COMPUTER ORGANIZATION AND ARCHITECTURE

[5 credits]

Book: Computer System Architecture by Morris Mano.

Internal
- Class test – 10
- Attendance – 5
- Home assignment – 8
- VIVA – 7

* DIGITAL COMPUTERS — It's a digital system that performs various computational tasks and the information is represented by the variables that take unlimited no. of discrete values. It works on binary no. system i.e. 2 digits, 0 and 1.

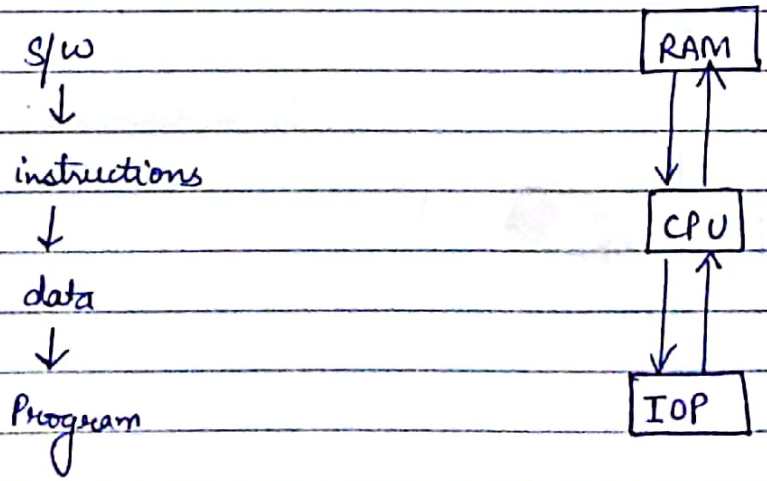
Q.1 $(1001011)_2 \longrightarrow (\ )_{10}$

$= 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 75$

Assignment-1 A set of conversions.

* HARDWARE — consists of all electronic and electromechanic devices.

* SOFTWARE — instructions and data that the computer manipulate to process various tasks.



* COMPUTER ORGANIZATION — operation of different components to work together and how they are connected.

* COMPILER ARCHITECTURE
- instruction format
- instruction set
- different addressing modes for addressing the memory

* COMPUTER DESIGN
- h/w design
- specifications

→ once specifications are formulated, it's the task of the designs for its ~~specif~~ implementation.
→ designs to design the system according to our specifications.

* LOGIC GATES — basic digital components of circuits. Gates are the basic building blocks of h/w that produces the signals of binary 1 or 0 when input logic requirements are satisfied.

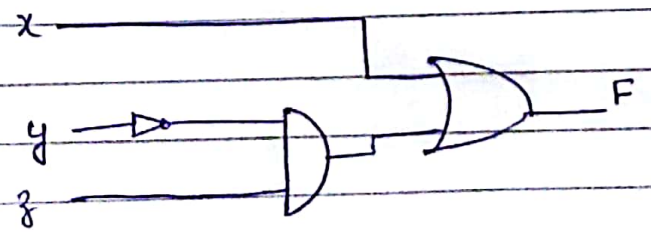
XOR — $A \oplus B = \bar{A}B + A\bar{B}$

XNOR — $\overline{A \oplus B} = AB + \bar{A}\bar{B}$

A	B	XOR	XNOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

* BOOLEAN ALGEBRA —
It deals with binary variables and logic operations.

$F = x + y'z$



components
ted.

des for
y

of the

our

. Gates
duces the
uirements

XOR	XNOR
0	1
1	0
1	0
0	1

18/7/18

x	y'	z	F
0	1	0	0
0	1	1	1
0	0	0	0
0	0	1	0
1	1	0	1
1	1	1	1
1	0	0	1
1	0	1	1

* IDENTITIES —

1) $X+0 = X$
2) $X+1 = 1$
3) $X+X' = 1$
4) $X.X' = 0$
5) $X+Y = Y+X$
6) $X.X = X$
7) $X(Y+Z) = XY+XZ$
8) $XY = YX$
9) $(X')' = X$
10) $(X+Y)' = X'Y'$
11) $(XY)' = X'+Y'$

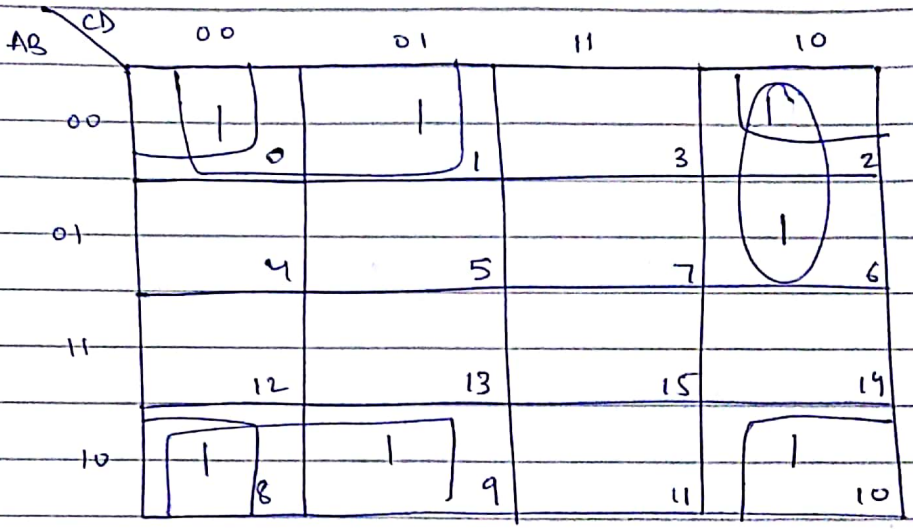
* K-MAP —

① $F = (x,y,z) = \Sigma(1,4,5,6,7)$

x \ yz	00	01	11	10
0	0	1	3	2
1	1 (4)	1 (5)	1 (7)	1 (6)

$F = x + \bar{y}z$

② $F(A,B,C,D) = \Sigma(0,1,2,6,8,9,10)$



$F = B'C' + B'D' + A'CD'$

* CIRCUITS ——— Sequential

→ if we want any storage element of the data to be included.

→ storage element is called flip flop.

Combinational

→ connected arrangement of logic gates with a set of inputs and outputs.

→ output is dependent on the input.

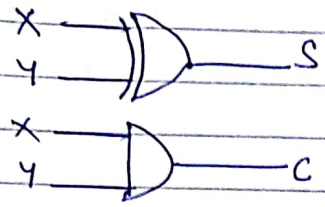
* HALF ADDER – addition of 2 bits (not considers carry)

{ full adder – addition of 3 bits }

19/7/18

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

$C = xy$

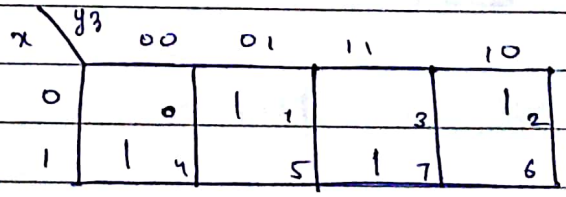


* FULL ADDER —

x	y	z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz = x \oplus y \oplus z$

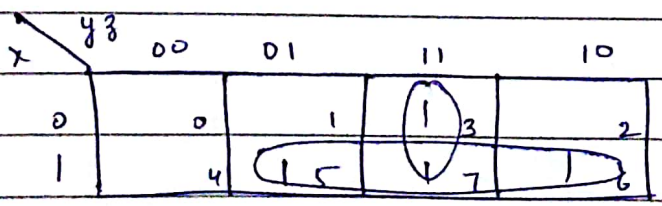
or through K-Map ;

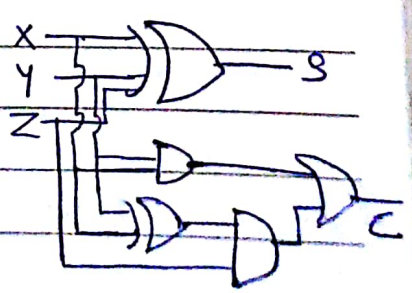


$C = \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz$

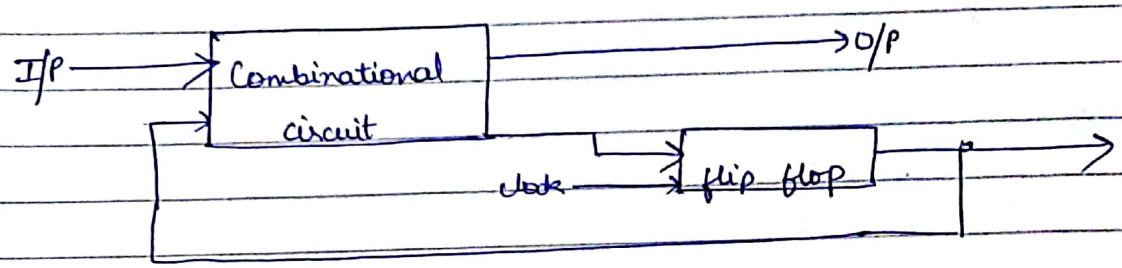
$= xy + xz + yz = xy + z(x+y)$

$= xy + z[x(y+y') + y(x+x')] = xy + z(x \oplus y)$





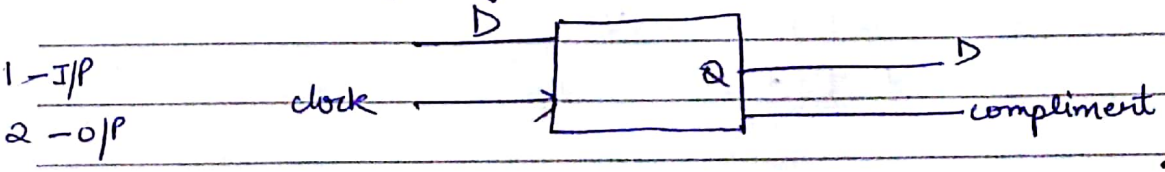
* SEQUENTIAL CIRCUITS – The storage element employed in the clocked sequential circuits is called flip flop. It's a binary cell capable of storing 1 bit of information. It has 2 outputs : 1 for normal value and 1 for the compliment value of bit stored in it.



Any combinational circuit receives binary signal from external I/P and from the O/P of the flip flops. The O/P of the combinational circuit go to external O/P and to the I/P to the flip flop. The gates in the combinational circuit determines the binary values to be stored in flip flop after each clock transition. The O/P of the flip flop in turn are applied to combinational circuit I/P and determines the circuit behaviour.

* TYPES OF FLIP-FLOP –

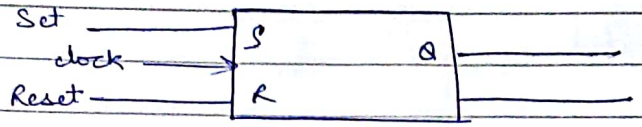
① D FLIP FLOP (Delay) –



D	Q(t+1)
0	0
1	1

② SR FLIP FLOP –

['→' : dynamic I/P]



(flip flop responds at the time of transition.)

If there's no signal at clock I/P, O/P of the circuit can't change irrespective of the values S and R. Only when the clock signal changes from 0 to 1 can the O/P be affected according to the values of S and R.

If S = 1 & R = 0 and clock also makes a transition from 0 to 1 then Q is set to 1.

S	R	Q(t+1)	
1	0	1	(set to 1)
0	1	0	(clear to 0) clock transition +ve
0	0	no change	(+ve " ")
1	1	unpredictable	(may go to 0 or 1)

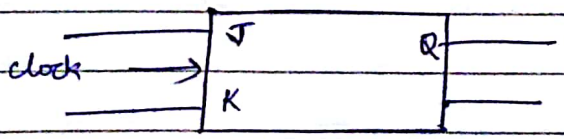
Q(t+1) = binary state of O/P after occurence of clock transition
Q(t) = binary state of O/P before transition.

(3) JK FLIPFLOP — refinement of SR flipflop in that the indeterminate condition of SR is defined in JK type.

O/P = Q'(t)

clock transition switches the O/P to its compliment state.

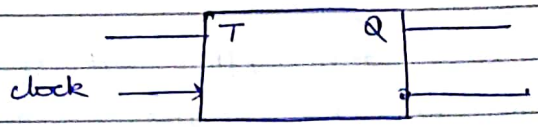
i.e. Q(t+1) = Q'(t)



J	K	Q'(t)
1	0	
0	1	
0	0	
1	1	compliment the state (inverting I/P)

(4) T FLIP FLOP (Toggle) –

1 – I/P



	T	Q(t+1)
(J=K=0)	0	no change
(J=K=1)	1	compliments state of flipflop

→ Clock transition doesn't change the state of flipflop.

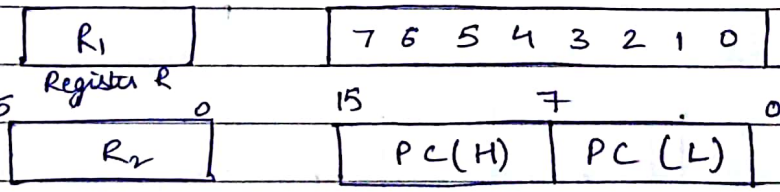
24/7/18

MODULE -I REGISTER TRANSFER LANGUAGE

* REGISTERS — temporary storage in a CPU that holds the data the processor is ~~holding~~ currently working on while RAM holds the program instructions and the data the program requires.

RAM is located external to the CPU whereas registers are storage locations internal to the processor. Registers have size that determine the max. amount of data that can be processed at a time.

Memory Address Register (MAR)
Program Counter (PC)
Instruction Register (IR)
Processor Register (R_1)

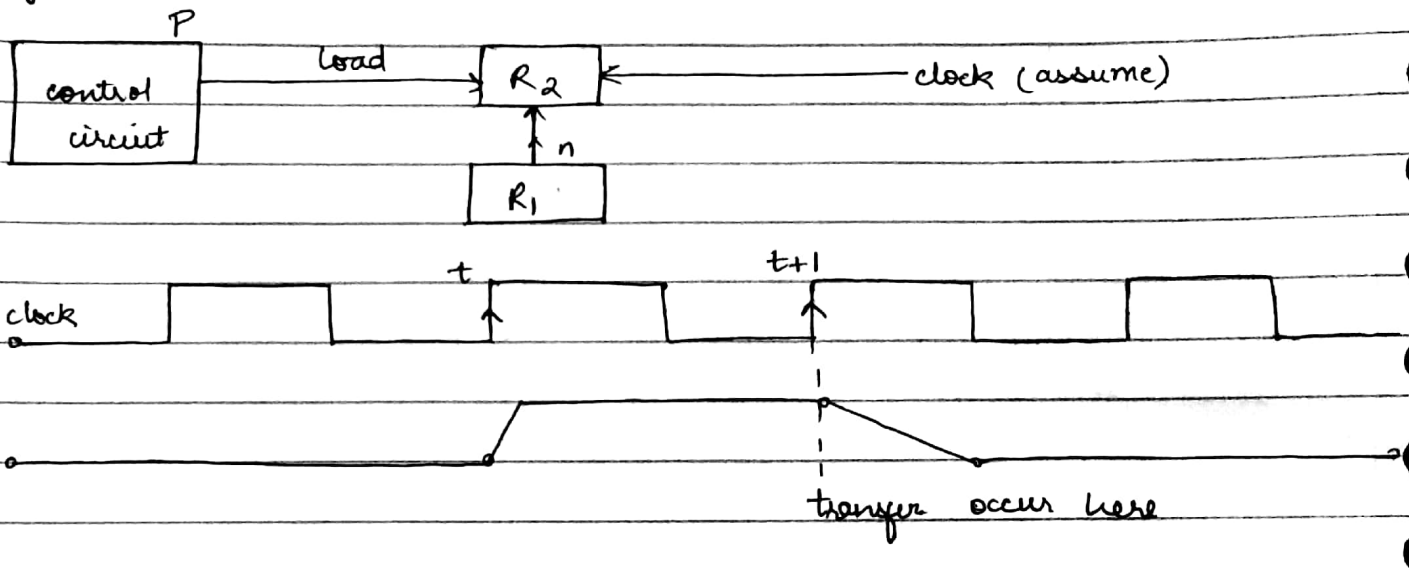


A register is a group of flipflops with each flip flop capable of storing 1 bit of information.

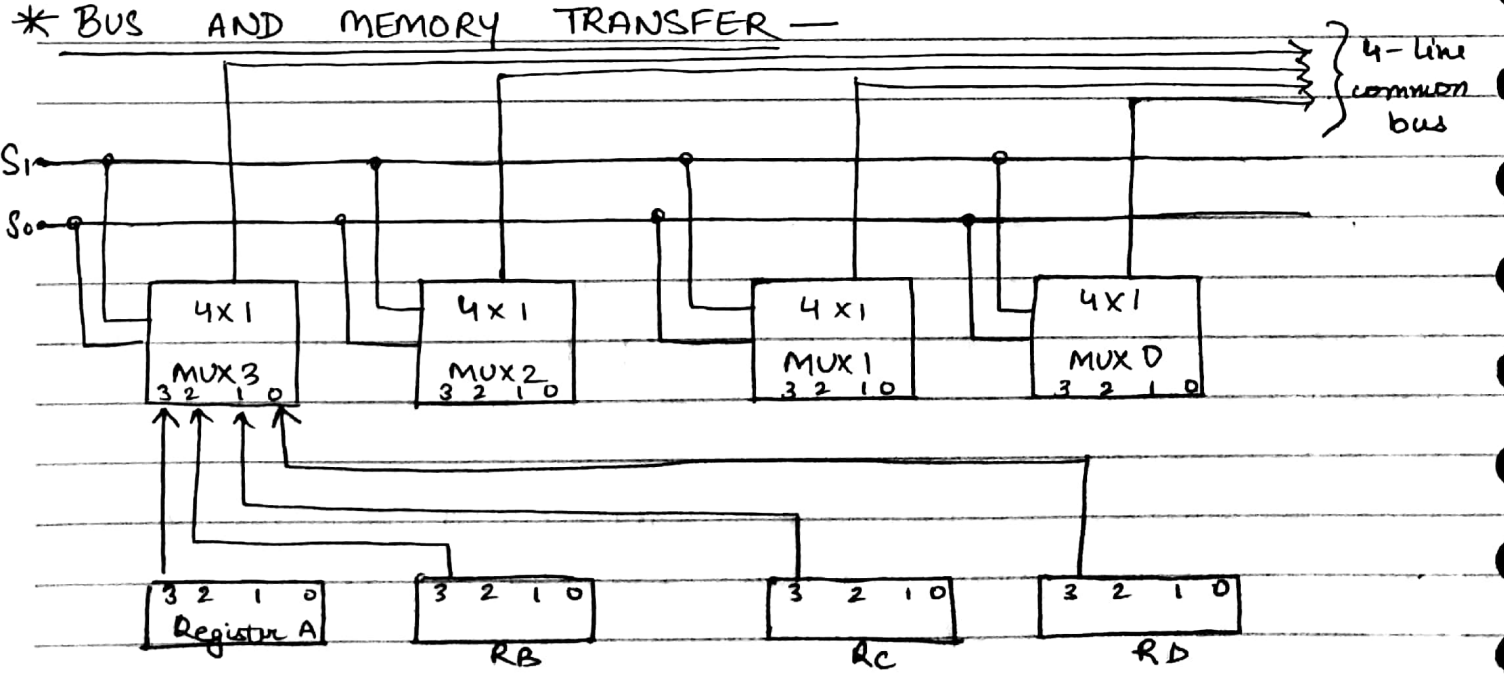
The operations executed on data stored in registers are called <u>microoperations.</u> (temp. processing of data)
Pentium chips — 32 bits
Eg: shift, clear, load, count.

$R_2 \leftarrow R_1$
(Subscript denotes bits)
This statement denotes transfer of contents of R_1 into R_2.

Eg:- If (P=1) then ($R_2 \leftarrow R_1$) or P: $R_2 \leftarrow R_1$



* BUS AND MEMORY TRANSFER —



[no. of bits of registers = no. of multiplexes.]

$\Rightarrow R_2 \leftarrow R_1$

Bus $\leftarrow R_1$

$R_2 \leftarrow$ Bus

A digital computer has many registers and path must be provided to transfer information from one

register to another. The efficient scheme for transfering information in multiple register configuration is a common bus system.

A bus structure consists of set of common lines — one for each bit of register through which binary information is transferred one at a time.

Two selection lines S_0 & S_1 are ~~collected~~ connected to selection inputs of multiplexes which chooses 4 bits of one register and transfer them into the 4 line common bus.

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

MEMORY TRASFER —

Memory Read — Read : DR ⟵ M(AR) [AR: address register]

Memory write — Write : M[AR] ⟵ R_1

In memory read, transfer of information into data register from memory selected by the address in AR.

Transfer of information from R_1 into memory word M selected by the address in AR.

* ARITHMETIC OPERATIONS —

4 categories

1) Register Transfer Microoperation
2) Arithmetic microop^n
3) Shift microop^n
4) Logic microop^n

$R_3 \leftarrow R_1 + R_2$ $\qquad R_2 + \overline{R_2} = 1$

same
- $R_3 \leftarrow R_1 - R_2$
- $R_3 \leftarrow \overline{R_3}$
- $R_3 \leftarrow \overline{R_3} + 1$
- $R_3 \leftarrow R_1 + \overline{R_2} + 1$

$R_1 \leftarrow R_1 + 1$

$R_1 \leftarrow R_1 - 1$

Q.1 Subtract $(100011)_2$ from $(010010)_2$ using 2's complement method.

Q.2 $(01110)_2$ from $(10101)_2$. $(00111)_2$

Soln.1 $(100011)_2 = 35$ $\qquad (010010)_2 = 18$

$-17 \Rightarrow (-010001)_2$

$$\begin{array}{r} 010010 \\ -\ 100011 \end{array} \Rightarrow \begin{array}{r} 010010 \\ +\ 011100 \\ \hline 101110 \end{array} \Rightarrow (-010001)$$

same no of bits ∴ compliment

Soln.2

$$\begin{array}{r} 10101 \\ -\ 01110 \end{array} = \begin{array}{r} 10101 \\ +\ 10001 \\ \hline 100110 \end{array} \xrightarrow{+1} \Rightarrow 00111$$

(+ve as there's carry generated i.e. extra bit)

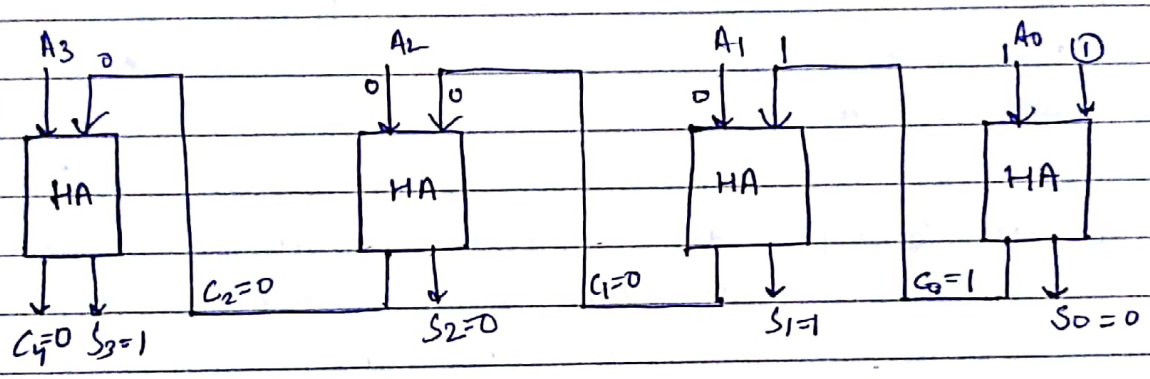
XOR gate with each full adder.

Subtraction = $\bar{R}_1 + \bar{R}_2 + 1$

XOR		
0	0	0
0	1	1
1	0	1
1	1	0

	B_3	B_2	B_1	B_0	
B	0	1	0	1	
XOR	1	1	1	1	M
$\bar{B}$	1	0	1	0	and M = 1

③ BINARY INCREMENTER –



A_3	A_2	A_1	A_0		
1	0	0	1	⇒	9
+			1		
1	0	1	0	⟹	10
2^3	2^2	2^1	2^0		

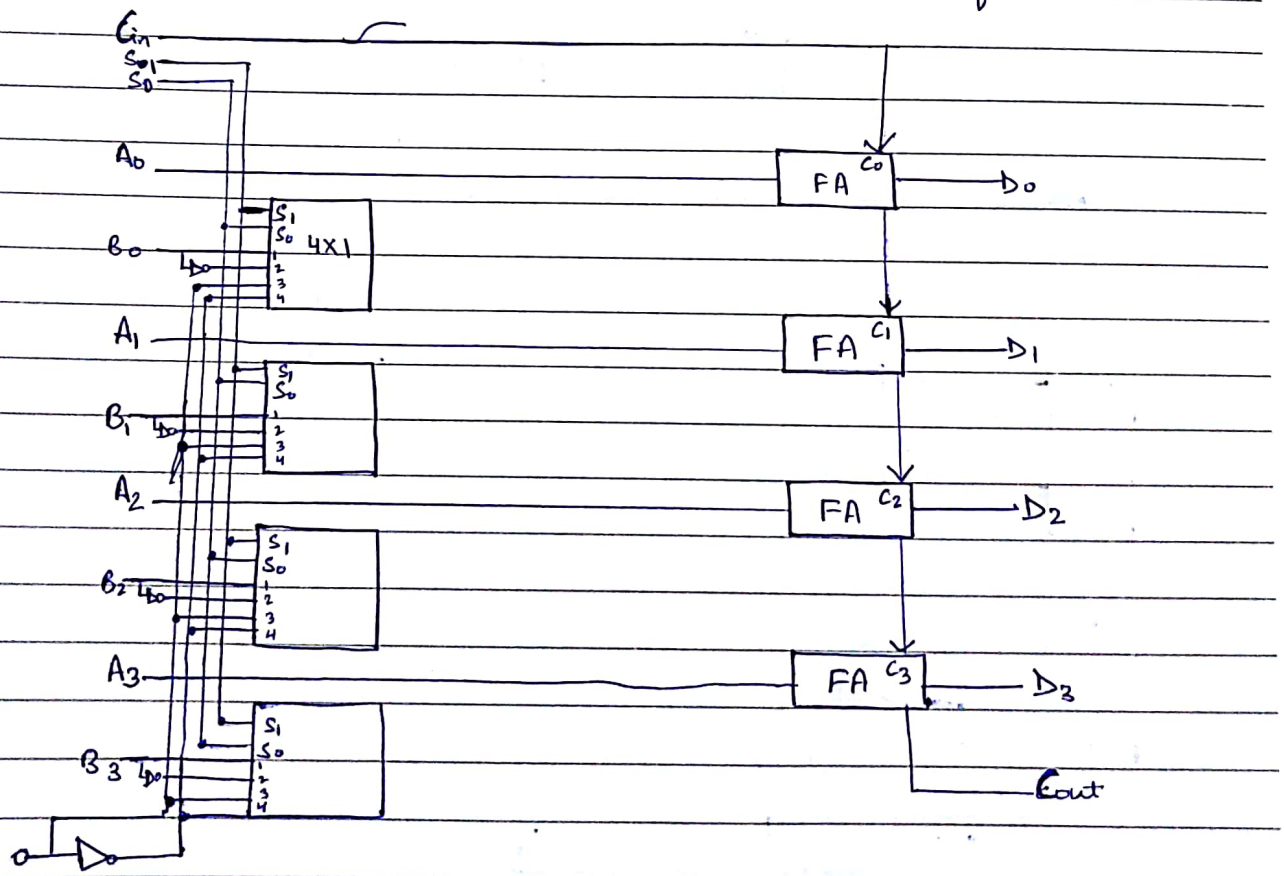
I/P = 1001

O/P = 1010

* 4-BIT BINARY ARITHMETIC CIRCUIT –

1/8/18.

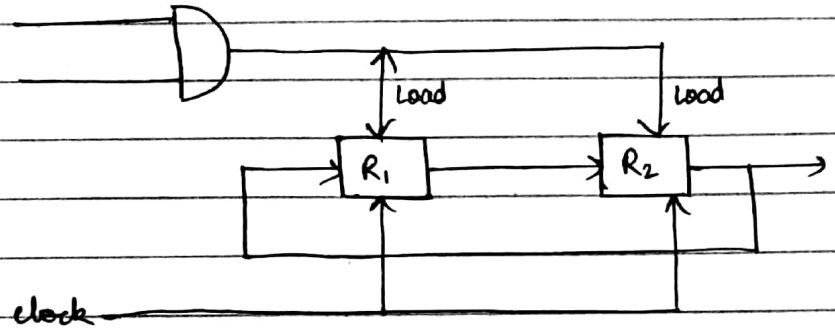
SELECT S_1	SELECT S_0	C_{in}	INPUT Y	OUTPUT	MICROOPERATION
0 (1st input)	0	0	B	$D = A + B$	Add
0 (1st input)	0	1	B	$D = A + B + 1$	Add with carry
0 (2nd input)	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0 (2nd input)	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1 (3rd input)	0	0	0	$D = A$	Transfer A
1 (3rd input)	0	1	0	$D = A + 1$	Increment A
1 (4th input)	1	0	1	$D = A - 1$	Decrement A.
1 (4th input)	1	1	1	$D = A$	Transfer A



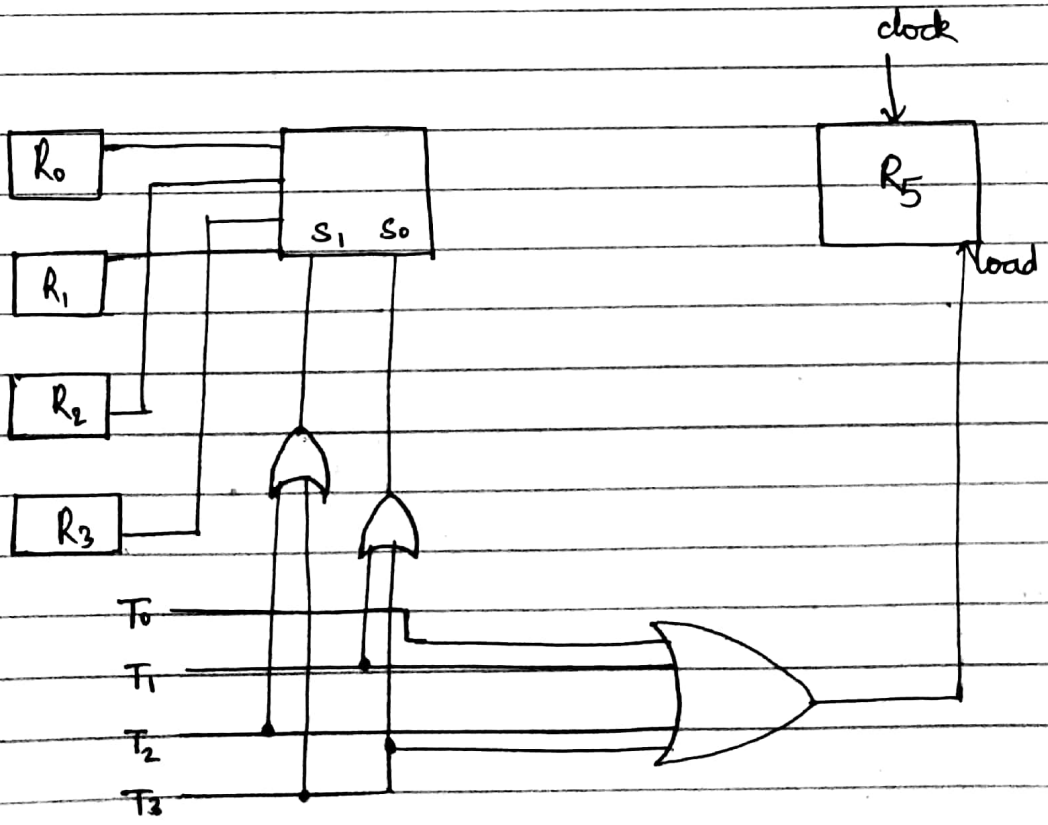
Q1 Block diagram.

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

Ans:



Ans.2



6/8/18

LOGIC MICROOPERATIONS

Boolean fn	Microopn	Name
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \bar{B}$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow \bar{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	Transfer B
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	Ex-OR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x+y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = (x \oplus y)'$		
$F_{10} = y'$		
$F_{11} = (x+y)'$		
$F_{12} = x'$		
$F_{13} = x' + y$		
$F_{14} = (xy)'$		
$F_{15} = y$	$F \leftarrow 1$	

16 combinations

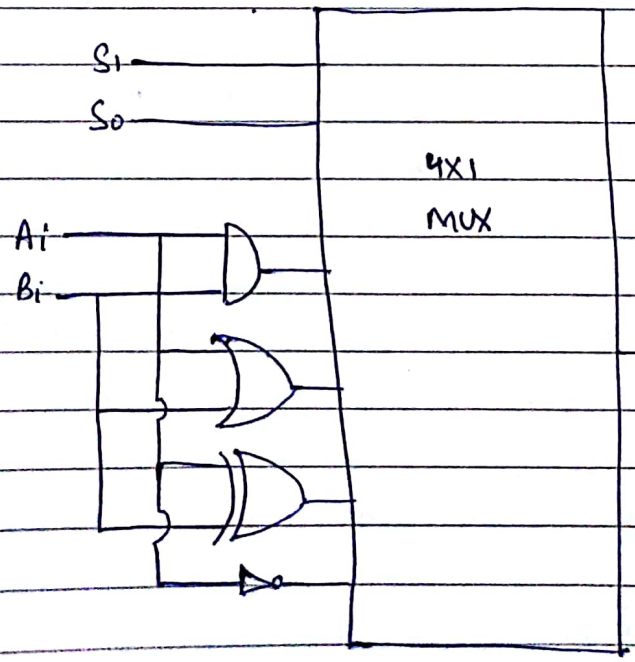
xy

4 logic microopn

xy → xy', x'y, (xy)'

x+y

x ⊕ y.



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

SELECTION SET (OR)

This op^n sets to 1, the bits of registers of A where there's corresponding 1 registers of B.

```
1010   A Before
1100   B
1110   A After
```

(where B → 1, A → 1)

SELECTIVE COMPLEMENT (EX-OR)

```
1010   A Before
1100   B
0110   A After
```

(where B → 1, A → A')

SELECTIVE CLEAR

```
1010
1100
0010
```

(where B → 1, A → 0)

MASK (AND)

Bits of A are cleared where there are corresponding 0's in B.

```
1010
1100
1000
```

[where B → 0, A → 0]

INSERT

This op^n inserts a new value into a group of bits by first masking the bits and then OR them with the req. value.

```
0110   1010   A Before
0000   1111   B (mask)
0000   1010   A after masking
```

insert new value 1001

```
0000   1010
1001   0000   + (OR)
1001   1010   A after insertion
```

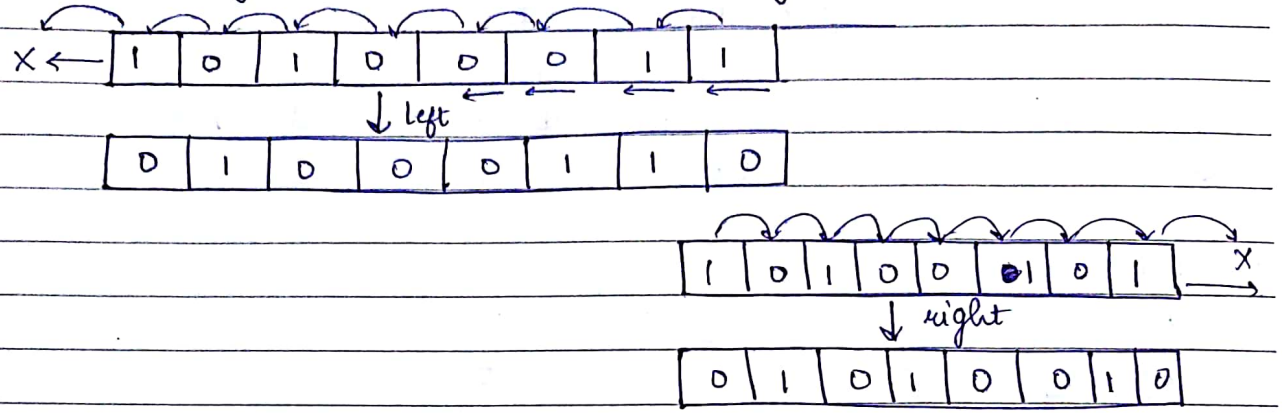
CLEAR

```
1010
1010
0000    A ← A ⊕ B
```

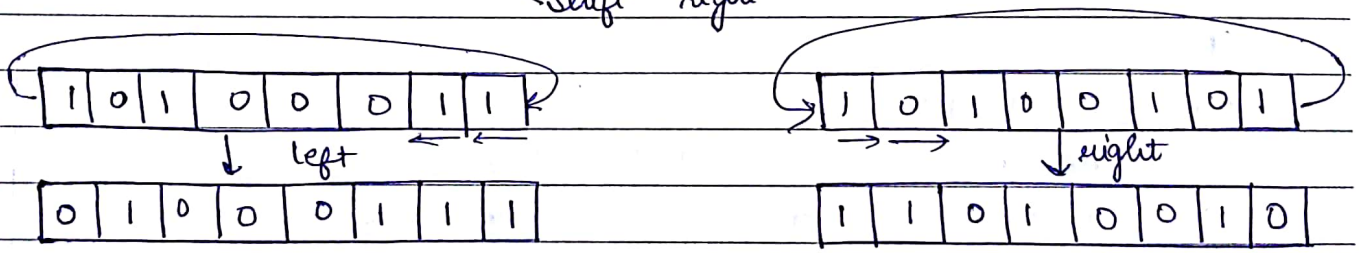
It compares the bits in A & B and produces an all 0 result if the two nos. are equal.

④ SHIFT MICROOPERATIONS –

① Logical Shift — Shift left / Shift right



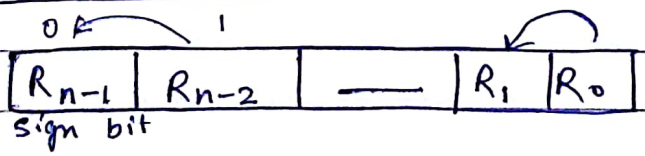
② Circular Shift — Shift left / Shift right



Q. R = 11011101

Logical shift left followed by : 10111010
~~Circular~~ Logical right shift : 01101110
Circular Shift right : 11011101
Circular left shift : 10111011

③ ARITHMETIC SHIFT –



Right: This is a ~~multiple~~ micro op^n whose ~~that~~ shifts right divides no. by 2.
Left: ~~If~~ multiplies a sign binary no. by 2.

AS left inserts a 0 at the right R_{n-1} is lost and replaced by the bit R_{n-2}.

Shift left divides by 2 => overflow (sign reversal)
Shift right multiplies by 2 =>

AS right leaves the sign bit unchanged and shifts the no. to the right.

Q.1 10011100
AS shift right followed by 11001110 ←
AS shift left : 10011100

Q.2 AR = 11110010
BR = 11111111
CR = 10111001
DR = 11101010

AR ← AR + BR (Add BR to AR)
CR ← CR ∧ DR , BR ← BR + 1 [AND ∧ increment. CR & DR
AR ← AR − CR (subtract CR from AR)

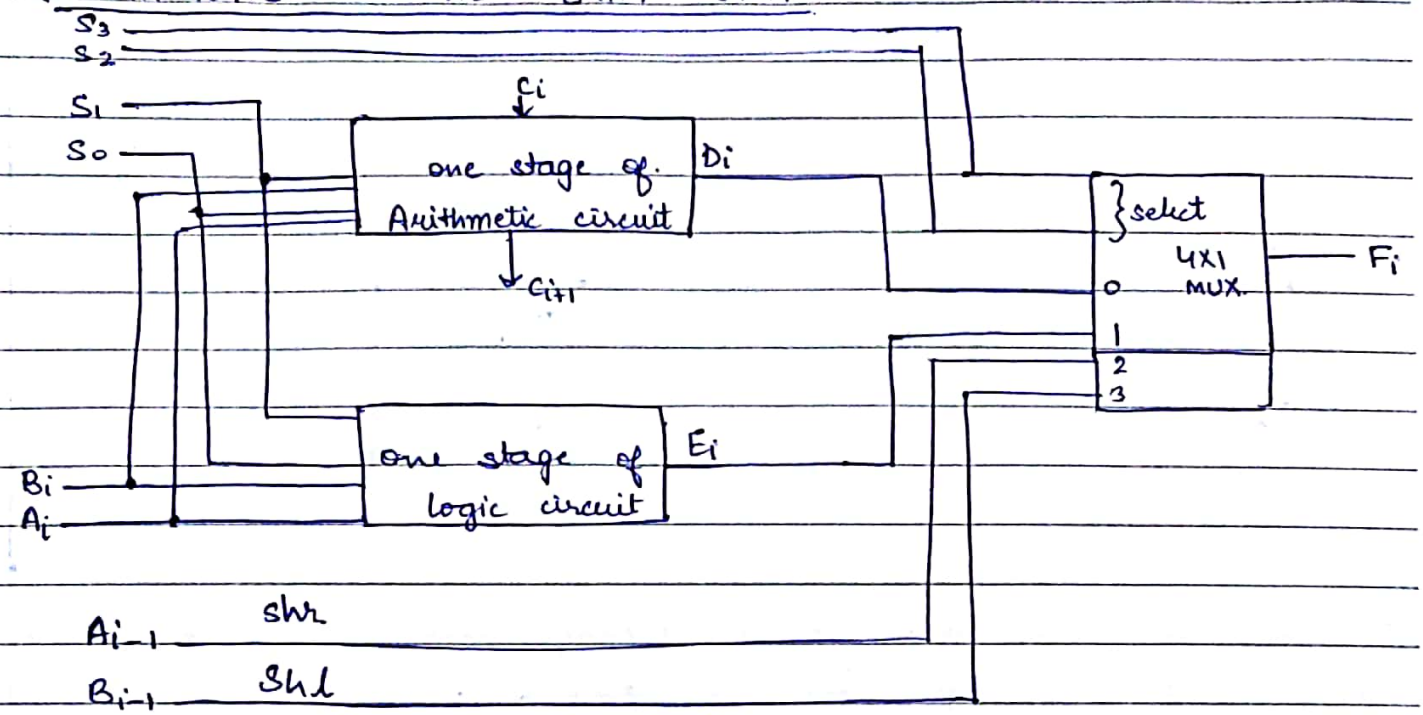
Ans (i) AR = 11110010
+ 11111111
(1) 11110001

(ii) CR = 10111001
DR = 11101010 AND
10101000

BR = 11111111
+1
(1) 00000000

(iii) AR − 11110001
CR − 10111001
00101000

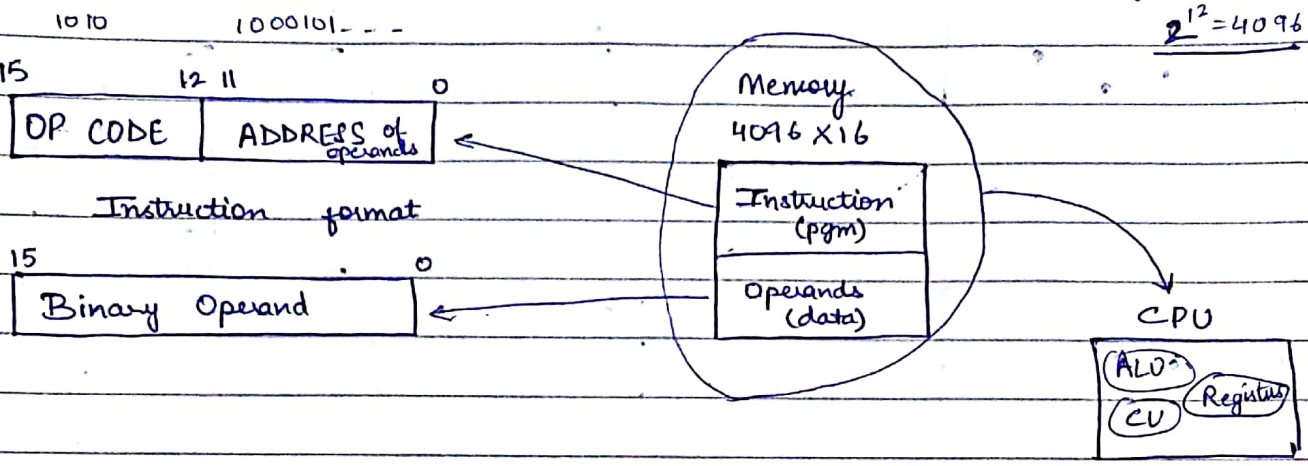
* ARITHMETIC LOGIC SHIFT UNIT –



	S_3	S_2	S_1	S_0	C_{in}	Op^n
Arithmetic Op^n	0	0	0	0	0	F = A
	0	0	0	0	1	F = A+1
	0	0	0	1	0	F = A+B
	0	0	0	1	1	F = A+B+1
	0	0	1	0	0	F = A+$\bar{B}$
	0	0	1	0	1	F = A+$\bar{B}$+1
	0	0	1	1	0	F = A−1
	0	0	1	1	1	F = A
Logic Op^n	0	1	0	0	X	F = A ∧ B
	0	1	0	1	X	F = A ∨ B
	0	1	1	0	X	F = A ⊕ B
	0	1	1	1	X	F = $\bar{A}$
	1	0	X	X	X	F = shr A
	1	1	X	X	X	F = shl A

MODULE - II BASIC COMPUTER ORGANIZATION AND DESIGN

* INSTRUCTION CODE —

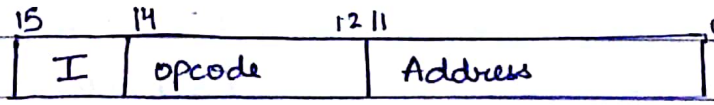


Eg:

$2^{12} = 4096$

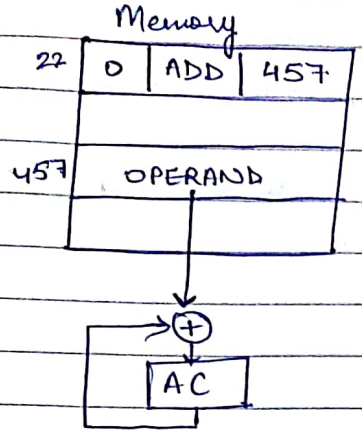
A computer instruction is a binary code that specifies a sequence of microop^n for the computer.
Instruction code together with data are stored in memory. The computer reads each instruction from memory and places it in control register. A control then interpreats the binary code and proceeds to execute by issuing sequence of microop^n. The op^n part of the instruction code specifies the op^n to be performed. The op^n must be performed on some data stored in processor registers or in memory.

* ADDRESSING —

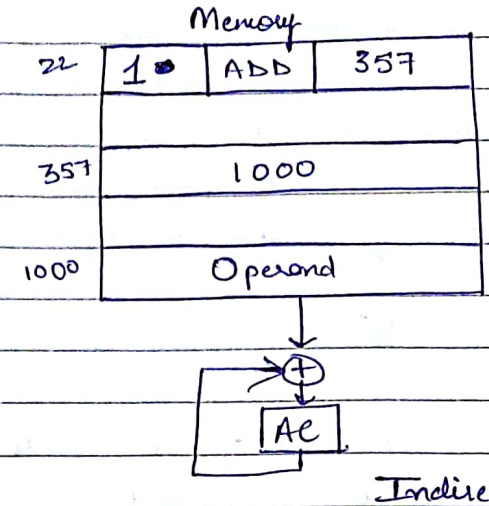


Instruction Format

When the address part of instruction code specifies operand, the instruction is ~~set~~ said an immediate instruction and it's a type of direct addressing.



Direct Address

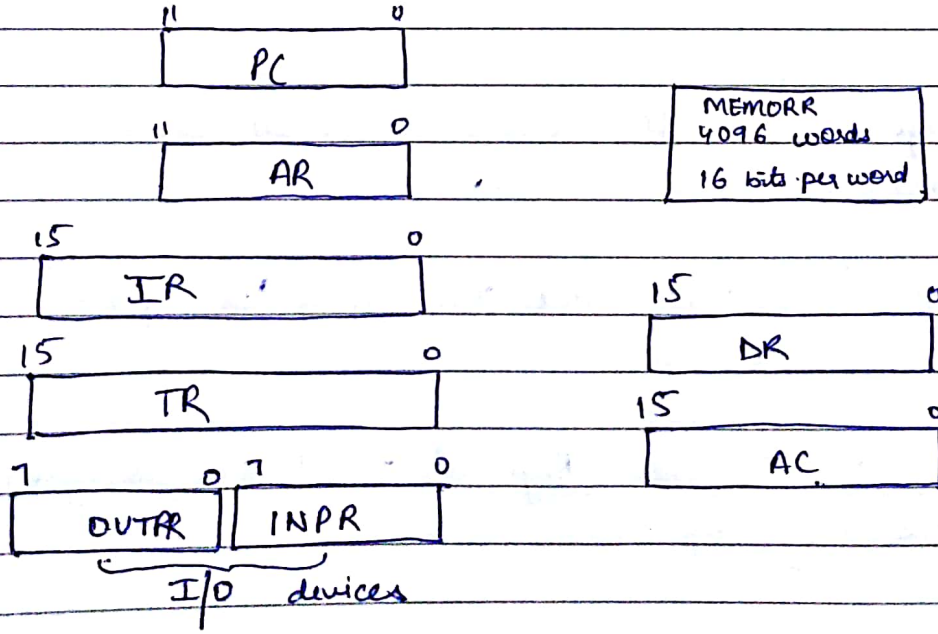


Indirect Address

Whereas, in the 2nd part of the instruction code designates an address of a memory word in which the address of operand is found then it's indirect addressing.

The mode bit is 0 for direct addressing and 1 for indirect addressing.

* REGISTERS –



Compiler instructions are normally stored in consecutive

13/8/18

memory locations and executed sequentially one at a time. The control reads an instruction from a specific address from memory and executes it.
The computer needs processor register for manipulating data and registers for storing memory locations.

* TYPES OF REGISTERS –

① PC (Program Counter) – It holds the add of the next instruction to be executed.

② AR (Address Register) – It holds the address for memory.

③ IR (Instruction Register) – It holds the instru code The instructions read from memory are stored in IR.

④ TR (Temporary Register) – It holds the temporary data.

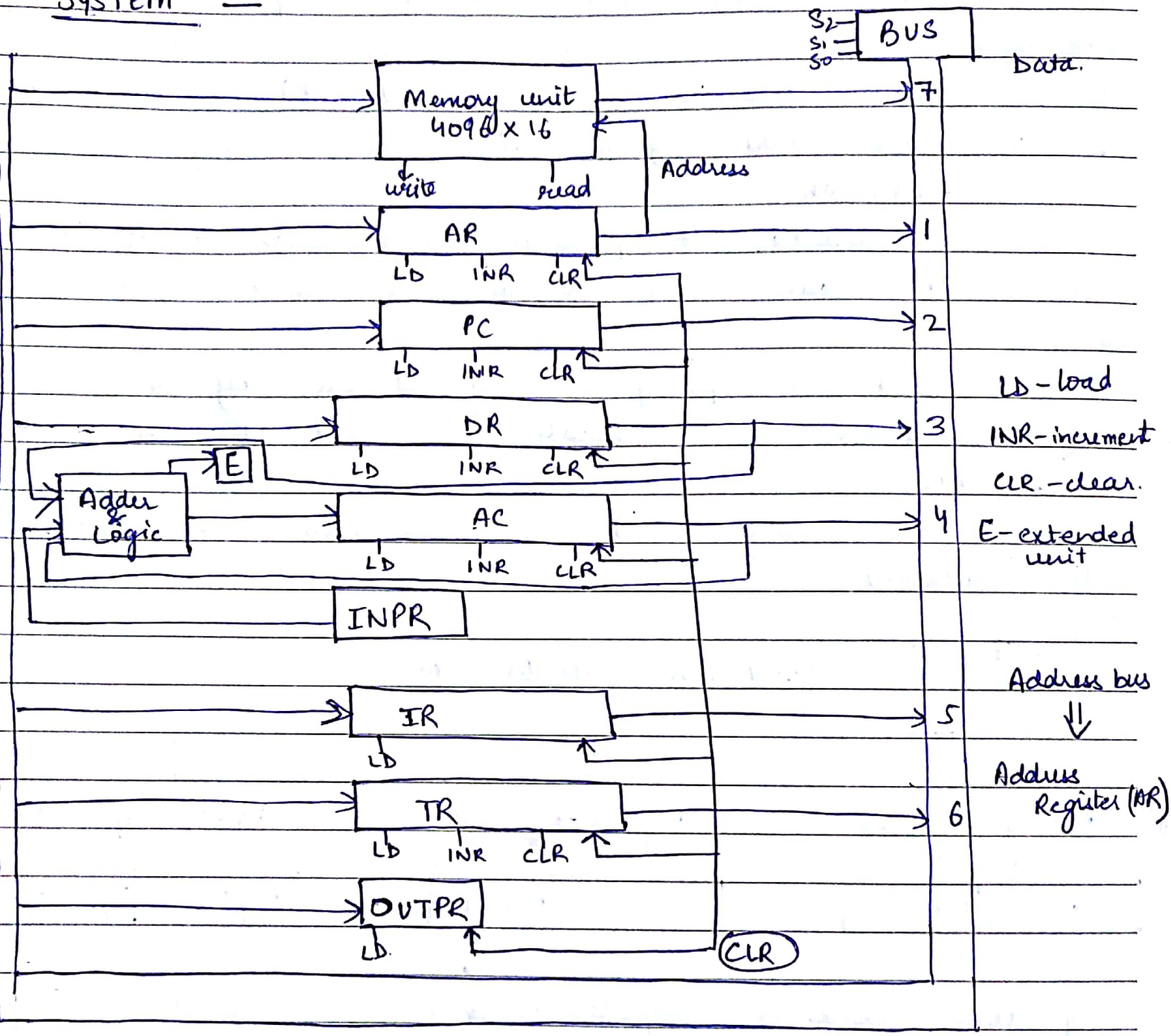
⑤ OUTR (Output Register) – It holds an 8-bit character for an output device.

⑥ INPR (Input Register) – It receives an 8-bit character from an input device.

⑦ DR (Data Register) – It holds the operands from the memory.

⑧ AC (Accumulator or Processor Register) – It is a general purpose processing register.

* Basic Computer Registers Connected to Bus System —



S_0	S_1	S_2	
0	0	0	→ 0
0	0	1	→ 1
0	1	0	→ 2

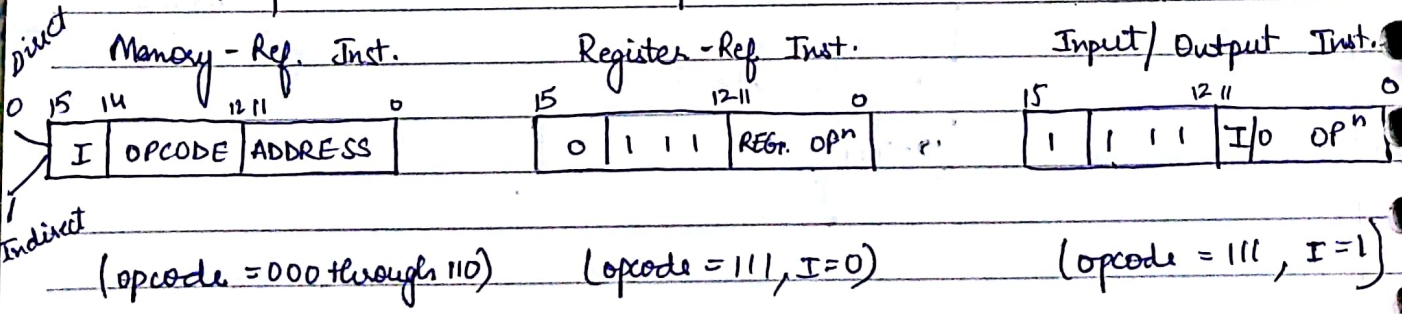
⇒ Extended unit (E) — 1-bit flip flop to store any carry generated.

The output of 7 registers and memory is connected with common bus. A specific output is selected through S_0, S_1 & S_2 at any given time.
A particular register whose load (LD) is enabled receives the data from the bus during the next block transition.
INPR is connected to provide information to the bus and OUTPR can only receive information from the bus.
AR is used to specify memory address by using this single register to eliminate the need for an address bus.
(increment is being used to move from one instruction to another.)

Imp.

(111) (7 is not used)

COMPUTER INSTRUCTIONS



Symbol	I=0	I=1
AND	0XXX	8XXX
ADD	1XXX	9XXX
LDA	2XXX	AXXX
STA	3XXX	B---
	4---	C---
	5---	D---
	6---	E---

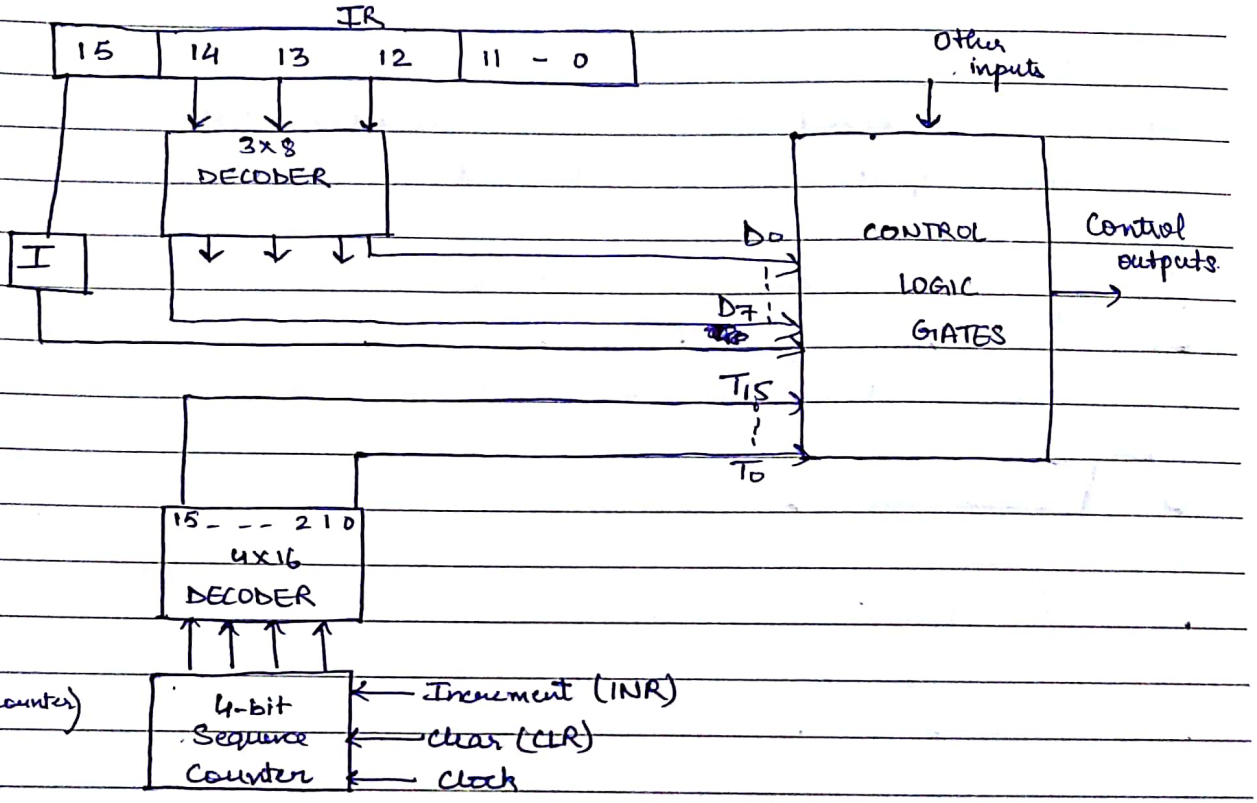
CLA	7800
CLE	7400
CMA	7200
CME	7100
	7080

INP	F800
OUT	F400
SKI	F200
SKO	F100

14/8/18

where CLA : Clear Accumulator, CLE : Clear Extended bit
CMA : Complement Accumulator, CME : Complement Extended bit

* TIMING AND CONTROL —



The timing for all registers in a basic computer is controlled by master clock generator. The clock pulses don't change the state of a register unless the register is enabled by a control signal.

There are 2 major types of control organizations :

① HARDWIRED - In it, the control logic is implemented with gates, flip flops, decoders and other digital circuits.

② MICROPRAMMED - In it, the control information is stored in a control memory. The control memory is programmed to initiate the required sequence of microop^n.

→ opcode decides the opⁿ.

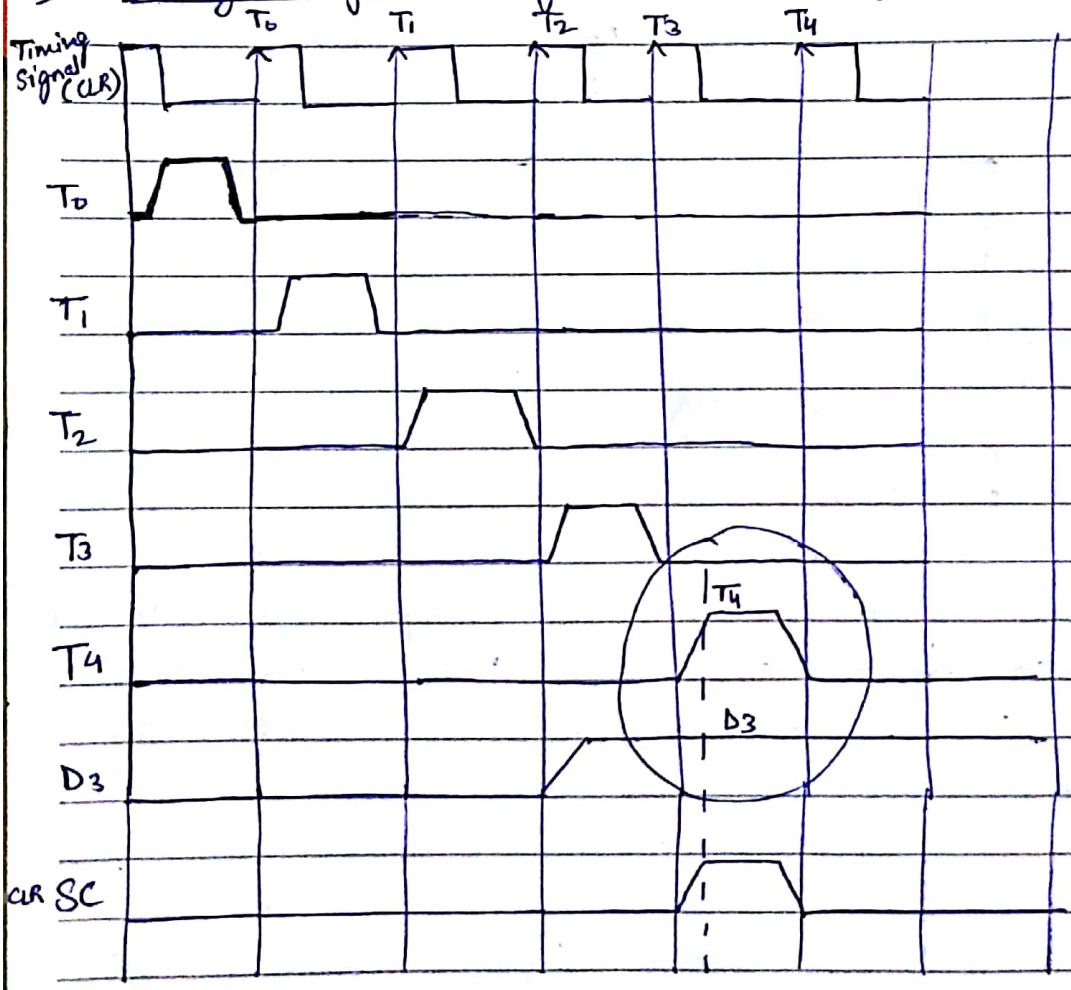
eg:- 1) D_5T_4 : PC + 1.
D_6T_5 : AC ← PC
D_7T_6 : IR ← AC
Decode - operands

} Reg. Ref. Instructions

2) D_3T_4 : SC ← 0

At T_4 SC is cleared to 0 (i.e. Sequence counter) if decoder output D_3 is activated.

⇒ Timing diagram of Statement 2,

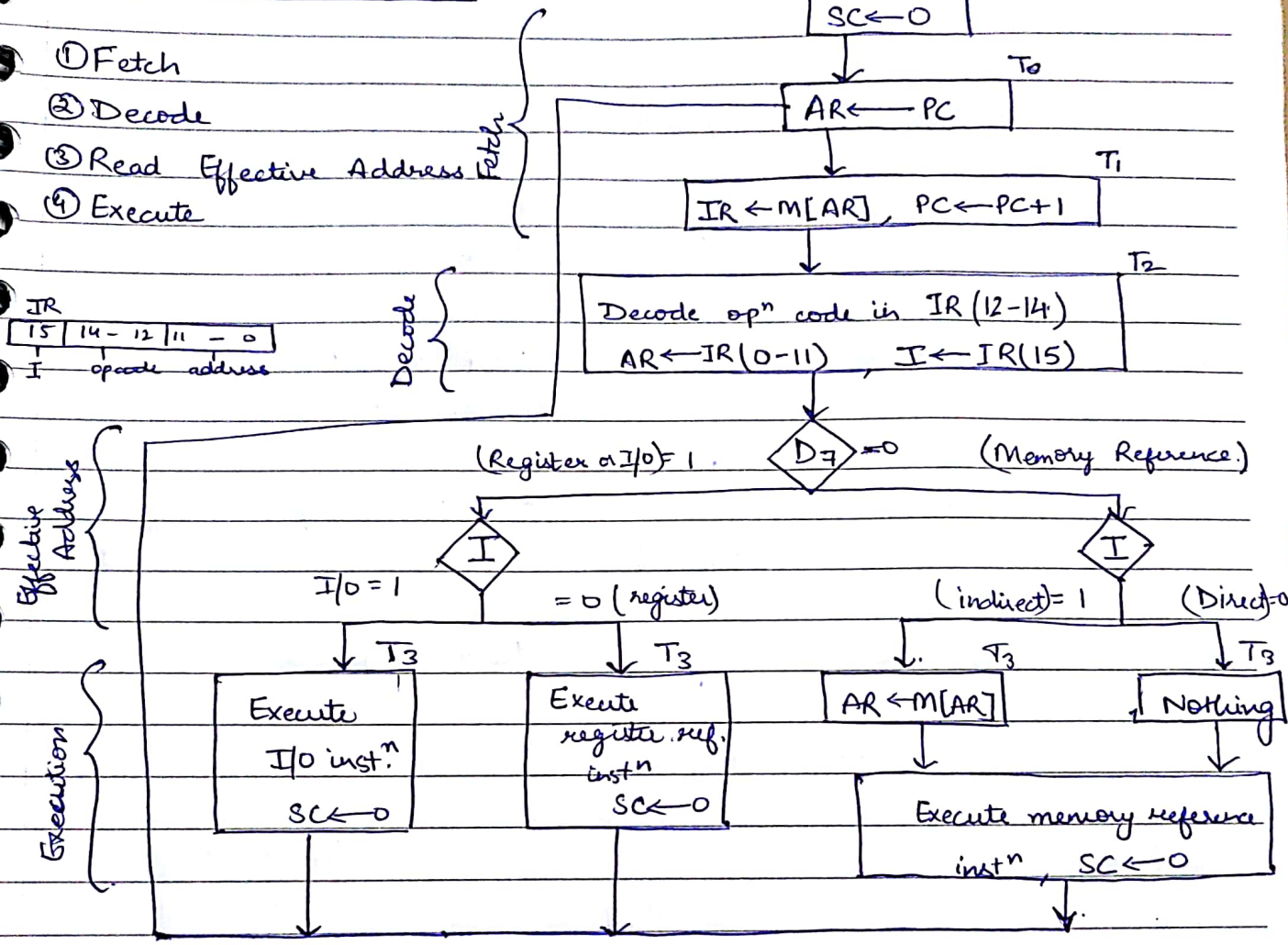


20/8/18

SC : sequence counter

* INSTRUCTION CYCLE –

1. Fetch
2. Decode
3. Read Effective Address
4. Execute



The program is executed in the computer by going through a cycle for each instruction. Each instⁿ cycle is in turn sub-divided into a sequence of sub-cycles or phases :

1) Fetch the instruction from memory
2) Decode the instruction.
3) Read the effective address from memory
4) Execute the instruction.

(* Refer D7 computer instruction flow diagram)

* EXECUTION OF REGISTER REFERENCE INSTRUCTION —

I=0

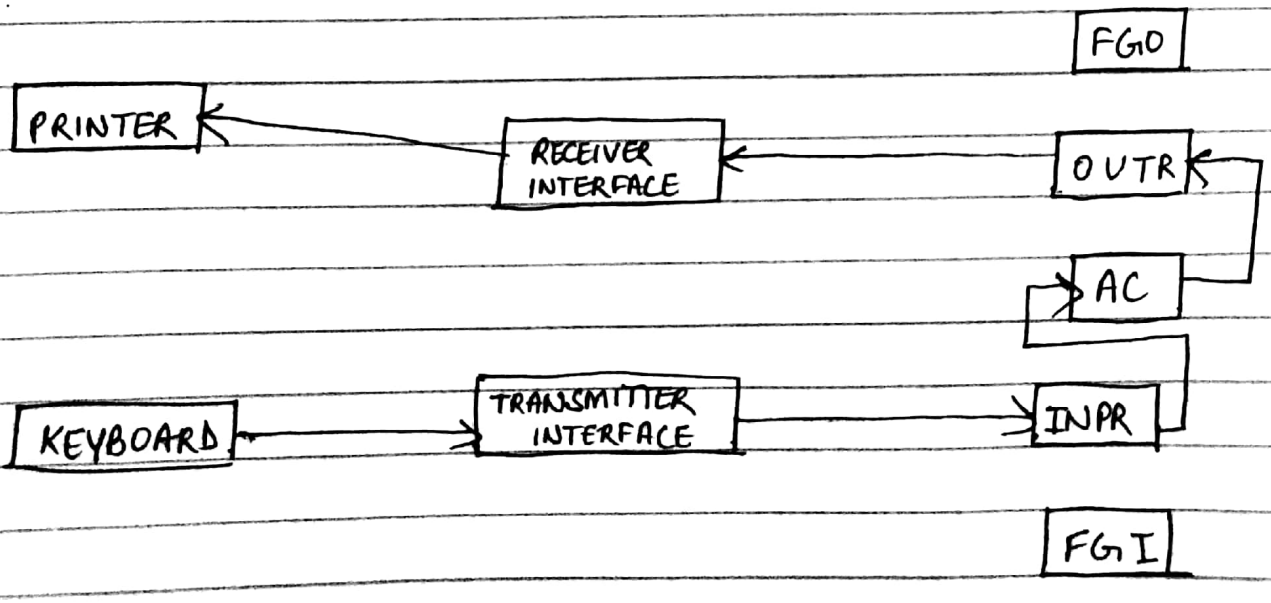
i.e. let $D_7 I' T_3 = r$.

I' $IR(i) = B_i$ (operation)

r:	$SC \leftarrow 0$	Clear SC
$r B_{11}$	$AC \leftarrow 0$	Clear AC
$r B_{10}$	$E \leftarrow 0$	Clear E
$r B_9$	$AC \leftarrow \overline{AC}$	complement AC
$r B_8$	$E \leftarrow \overline{E}$	complement E
$r B_7$	$AC \leftarrow shr\ AC$,	
	$AC(15) \leftarrow E$, $E \leftarrow AC(0)$ (temp. saving rightmost bit)	circulate right (all bits are shifted towards right)
$r B_0$	$S \leftarrow 0$	Halt computer

* INPUT / OUTPUT CONFIGURATION —

Input/Output Terminal — Serial Communication — Computer Registers & Flip flops



21/8/18

* I/O INSTRUCTIONS

I=1 set $D_7 I T_3 = p$ & $IR(6-11) = B$

P: $SC \leftarrow 0$

INP PB_{11} : $AC(0-7) \leftarrow INPR$, $FGI \leftarrow 0$

OUT PB_{10} : $OUTR \leftarrow AC(0-7)$, $FGO \leftarrow 0$

SKI PB_9 : If $(FGI = 1)$ then $(PC \leftarrow PC+1)$

SKO PB_8 : If $(FGO = 1)$ then $(PC \leftarrow PC+1)$

ION PB_7 : $IEN \leftarrow 1$

IOF PB_6 : $IEN \leftarrow 0$

The serial information from the keyboard is shifted into the I/P register INPR. The serial information for the printer is stored in O/P register OUTR. These 2 registers communicate with communication interface serially with AC in parallel. The 1 bit I/P flag FGI is a control flip flop. The flag bit is set to 1 when new information is available in the I/P device and is cleared to zero when the information is accepted by the computer. Initially, the O/P flag FGO is set to 1, the computer checks this flag if it is 1, the information from AC is transferred in parallel to OUTR and FGO is cleared to 0.

INP : it inputs a character from INPR to AC and FGI is set to 0.

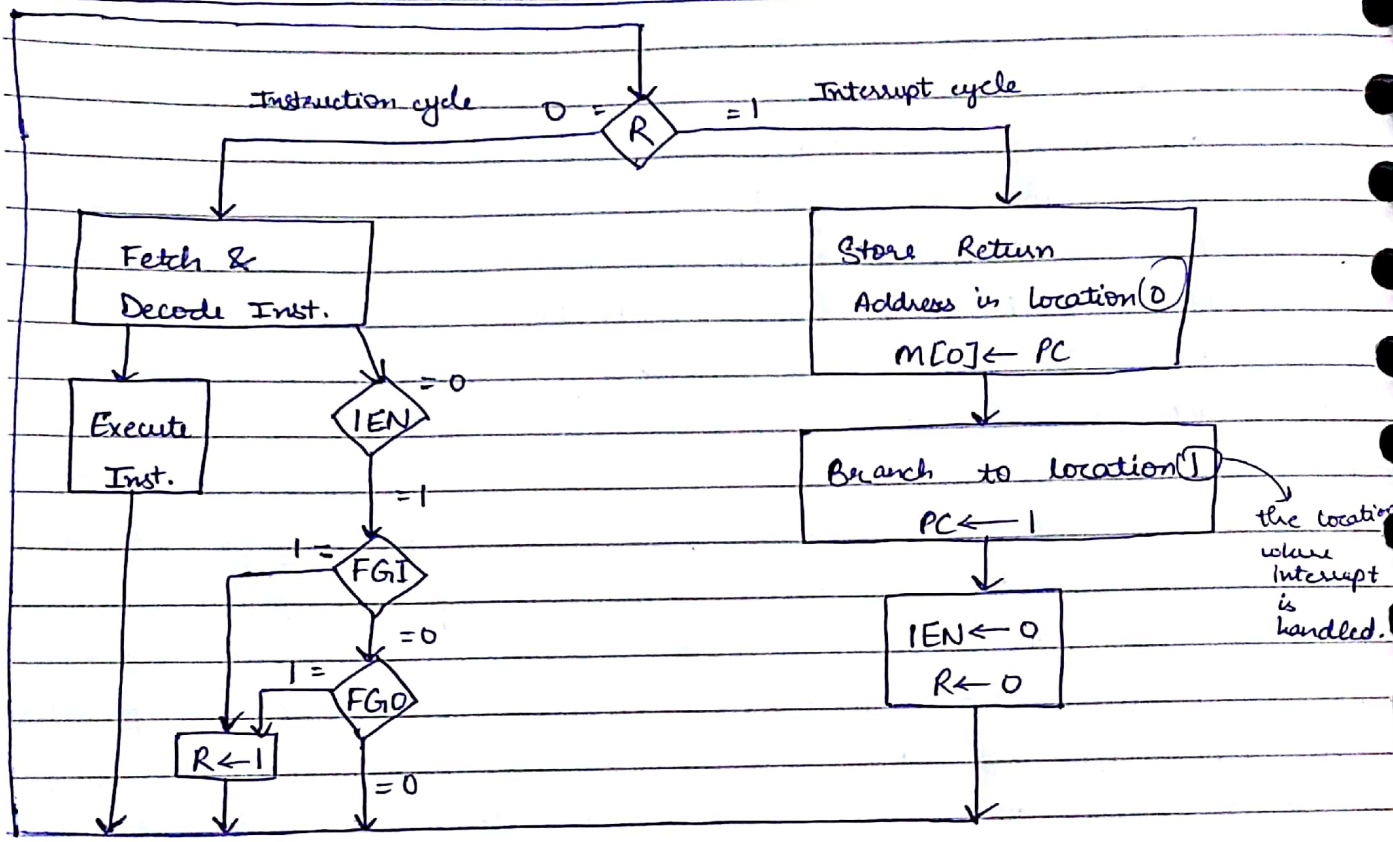
SKI : If FGI = 1, skip on input flag i.e. $PC \leftarrow PC+1$ (move on to the next instruction).

SKO : If FGO = 1, then skip on output flag.

ION : Interrupt enable ON

IOF : Interrupt enable OFF

* FLOWCHART FOR INTERRUPT CYCLE –



In program interrupt, the computer keep checking the flag bits and when it finds its set it initiates an information transfer. The difference of information flow rate b/w the computer and that of the I/O devic makes this type of transfer inefficient. The comp. is wasting time by checking the flag instead of doing some other useful processing task. An alternative to this approach is to let the external device inform the computer when its ready for the tranfer. The IEN (Interrupt enable flipflop) can be set and cleared with 2 instructions. When IEN is set to 0, the flags can't interrupt the computer. When IEN is set to 1, the flags can interrupt.

{Design of Accumulator Logic Instructions}

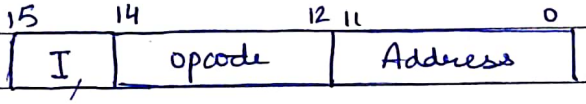
28/8/18

Q.1 1) 256 K words of 32 bits each.
A binary code instruction is stored in memory. Instruction has 4 parts : indirect bit, op code, register code specify one of 64 registers and an address part. How many bits are there in op code, register code part and address part..

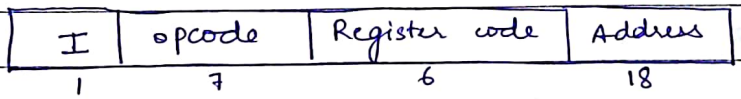
2) Draw the instruction word format and indicate no. of bits in each part.

Ans

For 16 bit :



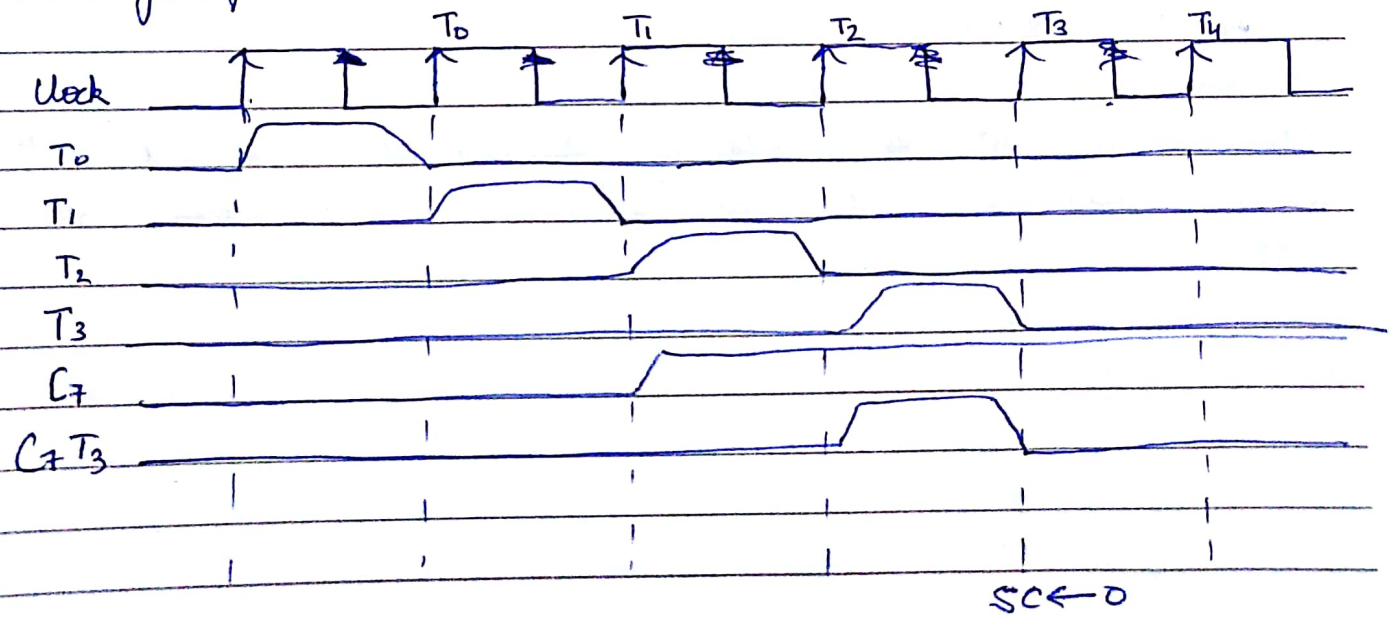
For 32 bits :



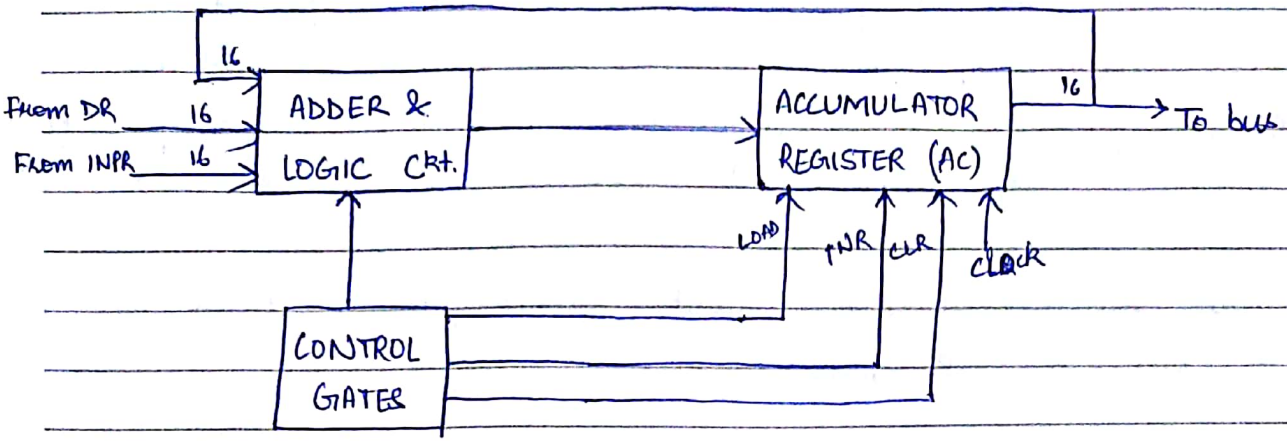
PTO

Q2 $C_7T_3 : SC \leftarrow 0$

Ans: Timing diagram.



DESIGN OF ACCUMULATOR LOGIC —



Memory
- D_0T_5 : $AC \leftarrow AC \wedge DR$
- D_1T_5 : $AC \leftarrow AC + DR$
- D_2T_5 : $AC \leftarrow DR$

I/O
- PB_{11} : $AC(0\text{-}7) \leftarrow INPR$

Register
- rB_9 : $AC \leftarrow \overline{AC}$
- rB_7 : $AC \leftarrow Shr\ AC$, $AC(15) \leftarrow E$
- rB_6 : $AC \leftarrow Shl\ AC$, $AC(0) \leftarrow E$
- rB_4 : $AC \leftarrow 0$
- rB_5 : $AC \leftarrow AC + 1$

Adder & Logic ckt. has 3 sets of I/P : 1 set from O/P of AC, another from DR and the 3rd set of I/P comes from INPR.

*continued

Ans 1.

256K = $2^8 \times 2^{10} = 2^{18}$

1 to 64 reg. = $64 = 2^6$

Address part = 18 bits

Register code = 6 bits

Indirect bit = 1 bit

opcode = 32 - 25 = 7 bits