BBDNIIT

2024-25(Odd Semester)

Branch : Computer Science & Engineering / Information Technology

Year : Second Year – CS-22, 24 / IT 2

Computer Organization & Architecture (BCS-302)

Model Paper with Solution

Name of Faculty: Dr. V. K. Singh

BABU BANARASI DAS

NORTHERN INDIA INSTITUTE OF TECHNOLOGY LUCKNOW

BABU BANARASI DAS
NORTHERN INDIA INSTITUTE OF TECHNOLOGY, LUCKNOW
B. Tech. Computer Science & Engineering / Information Technology
Academic Session 2024-25 (Odd Semester)

COMPUTER ORGANIZATION & ARCHITECURE (BCS 302)- MODEL PAPER SOLUTION

Time: 3 hrs **Max Marks: 70**

NOTE: Attempt all sections

SECTION A

Q01. Attempt ALL parts in brief: **[7 x 2 =14]**

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
1A	Write all functions & draw structure of Computer System with suitable example.	CO1	K1
1B	Draw flow chart of Unsigned Number Division Algorithm	CO2	K2
1C	"Hardwired control unit is faster than micro programmed control unit." Justify this statement.	CO3	K5
1D	A Microprocessor processor employs **RAM chips of 128 x 8** and **ROM chips of 256 x 8**. Then system needs **2K bytes of RAM**, **4K bytes of ROM**. Find the total number of RAM and ROM chip required for memory organization?	CO4	K1
1E	Explain about DMA and Interrupt Breakpoints?	CO5	K2
1F	Show the hardware implementation of following statement: $xy'T_0 + T_1 + x'yT_2 : AR \leftarrow AR + BR$	CO2	K3
1G	Draw the Chip diagram of RAM (256X8) and ROM(1024X8).	CO4	K2

SECTION B

Q 02 Attempt any THREE Questions from this section **[03 x 7 =21]**

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
2A	Describe the principle of designing Instruction Set of a program. Explain important field of instruction format. Write a program to evaluate following arithmetic statement, assuming that at memory locations A, B, C, D, E, F, G, H, K and X data are stored, by **X = A-B + C*(D*E – F) / G + H*K** **1.** Using a general register computer with **three address** instruction **2.** Using a general register computer with **two address** instruction **3.** Using a general register computer with **one address** instruction **4.** Using a general register computer with **zero address** instruction	CO1	K2
2B	Draw Flowchart for Signed Number Division Algorithm & implement on the following **(+7) ÷ (+3)**	CO2	K3
2C	Explain an Instruction cycles of a program execution with diagram & also write the micro operation for sub cycles involve in the program execution.	CO3	K2
2D	A Computer employs RAM organization of **512 bytes** and ROM organization of **512 bytes** with the help of **128 X 8bit RAM chip** and **128 X 8bit ROM chip**. Processor has 16 address lines. Find the following: **(1)** How many RAM and ROM chip required **(2)** Draw the circuit diagram for interfacing of RAM chips , ROM chips and processor &	CO4	K3

	Draw the memory address map.		
2E	What are the techniques used for I/O operation. With the help of block diagram of DMA & interfacing diagram of DMA-PROCESSOR-MEMORY & I/O, discuss the working of direct memory access (DMA).	CO5	K2

SECTION C

Q 03 Attempt any ONE Part of the Following **[7 x 1 = 7]**

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
3A	What do you understand by Address Bus, Data Bus and Control Bus? Explain with the help of Bus structure diagram. What do you understand by BUS ARBITRATION and what are the techniques used for it? Explain in detail.	CO1	K2
3B	Explain Register Organization in detail inside the PROCESSOR with the help of block diagram also write the importance of Memory Address Register (MAR) inside the processor.	CO1	K2

Q 04 Attempt any ONE Part of the Following **[7 x 1 = 7]**

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
4A	Explain the **BOOTH Multiplication Algorithm** for Signed Number Multiplication with the help of circuit diagram and flow chart & implement for : **(+20) X (-20)**	CO2	K3
4B	What is an ALU? Draw logic diagram of ALU that performs AND, OR logic operation and ADD, SUB arithmetic operations. Derive the combinational circuit that select and generates any of the following 8 logic operation	CO2	K3

Q 05 Attempt any ONE Part of the Following **[7 x 1 = 7]**

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
5A	What is the role of CONTROL UNIT in a computer system? Draw the architecture of MICROPROGRAMMED CONTROL UNIT ARCHITECTURE and explain the following: a. Sequencing operation for an instruction execution. b. Control Memory Organization **c.** Comparison of Hardwired and Micro programmed CU.	CO3	K2
5B	Briefly define the following terms: **1.** Micro operation **2.** Micro instruction **3.** Micro Program **4.** Micro Code A digital computer system has the following configurations: **1.** Memory: 4096 x 24 **2.** Can perform 2048 operations only **3.**Indirect Addressing dependent on single *I – bit* **4.** Control Unit implementation dependent on micro programmed control unit architecture. Control unit contains a control memory capable for **1K x 32** words. Find the following: **1.** Total number of bits required for OPCODE and OPERAND. **2.** Length of Program Counter (PC), Address Register (AR), Data Register (DR). **3.** Total number of micro – instruction can be stored in control memory. Length of micro – instruction, Control Address Register (CAR), Subroutine Buffer Register (SBR).	CO3	K3

Q 06 Attempt any ONE Part of the Following **[7 x 1 = 7]**

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
6A	Explain the memory hierarchy of memory organization & 2D & 2½D Memory Organization. Explain the following: **1.** Hit Ratio of memory organization **2.** Miss Ratio of memory organization **3.** Access Frequency **4.** Effective Access Time **5.** Cost of the memory	CO4	K2

	Consider a Cache (M_1), Main Memory (M_2) and Disk Storage (M_3) with following characteristic **M_1 : 16K word, 50ns access time, c_1 = \$ 1.25 M_2 : 1M word , 400 ns access time, c_2 = \$ 0.2** **M_3 : 256M word, 4ms access time, c_3 =?** The total cost of memory organization is upper bounded by **S 15000**. Find cost per word for memory level M_3 and Effective Access Time. Cache hit is given as 0.98; Main hit is given as 0.9.		
6B	What is the mapping for cache organization? Explain the following with example: **a. Associative Mapping b. Direct Mapping c. Set Associative Mapping** Consider a cache consisting of **256 blocks of 16 words each**, for a total of **4096 (4K) words**; and assume that the main memory is addressable by a **16-bit address** and it consists of **4K blocks**. How many bits are there in each of **TAG, BLOCK / SET, WORD** and word fields for all mapping technique.	CO4	K3

Q 07 Attempt any ONE Part of the Following [7 x 1 = 7]

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
7A	What are the techniques used for I/O operation. With the help of block diagram, discuss the working of Direct Memory Access (DMA). Also explain the main features of an IOP (Input / Output processor). Give a brief comparison of DMA and IOP.	CO5	K4
7B	Discuss the I/O processor organization with the help of block diagram and write algorithmic steps for CPU – IOP interaction.	CO5	K2

BABU BANARASI DAS
NORTHERN INDIA INSTITUTE OF TECHNOLOGY, LUCKNOW
B. Tech. Computer Science & Engineering / Information Technology
Academic Session 2024-25 (Odd Semester)

COMPUTER ORGANIZATION & ARCHITECURE (BCS 302)- MODEL PAPER SOLUTION

Time: 3 hrs **Max Marks: 70**

NOTE: Attempt all sections

SECTION A

Q01. Attempt ALL parts in brief: **[7 x 2 =14]**

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
1A	Write all functions & draw structure of Computer System with suitable example.	CO1	K1

2. STRUCTURE & FUNCTIONS OF COMPUTER (Ref William Stalling)

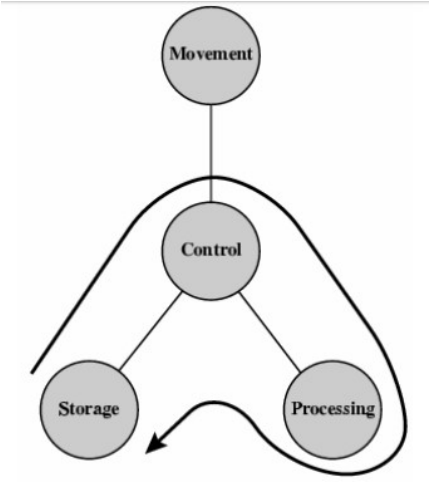
FUNCTIONS OF COMPUTER

Understanding of both structure and functioning of a computer are, important before start of complete functionality of computer at software and hardware level:

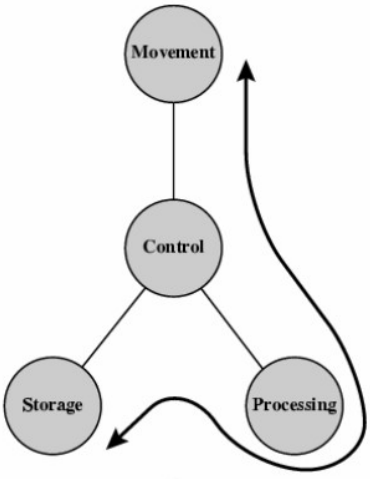
In general terms, there are only four basic functions that a computer can perform:

1. **Data Processing**
2. **Data Storage**
3. **Data Movement**
4. **Control**

1. DATA PROCESSING:



Processing from / to storage

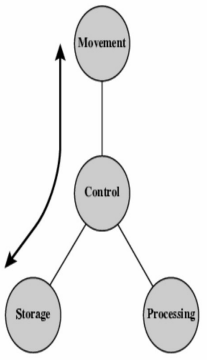


Processing from Storage to I/O

Data may take a wide variety of forms, and the range of processing requirements is

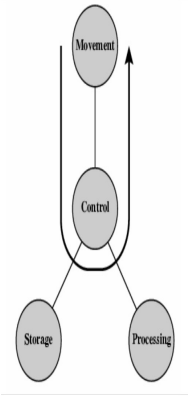
broad. However, we shall see that there are only a few fundamental methods or types of data processing.

2. DATA STORAGE:



Even if the computer is processing data on the fly (i.e., data come in and get processed, and the results go out immediately), the computer must temporarily store at least those pieces of data that are being worked on at any given moment. Thus, there is at least a short- term data storage function. Equally important, the computer performs a long-term data storage function. Files of data are stored on the computer for subsequent retrieval and update.

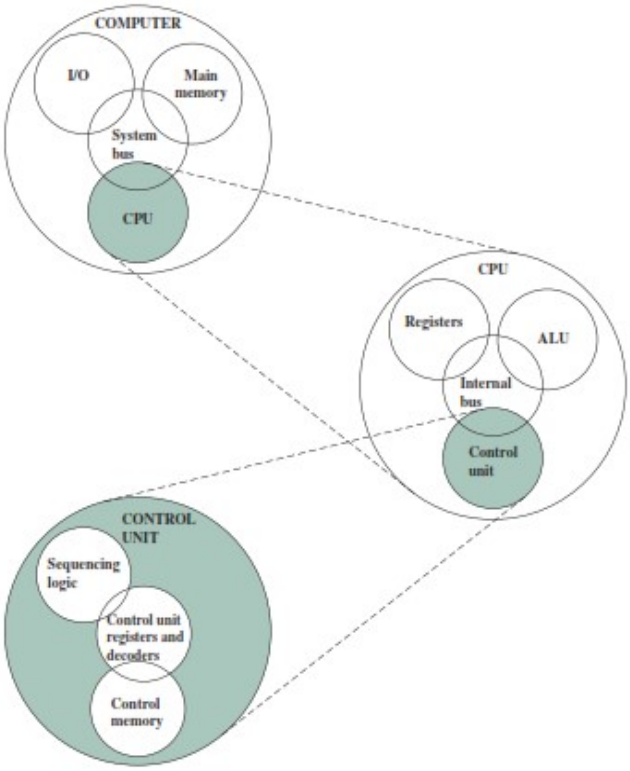
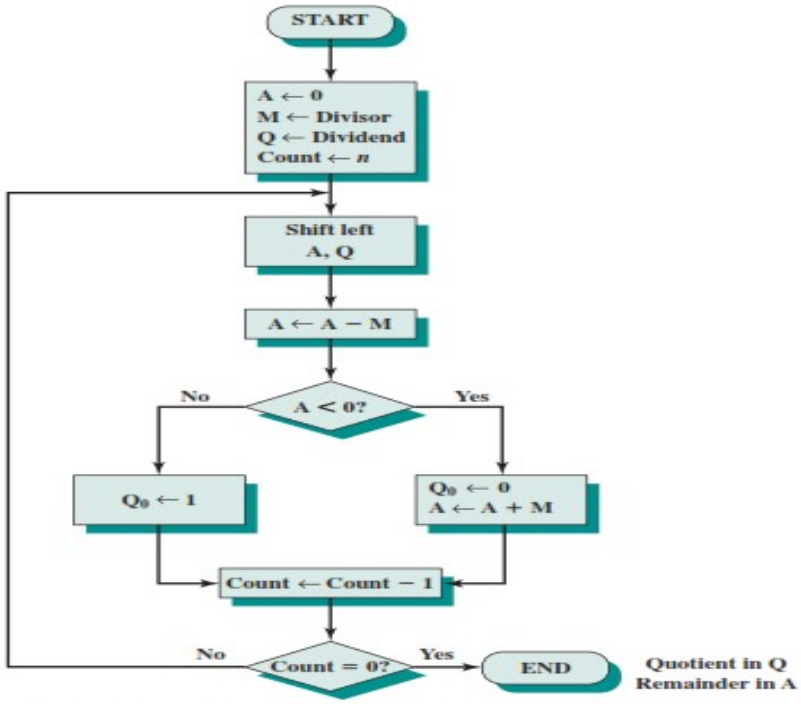
3. DATA MOVEMENT:



The computer's operating environment consists of devices that serve as either sources or destinations of data. When data are received from or delivered to a device that is directly connected to the computer, the process is known as input– output (I/O), and the device is referred to as a peripheral. When data are moved over longer distances, to or from a remote device, the process is known as data communications.

4. CONTROL:

Within the computer, a control unit manages the computer's resources and orchestrates the performance of its functional parts in response to instructions.

	STRUCTURE OF COMPUTER  Fig 2 : The Computer: Top- Level Structure Single-processor computer Figure 2 provides a hierarchical view of the internal structure of a traditional single-processor computer.		
1B	**Draw flow chart of Unsigned Number Division Algorithm**  Figure 10.16 Flowchart for Unsigned Binary Division	CO2	K2

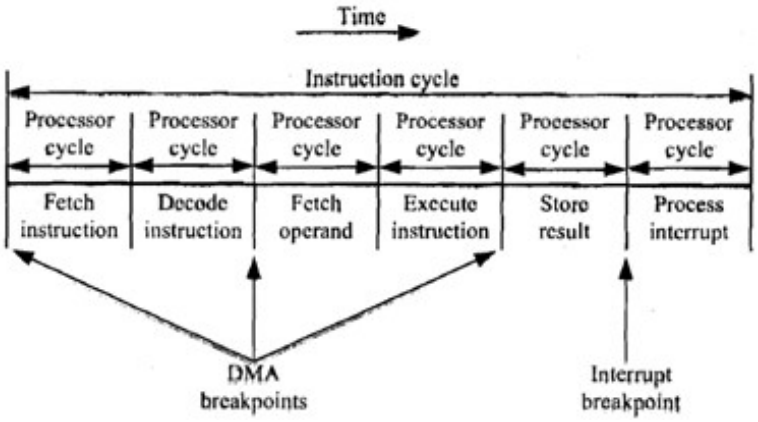
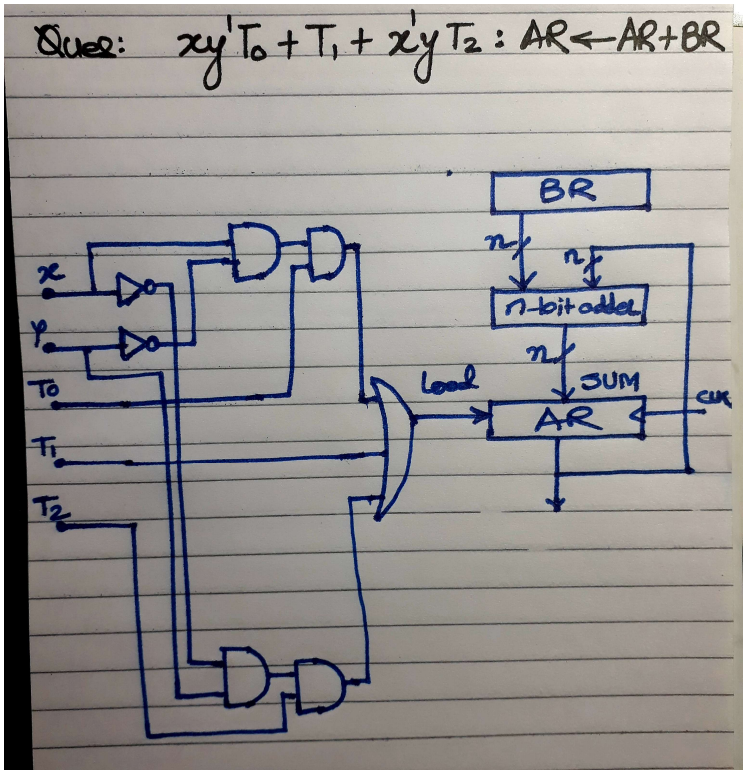
1C	**"Hardwired control unit is faster than micro programmed control unit." Justify this statement.**	CO3	K5

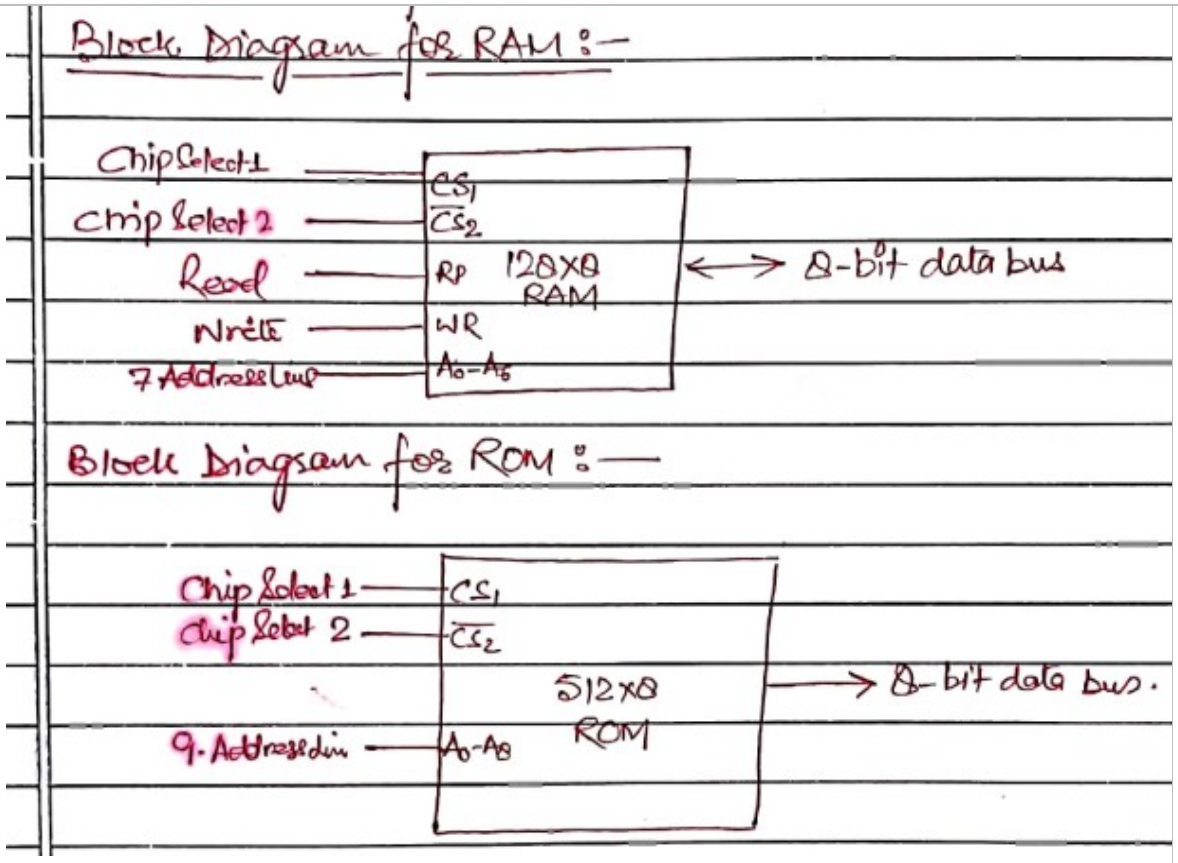
COMPARISON OF HARDWIRED CONTROL UNIT & MICRO-PROGRAMMED CONTROL UNIT:

1. To execute an instruction, there are two types of control units Hardwired Control unit and Micro-programmed control unit.
2. Hardwired control units are generally faster than micro-programmed designs. In hardwired control, we saw how all the control signals required inside the CPU can be generated using a state counter and a PLA circuit.
3. A micro-programmed control unit is a relatively simple logic circuit that is capable of
   1. Sequencing through microinstructions
   2. Generating control signals to execute each microinstruction.

HARDWIRED CONTROL UNIT	MICROPROGRAMMED CONTROL UNIT
Hardwired control unit generates the control signals needed for the processor using logic circuits	Micro-programmed control unit generates the control signals with the help of micro instructions stored in control memory
Hardwired control unit is faster when compared to micro-programmed control unit as the required control signals are generated with the help of hardware's	This is slower than the other as micro instructions are used for generating signals here
Difficult to modify as the control signals that need to be generated are hard wired	Easy to modify as the modification need to be done only at the instruction level
More costlier as everything has to be realized in terms of logic gates	Less costlier than hardwired control as only micro instructions are used for generating control signals

	It cannot handle complex instructions as the circuit design for it becomes complex	It can handle complex instructions		
	Only limited number of instructions are used due to the hardware implementation	Control signals for many instructions can be generated		
	Used in computer that makes use of Reduced Instruction Set Computers(RISC)	Used in computer that makes use of Complex Instruction Set Computers(CISC)		
1 D	A Microprocessor processor employs **RAM chips of 128 x 8** and **ROM chips of 256 x 8**. Then system needs **2K bytes of RAM**, **4K bytes of ROM**. Find the total number of RAM and ROM chip required for memory organization? **Number of RAM Chips required:** Total Number of RAM = Total RAM Capacity in Memory Organization / Capicity of Single RAM Chip Total Number of RAM = 2K Bytes / 128 x 8 = 2 x 1024 x 8 / 128 x 8 = 16 **Number of ROM Chips required:** Total Number of ROM = Total ROM Capacity in Memory Organization / Capicity of Single ROM Chip Total Number of ROM = 4K Bytes / 256 x 8 = 4 x 1024 x 8 / 256 x 8 = 16		CO4	K1
1E	**Explain about DMA and Interrupt Breakpoints?** DMA interface transfers complete block of data one word at a time directly to or from memory without going through processor. When transfer is complete, DMA interface transmits an interrupt signal to processor. So in DMA processor involvement can be restricted at beginning and end of transfer that can be displayed as in figure above. However question is when should DMA take control of bus? For this we would recall phenomenon of execution of an instruction by processor. Figure below displays five cycles for an instruction execution. Figure also displays five points where a DMA request can be responded to and a point where interrupt request can be responded to. Please note an interrupt request is acknowledged only at one point of an instruction cycle and that is at interrupt cycle.		CO5	K2

			
1F	Show the hardware implementation of following statement: **$xy'T_0 + T_1 + x'yT_2 : AR \leftarrow AR + BR$** 	CO2	K3
1 G	Draw the Chip diagram of RAM (256X8) and ROM(1024X8).	CO4	K2



For RAM 256 x 8 : Address Lines : 8lines (A0-A7) & 8 data lines
For ROM 1024 X 8 : Address Lines: 10lines (A0-A9) & 8 data lines

SECTION B

Q 02 Attempt any THREE Questions from this section [03 x 7 =21]

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
2A	Describe the principle of designing Instruction Set of a program. Explain important field of instruction format. Write a program to evaluate following arithmetic statement, assuming that at memory locations A, B, C, D, E, F, G, H, K and X data are stored, by **X = A-B + C*(D*E – F) / G + H*K** **1.** Using a general register computer with **three address** instruction **2.** Using a general register computer with **two address** instruction **3.** Using a general register computer with **one address** instruction **4.** Using a general register computer with **zero address** instruction	CO1	K2

S.1. **Instruction formal representation :—**

To explain the influence of the number of address on computers programs, we take an arithmatic expression

$$X = (A+B) * (C+D)$$

for the processing of
1. Zero Address Instruction
2. One Address Instruction
3. Two Address Instruction
4. Three Address Instruction

we will use the symbols ADD, SUB, MUL and DIV for the arithmatic operations; MOV for the transfer-type operation; and LOAD and STORE for transfers to and from memory and AC register. We also assumed that the operands are in memory addresses A, B, C and D and result must be stored in memory at address X.

S.1.1 **THREE ADDRESS INSTRUCTION :—**

1. Computers with three address instruction formats can use each address field to specify either a processor register or a memory operand.
2. The program in assembly language that evaluates

$$X = (A+B) * (C+D)$$

is shown below

```
ADD   R1, A, B    R1 ← M[A] + M[B]
ADD   R2, C, D    R2 ← M[C] + M[D]
MUL   X, R1, R2   M[X] ← R1 * R2
```

It is assumed that the computer has two processor registers R1 and R2 and M[A] represent the operand stored in memory at location A.

3. The advantage of the three-address format is that it results in short programs when evaluating arithmatic expressions.

4. The disadvantage of this format is that the binary-coded instructions require too many bits to specify three addresses.

Eg An example of a commercial computer that uses three-address instruction is the Cyber 170.

5.1.2. TWO ADDRESS INSTRUCTIONS :—

1. Two address instructions are the most common in commercial computers.

2. In this instruction format each address field can specify either a processor register or a memory word.

3. The program evaluate

$$X = (A+B) * (C+D)$$

is as follows :

MOV	R_1, A	$R_1 \leftarrow M[A]$	The mov instruction transfers the operands to and from memory & processor register
ADD	R_1, B	$R_1 \leftarrow R_1 + M[B]$	
MOV	R_2, C	$R_2 \leftarrow M[C]$	
ADD	R_2, D	$R_2 \leftarrow R_2 + M[D]$	
MUL	R_1, R_2	$R_1 \leftarrow R_1 * R_2$	
MOV	X, R_1	$M[X] \leftarrow R_1$	

5.1.3 ONE-ADDRESS INSTRUCTION :—

1. One-address instructions use an implied accumulator (AC) register for all data manipulation. for multiplication and division there is a need for a second register. In this case we will neglect the second register and assume that the AC contains the result of all operations.

2. The program to evaluate the expression

$$X = (A+B) * (C+D)$$

```
LOAD   A    ;  AC ← M[A]
ADD    B    ;  AC ← AC + M[B]
STORE  T    ;  M[T] ← AC
LOAD   C    ;  AC ← M[C]
ADD    D    ;  AC ← AC + M[D]
MUL    T    ;  AC ← AC * M[T]
STORE  X    ;  M[X] ← AC
```

All operations are done between the AC register and a memory operand. T is the address of a temporary memory location required for storing the intermediate result.

5.1.4 ZERO-ADDRESS INSTRUCTION :—

1. A stack organized computer does not use an address field for the instructions ADD and MUL.
2. The PUSH and POP instructions needs an address field to specify the operand that communicates with the stack.
3. The following program shows the execution of expression

$$X = (A+B) * (C+D)$$

In stack organized computers are

```
PUSH A ;  TOS ← A
PUSH B ;  TOS ← B
ADD    ;  TOS ← A+B
PUSH C ;  TOS ← C
PUSH D ;  TOS ← D
ADD    ;  TOS ← C+D
MUL    ;  TOS ← (C+D) * (A+B)
POP X  ;  M[X] ← TOS
```

4. To evaluate arithmetic expression in a stack computer it is necessary to convert the expression in reverse Polish Notation.

Ques. Write a program to evaluate the arithmatic statement 1. $(A-B)/(C+D*E)$.

2. $X = \dfrac{A-B+C*(D*E-F)}{G+H*K}$

a). Using a general register computer with three address instruction.

b). Using a general register computer with two address instruction

c). Using an accumulator type computer with one address instruction

d). Using a stack organised computer with zero address operation instructions.

Ans:- 1. Three address

SUB	R_1, A, B	$R_1 \leftarrow A-B$
MUL	T, D, E	$T \leftarrow D*E$
ADD	T, T, C	$T \leftarrow T+C$
DIV	R_1, R_1, T	$R_1 \leftarrow R_1 \div T$

2. Two address

MOV	R_1, A	$R_1 \leftarrow A$
SUB	R_1, B	$R_1 \leftarrow R_1 - B$
MOV	T, D	$T \leftarrow D$
MUL	T, E	$T \leftarrow T*E$
ADD	T, C	$T \leftarrow T+C$
DIV	R_1, T	$R_1 \leftarrow R_1 \div T$
MOV	Y, R_1	

One address $Y = (A-B) + (C+D \times E)$

1.	LOAD D	$AC \leftarrow D$
2.	MUL E	$AC \leftarrow AC * E$
3.	ADD C	$AC \leftarrow AC + C$
4.	STORE T	$T \leftarrow AC$
5.	LOAD A	$AC \leftarrow A$
6.	SUB B	$AC \leftarrow AC - B$
7.	DIV T	$AC \leftarrow AC \div T$
8.	STORE Y	$Y \leftarrow AC$

Zero Address $Y = (A-B) \div (C+D*E)$

PUSH	A	$TOS \leftarrow A$
PUSH	B	$TOS \leftarrow B$
SUB		$TOS \leftarrow A-B$
PUSH	D	$TOS \leftarrow D$
PUSH	E	$TOS \leftarrow E$
MUL		$TOS \leftarrow D*E$
PUSH	C	$TOS \leftarrow C$
ADD		$TOS \leftarrow (C+(D*E))$
DIV		$TOS \leftarrow (A-B) \div (C+(D*E))$
POP	Y	$M[Y] \leftarrow TOS$

$(A-B) \div (C+D*E)$
$= (AB-) \div (C+DE*)$
$= (AB-) \div (CDE*+)$
$= AB-CDE*+/$

Q. Write the program to evaluate the expression

$$X = \frac{(A-B)+C*(D*E-F)}{G+H*K}$$

Given expression

$$X = \frac{(A-B)+C*(D*E-F)}{G+H*K}$$

$$= \frac{(AB-)+C*((DE*)-F)}{G+(HK*)}$$

$$= \frac{(AB-)+C*(DE*F-)}{GHK*+}$$

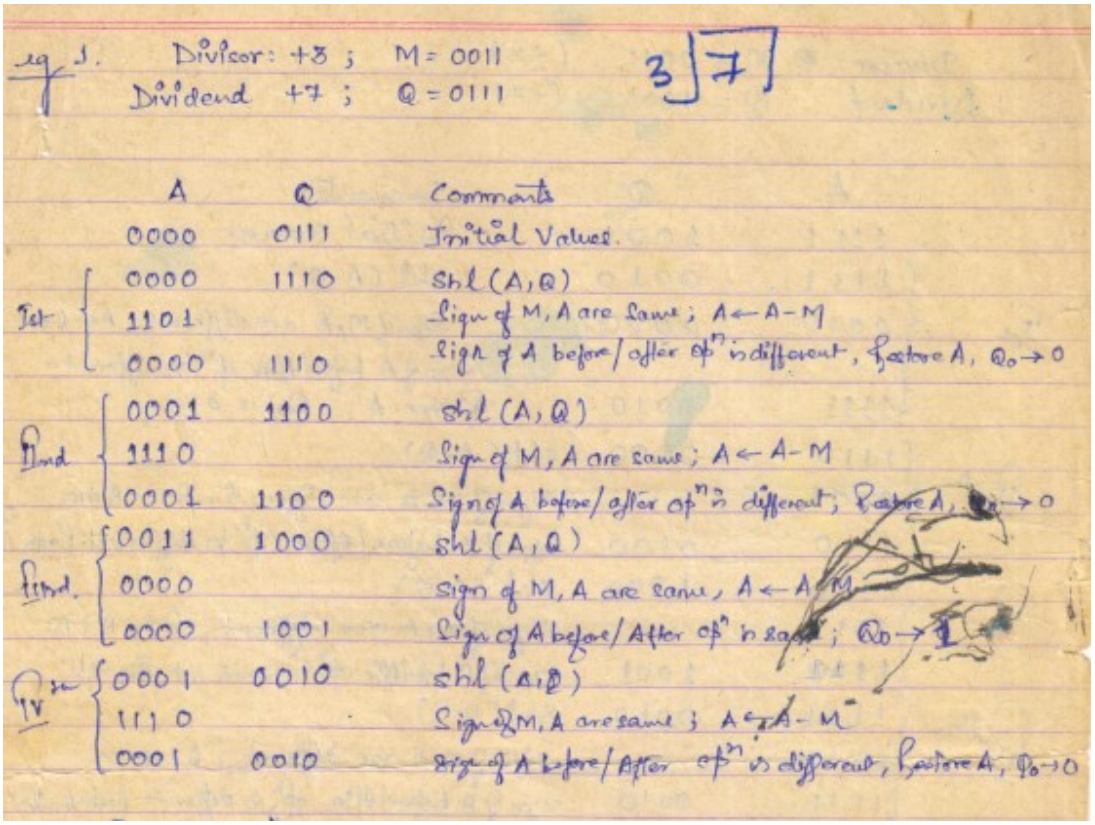
$$= \frac{(AB-)+(CDE*F-*)}{GHK*+}$$

$$= \frac{AB-CDE*F-*+}{GHK*+}$$

$$= AB-CDE*F-*+GHK*+/ \quad \text{RPN}$$

PUSH A	TOS ← A
PUSH B	TOS ← B
SUB	TOS (A-B)
PUSH C	TOS ← C
PUSH D	TOS ← D
PUSH E	TOS ← E
MUL	TOS ← D*E
PUSH F	TOS ← F
SUB	TOS ← (D*E) − F
MUL	TOS ← C * (D*E − F)
ADD	TOS ← (A-B)+C*(D*E-F)
PUSH G	TOS ← G
PUSH H	TOS ← H
PUSH K	TOS ← K
MUL	TOS ← H*K
ADD	TOS ← G+(H*K)
DIV	TOS ← (A-B)+C*(D*E-F) / G+(H*K)
POP X	M[X] ← TOS

2B	Draw Flowchart for Signed Number Division Algorithm & implement on the following **(+7) ÷ (+3)**	CO2	K3



eg 1. Divisor: +3 ; M = 0011 — 3] 7
Dividend +7 ; Q = 0111

	A	Q	Comments
	0000	0111	Initial Values.
1st	0000	1110	shl (A,Q)
	1101		Sign of M, A are same ; A ← A − M
	0000	1110	Sign of A before/after op^n is different, Restore A, $Q_0 \rightarrow 0$
2nd	0001	1100	shl (A,Q)
	1110		Sign of M, A are same ; A ← A − M
	0001	1100	Sign of A before/after op^n is different; Restore A, $Q_0 \rightarrow 0$
3rd	0011	1000	shl (A,Q)
	0000		Sign of M, A are same, A ← A − M
	0000	1001	Sign of A before/After op^n is same ; $Q_0 \rightarrow 1$
4th	0001	0010	shl (A,Q)
	1110		Sign of M, A are same ; A ← A − M
	0001	0010	Sign of A before/After op^n is different, Restore A, $Q_0 \rightarrow 0$

2C	**Explain an Instruction cycles of a program execution with diagram & also write the micro operation for sub cycles involve in the program execution.**	CO3	K2

6.0 MICRO-OPERATIONS FOR INSTRUCTION CYCLES:

1. We have seen that the operation of a computer, in executing a program, consists of a sequence of instruction cycles, with one machine instruction per cycle.
2. Instruction Cycles involves a series of steps, each of which involves the processor registers. We will refer to these steps as **micro-operations**. The prefix micro refers to the fact that each step is very simple and accomplishes very little. Micro-operations are the functional, or atomic, operations of a processor.
3. **Figure 24** depicts the relationship among the various concepts involve in program execution. To summarize, the execution of a program consists of the sequential execution of instructions. Each instruction is executed during an instruction cycle made up of shorter sub-cycles (e.g., fetch, indirect, execute, interrupt). The execution of each sub-cycle involves one or more shorter operations, that is, micro- operations.

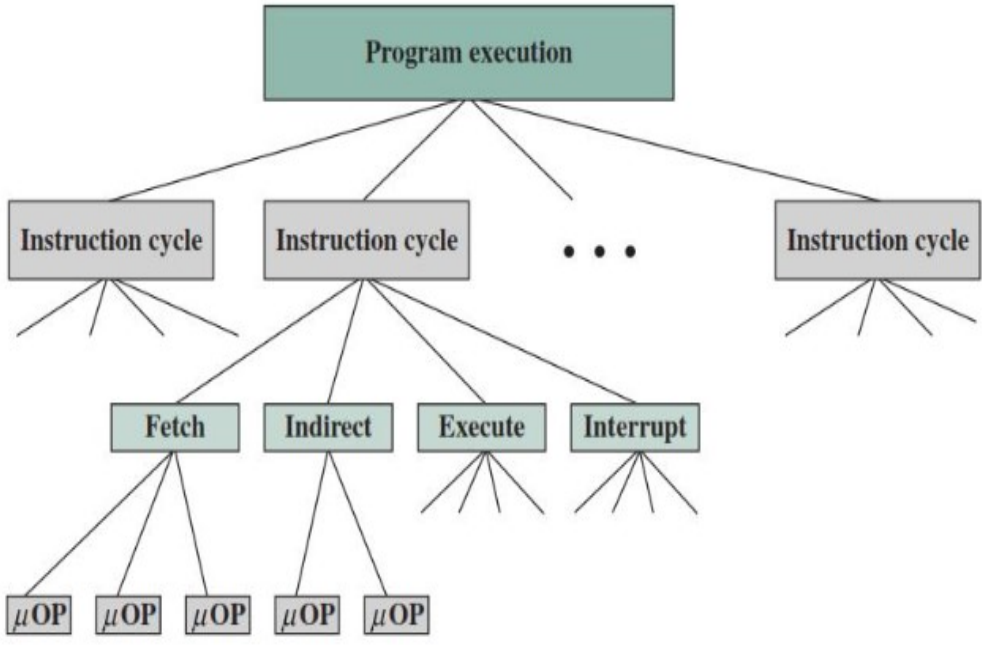
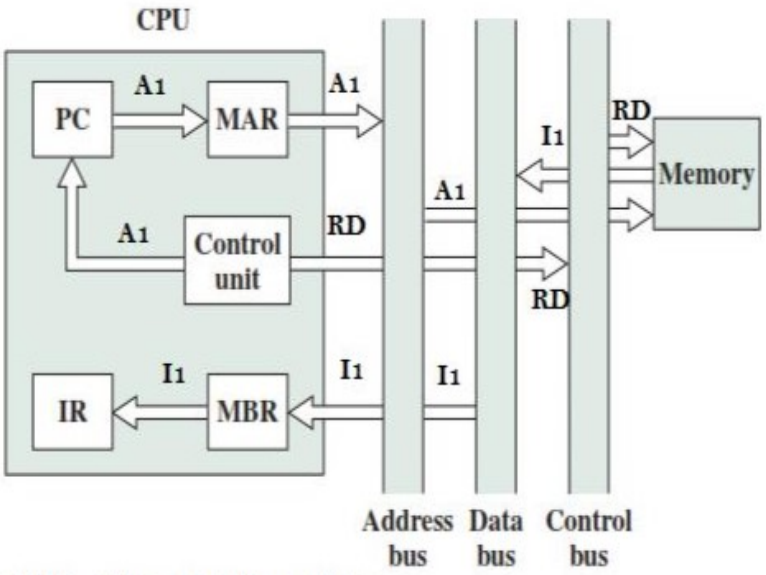


Fig 24: Constituent Elements of a Program Execution

Now in this part we will cover micro-operation associated with instruction cycles associated during the program execution ie:

1. *Micro-operations for Fetch Cycle*
2. *Micro-operations for Indirect Cycle*
3. *Micro-operations for Execute Cycle*
4. *Micro-operations for Interrupt Cycle*

6.1 MICRO-OPERATION FOR FETCH CYCLE:



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

Fetch Cycle

For the execution of **Fetch Cycle** of an instruction following micro-operations are required:

Micro-operation 1	**T1:**	**MAR ← (PC)**
Micro-operation 2	**T2:**	**MBR ← Memory**
		PC ← PC + 1
Micro-operation 3	**T3:**	**IR ← (MBR)**

Micro-operation 1 : *Transfer the address of PC to MAR. (Register Transfer)*

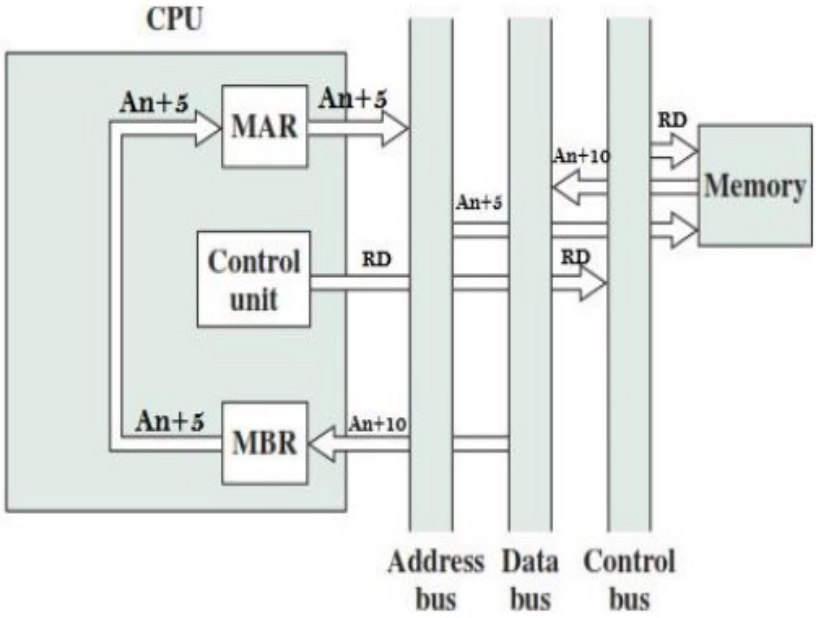
Micro-operation 2 : *MAR puts its contents on the address bus for main memory location selection, the control unit instructs the MAR to do so and also uses a memory read signal. The word so read is placed on the data bus where it is accepted by the Memory Buffer Register-MBR (Memory-read using bus. It may take more than one clock pulses depending on the* t_{cpu} *and* t_{mem}*)*

The PC is incremented by one memory word length to point to the next instruction in sequence. This micro-operation can be carried out in parallel to the micro-operation above.

Micro-operation 3 : *The instruction so obtained is transferred from Memory Buffer Register-MBR to the Instruction register for further processing. (Register Transfer)*

A clock is available for timing purposes and that it emits regularly spaced clock pulses. Each clock pulse defines a time unit. Thus, all time units are Each micro- operation can be performed within the time of a single time unit. The notation (T_1, T_2, T_3) represents successive time units.

6.2 MICRO-OPERATION FOR INDIRECT CYCLE:



Indirect Cycle

As we know Indirect Cycle will be run after the execution of Fetch Cycle if the current instruction belongs to Indirect Addressing Mode. For the execution of **Indirect Cycle** of following micro-operations are required:

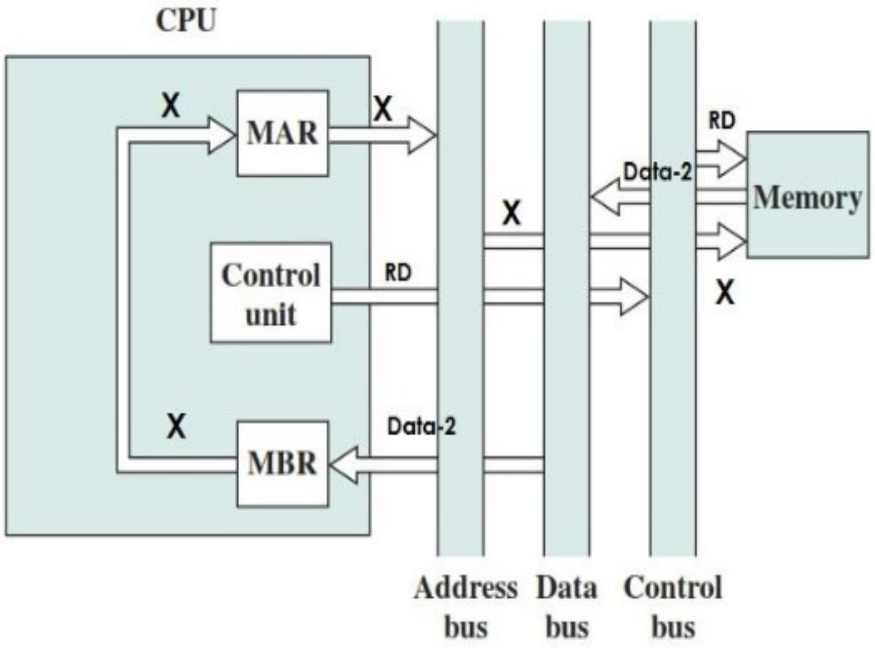
Micro-operation 1	**T1:**	**MAR ← MBR (Address)**
Micro-operation 2	**T2:**	**MBR ← Memory**
Micro-operation 3	**T3:**	**IR (Address) ← MBR (Address)**

Micro-operation 1 : *Transfer the address bits (operand address) of instruction to the MAR from MBR operand field (MBR(Address)). This transfer can be achieved using MBR, as MBR and IR at this point of time contain the same value. (Register Transfer)*

Micro-operation 2 : *Perform a memory read operation as done in fetch cycle and the desired address of the operand is obtained in the MBR(Address). (Memory Read)*

Micro-operation 3 : *Transfer the address part so obtained in MBR (MBR(Address)) as the address part of instruction register(IR(Address)). (Register Transfer)*

6.3 MICRO-OPERATION FOR EXECUTE CYCLE:



Execute Cycle

1. Now the instruction is ready for execution. A different opcode will require different sequence of steps for the execution.
2. In this case we have a simple case of addition instruction. Suppose, we have an instruction:

ADD R1, X

which adds the content of memory location X to R1 register storing the result in R1. This instruction will be executed with the help of following micro-operations:

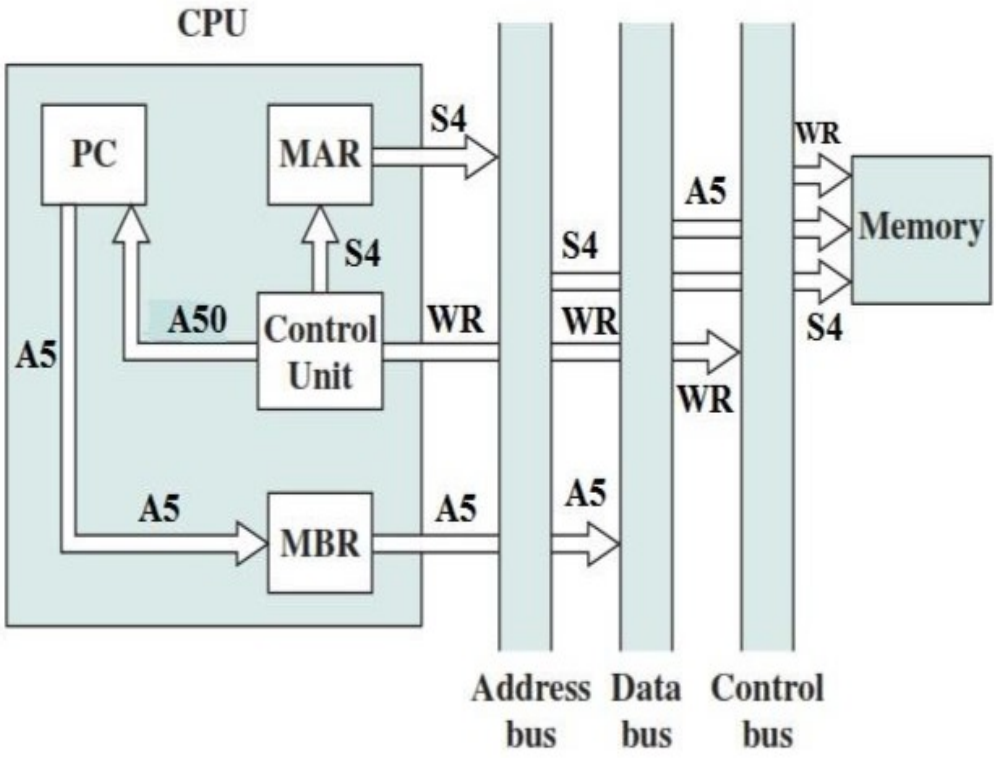
Micro-operation 1	**T1:**	**MAR ← IR (Address)**
Micro-operation 2	**T2:**	**MBR ← Memory**
Micro-operation 3	**T3:**	**R1 ← R1 + MBR**

Micro-operation 1 : *Transfer the address portion of the instruction stored in IR to the MAR. (Register transfer)*

Micro-operation 2 : *Reading operation takes place from the memory location X and bring the operand in the MBA through the data bus. (Memory read)*

Micro-operation 3 : *ADD the content of MBR with the content of register R1 using ALU and bring the results back to the register R1. (Add micro-operations)*

6.4 MICRO-OPERATION FOR INTERRUPT CYCLE:



Interrupt Cycle

1. On completion of the execution of an instruction, the machine checks whether there is any pending interrupt request for the interrupts that are enabled.
2. If an enabled interrupt has occurred then that Interrupt may be processed.
3. The nature of interrupt varies from machine to machine.
4. However, let us discuss one simple illustration of interrupt processing events. A simple sequence of steps followed in interrupt phase with the execution of following micro-operations:

Micro-operation 1	**T1:**	**MBR ← PC**
Micro-operation 2	**T2:**	**MAR ← Save Address** *(Stack Location)* **PC ← Routine Address**
Micro-operation 3	**T3:**	**MEMORY ← MBR**

Micro-operation 1 : *Transfer the current contents of PC to MBR, as this is the return address after the interrupt service program has been executed. This address must be saved.*

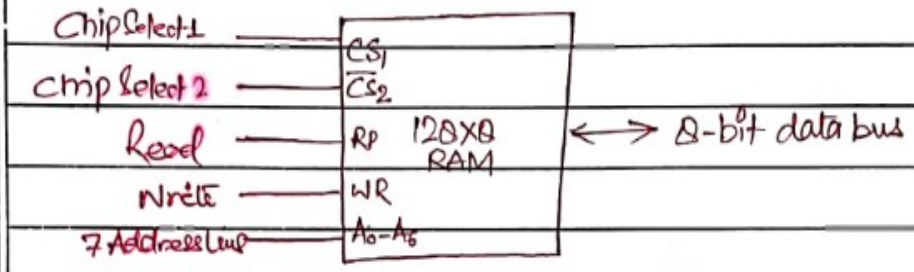
	Micro-operation 2 : *Place the address of location(Stack Location-Top of the Stack(TOS)), where the return address is to be saved, into MAR. Please note that this address is normally predetermined in computers.* *Transfer the address of the first instruction of interrupt servicing routine to the PC. This micro operation can be performed in parallel to the above micro-operation.* ***Micro-operation 3 :*** *Store the contents of PC in the memory using MBR and MAR. (Memory write)*		
2 D	A Computer employs RAM organization of **512 bytes** and ROM organization of **512 bytes** with the help of **128 X 8bit RAM chip** and **128 X 8bit ROM chip**. Processor has 16 address lines. Find the following: **(1)** How many RAM and ROM chip required **(2)** Draw the circuit diagram for interfacing of RAM chips , ROM chips and processor & Draw the memory address map.	CO4	K3

Main Memory Management :-

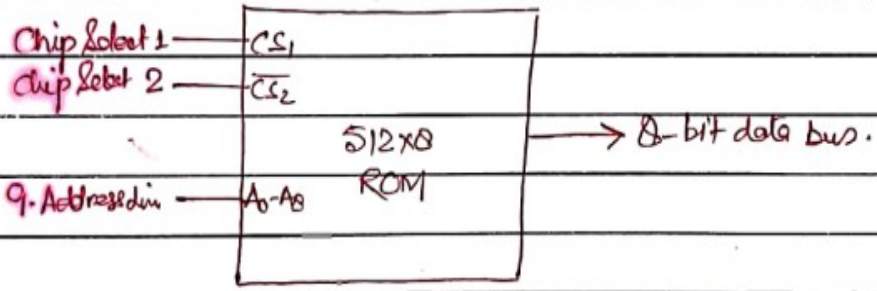
Requirement : 1024 X 8 memory with 128 X 8 RAM chips and 512 X 8 ROM chip

for 512 X 8 for RAM and
512 X 8 for ROM.

Block Diagram for RAM :-



Block Diagram for ROM :-



RAM requirement = 512 X 8

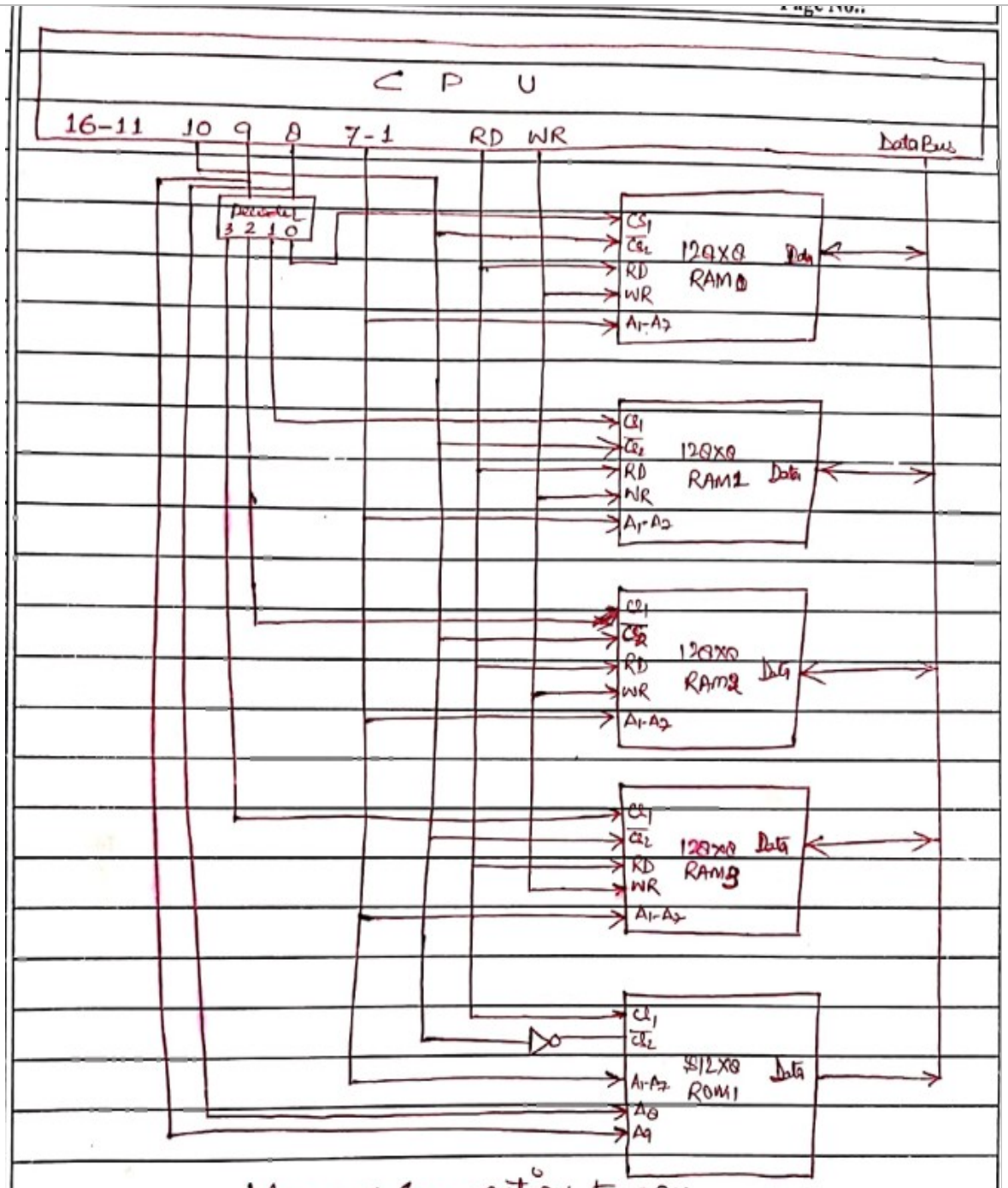
specification for single RAM chip = 128 X 8

No. of RAM chip required = $\frac{512 \times 8}{128 \times 8}$ = (4)

ROM requirement = 512 X 8

specification for single ROM chip = 512 X 8

No. of ROM chip required = $\frac{512 \times 8}{512 \times 8}$ = (1).



Memory Connection to CPU.

Memory Address Map for Microcomputer :-

Components	Hexadecimal address	Address Bus 16-11 10 9 8 7 6 5 4 3 2 1
RAM 0	0000 – 007F	0 0 0 0 0 x x x x x x x
RAM 1	0080 – 00FF	0 0 0 0 1 x x x x x x x
RAM 2	0100 – 017F	0 0 0 1 0 x x x x x x x
RAM 3	0180 – 01FF	0 0 0 1 1 x x x x x x x
ROM	0200 – 03FF	0 0 1 x x x x x x x x x

2E	**What are the techniques used for I/O operation. With the help of block diagram of DMA & interfacing diagram of DMA-PROCESSOR-MEMORY & I/O, discuss the working of direct memory access (DMA).**	CO5	K2

U-4.6 INPUT-OUTPUT TECHNIQUES

- ✓ 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.
- ✓ Data transfer between the central computer and I/O devices may be handled in a variety of modes. Three techniques are possible for I/O operation. These are:
  - Programmed input/output
  - Interrupt driven input/output
  - Direct memory access
  - I/O Processor or Channels

	Interrupt Required	I/O interface to/from memory transfer
Programmed I/O	No	Through CPU
Interrupt-driven I/O	Yes	Through CPU
DMA	Yes	Direct to Memory

Figure U5.4: Overview of Input/Output

- ➢ *In **programmed I/O**, the I/O operations are completely controlled by the processor. The processor executes a program that initiates, directs and terminate an I/O operation. It requires a little special I/O hardware, but is quite time consuming for the processor since the processor has to wait for slower I/O operations to complete.*

- ➢ *With **interrupt driven I/O**, 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 processor 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 which results in the waiting time by the processor being reduced.

- *With both programmed and interrupt-driven I/O, the processor is responsible for extracting data from the main memory for output and storing data in the main memory during input. What about having an alternative where I/O device may directly store data or retrieve data from memory? This alternative is known as* ***direct memory access (DMA).*** *In this mode, the I/O interface and main memory exchange data directly, without the involvement of processor.*

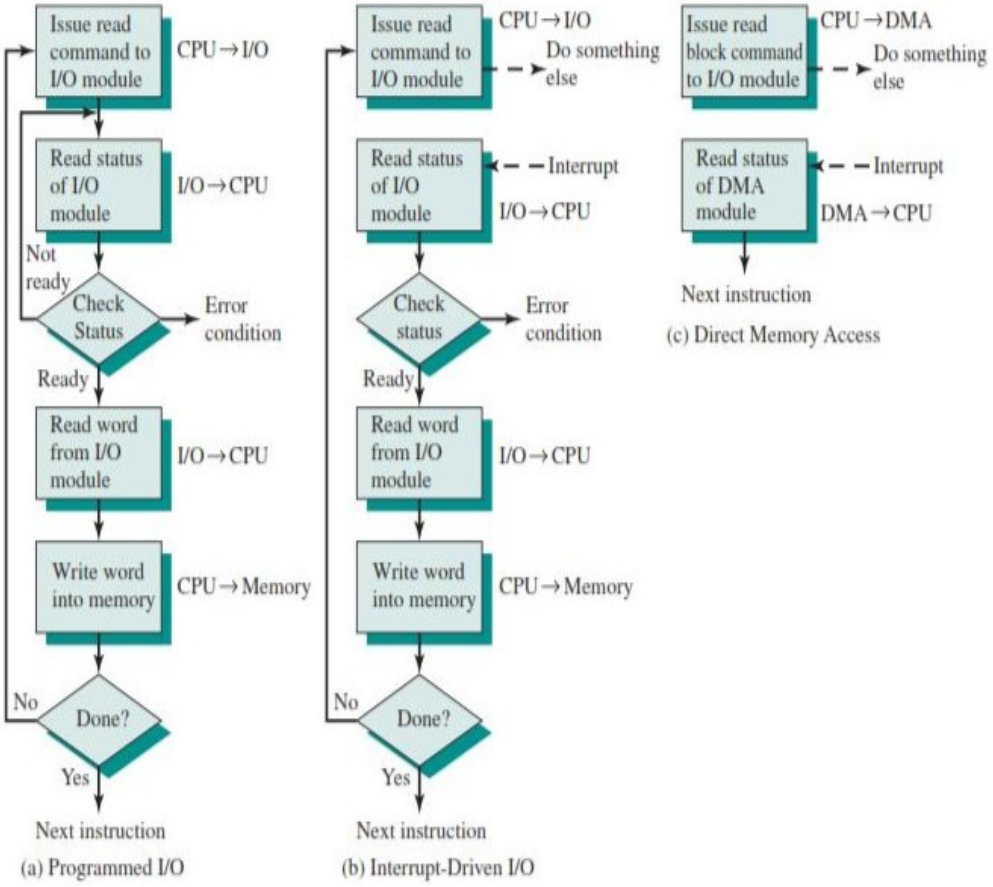


Figure U5.5 : Three Techniques for Input of a Block of Data

DMA TRANSFER:-

- ✓ The position of the DMA controller among the other components in a computer system is illustrated in Fig. U5.12

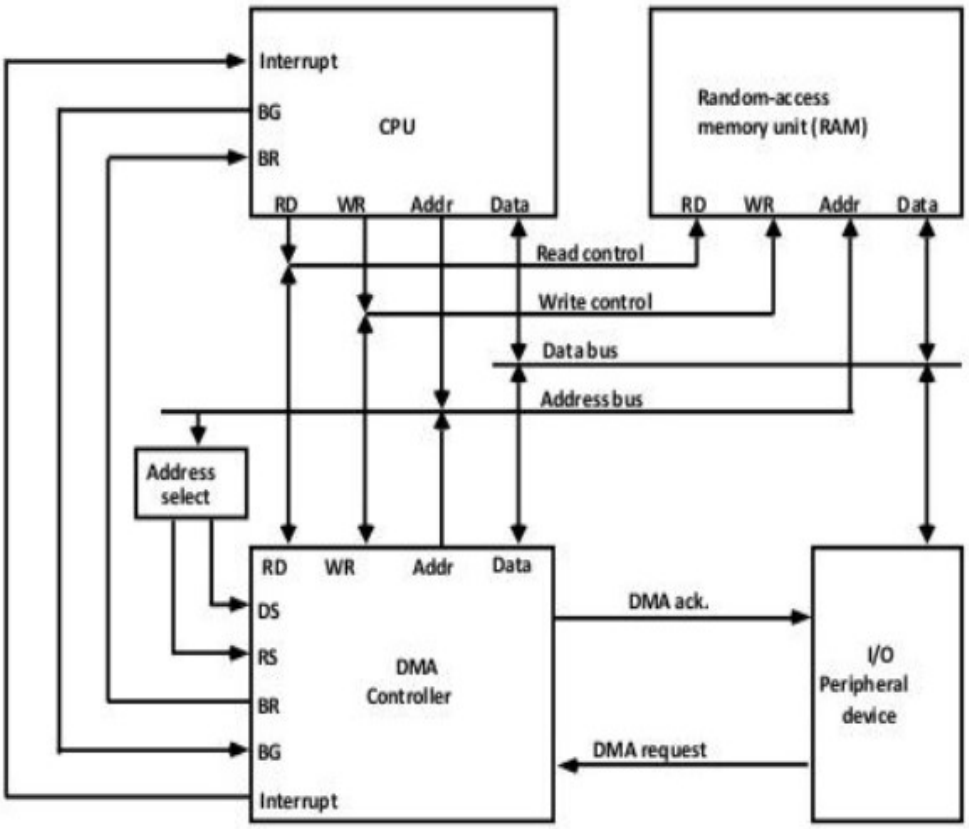


Figure U5.12: DMA transfer in a computer system

- ✓ 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 releases 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 and 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 us (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.
- ✓ 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 case the DMA disables the bus request line so that the CPU can continue to execute its program. When the peripheral requests a transfer, the DMA requests the buses again.
- ✓ It 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 on channel. In this case, each channel has a request and acknowledges 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.

SECTION C

Q 03 Attempt any ONE Part of the Following [7 x 1 = 7]

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
3A	**What do you understand by Address Bus, Data Bus and Control Bus? Explain with the help of Bus structure diagram. What do you understand by BUS ARBITRATION and what are the techniques used for it? Explain in detail.**	CO 1	K2

12.2.1 CENTRALIZED ARBITRATION

1. In the centralized arbitration scheme a hardware circuit device known as **Bus Controller** or bus arbiter processor the request to use the bus.
2. In centralized bus arbitration, a single bus arbiter performs the required arbitration. The bus arbiter may be the processor or a separate controller connected to the bus.
3. There are three different arbitration schemes that use the centralized bus arbitration approach. There schemes are:
   - **3.1** Daisy Chaining
   - **3.2** Polling Method
   - **3.3** Independent Request

3.1 DAISY CHAINING BUS ARBITRATION TECHNIQUE

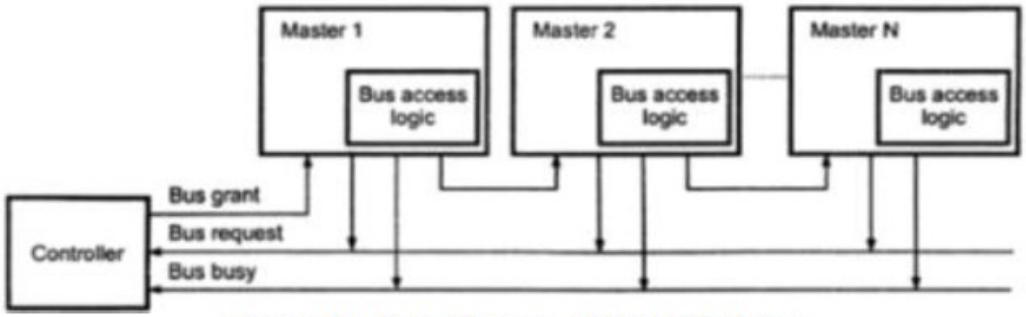


Figure :56 – Daisy Chaining Arbitration Technique

- ✓ The system connections for Daisy Chaining method are shown in Figure-57 above.
- ✓ In daisy chaining, the control of the bus is granted to any module by a "**Bus Grant (BG)**" signal.
- ✓ In this scheme, **Bus Grant** signal is distributed among the various modules and other two control signals are **Bus Request** and **Bus Busy** signals.
- ✓ The **Bus Request** line is used for grant request of system bus to various modules. The Bus Request is responded by the bus controller only if Bus Busy signal line is inactive or deactivated.
- ✓ The Bus Grant signal passed through module to module one by one.
- ✓ In this scheme, the priority of the position of the module dependent on the physical position of the module with respect to Bus Controller and cannot be changed by software commands or by programs. (Static Priority)

Module 1 Highest Priority ……*Module n* Lowest Priority

If two modules module1 and module n request the Bus at the same times, then the Bus will granted to that module which has highest priority ie to the module1.

- ✓ The basic drawback of this scheme is that if the bus request of module1 is occurring at the high rate, then the rest of the modules may not get the bus control.
- ✓ Another problem can occur when bus grant line between any two modules fails, then connection between all the modules is not possible.

3.2 POLLING ARBITRATION TECHNIQUE

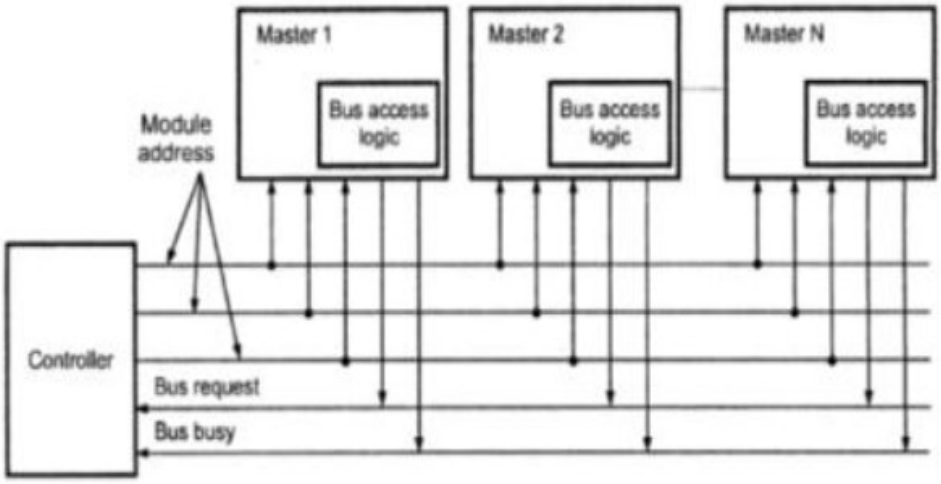


Figure :57 – Polling Arbitration Technique

- ✓ The system connections for polling method are shown in Figure-57 above.
- ✓ In this technique instead of Single BUS GRANT Line, Poll Count Lines are used. These lines are connected to all the modules connected on the BUS.
- ✓ The BUS REQUEST & BUS BUSY is the other two control lines for bus control. A request to use the BUS is made on the BUS REQUEST LINE while BUS REQUEST while not be responded to till BUS BUSY Line is active.
- ✓ The BUS CONTROLLER responds to a signal on BUS REQUEST line by generating sequences of binary address on POLL COUNT lines. These numbers are normally considered to be a unique address assigned to the connected module.
- ✓ When the Poll Count address matches the address of a particular module which is requesting for the bus, the module activates the BUS BUSY signals and start using the BUS.

- ✓ In Polling Technique, priority of Masters or Modules is Dynamic in nature and controlled by software. Priority of module can be change internally by changing the sequence of the generation of numbers on the poll counts lines.
- ✓ **Advantages :**
  - This method does not favor any particular device and processor.
  - The method is also quite simple.
  - If one device fails then entire system will not stop working.
- ✓ **Disadvantages :**
  - Adding bus masters is difficult as increases the number of address lines of the circuit. It requires more control lines which adds in cost & maximum number of modules which can share the bus in the Polling is restricted by number of poll count lines are used.

3.3 INDEPENDENT REQUEST ARBITRATION TECHNIQUE

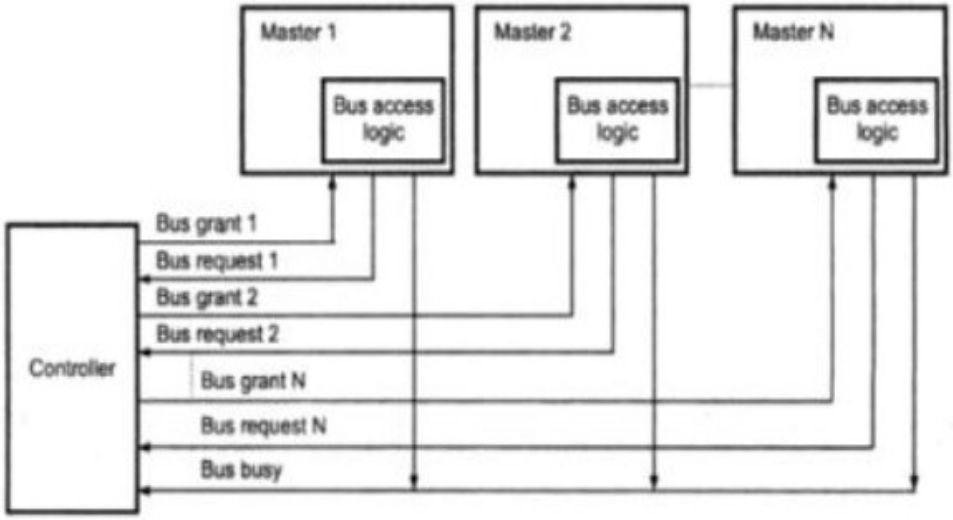
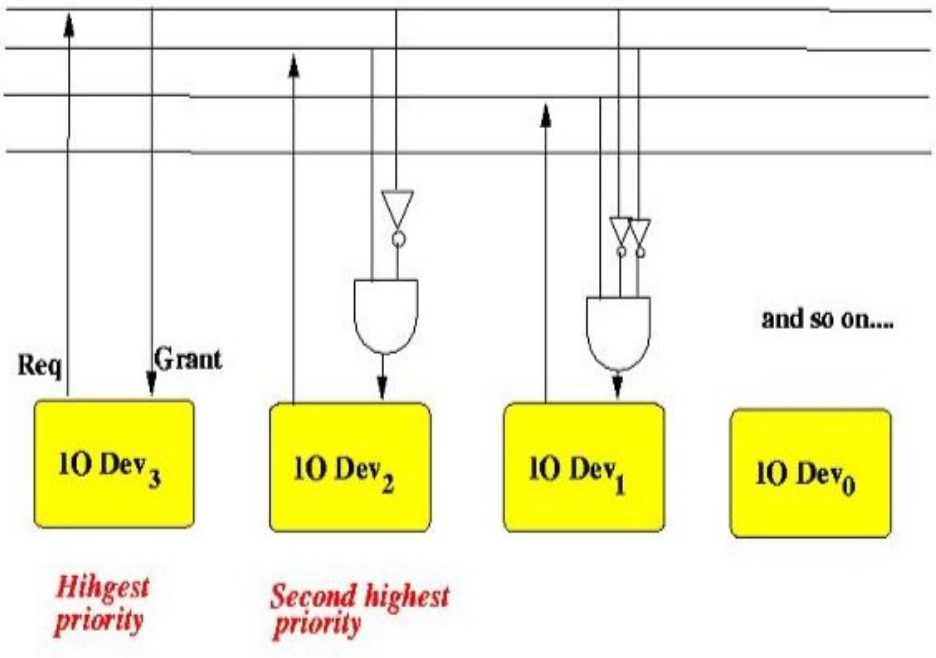


Figure :58 – Independent Request Arbitration Technique

- ✓ The figure below shows the system connections for the independent request scheme.
- ✓ In this, each master has a separate pair of bus request and bus grant lines and each pair has a priority assigned to it.
- ✓ The built-in priority decoder within the controller selects the highest priority request and asserts the corresponding bus grant signal.
- ✓ **Advantages** – This method generates fast response.
- ✓ **Disadvantages** – Hardware cost is high as large no. of control lines are required.

12.2.2 DISTRIBUTED ARBITRATION



1. Here, all the devices participate in the selection of the next bus master.
2. Each device on the bus is assigned a4 bit identification number.
3. When one or more devices request control of the bus, they assert the start arbitration signal and place their 4-bit identification numbers on arbitration lines through ARB3.
4. Each device compares the code and changes its bit position accordingly.
5. It does so by placing a 0 at the input of their drive.
6. The distributed arbitration is highly reliable because the bus operations are not dependant on devices.

3 B	**Explain Register Organization in detail inside the PROCESSOR with the help of block diagram also write the importance of Memory Address Register (MAR) inside the processor.**	CO 1	K2

3. REGISTER ORGANIZATION:

3.1 REGISTER:

A processor register is a quickly accessible location available to a computer's processors. Registers usually consist of a small amount of fast storage.

Processor registers are normally at the top of the memory hierarchy, and provide the fastest way to access data. The term normally refers only to the group of registers that are directly encoded as part of an instruction, as defined by the instruction set. Registers are normally measured by the number of bits they can hold, for example, an **"8-bit register"**, **"32-bit register"** or a **"64-bit register"** or even more.

3.2 GENERAL REGISTER ORGANIZATION FOR A PROCESSOR

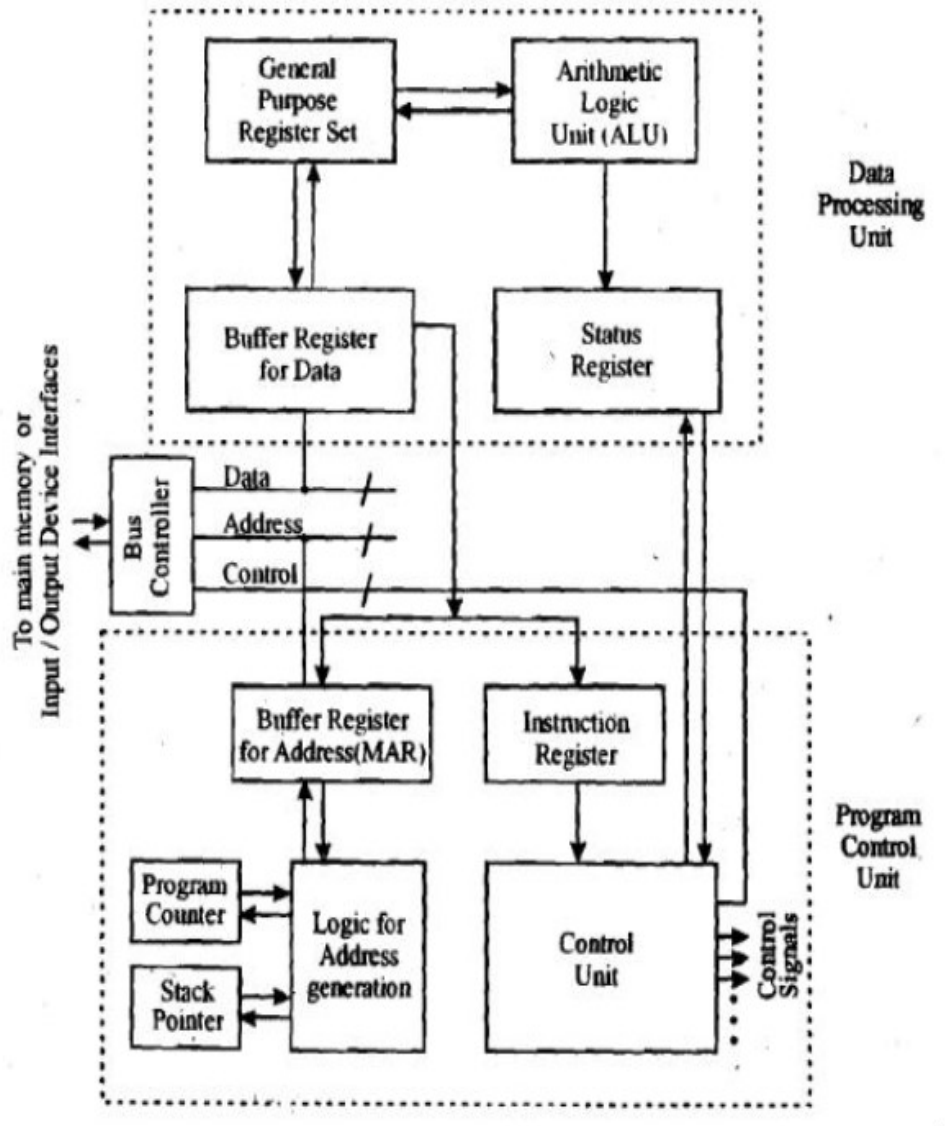
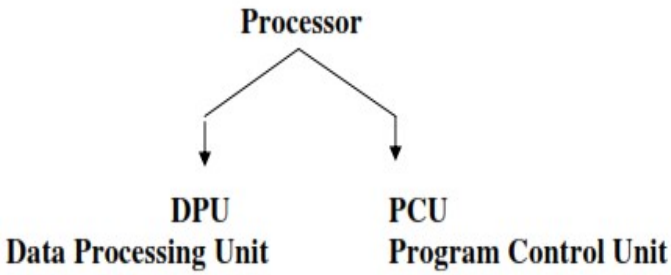


Fig 03 : CPU with general register organization



Date Processing Unit (DPU):

In a processor Data Processing Unit (DPU) is responsible for execution of program instruction with the help of available data in the registers. Data Processing Unit of processor contains:

1. Arithmetic Logic Unit
2. General Purpose Register Sets
3. Status Registers
4. Buffer for Registers

Program Control Unit (PCU) :

In a processor Program Control Unit (PCU) is responsible for to keep the track of all necessary instruction or data which are to be used for processing in the Data Processing Unit. Program Control Unit (PCU) contains:

1. Control Unit
2. Instruction Register
3. Unit of Address Generation
4. Program Counter Register
5. Stack Pointer Register
6. Buffer Register for address

Program:

A computer program is a collection of **instructions** that can be executed by a computer to perform a specific task. A computer program is usually written by a computer programmer in a programming language.

Instruction:

Computer instructions are a set of machine language instructions that a particular processor understands and executes. A computer performs tasks on the basis of the instruction provided.

An instruction comprises of groups called fields. These fields include:

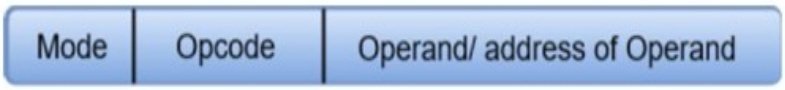


Fig 04 : Instruction Format with specific fields

1. The **Operation Code (OPCODE)** field which specifies the operation to be performed by an instruction.
2. The Address field which contains the location of the operand, i.e., register or memory location. **OPERAND** is defined as Data at which operation is to be performed after the execution of Opcode of the instruction.
3. The **Mode** field which specifies how the operand will be located. The mode of instruction will depend on the addressing mode associated with the instruction.

General Register Organization for a processor is categorized into two parts:

1. **Programmer Visible Register**
2. **Control and Status Registers**

PROGRAMMER VISIBLE REGISTER

These registers can be used by machine or assembly language programmers to minimize the memory references.

1. **General Purpose Register -** (GPR's) – General Purpose Registers are used for various functions of processor. GPR's contains operand of an instruction or can be used for calculation of address of operand for any operation code of an instruction.
2. **Accumulator -** Accumulator (AC or A) is used to temporarily store operands and results of ALU operations.
3. **Data Register -** (DR) Data registers are used as buffer storage between the main memory and processor. It also stores the operand of the instructions.
4. **Address Register (AR) -**

   Segment Pointer Register – is used to point out segment of the memory.

   Segments are specific areas defined in a program for containing data, code and stack. There are three main segments –

Code Segment (CS) − It contains all the instructions to be executed. A 16-bit Code Segment register or CS register stores the starting address of the code segment.

Data Segment (DS) − It contains data, constants and work areas. A 16-bit Data Segment register or DS register stores the starting address of the data segment.

Stack Segment (SS) − It contains data and return addresses of procedures or subroutines. It is implemented as a 'stack' data structure. The Stack Segment register or SS register stores the starting address of the stack.

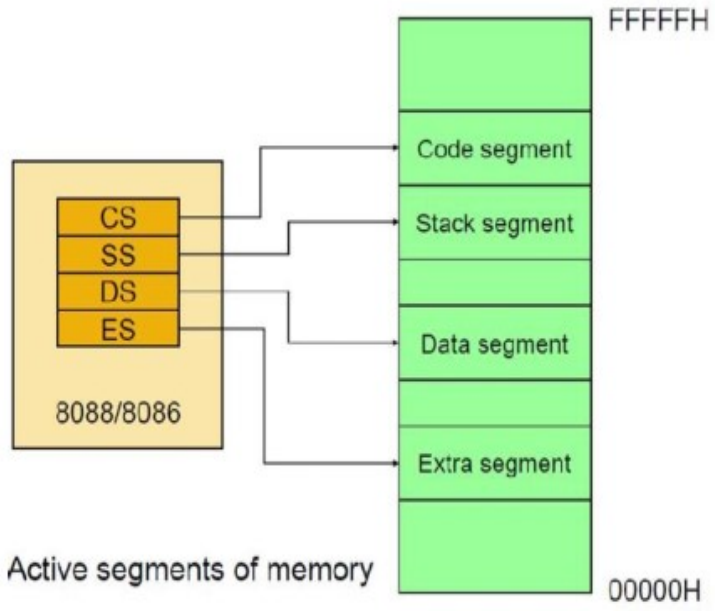


Fig 05 : Segment Registers

Stack Pointer Register – It contains the address of the stack memory.

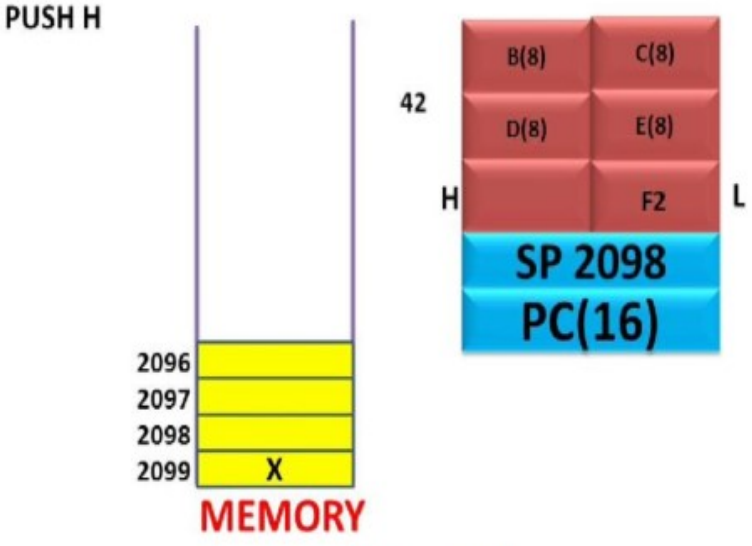


Fig 06 : Stack Pointer Register

Index Register – These are used for index addressing mode.

The 32-bit index registers, ESI and EDI, and their 16-bit rightmost portions. SI and DI, are used for indexed addressing and sometimes used in addition and subtraction. There are two sets of index pointers –

Source Index (SI) – It is used as source index for string operations.

Destination Index (DI) – It is used as destination index for string operations.

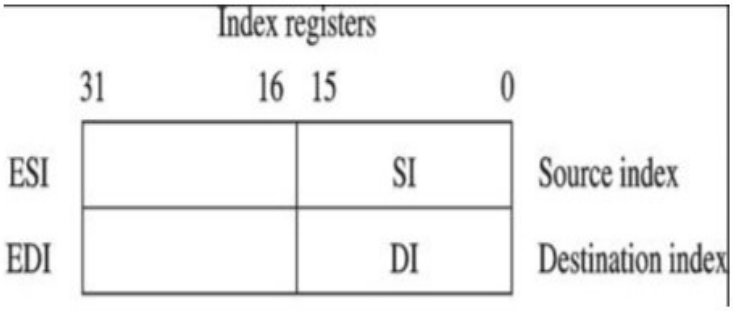


Fig 07 : Index Register

CONTROL AND STATUS REGISTER:

These registers cannot be used by programmers but are used to control the processor or execution of a program.

1. **Memory Buffer Register (MBR) –** MBR contains a word to restore in memory or is used to receive a word from memory.
2. **Memory Address Register (MAR) –** MAR represents the address in memory of the word to be written from or read into MBR.
3. **Program Counter (PC)** – PC contains the address of the next instruction to be executed.
4. **Instruction Register (IR)** – IR contains the operation code (OPCODE) being executed.
5. **Program status word (PSW)** – PSW is a set of register which holds the status information of different flags. Some of the commonly used flags or condition codes in such a register may be:

   Overflow Flag (OF) – It indicates the overflow of a high-order bit (leftmost bit) of data after a signed arithmetic operation.

   Direction Flag (DF) – It determines left or right direction for moving or comparing string data. When the DF value is 0, the string operation takes left-to-right direction and when the value is set to 1, the string operation takes right-to-left direction.

Interrupt Flag (IF) – It determines whether the external interrupts like keyboard entry, etc., are to be ignored or processed. It disables the external interrupt when the value is 0 and enables interrupts when set to 1.

Trap Flag (TF) – It allows setting the operation of the processor in single-step mode. The DEBUG program we used sets the trap flag, so we could step through the execution one instruction at a time.

Sign Flag (SF) – It shows the sign of the result of an arithmetic operation. This flag is set according to the sign of a data item following the arithmetic operation. The sign is indicated by the high-order of leftmost bit. A positive result clears the value of SF to 0 and negative result sets it to 1.

Zero Flag (ZF) – It indicates the result of an arithmetic or comparison operation. A nonzero result clears the zero flag to 0, and a zero result sets it to 1.

Auxiliary Carry Flag (AF) – It contains the carry from bit 3 to bit 4 following an arithmetic operation; used for specialized arithmetic. The AF is set when a 1-byte arithmetic operation causes a carry from bit 3 into bit 4.

Parity Flag (PF) – It indicates the total number of 1-bits in the result obtained from an arithmetic operation. An even number of 1-bits clears the parity flag to 0 and an odd number of 1-bits sets the parity flag to 1.

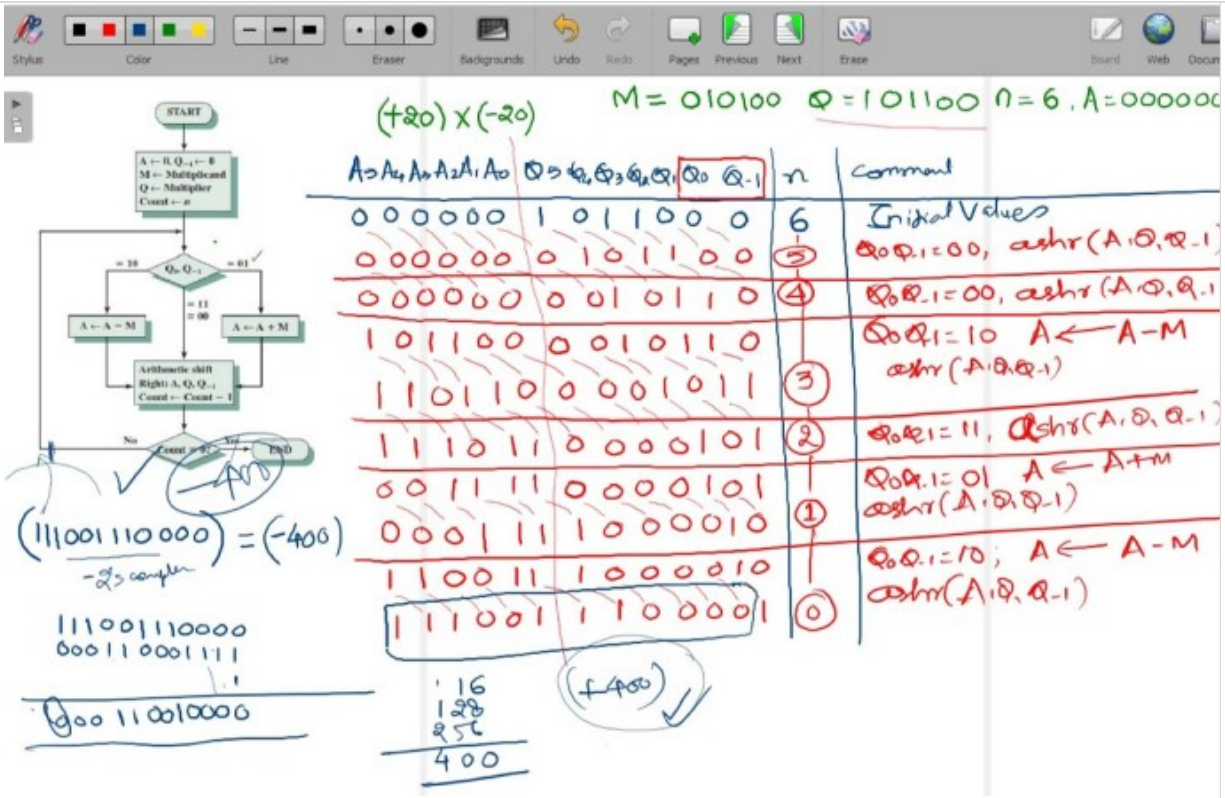
Carry Flag (CF) – It contains the carry of 0 or 1 from a high-order bit (leftmost) after an arithmetic operation. It also stores the contents of last bit of a shift or rotate operation.

The following table indicates the position of flag bits in the 16-bit Flags register:

Flag:					O	D	I	T	S	Z		A		P		C
Bit no:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Q 04 Attempt any ONE Part of the Following **[7 x 1 = 7]**

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
4 A	Explain the **BOOTH Multiplication Algorithm** for Signed Number Multiplication with the help of circuit diagram and flow chart & implement for : **(+20) X (-20)**	CO 2	K3



4 B	**What is an ALU? Draw logic diagram of ALU that performs AND, OR logic operation and ADD, SUB arithmetic operations. Derive the combinational circuit that select and generates any of the following 8 logic operation**	CO 2	K3

Operation Select			Output	Micro operation
S_1	S_2	S_3		
0	0	0	A+B	Add
0	0	1	A+B+1	Add With Carry
0	1	0	A+B	Subtract With Borrow
0	1	1	A-B	Subtract
1	0	0	B	Transfer B
1	0	1	A+1	Increment A
1	1	0	A-1	Decrement A
1	1	1	A	Transfer A

3. ARITHMETIC LOGIC UNIT (ALU) DESIGN:-

1. An ALU performs simple arithmetic-logic and shift operations.
2. The complexity of an ALU depends on the type of instruction set which has been realized for it.
3. The simple ALUs can be constructed for fixed-point numbers.
4. The floating-point arithmetic implementation requires more complex control logic and data processing capabilities, i.e., the hardware.
5. Several micro-processor families utilize only fixed-point arithmetic capabilities in the ALUs (Integer ALU).
6. For floating point arithmetic or other complex functions they may utilize an auxiliary special purpose unit. This unit is called arithmetic co-processor.

Now we discuss both the ALU's (Fixed Point ALU & Floating Point ALU) in detail

1. FIXED POINT ALU – (INTEGER ALU)

An ALU consists of circuits that perform data processing micro-operations. But how are these ALU circuits used in conjunction of other registers and control unit? The simplest organization in this respect for fixed point ALU (integer ALU) was suggested by John von Neumann in his IAS computer design

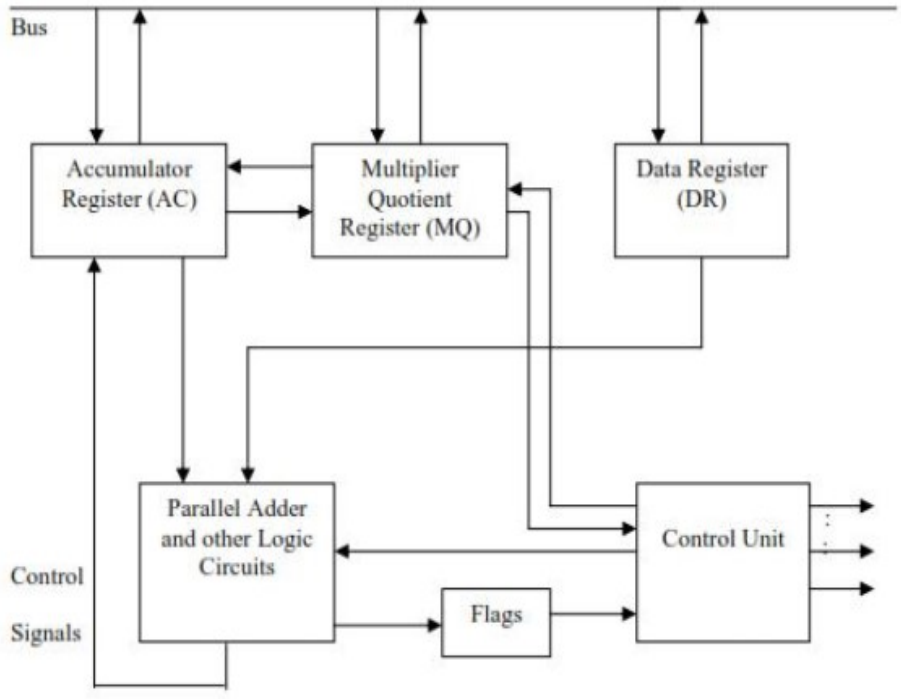


Figure 08: Structure of a Fixed point Arithmetic logic unit

1. An ALU performs the simple arithmetic, logic and shift operations.
2. The simple ALU can be constructed for fixed point number or integer number. Floating point arithmetic implementation required more complex control logic and data processing units.
3. In fixed point ALU organization, have three one-word registers **AC**(Accumulator Register), **MQ**(Multiplier & Quotient Register) and **DR**(Data Register).
4. **AC** and **MQ** are generally organized as a single **AC.MQ** register. This register is capable of left or right shift operations.
5. Some micro-operations which can be executed on this unit are:

ADDITION	**:**	**$AC \leftarrow AC + DR$**
SUBTRACTION	**:**	**$AC \leftarrow AC - DR$**
AND	**:**	**$AC \leftarrow AC \wedge DR$**
OR	**:**	**$AC \leftarrow AC \vee DR$**
EXCLUSIVE OR	**:**	**$AC \leftarrow AC \oplus DR$**
NOT	**:**	**$AC \leftarrow \overline{AC}$**

6. In this **ALU** organization, the multiplication and division are implemented using shift, addition/substation operations. The **MQ** is a special register used for implementation of multiplication and division algorithms. The **MQ** register stores the multiplier if multiplication is to be performed or the quotient value of a division operation.
7. For multiplication or division operation **DR** register stores the multiplicand or divisor value respectively.
8. The result of multiplication or division on applying certain algorithms can finally be obtained in **AC.MQ** register combination.
9. These operations are represented as

MULTIPLICATION	**:**	**$AC.MQ \leftarrow DR \times MQ$**
DIVISION	**:**	**$AC.MQ \leftarrow MQ \div DR$**

10. DR is another important register, which is used for storing second operand. In fact it acts as a buffer register, which stores the data brought from the memory for an instruction. In machines where we have general purpose registers any of the registers can be utilized as **AC, MQ** and **DR**

2. FLOATING POINT ARITHMATIC & LOGIC UNIT(FPU ALU)

A floating point ALU is implemented with the help of two fixed point arithmetic circuits known as **exponent** unit and **mantissa unit**.

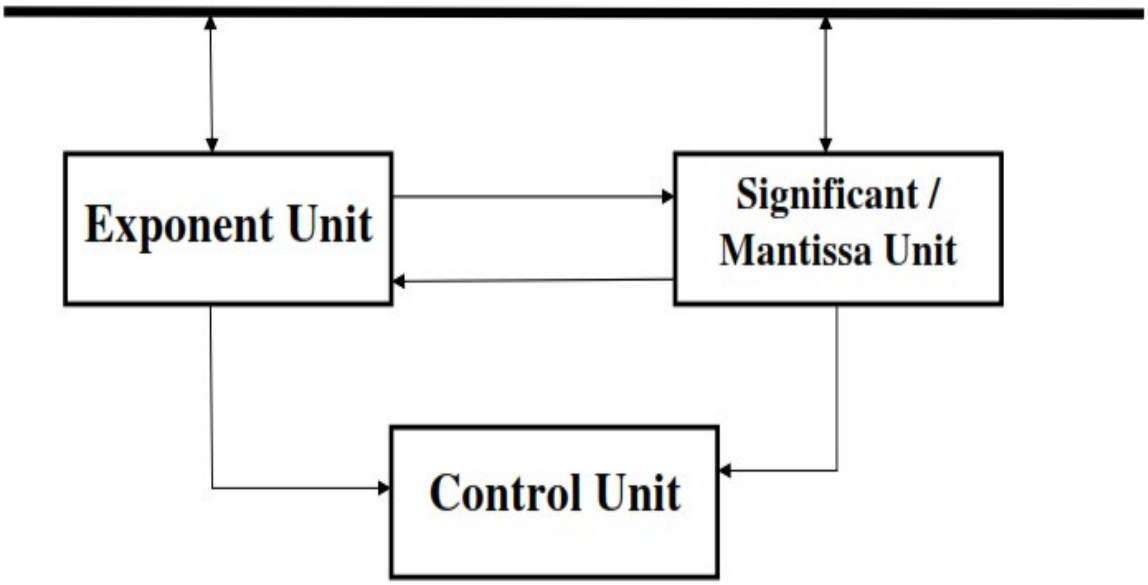
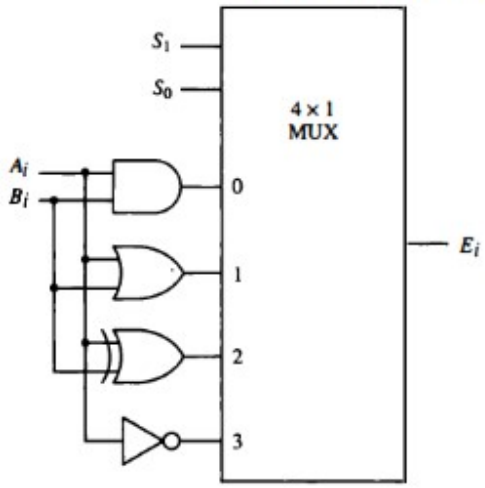


Figure 09: Structure of a Floating point ALU

1. Mantissa unit capable for performing all four arithmetic operations (addition, subtraction, multiplication & division) on the mantissa (equivalent to integer ALU design)
2. Exponent unit needs the circuit for addition and subtraction only.

Figure 4-10 One stage of logic circuit.



(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Function table

Use this concept for given question for MUX (8 X 1) with three selection lines S0, S1, S3

Q 05 Attempt any ONE Part of the Following [7 x 1 = 7]

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL
5A	**What is the role of CONTROL UNIT in a computer system? Draw the architecture of MICROPROGRAMMED CONTROL UNIT ARCHITECTURE and explain the following:** **a. Sequencing operation for an instruction execution.** **b. Control Memory Organization** **c. Comparison of Hardwired and Micro programmed CU.** The control unit is the main component of a central processing unit (CPU) in computers that can direct the operations during the execution of a program by the processor/computer. The main function of the control unit is to fetch and execute instructions from the memory of a computer. It receives the input instruction/information from the user and converts it into control signals, which are then given to the CPU for further execution. It is included as a part of Von Neumann architecture developed by John Neumann. It is responsible for providing the timing signals, and control signals and directs the execution of a program by the CPU. It is included as an internal part of the CPU in modern computers. This article describes complete information about the control unit	CO3	K2

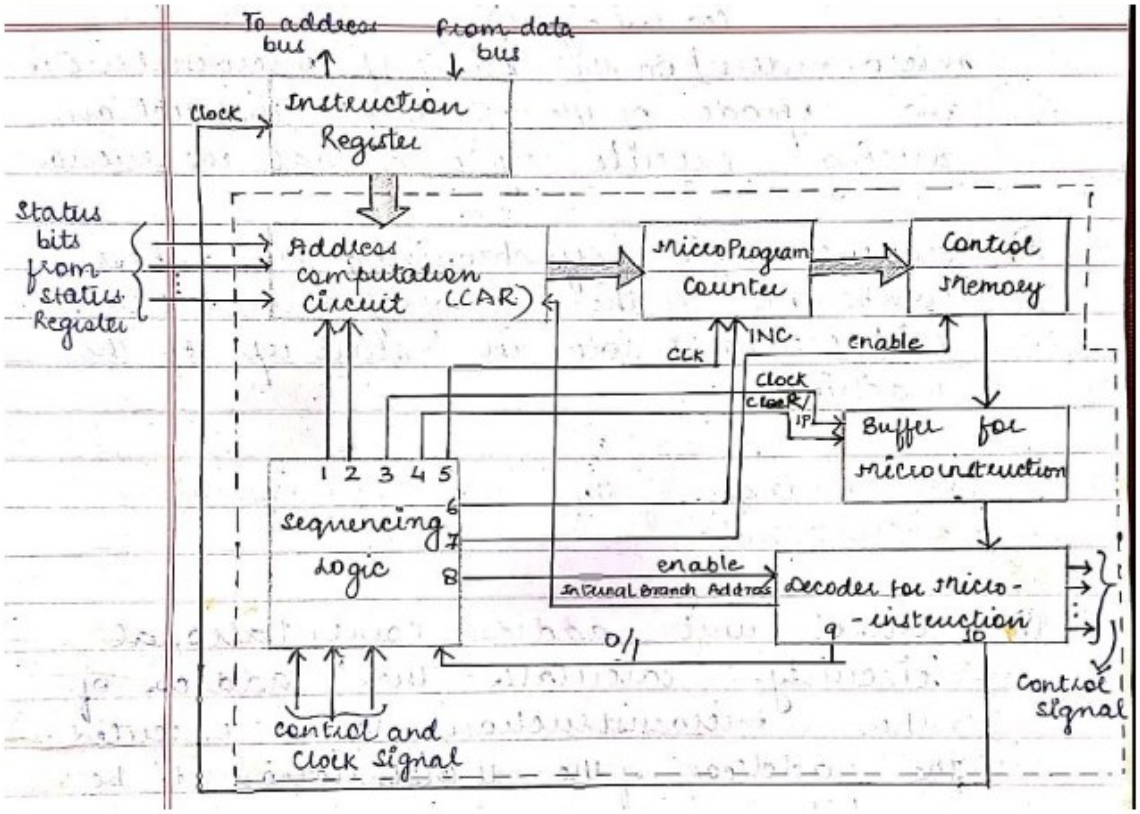


Figure 34-b : Micro-programmed Control Unit Architecture (1)

5. FUNCTIONING OF MICRO-PROGRAMMED CONTROL UNIT:

- Control unit address computational circuitry calculates the address of the micro instruction to be executed. The address of the micro-instructions to be executed is stored in the micro-program counter. Micro program counter can receive this address from

  1. Address computation circuit when clock signal no 5 is enabled.

  2. Incrementing its own content with the help of increment (INC) control signal no. 6.

- The micro instruction buffer register can be cleared during this time by enabling the signal no. 4
- Now reading process from the control memory takes place with the help of control signal no. 7.
- Signal 7 enables the control memory to transfer the micro instruction stored in the location addressed by the micro program counter into the micro instruction buffer. For this transfer operation, signal no. 3 should also be enable so that microinstruction buffer register can be loaded by micro instruction from the control memory.
- This micro instruction is then decoded and respective control signals or micro orders are generated in the micro instruction decode on receiving the control signal no. 8.

	➢ This sequence is continued till one micro program is executed. After this micro instruction decoder issues the control signal or micro order which enables the clock of the instruction register with the help of control signal no. 10. This signal causes loading of the next machine instruction to be executed into the instructions register. ➢ Control signal no. 9 is a deciding factor for branching and non-branching micro instruction. This control signal indicates to the sequence logic whether the micro instruction has been decoded is a non-branching or branching by two ways:- ✓ If value of signal no. 9 is 0, a non-branching micro instruction executed. Sequence logic-activates control signal no. 6 which increments the Micro program counter. ✓ The value of signal no. 9 is 1, a branching micro instruction executed. Decoder transfers the branch address to the address computation circuit with the help of internal Address Bus and Sequence logic activates control signal no. 5 which calls the address computation circuit to transfer the branch address to the Micro Program Counter.		
5B	Briefly define the following terms: **1.** Micro operation **2.** Micro instruction **3.** Micro Program **4.** Micro Code A digital computer system has the following configurations: **1.** Memory: 4096 x 24 **2.** Can perform 2048 operations only **3.** Indirect Addressing dependent on single *I – bit* **4.** Control Unit implementation dependent on micro programmed control unit architecture. Control unit contains a control memory capable for **1K x 32** words. Find the following: **1.** Total number of bits required for OPCODE and OPERAND. **2.** Length of Program Counter (PC), Address Register (AR), Data Register (DR). **3.** Total number of micro – instruction can be stored in control memory. Length of micro – instruction, Control Address Register (CAR), Subroutine Buffer Register (SBR). **Micro-Operation:** In computer central processing units, micro-operations (also known as micro-ops) are the functional or atomic, operations of a processor. These are low level instructions used in some designs to implement complex machine instructions. They generally perform operations on data stored in one or more registers. They transfer data between registers or between external buses of the CPU, also performs arithmetic and logical operations on registers. In executing a program, operation of a computer consists of a sequence of instruction cycles, with one machine instruction per cycle. Each instruction cycle is made up of a number of smaller units – Fetch, Indirect, Execute and Interrupt cycles. Each of these cycles involves series of steps, each of which involves the processor registers. These steps are referred as micro-operations. the prefix micro refers to the fact that each of the step is very simple and accomplishes very little. Figure below depicts the concept being discussed here.	CO3	K3

	Micro Instructions: The micro-instruction in control memory contains a set of bits to initiate micro operations in computer registers and other bits to specify the method by which the address is obtained. **Micro Program:** Microprogramming is a process of writing microcode for a microprocessor. Microcode is low-level code that defines how a microprocessor should function when it executes machine-language instructions. Typically, one machine language instruction translates into several microcode instructions, on some computers, the microcode is stored in ROM and can not be modified. **Micro Codes:** Microcode is a processor design technique that interposes a layer of computer organization between the CPU hardware and the programmer-visible instruction set architecture of the computer. As such, the microcode is a layer of hardware-level instructions that implement higher-level machine code instructions or internal state machine sequencing in many digital processing elements. Microcode is used in general-purpose central processing units.		

Q 06 Attempt any ONE Part of the Following [7 x 1 = 7]

Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BTL

6A	Explain the memory hierarchy of memory organization & 2D & 2½D Memory Organization. Explain the following: **1.** Hit Ratio of memory organization **2.** Miss Ratio of memory organization **3.** Access Frequency **4.** Effective Access Time **5.** Cost of the memory Consider a Cache (M_1), Main Memory (M_2) and Disk Storage (M_3) with following characteristic **M_1 : 16K word, 50ns access time, c_1 = $ 1.25 M_2 : 1M word , 400 ns access time, c_2 = $ 0.2** **M_3 : 256M word, 4ms access time, c_3 =?** The total cost of memory organization is upper bounded by **S 15000**. Find cost per word for memory level M_3 and Effective Access Time. Cache hit is given as 0.98; Main hit is given as 0.9.	CO4	K2

Memory Capacity Planning :-

The performance of a memory hierarchy is calculated by the effective access time T_{eff} to any level in the hierarchy. It depends on the hit ratio and access frequencies at successive levels.

Hit Ratio :-

→ Hit ratio is a concept defined for any two adjacent levels of a memory hierarchy.

→ When an information item is found in M_i, we call it hit, otherwise, a miss.

→ Consider memory levels M_i and M_{i-1} in a heirarchy, $i = 1, 2, 3, \ldots n$.

- The hit ratio h_i at M_i is the probability that an information item will be found in M_i. It is a function of characteristics of the two adjacent levels M_{i-1} and M_i. (memory capacities, management policies & programming)
- The miss ratio at M_i is $1 - h_i$.

→ The access frequency (f_i) to M_i is defined as

$$f_i = (1-h_1)(1-h_2)\cdots(1-h_{i-1})h_i$$

(The probability of successfully accessing M_i, where there are $i-1$ misses at the slower levels and a hit at M_i.

$$\sum_{i=1}^{n} f_i = 1 \text{ and } f_1 = h_1$$

→ Due to the locality property, the access frequencies decreases very rapidly from low to high levels i

$$f_1 \geq f_2 \geq f_3 \geq \cdots \geq f_n .$$

Effective Access Time :-

→ In the memory organisation maximum probability of a hit is always in lowest level say M_1. When the miss occurs in the lowest level then memory access or loading for next level is started.

→ The misses are called block misses in the cache, and page faults in the main memory. (because blocks and pages are the units of transfer between these levels)

→ The time penality for a page fault is much longer than that for a block miss. ($t_1 < t_2 < t_3$)
a cache miss is 2 to 4 times costly as a cache hit, but a page fault is 1000 to 10000 times costly as a page hit.

→ By access frequencies f_i for $i = 1, 2, 3, \ldots n$
Effective Access time for memory heirarchy

$$T_{eff} = \sum_{i=1}^{n} f_i \cdot t_i$$

$$T_{eff} = h_1 t_1 + (1-h_1) h_2 t_2 + (1-h_1)(1-h_2) h_3 t_3 + \cdots + (1-h_1)(1-h_2)\cdots(1-h_{n-1}) t_n .$$

Effective

→ The total cost of a memory hierarchy is calculated as

$$C_{total} = \sum_{i=1}^{n} c_i s_i$$

$s_i > 0,\ t_i > 0$ for $i = 1, 2, \ldots n$

c_i = cost per byte ; $c_1 > c_2 > c_3 > \cdots > c_n$

s_i = memory size ; $s_1 < s_2 < s_3 < \cdots < s_n$

⇒ $T_{eff} \rightarrow t_1$ of M_1

$C_n \rightarrow M_n$

$$C_{total} = \sum_{i=1}^{n} c_i \cdot s_i < C_0$$

C_0 – Ceiling Cost

Ques:- Consider the design of a three level memory hierarchy with the following specifications for memory characteristics.

Memory level	Access Time	Capacity	Cost / byte
Cache	$t_1 = 25ns$	$S_1 = 512$ Kbytes	$C_1 = \$ 1.25$
Main memory	$t_2 = ?$	$S_2 = 32$ Mbytes	$C_2 = \$ 0.2$
Disk Array	$t_3 = 4ms$	$S_3 = ?$	$C_3 = \$ 0.0002$

The design goal is to achieve an effective memory access time $t_{eff} = 10.04$ ns with a Cache hit $h_1 = 0.90$

Main hit $h_2 = 0.9$.
Memory

The total cost of the memory hierarchy (ceiling) is upper bounded by $15,000. Calculate S_2 and t_2

Ans:- According memory hierarchy cost-

for S_3

$$C = c_1 s_1 + c_2 s_2 + c_3 s_3 \leq 15000$$

$$1.25 \times 512 + 0.2 \times 32 \times 2 + 0.0002 \times S_3 \leq 15000$$

$$S_3 = 39.8 \text{ Gbytes}$$

for t_2 According to effective memory access time is given as

$$t_{eff} = h_1 t_1 + (1-h_1) h_2 t_2 + (1-h_1)(1-h_2) h_3 t_3 \leq 10.04$$

$$10.04 \times 10^{-6} = 0.90 \times 25 \times 10^{-9} + 0.02 \times 0.9 \times t_2 + 0.02 \times 0.1 \times 1 \times 4 \times 10^{-3}$$

6 B	What is the mapping for cache organization? Explain the following with example: **a. Associative Mapping b. Direct Mapping c. Set Associative Mapping** Consider a cache consisting of **256 blocks of 16 words each**, for a total of **4096 (4K) words**; and assume that the main memory is addressable by a **16-bit address** and it consists of **4K blocks**. How many bits are there in each of **TAG, BLOCK / SET, WORD** and word fields for all mapping technique.	CO4	K3

Case2:-

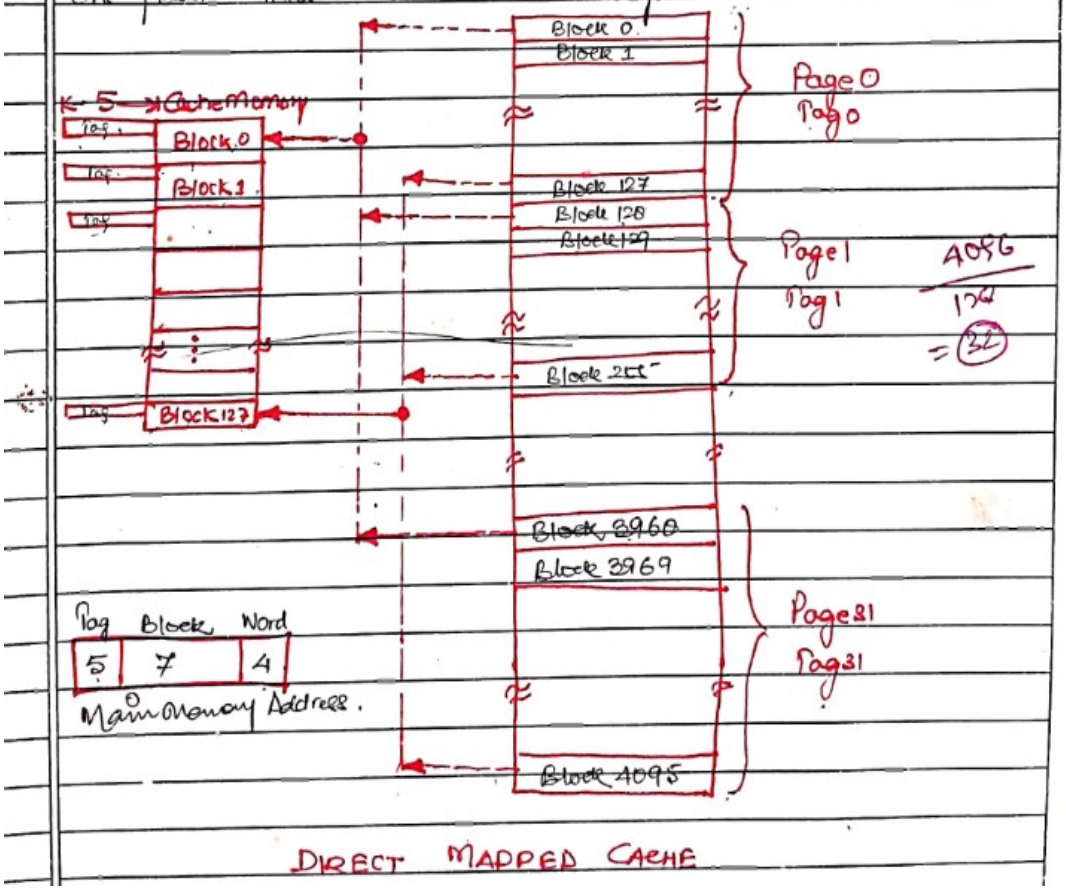
Cache consists of 128 blocks of 16 words each, for total 2048(2K) words, and assume that the main memory has 64K words. This 64K words of main memory is addressable by a 16-bit address and it can viewed as a 4K blocks of 16 word each. The group of 16 words each in main memory

$$\text{Total No. of blocks in cache} = \frac{\text{Total words in cache}}{\text{Total words in one Block}}$$

$$\text{Total No. of blocks in cache} = \frac{2048}{16} = 128 \text{ blocks.}$$

Direct Mapping:-

In this technique, each block from the main memory has only one possible location in the cache organization.



DIRECT MAPPED CACHE

2. In this case the block i of the main memory maps block i modulo 128 of the cache. ie Whenever one of the main memory blocks 0, 128, 256, ... 3968 is loaded in the cache it stored in the cache block 0.

Block 0 : 0, 128, 256, ... 3968

Block 1 : 1, 129, 257, ... 3969

⋮

Block 127 : 127, 255, ... 4095

Tag	Block	Word
5	7	4

Main Memory Address.

3. To implement such system, the address is divided into three fields, as shown in above fig.

3.1. The lower order 4-bits select ~~of~~ one of the 16 words in a block. This field is known as word field.

3.2. The second field known as block field is used to distinguish a block from other blocks. Its length is 7 bits since $2^7 = 128$. When a new block enters the cache, the 7-bit cache block field determines the cache position in which this block must be stored.

3.3 The third ~~bit~~ field is a tag field. It is used to ~~record~~ store the higher-order 5-bits of memory address of the block; These 5 bits (tag bits) are used to identify which of the 32 blocks (pages) that are mapped into cache.

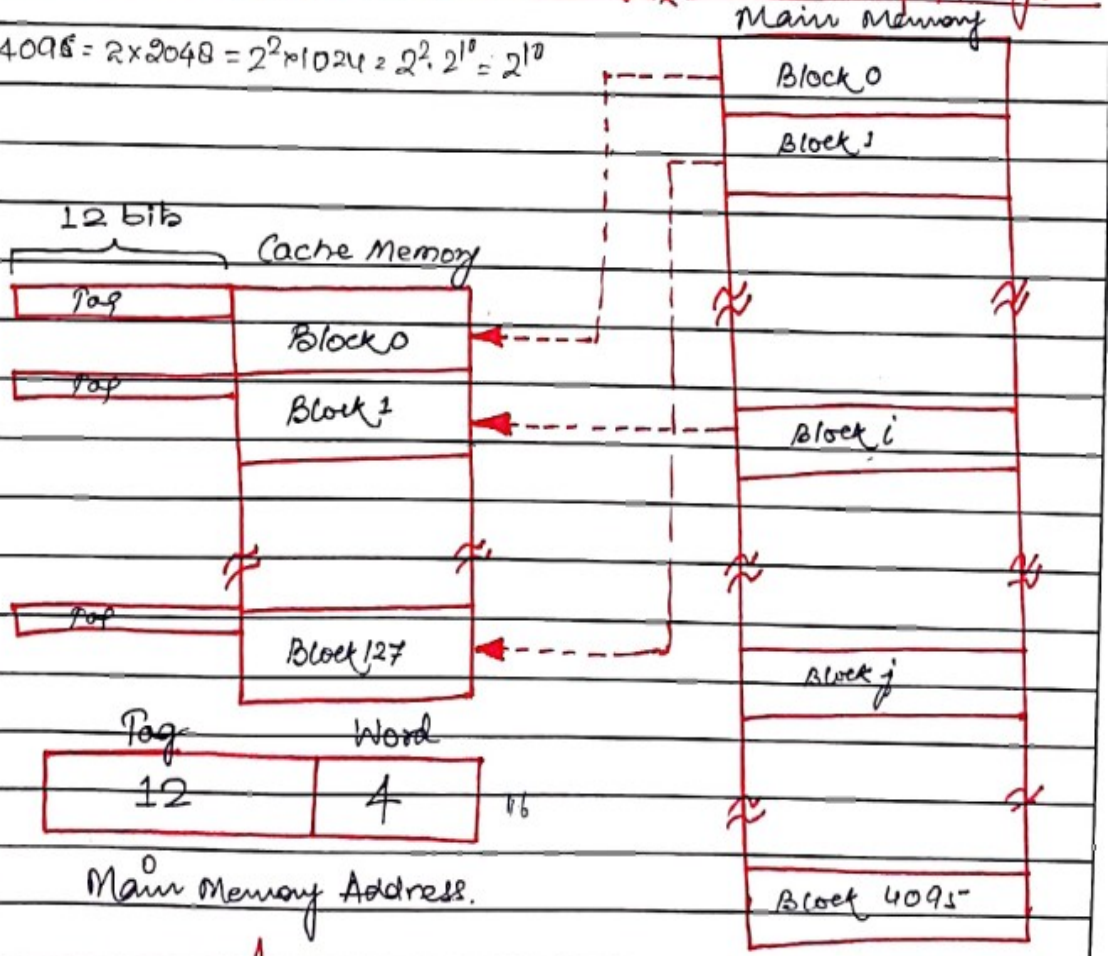
4. functioning :- 1. When memory is accessed, the 7 bit cache block field of each address generated by processor point to a particular block location in the cache.

2. The higher-order 5 bits of the address compared with the tag bits associated with that cache location. If they match, then the desired word is in that block of the cache. If there is no matched, then the block containing the required word must first be read from the main memory and loaded into cache. (Vivek Kumar Singh)

3. This means that to determine whether requested word is in the cache, only tag field is necessary to be compared. This needs only one comparison.

4. The main drawback of direct mapping cache is that if processor needs to access same memory locations from two different pages of the main memory frequently, the controller has to access main memory frequently. Since only one of these locations can be in the cache at a time.

ASSOCIATIVE MAPPING (fully Associative Mapping):-



ASSOCIATIVE-MAPPED CACHE

In this technique, a main ~~block~~ memory block can be placed into any cache position.

2. As there is no fix block, the memory address has only two field: Word field
Tag field.
This techniques is also known as fully-Associated Cache.

3. The 12-bits are required to identify a memory block when it is resident in the ~~cache~~ cache.

4. The higher order 12-bits of an address received from the processor are compared to the tag bits of each block of the cache to see if the desired block is present. Once the desired block is present, the 4-bit word is used to identify the necessary word from the cache. This technique gives complete freedom in selecting the cache location in which to place the memory block. Thus the memory space in the cache can be used more efficiently.

5. A new block, that has to be loaded into the cache has to replace (remove) an existing block, with the help of a replacement algorithm if the cache is full.

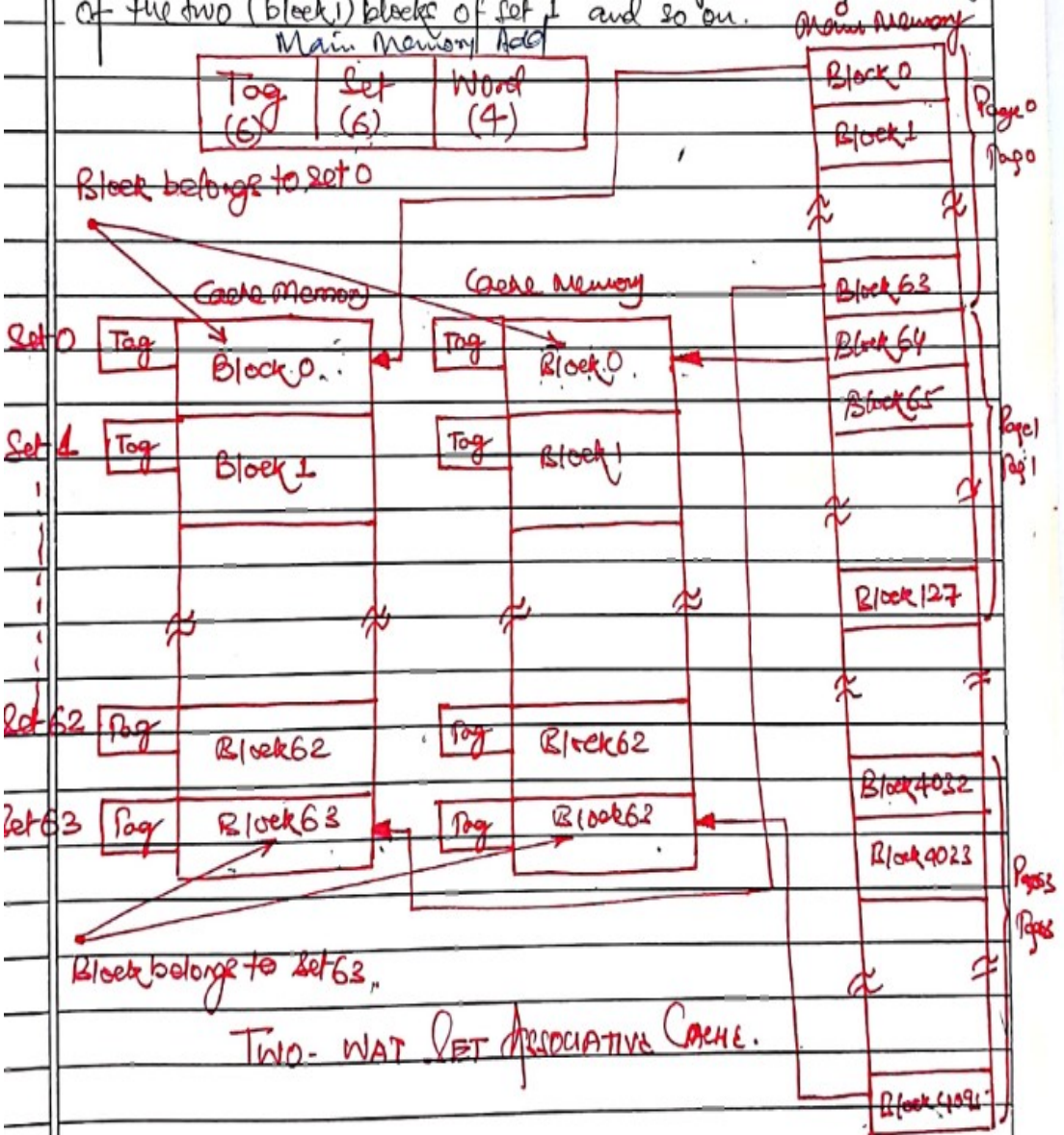
6. In associative mapped cache, it is necessary to compare the higher order bits of address of the main memory with all 128 tags corresponding to each block to determine whether a given block, is in the cache. This is the main disadvantage of associative-mapped Cache.

3. Set Associative Mapping :—

1. The set associative mapping is a combination of both direct- and associative mapping.
2. It contains several groups of direct mapped blocks that operates as several direct mapped cache in parallel.
3. A block of data from any page in the main memory can go into a particular block location of any direct-mapped Cache.
4. The above fig shows that, the two-way set-associative [illegible] each page

in the main memory is organized in such a way that the size of each page is same as the size of one directly mapped cache.

5. It is called two-way set associative cache because each block from main memory has two choices for block placement. In this technique block 0, 64, 128, ... 4032 of main memory can map into any of the two (block 0) blocks of set 0, block 1, 65, 129, ... 4033 of main memory can map into any of the two (block 1) blocks of set 1 and so on.



TWO-WAY SET ASSOCIATIVE CACHE.

5. As there are two choices, it is necessary to compare address of memory with the tag bits of corresponding two block locations of particular set.

for two way set associative cache we requires two comparision to determine whether a given block is in the cache. Since there are two direct mapped caches, any two byte having same offset from different page can be in the cache at a time. This improves the hit rate of cache system.

Que Consider a cache consisting of 256 blocks of 16 words each, for a total of 4096 (4K) words; and assume that the main memory is addressable by a 16 bit address and it consists of 4K blocks. How many bits are there in each of the TAG, BLOCK/SET and word fields for different mapping technique.

Ans

Block in cache = 256 blocks ; = 2^8 blocks.

Size of one block of cache = 16 words.

Total words in cache = 4096 = 4K words.

Size of main memory address = 16 bit

Total block is main memory = $4K = 2^2 \cdot 2^{10} = 2^{12}$ blocks.

1. Direct mapping

Tag	Block	Word	
4	8	4	Main memory Address.

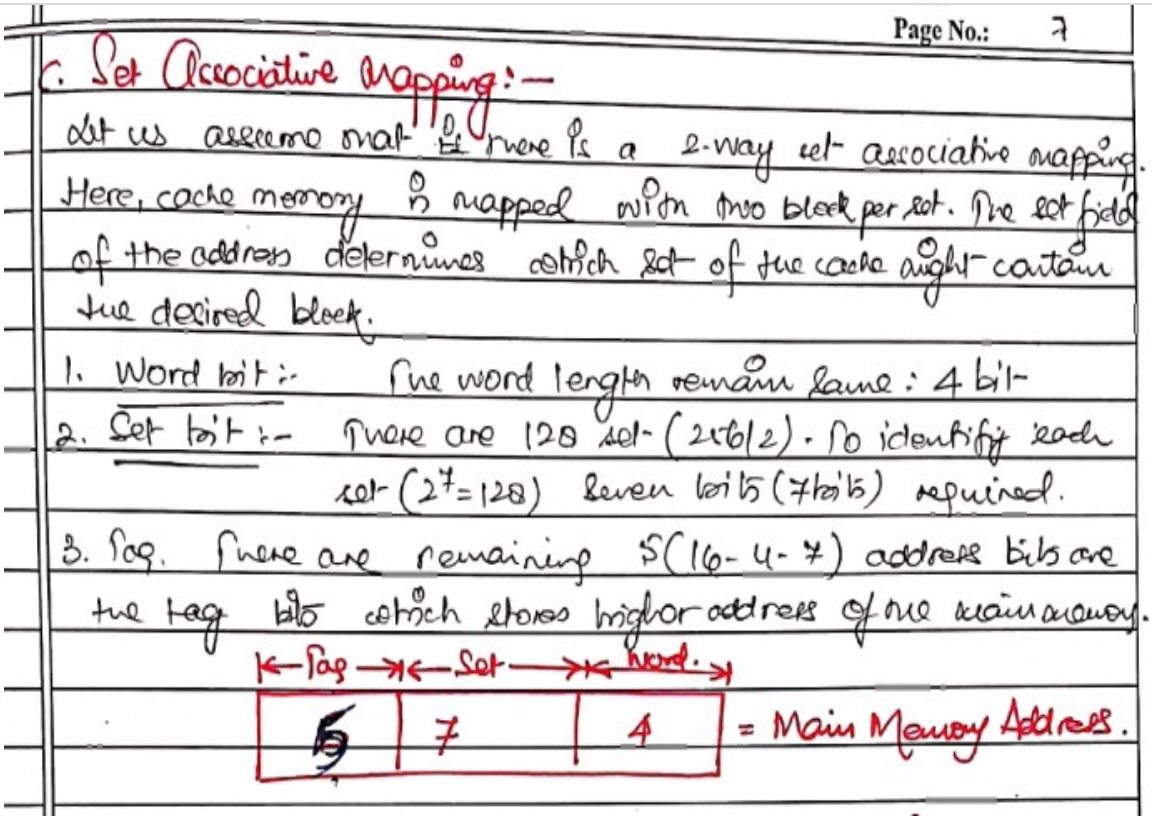
1. Word bit : each block consist of 16 word
2. Block bit : The cache memory used consist of 256 blocks and using direct mapping technique block of $256 = 2^8$
3. Tag bits : The remaing 4 (16-8-4) address bits are tag bits which stores the higher address of main memory.

2. Associative mapping :

1. Word bit : The word length is remain same i.e 4 bit.
2. Block bit : 0
3. Tag bit : To address each block in main memory ($2^{12} = 4096$) 12 bits are required and therefore, there are 12 tag bits.

(Vivek Kumar Singh)

Tag.	Word	
12	4	Main Memory Address.



Q 07 Attempt any ONE Part of the Following **[7 x 1 = 7]**

	Course Outcome(CO) & Bloom's Taxonomy Level(BTL) for each question	CO	BT L
7 A	**What are the techniques used for I/O operation. With the help of block diagram, discuss the working of Direct Memory Access (DMA). Also explain the main features of an IOP (Input / Output processor). Give a brief comparison of DMA and IOP.** **1. PROGRAMMED INPUT /OUTPUT TECHNIQUE** ✓ Programmed input/output is a useful I/O method for computers where hardware costs need to be minimized. The input or output operation in such cases may involve: ▪ Transfer of data from I/O device to the processor registers. ▪ Transfer of data from processor registers to memory.	CO 5	K4

- ✓ With the programmed I/O method, the responsibility of the processor is to constantly check the status of the I/O device to check whether it is free or it has finished inputting the data. Thus, this method is very time consuming where the processor wastes a lot of time in checking and verifying the status of an I/O device**. Figure U5.5(a)** gives an example of the use of programmed I/O to read in a block of data from a peripheral device into memory.
- ✓ **I/O COMMANDS :** There are four types of I/O commands that an I/O interface may receive when it is addressed by a processor:
  - **CONTROL:** These commands are device specific and are used to provide specific instructions to the device, e.g. a magnetic tape requiring rewinding and moving forward by a block.
  - **TEST:** This command checks the status such as if a device is ready or not or is in error condition.
  - **READ:** This command is useful for input of data from input device.
  - **WRITE:** this command is used for output of data to output device.

- ✓ **I/O INSTRUCTIONS:**
  - An I/O instruction is stored in the memory of the computer and is fetched and executed by the processor producing an I/O-related command for the I/O interface.
  - With programmed I/O, there is a close correspondence between the I/O-related instructions and the I/O commands that the processor issues to an I/O interface to execute the instructions.
  - In systems with programmed I/O, the I/O interface, the main memory and the processors normally share the system bus. Thus, each I/O interface should interpret the address lines to determine if the command is for itself.
  - There are two methods for doing so.
    - ➢ **MEMORY-MAPPED I/O**
    - ➢ **ISOLATED I/O.**

- **MEMORY-MAPPED I/O :-**
  - With memory-mapped I/O, there is a single address space for memory locations and I/O devices.
  - The processor treats the status and data registers of I/O interface as memory locations and uses the same machine instructions to access both memory and I/O devices.
  - For a memory-mapped I/O only a single read and a single write line are needed for memory or I/O interface read or write operations.
  - These lines are activated by the processor for either memory access or I/O device access.
  - Figure U5.6 shows the memory-mapped I/O system structure.

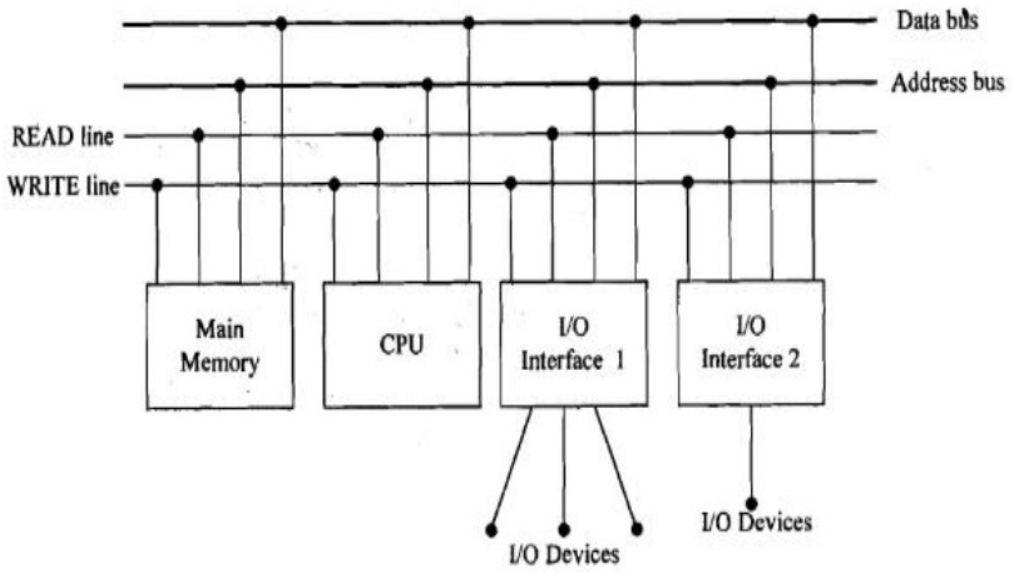


Figure U5.6 : Structure of Memory Mapped I/O

- **ISOLATED I/O :-**
  - With isolated I/O, there are separate control lines for both memory and I/O device read or write operations.
  - Thus a memory reference instruction does not affect an I/O device.
  - In isolated I/O, the I/O devices and memory are addressed separately; hence separate input/output instructions are needed which cause data transfer between addressed I/O interface and processor.
  - Figure 7 shows the structure of isolated I/O.

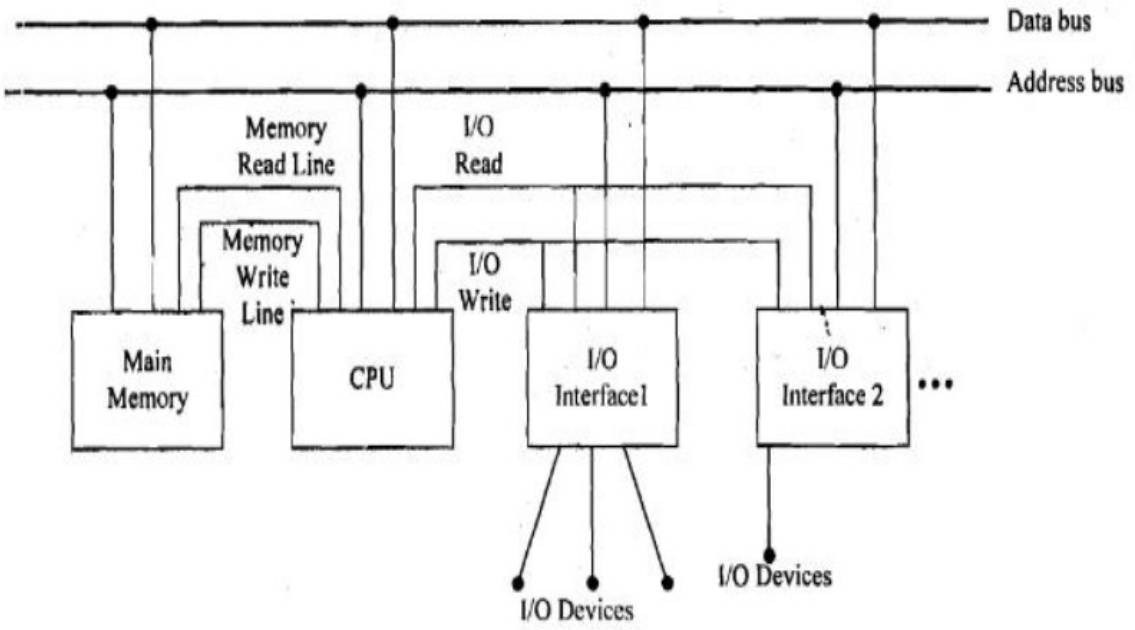


Figure U5.7: Structure of Isolated I/O

2. ITERRUPT DRIVEN INPUT /OUTPUT TECHNIQUE

- ✓ The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data.
- ✓ The processor, while waiting, must repeatedly interrogate the status of the I/O module. As a result, the level of the performance of the entire system is severely degraded.
- ✓ An alternative is for the processor to issue an I/O command to a module and then go on to do some other useful work. The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor.
- ✓ The processor then executes the data transfer, as before, and then resumes its former processing.

WORKING:

I. Let us consider how this works, first from the point of view of the I/O module. For input, the I/O module receives a READ command from the processor. The I/O module then proceeds to read data in from an associated peripheral. Once the data are in the module's data register, the module signals an interrupt to the processor over a control line. The module then waits until its data are requested by the processor. When the request is made, the module places its data on the data bus and is then ready for another I/O operation.

II. From the processor's point of view, the action for input is as follows. The processor issues a READ command. It then goes off and does something else (e.g., the processor may be working on several different programs at the same time). At the end of each instruction cycle, the processor checks for interrupts. When the interrupt from the I/O module occurs, the processor saves the context (e.g., program counter and processor registers) of the current program and processes the interrupt. In this case, the processor reads the word of data from the I/O module and stores it in memory. It then restores the context of the program it was working on (or some other program) and resumes execution.

- ✓ Figure U5.5b shows the use of interrupt I/O for reading in a block of data. Compare this with Figure U5.5a. Interrupt I/O is more efficient than programmed I/O because it eliminates needless waiting. However, interrupt I/O still consumes a lot of processor time,

because every word of data that goes from memory to I/O module or from I/O module to memory must pass through the processor.

INTERRUPT PROCESSING

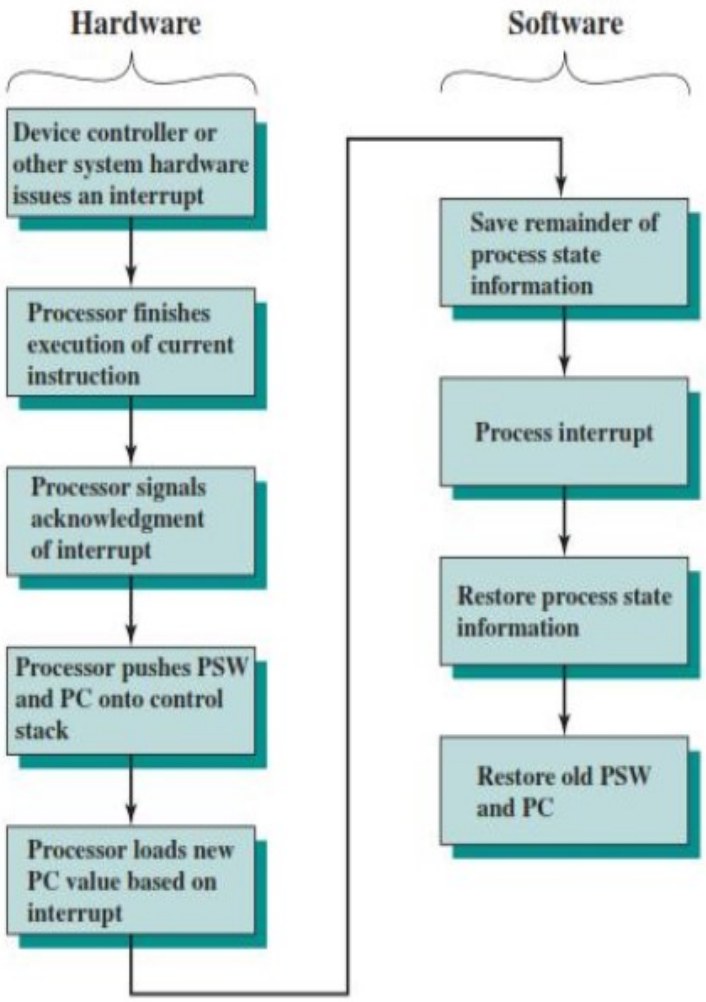


Figure U5.8: Simple Interrupt Processing

The occurrence of an interrupt triggers a number of events, both in the processor hardware and in software. Figure U5.8 shows a typical sequence. When an I/O device completes an I/O operation, the following sequence of hardware events occurs:

- ✓ The device issues an interrupt signal to the processor.
- ✓ The processor finishes execution of the current instruction before responding to the interrupt.
- ✓ The processor tests for an interrupt, determines that there is one, and sends an acknowledgment signal to the device that issued the interrupt. The acknowledgment allows the device to remove its interrupt signal.

- ✓ The processor now needs to prepare to transfer control to the interrupt routine. To begin, it needs to save information needed to resume the current program at the point of interrupt. The minimum information required is
  - the status of the processor, which is contained in a register called the program status word (PSW)
  - the location of the next instruction to be executed, which is contained in the program counter. These can be pushed onto the system control stack.
- ✓ The processor now loads the program counter with the entry location of the interrupt-handling program that will respond to this interrupt.
- ✓ Once the program counter has been loaded, the processor proceeds to the next instruction cycle, which begins with an instruction fetch. Because the instruction fetch is determined by the contents of the program counter, the result is that control transferred to the interrupt-handler program. The execution of this program results in the following operations.
- ✓ At this point, the program counter and PSW relating to the interrupted program have been saved on the system stack.
- ✓ The interrupt handler next processes the interrupt. This includes an examination of status information relating to the I/O operation or other event that caused an interrupt. It may also involve sending additional commands or acknowledgments to the I/O device.
- ✓ When interrupt processing is complete, the saved register values are retrieved from the stack and restored to the registers.
- ✓ The final act is to restore the PSW and program counter values from the stack. As a result, the next instruction to be executed will be from the previously interrupted program.

3. DIRECT MEMORY ACCESS (DMA)

- ✓ **DRAWBACKS OF PROFRAMMED & INTERRUPT-DRIVEN I/O:-** Interrupt-driven I/O, though more efficient than simple programmed I/O, still requires the active intervention of the processor to transfer data between memory and an I/O module, and any data transfer must traverse a path through the processor.
- ✓ Thus, both these forms of I/O suffer from two inherent drawbacks:
  - ❖ The I/O transfer rate is limited by the speed with which the processor can test and service a device.
  - ❖ The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer
- ✓ When large volumes of data are to be moved, a more efficient technique is required: **DIRECT MEMORY ACCESS (DMA).**
- ✓ DMA involves an additional module on the system bus.
- ✓ During DMA transfer, the CPU (Processor) 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.

✓ Figure U5.10 shows two control signals in the CPU that facilitate the DMA transfer.

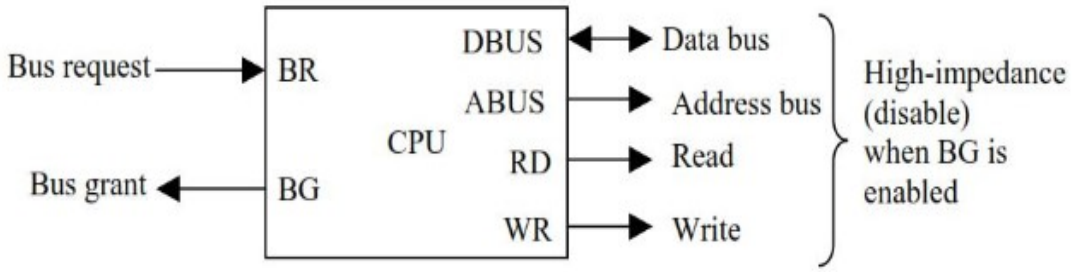


Figure U5.10: CPU bus signals for DMA transfer

✓ The **BUS REQUEST (BR)** input is used by the DMA controller to request the CPU to release 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 behaves like an open circuit, which means that the output is disconnected and dies not have a logic significance.

✓ 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(BG)**, 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 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 ARCHITECTURE:-

- ✓ The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device.

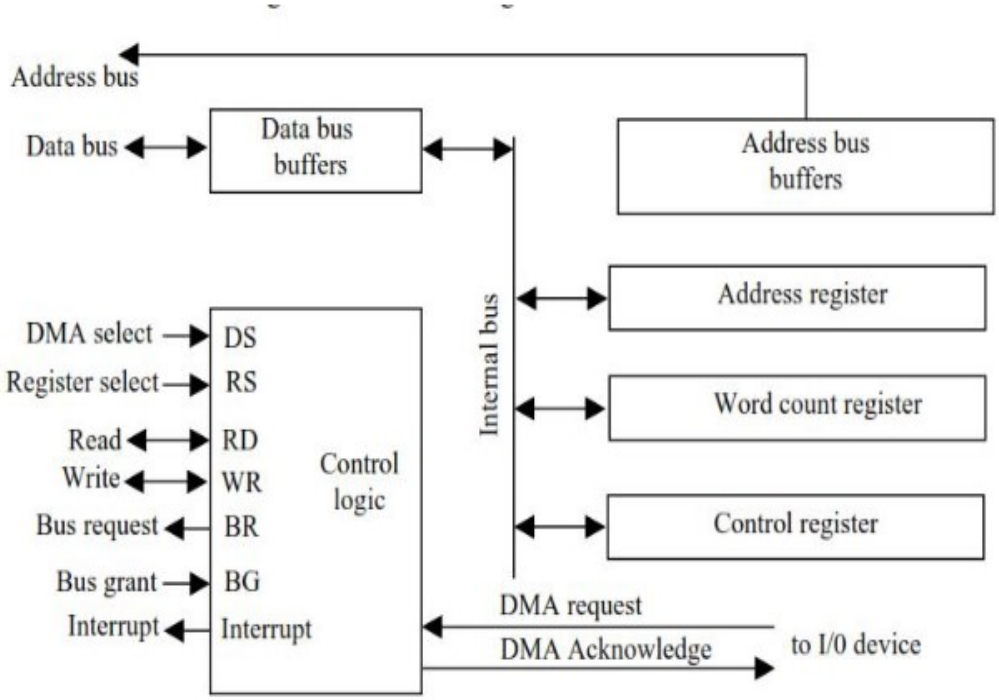


Figure U5.11: Block diagram of DMA Controller

- ✓ 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.
- ✓ Figure U5.11 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 with following reason:
  - ❖ 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

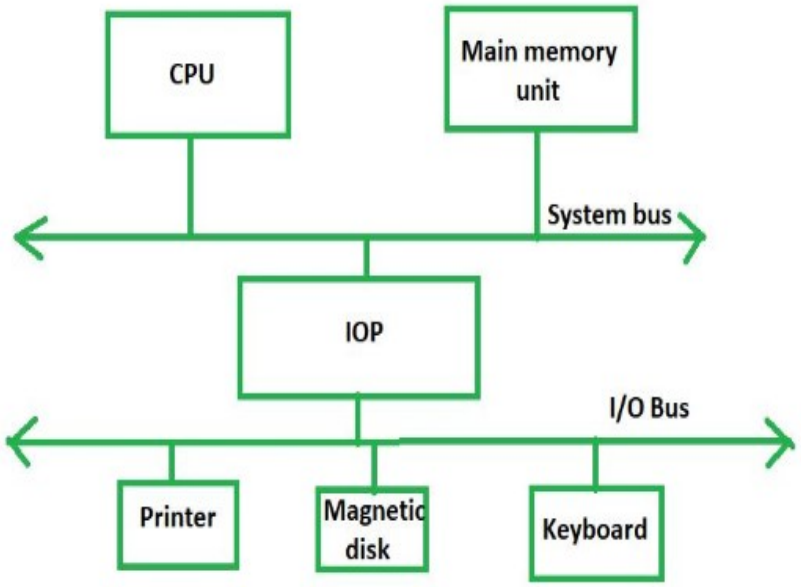
activating the RD or WR control. The DMA communicates with the external peripheral through the request and acknowledge lines by using a prescribed handshaking procedure.

- ✓ The DMA controller has three registers:
  - ❖ An Address Register
  - ❖ A Word Count Register
  - ❖ A Control Register
- ✓ **AN ADDRESS 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.
- ✓ **A WORD COUNT REGISTER:** The word count register is decremented 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.
- ✓ **A CONTROL REGISTER:**- The control register specifies the mode of transfer.
- ✓ All registers in the DMA appear to the CPU as I/O interface registers. Thus the CPU can read from or write into the DMA registers under program control via the data bus.
- ✓ 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:
  - ❖ The starting address of the memory block where data are available (for read) or where data are to be stored (for write)
  - ❖ The word count, which is the number of words in the memory block
  - ❖ Control to specify the mode of transfer such as read or write
  - ❖ 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.

7 B	**Discuss the I/O processor organization with the help of block diagram and write algorithmic steps for CPU – IOP interaction.**	CO 5	K2

4. I/O PROCESSOR:-

- ✓ The DMA mode of data transfer reduces CPU's overhead in handling I/O operations.
- ✓ It also allows parallelism in CPU and I/O operations. Such parallelism is necessary to avoid wastage of valuable CPU time while handling I/O devices whose speeds are much slower as compared to CPU.
- ✓ The concept of DMA operation can be extended to relieve the CPU further from getting involved with the execution of I/O operations. This gives rises to the development of special purpose processor called Input-Output Processor (IOP) or IO channel.
- ✓ The Input Output Processor (IOP) is just like a CPU that handles the details of I/O operations.
- ✓ It is more equipped with facilities than those are available in typical DMA controller.



Input-Output Processor

Figure U5.15: Input-Output Organization

- ✓ The IOP can fetch and execute its own instructions that are specifically designed to characterize I/O transfers.
- ✓ In addition to the I/O – related tasks, it can perform other processing tasks like arithmetic, logic, and branching and code translation.
- ✓ The main memory unit takes the pivotal role. It communicates with processor by the means of DMA.

	✓ The Input Output Processor is a specialized processor which loads and stores data into memory along with the execution of I/O instructions. ✓ It acts as an interface between system and devices. ✓ It involves a sequence of events to executing I/O operations and then store the results into the memory. **ADVANTAGE:** ✓ The I/O devices can directly access the main memory without the intervention by the processor in I/O processor based systems. ✓ It is used to address the problems that are arises in Direct memory access method.		

I/O Processor Technique -

(1) The concept of I/O Processor is an extension of the concept of DMA.

(2) The I/O processor can execute specialised I/O program stored in the memory without involvement of processor.

(3) Processor only needs to specify a sequence of I/O activity to the I/O processor. I/O processor then execute the necessary I/O instructions which are required for I/O operation and interrupt the processor only after the entire sequence of I/O activity as specified by the processor is completed.

(4) An advanced I/O processor can have its own memory enabling a large set of I/O devices to be controlled without much involvement of the processor.

(5) I/O processor has an additional ability to execute I/O instructions which provide a complete control on the I/O operation.

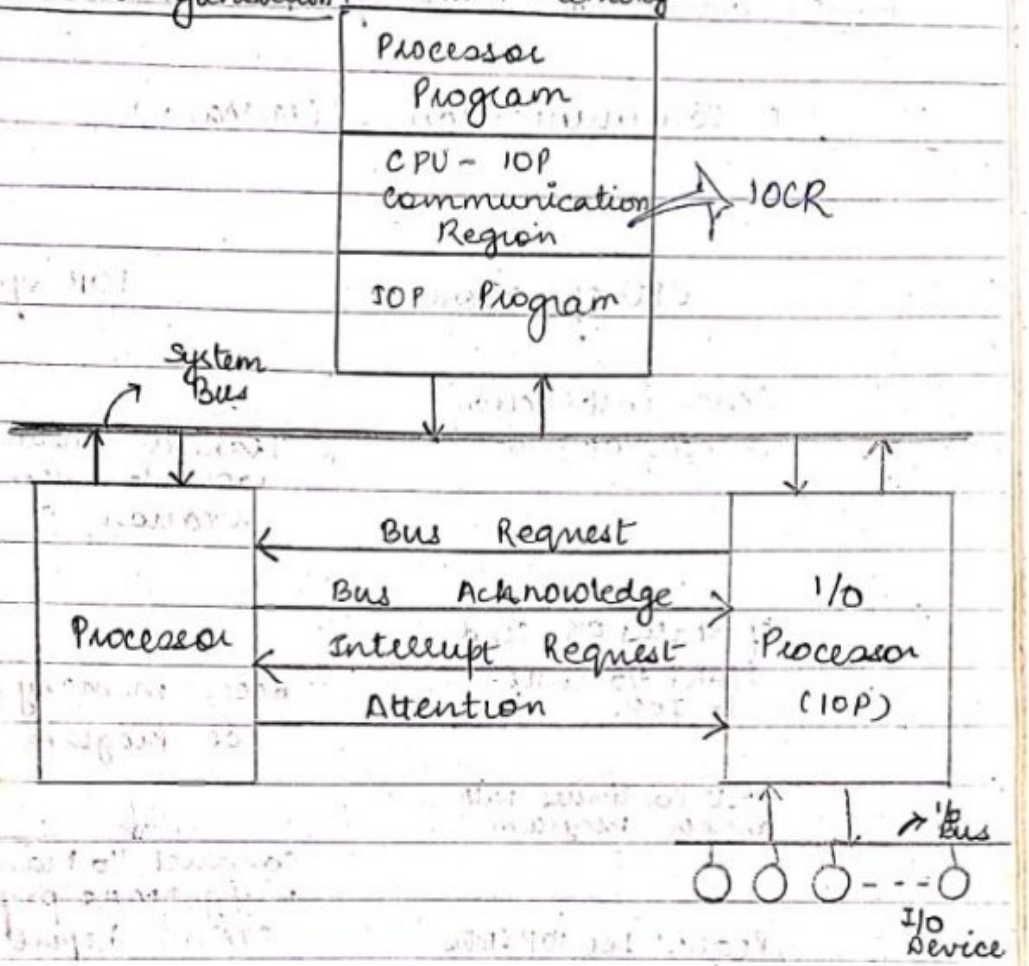
⑥ I/O processors are much powerful than DMA which provides the control to the I/O device.

⑦ The communication b/w I/O processor and processor can be made by writing a message in the memory area shared by both the processors.

⑧ Processor instruct I/O processor to execute an I/O program in memory. The program will then specify the device or devices and the area of the memory where the I/O data is stored or to be stored.

⑨ This program also contains what actions are to be taken in case of error or what priority is to be given to various input-output devices.

I/O Processor Organization:



Processor - IOP Interaction:- (+ CPU - IOP communi.)

WAIT : if ATTENTION = 1, then begin
Fetch parameters from IOCR;

SETUP : setup DMA control Registers;
Begin I/O program execution;
send command to I/O devices;

SEND : Transmit data word;
if transmission error then go to EXIT;
if not end of data, then go to SETUP;

EXIT : Place termination status in IOCR;
end
go to WAIT;

CPU - IOP Communication :- (M. Mano)