

LORDS[®]

Refresher[®] on
COMPUTER ORGANIZATION
&
ARCHITECTURE

(BTES-401) P.T.U.
(BCSES1-401) M.R.S.P.T.U

WITH UPTO-DATE PREVIOUS QUESTION PAPERS CHAPTERWISE
& LORDS MODEL TEST PAPERS (UNSOLVED)

For
B.E., B.Tech (P.T.U./M.R.S.P.T.U)

4th Semester
(Computer Science Engg.)

New Edition February 2020

By
Ms. POOJA CHOPRA
MCA, M.Phil
Asst. Prof. in (CSE & IT)
DIPS Institute of Management & Technology
Jalandhar City.

LORDS PUBLICATIONS (Regd.)

36, Chandan Nagar, Jalandhar City. Tel. : 0181-2621630

~ Syllabus ~

LTP
3 --

1. FUNCTIONAL BLOCKS OF A COMPUTER

CPU, memory, input-output subsystems, control unit. Instruction set architecture of a CPU-registers, instruction execution cycle, RTL interpretation of instructions, addressing modes, instruction set. Case study - instruction set of 8085 processor.

Data representation : Signed number representation, fixed and floating point representations, character representation. Computer arithmetic - integer addition and subtraction, ripple carry adder, carry look-ahead adder, etc. multiplication - shift and add, Booth multiplier, carry save multiplier, etc. Division restoring and non-restoring techniques, floating point arithmetic.

2. INTRODUCTION TO X86 ARCHITECTURE

CPU Control Unit Design : Hardwired and micro-programmed design approaches, Case study - design of a simple hypothetical CPU.

Memory system design : Semiconductor memory technologies, memory organization.

Peripheral devices and their characteristics : Input-output subsystems, I/O device interface, I/O transfers - program controlled, interrupt driven and DMA, privileged and non-privileged instructions, software interrupts and exceptions. Programs and processes - role of interrupts in process state transitions, I/O device interfaces - SCII, USB.

3. PIPELINING

Basic concept of pipelining, throughput and speedup, pipeline hazards.

Parallel Processors : Introduction to parallel processors, Concurrent access to memory and cache coherency.

4. MEMORY ORGANISATION

Memory interleaving, concept of hierarchy memory organization, cache memory, cache size vs. block size, mapping functions, replacement algorithms, write policies.

~ Contents ~

Punjab Technical University Question Papers		5-8
<i>(Dec. 2019)</i>		
1.	Functional Blocks of a Computer	9-71
2.	Introduction to x86 Architecture	72-132
3.	Pipelining	133-165
4.	Memory Organisation	166-190
	Model Test Papers	191-192

PUNJAB TECHNICAL UNIVERSITY QUESTION PAPERS

UNIVERSITY QUESTION PAPER, DEC.-2019

SECTION – A

Q 1. What is the difference between machine and instruction cycles ?

Ans. Refer to Chapter No. 1 Q.No. 49

Q 2. What are the memory reference instructions ? Give examples.

Ans. Refer to Chapter No. 1 Q.No. 12

Q 3. What is hardwired control ? What are its advantages ?

Ans. Refer to Chapter No. 2 Q.No. 117

Q 4. What is control memory ?

Ans. Refer to Chapter No. 2 Q.No. 31

Q 5. Explain the concept of virtual memory.

Ans. Refer to Chapter No. 4 Q.No. 28

Q 6. What is the role of ROM memory in a computer system ?

Ans. Refer to Chapter No. 4 Q.No. 57

Q 7. What is register transfer language ?

Ans. Refer to Chapter No. 1 Q.No. 1

Q 8. What is an instruction pipeline ?

Ans. Refer to Chapter No. 3 Q.No. 3

Q 9. What are registers ? Can they be called memory ?

Ans. Refer to Chapter No. 4 Q.No. 58

Q 10. What is Microprocessor ?

Ans. Refer to Chapter No. 2 Q.No. 118

SECTION – B

Q 11. What is memory management hardware ? Explain.

Ans. Refer to Chapter No. 4 Q.No. 22

Q 12. Explain the organization of a typical computer system.

Ans. Refer to Chapter No. 1 Q.No. 11

Q 13. What is pipelined control ? Explain.

Ans. Refer to Chapter No. 3 Q.No. 5

Q 14. What are multilevel memory systems ? Explain with the help of a diagram.

Ans. Refer to Chapter No. 4 Q.No. 27

Q 15. How does a RISC organize CPU works ? What are its characteristics and advantages?

Ans. Refer to Chapter No. 1 Q.No. 51, 52 & 53

Chapter

1

Functional Blocks of a Computer

Contents

CPU, memory, input-output subsystems, control unit. Instruction set architecture of a CPU- registers, instruction execution cycle, RTL interpretation of instructions, addressing modes, instruction set. Case study - instruction set of 8085 processor.

Data representation : Signed number representation, fixed and floating point representations, character representation. Computer arithmetic - integer addition and subtraction, ripple carry adder, carry look-ahead adder, etc. multiplication - shift and add, Booth multiplier, carry save multiplier, etc. Division restoring and non-restoring techniques, floating point arithmetic.

POINTS TO REMEMBER

- ☞ A register transfer language is a system for expressing in symbolic form the micro operation sequences among the registers of digital module.
- ☞ The symbolic notation used to describe the micro-operation transfer among registers is known as register transfer language.
- ☞ Microoperation is an elementary operation performed with the data stored in registers.
- ☞ Register transfer microoperations transfer binary information from one register to another.
- ☞ Arithmetic microoperations perform arithmetic operations on numeric data stored in registers.
- ☞ Logic microoperations perform bit manipulation operations on non-numeric data stored in registers.
- ☞ Shift microoperations perform shift operations on data stored in registers.
- ☞ Register transfer language can also be used to facilitate the design process of digital systems.
- ☞ Computer registers are designated by capital letters to denote the function of the register.
- ☞ Control condition is terminated with a colon.
- ☞ Register transfer microoperation does not change the information content when the binary information moves from the source register to the destination register.
- ☞ The basic arithmetic micro operations are addition, subtraction, increment, decrement, and shift.
- ☞ Arithmetic shifts are explained later in conjunction with the shift microoperations.
- ☞ These microoperations are implemented with a combinational circuit or with a binary upward counter.
- ☞ The increment and decrement microoperations are symbolized by plus-one and minus-one operations, respectively.
- ☞ The arithmetic microoperations can be implemented in one composite arithmetic circuit.
- ☞ The hardware implementation of logic microoperations requires that logic gates be inserted for each bit or pair of bits in the registers to perform the required logic function.
- ☞ The digital circuit that forms the arithmetic sum of two bits and a previous carry is called a full-adder.

- ☞ The digital circuit that generates the arithmetic sum of two binary numbers of any lengths is called a binary adder.
- ☞ Shift register is a register that is capable of shifting its binary information in one or both directions.
- ☞ The internal organization of a digital system is defined by the sequence of microoperations it performs on data stored in its registers. The general purpose digital computer is capable of executing various micro-operations.
- ☞ An instruction code is a binary code that specifies a sequence of micro operations for the computer. Instruction code together with data stored in memory.
- ☞ The operation code of an instruction is a group of bits that define operations such as add, subtract, multiply, shift, complement etc.
- ☞ Computer instructions are normally stored in consecutive memory locations and are executed sequentially one at a time. The control reads an instruction from a specific address in a memory and executes it.
- ☞ The program counter (P.C) has 12 bits and it holds address of the next instruction to be needed from memory after the current instruction is executed.
- ☞ A basic computer has eight registers, a memory unit and a control unit. Paths must be provided to transfer information from one register to another and between memory and registers.
- ☞ A computer has three instruction code format. These are :
 - (i) Memory Instruction format
 - (ii) Register Instruction format
 - (iii) I/O Instruction format.
- ☞ There are two types of control organisation in a computer, that are, hardware control and microprogrammed control.
- ☞ In a hardwired organisation, the control logics are implemented with gates, flip-flops, decoders and other digital circuits.
- ☞ In a microprogrammed control, any required changes or modification can be done by operating the microprocessor in the control memory.
- ☞ The program is executed in a computer by going through a cycle for each instruction. Each instruction cycle is subdivided into a sequence of subcycles or phases.
- ☞ In a basic computer each instruction cycle consists of the following phases :
 - (i) Fetch an instruction from the memory.
 - (ii) Decode the instruction.
 - (iii) Read the effective address from memory if the instruction has an indirect address.
 - (iv) Execute the instruction.
- ☞ A pseudo instruction is not a machine instruction but rather an instruction to the assembler giving information about some phases of the translation.
- ☞ The translation of the symbolic program into binary is done by a special program called an assembler.
- ☞ A system program that translate a program written in a high level programming language to a machine language is called a compiler.
- ☞ Subroutine is a set of common instructions that can be used in a program many times. Each time a subroutine is used in the main part of program, a branch is executed to the beginning of the subroutine. After the subroutine is executed, a branch is made back to the main program.
- ☞ A control unit whose binary control variables are stored in memory is called a microprogrammed control unit. Each word in control memory contains within it a micro instruction.

- ☞ The micro instruction specifies one or more operations for the system. A sequence of micro instruction specifies constitutes a micro program.
- ☞ Mapping is a process of transformation the instruction code bits to an address in control memory where the routine is located.
- ☞ Most computer instructors can be classified into three categories :
 - (i) Data transfer instructions
 - (ii) Data manipulation instructions
 - (iii) Program control instructions.
- ☞ A computer with large number of instructions is known as 'complex instruction set computer', abbreviated as CISC.
- ☞ A computer which uses fewer instructions with simple constructs, so they can be classified much faster within CPU are known as 'Reduced Instruction Intractor Set Computers', abbreviated as RISC.

QUESTION-ANSWERS

Q 1. What is register transfer language? Explain it with example.

(PTU, Dec. 2019 ; May 2018)

Ans. Register Transfer Language : The symbolic notation used to describe the micro-operation transfer among registers is known as register transfer language. A register transfer language is a system for expressing in symbolic form the microoperation sequences among the registers of digital module. A kind of hardware description language (HDL) used in describing the registers of a computer or digital electronic system, and the way in which data is transferred between them.

Register transfer language is :

- A symbolic language
- A convenient tool for describing the internal organization of digital computers
- Can also be used to facilitate design process of digital systems.

Example of Register Transfer Language : Suppose let us take an example of information transfer from one register to another by means of replacement operator

$$R2 < R1$$

Which denotes transfer of content of register R1 into register R2.

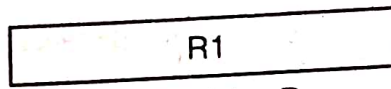
Q 2. Explain the concept of register transfer?

Or

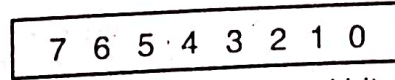
Explain the block diagram of register with its timing diagram.

Ans. Computer registers are designated by capital letters (sometimes followed by numerals) to denote the function of the register. For example, the register that holds an address for the memory unit is usually called a memory address register and is designated by the name MAR. Other designations for registers are PC (for program counter), IR (for instruction register), and R_1 (for processor register). The individual flip-flops in an n-bit register are numbered in sequence from 0 through $n - 1$, starting from 0 in the rightmost position and increasing the numbers toward the left. Fig. shows the representation of registers in block diagram form. The most common way to represent a register is by a rectangular box with the name of the register inside, as in Fig.(a). The individual bits can be distinguished as in (b). The numbering of bits in a 16-bit register can be marked on top of the box as shown in (c). A 16-bit register is partitioned into two parts in (d). Bits 0 through 7 are assigned the symbol L (for low byte) and bits 8 through 15 are assigned the symbol H (for high byte). The name of the 16-bit register is PC.

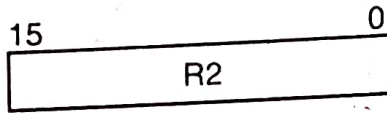
The symbol PC (0 – 7) or PC (L) refers to the low-order byte and PC (8 – 15) or PC (H) to the high-order byte.



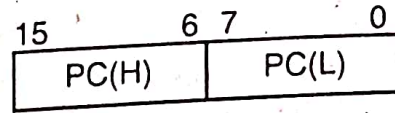
(a) Register R



(b) Showing individual bits



(c) Numbering of bits



(d) Divided into two parts

Block diagram of register

Information transfer from one register to another is designated in symbolic form by means of a replacement operator. The statement $R_2 \rightarrow R_1$ denotes a transfer of the content of register R_1 into register R_2 . It designates a replacement of the content of R_2 by the content of R_1 . By definition, the content of the source register R_1 does not change after the transfer.

A statement that specifies a register transfer implies that circuits are available from the output of the source register to the inputs of the destination register and that the destination register has a parallel load capability. Normally, we want the transfer to occur only under a predetermined control condition. This can be shown by means of an if-then statement.

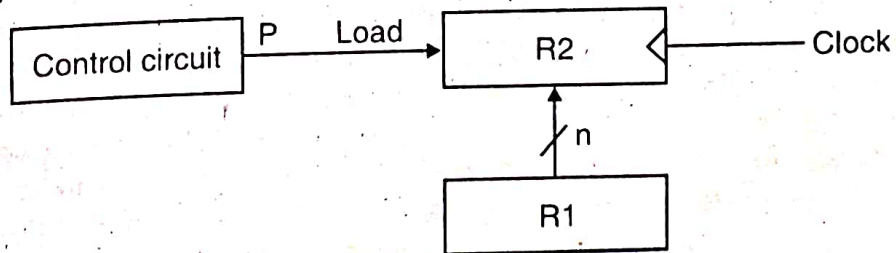
If $(P = 1)$ then $(R_2 \leftarrow R_1)$

where P is a control signal generated in the control section. It is sometimes convenient to separate the control variables from the register transfer operation by specifying a **control function**. A control function is a Boolean variable that is equal to 1 or 0. The control function is included in the statement as follows :

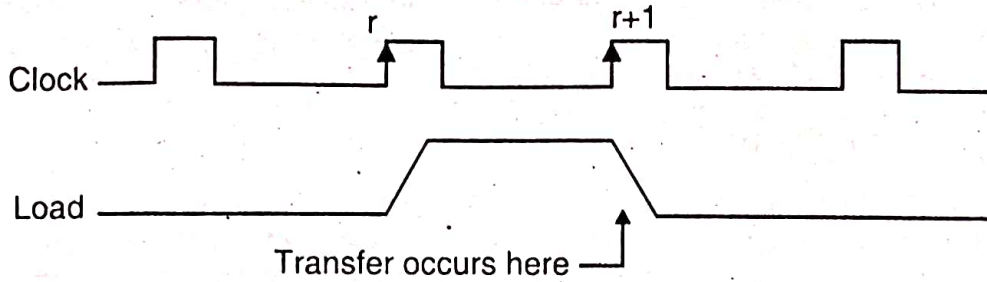
$P : R_2 \leftarrow R_1$

The control condition is terminated with a colon. It symbolizes the requirement that the transfer operation be executed by the hardware only if $P = 1$.

Every statement written in a register transfer notation implies a hardware construction for implementing the transfer. The above fig. shows the block diagram that depicts the transfer from R_1 to R_2 . The n outputs of register R_1 are connected to the n inputs of register R_2 . The letter n will be used to indicate any number of bits for the register. It will be replaced by an actual number when the length of the register is known. Register R_2 has a load input that is activated by the control variable P . It is assumed that the control variable is synchronized with the same clock as the one applied to the register. As shown in the timing diagram (b), P is activated in the control section by the rising edge of a clock pulse at time t . The next positive transition of the clock at time $t + 1$ finds the load input active and the data inputs of R_2 are then loaded into the register in parallel. P may go back to 0 at time $t + 1$; otherwise, the transfer will occur with every clock pulse transition while P remains active.



(a) Block diagram



(b) Timing diagram

Timing diagram

The basic symbols of the register transfer notations are listed in table. Registers are denoted by capital letters, and numerals may follow the letters. Parentheses are used to denote a part of a register by specifying the range of bits or by giving a symbol name to a portion of a register. The arrow denotes a transfer of information and the direction of transfer. A comma is used to separate two or more operations that are executed at the same time. The statement

$$T : R_2 \leftarrow R_1, R_1 \leftarrow R_2$$

denotes an operation that exchanges the contents of two registers during one common clock pulse provided that $T = 1$. This simultaneous operation is possible with registers that have edge-triggered flip-flops.

Symbol	Description	Example
Letters (and numerals)	Denotes a register	MAR, R2
Parentheses ()	Denotes a part of a register	R2 (0 - 7), R2 (L)
Arrow ←	Denotes transfer of information	R2 ← R1
Comma ,	Separates two microoperations	R2 ← R1, R1 ← R2

Table : Basic Symbols for Register Transfers

Q 3. Explain the various microoperations performed in registers.

Ans. A microoperation is an elementary operation performed with the data stored in registers. The microoperations most often encountered in digital computers are classified into four categories :

- 1. Register transfer microoperations** transfer binary information from one register to another.
- 2. Arithmetic microoperations** perform arithmetic operations on numeric data stored in registers.
- 3. Logic microoperations** perform bits manipulation operations on non-numeric data stored in registers.

4. Shift microoperations perform shift operations on data stored in registers.

The register transfer microoperation does not change the information content when the binary information moves from the source register to the destination register. The other three types of microoperations change the information content during the transfer.

Q 4. Explain arithmetic microoperations.

Ans. The basic arithmetic micro operations are addition, subtraction, increment, decrement, and shift. Arithmetic shifts are explained later in conjunction with the shift microoperations. The arithmetic microoperation defined by the statement.

$R_3 \leftarrow R_1 + R_2$ specifies an add microoperation. It states that the contents of register R_1 are added to the contents of register R_2 and the sum transferred to register R_3 . To implement this statement with hardware we need three registers and the digital component that performs the addition

operation. The other basic arithmetic microoperations are listed in Table. Subtraction is most often implemented through complementation and addition. Instead of using the minus operator, we can specify the subtraction by the following statement :

$$R3 \leftarrow R1 + \overline{R2} + 1$$

$\overline{R2}$ is the symbol for the 1's complement of R2. Adding 1 to the 1's complement produces the 2's complement. Adding the contents of R1 to the 2's complement of R2 is equivalent to $R1 - R2$.

Symbolic Designation	Description
$R3 \leftarrow R1 + R2$	Contents of R1 plus R2 transferred to R3
$R3 \leftarrow R1 - R2$	Contents of R1 minus R2 transferred to R3
$R2 \leftarrow \overline{R2}$	Complement the contents of R2 (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement the contents of R2 (
$R3 \leftarrow R1 + \overline{R2} + 1$	R1 plus the 2's complement of R2 (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of R1 by one
$R1 \leftarrow R1 - 1$	Decrement the contents of R1 by one

Table : Arithmetic Microoperations

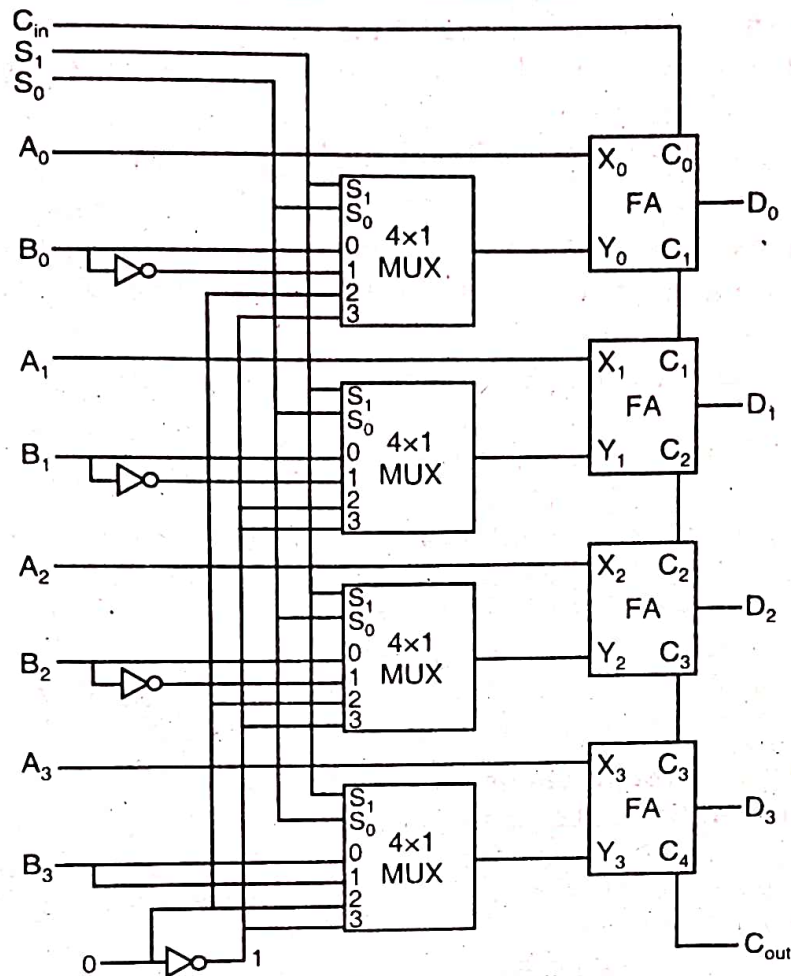
The increment and decrement microoperations are symbolized by plus-one and minus-one operations, respectively. These microoperations are implemented with a combinational circuit or with a binary up-down counter.

The arithmetic operations of multiply and divide are not listed in table. These two operations are valid arithmetic operations but are not included in the basic set of microoperations. The only place where these operations can be considered as microoperations is in a digital system, where they are implemented by means of a combinational circuit. In such a case, the signals that perform these operations propagate through gates, and the result of the operation can be transferred into a destination register by a clock pulse as soon as the output signal propagates through the combinational circuit. In most computers, the multiplication operation is implemented with the sequence of add and shift microoperations. Division is implemented with a sequence of subtract and shift microoperations. To specify the hardware in such a case requires a list of statements that use the basic microoperations of add, subtract and shift.

Q 5. Implement the arithmetic circuit using arithmetic microoperations.(PTU, May 2017)

Ans. The arithmetic microoperations can be implemented in one composite arithmetic circuit. The basic component of an arithmetic circuit is the parallel adder. By controlling the data inputs to the adder, it is possible to obtain different types of arithmetic operations.

The diagram of a 4-bit arithmetic circuit is shown in fig. It has four full-adder circuits that constitute the 4-bit adder and four multiplexers for choosing different operations. There are two 4-bit inputs A and B and a 4-bit output D. The four inputs from A go directly to the X inputs of the binary adder. Each of the four inputs from B is connected to the data inputs of the multiplexers. The multiplexers data inputs also receive the complement of B. The other two data inputs are connected to logic-0 and logic-1. Logic-0 is a fixed voltage value (0 volts for TTL integrated circuits) and the logic-1 signal can be generated through an inverter whose input is 0. The four multiplexers are controlled by two selection inputs S_1 and S_0 . The input carry C_{in} goes to the carry input of the FA in the least significant position. The other carries are connected from one stage to the next.



4-bit arithmetic circuit

The output of the binary adder is calculated from the following arithmetic sum :

$$D = A + Y + C_{in}$$

where A is the 4-bit binary number at the X inputs and Y is the 4-bit binary number at the Y inputs of the binary adder. C_{in} is the input carry, which can be equal to 0 or 1. Note that the symbol + in the equation above denotes an arithmetic plus. By controlling the value of Y with the two selection inputs S_1 and S_0 and making C_{in} equal to 0 or 1, it is possible to generate the eight arithmetic microoperations listed in table.

Select			Input	Output	Microoperation
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	
0	0	0	B	$D = A + B$	Add
0	0	1	\overline{B}	$D = A + \overline{B} - 1$	Add with carry
0	1	0	\overline{B}	$D = A + \overline{B}$	Subtract with borrow
0	1	1	B	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A + 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

Table : Arithmetic Circuit Function Table

When $S_1 S_0 = 00$, the value of B is applied to the Y inputs of the adder. If $C_{in} = 0$, the output $D = A + B$. If $C_{in} = 1$, output $D = A + B + 1$. Both cases perform the add microoperation with or without adding the input carry.

When $S_1 S_0 = 01$, the complement of B is applied to the Y inputs of the adder. If $C_{in} = 1$, then $D = A + \bar{B} + 1$. This produces A plus the 2's complement of B , which is equivalent to a subtraction of $A - B$. When $C_{in} = 0$, then $D = A + \bar{B} + 1$. This produces A plus the 2's complement of B , which is equivalent to a subtraction of $A - B$. When $C_{in} = 0$, then $D = A + \bar{B}$. This is equivalent to a subtract with borrow, that is, $A - B - 1$.

When $S_1 S_0 = 10$, the inputs from B are neglected, and instead, all 0's are inserted into the Y inputs. The output becomes $D = A + 0 + C_{in}$. This gives $D = A$ when $C_{in} = 0$ and $D = A + 1$ when $C_{in} = 1$. In the first case we have a direct transfer from input A to output D . In the second case, the value of A is increment by 1.

When $S_1 S_0 = 11$, all 1's are inserted into the Y inputs of the adder to produce the decrement operation $D = A - 1$ when $C_{in} = 0$. This is because a number with all 1's is equal to the 2's complement of 1 (the 2's complement of binary 0001 is 1111). Adding a number A to the 2's complement of 1 produces $F = A + 2$'s complement of 1 = $A - 1$. When $C_{in} = 1$, then $D = A - 1 + 1 = A$, which causes a direct transfer from input A to output D . Note that the microoperation $D = A$ is generated twice, so there are only seven distinct microoperations in the arithmetic circuit.

Q 6. What are logic microoperations?

Ans. Logic microoperations : Logic microoperations specify binary operations for strings of bits stored in registers. These operations consider each bit of the register separately and treat them as binary variables. For example, the exclusive-OR microoperation with the contents of two registers $R1$ and $R2$ is symbolized by the statement.

$$P : R1 \leftarrow R1 \oplus R2$$

It specifies a logic microoperation to be executed on the individual bits of the registers provided that the control variable $P = 1$. As a numerical example, assume that each register has four bits. Let the content of $R1$ be 1010 and the content of $R2$ be 1100. The exclusive-OR microoperation stated above symbolizes the following logic computation :

1010 Content of R1

1100 Content of R2

0110 Content of R1 after $P = 1$

The content of $R1$, after the execution of the microoperation, is equal to the bit-by-bit exclusive-OR operation on pairs of bits in $R2$ and previous values of $R1$. The logic microoperations are seldom used in scientific computations, but they are very useful for bit manipulation of binary data and for making logical decisions.

Special symbols will be adopted for the logic microoperations OR, AND and complement, to distinguish them from the corresponding symbols used to express Boolean functions. The symbol \vee will be used to denote an OR microoperation and the symbol \wedge to denote an AND microoperation. The complement microoperation is the same as the 1's complement and uses a bar on top of the symbol that denotes the register name. By using different symbols, it will be possible to differentiate between a logic microoperation and a control (or Boolean) function. Another reason for adopting two sets of symbols is to be able to distinguish the symbol $+$, when used to symbolize an arithmetic plus, from a logic OR operation. Although the $+$ symbol has two meanings, it will be possible to distinguish between them by noting where the symbol occurs. When the symbol $+$ occurs in a microoperation, it will denote

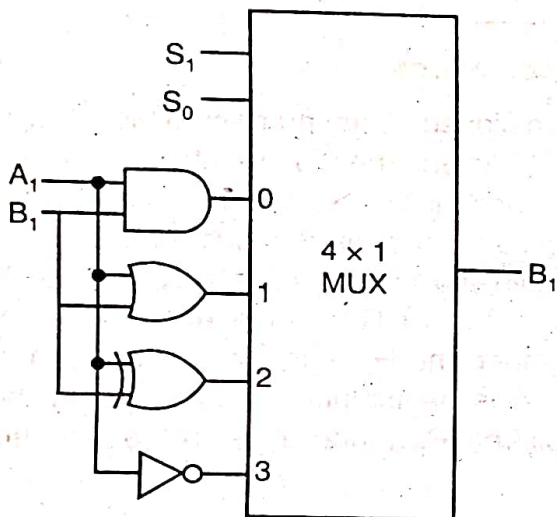
an OR operation. We will never use it to symbolize an OR microoperation. For example, in the statement $P + Q : R1 \leftarrow R2 + R3, R4 \leftarrow R5 \vee R6$

the + between P and Q is an OR operation between two binary variables of a control function. The + between R2 and R3 specifies an add microoperation. The OR microoperation is designated by the symbol \vee between registers R5 and R6.

Q 7. Explain Hardware implementation of logic microoperation.

Ans. Hardware Implementation : The hardware implementation of logic microoperations requires that logic gates be inserted for each bit or pair of bits in the registers to perform the required logic function. Although there are 16 logic microoperations, most computers use only four – AND, OR, XOR (exclusive-OR), and complement-from which all others can be derived.

Fig. shows one stage of a circuit that generates the four basic logic microoperations. It consists of four gates and a multiplexer. Each of the four logic operations is generated through a gate that performs the required logic. The outputs of the gates are applied to the data inputs of the multiplexer. The two selection inputs S_1 and S_0 choose one of the data inputs of the multiplexer and direct its value to the output. The diagram shows one typical stage with subscript i. For a logic circuit with n bits, the diagram must be repeated n times for $i = 0, 1, 2, \dots, n - 1$. The selection variables are applied to all stages. The function table in fig. (b) lists the logic microoperations obtained for each combination of the selection variables.



(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \cap B$	AND
0	1	$E = A \cup B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \bar{A}$	Complement

(b) Function circuits

One stage of logic circuit

Q 8. What are shift microoperations? Explain with example. Design 4 bit combinational circuit for shifter. (PTU, May 2018, 2015)

Ans. Shift Microoperations : Shift microoperations are used for serial transfer of data. They are also used in conjunction with arithmetic, logic and other data-processing operations. The contents of a register can be shifted to the left or the right. At the same time that the bits are shifted, the first flip-flop receives its binary information from the serial input. During a shift-left operation the serial input transfers a bit into the rightmost position. During a shift-right operation the serial input transfers a bit into the leftmost position. The information transferred through the serial input determines the type of shift. There are three types of shifts : logical, circular, and arithmetic.

A logical shift is one that transfers 0 through the serial input. We will adopt the symbols shl and shr for logical shift-left and shift-right microoperations. For example :

$$R1 \leftarrow \text{shl } R1$$

$$R2 \leftarrow \text{shr } R2$$

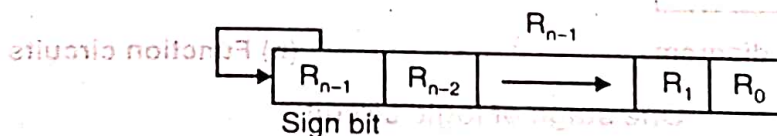
are two microoperations that specify a 1-bit shift to the left of the content of register R1 and a 1-bit shift to the right of the content of register R2. The register symbol must be the same on both sides of the arrow. The bit transferred to the end position through the serial input is assumed to be 0 during a logical shift.

The circular shift (also known as a rotate operation) circulates the bits of the register around the two ends without loss of information. This is accomplished by connecting the serial output of the shift register to its serial input. We will use the symbols cill and cir for the circular shift left and right, respectively. The symbolic notation for the shift microoperations is shown in table.

Symbolic designation	Description
$R \leftarrow \text{shl } R$	Shift-left register R
$R \leftarrow \text{shr } R$	Shift-right register R
$R \leftarrow \text{cill } R$	Circular shift-left register R
$R \leftarrow \text{cir } R$	Circular shift-right register R
$R \leftarrow \text{ashr } R$	Arithmetic shift-left R
$R \leftarrow \text{ashr } R$	Arithmetic shift-right R

Table : Shift Microoperations

An arithmetic shift is a microoperation that shifts a signed binary number to the left or right. An arithmetic shift-left multiplies a signed binary number by 2. An arithmetic shift-right divides the number by 2. Arithmetic shifts must leave the sign bit unchanged because the sign of the number remains the same when it is multiplied or divided by 2. The leftmost bit in a register holds the sign bit, and the remaining bits hold the number. The sign bit is 0 for positive and 1 for negative. Negative numbers are in 2's complement form. Fig. shows a typical register of n bits. Bit R_{n-1} in the leftmost position holds the sign bit. R_{n-2} is the most significant bit of the number and R_0 is the least significant bit. The arithmetic shift-right leaves the sign bit unchanged and shifts the number (including the sign bit) to the right. Thus R_{n-1} remains the same, R_{n-2} receives the bit from R_{n-1} , and so on for the other bits in the register. The bit in R_0 is lost.



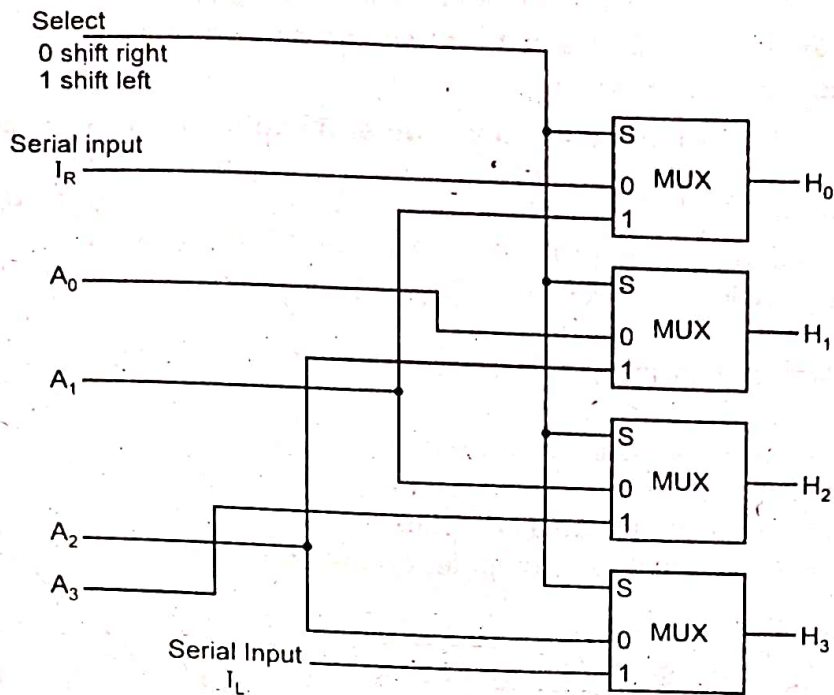
Arithmetic shift right

The arithmetic shift-left inserts a 0 into R_0 , and shifts all other bits to the left. The initial bit of R_{n-1} is lost and replaced by the bit from R_{n-2} . A sign reversal occurs if the bit in R_{n-1} changes in value after the shift. This happens if the multiplication by 2 causes an overflow. An overflow occurs after an arithmetic shift left if initially, before the shift, R_{n-1} is not equal to R_{n-2} . An overflow flip-flop V_s can be used to detect an arithmetic shift-left overflow.

$$V_s = R_{n-1} \oplus R_{n-2}$$

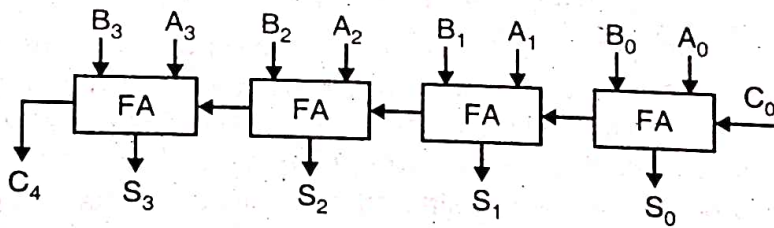
If $V_s = 0$, there is no overflow, but if $V_s = 1$, there is an overflow and a sign reversal after the shift. V_s must be transferred into the overflow flip-flop with the same clock pulse that shifts the register.

4-bit combinational circuit shifter



Q 9. To implement add microoperation with hardware.

Ans. Binary Adder : To implement the add microoperation with hardware, we need the registers that hold the data and the digital component that performs the arithmetic addition. The digital circuit that forms the arithmetic sum of two bits and a previous carry is called a full-adder. The digital circuit that generates the arithmetic sum of two binary numbers of any lengths is called a binary adder. The binary adder is constructed with full-adder circuits connected in cascade, with the output carry from one full-adder connected to the input carry of the next full-adder. Fig. shows the interconnections of four full-adders (FA) to provide a 4-bit binary adder. The augend bits of A and the addend bits of B are designated by subscript numbers from right to left, with subscript 0 denoting the low-order bit. The carries are connected in a chain through the full-adders. The input carry to the binary adder is C_0 and the output carry is C_4 . The S outputs of the full-adders generate the required sum bits.



4-bit binary adder

An n-bit binary adder requires n full-adders. The output carry from each full-adder is connected to the input carry of the next-high-order full-adder. The n data bits for the A inputs come from one register (such as R1), and the n data bits for the B inputs come from another register (such as R2). The sum can be transferred to a third register or to one of the source registers (R1 or R2), replacing its previous content.

Q 10. Differentiate between register and memory. (PTU, Dec. 2011)

Ans. Registers are the memory space provided by the processor for storing temporary values

on which the operation is currently performing. And the memory locations are usually referred to as physical memory addresses i.e. RAM addresses. The data are brought to and from the registers are from the physical memory. All data and instructions are stored in the physical memory and the data are brought to the registers for execution.

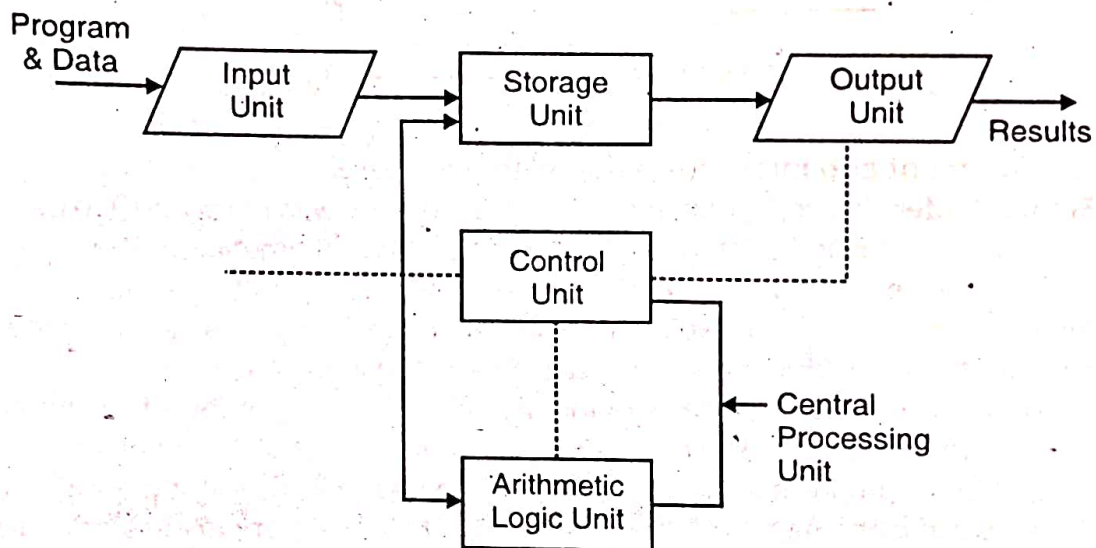
Q 11. Draw the block diagram of computer and explain its various components.

(PTU, Dec. 2019)

Ans. Block Diagram of Computer : A computer can process data, pictures, sound and graphics. They can solve highly complicated problems quickly and accurately. A computer as shown in fig. performs basically five major computer operations or functions irrespective of their size and make. These are

1. It accepts data or instructions by way of input
2. It stores data
3. It can process data as required by the user
4. It gives results in the form of output, and
5. It controls all operations inside a computer.

We discuss below each of these computer operations :



Basic Computer Operations

1. Input : This is the process of entering data and programs into the computer system. You should know that computer is an electronic machine like any other machine which takes as inputs raw data and performs some processing giving out processed data. Therefore, the input unit takes data from us to the computer in an organized manner for processing.

2. Storage : The process of saving data and instructions permanently is known as storage. Data has to be fed into the system before the actual processing starts. It is because the processing speed of Central Processing Unit (CPU) is so fast that the data has to be provided to CPU with the same speed. Therefore the data is first stored in the storage unit for faster access and processing. This storage unit or the primary storage of the computer system is designed to do the above functionality. It provides space for storing data and instructions.

The storage unit performs the following major functions :

- All data and instructions are stored here before and after processing.
- Intermediate results of processing are also stored here.

3. Processing : The task of performing operations like arithmetic and logical operations is

called processing. The Central Processing Unit (CPU) takes data and instructions from the storage unit and makes all sorts of calculations based on the instructions given and the type of data provided. It is then sent back to the storage unit.

4. Output : This is the process of producing results from the data for getting useful information. Similarly, the output produced by the computer after processing must also be kept somewhere inside the computer before being given to you in human readable form. Again the output is also stored inside the computer for further processing.

5. Control : The manner how instructions are executed and the above operations are performed. Controlling of all operations like input, processing and output are performed by control unit. It takes care of step by step processing of all operations inside the computer.

Functional Units : In order to carry out the operations mentioned in the previous section the computer allocates the task between its various functional units. The computer system is divided into three separate units for its operation. They are :

1. Arithmetic logical unit
2. Control unit
3. Central processing unit.

1. Arithmetic Logical Unit (ALU) :

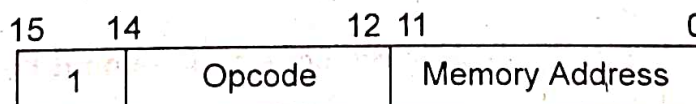
Logical Unit : After you enter data through the input device it is stored in the primary storage unit. The actual processing of the data and instruction are performed by Arithmetic Logical Unit. The major operations performed by the ALU are addition, subtraction, multiplication, division, logic and compression. Data is transferred to ALU from storage unit when required. After processing the output is returned back to storage unit for further processing or getting stored.

2. Control Unit (CU) : The next component of computer is the Control Unit, which acts like the supervisor seeing that things are done in proper fashion. Control unit is responsible for co-ordinating various operations using time signal. The control unit determines the sequence in which computer programs and instructions are executed. Things like processing of programs stored in the main memory, interpretation of the instructions and issuing of signals for other units of the computer to execute them. It also acts as a switch board operator when several users access the computer simultaneously. Thereby it coordinates the activities of computer's peripheral equipment as they perform the input and output.

3. Central Processing Unit (CPU) : The ALU and the CPU of a computer system are jointly known as the central processing unit. You may call CPU as the brain of any computer system, It is just like brain that takes all major decisions, makes all sorts of calculations and directs different parts of the computer functions by activating and controlling the operations.

Q 12. What are the memory reference instructions ? Give examples. (PTU, Dec. 2019)

Ans. These instructions refer to memory address as an operand. The other operand is always accumulator. Specifies 12-bit address 3-bit opcode and 1-bit addressing mode for direct and indirect addressing.



Example : IR register contains = 000/xxxxxxxxxxx, i.e. ADD after fetching and decoding of instruction we find out that it is a memory reference instruction for ADD operation.

Hence,

$DR \leftarrow M[AR]$

$AC \leftarrow AC + DR, SC \leftarrow 0$

Q 13. What do you understand by floating point arithmetic? (PTU, Dec. 2011)

Ans. Floating-point arithmetic is considered an esoteric subject by many people. This is rather surprising because floating-point is ubiquitous in computer systems. Almost every language has a floating-point data type; computers from PCs to supercomputers have floating-point accelerators; most compilers will be called upon to compile floating-point algorithms from time to time and virtually every operating system must respond to floating-point exceptions such as overflow.

Q 14. List the functions of 8251. (PTU, May 2012)

Ans. The functions of 8251 are :

1. 8251 can be used to transmit receive serial data. Data transmission to a CRT terminal using the 8251 in status check mode.

2. A programmable chip designed for synchronous/asynchronous serial data communication.

Q 15. Perform the subtraction with the following assigned binary number by taking the 2's complement of the subtrahend 11010-11111. (PTU, Dec. 2010)

Ans. Let $X = 11010$ and $Y = 11111$. So $X - Y$

Using 2's complement will be

$$X = 11010$$

2's complement of $Y = 00001$

$$\text{Sum} = 11011$$

Here no carry means, the $X - Y$ will be negative 2's complement of sum above i.e. 00101.

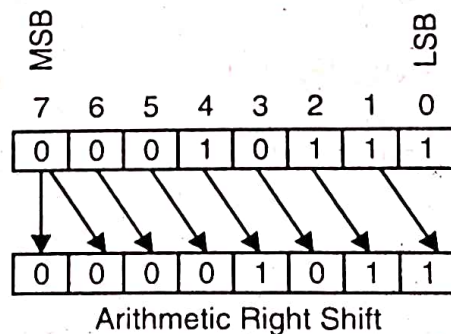
Q 16. Where ASCII code is used in computers? (PTU, Dec. 2010)

Ans. The American standard code for information interchange (ASCII) is a character-encoding scheme based on the ordering of the English alphabet. ASCII codes represent text in computers communications equipment and other devices that use text. Most modern character encoding schemes are based on ASCII though they support many more characters than ASCII does.

ASCII is the Internet Assigned Numbers Authority (IANA) preferred charset name for ASCII.

Q 17. An 8-bit register contains the binary value 10011100. What is the register value after arithmetic shift right? (PTU, May 2011)

Ans. 00010111 Right shift = 00001011



In this case, the rightmost 1 was shifted out a new 0 was copied into the leftmost portion, preserving the sign of the computer.

Q 18. Represent the following conditional control statement by two register transfer statements with the control functions.

If $(P = 1)$ then $(R_1 \leftarrow R_2)$ else if $(Q = 1)$ then $(R_1 \leftarrow R_3)$. (PTU, May 2011)

Ans. $P : R_1 \leftarrow R_2 ;$ if $(P = 1)$ then $(R_1 \leftarrow R_2)$

$P'Q : R_1 \leftarrow R_3 ;$ Else if $(Q = 1)$ then $(R_1 \leftarrow R_3)$

Q 19. Write a symbolic microprogram for the ADD operation. (PTU, May 2011)

Ans. The execution of the ADD instruction is carried out by the micro in structures at addresses

1 and 2. The first microstruction reads the operand from memory into DR. The second microinstruction performs and add micro-operation with the content of DR and AC and then jumps back to the beginning of the getch routine.

Lable	Microoperations	CD	BR	AD
ADD :	ORGO	1	CALL	Indirect
	NOP	u	JMP	Next
	READ			
	ADD	u	JUMP	FETCH

Q 20. With the help of circuits discuss look ahead carry generator. Show how it makes faster additions. (PTU, Dec. 2010)

Ans. These circuits are high-speed, look-ahead carry generators, capable of anticipating a carry across four binary adders or groups of adders. They are cascaded to perform full look ahead across n-bit adders. Carry-generate carry and propagate carry functions are provided as shown in the pin designation table.

When used in conjunction with the 181 arithmetic logic unit these generators provides high-speed carry look-ahead capability for any word length. Each DM74SI82 generators the look-ahead (anticipated carry) across a group of four ALU's and in addition, other carry look-ahead circuits may be employed to anticipate carry across sections of four look-ahead perform multi-level look-ahead is illustrated under typical application data.

Carry input and output of the ALU's are in their true form and the carry propagate (P) and carry generate (G) are in negated form therefore the carry functions (inputs, outputs, generate and propagate) of the look-ahead generators are implemented in the compatible forms for direct connection to the ALU reinterpretations of carry functions as explained on the 181 data sheet are also applicable to and compatible with the look-ahead generator positive logic equations for the DM748182 are :

$$C_{n+x} = \overline{G}_0 + \overline{P}_0 C_n$$

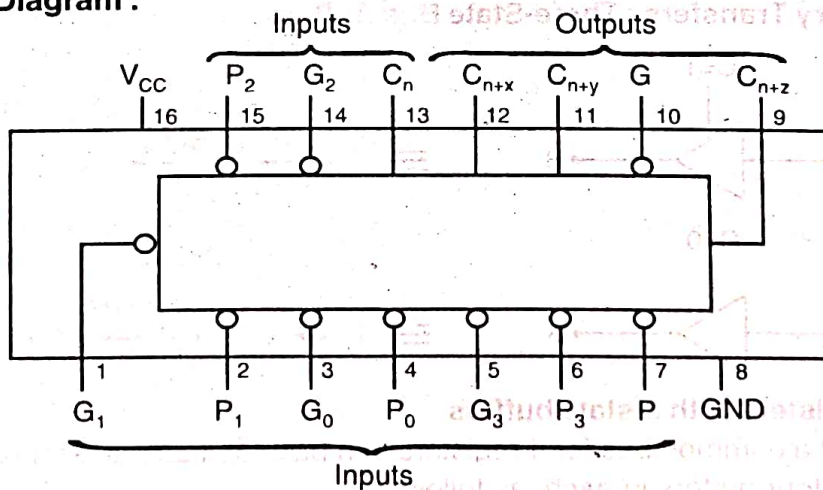
$$C_{n+y} = G_1 + \overline{P}_1 G_0 + \overline{P}_1 \overline{P}_0 C_n$$

$$C_{n+z} = \overline{G}_2 + \overline{P}_2 G_1 + \overline{P}_2 \overline{P}_1 G_0 + \overline{P}_2 \overline{P}_1 \overline{P}_0 C_n$$

$$= \overline{G}_3 (\overline{P}_3 + \overline{G}_2) (\overline{P}_3 + \overline{P}_2 + \overline{G}_1) (\overline{P}_3 + \overline{P}_2 + \overline{P}_1 + \overline{G}_0)$$

$$\overline{P} = \overline{P}_3 \overline{P}_2 \overline{P}_1 \overline{P}_0$$

Connection Diagram :



Designation	Pin Nos	Function
G_0, G_1, G_2, G_3	3, 1, 14, 5	Active Low Carry generate input
P_0, P_1, P_2, P_3	4, 2, 15, 6	Active low Carry propagate input
C_n	13	Carry input
C_{n+x}, C_{n+y}	12, 11, 9	Carry outputs
C_{n+2}		
G	10	Active low Carry propagate output
P	7	
V_{CE}	16	Supply voltage
GND	8	Ground

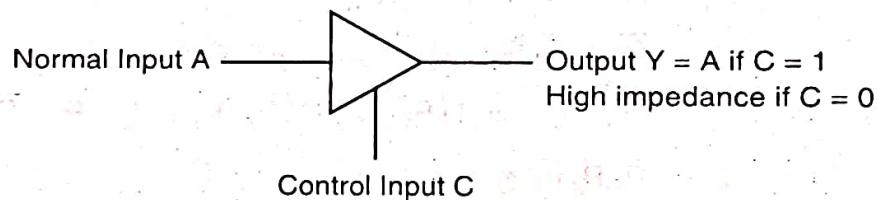
Q 21. Draw the diagram for a common bus system using tri-state buffers and a decoder instead of multiplexers. (PTU, May 2011)

Ans. Bus transfers

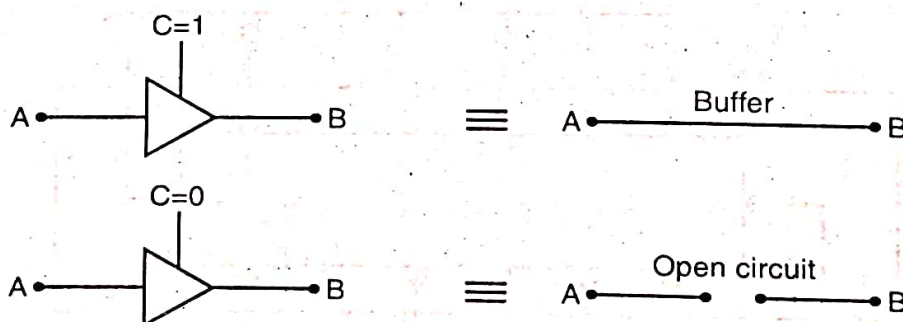
- The transfer of information from a bus into one of many destination registers can be accomplished by connecting the bus lines to the inputs of all destination registers and activating the load control of the particular destination register selected.
- The symbolic statement for a bus transfer may mention the bus or its presence may be implied in the statement. When bus is included in the statement we write :
 $BUS \leftarrow C, R1 \leftarrow BUS$ (however it is $R1 \leftarrow C$)

Tri State Bus Buffers

- A three-state gate is a digital circuit that exhibits three states. Two of the states are signals equivalent to logic 1 and 0 as in a conventional gate.
- The third state is called high impedance state
- The high impedance state behaves like an open circuit which means that the output is disconnected and does not have a logic significance.

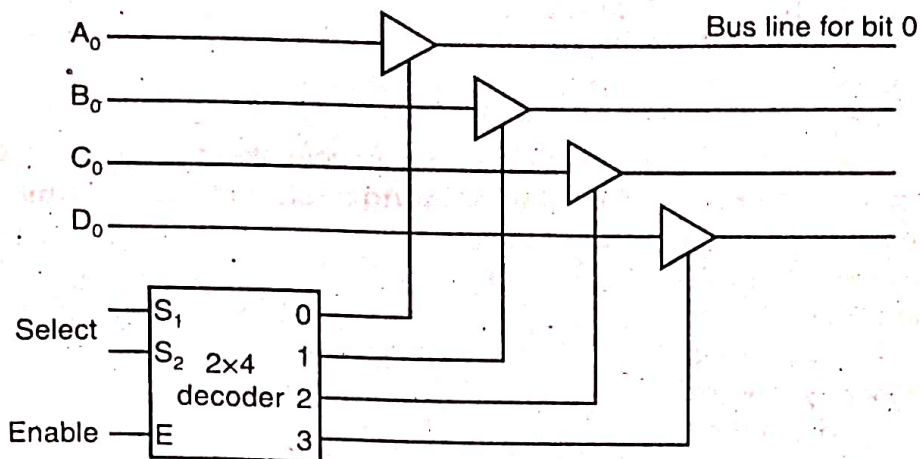


Bus and Memory Transfers : Three-State Bus Buffers



Connecting registers with 3-state buffers

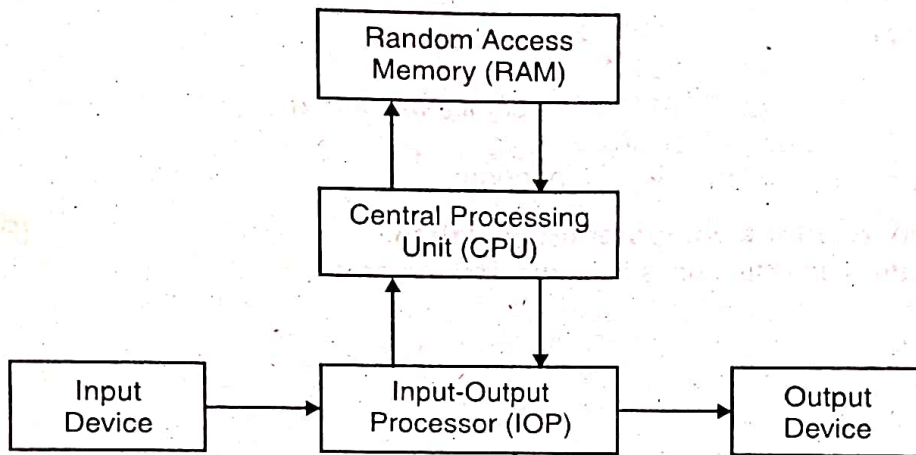
- To construct a common bus for 4 registers of n bits each using 3-state buffers, we need n circuits with four buffers in each, as follows :



Q 22. Draw the block diagram of a computer and explain the function of each block.

(PTU, Dec. 2007 ; May 2016, 2004)

Ans.



Block diagram of a Computer

The block diagram of a computer is shown above. Basically, the hardware of computer is divided into three major parts.

1. CPU : CPU is Central Processing Unit and it contains an Arithmetic and logic unit for manipulating data, a number of registers for storing data, and control circuits for fetching and executing instructions.

2. Memory : The memory of a computer consists of storage for instruction and data. Normally, it is called Random Access Memory (RAM) because CPU can access any location in memory at random and retrieve any information within a fixed interval to time.

The Input and Output processor/Job consists of electronic circuits and is used for communicating and controlling the transfer of information or between computer and outside world. i.e. it performs the basic input and output operations when CPU sends a request to it.

3. I/O Devices : The input and output devices are peripheral input devices, such as keyboard, mouse, etc are used to input any data to the processor monitor, printer, etc are used result to the user which inputs the data.

Q 23. Perform the subtraction with the following unsigned binary number by taking the 2's complement of the subtrahend 100-11000.

(PTU, Dec. 2004)

Ans. Let $x = 00100$ and $y = 11000$

So $x-y$ using 2' complement will be

$$x = 00100$$

2's complement of $y = 01000$

$$\text{Sum} = 01100$$

Here no carry means, the $x-y$ will be -ve 2's complement of sum above i.e. 10100.

Q 24. Perform the subtraction with the following unsigned binary number by taking the 2's complement of the subtrahend 11010-11111. (PTU, May 2005)

Ans. Let $x = 11010$ and $y = 11111$. So $x-y$

Using 2's complement will be

$$x = 11010$$

2's complement of $y = 00001$

$$\text{Sum} = 11011$$

Here no carry means, the $x-y$ will be negative 2's complement of sum above i.e. 00101.

Q 25. Perform the subtraction with the following unsigned binary number by taking the 2's complement of the subtrahend 1010100-1010100. (PTU, Dec. 2009, 2005)

Ans. Let $x = 1010100$ and $y = 1010100$

So $x - y$ using 2's complement will be.

$$x = 1010100$$

$$y = 0101100 \text{ (2's complement of } y)$$

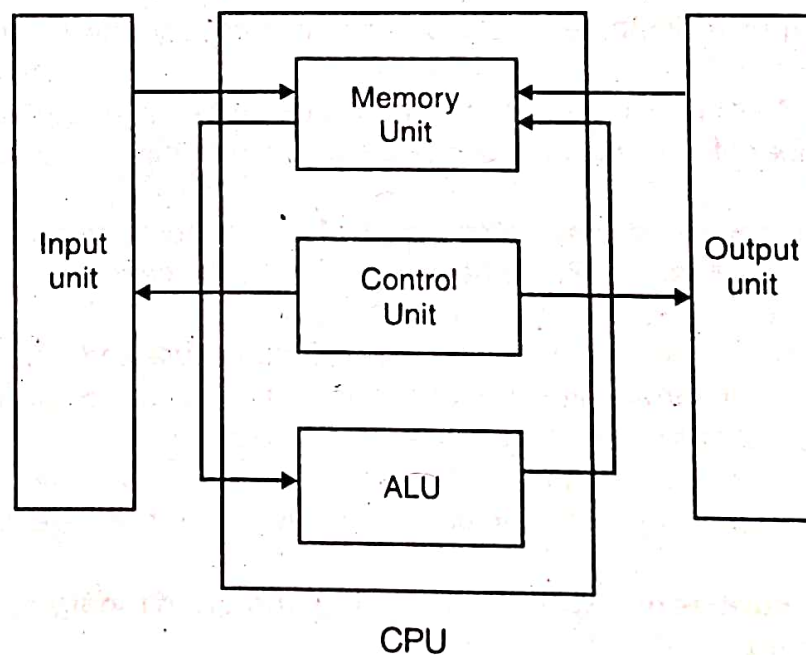
$$\text{Sum} = 10000000$$

Discarding the end carry then $x - y = 000000$

Q 26. Explain about the computer organization. (PTU, Dec. 2006)

Ans. There are mainly five units in a computer organization :

1. Input Unit
2. Output Unit
3. Memory
4. Control Unit
5. Arithmetic and logic unit



Block Diagram of Computer

Q 27. Explain the meaning of the memory – reference instruction STA.

(PTU, Dec. 2009)

Ans. Memory reference instruction STA occupies memory address which is 8 bit number. Memory reference instruction STA occupy 2 consecutive bytes of RAM. The first byte is reserved for opcode and the second byte indicates the 8 bit memory address. STA instruction stores the accumulator to memory. The instruction STA 100 means "Store the value currently in the accumulator in memory address 100".

The following program illustrates how can translate the statement $Z = A * B + C * D - E * F$ into a sequence of one-address instructions :

```
LDA    E    ; Acc ← E
MUL    F    ; Acc ← Acc * F
STA    T1   ; T1 ← Acc
LDA    C    ; Acc ← C
MUL    D    ; Acc ← Acc * D
STA    T2   ; T2 ← Acc
LDA    A    ; Acc ← A
MUL    B    ; Acc ← Acc * B
ADD    T2   ; Acc ← Acc + T2
SUB    T1   ; Acc ← Acc - T1
STA    Z    ; Z ← Acc
```

In this program, T1 and T2 represent the addresses of memory locations used to store temporary results. Instructions that do not require any addresses are called "zero-address instruction." All microprocessors include some zero-address instructions in the instruction set. Typical examples of zero-address instructions are CLC (clear carry) and NOP.

Q 28. Differentiate between arithmetic shift and logical shift.

(PTU, Dec. 2007)

Ans. The arithmetic instruction for logical operation are :

Clear, compliment, AND, OR, XOR, Clear and Carry, set array, complement carry, enable or disable interrupt.

The instructions for shift operation are :

1. Logical shift right
2. Logical shift left
3. Arithmetic shift right
4. Arithmetic shift left
5. Rotate right
6. Rotate left
7. Rotate right through carry
8. Rotate left through carry

Shift Operation : The circular shift operations are machine instructions in basic computer. The other shifts of interest are logic shift and arithmetic shifts. These two shifts can be programmed with a small number of instructions.

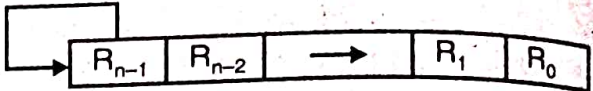
The logical shift requires that zeros to be added to extreme positions. This is easily accomplished by clearing E and circulating AC and E. Thus a logical shift right operation need two instruction :

CLE
CIR

For a logical shift left operation we need two instructions :

- CLE
- CIL

For a basic computer we have adopted signed 2's complement representation :

Arithmetic shift	Logical shift
<ol style="list-style-type: none"> 1. An arithmetic shift via micro operation that shifts a signed binary number to the left and right. 2. An arithmetic shift left multiplies assigned binary number by. An arithmetic shift right divides number by 2. 3. For example : R1 ← Shift R1 R2 ← Shift R2 an micro operations that specify a 1-bit shift to left of content of register R1 and 1-bit shift to right of content of register R2. 4. The register symbol must be same on both sides of the arrow. 5. The bit transferred to the end position through serial input is assumed to be 0 during a logical shift. 	<ol style="list-style-type: none"> 1. A logical shift is one that transforms through the serial input. 2. The symbols strand and for logical shift left and logical shift right microoperations. 3. For example :  <p>Sign bit</p> <p>The left most bit is a register that holds sign bit and remaining bits and remaining bits hold the number.</p> 4. It does not follow the conditions as followed by arithmetic shift micro operations. 5. The bit is not tranferred to the end position through serial input, since it follow difference arithmetic logic.

Q 29. Discuss Booth Multiplication algorithm with the help of suitable example.

(PTU, Dec. 2011, 2010, 2008 ; May 2005)

Ans. Booth Multiplication Algorithm : Booth algorithm gives a procedure for multiplying binary integers in signed-2's complement representation. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting, and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as $2^{k+1} - 2^m$. For example, the binary number 001110 (+14) has a string of 1's from 2^3 to 2^1 ($k = 3, m = 1$). The number can be represented as $2^{k+1} - 2^m = 2^4 - 2^1 = 16 - 2 = 14$. Therefore, the multiplication $M \times 14$, where M is the multiplicand and 14 the multiplier, can be done as $M \times 2^4 - M \times 2^1$. Thus the product can be obtained by shifting the binary multiplicand M four times to the left and subtracting M shifted left once.

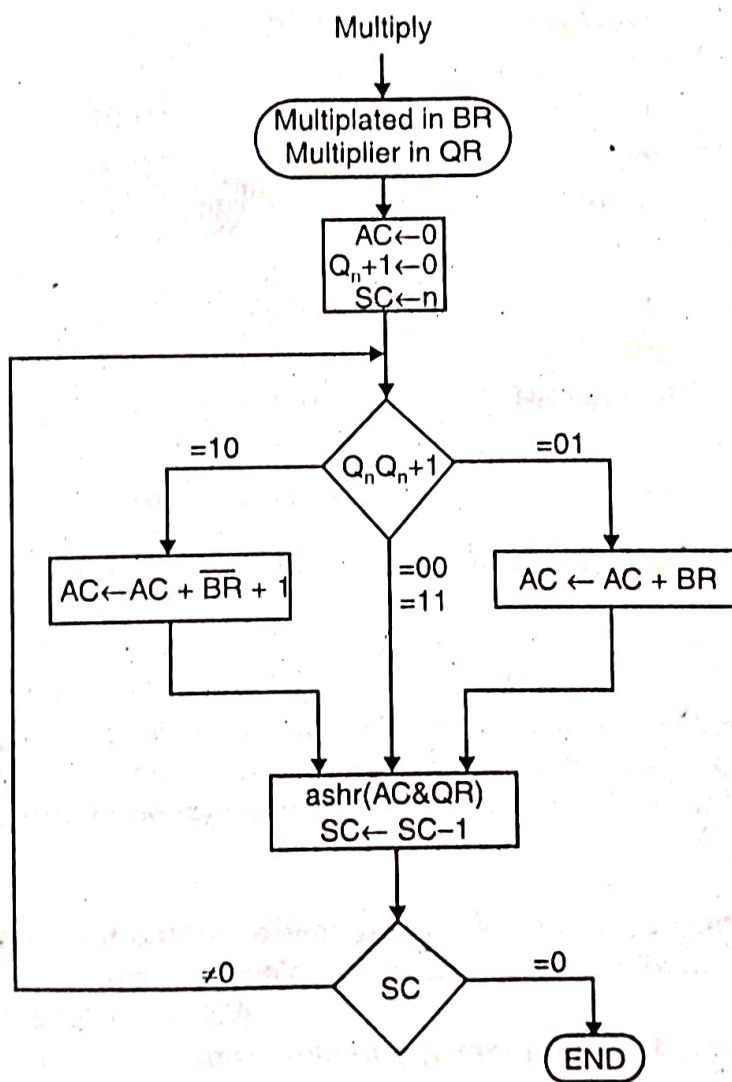
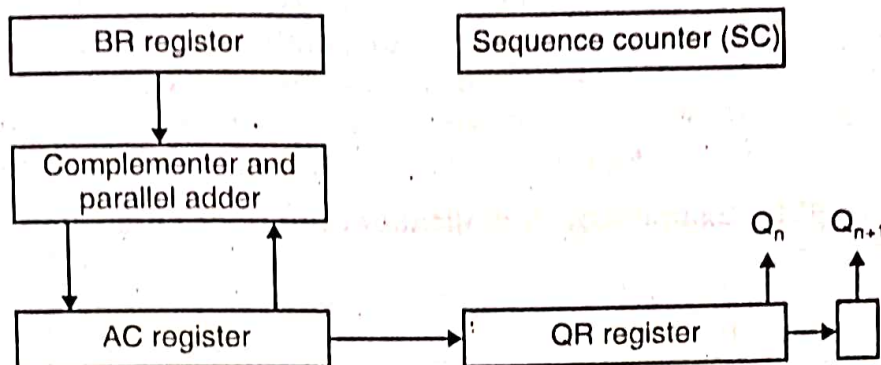
As in all multiplication schemes, Booth algorithm requires examination of the multiplier bits and shifting of the partial product, Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to the following rules :

1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier.
2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous 1) in a string of 0's in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

The algorithm works for positive or negative multipliers in 2's complement representation. This is because a negative multiplier ends with a string of 1's and the last operation will be a subtraction of

the appropriate weight. For example, a multiplier equal to -14 is represented in 2's complement as 110010 and is treated as $-2^4 + 2^2 - 2^1 = -14$.

The hardware implementation of Booth algorithm requires the register configuration shown in Fig. This is similar to Fig. except that the sign bits are not separated from the rest of the registers. To show this difference, we rename registers A, B, and Q, as AC, BR, and QR, respectively. Q_n designates the least significant bit of the multiplier in register QR. An extra flip-flop Q_{n+1} is appended to QR to facilitate a double bit inspection of the multiplier. The flowchart for Booth algorithm is shown in Fig. AC and the appended bit Q_{n+1} are initially cleared to 0 and the sequence counter SC is set to a number n , equal to the number of bits in the multiplier.



The two bits of the multiplier in Q_n and Q_{n+1} are inspected. If the two bits are equal to 10, it means that the first 1 in a string of 1's has been encountered. This requires a subtraction of the multiplicand from the partial product in AC. If the two bits are equal to 01, it means that the first 0 in a string of 0's has been encountered. This requires the addition of the multiplicand to the partial product in AC. When the two bits are equal, the partial product does not change. An overflow cannot occur because the addition and subtraction of the multiplicand follow each other. As a consequence, the two numbers that are added always have opposite signs, a condition that excludes an overflow. The next step is to shift right the partial product and the multiplier (including bit Q_{n+1}). This is an arithmetic shift right (ashr) operation which shifts AC and QR to the right and leaves the sign bit in AC unchanged (see Sec. 4-6). The sequence counter is decremented and the computational loop is repeated n times.

A numerical example of Booth algorithm is shown in Table for $n = 5$. It shows the step-by-step multiplication of $(-9) \times (-13) = +117$. Note that the multiplier in QR is negative and that the multiplicand in BR is also negative. The 10-bit product appears in AC and QR and is positive. The final value of Q_{n+1} is the original sign bit of the multiplier and should not be taken as part of the product.

TABLE : Example of Multiplication with Booth Algorithm

Q_n	Q_{n+1}	BR = 10111 $\overline{BR} + 1 = 01001$	AC	QR	Q_{n+1}	SC
		Initial	00000	10011	0	101
1	0	Subtract BR	<u>01001</u> 01001			
		ashr	00100	11001	1	100
1	1	ashr	00010	01100	1	011
0	1	Add BR	<u>10111</u> 11001			
		ashr	11100	10110	0	010
0	0	ashr	11110	01011	0	001
1	0	Subtract BR	<u>01001</u> 00111			
		ashr	00011	10101	1	000

Q 30. What are the two instructions needed in the basic computer in order to set the E flip flop to 1?

Ans. Electronics a flip-flop or latch is a circuit that has two stable states and can be used to store state information. The circuit can be made to change state by signals applied to one or more control input and will have one two output flip-flop and latches are fundamental building block of digital electronics systems used in computers, communications and many other types of systems.

Q 31. Explain addition subtraction algorithm with signed magnitude data, also give the hardware implementation.

OR

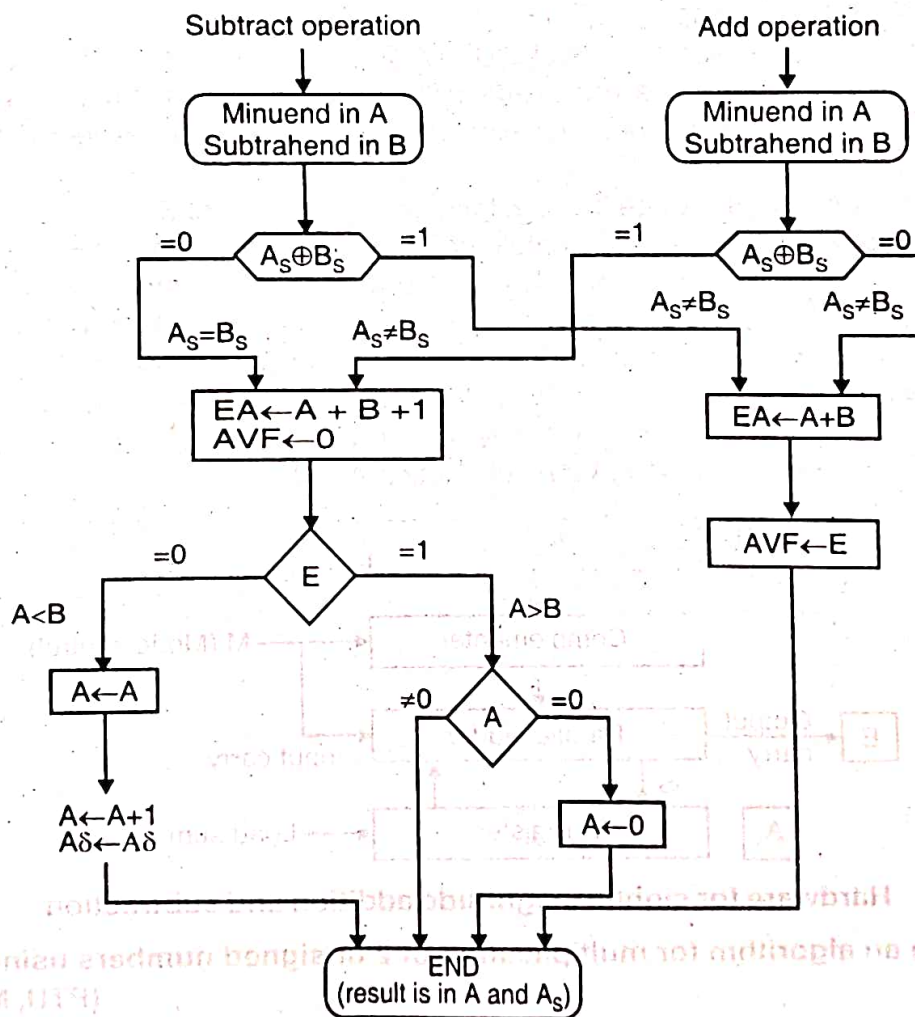
With the help of a flow chart discuss the process of subtraction of floating point numbers. Also explain the issues involved in its hardware implementation.

Ans. Flowchart for add and subtract operations : The flowchart for the hardware algorithm is

(PTU, May 2012, 2006 ; Dec. 2017, 2004)

presented in Fig. The two signs A, and B, are compared by an exclusive-OR gate. If the output of the gate is 0, the signs are identical; if it is 1, the signs are different. For an add operation, identical signs indicate that the magnitudes be added. For a subtract operation, different signs indicate that the magnitudes be added. The magnitudes are added with a microoperation $EA \leftarrow A + B$, where EA is a register that combines E and A. The carry in E after the addition constitutes an overflow if it is equal to 1. The value of E is transferred into the add-overflow flip-flop AVF.

The two magnitudes are subtracted if the signs are different for an add operation or identical for a subtract operation. The magnitudes are subtracted by adding A to the 2's complement of B. No overflow can occur if the numbers are subtracted so AVF is cleared to 0. A 1 in E indicates that $A \geq B$ and the number in A is the correct result, If this number is zero, the sign A_s must be made positive to avoid a negative zero. A 0 in E indicates that $A < B$. For this case it is necessary to take the 2's complement of the value in A. This operation can be done with one microoperation $A \leftarrow \bar{A} + 1$. However, we assume that the A register has circuits for microoperations complement and increment, so the 2's complement is obtained from these two microoperations. In other paths of the flowchart, the sign of the result is the same as the sign of A, so no change in A, is required. However, when $A < B$, the sign of the result is the complement of the original sign of A. It is then necessary to complement A, to obtain the correct sign. The final result is found in register A.



Flowchart for add and subtract operations

Hardware Implementation : To implement the two arithmetic operation with hardware, it is first necessary that the two numbers be stored in registers. Let A and B be two registers that hold the magnitudes of the numbers, and A_s and B_s be two flip-flop that hold the corresponding signs. The result of the operation may be transferred to a third register: however, a saving is achieved if the result is transferred into A and A_s . Thus A and A_s together form an accumulator register.

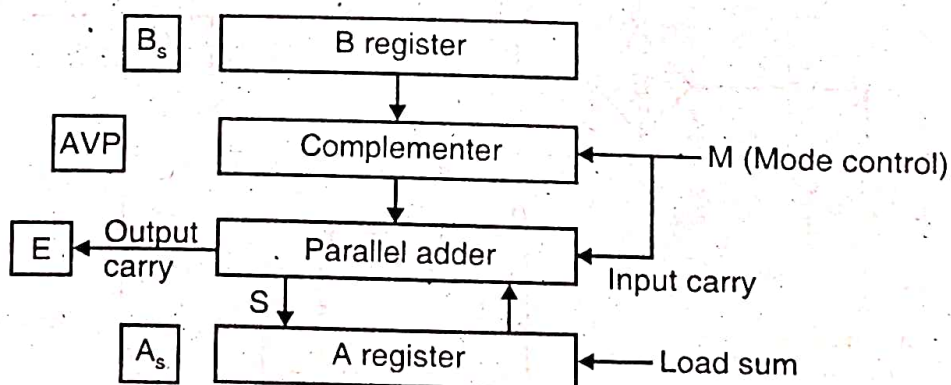
Consider now the hardware implementation of the algorithms above. First, a parallel-adder is needed to perform the microoperation $A + B$. Second, a comparator circuit is needed to establish if $A > B$, $A = B$, or $A < B$. Third, two parallel-subtractor circuits are needed to perform the microoperations $A - B$ and $B - A$. The sign relationship can be determined from an exclusive OR gate with A_s and B_s inputs.

This procedure requires a magnitude comparator, an adder, and two subtractors. However, a different procedure can be found that required less equipment. First, we know that subtraction can be accomplished by means of complement and add. Second, the result of a comparison can be determined from the end carry after the subtraction. Careful investigation of the alternatives reveals that the use of 2's complement for subtraction and comparison is an efficient procedure that requires only an adder and a complementer.

Fig. shows a block diagram of the hardware of the hardware for implementing the addition and subtraction operations. It consists of registers A and B and sign flip-flop A_s and B_s . Subtraction is done by adding A to the 2's complement of B.

The output carry is transferred to flip-flop E, where it can be checked to determine the relative magnitudes of the two numbers. The add-overflow flip-flop. AVF holds the overflow bit when A and B are added. The A register provides other microoperations that may be needed when we specify the sequence of steps in the algorithm.

The addition of A plus B is done through the parallel adder. The S (sum) output of the adder is applied to the input of the A register. The complementer provides an output of B or the complement of B depending on the state of the mode control M. The complement consists of exclusive-OR gates and the parallel adder consists of full-adder circuits as shown in Fig. in Chap 4. The M signal is also applied to the input carry of the adder. When $M = 0$, the output of B is transferred to the adder, the input carry is 0, and the output of the adder is equal to the sum $A + B$. When $M = 1$, the 1's complement of B is applied to the adder, the input carry is 1, and output $S = A + B + 1$. This is equal to A plus the 2's complement of B, which is equivalent to the subtraction $A - B$.

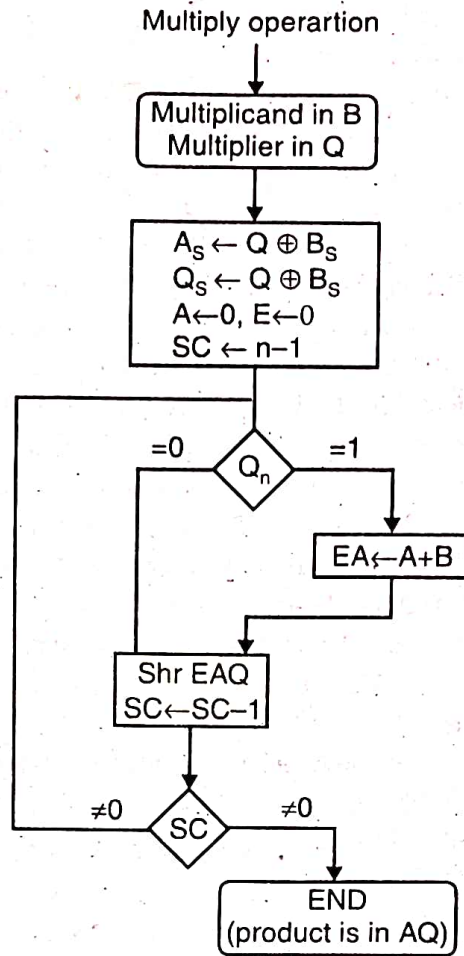


Hardware for signed magnitude addition and subtraction

Q 32. Write an algorithm for multiplication of 2 unsigned numbers using shift and add method.

Ans.

(PTU, May 2009, 2004)



Initially, multiplicand is in register B and multiplier in Q. The sum of A and B forms a partial product which is transferred to the EA register. Both partial product and multiplier are shifted to right. The sequence counter SC is initially set to a no equal to no of bits in multiplier. The counter is decremental by 1 after forming each partial product. When content of counter reaches zero, the product is formed and process stops. The best significant bit of A is shifted into the most significant position if g. the of bit from E is shifted to MS most signitacat position of A and O is shifted into 'E'.

e.g.

	E	A	Cs	SC
Multiplicend B = 10111				
Multiplier in Cs	0	00000	10011	101
$\theta_M = 1$; add B		<u>10111</u>		
I st partial product	0	10111		
Shift right E A Cs	0	01011	11001	100
$Q_n = 1$, add B		<u>10111</u>		
II nd partial product	1	00010		
Shift right E A Q	0	10001	01100	011
$Q_n = 0$; shift right EA Q	0	01000	10110	010
$Q_n = 0$; shift right EA Q	0	00100	01011	001
$Q_n = 1$; add B		<u>10111</u>		
Fifth Partial product	0	11011		
Shift right E A Q	0	01101	10101	000
Final product = 0110110101				

Q 33. Write an Algorithm of multiplication.

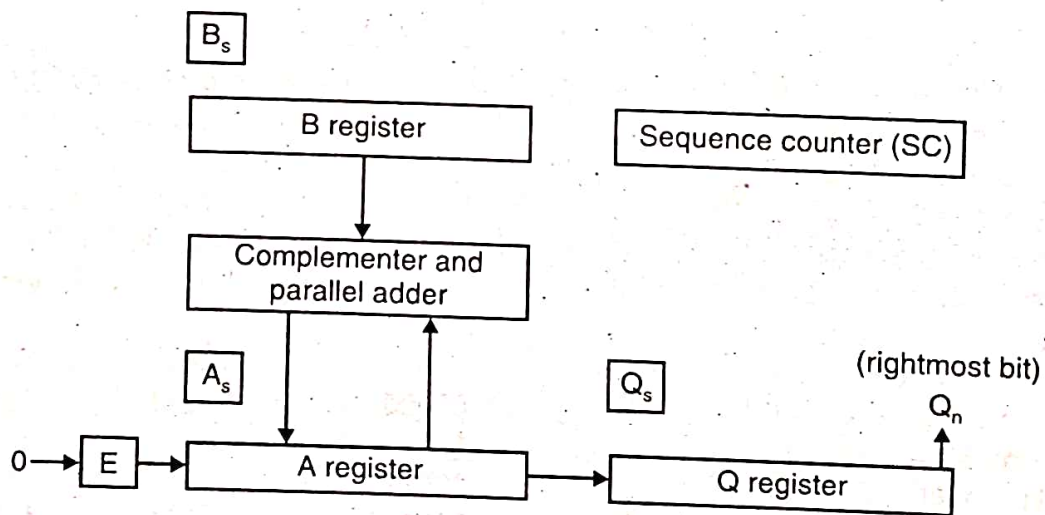
(PTU, Dec. 2006)

Ans. Algorithm of Multiplication.

Hardware Implementation for Signed-Magnitude Data : When multiplication is implemented in a digital computer, it is convenient to change the process slightly. First, instead of providing registers to store and add simultaneously as many binary numbers as there are bits in the multiplier, it is convenient to provide an adder for the summation of only two binary numbers and successively accumulate the partial products in a register. Second, instead of shifting the multiplicand and to the left, the partial product is shifted to the right, which results in leaving the partial product and the multiplicand in the required relative positions. Third, when the corresponding bit of the multiplier is 0, there is no need to add all zeros to the partial product since it will not alter its value.

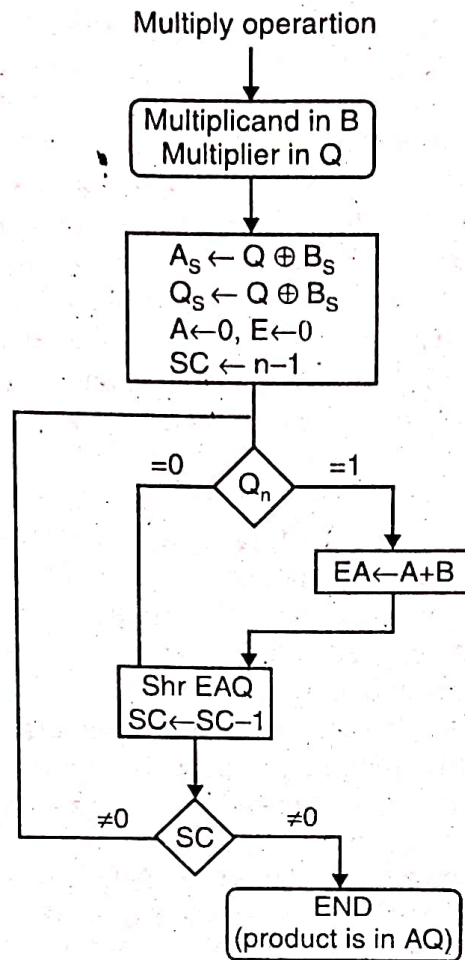
The hardware for multiplication consists of the equipment shown in Fig. plus two more registers. These registers together with registers A and B are shown in Fig. The multiplier is stored in the Q register and its sign in Q_s . The sequence counter SC is initially set to a number equal to the number of bits in the multiplier. The counter is decremented by 1 after forming each partial product. When the content of the counter reaches zero, the product is formed and the process stops.

Initially, the multiplicand is in register B and the multiplier in Q. The sum of A and B forms a partial product which is transferred to the EA register. Both partial product and multiplier are shifted to the right. This shift will be denoted by the statement shr EAQ to designate the right shift depicted in Fig. The least significant bit of A is shifted into the most significant position of Q, the bit from E is shifted into the most significant position of A, and 0 is shifted into E. After the shift, one bit of the partial product is shifted into Q, pushing the multiplier bits one position to the right. In this manner, the rightmost flip-flop in register Q, designated by Q_n , will hold the bit of the multiplier, which must be inspected next.



Hardware for multiply operation

Hardware Algorithm : Fig. is a flowchart of the hardware multiply algorithm. Initially, the multiplicand is in B and the multiplier in Q. Their corresponding signs are in B_s and Q_s , respectively. The signs are compared, and both A and Q are set to correspond to the sign of the product since a double-length product will be stored in registers A and Q. Registers A and E are cleared and the sequence counter SC is set to a number equal to the number of bits of the multiplier. We are assuming here that operands are transferred to registers from a memory unit that has words of a bits. Since an operand must be stored with its sign, one bit of the word will be occupied by the sign and the magnitude will consist of $n-1$ bits.



Flowchart for multiply operation

After the initialization, the low-order bit of the multiplier in Q_n is tested. If it is a 1, the multiplicand in B is added to the present partial product in A. If it is a 0, nothing is done. Register EAQ is then shifted once to the right to form the new partial product. The sequence counter is decremented by 1 and its new value checked. If it is not equal to zero, the process is repeated and a new partial product is formed. The process stops when $SC = 0$. Note that the partial product formed in A is shifted into Q one bit at a time and eventually replaces the multiplier. The final product is available in both A and Q, with A holding the most significant bits and Q holding the least significant bits.

Q 34. What are the various issues for designing the instruction set of a processor?

(PTU, May 2009)

Ans. Instruction Set : An instruction set, or instruction set architecture (ISA), is the part of the computer architecture related to programming, including the native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O. An ISA includes a specification of the set of opcodes (machine language), and the native commands implemented by a particular processor.

Instruction set architecture is distinguished from the microarchitecture, which is the set of processor design techniques used to implement the instruction set. Computers with different microarchitectures can share a common instruction set. For example, the Intel Pentium and the AMD Athlon implement nearly identical versions of the x86 instruction set, but have radically different internal designs.

This concept can be extended to unique ISAs like TIMI (Technology-Independent Machine

Interface) present in the IBM System/38 and IBM AS/400. TIMI is an ISA that is implemented by low-level software translating TIMI code into "native" machine code, and functionally resembles what is now referred to as a virtual machine. It was designed to increase the longevity of the platform and applications written for it, allowing the entire platform to be moved to very different hardware without having to modify any software except that which translates TIMI into native machine code, and the code that implements services used by the resulting native code. This allowed IBM to move the AS/400 platform from an older CISC architecture to the newer POWER architecture without having to rewrite or recompile any parts of the OS or software associated with it other than the aforementioned low-level code. Some virtual machines that support bytecode for Smalltalk, the Java virtual machine, and Microsoft's Common Language Runtime virtual machine as their ISA implement it by translating the bytecode for commonly-used code paths into native machine code, and executing less-frequently-used code paths by interpretation; Transmeta implemented the x86 instruction set atop VLIW processors in the same fashion.

Instruction set implementation

Any given instruction set can be implemented in a variety of ways. All ways of implementing an instruction set give the same programming model, and they all are able to run the same binary executables. The various ways of implementing an instruction set give different tradeoffs between cost, performance, power consumption, size, etc.

When designing microarchitectures, engineers use blocks of "hard-wired" electronic circuitry (often designed separately) such as adders, multiplexers, counters, registers, ALUs etc. Some kind of register transfer language is then often used to describe the decoding and sequencing of each instruction of an ISA using this physical microarchitecture. There are two basic ways to build a control unit to implement this description (although many designs use middle ways or compromises) :

1. Early computer designs and some of the simpler RISC computers "hard-wired" the complete instruction set decoding and sequencing (just like the rest of the microarchitecture).
2. Other designs employ microcode routines and/or tables to do this-typically as on chip ROMs and/or PLAs (although separate RAMs have been used historically).

There are also some new CPU designs which compile the instruction set to a writable RAM or FLASH inside the CPU (such as the Rekursiv processor and the Imsys Cjip)[1], or an FPGA (reconfigurable computing). The Western Digital MCP-1600 is an older example, using a dedicated, separate ROM for microcode.

An ISA can also be emulated in software by an interpreter. Naturally, due to the interpretation overhead, this is slower than directly running programs on the emulated hardware, unless the hardware running the emulator is an order of magnitude faster. Today, it is common practice for vendors of new ISAs or microarchitectures to make software emulators available to software developers before the hardware implementation is ready.

Often the details of the implementation have a strong influence on the particular instructions selected for the instruction set. For example, many implementations of the instruction pipeline only allow a single memory load or memory store per instruction, leading to a load-store architecture (RISC). For another example, some early ways of implementing the instruction pipeline led to a delay slot.

The demands of high-speed digital signal processing have pushed in the opposite direction-forcing instructions to be implemented in a particular way. For example, in order to perform digital filters fast enough, the MAC instruction in a typical digital signal processor (DSP) must be implemented using a kind of Harvard architecture that can fetch an instruction and two data words simultaneously, and it requires a single-cycle multiply-accumulate multiplier.

Instruction set design : Some instruction set designers reserve one or more opcodes for

some kind of software interrupt. For example, MOS Technology 6502 uses 00H, Zilog Z80 uses the eight codes C7,CF,D7,DF,E7,EF,F7,FFH[2] while Motorola 68000 use codes in the range A000..AFFFH.

Fast virtual machines are much easier to implement if an instruction set meets the Popek and Goldberg virtualization requirements.

The NOP slide used in Immunity Aware Programming is much easier to implement if the "unprogrammed" state of the memory is interpreted as a NOP.

On systems with multiple processors, non-blocking synchronization algorithms are much easier to implement if the instruction set includes support for something like "fetch-and-increment" or "load linked/store conditional (LL/SC)" or "atomic compare and swap".

Code density : In early computers, program memory was expensive, so minimizing the size of a program to make sure it would fit in the limited memory was often central. Thus the combined size of all the instructions needed to perform a particular task, the *code density*, was an important characteristic of any instruction set. Computers with high code density also often had (and have still) complex instructions for procedure entry, parameterized returns, loops etc. (therefore retroactively named *Complex Instruction Set Computers*, CISC). However, more typical, or frequent, "CISC" instructions merely combine a basic ALU operation, such as "add", with the access of one or more operands in memory (using addressing modes such as direct, indirect, indexed etc.). Certain architectures may allow two or three operands (including the result) directly in memory or may be able to perform functions such as automatic pointer increment etc. Software-implemented instruction sets may have even more complex and powerful instructions.

Reduced instruction-set computers, RISC, were first widely implemented during a period of rapidly-growing memory subsystems and sacrifice code density in order to simplify implementation circuitry and thereby try to increase performance via higher clock frequencies and more registers. RISC instructions typically perform only a single operation, such as an "add" of registers or a "load" from a memory location into a register; they also normally use a fixed instruction width, whereas a typical CISC instruction set has many instructions shorter than this fixed length. Fixed-width instructions are less complicated to handle than variable-width instructions for several reasons (not having to check whether an instruction straddles a cache line or virtual memory page boundary[3] for instance), and are therefore somewhat easier to optimize for speed. However, as RISC computers normally require more and often longer instructions to implement a given task, they inherently make less optimal use of bus bandwidth and cache memories.

Minimal instruction set computers (MISC) are a form of stack machine, where there are few separate instructions (16-64), so that multiple instructions can be fit into a single machine word. These type of cores often take little silicon to implement, so they can be easily realized in an FPGA or in a multi-core form. Code density is similar to RISC; the increased instruction density is offset by requiring more of the primitive instructions to do a task.

Issues in compiling

It is easy to interpret the individual instructions of a stack-based virtual machine (SM) such as the Java virtual machine (JVM), because most operands are implicit. However, a high-level language implementation of the interpreter generates more memory traffic than the equivalent set of instructions in a memory transfer machine (MM) such as Dis. Consider the code to execute

$$c = a + b;$$

An SM would execute this by a code burst such as this, which we have annotated with its memory traffic using *L* for load and *S* for store :

```

push a      # LS
push b      # LS
add         # LLS
store c     # LS

```

The corresponding MM code burst would be the plain three-operand instruction

```
add a,b,c # LLS
```

When interpreting, the extra memory traffic of the SM is masked by the time saved by not decoding any operand fields. The operand fields are implicit in the SM instructions, while the MM they are explicit: three operand fields must be decoded in every instruction, even those without operands.

When compiling, the tradeoffs are different. Clearly, either design can produce the same native instructions from its just-in-time compiler (JIT), but in the SM case most of the work must be done in the JIT, whereas in the MM design the front end has done most of the work and the JIT can be substantially simpler and faster.

A JIT for an SM is forced to do most of the work of register allocation in the JIT itself. Because the types of stack cells change as the program executes, the JIT must track their types as it compiles. In an MM, however, the architecture maps well to native instructions. This produces a continuum of register allocation strategies from none, to simple mapping of known cells to registers, to flow-based register allocation. Most of the work of any of these strategies can be done in the language-to-VM compiler. It can generate code for an infinite-register machine, and the JIT can then allocate as many as are available in the native architecture.

Q 35. Explain the working of floating point adder/subtractor.

(PTU, May 2010)

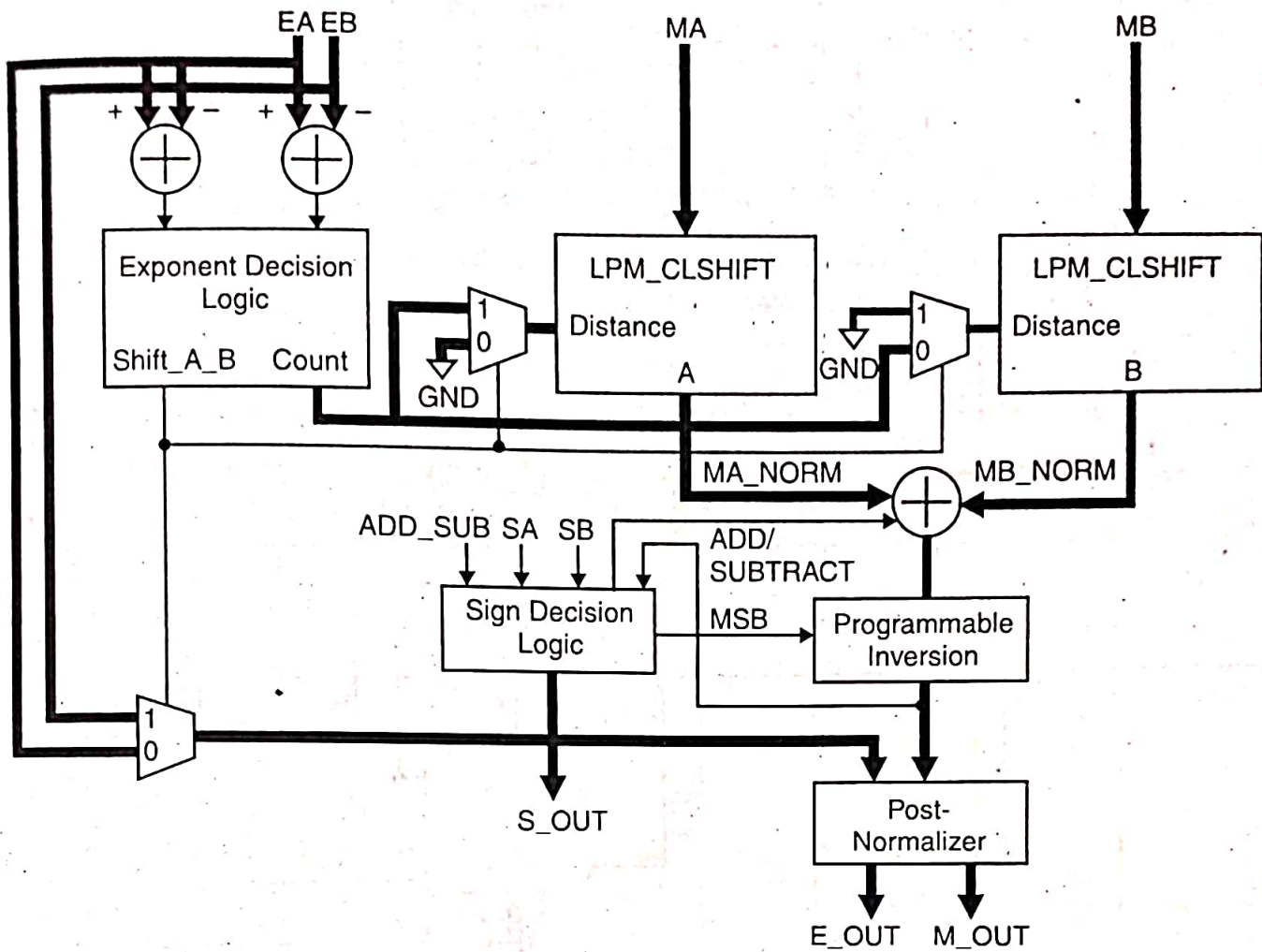
Ans. To add or subtract two floating-point numbers, the mantissas must be aligned. Then, the exponents must be compared to determine which number is larger. If the difference between the exponents is slight, the number with the smaller exponent may be larger if mantissas MA and MB are not normalized, which reduces the functions precision. To avoid this problem, designers should ensure that all inputs are normalized by making sure that the MSB of each mantissa is 1.

The relative values of the exponents are checked by subtracting one exponent from the other. The mantissa with the larger exponent is retained, and the mantissa with the smaller exponent is right-shifted until the radix point is properly aligned (i.e., until the exponents are equal). If the exponents differ by more than the number of bits in the mantissa, the smaller number becomes insignificant. The shifting is performed by the LPM function `lpm_clshift`.

After the mantissas pass through the shifters, an unsigned integer adder/subtractor performs an operation that is determined by the sign of the inputs (`sa` and `sb`) and the `add_sub` port. The result of the adder is then passed through a programmable inverter, which is controlled by the sign decision logic. This process ensures that the mantissa has the proper sign. After the addition or subtraction has taken place, the post-normalizer normalizes the result, if necessary, by adjusting the mantissa and exponent of the result so that the MSB of the mantissa is 1.

Input and Output Ports

Port Type	Name	Description
Input	<code>sa</code>	Sign bit for the a input : 1 = positive, 0 = negative
Input	<code>ma [mantissa_width1]</code>	Mantissa for the a input
Input	<code>ea [exponent_width1]</code>	Exponent for the a input
Input	<code>sb</code>	Sign bit for the b input : 1 = positive, 0 = negative
Input	<code>mb [mantissa_width1]</code>	Mantissa for the b input
Input	<code>eb [exponent_width1]</code>	Exponent for the b input
Input	<code>add_sub</code>	Operation : 1 = add, 0 = subtract
Output	<code>m_out [mantissa_width1]</code>	Mantissa for the output
Output	<code>e_out [exponent_width1]</code>	Exponent for the output
Output	<code>s_out</code>	Sign bit for the output : 1 = positive, 0 = negative



Block Diagram Floating-Point Adder/Subtracted

Q 36. What is the use of Condition Code register?

(PTU, May 2010)

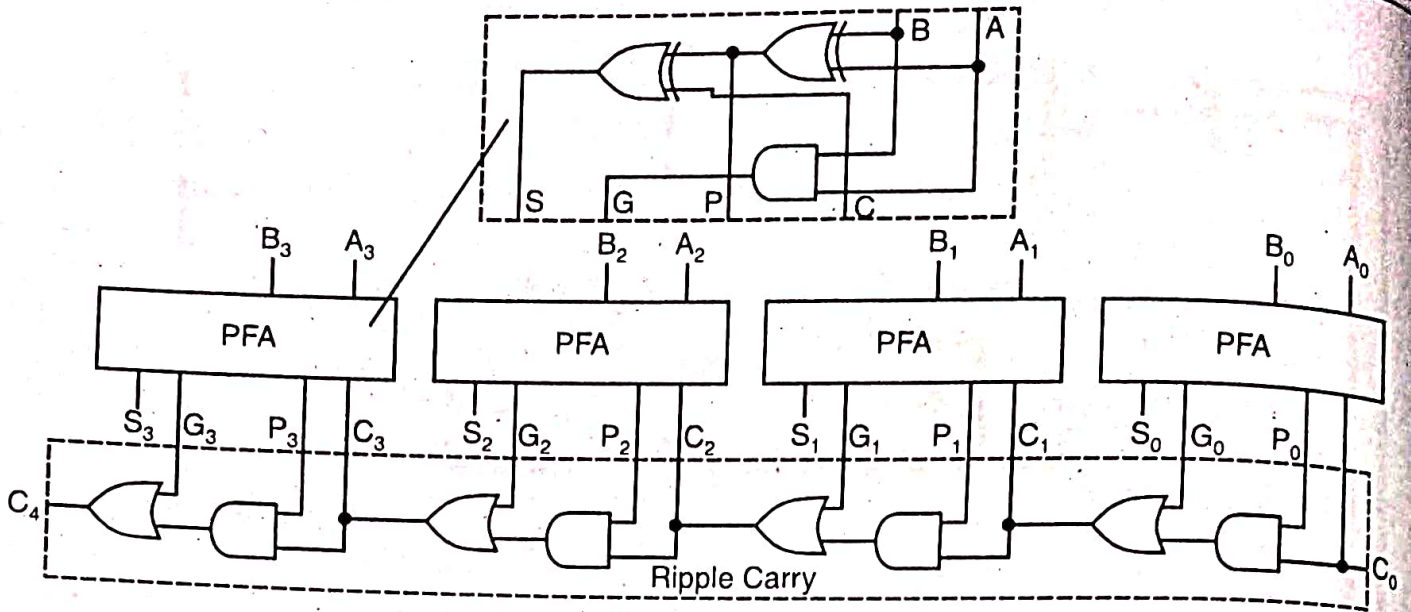
Ans. Condition-code register (qualifier register) : A set of indicators that records the status or condition of a previous result output from the ALU. It forms part of the program status word which is a collection of information that encapsulates the basic execution state of a program at any instant. It permits an interrupted process to resume operation after the interrupt has been handled. The information is held in the program status register, and usually contains the value of the program counter and bits indicating the status of various conditions in the ALU.

Q 37. Explain in detail the principle of carry-look-ahead adder. Show how 16-bit CLAs can be constructed from 4-bit address.

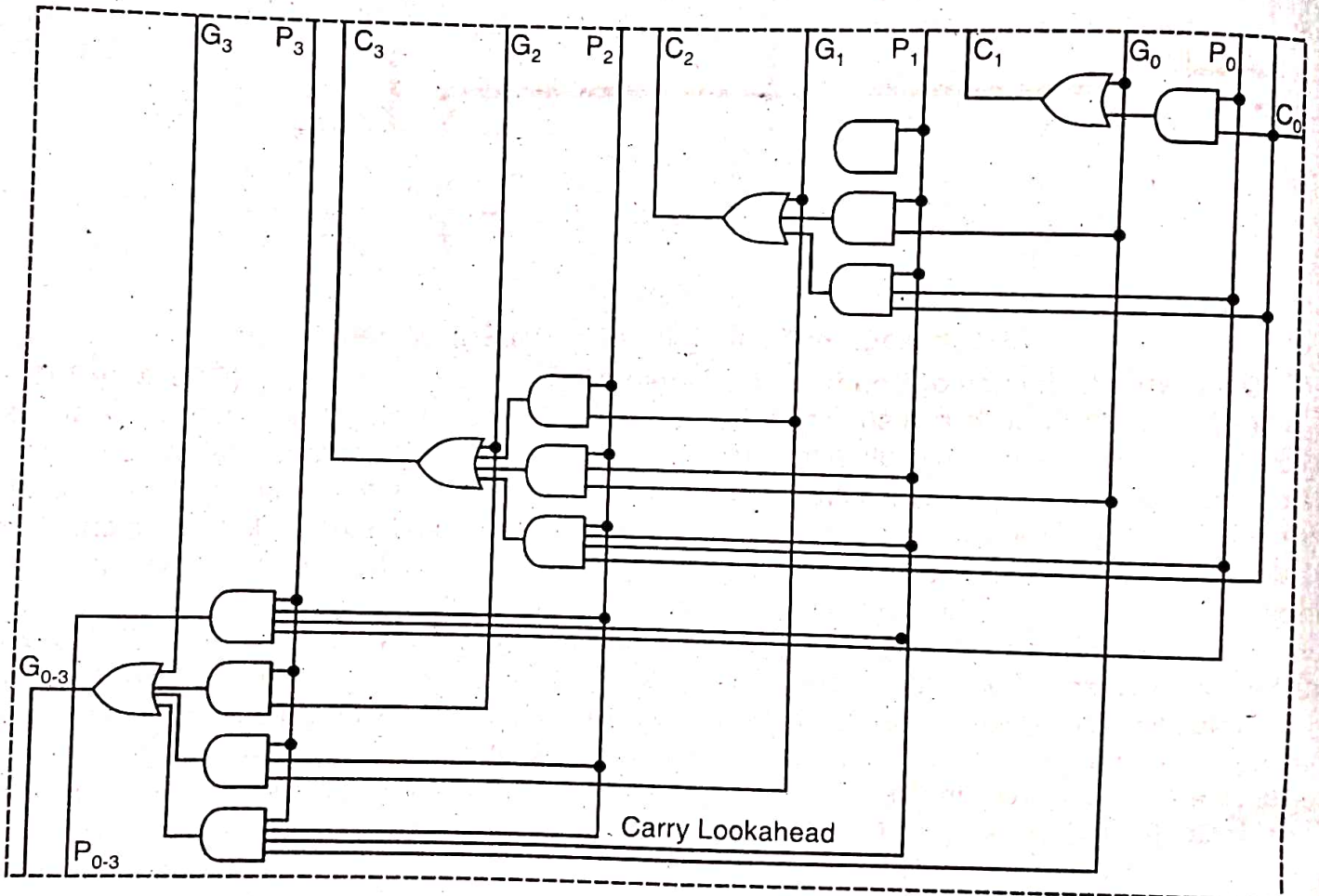
(PTU, May 2010, 2009)

Solution. The ripple carry adder, through simple in concept, has a long circuit delay due to the many gates in the carry path from the least significant bit to the most significant bit. For a typical design, the longest delay path through an n-bit ripple carry adder is approximately $2n + 2$ gate delays. Thus, for a 16-bit ripple carry adder, the delay is 34 gate delays. This delay tends to be one of the largest in a typical computer design. Accordingly, we find an alternative design, the carry lookahead adder, attractive. This adder is a practical design with reduced delay at the price of more complex hardware. The carry lookahead design can be obtained by a transformation of the ripple carry design into a design in which the carry logic over fixed groups of bits of the adder is reduced to two-level logic. The transformation is shown for a 4-bit adder group in fig.

First, we construct a new logic hierarchy, separating the parts of the full adders not involving the carry propagation path from those containing the path.



(a)



(b) Development of Carry-Look Ahead Adder

We call the first part of each full adder a partial full adder (PFA). This separation is shown 1 (a), which presents a diagram of a PFA and a diagram of four PFAs connected to the carry path. We have removed the OR gate and one of the AND gates from each of the full adders to form the ripple carry path.

There are two outputs, P_i and G_i , from each PFA to the ripple carry path and one output C_i , the carry input, from the carry path to each PFA. The function $P_i = A_i B_i$ is called the propagate function. Whenever P_i is equal to 1, an incoming carry is propagated through the bit position from C_i to C_{i+1} . For P_i equal to 0, carry propagation through the bit position is blocked. The function $G_i = A_i + B_i$ and is called the generate function. Whenever G_i is equal to 1, the carry output from the position is 1, regardless of the value of P_i , so a carry has been generated in the position. When G_i is 0, a carry is not generated, so that C_{i+1} is 0 if the carry propagated through the position from C_i is also 0. The generate and propagate functions correspond exactly to the half adder and are essential in controlling the values in the ripple carry path. Also, as in the full adder, the PFA generates the sum function by the exclusive-OR of the incoming carry C_i and the propagate function P_i .

The carry path remaining in the 4-bit ripple carry adder has a total of eight gates in cascade, so the circuit has a delay of eight gate delays. Since only AND and OR gates are involved in the carry path, ideally, the delay for each of the four carry signals produced, C_1 through C_4 , would be just two gate delays. The basic carry lookahead circuit is simply a circuit in which functions C_1 through C_3 have a delay of only two gate delays. The implementation of C_4 is more complicated in order to allow the 4-bit carry lookahead adder to be extended to multiples of 4 bits, such as 16 bits. The 4-bit carry lookahead circuit is shown in fig. 1 (b). It is designed to directly replace the ripple carry path in fig. 1 (a). Since the logic generating C_1 is already two-level, it remains unchanged. The logic for C_2 , however, has four levels. So to find the carry lookahead logic for C_2 , we must reduce the logic to two levels. The equation for C_2 is found from fig. 1 (a) and the distributive law is applied to obtain

$$\begin{aligned} C_2 &= G_1 + P_1 (G_0 + P_0 C_0) \\ &= G_1 + P_1 G_0 + P_1 P_0 C_0 \end{aligned}$$

This equation is implemented by the logic with output C_2 in figure 5-6 (b). We obtain the two-level logic for C_3 by finding its equation from the carry path in figure 1 (a) and applying the distributive law :

$$\begin{aligned} C_3 &= G_2 + P_2 (G_1 + P_1 (G_0 + P_0 C_0)) \\ &= G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_0) \\ &= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0 \end{aligned}$$

The two-level logic with output C_3 in fig. 1(b) implements this function.

We could implement C_4 using the same method. But some of the gates would have a fan-in of five, which may increase the delay. Also, we are interested in reusing this same circuit for higher numbered bits (e.g., 4 through 7, 8 through 11, and 12 through 15 of a 16-bit adder). For this adder, in positions 4, 8 and 12 we would like the carry to be produced as fast as possible without using excessive fan-in. Accordingly, we want to repeat the same carry lookahead trick for 4-bit groups that we used to handle the 4 bits. This will allow us to reuse the carry lookahead circuit design for each group of 4 bits, and also to use the same circuit for four 4-bit groups as if they were individual bits. So instead of generating C_4 , we produce generate and propagate functions that apply to 4-bit groups instead of a single bit to act as the inputs for the group carry lookahead circuit. To propagate a carry from C_0 to C_4 , we need to have all four of the propagate functions equal to 1, giving the group propagate function.

To represent the generation of a carry in positions 0, 1, 2 and 3, and its propagation to C_4 , we need to consider the generation of a carry in each of the positions, as represented by G_0 through G_3 , and the propagation of each of these four generated carries to position 4. This gives the group generate function. The group propagate and group generate equations are implemented by the logic in the lower part of fig. 1 (b). If there are only four bits in the adder, then the logic circuit used for C_1 can be used to generate C_4 from these two outputs; we will later refer to the C_1 logic block as OC (Output Carry) for generating the output carry from an adder, in this case, C_4 .

In a longer adder, for $4n$ bits, where $n = 2, 3$ and so on, one or more carry lookahead circuits identical to that in the figure, except for labelling, are placed at the second level to generate C_4, C_8, C_{12} and so on. As the number of adder bits crosses values equal to $4m$ for $m = 2, 3$ and so on, additional carry-lookahead circuit (CLC) levels are added. Assuming that an exclusive OR contributes 2 gate delays, the longest delay in the 4-bit carry lookahead adder is 6 gate delays, compared with 10 gate delays in the ripple carry adder. The improvement is very modest and perhaps not worth all the extra logic.

Q 38. Write short notes on the following :

(a) Superscalar machines (b) 8255 chip.

(PTU, Dec. 2011, 2010)

Ans. (a) Superscalar Machines : A superscalar CPU architecture implements a form of parallelism called instruction level parallelism within a single processor. It therefore allows faster CPU throughput than would otherwise be possible at a given clock rate. A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor. Each functional unit is not a separate CPU core but an execution resource within a single CPU such as an arithmetic logic unit, a bit shifter, or a multiplier.

In the Flynn taxonomy, a single-core superscalar processor is classified as an SISD processor (Single Instructions, Single Data), while a multi-core superscalar processor is classified as an MIMD processor (Multiple Instructions, Multiple Data).

While a superscalar CPU is typically also pipelined, pipelining and superscalar architecture are considered different performance enhancement techniques.

The superscalar technique is traditionally associated with several identifying characteristics (within a given CPU core) :

- Instructions are issued from a sequential instruction stream
- CPU hardware dynamically checks for data dependencies between instructions at run time (versus software checking at compile time)
- The CPU accepts multiple instructions per clock cycle.

(b) 8255 Chip : The Intel 8255 (or i8255) Programmable Peripheral Interface chip is a peripheral chip originally developed for the Intel 8085 microprocessor, and as such is a member of a large array of such chips, known as the **MCS-85 Family**. This chip was later also used with the Intel 8086 and its descendants. It was later made (cloned) by many other manufacturers. It is made in DIP 40 and PLCC 44 pins encapsulated versions.

The 8255 is widely used not only in many microcomputer/microcontroller systems especially Z-80 based, home computers such as SV-328 and all MSX, but also in the system board of the best known original IBM-PC, PC/XT, PC/jr, etc. and clones, along with numerous homebuilt computer computers such as the N8VEM.

Operational modes of 8255 : There are two basic operational modes of 8255 :

1. Bit Set/Reset Mode (BSR Mode)
2. Input/Output Mode (I/O Mode)

The two modes are selected on the basis of the value present at the D_7 bit of the Control Word Register. When $D_7 = 1$, 8255 operates in I/O mode and when $D_7 = 0$, it operates in the BSR mode.

Bit Set/Reset Mode : The Bit Set/Reset (BSR) mode is applicable to port C only. Each line of port C ($PC_0 - PC_7$) can be set/reset by suitably loading the control word register. BSR mode and I/O mode are independent and selection of BSR mode does not affect the operation of other ports in I/O mode.

- D_7 bit is always 0 for BSR mode.
- Bits D_6, D_5 and D_4 are don't care bits.
- Bits D_3, D_2 and D_1 are used to select the pin of Port C.
- Bit D_0 is used to set/reset the selected pin of Port C.

Q 39. Mention the limitations of 8085.

(PTU, Dec. 2010)

Ans. (i) The lower order address bus of the 8085 microprocessor is multiplexed (time shared) with the data bus. The buses need to be demultiplexed.

(ii) Appropriate control signals need to be generated to interface memory and I/O with the 8085.

Q 40. Describe Booth's multiplication algorithm.

(PTU, May 2011)

Ans. Booth's algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm used desk calculators that were faster at shifting them adding and created the algorithm to increase their speed.

Procedure : Booth's algorithm involves repeatedly adding one of two predetermined value A and S to a product P, then performing a right ward arithmetic shift on P. Let m and r be the multiple C and multiplier, respectively and let x and y represent the number of bits in m and r.

1. Determine the value of A and S, and the initial value of P. All of these numbers should have length equal to $(x + y + 1)$.

(i) **A :** Fill the most significant (leftmost) bits with the value of m. Fill the remaining $(y + 1)$ bits with zeros.

(ii) **S :** Fill the most significant bits with the value of $(-m)$ in two's complement notation. Fill the remaining $(y + 1)$ bits with zeros.

(iii) **P :** Fill the most significant x bits with zeros. To the right of this, append the value of r. Fill the least significant (rightmost) bits with a zero.

2. Determine the two least significant (rightmost) bit of P.

(i) If they are 01, find the value of $P + A$. Ignore any overflow.

(ii) If they are 10, find the value $P + S$. Ignore any overflow.

(iii) If they are 00, do nothing. Use P directly in next step.

(iv) If they are 11, do nothing. Use P directly in the next step.

3. Arithmetically shift : The value obtained in the 2nd step by a single place to the right. Let P now equal this new value.

4. Repeat steps 2 and 3 until they have been done y time.

5. Drop the least significant (rightmost) bit from P. This is the product of m and r.

Q 41. Explain the meaning of the memory-reference instruction LDA.

(PTU, Dec. 2004)

Ans. LDA : Load to AC.

This instruction transfers the memory word specified by the effective address to AC. The micro operations needed to execute this information are.

$DR \leftarrow M[AR]$

$AC \leftarrow DR, SC \leftarrow O$

Q 42. Define Addressing Modes. What are the different types of addressing modes?

(PTU, May 2018, 2007 ; Dec. 2015)

Ans. The addressing mode specifies a rule for interpreting or modifying the address field of Instruction before the operand is actually referenced. Computers use addressing mode techniques for the purpose of accommodating one or both of the following processors.

1. To give programming versatility to the user by providing such facilities as pointers to memory, counters for loop control.

2. To reduce the number of bits in the address field of the Instruction.

The various types of addressing modes are as follow :

1. Implied mode

2. Immediate mode

3. Register mode

- 4. Direct addressing mode
- 5. Indirect addressing mode
- 6. Relative addressing mode

Q 43. Explain with the help of an example, the use of hamming code as error detection and correction code. (PTU, Dec. 2017)

Ans. Hamming code is technique developed by R.W. Hamming for error correction. This method corrects the error by finding the state at which the error has occurred.

Example : Suppose a binary data 1001101 is to be transmitted. To implement hamming code for this, following steps are used :

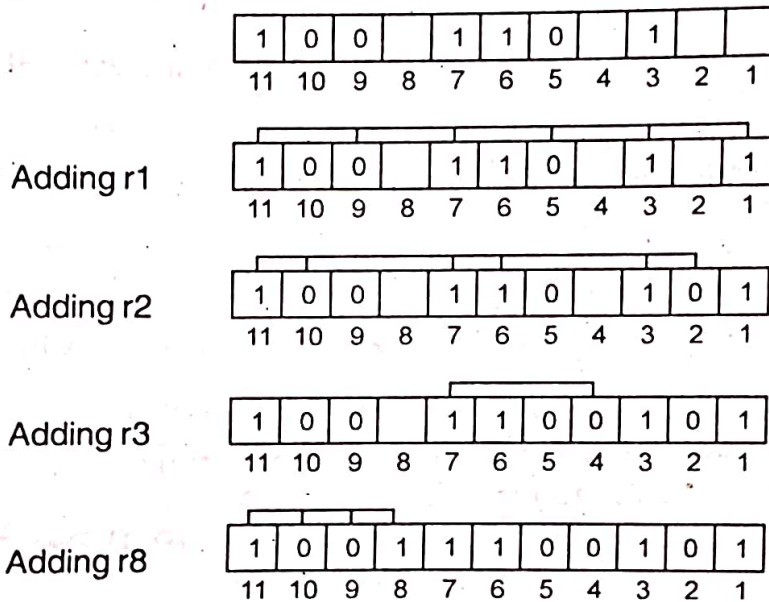
1. Calculating the number of redundancy bits required, Since number of data bits is 7, the value of r is calculated as

$$2^r \geq m + r + 1$$

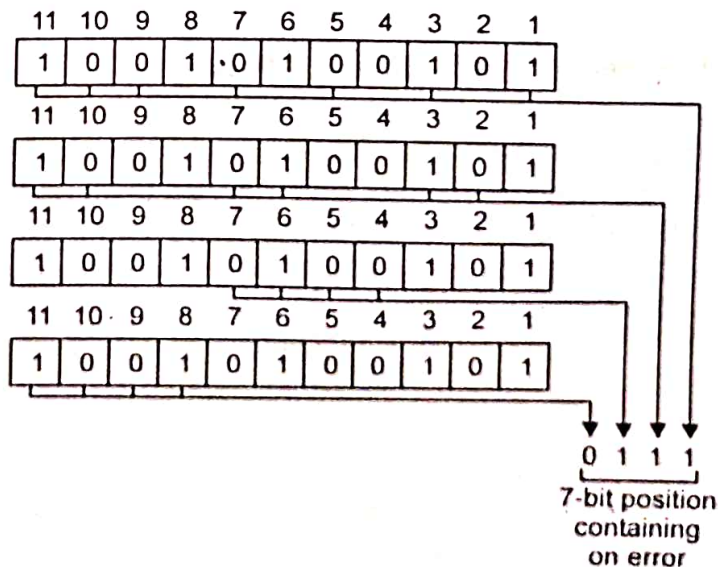
$$2^4 \geq 7 + 4 + 1$$

Therefore, no. of redundancy bits = 4

2. Determining the positions of various bits and redundancy bits. The various r bits are placed at the position that corresponds to the power of 2 i.e. 1, 2, 4, 8.



Error Detection & Correction : Considering a case of above discussed example. If bit number 7 has been changed from 1 to 0. The data will be erroneous.



Data sent !	10011100101
Data received	10010100101

The receiver takes the transmission and recalculates four new VRCs using the same set of bits used by sender plus the relevant parity (r) bit for each set. Then it assembles the new parity values into a binary number in order of r position (r_8, r_4, r_2, r_1). In the above example, this step gives us the binary number 0111. This corresponds to decimal 7. Therefore bit number 7 contains an error. To correct this error, bit 7 is reversed from 0 to 1.

Q 44. Write any four "Zero Byte" instruction. (PTU, May 2007)

Ans. Zero Byte Instructions :

1. Accumulator
2. A stack organized computer does not use an address field for the Instruetrons ADD & MUL. The name "Zero Byte" is given to this instruction because of the absence of an address field in the computational instructions.

3. Memory-memory : All three operands of each instruction are in memory.

4. Load Store : All operations occus in registers and register to register instruction have three operands per instruction.

Q 45. What is the basic difference between a branch instruction, a call subroutine instruction and program interrupt ? (PTU, May 2007)

Ans. Branch Instruction : The branch is usually one-add instruction. It is written in assembly language as BR and ADR, where ADR is a Symbolic name for an address. When executed the branch instruction causes a transfer of the value of ADR into the program counter.

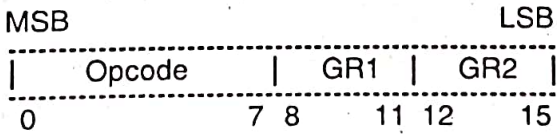
Subroutine : A subroutine is a self contained sequence of Instructions that carries out a given task. It is a set of common instructions that can be used in program many times. Each time that a subroutine is used in the main part of the program, a branch is executed to the beginning of the subroutine. After the subroutine has been executed, branch is made back to the main program.

Program Interrupt : The concept of program Interrupt is used to handle a variety of problems that arise out of normal program sequence. Program interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or Internal generated request. Control Hetarns to the original program after the service program is executed.

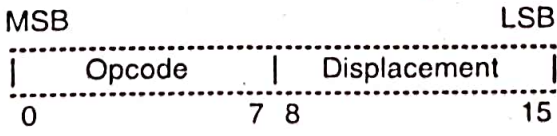
Q 46. Briefly explain an instruction format. (PTU, May 2018 ; Dec. 2014, 2008)

Ans. Instruction Formats : Six basic instruction formats shall support 16 and 32 bit instructions. The operation code (opcode) shall normally consist of the 8 most significant bits of the instruction.

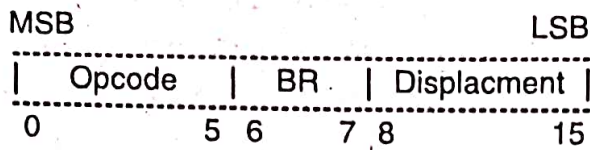
1. Register-to-Register Format : The register-to-register format is a 16-bit instruction consisting of an 8-bit opcode and two 4-bit general register (GR) fields that typically specify any of 16 general registers. In addition, these fields may contain a shift count, condition code, opcode extension, bit number, or the operand for immediate short instructions.



2. Instruction Counter Relative Format : The Instruction Counter (IC) Relative Format is a 16-bit instruction consisting of an 8-bit opcode and an 8-bit displacement field.



3. Base Relative Format : The base relative instruction format is a 16-bit instruction consisting of a 16-bit opcode, a 2-bit base register field and an 8-bit displacement field. The base register (BR) field allows the designation of one of four different registers.



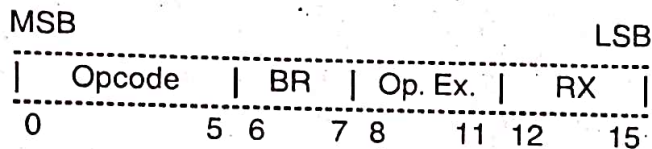
BR = 0 implies general register 12

BR = 1 implies general register 13

BR = 2 implies general register 14

BR = 3 implies general register 15

4. Base Relative Indexed Format : The base relative indexed instruction format is a 16-bit instruction consisting of a 16-bit opcode, a 2-bit base register field, a 4-bit opcode extension and a 4-bit index register field. The base register (BR) field allows the designation of one of four different base registers and the index register (RX) field allows the designation of one of fifteen different index registers.



BR = 0 implies general register 12

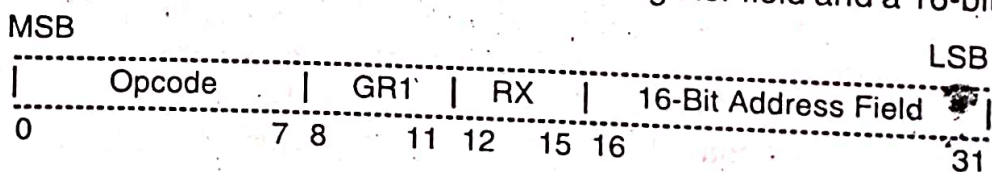
BR = 1 implies general register 13

BR = 2 implies general register 14

BR = 3 implies general register 15

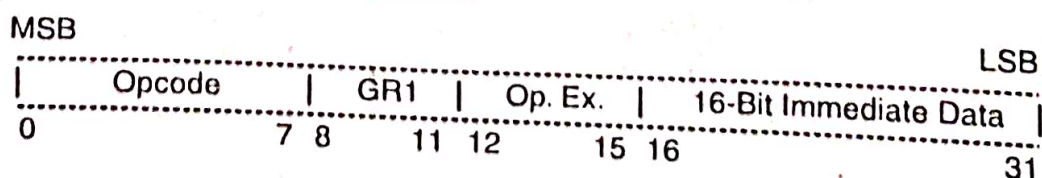
RX = 0 implies no indexing

5. Long Instruction Format : The Long Instruction Format is a 32-bit instruction consisting of an 8-bit opcode, a 4-bit general register field, a 4-bit index register field and a 16-bit address field.

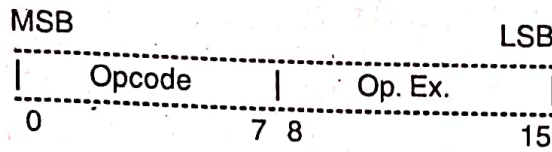


Typically, GR1 is one of the 16 general registers on which the instruction is performing the operation. RX is one of the 15 general registers being used as an index register. The 16-bit address field is either a full 16-bit memory address or a 16-bit operand if the instruction specifies immediate addressing.

6. Immediate Opcode Extension Format : The immediate opcode extension format is a 32-bit instruction consisting of an 8-bit opcode, a 4-bit general register field, a 4-bit opcode extension and 16-bit data field. Typically, GR1 is one of the 16 general registers on which the instruction is performing the operation. Op. Ex. is an opcode extension.

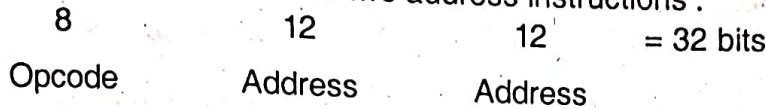


7. Special Format : The special instruction format is a 16-bit instruction consisting of an 8-bit opcode followed by an 8-bit opcode extension (Op. Ex.).



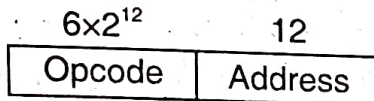
Q 47. A computer has 32-bit instruction and 12-bit address. If there are 250 two address instructions, how many one address instructions can be formulated ? (PTU, May 2007)

Ans. An instruction format for two address instructions :



There are 8 bits used by opcode so $2^8 = 256$ instructions can be possible. But in our problem, there are 250 two address instructions. Thus remaining $256 - 250 = 6$ combinations can be used for one address instructions.

An Instruction for one address instructions



Maximum number of one address instructions.
 $= 6 \times 2^{12} = 24, 576.$

Q 48. Write instructions (8085) to : Load 00H in the accumulator; Decrement the accumulator; Display the answer. (PTU, Dec. 2007)

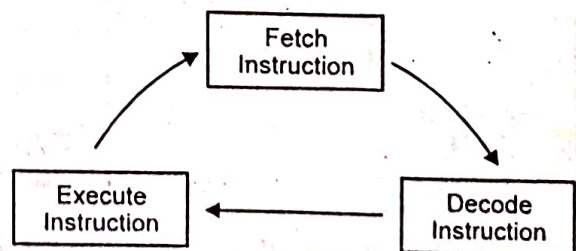
Ans.

Memory Address	Machine Code	Mnemone	Operands	Common
FC00	3A, 00, FC	LDA	FC00	Get the content of memory location FC00 into accumulator
FC03	46	MOV	13, 4	Move the content IC00 to 13
FC04	2B	DCX	H	decrement H-L pair by one.
FC05	4E	MOV	G, M	Move the content of FC00 to C
FC06	76	HLT		Halt

Q 49. What do you mean by instruction cycle, fetch cycle, machine cycle and interrupt acknowledgement cycle? (PTU, Dec. 2019, 2017, 2014 ; May 2015, 2008)

Ans. (a) Instruction cycle : An instruction cycle (also called **fetch-and-execute cycle**, and **FDX**) is the time period during which a computer processes a machine language instruction from its memory or the sequence of actions that the central processing unit (CPU) performs to execute each machine code instruction in a program.

The name fetch-and-execute cycle is commonly used. The instruction must be fetched from main memory, and then executed by the CPU. This is fundamentally how a computer operates, with its CPU reading and executing a series of instructions written in its machine language. From this arise all functions of a computer familiar from the user's end.



Each computer's CPU can have different cycles based on different instruction sets.

1. Fetch the instruction from main memory : The CPU presents the value of the program counter (PC) on the address bus. The CPU then fetches the instruction from main memory via the data bus into the memory data register (MDR). The value from the MDR is then placed into the current instruction register (CIR), a circuit that holds the instruction temporarily so that it can be decoded and executed.

2. Decode the instruction : The instruction decoder interprets and implements the instruction. The instruction register (IR) holds the current instruction, while the program counter (PC) holds the address in memory of the next instruction to be executed.

Fetch data from main memory : Read the effective address from main memory if the instruction has an indirect address. Fetch required data from main memory to be processed and placed into registers.

3. Execute the instruction : From the instruction register, the data forming the instruction is decoded by the control unit. It then passes the decoded information as a sequence of control signal to the relevant function units of the CPU to perform the actions required by the instruction such as reading values from registers, passing them to the Arithmetic logic unit (ALU) to add them together and writing the result back to a register. A condition signal is sent back to the control unit by the ALU if it is involved.

4. Store results : Also called write back to memory. The result generated by the operation is stored in the main memory, or sent to an output device. Based on the condition feedback from the ALU, the PC is either incremented to address the next instruction or updated to a different address where the next instruction will be fetched. The cycle is then repeated.

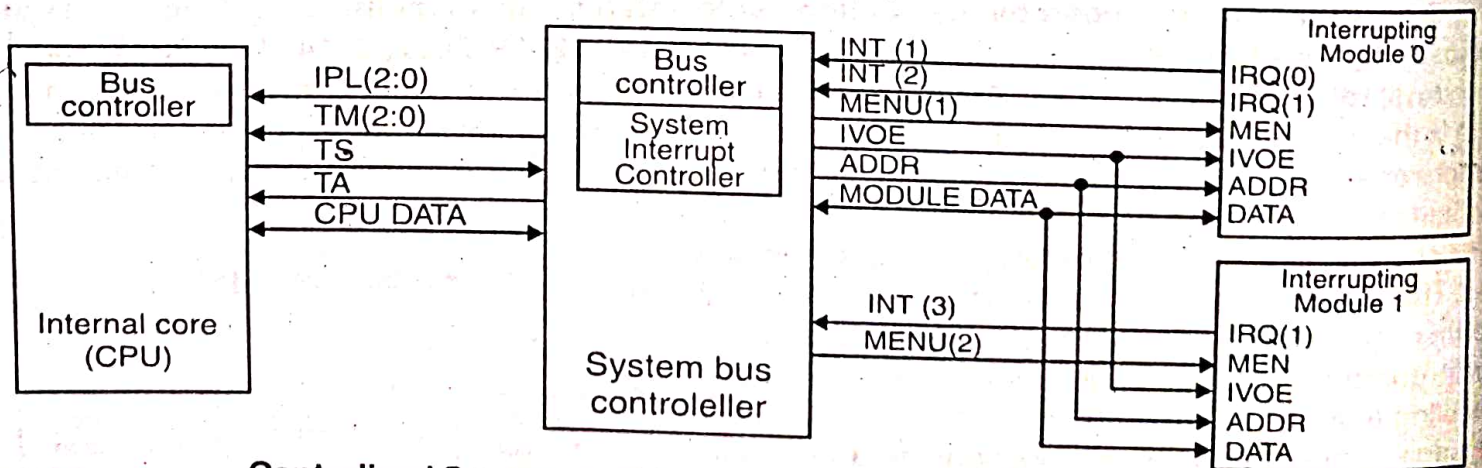
(b) Fetch cycle : Steps 1 and 2 of the Instruction Cycle are called the Fetch Cycle. These steps are the same for each instruction. The fetch cycle processes the instruction from the instruction word which contains an opcode and an operand.

(c) Machine cycle : The steps performed by the computer processor for each machine language instruction received. The **machine cycle** is a 4 process cycle that includes reading and interpreting the machine language, executing the code and then storing that code.

(d) Interrupt acknowledge cycle protocol for integrated modules of embedded microprocessor systems

Within integrated microprocessor based devices, it is desirable to provide an internal bus protocol that allows individual functional blocks to be developed, verified and integrated as independent modules. The internal bus protocol employed should allow each module to have :

Interrupt acknowledge cycle protocol solves the above problems for an integrated device with centralized interrupt control.



Centralized System Architecture for Internal Interrupt Control

The system architecture employing centralized interrupt control for integrated embedded micro-processor systems is shown in Fig. With this architecture, interrupt control and arbitration is handled by the system bus controller (SBC). An interrupting module may provide an individual interrupt request line (INT), to the SBC for each of its interrupt sources. The priority and auto vector features of each interrupt are programmable via registers in the system bus controller.

Q 50. Design a CPU that meets the following specifications :

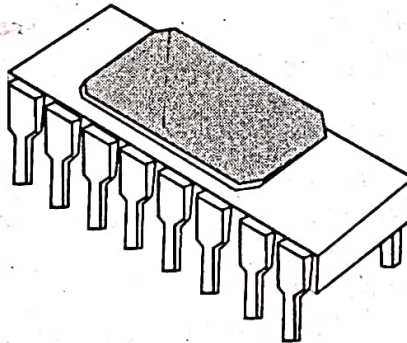
It can access 64 words of memory, each word being 8-bit long. The CPU does this by outputting a 6-bit address on its output pins A [5, 0] and reading in the 8-bit value from memory on inputs D [7,0]. It has one 8-bit accumulator, 8-bit address register, 6-bit program counter, 2-bit instruction register, 8-bit data register.

The CPU must realise the following instruction set :

Instruction	Instruction Code	Operation
ADD	00 AAAAAA	$AC \leftarrow AC + M [AAAAAA]$
AND	01 AAAAAA	$AC \leftarrow AC \wedge M [AAAAAA]$
JMP	10 AAAAAA	Go to AAAAAA
INC	11 xxxxxx	$AC \leftarrow AC + 1$

(PTU, May 2008)

Ans. The **processor (CPU**, for Central Processing Unit) is the computer's brain. It allows the processing of numeric data, meaning information entered in binary form, and the execution of instructions stored in memory.



4 16-bit General purpose registers.

Also assigned specific functions or paired with certain actions.

A – accumulator, B, C - counter, D

Also addressable at the byte level AH, AL, etc.

4 16-bit dedicated purpose address registers.

SP - stack pointer (user programmable).

BP - base pointer (used to with blocks of data).

SI - source index (used by specific commands with DI to move blocks of data)

DI - destination index

20-bit (multiplexed) memory address register.

16-bit instruction pointer (combined with CS register *4)

4 16 bit segment registers.

Used to address memory in 64 KB blocks on 6 byte boundaries.

Combined with GP registers for a complete address.

CS - code segment

DS - data segment

SS - stack segment

ES - extra segment (usually used with DS)

16-bit Condition register

Status flags

Carry flag - carry or borrow for unsigned number.

Auxiliary carry flag - monitors lowest 4 bits of accumulator, used for bcd carry.

Overflow flag - carry or borrow for a signed number.

Sign flag - result of an action created a negative value.

Zero flag - result of an action was zero or equal.

Parity flag - action detected an even or odd parity.

Control flags

Direction flags - determines index direction of a looping instruction, up or down.

Interrupt enable flag - block or accept maskable interrupts.

Trap flag - causes cpu to pause between commands (debugging)

It can access 64 words of memory, each word being 8-bit long. The CPU does this by outputting a 6-bit address on its output pins A [5, 0] and reading in the 8-bit value from memory on inputs D [7,0]. It has one 8-bit accumulator, 8-bit address register, 6-bit program counter, 2-bit instruction register, 8-bit data register.

The CPU must realise the following instruction set :

Instruction	Instruction Code	Operation
ADD	00 AAAAAA	$AC \leftarrow AC + M [AAAAAA]$
AND	01 AAAAAA	$AC \leftarrow AC \wedge M [AAAAAA]$
JMP	10 AAAAAA	Go to AAAAAA
INC	11 xxxxxx	$AC \leftarrow AC + 1$

Q 51. Explain in detail the difference between RISC and CISC Architecture ?

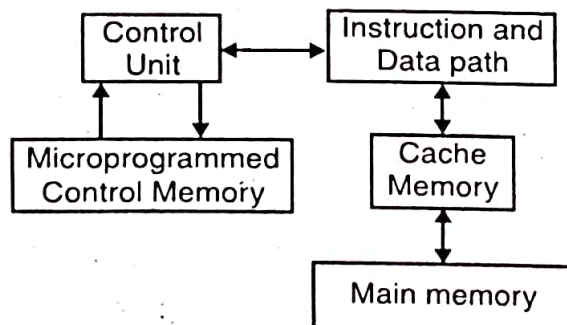
(PTU, May 2017, 2010 ; Dec. 2019, 2016, 2014, 2004)

OR

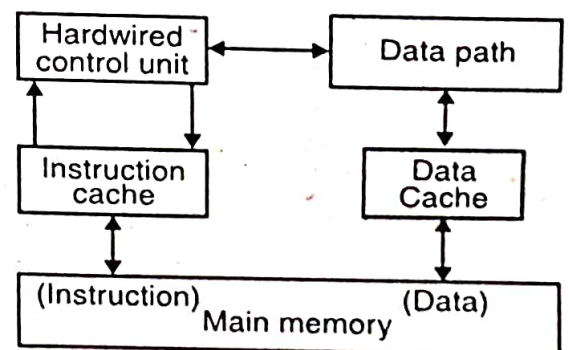
Compare the instruction set Architecture in RISC and CISC processors in terms of instruction formats, addressing modes and cycles per instruction (CPI)?

(PTU, May 2018, 2006)

Ans. RISC Vs CISC : Fig. shows the architectural distinctions between modern CISC and traditional RISC. Conventional CISC architecture uses a unified cache for holding both instructions and data. Therefore, they must share the same data/instruction path. In a RISC processor, separate, **instruction** and **data caches** are used with different access paths. However, exceptions do exist. In other words, CISC processors may also use split codes. The use of microprogrammed control can be found in traditional CISC, and hard wired control in most RISC. Thus control memory (ROM) is needed in earlier CISC processors, which may significantly slow down the instruction execution. However, modern CISC may also use hard wired control. Therefore, split caches and hardwired control are not exclusive in RISC machines.



The CISC architecture with microprogrammed control and unified cache



The RISC architecture with hardwired control and split instruction and data cache

Using hardwired control will reduce the CPI effectively to one instruction per cycle if pipelining is carried out perfectly. Some CISC processors also use split caches and hard wired control, such as the MC68040 and 1586.

We have compared the main features of RISC and CISC processors. The comparison involves five areas : **instruction sets, addressing modes, register file and cache design, clock rate and expected CPI, and control mechanisms.**

The large number of instructions used in a CISC processor is the result of using variable- format instructions- integer, floating-point, and vector data- and of using over a dozen different addressing modes. Furthermore, with few GPRs (Genetal Purpose Registers) many more instructions access the memory for operands. The CPI is thus high as a result of the long microcodes used to control the execution of some complex instructions.

On the other hand, most RISC processors use 32-bit instructions which are predominantly register-based. With few simple addressing modes, the memory-access cycle is broken into pipelined access operations involving the use of caches and working registers. Using a large register file and separate 1- and D-caches benefits internal data forwarding and eliminates unnecessary storage of intermediate results. With hardwired control, the CPI is reduced to 1 for most RISC instructions.

Architectural Characteristic	Complex Instruction Set Computer (CISC)	Reduced Instruction Set Computer (RISC)
1. Instruction-set size and instruction formats	Large set of instructions with variable formats (16-64 bits per instruction).	Small set of instructions with fixed (32-bit) format and most register-based instructions.
2. Addressing modes	12 – 24	Limited to 3 – 5.
3. General purpose registers and cache design	8-24 GPRs, mostly with a unified cache for instructions and data, recent designs also use split caches.	Large numbers (32 – 192) of GPRs with mostly split data cache and instruction cache.
4. Clock rate and CPI	33-50 MHz in 1992 with a CPI between 2 and 15.	50-150 MHz in 1993 with one CPI cycle for almost all instructions and an average CPI < 1.5.
5. CPU Control	Most microcoded using control memory (ROM), but modern CISC also uses hard wired control.	Most hardwired without control memory.
6. Instructions Type	Not register based instructions	Register based instructions
7. Memory Access	More memory access	Less memory access
8. Emphasis	Emphasis on hardware	Emphasis on software
9. Operating clocks	Includes multi-clock	Single-clock
10. Instructions nature	Complex instructions	Reduced instruction only
11. Data Transfer	Memory-to-memory	Register to register
12. LOAD and STORE	"LOAD" and "STORE" incorporated in instructions	"LOAD" and "STORE" are independent instructions
13. Opcode Size	Small code sizes,	Large code sizes

Table : Characteristics of CISC and RISC Architectures

Q 52. What are the advantage of RISC over CISC.

(PTU, Dec. 2019)

Ans. Advantages of RISC over CISC

One of the major advantages of RISC is its ability to execute instructions at the rate of one per clock cycle. It is not possible to expect that every instruction be fetched from memory and executed in one clock cycle. What is done, in effect, is to start each instruction with each clock cycle and to pipeline the processor to achieve the goal of single-cycle instruction execution. The advantage of RISC over CISC is that RISC can achieve pipeline segments, requiring just one clock cycle, while CISC uses many segments in its pipeline, with the longest segment requiring two or more clock cycles. Another characteristic of RISC is the support given by the compiler that translates the high-level language program into machine language program. Instead of designing hardware to handle the difficulties associated with data conflicts and branch penalties, RISC processors rely on the efficiency of the compiler to detect and minimize the delays encountered with these problems.

Q 53. What are characteristics of RISC?

(PTU, Dec. 2019)

Ans. RISC processors has following characteristics :

1. Relatively few instructions.
2. Relatively few addressing modes.
3. Memory access limited to load and store instructions.
4. All operations done with the Registers of the CPU.
5. Fixed length easily decoded instruction format.
6. Single cycle instruction execution
7. Hardwired rather than micro programmed control.

Q 54. A given processor has 32 registers, uses 16-bit immediate, and has 142 instruction in its ISA. In a given program, 20% of the instructions take one input register and have one output register, 30% have two input registers and one output register, 25% have one output and one input register and take an immediate input as well, and the remaining 25% have one immediate input register and one output register. For each of the four types of instruction, how many bits are required ? Assume that the ISA requires that all instructions be a multiple of 8 bits in length.

(PTU, Dec. 2005)

Ans. A 32-KB cache with 256 byte line contact lines. Since cache is four way set associative, it has 32 sets ($2^5 = 32$) and require 5 bits lines that are 256 bytes lay means it require 8 bits ($2^8 = 256$) so 13 bits of address are used to select a set and determine the byte within line that an address points to. Therefore a tag field of each tag array entry is $32 - 12 = 19$ bits long. Adding 2 bits for duty and valid bits we get 21 bits per tag array. Multiplying by 128 lines is cache gives 2688 bits, storage in the tag array.

Q 55. Explain instruction set of SPARC with Descriptions.

(PTU, Dec. 2007)

Ans. The basic set of SPARC instruction set are as follows :

Sparc Instruction Set

□ Instruction groups are consists of the follows :

- load/store (ld, st,)
- integer arithmetic (add, sub,)
- bit-wise logical (and, or, xor,)
- bit-wise shift (sll, srl,)
- integer branch (be, bne, bl, bg,)
- Trap (ta, te,)
- control transfer (call, save,)
- floating point (ldf, stf, fadds, fsubs,)
- floating point branch (fbe, fbne, fbl, fbg,)

1. Load Instructions

(i) Move data from memory to a register

$$\bullet \text{ ld } \begin{bmatrix} \text{u} \\ \text{s} \end{bmatrix} \begin{bmatrix} \text{b} \\ \text{d} \end{bmatrix} \{a\} [\text{address}], \text{reg}$$

□ Examples :

- `ld [% i1], %g2`
- `ldud [%i1 + %i2], %g3`

(ii) Move data from memory to a register

$$\bullet \text{ ld } \begin{bmatrix} \text{u} \\ \text{s} \end{bmatrix} \begin{bmatrix} \text{b} \\ \text{d} \end{bmatrix} \{a\} [\text{address}], \text{reg}$$

□ Details

- fetched byte/halfword is right-justified
- leftmost bits are zero-filled or sign-extended
- double-word loaded into register pair ; most significant word in reg (must be even) ; least significant in reg+1
- address must be appropriately aligned

2. Store Instructions

(i) Move data from a register to memory

$$\bullet \text{ st } \begin{bmatrix} \text{b} \\ \text{h} \\ \text{d} \end{bmatrix} \{a\} \text{reg}, [\text{address}]$$

□ Examples

- `st %g1, [% o2]`
- `stb %g1, [%o2 + o3]`

(ii) Move data from a register to memory

$$\bullet \text{ st } \begin{bmatrix} \text{b} \\ \text{h} \\ \text{d} \end{bmatrix} \{a\}, \text{reg}, [\text{address}]$$

□ Details

- rightmost bit of byte/halfword are stored
- leftmost bits of byte/halfword are ignored
- reg must be even when storing double words

3. Arithmetic Instructions

Arithmetic operations on data in registers

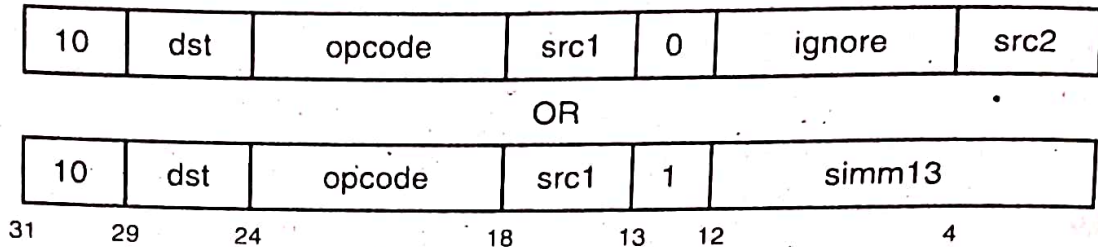
- `add {x} {cc} src1, src2, dst dst = src1 + src2`
- `sub {x} {cc} src1, src2, dst dst = src1 - src2`

□ **Examples :**

- add %1, %02, %03
- sub %01, 2, %g03

□ **Details**

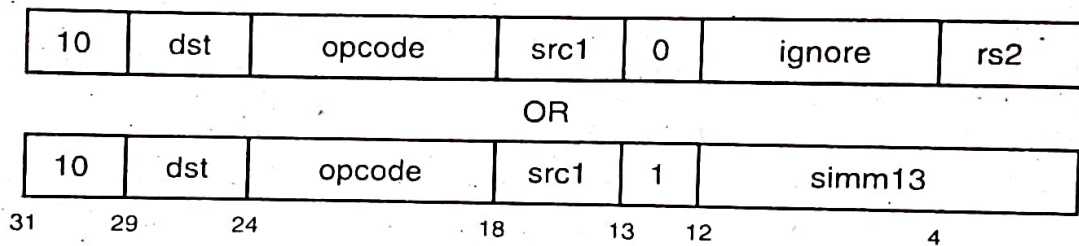
- tsd1 and et umust be registers
- tsd3 may be a register or asigned 13-bit immediate



4. Bitwise Logical Instructions

Logical operations on data in registers

- | | | |
|-------------|-----------------|-------------------------------|
| ● and {cc} | src1, src2, dst | $dst = src1 \& src2$ |
| ● andn {cc} | src1, src2, dst | $dst = src1 \& \sim src2$ |
| ● or {cc} | src1, src2, dst | $dst = src1 src2$ |
| ● orn {cc} | src1, src2, dst | $dst = src1 \sim src2$ |
| ● xor {cc} | src1, src2, dst | $dst = src1 \wedge src2$ |
| ● xnor {cc} | src1, src2, dst | $dst = src1 \wedge \sim src2$ |



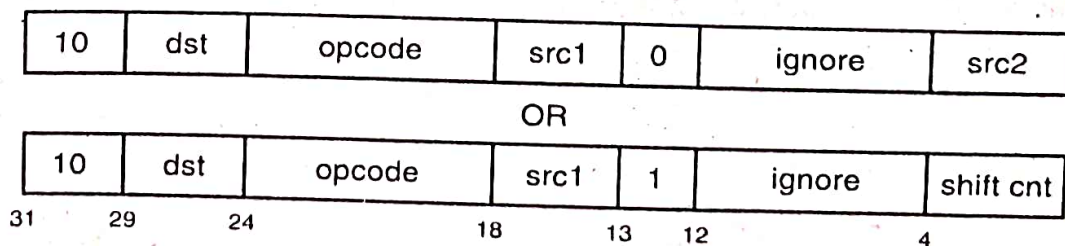
5. Shift Instructions

Shift bits of data in registers

- $s \begin{bmatrix} | \\ | \\ r \end{bmatrix} \begin{bmatrix} | \\ | \\ a \end{bmatrix} src1, \begin{bmatrix} src2 \\ 0..31 \end{bmatrix}, dst$
- sll : $dst = src1 \ll src2 ;$
- srl : $dst = src1 \gg src2 ;$

Details

- do not modify condition codes
- sll and srl fill with 0, sra fills with sign bit
- no sla



6. Floating Point Instructions

Performed by floating point unit (FPU)
 Use 32 floating point registers : %f OA% 31
 Load and store instructions

- ~ N ld [address], freg
- ~ N ldd [address], freg
- ~ N st freg, [address]
- ~ N std freg, [address]

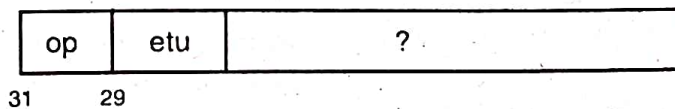
Other instructions are FPU-specific

- ~ N fmovs, fsqrt, fadd, fsub, fmul, fdiv, Ä

7. Data Movement

We load a constant (e.g., address) into a register as follow :

Instruction format



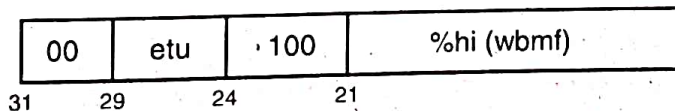
Loading a constant (e.g., address) into a register

- set hi %hi (value), dst
- or dst, %o (value), dst

Details

- if hi (value) = = 0, omit set hi
- if % o (value) = = 0, omit or

sethi instruction format



Example : direct addressing

- | | |
|---------------|----------------------|
| set a, %g1 | set hi % hi (a), %g1 |
| ld [%g1], %g2 | or %o (a), %g1 |
| | ld [%g1], %g2 |

Faster alternative

- set hi %hi (a), %g1
- ld [%g1 + % 0 (a)], %g2

Q 56. Explain the various Addressing modes in detail.

(PTU, May 2018, 2016, 2015, 2014 ; Dec. 2016, 2015, 2014, 2008)

Ans. To understand the various addressing modes to be presented in this section, it is imperative that we understand the basic operation cycle of the computer. The control unit of a computer is designed to go through an instruction cycle that is divided into three major phases :

1. Fetch the instruction from memory.
2. Decode the instruction.
3. Execute the instruction.

There is one register in the computer called the program counter or PC that keeps track of the

instructions in the program stored in memory. PC holds the address of the instruction to be executed next and is incremented each time an instruction is fetched from memory. The decoding done in step 2 determines the operation to be performed, the addressing mode of the instruction, and the location of the operands. The computer then executes the instruction and returns to step 1 to fetch the next instruction in sequence.

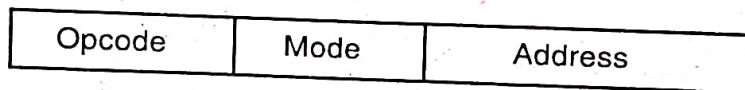
In some computers the addressing mode of the instruction is specified with a distinct binary code, just like the operation code is specified. Other computers use a single binary code that designates both the operation and the mode of the instruction. Instructions may be defined with a variety of addressing modes are combined in one instruction.

An example of an instruction format with a distinct addressing mode field is shown in fig. The operation code specifies the operation to be performed. The mode field is used to locate the operands needed for the operation. There may or may not be an address field in the instruction. If there is an address field, it may designate a memory address or a processor register. Moreover, as discussed in the preceding section, the instruction may have more than one address field, and each address field may be associated with its own particular addressing mode.

Although most addressing modes modify the address field of the instruction, there are two modes that need no address field at all. These are the implied and immediate modes.

Following are the various Addressing modes :

1. Implied Mode : In this mode the operands are specified implicitly in the definition of the instruction. For example, the instruction "complement accumulator" is an implied-mode instruction because the operand in the accumulator register is implied in the definition of the instruction. In fact, all register reference instructions that use an accumulator are implied-mode instructions. Zero-address instructions in a stack-organized computer are implied-mode instructions since the operands are implied to be on top of the stack.



Instruction format with mode field

2. Immediate Mode : In this mode the operand is specified in the instruction itself. In other words, an immediate-mode instruction has an operand field rather than an address field. The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction. Immediate-mode instructions are useful for initializing registers to a constant value.

It was mentioned previously that the address field of an instruction may specify either a memory word or a processor register. When the address field specifies a processor register, the instruction is said to be in the register mode.

3. Register Mode : In this mode the operands are in registers that reside within the CPU. The particular register is selected from a register field in the instruction. A k-bit field can specify any one of 2^k registers.

4. Register Indirect Mode : In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory. In other words, the selected register contains the address of the operand rather than the operand itself. Before using a register indirect mode instruction, the programmer must ensure that the memory address of the operand is placed in the processor register with a previous instruction. A reference to the register is then equivalent to specifying a memory address. The advantage of a register indirect mode instruction is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.

5. Autoincrement or Autodecrement Mode : This is similar to the register indirect mode except that the register is incremented or decremented after (or before) its value is used to access memory. When the address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register after every access to the table. This can be achieved by using the increment or decrement instruction. However, because it is such a common requirement, some computers incorporate a special mode that automatically increments or decrements the content of the register after data access.

The address field of an instruction is used by the control unit in the CPU to obtain the operand from memory. Sometimes the value given in the address field is the address of the operand, but sometimes it is just an address from which the address of the operand is calculated. To differentiate among the various addressing modes it is necessary to distinguish between the address part of the instruction and the effective address used by the control when executing the instruction. The effective address is defined to be the memory address obtained from the computation dictated by the given addressing mode. The effective address is the address of the operand in a computational-type instruction. It is the address where control branches in response to a branch-type instruction. We have already defined two addressing modes in. They are summarized here for reference.

6. Direct Address Mode : In this mode the effective address is equal to the address part of the instruction. The operand resides in memory and its address is given directly by the address field of the instruction. In a branch-type instruction the address field specifies the actual branch address.

7. Indirect Address Mode : In this mode the address field of the instruction gives the address where the effective address is stored in memory. Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.

A few addressing modes require that the address field of the instruction be added to the content of a specific register in the CPU. The effective address in these modes is obtained from the following computation :

$$\text{effective address} = \text{address part of instruction} + \text{content of CPU register}$$

The CPU register used in the computation may be the program counter, an index register, or a base register. In either case we have a different addressing mode which is used for a different application.

8. Relative Address Mode : In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address. The address part of the instruction is usually a signed number (in 2's complement representation) which can be either positive or negative. When this number is added to the content of the program counter, the result produces an effective address whose positive in memory is relative to the address of the next instruction. To clarify with an example, assume that the program counter contains the number 825 and the address part of the instruction contains the number 24. The instruction at location 825 is read from memory during the fetch phase and the program counter is then incremented by one to 826. The effective address computation for the relative address mode is $826 + 24 = 850$. This is 24 memory locations forward from the address of the next instruction. Relative addressing is often used with branch-type instructions when the branch address is in the area surrounding the instruction word itself. It results in a shorter address field in the instruction format since the relative address can be specified with a smaller number of bits compared to the number of bits required to designate the entire memory address.

9. Indexed Addressing Mode : In this mode the content of an index register is added to the address part of the instruction to obtain the effective address. The index register is a special CPU register that contains an index value. The address field of the instruction defines the beginning address of a data array in memory. Each operand in the array is stored in memory relative to the beginning address. The distance between the beginning address and the address of the operand is the index

value stored in the index register. Any operand in the array can be accessed with the same instruction provided that the index register contains the correct index value. The index register can be incremented to facilitate access to consecutive operands. Note that if an index-type instruction does not include an address field in its format, the instruction converts to the register indirect mode of operation.

Some computers dedicate one CPU register to function solely as an index register. This register is involved implicitly when the index-mode instruction is used. In computers with many processor registers, any one of the CPU registers can contain the index number. In such a case the register must be specified explicitly in a register field within the instruction format.

10. Base Register Addressing Mode : In this mode the content of a base register is added to the address part of the instruction to obtain the effective address. This is similar to the indexed addressing mode except that the register is now called a base register instead of an index register. The difference between the two modes is in the way they are used rather than in the way that they are computed. An index register is assumed to hold an index number that is relative to the address part of the instruction. A base register is assumed to hold a base address and the address field of the instruction gives a displacement relative to this base address. The base register addressing mode is used in computers to facilitate the relocation of programs in memory. When programs and data are moved from one segment of memory to another, as required in multiprogramming systems, the address values of instructions must reflect this change of position. With a base register, the displacement values of instructions do not have to change. Only the value of the base register requires updating to reflect the beginning of a new memory segment.

Q 57. Write a note on general register organization.

(PTU, Dec. 2016)

Ans. The number of registers in a processor unit may vary from just one processor register to as many as 64 registers or more.

1. One of the CPU register is called as an accumulator AC or 'A' register. It is the main operand register of the ALU.
2. The Data Register (DR) acts as a buffer between the CPU and main memory. It is used as an input operand register with the accumulator.
3. The instruction register (IR) holds the opcode of the current instructions.
4. The address register (AR) holds the address of the memory in which the operand resides.
5. The program counter (PC) holds the address of the next instruction to be fetched for execution.

Additional addressable registers can be provided for storing operand and address. This can be viewed as replacing the single accumulator by a set of registers.

Q 58. What is stack organization?

(PTU, May 2016, 2014)

Ans. A stack is a storage structure that stores information in such a way that the last item stored is the first item retrieved. It is based on the principle of LIFO. The stack is a digital computer in a group of memory locations with a register that holds the address of top of element. This register that holds the address of top of element of the stack is called stack pointer.

The two operations of a stack are :

- (i) Push
- (ii) Pop

Push inserts an item on top of the stack where as pop deletes an item from top of stack.

Q 59. How stack is implemented?

(PTU, May 2016, 2014)

Ans. In digital computers, stack can be implemented in two ways :

1. Register Stack
2. Memory Stack

1. Register Stack : A stack can be organized as a collection of finite number of registers that are used to store temporary information during the execution of a program. The stack pointer (SP) is a register that holds the address of top of element of the stack. Register or memory words can be organized to form a stack. The stack pointer is a register that holds the memory address of the top of the stack. When an item need to be deleted from the stack, item on the top of the stack is deleted and the stack pointer is decremented. Similarly, when an item needs to be added, the stack pointer is incremented and writing the word at the position indicated by the stack pointer. There are two 1 bit register ; FULL and EMTY that are used for describing the stack overflow and underflow conditions. Following micro-operations are performed during inserting and deleting an item in/from the stack.

Insert :

SP < SP + 1 // Increment the stack pointer to point the next higher address //
 M [SP] < DR // Write the item on the top of the stack //
 If (SP = 0) then (Full < 1) // Check overflow condition //
 EMTY < 0 // Mark that the stack is not empty //

Delete :

DR < M [SP] // Read an item from the top of the stack //
 SP < SP - 1 // Decrement the stack pointer //
 If (SP = 0) then (EMTY < 1) // Check underflow condition //
 FULL < 0 // Mark that the stack is not full //

2. Memory Stack : A stack can be implemented in a random access memory (RAM) attached to a CPU. The implementation of a stack in the CPU is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer. The starting memory location of the stack is specified by the processor register as stack pointer.

Stack is the storage method of the items in which the last item included is the first one to be removed/taken from the stack. Generally a stack in the computer is a memory unit with an address register and the register holding the address of the stack is known as the Stack Pointer (SP). A stack performs Insertion and Deletion operation, were the operation of inserting an item is known as Push and operation of deleting an item is known as Pop. Both push and pop operation results in incrementing and decrementing the stack pointer respectively.

Q 60. Write a note on Data Manipulation Instruction.

Ans. Data manipulation instructions performs operations on data and provide the computational capabilities for the computer. The data manipulation instructions in a typical computer are usually divided into three basic types :

1. Arithmetic instructions
2. Logical and bit manipulation instructions
3. Shift instructions.

Q 61. What are the different types of computer instructions?

Ans. Most of the computer instructions are classified into three categories :

1. Data transfer instructions
2. Data manipulation instructions
3. Program control instructions.

Data transfer instructions cause transfer of data from one location to another without changing the binary information content.

Data manipulation instructions are those that perform arithmetic, logic and shift operations.

Program control instruction provides decision making capabilities and change the path taken by the program when executed in the computer.

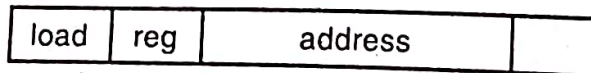
Q 62. Define the following address mode of the instruction set

(a) Direct (b) Indirect (c) Relative (d) Indexed.

Ans. Addressing modes are an aspect of the instruction set architecture in most central processing unit (CPU) designs. The various addressing modes that are defined in a given instruction set architecture define how machine language instructions in that architecture identify the operand (or operands) of each instruction. An addressing mode specifies how to calculate the effective memory address of an operand by using information held in registers and/or constants contained within a machine instruction or elsewhere.

In computer programming, addressing modes are primarily of interest to compiler writers and to those who write code directly in assembly language.

Absolute/Direct

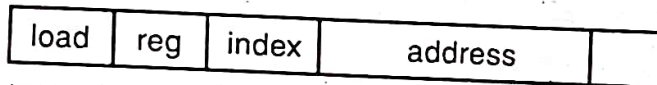


(Effective address = address as given in instruction)

This requires space in an instruction for quite a large address. It is often available on CISC machines which have variable-length instructions, such as x86.

Some RISC machines have a special Load Upper Literal instruction which places a 16-bit constant in the top half of a register. An OR literal instruction can be used to insert a 16-bit constant in the lower half of that register, so that a full 32-bit address can then be used via the register-indirect addressing mode, which itself is provided as "base-plus-offset" with an offset of 0.

Indexed absolute

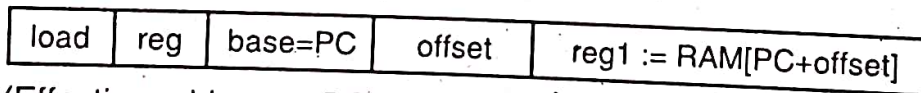


(Effective address = address + contents of specified index register)

This also requires space in an instruction for quite a large address. The address could be the start of an array or vector, and the index could select the particular array element required. The processor may scale the index register to allow for the size of each array element.

Note that this is more or less the same as base-plus-offset addressing mode, except that the offset in this case is large enough to address any memory location.

PC-relative



(Effective address = PC + offset)

The PC-relative addressing mode is used to load a register from a "constant" stored in program memory a short distance away from the current instruction. It can be seen as a special case of the "base plus offset" addressing mode, one that selects the program counter (PC) as the "base register". There are a few CPUs that support PC-relative data loads.

Q 63. Explain the meaning of the memory-reference instruction STA. (PTU, Dec. 2005)

Ans. STA : Store AC.

This Instruction stores the content of AC into the memory word specified by the effective address since the output of AC is applied to the bus and the data input of memory is connected.

Q 64. What is the role of shift register in digital Computer? (PTU, Dec. 2008)

Ans. Shift register is a group of flip flop in digital circuits, a **flip-flop** is a term referring to an

electronic circuit that has two stable states and thereby is capable of serving as one bit of computer storage.

Shift Register set up in a linear fashion which have their inputs and outputs connected together in such a way that the data is shifted down the line when the circuit is activated. Shift registers can have co parallel port.

A **parallel port** is a type of interface found on computers for connecting various peripherals. It is also known as a **printer port**. The interface, inputs and outputs, including **serial-in, parallel-out** (SIPO) and **parallel-in, serial-out** (PISO) types. There are also types that have both serial and parallel input and types with serial and parallel output. There are also **bi-directional** shift registers which allow you to vary the direction of the shift register. The serial input and outputs of a register can also be connected together to create a **circular shift register**. One could also create multi-dimensional shift registers, which can perform more complex computation.

Q 65. Give the case study on 8085 microprocessor.

Ans. 8085 microprocessor is a single board microprocessor training/development kit configured around the most widely used microprocessor of today's world 8085 provides 8K/32K bytes of RAM and 8K bytes of EPROM. The total on board memory can be very easily expanded to 64 K bytes in an appropriate combination of RAM and ROM. 8085 is configured around the internationally adopted STD bus, which is the most popular bus for process control and real time applications. All the address data and control lines are available at the edge connector. The bits fully expandable for any kind of application.

System Specification of 8085 kit :

CPU – 8 bit microprocessor, the 8085

MEMORY – Total on board capacity of 64K bytes

RAM – 8K/32K bytes and space for further expansion

ROM – 8k bytes of EPROM loaded with powerful program

TIMER – 16 bit programmable timer/counter using 8253

KEYBOARD – 10 keys for command.

16 Keys for hexadecimal data entry

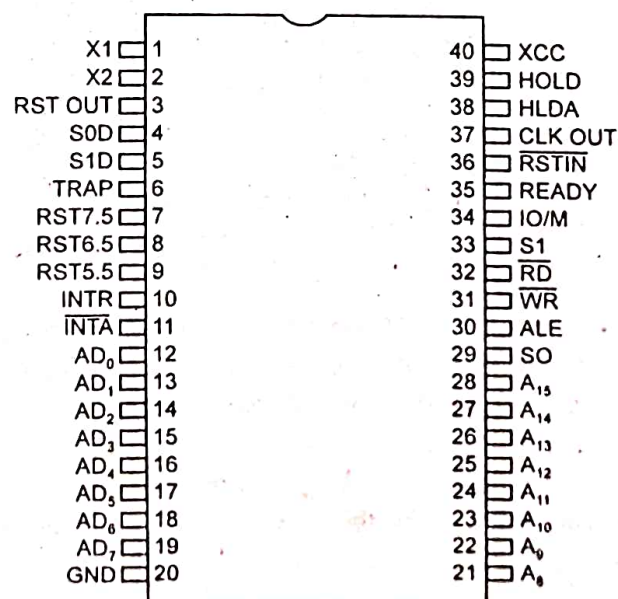
1 Key for vector interrupt & 1 key for reset

LED DISPLAY – 6 seven segment display

4 for address field & 2 for data field

BUS– All data, address and control signals.

General



The system has got 8085 as the central processing unit. The clock frequency for the system is 3.07 MHz. 8085 has got 8 data lines and 16 address lines. The lower 8 address lines and 8 bit data lines are multiplexed. Since the lower 8 address bit appear on the bus during the first clock cycle of a machine cycle and the 8 bit data appears on the bus during the 2nd and 3rd clock cycle, it becomes necessary to latch the lower 8 address bit during the first clock cycle so that the 16 bit address remains available in subsequent cycles. This is achieved using a latch 74-LS 373.

Memory : 8085 provides 8/32k bytes of RAM using 6264/62256 chip and 8k bytes of EPROM for monitor. There is one memory space provided on 8085. This one space can be defined by address slots from 8000 DFFF depending upon the size of the memory chip to be used. Total on board memory can be extended to 64K bytes.

I/O Devices : The various I/O chips, used in 8085 are 8279, 8255 and 8253

Registers : The 8085/8080A programming model includes six registers, one accumulator and one flag register as shown in figure. In addition, it has two 16-bit registers the stack pointer and the program counter. They are described briefly as follows :

The 8085/8080A has six general purpose registers to store 8-bit data, these are identified as B, C, D, E, H and L as shown in the figure. They can be combined as register pairs BC, DE and HL to perform some 16 bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

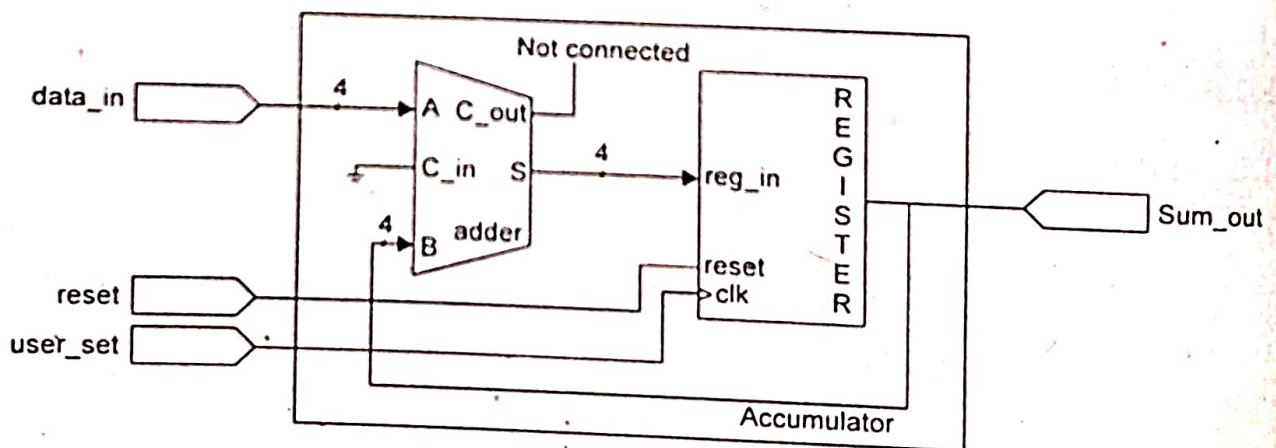
Accumulator : The accumulator is an 8 bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8 bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flags : The ALU includes five flip flops which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called zero (z), carry (cy), sign (s), Parity (P), and Auxillary carry (Ac) flags. The most commonly used flags are zero, carry and sign. The microprocessor uses these flags to test data conditions.

Q 66. Draw a schematic diagram of Accumulator logic.

Ans.

(PTU, May 2014)



Q 67. Explain serial communication.

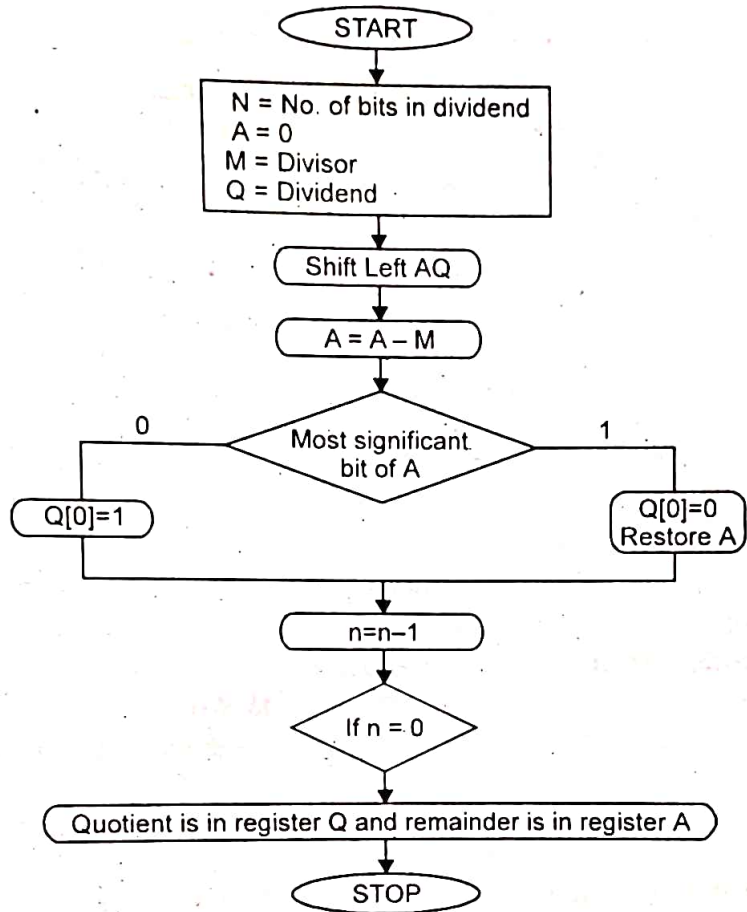
(PTU, May 2014)

Ans. Serial communication is a device communication protocol that is standard on almost every PC. The concept of serial communication is very simple. In serial communication, each bit of the message is sent in sequence, one bit at a time. Serial communication requires the use of one or two signal lines. It is slower, but less expensive, since only one conductor is required.

It is increasingly important because smaller cables are easier to connect. In this signal skew is less important because there are fewer signals. For serial communication there are at most two signals that signal skew can impact.

Q 68. Explain Division restoring algorithm for unsigned integer.

Ans. A division algorithm provides a quotient and a remainder when we divide two number. They are generally of two type slow algorithm and fast algorithm. Slow division algorithm are restoring, non-restoring, SRT algorithm and underfast comes Newton-Raphson and Goldschmidt Here register Q contains quotient and register A contain remainder. Here, n-bit dividend is loaded in Q and divisor is loaded in M. Value of Register is initially kept 0 and that is the register whose value is restored during iteration due to which it is named Restoring.



Steps :

Step 1 : First the registers are initialized with corresponding values (Q = Dividend, M = Divisor, A = 0, n = number of bits individuals)

Step 2 : Then that content of register A and Q is shifted right as if they are a single unit.

Step 3 : Then content of register M is subtracted from A and result is stored in A.

Step 4 : Then the most significant bit of the A is checked if it is 0 the least significant bit of Q is set to 1 otherwise if it is the least significant bit of Q is set to 0 and value of register A is restored i.e. the value of A before the subtraction with M.

Step 5 : The value of counter n is decremented.

Step 6 : If the value of n becomes zero we get of the loop otherwise we repeat from step 2.

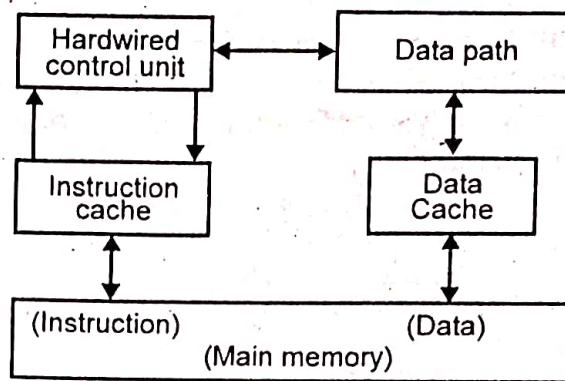
Step 7 : Finally, the register Q contains the quotient and A contain remainder.

Q 69. What are the implications of Moore's law in the development of computer technology? (PTU, May 2014)

Ans. Moore's law states that processor speeds or overall processing power for computers will double about every 18 month's. Moore's law makes it virtually certain that two or four or six or more years from now we'll be doing more things we did not expect to do with electronic devices. Thus Moore's law have the high implications in the development of computer technology.

Q 70. Explain in detail, the RISC architecture.

Ans.



RISC architecture

The concept of RISC architecture involves an attempt to reduce execution time by simplifying the instruction set of the computer. The major characteristics of a RISC processor are :

- (i) Relatively few instructions.
- (ii) Relatively few addressing modes.
- (iii) Memory access limited to load & store instructions.
- (iv) All operations done with in the register of the CPU.
- (v) Fixed-length, easily decoded instruction format.
- (vi) Single-cycle instruction execution.
- (vii) Hardwired rather than microprogrammed control.
- (viii) Faster execution.

Other characteristics attributed to RISC architecture are :

- (i) A relatively large number of registers in the processor unit.
- (ii) Use of overlapped register windows to speed-up procedure call and return.
- (iii) Efficient instruction pipeline.
- (iv) Compiler support for efficient translation of high-level language program into machine language programs.

A large number of registers is useful for storing intermediate results and for optimizing operand references. The advantage of register storage as opposed to memory storage is that registers can transfer information to other registers much faster than the transfer of information to and from memory. Thus register-to-memory operations can be minimized by keeping the most frequent accessed operands in registers. Studies that show improved performance for RISC architecture do not differentiate between the effects to the reduced instruction set and the effects of a large register file.

Q 71. Explain BCD addition and subtraction with suitable example.

(PTU, May 2014)

Ans. BCD addition : BCD addition is performed by adding two BCD numbers similar to binary addition. If the sum of two BCD numbers is not valid BCD code (1010 through 1111), then add 0110 (6) to the sum to get the valid BCD code. If carry is obtained, add the carry to the next BCD number.

For example : Add the following in BCD numbers : 437 and 721

437	→	0100	0011	0111
721	→	0111	0010	0001
		1011	0101	1000
		↑	⏟	
		Invalid BCD code	Valid BCD code (∴ 4-bit greater than 9)	

Now add 0110 (6) to the invalid result

$$\begin{array}{r}
 \text{i.e.} \quad 1011 \\
 \quad \quad 0110 \\
 \hline
 \text{Carry} \quad 10001
 \end{array}$$

Then the final result is

437	0100	0011	0111
+ 721	+ 0111	0010	0001
1158	0001	0001	0101
	1	1	5
			8

BCD subtraction : BCD subtraction follows the same rules as binary subtraction. However, if the subtraction causes a borrow and/or creates an invalid BCD numbers, an adjustment is required to correct the answer. The correction method is to subtract 6 from the difference in any digit position that has caused an error.

Example :

$$\begin{array}{r}
 65 - 19 = 46 \\
 0110 \ 0101 = 65 \\
 - 0001 \ 1001 = 19 \\
 \hline
 0100 \ 1100 = 4 ? \text{ (invalid)} \\
 \\
 0100 \ 1100 = 4 ? \text{ (invalid)} \\
 - 0000 \ 0110 = 6 \text{ (adjustment)} \\
 \hline
 0100 \ 0110 = 46
 \end{array}$$

Q 72. What is a Co-Processor and its use ?

(PTU, Dec. 2014)

Ans. A coprocessor is a computer processor used to supplement the functions of the primary processor (the CPU). Operations performed by the coprocessor may be floating point arithmetic, graphics, signal processing, encryption or I/O interfacing with peripheral devices. By offloading processor-intensive tasks from the main processor, coprocessors can accelerate system performance. Coprocessors allow a line of computers to be customized, so that customers who do not need the extra performance don't need to pay for it.

In general, a coprocessor is a separate instruction set processor. It has its own instruction set supporting the special exponential and trigonometric functions. The instruction and registers of coprocessor are just extension of the CPU's instruction set and registers. Both CPU and coprocessor execute their instructions from the same program. The instructions intended for the coprocessor are fetched by the CPU, and jointly decoded by both the CPU and the coprocessor, and executed by coprocessor.

Q 73. Differentiate synchronous and asynchronous bus.

(PTU, Dec. 2014)

Ans. Data transfers over the system bus may be synchronous or asynchronous. In a synchronous bus, each data item is transferred during a time slice known in advance to both source and destination units. Synchronization is achieved by driving both units from a common clock source. An alternative procedure is to have separate clocks of approximately the same frequency in each unit. Synchronization signals are transmitted periodically in order to keep all clocks in the system in step with each other.

In an asynchronous bus, each data item being transferred is accompanied by handshaking control signals to indicate when the data are transferred from the source and received by the destination.

Q 74. What is IO control method ?

(PTU, Dec. 2014)

Ans. IO control method : It refers the data transfer between the IO device and the memory or

between the IO device and the CPU. For e.g., testing the status of device and to determine if they are required service by the CPU.

Q 75. What is a Flip Flop ?

Ans. A flip flop is a binary storage device that store binary bit either 0 or 1. It has two stable states High and Low i.e. 1 and 0. It has the property to remain in one state indefinitely until it is directed by an input signal to switch over to the other state. (PTU, Dec. 2017)

Q 76. What is Multiplexer ?

Ans. Multiplexer which is also known by name MUX is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal. (PTU, Dec. 2017)

Q 77. What are the merits and demerits of single address instructions ?

Ans. Merits of single address instruction is that it is suitable for small set for instruction and the demerits of using single type of instruction is that in practice the codes in an instruction may be fairly small e.g. 1.8 bits. However if the instruction is to be able to reference large quantities of data then the addresses must be large e.g. 16.32 bits. (PTU, Dec. 2015)

Q 78. List various memory reference instructions.

Ans. Various memory reference instructions are :

1. ADD 2. AND 3. LDA 4. STA 5. BUN 6. BSA 7. ISZ

Q 79. Define Guard bit and truncation.

Ans. Guard bit : While adding two floating point numbers with 24 bit mantissas, we shift the mantissa of the number with the smaller exponent to the right until the two exponents are equalized. This implies that mantissa bits may be lost during the right shift (that is, bits of precision may be shifted out of the mantissa being shifted). To prevent this, floating point operations are implemented by keeping guard bits, that is, extra bits of precision at the least significant end of the mantissa. The arithmetic on the mantissas is performed with these extra bits of precision. After an arithmetic operation, guarded mantissas are : (PTU, Dec. 2016)

- Normalized (if necessary)
- Converted back by a process called truncation/rounding to a 24 bit mantissa.

Truncation and rounding :

1. Straight chopping :

- The guard bits (excess bits or precision) are dropped.

2. Von Neumann rounding :

- If the guard bits are all 0, they are dropped.
- However, if any bit of the guard bit is a 1, then the LSB of the retained bit is set to 1

3. Rounding : If there is a 1 in the MSB of the guard bit then a 1 is added to the LSB of the retained bits.

Q 80. State the basic performance equation.

Ans. Basic performance equation :

$$T = \frac{N \times S}{R}$$

Where N is the actual number of instructions executed by the processor for execution of a program, R is a clock rate measured in clocks per second, and S is the average number of steps needed to execute one machine instruction. (PTU, Dec. 2015)

Q 81. What are the types of ALU ? Give advanced features of ALU.

Ans. The arithmetic logic unit perform the arithmetic (add, subtract) and logical operations (and, (PTU, Dec. 2015)

or) on the data made available to it. Whenever an arithmetic or logical operation is to be performed, the required data is transferred from the memory unit to ALU, the operation is performed and the result is returned to memory unit. Before the completion of the processing, data may need to be transferred back and forth several times between these two sections. Subsequently the result are transferred from internal storage to an output device.

In some processors, the ALU is divided into two units :

1. an arithmetic unit (AU) and
2. a logic unit (LU)

The major performance parameters for an ALU are :

1. The variety of arithmetic and logic functions it can perform.
2. The speed of operation.

The ALU subsystem has thus been enhanced in the following ways :

1. Faster algorithms for ALU operations and corresponding hardware implementations.
2. Use of a large number of general-purpose registers in the processor structure, wherein most of the ALU operations are on the data items residing in those registers. This reduces the number of accesses to the memory and hence increases the speed of the ALU.
3. Stack-based ALUs increased the speed of operation since operands and intermediate results could be maintained in stack registers, thus reducing the memory access requirements. Note that this concept refers to the so called zero-address machines, in which the arithmetic and logic operations always refer to the operands on the top one or two levels of a stack. A set of registers in the processor are interconnected to form a stack.

Q 82. List and explain the steps involved in the execution of a complete instruction.

(PTU, May 2018 ; Dec. 2015)

Ans. Steps involved in the execution of a complete instruction :

1. Fetch the instruction
 2. Fetch the operand from memory location pointed by R_2
 3. Perform the addition
 4. Store the result in R_1 .
1. The instruction fetch operation is initiated by loading the contents of the PC into the MAR and activating Read signal. PC contents are also loaded into register Y and added with constant number by activating select C input of multiplexer and add input of the ALU. By activating Z_{in} signal result is stored in the register Z.
 $PC_{out}, MAR_{in}, Read, Y_{in}, select\ C, Add, Z_{in}$.
 2. The contents of register Z are transferred to PC register by activating Z_{out} and PC_{in} signal. This completes the PC increment operation and PC will now point to next instruction. After receiving WMFC signal, the contents of specified location are available in MDR register.
 $Z_{out}, PC_{in}, WMFC$
 3. In step 3 contents of MDR register are transferred to the instruction register (IR) of the processor. The step 1 through 3 constitute the instruction fetch phase. At the beginning of step 4, the instruction decoder interprets the contents of the IR. This enables the control circuitry to activate the control signals for steps 4 through 1, which constitute the execution phase.
 MDR_{out}, IR_{in}
 4. In step 4, the contents of register R_2 are transferred to register MAR by activating R_{2out} and MAR_{in} signals and Read signal is activated.
 $R_{2out}, MAR_{in}, Read$
 5. In this the contents of register R_1 are transferred to register Y by activating R_{1out} and Y_{in} signals.

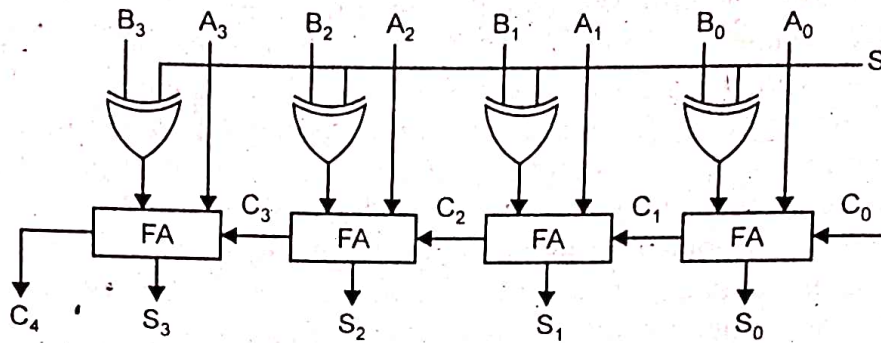
After receiving WMFC signal, the contents of specified location are available in MDR register
 $R_{1out}, Y_{in}, WMFC$

6. In this MDR_{out} , select Y, Add and Z_{in} signals are activated to perform addition of contents of register, and the contents of MDR. The result is stored in the register Z.
 $MDR_{out}, Select Y, Add, Z_{in}$
7. In this the contents of register Z are transferred to register R_1 by activating Z_{out} and R_{1in} signals. The end signal causes a new instruction fetch cycle to begin by returning to step 1.
 Z_{out}, R_{1in}, END

Q 83. Draw the block diagram of a 2's complement Subtractor.

(PTU, May 2016)

Ans.



Q 84. Give two applications of three-Address instruction.

(PTU, Dec. 2016)

Ans. Computers with three address instruction format can use each address field to specify either processor register or memory operand :

ADD R1, A, B $A1 \text{ (R) } M[A] + M[B]$

ADD R2, C, D $R2 \text{ (R) } M[C] + M[D]$
 $x = (A + B) * (C + A)$

MUL X, R1, R2 $M[X] R1 * R2$

The advantage of three address formats is that it results in short program when evaluating arithmetic expression.

Q 85. What are the two approaches to reduce delay in adders ?

(PTU, May 2017)

Ans. Carry select adder (CSLA) and carry look ahead adders are used to reduce delay in adders.

Q 86. Show that the dual of EX OR is also its complement.

(PTU, Dec. 2017)

Ans. $A \text{ XOR } B = A'B + AB'$
 $(A \text{ XOR } B)' = AB + A'B'$

In dual we replace AND with OR and OR with AND

Dual of $(A \text{ XOR } B) = (A' + B)(A + B') = A'A + A'B' + AB + BB' = AB + A'B'$

from (1) and (2)

Dual of XOR = Complement of XOR.

Q 87. What is the difference between static and dynamic network interconnections ? Discuss any three network interconnection networks.

(PTU, May 2017)

Ans. Static validation techniques look at configurations and network topologies in order to identify security. Configuration errors, while dynamic verification supplements that with actual traffic logs. These techniques are commonly used to verify firewall configurations.

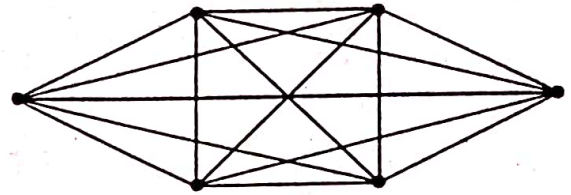
Static validation has the advantage of being performed offline, and it can be completed prior to deploying a security configuration. It can detect errors such as shadowed rules (These are rules that will never be triggered because an earlier rule covers all of the traffic that would be covered by the shadowed rule)

Dynamic analysis provides deeper insight into a rule base. For example, only dynamic analysis can detect orphaned rules : rules that are syntactically correct but will never be triggered due to changes in the way the network operates. For example static analysis will never reveal that a database

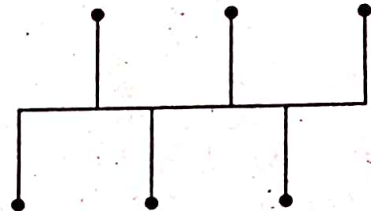
server has been decommissioned, while dynamic analysis will identify that the rule has not been triggered in a long time, allowing you to proactively clean up the rule base.

Network interconnections

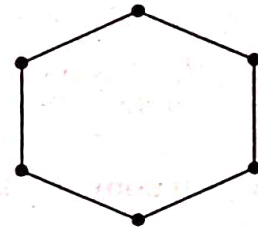
Mesh : In a mesh network, multiple nodes are connected with each other. Each node in the network is connected to every other node in the network. This arrangement allows proper communication of the data between the nodes. But, there are lot of common overheads due to the increased number of node connections.



Shared bus : This network topology involves connection of the nodes with each other over a bus. Every node communicate with every other node using the bus. The bus utility ensures that no data is sent to the wrong node. But the bus traffic is an important parameter which can affect the system.



Ring : This is one of the simplest ways of connecting nodes with each other. The nodes are connected with each other to form a ring. For a node to communicate with some other node, it has to send the message to its neighbour. Therefore the data message passes through a series of other node before reaching the destination. This involves increased latency in the system.



Q 88. State the condition in which overflow occurs in case of addition & subtraction of two signed 2's complement number. How is it detected ? (PTU, Dec. 2017)

Ans. Overflow : When two numbers of n digits each are added and sum occupies n + 1 digits an overflow occurs

- Addition of two unsigned numbers, detected from the end carry of the most significant position.
- Addition of two signed numbers. May occur If the two numbers are both positive or both negative.

Signed numbers

- The leftmost bit always represents the sign and negative numbers are in 2's complement form.
- Addition : the sign bit is treated as part of the number and the end carry does not indicate an overflow.

Overflow for signed numbers

Example : Two signed binary numbers +70 and +80 stored in two bit register (+127 to -128)

Carries :	0 1	
+ 70	0 1000110	
+ 80	0 1010000	
+ 150	1 0010110	

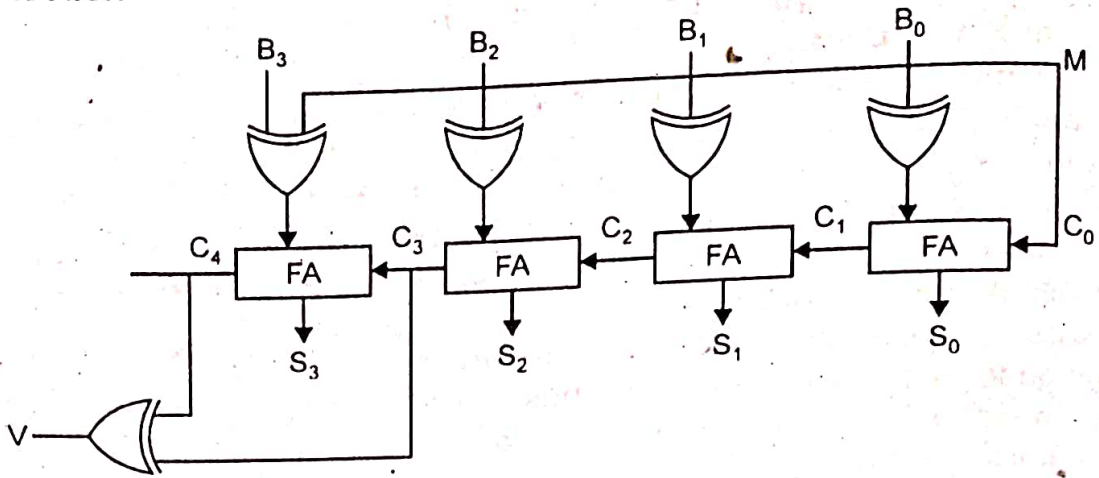
Carries :	1 0	
- 70	1 0111010	
- 80	1 0110000	
- 150	0 1101010	

Detecting overflow condition by observing

- the carry into the sign bit position
- the carry out of the signbit position.

An overflow has occurred If two carries are not equal (V bit)

Overflow Detection



4-bit adder subtractor

- Overflow for unsigned numbers C bit detects a carry after addition or a borrow after subtraction.
- Overflow for signed numbers.
 - $V = 0$ after an addition or subtraction : indicate no overflow
 - $V = 1$: the result of the operation contains $n + 1$ bits
- An overflow has occurred
- The $(n + 1)$ th bit is the actual sign and has been shifted out of position.

Q 89. Design a combinational circuit that generates 9's complement of a BCD digit. (PTU, Dec. 2017)

Ans.

Digit	x_3	x_2	x_1	x_0	y_3	y_2	y_1	y_0
0	0	0	0	0	1	0	0	1
1	0	0	0	1	1	0	0	0
2	0	0	1	0	0	1	1	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	0	1
5	0	1	0	1	0	1	0	0
6	0	1	1	0	0	0	1	1
7	0	1	1	1	0	0	1	0
8	1	0	0	0	0	0	0	1
9	1	0	0	0	0	0	0	0

$y_0 = \bar{x}_0$

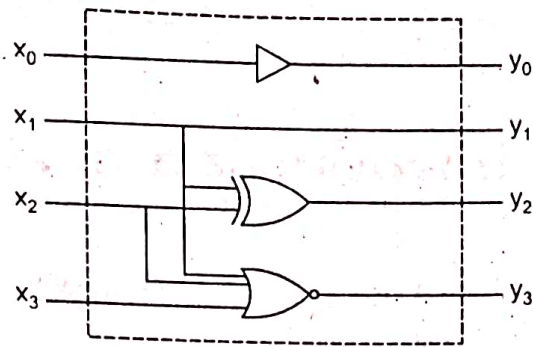
$y_1 = x_1$

x_3x_2	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	X	X	X	X
10	0	0	X	X

$y_2 = x_2 \bar{x}_1 + \bar{x}_2 x_1$
 $= x_2 \oplus x_1$

	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	X	X

$y_3 = \bar{x}_3 \bar{x}_2 \bar{x}_1$



Q 90. Design a sequential circuit with JK flip flop to satisfy the following state equation.

$$A(t+1) = A'B'CD + A'B'C + ACD + AC'D'B(t+1) = A'C + CD' + A'B'C(t+1) = BD(t+1) = D'$$

(PTU, Dec. 2017)

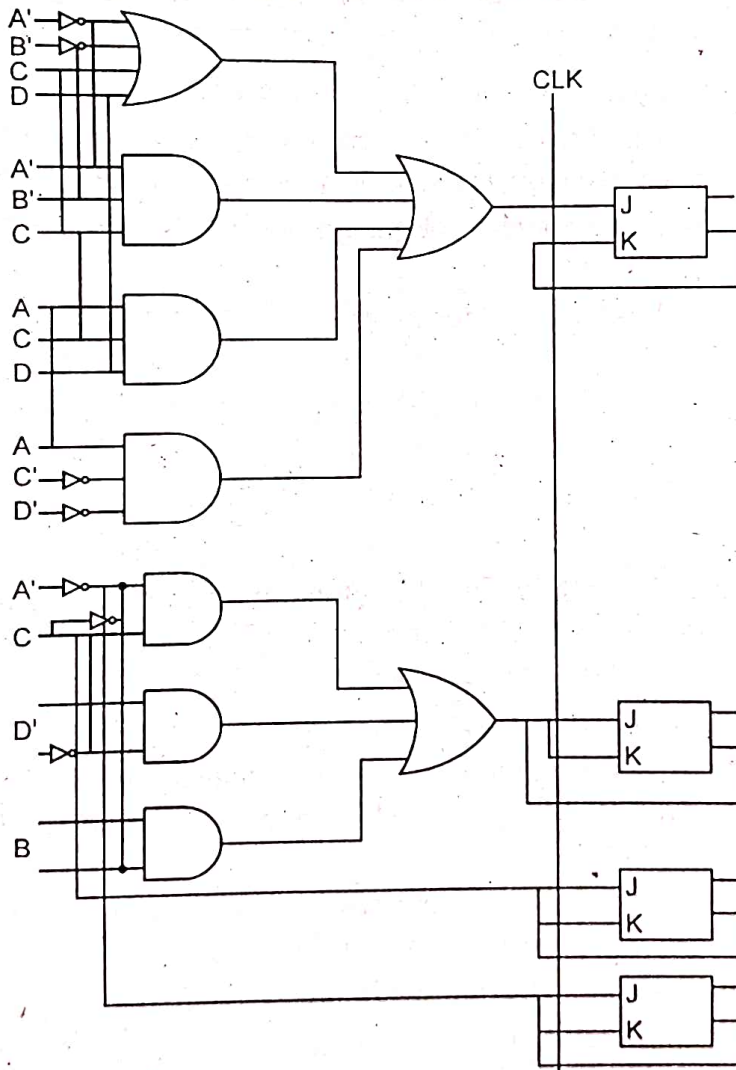
Ans.

$$A(t+1) = A'B'CD + A'B'C + ACD + ACD'$$

$$B(t+1) = A'C + CD' + A'BC'$$

$$C(t+1) = B$$

$$D(t+1) = D'$$



Chapter

2

Introduction to x86 Architecture

Contents

CPU Control Unit Design : Hardwired and micro-programmed design approaches, Case study - design of a simple hypothetical CPU.

Memory system design : Semiconductor memory technologies, memory organization.

Peripheral devices and their characteristics : Input-output subsystems, I/O device interface.

I/O transfers - program controlled, interrupt driven and DMA, privileged and non-privileged

instructions, software interrupts and exceptions. Programs and processes - role of interrupts

in process state transitions, I/O device interfaces - SCII, USB.

POINTS TO REMEMBER

- ☞ The digital computer is a digital system that performs various computational tasks.
- ☞ Digital computers use the binary number system, which has two digits 0 and 1. A binary digit called a bit and information in a digital computer is represented in the group of bits.
- ☞ Computer architecture is the conceptual design and fundamental operational structure of computer system
- ☞ The various abstraction layer in CPU design are :
 - (i) Hardware
 - (ii) Firmware
 - (iii) Assembler
 - (iv) Kernel
 - (v) Application.
- ☞ A machine language is a collection of all the fundamental instructions that the machine can execute, expressed as a pattern of 1's and 0's. However a high level language needs to be translated to machine language for execution.
- ☞ A real time computer performs the given task or operation within or by the given deadline.
- ☞ Process control system : After completion of one process the other process starts, which is managed by operating system. It makes the efficient use of computer resources. These processes may be in queue or in a stack.
- ☞ Computer organization is concerned with the way the hardware components operate and the way they are connected together to form the computer system.
- ☞ Computer design concerned with the hardware design of the computer.
- ☞ Computer architecture is concerned with the structure and behaviour of the computer as seen by the user. It includes the information, formats, the instruction set and techniques for addressing.

memory. The architectural design of a computer system is concerned with the specifications of the various functional modules such as processors and memories and structuring them together into a computer system.

- ☞ The basic type of computer architectures are non-Neumann architecture and hardware architecture.
- ☞ The manipulation of binary information is done by logic circuits called gates. Gates are blocks of hardware that produce signals of binary 0 or 1 when input requirements are satisfied.
- ☞ The input-output relationship of a binary variable for each gate can be represented in the form of a table called 'Truth Table'.
- ☞ Compliments are used in digital computers for simplifying the subtraction operation and logical manipulation.
- ☞ Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplication from bit weight 2^k to weigh 2^m can be treated as $2^{k+1} - 2^m$.
- ☞ During addition and subtraction, the two floating-point operands are in AC and BR. The sum or difference is formed in the AC. The algorithm can be divided into four consecutive parts as below :
 - (i) Check zeros
 - (ii) Align the mantissas
 - (iii) Add or subtract the mantissas
 - (iv) Normalize the result.
- ☞ Control functions specify a micro operation is a binary variable.
- ☞ Control variable does not change the state of the register in the system.
- ☞ Control variable at a given time can be represented by a string of 1's and 0's called a control word.
- ☞ Micro programmed control unit in a control unit whose binary control variables are stored in memory.
- ☞ Each word in control memory contains within it a micro instruction.
- ☞ A sequence of micro instructions constitutes a micro program.
- ☞ A memory that is the part of control unit is referred to as a control memory.
- ☞ Control memory address register specifies the address of the micro instructions.
- ☞ Control data register holds the micro instruction read from memory.
- ☞ Micro instructions are stored in control memory in a group, with each group specifying a routine.
- ☞ Hardwired control unit is implemented as a logic circuit in the hardware.
- ☞ Program interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request. Control returns to the original program after the service program is executed.
- ☞ There are three types of interrupts that cause a break in the normal execution of a program. These can be classified as :
 - (i) External Interrupts

- (ii) Internal Interrupts
- (iii) Software Interrupts.
- ☞ Data transfer to and from peripherals may be handled in one of three possible modes :
 - (i) Programmed I/O
 - (ii) Interrupt initiated I/O
 - (iii) Direct memory access (DMA).
- ☞ A polling procedure is used to identify the highest-priority source by software means. On this method there is one common branch address for all interrupts.
- ☞ The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path and lettering the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called 'Direct Memory Access' (DMA).
- ☞ A DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device. In addition, it needs an address register, a word count register and a set of address lines.
- ☞ The input-output processor (IOP) is similar to a CPU that it is designed to handle the details of I/O processing. Unlike the DMA controller that must be set up entirely by the CPU, the IOP can be fetch and execute its own instructions.
- ☞ The most striking difference between an I/O processor and a data communication processor is the way the processor communicate with the I/O devices.
- ☞ An I/O processor communicate with the peripherals through a common I/O bus that is comprised of many data and control lines. A data communication processor communicates with each peripheral through a single pair of wires.
- ☞ I/O controller is the module that controls an I/O function.

QUESTION-ANSWERS

Q 1. What is the difference between Machine Language and High Level Language?

(PTU, May 2004)

Ans. A machine language is the collection of all the fundamental instructions that the machine can execute, expressed as a pattern of 1s and 0s. However a high level language needs to be translated to machine language for execution.

Q 2. Define the terms real time computer and process control computer.

(PTU, Dec. 2009, 2005)

Ans. Real time Computer : A real time computer performs the given task or operation within or by the given deadline.

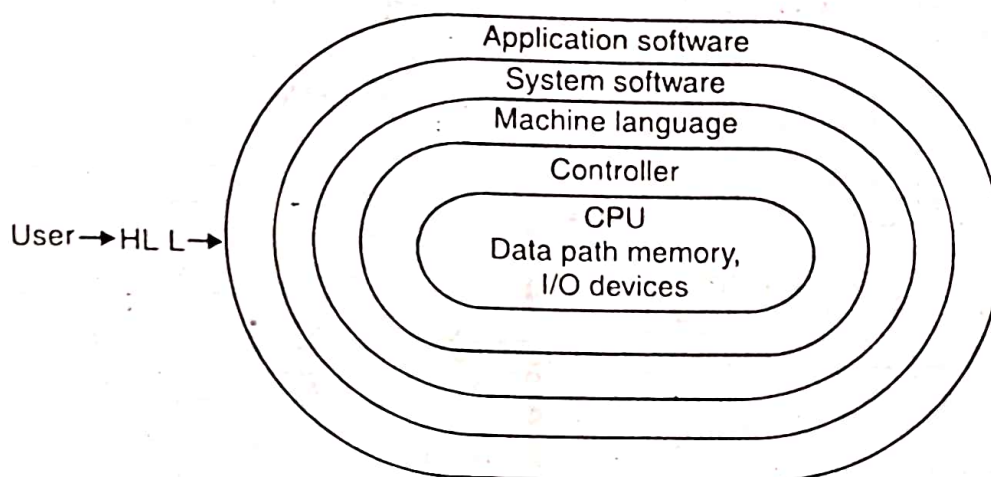
For example : Controlling the air craft system
Process controlled computer : It controls the given process but it has deadline for processing it.

Process control system : The various processes can be control by software of a computer system. These process may be in Queue or a Stack. After completion of one process the other process starts, which is managed by operating system. It makes the efficient use of computer resources.

Q 3. Give the layered view of a computer system.

Ans.

(PTU, May 2005 ; Dec. 2009, 2005, 2004)



A layered view of a computer system

It is the layered view of a computer system. The first two layers form hardware of computer system. The third layer when combines with fourth layer form software. User interact through high level language.

Q 4. Draw top leveled view of computer components.

(PTU, Dec. 2007)

Ans. A **computer** is a machine that manipulates data according to a list of instructions. A

Program consists of the following :

- A sequence of steps.
- For each step, an arithmetic or logical operation is done.
- For each operation, a different set of control signals is needed.

Function of Control Unit

- For each operation a unique code is provided
e.g. ADD, MOVE
- A hardware segment accepts the code and issues the control signals.

Instruction Cycle

There are two steps in instruction cycle :

- Fetch.
- Execute

A levelled view of instruction cycle is shown in the Fig. 1.

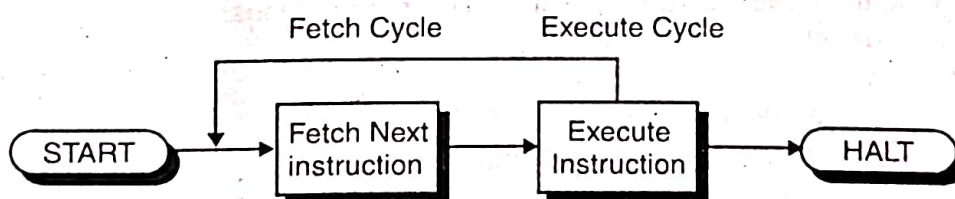


Fig. 1 Instruction cycle

Components

The **computer components** : Top Level View is shown in the fig. 2. The computer components of top levelled view must contains the following :

- The Control Unit and the Arithmetic and Logic Unit constitute the Central Processing Unit
- Data and instructions need to get into the system and results out
 - Input/output
- Temporary storage of code and results is needed
 - Main memory.

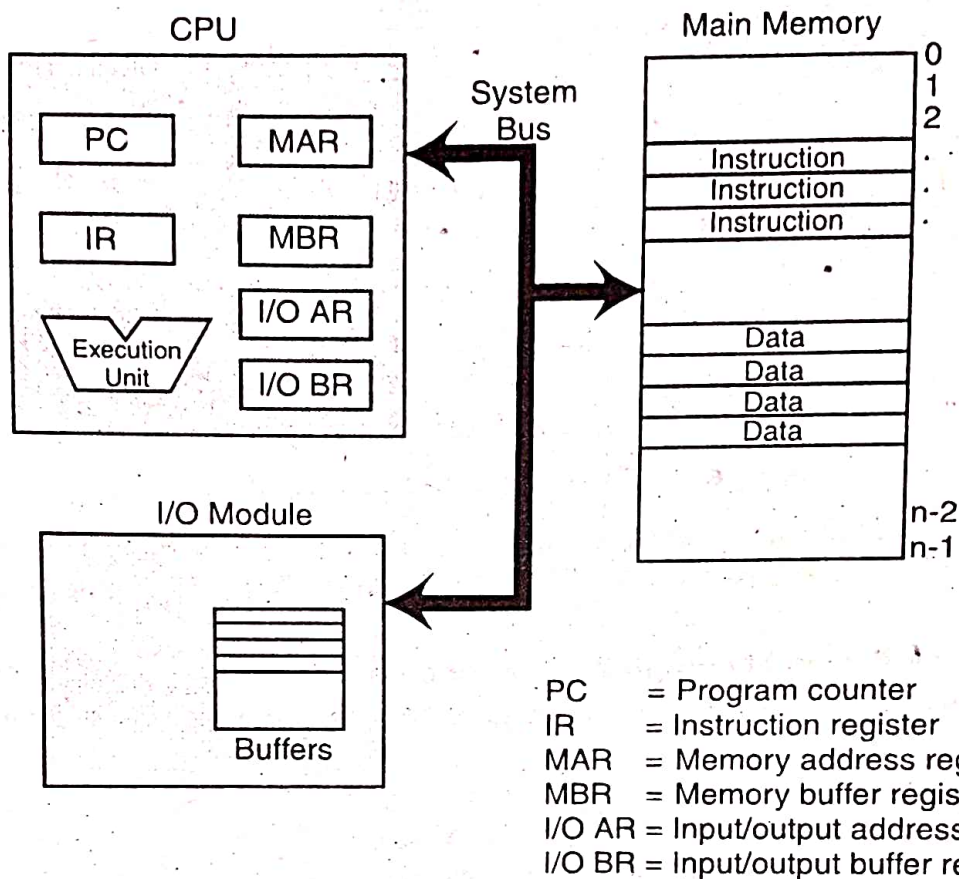


Fig. 2 Computer Components : Top Level View

Q 5. Write typical physical realization of bus architecture.

(PTU, Dec. 2007)

Ans. The Bus architecture may compose of :

- There are a number of possible interconnection systems.
- Single and multiple BUS structures are most common.
- e.g. Control/Address/Data bus (PC).
- e.g. Unibus (DEC-PDP)

Bus

- A communication pathway connecting two or more devices.
- Usually broadcast
- Often grouped
 - A number of channels in one bus
 - e.g. 32 bit data bus is 32 separate single bit channels
- Power lines may not be shown

There are three basic types of buses :

(i) Data Bus (ii) Address Bus (iii) Control Bus

(i) Data Bus

- Carries data
 - Remember that there is no difference between "data" and "instruction" at this level.

- ❑ Width is a key determinant of performance
 - 8, 16, 32, 64 bit
- (ii) **Address Bus**
 - ❑ Identify the source or destination of data
 - ❑ e.g. CPU needs to read an instruction (data) from a given location in memory
 - ❑ Bus width determines maximum memory capacity of system.
 - e.g. 8080 has 16 bit address bus giving 64k address space.
- (iii) **Control Bus**
 - ❑ Control and timing information
 - Memory read/write signal
 - Interrupt request
 - Clock signals

Bus Interconnection Scheme : There exist the three types of basis buses - data, address and control buses. The bus interconnection scheme is as shown in fig. 3.

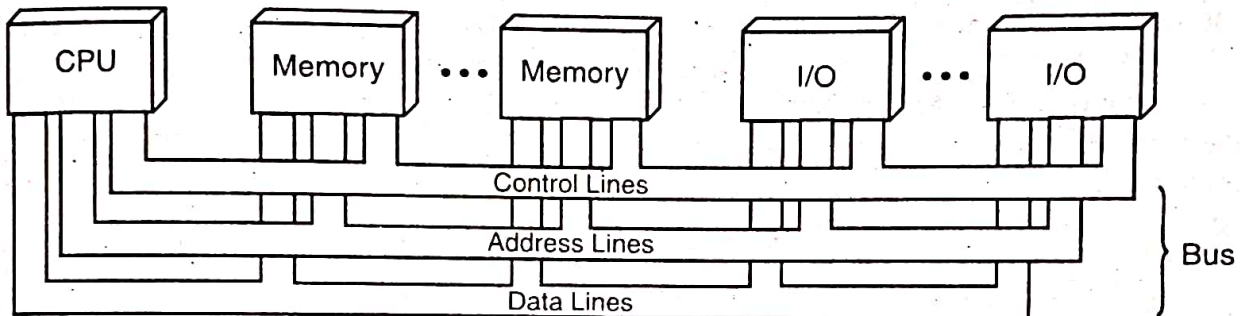


Fig. 3 Bus Interconnection Scheme

The buses look like :

- Parallel lines on circuit boards
- Ribbon cables
- Strip connectors on mother boards
- e.g. PCI
- Sets of wires

Physical Realization of Bus Architecture : The typical physical realization of bus architecture must contain CPU, memory and I/O. The physical realization of bus architecture is shown in fig. 4.

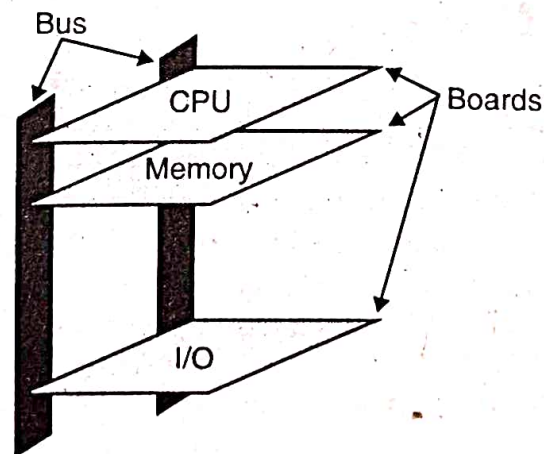


Fig. 4 Physical realization of Bus Architecture

Q 6. What is the concept of layers in architectural design?

(PTU, Dec. 2008)

OR

Explain the significance of a layered architecture.

(PTU, May 2009)

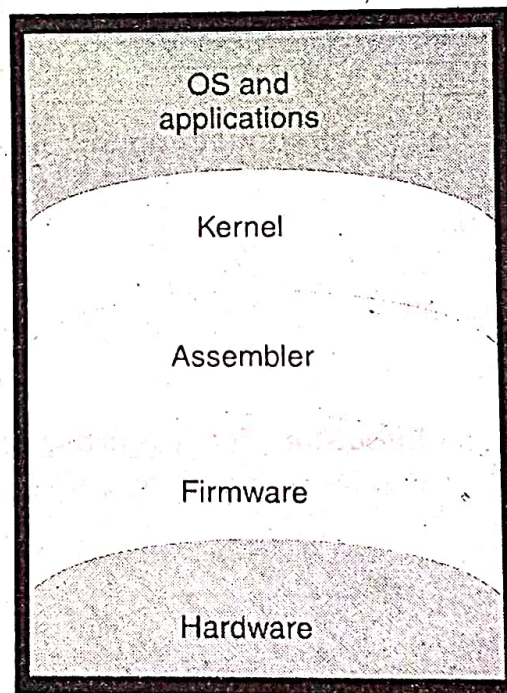
Ans. Computer Architecture is the conceptual design and fundamental operational structure of a computer system. It is a blueprint and functional description of requirements (especially speeds and interconnections) and design implementations for the various parts of a computer.

It focuses largely on the way by which the **central processing unit (CPU)** performs internally and accesses addresses in **memory**.

A typical vision of a computer architecture as a series of Abstraction Layers :

1. Hardware
2. Firmware
3. Assembler
4. Kernel
5. Applications

In computer system architecture design, a **layer** is a group of classes that have the same set of link-time module dependencies to other modules. In other words, a layer is a group of reusable components that are reusable in similar circumstances. Layers are often arranged in a tree-form hierarchy, with dependency relationships as links between the layers. Dependency relationships between layers are often either inheritance, composition or aggregation relationships, but other kinds of dependencies can also be used.



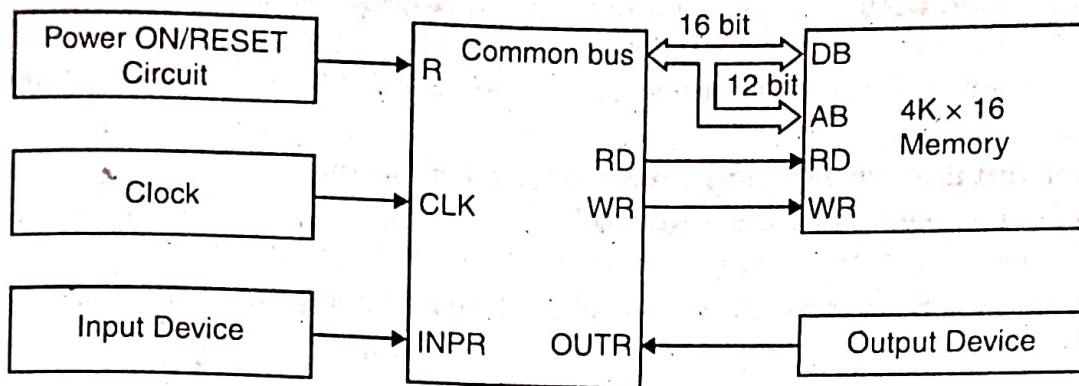
Q 7. What are the principles of computer design?

(PTU, May 2010)

Ans. In a basic principle of computer design the following hardware components are required

1. A memory unit with 4096 words of 16 bits each.
2. Nine registers : AR, PC, DR, AC, IR, TR, QUTR, INPR and SC.
3. Seven flip flops : I, S, E, R, IEN, FGI and FGO.
4. Two decoders : a 3×3 operation decoder and a 4×16 timing decoder.
5. A 16-bit common bus.
6. Control logic gates.
7. Adder and logic circuit connected to the input of AC.

The functional block diagram of a basic computer is as shown below :



Q 8. Give the features of a ROM cell.

(PTU, May 2016, 2010)

Ans. ROM cell is a small part of a computer which means read-only memory cell and it provides space for the programs for booting and the startup of PC. Read only memory, shortly ROM enables an electronic device to regulate their work motion which is inserted in the integrated circuit. ROM is primarily used in various electro-mechanical devices and is less used in computer IC. ROM is a device which is mainly used for storage purposes. The data stored in ROM is not readily modifiable and the data is used to run a specific hardware with certain software, which in terms is said firmware.

The ROM cell usually consists of a single transistor (ROM and EPROM cells consist of one transistor, EEPROM cells consist of one, one-and-a-half, or two transistors). The threshold voltage of the transistor determines whether it is a "1" or "0". During the read cycle, a voltage is placed on the gate of the cell. Depending on the programmed threshold voltage, the transistor will or will not drive a current. The sense amplifier will transform this current, or lack of current, into a "1" or "0".

ROM is classified into five main classes :

- ❑ **Classical read-only memory (ROM) :** Classic mask programmed ROM are the oldest model of ROM. The data and program is stored permanently, which can't be changed after manufacturing.
- ❑ **Programmable read-only memory (PROM) :** PROM is also a one time programmable ROM which is programmed with the aid of special devices. Consequently this ROM creates internal links within the IC which is inerasable.
- ❑ **Erasable programmable read-only memory (EPROM) :** This is a type of ROM which enables a device to erase the data in ROM. When the ROM is exposed to ultra-violet (UV) ray for more than 10 minutes, the stored data is deleted and it enables to input new programs. This erasing cycle can be done 1000 times with an EPROM.
- ❑ **Electrically erasable programmable read-only memory (EEPROM) :** EEPROM is very much similar to EPROM. The main difference is that the data is erased electrically, where with EPROM UV ray is used.
- ❑ **Flash memory :** Flash memory is the modified version of EEPROM. It is faster than EEPROM and rewriting and deleting is easy with it. Recently NAND flash has made this easier and flash memory is mostly used in electronic devices now a day.

Q 9. Convert the following logic function into z minterm :

$$A'B'CDE' + A'BCDE + AB'CD'E'' + ABCD'E.$$

(PTU, Dec. 2004)

Ans. The minterm of given logic function will be $\Sigma(6, 15, 20, 29)$

Q 10. Convert the following logic function into minterm :

$$A'BC'D'E'F + A'BC'DEF' + AB'C'DEF' + AB'C'DEF' + ABC'DE'F'$$

(PTU, May 2005)

Ans. The minterm of the given logic function will be $\Sigma(17, 22, 38, 52)$

Q 11. What is the role of Binary Counters in digital computers. (PTU, May 2005)

Ans. Binary Counters : A register that goes through a predetermined sequence of states upon the application of Input pulses is called counter. They are used for counting the number of occurrences of an event and are useful for generating timing signals to control the sequence of operations in digital computers.

Q 12. Convert the following logic function in to minterm.

$$ABC'DE' + AB'C'DE' + ABCDE' + AB'CD'E'$$

(PTU, Dec. 2009, 2005)

Ans. The minterm of the given logic function will be $\Sigma(18, 20, 26, 30)$

Q 13. Difference between Computer Architecture and computer organization.

(PTU, Dec. 2008 ; May 2007)

Ans.

Computer Architecture	Computer Organization
1. Computer architecture is defined as the functional operation of the individual hardware units in a computer system and the flow of information among these units and also the controlling of those units.	1. The function and design of the various units of digital computers that store and process information. It also deals with the units of the computer that receive information from and that send computer resumes to the outside worlds.
2. The computer architecture also refers to those attributes of a system which are visible to a programmer or those attributes that have a direct impact on the logical execution of a program.	2. Computer hardware is the electronic circuit and electronic mechanical equipment that build the computer so simply computer organization refers to the hardware.

Q 14. Differentiate between program interrupt and subroutine call.

(PTU, May 2015 ; Dec. 2016, 2013, 2011)

Ans. The main difference between an interrupt and a sub-routine call is that an interrupt can occur at any time during the execution of the main program, whereas a sub-routine call can only be carried out when, during the scan time of the main program, the call instruction is executed.

Q 15. What are the issues in computer design?

(PTU, May 2015 ; Dec. 2011)

Ans. The issues in computer design are as follows :

- (i) Addressing
- (ii) Error control
- (iii) Flow control
- (iv) Multiplexing
- (v) Demultiplexing
- (vi) Routing.

Q 16. What is superscalar machine?

(PTU, May 2012)

Ans. A superscalar CPU architecture implements a form of parallelism called instruction level parallelism within a single processor. It therefore allows faster CPU throughput than would otherwise be possible at a given clock rate. A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor. Each functional unit is not a separate CPU core but an execution resource within a single CPU such as an arithmetic logic unit, a bit shifter, or a multiplier.

Q 17. Discuss the importance of performance measure in computer hardware design. Also state the advantages and disadvantages of layers in architectural design.

(PTU, Dec. 2011)

Ans. Modern computer architectural performance is often described in MIPS per MHz (millions of instructions per second per millions of cycles per second of clock speed). This metric explicitly measures the efficiency of the architecture at any clock speed. Since a faster clock can make a faster computer, this is a useful, widely applicable measurement. Historic complex instruction set computers had MIPS/MHz as low as 0.1. Simple modern processors easily reach near 1. Superscalar processors may reach three to five by executing several instructions per clock cycle. Multicore and vector processing CPUs can multiply this further by acting on a lot of data per instruction, and have several CPUs executing in parallel. Counting machine language instructions would be misleading because they can do varying amounts of work in different ISAs. The "instruction" in the standard measurements is not a count of the ISA's actual machine language instructions, but a historical unit of measurement, usually based on the speed of the VAX computer architecture. Historically, many people measured the speed by the clock rate (usually in MHz or GHz). This refers to the cycles per second of the main clock rate may not necessarily have higher performance. As a result manufacturers have moved away from clock speed as a measure of performance.

Advantages and disadvantages of Layered Architecture Design : The following are the advantages of a layered architecture :

Layered architecture increases flexibility, maintainability and scalability. In a layered architecture we separate the user interface from the business logic, and the business logic from the data access logic. Separation of concerns among these logical layers and components is easily achieved with the help of layered architecture.

Multiple applications can reuse the components : For example if we want a windows user interface, this can be done in an easy and fast way by just replacing the UI component. All the other components like business logic, data access and the database remains the same. Layered architecture allows to swap and reuse components at will.

Layered architecture enables teams to work on different parts of the application parallelly with minimal dependencies on other teams.

Layered architecture enables develop loosely coupled systems.

Different components of the application can be independently deployed, maintained, and updated, on different time schedules.

Layered architecture also makes it possible to configure different levels of security to different components deployed on different boxes. So layered architecture, enables you to ensure portions of the application behind the firewall and make other components accessible from the Internet.

Layered architecture also helps you to test the components independently of each other.

The following are the disadvantages of a layered architecture : There might be a negative impact on the performance as we have the extra overhead of passing through layers instead of calling a component directly.

Development of user-intensive applications can sometime take longer if the layering prevents the use of user interface components that directly interact with the database.

The use of layers helps to control and encapsulate the complexity of large applications, but adds complexity to simple applications.

Changes to lower level interfaces tend to percolate to higher levels, especially if the relaxed layered approach is used.

Q 18. Explain the role of a compiler.

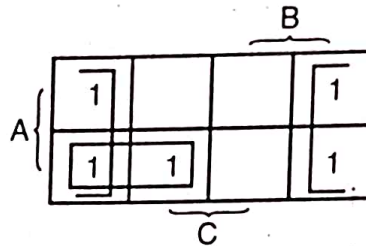
Ans. Compiler is language processor used to translate program written in high level language into the machine level language. It is also use to cover the "gap" between humans.

Computer language : A program written in high level programming language is called the **source program**. The source program is stored on the disk in a file is called the object file. The object file contains the translated program files, source and object are saved on the disk permanently.

Q 19. Simplify the following boolean function using three variable K-map.

$$F(x, y, z) = \Sigma(1, 2, 3, 6, 7).$$

Ans. The five minterms are marked with 1's in the corresponding squares of the three variable map as shown in fig. The four squares in the first and fourth columns are adjacent and represent the term C' . The remaining, square marked with a 1 belong to minterm 5 and can be combined with the square of minterm 4 to produce the term AB' . The simplified function is $F = C' + AB'$



Q 20. Explain microprogrammed control.

Ans. Microprogrammed Control Implementation is a software approach in contrast to the hardware approach. It deals with various units of software but at the micro level i.e. microoperation, microinstruction, microprogram etc. It consists of the following elements :

1. Control Memory (CM) : The set of micro-instructions is stored in control memory also called the 'Control Store'.

2. Control Address Register : It contains the address of next micro instructions to be read. This is similar to program counter which stores the address of next instruction.

3. Control Buffer Register : When a micro-instruction is read from control memory, it is transferred to control buffer register.

4. Sequencing Logic : The fourth element is a sequencing unit that loads the control address register that with the address of the next instruction to be read and issues a read command to control memory.

Q 21. Give the comparison between and examples of hardwired control unit and microprogrammed control unit.

Ans. 1. Hardware Control : A hardware control unit is implemented as a logic circuits (gates, flip flops, decoders etc.) in the hardware. The inputs to control unit are the instruction register, flags, timing signals and control bus signals. On the basis of input and output signal sequence are generated. These output control signal are functions of input. Thus we can derive a logical function for each control signal. This organization is very complicated, if we have a large control unit. In this organization if design has to be modified or changed, requires changes in wiring among various components. Thus combination circuits modification found to be very difficult. Therefore, a new approach of micro programming is used. The advantage of this control is to produce fast mode of operation.

A hardware control unit has a processor that generates signals or instructions to be implemented in correct sequence. This was the older method of control that works through the use of distinct components, drums, a sequential circuit design, or flip chips.

2. Microprogrammed Control : A micro programmed control unit on the other hand makes use of a micro sequencer from which instruction bits are decoded to be implemented. It acts as the device

supervisor that controls the rest of the subsystems including arithmetic and logic units, registers, instruction registers, off-chip input/output and buses. In microprogrammed organization, the control information is stored in a control memory. The control memory is programmed to initiate a required sequence of microoperations. In microprogrammed control, any required change or modification can be done by updating microprogram in control memory by the programmer.

Q 22. Explain the difference between Hardwired control and micro programmed control.

Is it possible to have hardwired control associated with a control memory ?

(PTU, Dec. 2015, 2014 ; May 2011, 2010, 2007)

OR

What is the difference between a hardwired implementation and a micro programmed implementation of a control unit?

(PTU, May 2018 ; Dec. 2013, 2007)

Ans. 1. Hardwired Control : A hardwired control unit is implemented as a logic circuits (gates, flip-flop, decoders etc.) in the hardware. The inputs to control unit are the instruction register, flags, timing signals and control bus signals. On the basis of these inputs the output signal sequence are generated. Thus output control signals are functions of inputs. Thus always we can derive a logical function for each control signal. This organization is very complicated if we have a large control unit. In this organization if the design has to be modified or changed, requires changes in the wiring among the various components. Thus the modification of all the combinational circuits may be very difficult. Therefore a new approach microprogramming was used. The advantage of this control is to produce a fast mode of operation.

2. Microprogrammed Control : In the microprogrammed organization. the control information is stored in a control memory. The control memory is programmed to initiate the required sequence of microoperations. In the microprogrammed control any required changes or modification can be done by updating the microprogram in the control memory by the programmer.

Difference between hardwired control and microprogrammed control

Hardwired Control	Microprogrammed Control
The Technology is circuit based.	The technology is software based instructions and is stored in control memory.
Hardwired is implemented through flip-flops, gates, decoder, and other circuits.	The decoding of microinstructions generates appropriate signals to control the execution of an instruction.
The Instruction format criteria is fixed.	The Instruction formats types are variable. (16-64) bits per instruction.
The instructions are Register based.	The instructions are not Register based.
Faster Decoding	Slow Decoding.
Large number of (32-192) general purpose registers	(8-24) general purpose register.
Use of split Cache.	Use of unified Cache.
Clock Rate is 50-150 MHz	Clock Rate is 33-50 MHz
Cycle per instruction (CPI) lies as less than 1.5.	Cycle per instruction (CPI) lies between 2 and 15.
No use of ROM.	ROM is used.
It is used in RICS	It is used in CISC
By using hardwired, it reduces CPI of RISC to one to two cycles.	Once the hardware configuration is stabilized, there is no need to further wiring or hardware changes.

Q 23. What is the necessity of connecting variety of memory devices to a computer? (PTU, Dec. 2010)

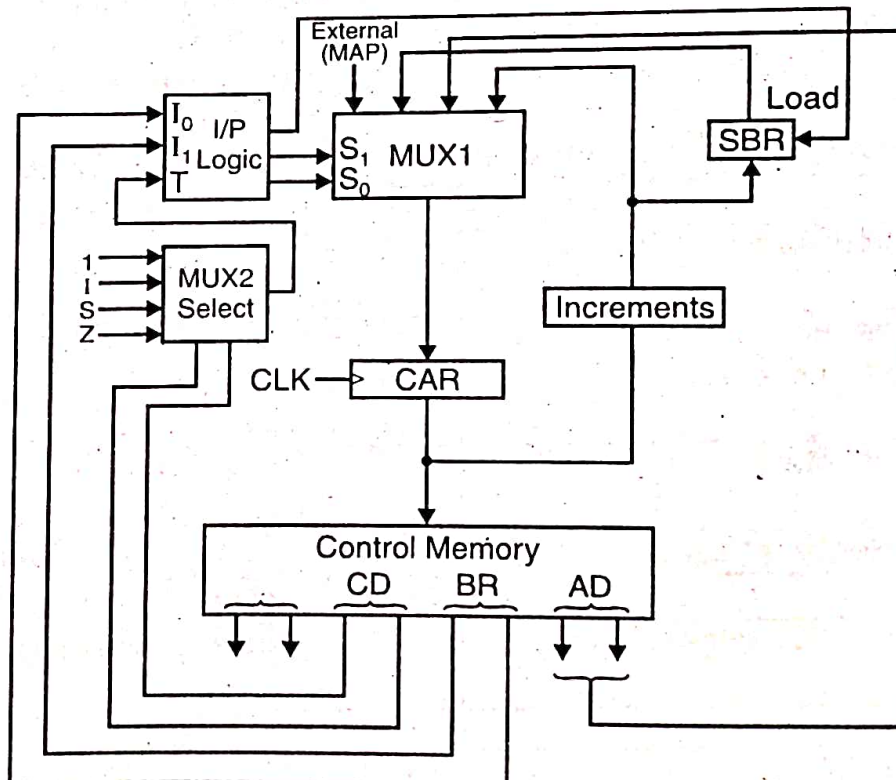
Ans. Computer memory device that is used to store data or programs (sequences of instructions) on temporary or permanent basis for use in an electronic digital computer. Computers represent information in binary code, written as sequences 0s and 1s. Each binary digit (or "bit") may be stored by any physical system that can be in either of two stable states, to represent 0 and 1. Such a system is called bistable. This could be an on-off switch an electrical capacital that can have a pit or not today capacitors and transistors ; functioning as tiny electrical switches, are used for temporary storage, and either disks or taps with a magnetic coating or plastic discs with patterns of pits are used for long-term storage.

Q 24. What do you understand by a microoperation ? Explain. Draw the diagram of a microprogram sequence and explain its working. (PTU, May 2017, 2004)

Ans. The operations executed on data stored in registers are called microoperations. A microoperation is an elementary operation performed on information stored in one or more registers. The result of operation may replace the previous binary information of a register or may transfer it to another register.

e.g. Shifts, cont, clear and Load.

Microprogram sequencer : The basic components of a microprogrammed control unit are the control memory and circuits that select next address, the address selection part is called microprogram sequencer.



There are two multiplexes in the circuit. The 1st selects and address from one of 4 sources and routes it into control address register CAR. The second tests the value of a selected status bit and the result of test is applied to an I/P logic circuit.

The content of CAR is incremented and applied to one of multiplexer inputs and to over out register SBR. The CD (Condition) field of micro instruction selects one of the status bits in the second

multiplexer. If the bit selected is '1' the 'T' (test) variable is 1, also, it is 0. The 'T' value together with the 2 bits from the BR (branch) field go to an input logic circuit. The I/O logic circuit has 3 inputs I_0 , I_1 , and T and 3 O/P S_0 , S_1 , and L. variables S_1 and S_0 select one of the source addresses for CAR. Variable 'L' enables the load I/P in SBR. The binary value of 2 selection variables determine the path in MUX.

BR field	Input			MUXI		Load SBR
	I_1	I_0	T	S_1	S_0	L
0 0	0	0	0	0	0	0
0 0	0	0	1	0	1	0
0 1	0	1	0	0	0	0
0 1	0	1	1	0	1	1
1 0	1	0	x	1	0	0
1 1	1	1	x	1	1	0

Note that incrementer circuit in sequence is not a counter constructed with flip-flops, but rather a combinational circuit constructed with gates. A combinational circuit incrementer can be designed by cascading a series of half adder circuits. The I/O carry from one stage must be applied to I/O of next stage one I/O in 1st Last significant stage must be equal to 1 to provide the increment by one operation.

Q 25. Role of microprogrammed control over hardwired control. (PTU, Dec. 2010)

Ans. 1. Control Memory (CM) : The set of micro instructions is stored in control memory also called the control store.

2. Control Address Register : It contains the address of next micro instructions to be read. This is similar to program counter which stores the address of next instruction.

3. Control Buffer Register : When a micro instruction is read from control memory, it is transferred to control buffer register.

4. Sequencing Logic : The fourth element is a sequencing unit that loads the control address register that with the address of the next instruction to be read and issues a read command to control memory.

Q 26. Explain and show diagrammatically how address sequencing is done in microprogrammed control unit. (PTU, Dec. 2013, 2011)

Ans. A microprogram consist of a sequence of micro-instructions in a microprogramming. The ideal of microprogrammed Control Unit is that the Control Unit design must include the logics for sequencing through micro-operations, for executing micro-operation, for executing micro-instructions, for interrupting opcodes and for making decision based on ALU flags. So the design is relatively inflexible. It is difficult to change the design if one wishes to add a new machine instruction.

The principal disadvantage of a microprogrammed control unit is that it will be slower than hardwired unit of comparable technology. Despite this, microprogramming is the dominant technique for implementing control unit in the contemporary CISC processor, due to its ease of implementation.

The control unit operates by performing consecutive control storage reads to generate the next set of control function outputs. Performing the series of control memory accesses is, in effect, executing a program for each instruction in the machine's instruction set – hence the term microprogramming.

The two basic tasks performed by a microprogrammed control unit are as follows :

□ **Micro-instruction sequencing :** The microprogrammed control unit get the next micro-instruction from the control memory.

- **Micro-instruction execution** : The microprogrammed control unit generate the control signals needed to execute the micro-instruction.

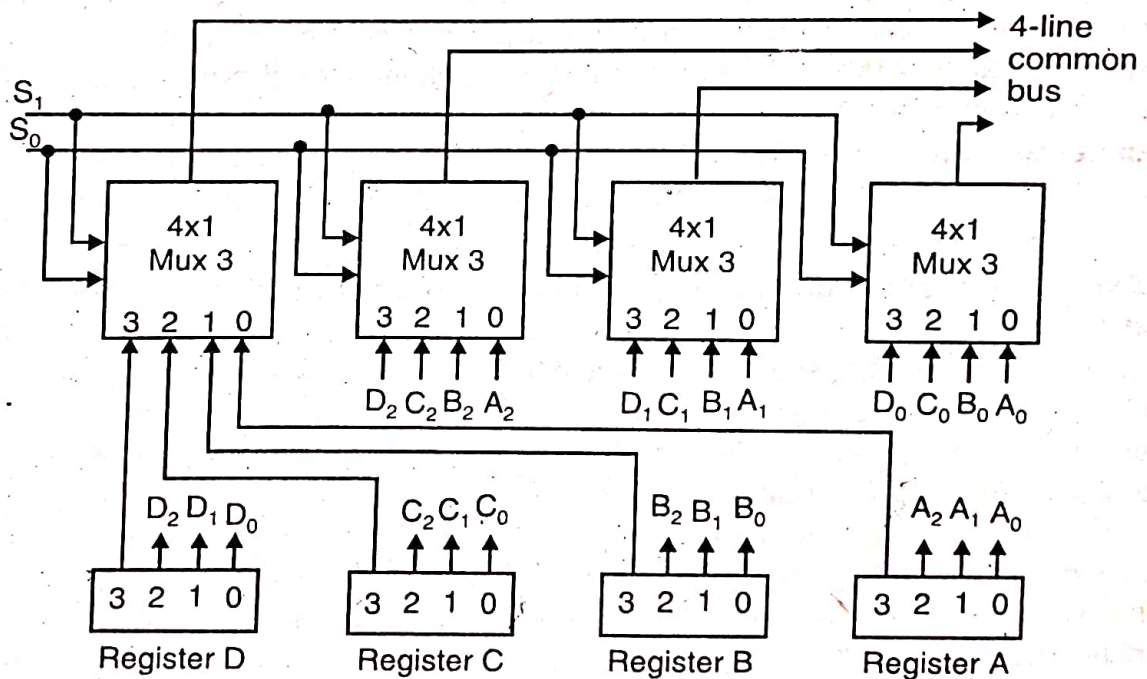
The control unit design must consider both affect the format of the micro-instruction and the timing of the control unit.

Q 27. What is meant by hierarchical bus system for multiprocessing systems ?

(PTU, May 2006)

Ans. Hierarchical Bus System for Multiprocessing Systems : A typical digital computer has many registers, and paths must be provided to transfer information from one register to another. The number of wires will be excessive if separate lines are used between each register and all other registers in the system. A more efficient scheme for transferring information between register configuration is a common bus system. A bus structure consists of a set of common lines, one for each bit of a register, which binary information is transferred one at a time. Control signals determine which register is selected by the bus during each particular register transfer.

One way of constructing a common bus system is with multiplexers. The multiplexers select the source register whose binary information is then placed on the bus. The construction of a bus system for four registers is shown in Fig. Each register has four bits, numbered 0 through 3. The bus consists of four 4×1 multiplexers each having four data inputs, 0 through 3, and two selection inputs, S_1 and S_0 . In order not to complicate the diagram with 16 lines crossing each other, we use labels to show the connections from the outputs of the registers to the inputs of the multiplexers. For example, output 1 of register A is connected to input 0 of MUX 1 because this input is labeled A_1 . The diagram shows that the bits in the same significant position in each register are connected to the inputs of one multiplexer to form one line of the bus. Thus MUX 0 multiplexes the four 0 bits of the registers, MUX 1 multiplexes the four 1 bits of the registers, and similarly for the other two bits.



Bus system for four registers

The two selection lines S_1 and S_0 are connected to the selection inputs of all four multiplexers. The selection lines choose the four bits of one register and transfer them into four-line common bus. When $S_1 S_0 = 00$, the 0 data inputs of all four multiplexers are selected and applied to the outputs

that from the bus. This causes the bus lines to receive the content of register A since the outputs of this register are connected to the 0 data inputs of the multiplexers. Similarly, register B is selected if $S_1S_0 = 01$, and so on. Table shows the register that is selected by the bus for each of the four possible binary value of the selection lines.

TABLE : Function Table for Bus of Fig.

S_1	S_0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

In general, a bus system will multiplex k registers of n bits each to produce an n -line common bus. The number of multiplexers needed to construct the bus is equal to n , the number of bits in each register. The size of each multiplexer must be $k \times 1$ since it multiplexes k data lines. For example, common bus for eight registers of 16 bits each requires 16 multiplexers, one for each line in the bus. Each multiplexer must have eight data input lines and three selection lines to multiplex one significant bit in the eight registers.

The transfer of information from a bus into one of many destination registers can be accomplished by connecting the bus lines to the inputs of all destination registers and activating the load control of the particular destination register selected. The symbolic statement for a bus transfer may mention the bus or its presence may be implied in the statement. When the bus is included in the statement, the register transfer is symbolized as follows :

$$\text{BUS} \leftarrow C, \quad R1 \leftarrow \text{BUS}$$

The content of register C is placed on the bus, and the content of the bus is loaded into register R1 by activating its load control input. If the bus is known to exist in the system, it may be convenient just to show the direct transfer.

$$R1 \leftarrow C$$

From this statement the designer knows which control signals must be activated to produce the transfer through the bus.

Q 28. What are system attributes to performance ? How do we calculate MIPS rate and throughput rate ? (PTU, May 2007)

Ans. System attributes to performance : In order to compare performance of two machines with different instruction sets, and even different styles of instruction sets (e.g. RISC vs, CISC), we can break the total execution time into constituent parts. The total time (T) to execute any given program is the product of the number of machine cycles (n_C) required to execute the program and the processor cycle time (t_C) :

$$T = n_C \times t_C$$

The number of cycles executed can be rewritten as the number of instructions executed (n_i) times the average number of cycles per instruction (n_C / n_i) :

$$T = n_i \times (n_C / n_i) \times t_C$$

The middle factor in this expression describes the average number of machine cycles the processor devotes to each instruction. It is the number of cycles per instruction or CPI. The basic performance model for a single processor computer system is thus,

$$T = n_i \times \text{CPI} \times t_C \quad \dots(1)$$

Where $\text{CPI} = (n_C / n_i)$.

The above three factors each describe different attributes of the execution of a program.

The first factor i.e. number of instructions depends on the algorithm, the compiler, and to some extent the instruction set of the machine. Total execution time can be reduced by lowering the instruction count, either through a better algorithm (one that executes an inner loop fewer times, for example), a better compiler (one that generates fewer instructions for the body of the loop), or perhaps by changing the instruction set so it requires fewer instructions to encode the same algorithm.

The second factor in the performance model is CPI. At first it would seem this factor is simply a measure of the complexity of the instruction set: simple instructions require fewer cycles, so RISC machines should have lower CPI values. That view is misleading, however, since it concerns a static quantity. The performance equation describes the average number of cycles per instruction measured during the execution of a program. The difference is crucial. Implementation techniques such as pipelining allow a processor to overlap instructions by working on several instructions at one time. These techniques will lower CPI & improve performance since more instructions are executed in given time period.

The third factor in the performance model is the processor cycle time t_C . This is usually in the realm of computer engineering: a better layout of the components on the surface of the chip might shorten wire lengths and allow for a faster clock, or a different material (e.g. gallium arsenide vs. silicon based semiconductors) might have a faster switching time. However, the architecture can also affect cycle time. One of the reasons RISC is such a good fit for current VLSI technology is that if the instruction set is small, it requires less logic to implement. Less logic means less space on the chip, and smaller circuits run faster and consume less power. Thus the design of the instruction set, the organization of pipelines, and other attributes of the architecture and its implementation can impact cycle time.

The execution of an instruction requires going through a cycle of events involving the instruction fetch, decode, operand (s) fetch, execution, and store results. In this cycle, only the instruction decode and execution phases are carried out in the CPU. The remaining three operations may be required to access the memory. We define a memory cycle as the time needed to complete one memory reference. Usually, memory cycle is k times the processor cycle t_C . The value of k depends on the speed of the memory technology and processor-memory interconnection scheme used.

The CPI of an instruction type can be divided into two component terms (1) the total processor cycles and (2) memory cycles needed to complete the execution of the instruction. Depending on the instruction type, complete instruction cycle may involve one to four memory references (one of instruction fetch, two for operand fetch, & one for store results). Therefore we can rewrite Eq. 1 as follows

$$T = n_i \times (p + m + k) \times t_C \quad \dots(2)$$

where p is the number of processor cycles needed for the instruction decode and execution, m is the number of memory references needed, k is the ratio between memory cycle and processor cycle, n_i is the instruction count, and t_C is the processor cycle time. Equation 2 can be further refined once the CPI components (p, m, k) are weighted over the entire instruction set.

MIPS RATE

Let C be the total number of clock cycles needed to execute a given program. Then the CPU time in Eq. 2 can be estimated as $T = C \times t_C = C / f$. Furthermore, $CPI = C / n_i$ and $T = n_i \times CPI \times t_C = n_i \times CPI / f$.

The processor speed is often measured in terms of million instructions per second (MIPS). We simply call it the MIPS rate of a given processor. It should be emphasized that the MIPS rate varies with respect to a number of factors, including the clock rate (f), the instruction count (n_i), and the CPI of a given machine, as defined below:

$$\text{MIPS rate} = \frac{n_i}{T \times 10^6} = \frac{f}{\text{CPI} \times 10^6} = \frac{f \times n_i}{C \times 10^6} \quad \dots(3)$$

THROUGHPUT RATE

Another important concept is related to how many programs a system can execute per unit time, called the system throughput W_s (in programs/second). In a multi-programmed system, the system throughput is often lower than the CPU throughput W_p defined by:

$$W_p = \frac{f}{n_i \times \text{CPI}}$$

From eq. 3 Note that

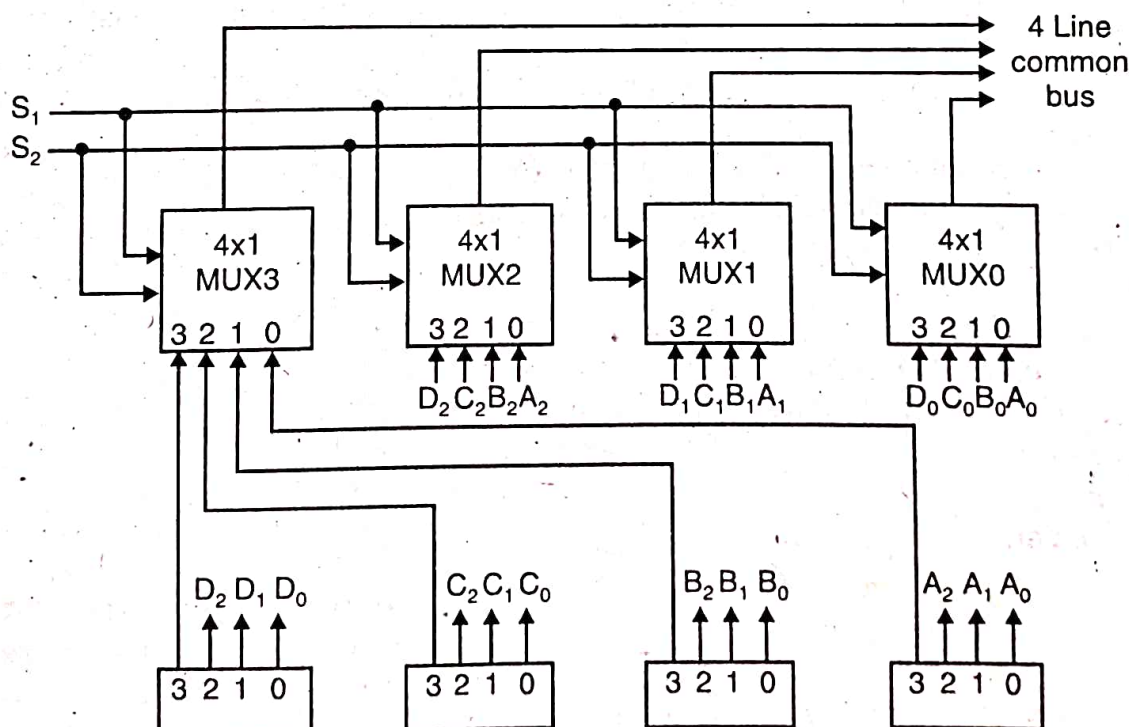
$$W_p = \frac{\text{MIPS} \times 10^6}{n_i}$$

The unit for W_p is programs/second. The CPU throughput is a measure of how many programs can be executed per second, based on the MIPS rate and average program length (n_i). The reason why $W_s < W_p$ is due to the additional system overheads caused by the I/O, compiler, and OS when multiple programs are interleaved for CPU execution by multiprogramming or timesharing operations. If the CPU is kept busy in a perfect program-interleaving fashion, then $W_s < W_p$. This will probably never happen, since the system overhead often causes an extra delay and the CPU may be left idle for some cycles.

Q 29. Which is the elements considered in Bus Design?

(PTU, Dec. 2007)

Ans. Bus Design : A digital computer has many registers and paths must be provided to transfer information from one register to another. The number of wires will be excessive if separate lines are used between each register. A more efficient scheme for transferring information between registers in the system. A bus structure consists of a set of common lines, one for each bit of a register through which binary information is transferred one at a time. The control signals determine which register is selected by the bus during each particular register transfer.



One optimal way of constructing a common bus system is with multiplexers. The multiplexers select source register whose binary information is then placed on bus. The construction of a bus system is shown in above figure. Each register has four bits numbered 0 through 3. The bus consists of 4X1 multiplexers each having four data input 0 to 3 and two selection S0 and S1. The diagram is equipped with 16 data lines crossing each other. For example output 1 of register A is connected with input 0 MUX1 because this input is labelled A1. The bits in the significant position in each register are connected to data inputs of one multiplexer to form one line of the bus. This MUX 0 multiplexes four 0 bits of register, MUX1 multiplexes the four 1 bits of register, similarly for other two bits.

S1	S0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

Q 30. What is micro-programming ? What are the advantages and disadvantages of micro programming ? (PTU, Dec. 2015, 2014 ; May 2004)

Ans. Control words can be programmed to perform various operations on the components of the system. A control unit whose binary control variables are stored in memory is called a microprogrammed control unit. Each word in control memory contains within it a micro instruction, the microinstructions specifies one or more micro operations for the system. A sequence of micro instructions constitutes a microprogram and the process is known as microprogramming. Since alteration of microprogram are not needed are the control unit is in operation, the control memory can be read only memory (ROM). In microprogrammed organization, control information is stored in a controlled memory. The control memory is programmed to initiate required sequence of microoperation, Here any required changes or modifications can be done by updating the microprogram in control memory.

Advantages of microprogramming :

1. Great sophistication in the user instruction set can be achieved for relatively low cost. Adding new instructions is cheap.
2. For specialized applications (e.g. real-time systems), critical loops can be microprogrammed for faster execution time.
3. Multiple user instruction sets can be available on the same machine. This allows a new machine to emulate a previous model to aid in the conversion process.
4. Simplifies design of control unit
 - Cheaper to design
 - Less error-prone
 - Much easier to modify
 - Supports having multiple versions/models.

Disadvantage :

- Slower
- More expensive to produce in quantities.

Q 31. What is control memory?

(PTU, May 2017 ; Dec. 2019, 2011)

Ans. Control memory is a random access memory (RAM) consisting of addressable storage registers. It is primarily used in mini and mainframe computers. It is used as a temporary storage for data. Access to control memory data requires less time than to main memory; this speeds up CPU

operation by reducing the number of memory references for data storage and retrieval. Access is performed as part of a control section sequence while the master clock oscillator is running.

Q 32. What is Cache Memory?

(PTU, May 2014)

Ans. Cache memory is a small, high-speed RAM buffer located between the CPU and main memory. Cache memory buffers or holds a copy of the instructions (instruction cache) or data (operand or data cache) currently being used by the CPU. The instructions and data are copies of those in main memory. Cache memory provides two benefits. One, the average access time for CPU's memory requests is reduced, increasing the CPU's speed by providing rapid access to currently used instructions and data. Two, the CPU's use of the available memory bandwidth is reduced. This allows other devices on the system bus to use the memory without interfering with the CPU.

Q 33. What are the properties of cache memory?

(PTU, May 2014)

Ans. The various properties of cache memory are as follow :

- (a) A buffered memory or cache memory consists of a small high-speed memory with main memory information. This information may be addresses, data, or instructions. The speed of the small memory is usually on the order of one magnitude faster than main memory, and its capacity is typically one or two orders of magnitude less than main memory.
- (b) A cache memory system requires an **identifier** or **tag store** to indicate which entries of main memory have been copied into it. Such an area is usually referred to as the directory or tag store.
- (c) A cache memory requires a logical network and method of replacing old entries.
- (d) A cache memory uses timing and control.

Q 34. What is hardwired control?

Ans. Hardwired control units are implemented through use of sequential logic units, featuring a finite number of gates that can generate specific results based on the instructions that were used to invoke those responses. Hardwired control units are generally faster than micro-programmed designs.

Q 35. Explain the function of Control Unit ?

Ans. The control unit implements the architecture of the CPU. It performs the tasks of fetching, decoding, managing execution and then storing results. It may manage the translation of instructions to micro-instructions and manage scheduling the micro-instructions between the various execution units. On some processors the control unit may be further broken down into other units, such as a scheduling unit to handle scheduling and a retirement unit to deal with results coming from the pipeline.

Q 36. What is micro-instruction?

Ans. An instruction that controls data flow and instruction-execution sequencing in a processor at a more fundamental level than machine instructions.

Q 37. What is Microcode?

Ans. **Microcode** is a layer of hardware-level instructions or data structures involved in the implementation of higher level machine code instructions in many computers and other processors; it resides in special high-speed memory and translates machine instructions into sequences of detailed circuit-level operations. It helps separate the machine instructions from the underlying electronics so that instructions can be designed and altered more freely. It also makes it feasible to build complex multi-step instructions while still reducing the complexity of the electronic circuitry compared to other methods. Writing microcode is often called microprogramming and the microcode in a particular processor implementation is sometimes called a microprogram.

Q 38. Explain Micro-programmed Control Unit in detail.

(PTU, May 2016)

Ans. The controller causes instructions to be executed by issuing a specific set of control

signals at each beat of the system clock. Each set of control signals issued causes one basic operation (micro-operation), such as a register transfer, to occur within the data path section of the computer. In the case of a hard-wired control unit the control matrix is responsible for sending out the required sequence of signals.

An alternative way of generating the control signals is that of micro-programmed control. In order to understand this method it is convenient to think of sets of control signals that cause specific micro-operations to occur as being "microinstructions" that could be stored in a memory. Each bit of a microinstruction might correspond to one control signal. If the bit is set it means that the control signal will be active; if cleared the signal will be inactive. Sequences of microinstructions could be stored in an internal "control" memory. Execution of a machine language instruction could then be caused by fetching the proper sequence of microinstructions from the control memory and sending them out to the data path section of the computer. A sequence of microinstructions that implements an instruction on the external computer is known as a micro-routine. The instruction set of the computer is thus determined by the set of micro-routines, the "microprogram," stored in the controller's memory. The control unit of a microprogram-controlled computer is essentially a computer within a computer.

Q 39. What is microprogram sequencing?

Ans. The basic component of a microprogram control unit are the control memory and the circuit that select the next address. The address selection part is called a microprogram sequencer. The purpose of the microprogram sequencer is to present an address to the control memory so that a micro instruction may be read and executed.

Q 40. Compare Microprogrammed and hardwired control organization.

Ans. In hardwired organization, the control logic is implemented with gates, flip-flops, decoders and other digital circuits. It can be optimized to produce fast mode of operation. In the microprogrammed organization, the control information is stored in the control memory. The control memory is programmed to initiate the required set of microoperations. A hardwired control, requires changes in the wiring among the various components if the design has to be modified or changed. In the microprogrammed control, any required changes or modifications can be done by updating the microprogram in the control memory.

Q 41. Compare Register reference and memory reference instructions.

Ans. A register reference instruction is recognized by the operation code 111 with a 0 in the leftmost bit of the instruction. An operand from the memory is not needed, so other 12 bits are used to specify the instruction to be executed. A memory reference instruction uses 12 bits to specify an address and 1 bit for addressing mode I . $I=0$ for direct address and $I=1$ for indirect address. If the three opcode bits in position 12 though 14 are not equal to 111, the instruction is a memory reference type and the bit in 15 is taken as the addressing mode I . If the 3-bit opcode is taken as 111, control then checks the bit in position 15 and if it is 0, it is a register reference type.

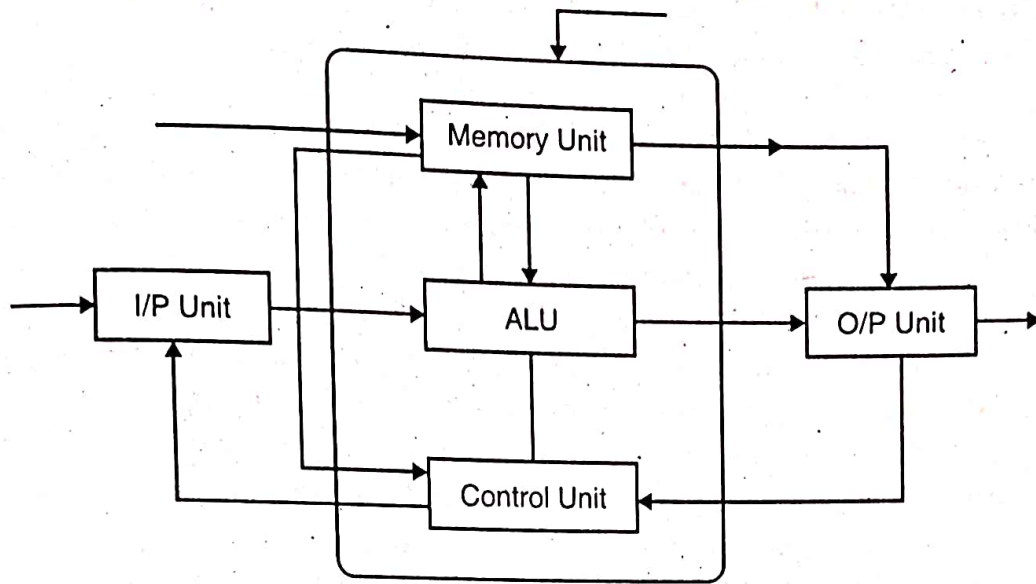
Q 42. What is the need of control unit in computer? Draw the control unit of a basic computer. Discuss how fetch and decode phases are carried out.

Ans. The control unit is one of the most important parts of a microprocessor for the reason that it is in charge of the entire process, that is the machine cycle. The CPU deals with each instruction it is given in a series of steps. Each step is repeated for each instruction. This series of steps is called the machine cycle. It involves :

- Fetching an instruction from memory ;
- Decoding the instruction ;
- Transferring the data ;
- Executing the instruction.

(PTU, Dec. 2016 ; May 2012)

The control unit makes sure that all of those actions are carried out. It also manages all the other components on the CPU.



Input Unit : Computers need to receive data and instruction in order to solve any problem. Therefore we need to input the data and instructions into the computers. The input unit consists of one or more input devices. Keyboard is the one of the most commonly used input device. Other commonly used input devices are the mouse, floppy disk drive, magnetic tape, etc. All the input devices perform the following functions.

- Accept the data and instructions from the outside world.
- Convert it to a form that the computer can understand.
- Supply the converted data to the computer system for further processing.

Storage Unit : The storage unit of the computer holds data and instructions that are entered through the input unit, before they are processed. It preserves the intermediate and final results before these are sent to the output devices. It also saves the data for the later use. The various storage devices of a computer system are divided into two categories :

1. Primary Storage : Stores and provides very fast. This memory is generally used to hold the program being currently executed in the computer, the data being received from the input unit, the intermediate and final results of the program. The primary memory is temporary in nature. The data is lost, when the computer is switched off. In order to store the data permanently, the data has to be transferred to the secondary memory. The cost of the primary storage is more compared to the secondary storage. Therefore most computers have limited primary storage capacity.

2. Secondary Storage : Secondary storage is used like an archive. It stores several programs, documents, data bases etc. The programs that you run on the computer are first transferred to the primary memory before it is actually run. Whenever, the results are saved, again they get stored in the secondary memory. The secondary memory is slower and cheaper than the primary memory. Some of the commonly used secondary memory devices are Hard disk, CD, etc.

Memory Size : All digital computers use the binary system, i.e. 0's and 1's. Each character or a number is represented by an 8 bit code.

The set of 8 bits is called a byte.

A character occupies 1 byte space.

A numeric occupies 2 byte space.

Byte is the space occupied in the memory.

The size of the primary storage is specified in KB (Kilobytes) or MB (Megabyte). One KB is equal to 1024 bytes and one MB is equal to 1000 KB. The size of the primary storage in a typical PC usually starts at 16MB. PCs having 32 MB, 48 MB, 128 MB, 256 MB memory are quite common.

Output Unit : The output unit of a computer provides the information and results of a computation to outside world. Printers, Visual Display Unit (VDU) are the commonly used output devices. Other commonly used output devices are floppy disk drive, hard disk drive, and magnetic tape drive.

Arithmetic Logical Unit : All calculations are performed in the Arithmetic Logic Unit (ALU) of the computer. It also does comparison and takes decision. The ALU can perform basic operations such as addition, subtraction, multiplication, division etc. and does logic operations viz, >, <, =, 'etc. Whenever calculations are required, the control unit transfers the data from storage unit to ALU once the computations are done, the results are transferred to the storage unit by the control unit and then it is send to the output unit for displaying results.

Control Unit : It controls all other units in the computer. The control unit instructs the input unit, where to store the data after receiving it from the user. It controls the flow of data and instructions from the storage unit to ALU. It also controls the flow of results from the ALU to the storage unit. The control unit is generally referred as the central nervous system of the computer that control and synchronizes its working.

Central Processing Unit : The control unit and ALU of the computer are together known as the Central Processing Unit (CPU). The CPU is like brain performs the following functions :

- It performs all calculations.
- It takes all decisions.
- It controls all units of the computer.

A PC may have CPU-IC such as Intel 8088, 80286, 80386, 80486, Celeron, Pentium, Pentium Pro, Pentium II, Pentium III, Pentium IV, Dual Core, and AMD etc.

Q 43. List four peripheral devices that produce an acceptable output for a person to understand. (PTU, May 2011)

Ans. 1. Monitor 2. Printer 3. Speaker 4. Plotter

Q 44. What is the difference between micro program and micro code?

(PTU, Dec. 2016, 2009, 2005 ; May 2005)

Ans. Microcode is a series of microoperations which widely control the computers central processing unit at a very fundamental level.

Micro-program : Each word in control memory contains within it a microinstruction. The micro instruction specifies one or more microoperations for the system. A sequence of microinstructions for the system. A sequence of microinstruction constitutes a microprogram.

Q 45. What is a stack ? List its applications .

(PTU, Dec. 2017 ; May 2004)

Ans. A stack is a linear list data structure that permits insertion (Push operation) and deletion (Pop operation) of elements only at the one end of the list. It is a LIFO structure. Some of the applications of stacks : Conversions to postfix or prefix notations; in the organization of computers; Implementing computer solutions to certain problems etc.

Q 46. Give two examples of program control instructions.

(PTU, May 2005)

Ans. Two Program control instructions.

1. Call
2. Jump

Q 47. What is the role of Shift Registers in digital computers ?

(PTU, Dec. 2009, 2005)

Ans. Shift Register not only store the data bits but also shift the data bits. They are used in digital computers for time delays. counters, parallel or serial data storage.

Q 48. Explain about RISC processors.

(PTU, Dec. 2006)

OR

Write any six characteristics of RISC.

(PTU, May 2009, 2006)

Ans. RISC processors has following characteristics :

1. Relatively few Instructions.
2. Relatively few addressing modes.
3. Memory acces limited to load and store Instructions.
4. All operations done within the Registers of the CPU.
5. Fixed length, easily decoded instruction format.
6. Single-Cycle instruction execution.
7. Hardwired rather than microprogrammed control.

Q 49. What do you mean by instruction parallism ?

(PTU, May 2007)

Ans. An Instruction parallel refers to the degree. On average how many instructions can be executed in parallel in various pipeline stages.

Q 50. Differentiate between RISC and CISC processors.

(PTU, Dec. 2015, 2013, 2010, 2007 ; May 2007)

Ans. The main difference between RISC and CISC processors are as follow :

RISC	CISC
1. Instruction with fixed format and almost register based Instruction.	1. Instructions with variable formats.
2. No. of Instructions less than 100.	2. No. of Instructions 120 to 350.
3. Addressing Modes 12-24	3. Addressing modes limited to 3-5
4. Mostly microcoded using control memory but modern CISC also uses hardwired control.	4. CPU control mostly hardwired without control memory.

Q 51. Define the terms : Seek time, Rotational Delay, Access time.

(PTU, Dec. 2014, 2007)

Ans. **Seek time** is one of the three delays associated with reading or writing data on a computer's disk drive, and somewhat similar for CD or DVD drives. The others are rotational delay and transfer time, and their sum is access time. In order to read or write data in a particular place on the disk, the read/write head of the disk needs to be physically moved to the correct place. This process is known as seeking, and the time it takes for the head to move to the right place is the seek time. Seek time for a given disk varies depending on how far the head's destination is from its origin at the time of each read or write instruction ; usually one discusses a disk's average seek time.

It is the amount of time between when the CPU requests a file and when the first byte of the file is sent to CPU. Times between 10 to 20 milliseconds are common.

Spindle RPM

Average latency (ms)

4200

7.14

5400

5.55

Rotational delay is one of the three delays associated with reading or writing data on a computer's disk drive, and somewhat similar for CD or DVD drives. The others are seek time and transfer time,

Q 47. What is the role of Shift Registers in digital computers ?

Ans. Shift Register not only store the data bits but also shift the data bits. They are used in digital computers for time delays. counters, parallel or serial data storage. (PTU, Dec. 2009, 2005)

Q 48. Explain about RISC processors.

OR

Write any six characteristics of RISC.

Ans. RISC processors has following characteristics :

1. Relatively few Instructions.
2. Relatively few addressing modes.
3. Memory acces limited to load and store Instructions.
4. All operations done within the Registers of the CPU.
5. Fixed length, easily decoded instruction format.
6. Single-Cycle instruction execution.
7. Hardwired rather than microprogrammed control.

(PTU, May 2009, 2006)

Q 49. What do you mean by instruction parallism ?

(PTU, May 2007)

Ans. An Instruction parallel refers to the degree. On average how many instructions can be executed in parallel in various pipeline stages.

Q 50. Differentiate between RISC and CISC processors.

(PTU, Dec. 2015, 2013, 2010, 2007 ; May 2007)

Ans. The main difference between RISC and CISC processors are as follow :

RISC	CISC
1. Instruction with fixed format and almost register based Instruction.	1. Instructions with variable formats.
2. No. of Instructions less than 100.	2. No. of Instructions 120 to 350.
3. Addressing Modes 12-24	3. Addressing modes limited to 3-5
4. Mostly microcoded using control memory but modern CISC also uses hardwired control.	4. CPU control mostly hardwared without control memory.

Q 51. Define the terms : Seek time, Rotational Delay, Access time.

(PTU, Dec. 2014, 2007)

Ans. Seek time is one of the three delays associated with reading or writing data on a computer's disk drive, and somewhat similar for CD or DVD drives. The others are rotational delay and transfer time, and their sum is access time. In order to read or write data in a particular place on the disk, the read/write head of the disk needs to be physically moved to the correct place. This process is known as seeking, and the time it takes for the head to move to the right place is the seek time. Seek time for a given disk varies depending on how far the head's destination is from its origin at the time of each read or write instruction ; usually one discusses a disk's average seek time.

It is the amount of time between when the CPU requests a file and when the first byte of the file is sent to CPU. Times between 10 to 20 milliseconds are common.

Spindle RPM

Average latency (ms)

4200

7.14

5400

5.55

Rotational delay is one of the three delays associated with reading or writing data on a computer's disk drive, and somewhat similar for CD or DVD drives. The others are seek time and transfer time,

and their sum is access time. The term applies to rotating storage devices (such as a hard disk or floppy disk drive, and to the older drum memory systems). The rotational delay is the time required for the addressed area of the disk (or drum) to rotate into a position where it is accessible by the read/write head.

Maximum rotational delay is the time it takes to do a full rotation (as the relevant part of the disk may have just passed the head when the request arrived). Most rotating storage devices rotate at a constant angular rate (constant number of revolutions per second). The maximum rotational delay is simply the reciprocal of the rotational speed (appropriately scaled). In 2001, 7200 revolutions per minute is typical for a hard disk drive; its maximum rotational delay will be $60/7200$ s or about 8 ms.

Average rotational delay is also a useful concept - it is half the maximum rotational delay.

Access time is the time delay or latency between a request to an electronic system, and the access being completed or the requested data returned.

- In a telecommunications system, access time is the delay between the start of an access attempt and successful access. Access time values are measured only on access attempts that result in successful access.
- In a computer, it is the time interval between the instant at which an instruction control unit initiates a call for data or a request to store data, and the instant at which delivery of the data is completed or the storage is started.
- In disk drives, **disk access time** is the time required for a computer to process data from a storage device, such as a hard drive. For hard drives, disk access time is determined by a sum of the spin-up time, seek time, rotational delay and transfer time.
 - Spin-in time - is the time required to accelerate the disk to operating speed. Frequently used drives are often left spinning to improve access time, but drives may be spun down to reduce energy use or noise.
 - Seek time - is the time for the access arm to reach the desired disk track.
 - Rotational delay - the delay for the rotation of the disk to bring the required disk sector under the read-write mechanism. It greatly depends on rotational speed of a disk, measured in revolutions per minute (RPM).
 - Transfer time - time during which data is actually read or written to medium, with a certain throughput.

Theoretical averages of the access time or latency are shown in the table below, based on the empirical relation that the average latency in milliseconds for such a drive is about 30000/RPM.

Q 52. What is bus system?

(PTU, May 2015)

Ans. A group of wires through which binary information is transferred one at a time among register is called a bus. A common bus system can be constructed by using multiplexers, source registers, decoders and destination registers.

Q 53. Explain program control.

Ans. Program control instructions specify condition for altering the content of the program counter. On the other hand, a program control type of instruction when executed may change the address value in the program counter and cause the flow of control to be altered.

Q 54. Explain the concept of program interrupt.

Ans. The concept of program interrupt is used to handle a variety of problems that arise out of normal program sequence. Program interrupt refer to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request. Control returns to the original program after the service program is executed.

Q 55. Discuss cost/benefit concept of layers in architecture design. (PTU, Dec. 2010)

Ans. The cost/benefit analysis method in an architecture centric method for analyzing the costs, benefits and schedule implication of architectural decisions. It also enables assessment of the uncertainty surrounding judgements of costs and benefits, thereby providing a basis for informed decision making about the architectural design.

Q 56. Differentiate between Programmed I/O and mapped I/O. (PTU, May 2004)

Ans. The statement is incorrect.

However, by taking I/O mapped I/O mean memory mapped I/O, the difference between programmed I/O and memory mapped I/O is that in former the I/O is controlled through program, and in later case I/O is through designated memory locations.

Q 57. Explain Maskable and unmaskable interrupts. (PTU, May 2004)

Ans. Maskable interrupts are hardware interrupts that can be enabled and disabled by software otherwise they are nonmaskable.

Q 58. List various methods of data transfer ? (PTU, May 2004)

Ans. There are many methods of data transfer like serial, parallel, synchronous, asynchronous, programmed, memory mapped, DMA etc.

Q 59. What is the difference between external interrupts and internal interrupts ?

(PTU, Dec. 2004)

Ans. External Interrupt : It is generated by external services like keyboard. It is asynchronous.

Internal Interrupts : It is generated from illegal or erroneous use of an instruction. Internal Interrupts are also called traps. Example of Interrupts caused by internal error condition are register overflow and stack overflow.

Q 60. What do you mean by DMA I/O Concept ? (PTU, Dec. 2004)

Ans . DMA : The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called Direct Memory access.

Q 61. Define the terms I/O processor and I/O controller. (PTU, May 2005)

Ans. I/O Processor : It is a processor which is required to communicate with the system for performing a specific task. For example DMA processor is used to transfer data from processor to I/O.

I/O Controller : It is the module that controls an I/O function.

Q 62. What do you mean by programmed I/O Concept ? (PTU, May 2012, 2005)

Ans. Programmed I/O operation are the result of I/O concept Instructions written in the computer Program. Each data item transfer is initiated by an Instruction in the program. Usually, the transfer is to and from a CPU register and Peripheral. Other Instructions are needed to transfer the data to and from CPU and memory. Transferring data under program control requires constant monitoring of the peripheral by the CPU. Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made.

Q 63. What do you mean by software interrupt ? (PTU, Dec. 2009, 2005)

Ans. Software Interrupt : A Software interrupt is initiated by executing an instruction. Software Interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call.

Q 64. What do you mean by Interrupt-initiated I/O Concept ? (PTU, Dec. 2009, 2005)

Ans. In the programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it keys the processor busy needlessly. It can be avoided by using an Interrupt request signal when the data are

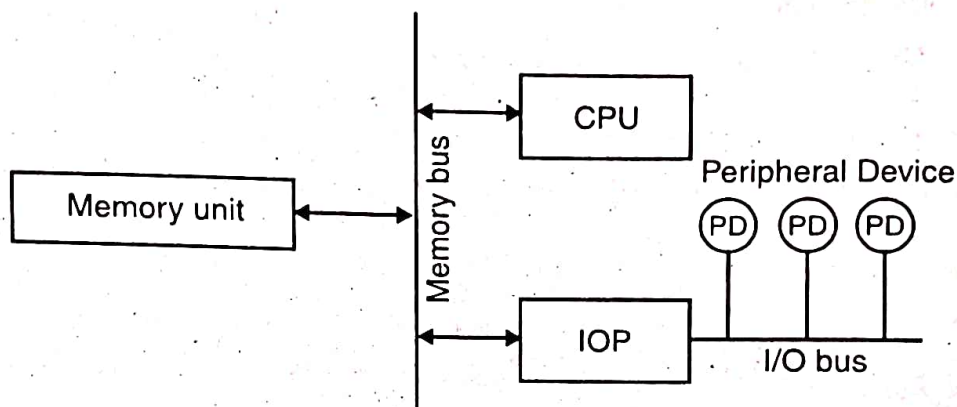
available from device. In the meantime the CPU can proceed to execute another program. The Interface meantime keeps monitoring the device. When the Interface determines that the device is ready for data transfer, it generates an Interrupt request signal to computer. Upon detecting the external interrupt signal, the CPU momentarily stops the tasks at is processing, branches to the service program process the I/O transfer and then returns to the task it was originally performing.

Q 65. Write about DMA Transfer.

Ans. DMA Transfer : DMA transfer is used to move the data from one memory location to another without using the CPU. (PTU, May 2006)

Q 66. Explain about I/O processor.

Ans. An Input-Output processor may be classified as a processor with direct memory access capability that communicates with I/O devices. Each IOP takes care of input and output tasks, relieving the CPU from housekeeping chores involved in I/O transfers. (PTU, Dec. 2011, 2010 ; May 2006)



Q 67. What is channel ?

OR

What do you mean by the term DMA channel?

Ans. Channel : A channel is used to transfer the data from source device to the destination device. (PTU, May 2006)

Q 68. Explain about I/O modes.

Ans. Binary Information recieved from an external device is usually stroed in memory for later Processing Information transferred from the cental computer into an external device originates in the memory unit. The CPU merely executes the I/O Instructions and may accept data temporarily, but the ultimate source or destination is the memory unit. Data transfer between the central computer and I/O devices may be handled in variety of modes. These are :

1. Programmed I/O
2. Interrupt Initiated I/O
3. Direct Memory Access (DMA)

Q 69. What is meant by DMA?

Ans. DMA stands for Direct Memory Access, a capability in modern computers that allows peripheral devices to send data to the motherboard's memory without intervention from the CPU. (PTU, May 2016, 2010 ; Dec. 2017, 2006)

The DMA controllers are special hardware-now embedded into the chip in modern integrated processors-that manage the data transfers and arbitrate access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.

Transfers are initiated when the DMA controller is notified of the need to move data to the

memory by some event (keyboard press or mouse click, for examples). The controller asserts a DMA request signal to the CPU to use the system bus. The CPU completes its current operation and yields control of the bus to the DMA controller via a DMA acknowledge signal. The controller then reads and writes data and controls signals as if it is the CPU, which at that instant is tri-stated (idled). Upon completion of the transfer, DMA controller de-asserts the DMA request signal and the CPU in turn removes its DMA acknowledge signal and resumes control of the bus.

DMA is implemented in computer bus architectures to speed up computer operations and allow multitasking. Normally, the CPU will be fully occupied in any read/write operation, enabling DMA allows reading/writing data in the internal memory, external memory and peripherals without CPU involvement, thus making the processor available for other tasks. This ensures streamlined operations, as movement of data to/from memory is one of the most common computer operations and freeing the CPU of this overhead can lead to a significant improvement in performance.

DMA is useful in real-time computing applications where critical operations must be done concurrently. Stream processing is another application of DMA, where transfer and data processing are done simultaneously. Many hardware systems use DMA including floppy and disk drive controllers, graphics cards.

Q 70. Write characteristics of I/O Channels.

(PTU, May 2012 ; Dec. 2010, 2007)

Ans. I/O Channel : In computer science, **channel I/O** is a generic term that refers to a high-performance input/output (I/O) architecture that is implemented in various forms on a number of computer architectures, especially on mainframe computers. In the past they were generally implemented with a custom processor, known alternately as **peripheral processor, I/O processor, I/O controller, or DMA controller.**

Basic principles and characteristics : Many I/O tasks can be complex and require logic to be applied to the data to convert formats and other similar duties. In these situations, the simplest solution is to ask the CPU to handle the logic, but because I/O devices are relatively slow, a CPU could waste time (in computer perspective) waiting for the data from the device. This situation is called 'I/O bound'.

Channel architecture avoids this problem by using a separate, independent, low-cost processor. Channel processors are simple, but self-contained, with minimal logic and sufficient on-board scratchpad memory (working storage) to handle I/O tasks. They are typically not powerful or flexible enough to be used as a computer on their own and can be construed as a form of subprocessor.

An attached CPU sends small channel programs to the controller to handle I/O tasks, which the channel controller can normally complete without further intervention from the CPU.

When I/O transfer is complete or an error is detected, the channel controller communicates with the CPU using an interrupt. Since the channel controller has direct access to the main memory, it is also often referred to as DMA controller (where DMA stands for direct memory access), although that term is looser in definition and is often applied to non-programmable devices as well.

Description : The reference implementation of channel I/O is that of the IBM System/360 family of mainframes and its successors, but similar implementations have been adopted by other mainframe vendors, such as Control Data, Bull (General Electric/Honeywell) and Unisys.

Computer systems that use channel I/O have special hardware components that handle all input/output operations in their entirety independently of the systems' CPU (s). The CPU of a system that uses channel I/O typically has only one machine instruction in its repertoire for input and output ; this instruction is used to pass input/output commands to the specialized I/O hardware in the form of channel programs. I/O thereafter proceeds without intervention from the CPU until an event requiring notification of the operating system occurs, at which point the I/O hardware signals an interrupt to the CPU.

A channel is an independent hardware component that co-ordinates all I/O to a set of controllers or devices. It is not merely a medium of communication, despite the name ; it is a programmable device that handles all details of I/O after being given a list of I/O operations to carry out (the channel program).

Each channel may support one or more controllers and/or devices. Channel programs contain lists of commands to the channel itself and to various controllers and devices to which it is connected. Once the operating system has prepared a complete list of I/O commands, it executes a single I/O machine instruction to initiate the channel program ; the channel thereafter assumes control of the I/O operations until they are completed.

It is possible to develop very complex channel programs ; initiating many different I/O operations on many different I/O devices simultaneously. This flexibility frees the CPU from the overhead of starting, monitoring and managing individual I/O operations. The specialized channel hardware, in turn, is dedicated to I/O and can carry it out more efficiently than the CPU (and entirely in parallel with the CPU). Channel I/O is not unlike the Direct Memory Access (DMA) of microcomputers, only more complex and advanced. Most mainframe operating systems do not fully exploit all the features of channel I/O.

On large mainframe computer systems, CPUs are only one of several powerful hardware components that work in parallel. Special input/output controllers (the exact names of which vary from one manufacturer to another) handle I/O exclusively, and these in turn are connected to hardware channels that also are dedicated to input and output. There may be several CPUs and several I/O processors. The overall architecture optimizes input/output performance without degrading pure CPU performance. Since most real-world applications of mainframe systems are heavily I/O-intensive business applications, this architecture helps provide the very high levels of throughput that distinguish mainframes from other types of computer.

In IBM ESA/390 terminology, a channel is a parallel data connection inside the tree-like or hierarchically organized I/O subsystem. In System/390 I/O cages, channels either directly connect to devices which are installed inside the cage (communication adapter such as ESCON, FICON, Open Systems Adapter) or they run outside of the cage, below the raised floor as cables of the thickness of a thumb and directly connect to channel interfaces on bigger devices like tape subsystems, direct access storage devices (DASDs), terminal concentrators and other ESA/390 systems.

Q 71. Write major requirement for I/O module.

(PTU, Dec. 2007)

Ans. In computer science, I/O module is a generic term that refers to a high performance input/output (I/O) architecture that is implemented in various form on a number of computer architecture, especially on mainframe computers: The basic requirements for I/O module function are as follows.

1. **Write** : Transfer data from memory to I/O device.
2. **Read** : Transfer data from I/O device to memory.
3. **Read Backwards** : Read magnetic tape with tape moving backward.
4. **Control** : Used to initiate an operation not involving transfer of data such as rewinding of tape or positioning a disk access mechanism.

5. **Sense** : Informs the channel to transfer its channel status word to memory location 64.

6. **Transfer in channel** : Used instead of a jump instruction. Here the data address field specifies the address of next command word to be executed by the channel.

Q 72. What is an I/O processor? Briefly discuss.

(PTU, Dec. 2006)

Ans. In computer science, channel I/O is a generic term that refers to a high-performance input/output (I/O) architecture that is implemented in various forms on a number of computer architectures, especially on mainframe computers. In the past they were generally implemented with

a custom processor, known alternately as **peripheral processor, I/O processor, I/O controller, or DMA controller.**

Many I/O tasks can be complex and require logic to be applied to the data to convert formats and other similar duties. In these situations, the simplest solution is to ask the CPU to handle the logic, but because I/O devices are relatively slow, a CPU could waste time (in computer perspective) waiting for the data from the device. This situation is called 'I/O bound.' Channel architecture avoids this problem by using a separate, independent, low-cost processor. Channel processors are simple, but self-contained, with minimal logic and sufficient on-board scratchpad memory (working storage) to handle I/O tasks. They are typically not powerful or flexible enough to be used as a computer on their own and can be construed as a form of subprocessor.

Q 73. A DMA controller transfers 16-bit words to memory using cycle stealing. The words are assembled from a device that transmits characters at a rate of 2400 characters per second. The CPU is fetching and executing instructions at an average rate of 1 million instructions per second. By how much will the CPU be slowed down because of the DMA transfer ?

(PTU, May 2007 ; Dec. 2009, 2005)

Ans. CPU refers to memory on the average once every micro-seconds ($1/10^6$ Sec).

Characters arrive one every $1/2400 = 416.6$ micro seconds.

Two characters of 8 bits each are packed in to a 16 bit words every $2 \times 416.6 = 833.3$ micro seconds. The CPU is slowed down by no more than $(1/833.3) \times 100 = 0.12 \%$

Q 74. Compare interrupt I/O control with DMA I/O.

(PTU, May 2006)

Ans. Interrupt I/O control with DMA I/O : Binary information received from an external device is usually stored in memory for later processing. Information transferred from the central computer into an external device originates in the memory unit. The CPU merely executes the I/O instructions and may accept the data temporarily, but the ultimate source or destinations is the memory unit. Data transfer between the central computer and I/O devices may be handled in a variety of modes. Some modes use the CPU as an intermediate path; others transfer the data directly to and from the memory unit. Data transfer to and from peripherals may be handle in one of three possible modes:

1. Programmed I/O
2. Interrupt-initiated I/O
3. Direct memory access (DMA)

1. Programmed I/O : Programmed I/O operations are the result of I/O instructions written in the computer program. Each data item transfer is initiated by an instruction in the program. Usually, the transfer is to and from a CPU register and peripheral. Other instructions are needed to transfer the data to and from CPU and memory. Transferring data under program control requires constant monitoring of the peripheral by the CPU. Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made. It is up to the programmed instructions executed in the CPU to keep close tabs on everything that is taking place in the interface unit and the I/O device.

2. Interrupt-initiated I/O : In the programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time-consuming process since it keeps the processor busy needlessly. It can be avoided by interrupt using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device. In the meantime the CPU can proceed to execute another program. The interface meanwhile keeps monitoring the device. When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer. Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program, to process the I/O transfer, and then returns to the task it was originally performing.

3. Direct memory access (DMA) : Transfer of data under programmed I/O is between CPU and peripheral. In direct memory access (DMA), the interface transfers data into and out of the memory unit through the memory bus. The CPU initiates the transfer by supplying the interface with the starting address and the number of words needed to be transferred and then proceeds to execute other tasks. When the transfer is made, the DMA requests memory cycles through the memory bus. When the request is granted by the memory controller, the DMA transfers the data directly into memory. The CPU merely delays its memory access operation to allow the direct memory I/O transfer. Since peripheral speed is usually slower than processor speed, I/O-memory transfers are infrequent compared to processor access to memory. DMA transfer is discussed in more detail in sec.

IOP : Many computers combine the interface logic with the requirements for direct memory access into one unit and call it an I/O processor (IOP). The IOP can handle many peripherals through a DMA and interrupt facility. In such a system, the computer is divided into three separate modules: the memory unit, the CPU, and the IOP. I/O processors are presented in sec.

Q 75. What is the basic function of an interrupt controller?

(PTU, May 2009)

Ans. The interrupt controller provides the means by which I/O devices request attention from the processor to deal with data transfers. It accepts requests from the peripheral equipment, determines which of the incoming requests is of the highest priority, ascertains whether the incoming request has a higher priority value than the level currently being serviced, and issues an interrupt to the CPU based on this determination.

Q 76. What is program interrupt?

(PTU, May 2010)

Ans. External and Internal interrupts are initiated from signals that occur in hardware of CPU. A software interrupt is initiated by executing an instruction. Software interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call. It can be used by the programmer to initiate an interrupt procedure at any desired point in the program. The most common use of software interrupt is associated with a supervisor call instruction. This instruction provides means for switching from CPU user mode to superior mode. Certain operations in computer may be assigned to supervisor mode only, for example, a complex input and output procedure. A program written by user must most run in user mode. When an input or output transfer is required the supervisor mode requested by means of a supervisor call instruction. This instruction causes a software interrupt that stores the old CPU state and bring a new PSW (program status word) that belongs to supervisor mode. The calling program must pass information to operating system in order to specify particular task requires.

Q 77. What do you mean by initialisation of DMA controller? How DMA controller works? Explain with suitable block diagram.

(PTU, May 2018, 2014, 2012, 2008 ; Dec. 2013)

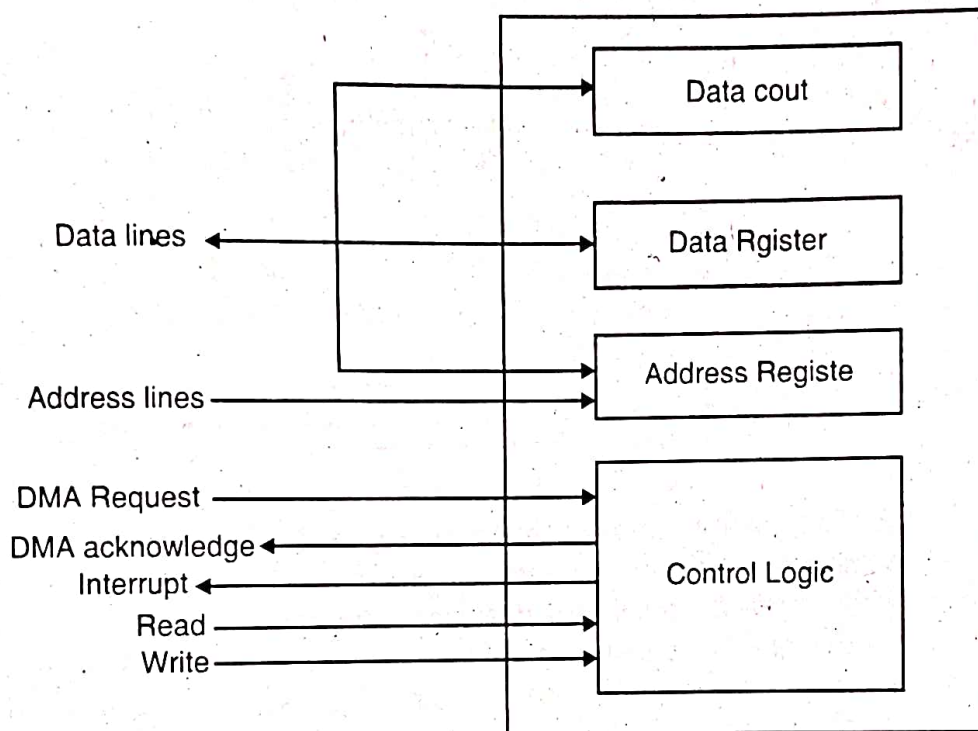
OR

Discuss DMA with help of schematic diagram of controller.

(PTU, Dec. 2015, 2008)

Ans. Direct memory access (DMA) is a feature of modern computers and microprocessors that allows certain hardware subsystems within the computer to access system memory for reading and/or writing independently of the central processing unit. Many hardware systems use DMA including disk drive controllers, graphics cards, network cards, sound cards and GPUs. DMA is also used for intra-chip data transfer in multi-core processors, especially in multiprocessor system-on-chips, where its processing element is equipped with a local memory (often called scratchpad memory) and DMA is used for transferring data between the local memory and the main memory. Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without a DMA channel. Similarly a processing element inside a multi-core processor can transfer data to and from its local memory without occupying its processor time and allowing computation and data transfer concurrency.

Without DMA, using programmed input/output (PIO) mode for communication with peripheral devices, or load/store instructions in the case of multicore chips, the CPU is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work. With DMA, the CPU would initiate the transfer, do other operations while the transfer is in progress, and receive an interrupt from the DMA controller once the operation has been done. This is especially useful in real-time computing applications where not stalling behind concurrent operations is critical. Another and related application area is various forms of stream processing where it is essential to have data processing and transfer in parallel, in order to achieve sufficient throughput.



DMA Block Diagram

The Direct Memory Access (DMA) module handles all data transfers between the computer's peripherals and memory without intervention by the processor.

A DMA module transfers a sector from the hard disk to memory by first accessing the system bus. It may either use the bus when the processor is not engaging it, or it must conduct cycle stealing. Cycle stealing is to force the processor to temporarily suspend its current operation.

While the processor does some other operation not involving use of the system bus, the DMA module transfers the entire sector of data, one word at a time from the hard disk to memory. When the transfer is complete, the DMA module informs the processor by sending an interrupt signal to it.

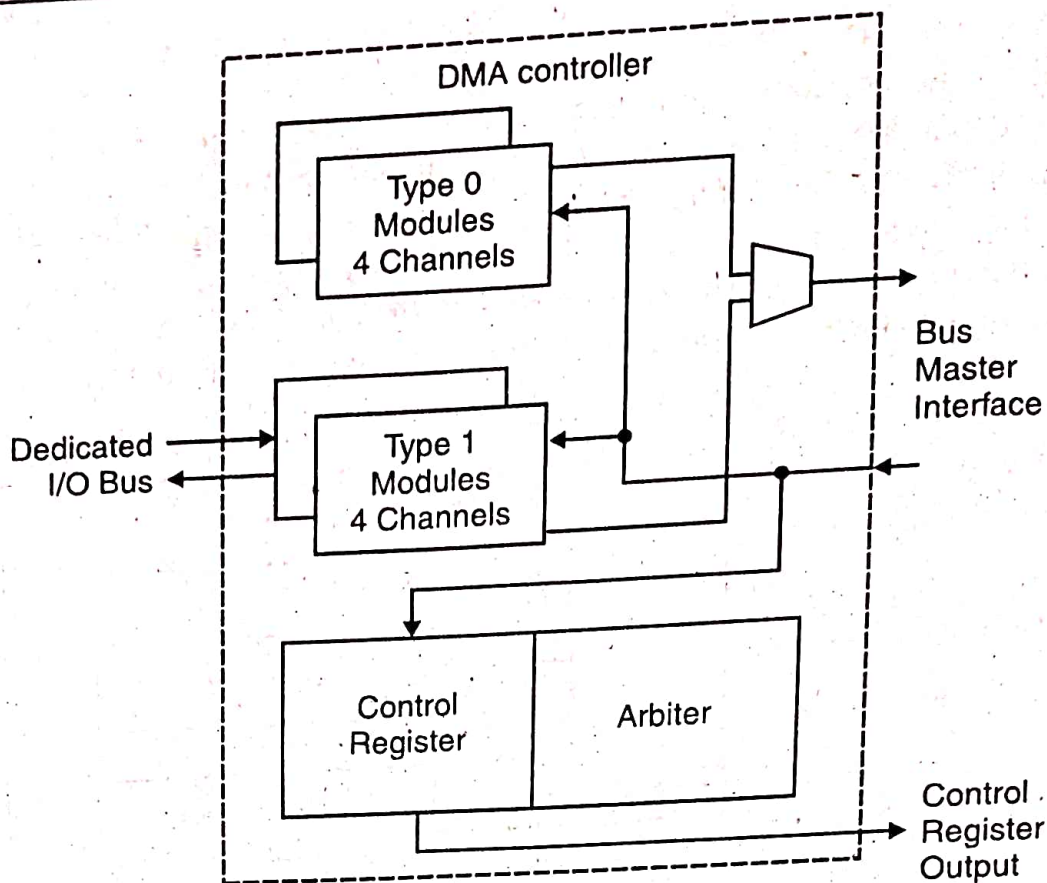
DMA Controller

Features

- General-purpose direct-memory access (DMA) controller.
- Upto 16 DMA channels
- Supports both synchronous and asynchronous DMA transfers
- Designed for peripheral component interconnect (PCI) and other central processing unit (CPU) bus systems.

Block Diagram

Fig. shows the block diagram for the DMA controller



Description : The DMA controller megafunction is designed for data transfer in different system environments. Two module types – type 0 and type 1 – are provided, and the user can choose the number of each module type. Type 0 modules are designed to transfer data residing on the same bus, and Type 1 modules are designed to transfer data between two different buses. Each module can support up to 4 DMA channels ; the megafunction supports up to 16 total DMA channels.

Each DMA channel can be programmed for various features, such as transfer size, synchronized and unsynchronized transfer control, transfer priority, interrupt generation, memory and I/O address space, and address change direction. This megafunction is designed to work with 32-bit and 64-bit bus systems, including the PCI bus, power PC bus, and other CPU host buses. It can also be integrated with other megafunctions to form a complete functional block.

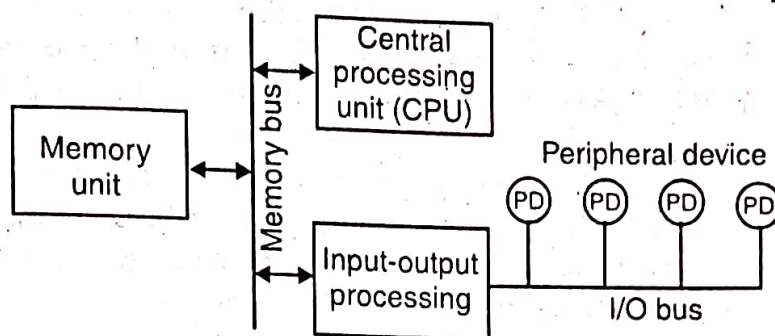
Q 78. Explain about I/O processor/Information Processor.

(PTU, Dec. 2006)

Ans. Input-Output Processor (IOP) : Instead of having each interface communicate with the CPU, a computer may incorporate one or more external processors and assign them the task of communicating directly with all I/O devices. An input-output processor (IOP) may be classified as a processor with direct memory access capability that communication with I/O devices. In this configuration, the computer system can be divided into a memory unit, and a number of processors comprised of the CPU and one or more IOPs. Each IOP takes care of input and output tasks, relieving the CPU from the housekeeping chores involved in I/O transfers. A processor that communicates with remote terminals over telephone and other communication media in a serial fashion is called a data communication processor (DCF).

The IOP is similar to a CPU except that it is designed to handle the details of I/O processing. Unlike the DMA controller that must be set up entirely by the CPU, the IOP can fetch and execute its own instructions. IOP instructions are specifically designed to facilitate I/O transfer. In addition, the IOP can perform other processing tasks, such as arithmetic, logic, branching, and code translation.

The block diagram of a computer with two processors is shown in Fig. The memory unit occupies a central position and can communicate with each processor by means of direct memory access. The CPU is responsible for processing data needed in the solution of computational tasks. The IOP provides a path for transfer of data between various peripheral devices and the memory unit. The CPU is usually assigned the task of initiating the I/O program. From then on the IOP operates independent of the CPU and continues to transfer data from external devices and memory.



Block diagram of a computer with I/O processor

The data formats of peripheral devices differ from memory and CPU data formats. The IOP must structure data words from many different sources. For example, it may be necessary to take four bytes from an input device and pack them into one 32-bit word before the transfer to memory. Data are gathered in the IOP at the device rate and bit capacity while the CPU is executing its own program. After the input data are assembled into a memory word, they are transferred from IOP directly into memory by "stealing" one memory cycle from the CPU. Similarly, an output word transferred from memory to the IOP is directed from the IOP to the output device at the device rate and bit capacity.

The communication between the IOP and the devices attached to it is similar to the program control method of transfer. Communication with the memory is similar to the direct memory access method. The way by which the CPU and IOP communicate depends on the level of sophistication included in the system. In very-large-scale computers, each processor is independent of all others and any one processor can initiate an operation. In most computer systems, the CPU is the master while the IOP is a slave processor. The CPU is assigned the task of initiating all operations, but I/O instructions are executed in the IOP. CPU instructions provide operations to start an I/O transfer and also to test I/O status conditions needed for making decisions on various I/O activities. The IOP, in turn, typically asks for CPU attention by means of an interrupt. It also responds to CPU requests by placing a status word in a prescribed location in memory to be examined later by a CPU program. When an I/O operation is desired, the CPU informs the IOP where to find the I/O program and then leaves the transfer details to the IOP.

Instructions that are read from memory by an IOP are sometimes called commands, to distinguish them from instructions that are read by the CPU. Otherwise, an instruction and a command have similar functions. Commands are prepared by experienced programmers and are stored in memory. The command words constitute the program for the IOP. The CPU informs the IOP where to find the commands in memory when it is time to execute the I/O program.

Q 79. Explain Maskable Interrupts.

(PTU, May 2004)

Ans. Maskable interrupts : Maskable interrupts are hardware interrupts that can be enabled and disabled by software and which can avoid other interrupts, otherwise they are non-markable, if any other interrupt occur in between the control goes to that interrupt and after its task is complete, the control is switched back to the previous instruction.

Q 80. Write about DMA transfer.

(PTU, Dec. 2014, 2007, 2008)

OR

Explain in detail how Direct Memory Access system works by taking a suitable example showing the various stages. (PTU, May 2005)

Ans. DMA : The transfer of data between a fast storage device as magnetic disk and memory is often limited by the speed of the CPU. Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called direct memory access (DMA). During DMA transfer, the CPU is idle and has no control of the memory buses. A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.

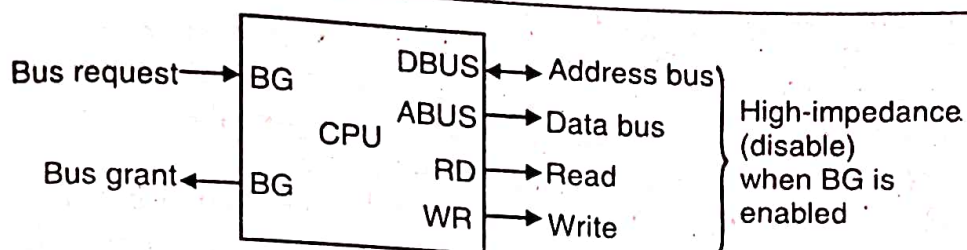
The CPU may be placed in an idle state in a variety of ways. One common method extensively used in microprocessors is to disable the buses through special control signals. Fig. shows two control signals in the CPU that facilitate the DMA transfer. The bus request (BR) input is used by the DMA controller to request the CPU to relinquish control of the buses. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus, and the read and write lines into a high-impedance state. The high-impedance state behaves like an open circuit, which means that the output is disconnected and does not have a logic significance bus grant (see Sec. 4-3). The CPU activates the bus grant (BG) output to inform the external DMA that the buses are in the high-impedance state. The DMA that originated the bus request can now take control of the buses to conduct memory transfers without processor intervention. When the DMA terminates the transfer, it disables the bus request line. The CPU disables the bus grant, takes control of the buses, and returns to its normal operation.

When the DMA takes control of the bus system, it communicates directly with the memory. The transfer can be made in several ways. In DMA burst transfer, a block sequence consisting of a number of memory words is transferred in a continuous burst while the DMA controller is master of the memory buses. This mode of transfer is needed for fast devices such as magnetic disks, where data transmission cannot be stopped or slowed down until cycle stealing an entire block is transferred. An alternative technique called cycle stealing allows the DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU. The CPU merely delays its operation for one memory cycle to allow the direct memory I/O transfer to "steal" one memory cycle.

DMA Controller

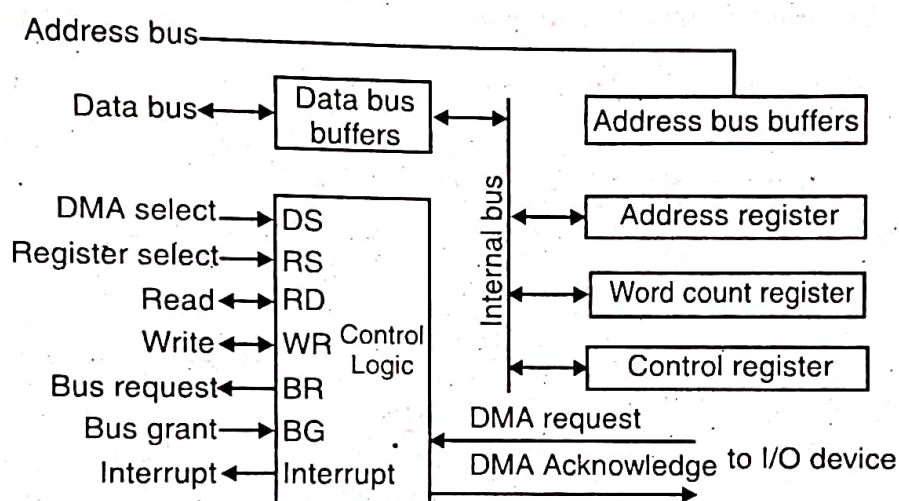
The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device. In addition, it needs an address register, a word count register, and a set of address lines. The address register and address lines are used for direct communication with the memory. The word count register specifies the number of words that must be transferred. The data transfer may be done directly between the device and memory under control of the DMA.

Fig. shows the block diagram of a typical DMA controller. The unit communicates with the CPU via the data bus and control lines. The registers in the DMA are selected by the CPU through the address bus by enabling the DS (DMA select) and RS (register select) inputs. The RD (read) and WR (write) inputs are bidirectional. When the BG (bus grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers. When BG = 1, the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control. The DMA communicates with the external peripheral through the request and acknowledge lines by using a prescribed handshaking procedure.



CPU bus signals for DMA transfer

The DMA controller has three registers: an address register, a word count register, and a control register. The address register contains an address to specify the desired location in memory. The address bits go through bus buffers into the address bus. The address register is incremented after each word that is transferred to memory. The word count register holds the number of words to be transferred. This register is decremented by one after each word transfer and internally tested for zero. The control register specifies the mode of transfer. All registers in the DMA appear to the CPU as I/O interface register. Thus the CPU can read from or write into the DMA registers under program control via the data bus.



Block diagram of DMA controller

The DMA is first initialized by the CPU. After that, the DMA starts and continues to transfer data between memory and peripheral unit until an entire block is transferred. The initialization process is essentially a program consisting of I/O instructions that include the address for selecting particular DMA registers. The CPU initializes the DMA by sending the following information through the data bus:

1. The starting address of the memory block where data are available (for read) or where data are to be stored (for write)
2. The word count, which is the number of words in the memory block
3. Control to specify the mode of transfer such as read or write
4. A control to start the DMA transfer

The starting address is stored in the address register. The word count is stored in the word count register, and the control information in the control register. Once the DMA is initialized, the CPU stops communicating with the DMA unless it receives an interrupt signal or if it wants to check how many words have been transferred.

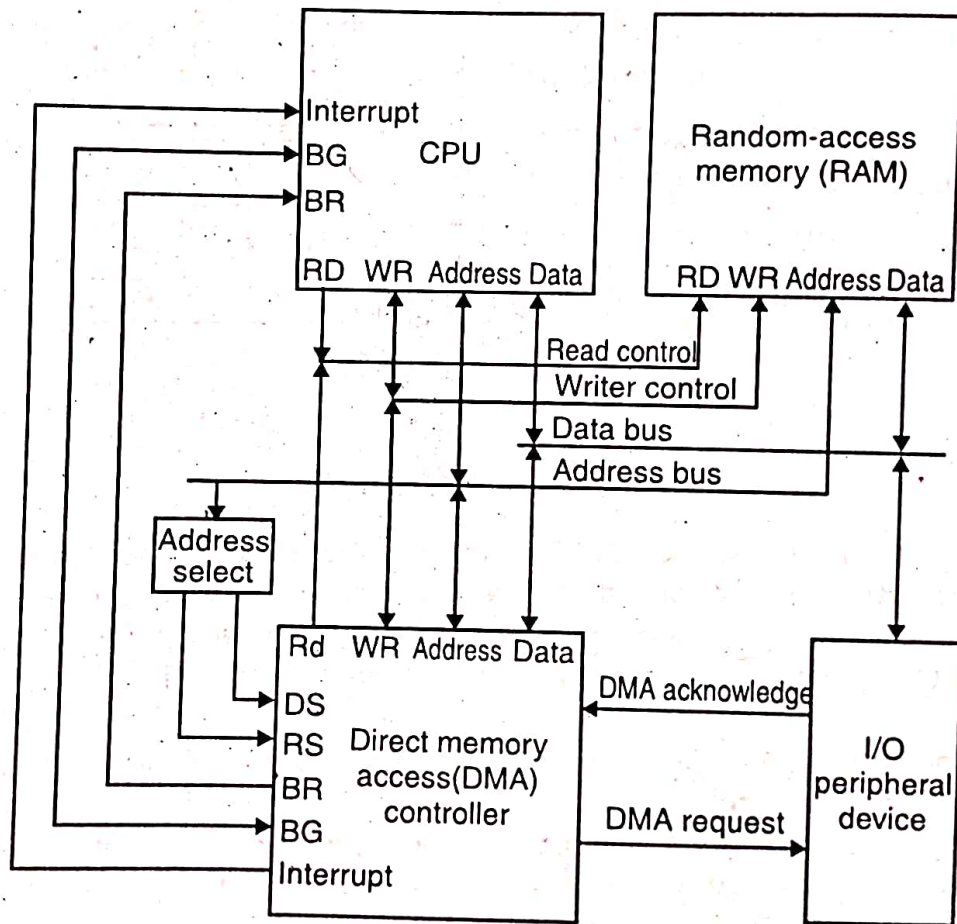
DMA Transfer

The position of the DMA controller among the other components in a computer system is

illustrated in Fig. The CPU communicates with the DMA through the address and data buses as with any interface unit. The DMA has its own address, which activates the DS and RS lines. The CPU initializes the DMA through the data bus. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory.

When the peripheral device sends a DMA request, the DMA controller activates the BR line, informing the CPU to relinquish the buses. The CPU responds with its BG line, informing the DMA that its buses are disabled. The DMA then puts the current value of its address register into the address bus, initiates the RD or WR signal, and sends a DMA acknowledge to the peripheral device. Note that the RD and WR lines in the DMA controller are bidirectional. The direction of transfer depends on the status of the BG line. When BG = 0, the RD and WR are input lines allowing the CPU to communicate with the internal DMA registers. When BG = 1, the RD and WR are output lines from the DMA controller to the random-access memory to specify the read or write operation for the data.

When the peripheral device receives a DMA acknowledge, it puts a word in the data bus (for write) or receives a word from the data bus (for read). Thus the DMA controls the read or write operations and supplies the address for the memory. The peripheral unit can then communicate with memory through the data bus for direct transfer between the two units while the CPU is momentarily disabled.



DMA transfer in a computer system

For each word that is transferred, the DMA increments its address register and decrements its word count register. If the word count does not reach zero, the DMA checks the request line coming from the peripheral. For a high-speed device, the line will be active as soon as the previous transfer is completed. A second transfer is then initiated, and the process continues until the entire block is transferred. If the peripheral speed is slower, the DMA request line may come somewhat later. In this

When the peripheral requests a transfer, the DMA requests the buses again. If the word count register reaches zero, the DMA stops any further transfer and removes its bus request. It also informs the CPU of the termination by means of an interrupt. When the CPU responds to the interrupt, it reads the content of the word count register. The zero value of this register indicates that all the words were transferred successfully. The CPU can read this register at any time to check the number of words already transferred.

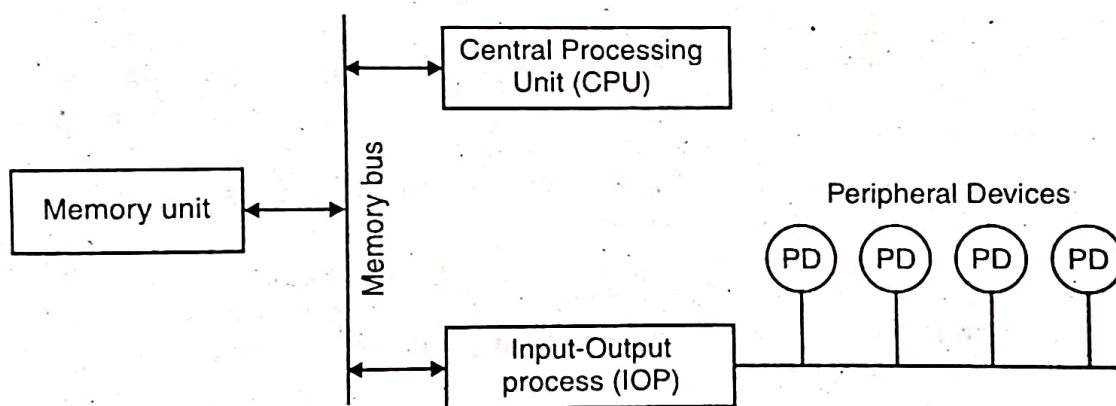
A DMA controller may have more than one channel. In this case, each channel has a request and acknowledge pair of control signals which are connected to separate peripheral devices. Each channel also has its own address register and word count register within the DMA controller. A priority among the channels may be established so that channels with high priority are serviced before channels with lower priority.

DMA transfer is very useful in many applications. It is used for fast transfer of information between magnetic disks and memory. It is also useful for updating the display in an interactive terminal. Typically, an image of the screen display of the terminal is kept in memory which can be updated under program control. The contents of the memory can be transferred to the screen periodically by means of DMA transfer.

DMA transfer is very useful in many applications. It is used for fast transfer of information between magnetic disks and memory. It is also useful for updating the display in an interactive terminal. Typically, an image of the screen display of the terminal is kept in memory which can be updated under program control. The contents of the memory can be transferred to the screen periodically by means of DMA transfer.

Q 81. When a device interrupt occurs, how does the processor determine which device issued the interrupt? (PTU, Dec. 2007)

Ans. Instead of having each interface communicate with CPU, a computer may incorporate one or more external processor and assign them the task of communicating directly with I/O devices. An input-output processor (IOP) may be classified as a processor with direct memory access capability that can be divided into a memory unit and a number of processors comprised of CPU and one or more IOPs.



Block diagram of computer with I/O processor

So each IOP takes care of input and output tasks, relieving the CPU from housekeeping chores involved in I/O transfers. A processor that communicates with remote terminals over telephone and other communication media in a serial fashion is called data communication processor (DCP).

The IOP is similar to CPU except it is designed to handle the details of I/O processing. Unlike DMA controller that must be set up entirely by the CPU, the IOP can fetch and execute own instructions. In addition, the IOP can perform other processing tasks such as arithmetic logic, branching and code translation.

The block diagram of computer with I/O processor shown above. The memory unit occupies a central position and can communicate with processor by means of direct memory access. The CPU

is responsible for processing data needed in solution of computational tasks. The IOP provides a path for transfer of data between various peripheral devices and the memory unit. Its assumed that a device is sending a sequence of bytes that must be stored in memory. The transfer of byte requires three instructions :

1. Read the status register.
2. Check the status of flag bit and branch to step 3 if not set or to step 3 if set.
3. Read the data register.

Q 82. What do you mean by software and hardware interrupts? How these are used in a microprocessor system?

(PTU, Dec. 2013 ; May 2008)

OR

Software interrupt and hardware interrupt.

(PTU, Dec. 2008)

Ans. 1. External Interrupts : External interrupts come from input-output (I/O) devices, from a timing device, from a circuit monitoring the power supply, or from any other external resource. Examples that cause external interrupts are I/O device requesting transfer of data, I/O device finished transfer of data, elapsed time of an event, or power failure.

2. Internal Interrupt : Internal interrupts arise from illegal or erroneous use of an instruction or data. Internal interrupts are also called traps. Examples of interrupts caused by internal error conditions are register overflow, attempt to divide by zero, stack overflow, protection violation and invalid operation code.

3. Software Interrupt : External and Internal interrupts are initiated from signals that occur in hardware of CPU. A software interrupt is initiated by executing an instruction. Software interrupt is a special call instruction that behaves like an interrupt rather than a subroutine call. It can be used by the programmer to initiate an interrupt procedure at any desired point in the program. The most common use of software interrupt is associated with a supervisor call instruction. This instruction provides means for switching from CPU user mode to supervisor mode. Certain operations in computer may be assigned to supervisor mode only, for example, a complex input and output procedure. A program written by user must most run in user mode. When an input or output transfer is required the supervisor mode requested by means of a supervisor call instruction. This instruction causes a software interrupt that stores the old CPU state and bring a new PSW (program status word) that belongs to supervisor mode. The calling program must pass information to operating system in order to specify particular task requires.

In computing, an **interrupt** is an asynchronous signal indicating the need for attention or a synchronous event in software indicating the need for a change in execution.

A hardware interrupt causes the processor to save its state of execution via a context switch, and begin execution of an interrupt handler.

Software interrupts are usually implemented as instructions in the instruction set, which cause a context switch to an interrupt handler similar to a hardware interrupt.

Interrupts are a commonly used technique for computer multitasking, especially in real-time computing. Such a system is said to be interrupt-driven.

An act of interrupting is referred to as an interrupt request (IRQ).

The interrupt are used in microprocessor system follows this technique :

Hardware Interrupts (IRQ) and Conflicts

The Concept of Interrupts : Because the processor cannot simultaneously process several pieces of information (it processes one piece of information at a time), a program being run can, thanks to an interrupt request, be momentarily suspended while an interrupt takes place. The interrupted program can then continue running. There are 256 different interrupt addresses.

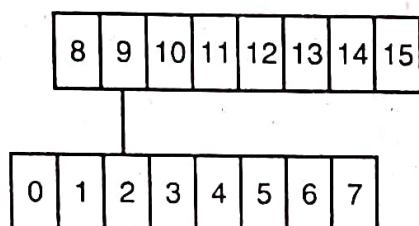
An interrupt becomes a hardware interrupt when it is requested by one of the computer's hardware

components. There are many peripherals in a computer. These peripherals generally need to use the system resources if only to communicate with the system itself.

When a peripheral wants to access a resource, it sends an interrupt request to the processor in order to get its attention. The peripherals have an interrupt number that is called an **IRQ** (Interrupt Request). It is as if each peripheral pulls a "string" that is attached to a bell in order to tell the computer that it wants the computer to pay attention to it.

This "string" is in fact a physical line that links each expansion slot as well as each I/O interface to the motherboard. For an 8-bit ISA slot, for example, there are 8 IRQ lines that link the 8-bit ISA slots to the motherboard (IRQ0 to IRQ7). These IRQs are controlled by an "interrupt controller" that is in charge of allowing the IRQ with the greatest priority "to speak".

When 16-bit slots were introduced, IRQs 8 to 15 were added, as was a second interrupt controller. The two groups of interrupts are linked by IRQ2 which is connected (or "cascaded") to IRQ9. In a way, this cascade "inserts" IRQs 8 to 15 between IRQs 1 and 3:



Given that priority goes from lowest to highest IRQ, and IRQs 8 to 15 are inserted between IRQs 1 and 3, the order of priority is as follows:

0 > 1 > 8 > 9 > 10 > 11 > 12 > 13 > 14 > 15 > 3 > 4 > 5 > 6 > 7

Physical devices within the system initiate hardware interrupts. These devices may be located physically on the platform or located on an add-in board. In both cases, a special signal from the device is connected to an interrupt request (IRQ) line of a system interrupt controller. The primary function of hardware interrupts is to relieve the CPU from having to poll devices waiting for hardware events to happen. In other words, microprocessor bandwidth to process external hardware events is consumed only when the hardware interrupt is detected and serviced.

Unlike software interrupts that are under programmer control, hardware interrupts may be generated by system events in a random fashion. For instance, when a key is pressed on the keyboard or when a mouse is moved, a hardware interrupt request is generated. Multiple hardware interrupts may occur simultaneously. The system interrupt controllers are responsible for detecting that a device has requested service, prioritizing the interrupt and generating a request to the CPU to execute the interrupt handler specific to the interrupt request. The only control the programmer has over an interrupt generated by hardware is what the system does when the interrupt occurs.

Q 83. What is the difference between I/O mapped input/output and memory mapped input/output? What are the advantages and disadvantages of each? (PTU, Dec. 2008)

Ans. Memory-mapped I/O (MMIO) and port I/O (also called port-mapped I/O or PMIO) are two complementary methods of performing input/output between the CPU and peripheral devices in a computer. Another method, not discussed in this article, is using dedicated I/O processors – commonly known as channels or mainframe computers – that execute their own instructions.

Memory-mapped I/O uses the same address bus to address both memory and I/O devices, and the CPU instructions used to access the memory are also used for accessing devices. In order to accommodate the I/O devices, areas of CPU's addressable space must be reserved for I/O. The reservation might be temporary – the Commodore 64 could bank switch between its I/O devices and regular memory – or permanent. Each I/O device monitors the CPU's address bus and responds to

any CPU's access of device-assigned address space, connecting the data bus to a desirable device hardware register.

Port-mapped I/O uses a special class of CPU instructions specifically for performing I/O. It is generally found on Intel microprocessors, specifically the IN and OUT instructions which can read and write a single byte to an I/O device. I/O devices have a separate address space from general memory, either accomplished by an extra "I/O" pin on the CPU's physical interface, or an entire bus dedicated to I/O.

A device's direct memory access (DMA) is not affected by those CPU-to-device communication methods, especially it is not affected by memory mapping. This is because, by definition, DMA is a memory-to-device communication method, that bypasses the CPU.

Hardware interrupt is yet another communication method between CPU and peripheral device. However, it is always treated separately for a number of reasons. It is device-initiated, as opposed to CPU-initiated methods. It is also unidirectional, as information flows only from device to CPU. Lastly, each interrupt line carries itself only one bit of information with a fixed meaning, namely "there is an interrupt".

Relative merits/demerits of the two I/O methods

The main advantage of using port-mapped I/O is an CPU's with a limited addressing capability. Because port-mapped I/O separates I/O access from memory access, the full address space can be used for memory. It is also obvious to a person reading an assembly language program listing when I/O is being performed, due to the special instructions that can only be used for that purpose.

I/O operations can slow the memory access, if the address and data buses are shared. This is because the peripheral device is usually much slower than main memory. In some architecture, port-mapped I/O operates via a dedicated I/O bus, alleviating the problem.

Advantages : The advantages of having an IOMMU, compared to direct physical addressing of the memory, include :

- ❑ Large regions of memory can be allocated without the need to be contiguous in physical memory – the IOMMU will take care of mapping contiguous virtual addresses to the underlying fragmented physical addresses. Thus, the use of vectored I/O (scatter-gather lists) can sometimes be avoided.
- ❑ For devices that do not support memory addresses long enough to address the entire physical memory, the device can still address the entire memory through the IOMMU. This avoids overhead associated with copying buffers to and from the memory space the peripheral can address.
- ❑ For example, on contemporary x86 computers, more than 4 GiB of memory can be used, enabled by the PAE feature in an x86 processor. Still, an ordinary 32-bit PCI device simply cannot address the memory above the 4 GiB boundary, and thus it cannot perform DMA to it. Without an IOMMU, the operating system is forced to implement time consuming double buffers (Windows nomenclature) also known as bounce buffers (Linux).
- ❑ Memory protection from malicious or misbehaving devices : a device cannot read or write to memory that hasn't been explicitly allocated (mapped) for it. The memory protection is based on the fact that OS running on the CPU exclusively controls both the MMU and the IOMMU. The devices are physically unable to circumvent or corrupt configured memory management tables.
- ❑ With virtualization, guest operating systems can use hardware that is not specifically made for virtualization. Higher performance hardware such as graphics cards use DMA to access memory directly ; in a virtual environment all the memory addresses are remapped by the

- virtual machine software, which causes DMA devices to fail. The IOMMU handles this remapping, allowing for the native device drivers to be used in a guest operating system.
- ❑ In some architectures IOMMU performs also hardware interrupt remapping, in a manner similar to standard memory address remapping.
 - ❑ Peripheral memory paging can be supported by an IOMMU. A peripheral using the PCI-SIG PCIe Address Translation Services (ATS) Page Request Interface (PRI) extension can detect and signal the need for memory manager services.

For system architecture in which port I/O is a distinct address space from the memory address space, an IOMMU is not used when the CPU communicates with devices via I/O ports. In system architectures in which part I/O and memory are mapped into a suitable address space, an IOMMU can translate port I/O accesses.

Disadvantages :

The disadvantages of having an IOMMU, compared to direct physical addressing of the memory, include :

- ❑ Some degradation of performance from translation and management overhead (e.g., page table walks).
- ❑ Consumption of physical memory for the added I/O page (translation) tables. This can be mitigated if the tables can be shared with the processor.

Q 84. How many characters per second can be transmitted over a 1200-band line in each of the following modes considering a character code of 8 bits :

(a) Synchronous serial transmission.

(b) Asynchronous serial transmission with 2 stop bit.

(PTU, May 2005)

Ans. (a) Serial and Parallel data transmission : The need to provide data transfer between a computer and a remote terminal has led to the development of serial communication.

Serial data transmission implies transfer data transfer bit by bit on the single (serial) communication line.

In case of serial transmission data is sent in a serial form i.e. bit on a single line. Also, the cost of communication hardware is considerable reduced since only a single wire or channel is require for the serial bit transmission. Serial data transmission is slow as compared to parallel transmission.

However, parallel data transmission is less common but faster than serial transmission. Most data are organized into 8 bit types. In some computers, data are further organized into multiple bits called half words, full words. Accordingly data is transferred some times a byte or word at a time on multiple wires with each wire carrying individual data bits. Thus transmitting all bits of a given data byte or word at the same time is known as parallel data transmission.

Parallel transmission is used primarily for transferring data between devices at the same site. For eg : communication between a computer and printer is most often parallel, so that entire byte can be transferred in one operation.

Synchronous Communication : In Synchronous communication scheme, after a fixed number of data bytes a special bit pattern in send called SYNC by the sending end.

Data transmission take place without any gap between two adjacent characters, however data is send block by block. A block is a continuous steam of characters or data bit pattern coming at a fixed speed. You will find a Sync bit pattern between any two blocks of data and hence the data transmission is synchronized.

Synchronous communication is used generally when two computers are communicating to each other at a high speed or a buffered terminal is communicating to the computer.

Advantages and Disadvantages of Synchronous Communication : Main advantage of

Synchronous data communication is the high speed. The synchronous communications require high-speed peripherals/devices and a good-quality, high bandwidth communication channel.

The disadvantage include the possible in accuracy. Because when a receiver goes out of synchronization, loosing tracks of where individual characters begin and end. Correction of error takes additional time.

(b) Asynchronous data transmission : Serial data communication generally employs either synchronous or asynchronous communication scheme. This two scheme used different techniques for synchronizing in the circuits in sending and receiving end.

In asynchronous transmission each character is transmitted separately, that is one character at a time. The character is preceded by a start bit, which tells the receiving end where the character coding begins, and is followed by a stop bit, which tells the receiver where the character coding ends. There will be intervals of ideal time on the channell shown as gaps. Thus there can be gaps between two adjacent characters in the asynchronous communication scheme. In this scheme, the bits within the character frame (including start, parity and stop bits) are sent at the baud rate.

The START BIT and STOP BIT including gaps allow the receiving and sending computers to synchronise the data transmission. Asynchronous communication is used when slow speed peripherals communicate with the computer. The main disadvantage of asynchronous communication is slow speed transmission. Asynchronous communication however, does not require the complex and costly hardware equipments as is required for synchronous transmission.

Advantages and Disadvantages of Asynchronous Transmission : The advantage of asynchronous transmission is that it does not required any local storage at the terminal or the computer and is thus cheaper to implement.

Major disadvantage of asynchronous transmission is that the transmission lines is idle during the time intervals between transmitting characters.

Synchronous and Asynchronous serial data transmission : Consider a parallel port with 8 data lines. It can send/receive 8 bits in parallel (simultaneously). In contrast, a serial port only needs one data line to transmits the 8-bit byte. To do this the byte is transmitted bit by bit, in a serial manner. There are two serial data transfer schemes : sychronous and asynchronous transmissions.

In the synchronous data transfer scheme, additional lines are required to transmit handshake or timing signals along with the data line to indicate when the next bit is to be transmitted on the line. The advantage of this data transfer is that the receiver is able to respond to the clock rate of the transmitter automatically. For asynchronous data transfer, the transmitted data themselves contain the information required for sychronization and neither handshake nor clock signals are needed. The transmitted serial data comprises a Start Bit, which indicates the beginning of data transmission. It is followed by Serial Data Bits and then Stop Bits indicating th end of the transmission. An optional Parity Bit can be added between the Serial Data Bits and the Stop Bit for parity checking. The receiver device detects the start bit and receives the subsequent data bits. This data transfer requires that the transmitter and the receiver must have the same clock frequency.

Asynchronous data transmission is used in most personal computers. In practice, the asynchronous communication is facilitated by a family of industrial standard computer peripheral ICs known as the UARTs (Universal Asynchronous Receivers and Transmitters).

Q 85. Write short note on the following :

- (a) Parallel Computing.
- (b) Distributed Computing.
- (c) Serial and Parallel interface.

Ans. (a) Parallel computing is a form of computation in which many calculations are carried

(PTU, May 2010 ; Dec. 2008)

out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel"). There are several different forms of parallel computing: bit-level, instruction level, data and task parallelism. Parallelism has been employed for many years, mainly in high-performance computing, but interest in it has grown lately due to the physical constraints preventing frequency scaling. As power consumption (and consequently heat generation) by computers has become a concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multicore processors.

Parallel computers can be roughly classified according to the level at which the hardware supports parallelism – with multi-core and multi-processor computers having multiple processing elements within a single machine, while clusters, MPPs and grids use multiple computers to work on the same task. Specialized parallel computer architectures are sometimes used alongside traditional processors, for accelerating specified tasks.

Parallel computer programs are more difficult to write than sequential ones, because concurrency introduces several new classes of potential software bugs, of which race conditions are the most common. Communication and synchronization between the different subtasks are typically one of the greatest obstacles to getting good parallel program performance. The speed-up of a program as a result of parallelization is governed by Amdahl's law.

Applications :

As parallel computers become larger and faster, it becomes feasible to solve problems that previously took too long to run. Parallel computing is used in a wide range of fields, from bioinformatics (to do protein folding) to economics (to do simulation in mathematical finance). Common types of problems found in parallel computing applications are :

- Dense linear algebra
- Sparse linear algebra
- Spectral methods (such as Cooley-Tukey Fast Fourier transform)
- N-body problems (such as Barnes-Hut simulation)
- Structured grid problems (such as Lattice Boltzmann methods)
- Unstructured grid problems (such as found in finite element analysis)
- Monte Carlo simulation
- Combinational logic (such as brute-force cryptographic techniques)
- Graph traversal (such as Sorting algorithms)
- Dynamic programming
- Branch and bound methods
- Graphical models (such as detecting Hidden Markov models and constructing Bayesian networks)
- Finite State Machine simulation

(b) Distributed computing deals with hardware and software systems containing more than one processing element or storage element, concurrent processes, or multiple programs, running under a loosely or tightly controlled regime.

In distributed computing a program is split up into parts that run simultaneously on multiple computers communicating over a network. Distributed computing is a form of parallel computing, but parallel computing is most commonly used to describe program parts running simultaneously on multiple processors in the same computer. Both types of processing require dividing a program into parts that can run simultaneously, but distributed programs often must deal with heterogeneous environments, network links of varying latencies, and unpredictable failures in the network or the computers.

There are many different types of distributed computing systems and many challenges to overcome in successfully designing one. The main goal of a distributed computing system is to connect users and resources in a transparent, open, and scalable way. Ideally this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems. Openness is the property of distributed systems such that each subsystem is continually open to interaction with other systems (see references). Web services protocols are standard which enable distributed systems to be extended and scaled. In general, an open system that scales has an advantage over a perfectly closed and self-contained system. Openness cannot be achieved unless the specification and documentation of the key software interface of the component of a system are made available to the software developer.

Different subsystems of an open distributed system include heterogeneous, overlapping and possibly conflicting information. There is no central arbiter of Unbounded Nondeterminism

Asynchronously, different subsystems can come up and go down and communication links can come in and go out between subsystems of an open distributed system. Therefore the time that it will take to complete an operation cannot be bounded in advance.

If not planned properly, a distributed system can decrease the overall reliability of computations if the unavailability of a node can cause disruption of the other nodes. Leslie Lamport famously quipped that : "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."

Troubleshooting and diagnosing problems in a distributed system can also become more difficult, because the analysis may require connecting to remote nodes or inspecting communication between nodes.

Many types of computation are not well suited for distributed environments, typically owing to the amount of network communication or synchronization that would be required between nodes. If bandwidth, latency, or communication requirements are too significant, then the benefits of distributed computing may be negated and the performance may be worse than a non-distributed environment.

(c) Serial and Parallel Interface : In telecommunication and computer science, **serial communication** is the process of sending data one bit at one time, sequentially, over a communication channel or computer bus. This is in contrast to parallel communication, where several bits are sent together, on a link with several parallel channels. Serial communication is used for all long-haul communication and most computer networks, where the cost of cable and synchronization difficulties make parallel communication impractical. At shorter distances, serial computer buses are becoming more common because of a tipping point where the disadvantages of parallel busses (clock skew, interconnect density) outweigh their advantage of simplicity (no need for serializer and deserializer (SERDES)). Improved technology to ensure signal integrity and to transmit and receive at a sufficiently high speed per lane have made serial links competitive. The migration from PCI to PIC-Express is an example.

In telecommunication and computer science, **parallel communication** is a method of sending several data signals simultaneously over several parallel channels. It contrasts with serial communication ; this distinction is one way of characterizing a communication link.

The basic difference between a parallel and a serial communication channel is the number of distinct wires or strands at the physical layer used for simultaneous transmission from a device. Parallel communication implies more than one such wire/strand, in addition to a ground connection. An 8-bit parallel channel transmits eight bits (or a byte) simultaneously. A serial channel would transmit those bits one at a time. If both operated at the same clock speed, the parallel channel would be eight times faster. A parallel channel will generally have additional control signals such as a clock.

to indicate that the data is valid, and possibly other signals for handshaking and directional control of data transmission.

Q 86. What is the difference between memory-mapped and peripheral mapped I/O?

(PTU, May 2009)

Ans. Memory-mapped I/O uses a section of memory for I/O. The idea is simple. Instead of having "real" memory (i.e., RAM) at that address, place an I/O device.

Thus, communicating to an I/O device can be the same as reading and writing to memory addresses devoted to the I/O device. The I/O device merely has to use the same protocol to communicate with the CPU as memory uses. Some ISAs use special I/O instructions. However, the signals generated by the CPU for I/O instructions and for memory-mapped I/O are nearly the same. Usually, there's just one special I/O pin that lets you know whether it's a memory address or an I/O address. The main advantage of using port-mapped I/O is on CPUs with a limited addressing capability. Because port-mapped I/O separates I/O access from memory access, the full address space can be used for memory.

I/O operations can slow the memory access, if the address and data buses are shared. This is because the peripheral device is usually much slower than main memory. In some architectures, port-mapped I/O operates via a dedicated I/O bus, alleviating the problem. A peripheral device that assigns specific memory locations to input and output. For example, in a memory mapped display, each pixel or text character derives its data from a specific memory byte or bytes. The instant this memory is updated by software, the screen is displaying the new data.

Q 87. What is the transfer rate of an eight track magnetic tape whose speed is 120 inches per second and whose density is 1600 bit per inch?

(PTU, May 2011)

Ans. The formula is : number of bytes on a track \times Rotation time

Therefore transfer rate = $1600 \times 120 = 192000$ ms/track.

Q 88. Explain input output interface.

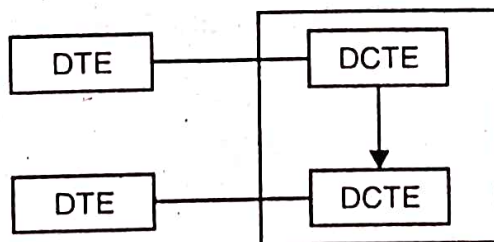
Ans. Input output interface provides a method for transferring information between internal storage and external input/output devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit. The purpose of the communication link is to be resolve the differences that exist between the central computer and each peripheral.

Q 89. Describe various modes of data transfer. Why does DMA have priority over the CPU when request a memory transfer?

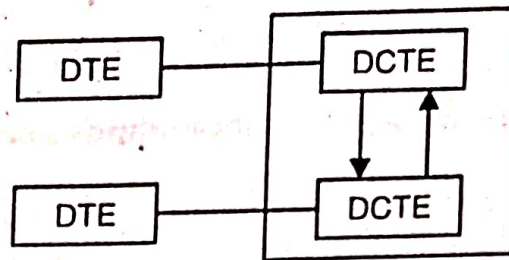
(PTU, May 2011)

Ans. Data can be transferred in three different modes, these 3 modes are equally popular and used in common everyday situation.

Simplex : If data can be transferred in a single direction it is known as simplex. Television transmission is an example of simplex transmission, here data can be transferred only to a television but no data can be transferred in the opposite direction.

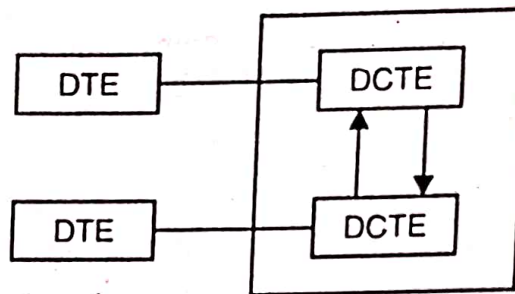


Duplex : When data may be transferred in two directions simultaneously it is known as Duplex, a simple telephone conversation would be an example, while we usually don't talk simultaneously a telephone provides that 'duplex' functionality.



Full-duplex Connection

In this mode of transmission as well data may be transmitted in both directions however it cannot be done simultaneously. A Walkie Talkie would be an example of this mode of transmission.



Half-duplex Connection

Q 90. Where ASCII code is used in computers?

(PTU, Dec. 2010)

Ans. The American standard code for information interchange (ASCII) is a character-encoding scheme based on the ordering of the English alphabet. ASCII codes represent text in computers communications equipment and other devices that use text. Most modern character encoding schemes are based on ASCII though they support many more characters than ASCII does.

ASCII is the Internet Assigned Numbers Authority (IANA) preferred charset name for ASCII.

Q 91. What do you mean by synchronous data transfer?

Ans. Synchronous data transfer usually occurs when peripherals are located within the same computer as the CPU because their close proximity allows them to share a common clock and because data does not have to travel very far physically which becomes a concern at a higher clock frequencies.

In synchronous data transfer a bus clock signal provides the timing information for all actions on the bus. Change in other signals is relative to the falling or rising edge of the clock. Synchronous transmissions are synchronized by an external clock.

Q 92. What is the need of synchronization?

Ans. Whenever an electronic device transmits digital data to another electronic device, there must be a certain rhythm established between the two devices i.e. the receiving device must have some way of knowing within the context of the fluctuating signal that is receiving, where each unit of data begins and where it ends.

Q 93. What do you mean by Asynchronous Data Transfer?

(PTU, May 2017)

Ans. In asynchronous data transfer, there is no clock signal. Asynchronous transmissions are synchronized by special signals along the transmission medium. Asynchronous transfers use control signals and their associated hardware to co-ordinate the movement of data. These data transfers do not require that the source and destination use the same system clock.

There are four types of asynchronous data transfers :

1. Source-initiated data transfer without handshaking.
2. Destination initiated data transfer without handshaking.

3. Source-initiated data transfer with handshaking.
4. Destination-initiated data transfer with handshaking.

Q 94. What is handshaking?

Ans. The unit receiving the data item respond with another control signal to acknowledge receipt of the data. This type of agreement between two independent units is referred to as handshaking.

Q 95. What are the various modes of transfer?

Ans. Data transfer to and from peripherals may be handled in one of the three possible modes:

1. Programmed Input/Output
2. Interrupt-initiated Input/Output
3. Direct memory access (DMA).

Q 96. What is priority interrupt?

(PTU, May 2018, 2016 ; Dec. 2016)

Ans. A priority interrupt is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously.

The system may also determine which condition are permitted to interrupt the computer while another interrupt is being serviced. Higher priority interrupt levels are assigned to the requests which if delayed or interrupted could have serious consequences.

Q 97. What are the advantages of interrupts in computers?

(PTU, May 2012)

Ans. An interrupt is a break up and exception is exclusion.

1. Often used in interfacing applications that require synchronization.
2. Many operations can performs simultaneously.
3. Comfortable of using and handles of huge amount of operations.
4. It is used to stop unwanted processes.

Q 98. Discuss the various I/O data transfer techniques alongwith their merits and demerits.

(PTU, Dec. 2019, 2010)

Ans. The READ statement obtains data from an external or internal file and transfers the data to internal storage. If you specify an input list, values transfer from the file to the data item your specify.

The PRINT statement transfer data from internal storage into an external file. Specifying the `port = tpestmt` compiler option enables the type statement which supports functionality identical to PRINT. If you specify an output list and format specification values transfer to the file from the data items you specify. If you do not specify an output list. The PRINT statement transfers a blank record to the output device unless the **FORMAT** statement it refers to contains, as the first specification, a character string edit descriptor or a splash edit descriptor. In this case, the records these specifications indicate transfer to the output device.

Execution of a WRITE or PRINT statement for a file that does not exist creates that file unless an error occurs.

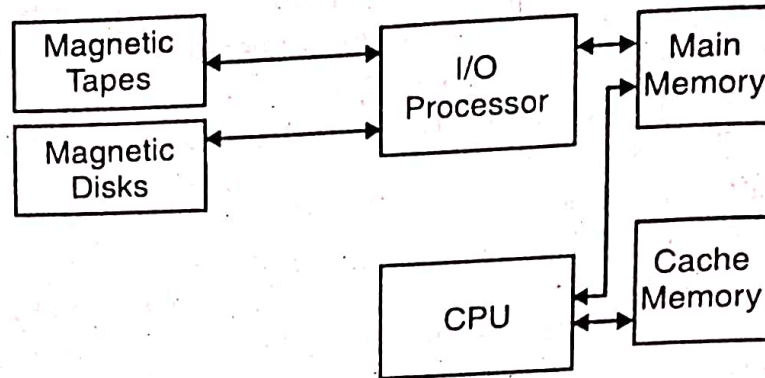
If an input/output item is a pointer data transfers between the file and the associated target.

Q 99. What is memory organisation ? Explain various memories.

(PTU, May 2016, 2006 ; Dec. 2013)

Ans. Memory Organization : The memory unit that communicates directly with the CPU is called the main memory. The total memory capacity of a computer can be visualized as being a hierarchy of components. The memory hierarchy system consists of all storage devices employed in a computer system from the slow but high-capacity auxiliary memory accessible to the high speed processing logic illustrates the components in a typical memory hierarchy. As the bottom of the hierarchy are the relative, slow magnetic tapes used to store removable files. Next are the magnetic disks used as back up storage. The main memory occupies a central position by being able to

communicate directly with the CPU and with auxiliary memory devices through an I/O processor. When programs not residing in main memory are needed by the CPU, they are brought in from auxiliary memory. Programs not currently needed in main memory are transferred into auxiliary memory to provide space for currently used program and data.



Memory hierarchy in a Computer system

Various Memories in Computer System

1. Main Memory : The main memory is the central storage unit in a computer system. It is a relatively large and fast memory used to store program and data during the computer operation. The principal technology used for the main memory is based on semiconductor integrated circuits. Integrated circuit RAM chips are available in two possible operating modes, static and dynamic. The static RAM consists essentially of internal flip-flops that store the binary information. The stored information remains valid as long as power is applied to the unit. The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors.

2. Auxiliary Memory : The most common auxiliary memory devices used in computer systems are magnetic disk and tapes. Other components used, but not as frequently, are magnetic drums, magnetic bubble memory and optical disks. To understand fully the physical mechanism of auxiliary memory device one must have a knowledge of magnetics, electronics and electromechanical systems.

3. Associative Memory : The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address. A memory unit accessed by content is called an associative memory or content addressable memory (CAM). This type of memory is accessed simultaneously and in parallel on the basis of data content. This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location.

4. Cache Memory : Analysis of a large number of typical programs has shown that the references to memory at any given interval of time tend to be confined within a few localized areas in memory. This phenomenon is known as the property of locality of reference. The reason for this property may be understood considering that a typical computer program flows in a straight-line fashion with program loops and subroutine calls encountered frequently. When a program loop is executed, the CPU repeatedly refers to the set of instructions in memory that constitute the loop.

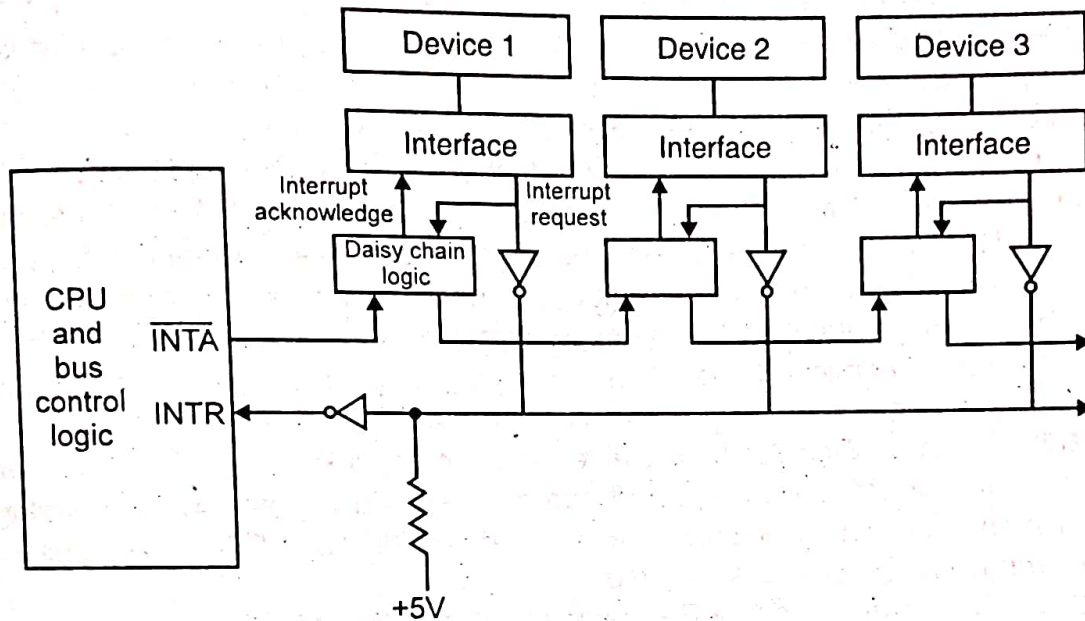
5. Virtual Memory : In a memory hierarchy system, programs and data are first stored in auxiliary memory. Portions of a program or data are brought into main memory as they are needed by the CPU. Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory. Each address that is referred by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory. Virtual memory is used to give programmes

the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory.

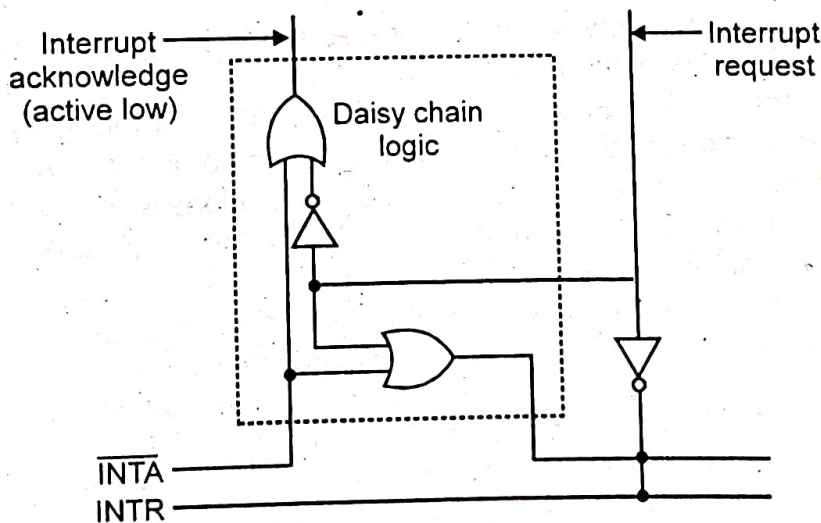
Q 100. Describe in detail the Daisy-chaining priority to handle interrupts with suitable example. (PTU, Dec. 2016 ; May 2012)

Ans. The hardware priority function can be established by either a serial or a parallel connection of interrupt lines. The serial connection is called daisy chaining method. In daisy chaining method all the devices are connected in serial. The device with the highest priority is placed in the first position, followed by lower priority devices.

A daisy chain is used to identify the device requesting service.



(a) Daisy Chain



(b) Logic

Daisy chaining is used for level sensitive interrupts, which act like a wired 'OR' gate. Any requesting device can take the interrupt line low, and keep it asserted low until it is serviced.

Because more than one device can assert the shared interrupt line simultaneously, some method must be employed to ensure device priority. This is done using the interrupt acknowledge signal generated by the processor in response to an interrupt request.

Each device is connected to the same interrupt request line, but the interrupt acknowledge line is passed through each device, from the highest priority device first, to the lowest priority device last.

After preserving the required registers, the microprocessor generates an interrupt acknowledge signal. This is gated through each device. If device 1 generated the interrupt, it will place its identification signal on the data bus, which is read by the processor, and used to generate the address of the interrupt-service routine. If device 1 did not request the servicing, it will pass the interrupt acknowledge signal on the next device in the chain.

Q 101. What are privileged and non-privileged instructions ?

Ans. Instructions are divided into two categories : the non privileged instructions and the privileged instructions. A non-privileged instruction is an instruction that any application or user can execute. A privileged instruction, on the other hand, is an instruction that can only be executed in kernel mode. Instructions are divided in this manner because privileged instructions could the kernel.

Q 102. Explain various semiconductor memory technologies.

Ans. The different memory types or memory technologies are detailed below :

1. DRAM : Dynamic RAM is a form of random access memory. DRAM uses a capacitor to store each bit of data and the level of charge on each capacitor determines whether that bit is a logical 1 or 0. However these capacitors do not hold their charge indefinitely and therefore the data needs to be refreshed periodically. As a result of this dynamic refreshing it gains its name of being a dynamic RAM. DRAM is the form of semiconductor memory that is often used in equipment including personal computers and workstations where it forms the main RAM to the computer.

2. EEPROM : This an Electrically Erasable Programmable Read only memory. Data can be written to these semiconductor devices and it can be erased using an electrical voltage. This is typically applied to an erase pin on the chip. Like other types of PROM, EEPROM retains the contents of the memory even when the power is turned off.

3. EPROM : This is an Erasable Programmable Read Only Memory. These semiconductor devices can be programmed and then erased at a later time. This is normally achieved by exposing the semiconductor device itself to ultraviolet light.

4. Flash Memory : Flash memory may be considered as a development of EEPROM technology. Data can be written to it and it can be erased, although only in blocks, but data can be read on an individual cell basis. To erase and re-programme areas of the chip, programming voltage of levels that are available within electronic equipment are used. It is also non-volatile and this makes it particularly useful. As a result flash memory is widely used in many applications including USP memory sticks.

Compact Flash memory cards, SD memory cards and also now solid state harddrives for computers and many other applications.

5. F-RAM : Ferroelectric RAM is a random access memory technology that has many similarities to the standard DRAM technology. The major difference is that it incorporates a ferroelectric layer instead of the more usual dielectric layer and this provides its non-volatile capability. As it offers a non-volatile capability, F-RAM is a direct competitor to flash.

6. MRAM : This is magneto-resistive RAM, or magnetic RAM. It is a non volatile RAM memory technology that uses magnetic charges to store data instead of electric charges. MRAM relations data even when the power is removed. An additional advantage is that it only requires low power for active operation.

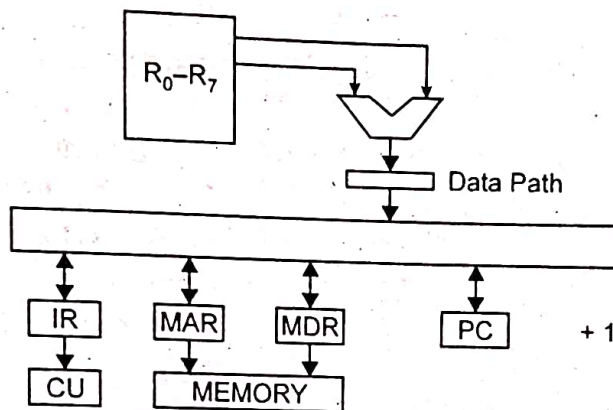
7. PRAM : This type of semiconductor memory is known as Phase change Random Access Memory. It is based around a phenomenon where a form of chalcogenide glass changes its state of phase between an amorphous state and a poly crystalline state. It is possible to detect the state of an individual cell and hence use this for data storage. Currently, this type of memory has not been widely commercialised but it is expected to be competitor for flash memory.

8. **PROM** : This stands for programmable Read Only Memory. It is a semiconductor memory which can only have data written to it once the data written to it is permanent. These memories are bought in a blank format and they are programmed using a special PROM programmer.

9. **SRAM** : Static Random Access Memory. This form of semiconductor memory gains its name from the fact that unlike DRAM, the data does not need to be refreshed dynamically. These semiconductor devices are able to support faster read and write time than DRAM and in addition its cycle time is much shorter because it does not need to pause between accesses. However they consume more power, they are less dense and more expensive than DRAM. As a result of this SRAM is normally used for Caches, while DRAM is used as the main semiconductor memory technology.

Q 103. Describe case study of a design of simple hypothetical CPU.

Ans. To illustrate how a processor can be designed, we will describe the design of a simple hypothetical CPU called S1. S1 contains all the important elements of a real processor. It is aimed to be as simple as possible so that every one can understand it easily. A simulator of S1 at instruction level is provided.



Instruction set design

- 16 bit word, fixed length
- address space 1024 words
- load store architecture
- 8 registers, R7 as stack pointer
- Condition code, Z, S Zero, Sign

Instruction formula :

1. Op : 3, ro : 3, ads : 10
 15 .. 13 12 .. 10 9 .. 0
 op ro ads
2. Op : 3, xop : 4, r1 : 3, r2 : 3
 15 .. 13 12 .. 9 8 .. 6 5 .. 3 2 .. 0
 7 xop 0 r1 r2

Instruction

		op
load M,r	0	M → r
store r, M	1	r → M
jump c, ads	2	
call o, ads	3	push (PC), goto ads
xop	7	
mov r1, r2	0	r1 → r2
load (r1), r2	1	(r1) → r2 load indirect
store r1, (r2)	2	r1 → (r2) store indirect

add r1, r2	3	$r1 + r2 \rightarrow r1$
cmp r1, r2	4	compare, affect z, s
inc r1	5	
ret	6	pop (PC)
r 0 7		
c 0 6		Conditional : 0 always 1z, 2 Nz, 3LT, 4LE, 5GE, 6GT

S1 micro architecture

Notation

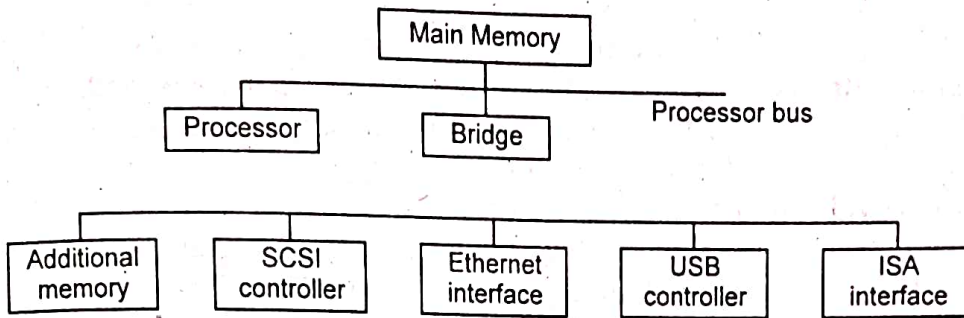
//Comment	<move>
des t = source	$T = R [IR : R1]$
e 1 ; e2	$R [IR : R2] = T$
M[a]	<add>
IR : a	$T = \text{add} (R [IR : R1], R[IR : R2])$
<name>	$R [IR : R1] = T$
op ()	<compare>
//main loop	$CC = \text{Cmp} (R[IR : R1], R[IR : R2])$
repeat	<increment>
<ifetch>	$T = \text{inc} (R [IR : R1])$
<decode>	$R [IR : R1] = T$
<execute>	<jump>
<ifetch>	If test cc (IR : R0)
MAR = PC	then DC = IR : ADS
MDR = M [MAR]	<call>
IR = MDR; PC = PC + 1	$T = \text{add1} (R[7])$
<load>	$R [T] = T$
MAR = IR : ADS	MAR = R[T]
MDR = M[MAR]	MDR = PC
$R [R - R_0] = \text{MDR}$	$M [MAR] = \text{MDR}$
<store>	PC = IR : ADS
MAR = IR : ADS	<return>
MDR = R[IR : R0]	MAR = R[7]
$M[MAR] = \text{MDR}$	MDR = M [MAR]
<loadr>	PC = MDR
MAR = R [IR : R1]	$T = \text{sub1} (R[7])$
MDR = M[MAR]	$R[7] = T$
$R(IR : R2) = \text{MDR}$	<i fetch 2>
<storer>	MAR = PC
MAR = R [IR : R2]	IR = MDR = M [MAR]; PC = PC+1
MAR = R [IR : R1]	

How to run S1 simulator

The input file is an object file with the name "in. obj". When run S1 the simulator will start and load "in.obj" and execute starting from PC = 0 until stop with the instruction call 1000.

Q 104. Explain I/O device interfaces.

Ans. The processor bus is the bus defined by the signals on the processor chip itself. Devices that require a very high speed connection to the processor such as the main memory may be connected directly to this manner. The two buses are inter connected by a circuit which we will call a bridge that translates the signals and protocols of one bus into those of the other. Device connected to the expansion bus appear to the processor as if they were connected directly to the processors own bus.



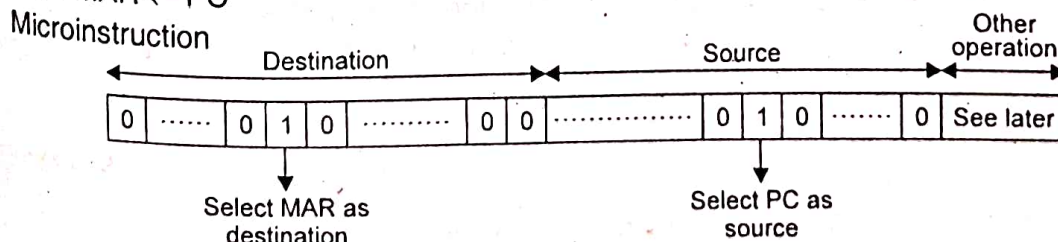
PCI (Peripheral Component Interconnect) Bus : The PCI bus is a good example of a system bus that grew out of the need for standardization. It supports the functions found on a processor bus but in a standardized format that is independent of any particular processor. Devices connected to the PCI bus appear to the processor as if they were connected directly to the processor bus. They are assigned addresses in the memory address space of the processor. The PCI follows a sequence of bus standards that were used primarily in IBM, PC's, Early PC's used the 8 bit XT bus, whose signals closely mimicked those of Intel's 8086 processors, Later the 16 bit bus used on the PC. At computers became known as the ISA bus. Its extended 32 bits version is known as EISA bus. The PCI bus was developed as a low cost but that is truly processor independent. A very important feature that the PCI pioneered is a plug and play capability for connection I/O devices. To connect a new device the user simply connects the device interface board to the bus. The software takes care of the rest.

2. SCSI bus : The acronym SCSI stands for small computer system Interface. It refers to a standard bus defined by the American National Standards Institute (ANSI) under the designation X 3.131. In the original specifications of the standard, devices such as disks are connected to a computer via a 50 wirecable which can be upto 25 meters in length and can transfer data at rates upto 5mb. A SCSI bus may have eight data lines in which case its called a narrow bus and transfers data one byte at a time. Alternatively a wide SCSI bus has 16 data lines and transfer data 16 bits at a time. Devices connected to the SCSI bus are not part of the address space of the processor in the same way as devices connected to the processor bus The SCSI bus is connected to the processor bus through a SCSI controller. This controller uses DMA to transfer data packets from the main memory to the device or vice versa. A packet may contain a block of data, commands from the processor to the device or status information about the device.

Q 105. Give an example of horizontal microinstruction.

(PTU, May 2014)

Ans. MAR ← PC



Q 106. What are characteristics of I/O channels?

(PTU, Dec. 2013)

Ans. An I/O channel has a special purpose processor. This processor has an ability to execute

I/O instructions and it can have complete control over I/O operation. The I/O instructions are stored in main memory. When I/O transfer is required, the CPU initiates an I/O transfer by instructing the device channel to execute an I/O program stored in the main memory. The I/O program specifies the device or devices, the area of memory storage, priority and actions to be taken for certain error conditions.

Q 107. Differentiate between maskable and non-maskable interrupt. (PTU, May 2014)

Ans. Maskable interrupt : The interrupt which can be ignored by the processor, while performing its operations are called maskable interrupts. Generally maskable interrupts are the interrupts that comes from the peripheral devices.

Example : Mouse-click, memory read etc.

Non Maskable interrupt : The non-maskable interrupt are the interrupts which can not be ignored. Generally these type of interrupts are specified to be software interrupts.

Example : powerfailure, software corrupted etc.

Q 108. What is cycle stealing DMA transfer? (PTU, May 2014)

Ans. DMA controllers can operate in a cycle stealing mode in which they take over the bus for each byte of data to be transferred and then return control to the CPU. They can also operate in burst mode in which a block of data is transferred before returning bus control to the CPU. The choice depends on the speed at which data is arriving relative to the bus bandwidth and whether a particular application will allow the CPU to be locked off the bus for the duration of one transfer.

Q 109. Explain instruction cycle. Give the RTL statement for each sub cycle. How the instruction cycle is to accommodate the interrupt from I/O devices? (PTU, May 2014)

Ans. A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction. In the basic computer each instruction cycle consists of the following phases.

- (i) Fetch an instruction from memory.
- (ii) Decode the instruction
- (iii) Read the effective address from memory if the instruction has an indirect address.
- (iv) Execute the instruction.

Fetch and decode

The Program Counter (PC) is loaded with the address of the first instruction in the program. The sequence counter cleared to 0, providing a decoded timing signal T_0 . After each clock pulse, SC is incremented by one, so that the timing signals go through a sequence T_0 , T_1 and so on. The micro instruction for the fetch and decode phase can be specified by the following register transfer statements.

$T_0 : AR \leftarrow PC$

$T_1 : IR \leftarrow M[AR], PC \leftarrow PC + 1$

$T_2 : D_0, \dots, D_7 \leftarrow \text{Decode } IR(12-14), AR \leftarrow IR(0-11),$

$I \leftarrow IR(15)$

To provide the data path for the transfer of PC to AR we must apply timing signal T_0 to achieve the following connection :

- (i) Place the content of PC on to the bus by making the bus selection inputs $S_2 S_1 S_0$ equal to 010.

- (ii) Transfer the content of the bus to AR by enabling the LD input of AR.

The next clock transition initiates the transfer from PC to AR since $T_0 = 1$. In order to implement the second statement,

$T_1 : IR \leftarrow M[AR], PC \leftarrow PC + 1.$

It is necessary to use timing signal T_1 to provide the following connections in the bus system.

- (i) Enable to read input of memory.
- (ii) Place the content of memory on to the bus by making $S_2 S_1 S_0 = 111$.
- (iii) Transfer the content of the bus to IR by enabling the LD input of IR.
- (iv) Increment PC by enabling the INR input of PC.

The next clock transition initiates the read and increment operations since $T_1 = 1$.
 Now, the microoperation for the indirect address condition can be symbolized by the R.T.L.

$$AR \leftarrow M[AR]$$

The three instruction types are subdivided into four separate paths. The selected operation is activated with the clock transition associated with timing signal T_3 .
 This can be symbolized as follows :

$$D_7'I T_3 : AR \leftarrow M[AR]$$

$$D_7'I T_3 : \text{Nothing}$$

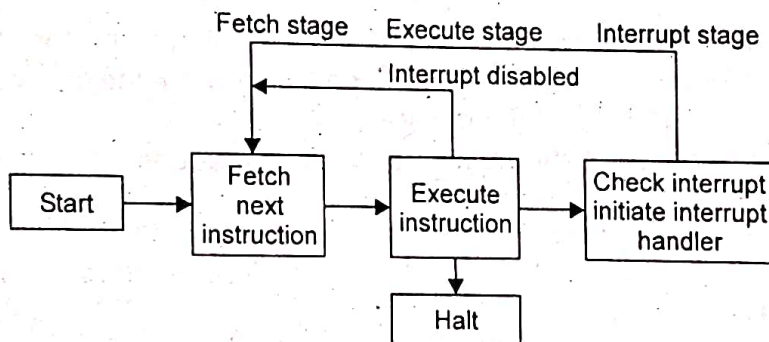
$$D_7'I T_3 : \text{Execute a register-reference instruction}$$

$$D_7'I T_3 : \text{Execute an I/O instruction.}$$

When a memory reference instruction with $I = 0$ is encountered, it is not necessary to do anything since the effective address is already in AR. However the sequence counter SC must be incremented when $D_7'T_3 = 1$, So that the execution of the memory reference instruction can be continued with timing variable T_4 . A register-reference or input-output instruction can be executed with the clock associated with timing signal T_3 . After the instruction is executed, SC is cleared to 0 and control returns to the fetch phase with $T_0 = 1$.

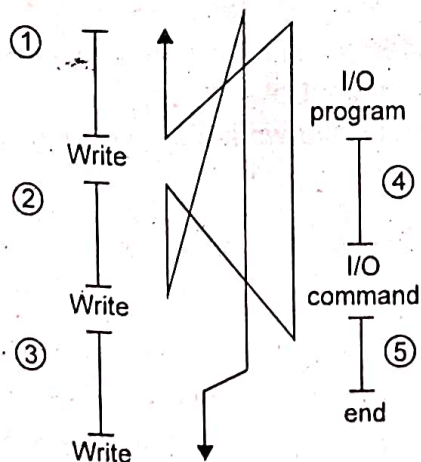
Note that the sequence counter SC is either incremented or cleared to 0 with every positive clock transition. We will adopt the convention that if SC is incremented, we will not write the statement $SC \leftarrow SC + 1$, but it will be implied that the control goes to the next timing signal in sequence. When SC is to be cleared, we will include the statement $SC \leftarrow 0$.

How the instruction cycle is to accommodate the interrupt from I/O devices :



I/O Interrupt generated by I/O controller, to signal normal completion of an operation or to signal a variety of error conditions. Most I/O devices are slower than processor without interrupts processor has to wait for device.

Program Flow :



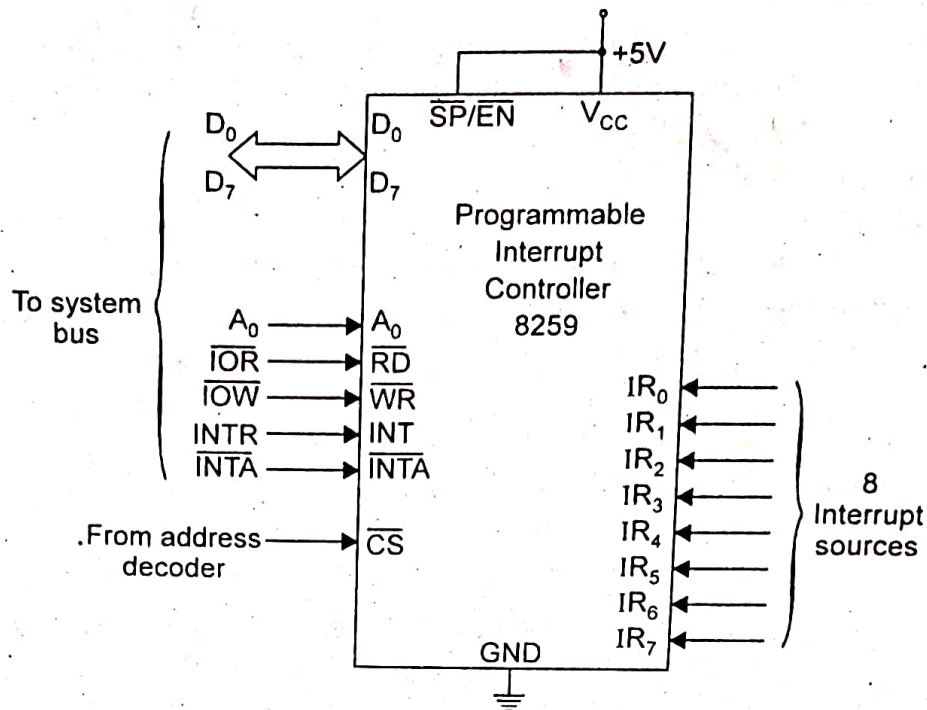
Processor checks the interrupts.
 It interrupt :

- ① Suspend execution of program
- ② Execution interrupt routine

Q 110. Design a parallel priority interrupt hardware for a system with eight interrupt sources.

(PTU, May 2014)

Ans.



Q 111. Define Micro-operation with example ?

(PTU, Dec. 2014)

Ans. **Micro-operations** : A micro-operation is a basic elementary operation defined on the data stored in registers.

We can also say that a microoperation is an elementary operation performed on the information stored in one or more registers. The result of the operation may replace the previous binary information of a register or may be transferred to another register.

The microoperations are classified into following four categories :

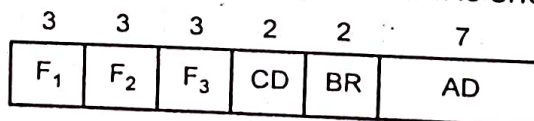
1. Arithmetic microoperation
 $R_3 \leftarrow R_1 + R_2$
2. Logic microoperation.
 $P : R_1 \leftarrow R_1 R_2$
3. Shift microoperation
 $R_1 \leftarrow \text{Shl } R_1$
 $R_2 \leftarrow \text{Shr } R_2$
4. Arithmetic logic shift unit.

Q 112. What are microinstructions ? Give the microinstruction format ?

(PTU, May 2015 ; Dec. 2014)

Ans. A microinstruction specifies one or more microoperation for the system and a sequence of microinstruction constitutes a microprogram, which resides in control memory.

Microinstruction format : The microinstruction format is shown in the following figure.



- F₁, F₂, F₃ : Microoperation fields.
- CD : Condition for branching
- BR : Branch field
- AD : Address field

Microinstruction code format (20 bits).

The 20 bits of the microinstruction are divided into four functional parts. The three fields F_1 , F_2 and F_3 specify microoperations for the computer.

CD stands for conditional branching that selects status bit conditions. The BR field specifies the type or branch to be use. The AD field contains a branch address. The address field is seven bits wide, since the control memory has $128=2^7$ words.

Q 113. What is bus arbitration and its types ?

(PTU, Dec. 2014)

Ans. Bus arbitration : The possibility exists that several master or slave units connected to a shared bus will request access to the bus at the same time. A selection mechanism called bus arbitration is therefore required to enable the current master which will still refer to a bus controller to decide among such competing request. Centralized and distributed are the two types of bus arbitration techniques.

In this we mention three types of bus arbitration

- Daisy chaining
- Polling
- Independent requesting

Q 114. How asynchronous data transfer is achieved with the help of handshaking method?

(PTU, May 2015)

Ans. In a computer system, CPU and an I/O interface are designed independently of each other. When internal timing in each unit is independent from the other and when registers in interface and registers of CPU uses its own private clock. In that case the two units are said to be asynchronous to each other. CPU and I/O device must coordinate for data transfers.

Methods used in asynchronous data transfer

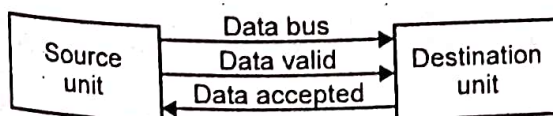
- Strobe control
- Handshaking.

Handshaking method is used to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data.

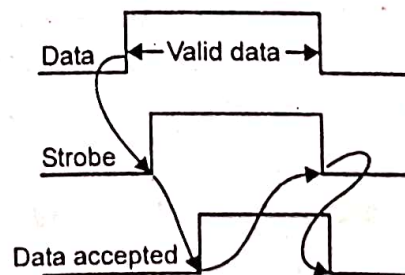
In case of source initiated data transfer under strobe control method, the source unit has no way of knowing whether destination unit has received the data or not. Similarly, destination initiated transfer has no method of knowing whether the source unit has placed the data on the data bus. Handshaking mechanism solve this problem by introducing a second control signal that provides a reply to the unit that initiate the transfer. There are two control lines in handshaking technique :

- Source to destination unit
- Destination to source unit.

Source Initiated Transfer : Handshaking signals are used to synchronize the bus activities. The two handshaking lines are data valid, which is generated by the source unit, and data accepted, generated by the destination unit. The timing diagram shows exchange of signals between two units.



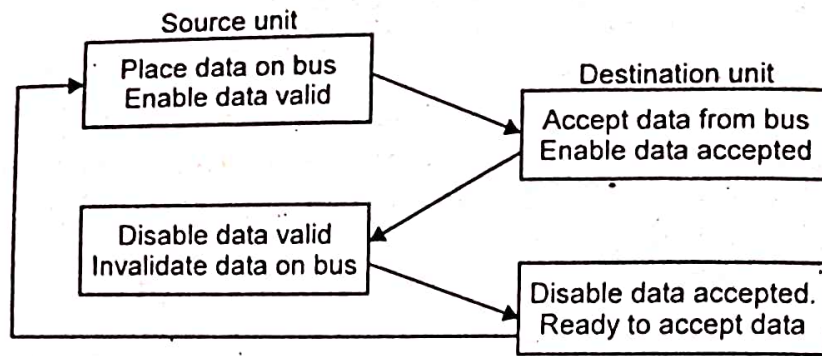
Block diagram



Timing diagram

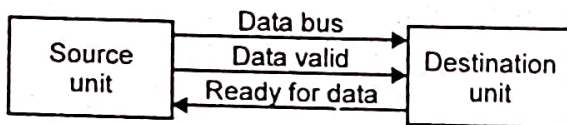
The sequence of events : The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal. The data accepted signal is activated by the destination unit after it

accepts the data from the bus. The source unit then disables its data valid signal, which invalidates the data on the bus. The destination unit then disables its data accepted signal and the system goes into its initial state.

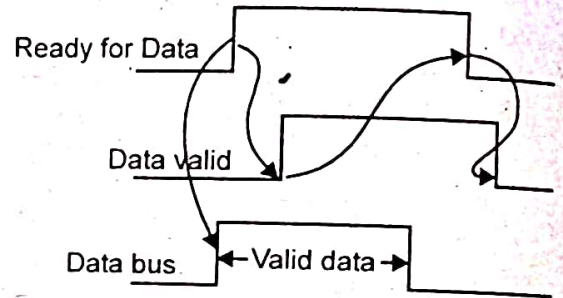


Sequence of events

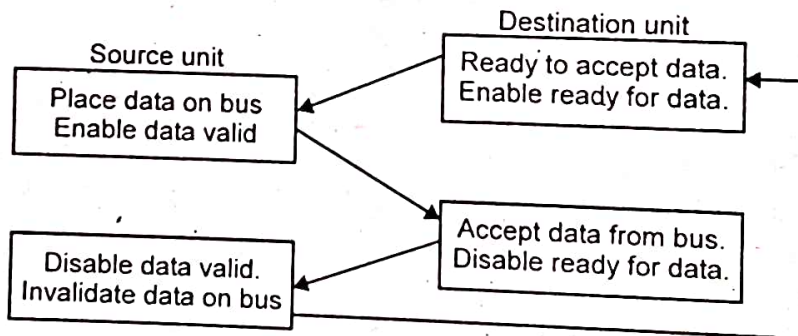
Destination initiated transfer using handshaking : In this case the name of the signal generated by the destination unit is ready for data. The source unit does not place the data on the bus until it receives the ready for data signal from the destination unit. The handshaking procedure follows the same pattern as in source initiated case. The sequence of events in both the cases is almost same except the ready for signal has been converted from data accepted in case of source initiated.



Block diagram



Timing diagram



Sequence of events

Q 115. What are the components of an I/O interface ?

(PTU, Dec. 2015)

Ans. I/O units consist of a mechanical part and an electronic part. The electronic part is often called the controller while the mechanical component is the I/O device itself.

I/O components : Functionally we can identify several components in an I/O device.

1. External mechanism : Within an I/O device there must be some kind of mechanism which is able to interact with the external medium to perform the I/O operations.

2. Electronic Control circuits : To control the external I/O mechanism requires electronic control circuit.

3. Interface circuit : The third main component within an I/O device are the interface circuits.

Microcomputer interfaces may be classified in the following two general categories :

- (i) External orientated interface
- (ii) Computer orientated interface

Q 116. Name various data transfer modes.

(PTU, Dec. 2015)

Ans. Different modes of data transfer are :

1. Programmed I/O
2. Interrupt initiated I/O
3. Direct memory access (DMA)

Q 117. What is hardwired control ? What are its advantages ?

(PTU, Dec. 2019)

Ans. Hardwired control is a control mechanism that generates control signals by using an appropriate control signals by using an appropriate finite state machine.

Advantages :

1. Faster mode of operation.
2. Faster than micro programmed control unit.
3. User in RISC processor
4. Inflexible to modify
5. It is also flexible as changes could be easily made to the design.
6. It simplifies the design of the control unit.

Q 118. What is Microprocessor ?

(PTU, Dec. 2019)

Ans. A microprocessor is an electronic component that is used by a computer to do its work. It is a central processing unit on a single integrated circuit chip containing millions of very small components including transistors, resistors and diodes that work together.

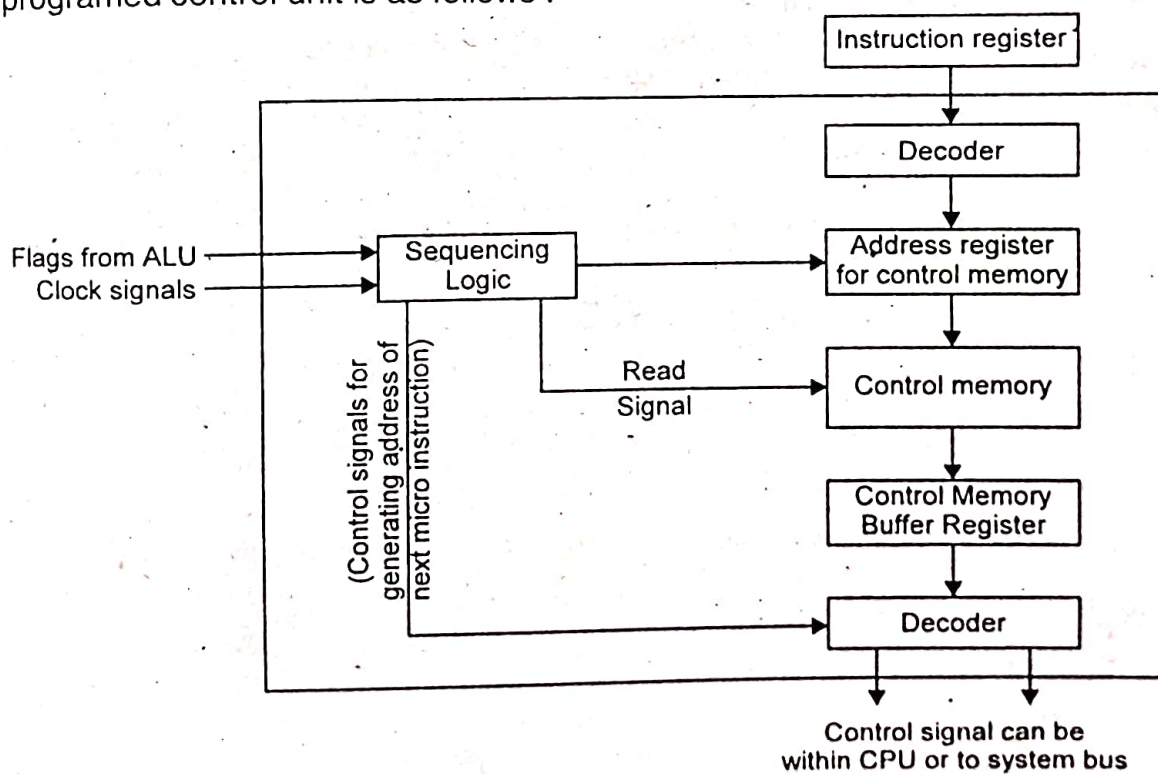
Q 119. Explain the design and working of a micro programmed control unit.

(PTU, Dec. 2019)

Ans. The logic of the control unit is specified by a micro-program. A micro program is also called firmware. It consists of :

(a) One or more micro operations to be executed and

(b) The information about the micro instruction to be executed next. The general configuration of a micro programmed control unit is as follows :



Operation of micro-programmed control unit

The micro-instructions are stored in the control memory. The address register for the control

memory contain the address of the next instruction that is to be read. The control memory Buffer Register receives the micro-instruction that has been read. A micro instruction execution primarily involves the generations of desired control signals and signals used to determine the next micro instruction to be executed. The sequencing logic section loads the control memory address register. It also issues a read command to control memory. The following functions are performed by the micro programmed control unit :

1. The sequence logic unit specifies the address of the control memory word that is to be read in the Address Register of the control memory. It also issues the READ signal.
 2. The desired control memory word is read into control memory Buffer Register.
 3. The content of control memory buffer register is decoded to create control signals and next address information for the sequencing logic unit.
 4. The sequencing logic unit finds the address of the next control word on the basis of the next address information from the decoder and the ALU flags.
- This decoder translates the opcode of the IR into a control memory address.

Q 120. Give five examples of external interrupt and five examples of internal interrupt.
(PTU, Dec. 2016)

Ans. External Interrupts : External interrupts come from I/O devices, from a timing device, from a circuit monitoring the power supply or from any other external resource.

Examples of external interrupts are :

1. User requests for process changes.
2. Any request from an Input/output device.
3. Elapsed time of an event.
4. Power failure.
5. Any error/event that changes the way that the computer works on any given set of instructions.

Internal Interrupts : Internal interrupts arise from illegal or erroneous use of an instruction or data. Internal interrupts are also called the traps.

Examples of internal interrupts :

1. When a program requires the operating system to read through a cache or buffer before stopping and becoming available for other tasks.
2. Register overflow
3. Attempt to divide by zero.
4. Stack overflow
5. Protection violation & invalid operation.

□□□

Chapter

3

Pipelining

Contents

Basic concept of pipelining, throughput and speedup, pipeline hazards.

Parallel Processors : Introduction to parallel processors, Concurrent access to memory and cache coherency.

POINTS TO REMEMBER



- ✎ Pipelining is the technique of decomposing a sequential process into suboperations, with each suboperation beginning executed in a special dedicated segment that operate concurrently with all the other segments.
- ✎ A super scalar processor has a form of parallelism on a single chips allowing the system as a whole to run much faster that it would otherwise be able to at a given clock speed. A superscalar processor fetches, executes and returns results from more than one instruction during a single pipeline stage.
- ✎ A scalar processor processes one data item at a time on the other hand. In a vector processor single instruction operates simultaneously on multiple data items.
- ✎ A computer with vector instructions and pipelined floating point arithmetic operations is referred to as a supercomputer.
- ✎ An array processor is a processor that performs computations on large arrays of data. The term is used to refer two different types of processors. An attached array processor is an auxillary processor attached to a general purpose computer.
- ✎ Parallel processing is a term used to denotes a large class of techniques that are used to provide simultaneous data processing tasks for the purpose of increasing the computational speed of a computer system.
- ✎ The purpose of parallel processing is to speed up the computer processing capability and increase its throughput, that is, the amount of processing that can be accomplished during a given interval of time.
- ✎ A multiprocessor system is the interconnection of two or more CPUs with memory and input output equipment.
- ✎ In a synchronous bus, each data item is tranferred during a time slice known in advance to both source and destination. Synchronization is achieved by during both units from a common clock source.
- ✎ In an asynchronous bus, each data item being transferred is accomplished by control signals to indicate when the data is transfer from the source and received by destination.
- ✎ Abbreviations :
 - SISD** : Single Instruction Stream, single data stream.
 - SIMD** : Single Instruction Stream, Multiple Data Stream.
 - MISD** : Multiple Instruction Stream, Single Data Stream.

MIMD : Multiple Instruction Stream, Multiple Data Stream.

MDH : Message Passing Distributed Memory.

DSH : Distributed Shared Memory.

☞ **Mutual Exclusion** : A properly functioning multi processor system must provide a mechanism that will guarantee overly access to shared memory and other shared resources. This necessary to protect data from being changed simultaneously by two or more processors. This mechanism has been termed as 'Mutual Exclusion'.

QUESTION-ANSWERS

Q 1. How pipeline helps in a faster execution of a instruction?

(PTU, May 2004)

Ans. The instructions are exected in stages in pipeline processing. When an instruction completes a stage, another instruction can move to the stage. This way instructions can be executed in parallel, and results in faster execution.

Q 2. What is super pipelining ?

(PTU, Dec. 2006)

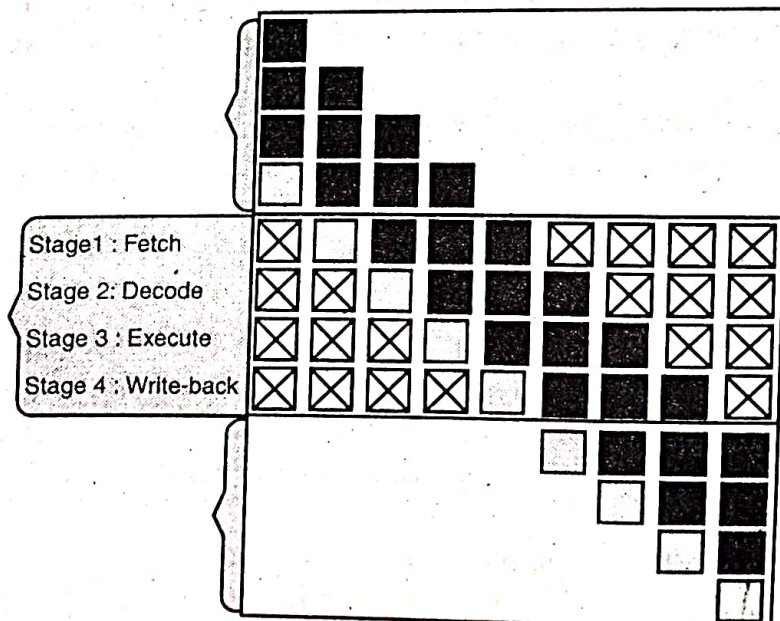
Ans. Pipelining is an effective way of organizing 11 activity in Computer System. An super instruction pipeline can overlap the processing of up or multiple instruction, so it desirable to use more than two stages to maximize the Instruction throughout. The value of multiple depends on the maximum number of stages into which instruction processing can be efficiently broken. The no. of pipeline stages range from 3 to dozen or more.

Q 3. What is instruction pipelining?

(PTU, May 2018 ; Dec. 2019, 2008)

Ans. An **instruction pipeline** is a technique used in the design of computers and other digital electronic devices to increase their instruction throughput (the number of instructions that can be executed in a unit of time).

The fundamental idea is to split the processing of a computer instruction into a series of independent steps, with storage at the end of each step. This allows the computer's control circuitry to issue instructions at the processing rate of the slowest step, which is much faster than the time needed to perform all steps at once. The term pipeline refers to the fact that each step is carrying data at once (like water), and each step is connected to the next (like the links of a pipe).



Generic 4-stage pipeline ; the colored boxes represent instructions independent of each other

Q 4. Explain pipelining in CPU design.

(PTU, Dec. 2006)

Ans. A pipeline is set of data processing elements connected in series, so that the output of one element is the input of the next one. The elements of a pipeline are often executed in 11 or in time-sliced fashion; in that case, some amount of buffer storage is often inserted between elements computer related pipelines includes :

- Instruction Pipeline
- Arithmetic Pipeline
- Graphics Pipeline

Q 5. What is pipeline processor ? What are advantages of it ?

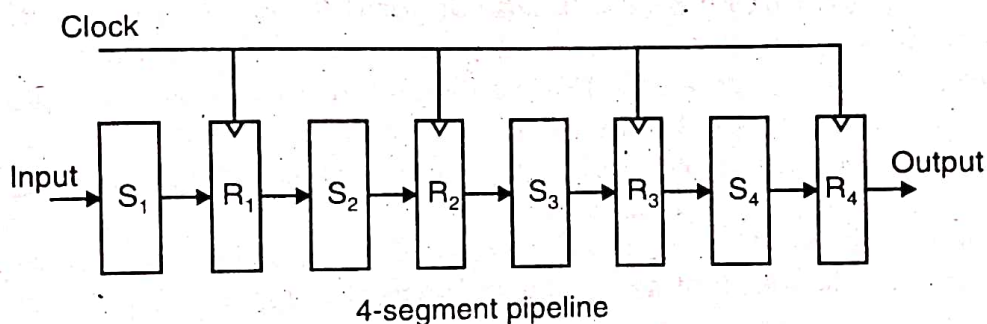
(PTU, Dec. 2019, 2014 ; May 2004)

Ans. Pipeline processor is a processor which involves pipelining technique for executing instructions. Pipelining is a technique and decomposing a sequential process into sub-operations, with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments.

Pipeline involves a collection of processing segments through which binary information flows. The result obtained from each segment is transferred to the next segment in the pipeline. Final result is obtained when data have passed through all the segments.

It is composed of registers which holds the data and also combinatorial circuits that performs suboperations in particular segment.

Information flows between adjacent stages under the control of a common clock applied to the registers simultaneously.



Basically, there are two types of pipelining.

1. Arithmetic Pipeline :- It divides an arithmetic operation into sub-operations for execution in pipeline segment.

2. Instruction Pipeline :- It operates on a stream of instructions by overlapping the fetch, decode and execute phases of instruction cycle.

Advantages :

1. Several computations can be performed simultaneously in distinct segments. so, it increases the speed.
2. It is efficient for those applications that need to repeat same task many times with different sets of data.
3. They are used to implement floating-point operations. so, useful in very high speed computers.
4. They are used in scientific problems as they can perform complex tasks at faster speed.
5. They are basically used for performing complex instructions.

Q 6. Determine the methods of handle branches in a pipeline instruction execution unit.

(PTU, May 2004)

Ans. A branch instruction can be of two types conditional or unconditional. An unconditional

branch always alters the sequential program flow by loading the program counter with the target address. In a conditional branch, the control selects the target instruction if the condition is satisfied or the next sequential instruction if not satisfied. Now, following methods are used to handle branch instructions.

1. Prefetch target instruction : One way of handling a conditional branch is to prefetch the target instruction in addition to the instruction following the branch. Both are saved until the branch is executed. If the branch condition is successful, the pipeline continues from the branch target instruction.

2. Branch Target Buffer : The branch target buffer (BTB) is an association memory included in the fetch segment of the pipeline. Each entry in BTB consists of the address of a previously executed branch instruction and the target instruction for that branch. It also stores the next few instructions after the branch target instruction. When the pipeline decodes a branch instruction, it searches the associative memory BTB for the address of the instruction. If it is in BTB, the instruction is available directly and prefetch conditions from the new path.

3. Loop Buffer : This is a small very high speed register file maintained by the instruction fetch segment of the pipeline. When a program loop is detected in the program, it is stored in the loop buffer.

4. Branch predictions : A pipeline with branch prediction uses some additional logic to guess the outcome of a conditional branch instruction before it is executed.

5. Delayed branch : It is a procedure employed in most RISC processors. In this procedure, the compiler detects the branch instructions and rearranges the machine language code sequence by inserting useful instructions that keep the pipeline operating without interruptions.

e.g. Insertion of a no-operation instruction after a branch instruction.

Q 7. What causes a processor pipeline to be underpipelined ?

(PTU, May 2006)

Ans. The causes of a processor pipeline to be underpipelined are as follows :

1. A pipelined processor with a plurality of execution pipelines each with a plurality of pipe stages, where each of the pipe stages has associated with it state information, and where instructions issued into an execution pipeline are processed in each pipe stage, thereby altering the associated state information for that pipe stage, comprising :

- (a) at least first and second execution pipelines with respective corresponding pipe stages ;
- (b) instruction issue circuitry that issues first and second sequences of instructions respectively into the first and second execution pipelines, with at least some of the instructions in the first sequence of instructions being issued into the first execution pipeline substantially simultaneously with instructions in the second sequence of instructions being issued into the second execution pipeline ; and
- (c) pipe control circuitry that monitors the execution of instructions issued into at least the first and second execution pipelines, including the state information associated with the respective corresponding pipe stages of the first and second execution pipelines ;
- (d) in response to such state information, the pipe control circuitry controlling the flow of the first and second sequences of instructions between the respective corresponding pipe stages of the first and second execution pipelines such that an instruction in the first execution pipeline moves from a current pipe stage to a next pipe stage independent of the movement of instructions in corresponding pipe stages of the second execution pipeline.

2. The pipelined processor of claim 1, where in the first and second sequences of instructions include respectively first and second instructions issued substantially simultaneously into respectively the first and second execution pipelines, with the first instruction being senior, and with the second instruction having a data dependency on the first instruction, and wherein the associated state information for a corresponding pipe stage of the second execution pipeline that is a current pipe stage for the

second instruction includes an indication of whether data dependency must be resolved to process the second instruction in the current pipe stage.

3. The pipelined processor of claim 1, wherein, for at least first and second corresponding pipe stages of respectively the first and second execution pipelines, the first and second sequences of instructions include both instructions that complete processing in a single clock and instructions that complete processing in more than one clock, and wherein the associated state information for the first and second pipe stages includes an indication of whether a current instruction in either the first or second corresponding pipe stage will require more than one clock to complete processing.

4. A pipelined processor with a plurality of execution pipelines each with a plurality of stages in which instructions issued into an execution pipeline are processed, thereby altering processor state, comprising:

- (a) instruction issue means for independently issuing first and second instructions respectively into the first and second execution pipelines, with the first instructions being senior, and with the second instruction having a data dependency on the first instruction, where data dependency refers to a read-after-write data hazard; and
- (b) pipe control means for monitoring the execution of instructions issued into at least the first and second execution pipelines, including data dependencies between issued instructions, so as to control the flow of instructions between the stages of such execution pipelines;
- (c) the pipe control means controlling the flow of the first and second instructions between the stages of respective execution pipelines such that the second instruction is not delayed due to the data dependency on the first instruction unless the data dependency must be resolved for proper processing of the second instruction in its current stage.

5. The pipelined processor of claim 4, wherein instructions reference a set of logical registers, and wherein the processor includes a register file with a greater number of physical registers than the number of logical registers, further comprising:

- (a) register translation means for selectively implementing register renaming by mapping a logical register referenced by an instruction to a physical register in the register file;
- (b) the instruction issue means independently issuing third and fourth instructions respectively into the first and second execution pipelines, with the third instruction being senior, and with the fourth instruction having a write-after-read or a write-after-write dependency on the third instruction; and
- (c) the register translation means using register renaming to eliminate such dependency for the fourth instruction.

6. The pipelined processor of claim 4, wherein the read-after-write data dependency results from the second instruction referencing a source register that is the same as a destination register of the first instruction, and wherein the pipe control means includes forwarding means for selectively forwarding data from the first instruction to the second instruction by modifying a source register referenced by the second instruction to eliminate the data dependency between the first and second instructions.

7. The pipelined processor of claim 4, wherein the issue means comprises an instruction decoder, and wherein each execution pipeline includes at least address calculation and execution stages.

8. A method of issuing and controlling the flow of instructions in a pipelined processor with at least first and second execution pipelines with respective corresponding pipe stages, where each of the pipe stages has associated with it state information, and where instructions issued into an execution pipeline are processed in each pipe stage, thereby altering the associated state information for that pipe stage, comprising the steps:

- (a) issuing first and second sequences of instructions respectively into the first and second execution pipelines, with at least some of the instructions in the first sequence of instructions being issued into the first execution pipeline substantially simultaneously with instructions in the second sequence of instructions being issued into the second execution pipeline ; and
- (b) monitoring the execution of instructions issued into at least the first and second execution pipelines, including the state information associated with the respective corresponding pipe stages of the first and second execution pipelines ; and
- (c) in response to such state information, controlling the flow of the first and second sequences of instructions between the respective corresponding pipe stages of the first and second execution pipelines such that an instruction in the first execution pipeline moves from a current pipe stage to a next stage independent of the movement of instructions in corresponding pipe stages of the second execution pipeline.

9. The method of claim 8, wherein instructions reference a set of logical registers, and wherein the processor includes a register file with a greater number of physical registers than the number of logical registers, further comprising the steps :

- (a) selectively implementing register renaming by mapping a logical register referenced by an instruction to a physical register in the register file ;
- (b) independently issuing third and fourth instructions respectively into the first and second execution pipelines, with the third instruction being senior, and with the fourth instruction having a write-after-read or a write-after-write dependency on the third instruction ; and
- (c) renaming a destination register of the fourth instruction to be a different physical register than a physical register that is a destination register of the third instruction, thereby eliminating such dependency for the fourth instruction.

10. The method of claim 8, wherein the read-after-write data dependency results from the second instruction referencing a source register that is the same as a destination register of the first instruction, further comprising the step of : selectively forwarding data from the first instruction to the second instruction by modifying a source register referenced by the second instruction to eliminate the data dependency between the first and second instructions.

11. The method of claim 8, wherein the associated state information for the corresponding pipe stages of the first and second execution pipelines includes an indication of whether a data dependency between first and second instructions has resolved, further comprising the steps of : issuing the first and second instructions issued substantially simultaneously into respectively the first and second execution pipelines, with the first instruction being senior, and with the second instruction having a data dependency on the first instruction, and then controlling the movement of the second interaction from the current pipe stage to a next pipe stage dependent on whether the data dependency must be resolved to process the second instruction in the current pipe stage.

12. The method of claim 8, wherein, for at least first and second corresponding pipe stages of respectively the first and second execution pipelines, the first and second sequences of instructions include both instructions that complete processing in a single clock and instructions that complete processing in more than one clock, and wherein the associated state information for the first and second pipe stages includes an indication of whether a current instruction in either the first or second corresponding pipe stages will require more than one clock to complete processing.

Q 8. What are the different types of hazards in case of instruction pipeline?

(PTU, May 2011)

Ans. ● Instruction hazard

- Data hazard
- Structural hazard
- Control hazard.

Q 9. What is meant by superscalar processor ? Explain the concept of Pipeline in superscalar processor ? (PTU, May 2010, 2009, Dec. 2006)

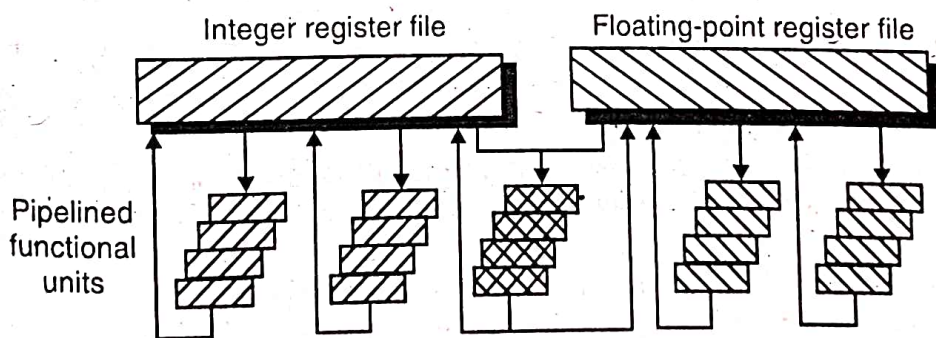
Ans. The term superscalar, first coined in 1987 (AGER 87), refers to a machine that is designed to improve the performance of the execution of scalar instructions. In most appreciations, the bulk of the operations are on scalar quantities. Accordingly, the superscalar approach represents the next step in the evolution of high-performance general-purpose processors.

The essence of the superscalar approach is the ability to execute instructions independently in different pipelines.

The concept can be further exploited by allowing instructions to be executed in an order different from the program order. Fig. shows, in general terms the superscalar approach. There are multiple functional units, each of which is implemented as a pipeline, which support parallel execution of several instructions. In this example, two integer, two floating-point, and one memory (either load or store) operations can be executing at the same time.

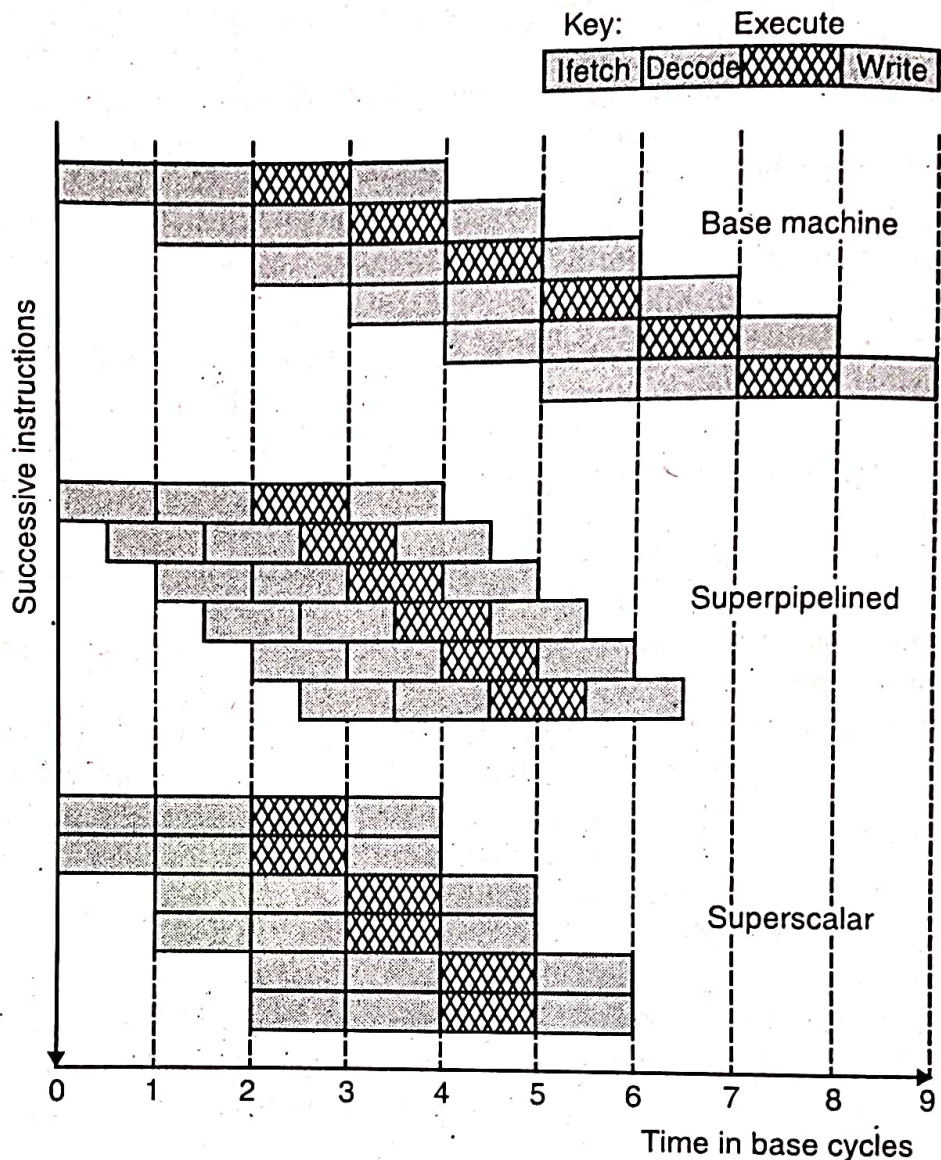
Many researchers have investigated superscalar-like processors and their research indicates that some degree of performance improvement is possible. present the reported performance advantages. The differences in the results arise from differences both in the hardware of the simulated machine and in the applications being simulated.

Superscalar Versus Superpipelined : An alternative approach to achieving greater performance is referred to as super pipelining, a term first coined in 1998 (JOUP 88) . Superpipelining exploits the facts that many pipeline stages perform tasks that require less than half a clock cycle. Thus, a doubled internal clock speed allows the performance of two tasks in one external clock cycle. We have seen one example of this approach with the MIPS Rs. 4000 Clock Cycle. We have seen one example of this approach with the MIPS Rs. 4000. Compares the two approaches. The base pipelines issues one instruction per clock cycle and can perform one pipelines stage per clock cycle. The pipeline has four stages. instruction fetch, operation decode, operation execution and result write back. The execution stage is cross hatched for clarity. Nowthan although several instrutions are executing concurrently, only one instruction its execution stage at anyone time.



The next part of the diagram shows a super pipelined implementation that is capable of performing two pipeline stage per clock cycle. An alternative way looking at this is that the function performed in each stage can be split into monoverlapping parts and each execute in half a clock cycle. An Superline implementantation that behaves in this fashion is said to be of degree 2. Finally the lowest part of the diagram shows a superscalar implementation capable of executors two instances of each stage in parallel. Higher-degree superpipeline and superscalar implementations are of course possible.

Both the superpipeline and the superscalar implementations depicted in have the same number of instructions executing at the same time in steady state. The Superpipelined processor falls behind the superscalar processor the start of the program and at each branch target.



Q 10. How pipelining would improve the performance of CPU? Justify. (PTU, Dec. 2008)

Ans. Pipelining is a technique used to improve the execution throughput of a CPU by using the processor resources in a more efficient manner.

The basic idea is to split the processor instructions into a serial of small independent stages. Each stage is designed to perform a certain part of the instruction. At a very basic level, these stages can be broken down into :

- | | |
|--|---------------------------------------|
| <input type="checkbox"/> Fetch Unit | Fetch an instruction from memory |
| <input type="checkbox"/> Decode Unit | Decode the instruction to be executed |
| <input type="checkbox"/> Execute Unit | Execute the instruction |
| <input type="checkbox"/> Write Unit | |

There will be a dedicated CPU module for each of the stages mentioned above.

On a non-pipelined CPU, when an instruction is being processed at a particular stage, the other stages are at an idle state – which is very inefficient. If you look at the diagram, when the 1st instruction is being decoded, the Fetch, Execute and Write Units of the CPU are not being used and it takes 8 clock cycles to execute the 2 instructions.

Pipelining

On the other hand, on a pipelined CPU, all the stages work in parallel. When the 1st instruction is being decoded by the Decoder Unit, the 2nd instruction is being fetched by the Fetch Unit. It only takes 5 clock cycles to execute 2 instructions on a pipelined CPU.

Instruction	1				2			
Fetch	█				█			
Decode		█				█		
Execute			█				█	
Write				█				█
Clock	1	2	3	4	5	6	7	8

Non-Pipelined

Instruction	1	2			
Fetch	█	█			
Decode		█	█		
Execute			█	█	
Write				█	█
Clock	1	2	3	4	5

Pipelined

Note that increasing the number of stages in the pipeline will not always result in an increase of the execution throughput. On a non-pipelined CPU, an instruction could only take 3 cycles, but a pipelined CPU it could take 4 cycles because of the different stages involved. Therefore, a single instruction might require more clock cycles to execute on a pipelined CPU. But the time taken to complete the execution of multiple instructions gets faster in pipelined CPUs.

Q 11. Compare and Contrast Super pipelined machine and Super scalar machines.
(PTU, Dec. 2008)

Ans. An **superpipeline** is a technique used in the design of computers and other digital electronic devices to increase their instruction throughput (the number of instructions that can be executed in a unit of time).

The fundamental idea is to split the processing of a computer instruction into a series of independent steps, with storage at the end of each step. This allows the computer's control circuitry to issue instructions at the processing rate of the slowest step, which is much faster than the time needed to perform all steps at once. The term pipeline refers to the fact that each step is carrying data at once (like water), and each step is connected to the next (like the links of a pipe).

Most modern CPUs are driven by a clock. The CPU consists internally of logic and memory (flip flops). When the clock signal arrives, the flip flops take their new value and the logic then requires a period of time to decode the new values. Then the next clock pulse arrives and the flip flops again take their new values, and so on. By breaking the logic into smaller pieces and inserting flip flops between the pieces of logic, the delay before the logic gives valid outputs is reduced. In this way the clock period can be reduced. For example, the RISC pipeline is broken into five stages with a set of flip flops between each stage.

1. Instruction fetch
2. Instruction decode and register fetch

3. Execute
4. Memory access
5. Register write back

Advantages and Disadvantages

Pipelining does not help in all cases. There are several possible disadvantages. An instruction pipeline is said to be fully pipelined if it can accept a new instruction every clock cycle. A pipeline that is not fully pipelined has wait cycles that delay the progress of the pipeline.

Advantages of Pipelining :

1. The cycle time of the processor is reduced, thus increasing instruction issue-rate in most cases.
2. Some combinatorial circuits such as address or multipliers can be made faster by adding more circuitry. If pipelining is used instead, it can save circuitry vs. a more complex combinatorial circuit.

Disadvantages of Pipelining :

1. A non-pipelined processor executes only a single instruction at a time. This prevents branch delays (in effect, every branch is delayed) and problems with serial instructions being executed concurrently. Consequently the design is simpler and cheaper to manufacture.
2. The instruction latency in a non-pipelined processor is slightly lower than in a pipelined equivalent. This is due to the fact that extra flip flops must be added to the data path of a pipelined processor.
3. A non-pipelined processor will have a stable instruction bandwidth. The performance of a pipelined processor is much harder to predict and may vary more widely between different programs.

A **superscalar** CPU architecture implements a form of parallelism called instruction-level parallelism within a single processor. It thereby allows faster CPU throughput than would otherwise be possible at the same clock rate. A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor. Each functional unit is not a separate CPU core but an execution resource within a single CPU such as an arithmetic logic unit, a bit shifter, or a multiplier.

While a superscalar CPU is typically also pipelined, they are two different performance enhancement techniques. It is theoretically possible to have a non-pipelined superscalar CPU or a pipelined non-superscalar CPU.

The superscalar technique is traditionally associated with several identifying characteristics. Note these are applied within a given CPU core.

- Instructions are issued from a sequential instruction stream.
- CPU hardware dynamically checks for data dependencies between instructions at run time (versus software checking at compile time).
- Accepts multiple instructions per clock cycle.

Q 12. What are the limits on how much a processor's performance can be improved using pipelining ? (PTU, Dec. 2004)

OR

How does pipelining improve performance ?

(PTU, Dec. 2009, 2005)

OR

How can you evaluate the performance of processor architecture?

(PTU, May 2009)

Ans. The limits on which processor's performance can be improved using pipelining are as follows :

(i) Speed up : The speed up of a pipeline processing over an equivalent non pipeline processing is defined as the ratio of time taken by the non pipeline system to the time taken by pipeline system on the same operation.

$$S = \frac{n + n}{(m + (n - 1)) p}$$

(ii) Cycle Time of Pipeline Processors : The cycle time of pipeline processor is dependent on four factors ÷ the cycle time of the unpipelined diversion of the processor, the number of pipeline stages, how evenly the datapath logic divided among the stages, and the latency of the pipeline registers. If the logic can be divided evenly among the pipeline stages, the clock period of the pipelined processor is

$$\text{Cycle Time Pipelined} = \frac{\text{Cycle Time Unpipelined}}{\text{No. of Pipeline Stages}} + \text{Pipeline Line Latency}$$

(iii) Pipeline Latency : The latency of a pipeline is the amount of time that a single Instruction takes to pass through the pipeline, which is the product of number of pipeline stages and clock cycle time.

$$\text{Pipeline Latency} = \text{Pipeline Stages} \times \text{Clock Cycle Time.}$$

(iv) Cycle Per Instruction : It is the number of clock cycles required to execute each Instruction, known as cycles per Instruction. The CPI of a given program on a given system is calculated by dividing the number of clock cycles required to execute the program by the number of Instructions executed in running the program.

Q 13. Write short note on Hazards of pipelining.

(PTU, May 2009)

Ans. Hazards of pipelining : When a programmer (or compiler) writes assembly code, they make the assumption that each instruction is executed before execution of the subsequent instruction is begun. This assumption is invalidated by pipelining. When this causes a program to behave incorrectly, the situation is known as a hazard. Various techniques for resolving hazards such as forwarding and stalling exist.

A non-pipeline architecture is inefficient because some CPU components (modules) are idle while another module is active during the instruction cycle. Pipelining does not completely cancel out idle time in a CPU but making those modules work in parallel improves program execution significantly.

Processors with pipelining are organized inside into stages which can semi-independently work on separate jobs. Each stage is organized and linked into a 'chain' so each stage's output is fed to another stage until the job is done. This organization of the processor allows overall processing time to be significantly reduced.

Unfortunately, not all instructions are independent. In a simple pipeline, completing an instruction may require 5 stages. To operate at full performance, this pipeline will need to run 4 subsequent independent instructions while the first is completing. If 4 instructions that do not depend on the output of the first instruction are not available, the pipeline control logic must insert a stall or wasted clock cycle into the pipeline until the dependency is resolved. Fortunately, techniques such as forwarding can significantly reduce the cases where stalling is required. While pipelining can in theory increase performance over an unpipelined core by a factor of the number of stages (assuming the clock frequency also scales with the number of stages), in reality, most code does not allow for ideal execution.

Disadvantages of Pipelining :

1. A non-pipelined processor executes only a single instruction at a time. This prevents branch

- delays (in effect, every branch is delayed) and problems with serial instructions being executed concurrently. Consequently the design is simpler and cheaper to manufacture.
- The instruction latency in a non-pipelined processor is slightly lower than in a pipelined equivalent. This is due to the fact that extra flip flops must be added to the data path of a pipelined processor.
 - A non-pipelined processor will have a stable instruction bandwidth. The performance of a pipelined processor is much harder to predict and may vary more widely between different programs.

Q 14. How many clock cycles are required to process 100 tasks in five segment pipeline?
(PTU, May 2017, 2015 ; Dec. 2011)

Ans. Number of clock cycles = number of segments (k) + (number of tasks (n) + 1)

$$= k + (n - 1)$$

$$= 5 + (100 - 1)$$

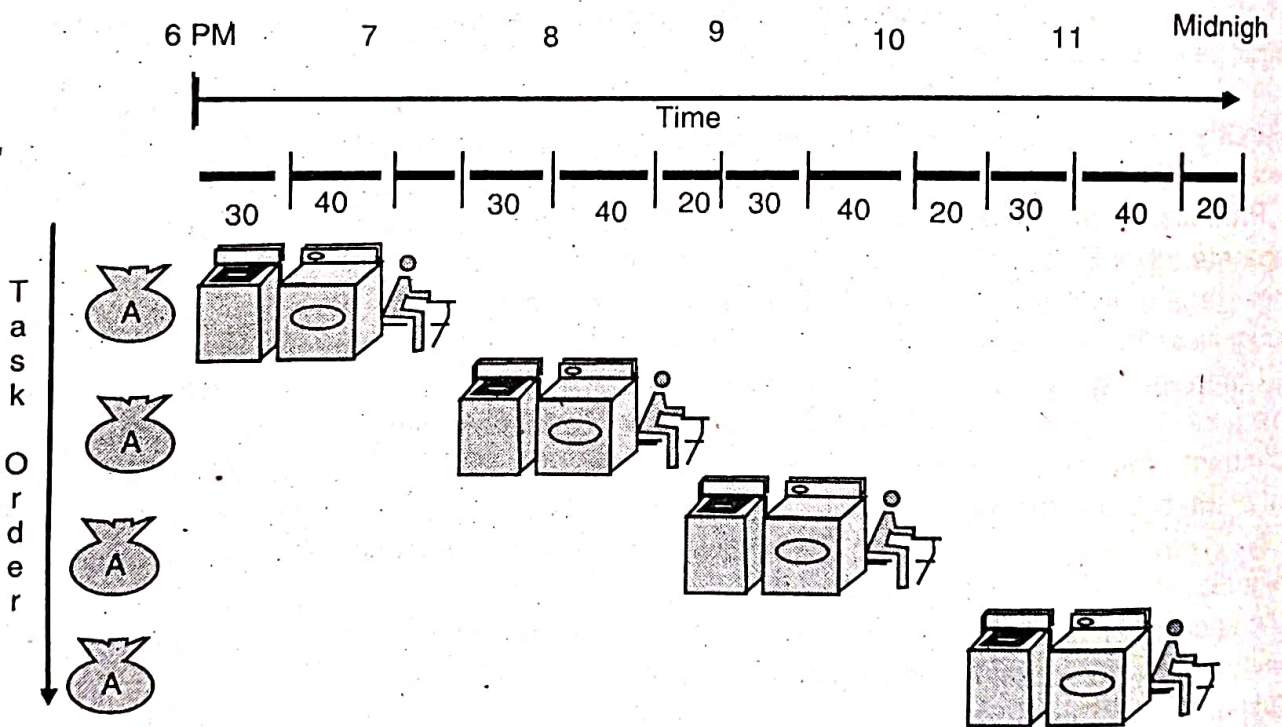
$$= 5 + 99$$

$$= 104 \text{ cycles.}$$

Q 15. What are the reasons of Pipe-Line conflicts in a Pipe Lined processor? How are they resolved?
(PTU, Dec. 2016 ; May 2008)

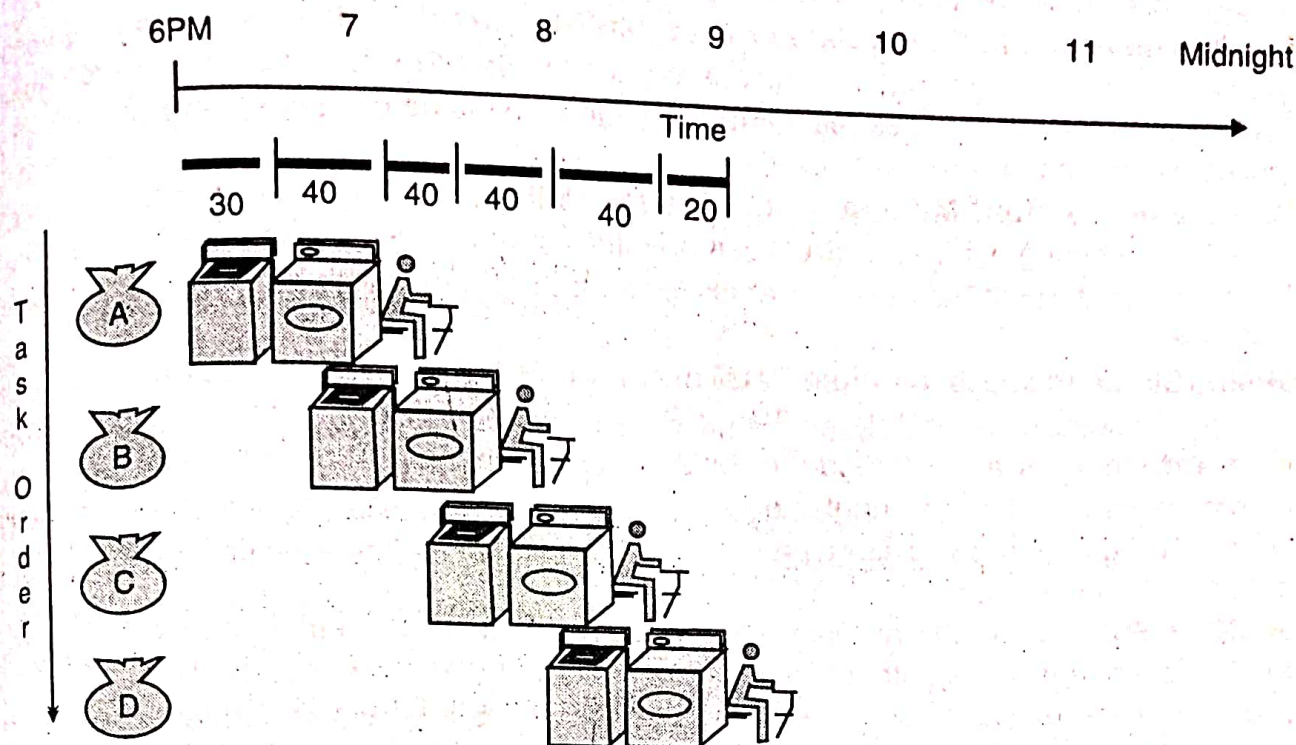
Ans. Pipelining, a standard feature in RISC processors, is much like an assembly line. Because the processor works on different steps of the instruction at the same time, more instructions can be executed in a shorter period of time.

A useful method of demonstrating this is the laundry analogy. Let's say that there are four loads of dirty laundry that need to be washed, dried, and folded. We could put the first load in the washer for 30 minutes, dry it for 40 minutes, and then take 20 minutes to fold the clothes. Then pick up the second load and wash, dry, and fold, and repeat for the third and fourth loads. Supposing we started at 6 PM and worked as efficiently as possible, we would still be doing laundry until midnight.



However, a smarter approach to the problem would be to put the second load of dirty laundry into the washer after the first was already clean and whirling happily in the dryer. Then, while the first

As the first load is being folded, the second load would dry, and a third load could be added to the pipeline of laundry. Using this method, the laundry would be finished by 9 : 30.



RISC Pipelines : A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor :

1. fetch instruction from memory
2. read registers and decode the instruction
3. execute the instruction or calculate an address
4. access an operand in data memory
5. write the result into a register.

Q 16. Explain about MIMD Machines.

(PTU, May 2006)

Ans. MIMD : It stands for multiple instruction multiple data stream. These are multiprocessors and multicomputer systems and unpracticable.

Q 17. Compare SPMD and MIMD machine.

(PTU, May 2008 ; Dec. 2013, 2007)

Ans.

Flynn's taxonomy

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

The four classifications defined by Flynn are based upon the number of concurrent instruction (or control) and data streams available in the architecture :

Single Instruction, Single Data stream (SISD) : A sequential computer which exploits no parallelism in either the instruction or data streams. Examples of SISD architecture are the traditional uniprocessor machines like a PC or old mainframes.

Single Instruction, Multiple Data streams (SIMD) : A computer which exploits multiple data streams against a single instruction stream to perform operations which may be naturally parallelized. For example, an array processor or GPU.

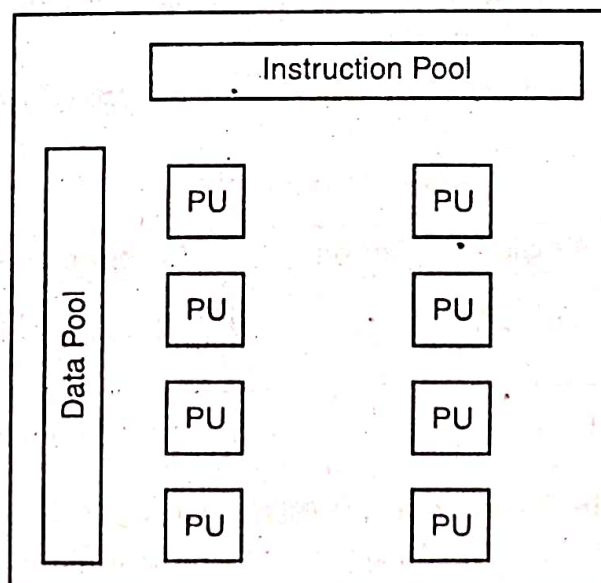
Multiple Instruction, Single Data stream (MISD) : Multiple instructions operate on a single data stream. Uncommon architecture which is generally used for fault tolerance. Heterogeneous systems operate on the same data stream and must agree on the result. Examples include the Space Shuttle flight control computer.

Multiple Instruction, Multiple Data streams (MIMD) : Multiple autonomous processors simultaneously executing different instructions on different data. Distributed systems are generally recognized to be MIMD architectures ; either exploiting a single shared memory space or a distributed memory space.

SPMD (Single Process, Multiple Data) or (Single Program, Multiple Data) : In computing, **SPMD** (Single Process, Multiple Data) or (Single Program, Multiple Data) is a technique employed to achieve parallelism ; it is a subcategory of MIMD. Tasks are split up and run simultaneously on multiple processors with different input in order to obtain results faster. SPMD is the most common style of parallel programming. It is also a prerequisite for research concepts such as active messages and distributed shared memory.

SPMD vs SIMD : In SPMD, multiple autonomous processors simultaneously execute the same program at independent points, rather than in the lockstep that SIMD imposes on different data. With SPMD, tasks can be executed on general purpose CPUs ; SIMD requires vector processors to manipulate data streams. Note that the two are not mutually exclusive.

MIMD (Multiple Instruction stream, Multiple Data stream) : In computing, **MIMD** (Multiple Instruction stream, Multiple Data stream) is a technique employed to achieve parallelism. Machines using MIMD have a number of processors that function asynchronously and independently. At any time, different processors may be executing different instructions on different pieces of data. MIMD architectures may be used in a number of application areas such as computer-aided design/computer-aided manufacturing, simulation, modeling and as communication switches. MIMD machines can be of either shared memory or distributed memory categories. These classifications are based on how MIMD processors access memory. Shared memory machines may be of the bus-based, extended, or hierarchical type. Distributed memory machines may have hypercube or mesh interconnection schemes.



Q 18. What is a multiprocessor? Explain the term SIMD. (PTU, May 2010 ; Dec. 2008)

Ans. Multiprocessing is the use of two or more central processing unit (CPUs) within a single computer system. The term also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them.

Multiprocessing sometimes refers to the execution of multiple concurrent software processes in a system as opposed to a single process at any one instant. However, the terms multitasking or multiprogramming are more appropriate to describe this concept, which is implemented mostly in software, whereas multiprocessing is more appropriate to describe the use of multiple hardware CPUs. A system can be both multiprocessing and multiprogramming, only one of the two, or neither of the two.

Flynn's taxonomy

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

SIMD multiprocessing

SIMD multiprocessing is well suited to parallel or vector processing, in which a very large set of data can be divided into parts that are individually subjected to identical but independent operations. A single instruction stream directs the operation of multiple processing units to perform the same manipulations simultaneously on potentially large amounts of data.

For certain types of computing applications, this type of architecture can produce enormous increases in performance, in terms of the elapsed time required to complete a given task. However, a drawback to this architecture is that a large part of the system falls idle when programs or system tasks are executed that cannot be divided into units that can be processed in parallel.

SIMD multiprocessing finds wide use in certain domains such as computer simulation, but is of little use in general-purpose desktop and business computing environments.

Q 19. Explain in brief about SPMD machines. (PTU, Dec. 2004)

Ans. Single Program Multiple Data (SPMD) : Single Program Multiple Data (SPMD) programs are a special class of SIMD programs which emphasize medium-grain parallelism and synchronization at the subprogram level rather than at the instruction level. In this sense, the data-parallel programming model applies to both synchronous SIMD and loosely coupled MIMD computers. Program conversion between different machine architectures is very much needed to broaden software portability. Procedure level corresponds to medium-grain size at the task, procedural, subroutine, and coroutine levels. A typical grain at this level contains less than 2000 instructions. Detection of parallelism at this level is much more difficult than at the finer-grain levels. Interprocedural dependence analysis is much more involved and history-sensitive.

The communication requirement is often less compared with that required in MIMD execution mode. SPMD execution mode is a special case at this level. Multitasking, also belongs in this category. Significant efforts by programmers may be needed to restructure a program at this level, and some compile assistance is also needed. Subprogram Level corresponds to the level of job steps and related subprograms. The grain size may typically contain thousands of instructions. Job steps can overlap across different jobs. Subprograms can be scheduled for different processors in SPMD mode.

Q 20. Give an overview of CISC Architecture. (PTU, May 2016 ; Dec. 2009, 2005)

Ans. CISC Characteristics : The design of an instruction set for a computer must take into consideration not only machine language constructs, but also the requirements imposed on the use of

high-level programming languages. The translation from high-level to machine language programs is done by means of a compiler program. One reason for the trend to provide a complex instruction set is the desire to simplify the compilation and improve the overall computer performance. The task of a compiler is to generate a sequence of machine instructions for each high-level language statement. The task is simplified if there are machine instructions that implement the statements directly. The essential goal of a CISC architecture is to attempt to provide a single machine instruction for each statement that is written in a high-level language. Examples of CISC architectures are the Digital Equipment Corporation VAX computer and the IBM 370 computer.

Another characteristic of CISC architecture is the incorporation of variable length instruction formats. Instructions that require register operands may be only two bytes in length, but instructions that need two memory addresses may need five bytes to include the entire instruction code. If the computer has 32-bit words (four bytes), the first instruction occupies half a word, while the second instruction needs one word in addition to one byte in the next word. Packing variable instruction formats in a fixed-length memory word requires special decoding circuits that count bytes within words and frame the instructions according to their byte length.

The instructions in a typical CISC processor provide direct manipulation of operands residing in memory. For example, an ADD instruction may specify one operand in memory through index addressing and a second operand in memory through a direct addressing. Another memory location may be included in the instruction to store the sum. This requires three memory references during execution of the instruction. Although CISC processors have instructions that use only processor register, the availability of other modes of operations tend to simplify high-level language compilation. However as more instructions and addressing modes are incorporated into a computer, the more hardware logic is needed to implement and support them, and this may cause the computations to slow down. In summary, the major characteristics of architecture are :

1. A large number of instructions-typically from 100 to 250 instruction.
2. Some instructions that perform specialized tasks and are used infrequently.
3. A large variety of addressing modes-typically from 5 to 20 different modes.
4. Variable-length instruction formats.
5. Instructions that manipulate operands in memory.

Q 21. What do you mean by parallel and distributed computers? Explain.

(PTU, Dec. 2010, 2006; May 2006)

Ans. Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel"). There are several different forms of parallel computing : bit-level, instruction level, data, and task parallelism. Parallelism has been employed for many years, mainly in high-performance computing, but interest in it has grown lately due to the physical constraints preventing frequency scaling. As power consumption (and consequently heat generation) by computers has become a concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multicore processors.

Parallel computers can be roughly classified according to the level at which the hardware supports parallelism – with multi-core and multi-processor computers having multiple processing elements within a single machine, while clusters, MPPs and grids use multiple computers to work on the same task. Specialized parallel computer architectures are sometimes used alongside traditional processors, for accelerating specific tasks.

Distributed computing : Distributed computing deals with hardware and software systems containing more than one processing element or storage element, concurrent processes, or multiple programs, running under a loosely or tightly controlled regime.

In distributed computing a program is split up into parts that run simultaneously on multiple computers communicating over a network. Distributed computing is a form of parallel computing, but parallel computing is most commonly used to describe program parts running simultaneously on multiple processors in the same computer. Both types of processing require dividing a program into parts that can run simultaneously, but distributed programmes often must deal with heterogenous environments, network links of varying latencies, and unpredictable failures in the network or the computers.

Q 22. Explain about the multiprocessor.

(PTU, Dec. 2006)

OR

Explain in brief about MIMD machines.

(PTU, Dec. 2009, 2005)

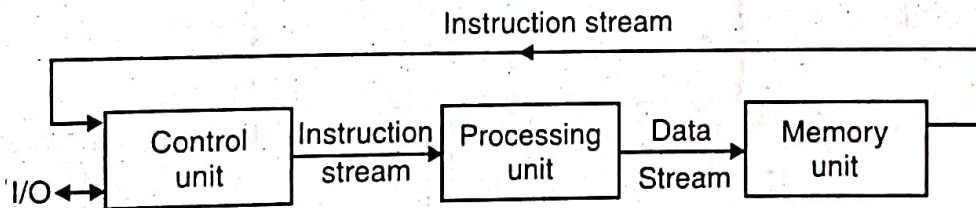
OR

Explain briefly Array Processor.

(PTU, Dec. 2019)

Ans. Multiprocessor (Array Processors) : An array processors that performs computations on large arrays of data. The term is used to refer to two different types of processors. An attached array processor is an auxiliary processor attached to a general-purpose computer. It is intended to improve the performance of the host computer in specific numerical computation tasks. The normal operation of a computer is to fetch instructions from memory and execute them in the processor. Parallel processing may occur in the instruction stream, in the data stream or in both. Flynn classified computers into four major groups.

SISD Uniprocessor Architecture : SISD (single instruction stream, single data stream) is the organization of a single computer containing a processing unit, a control unit and a memory unit as show in Fig. 1. Instructions are executed sequentially (one by one with system clock time period) and system may or may not have internal parallel processing capabilities. The sequence of instructions read from memory unit constitutes an instruction stream and the operation performed on the data in the processor constitutes a data stream, Parallel processing can be achieved by using pipeline processing technique or by using multiple functional units.



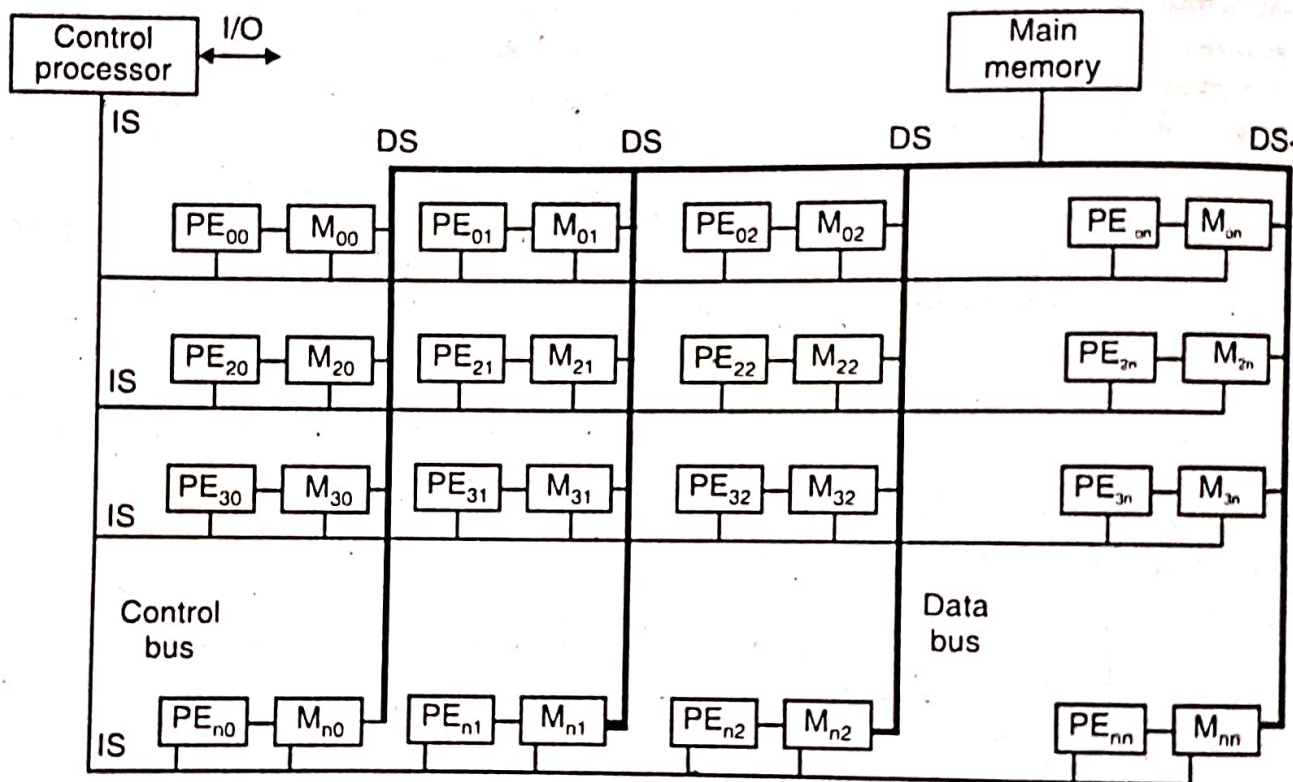
SISD uniprocessor architecture

SIMD Array Processor : SIMD (single instruction stream, multiple data stream) is the organization of a single computer containing multiple processors connected with a common control unit and a shared memory unit. All processing elements (PEs) receive the same instruction stream (IS) from the control unit but operate on different data stream (DS). The shared memory unit contain multiple modules so that it can communicate with all the processors simultaneously.

A general block diagram of an array processor is shown in Fig. It contains a set of identical processing elements (PEs), each having a local memory M. Each processor element includes an ALU, a floating-point arithmetic unit, and working registers. The master control unit (control processor) controls the operations in the processor elements. The main memory is used for storage of the program. The function of the master control unit is to decode the instructions and determine how the instruction is to be executed. The common control unit is responsible for fetching and interpreting instructions. When it encounters an arithmetic or other data processing instruction, it broadcasts the instruction to all PEs, when then all perform the same operation.

Scalar and program control instructions are directly executed within the master control unit. Vector instructions are broadcast to all PEs simultaneously. Each PE uses operands stored in its local memory. Vector operands are distributed to the local memories prior to the parallel execution of the instruction. On each clock cycle, the control processor issues an instruction to each processor using the control bus. Each processor performs that instruction and (optionally) returns a result to the memory via the data bus.

Consider for example the vector addition $C = A + B$. The master control unit first stores the i^{th} components a_i and b_i of A and B in local memory M_i for $i = 1, 2, 3, \dots, n$. It then broadcasts the floating-point add instruction $C_i = a_i + b_i$ to all PEs, causing the addition to take place simultaneously. The components of C_i are stored in fixed locations in each local memory. This produces the desired vector sum in one add cycle. A key division of vector processors arises from the way the instructions access their operands.



SIMD architecture with distributive memory

In the memory to memory organization the operands are fetched from memory and routed directly to the functional unit. Results are streamed back out to memory as the operation proceeds.

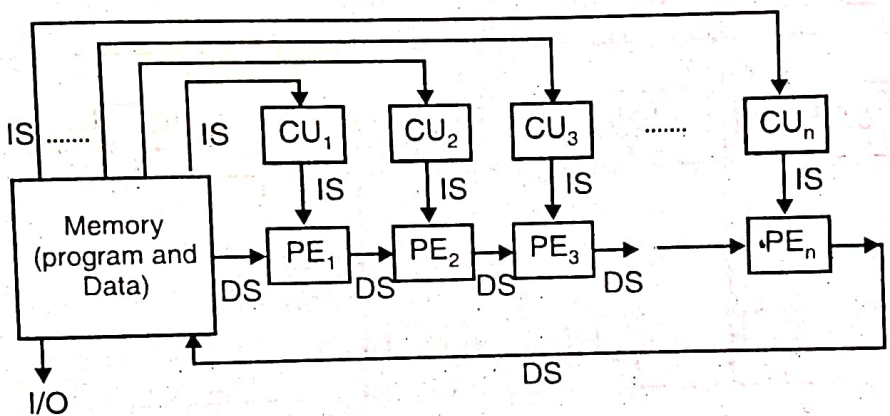
In the register to register organization operands are first loaded into a set of vector registers, each of which can hold a segment of a register, for example 64 elements. The vector operation then proceeds by fetching the operands from the vector registers and registers and returning the results to a vector register.

The advantage of memory to memory machines is the ability to process very long vectors, whereas register to register machines must break long vectors into fixed length segments. Unfortunately, this flexibility is offset by a relatively large overhead known as the startup time, which is the time between the initialization of the instruction and the time the first result emerges from the pipeline. The long startup time on a memory to memory machine is a function of memory latency, which is longer than the time it takes to access a value in an internal register. Once the pipeline is full, however, a result is produced every cycle or perhaps every other cycle.

Masking schemes are used to control the status of each PE during the execution of vector instructions. Each PE has a flag that is set when the PE active and reset when the PE is inactive. This ensures that only those PEs that need to participate are active during the execution of the instruction. For example, suppose that the array processor contains a set of 64 PEs. If a vector of length of less than 64 data items is to be processed, the control unit selects the proper number of PEs to be active. Vectors of greater length than 64 must be divided into 64-word portions by the control unit.

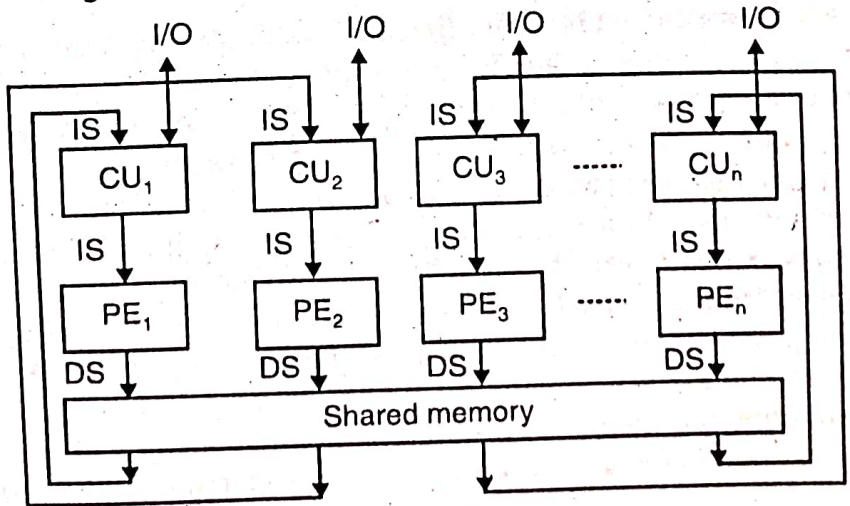
The best known SIMD array processor is the ILLIAC IV computer developed at the University of Illinois and manufactured by the Burroughs Corp. This computer is no longer in operation. SIMD processors are highly specialized computers. They are suited primarily for numerical problems that can be expressed in vector or matrix form. However, they are not very efficient in other types of computations or in dealing with conventional data-processing programs.

MISD Architecture : MISD (multiple instruction stream, single data stream) is the organization of a single computer containing multiple processors connected with multiple control units and a common memory unit as shown in Fig.. All processing elements (PEs) receive the different instruction stream (IS) of a program from the control units and operate simultaneously over single data stream (DS) given by common memory unit. Therefore MISD organization refers to a computer system capable of processing several instructions over single data stream at the same time.



MISD architecture

MIMD Architecture : MIMD (multiple instruction stream, multiple data stream) is the organization of a single computer containing multiple processors connected with multiple control units and a shared memory unit as shown in Fig. All processing elements (PEs) receive the different instruction stream (IS)

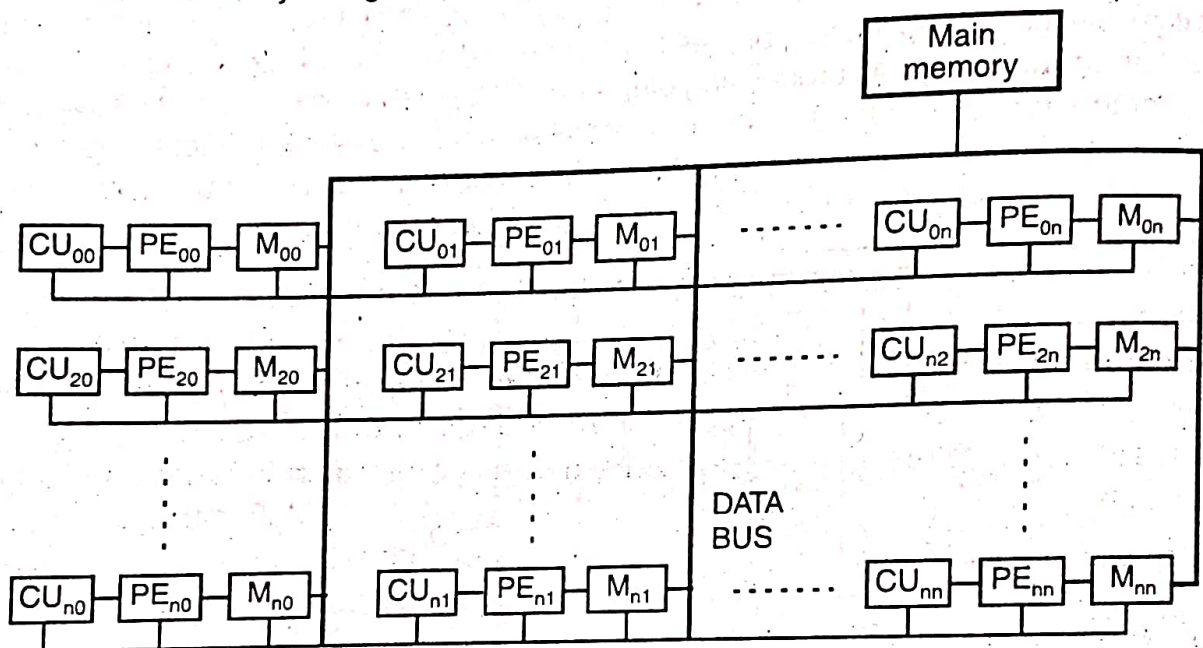


MIMD Architecture

of a program from the control units (CUs) and operate simultaneously over the multiple data streams (DS). The shared memory unit contain multiple modules so that it can communicate with all the processors simultaneously. Therefore MIMD organization refers to a computer system capable of processing several programs at the same time.

Non-Uniform Memory Access - MIMD Technique : One disadvantage of the MIMD shared memory model is that all the processors must communicate with a single bank of memory, and this gives rise to the danger of memory bottlenecks. Earlier designs of MIMDs used a single memory bus to connect the processors with the memory; with only so much bus bandwidth available, adding more processors quickly saturates the bus capacity and kills scalability.

To get around this, MIMD machines have borrowed ideas from distributed memory Massively Parallel Processing (MPP) systems to produce a hybrid Non-Uniform Memory Access (NUMA) or Distributed Shared Memory design. The Fig. illustrates this idea.



MIMD Computer with distributive memory

In this model the memory is physically distributed and resides with each individual processor, but the combination of the operating system and extra hardware controllers maintain the single shared memory image at the user level.

Message-passing Distributed Memory (MDM) : Distributed memory systems employing so-called message passing can accommodate thousands of processors. Each processor has a private block of memory from which to draw data. The drawbacks? Such systems tend to be slower because data often must be shuffled back and forth between the processors and they're more difficult to program. Also there's less available software as compared with the number programs that can be run on SMP and DSM architectures. As a result, a lot of applications have either to be "ported" from other types of processors, or custom-built--an expensive proposition.

Distributed Shared Memory (DSM) : New, hybrid or distributed memory systems are emerging that maximize the strengths of each of the previous systems. Groups of processors (called nodes) share a local memory; the nodes are networked so that any processor can access any portion of memory. These system should have the advantage of being scalable and quite easy to program. Though these system are still somewhat experimental more and more software for them is becoming available.

Q 23. Describe the following terminology associated with multiprocessor :

- (a) Mutual Exclusion
- (b) Critical Section
- (c) Hardware Lock
- (d) Semaphores
- (e) Test and Set instruction

(PTU, May 2007)

Ans. The following terminology associated with multiprocessors :

A properly functioning multiprocessor system must provide a mechanism that will guarantee orderly access to shared memory and other shared resources. This necessary to protect data from being changed simultaneously by two or more processors. This mechanism has been termed 'mutual exclusion'. Mutual exclusion must be provided in a multiprocessor system to enable one processor to exclude or lock out access to a shared resource by critical section other processors when it is in a critical section. A critical section is a program sequence that, once begun, must complete execution before another processor access the same shared resource.

A binary variable called a semaphore is often used to indicate whether or not a processor is executing a critical section. A semaphore is a software controlled flag that is stored in a memory location that all processors can access. When the semaphore is equal to 1, it means that a processor is executing a critical program, so that the shared memory is not available to other processors. When the semaphore is equal to 0, the shared memory is available to any requesting processor. Processors that share the same memory segment agree by convention not to use the memory segment unless the semaphore is equal to 0, indicating that memory is available. They also agree to set the semaphore to 1 when they are executing a critical section and to clear it to 0 when they are finished.

Testing and setting the semaphore is itself a critical operation and must be performed as a single indivisible operation. If it is not, two or more processors may test the semaphore simultaneously and then each set it, allowing them to enter a critical section at the same time. This action would allow simultaneous execution of critical section, which can result in erroneous initialization of control parameters and a loss of essential information.

A semaphore can be initialized by means of a test and set instruction in hardware lock conjunction with a hardware lock mechanism. A hardware lock is a processor generated signal that serves to prevent other processors from using the system bus as long as the signal is active. The test-and-set instruction tests and sets a semaphore and activates the lock mechanism during the time that the instruction is being executed. This prevents other processors from changing the semaphore between the time that the processor is testing it and the time that it is setting it. Assume that the semaphore is a bit in the least significant position of a memory word whose address is symbolized by SEM. Let the mnemonic TSL designate the "test and set while locked" operation. The instruction

TSL SEM

will be executed in two memory cycles (the first to read and the second to write) without interference as follows :

$R \leftarrow M[\text{SEM}]$ Test semaphore

$M[\text{SEM}] \leftarrow 1$ Set semaphore

The semaphore is tested by transferring its value to a processor register R and then it is set to 1. The value in R determines what to do next. If the processor finds that $R = 1$, it knows that the semaphore was originally set. (The fact that it is set again does not change the semaphore value.) That means that another processor is executing a critical section, so the processor that checked the

semaphore does not access the shared memory. If $R = 0$, it means that the common memory (or the shared resource that the semaphore represents) is available. The semaphore is set to 1 to prevent other processors from accessing memory. The processor can now execute the critical section. The last instruction in the program must clear location SEM to zero to release the shared resource to other processors.

Q 24. Write and explain types of parallel processor systems.

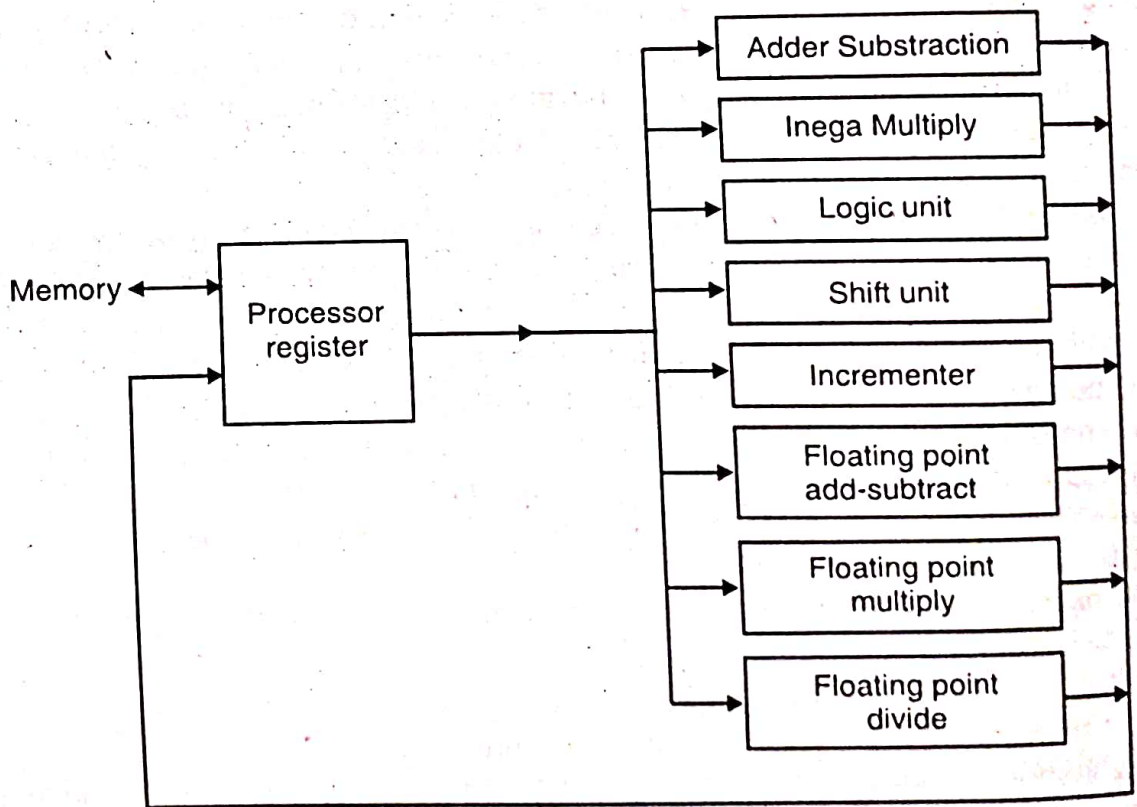
(PTU, Dec. 2013, 2007)

Ans. Parallel processing is a term used to denote a large class of techniques that are used to provide simultaneous data processing tasks for purpose of increasing computational speed of a computer system. Instead of processing each instruction sequentially as in a conventional computer system, a parallel system is able to perform concurrent data processing to achieve faster execution time for example, while an instruction is being executed by an ALU, the next instruction can read for memory. The system have two or more ALU and be able to execute two or more instruction at some time. Furthermore, the system have two or more processor operating concurrently.

The purpose of parallel processing is to speed up for computer processing capabilities and increase its throughput. In other words amount of processing that can be accomplished during again interval of time.

The amount of hardware increases with parallel processing and with this cost of system increases. However technological developments have reduced hardware costs to the point where parallel processing techniques are economically feasible.

Parallel process can be viewed from various levels of complexity. At lowest level we distinguish between parallel and serial operations by the type of registers used. Processor with multiple functional unit discussion diagram.



Processor with multiple functional unit

The diagram shown one possible way of separating execution unit into eight functional units operating in parallel. A multifunctional organization is usually associated with complex control unit to coordinate all the activities among the various component.

There are variety of ways that parallel processing can be classified. It can considered from the connection struct between processors or from flow of information through the system. One classification introduced by M.J. Flynn considers the organization of a computer system by the number of instruction and data items that are manipulated simultaneously. The normal operation of a computer is to fetch instruction from memory and execute them in the processor. The sequence of instructions fetched from memory constitutes an instruction stream. The operations read from memory constitutes data stream. Parallel processing may occur in instruction stream or in both. Flynn's classification divides computers into four major groups as follows :

Single instruction stream, single data stream (SISD)

Single instruction stream, multiple data stream (SIMD)

Multiple instruction stream, single data stream (MISD)

Multiple instruction stream, multiple data stream (MIMD)

SISD (single instruction stream, single data stream) represent the organization of a single computer containing a control unit, a processor unit and a memory unit. Instructions are executed sequentially and the system may or may not have internal parallel processing capabilities. Parallel processing in this case, may be means of multiple functional units or by pipeline processing.

SIMD represents an organization that includes many processing units under supervision of a common control unit. All processors receive same instruction from control unit but operate on different items of data. The shared memory unit must contain multiple modules so that it can communicate with all processors simultaneously.

MISD structure is only of theoretical interest since no practical system has been constructed using this organisation.

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

MIMD organization refers to a computer system capable of processing several programs at same time. Most multiprocessor and multi computer systems can be classified in few category.

Flynn's classification depends upon dis function between performance of control unit and data processing unit. It emphasize the behaviour characteristics of computer system rather than its operational and structural interconnections one type of parallel processing that doesn't fit flynn's classification is pipelining. Parallel processing computers are required to meet demand of large computer it may scientific, engineering, military, medical, artificial intelligence and basic research areas.

Q 25. Distinguish between SIMD and MIMD.

(PTU, Dec. 2016 ; May 2009)

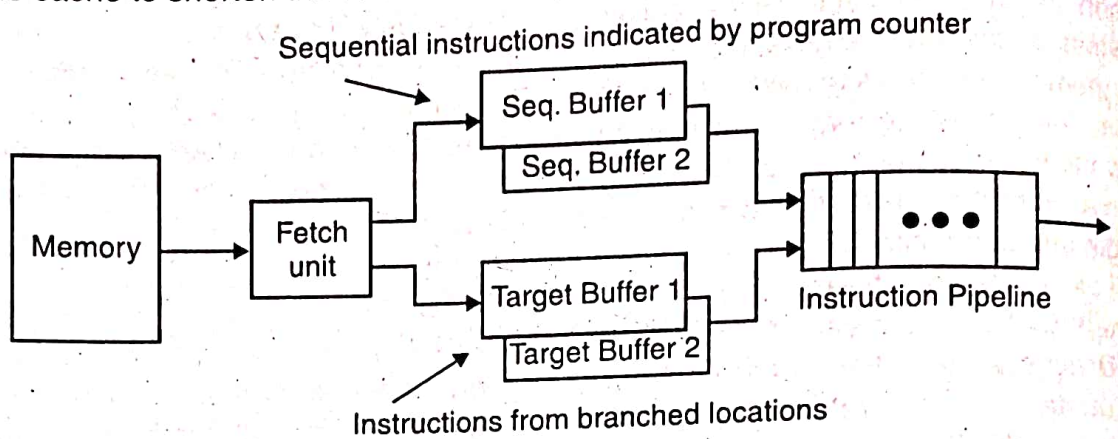
Ans. SIMD (single instruction stream, multiple data stream) is the organization of a single computer containing multiple processors connected with a common control unit and a shared memory unit. All processing elements (PEs) receive the same instruction stream (IS) from the control unit but operate on different data stream (DS).

MIMD (multiple instruction stream, multiple data stream) is the organization of a single computer containing multiple processors connected with multiple control units and a shared memory unit as shown in Fig. All processing elements (PEs) receive the different instruction stream (IS) of a program from the control units (CUs) and operate simultaneously over the multiple data streams (DS).

Q 26. What is the purpose of prefetch buffers in instruction pipelining?

(PTU, May 2010)

Ans. Prefetch buffer is used to match the instruction fetch rate to the pipeline consumption rate. In one memory access time a block of consecutive instructions are fetched into a prefetch buffer as shown in the fig. below. The block access can be achieved by interleaved memory modules or using the cache to shorten the effective memory access time.



The use of sequential and target buffers

Q 27. What is parallel computers?

(PTU, May 2010)

Ans. Parallel computer can be roughly classified according to the level at which the hardware supports parallelism, with multi-core and multi-processor computers having multiple processor elements within a single machine. Special computer architectures are sometimes used alongside traditional processors for accelerating specified tasks. Parallel computer programs are more difficult to write than sequential ones, because concurrency introduces several new classes of potential software bugs. Communication and synchronization between the different subtasks are typically one of the greatest obstacles to getting good parallel program performance. Being faster parallel computers are used in wide range of application like bioinformatics, simulation etc.

Q 28. Explain arithmetic pipeline.

Ans. Pipeline arithmetic units are usually found in very high speed computers. They are used to implement floating point operations, multiplication of fixed points numbers and similar computations encountered in scientific problems.

Q 29. What is array processor?

Ans. An array processor is a processor that performs computations on large arrays of data.

Q 30. What is attached array processor?

Ans. An attached array processor is designed as a peripheral for a conventional host computer and its purpose is to enhance the performance of the computer by providing vector processing for complex scientific applications. It achieves high performance by means of parallel processing with multiple functional units. It includes an arithmetic unit containing one or more pipelined floating point address and multipliers.

Q 31. What is vector processing?

(PTU, May 2017)

Ans. There is a class of computational problems that are beyond the capabilities of a conventional computer. These problems are characterized by the fact that they require a vast number of computations that will take a conventional computer days or even weeks to complete. In many science and engineering applications, the problems can be formulated in terms of vectors and matrices that lend themselves to vector processing.

Q32. What are the various applications of vector processing?

Ans. The following are the representative application areas where the vector processing is of important.

- Petroleum Exploration
- Medical diagnosis
- Seismic data analysis
- Long range weather forecasting
- Aerodynamics and space flight simulations
- Image processing
- Artificial intelligence and expert system
- Mapping the human genome.

Q33. What is parallel processing?

(PTU, Dec. 2016)

Ans. Parallel processing is a term used to denote a large class of techniques that are used to perform simultaneous data processing tasks for the purpose of increasing the computational speed of a computer system.

A parallel processing system is able to perform concurrent data processing to achieve the minimum execution time. The main purpose of parallel processing is to speed up the computer processing capability and increase its throughput i.e. the amount of processing that can be accomplished during a given interval of time.

Q34. What is interprocess synchronization?

(PTU, May 2016)

Ans. All processes prevent casual exchange of data. However, occasionally two processes need to communicate with each other. One method that enables processes to communicate is called interprocess synchronization.

Q35. What is semaphore?

Ans. Semaphore is a bucket. When a semaphore is allocated, a bucket that contains a fixed number of keys is created. Processes using semaphores must first procure a key from the bucket before they can continue to execute. If a specific process requires a key, only a fixed number of instances of that process can be in progress simultaneously. All others must wait until a sufficient number of keys is returned to the bucket. Semaphores are typically used for mutual exclusion, access control to shared resources, and for basic synchronization.

Semaphore is a built-in class that provides the following methods :

- Create a semaphore with a specified number of keys : new ()
- Obtain one or more keys from the bucket : get ()
- Return one or more keys into the bucket : put ()
- Try to obtain one or more keys without blocking : try_get ()

Q36. What is inter-process communication?

Ans. Inter-process communication, which is short is known as IPC, deals mainly with the techniques and mechanisms that facilitate communication between processes.

In computing, **Inter-process communication (IPC)** is a set of methods for the exchange of data among multiple threads in one or more processes. Processes may be running on one or more computers connected by a network. IPC methods are divided into methods for message passing, synchronization, shared memory, and remote procedure calls (RPC). The method of IPC used may vary based on the bandwidth and latency of communication between the threads, and the type of data being communicated.

There are several reasons for providing an environment that allows process cooperation :

- Information sharing

- Computational speedup
- Modularity
- Convenience
- Privilege separation

IPC may also be referred to as inter-thread communication and inter-application communication. (PTU, Dec. 2013, 2011)

Q 37. What is pipelining?

Ans. Pipelining is a key implementation technique used to build fast processors. It allows the execution of multiple instructions to overlap in time. A pipeline within a processor is similar to a car assembly line. Each assembly station is called a pipe stage or a pipe segment. The throughput of an instruction pipeline is the measure of how often an instruction exits the pipeline.

Q 38. List some properties of SIMD.

(PTU, May 2015 ; Dec. 2011)

Ans. Single instruction, multiple data (SIMD), is a class of parallel computers in Flynn's taxonomy. It describes computers with multiple processing elements that perform the same operation on multiple data simultaneously. Thus, such machines exploit data level parallelism.

Q 39. A given program consists of 50 instruction loop that is executed 35 times. It takes 6,000 cycles to execute the program on a given system. Find the computer performance in CPI (cycles per instruction).

(PTU, May 2012)

Ans. The 50-instruction loop is executed 35 times,
so the total number of instructions executed is $50 \times 35 = 1750$.
It takes 6,000 cycles to execute the program,
so the CPI is $6,000/1750 = 3.43$.

Q 40. What do you understand by the terms "loosely coupled" and "tightly coupled" in parallel computers?

(PTU, May 2012)

Ans. Loosely coupled is a type of coupling that describes how multiple computer systems, even those using incompatible technologies, can be joined together for transactions, regardless of hardware, software and other functional components. Loosely coupled systems describe those that work on an exchange relationship where little input is needed from each of the additional systems. In a loosely coupled system hardware and software may interact but they are not dependent on each other to work. Computers in a network are considered loose-coupled systems as a client machine may request data from the server, but the two systems also work independently of each other.

Tightly coupled is a type of coupling that describes a system in which hardware and software are not only linked together, but are also dependent upon each other. In a tightly coupled system where multiple systems share a workload, the entire system usually would need to be powered down to fix a major hardware problem, not just the single system with the issue.

Q 41. What do you understand by instruction pipeline? Discuss the major difficulties that cause the instruction pipeline to deviate from this normal operation.

(PTU, Dec. 2011)

Ans. An instruction pipeline is a technique used in the design of computers to increase their instruction throughput (the number of instructions that can be executed in a unit of time). Pipelining does not reduce the time to complete an instruction, but increases the number of instructions that can be processed at once.

Each instruction is split into a sequence of dependent steps. The first step is always to fetch the instruction from memory; the final step is usually writing the results of the instruction to processor registers or to memory. Pipelining seeks to let the processor work on as many instructions as there are dependent steps, but as an assembly line builds many vehicles at once, rather than waiting until one vehicle has passed through the line before admitting the next one. As the goal of the assembly

The aim is to keep each assembler productive at all times, pipelining seeks to use every portion of the processor busy with some instructions. Pipelining lets the computer's cycle time be the time of the slowest step, and ideally lets one instruction complete in every cycle.

Problems in Instruction Pipelining : Several difficulties prevent instruction pipelining from being as simple as the above description suggests. The principal problems are :

Timing Variations : Not all stages take the same amount of time. This means that the speed of a pipeline will be determined by its slowest stage. This problem is particularly acute in instruction processing, since different instructions have different operand requirements and sometimes vastly different processing time. Moreover, synchronization mechanisms are required to ensure that data is passed from stage to stage only when both stages are ready.

Data Hazards : When several instructions are in partial execution, a problem arises if they reference the same data. We must ensure that a later instruction does not attempt to access data sooner than a preceding instruction, if this will lead to incorrect results. For example, instruction $N + 1$ must not be permitted to fetch an operand that is yet to be stored into by instruction N .

Branching : In order to fetch the "next" instruction, we must know which one is required. If the present instruction is a conditional branch, the next instruction may not be known until the current one is processed.

Interrupts : Interrupts insert unplanned "extra" instructions into the instruction stream. The interrupt must take effect between instructions, that is, when one instruction has completed and the next has not yet begun. With pipelining, the next instruction has usually begun before the current one has completed.

Q 42. Describe the principle of operation and role of stack memory in program execution.

State the microinstructions executed in stack operation.

(PTU, May 2012)

Ans. In computer science, a **stack** is a last in, first out (LIFO) abstract data type and linear data structure. A stack can have any abstract data type as an element, but is characterized by two fundamental operations, called push and pop (or pull). The push operation adds a new item to the top of the stack, or initializes the stack if it is empty. If the stack is full and does not contain enough space to accept the given item, the stack is then considered to be in an overflow state. The pop operation removes an item from the top of the stack. A pop either reveals previously concealed items, or results in an empty stack, but if the stack is empty then it goes into underflow state (it means no items are present in stack to be removed). A stack pointer is the register which holds the value of the stack. The stack pointer always points to the top value of the stack.

A stack is a restricted data structure, because only a small number of operations are performed on it. The nature of the pop and push operations also means that stack elements have a natural order. Elements are removed from the stack in the reverse order to the order of their addition : therefore, the lower elements are those that have been on the stack the longest.

At the lowest level the stack is the place where certain instructions store or retrieve data and where data is stored when an interrupt occurs. Microprocessors vary, but there are 5 general types of stack-specific instructions :

1. **Push** : Put data onto the stack
 2. **POP (or PULL)** : "Remove" data from the stack
 3. **CALL** : Jump to a subroutine and put the return address on the stack.
 4. **Return** : Return from a subroutine by loading the program counter with the stack top
 5. **INT (or SWI)** : Software interrupt ; a specialized CALL
- When a processor interrupt occurs (due to an external device), the CPU will save the current program counter and (usually) the flags register on the stack and jump to the handling subroutine. This

Pipelining

is to keep each assembler productive at all times, pipelining seeks to use every portion of the processor busy with some instructions. Pipelining lets the computer's cycle time be the time of the slowest step, and ideally lets one instruction complete in every cycle.

Problems in Instruction Pipelining : Several difficulties prevent instruction pipelining from being as simple as the above description suggests. The principal problems are :

Timing Variations : Not all stages take the same amount of time. This means that the speed gain of a pipeline will be determined by its slowest stage. This problem is particularly acute in instruction processing, since different instructions have different operand requirements and sometimes vastly different processing time. Moreover, synchronization mechanisms are required to ensure that data is passed from stage to stage only when both stages are ready.

Data Hazards : When several instructions are in partial execution, a problem arises if they reference the same data. We must ensure that a later instruction does not attempt to access data sooner than a preceding instruction, if this will lead to incorrect results. For example, instruction $N + 1$ must not be permitted to fetch an operand that is yet to be stored into by instruction N .

Branching : In order to fetch the "next" instruction, we must know which one is required. If the present instruction is a conditional branch, the next instruction may not be known until the current one is processed.

Interrupts : Interrupts insert unplanned "extra" instructions into the instruction stream. The interrupt must take effect between instructions, that is, when one instruction has completed and the next has not yet begun. With pipelining, the next instruction has usually begun before the current one has completed.

Q 42. Describe the principle of operation and role of stack memory in program execution. State the microinstructions executed in stack operation. (PTU, May 2012)

Ans. In computer science, a **stack** is a last in, first out (LIFO) abstract data type and linear data structure. A stack can have any abstract data type as an element, but is characterized by two fundamental operations, called push and pop (or pull). The push operation adds a new item to the top of the stack, or initializes the stack if it is empty. If the stack is full and does not contain enough space to accept the given item, the stack is then considered to be in an overflow state. The pop operation removes an item from the top of the stack. A pop either reveals previously concealed items, or results in an empty stack, but if the stack is empty then it goes into underflow state (It means no items are present in stack to be removed). A stack pointer is the register which holds the value of the stack. The stack pointer always points to the top value of the stack.

A stack is a restricted data structure, because only a small number of operations are performed on it. The nature of the pop and push operations also means that stack elements have a natural order. Elements are removed from the stack in the reverse order to the order of their addition : therefore, the lower elements are those that have been on the stack the longest.

At the lowest level the stack is the place where certain instructions store or retrieve data and where data is stored when an interrupt occurs. Microprocessors vary, but there are 5 general types of stack-specific instructions :

1. **Push** : Put data onto the stack
 2. **POP (or PULL)** : "Remove" data from the stack
 3. **CALL** : Jump to a subroutine and put the return address on the stack.
 4. **Return** : Return from a subroutine by loading the program counter with the stack top
 5. **INT (or SWI)** : Software interrupt ; a specialized CALL
- When a processor interrupt occurs (due to an external device), the CPU will save the current program counter and (usually) the flags register on the stack and jump to the handling subroutine. This

allows the handling subroutine to process the interrupt and return to whatever the CPU was doing with its current state preserved.

While a microprocessor has only one stack active at a time, the operating system can make it appear as if there are multiple stacks. At least one for the OS, one for each process and one for each thread. In fact, the threads themselves may implement multiple stacks.

At a higher level, whatever language is used to implement a thread will often use the stack for its own purposes to store functional call parameters, local variables and function call return values.

Role of stack memory in program execution : Stack is used largely during a function call but depending on the language and level of programming it may be used to temporarily store processor register data or other variables.

Further, the stack may also be used for short-term large-scale storage of data when using **recursive functions** that store partial data in the stack and call themselves again.

1. Return address
2. Return value
3. Parameters to called function
4. Local variables in the called function
5. Processor registers that will be reused in the called function.

Q 43. Write short notes on the following :

(a) Transaction processing benchmarks

(b) SPMD.

(PTU, May 2012)

Ans. (a) Transaction Processing Benchmarks and Klips Ratings : On-line transaction processing applications demand rapid, interactive processing for large number of relatively simple transactions. They are typically supported by very large database. Automated teller machines and airline reservation systems are familiar examples. The throughput of computers for on-line transaction processing is often measured in transactions per second (TPS). Each transaction may involve a database search, query answering, and database update operations. Business computers should be designed to deliver a high TPS rate.

The TPI (Transaction processing) benchmark was originally proposed in 1985 for measuring the transaction processing of business application computers. This benchmark has also become a standard for gauging relational database performances. The benchmark is closely related to the Debit Credit benchmark. Based on today's standards, any computer exceeding 100 TPS is considered fast for transaction processing. For example, the VAX 9000 executed 70 TPS processor. The Sequential Symmetry multiprocessor achieved 140 TPS using a shared memory system with 24 processors.

(b) SPMD (single process, multiple data ; or single program, multiple data) is a technique employed to achieve parallelism ; it is a subcategory of MIMD. Tasks are split up and run simultaneously on multiple processors with different input in order to obtain results faster. SPMD is the most common style of parallel programming. It is also a prerequisite for research concepts such as active messages and distributed shared memory.

In SPMD, multiple autonomous processors simultaneously execute the same program at independent points. With SPMD, tasks can be executed on general purpose CPUs.

Q 44. What are the benchmarks for evaluating the performance of a multiprocessor system (MIMD)? Explain with example.

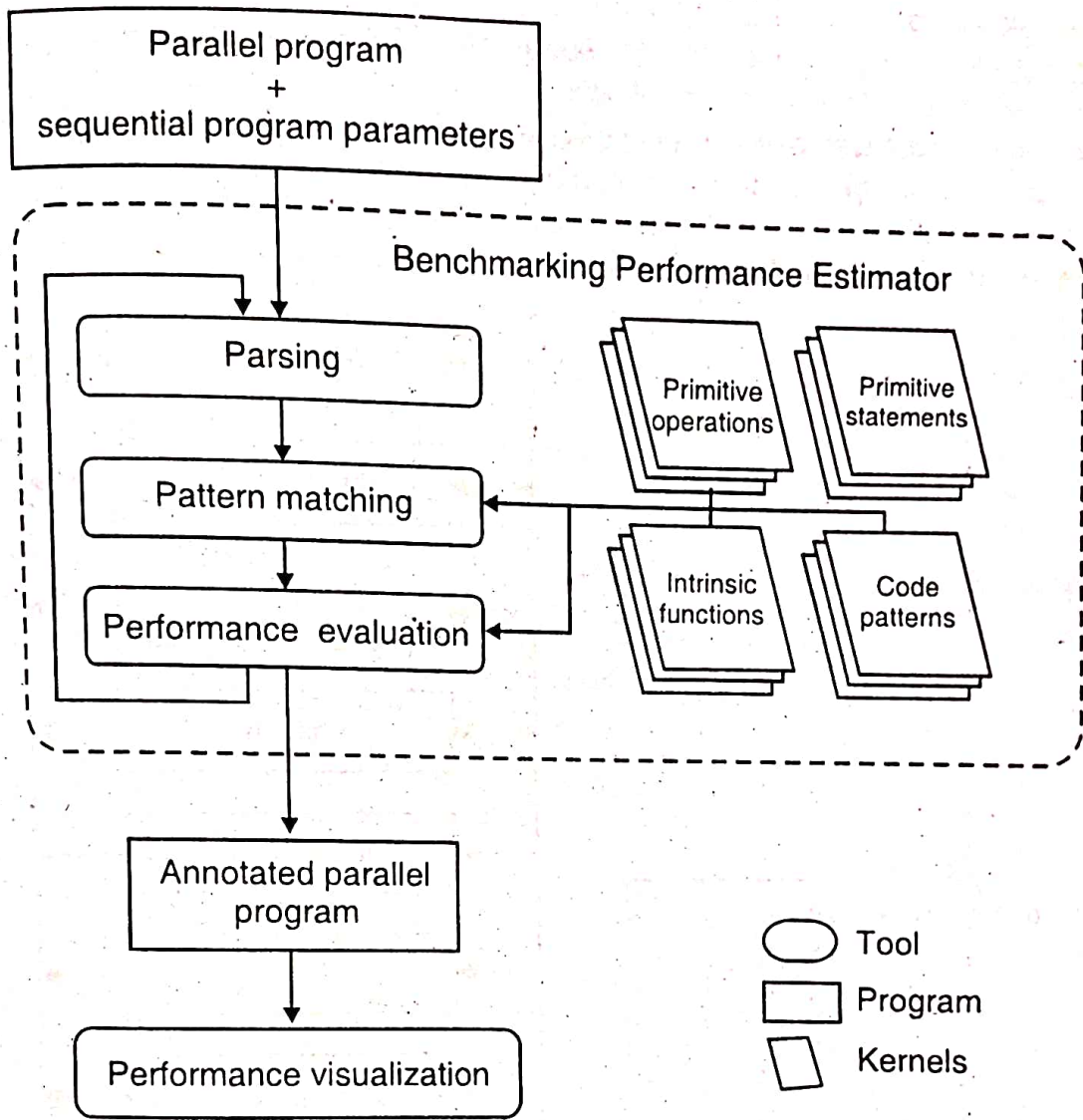
(PTU, Dec. 2011)

Ans. The Benchmark approach is evaluated with the following issues :

1. Time effort to build and maintain a benchmark performance estimator
2. Probability
3. Measurement complexity

- 4. Local and global performance influence of target machine and compiler
- 5. Pattern matching
- 6. Prediction accuracy

The following fig. shows the performance evaluator based on
A performance evaluator based on bench marking approach



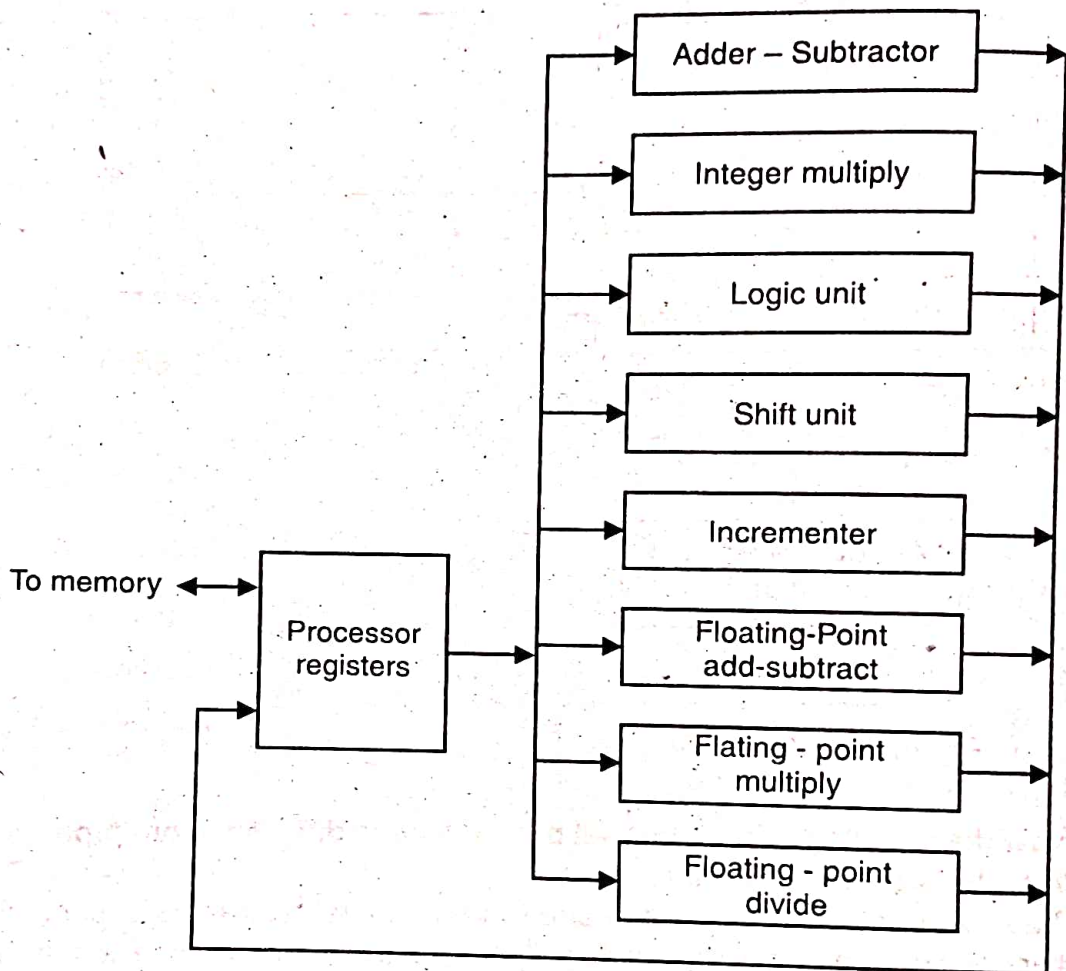
Q 45. How the architecture of parallel processors is different from pipeline processors? Give the application areas of the both. (PTU, May 2011)

Ans. Parallel processing is a term used to denote a large class of techniques are used to provide simultaneous data-processing tasks for the purpose of increasing the computational speed of a computer system. Instead of processing each instruction sequentially as in a conventional computer, a parallel processing system is able to perform concurrent data processing to achieve faster execution time for example, while an instruction is being executed in the ALU, the next instructions can be read from memory. The system may have two or more ALUs and be able to execute two or more instructions at the same time furthermore the system may have two or more processes operating concurrently. The purpose of parallel processing is to speed up the computer processing capability and increase its throughput, that is, the amount of processing that can be accomplished during a given interval of time. The amount of hardware increases with parallel processing, and with it, the cost of the system increases. However, technological developments have reduced hardware costs to the point where parallel processing techniques are economically feasible.

Parallel processing can be viewed from various levels of complexity. At the lowest level, we distinguish between parallel and serial operations by the type of registers used. Shift registers operate in serial fashion one bit at a time. While registers with parallel load operate with all the bits of the word. Simultaneously parallel processing at a higher level of complexity can be achieved by having a multiplicity of functional units that perform identical or different operations simultaneously. Parallel processing is established by distributing the data among the multiple functional units. For example, the arithmetic logic and shift operations can be separated into three units and the operands diverted to each unit under the supervision of a control unit.

Shows one possible way of separating the execution unit into eight functional units operating in parallel. The operands in the registers are applied to one of the units depending on the operation specified by the instruction associated with the operands.

Processor with multiple functional units



Instruction associated with the operands

Pipelining : Pipelining is a technique of decomposing a sequential process into suboperations, with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments. A pipeline can be visualized as a collection of processing segments through which binary informations flows. Each segment performs partial processing dictated by the way the task is partitioned. The result obtained from the computation in each segment is transferred to the next segment in the pipeline. The final result is obtained after the data have passed through all segments. The name "pipeline" implies a flow of information analogous to an industrial assembly line. It is characteristic of pipelines that several computations can be in progress in distinct segments at

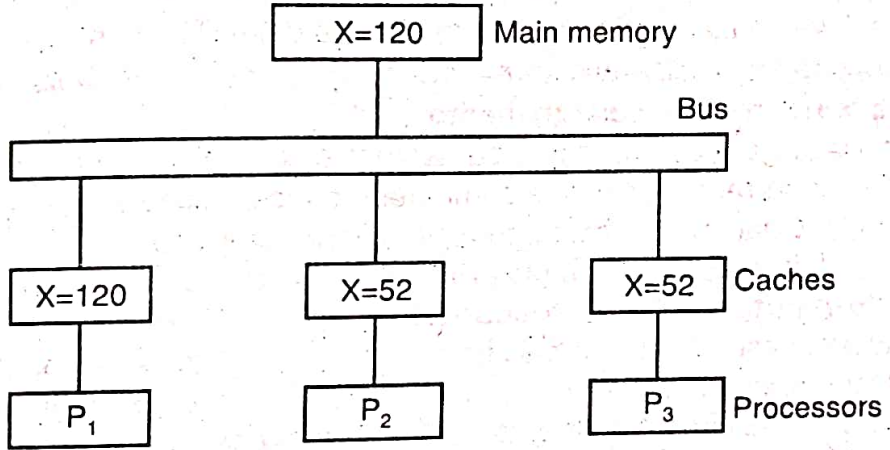
the same time. The overlapping of computation is made possible by associating a register with each segment in the pipeline. The registers provide isolation between each segment so that each can operate on distinct data simultaneously.

Perhaps the simplest way of viewing the pipeline structure is to imagine that each segment consists of an input register followed by a combinational circuit. The register holds the data and the combinational circuit performs the suboperation in the particular segment. The output of the combinational circuit in the given segment is applied to the input register of the next segment. A clock is applied to all registers after enough time has elapsed to perform all segment activity. In this way the information flows through the pipeline one step to a time.

Q 46. What is a Cache Coherence ? (PTU, May 2018, 2004)

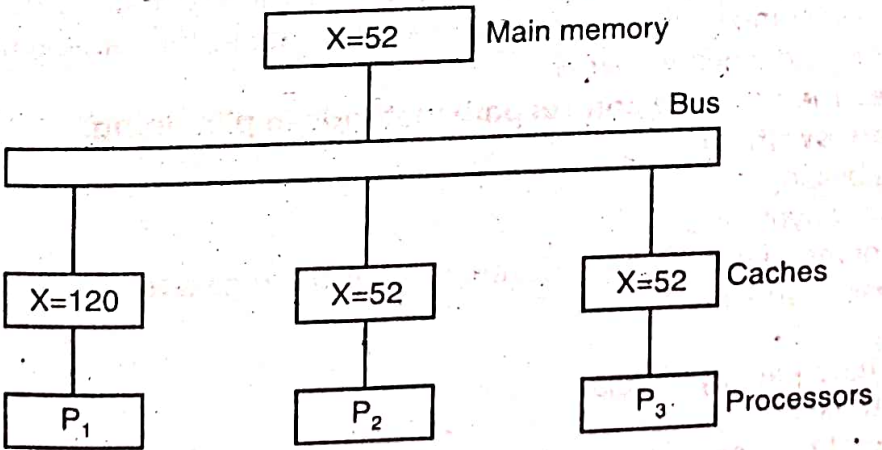
Ans. In a shared memory multiprocessor system, all the processors share common memory in addition, each processor may have a local memory, part or all of which may be a cache. The same information may reside in a number of copies in some cache and main memory. To ensure the ability of the system to execute memory operations correctly, the multiple copies must be kept identical. This requirement imposes a cache coherence problem. A memory scheme is coherent if the value returned on a load instruction is always the value given by latest store instruction with same address. Two types of cache coherence techniques are :

1. Write through Cache policy : In a write through policy, main memory and cache are consistent with each other, i.e. only change made in cache memory is reflected in main memory.



Write through Cache policy

2. Write back Cache policy : In write back policy, main memory is not updated at time of store, the copies in the other two cache and main memory are inconsistent.



Write back Cache policy

Q 47. The time delay for the four segments in a pipeline are as follows : $t_1 = 50$ ns, $t_2 = 30$ ns, $t_3 = 95$ ns, and $t_4 = 45$ ns. The interface registers delay time $t_r = 5$ ns.

(a) How long would it take to add 100 pairs of numbers in the pipeline?

(b) How can we reduce the total time to about one-half of the time calculated in part (a)?
(PTU, May 2011)

Ans. The time taken to add n pairs of numbers in an K -segment pipeline = $(K + n - 1) + P$

Given $K = 4$ and $n = 100$

Calculating the clock cycle t_P :

The total time delay in a segment °C.

TD_i = the time delay in the segment circuit and the time delay of the interface register.

The maximum time delay of all segments

TD_{max} = the maximum time delay in segments circuits + the time delay of the interface register

$$= 95\text{ns} + 5\text{ns} = 100\text{ns}$$

As we have a different time delay in each segment, the clock cycle must be larger than or equal to the maximum delay.

The clock cycle $t_p = 100$ ns.

The time taken to add 100 pairs of number in the 4 segment pipeline = $(K + n - 1) t_p$
= $(4 + 100 - 1) 100$ ns = 10.3 ms.

(b) To reduce the time calculated in part (a) to one half, we have to reduce the clock cycle t_p .

We can achieve this if we could reduce the maximum time delay TD_{max} to 50ns.

This can be reached by reducing t_1 from 50ns to 45 ns and t_3 from 95_{ns} to 45_{ns}.

Q 48. What is concurrent access to memory ?

Ans. The ability to gain admittance to a system or component by more than one user or process—for example concurrent access to a computer means multiple users are interacting with the system simultaneously. Concurrent access to a hardware component, such as a memory chip or memory bank, means that the circuits are built with two or more input or signaling channels.

Q 49. Write short note on Array Processors.

(PTU, Dec. 2013)

Ans. An array processor that performs computations on a large array of data. The term is used to refer to two different types of processors. An attached array processor is an auxiliary processor attached to a general purpose computer. It is intended to improve the performance of the host computer in specific numerical computation tasks. The normal operation of a computer is to fetch instructions from memory and execute them in the processor.

Q 50. What is the role of cache in pipelining ?

(PTU, Dec. 2015)

Ans. The use of cache memory is to solve the memory access problem. When cache is included in the processor the access time to the cache is usually the same time needed to perform other basic operation inside the processor.

Q 51. Discuss the data and control path methods in pipelining.

(PTU, Dec. 2015)

Ans. Data path synthesis

- Resource binding
- Connectivity synthesis

Connection of resources to : multiplexers busses and registers.

- Control unit interface
- I/O ports.

- Physical data-path synthesis

Control synthesis

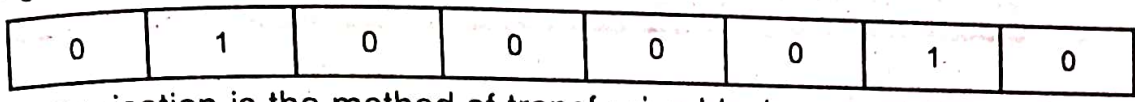
- Synthesis of the control unit

- Logic model : synchronous FSM
- Physical implementation :
 - Microcode (ROM,PLA).
 - Hard-wired FSM
 - Distributed FSM

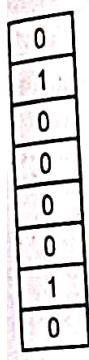
52. Differentiate serial and parallel communication.

(PTU, May 2016)

Ans. Serial and Parallel Communication : Data can be transmitted between a sender and a receiver in two main ways : Serial and Parallel. Serial communication is the method of transferring one bit at a time through a medium.



Parallel communication is the method of transferring blocks e.g. BYTES of data at the same time.



Q 53. What is the difference between serial and parallel processing ? (PTU, Dec. 2017)

Ans. The following are the differences between serial and parallel processing :

1. In serial processing, same tasks are completed at the same time but in Parallel processing completion time may vary.
2. Parallel processor is costly as compared to serial processor.
3. Serial processing takes more time than parallel processor.
4. In serial processing data transfers in bit by bit form while in parallel processing data transfers in byte form i.e in 8 bits form.

(PTU, May 2018)

Q 54. Write short note on Vector processors.

Ans. Vector Processors : Vector processor is a processor that solves computational problem that are beyond the capabilities of a conventional computer. These problems are characterized by the fact that they requires a vast number of computations that will take a conventional computer days or even weeks to complete.



Chapter

4

Memory Organisation

Contents

Memory interleaving, concept of hierarchy memory organization, cache memory, cache size vs. block size, mapping functions, replacement algorithms, write policies.

POINTS TO REMEMBER



- ☞ A memory unit is the collection of storage cells together with associated circuits needed to transfer information in and out of storage. The memory stores binary information in the form of bits known as 'words'.
- ☞ ROM memory can be classified into :
Masked programmed ROMs and
User programmed ROMs.
- ☞ A control unit consists of two decoders, a sequence counter and a number of control gates.
- ☞ The memory unit that communicate directly with the CPU is called main memory and the devices that provides backup storage are called Auxilliary memory.
- ☞ A special very high speed memory called a cache is used to increase the speed of processing by making current programs and data available to CPU at a rapid speed.
- ☞ Many operating systems are designed to enable the CPU to process a number of independent programs concurrently, this concept is called multiprogramming, refers to the existance of two or more programs in different parts of the memory hierachy at the same time.
- ☞ The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address. A memory unit accessed by content is called an associative memory or content addressable memory (CAM).
- ☞ Analysis of a large number of typical programs has shown that the references to memory at any given interval of time tend to be confined within a few localized areas in memory. This phenomenon is known as the property of 'locality of reference'.
- ☞ The performance of cache memory is frequently measured in terms of a quantity called hit ratio. When the CPU refers to memory and finds the word is cache, it is said to produce a hit. If the word is not found in cache, it is in main memory it counts as a miss.
- ☞ Mapping is a process of tranformation of data from the main memory to the cache memory.
- ☞ There are three types of mapping procedures which are of practical interest when consisdering the organization of cache memory. These are :
(i) Associative mapping

- (ii) Direct mapping
- (iii) Set associative mapping.

Virtual memory is a concept used in large computer systems that permits the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory. A virtual memory system provides a mechanism for translating program-generated addresses into correct main memory locations.

QUESTION-ANSWERS

Q 1. What do you mean by Associative memory and its application ?

(PTU, May 2018, 2017, 2004 ; Dec. 2016, 2015, 2014)

Ans. Associative memory also known as Content Addressable Memory (CAM), is a memory chip in which each bit position can be compared. In regular dynamic RAM (DRAM) and static RAM (SRAM) chips, the contents are addressed by bit location and then transferred to the arithmetic logic unit (ALU) in the CPU for comparison. In CAM chips, the content is compared in each bit cell, allowing for very fast table lookup. Since the entire chip is compared, the data content can often be randomly stored without regard to an addressing scheme which would otherwise be required. However, CAM chips are considerably smaller in storage capacity than regular memory chips.

Application : Associative memory will be significantly more expensive than the corresponding regular memories. Associative memory are used only in applications where the time available for the associative search is very limited.

Q 2. What is the need for a cache memory? Where is it located in a computer?

(PTU, May 2018, 2012, 2004 ; Dec. 2010, 2008, 2006)

Ans. A CPU cache is used by the central processing unit of a computer to reduce the average time to access memory. The cache is a smaller, faster memory which stores copies of the data from the most frequently used main memory locations. As long as most memory accesses are cached memory locations, the average latency of memory accesses will be closer to the cache latency than to the latency of main memory.

When the processor needs to read from or write to a location in main memory, it first checks whether a copy of that data is in the cache. If so, the processor immediately reads from or writes to the cache, which is much faster than reading from or writing to main memory.

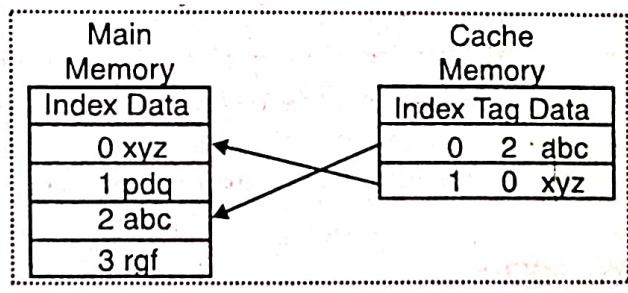


Diagram of CPU cache

The diagram above shows two memories. Each location in each memory has a datum (a cache line), which in different designs ranges in size from 8 to 512. The size of the cache line is usually larger than the size of the usual access requested by a CPU instruction, which ranges from 1 to 16 bytes. Each location in each memory also has an index, which is a unique number used to refer to that location. The index for a location in main memory is called an address. Each location in the cache has

a tag that contains the index of the datum in main memory that has been cached. In a CPU's data cache these entries are called cache lines or cache blocks. (PTU, May 2015, 2014, 2004)

Q 3. Explain Interleaved Memories ?

Ans. Memory Interleaving is a category of techniques for increasing memory speed. For example, with separate memory banks for odd and even addresses, the next byte of memory can be accessed while the current byte is being refreshed. (PTU, Dec. 2004)

Q 4. What is the role of Registers in digital computers ?

Ans. Registers are usually employed for the temporary storage of data along with the capability to shift the data bits. An n bit register has a group of n flipflops and is capable of storing and binary information of n bits. (PTU, Dec. 2004)

Q 5. How associative memory is useful in memory hierarchy ?

Ans. With Associative memory map, each bit cell is compared with the content and hence gives very fast table lookup eg : CAM. (Content Addressable Memory) (PTU, Dec. 2004)

Q 6. Explain the meaning of the memory-reference instruction BUN. (PTU, May 2005)

Ans. BUN : This Instruction transfers the program to the Instruction specified by the effective address. Remember that PC holds the address of the instruction to be read from memory in the next Instruction cycle. PC is Incremented at time T, to prepare it for the address of the next Instruction in the program sequence. The Bun Instruction allows the programmer to specify an Instruction out of sequence and we say that the program branches (or jumps) unconditionally. The Instruction is executed with one micro-operation :

$PC \leftarrow AR, SC \leftarrow 0$

Q 7. How Cache Memory is useful in memory hierarchy ? (PTU, Dec. 2009, 2005)

Ans. If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. Such a fast small memory is referred to as a Cache memory.

Q 8. Compare and contrast characteristic features of different memories. (PTU, May 2006)

Ans. Main memory access time is not uniform. It can be stored memory or local memory. Cache memory is a fast access memory but It has less storage capability.

Disk storage memory is very large than main memory but has low access time.

Q 9. How Virtual Memory is useful in memory hierarchy ? (PTU, May 2005)

Ans. In a memory hierarchy system, Programs and data are first stored in auxiliary memory. Portions of a Program or data are brought into main memory as they are needed by the CPU. Virtual Memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available equal to the totality of auxiliary memory. A virtual memory system provides a mechanism for translating program-generated addresses into correct main locations.

Q 10. Explain about main memory. (PTU, Dec. 2006)

Ans. Main Memory : The main memory is the central storage unit in a computer system. It is a relatively large and fast memory used to store programs and data during the computer operation. The Principal technology used for main memory is based on semiconductor circuits.

Q 11. Differentiate between block and pages. (PTU, May 2007)

Ans. Block : The physical memory is broken into groups equal size called blocks, which may range from 4096 words each.

Page : The term page refers to groups of address space of same size.

Q 12. Write to techniques to implement virtual memory.

(PTU, May 2007)

Ans. Techniques that automatically move program and data blocks into the physical main memory. When they are required for execution are called virtual memory techniques. Two techniques used to implement virtual and memory are

1. Virtual Demand Paging
2. Virtual Demand Segmentation.

Q 13. What is the difference between physical address and logical address ?

(PTU, May 2007)

Ans.

Physical Address	Logical Address
<ol style="list-style-type: none"> 1. The concept of a physical address space is central to proper memory management. 2. Physical address seen by the memory unit. 	<ol style="list-style-type: none"> 1. The concept of a logical address space that is bound to a separate. 2. Logical address generated by the CPU referred to as virtual address.

Q 14. Differentiate among direct mapping and associative mapping.

(PTU, Dec. 2013, 2007)

Ans. Direct Mapping : In direct mapping, associative memories are compared to random access memories because of added logic associated with each cell. The CPU address of 15 bits is divided into two fields. The nine least significant bits constitute the index field and the remaining six bits from tag field. The number of bits in the index field is equal to number of address units required to access cache memory.

Associative mapping : The fastest and most flexible cache organization uses an associative mapping. The associative memory stores both address and content of memory word. This permit any location in cache to store any word in main memory. The address value of 15 bits is five digit octal number and it corresponding 12 bits word in four digit octal number. A CPU address of 15 bits is placed in argument register and associative memory is searched for matching address.

The properties differences between the Direct mapping and Associative mapping is as follows:

Properties of Direct mapping	Properties of Associative mapping
This can be very simple.	A main memory block can load into any line of cache.
Consider the original CERN server.	Memory address is interpreted as tag and word.
Based on Unix	Tag uniquely identifies block of memory.
Poor performance.	Every line's tage is examined for a match.
Cache management and garbage collection is difficult.	Cache searching gets expensive.

Q 15. What do you mean by memory hierarchy? Briefly discuss.

(PTU, May 2010, Dec. 2008)

Ans. The hierarchical arrangement of storage in current computer architecture is called the **memory hierarchy**. It is designed to take advantage of memory locality in computer programs. Each level of the hierachy has the properties of higher bandwidth, smaller size, and lower latency than lower levels.

Most modern CPUs are so fast that for most program workloads, the locality of reference of memory accesses and the efficiency of the caching and memory transfer between different levels of the hierarchy are the practical limitation on processing speed. As a result, the CPU spends much of

its time idling, waiting for memory I/O to complete. This is sometimes called the space cost, as a larger memory object is more likely to overflow a small/fast level and require use of a larger/slower level.

Q 16. What do you mean by interleaved memory?

(PTU, Dec. 2008 ; May 2006)

OR

Explain the importance of interleave memory.

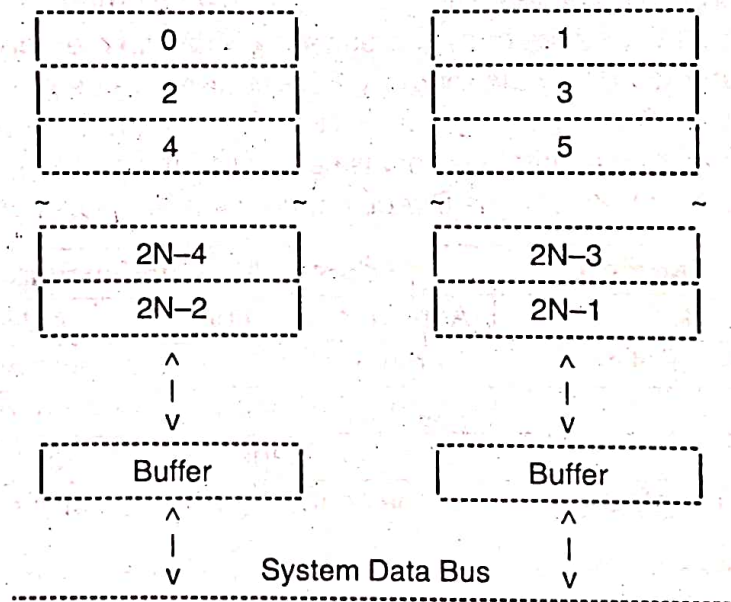
(PTU, May 2009)

Ans. Interleaved memory is a technique for compensating the relatively slow speed of DRAM. The CPU can access alternative sections immediately without waiting for memory to be cached. Multiple memory banks take turns supplying data.

An interleaved memory with "n" banks is said to be n-way interleaved. If there are "n" banks, memory location "i" would reside in bank number $i \bmod n$.

One way of allocating virtual addresses to memory modules is to divide the memory space into contiguous blocks. The CPU can access alternate sections immediately, without waiting for memory to catch up (through wait states). Interleaved memory is one technique for compensating for the relatively slow speed of dynamic RAM (DRAM). Other techniques include page-mode memory and memory caches.

In an interleaved memory system, there are still two physical banks of DRAM, but logically the system sees one bank of memory that is twice as large. In the interleaved bank, the first long word of bank 0 is followed by the first long word of bank 1, which is followed by the second long word of bank 0, which is followed by the second long word of bank 1, and so on. The below fig. shows this organization for two physical banks of N long words. All even long words of the logical bank are located in physical bank 0 and all odd long words are located in physical bank 1.



Interleaved Memory Organization

Q 17. A computer employs RAM chips of 256×8 and ROM chips of 1024×8 . The computer system needs 2k bytes of RAM, 4k bytes of ROM, and four interface units, each with four registers. A memory-mapped I/O configuration is used. The two highest-order bits of the address bus are assigned 00 for RAM, 01 for ROM, and 10 for interface registers. Draw a memory address map for the system.

(PTU, Dec. 2009, 2005, 2004)

Ans. Number of RAM chips = $\frac{2048}{256} = 8$. Therefore 3×8 decoders are needed to select each of 8 RAM Chips. Also $256 = 2^8$, first 8 lines are used as a address lines for a selected RAM.

Number of ROM Chips = $\frac{4096}{1024} = 4$. $\therefore 2 \times 4$ decoders are needed to select each of 4 ROM chips. Also $1024 = 2^{10}$ first 10 lines are used as address lines for a selected ROM.

Since each interface unit has four registers, thus 4 interface units \times 4 register = 16 registers are needed to select 16 registers, 4 lines are used. The memory address map is tabulated as follows :

Device	Hexadecimal Address	Address Bus															
Select	Address	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
RAM	0000-07FF	0	0	0	0	0	3	\times 8dec.	x	x	x	x	x	x	x	x	x
ROM	4000-4FFF	0	1	0	0	2	\times 4dec	x	x	x	x	x	x	x	x	x	x
Interface	8000-8000F	1	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x

Q 18. Information is inserted into a FIFO buffer at a rate of m bytes per second. The information is deleted at a rate of n bytes per second. The maximum capacity of the buffer is k bytes. How long does it take for an empty buffer to fill up when $m > n$ and how long does it take for a full buffer to empty when $m < n$?

(PTU, Dec. 2004)

Ans. $\therefore t = \frac{5}{4} \therefore \frac{x \times n}{k}$ where $n = 1, 2, \dots, k$.

Q 19. A computer uses RAM chips of 1024×1 capacity. How many chips are needed, and how should their address lines be connected to provide a memory capacity of 1024 bytes ? Also explain in words how the chips are to be connected to the address bus ?

(PTU, May 2005)

Ans. (a) Number of RAM chips required = $\frac{1024}{1024} = 1$

(b) Number of RAM chips required = $\frac{16K}{1024} = \frac{16 \times 1024}{1024} = 16$, $16 K = 2^4 \times 2^{10} = 2^{14}$, therefore

14 lines of the address bus must be used to access 16K bytes of memory $1024 = 2^{10}$, therefore 10 lines are required to address each chip. Remaining $14 - 10 = 4$ lines are required to decoder for selecting 16 chips. \therefore size of decoder 4×16 decoder.

Q 20. What are the advantages you got with virtual memory ?

(PTU, May 2017 ; Dec. 2006)

Ans. **Virtual memory** is the separation of user logical memory from physical memory. This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available (Fig.). Virtual memory makes the task of programming much easier, because the programmer no longer needs to worry about the amount of physical memory available, or about what code can be placed in overlays ;

In addition to separating logical memory from physical memory, virtual memory also allows files and memory to be shared by several different processes through page sharing (Section 94.5). The sharing of pages further allows performance improvements during process creation.

Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Several systems provide a paged segmentation scheme, where segments are broken into pages. Thus, the user view is segmentation, but the operating system can implement this view with demand paging. Demand segmentation can also be used to provide virtual memory. Burroughs' computer systems have used demand segmentation. The IBM OS/2 operating system also uses demand segmentation. However, segment-replacement algorithms are more complex than are page-replacement algorithms because the segments have variable sizes. We do not cover demand segmentation in this text; refer to the Bibliographical Notes for relevant references.

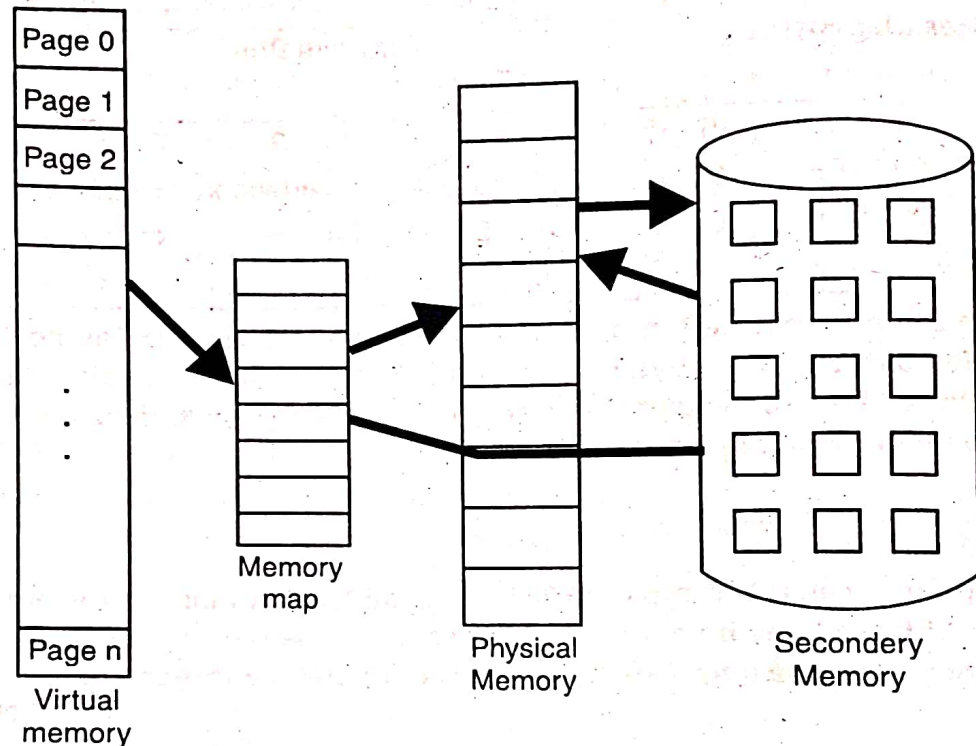
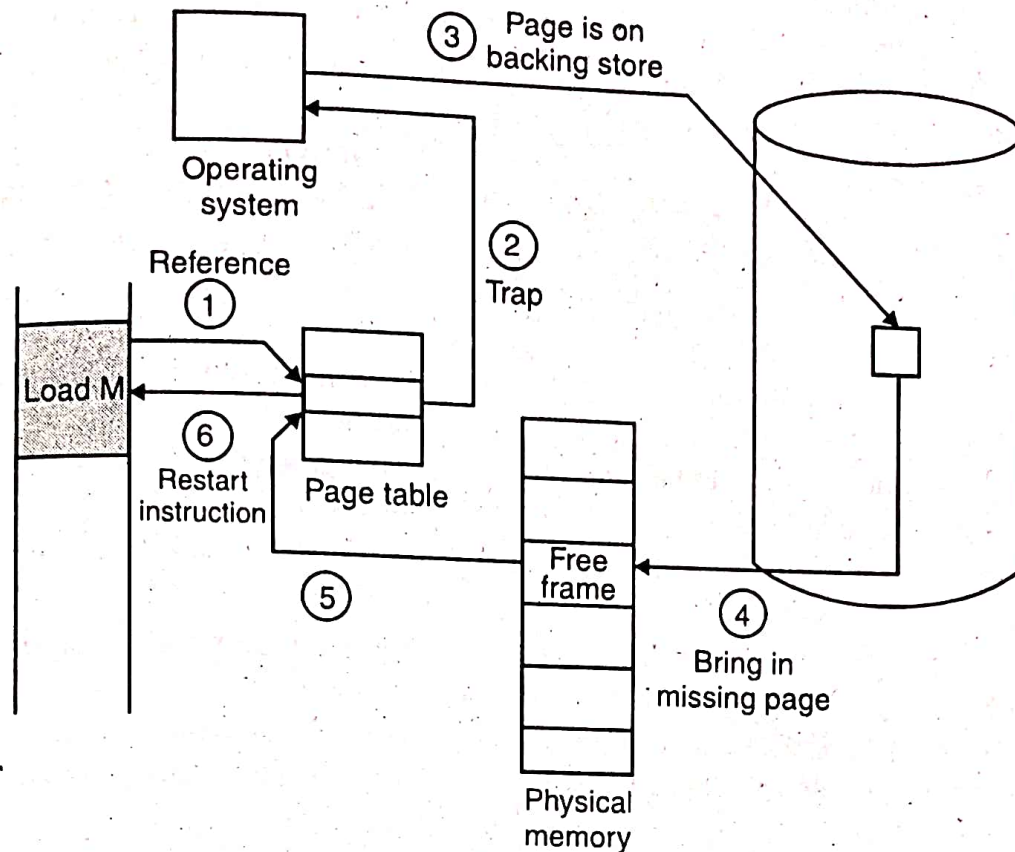


Diagram showing virtual memory that is larger than physical memory

But what happens if the process tries to access a page that was not brought into memory? Access to a page marked invalid causes a page-fault trap. The paging hardware, in translating the address through the page table, will notice that the invalid bit is set, causing a trap to the operating system. This trap is the result of the operating system's failure to bring the desired page into memory (in an attempt to minimize disk-transfer overhead and memory requirements), rather than an invalid address error as a result of an attempt to use an illegal memory address (such as an incorrect array subscript). We must therefore correct this oversight. The procedure for handling this page fault is straightforward (Fig.) :

1. We check an internal table (usually kept with the process control block) for this process, to determine whether the reference was a valid or invalid memory access.
2. If the reference was invalid, we terminate the process. If it was valid, but we have not yet brought in that page, we now page it in.
3. We find a free frame (by taking one from the free-frame list, for example).
4. We schedule a disk operation to read the desired page into the newly allocated frame.



- When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
- We restart the instruction that was interrupted by the illegal address trap. The process can now access the page as though it had always been in memory.

It is important to realize that, because we save the state (registers, condition code, instruction counter) of the interrupt process when the page fault occurs, we can restart the process in exactly the same place and state, except that the desired page is now in memory and is accessible. In this way, we are able to execute a process, even though portions of it are not (yet) in memory. When the process tries to access locations that are not in memory, the hardware traps to the operating system (page fault). The operating system reads the desired page into memory and restarts the process as though the page had always been in memory.

Performance of Demand Paging

Demand paging can have a significant effect on the performance of a computer system. To see why, let us compute the effective access time for a demand-paged memory. For most computer systems, the memory-access time, denoted m_a , now ranges from 10 to 200 nanoseconds. As long as we have no page faults, the effective access time is equal to the memory access time. If, however, a page fault occurs, we must first read the relevant page from disk, and then access the desired word.

Let p be the probability of a page fault ($0 \leq p \leq 1$). We would expect p to be close to zero; that is, there will be only a few page faults. The effective access time is then

$$\text{effective access time} = (1-p) \times m_a + p \times \text{page fault time.}$$

Q 21. What is the difference between memories mapped I/O and Isolated I/O? What are the advantages and disadvantages of each? (PTU, May 2017, 2015 ; Dec. 2016, 2013, 2007)

Ans. The differences between mapped I/O and isolated I/O memories are as follows : Many computers use one common bus to transfer data between memory, I/O and CPU. The distinction

between a memory transfer and I/O transfer made through separate read and write lines. The I/O read and I/O write control lines are enabling during a I/O transfer. The memory read write lines are enabled during memory transfer. This configuration isolates are I/O interfaces addresses from address assigned to memory and referred as isolated I/O method for assigning address in common bus.

Memory mapped I/O	Isolated I/O
<ol style="list-style-type: none"> 1. In memory mapped I/O there are no specific input or output instructions. 2. In memory mapped I/O, it uses same address space for both memory and I/O. 3. In memory mapped, at a same time it not enables I/O read and I/O write. 4. There is no specific input and output instruction. 5. The memory mapped I/O maps memory and I/O address. 6. In memory mapped I/O are instructions refers to memory are also available for I/O. 7. The advantage of memory mapped I/O is that load and store instructions are used for reading and writing from memory can used to input and output data from I/O registers. 	<ol style="list-style-type: none"> 1. In isolated I/O configuration, CPU has distinct input and output instructions each of these instruction is associated with address of an interface register. 2. In isolated I/O it uses different address space for both memory and I/O. 3. In isolates I/O, at a same time it enable I/O read (for input) and I/O write C for outputs. 4. There are specific input and output instructions. 5. Isolated I/O method isolates memory and I/O address. 6. This informs external component that the address is for a memory word and not for I/O interface. 7. In isolated I/O it places memory address on address line and enables memory read or many write control lines.

Advantages and disadvantages of Isolated I/O :

Advantages :

1. Uses separate memory space.
2. Limited instructions can be used. Those are IN, OUT, INS, OUTS.
3. Efficient I/O operations due to using separate bus

Disadvantages :

1. Comparatively larger in size
2. Uses complex internal logic
3. Slower operations.

Advantages and disadvantages of Memory mapped I/O :

Advantages :

1. Many operations, especially I/O intensive operations can be faster since content does not need to be copied between kernel space and user space.
2. Computers with memory mapped I/O can use the memory type instructions to access I/O data. It allows the computer to use the same instructions for either I/O transfers or for memory transfers.

Disadvantages :

1. Performance does not always increase with memory mapped I/O.
2. An I/O error on a memory mapped file cannot be caught.

Q 22. Explain memory management hardware.

(PTU, Dec. 2019 ; May 2015)

Ans. A memory management system is a collection of hardware and software procedures for managing the various programs residing in memory. The memory management software is part of an operating system available in many computers. Here we are concerned with the hardware unit associated with the memory management system. The basic components of memory management are:

- 1. A facility for dynamic storage relocation that maps logical memory references into physical memory address.
- 2. A provision for sharing common programs stored in memory by different users.
- 3. Protection of information against unauthorized access between users and preventing users from changing operating system functions.

The dynamic storage relocation hardware is a mapping process similar to the paging system.

Q 23. A RAM chip 4096 × 8 bits has two enable lines. How many pins are needed for the integrated circuit package? Draw a block diagram and label all input and outputs pins of the chip. What is the main feature of random access memory?

(PTU, May 2008)

Ans. 4096 × 8

(a) Number of RAM chips are required = $\frac{4096}{1024} = 4$

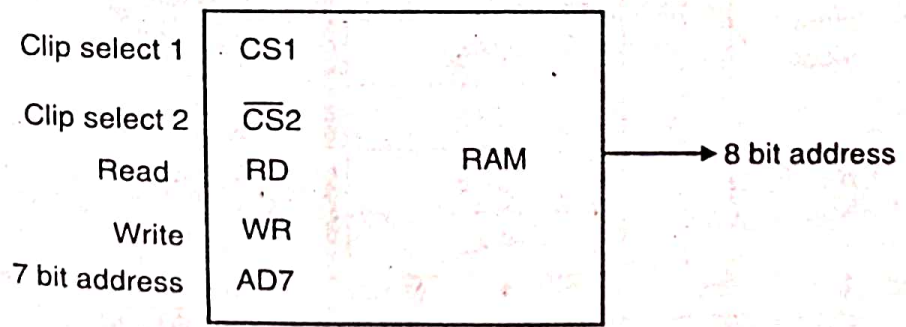
(b) Number of RAM chips/pins are required = $\frac{32K}{1024} = \frac{32 \times 1024}{1024} = 32K$
 $= 2^5 \times 2^{10} = 2^{15}$

Therefore 15 lines of address bus must be used to access 32K bytes of memory. The bytes of memory are $1024 = 2^{10}$.

Therefore, 10 address lines are required to access addresses of each chips. Remaining 15 – 10 lines are required 10 decoder for selecting 16 chips.

(c) Block diagram of RAM chip : A RAM chip is better suited for communication with CPU if it has one or more control chips inputs that select the chip only when needed. Another common feature is a bidirectional data bus that allows the transfer of data either from memory CPU during a read operation or from CPU to memory during write operation.

A bidirectional bus must be constructed with three state buffers. A three state buffer outputs can be placed in three possible states : a signal equivalent to logic 1 ; a signal equivalent to logic 0 ; a high impedance state. The logic 0 and 1 are normal digital signals. The high impedance state behaves like an open circuit, which means that output does not carry a signal and has no logic significance.



A basic block diagram of RAM

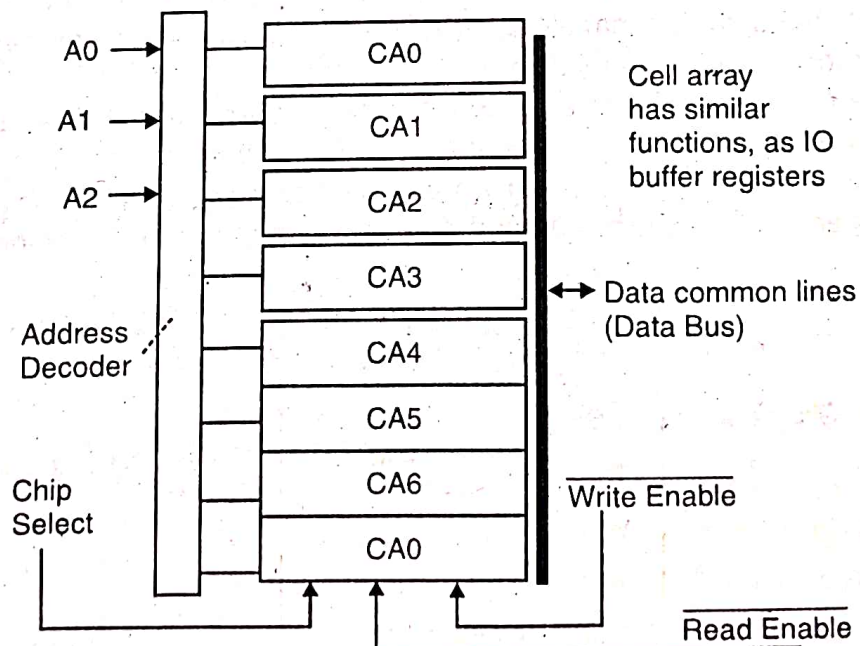
- (i) The main memory is central storage unit in a computer system.
- (ii) It is relatively large and fast used memory to store programs and data during the computer operation.
- (iii) The principal technology used for main memory is based on semi conductor integrated circuit.
- (iv) Integrated circuits RAM chips are available in two operating mode static and dynamic.
- (v) The static RAM consist of essentially internally flip-flops that stores the binary information.
- (vi) The dynamic RAM stores the binary information in form of electronic charges that are applied to capacitors.
- (vii) The dynamic RAM offers reduced power consumption and larger storage capacity in a single memory chip.
- (viii) The static RAM is easier to use and has shorter read and write cycles.
- (ix) One of major application of static RAM is in implementing cache memories.
- (x) Most of the desktop personal computer are dynamic RAM with improved performance characteristics such as multi bank DRAM, extended data out DRAM, synchronous DRAM, and direct RAM bus DRAM.

A RAM chip is better suited for communication with CPU if it has one or more control inputs that select chip only when needed. Another common feature is a bidirectional data bus that allows the transfer of data either from memory CPU during a read operation or from CPU to memory during write operation. A bidirectional data bus must be constructed with three state buffers. A three state buffer outputs can be placed in three possible states : a signal equivalent logic 1, a signal equivalence logic 0 or high impedance state. The request RAM IC discussed below :

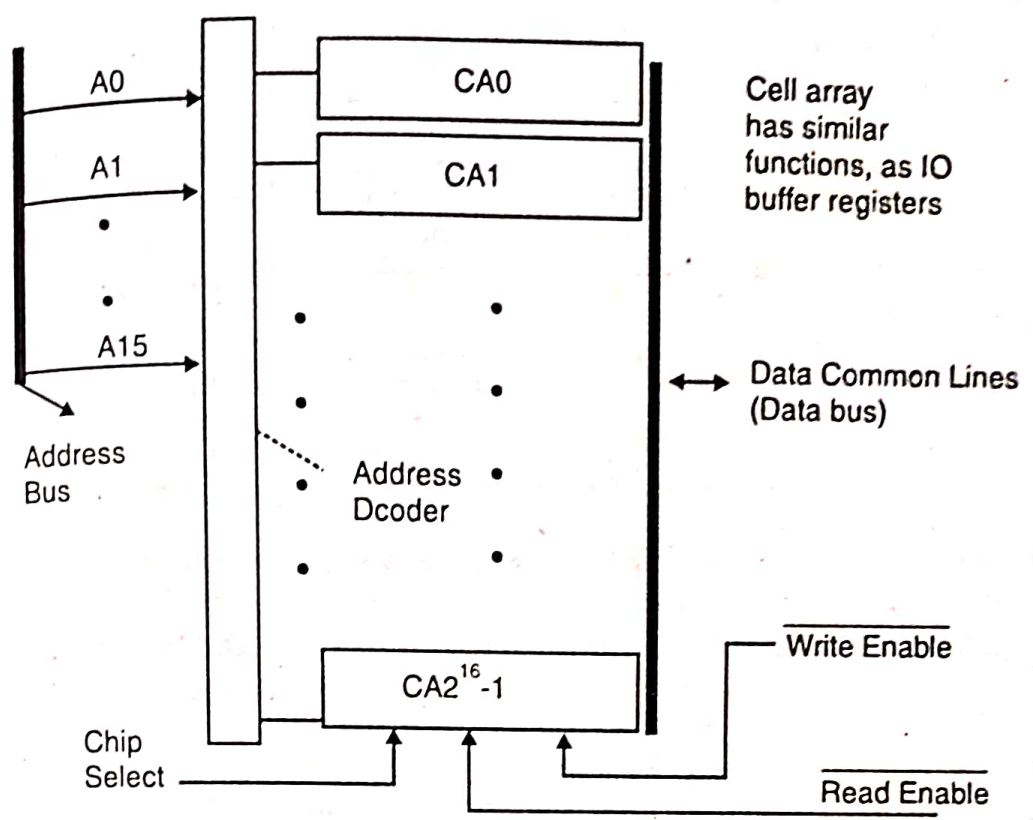
- (a) Internal address decoding
- (b) IC for RAM

Internal address decoding of RAM (Random Access Memory)

Memory Cell Arrays of 8 Bytes



2^{16} Memory Cell Arrays of 8 Bytes



Q 24. Explain the following terms :

- (a) Cache Memory
- (b) Interleaved Memories
- (c) Associative Memory

(PTU, May 2018, 2017, 2004 ; Dec. 2017, 2016, 2015)

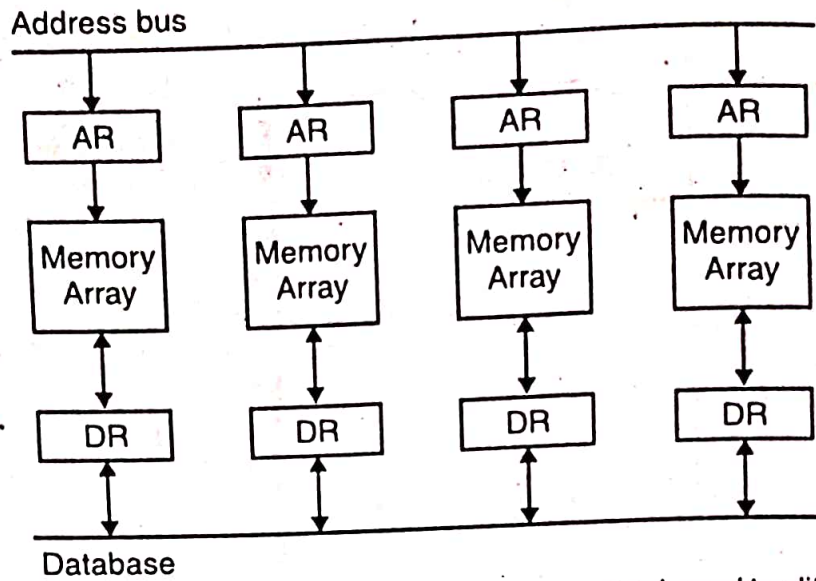
Ans. (a) **Cache Memory** : Cache memory is small, very high speed memory connected directly to CPU. It stores currently needed instructions and data. Its access time is 10 nanoseconds and its capacity 2-3% of main memory.

When the CPU needs to access memory, the cache is examined. If the word is found in cache, it is read from the fast memory. If the word addressed by CPU isn't found in the cache, main memory is accessed to read the word. The performance of cache memory is frequently measured in terms of a quality called hit ratio. When CPU refers to memory and finds the word in cache, it is said to produce a hit. If the word isn't found in the cache, it is said to have a miss. The ratio of no. of hits divided by total CPU references to memory is hit ratio. Hit ratio of 0.9 & higher have been reported.

The basic characteristic of cache memory is its fast access time thus very little or no time must be wasted when searching for words in cache. The transfer time of data from main memory to cache memory is referred to as a mapping process. There are three types of mappings:

- (1) Associative mapping
- (2) Direct mapping
- (3) Set associative mapping

(b) **Interleaved memories** : In case of interleaved memory, the memory can be partitioned into a number of modules connected to a common memory address and data bus. A memory module is a memory array together with its own address and data registers. Each memory array has its own address register (ARR) and data register (DR). The address register receives information from a common address bus and the data register receives information via a bidirectional data bus. The 2 L.S.B's of address can be used to distinguish between 4 modules.

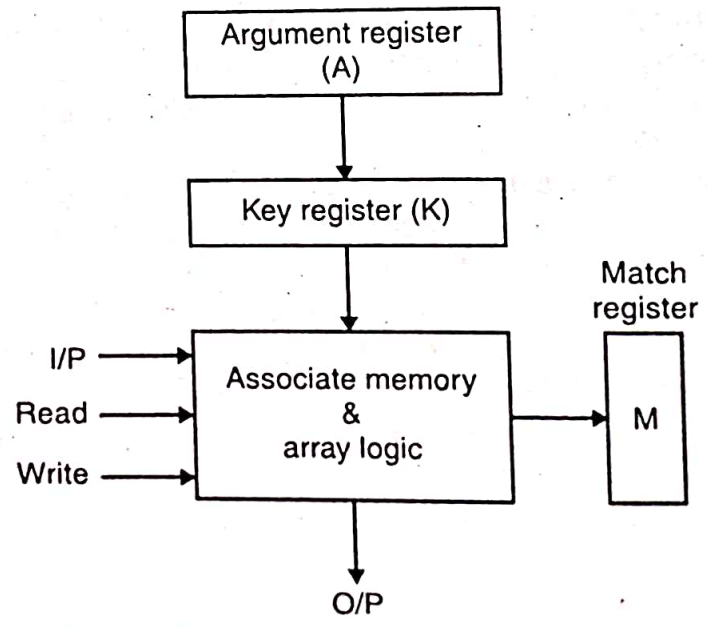


In an interleaved memory, different sets of addresses are assigned to different memory modules. A modular memory is useful in system with pipeline and vector processing.

(c) Associative memory : The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by content of data itself rather than by an address.

A memory unit accessed by content is called an associative memory or content addressable memory (CAM). When a word is written in an associative memory, no address is given when a word is to be used from an associative memory, the content of word, or part of word, is specified.

An associative memory is more expensive than a RAM because each cell must have storage capability as well as logic circuit for matching its content with external original.



Block diagram of Associative memory

Q 25. How many bits of storage are required for tag array of 32-KB cache with 256-byte cache lines and four-way-set associativity if the cache is write-back but does not require any additional bits of data in the tag array to implement the write-back policy? Assume that the system containing the cache uses 32-bit addresses.

Ans. A 32-KB cache with 256-bytes lines contain 128 lines. Since cache is four way set

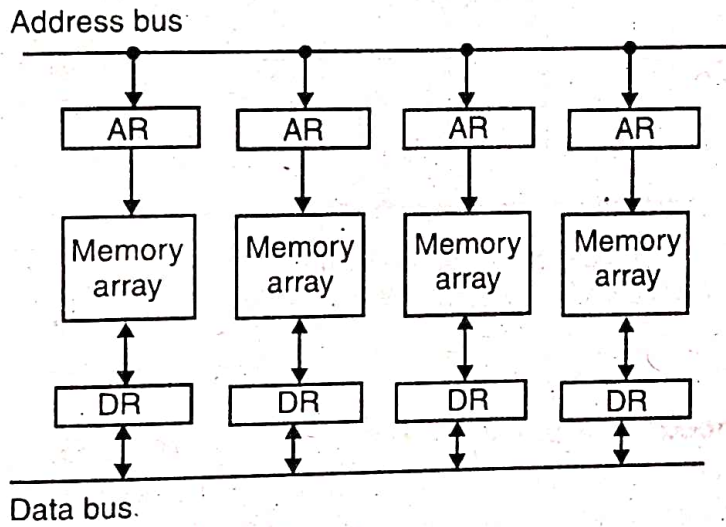
(PTU, May 2005 ; Dec. 2004)

associative, it has 32 sets (ie $2^5 = 32$) and require 5 bits. Lines that are 256 bytes long means require 8 bits ($2^8 = 256$). So 13 bits of the address are used to select a set and determine the byte within the line that an address points to therefore, the tag field of each tag array entry is $32 - 13 = 19$ bits long. Adding 2 bits for the duty and valid bits, we get 21 bits per tag array. Multiplying by the 128 lines in the cache gives 2688 bits of storage in the tag array.

Q 26. What is memory interleaving ? How is it different from Cache memory ?

(PTU, Dec. 2006)

Ans. Memory Interleaving : Pipeline and vector processors often require simultaneous access to memory from two or more sources. An instruction pipeline may require the fetching of an instruction and an operand at the same time from two different segments. Similarly, an arithmetic pipeline usually requires two or more operands to enter the pipeline at the same time. Instead of using two memory buses for simultaneous access, the memory can be partitioned into a number of modules connected to a common memory address and data buses. A memory module is a memory array together with its own address and data registers. Fig. shows a memory unit with four medules. Each memory array has its own address register AR and data register DR. The address registers receive information from a common address bus and the data registers communicate with a bidirectional data bus. The two least significant bits of the address can be used to distinguish between the four modules. The modular system permits one module to initiate a memory access while other modules are in the process of reading or writing a word and each module can honor a memory request indenpent of the state of the other moules.



Multiple module memory organisation

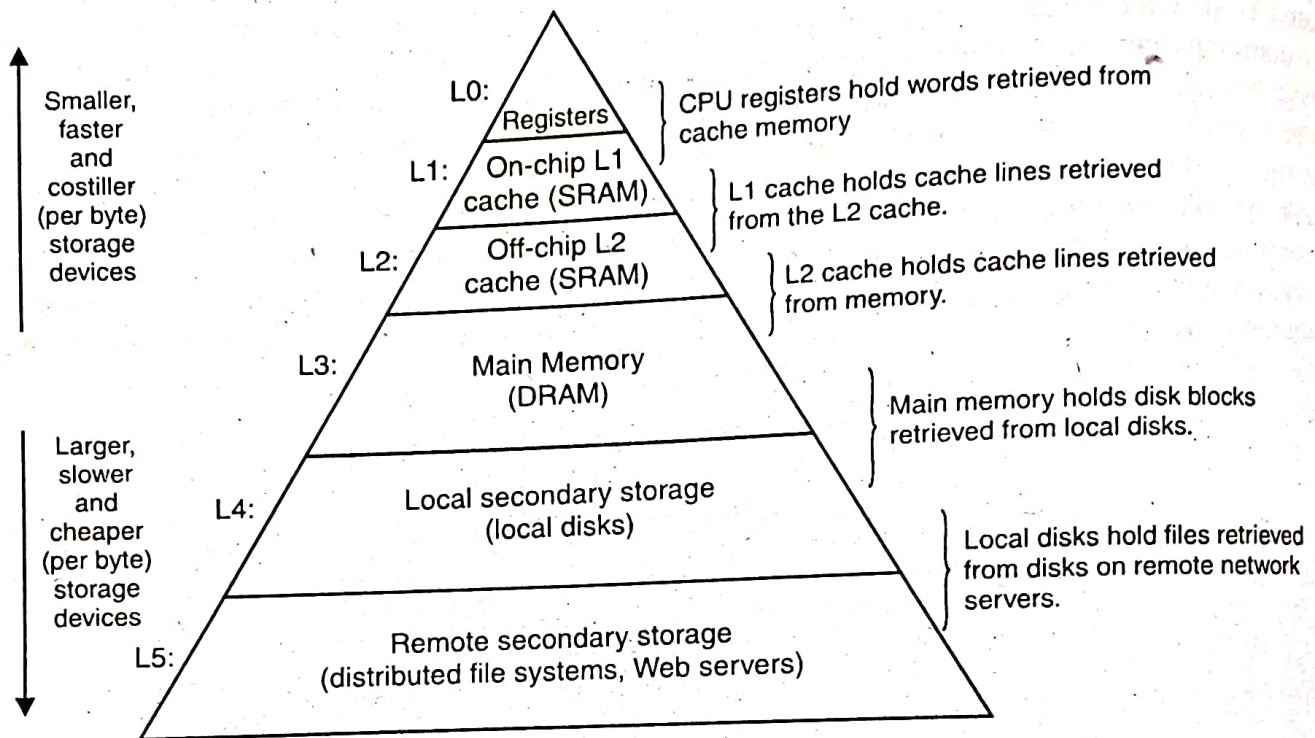
The advantage of a modular memory is that it allows the use of a technique called interleaving. In an interleaved memory, different sets of addresses are assigned to different memory modules. For example, in a two-module memory system, the even addresses may be in one module and the odd addresses in the other. When the number of modules is a power of 2, the least significant bits of the address select a memory module and the remaining bits designate the specific location to be accessed within the selected module.

A modular memory is useful in systems with pipeline and vector processing. A vector processor that uses an n-way interleaved memory can fetch n operands from n different modules. By staggering the memory access, the effective memory cycle time can be reduced by a factor close to the number of modules. A CPU with instruction pipeline can take advantage of multiple memory modules so that each segment in the pipeline can access memory independent of memory access from other segments.

Q 27. What is memory hierarchy? Explain with the help of diagram. (PTU, Dec. 2019 ; May 2016, 2009)

Ans. The hierarchical arrangement of storage in current computer architectures is called the **memory hierarchy**. Each level of the hierarchy is of higher speed and lower latency, and is of smaller size, than lower levels.

Most modern CPUs are so fast that for most program workloads the **locality of reference** of memory accesses, and the efficiency of the **caching** and memory transfer between different levels of the hierarchy, is the practical limitation on processing speed. As a result, the CPU spends much of its time idling, waiting for memory I/O to complete.



Q 28. What is the need of virtual memory? (PTU, Dec. 2019, 2014 ; May 2009)

Ans. **Virtual memory** is a memory management technique developed for multitasking kernels; this technique virtualizes a computer architecture's various hardware memory devices (such as RAM modules and disk storage drives).

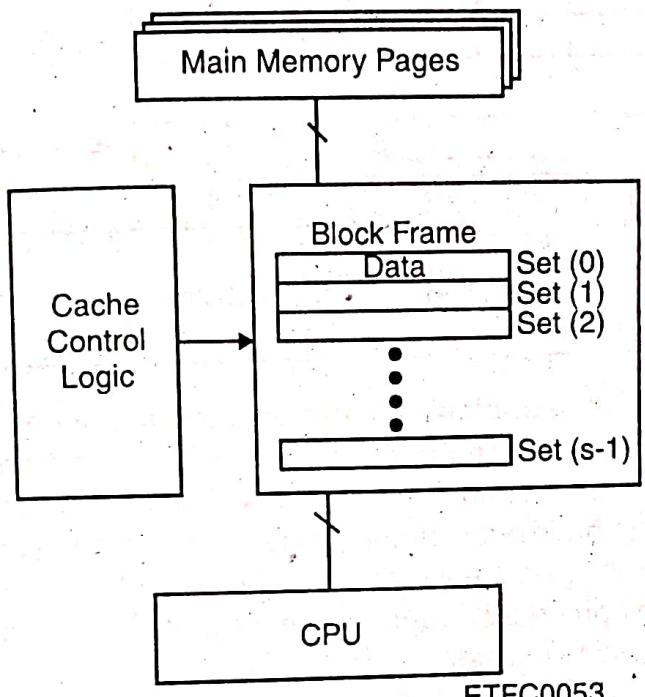
There is a lot of examples where VM is necessary :

- **No memory limit** : a virtual memory helps OS and applications to works with huge virtual memory whatever the physical memory capacity is.
- **File mapping** : a way not to be forced to load all the unnecessary parts of a file on memory, a way to work on a file like a memory block. A lot of databases used to work on very huge files though file mappings.
- **Shared access memory mapping** : a way to let processes to share a memory with the same or different access rights, with the same or different content for each block of this memory. An example is the data you share when forking a process, the data of the new process would be shared until it is modified (Copy-On-Write mechanism).

Q 29. Discuss the various mapping schemes used in cache design. Compare the schemes in terms of cost and performance. (PTU, Dec. 2016 ; May 2015, 2011, 2010)

Ans. Cache Mapping Techniques : Cache mapping is the method by which the contents of main memory are brought into the cache and referenced by the CPU. The mapping method used directly affects the performance of the entire computer system.

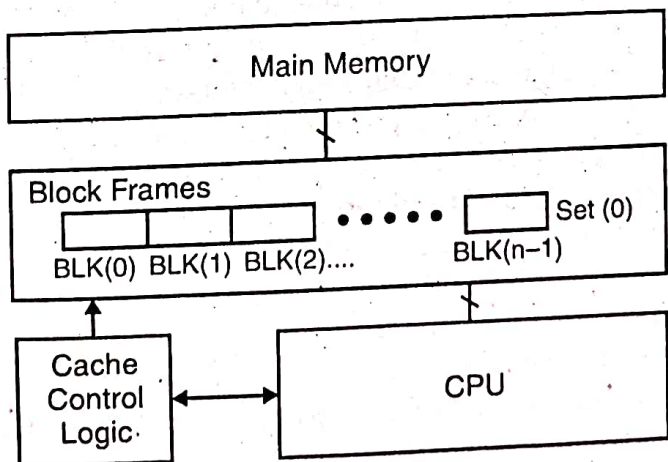
1. Direct mapping : Main memory locations can only be copied into one location in the cache. This is accomplished by dividing main memory into pages that correspond in size with the cache.



ETFC0053

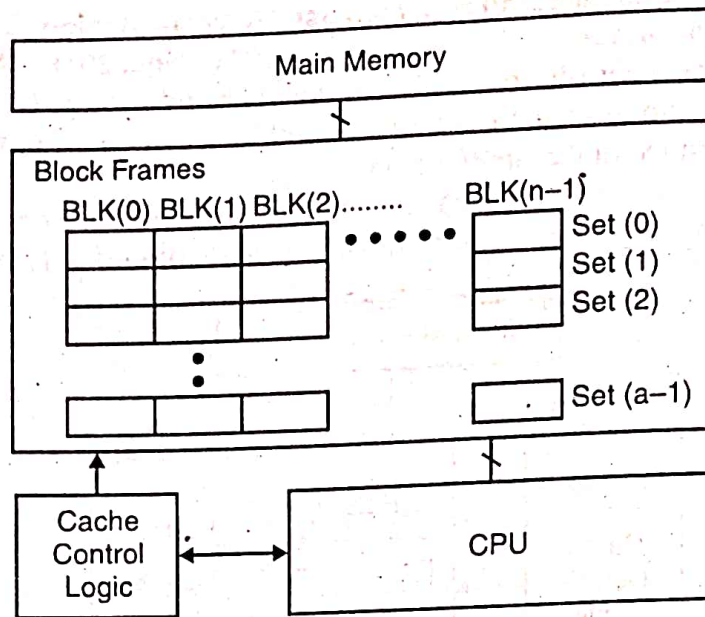
Example of direct mapping used in cache memory

2. Fully associative mapping : Fully associative cache mapping is the most complex, but it is most flexible with regards to where data can reside. A newly read block of main memory can be placed anywhere in a fully associative cache. If the cache is full, a replacement algorithm is used to determine which block in the cache gets replaced by the new data.



Example of fully associated mapping used in cache memory

3. Set associative mapping : Set associative cache mapping combines the best of direct and associative cache mapping techniques. As with a direct mapped cache, blocks of main memory data will still map into as specific set, but they can now be in any N-cache block frames within each set.



Example of set association mapping used in cache memory

Comparison of Cache Mapping Techniques : There is a critical tradeoff in cache performance that has led to the creation of the various cache mapping techniques described in the previous section. In order for the cache to have good performance you want to maximize both of the following:

- **Hit Ratio :** You want to increase as much as possible the likelihood of the cache containing the memory addresses that the processor wants. Otherwise, you lose much of the benefit of caching because there will be too many misses.
- **Search Speed :** You want to be able to determine as quickly as possible if you have scored a hit in the cache. Otherwise, you lose a small amount of time on every access, hit or miss, while you search the cache.

Now let's look at the three cache types and see how they differ :

- **Direct Mapped Cache :** The direct mapped cache is the simplest form of cache and the easiest to check for a hit. Since there is only one possible place that any memory location can be cached, there is nothing to search ; the line either contains the memory information we are looking for, or it doesn't. Unfortunately, the direct mapped cache also has the worst performance, because again there is only one place that any address can be stored. Let's look again at our 512 KB level 2 cache and 64 MB of system memory. As you recall this cache has 16,384 lines (assuming 32-byte cache lines) and so each one is shared by 4,096 memory addresses. In the absolute worst case, imagine that the processor needs 2 different addresses (call them X and Y) that both map to the same cache line, in alternating sequence (X, Y, X, Y). This could happen in a small loop if you were unlucky. The processor will load X from memory and store it in cache. Then it will look in the cache for Y, but Y uses the same cache line as X, so it won't be there. So Y is loaded from memory, and stored in the cache for future use. But then the processor requests X, and looks in the cache only to find Y. This conflict repeats over and over. The net result is that the hit ratio here is 0%. This is a worst case scenario, but in general the performance is worst for this type of mapping.
- **Fully Associative Cache :** The fully associative cache has the best hit ratio because any line in the cache can hold any address that needs to be cached. This means the problem seen in the direct mapped cache disappears, because there is no dedicated single line that an address must use. However (you knew it was coming), this cache suffers from problems involving searching the cache. If a given address can be stored in any of 16,384 lines, how do you know where it is? Even

with specialized hardware to do the searching, a performance penalty is incurred. And this penalty occurs for all accesses to memory, whether a cache hit occurs or not, because it is part of searching the cache to determine a hit. In addition, more logic must be added to determine which of the various lines to use when a new entry must be added (usually some form of a "least recently used" algorithm is employed to decide which cache line to use next). All this overhead adds cost, complexity and execution time.

- **N-Way Set Associative Cache** : The set associative cache is a good compromise between the direct mapped and set associative caches. Let's consider the 4-way set associative cache. Here, each address can be cached in any of 4 places. This means that in the example described in the direct mapped cache description above, where we accessed alternately two addresses that map at the same cache line, they would now map to the same cache set instead. This set has 4 lines in it, so one could hold X and another could hold Y. This raises the hit ratio from 0% to near 100%! Again an extreme example, of course. As for searching, since the set only has 4 lines to examine this is not very complicated to deal with, although it does have to do this small search, and it also requires additional circuitry to decide which cache line to use when saving a fresh read from memory.
- Some form of LRU (least recently used) algorithm is typically used.

Here's a summary table of the different cache mapping techniques and their relative performance:

Cache Type	Hit Ratio	Search Speed
Direct Mapped	Good	Best
Fully Associative	Best	Moderate
N-Way Set Associative, $N > 1$	Very Good, Better as N Increases	Good, Worse as N Increases

Q 30. What is logical address?

Ans. The address generated by the segmented program is called a logical address.

Q 31. Explain the different levels of cache.

Ans. **Level 1 (Primary) Cache** : Level 1 or primary cache is the fastest memory on the PC. It is in fact, built directly into the processor itself. This cache is very small, generally from 8 KB to 64 KB, but it is extremely fast. It runs at the same speed as the processor. If the processor request information and can find it in the level 1 cache, that is the best case because the information in these is immediately and the system does not have to wait.

Level 2 Cache : In level 2 cache, cache memory that is external to the micro processor. It is also called the secondary cache, resides on a separate chip from the micro processor chip. Although micro processor are including level 2 cache into their architectures.

Q 32. What is a segment?

Ans. A segment is a logically related instructions or data elements associated with a given name. Segments may be generated by the programmer or by the operating system. Example of segments are a subroutine an array of data, a table of symbols or a user's program.

Q 33. What do you understand by locality of reference? How is it helpful in improving the performance of memory? Discuss with example. (PTU, Dec. 2011)

Ans. **Locality of reference** is a phenomenon that can be seen in Computer Science. It says that the same value, or the place where it is stored is accessed often. There are two different kinds of locality of reference :

- **Temporal locality** : If a value is referenced, there is a high probability it will be referenced again a short time from now.
- **Spatial locality** : A value is stored in a certain location in storage (memory, disk, ...). If this location is referenced, it is very probably that locations that are physically close to do it will be referenced as well.

There are many reasons why locality of reference occurs :

- ❑ **Predictability** : Well-written programs only behave in a predictable way.
- ❑ **Structure of the program** : Often programs act on several items, one at a time. This item will be accessed more than once as computation or processing is done.
- ❑ **Linear data structures** : Programs may have loops that act on arrays or lists. They reference an item by its index in the array or list.

Q 34. What a main memory of a computer consists of?

(PTU, Dec. 2010)

Ans. The memory unit that directly communicates with the CPU is called the main memory. Main memory or primary memory is a fast memory storing data and code for the CPU to process. It can be used for both reading and writing and is not sequential i.e. the contents can be accessed in any order, hence the name RAM (Random Access Memory). Modern memories usually DRAM (Dynamic Random Access Memory) while in the good old days they were core memories, working with magnetic induction.

Both the cache and the main memory are short-term or volatile memories. They are not capable of holding the stored information after the computer has been switched off. Non-volatile memories or storage devices as some are mostly called come in several different flavours.

Q 35. How many 128×8 memory chips are needed to provide a memory capacity of 4096×16 ?

(PTU, May 2011)

Ans. The total energy capacity = $4096 \times 16 = 2^{12} \times 2^4$

The capacity of one RAM chip = $128 = 2^7$

The number of memory chip needed = $2^{12} \times \frac{2^4}{2^7} = 512$ chips.

Q 36. Cache size is 64 KB, Block size is 32B and the cache is two-way set associative. For a 32 bit physical address, give the division between block offset, indese and tag.

(PTU, May 2011)

Ans. $64K/32 = 2000$ blocks

2 way set assoc – $2000 = 1000$ lines = 10 bits for index.

Q 37. List the various memories in order of their features and discuss their comparison.

(PTU, Dec. 2010)

Ans. Cache : The cache memory is the fastest memory. It is used as a high-speed buffer between main memory and CPU, storing instructions and data the CPU is likely to need in the near future. The cache +is constructed with semiconductor technology.

Main Memory : Main memory or primary memory is fast memory storing data and code for the CPU to process. It can be used for both reading and writing and is not sequential, i.e. the contents can be accessed in any order, hence the name RAM (Random Access Memory). Modern memories are usually DRAM (Dynamic Random Access Memory) while in the good old days they were core memories, working with magnetic induction, both the cache and the main memory are short term or volatile memories. They are not capable of holding the stored information after the computer has been switched off. Non volatile memories or storage devices as some are mostly called come in several different flavours.

Magnetic Discs : Magnetic discs hold their information even when the computer has been shut off. These are often called secondary memory or hard discs and is the internal long-term storage form used in most computers.

Other : Other kinds of non-volatile storage devices/memories are :

- Magnetic tapes, often used for systems backup.
- CD ROM (Compact Disc Read Only Memory). The CD-ROM platter is very fast and has

high capacity the disadvantage is the incapability of dynamically writing to the disc.

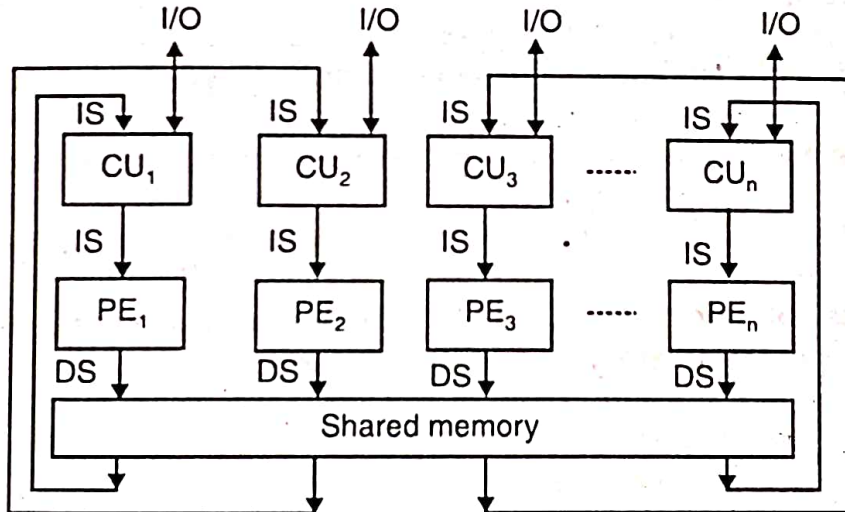
- Writeable CD ROM
- Magnet-optical discs
- Main memory (RAM or ROM).

Q 38. Discuss Virtual memory. (PTU, Dec. 2017, 2016, 2014, 2010 ; May 2016, 2014)

Ans. Virtual Memory : In a memory hierarchy system, programs and data are first stored in auxiliary memory. Portions of a program or data are brought into main memory as they are needed by the CPU. Virtual memory is a concept used in some large computer system that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory. Each address that is referred by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory. Virtual memory is used to give programmes the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory.

Q 39. Write note on the MIMD machines. (PTU, Dec. 2010)

Ans. MIMD Machines : MIMD (Multiple instruction stream, multiple data stream) is the organization of a single computer containing multiple processors connected with multiple control units and a shared memory unit. All processing elements (PEs) receive the different instruction stream (IS) of a program from the control unit (CUs) and operate simultaneously over the multiple data streams (DS). The shared memory unit contain multiple modules so that it can communicate with all the processors simultaneously. Therefore MIMD organization refers to a computer system capable of processing several programs at the same time.



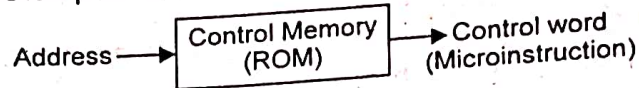
Q 40. What is the difference between Cache size and block size ?

Ans. The storage array's controller organizes its cache into blocks which are chunks of memory that can be 4, 8, 16 or 32 kbs in size. All volumes on the storage system share the same cache space, therefore, the volume can have only one cache block size. Cache blocks are not the same as the 512 byte blocks that are used by the logical block system of the disks. Applications use different blocks sizes, which can have an impact on storage performance. By default, the block size in system Manager is 8KB but you can set the value to 4, 8, 16 or 32 KiBs. A smaller size is a good choice for file systems or database applications. A large size is a good choice for applications that require large data transfers sequential I/O, or high bandwidth, such as multimedia.

Q 41. What do you mean by control memory? (PTU, Dec. 2013)

Ans. The control memory holds a fixed microprogram that can not be altered by the occasional user. The microprogram consists of microinstructions that specify various internal control signals for

execution of register microoperations. Each machine instruction initiates a series of microinstructions in control memory. These microinstructions generate the microoperations to fetch the instruction from main memory; to evaluate the effective address, to execute the operation specified by the instruction, and to return control to the fetch phase in order to repeat the cycle for the next instruction.



Computer Organization

The control memory is assumed to be a ROM.

Q 42. Give difference between register and memory.

(PTU, Dec. 2013)

Ans. Registers are storage locations internal in the processor. CPU instructions operate on these values directly. On RISC processors, all data must be moved into a register before it can be operated. On CISC chips, there are a few operations that can load data from RAM, process it and save the result back out, but the fastest operations work directly with registers.

Also, there are registers that are set aside for certain tasks, these generally include a program counter, stack and flags.

Each register also has a size that determines the maximum amount of data that can be processed at a time. The registers on Pentium chips, for example are 32 bits.

Memory, or RAM is located external to the CPU. Data has to be loaded into a CPU register from memory before the CPU can process it. RAM is much slower than registers, there is a lot more RAM than registers, and generally memory can be addressed on a byte boundaries, where registers may not be able to access all the bytes in a register.

Other words, Registers are temporary storage in the CPU that holds the data the processor is currently working on, while RAM holds the program instructions and the data the program requires.

Q 43. What is the difference between access time and cycle time of a memory?

(PTU, May 2014)

Ans. Access time : Access time is the time between the read is requested and the desired word arrives.

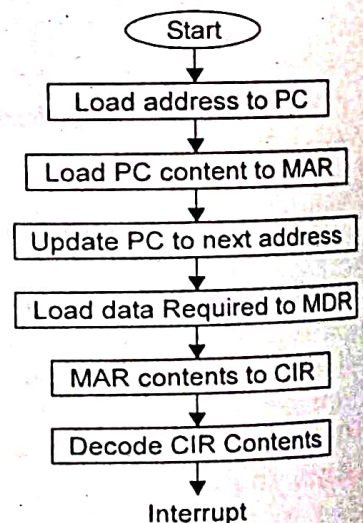
Cycle time : Cycle time is the minimum time between requests to memory (cycle time > access time because need for stabilization of address lines).

Q 44. The address of a memory location to be accessed is in register R_1 , and the memory data are to be loaded into Register R_2 . Specify the sequence of operations involved.

(PTU, May 2014)

Ans. Sequence of operation :

- Fetching the instruction of R_1 from PC.
- Decode the instruction of R_1 .
- The effective address is read from an indirect address.
- Execute the instruction.



Q 45. Why page-table is required in a virtual memory system. Explain different ways of organizing a table. (PTU, Dec. 2017 ; May 2014)

Ans. In any computer the address space is larger than memory space i.e. secondary memory is larger than the main memory, physically available to processor for execution of program. So programs and data are transferred to and from auxiliary memory and main memory based on demand imposed by the CPU. As the address of virtual memory is of larger bit then that of main memory, so mapping technique is required. To obtain the actual main memory address of the data from its virtual memory address. For this purpose a page table is required which holds the page number of virtual memory and the block numbers of the main memory. Further, each word of page table also has 'presence bit' to denote whether this page is presently available in main memory or not.

The different ways of organizing a page table are :

(i) **In the R/W memory** : It is called memory page table. But it is inefficient w.r.t. storage utilization and it required two main memory references to read a data, thus reducing the speed of execution of program.

(ii) **By using associative logic** : It is more efficient way to organize the page table, as it can be constructed with no. of words equal to no. of blocks in main memory.

Q 46. Name two types of memory interleaving.

Ans. High - Order
Low - Order

(PTU, Dec. 2014)

Q 47. Define latency time.

Ans. Latency : Latency is a time measure of the communication overhead incurred between

(PTU, Dec. 2014)

machine subsystems. For example, the memory latency is the time required by a processor to access the memory. Or in 2nd example, the hard disk platters are spinning around at high speed, and the spin speed is not synchronized to the process that moves the read/write heads to the correct cylinder on a random access on the hard disk. Therefore, at the time that the heads arrive at the correct cylinder, the actual sector that is needed may be anywhere. After the actuator assembly has completed its seek to the correct track, the drive must wait for the correct sector to come around to where the read/write heads are located. This time is called latency. Latency is directly related to the spindle speed of the drive and such is influenced solely by the drive's spindle characteristics. This operation page discussing spindle speeds also contains information relevant to latency.

Conceptually, latency is rather simple to understand it is also easy to calculate. The faster the disk is spinning, the quicker the correct sector will rotate under the heads, and the lower latency will be. Sometimes the sector will be at just the right spot when the seek is completed, and the latency for that access will be close to zero. Sometimes the needed sector will have just passed the head and in this "worst case", a full rotation will be needed before the sector can be read. On average, latency will be half the time it takes for a full rotation of the disk.

Q 48. Why does the DMA gets priority over CPU when both request memory transfer?

(PTU, Dec. 2014)

Ans. Since the data transfer rate using DMA is quite higher than the CPU and memory transfer rate, the DMA have priority over the CPU when both request a memory transfer.

Q 49. An address space is specified by 24 bits and corresponding memory space by 16 bits. How many words are there in address space and in memory space ? (PTU, May 2015)

Ans. Address space = 24 bits
 $2^{24} = 2^4 \cdot 2^{20} = 16M$ words

Memory space : 16 bits
 $2^{16} = 64$ K words.

Q 50. Write register transfer sequence for read and write from memory. (PTU, May 2015)

Ans. Read : $DR \leftarrow M[AR]$
Write : $M[AR] \leftarrow R_1$

Q 51. The address of a memory location to be accessed is in register R_1 and the memory data are to be loaded into register R_2 . Specify the sequence of operations involved. (PTU, May 2016)

Ans. LDR $R_2, [R_1]$

Will load R_2 with the contents of memory pointed at by register R_1 .

Q 52. What are the registers generally contained in the processor? (PTU, Dec. 2017 ; May 2018, 2015)

- Ans.** MAR – Memory address register
- MDR – Memory data register
- IR – Instruction register
- R_0-R_n – General purpose register
- PC – Program counter.

Q 53. What is the role of registers in digital computers? (PTU, Dec. 2017, 2016)

Ans. Registers are groups of flipflops where each flipflop is capable of storing one bit of information. The basic function of a register is to hold information in a digital system and make it available to the logic elements for the computing process.

Q 54. Explain how cache memory is different from virtual memory. Also, discuss various page replacement policies for virtual memory with examples. (PTU, May 2016)

Ans. 1. Cache memory is a type of memory used for improving the main memory access time. It is a faster type of memory that resides between CPU and RAM to reduce the average memory access latency. Virtual memory is a memory management method where it is a concept that lets program get its own virtual memory space, which is even larger than the real physical RAM available.

2. Cache memory is a type of hardware memory that actually exists physically. On the other hand, there is no hardware called virtual memory as it is a concept that uses RAM, harddisk, Memory management unit and software to provide a virtual type of memory.

3. Cache memory management is done fully by hardware. Virtual memory is managed by the operating system.

4. Virtual memory involves data structures such as page tables but this type of data structures is not necessary for cache memory.

5. Cache memories take small sizes such as KB and MB. Virtual memory on the other hand involves huge sizes that take GBs.

Page Replacement Policies

1. Static Replacement Algorithms : The static paging algorithms implement the replacement policy when the frame allocation to a process is fixed.

(a) FIFO (First in First out) Replacement : On the page fault, the frame that has been in memory the longest is replaced.

Page reference stream :

1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
1	1	1	1	1	2	2	3	5	1	6	6	2	5	5	3	3	1	6	2
	2	2	2	2	3	3	5	1	6	2	2	5	3	3	1	1	6	2	4
		3	3	3	5	5	1	6	2	5	5	3	1	1	6	6	2	4	3

FIFO

Total 14 page faults.

(b) Optimal Replacement : The Belady's optimal algorithm cheats. It looks forward in time to see which frame to replace on a page fault. This it is not a real replacement algorithm. It gives us a frame of reference for a given static frame access sequence.

Page reference stream :

1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
1	1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	2	2	2
	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	4	4
		3	3	3	5	5	5	5	5	5	5	3	3	3	3	3	3	3	3

Optimal – Total 9 Page faults.

(c) **Least Recently used (LRU) Replacement** : On a page fault, the frame that was least recently used is replaced.

Page reference stream :

1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
1	1	1	1	3	2	1	5	2	1	6	2	5	6	6	1	3	6	1	2
	2	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4
		3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3

LRU

Total 11 page Faults

2. LRU approximation : Pages with a current copy on disk are the first choice for pages to be removed. When more memory is needed to facilitate page replacement algorithm, a table of valid or invalid bits is maintained. With each page table entry a valid-invalid bit is associated.

- 1 (in memory)
- 0 (not in memory)
- Initially valid-invalid but is set to 0 on all entries.
- In idle times, dirty frames are copied to disk.
- Additional reference bit whether the frame was recently referenced. The reference bits are refreshed on a periodic basis.

Q 55. What is the relation between address and memory space in a virtual memory system? (PTU, Dec. 2016)

Ans. In a memory hierarchy system, programs and data are first stored in auxiliary memory. Portions of a program or data are brought into main memory as they are needed by the CPU. Virtual memory is a concept used in some large computer system that permits the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory. Each address that is referred by the CPU goes through an address mapping from the so-called virtual address to a physical address in main-memory. Virtual memory is used to give programmes the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory.

Q 56. Explain any five memory reference instructions in detail. (PTU, May 2017)

Ans. Memory reference instructions are those commands or instructions which are used to make a reference to the memory and allow a program to access the required data from where the data is stored. The following are five memory reference instructions.

1. LDA : This instruction is used to load the accumulator with a value that is located at a specific memory location.

Syntax	Opcode	Operand	Description
Eg	LDA	16 bit address	Load the accumulator
	LDA	2500 H	

This command says that load the value that is stored at the memory location 2500 into the accumulator. H represents that the entered address is a hexadecimal memory address.

LORDS MODEL TEST PAPERS

(Unsolved)

LORDS MODEL TEST PAPER – 1

Instruction to Candidates :

1. Section – A is Compulsory consisting of TEN questions carrying TWO marks each.
2. Section – B contains FIVE questions carrying FIVE marks each and students have to attempt any FOUR questions.
3. Section – C contains THREE questions carrying TEN marks each and students have to attempt any TWO questions.

SECTION – A

- Q 1. (a) Discuss Virtual memory.
 (b) What is the difference between machine and instruction cycles?
 (c) What is the role of ROM memory in a computer system?
 (d) What is the role of shift register in digital Computer?
 (e) What do you understand by micro-programming ? Explain.
 (f) Differentiate between RISC and CISC processors.
 (g) Difference between computer architecture and computer organization.
 (h) What do you understand by transaction processing benchmarks?
 (i) What is the need for a cache memory? Where is it located in a computer?
 (j) Mention the limitations of 8085.

SECTION – B

- Q 2. (a) Explain about the multiprocessor.
 (b) Explain the various Addressing modes in detail.
- Q 3. Give five examples of external interrupt and internal interrupt.
- Q 4. (a) Draw the block diagram of computer and explain its various components.
 (b) What is meant by DMA?
- Q 5. What are multilevel memory systems? Explain with the help of a diagram.
- Q 6. Explain any five memory reference instructions in detail.

SECTION – C

- Q 7. What is the difference between static and dynamic network interconnections? Discuss any three network interconnection networks.
- Q 8. What is the need of control unit in computer? Draw the control unit of a basic computer. Discuss how fetch and decode phases are carried out.
- Q 9. Write short notes on the following :
 (a) Superscalar machines
 (b) 8255 chip.

LORDS MODEL TEST PAPER – 2**Instruction to Candidates :**

1. Section – A is Compulsory consisting of TEN questions carrying TWO marks each.
2. Section – B contains FIVE questions carrying FIVE marks each and students have to attempt any FOUR questions.
3. Section – C contains THREE questions carrying TEN marks each and students have to attempt any TWO questions.

SECTION – A

- Q 1. (a) What is control memory and Cache memory?
(b) Explain about I/O processor.
(c) Role of microprogrammed control over hardwired control.
(d) What is pipelining?
(e) Give the layered view of a computer system.
(f) What is register transfer language?
(g) What is the difference between physical address and logical address ?
(h) What do you mean by programmed I/O Concept ?
(i) Briefly explain an instruction format.
(j) Write any six characteristics of RISC.

SECTION – B

- Q 2. (a) Explain the concept of register transfer?
(b) Explain the significance of a layered architecture.
- Q 3. What is meant by superscalar processor ? Explain the concept of Pipeline in superscalar processor ?
- Q 4. What do you mean by instruction cycle, Fetch cycle, machine cycle and interrupt acknowledgement cycle?
- Q 5. What is memory organisation ? Explain various memories.
- Q 6. What do you mean by software and hardware interrupts? How these are used in a microprocessor system?

SECTION – C

- Q 7. Explain the difference between Hardwired control and micro programmed control. Is it possible to have hardwired control associated with a control memory ?
- Q 8. (a) What is the concept of layers in architectural design?
(b) Give the features of a ROM cell.
- Q 9. Write short notes on the following :
(a) DMA
(b) Distributed Computing.
(c) Serial and Parallel interface.

