Date 11.07.19

Unit-1

Introduction

Difference b/w :-

Computer Organization	Computer Architecture
1. Computer organisation refers to all physical aspects of a computer system.	1. Computer architecture refers to all logical aspects of computer system.
2. It describes how computer system works.	2. It describes how computer system designed.
3. It involve circuit design, operation units (CPU, Memory, I/O) inter connections, control signal generation.	3. It involve instruction formats, data types, instruction sets, address modes.

Buses :-

Set of electrical wires used to connect multiple devices in a computer system.

System Bus :-

A Bus used to connect major components i.e., CPU, memory and I/O devices.

Local Bus :-

A Bus used to connect minor components inside major components or other major additional

elements or components to the computer system.

Classification of System Bus :-

Bus Architecture :

1. Data Bus →

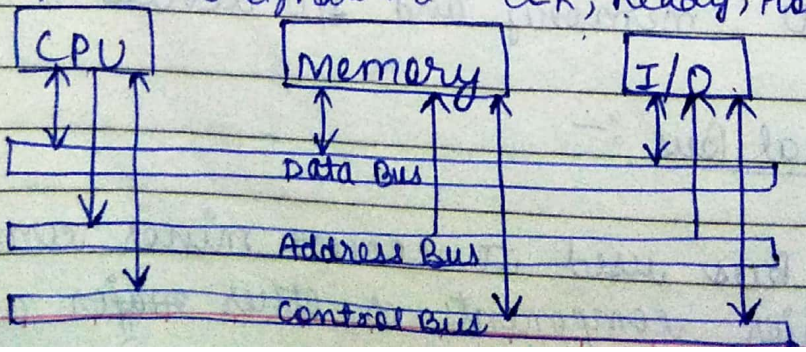
(i) It is bi-directional in nature.
(ii) It carries data as well as instruction.
(iii) It carries data from CPU to memory or input/output devices and vice-versa.
(iv) consist of 8, 16, 32 or more parallel signal lines

2. Address Bus →

(i) It is unidirectional in nature.
(ii) It carries address only.
(iii) It carries address from CPU to memory or Input/output devices only.
(iv) It consist of 16, 20, 24 or more parallel lines.

3. Control Bus →

(i) It is bi-directional in nature.
(ii) It carries control or timing signals.
(iii) It Normally carry control or timing signals generated by CPU but sometime carry status from I/O devices or from memory.
(iv) Typical CB signals are - CLK, Ready, Hold, INTR, etc.



Date _____

Bus Width :

The no. of bits carries by a bus is called Bus width.

* 32 bit data bus carries 32 bits of data.
* 16 bit address bus carries 16 bits address.

Classification of Bus on the basis of Structure :

1. Dedicated Bus.	2. Multiplexed Bus.
(i) Bus permanently assigned to one function.	(i) Same Bus is used for different functions.
(ii) It requires separate bus for data and address.	(ii) Same Bus for data and address.
(iii) Increased cost due to multiple no. of buses.	(iii) Reduced cost due to sharing bus.
(iv) High throughput	(iv) Low throughput.
(v) Less bus contention.	(v) High bus contention.
(vi) Less complexity.	(vi) High complexity.

Bus Arbitration :

It is a process which decides among a no. of devices requesting to use the system bus, which one to grant access of bus.

2^{10} = 1 KB
2^{20} = 1 MB
2^{30} = 1 GB

Date

Types of Bus Arbitration :-

1. Centralized Arbitration → A single hardware device (CPU, DMA controller) is used to acts as arbiter (selector) to decide among a no. of devices that which one will use the bus firstly. There are three types of centralised arbitration:
   (i) Daisy chaining method.
   (ii) Polling method.
   (iii) Independent request method.

2. Distributed Arbitration → Every device is involved in the selection of which one will use the bus firstly.

Bus Master :-

A device that initiates, ~~bus~~ data transfer on the bus at any given time is called Bus master. and other devices are called Bus slaves or devices.

Ques 1. Find the no. of address lines (bits) of 4MB RAM.

Sol.

$1 MB = 2^{20}$

$4 MB = 2^2 \times 2^{20}$

no. of add. lines $= 2^{22}$

no. of add. lines = 22.

Date

Ques. Find the memory capacity of 14 bits of address lines.

Sol. Address lines = 14 bits.

$= 2^{14}$

$= 2^4 \times 2^{10}$

$= 16 \times 1K$

Memory capacity = 16 K

1. Daisy chaining Method :

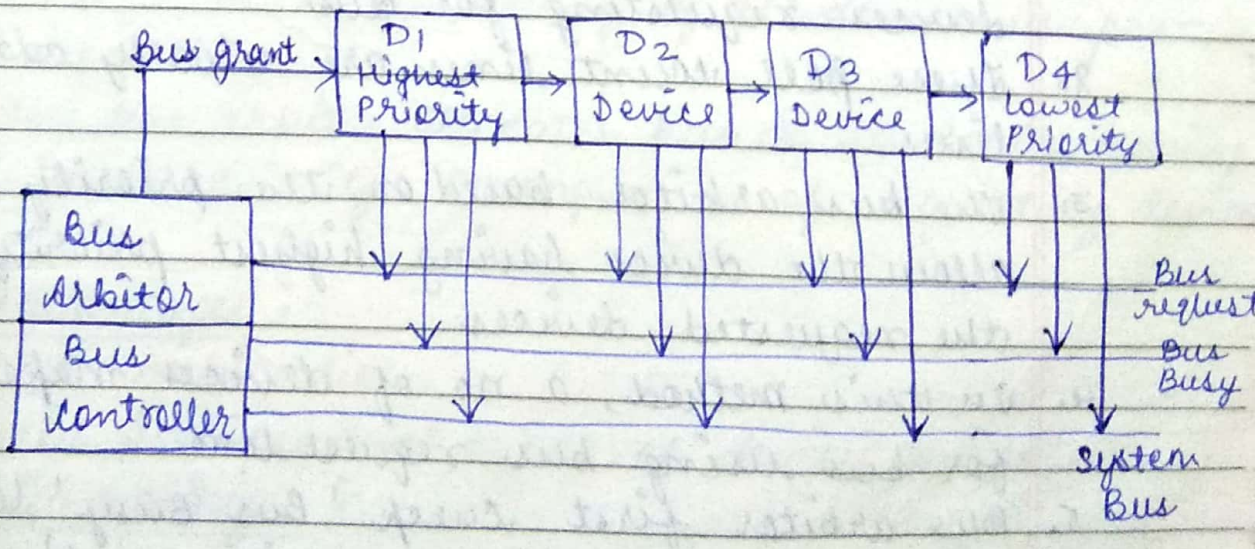
The device which is physically closer to the Bus controller / Arbitor will be allowed to use the bus first.

The three control signals are used -

(i) Bus Request : It is used by devices to make a request for bus.

(ii) Bus Busy : It is used to show that bus is already used by other devices.

(iii) Bus Grant : It is used to allow a device to grant access of bus.



Date

Advantages :

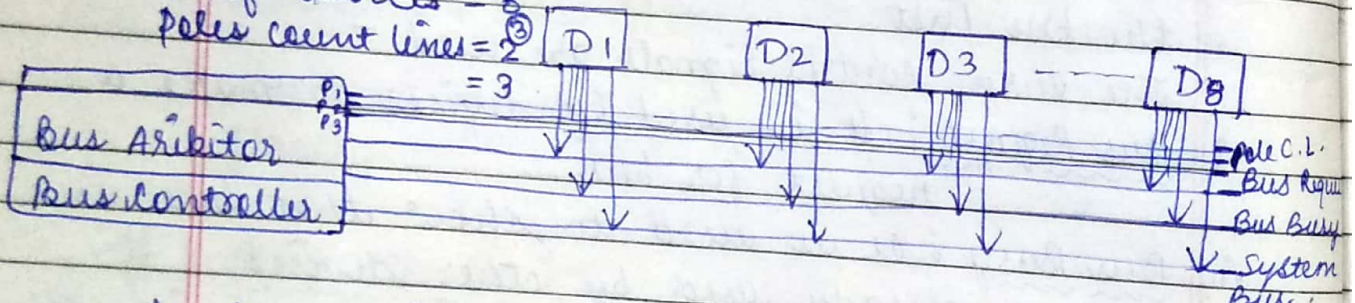
1. Simple circuit
2. Low cost.
3. Scalable.
4. Require least no. of lines & this no. is independent of no. of master in system.

Disadvantages :

1. Low speed.
2. Less Reliability, failure of any master causes whole system to fail.
3. Starvation.
4. Priority of master is fixed by its physical location.

2. Polling Method :

no. of devices = 8
poles count lines = $\log_2 8 = 3$

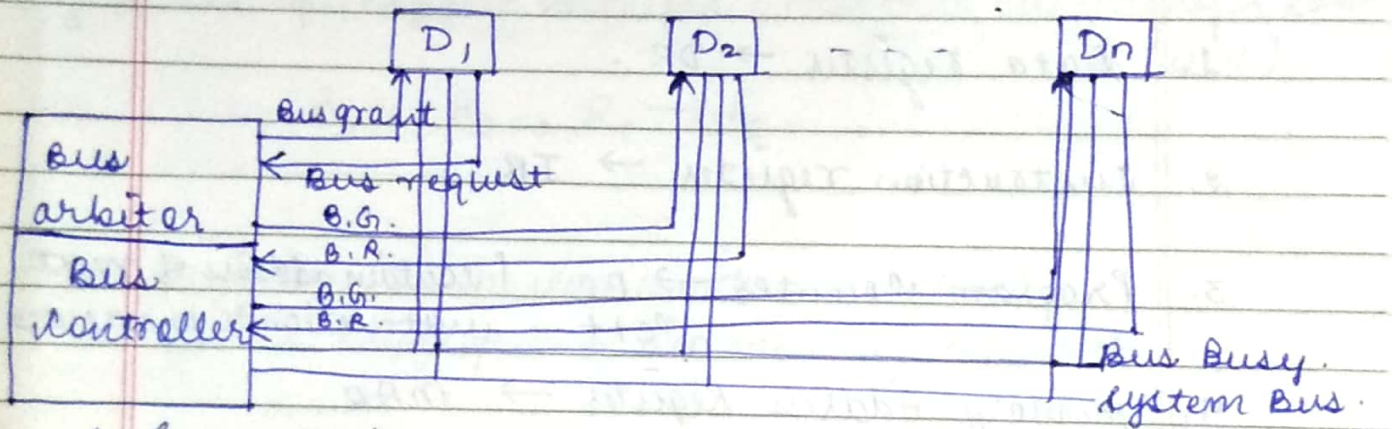


1. In this method, a set of poll count lines are used to select a device from number of devices requesting for bus.
2. These poll count lines are actually address lines.
3. The bus arbitor based on the priority of devices allow the device having highest priority among the requested devices.
4. In this method, a no. of devices make a request for bus using bus request line.
5. Bus arbitor first check 'Bus Busy' line and if it is free then Bus arbitor active poll count lines based on priority & grant permission to bus transfer.

Advantages:
1. Reduced cost due to common bus.
2. One module fail, entire system does not fail.
3. Flexible priority.

Disadvantages:
Poll count lines increase as no. of devices increase

3. Independent Request Method:



1. Each device make a request using separate 'Bus Request' control lines.
2. Bus arbitor grant Bus access to a device using separate Bus Grant 'control lines'.
3. All devices use common Bus Busy & control lines.
4. In this method, priority of all devices is pre-defined.
5. The bus arbitor allocates bus to the device having 'highest priority' among a no. of requesting devic

Advantages:
1. Fast Arbitration.
2. Speed Independent of no. of devices connect.

Disadvantages:
1. Large no. of control lines.
2. High cost due to large no. of bus lines.

1) Information Transfer from 1 - Reg. to another is called Register transfer, $R_2 \leftarrow R_1$.

2) It shows the transfer of content of Reg. R_1 to Reg. R_2.

3) Content of R_1 does not change.

Register Transfer Language :-

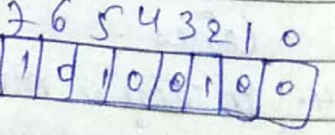
Symbolic notation used to represent the transfer of information between register.

Representation of Registers :-

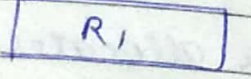
1. Data register → DR.
2. Instruction register → IR.
3. Program Counter → PC. [contain address of next instruction to be executed] PC++.
4. Memory Address Register → MAR.
5. General Purpose Register → $R_1, R_2, R_3 \ldots$ [carry data, Add, Instⁿ]

(i) Letter → $[R_1]$ ✓ ①
Numeral

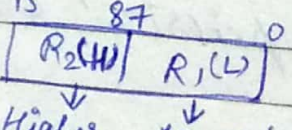
(ii) Only Symbol → MAR, PC, IR,

(iii) Individual Bit Representation → 

✓ ⑥

(iv) Bit System → 

✓ ③

(v) 16 - Bit System → 

✓ ④

(vi) $R_1(0-5)$ → Part of register.

② → It can be shown by if-else statement.

(vii) Register Transfer → P: $R_2 \rightarrow R_1$, ~~R: MAR ← PC~~
If P = 1 then $R_1 \leftarrow R_2$

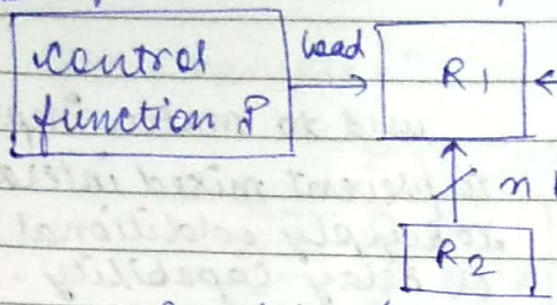
① → [Conditional Transfer of information] occur only under specific condition

4) Content of R_2 is replaced by content of R_1.
5) This transfer happen in 1 cycle.

Date

③ → It symbolise transfer operation takes place by hardware only if [P = 1]

④ →



Block diagram

⑤ Parallel Reg. transfer.
(microoperation)

Set of Register transfer (more than one transfer, comma is used).

$R_3 \leftarrow R_2$, $R_1 \rightarrow R_2$

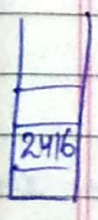
★ Register is a combination of flipflop.
1 Flipflop = 1 bit memory elements

Memory Transfer :-

1. Memory Read → To transfer any information from memory to external environment (CPU)

if AR = 300

DR ← M[AR]
(memory content at address stored in AR register) words is transferred into data Register ~~from memory words~~



2. Memory Write → To transfer any information from external environment (CPU) to memory.

M[AR] ← DR.

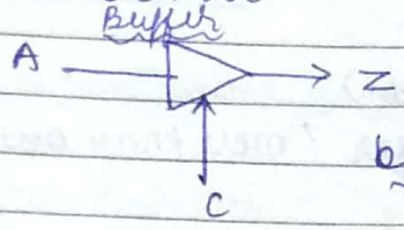
Content of data Register transfer to memory words at address stored in AR register.

→ A 3-state gate can be any logic gate But in Bus System we use only 3-state Buffer gate.

Bus Transfer :-

★ Representation of Tristate Gate -

used to match impedance, to prevent mixed interaction to supply additional drive or delay capability.

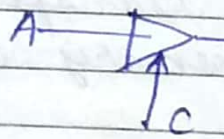


Tristate used for buffer gate (time delay)

If $C = 1$, $Z = A$ i.e., A Gate behaves as like its actual behaviour or Gate is active.

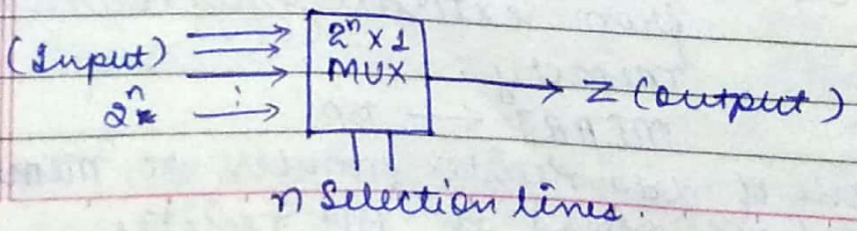
If $C = 0$, High impedance state, circuit act as open which is disable state.

★ Tri state Buffer Gate → A buffer gate having one input & one output as in conventional buffer gate along with one control input.



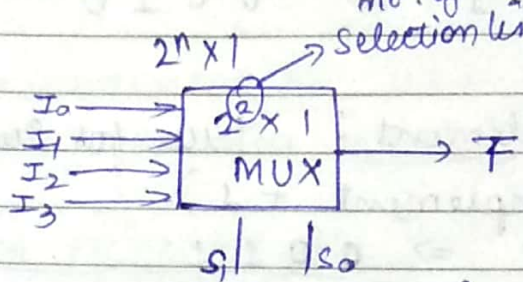
Multiplexer (MUX) :-

It is a digital circuit used to pass one output, 2^n (or less) input. It is called data selector from



High Impedance state behaves like open circuit i.e, the output is disconnected & does not have a logic significance.

for example :-



S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

* No. of Mux Required = No. of bits in each registers.
* Size of MUX = NO. of Registers.

* Three State Bus Buffer :-

A Bus system that can be constructed with three state buffer gate.

* Three state gate :-

A 3- state gate is a digital circuit that shows 3- states. (i) State - 1 (ii) state - 0
(iii) High Impedance state.

Date

1's complement:

1 1 0 1 ⇒ 0 0 1 0

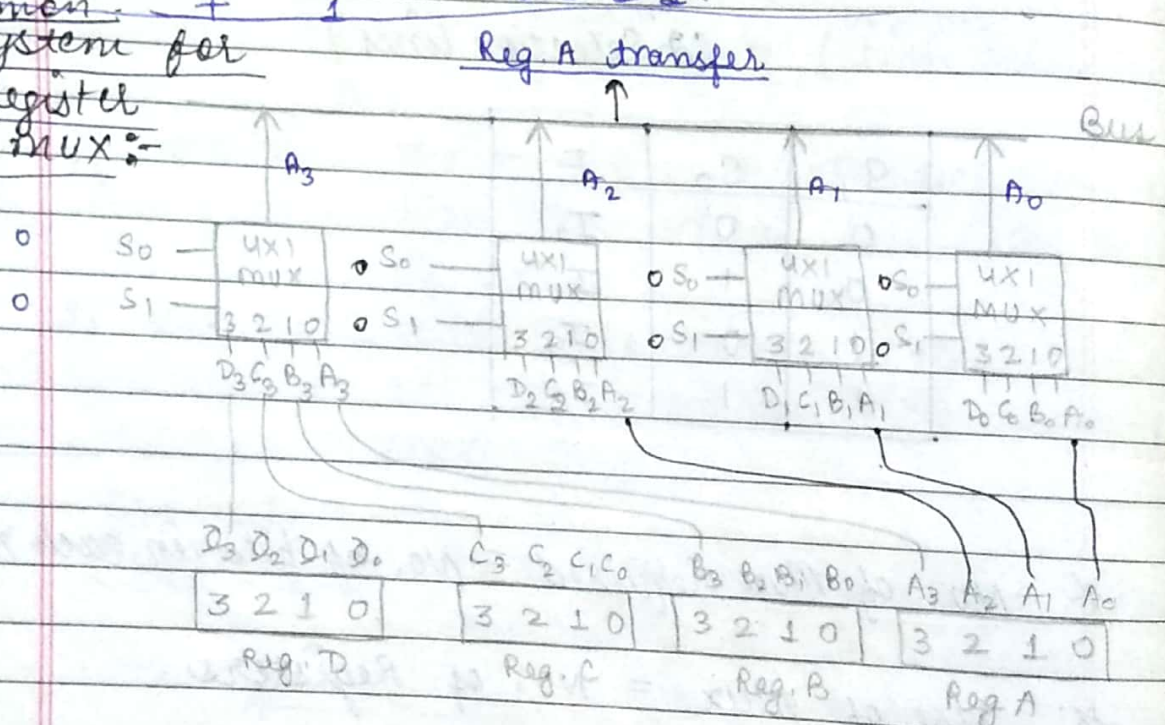
2's complement: (use for Subtraction)

(1's complement + 1)

0 0 1 0 ⇒ 0 0 1 1

\+ 1

Common Bus system for four-Register using MUX:-



	S_0	S_1	Reg. Select
I_0	0	0	A
I_1	0	1	B
I_2	1	0	C
I_3	1	1	D

A − B = A + 2's complement of B. = A + B' + 1

for eg. A + 0 = 0 0 0 0

− 0 = ?

1's complement of +0 = 1 1 1 1.

1's complement of +0 + 1 = 1 1 1 1

\+ 1

neglected ← (1) 0 0 0 0

in 4-bit reg.

$$A - B = A + B' + 1$$

Date

Microoperation :-

The fundamental or basic operations that are performed on the content of registers.

Types of Microperations :-

1. Arithmetic micro operation – perform arithmetic operation on numeric data stored in register
2. Logical m.o – perform bit manipulation operation on numeric data stored in registers
3. Shift m.o. – perform shift operation on data stored in register

Ques 1 A digital computer has a common work system for 16 registers of 32 bits each. The bus is constructed with multiplexers.

(i) How many multiplexers are there in the bus?

(ii) What size of multiplexer are needed.

(iii) How many selection inputs are there in each multiplexers

Sol

(i) No. of multiplers = No. of bits in each register
= 32

(ii) Size of MUX = No. of registers used.
= 16 × 1 MUX
= 2^4 × 1 MUX

(iii) No. of selection input = 4
= 4 lines

Date

Ques.2 The following transfer statement specify a memory. Explain the memory operation in each case.

(i) $R_2 \leftarrow M[AR]$.

(ii) $M[AR] \leftarrow R_3$

(iii) $R_5 \leftarrow M[R_5]$

(i) Memory Read operation, the memory content at an address specified by Register AR is transferred to Register R_2.

(ii) Memory Write Operation, the Register R_3 value is stored as memory content at an address specified by Register AR.

(iii) Memory Read operation, the memory content at an address specified by Register R_5 is transferred to Register R_5.

Ques.3 Represent the following conditional control statement by two register transfer statement with control function.

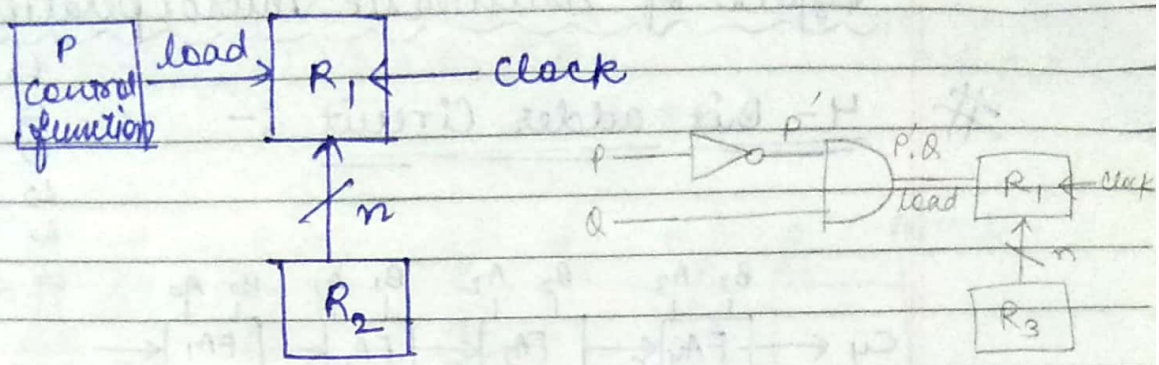
Ans (i) If $P = 1$ then $(R_1 \leftarrow R_2)$, else if $Q = 1$ then $(R_1 \leftarrow R_3)$

Sol. $P : R_1 \leftarrow R_2$

$P' \cdot Q : R_1 \leftarrow R_3$.

* full adder = 1-bit binary Adder.

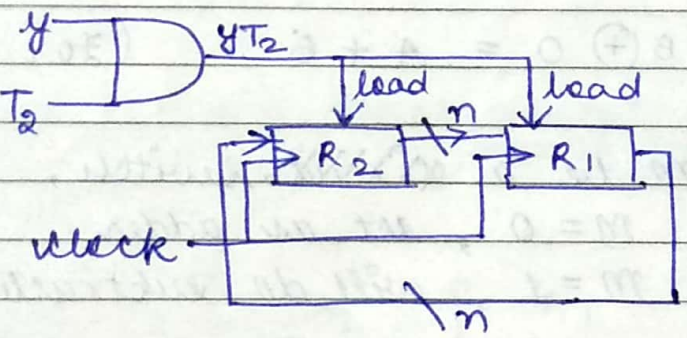
Date



Ques.4. Show the block diagram of a hardware that implements the following register transfer statement.

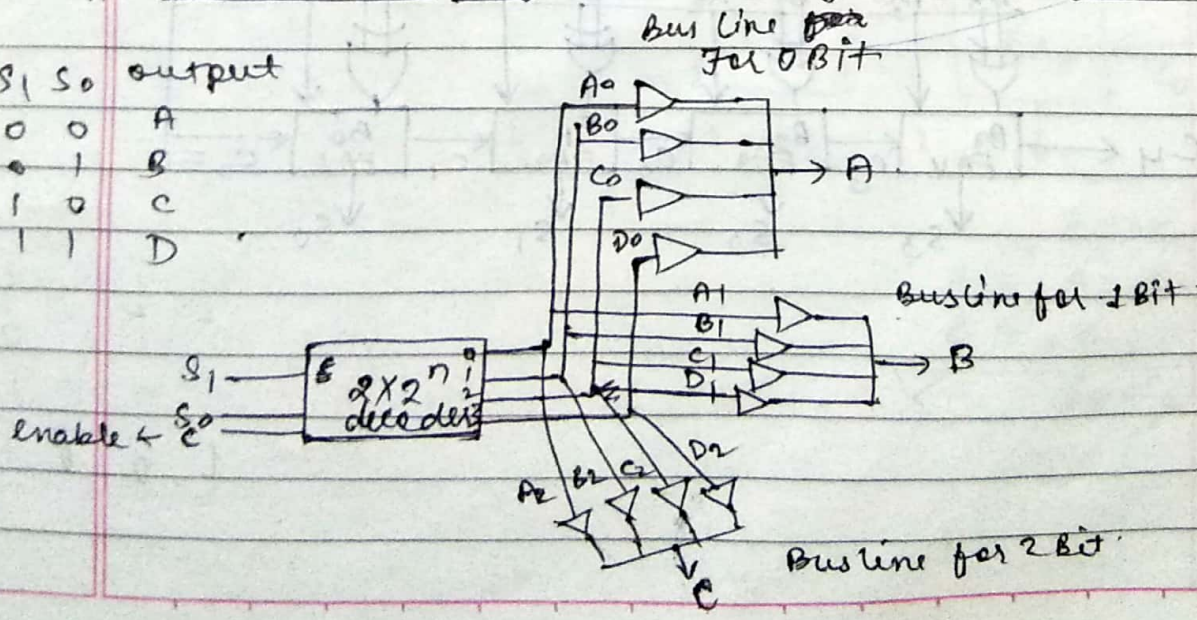
$yT_2 : R_2 \leftarrow R_1 , R_1 \leftarrow R_2$.

Sol.



Common bus system using decoder & Buffer gate.

S1	S0	output
0	0	A
0	1	B
1	0	C
1	1	D



Date

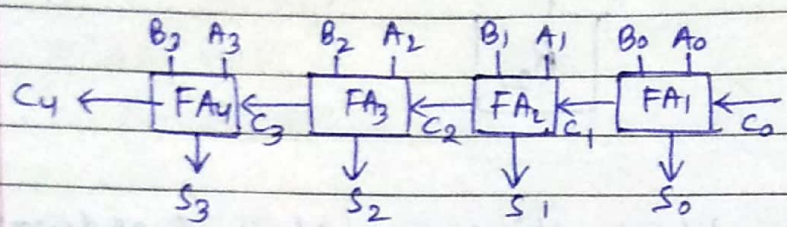
$x \oplus y = x'y + xy'$.

$x \odot y = x'y' + xy$.

Types of Arithmetic Microoperation :-

- ↳ Addition
- ↳ Subtraction
- ↳ Decrement
- ↳ Increment.
- ↳ 1's complement
- ↳ 2's complement

4-Bit adder Circuit :-



	FA_3	FA_2	FA_1	FA_0
C_4	C_3	C_2	C_1	C_0
A =	A_3	A_2	A_1	A_0
B =	B_3	B_2	B_1	B_0
	S_3	S_2	S_1	S_0

Design a 4-Bit Adder-Subtractor Circuit :

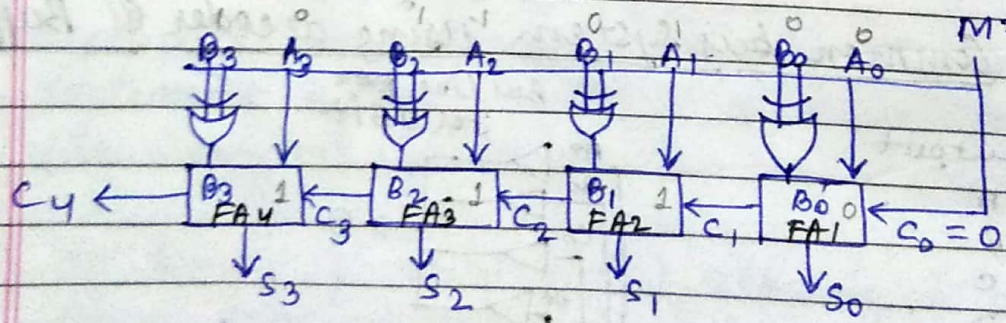
Case 1. Addition.

$A + B \oplus 0 = A + B$. (For $m = 0$).

Let m be a switch,

$m = 0$, act as adder.

$m = 1$, will do subtraction.



	8	4	2	1
A =	0	1	1	0
+ B =	0	0	1	1
C =	1	0	0	1

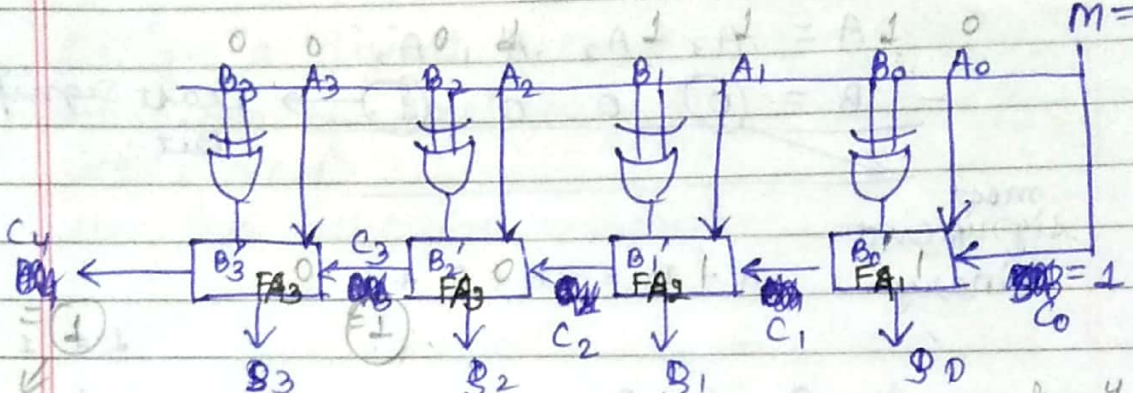
Half adder. $S = a \oplus b$
$C = a \wedge b$

Full adder $S = a \oplus b \oplus c$
$C = (a \oplus b)c \vee a \wedge b$

$A - B = A + B' + 1$

$A = B$
$B = 1$
$AB' + A'B$
$B(1)' + B'(1)$
$B \cdot 0 + B'$
$= B'$

Case 2. $A + B \oplus 1 = A_0 + B_0'$ (for $M = 1$).
Subtraction.

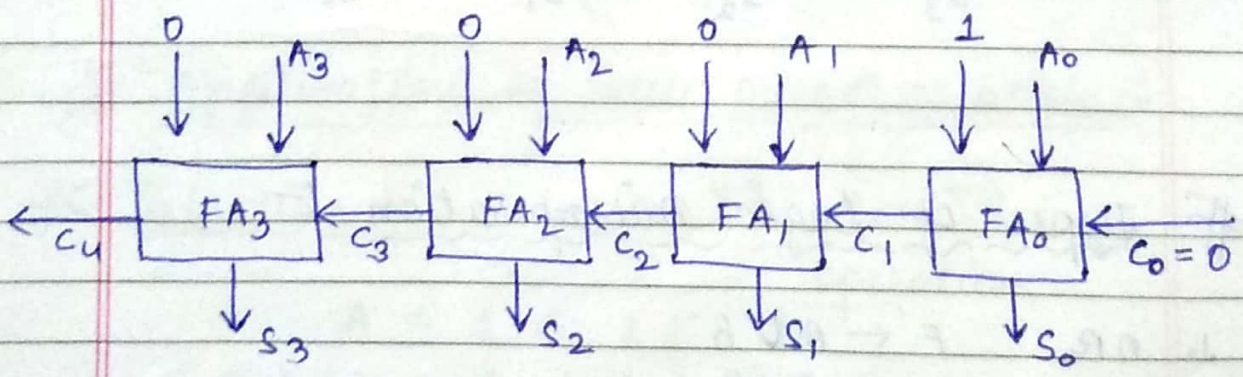


neglected

	B	4	2	1
A =	0	1	1	0
B =	+0	0	1	1
C =	0	0	1	1

Design 4-bit Incremental Circuit :

→ A + 1



	C_4	C_3	C_2	C_1	C_0
A =	A_3	A_2	A_1	A_0	
B =	+0	0	0	0	1
	S_3	S_2	S_1	S_0	

Design of 4-bit binary decrementer :-

$A = A_3\ A_2\ A_1\ A_0$

$-\ B = 0\ 0\ 0\ 1$ → Least Significant bit.

most significant binary.

$A - B = A + B' + 1$

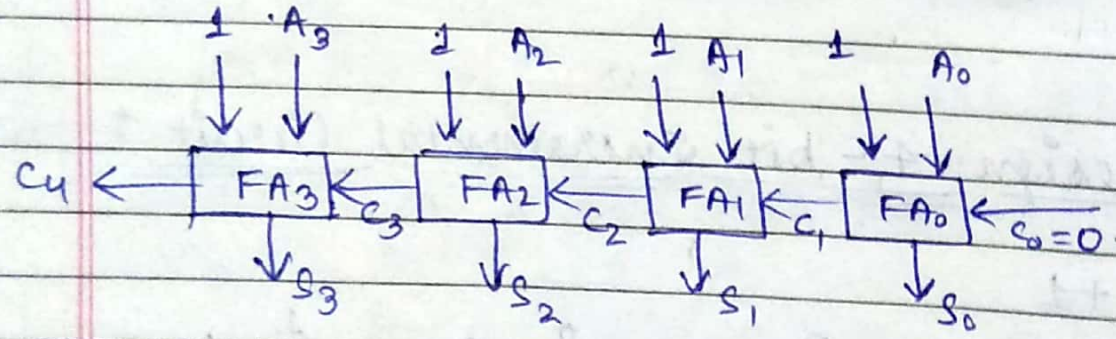
$A_3\ A_2\ A_1\ A_0$

$+\ 1\ 1\ 1\ 1$

1 1 1 0 (B

\+ 1

1 1 1 1



Types of Logic Microperation :-

1. OR $F \leftarrow A \vee B$
2. AND $F \leftarrow A \wedge B$
3. NOT (Inverter) $F \leftarrow \bar{A}$
4. NAND $F \leftarrow (A \wedge B)'$
5. XOR $F \leftarrow (A \oplus B)$
6. XNOR $F \leftarrow (A \oplus B)'$
7. Transfer (Buffer) $F \leftarrow A$
8. NOR $F \leftarrow (A \vee B)'$

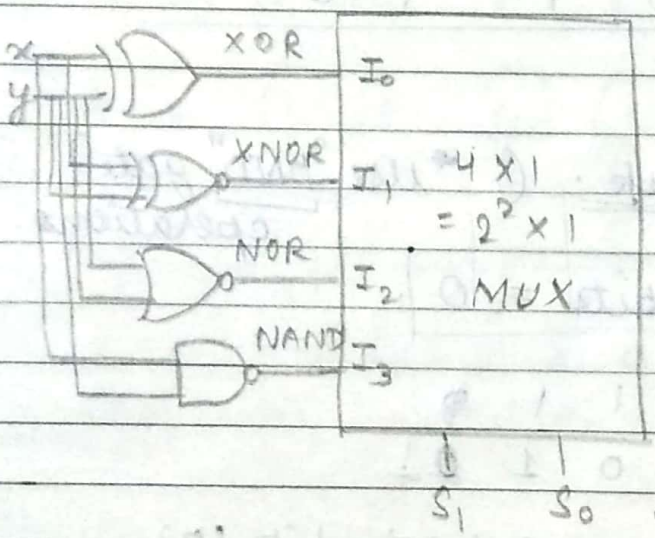
Date

9. All clear (0's) $F \leftarrow 0$

10 All set (1's) $F \leftarrow$ all 1's

Ques. Design a digital circuit that performs all logic operation of exclusive OR (XOR), exclusive NOR (XNOR), NOR, NAND. Use two selection variables.

Sol.



S_1	S_0	F (Micro op)
0	0	XOR
0	1	XNOR
1	0	NOR
1	1	NAND

Application of logic microoperation :

★ **Selective Set** (use "OR Gate" operation.)

A = 1 0 1 1.

result = 1 1 1 1.

Sol. Selective bit (1)

B = 0 1 0 0

A = 1 0 1 1

B = 0 1 0 0

1 1 1 1. H.P.

✱ Selective Complement (use "XOR" gate operation.

A = 1 0 1 1

result 0 1 1 0

Selective bits (1) | B = 1 1 0 1.

Sol.

A = (1) (0) 1 (1)

⊕ 1 1 1

0 1 1 0

✱ Selective mask. (use "AND" gate operation.

Selective bits = 0.

A = 1 1 1 1

result 0 0 1 1

Selective bits (0)

Sol.

A = (1) (1) 1 (1)

∩ 0 0 1 0

0 0 1 1

B = 0 0 1 1

✱ Selective clear (use "A ∩ $\bar{B}$" operation).

A = 1 1 0 1

result = 0 1 0 0

Date

Selective bits (1)

$A = (1)\ 1\ 0\ (1)$
$B = 1\ 0\ 0\ 1$
$B' = 0\ 1\ 1\ 0$
$(A \wedge B') = 0\ 1\ 0\ 0$

Ques. Register A holds the 8-bit binary value 1 1 0 1 1 0 0 1. Determine the B operand & the logic microoperation to be performed to change the value of A to 0 1 1 0 1 1 0 1

$A = (1)\ 1\ (0)\ (1)\ 1\ (0)\ 0\ 1$
$\oplus\ B = 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0$
$0\ 1\ 1\ 0\ 1\ 1\ 0\ 1$

$B = 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0$
Operation = XOR.

Ques.

$A = 1\ (1)\ (0)\ (1)$
$\oplus\ B = 0\ 1\ 1\ 1$
$1\ 0\ 1\ 0$

XOR gate

* **Use clear logic microoperation:**

(Use "XOR" gate)

$A = 1\ 0\ 1\ 1$
result $= 0\ 0\ 0\ 0$

$A = 1\ 0\ 1\ 1$
$\oplus\ B = 1\ 0\ 1\ 1$ (same as A)
$0\ 0\ 0\ 0$

Date

Ques. Register A holds the value 1 1 0 1 1 0 0 1. Find B, if result is 1 1 1 1 1 1 0 1.

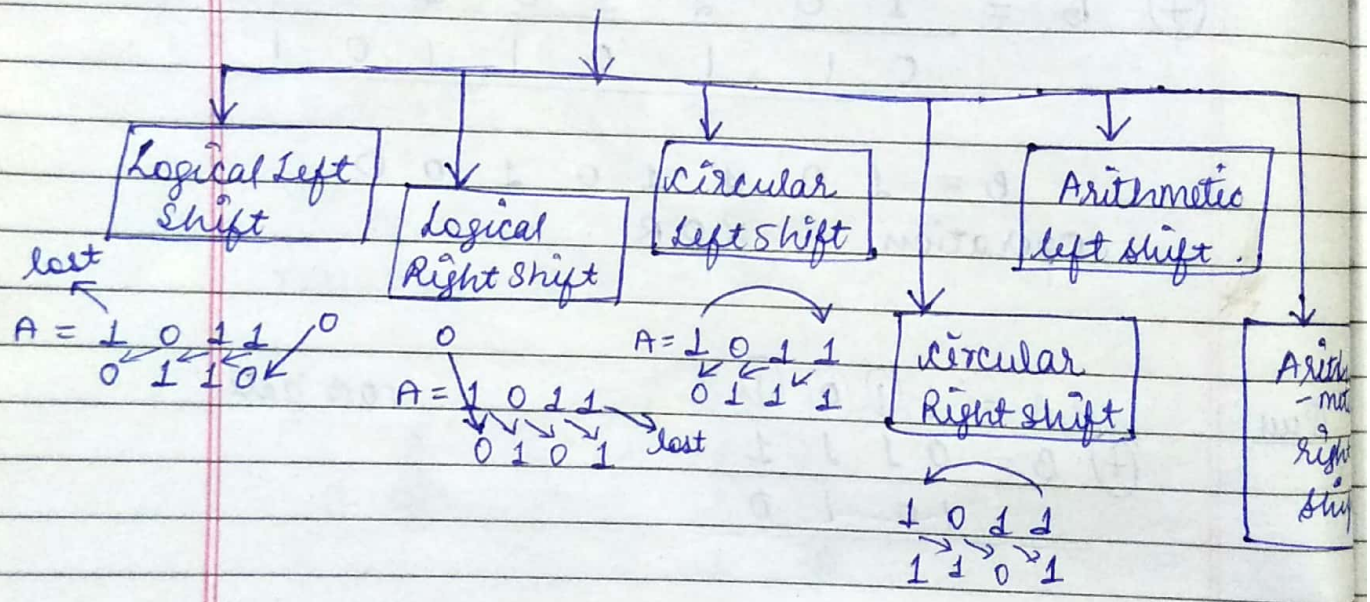
Sol.

$$\begin{array}{r} 1\ 1\ \textcircled{0}\ 1\ 1\ \textcircled{0}\ 0\ 1 \\ +\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1 \Rightarrow B \\ \hline 1\ 1\ 1\ 1\ 1\ 1\ 0\ 1 \end{array}$$

using selective set microoperation

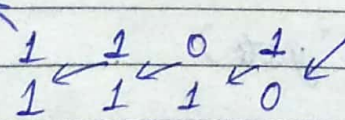
B = 0 0 1 0 0 1 0 1.

Shift micro-operation :

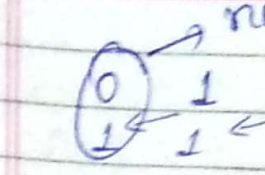


Arithmetic Left shift.

1.

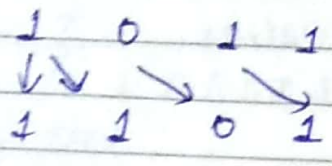


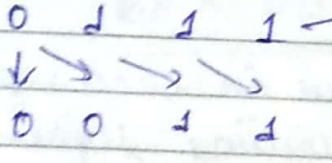
Date

2. 

↳ 0 1 1 0

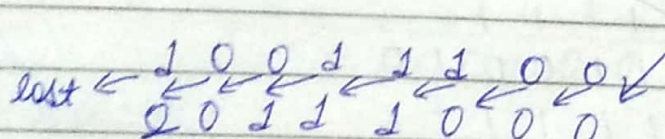
★ Arithmetic Right Shift.

1. 

2. 

Ques.1 An 8 bits register contains value 10011100. Determine the register value after arithmatic shift left, followed by circular right shift, followed by logical left shift.

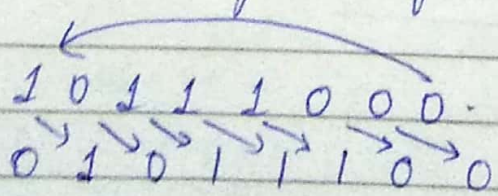
Sol. 1. Arithmetic left shift.



By sign reversal.

1 0 1 1 1 0 0 0.

2. Circular Right shift.



Result = 0 1 0 1 1 1 0 0

Date _____

3. Logical Left Shift.

lost ← 0 1 0 1 1 1 0 0 ← 0

1 0 1 1 1 0 0 0

Result = 10111000.

Qus. 2. An 8 bits register AR, BR, CR, DR initially have the following values.

AR = 11110010

BR = 11111111

CR = 10111001

DR = 11101010.

Determine 8 bits value in each register after the execution of following sequence of micro-operations.

① AR ← AR + BR (Add AR to BR)

② CR ← CR ∩ DR, ③ BR ← BR + 1 (AND CR & DR, increment BR)

④ AR ← AR - CR (Subtract CR from AR).

Sol. ① AR = 11110010

BR = 11111111

① 11110001

negated.

AR = 11110001

Date

(2) CR = 10111001

AND DR = 11101010

10101000

CR = 10101000

(3) BR = 11111111

\+ 1

① 00000000

neglected.

BR = 00000000

(4) AR = 11110001

− CR = 10101000

AR − CR = AR + CR' + 1.

AR = 11110001

CR' = 01010111

\+ 1

① 01001001

neglected.

AR = 01001001

Final Ans :-

AR = 01001001

BR = 00000000

CR = 10101000

DR = 11101010.

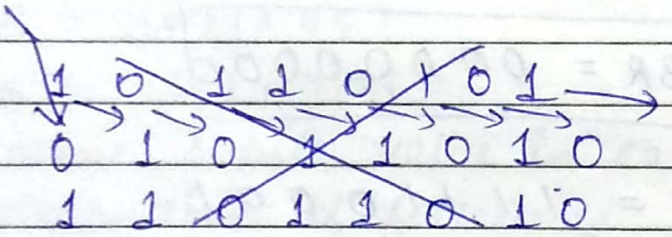
Date

Ques. 3. An 8-bit register A has value 10110101. Find the value of register A after each operater.

(1) Arithmetic right shift followed by
(2) Logical right shift followed by
(3) Circular left shift followed by
(4) Arithmetic left shift followed by
(5) Circular right shift ~~followed~~

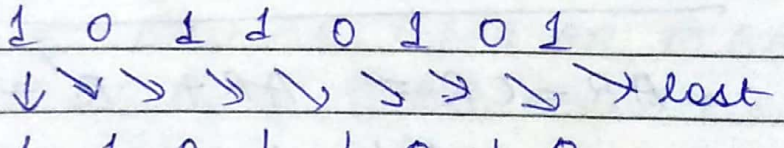
Sol. A = 10110101.

(1)

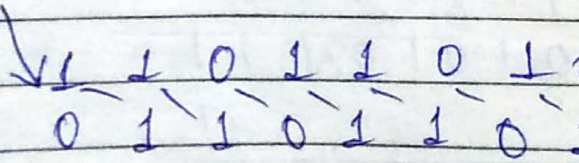


~~(2)~~

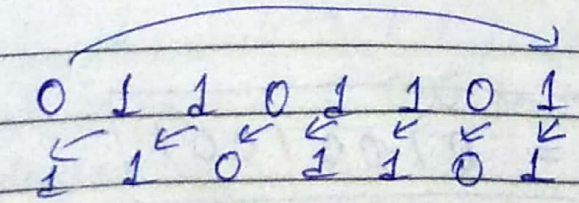
(1)



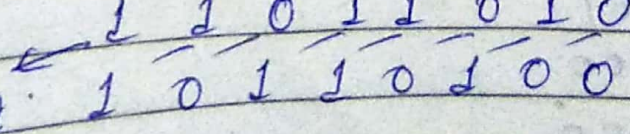
(2)



(3)

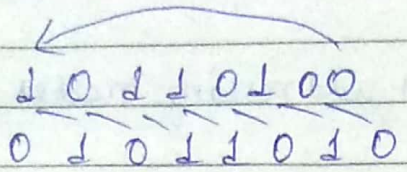


(4)



Date

(5.)



Instruction Code :

instruct computer to perform specific operation.

A group of bits used in the instruction, used to specify what to do.

called as macroopration as specifies set of micro-operations

1. Operation Code :- Specifies the operation to be performed by that instruction such as add, subtract, shift, complement, multiply.
2. Address Field(s) :- On what operation should be performed either stored in memory address or in register address. ~~it specifies the rule for interpreting or modifying address~~
3. mode field → it specifies how to access the address of operands.

Instruction format :-

mode	Op code	address field(s)

Ques. 1. The memory unit of a computer has 256 K words of 32 bit each. The computer has an instruction format with 4-fields, ① an op code field, ② a mode field to specify one of 7 addressing modes ③ a register address field to specify one of 60 processor register and ④ a memory address. Specify the instruction format & the no. of bits in each field if the inst^n is in 1 memory word.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1K	512	256	128	64	32	16	8	4	2	1

Date

Sol. (ii) mode field - 7 addressing modes = 2^3 = 3 bits

(iii) Reg. add. field - 60 processor reg = 2^6 = 6 bits

(iv) Memory add. = 256K = $2^8 \times 2^{10}$

= 2^{18} = 18 bits

5	3	6	18
OP code	mode field	Reg add field	mem add

← 32 bits →

(i) OP code = 32 - (3+6+18)

= 32 - 27

= 5 bits.

Types of Instructions :-

It is Based on CPU organization.

CPU Organization

- Single Accumulator organization
- General Register organization
- stack Organization

1. Single Accumulator Organization :-

* In each operation there should be a use of single accumulator Organization. It is a specialized register itself
* It has one address field.
* denoted by AC → Accumulator Register.

3-address = has Three address field. It may be Reg. add field, memory add field

Date

2. General Register Organization :

- No specialised register.
- 2-address field or 3-address field.

3. Stack Organization :

- There are two operations performed i.e. push and pop (deletion)
- Work on LIFO.
- It has 0 address field.
- Storage 1. Memory stack : portion of main memory.
  2. Register stack : set of reg. work on concept of stack.

Ques 1 Solve the following arithmetic operation.
$X = (A+B) * (C+D)$ using
1) 3-address inst" ↓ memory add.
2) 2-address inst"
3) 1-address inst"
4) 0-address inst"

Sol 1.

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

2.

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

Date

MOV X, R1 [M[X] ← R1].

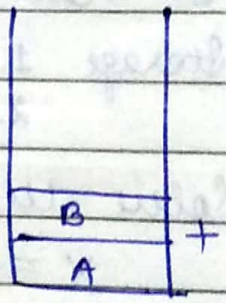
3.

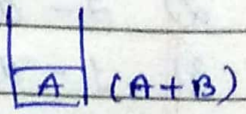
Accumulator Register ← AC.

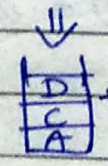
Load A	AC ← M[A].
ADD B	AC ← AC + M[B].
Store T	M[T] ← AC.
Load C	AC ← M[C].
ADD D	AC ← AC + M[D].
MUL T	AC ← AC * M[T].
Store X	M[X] ← AC.

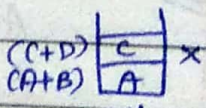
4.

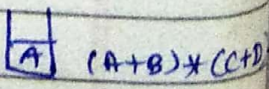
PUSH A	TOS ← A
PUSH B	TOS ← B
ADD	TOS ← A + (A+B)
PUSH C	TOS ← C
PUSH D	TOS ← D
ADD	TOS ← (C+D)
MUL	TOS ← (A+B) * (C+D)
POP X	M[X] ← TOS.











Date

8.12

Ques 2. WAP to evaluate the arithmetic statement

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

Sol. (1) 3 address Instⁿ

```
SUB  R1, A, B      [R1 ← M[A] - M[B]]
ADD  R2, R1, C     [R2 ← R1 + M[C]]
MUL  R3, D, E      [R3 ← M[D] * M[E]]
SUB  R4, R3, F     [R4 ← R3 - M[F]]
MUL  R5, R2, R4    [R5 ← R2 * R4]
MUL  R6, H, K      [R6 ← M[H] + M[K]]
ADD  R7, G, R6     [R7 ← M[G] + R6]
DIV  X, R7, R5     [X ← R5 / R7]
```

(2) 2 address Instⁿ

```
MOV  R1, A     [R1 ← M[A]]
SUB  R1, B     [R1 ← R1 - M[B]]
MOV  R2, D     [R2 ← M[D]]
MUL  R2, E     [R2 ← R2 * M[E]]
SUB  R2, F     [R2 ← R2 - M[F]]
MUL  R2, C     [R2 ← R2 * M[C]]
ADD  R1, R2    [R1 ← R1 + R2]
MOV  R3, H     [R3 ← M[H]]
ADD  R3, G     [R3 ← R3 + M[G]]
MUL  R3, K     [R3 ← R3 * M[K]]
DIV  R1, R3    [R1 ← R1 / R3]
MOV  X, R1     [M[X] ← R1]
```

Date

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

3. 1 adder Instruction :-

LOAD A	[AC ← M[A]]
Sub B	[AC ← AC − M[B]]
Store T	[M[T] ← AC]
Load D	[AC ← M[D]]
MUL E	[AC ← AC * M[E]]
SUb F	[AC ← AC − M[F]]
MUL C	[AC ← M[C] * AC]
Store Add T	[AC ← AC + M[T]]
Store T	[M[T] ← AC]
Load H	[AC ← M[H]]
MUL K	[AC ← AC * M[K]]
Add G	[AC ← AC + M[G]]
~~Store T1~~	~~[M[T1] ← AC]~~
~~Load T~~	~~[AC ← M[T]]~~
Div T1	[AC ← AC / M[T1]]
Store X	M[X] ← AC

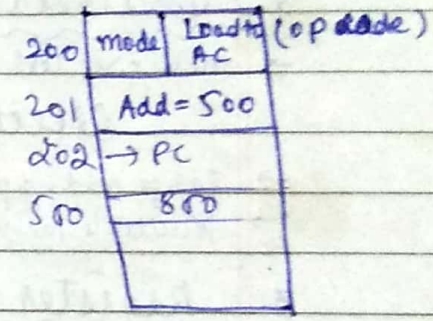
4. 0 adder Inst" :-

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))

Date

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.

Ques. PC = 202 (Program Counter)
R_1 = 400 (Processor Register)
AC (Accumulator)
XR = 100 (Index Register)



Sol. (i) Direct = EA = M[A] = 500

(ii) Indirect = EA = M[A] = M[500] = 800.

(iii) Immediate = EA = 201

(iv) Register direct = EA = R_1

(v) Register indirect = EA = 400.

(vi) Relative = PC + Add. = 202 + 500 = 702.

(vii) Index = Index + Add. = 100 + 500 = 600

(viii) Autoincrement = 400 (Reg. value)

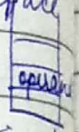
(ix) Autodecrement = 399 (Reg. value)

Date

Addressing Mode :-

It specifies the different ways of determining the effective (actual) address of an operand.

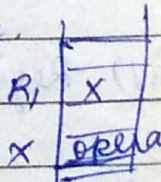
1. **Direct Addressing Mode :**
   → Add. field contain add. of operand → limited add. space
   Effective address = address part
   = M[X] → no additional calculation to work out effective add.
   Single memory reference to fetch data



2. **Indirect Addressing Mode :**
   → memory cell pointed to add. field contains add. of operand.
   Effective address = ~~address part~~
   = M[Y]
   → large add. space
   → Multiple memory accesses to find operand. → 2^n where n = word length.

3. **Register Addressing Mode :**
   → operand is held in register named in add. field
   Effective address = R1
   → limited no. of registers → very small add. field.
   → fast execution → no memory access

4. **Register Indirect Addressing Mode :**
   → fewer memory access than indirect addressing.
   Effective address = X = M[R1]
   → large add. space (2^n)
   → operand is in memory cell pointed to contents of register.



5. **Immediate Addressing Mode :**
   → operand is given in the instruction itself.
   → operand = value → fast → limited range → no memory reference to fetch data.

6. **Implied Addressing Mode :**
   Example - CMA (Complement the content of Accumulator).

Date

operand = Accumulator.

Effective Address = Accumulator.

(7) Relative Addressing mode.

$$EA = PC + \text{Address part}$$

PC = Program Counter (next to inst^n)

(8) Index Addressing mode.

$$EA = \text{Index Register} + \text{Address part}$$

$$EA = X + M[R]$$

(9) Base Register Addressing mode.

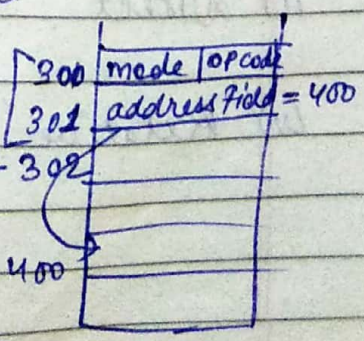
$$EA = \text{Base Register} + \text{Address part}$$

Ques.1 An Instruction is stored at Location 300 with its address field at Location 301. The address field has value 400. A processor register R_1 contains the value 200.
Determine effective address for.

8.18

1. Direct Addressing.
2. Immediate Add.
3. Relative Add.
4. Register Indirect Add.
5. Index addressing with R_1 as index register.
6. Indirect
7. Register direct.

Sol. $R_1, \#200$



1. EA (Direct add.) = 400.
2. EA (Immediate) = 301.
6. EA (Indirect) = M[400]

3. Relative Add.

EA = PC + Add. part.
= 302 + 400
= 702

4. Register Indirect.

EA = M[R,] = 200.

5. Index Addressing.

EA = Index + Add. part.
= 200 + 400.
EA = 600

7. Register direct.

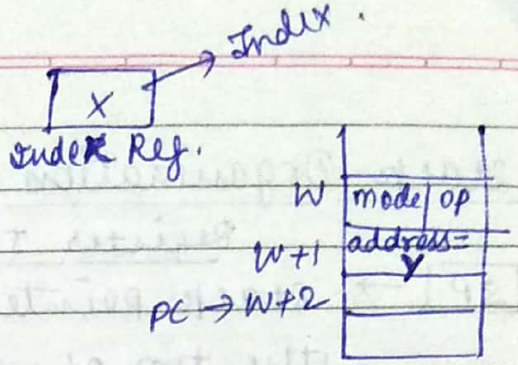
EA = R,.

Ques.2. An instruction is stored at memory address W. The address field is stored at address W+1 is designated by symbol Y. The operand used is stored at address Z. An index register contains the value X. Determine how Z is calculated using addressing modes.

(i) Direct (ii) Indirect (iii) Register (iv) Regi Indir

(v) Relative (vi) Index (vii) Immediate.

Date _____



Sol.

(i) EA = Address field
= Y

(ii) EA = M[Y].

(iii) EA = Index Register

(iv) EA = X

(v) Relative

EA = PC + Add.
= W+2 + Y
= W + Y + 2

(vi) Index.

EA = Index Reg + Add.
= X + Y

(vii) Immediate

EA = W+1.