K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109

FIRST INTERNAL TEST QUESTION PAPER 2023-24 ODD SEMESTER

SET: A

USN | | | | | | | | | | |

Degree :	B.E	Semester :	3rd
Branch - Stream :	CSE - ICB	Course Type / Code :	Integrated \ BCS302
Course Title :	Digital Design and Computer Organization	Date :	02/01/2024
Duration :	1 ½ Hr (90 minutes)	Max Marks :	50

Note: Answer **ONE full** question from each Module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module 1			
1(a)	**Solve** the following function using K-Map and obtain simplified Boolean expressions. (i) f1 (a, b, c, d) = $\sum$ m (1, 3, 4, 5, 7, 10, 12) (ii) f2 (a, b, c, d) = $\sum$ m (5, 8, 9, 10, 11, 12, 13, 14, 15)	10	CO1	K3
(b)	By **Making use of** the K Map technique solve the following function for SOP and implement it using the basic gates. (i) $f(a,b,c,d) = \prod M(5,7,13,14,15) + d(1,2,3,9)$	10	CO1	K3
	OR			
2(a)	**Solve** $f(a, b, c, d) = \prod M(0,6,7,8,12,13,14,15)$ using K Map to get minimum POS expression. Implement using **BASIC GATES, NAND only and NOR only gates.**	10	CO1	K3
(b)	By **Making use of** the K Map technique solve the following function for POS and implement it using the basic gates. (i) $f(a,b,c,d) = \prod M(0,1,6,8,11,12) + d(3,7,4,15)$	10	CO1	K3
	Module 2			
3(a)	**Construct** a 3:8 decoder for the following functions and define multiplexer and explain 8 to 1 mux with the help of a logic diagram and corresponding expression. (i) f1 (a, b, c) = $\sum$ m (0,4,6,7) (ii) f2(a, b, c) = $\sum$ m (1,4,5)	10	CO2	K3
(b)	By **Making use of** EVM method, simplify the following function and implement it by using 8:1 MUX. Assume D as MEV variable. f (a, b, c, d) = $\sum$ m (0,1,2,4,5,6,9,10,12,13,14,15)	10	CO2	K3
	OR			
4(a)	**Construct** a 32-to-1 multiplexer using two 16-to-1 multiplexer and one 2-to-1 multiplexer and explain octal to binary encoder with truth table and write the boolean output functions.	10	CO2	K3
(b)	**Design** a **3-to-8-line Decoder** with circuit, truth table and logic diagram.	10	CO2	K3
	Module 3			
5(a)	**Explain** the basic operation concept behind working of computers.	10	CO3	K2
	OR			
6(a)	**Write a note** on addressing mode and explain any four types of addressing modes with suitable examples.	10	CO3	K2

Rachana V Murthy [illegible]
Name & Signature of Course In charge:

Kushal Kumar BN [illegible]
Name & Signature of Module Coordinator:

[illegible]
HOD

[illegible]
Principal

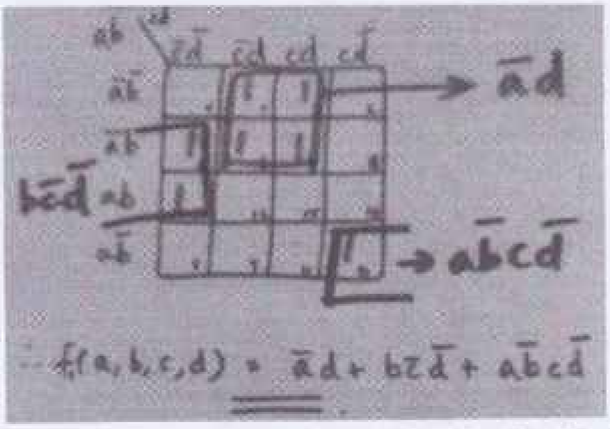
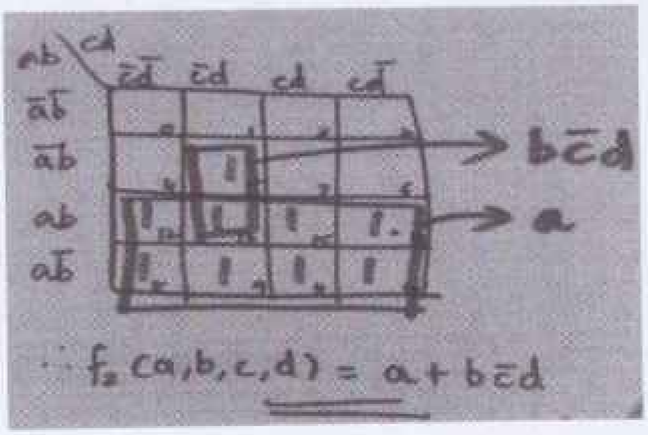
K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

III SESSIONAL TEST QUESTION PAPER 2022 – 23 ODD SEMESTER

Set A

SCHEME AND SOLUTION

Degree	:	B. E	Semester	:	III
Branch	:	Computer Science - ICB	Course Code	:	Integrated \ BCS302
Course Title	:	Digital Design and Computer Organization	Max Marks	:	50

Q.NO.	POINTS	MARKS
1(a)	**Solve** the following function using K-Map and obtain simplified Boolean expressions. **(i)** **f1 (a, b, c, d) = ∑ m (1, 3, 4, 5, 7, 10, 12)** **(ii)** **f2 (a, b, c, d) = ∑ m (5, 8, 9, 10, 11, 12, 13, 14, 15)** i) f_1 (a, b, c, d) = ∑ m (1, 3, 4, 5, 7, 10, 12)  ab cd $\bar{c}\bar{d}$ $\bar{c}d$ cd $c\bar{d}$ $\bar{a}\bar{b}$ $\bar{a}b$ ab $a\bar{b}$ $\rightarrow \bar{a}d$ $b\bar{c}\bar{d}$ $\rightarrow a\bar{b}c\bar{d}$ $\therefore f(a,b,c,d) = \bar{a}d + b\bar{c}\bar{d} + a\bar{b}c\bar{d}$ ii) f_2 (a, b, c, d) = ∑ m (5, 8, 9, 10, 11, 12, 13, 14, 15)  ab cd $\bar{c}\bar{d}$ $\bar{c}d$ cd $c\bar{d}$ $\bar{a}\bar{b}$ $\bar{a}b$ ab $a\bar{b}$ $\rightarrow b\bar{c}d$ $\rightarrow a$ $\therefore f_2(a,b,c,d) = a + b\bar{c}d$	5+5=10M

(b) By **Making use of** the K Map technique solve the following function for SOP and implement it using the basic gates.

f (a, b, c, d) = $\pi M(5,7,13,14,15) + d(1,2,3,9)$

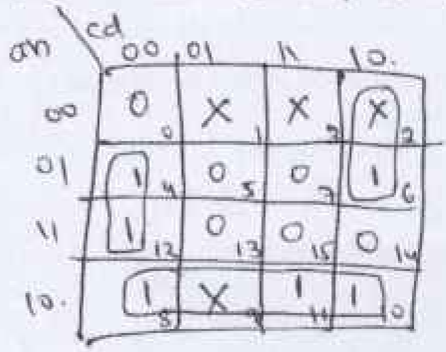
$2+3+1$
$+4=10M$

$f(a,b,c,d) = \pi m(5,7,13,14,15) + d(1,2,3,9)$

Given $f(a,b,c,d) = \pi m(5,7,13,14,15) + d(1,2,3,9)$ (2M)

converting into minterm form for Sop.

$f(a,b,c,d) = \Sigma m(4,6,8,10,11,12) + d(1,2,3,9)$



ab \ cd	00	01	11	10
00	0 (0)	X (1)	X (3)	X (2)
01	1 (4)	0 (5)	0 (7)	1 (6)
11	1 (12)	0 (13)	0 (15)	0 (14)
10	1 (8)	X (9)	1 (11)	1 (10)

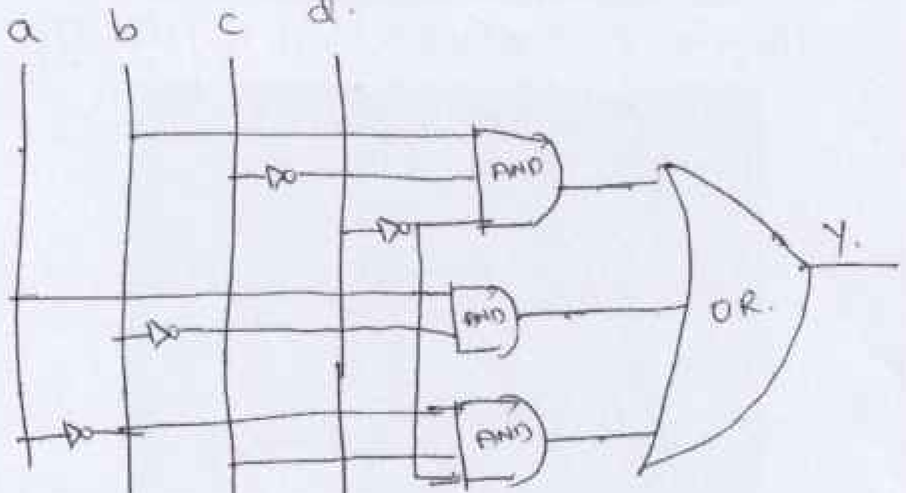
(3M)

sop expression
$= b\bar{c}\bar{d} + a\bar{b} + \bar{a}c\bar{d}$

Sop expression (1M)

$y = b\bar{c}\bar{d} + a\bar{b} + \bar{a}c\bar{d}$

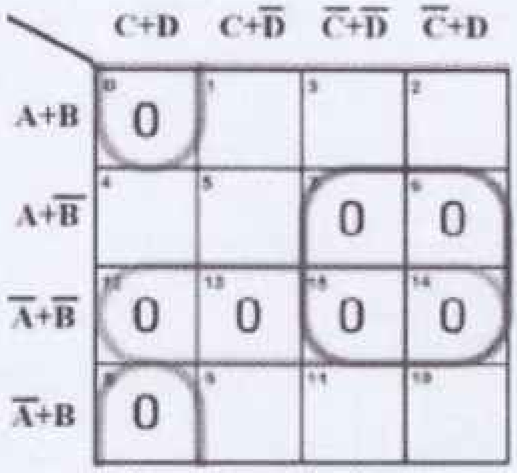
Realize the circuit using basic gates.



(4M)

2(a) Solve **f (a, b, c, d) =** $\pi M(0,6,7,8,12,13,14,15)$ using K Map to get minimum POS expression. Implement using **BASIC GATES, NAND only and NOR only gates.**

3+1+6 =10M

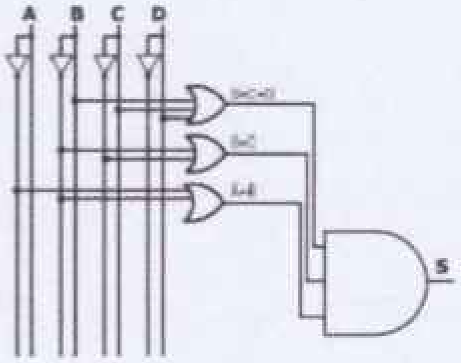


(3M)

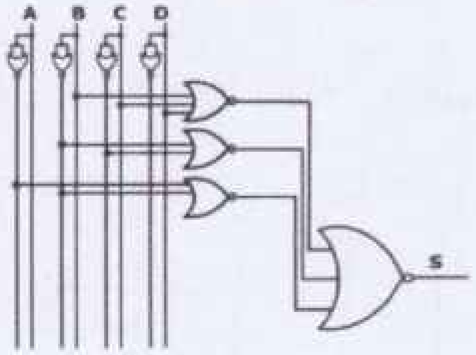
S = (B+C+D) · ($\bar{B}$+$\bar{C}$) · ($\bar{A}$+$\bar{B}$)

(1M)

Common Inverted Circuit

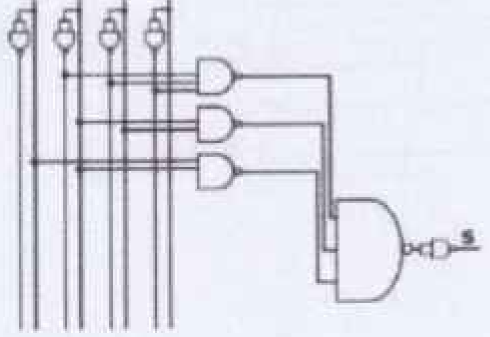


Nor Only Circuit



(2M each)

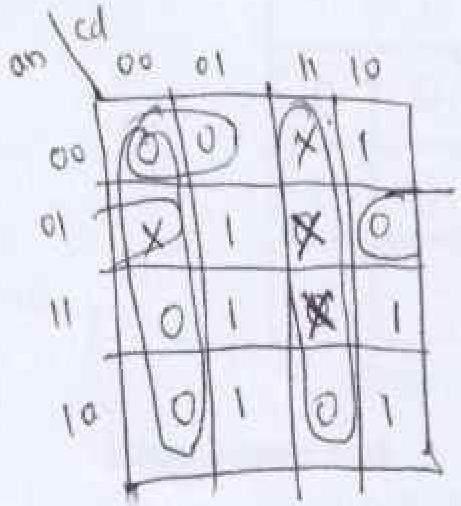
Nand Only Circuit



2(b) By **Making use of** the K Map technique solve the following function for POS and implement it using the basic gates.

f (a, b, c, d) = $\pi M(0, 1, 6, 8, 11, 12) + d(3, 7, 4, 15)$

4+1+5
=10M

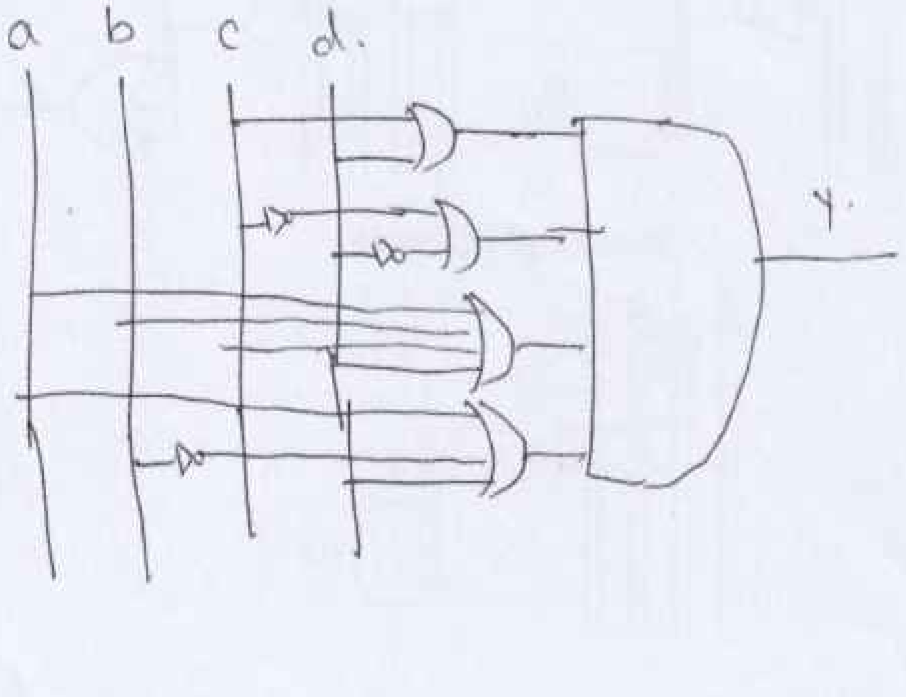


(4M)

$Y = (c+d)(\bar{c}+\bar{d})(a+b+$
$(a+\bar{b}+d)$

(1M)

Realize the expression Y.



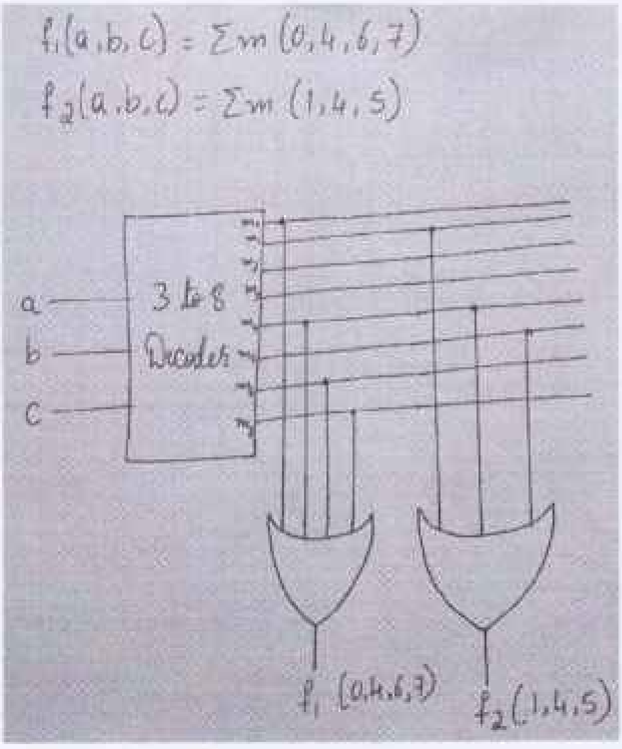
(5M)

3(a) **Construct** a 3:8 decoder for the following functions and define multiplexer and explain 8 to 1 mux with the help of a logic diagram and corresponding expression.

(i) **f1 (a, b, c) = $\sum$ m (0,4,6,7)**

(ii) **f2(a, b, c) = $\sum$ m (1,4,5)**

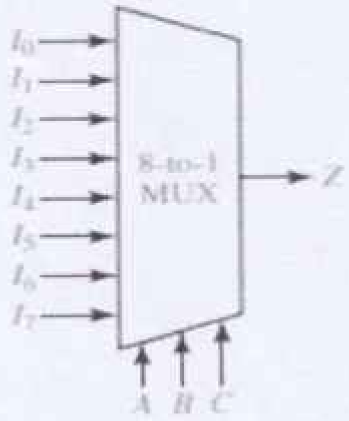
5+2+3 =10M



$f_1(a,b,c) = \Sigma m(0,4,6,7)$
$f_2(a,b,c) = \Sigma m(1,4,5)$
a
b
c
3 to 8 Decoder
f_1 (0,4,6,7)
f_2 (1,4,5)

(5M)

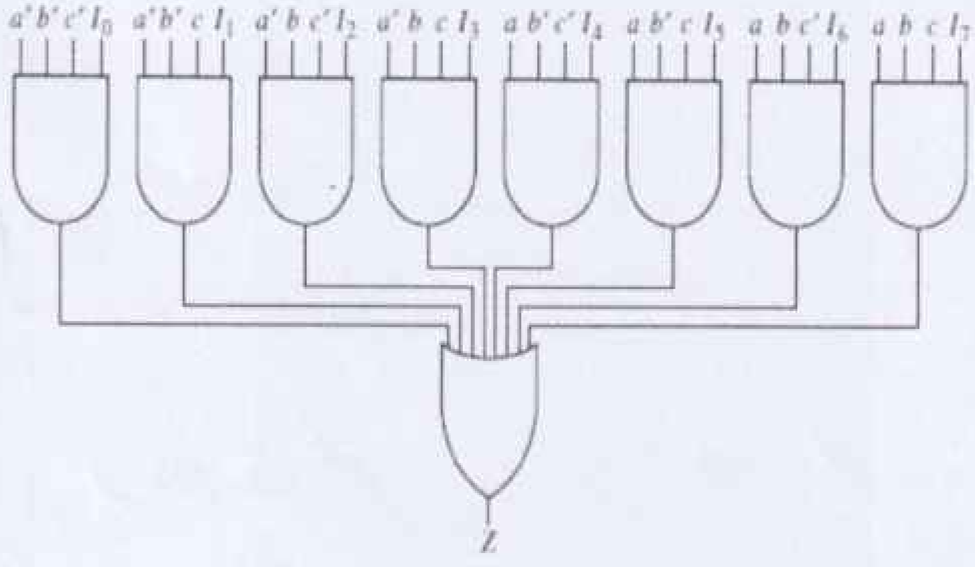
The multiplexer or MUX is a digital switch, also called as data selector. It is a Combinational Logic Circuit with more than one input line, one output line and more than one select line. It accepts the binary information from several input lines or sources and depending on the set of select lines, a particular input line is routed onto a single output line.



8-to-1 multiplexer

(2M)

It is described by the equation: $Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$. Multiplexers can also have an additional input called an *enable* input.



(3M)

3(b) By **Making use of** EVM method, simplify the following function and implement it by using 8:1 MUX. Assume D as MEV variable.

f (a, b, c, d) = ∑ m (0,1,2,4,5,6,9,10,12,13,14,15)

5+2+3 =10M

$f(a,b,c,d) = \sum m(0,1,2,4,5,6,9,10,12,13,14,15)$

Truth Table

(5M)

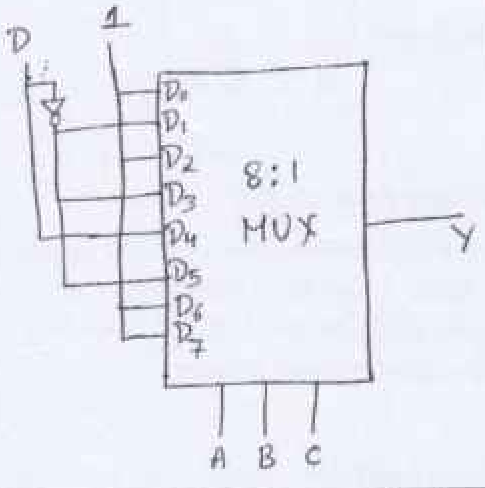
A	B	C	D	Y	
0	0	0	0	1	1
0	0	0	1	1	
0	0	1	0	1	$\overline{D}$
0	0	1	1	0	
0	1	0	0	1	1
0	1	0	1	1	
0	1	1	0	1	$\overline{D}$
0	1	1	1	0	
1	0	0	0	0	D
1	0	0	1	1	
1	0	1	0	1	$\overline{D}$
1	0	1	1	0	
1	1	0	0	1	1
1	1	0	1	1	
1	1	1	0	1	1
1	1	1	1	1	

MEV Table.

(2M)

	A	B	C	Y
0	0	0	0	1
1	0	0	1	$\overline{D}$
2	0	1	0	1
3	0	1	1	$\overline{D}$
4	1	0	0	D
5	1	0	1	$\overline{D}$
6	1	1	0	1
7	1	1	1	1

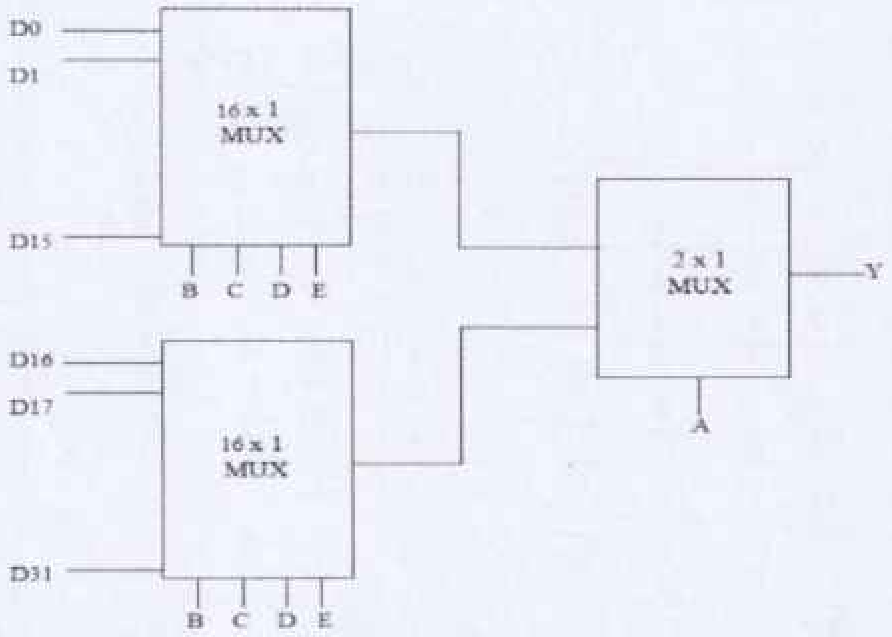
(3M)



4(a) **Construct** a 32-to-1 multiplexer using two 16-to-1 multiplexer and one 2-to-1 multiplexer and explain octal to binary encoder with truth table and write the boolean output functions.

5+3+2
=10M

32-to-1 multiplexer using two 16-to-1 multiplexer and one 2-to-1 multiplexer



(5M)

Octal To Binary Encoder

An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2n (or fewer) input lines and n output lines. The output lines, as an aggregate, generate the binary code corresponding to the input value. An example of an encoder is the octal-to-binary encoder.

Boolean output functions:

$$z = D1 + D3 + D5 + D7$$
$$y = D2 + D3 + D6 + D7$$
$$x = D4 + D5 + D6 + D7$$

(3M)

Truth Table of an Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

(2M)

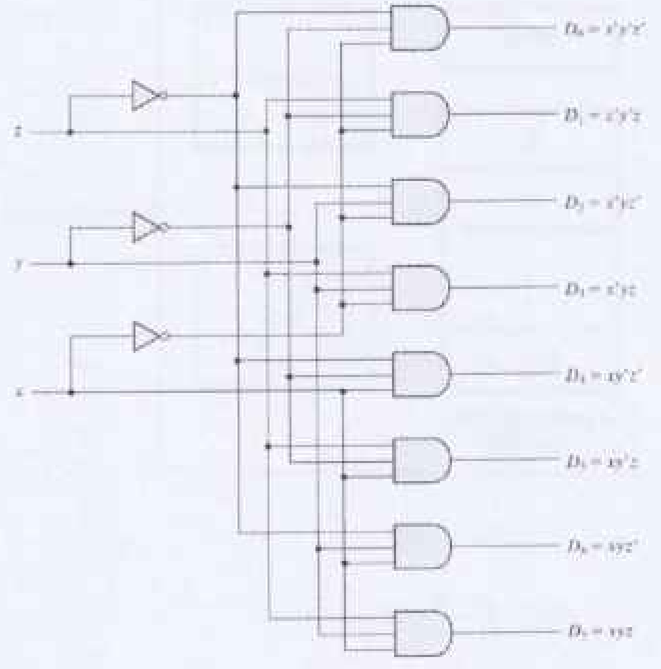
4(b) **Design** a **3-to-8-line Decoder** with circuit, truth table and logic diagram

2+5+3 =10M

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2n unique output lines. If the n -bit coded information has unused combinations, the decoder may have fewer than 2n outputs.

A particular application of this decoder is binary-to-octal conversion. The input variables represent a binary number, and the outputs represent the eight digits of a number in the octal number system. However, a three-to-eight-line decoder can be used for decoding any three-bit code to provide eight outputs, one for each element of the code.

(2M)



(5M)

Three-to-eight-line decoder

MAIN PARTS OF PROCESSOR

- The **processor** contains ALU, contro -circuitry and many registers.
- The processor contains „n" general-purpose registers $\mathbf{R_0}$ through $\mathbf{R_{n-1}}$
- The **IR** holds the instruction that is currently being executed.
- The **control-unit** generates the timing-signals that determine when a given action is to take place
- The **PC** contains the memory-address of the next-instruction to be fetched & executed.
- During the execution of an instruction, the contents of PC are updated to point to next instruction.
- The **MAR** holds the address of the memory-location to be accessed.
- The **MDR** contains the data to be written into or read out of the addressed location.

MAR and MDR facilitates the communication with memory.

(IR → Instruction-Register, PC → Program Counter)

(MAR → Memory Address Register, MDR→ Memory Data Register)

(5M)

STEPS TO EXECUTE AN INSTRUCTION

1) The address of first instruction (to be executed) gets loaded into PC.

2) The contents of PC (i.e. address) are transferred to the MAR & control-unit issues Read signal to memory.

3) After certain amount of elapsed time, the first instruction is read out of memory and placed into MDR.

4) Next, the contents of MDR are transferred to IR. At this point, the instruction can be decoded & executed.

5) To fetch an operand, it's address is placed into MAR & control-unit issues Read signal. As a result, the operand is transferred from memory into MDR, and then it is transferred from MDR to ALU.

6) Likewise required number of operands is fetched into processor.

7) Finally, ALU performs the desired operation.

8) If the result of this operation is to be stored in the memory, then the result is sent to the MDR.

9) The address of the location where the result is to be stored is sent to the MAR and a Write cycle is initiated.

10) At some point during execution, contents of PC are incremented to point to next instruction in the program.

6(a) **Write a note** on addressing mode and explain any four types of addressing modes with suitable examples.

2×5 = 10M

1. **Absolute (Direct) Mode**

- The operand is in a memory-location.
- The address of memory-location is given explicitly in the instruction.
- The absolute mode can represent global variables in the program.
- For example, the instruction

  Move LOC, R2 ;Copy content of memory-location LOC into register R2

2. **Indirect Mode**

- The EA of the operand is the contents of a register(or memory-location).
- The register (or memory-location) that contains the address of an operand is called a **Pointer**
- We denote the indirection by

  → name of the register or

  → new address given in the instruction.

  E.g: *Add (R1),R0* ;The operand is in memory. Register R1 gives the effective-address (B) of the operand. The data is read from location B and added to contents of register R0.

3. **Immediate Mode**

- The operand is given explicitly in the instruction.
- For example, the instruction

  Move #200, R0 ;Place the value 200 in register R0.

- Clearly, the immediate mode is only used to specify the value of a source-operand.

INDEXING AND ARRAYS

- A different kind of flexibility for accessing operands is usefu

4. **Index mode**

- The operation is indicated as X(Ri)

  where X=the constant value which defines an offset(also called a displacement).

  Ri=the name of the index register which contains address of a new location

- The effective-address of the operand is given by EA=X+[Ri]

  The contents of the index-register are not changed in the process of generating the effective-address.

- The constant X may be given either

  → as an explicit number or

  → as a symbolic-name representing a numerical value.

5. **RELATIVE MODE**

- This is similar to index-mode with one difference:

  The effective-address is determined using the PC in

- The operation is indicated as X(PC).

  X(PC) denotes an effective-address of the operand which is X contents of PC.

  Since the addressed-location is identified "relative" to the PC, the name Relative mode is associated with this type of addressing.

- This mode is used commonly in conditional branch instructions.
- An instruction such as

  Branch > 0 LOOP ;Causes program execution to go to the branch target location identified by name LOOP if branch condition is satisfied.

Rachana V Murthy
[illegible]
Course in-charge

[illegible]
Module Coordinator

[illegible]
HOD

K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109

FIRST INTERNAL TEST QUESTION PAPER 2023-24 ODD SEMESTER

SET: B

USN										

Degree :	B.E	Semester :	3rd
Branch - Stream :	CSE - ICB	Course Type / Code :	Integrated \ BCS302
Course Title :	Digital Design and Computer Organization	Date :	02/01/2024
Duration :	1 ½ Hr (90 minutes)	Max Marks :	50

Note: Answer **ONE full** question from each Module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module 1			
1(a)	**Solve** the following function using K-Map and obtain simplified Boolean expressions. (i) **F(A,B,C,D)=Σm(1,2,3,6,8,9,10,12,13,14)** (ii) **F(A,B,C,D)=Σm(7)+d(10,11,12,13,14,15)**	10	CO1	K3
(b)	By **Making use of** the K Map technique solve the following function for SOP and implement it using the basic gates. (i) **F(A,B,C,D)=Σ m(0,1, 2, 4, 5,6, 8,9,10,12,13)**	10	CO1	K3
	OR			
2(a)	By **Making use of** the K Map technique, solve the following function for POS and implement it using the basic gates. (i) **f(a, b, c, d) =** $\prod M(0,3,4,7,8,10,12,14) + d(2,6)$	10	CO1	K3
(b)	**Solve f (a, b, c, d) = Σ m**(0, 1, 3, 5, 6, 7, 11, 12, 14) using K Map to get minimum SOP expression. Implement using **BASIC GATES, NAND only and NOR only gates.**	10	CO1	K3
	Module 2			
3(a)	**Construct** a 3:8 decoder for the following functions (i)**f1 (a, b, c) = ∑ m (0,4,6) , f2(a, b, c) = ∑ m (0,5) , f3(a, b, c) = ∑ m (1,2,3,7)** (ii)**S(x, y, z) = ∑ m (1, 2, 4, 7) , C(x, y, z) = ∑ m (3, 5, 6, 7)**	10	CO2	K3
(b)	By **Making use of** EVM method, simplify the following function and implement it by using 8:1 MUX. Assume D as MEV variable. **f (a, b, c, d) = ∑ m (0,2,3,4,5,8,9,10,11,12,13,15)**	10	CO2	K3
	OR			
4(a)	**Construct** a structural, dataflow and test bench model for the Full Adder Circuit	10	CO2	K3
(b)	**Design** the following Boolean function with a multiplexer (i) **F (x, y, z) = Σ m (1, 2, 6, 7)** (ii) **F (A, B, C, D) = Σ m (1, 3, 4, 11, 12, 13, 14, 15)**	10	CO2	K3
	Module 3			
5(a)	**Explain** the basic operation concept behind working of computers.	10	CO3	K2
	OR			
6(a)	**Write a note** on (i) Basic Performance Equation and (ii) Clock Rate.	10	CO3	K2

Rachana V Murthy Kushal Kumar BN

[illegible] [illegible] [illegible] [illegible]

Name & Signature of Course In charge: Name & Signature of Module Coordinator: HOD Principal

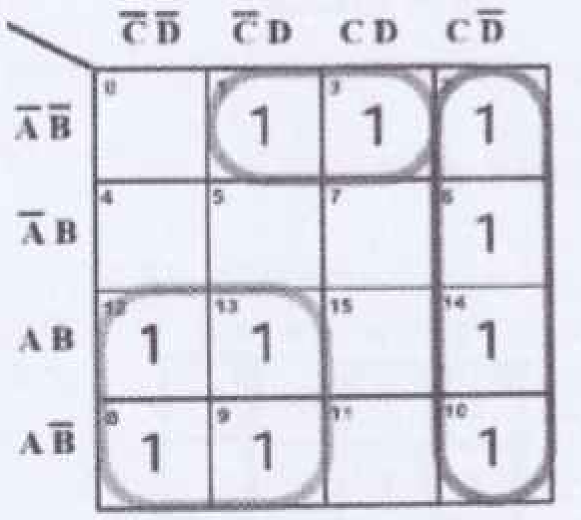
Selected

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
III SESSIONAL TEST QUESTION PAPER 2022 – 23 ODD SEMESTER

Set B

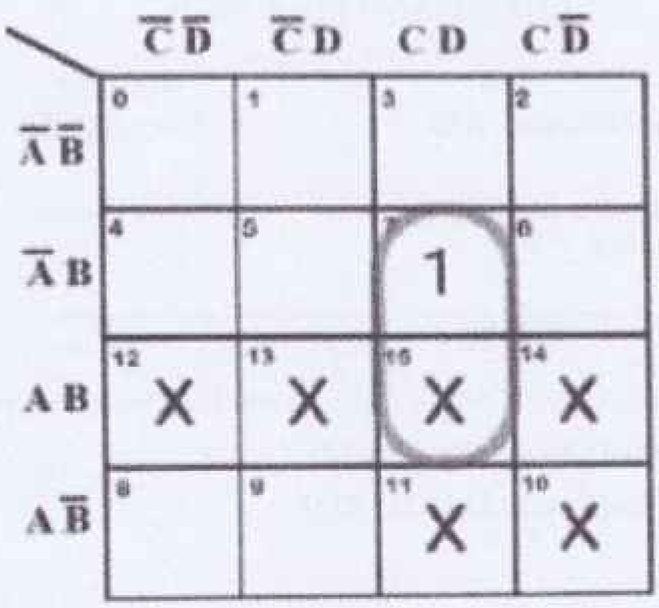
SCHEME AND SOLUTION

Degree	:	B. E	Semester	:	III
Branch	:	Computer Science - ICB	Course Code	:	Integrated \ BCS302
Course Title	:	Digital Design and Computer Organization	Max Marks	:	50

Q.NO.	POINTS	MARKS
1(a)	**Solve** the following function using K-Map and obtain simplified Boolean expressions. (i) F(A,B,C,D)=Σm(1,2,3,6,8,9,10,12,13,14) (ii) F(A,B,C,D)=Σm(7)+d(10,11,12,13,14,15)	5+5=10M
	(i) F(A,B,C,D)=Σ m (1,2,3,6,8,9,10,12,13,14)  $S = \bar{A}\bar{B}D + C\bar{D} + A\bar{C}$	(5M)

(ii) $F(A,B,C,D)=\Sigma m(7)+d(10,11,12,13,14,15)$

(5M)

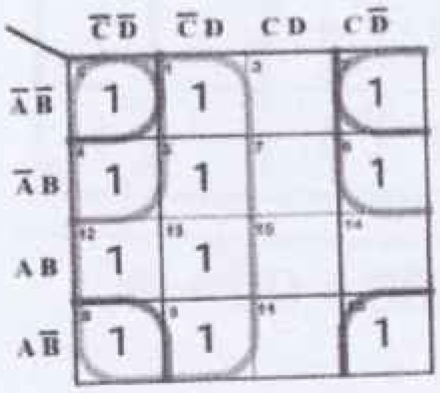


$S = BCD$

1(b) By **Making use of** the K Map technique solve the following function for SOP and implement it using the basic gates.

$F(A,B,C,D)=\Sigma m(0,1,2,4,5,6,8,9,10,12,13)$

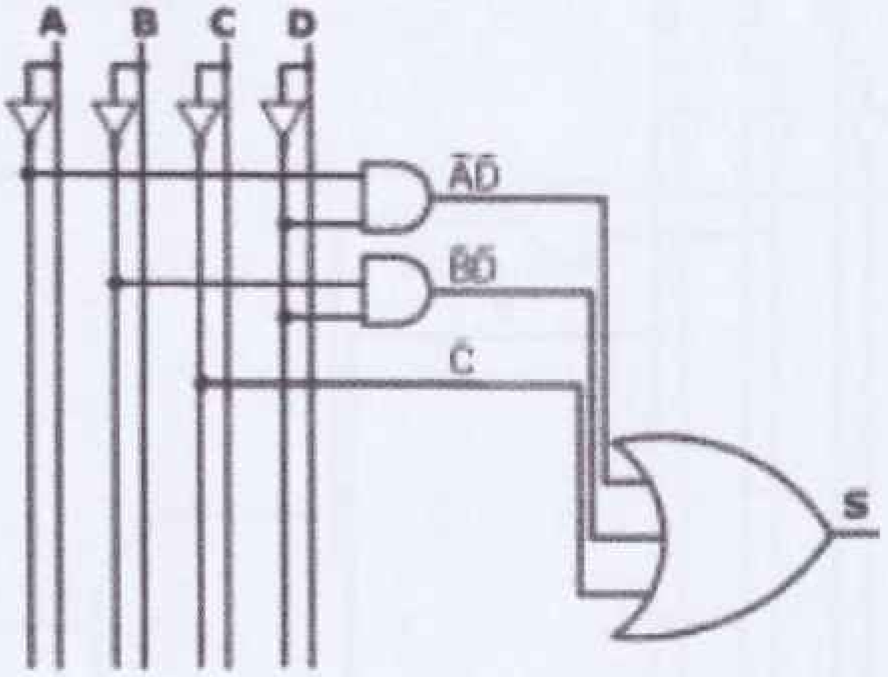
3+1+6 = 10M



(3M)

$S = \bar{A}\bar{D} + \bar{B}\bar{D} + \bar{C}$

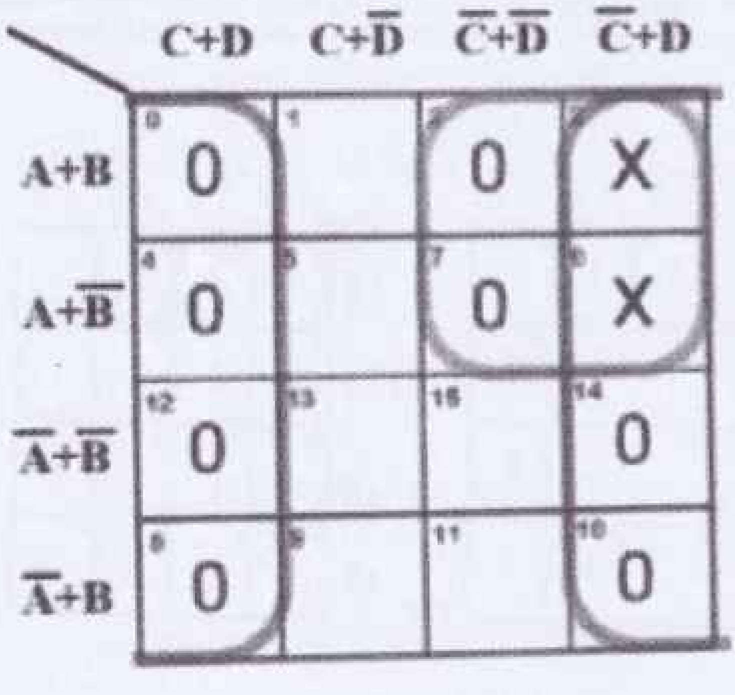
(1M)



(6M)

2(a) By **Making use of** the K Map technique, solve the following function for POS and implement it using the basic gates. 3+1+6 =10M

(i) **f (a, b, c, d)** = $\prod M(0,3,4,7,8,10,12,14) + d(2,6)$

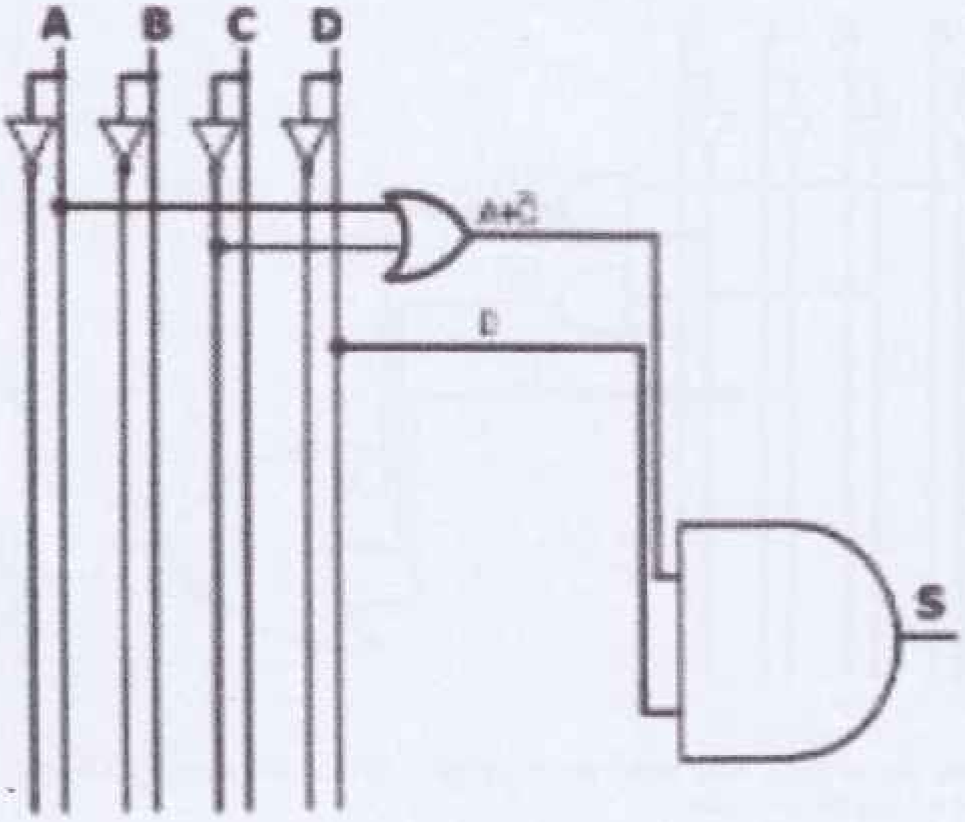


	C+D	C+$\overline{D}$	$\overline{C}$+$\overline{D}$	$\overline{C}$+D
A+B	0 0	1	3 0	2 X
A+$\overline{B}$	4 0	5	7 0	6 X
$\overline{A}$+$\overline{B}$	12 0	13	15	14 0
$\overline{A}$+B	8 0	9	11	10 0

(3M)

(1M)

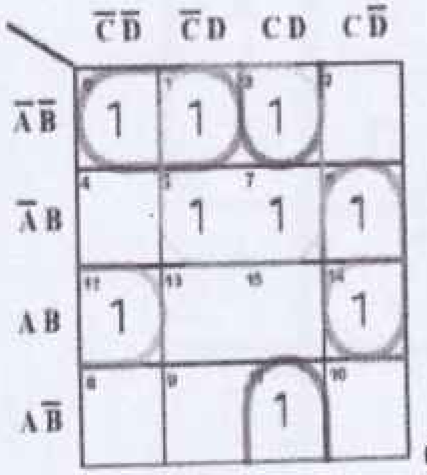
$$S = (A+\bar{C}) \cdot (D)$$



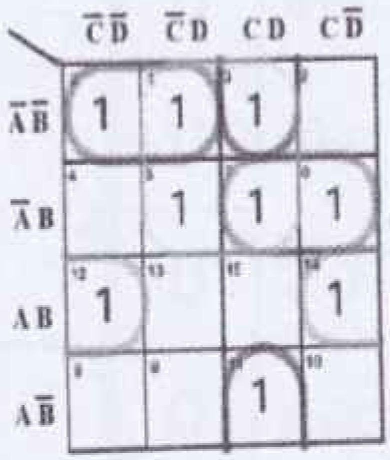
(6M)

2(b) Solve f (a, b, c, d) = Σ m(0, 1, 3, 5, 6, 7, 11, 12, 14) using K Map to get minimum SOP expression. Implement using **BASIC GATES, NAND only and NOR only gates.**

3+1+6
=10M



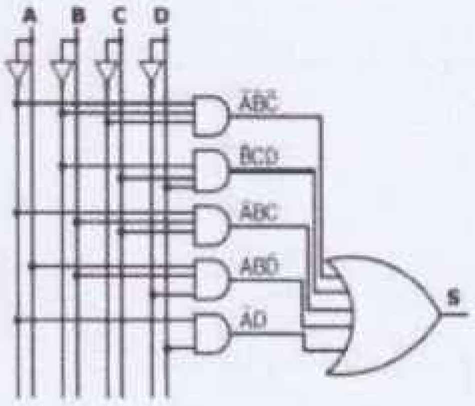
OR

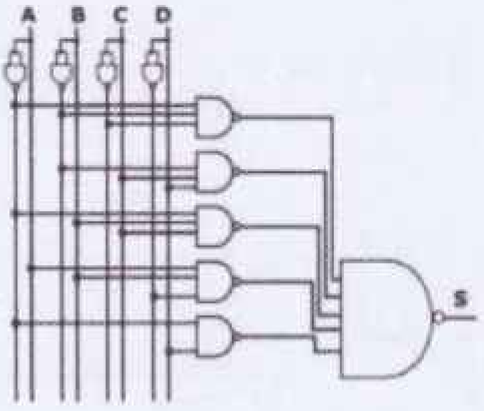


(3M)

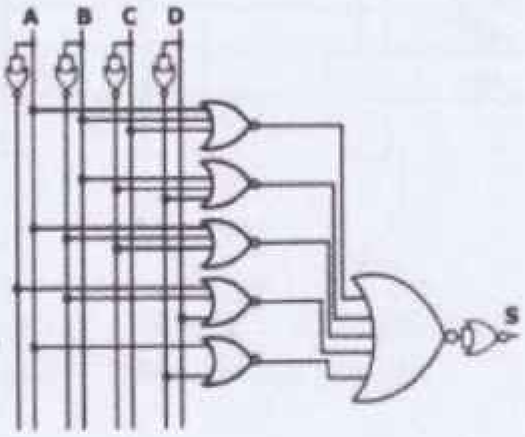
$S=\bar{A}\bar{B}\bar{C}+\bar{B}CD+BC\bar{D}+AB\bar{D}+\bar{A}D$ $S=\bar{A}\bar{B}\bar{C}+\bar{B}CD+\bar{A}BC+AB\bar{D}+\bar{A}D$ (1M)

Circuit diagram for $S=\bar{A}\bar{B}\bar{C}+\bar{B}CD+\bar{A}BC+AB\bar{D}+\bar{A}D$





2M×3 =(6M)



3(a) **Construct** a 3:8 decoder for the following functions

5+5

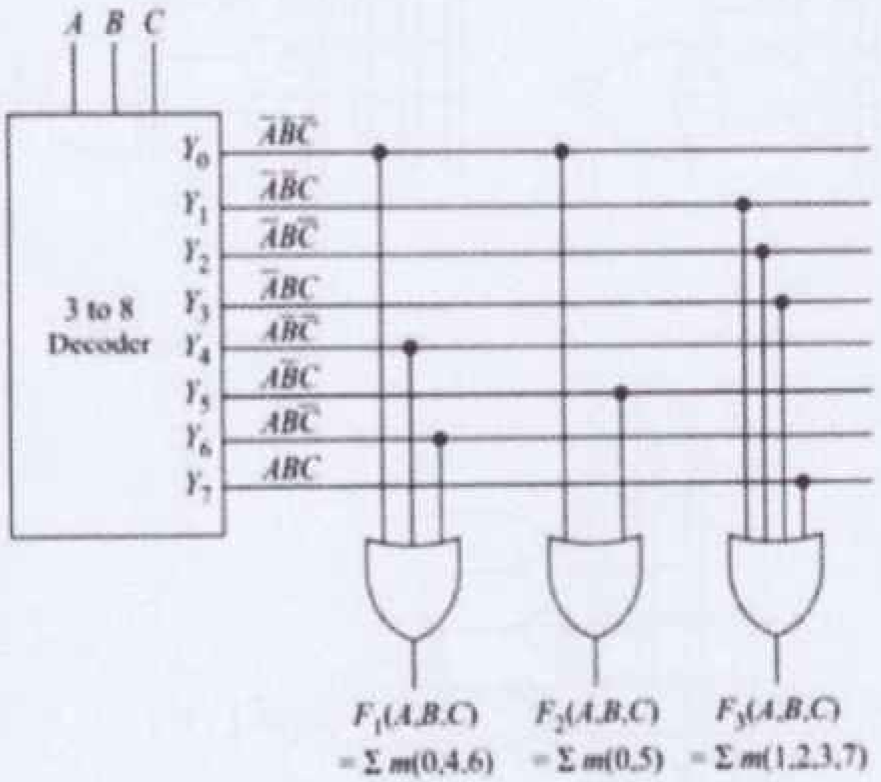
=10M

(i)f1 (a, b, c) = $\sum$ m (0,4,6) , f2(a, b, c) = $\sum$ m (0,5) , f3(a, b, c) = $\sum$ m (1,2,3,7)

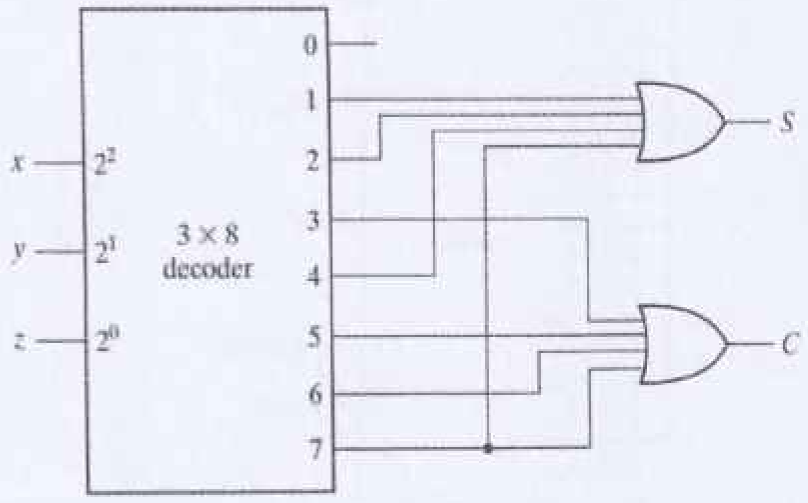
(ii)S(x, y, z) = $\sum$ m (1, 2, 4, 7) , C(x, y, z) = $\sum$ m (3, 5, 6, 7)

(i)f1 (a, b, c) = $\sum$ m (0,4,6) , f2(a, b, c) = $\sum$ m (0,5) , f3(a, b, c) = $\sum$ m (1,2,3,7)

(5M)



A B C
3 to 8 Decoder
Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7
$\bar{A}\bar{B}\bar{C}$
$\bar{A}\bar{B}C$
$\bar{A}B\bar{C}$
$\bar{A}BC$
$A\bar{B}\bar{C}$
$A\bar{B}C$
$AB\bar{C}$
ABC
$F_1(A,B,C) = \Sigma m(0,4,6)$
$F_2(A,B,C) = \Sigma m(0,5)$
$F_3(A,B,C) = \Sigma m(1,2,3,7)$



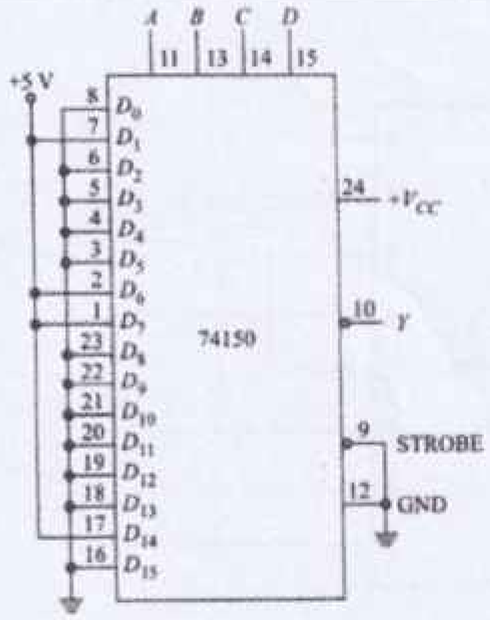
(5M)

3(b) By **Making use of** EVM method, simplify the following function and implement it by using 8:1 MUX. Assume D as MEV variable.
f (a, b, c, d) = ∑ m (0,2,3,4,5,8,9,10,11,12,13,15)

3+3+3+1=10M

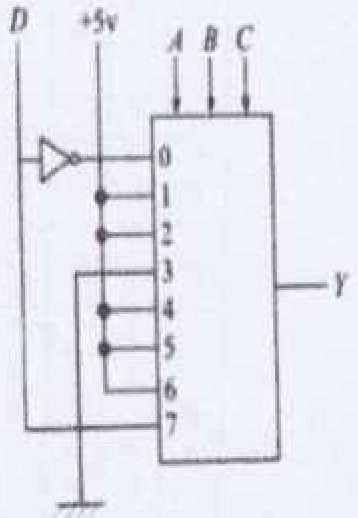
A	B	C	D	Y	8-to-1 MUX Data Inputs
0	0	0	0	1	$\overline{D}$
0	0	0	1	0	
0	0	1	0	1	1
0	0	1	1	1	
0	1	0	0	1	1
0	1	0	1	1	
0	1	1	0	0	0
0	1	1	1	0	
1	0	0	0	1	1
1	0	0	1	1	
1	0	1	0	1	1
1	0	1	1	1	
1	1	0	0	1	1
1	1	0	1	1	
1	1	1	0	0	D

(3M)



(3M)

A	B	C	8-to-1 MUX Data Inputs
0	0	0	$\bar{D}$
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	D



(3M+1M)

4(a) **Construct** a structural, dataflow and test bench model for the Full Adder Circuit

4+2+4 =10M

❖ **STRUCTURAL MODELING**

(4M)

```
module full_adder_s (
   input a, b, cin,
   output sum, carry
);
   wire w1, w2, w3, w4;  // Internal connections
   xor w1(a, b);         // XOR gate for sum
   xor sum(w1, cin);     // Sum output
   and w2(a, b);         // AND gate for carry generation
   and w3(b, cin);
   and w4(cin, a);
   or carry(w2, w3, w4); // OR gate for carry output
endmodule
```

❖ **DATAFLOW MODELING**

(2M)

```
module full_adder_d (
   input a, b, cin,
   output sum, carry
);
   assign sum = a ^ b ^ cin;
   assign carry = (a & b) | (b & cin) | (cin & a);
endmodule
```

❖ **Test Bench**

(4M)

```
module full_adder_tb;
reg a, b, cin;
wire sum, carry;
full_adder_s uut(a, b, cin, sum, carry);
 initial begin
   a = 0; b = 0; cin = 0; #10;
   a = 0; b = 0; cin = 1; #10;
   a = 0; b = 1; cin = 0; #10;
   a = 0; b = 1; cin = 1; #10;
   a = 1; b = 0; cin = 0; #10;
   a = 1; b = 0; cin = 1; #10;
   a = 1; b = 1; cin = 0; #10;
   a = 1; b = 1; cin = 1; #10;
   $finish();
 end
endmodule
```

4(b) **Design** the following Boolean function with a multiplexer

2+3+2+3=10M

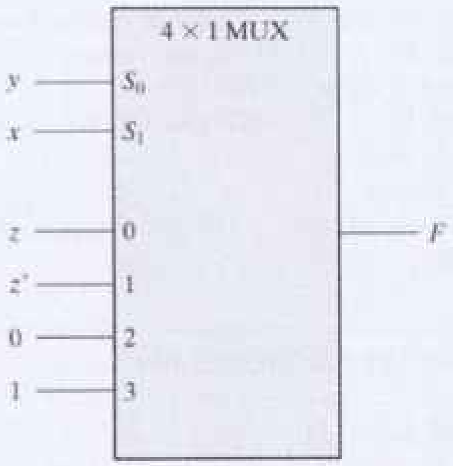
(i) **F (x, y, z) = Σ m (1, 2, 6, 7)**

(ii) **F (A, B, C, D) = Σ m (1, 3, 4, 11, 12, 13, 14, 15)**

(i) **F (x, y, z) = Σ m (1, 2, 6, 7)**

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table

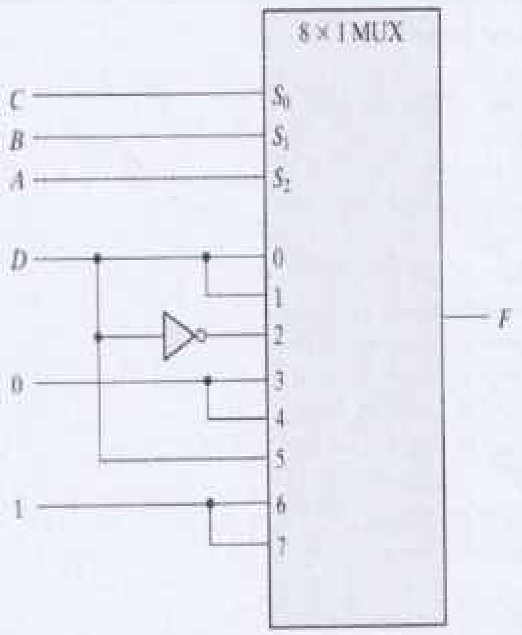


(b) Multiplexer implementation

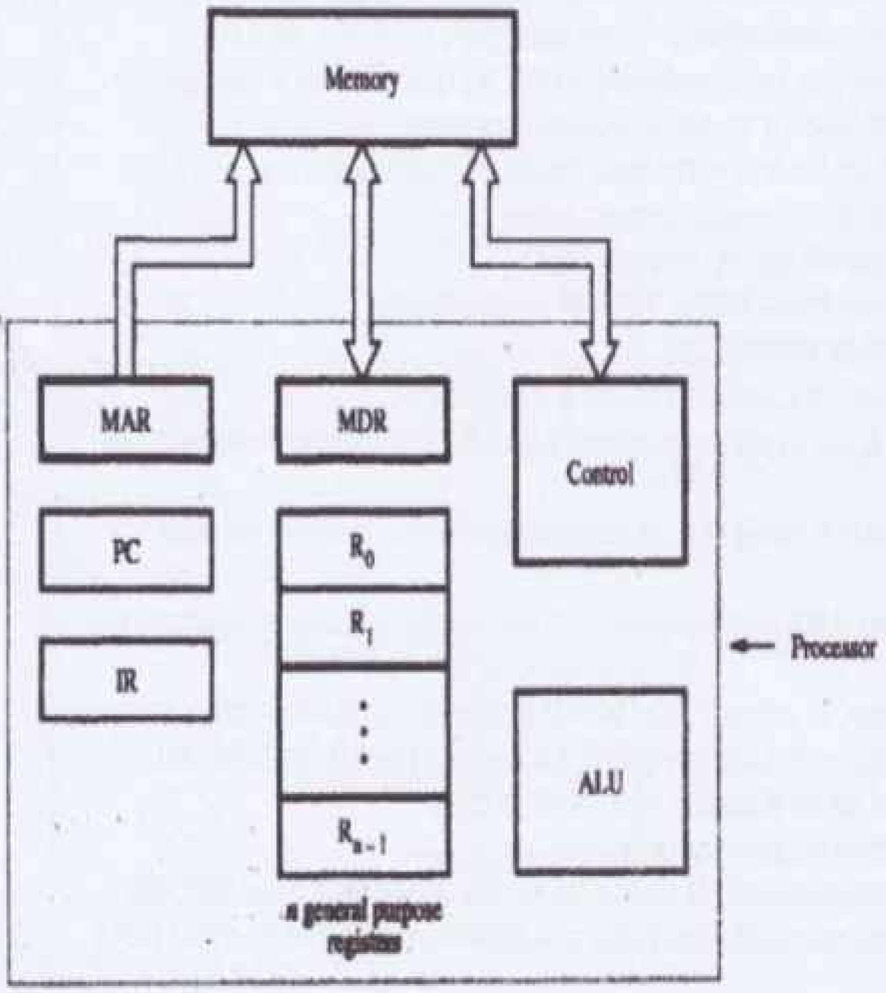
(2+3M)

(ii) **F (A, B, C, D) = Σ m (1, 3, 4, 11, 12, 13, 14, 15)**

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



(2M+3M)

5(a)	**Explain** the basic operation concept behind working of computers.  Memory MAR MDR Control PC R_0 R_1 IR Processor ALU R_{n-1} n general purpose registers	5+5= 10M (5M)

MAIN PARTS OF PROCESSOR

- The **processor** contains ALU, contro -circuitry and many registers.
- The processor contains „n" general-purpose registers **R_0** through **R_{n-1}**
- The **IR** holds the instruction that is currently being executed.
- The **control-unit** generates the timing-signals that determine when a given action is to take place
- The **PC** contains the memory-address of the next-instruction to be fetched & executed.
- During the execution of an instruction, the contents of PC are updated to point to next instruction.
- The **MAR** holds the address of the memory-location to be accessed.
- The **MDR** contains the data to be written into or read out of the addressed location.

MAR and MDR facilitates the communication with memory.

(IR → Instruction-Register, PC → Program Counter)

(MAR → Memory Address Register, MDR→ Memory Data Register)

(5M)

STEPS TO EXECUTE AN INSTRUCTION

1) The address of first instruction (to be executed) gets loaded into PC.

2) The contents of PC (i.e. address) are transferred to the MAR & control-unit issues Read signal to memory.

3) After certain amount of elapsed time, the first instruction is read out of memory and placed into MDR.

4) Next, the contents of MDR are transferred to IR. At this point, the instruction can be decoded & executed.

5) To fetch an operand, it's address is placed into MAR & control-unit issues Read signal. As a result, the operand is transferred from memory into MDR, and then it is transferred from MDR to ALU.

6) Likewise required number of operands is fetched into processor.

7) Finally, ALU performs the desired operation.

8) If the result of this operation is to be stored in the memory, then the result is sent to the MDR.

9) The address of the location where the result is to be stored is sent to the MAR and a Write cycle is initiated.

10) At some point during execution, contents of PC are incremented to point to next instruction in the program.

6(a) **Write a note** on (i) Basic Performance Equation and (ii) Clock Rate. 4+1+3+2 =10M

BASIC PERFORMANCE EQUATION

Let T = Processor time required to executed a program

N = Actual number of instruction executions.

S = Average number of basic steps needed to execute one machine instruction (4M)

R = Clock rate in cycles per second.

• The program execution time is given by

$$T = \frac{N \times S}{R}$$ (1M)

Equ1 is referred to as the basic performance equation

To achieve high performance, the computer designer must reduce the value of T, which reducing N and S, and increasing R.

- The value of N is reduced if source program is compiled into fewer machine instructions
- The value of S is reduced if instructions have a smaller number of basic steps to perform
- The value of R can be increased by using a higher frequency clock.

• Care has to be taken while modifying values since changes in one parameter may affect the other

CLOCK RATE

• There are 2 possibilities for increasing the clock rate R:

1) Improving the IC technology makes logic-circuits faster.

This reduces the time needed to compute a basic step. (IC → integrated circuits) (3M)

This allows the clock period P to be reduced and the clock rate R to be increased

2) Reducing the amount of processing done in one basic step also reduces the clock period P.

In presence of a cache, the percentage of accesses to the main-memory is small.

Hence, much of performance-gain expected from the use of faster technology can be realized. (2M)

The value of T will be reduced by same factor as R is increased „" S & N are not affected

Rachana V Murthy.
[illegible]
Course in-charge

[illegible]
Module Coordinator

[illegible]
HOD

K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109

SECOND INTERNAL TEST QUESTION PAPER 2023-24 ODD SEMESTER

SET: A

Degree :	B.E	USN	
Branch - Stream :	CSE - ICB	Semester :	3rd
Course Title :	Digital Design and Computer Organization	Course Type / Code :	Integrated/BCS302
		Date :	05/02/2024
Duration :	1 Hr (60 minutes)	Max Marks :	25

Note: Answer **ONE full** question from each Module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module 3			
1(a)	**Make Use Of** appropriate equations to explain the following (i) Bus Structure. (ii) Processor Clock. (iii) Basic Performance Equation. (iv) Performance Measurement.	10	CO3	K3
	OR			
2(a)	**Identify** all the addressing modes and explain with an example.	10	CO3	K3
	Module 4			
3(a)	**Make Use Of** Memory addressing mode using Branching technique with an example.	10	CO4	K3
3(b)	**Explain** Accessing of I/O Devices.	5	CO4	K2
	OR			
4(a)	**Construct** the Mechanism used for Interfacing I/O Devices.	10	CO4	K3
4(b)	**Explain** Polling and Vectored Interrupt.	5	CO4	K2

[illegible]
Rachana V Murthy
Name & Signature of Course In charge

[illegible]
Kushal Kumar B N
Name & Signature of Module Coordinator

[illegible]
HOD

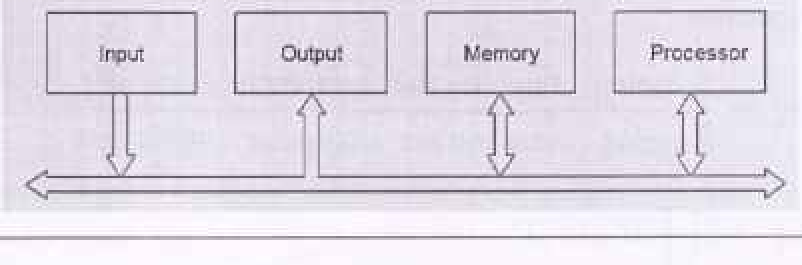
[illegible]
Principal
[illegible]

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
II SESSIONAL TEST QUESTION PAPER 2022 – 23 ODD SEMESTER

Set A

SCHEME AND SOLUTION

Degree	: **B. E**	**Semester**	: **III**
Branch	: **Computer Science - ICB**	**Course Code**	: **Integrated/ BCS302**
Course Title	: **Digital Design and Computer Organization**	**Max Marks**	: **25**

Q.NO.	*POINTS*	*MARKS*
1(a)	By **Making Use Of** appropriate equations explain the following (i) Bus Structure. (ii) Processor Clock. (iii) Basic Performance Equation. (iv) Performance Measurement.	
(i)	**Bus Structure.** **BUS STRUCTURE** ➢ A *bus* is a group of lines that serves as a connecting path for several devices. ➢ Bus must have lines for data transfer, address & control purposes. ➢ Because the bus can be used for only one transfer at a time, only 2 units can actively use the bus at any given time. ➢ Bus control lines are used to arbitrate multiple requests for use of the bus. ➢ Main advantage of single bus: Low cost and flexibility for attaching peripheral devices. ➢ Systems that contain multiple buses achieve more concurrency in operations by allowing 2 or more transfers to be carried out at the same time. Advantage: better performance. Disadvantage: increased cost. ➢ The devices connected to a bus vary widely in their speed of operation. To synchronize their operational speed, the approach is to include buffer registers with the devices to hold the information during transfers. ➢ Buffer registers prevent a high-speed processor from being locked to a slow I/O device during a sequence of data transfers. 	(3M)

(ii) Processor Clock.

PROCESSOR CLOCK

- Processor circuits are controlled by a timing signal called a clock.
- The clock defines regular time intervals called *clock cycles*.
- To execute a machine instruction, the processor divides the action to be performed into a sequence of basic steps such that each step can be completed in one clock cycle.
- Let P=length of one clock cycle R=clock rate. Relation between **P** and **R** is given by

$$R=\frac{1}{P}$$

This is measured in cycles per second.

- Cycles per second is also called hertz(Hz)

(2M)

(iii) Basic Performance Equation.

T – processor time required to execute a program that has been prepared in high-level language

N – number of actual machine language instructions needed to complete the execution (note: loop)

S – average number of basic steps needed to execute one machine instruction. Each step completes in one clock cycle

R – clock rate

Note: these are not independent to each other

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

(2M)

(iv) Performance Measurement.

- T is difficult to compute.
- Measure computer performance using benchmark programs.
- System Performance Evaluation Corporation (SPEC) selects and publishes representative application programs for different application domains, together with test results for many commercially available computers.
- Compile and run (no simulation)
- Reference computer

$$SPEC\ \ rating\ =\frac{\text{Running time on the reference computer}}{\text{Running time on the computer under test}}$$

$$SPEC\ \ rating\ =(\prod_{i=1}^{n} SPEC_{\ i})^{\frac{1}{n}}$$

(3M)

(10M)

2(a) **Identify** all the addressing modes and explain with an example.

Name	Assem bler syn tax	Addressing function
Immediate	#Value	Op erand = Value
Register	R *i*	EA = R*i*
Absolute (Direct)	LOC	EA = LOC

Indirect	(R *i*)
	(LOC)
		EA = [R*i*]
		EA = [LOC]

Index	X(R *i*)	EA = [R*i*] + X
Base with index	(R *i*,R*j*)	EA = [R*i*] + [R*j*]
Base with index and offset	X(R *i*,R*j*)	EA = [R*i*] + [R*j*] + X .
Relative	X(PC)	EA = [PC] + X
Autoincremen t	(R *i*)+	EA = [R*i*] ; Incremen t R *i*
Autodecrement	−(R *i*)	Decremen t R *i* ; EA = [R*i*]

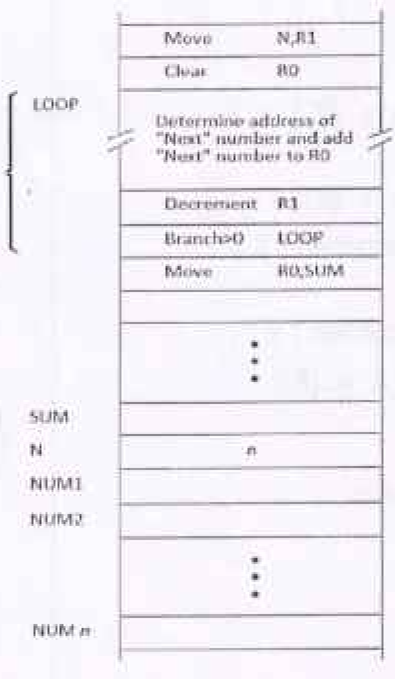
(10M)

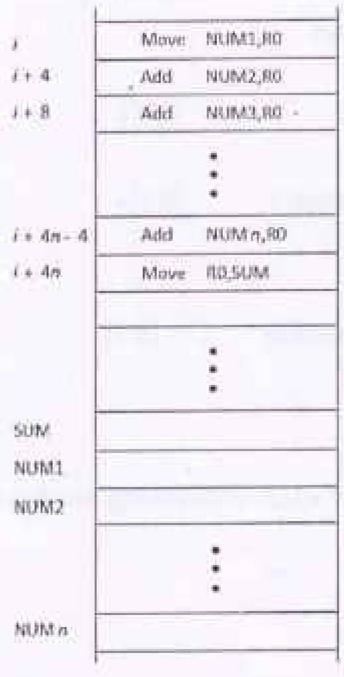
Explanation of each addressing modes with examples

3(a) **Make Use Of** Memory addressing mode using Branching technique with an example.

BRANCHING

- Consider the task of adding a list of n numbers (Figure 2.10).
- The loop is a straight line sequence of instructions executed as many times as needed. It starts at location LOOP and ends at the instruction Branch > 0.
- During each pass through this loop, the address of the next list entry is determined, and that entry is fetched and added to R0.
- Register R1 is used as a counter to determine the number of times the loop is executed. Hence, the contents of location N are loaded into register R1 at the beginning of the program.
- Within the body of the loop, the instruction *Decrement R1* reduces the contents of R1 by 1 each time through the loop.
- Then Branch instruction loads a new value into the program counter. As a result, the processor fetches and executes the instruction at this new address called the branch target.'
- A conditional branch instruction causes a branch only if a specified condition is satisfied. If the condition is not satisfied, the PC is incremented in the normal way, and the next instruction in sequential address order is fetched and executed.





(10M)

3(b) **Explain** Accessing of I/O Devices.

ACCESSING I/O DEVICES

There are 2 ways to deal with I/O devices (Figure 4.1).

1) Memory mapped I/O

When I/O devices and memory share the same address space, the arrangement is called *memory mapped I/O*.

- Memory and I/O devices share a common address-space.
- Any data-transfer instruction (like Move, Load) can be used to exchange information.

For example,

Move DATAIN, R0;

This instruction reads data from DATAIN (input-buffer associated with keyboard) & stores them into processor-register R0.

Advantage:

1.No need of separate address space.
2.No special instructions are needed.
3.Easy to implement.

Disadvantage:

Slow compare to other technique.

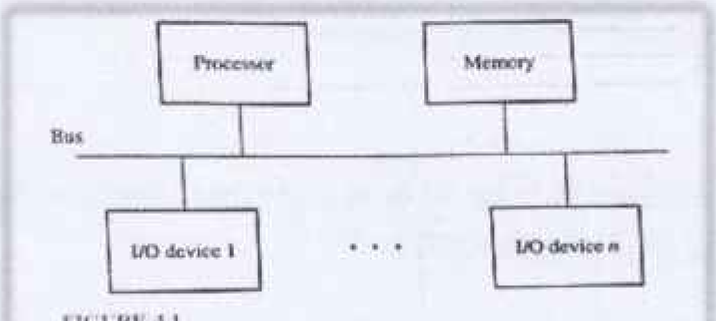


FIGURE 4.1
A single bus structure.

(5M)

2) In *I/O mapped I/O*

- Memory and I/O address-spaces are different.
- A special instruction named IN and OUT is used for data transfer.

Advantage:

I/O devices deal with fewer address-lines.

1. Faster compare to other technique.
2. Dedicated instructions for I/O operation.

Disadvantage:

Complex to design

4(a) **Construct** the Mechanism used for Interfacing I/O Devices.

MECHANISMS USED FOR INTERFACING I/O DEVICES

1) Program Controlled I/O

- Processor repeatedly checks a status-flag to achieve required synchronization between processor & input/output device. (We say that the processor polls the device).
- Main drawback: The processor wastes its time in checking the status of the device before actual data transfer takes place.

In figure 46 the status register are explained based on the program in figure 48. The information needed to determine whether a device is requesting an interrupt is available in its status register. When a device raises an interrupt request, it sets to 1 one of the bits in its status register, which we will call the IRQ bit. For example, bits KIRQ and DIRQ are the interrupt request bits for the keyboard and the display, respectively. The simplest way to identify the interrupting device is to have the interrupt-service routine poll all the I/O devices connected to the bus. The first device encountered with its IRQ bit set is the device that should be serviced. An appropriate subroutine is called to provide the requested service. 4

2) Interrupt I/O

- Synchronization is achieved by having I/O device send a special signal over bus whenever it is ready for a data transfer operation. 2

3) Direct Memory Access (DMA)

- This involves having the device-interface transfer data directly to or from the memory without continuous involvement by the processor. 2

Naming (i) Program controlled I/o
(ii) Interrupt I/o
(iii) Direct Memory Access 2

(10M)

4(b) Explain Polling and Vectored Interrupt.

HANDLING MULTIPLE DEVICES

The interrupt from multiple devices are handled by the

1. Polling
2. Vectored Interrupt

Polling

- ➢ Information needed to determine whether a device is requesting an interrupt is available in its status-register.
- ➢ When a device raises an interrupt-request, it sets IRQ bit to 1 in its status-register (Figure 4.3).
- ➢ KIRQ and DIRQ are the interrupt-request bits for keyboard & display.
- ➢ Simplest way to identify interrupting device is to have ISR poll all I/O devices connected to bus.
- ➢ The first device encountered with its IRQ bit set is the device that should be serviced. After servicing this device, next requests may be serviced.

Advantage:

1. Simple
2. Easy to implement.

Disadvantage:

More time spent polling IRQ bits of all devices (that may not be requesting any service).

(5M)

Vectored Interrupts

- ➢ In the program shown in figure 4.4, a device requesting an interrupt identifies itself by sending a special-code to processor over bus. (This enables processor to identify individual devices even if they share a single interrupt-request line).
- ➢ The code represents starting-address of ISR for that device.
- ➢ ISR for a given device must always start at same location.
- ➢ The address stored at the location pointed to by interrupting-device is called the interrupt-vector.
- ➢ Processor
  → loads interrupt-vector into PC &
  → executes appropriate ISR

[illegible] [illegible] [illegible]

Course in-charge   Module Coordinator   HOD

K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109
SECOND INTERNAL TEST QUESTION PAPER 2023-24 ODD SEMESTER

SET: B

USN										

Degree	:	B.E	Semester	:	3rd
Branch - Stream	:	CSE - ICB	Course Type / Code	:	Integrated/BCS302
Course Title	:	Digital Design and Computer Organization	Date	:	05/02/2024
Duration	:	1 Hr (60 minutes)	Max Marks	:	25

Note: Answer **ONE full** question from each Module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module 3			
1(a)	**Make Use Of** appropriate diagrams to explain the following: a. Byte addressability. b. Big-Endian assignment. c. Little-Endian assignment. d. Instruction and Instruction Sequencing.	10	CO3	K3
	OR			
2(a)	**Identify** addressing modes. Explain each with examples.	10	CO3	K3
	Module 4			
3(a)	**Make Use Of** memory address mode to explain Instruction Execution and Straight-Line Sequencing.	10	CO4	K3
3(b)	**Explain** I/O Interface for an Input Device.	5	CO4	K2
	OR			
4(a)	**Construct** Interrupt Hardware technique.	10	CO4	K3
4(b)	**Illustrate** a Program that reads one line from the keyboard and sends it to the display.	5	CO4	K2

[illegible]	[illegible]	[illegible]	[illegible]
Rachana V Murthy	Kushal Kumar B N	**HOD**	**Principal**
Name & Signature of Course In charge	**Name & Signature of Module Coordinator**		

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

II SESSIONAL TEST QUESTION PAPER 2022 – 23 ODD SEMESTER

Set B

SCHEME AND SOLUTION

Degree	: **B. E**	**Semester**	: **III**
Branch	: **Computer Science - ICB**	**Course Code**	: **Intégrated/ BCS302**
Course Title	: **Digital Design and Computer Organization**	**Max Marks**	: **25**

Q.NO.	*POINTS*	*MARKS*
1(a)	**Make Use Of** appropriate diagrams to explain the following: a. Byte addressability. b. Big-Endian assignment. c. Little-Endian assignment. d. Instruction and Instruction Sequencing.	
(a)	Byte addressability. **BYTE ADDRESSABILITY** ➢ In byte addressable memory, successive addresses refer to successive byte locations in the memory. ➢ Byte locations have addresses 0, 1, 2. . . . . ➢ If the word length is 32 bits, successive words are located at addresses 0, 4, 8. .with each word having 4 bytes.	
(b)	Big-Endian assignment.	
(c)	Little-Endian assignment. **BIG-ENDIAN AND LITTLE-ENDIAN ASSIGNMENTS** There are two ways in which byte addresses are arranged. 1. **Big-endian assignment**: lower byte addresses are used for the more significant bytes of the word (Figure 2.3). 2. **Little-endian**: lower byte addresses are used for the less significant bytes of the word In both cases, byte addresses 0, 4, 8. . . . . are taken as the addresses of successive words in the memory.	

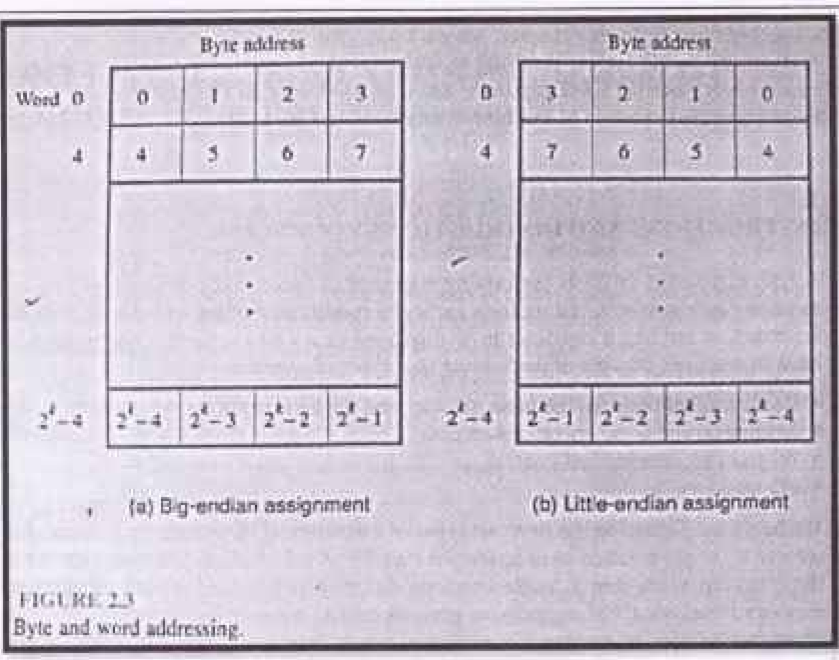


FIGURE 2.3
Byte and word addressing.

(d) Instruction and Instruction Sequencing.

REGISTER TRANSFER NOTATION (RTN)

We identify a memory location by a symbolic name (in uppercase alphabets). For example, LOC, PLACE, NUM etc indicate memory locations.R0, R5 etc indicate processor register. DATAIN, OUTSTATUS etc indicate I/O registers.

Example,

R ← [LOC]

Means that the contents of memory location LOC are transferred into processor register R1 (The contents of a location are denoted by placing [ ] square brackets around the name of the location).

R3← [R1] + [R2]

Indicates the operation that adds the contents of registers R1 and R2 ,and then places their sum into register R3.

- This type of notation is known as **RTN** (Register Transfer Notation).

Move LOC, R1;

This instruction transfers data from memory-location LOC to processor-register R1. The contents of LOC are unchanged by the execution of this instruction, but the old contents of register R1 are overwritten.

Add R1, R2, R3;

This instruction adds 2 numbers contained in processor-registers R1 and R2, and places their sum in R3.

ASSEMBLY LANGUAGE NOTATION

To represent machine instructions and programs, assembly language format can be used.

Example,

10

2(a) **Identify** addressing modes. Explain each with examples.

Name	Assem bler syn tax	Addressing function
Immediate	#Value	Op erand = Value
Register	R *i*	EA = R*i*
Absolute (Direct)	LOC	EA = LOC

Indirect	(R *i*)
	(LOC)
		EA = [R*i*]
		EA = [LOC]

Index	X(R *i*)	EA = [R*i*] + X
Base with index	(R *i*,R*j*)	EA = [R*i*] + [R*j*]
Base with index and offset	X(R *i*,R*j*)	EA = [R*i*] + [R*j*] + X
Relative	X(PC)	EA = [PC] + X
Autoincremen t	(R *i*)+	EA = [R*i*] ; Incremen t R *i*
Autodecrement	−(R *i*)	Decremen t R *i* ; EA = [R*i*]

Explanation of each addressing modes with examples

10

3(a) **Make Use Of** memory address mode to explain Instruction Execution and Straight-Line Sequencing.

INSTRUCTION EXECUTION & STRAIGHT LINE SEQUENCING

The program is executed as follows:

1. Initially, the address of the first instruction is loaded into PC (Program counter is a register which holds the address of the next instruction to be executed)
2. Then, the processor control circuits use the information in the PC to fetch and execute instructions, one at a time, in the order of increasing addresses. This is called *straight-line sequencing* (Figure 2.4)
3. During the execution of each instruction, the PC is incremented by 4 to point to the next instruction.
4. Executing given instruction is a two-phase procedure.
   - i. In fetch phase, the instruction is fetched from the memory location (whose address is in the PC) and placed in the IR of the processor
   - ii. In execute phase, the contents of IR is examined to determine which operation is to be performed. The specified operation is then performed by the processor.

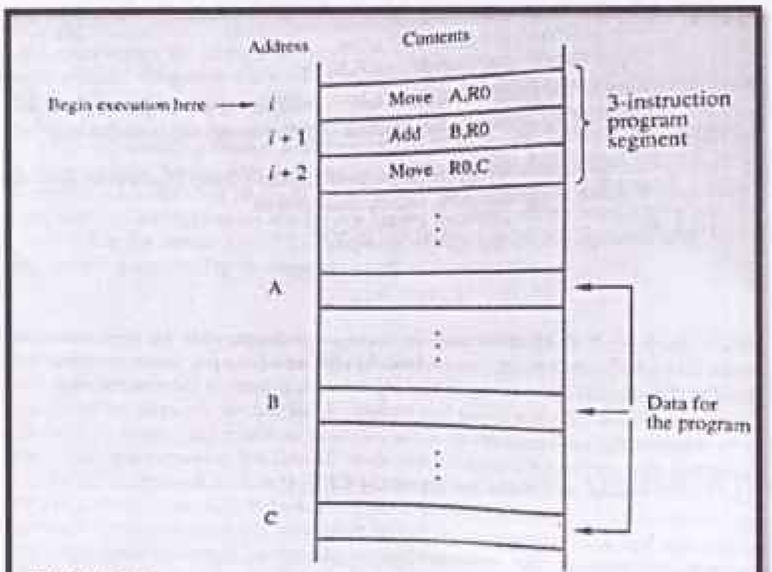


FIGURE 2.4
A program for C ← [A] + [B].

10

3(b) **Explain** I/O Interface for an Input Device.

I/O Interface for an Input Device

- *Address decoder:* decodes address sent on bus, so as to enable input-device (Figure 4.2).
- *Data register:* holds data being transferred to or from the processor.
- *Status register:* contains information relevant to operation of I/O device.
- Address decoder, data- and status-registers, and control-circuitry required to coordinate I/O transfers constitute device's interface-circuit.

5

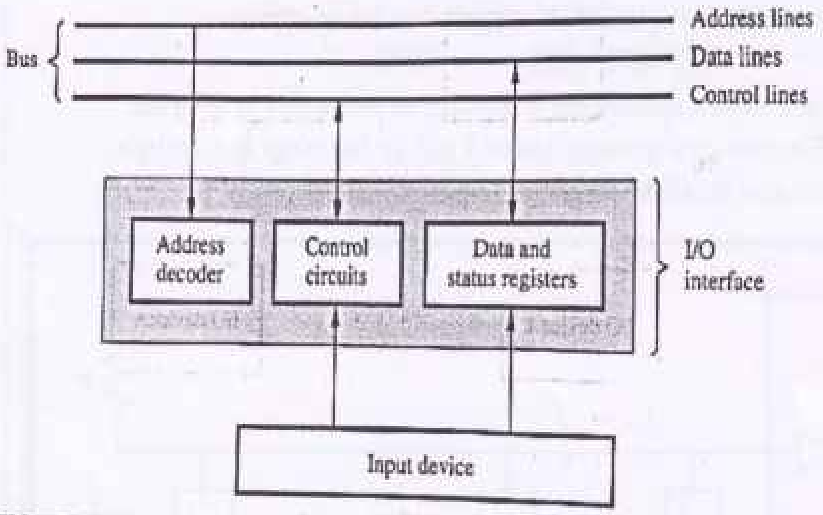


FIGURE 4.2
I/O interface for an input device.

4(a) **Construct** Interrupt Hardware technique.

INTERRUPT HARDWARE

- An I/O device requests an interrupt by activating a bus-line called interrupt-request (IR).
- A single IR line can be used to serve "n" devices (Figure 4.6).
- All devices are connected to IR line via switches to ground.
- To request an interrupt, a device closes its associated switch. Thus, if all IR signals are inactive (i.e. if all switches are open), the voltage on the IR line will be equal to V_{dd}.
- When a device requests an interrupt by closing its switch, the voltage on the line drops to 0, causing the INTR received by the processor to goto 1.
- The value of INTR is the logical OR of the requests from individual devices

$$INTR=INTR_1+INTR_2+\ldots\ldots+INTR_n$$

- A special gate known as open-collector or open-drain are used to drive the INTR line.
- Resistor R is called a *pull-up resistor* because it pulls the line voltage up to the high-voltage state when the switches are open.

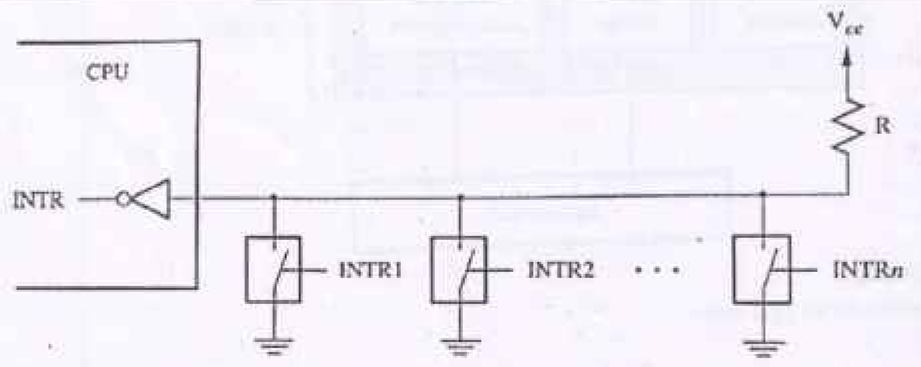
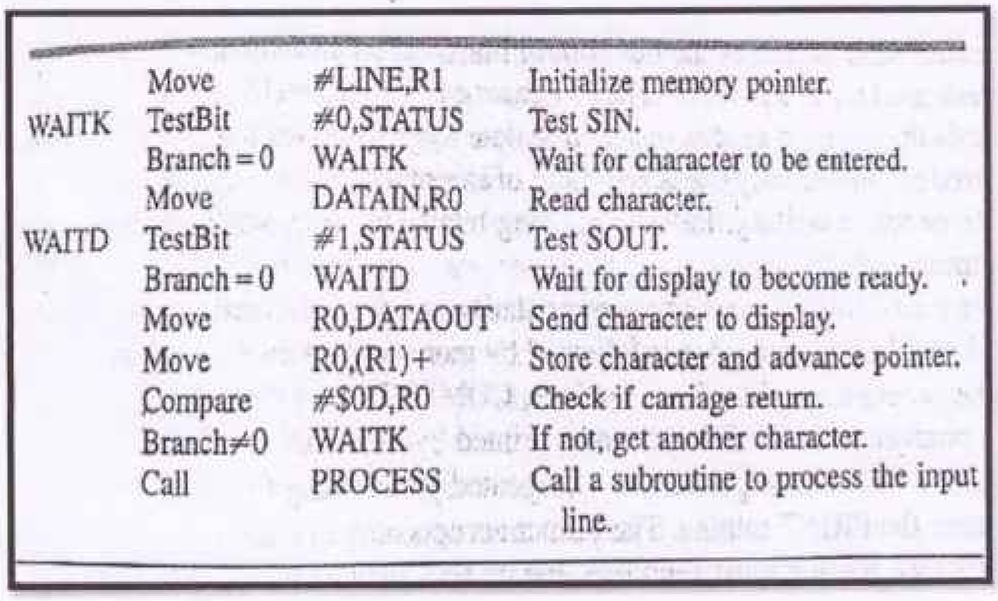


FIGURE 4.6
An equivalent circuit for an open-collector bus used to implement a common interrupt-request line.

10

4(b) **Illustrate** a Program that reads one line from the keyboard and sends it to the display.



```
          Move        #LINE,R1      Initialize memory pointer.
WAITK     TestBit     #0,STATUS     Test SIN.
          Branch=0    WAITK         Wait for character to be entered.
          Move        DATAIN,R0     Read character.
WAITD     TestBit     #1,STATUS     Test SOUT.
          Branch=0    WAITD         Wait for display to become ready.
          Move        R0,DATAOUT    Send character to display.
          Move        R0,(R1)+      Store character and advance pointer.
          Compare     #$0D,R0       Check if carriage return.
          Branch≠0    WAITK         If not, get another character.
          Call        PROCESS       Call a subroutine to process the input
                                       line.
```

5

[illegible]
Course in-charge

[illegible]
Module Coordinator

[illegible]
HOD

K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109

THIRD INTERNAL TEST QUESTION PAPER 2023-24 ODD SEMESTER

SET: B

USN

Degree	:	B.E	Semester	:	3RD
Branch - Stream	:	CSE – ICB	Course Type / Code	:	IPCC/BCS302
Course Title	:	DDCO	Date	:	04-03-2024
Duration	:	1 Hr (60 minutes)	Max Marks	:	25

Note: Answer **ONE full** question from each Module.

K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module 4			
1(a)	Make use of a block diagram explain Centralized Arbitration in detail.	5	CO4	K3
	OR			
2(a)	Make use of Mapping Techniques explain in detail the Direct Mapping briefly.	5	CO4	K3
	Module 5			
3(a)	Make use of MOVE (R_s), R_d instruction write the complete control sequence and brief about performing an arithmetic or logic operation.	10	CO5	K3
(b)	Explain the Role of Cache Memory along with 4 Stage Pipeline block diagram.	10	CO5	K2
	OR			
4(a)	Make use of block diagram and appropriate timing diagram explain how to fetch a WORD from MEMORY.	10	CO5	K3
(b)	Explain Execution of A COMPLETE INSTRUCTION briefing all the 7 Steps.	10	CO5	K2

[illegible]
Name & Signature of Course In charge:
Rachana V Murthy

[illegible]
Name & Signature of Module Coordinator:
KUSHAL KUMAR B N

[illegible]
HOD

[illegible]
Principal
Scrutinized

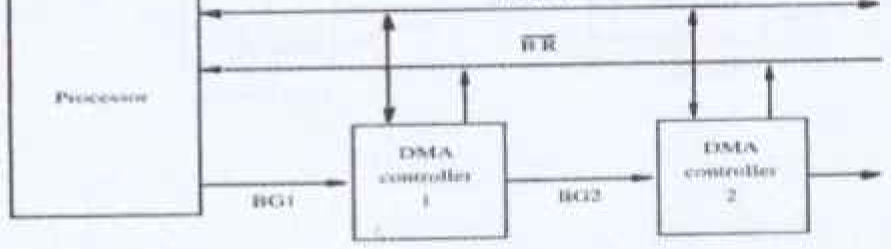
K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

III SESSIONAL TEST QUESTION PAPER 2023 – 24

ODD SEMESTER

Set B **SCHEME AND SOLUTION**

Degree	**: B. E**	**Semester**	**: III**
Branch	**: Computer Science - ICB**	**Course Code**	**: Integrated/ BCS302**
Course Title	**: Digital Design and Computer Organization**	**Max Marks**	**: 25**

Q.NO.	*POINTS*	*MARKS*
1(a)	Make use of a block diagram explain Centralized Arbitration in detail. **Centralized Arbitration**:- The bus arbiter may be the processor or a separate unit connected to the bus. A basic arrangement in which the processor contains the bus arbitration circuitry. In this case, the processor is normally the bus master unless it grants bus mastership to one of the DMA controllers. A DMA controller indicates that it needs to become the bus master by activating the Bus-Request line, BR . The signal on the Bus-Request line is the logical OR of the bus requests from all the devices connected to it. When Bus-Request is activated, the processor activates the Bus-Grant signal, BG1, indicating to the DMA controllers that they may use the bus when it becomes free. This signal is connected to all DMA controllers using a daisy-chain arrangement. Thus, if DMA controller 1 is requesting the bus, it blocks the propagation of the grant signal to other devices. Otherwise, it passes the grant downstream by asserting BG2. The current bus master indicates to all device that it is using the bus by activating another open-controller line called Bus-Busy, BBSY . Hence, after receiving the Bus-Grant signal, a DMA controller waits for Bus-Busy to become inactive, then assumes mastership of the bus. Centralized Bus Arbitration  $\overline{BBSY}$ $\overline{BR}$ Processor DMA controller 1 DMA controller 2 BG1 BG2	(2+3) 2→Exp 3→Diag Total 5M

2(a) Make use of Mapping Techniques explain in detail the Direct Mapping briefly.

Direct Mapping Technique The cache systems are divided into three categories, to implement cache system. As shown in figure, the lower order 4-bits from 16 words in a block constitute a word field. The second field is known as block field used to distinguish a block from other blocks. Its length is 7-bits, when a new block enters the cache, the 7-bit cache block field determines the cache position in which this block must be stored. The third field is a Tag field, used to store higher order 5-bits of the memory address of the block, and to identify which of the 32blocks are mapped into the cache.

Tag	Block	Word
5	7	4

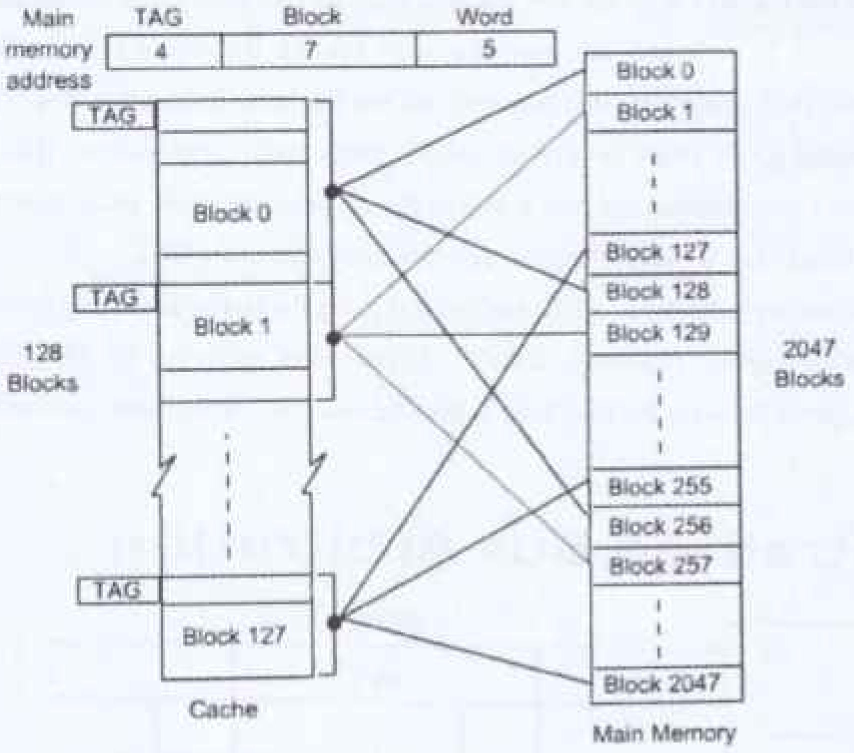
Figure 8:Main Memory Address

It is the simplest mapping technique, in which each block from the main memory has only one possible location in the cache organization. For example, the block I of the main memory maps on to block i module128 of the cache. Therefore, whenever one of the main memory blocks 0, 128, 256, ……. Is loaded in the cache, it is stored in the block 0. Block 1, 129, 257,….. are stored in block 1 of the cache and so on.

(2+3)

2→Exp

3→Diag



Total 5M

3(a) Make use of MOVE (R_s), R_d instruction write the complete control sequence and brief about performing an arithmetic or logic operation.

Write the complete control sequence for the instruction : Move (Rs),Rd

• This instruction copies the contents of memory-location pointed to by Rs into Rd. This is a memory read operation.

This requires the following actions

→ fetch the instruction

→ fetch the operand (i.e. the contents of the memory-location pointed by Rs).

→ transfer the data to Rd.

5M

5 -> Sequen

• The control-sequence is written as follows

1) PCout, MARin, Read, Select4, Add, Zin

2) Zout, PCin, Yin, WMFC

3) MDRout, IRin

4) Rs , MARin, Read

5) MDRin, WMFC

6) MDRout, Rd, End

PERFORMING AN ARITHMETIC OR LOGIC OPERATION

• The ALU performs arithmetic operations on the 2 operands applied to its A and B inputs.

• One of the operands is output of MUX & the other operand is obtained directly from bus.

• The result (produced by the ALU) is stored temporarily in register Z.

• The sequence of operations for [R3]←[R1]+[R2] is as follows.

5M

5->Pgm

```
1) R1out, Yin                   //transfer the contents of R1 to Y register
2) R2out, SelectY, Add, Zin   //R2 content s are transferred directly to B input of ALU.
                               // The numbers of added. Sum stored in register Z
3) Zout, R3in                 //sum is transferred to register R3
```

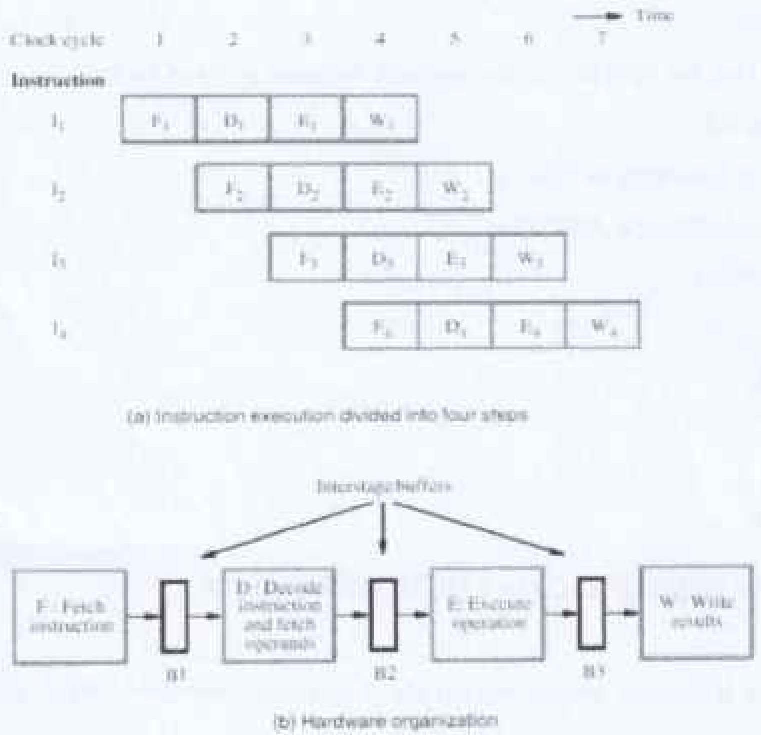
• The signals are activated for the duration of the clock cycle corresponding to that step. All other signals are inactive.

Total (5+5) 10M

3(b) Explain the Role of Cache Memory along with 4 Stage Pipeline block diagram.

ROLE OF CACHE MEMORY:

Each stage in a pipeline is expected to complete its operation in one clock cycle. Hence, the clock period should be sufficiently long to complete the task being performed in any stage. If different units require different amounts of time, the clock period must allow the longest task to be completed. A unit that completes its task early is idle for the remainder of the clock period. Hence, pipelining is most effective in improving performance if the tasks being performed in different stages require about the same amount of time.


Time
Clock cycle 1 2 3 4 5 6 7
Instruction
I_1 F_1 D_1 E_1 W_1
I_2 F_2 D_2 E_2 W_2
I_3 F_3 D_3 E_3 W_3
I_4 F_4 D_4 E_4 W_4
Interstage buffers
F: Fetch instruction
D: Decode instruction and fetch operands
E: Execute operation
W: Write results
B1
B2
B3

(a) Instruction execution divided into four steps

(b) Hardware organization

Figure 8.2 A 4-stage pipeline

(5M)
5 -> Diag

This consideration is particularly important for the instruction fetch step, which is assigned one clock period in Figure 8.2a. The clock cycle has to be equal to or greater than the time needed to complete a fetch operation. However, the acc ess time of the main memory may be as much as ten times greater than the time needed to perform basic pipeline stage operations inside the processor, such as adding two numbers. Thus if each instruction fetch required access to the main memory, pipelining would be of little value. The use of cache memories solves the memory access problem. In particular, when a cache is included on the same chip as the processor, access time to the cache usually the same as the time needed to perform other basic operations inside the processor. This makes it possible to divide instruction fetching and processing into steps that are more or less equal.

(5M)
5 -> Exp

Total
(5+5)
10M

4(a) Make use of block diagram and appropriate timing diagram explain how to fetch a WORD from MEMORY.

FETCHING A WORD FROM MEMORY

• To fetch instruction/data from memory, processor transfers required address to MAR (whose output is connected to address-lines of memory-bus).

At the same time, processor issues Read signal on control-lines of memory-bus.

• When requested-data are received from memory, they are stored in MDR. From MDR, they are transferred to other registers

• MFC (Memory Function Completed): Addressed-device sets MFC to 1 to indicate that the contents of the specified location.

→ have been read &

→ are available on data-lines of memory-bus

• Consider the instruction Move (R1),R2. The sequence of steps is:

1) R1out, MARin, Read	;desired address is loaded into MAR & Read command is issued
2) MDRinE, WMFC	;load MDR from memo ry bus & Wait for MFC response from memory
3) MDRout, R2in	;load R2 from MDR

where WMFC=control signal that causes processor's control circuitry to wait for arrival of MFC signal

5M

5→Exp

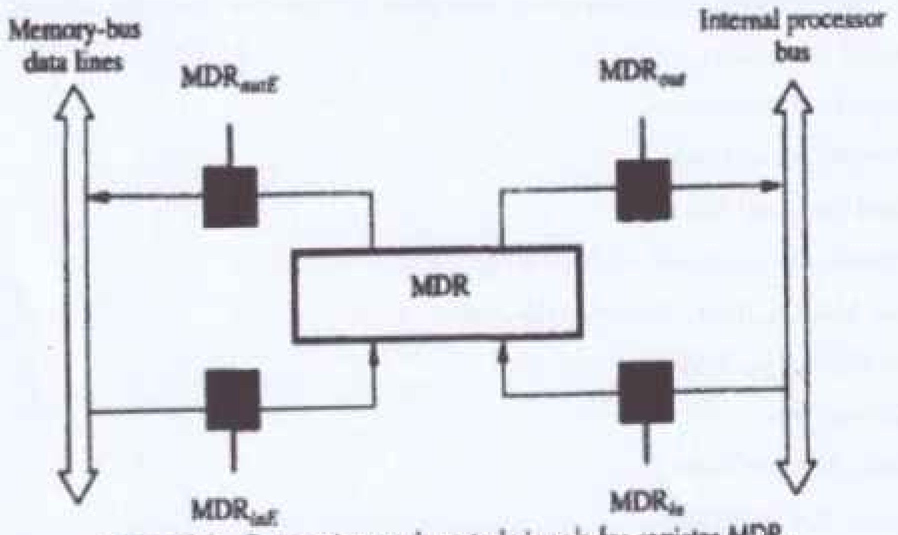


Figure 7.4 Connection and control signals for register MDR.

2M

2→Diag

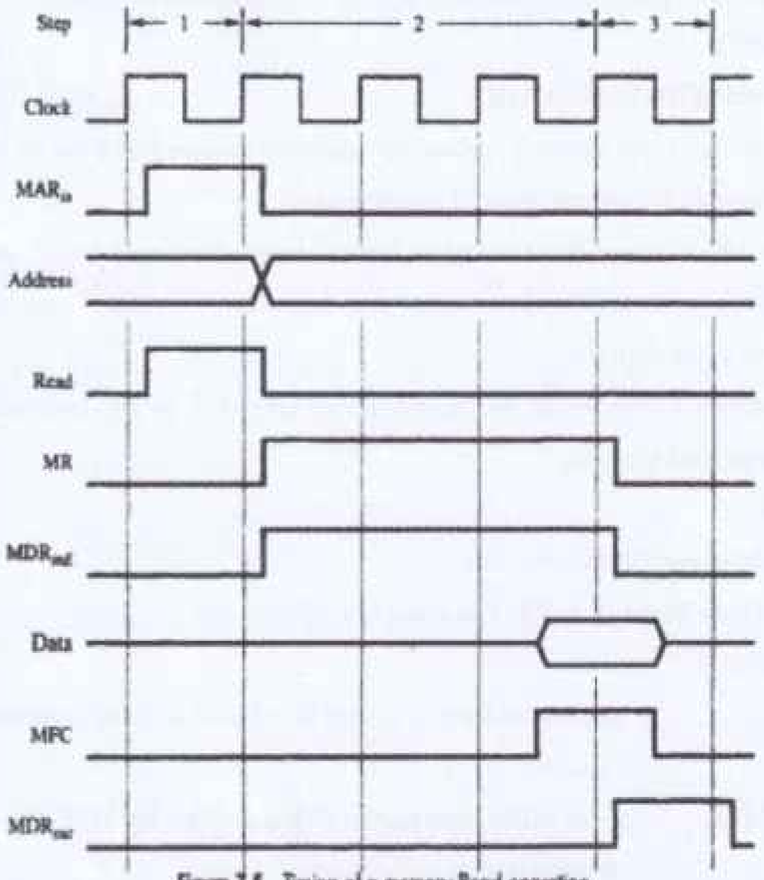


Figure 7.5 Timing of a memory Read operation.

3M
3 ⇒ Timing Diag
Total
(5+2+3)
10M

4(b) Explain Execution of A COMPLETE INSTRUCTION briefing all the 7 Steps.

EXECUTION OF A COMPLETE INSTRUCTION

• Consider the instruction Add (R3), R1 which adds the contents of a memory-location pointed by R3 to register R1. Executing this instruction requires the following actions:

1) Fetch the instruction.
2) 2) Fetch the first operand.
3) 3) Perform the addition.
4) 4) Load the result into R1.

• Control sequence for execution of this instruction is as follows

1) PCout, MARin, Read, Select4, Add, Zin
2) Zout, PCin, Yin, WMFC
3) MDRout, Irin
4) R3out, MARin, Read
5) R1out, Yin, WMFC
6) MDRout, SelectY, Add, Zin
7) Zout, R1in, End

5M
5 ⇒ Exp

	• Instruction execution proceeds as follows: Step1--> The instruction-fetch operation is initiated by loading contents of PC into MAR & sending a Read request to memory. The Select signal is set to Select4, which causes the Mux to select constant 4. This value is added to operand at input B (PC"s content), and the result is stored in Z. Step2--> Updated value in Z is moved to PC. Step3--> Fetched instruction is moved into MDR and then to IR. Step4--> Contents of R3 are loaded into MAR & a memory read signal is issued. Step5--> Contents of R1 are transferred to Y to prepare for addition. Step6--> When Read operation is completed, memory-operand is available in MDR, and the addition is performed. Step7--> Sum is stored in Z, then transferred to R1.The End signal causes a new instruction fetch cycle to begin by returning to step1.	5M 5 ⇒ Steps Total (5+5) 10M

[illegible]
Course in-charge
Rachana V. Murthy

[illegible]
Module Coordinator

[illegible]
HOD

K.S. INSTITUTE OF TECHNOLOGY, BENGALURU - 560109
THIRD INTERNAL TEST QUESTION PAPER 2023-24 ODD SEMESTER

SET: A

Degree	:	B.E	Semester	:	3RD
Branch - Stream	:	CSE – ICB	Course Type / Code	:	IPCC/BCS302
Course Title	:	DDCO	Date	:	04-03-2024
Duration	:	1 Hr (60 minutes)	Max Marks	:	25

USN

Note: Answer **ONE full** question from each Module.
K-Levels: K1-Remebering, K2-Understanding, K3-Applying, K4-Analyzing, K5-Evaluating, K6-Creating

Q No.	Questions	Marks	CO	K-Level
	Module 4			
1(a)	Make use of a block diagram explain Distributed Arbitration in detail.	5	CO4	K3
	OR			
2(a)	Make use of Mapping Techniques explain in detail the Associative Mapping briefly.	5	CO4	K3
	Module 5			
3(a)	Make use of the sequential execution and hardware organization diagram explain the basic concepts of Pipelining.	10	CO5	K3
(b)	Explain the Role of Cache Memory along with 4 Stage Pipeline block diagram.	10	CO5	K2
	OR			
4(a)	Make use of the date path inside the processor explain in detail the Single bus organization.	10	CO5	K3
(b)	Explain Input and Output Gating for one Register Bit.	10	CO5	K2

[illegible]
Name & Signature of Course In charge:
Rachana. V. Murthy

[illegible]
Name & Signature of Module Coordinator:
KOSHAL KUMAR BN

[illegible]
HOD

[illegible]
Principal

K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109

III SESSIONAL TEST QUESTION PAPER 2023 – 24

ODD SEMESTER

Set A — **SCHEME AND SOLUTION**

Degree	: **B. E**	**Semester**	: **III**
Branch	: **Computer Science - ICB**	**Course Code**	: **Integrated/ BCS302**
Course Title	: **Digital Design and Computer Organization**	**Max Marks**	: **25**

Q.NO.	*POINTS*	*MARKS*
1(a)	Make use of a block diagram explain Distributed Arbitration in detail.	

Distributed Arbitration: -Distributed arbitration means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process, without using a central arbiter. A simple method for distributed arbitration is illustrated in figure 6. Each device on the bus assigned a 4-bit identification number. When one or more devices request the bus, they assert the Start – Arbitration signal and place their 4-bit ID numbers on four open-collector lines, ARB0 through ARB3 . A winner is selected as a result of the interaction among the signals transmitted over those liens by all contenders. The net outcome is that the code on the four lines represents the request that has the highest ID number.

(2+3)

2⇒Exp

3⇒Diag

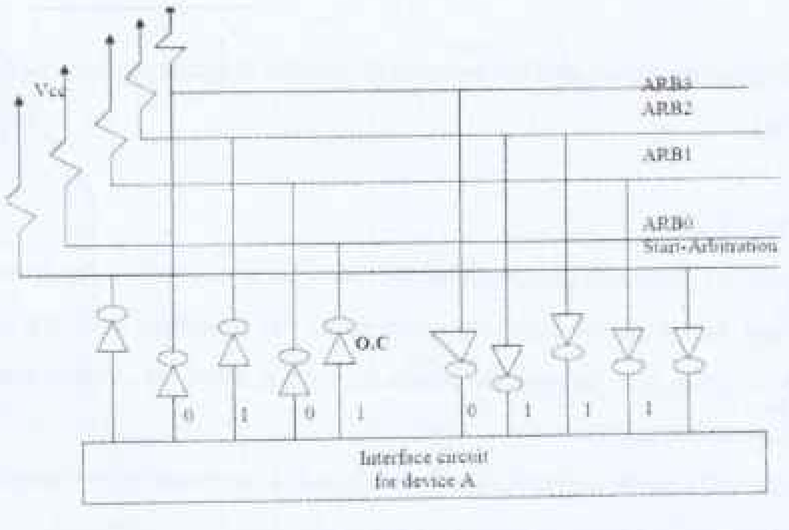


Fig 6 A distributed arbitration

Total 5M

2(a) Make use of Mapping Techniques explain in detail the Associative Mapping briefly.

Associative Mapping Technique The figure shows the associative mapping, where in which main memory block can be placed into any cache block position. in this case, 12 tag bits are required to identify a memory block when it is resident in the cache. The tag 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. This is called associative - mapping technique. It gives the complete freedom in choosing the cache location in which to place the memory block.

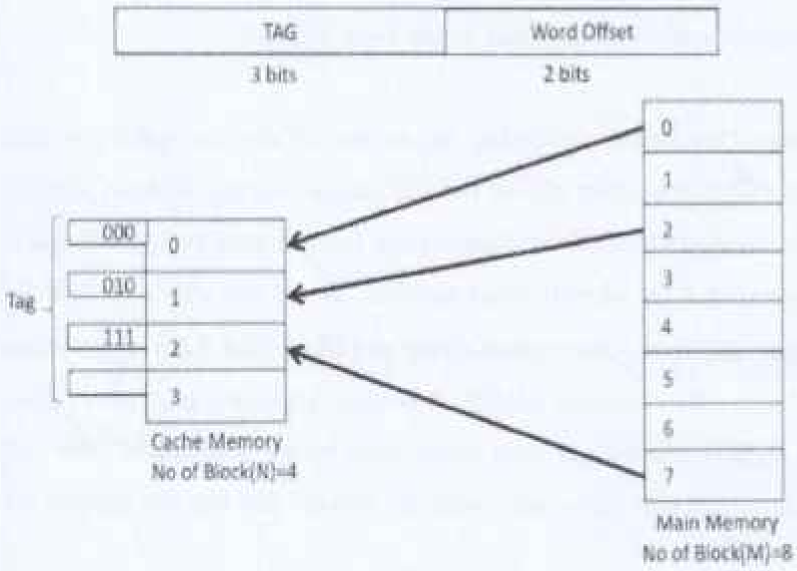
(2+3)

2→Exp

3→Diag

Associative Mapping

Any Block of main memory can be placed any where in the cache



Total 5M

3(a) Make use of the sequential execution and hardware organization diagram explain the basic concepts of Pipelining.

BASIC CONCEPTS

The speed of execution of programs is influenced by many factors. One way to improve performance is to use faster circuit technology to build the processor and the main memory. Another possibility is to arrange the hardware so that more than one operation can be performed at the same time.

In this way, the number of operations performed per second is increased even though the elapsed time needed to perform any one operation is not changed.

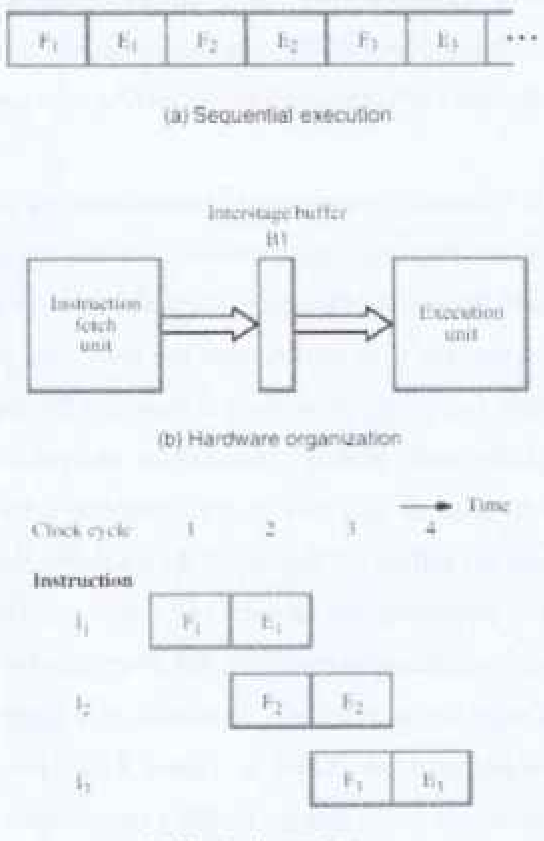
We have encountered concurrent activities several times before. Chapter 1 in-traduced the concept of multiprogramming and explained how it is possible for I/Transfers and computational activities to proceed simultaneously. DMA devices make this possible because they can perform I/O transfers independently once these transfers are initiated by the processor Pipelining is a particularly effective way of organizing concurrent activity in a computer system.

The basic idea is very simple. It is frequently encountered in manufacturing plants, where pipelining is commonly known as an assembly-line operation. Readers are undoubtedly familiar with the assembly line used in car manufacturing. The first station in an assembly line may prepare the chassis of a car, the next station adds the body, and the next one installs the engine, and so on. While one group of workers is installing the engine on one car, another group is fitting a car body on the chassis of another car, and yet another group is preparing a new chassis for a third car. It may take days to complete work on a given car, but it is possible to have a new car rolling off the end of the assembly line every few minutes. Consider how the idea of pipelining can be used in a computer. The processor executes a program by fetching and executing instructions, one after the other. Let Fi and Ei refer to the fetch and execute steps for instruction Ii. Execution of a program consists of a sequence of fetch and execute steps, as shown in Figure 8.1a. Now consider a computer that has two separate hardware units, one for fetching instructions and another for executing them, as shown in Figure 8.1 b.

(5M)

5→Exp

The instruction fetched by the fetch unit is deposited in an intermediate storage buffer, B1. This buffer is needed to enable the execution unit to execute the instruction while the fetch unit is fetching the next inst ruction. The results of execution are deposited in the destination location specified by the instruction. For the purposes of this discussion, we assume that both the source and the destination of the data operated on by the instructions are inside the block labelled Execution unit.



(a) Sequential execution

(b) Hardware organization

(c) Pipelined execution

Figure 8.1 Basic idea of instruction pipelining

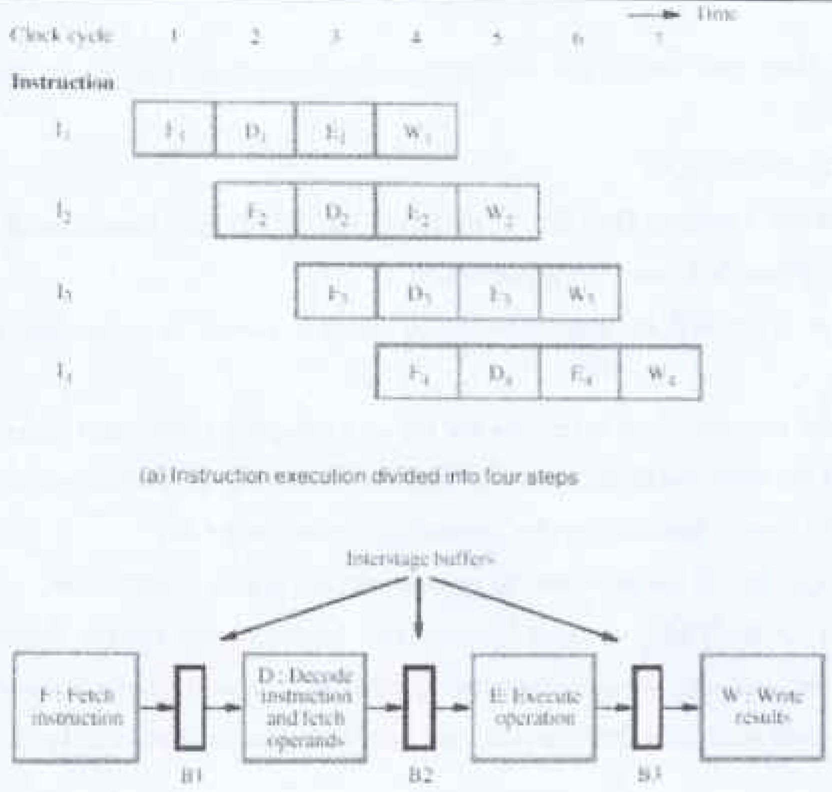
5M
5→Diag

Total
(5+5)
10M

3(b) Explain the Role of Cache Memory along with 4 Stage Pipeline block diagram.

ROLE OF CACHE MEMORY:

Each stage in a pipeline is expected to complete its operation in one clock cycle. Hence, the clock period should be sufficiently long to complete the task being performed in any stage. If different units require different amounts of time, the clock period must allow the longest task to be completed. A unit that completes its task early is idle for the remainder of the clock period. Hence, pipelining is most effective in improving performance if the tasks being performed in different stages require about the same amount of time.



Clock cycle 1 2 3 4 5 6 7
Time
Instruction
I_1 F_1 D_1 E_1 W_1
I_2 F_2 D_2 E_2 W_2
I_3 F_3 D_3 E_3 W_3
I_4 F_4 D_4 E_4 W_4
Interstage buffers
F : Fetch instruction
D : Decode instruction and fetch operands
E: Execute operation
W : Write results
B1
B2
B3

(a) Instruction execution divided into four steps

(b) Hardware organization

Figure 8.2 A 4-stage pipeline

5M
5⇒Diag

This consideration is particularly important for the instruction fetch step, which is assigned one clock period in Figure 8.2a. The clock cycle has to be equal to or greater than the time needed to complete a fetch operation. However, the acc ess time of the main memory may be as much as ten times greater than the time needed to perform basic pipeline stage operations inside the processor, such as adding two numbers. Thus if each instruction fetch required access to the main memory, pipelining would be of little value. The use of cache memories solves the memory access problem. In particular, when a cache is included on the same chip as the processor, access time to the cache usually the same as the time needed to perform other basic operations inside the processor. This makes it possible to divide instruction fetching and processing into steps that are more or less equal.

5M
5⇒Exp

Total
(5+5)
10M

4(a) Make use of the date path inside the processor explain in detail the Single bus organization.

SINGLE BUS ORGANIZATION

• MDR has 2 inputs and 2 outputs. Data may be loaded → into MDR either from memory-bus (external) or → From processor-bus (internal).

• MAR"s input is connected to internal-bus, and MAR"s output is connected to externalbus.

• Instruction-decoder & control-unit is responsible for → Issuing the signals that control the operation of all the units inside the processor (and for interacting with memory bus). → implementing the actions specified by the instruction (loaded in the IR)

• Registers R0 through R(n-1) are provided for general purpose use by programmer.

• Three registers Y, Z & TEMP are used by processor for temporary storage during execution of some instructions. These are transparent to the programmer i.e. programmer need not be concerned with them because they are never referenced explicitly by any instruction.

• MUX(Multiplexer) selects either → output of Y or → constant-value 4(is used to increment PC content). This is provided as input A of ALU.

• B input of ALU is obtained directly from processor-bus.

• As instruction execution progresses, data are transferred from one register to another, often passing through ALU to perform arithmetic or logic operation.

• An instruction can be executed by performing one or more of the following operations:

1) Transfer a word of data from one processor-register to another or to the ALU.

2) Perform arithmetic or a logic operation and store the result in a processor - register.

3) Fetch the contents of a given memory-location and load them into a processor - register.

4) Store a word of data from a processor-register into a given memory-location.

5M

5 → Exp

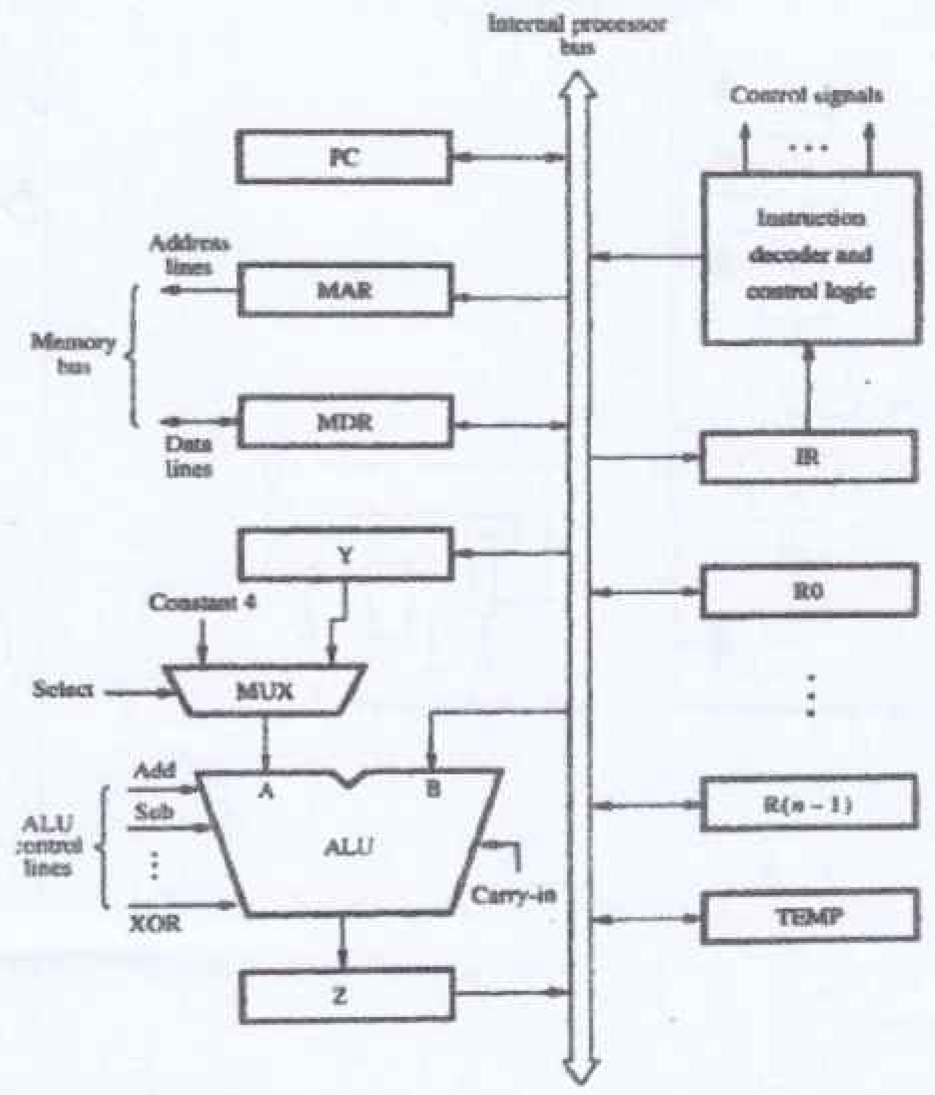


Figure 7.1 Single-bus organization of the datapath inside a processor.

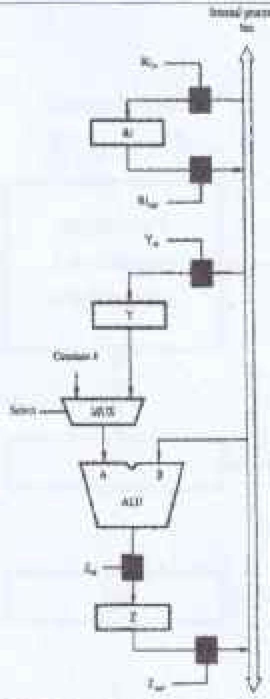
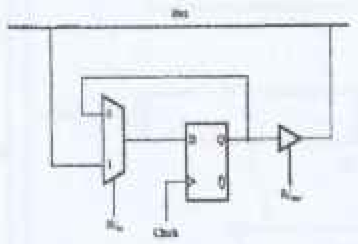
5M
5→Diag

Total
(5+5)
10M

4(b) Explain Input and Output Gating for one Register Bit.

- A 2-input multiplexer is used to select the data applied to the input of an edge-triggered D flip-flop.
- Riin=1 - mux selects data on bus. This data will be loaded into flip-flop at rising-edge of clock. Riin=0 - mux feeds back the value currently stored in flip-flop (Figure7.3).
- Q output of flip-flop is connected to bus via a tri-state gate. Riout=0 - gate's output is in the high-impedance state. Riout=1 - the gate drives the bus to 0 or 1, depending on the value of Q.

5M
5→Exp

	 Internal processor bus R_{in} Ri R_{iout} Y_{in} Y Constant 4 Select MUX A B ALU Z_{in} Z Z_{out} Figure 7.2 Input and output gating for the registers in  Bus 0 1 D Q Q Clock R_{in} R_{out} Figure 7.3 Input and output gating for one register bit.	5M 5→Diag Total (5+5) 10M

[illegible]
Course in-charge
Rachana V. Murthy

[illegible]
Module Coordinator

[illegible]
HOD