Aho Ullman

Date : / /
Page No :

C.W

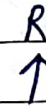
19/7/18

MODULE - I

⇒ **Micro operation** - The operations executed on data stored in registers are called as Microoperations.
Ex- Shifting info. from 1 register to other, count, clear, load.

The symbolic notation used to describe the microoperation transfers among registers is called a register transfer language.

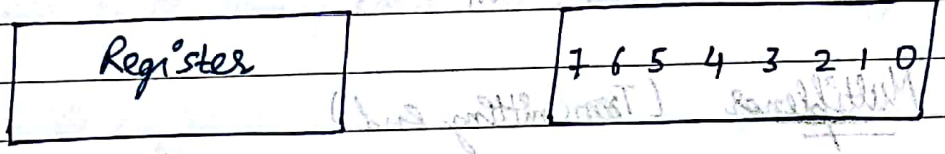
$R_2 \leftarrow R_1$

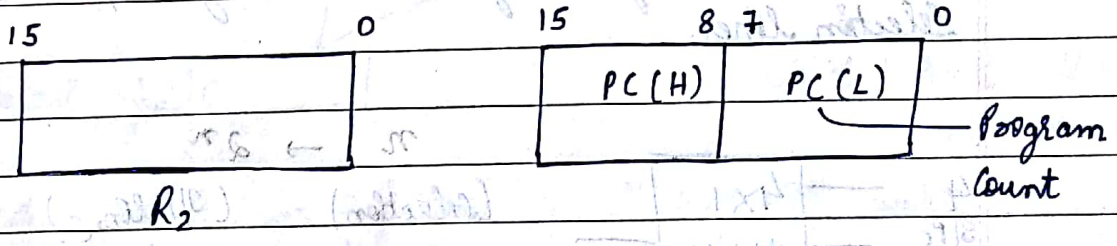


8	4	2	1	
MSB 1	1	1	0	LSB
A	B	C	D	

least significant bit

⇒ **Block diagram of Register**

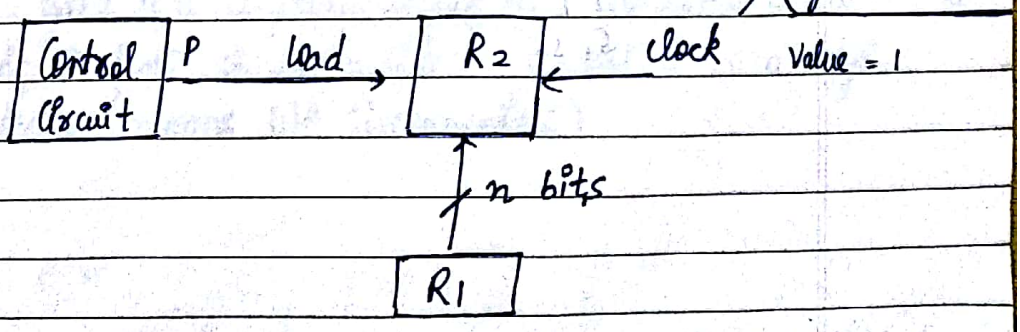




★ P = 1 $R_2 \leftarrow R_1$ P: $R_2 \leftarrow R_1$

P = 0

used for synchronization



SARAA

Date : / /
Page No :

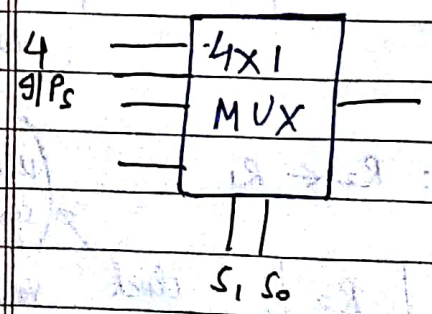
⇒ Basic Symbols for Register transfer

Symbol	Description	Example (Memory Address Regi.)
letters & Numbers	Denotes a Register	MAR, R_2
()	Denotes a part of a Register	$R_2(0-7)$, $R_2(L)$ ↓ low bits
←	Denotes transfer of Information	$R_2 \leftarrow R_1$
(comma) ,	Separates two microoperation	$R_2 \leftarrow R_1$, $R_1 \leftarrow R_2$

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

⇒ Multiplexor (Transmitting end)

A device having many I/P's & single O/P & some selection lines.



$n \rightarrow 2^n$

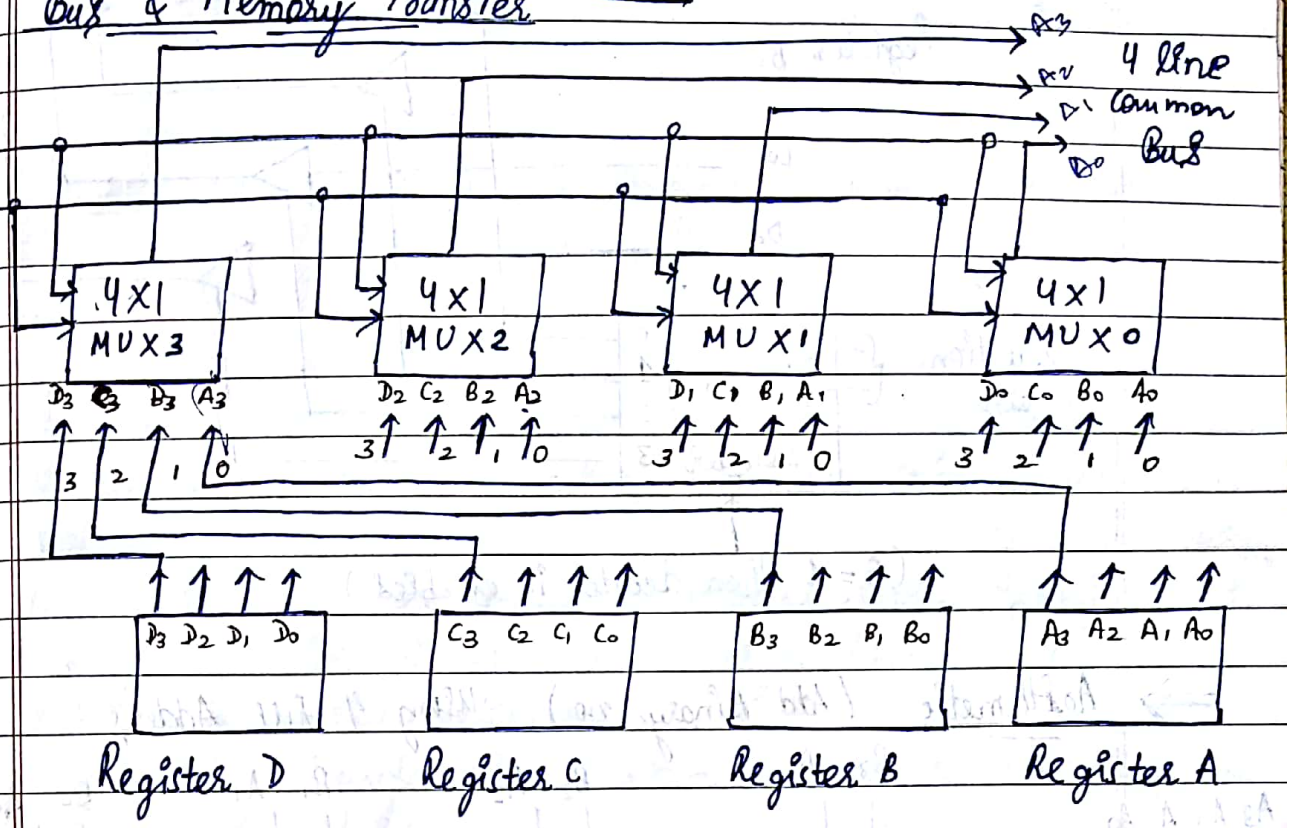
(selection lines) (I/P lines)

selected Register A ← Selection Lines

00	First Bit of every Multiplexer
01	2nd ...
10	3rd ...
11	4th ...
S_1	S_0

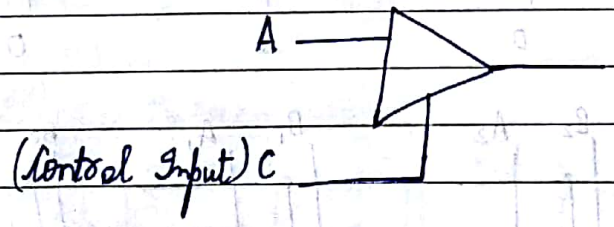
Date: / /
Page No:

⇒ Bus & Memory Transfer



In case of no multiplexer, three state bus buffer is used.

⇒ 3 state Buffer



$C = 1 \begin{cases} A = 1 & Y \to 1 \\ A = 0 & Y \to 0 \end{cases}$

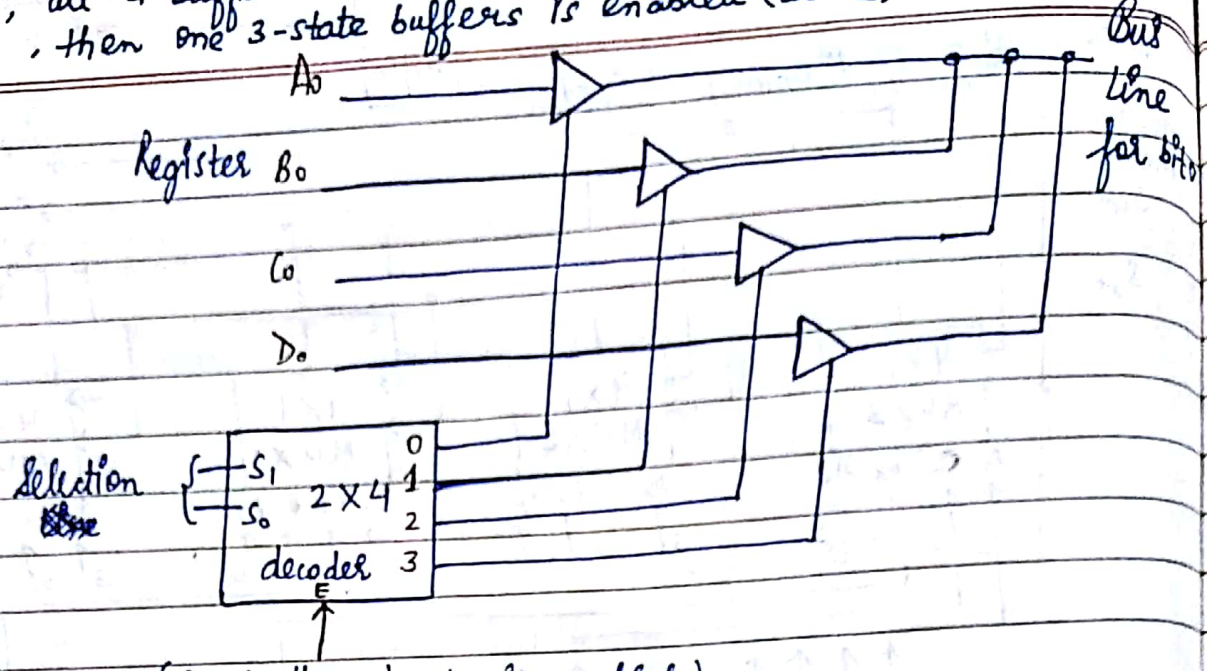
when C = 0 (high Impedance), it will not work.
(Open circuit condition)

A three-state gate is a digital circuit that exhibits 3 states. 2 states are equivalent to logic 0 & 1 & third is a high Impedance state (means O/P disconnected)

Encoder, decoder (Input, Output, enabled line)

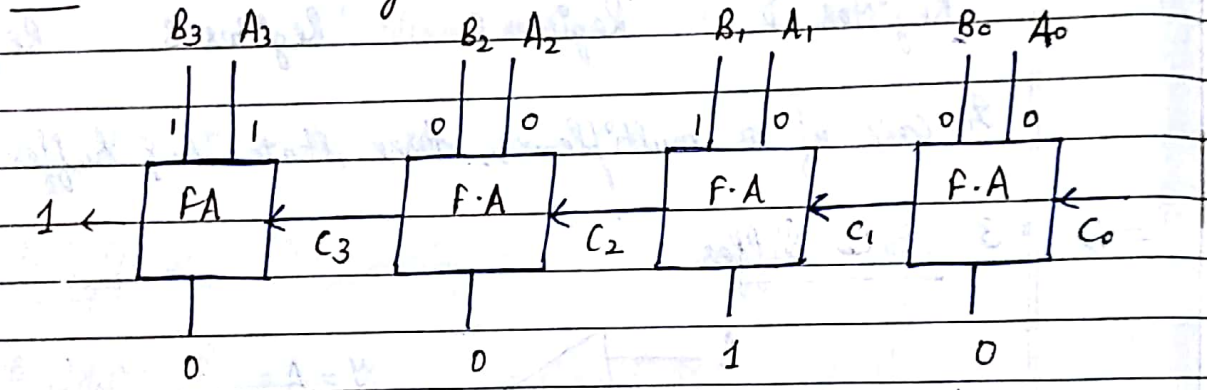
When E=0, all 4 buffers are disabled
& E=1, then one 3-state buffers is enabled (active).

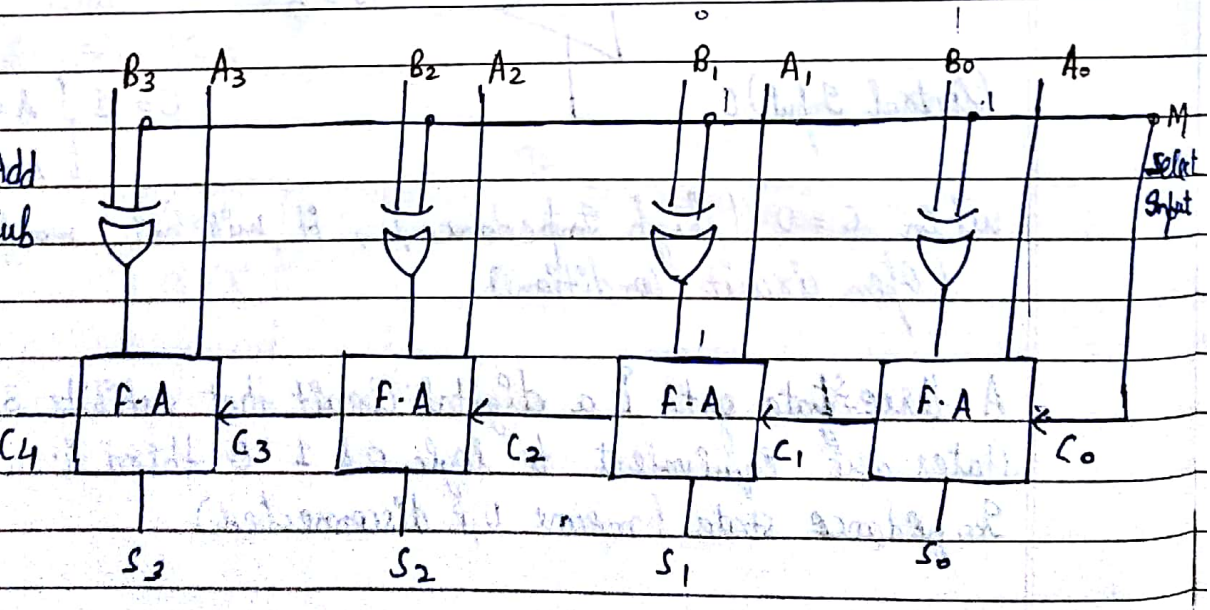
Date: / /
Page No:



⇒ Arithmetic (Add Binary no) using 4 full Adders

A_3	A_2	A_1	A_0
1	0	0	0
1	0	1	0
B_3	B_2	B_1	B_0





(B)	(M)	(y)
A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

M=0, B (same)
M=1, Complement

Adder Subtracter (Both)

SARAA

Date : / /
Page No :

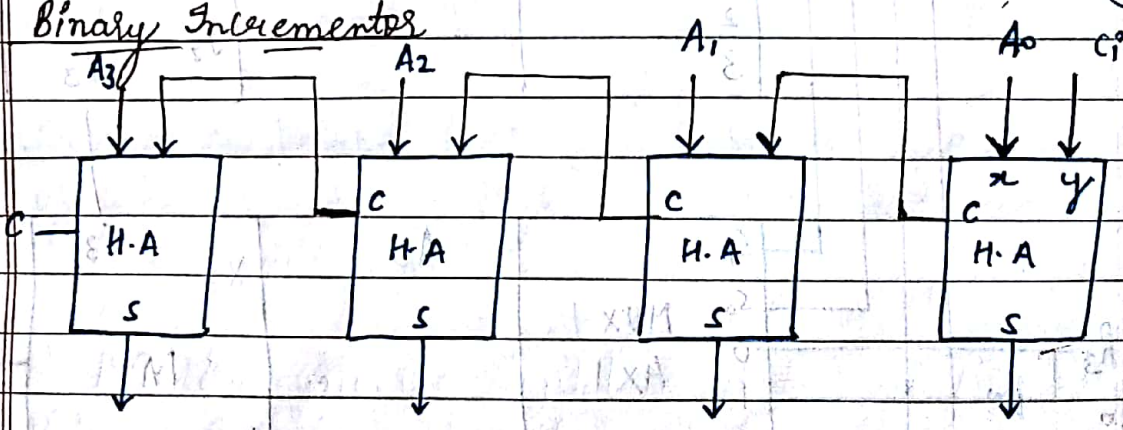
Ex- M = 1 5 → 0101
3 → 0011
Result will be 1010
Ignore the carry, then result
0010 i.e. (2)

A Microoperation is an elementary operation performed with the data stored in the registers. The M.O most often encounter in digital computers are classified under 4 categories -

NOTE- (we do not change the content of register during transfer in 1st only)

1) Register transfer microoperations transfer binary information from 1 to another register.
2) Arithmetic microoperation perform arithmetic operations on numeric data stored in the register.
3) Logical microoperation perform bit manipulations operations on non-numeric data stored in registers.
4) Shift microoperation performs shift operations on data stored in the registers.

⇒ Binary Incrementer (Previous carry 1)

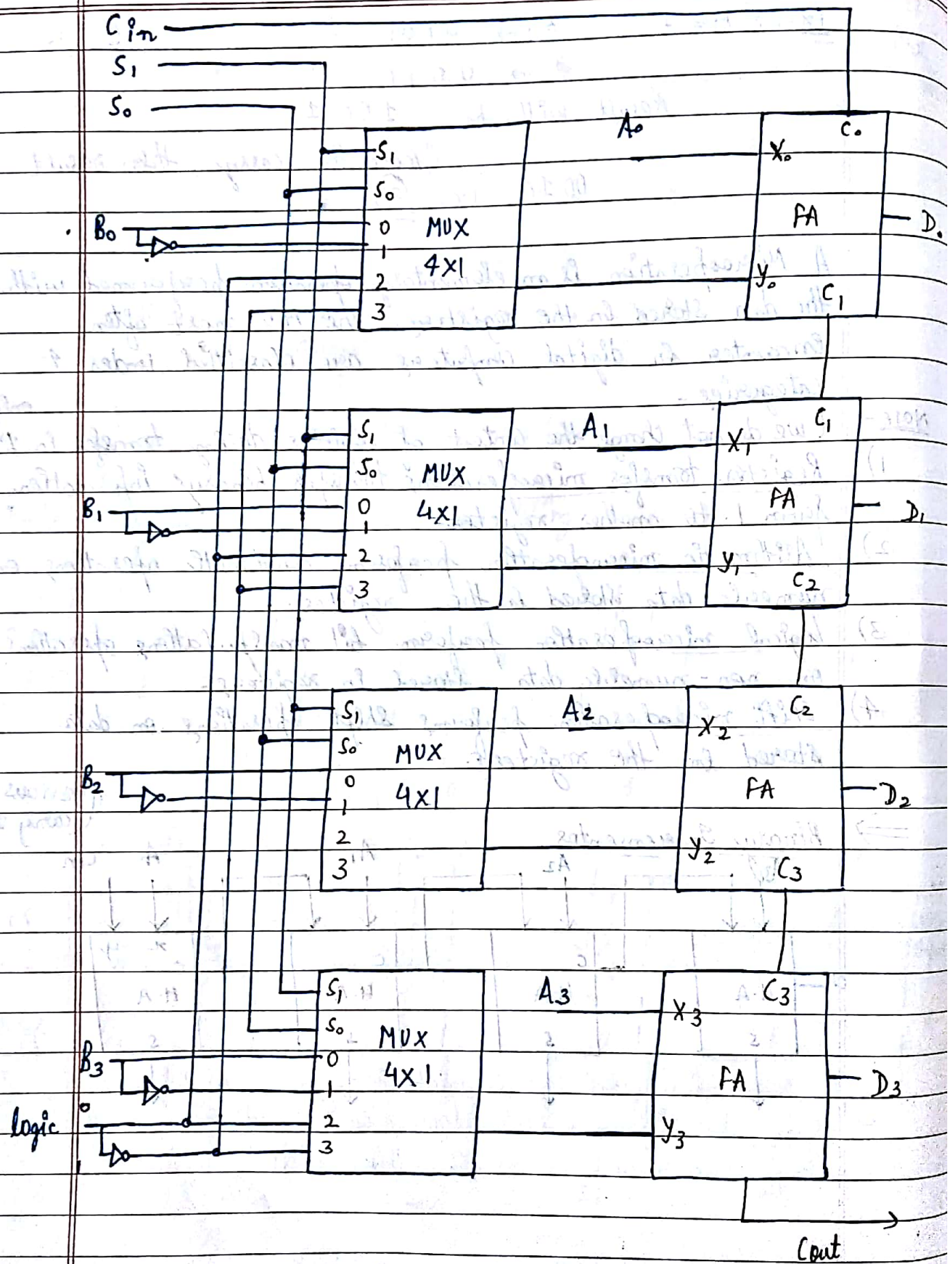


SARAA

Arithmetic Circuit for Microoperations of 4-bit

Date : / /

Page No :



SARAA

Table for Microoperations

Date: / /
Page No:

Select			Input	O/P	Microoperations
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\bar{B}$	$D = A + \bar{B} = A - B - 1$	Subtract with borrow
0	1	1	$\bar{B}$	$D = A + \bar{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

⇒ Logic Microoperations

① Selective Set (OR) → The selective set operation sets to 1, the bits in the register A where there are corresponding 1's in the register B.

```
1 0 1 0   A          A ← A+B
1 1 0 0   B
1 1 1 0   New A
```

② Selective complement (XOR) The selective complement operation complements bits in A where there are corresponding 1's in register B.

```
1 0 1 0   A
1 1 0 0   B
0 1 1 0   New A
```

SARAA

Date: / /
Page No:

③ Selective clear operation clears to 0 the bits in A only where there are corresponding 1's in B.

```
1 0 1 0   A
1 1 0 0   B
0 0 1 0   New A
```

④ Mask operation - The mask operation (AND) is similar to the selective clear operation, except that the bits of A are cleared only where there are corresponding 0's in B.

```
1 0 1 0   A
1 1 0 0   B
1 0 0 0   New A
```

⑤ Insert operation inserts a new value into a group of bits (Masking + ORing)

want

```
A →      0 1 1 0     1 0 1 0     1 1 0 1     1 0 1 0
Mask →   0 0 0 0     1 1 1 1
         0 0 0 0     1 0 1 0
Then OR → 1 1 0 1    1 0 1 0
         1 1 0 1     1 0 1 0
```

⑥ Clear operation compares the words or bits in A & B & produces 0 if 2 numbers are equal.

```
1 0 1 0
1 0 1 0
0 0 0 0
```

SARAA

④ symbol :-
Sh l
Sh R

Right
ash l
ash r

NOTE

+ve
Arith...
Right s...

Date : / /
Page No :

⑦ **Shift microoperation** - 3 types of shift operations logical, circular & Arithmetic shift.

symbol :-

Shl A Shift left
Shr A Shift right

A logical shift is one that transfers 0 through the serial input.

1 0 1 0 → shift left → 0

$R_1 \leftarrow shl\ R_1$
$R_1 \leftarrow shl\ R_2$
$R_1 \leftarrow shr\ R_1$

The circular shift is also known as rotate operations. Circulates the bits of register around the 2 ends without the loss of information.

1 1 0 0 → 0 1 1 0 (Right)

cil R_1 (left)
cir R_2 (right)

ashl R_1
ashr R_2

An arithmetic shift is an microoperation that shifts a assigned binary number to left & right. An arithmetic shift-l multiplies assigned binary no. by 2 & arithmetic shift-r divides assigned binary no. by 2.

NOTE

* Arithmetic shift must leave the signed bit unchanged because the sign of the no. remains the same when it is multiplied or divided by 2.

+ve | -ve number

Arith... Right shift

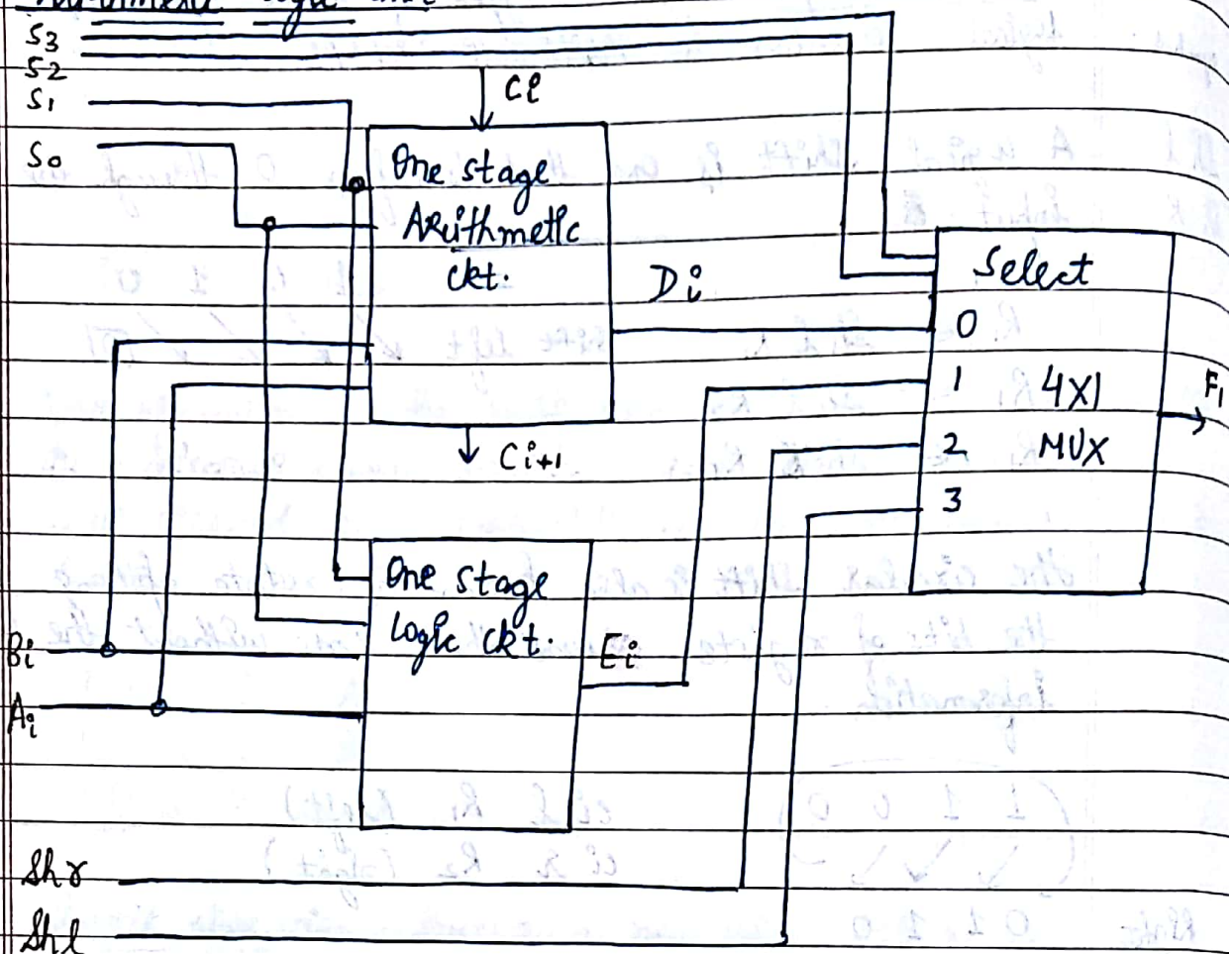
0 1000 8 → 0100 4 → 0010 2 (÷2)

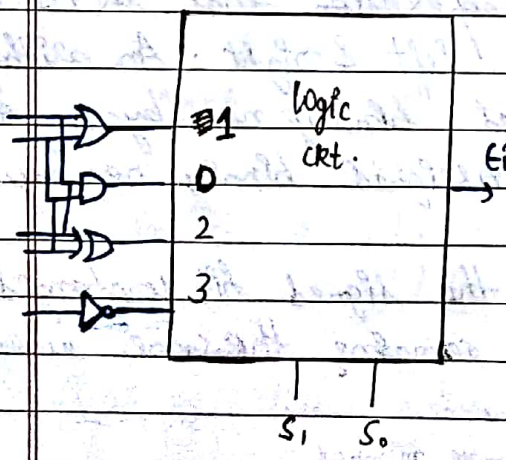
1 1000 −8 → 1100 −4 → 1110 −2

SARAA

Date : / /
Page No :

⇒ Arithmetic logic Unit





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

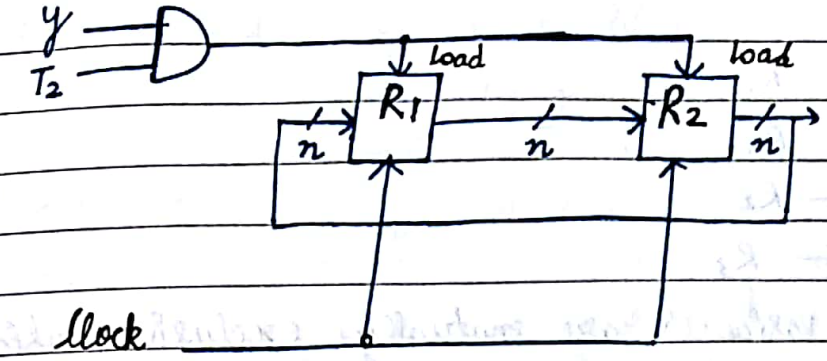
Q. Draw the block diagram of the Hardware that implements the following registes transfer statement.

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

SARAA

Date: / /
Page No:

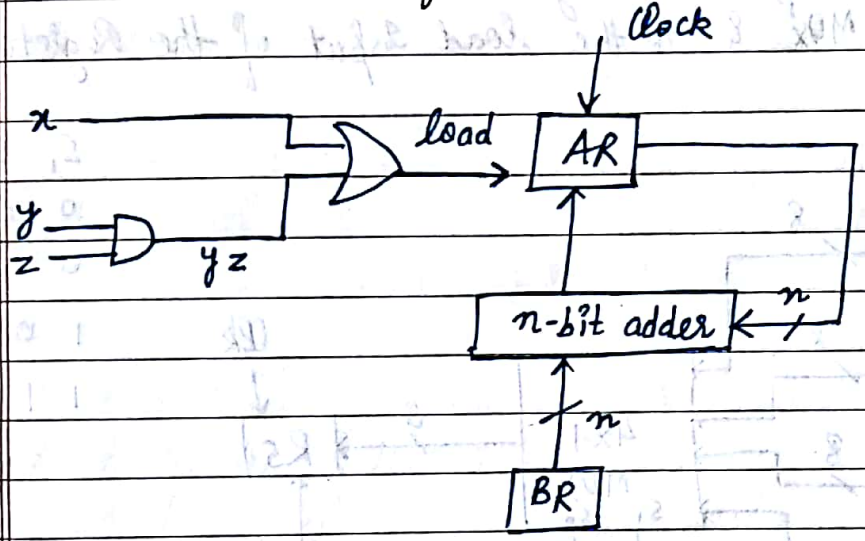
A.



Q. Draw the block diagram for the hardware that implements the statement:

$$x + yz : AR \leftarrow AR + BR$$

A.



Q. Represent the following control statement by 2 register transfer statements with control function

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

else if $(Q=1)$ then $R_1 \leftarrow R_3$

A.

$P: R_1 \leftarrow R_2$

$P'Q: R_1 \leftarrow R_3$

Q. The outputs of 4 registers R_0, R_1, R_2, R_3 are connected through a 4x1 Multiplexer to the inputs of 5th register (R_5). Each Register is 8 bits long. The required transfer are dictated by 4 timing variables

SARAA

T_0 to T_3 as.

T_0 : $R_5 \leftarrow R_0$

T_1 : $R_5 \leftarrow R_1$

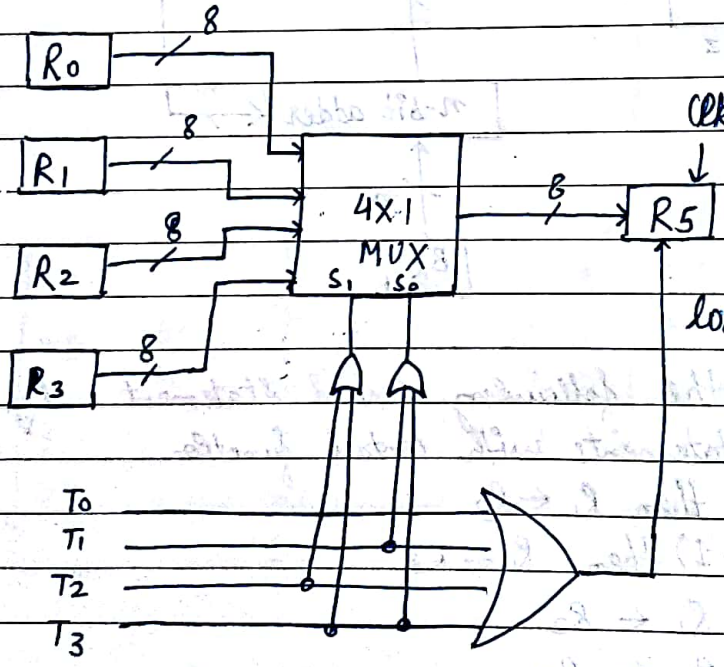
T_2 : $R_5 \leftarrow R_2$

T_3 : $R_5 \leftarrow R_3$

The timing variables are mutually exclusive which means that only 1 variable is equal to 1 at a given time while the other three are equal to 0. Draw the block diagram showing the hardware implementation of the Register transfer. Include the connections necessary from the four timing variables to the selection I/P of the MUX & to the load input of the Register R_5.

A.

S_1	S_0	
0	0	T_0
0	1	T_1
1	0	T_2
1	1	T_3



Q. Show the hardware that implements the following. Include the logic state for the control function & a block diagram for the binary counter with a count enabled input.

SARAA

A.

T_0 —

T_1 —

T_2 —

x —

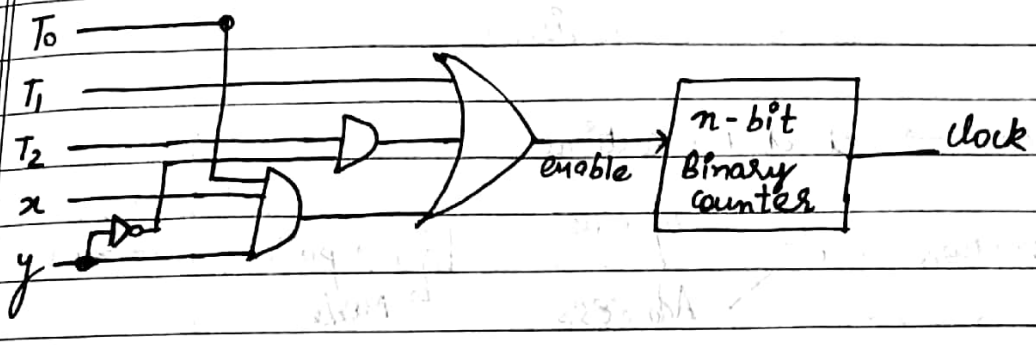
y —

SA

Date : / /
Page No :

$xyT_0 + T_1 + y'T_2 : AR \leftarrow AR + 1$

Ans



2/8/18

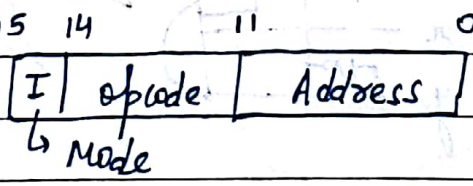
Date: / /
Page No:

MODULE - II

Computer organization
- Program
- Time & Control unit
- Registers

Program → set of instructions

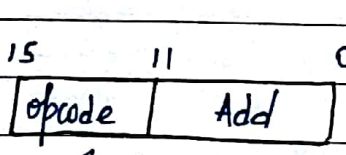
(16 Bits) Instruction code
- opcode
- Address



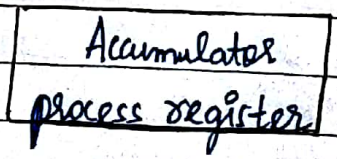
If I → 0 (direct addressing) &
I → 1 (Indirect addressing)

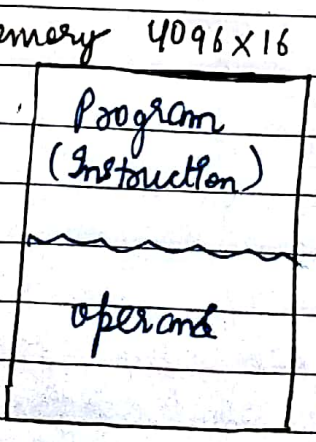
* The organization of computer is defined by its internal registers, the timing & control structure & the program. A program is a set of instructions that specify the operations, operands & the sequence by which processing has to occur. An instruction code is a group of bits that instructs the computer to perform a specific operation.

The operation code (op-code) of an instruction is a group of bits that define operations such as add, subtract, shift, complement etc.



Instruction format

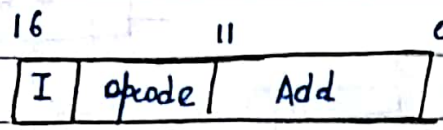


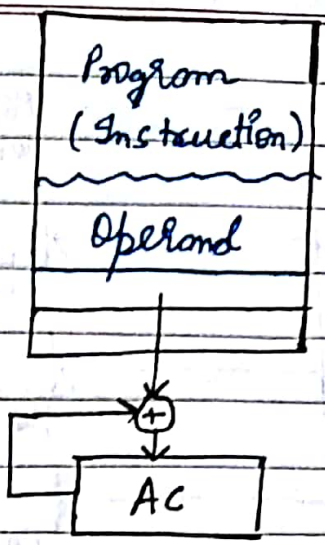


SARAA

Indirect → effective add. of the operand

Date : / /
Page No :





Direct & Indirect

⇒ Computer Registers

Registers	No. of bits	Register Name	function
DR	16	Data Register	Holds memory operands
AR	12	Address Register	Holds address for memory
AC	16	Accumulator	Processor Register
IR	16	Instruction Register	Holds Instruction code
TR	16	Temporary Register	Holds Temperary data
PC	12	Program Counter	Holds address of Instruction
INPR	8	I/P Register	Holds I/P characters
OUTR	8	O/P Register	Holds O/P characters

⇒ Common Bus System

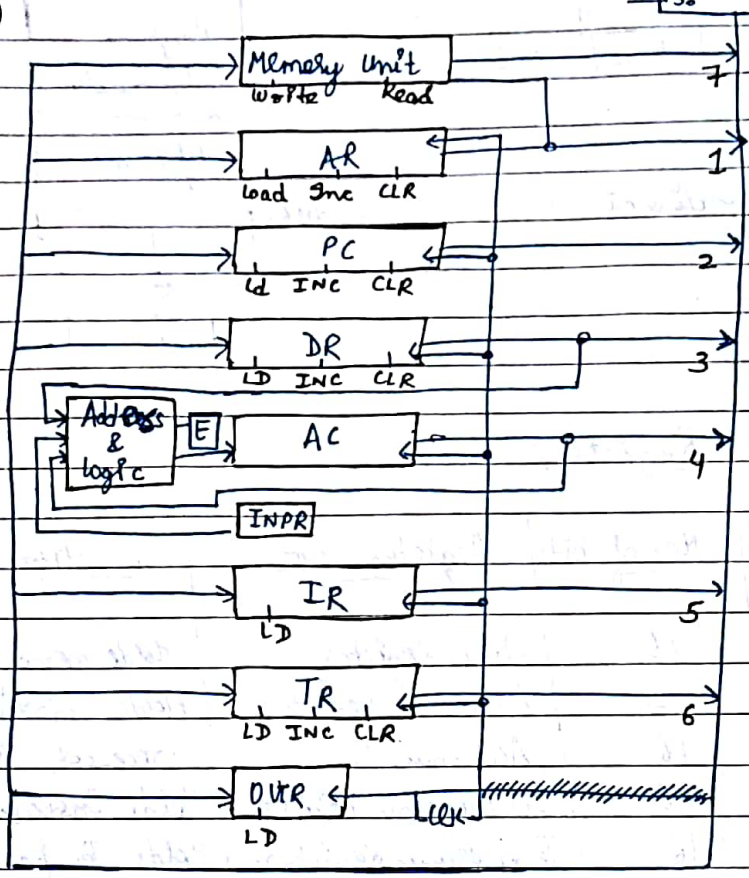
If $S_2 S_1 S_0 \Rightarrow 100$ AC will load data to Bus

[Bus ← AR, Bus ← PC] then by default last 4 will be 0.

(AR | PC ← Bus)

Most significant 4 will be lost.

Starting (0-11) will be transf..

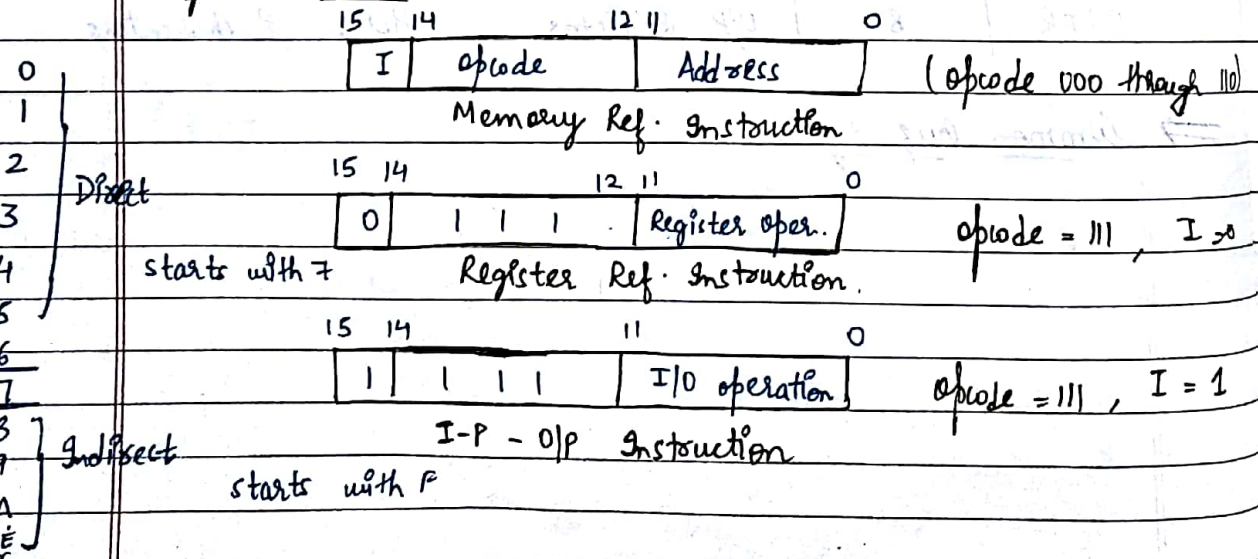


E → F/F

Flip/Flop

(1 bit Memory device)

⇒ Computer Instructions



I = 0

AND 0XXX

ADD 1XXX

Basic Com

⇒ Timing

Control

To

T4

To

T1

T2

T3

T4

D3

SARAA

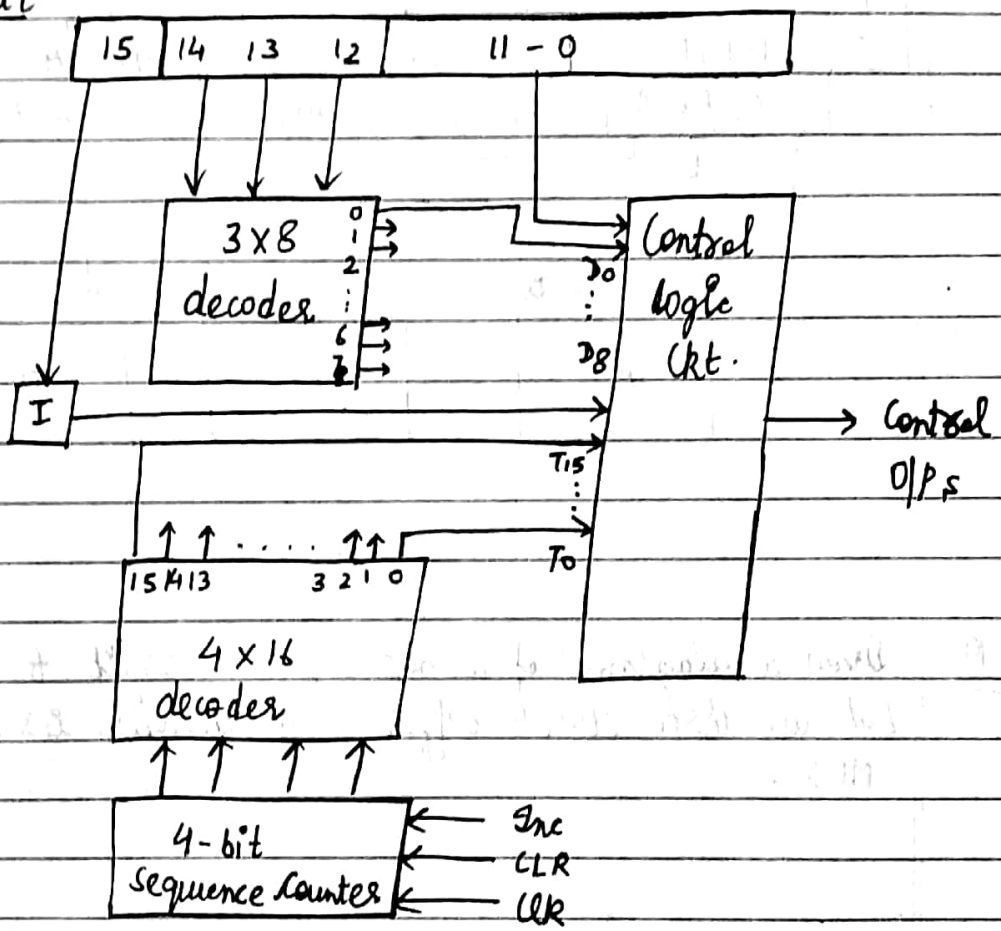
Date : / /
Page No :

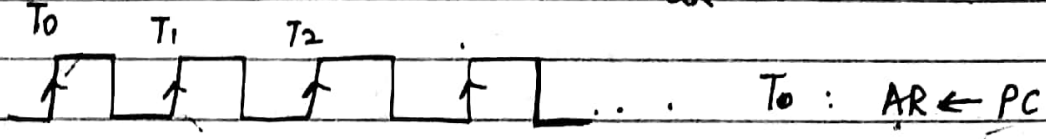
	I = 0	I = 1	
AND	0XXX	8XXX	[And memory word to AC]
ADD	1XXX	9XXX	[Add memory word to AC]

Basic Computer Instructions (Do yourself)

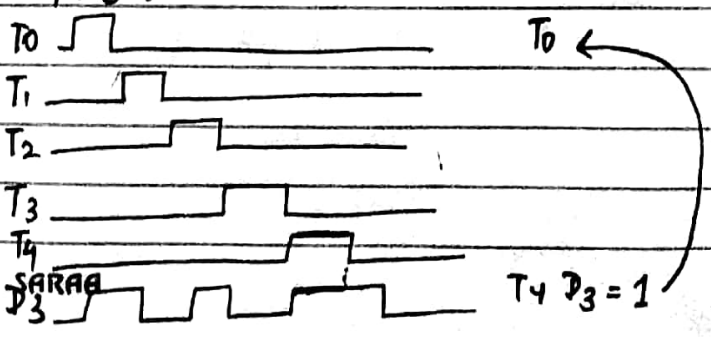
⇒ Timing & control

Control Unit





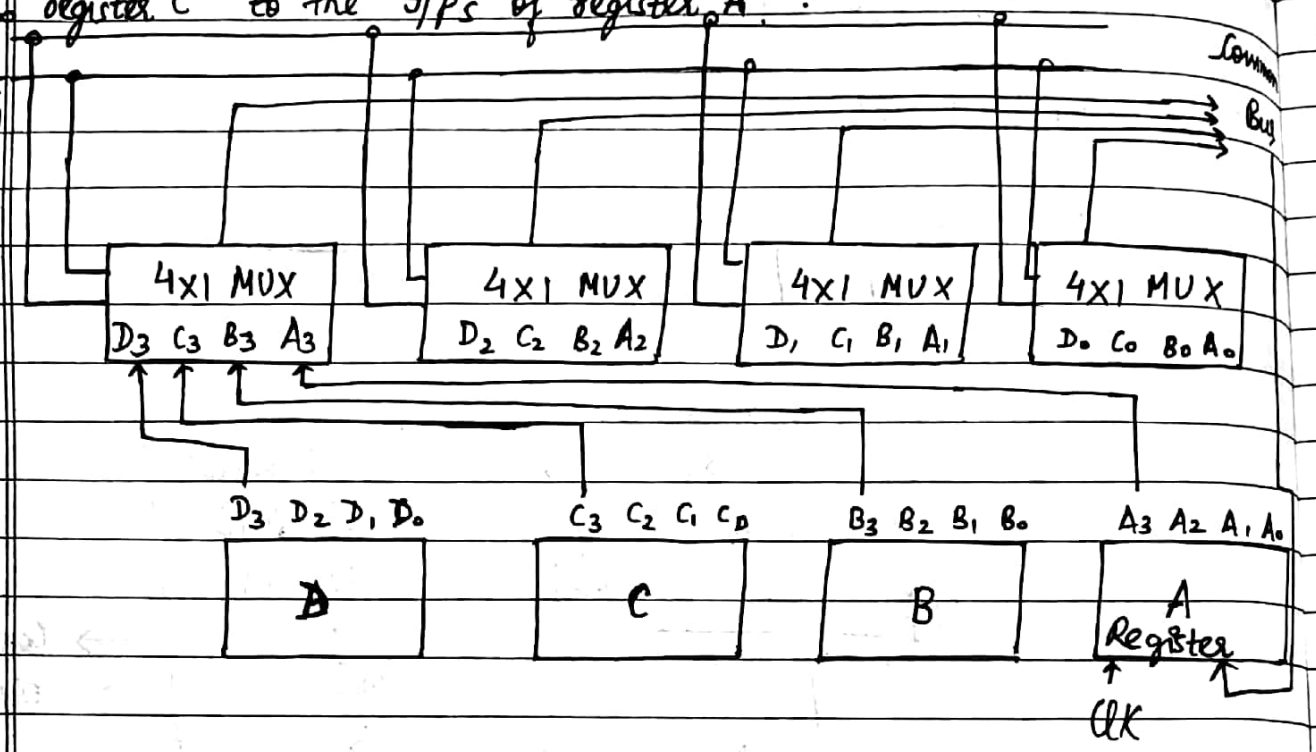
Ex- $T_4 D_3 : SC \leftarrow 0$



Q. What has to be done to the bus system of fig. (15 pg back) to be able to transfer information from any register to another register, Specifically show the connections that must be included to provide a path from the output of ~~register~~ register C to the I/Ps of register A.

Sol.

A.



Q. Draw a diagram of a bus system similar to fig. (common) but use three state buffer & a decoder instead of a 4x1 MUX.

SARAA

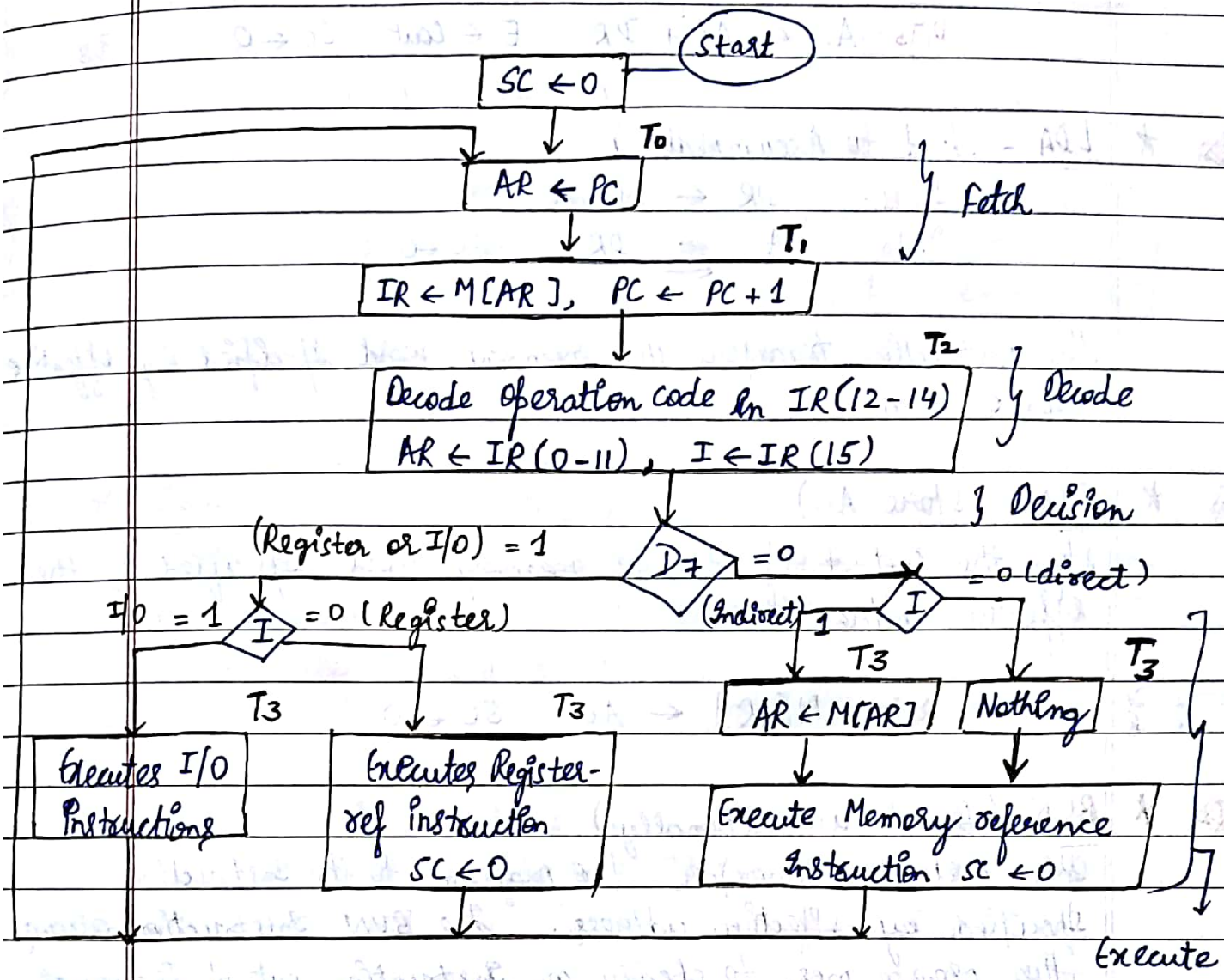
I=0 or I=1 & D7

Decision

Date : / /

Page No :

⇒ Instruction Cycle ① fetch ② Decode ③ Execute



⇒ Memory Reference Instruction

- AND to AC – In this instruction, and operation takes place. On the bits of AC & memory word specified by the effective address. The result of the operation is transferred to AC.

first data transferred to DR $DR \leftarrow M[AR]$: D0T4

bcoz no direct con. b/w Bus & Accumulator. $AC \leftarrow AC \wedge DR$ $SC \leftarrow 0$: D0T5

SARAA

Date : / /
Page No :

Do yourself 9/10

ADD to AC :

D_1T_4 : $DR \leftarrow M[AR]$

D_1T_5 : $AC \leftarrow AC + DR$ $E \leftarrow Cout$ $SC \leftarrow 0$

★ LDA - (load to Accumulator)

D_2T_4 : $DR \leftarrow M[AR]$

D_2T_5 : $AC \leftarrow DR$, $SC \leftarrow 0$

The Instruction transfers the memory word specified by effective address to AC.

★ STA (Store AC)

Store the content of AC into memory word specified by the effective address.

D_3T_4 : $M[AR] \leftarrow AC$, $SC \leftarrow 0$

★ BUN (Branch unconditionally)

This Instruction transfers the program to the Instruction specified by effective address. The BUN Instruction allows the programmer to specify an Instruction out of sequence.

D_4T_4 : $PC \leftarrow AR$

★ BSA (Branch & save Return address)

This Instruction is useful for branching to a portion of a program called a Subroutine.

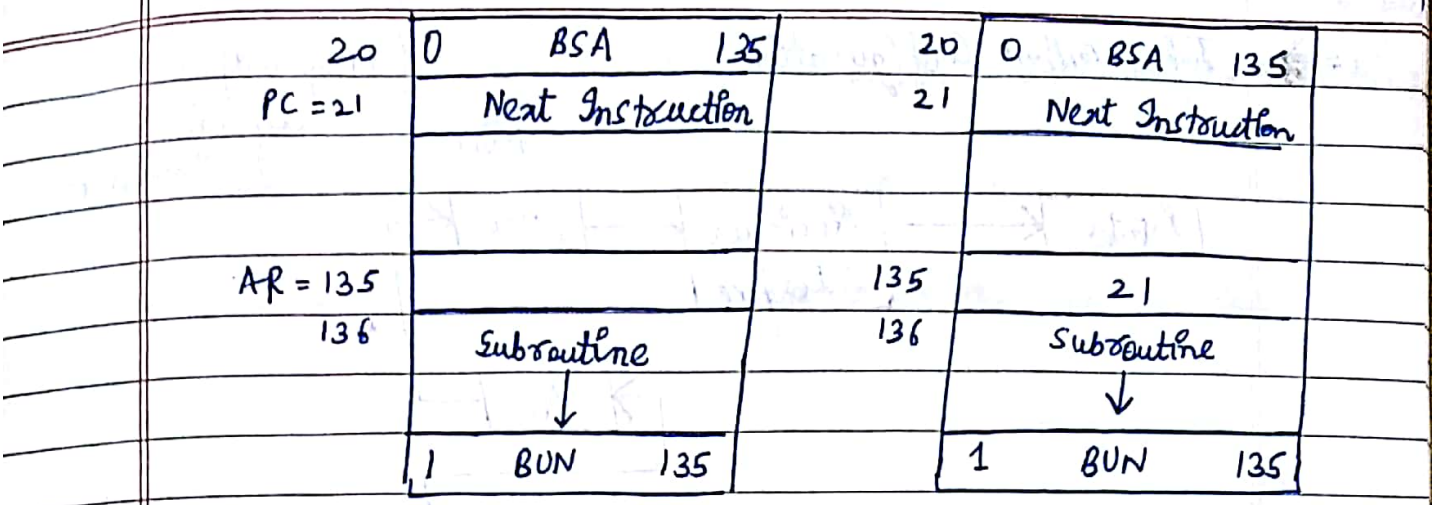
D_5T_4 : $M[AR] \leftarrow PC$, $PC \leftarrow AR + 1$

D_5T_5 : $PC \leftarrow AR$, $SC \leftarrow 0$

SARAA

Do yourself 9/10 Instruction Table

Date: / /
Page No:



★ **ISZ** (Increment & Skip if Zero)

This instruction increments the words specified by the eff. address & if the incremented value is equal to 0, PC is incremented by 1.

D_6T_4: $DR \leftarrow M[AR]$

D_6T_5: $DR \leftarrow DR + 1$ $SC \leftarrow 0$

D_6T_6: $M[AR] \leftarrow DR$, if $(DR = 0)$ $PC \leftarrow PC + 1$,

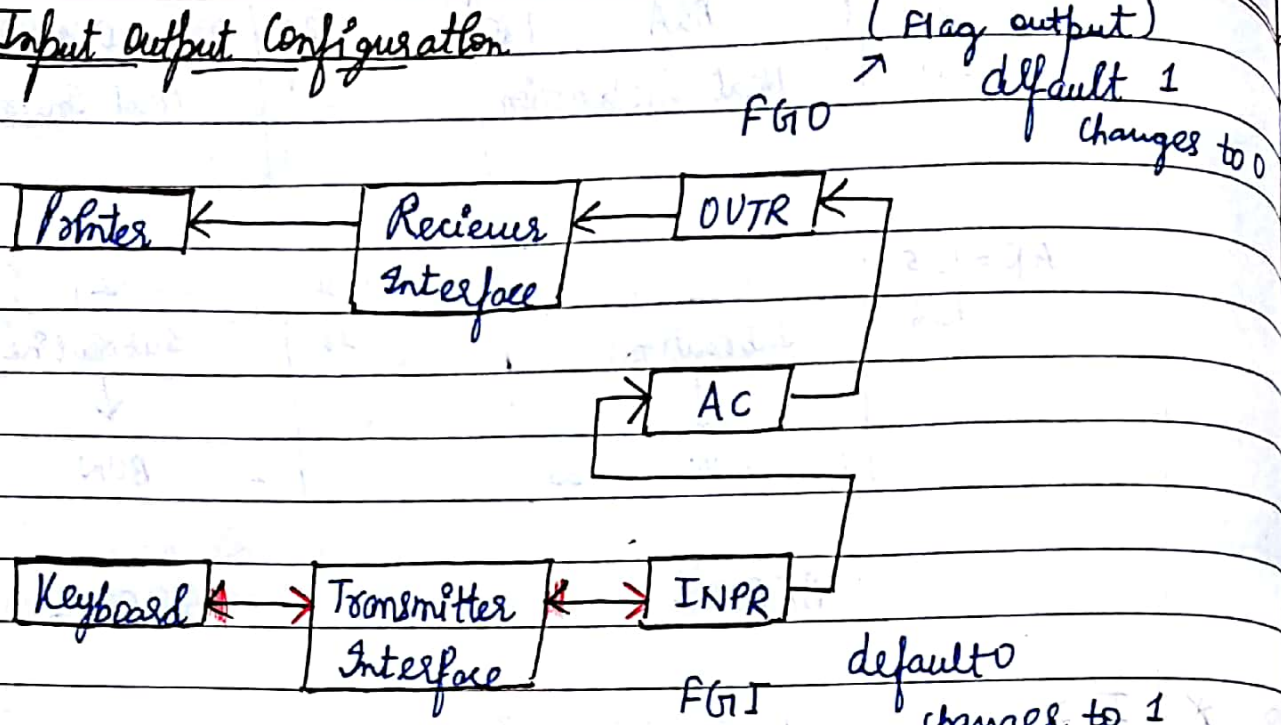
Register ref. instruction are recognized by $D_7 = 1$, $I = 0$

$I = 0$ $D_7 = 1$ $I' D_7 T_3$

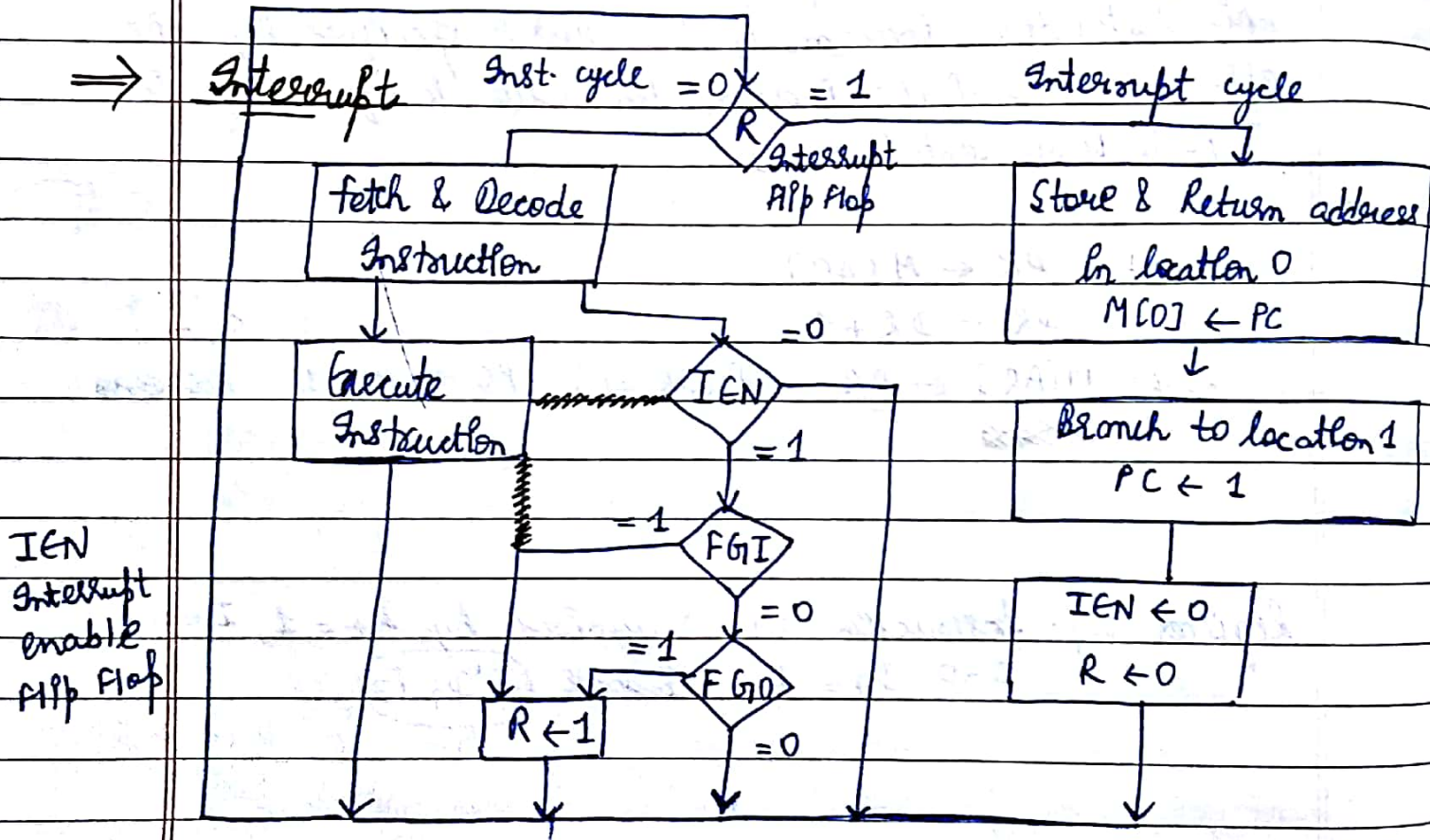
SARAA

Date : / /
Page No :

⇒ Input Output Configuration



⇒ Interrupt



Date: / /
Page No: (7 pg back)

Q. For the following control I/P active bus system is shown in fig. for each case specify the register transfer that will be executed during the next clock transition.

A.

	S_2 S_1 S_0	Ld. of Register	Memory	Adder & logic ckt.
	1 1 1	IR	Read	–
	1 1 0	PC	–	–
	1 0 0	DR	Write	–
Case ←	0 0 0	AC	–	Add

1) $IR \leftarrow M[AR]$
2) $PC \leftarrow TR$
3) $DR \leftarrow AC$
   $M[AR] \leftarrow DR$
4) $AC \leftarrow AC + DR$
   $AC \leftarrow DR + INPR$
   $AC \leftarrow INPR + AC$

Reverse of Above

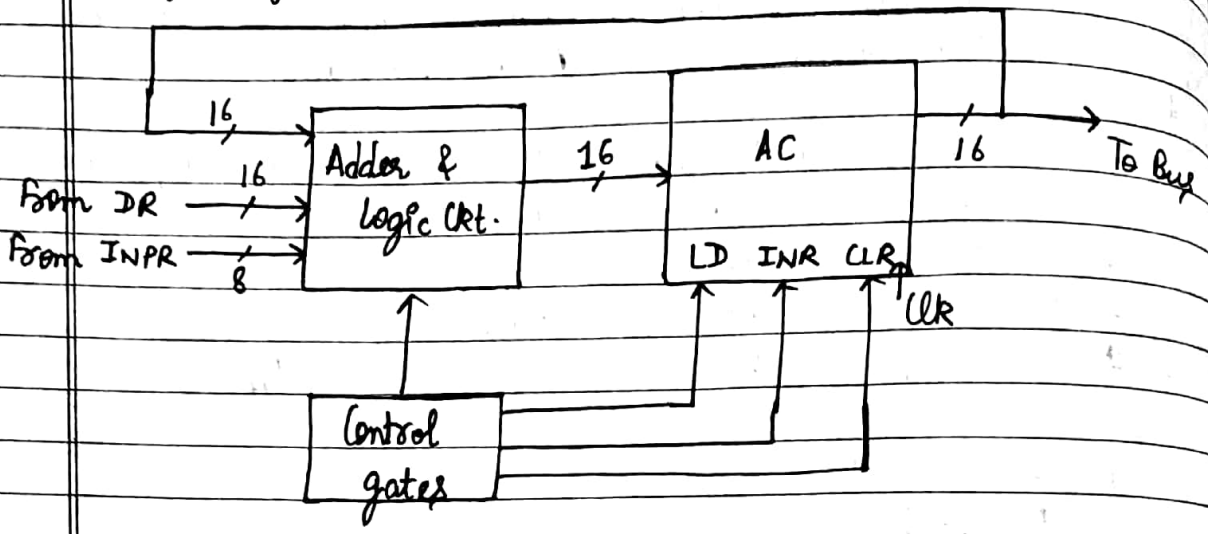
Q. ① $AR \leftarrow PC$
② $IR \leftarrow M[AR]$
③ $M[AR] \leftarrow TR$
④ $AC \leftarrow DR$, $DR \leftarrow AC$

	S_2 S_1 S_0	Ld. of Register	Memory	Adder & logic ckt.
①	0 0 1	AR	–	–
②	1 1 1	IR	Read	–
③	1 1 0	–	Write	–
④	0 1 1	DR & AC	–	Transfer

SARAA

Date : / /
Page No :

⇒ Design of a Accumulator logic



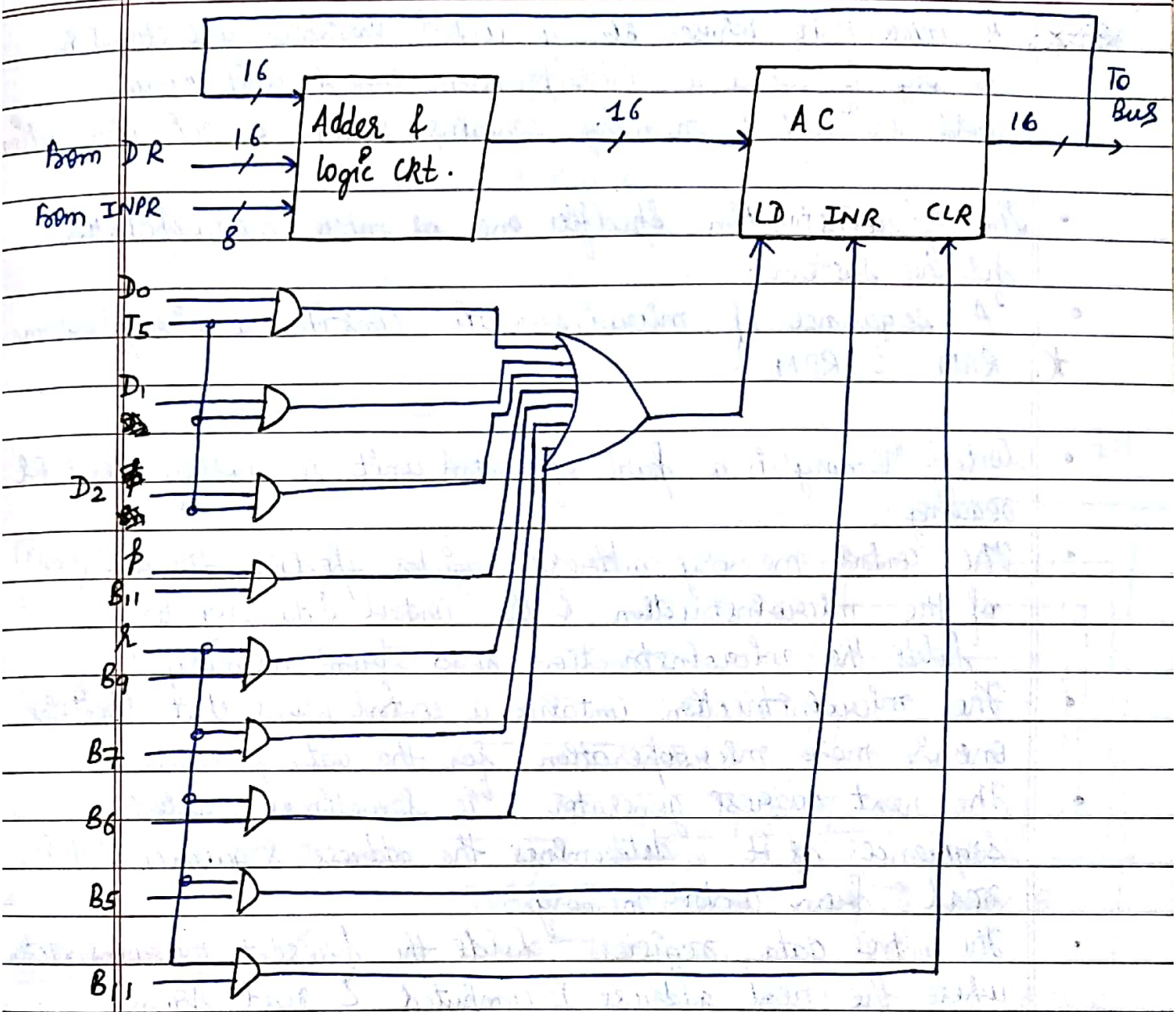
load

D_0T_5	:	$AC \leftarrow AC \wedge DR$
D_1T_5	:	$AC \leftarrow AC + DR$
D_2T_5	:	$AC \leftarrow DR$
pB_{11}	:	$AC(0-7) \leftarrow INPR$
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_{11}	:	$AC \leftarrow 0$
rB_5	:	$AC \leftarrow AC + 1$

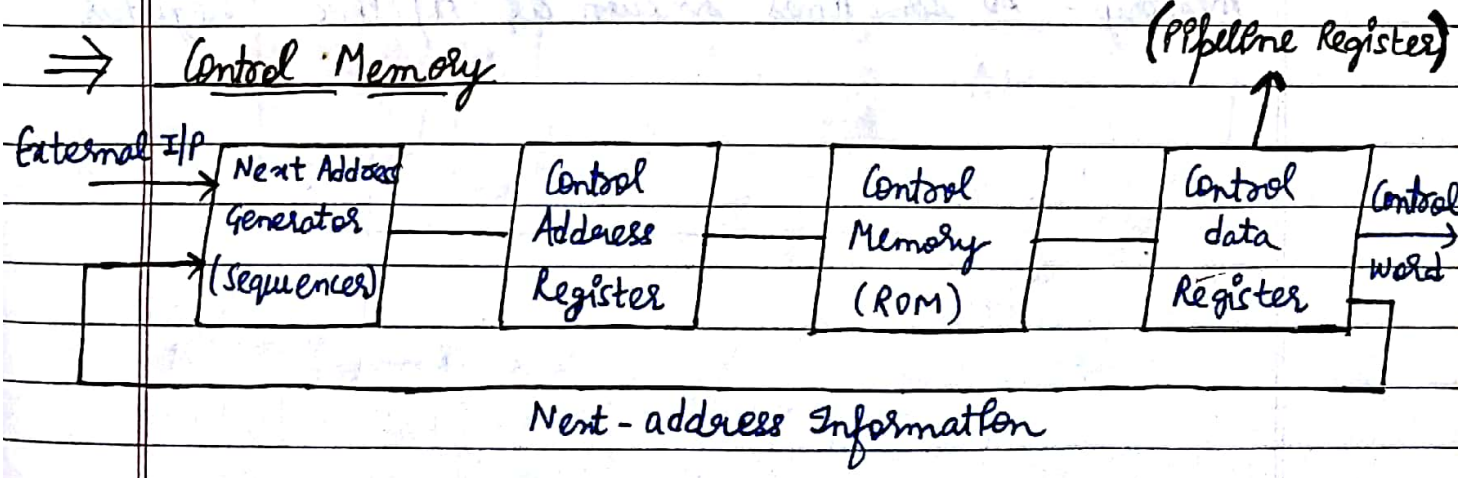
Expansion of control gates

(Next pg)

SARAA



⇒ Control Memory



Microprocessor

SARAA

Do yourself ★ Address Sequencing & Design of a Control Unit

ASSIGNMENT

Date: / /

Page No:

16/8/18

A control unit whose binary control variables are stored in memory is called a Microprogram control unit. • Each word in a control memory contains within a microinstruction.

- The microinstruction specifies one or more microoperation for the system.
- A sequence of microinstruction constitue a microprogram.

★ RAM & ROM . . .

- Control Memory is a part of control unit is mostly used for reading.
- The control memory address register specifies the address of the microinstruction & the control data register holds the microinstruction read from memory.
- The microinstruction contains a control word that specifies one or more microoperation for the data processor.
- The next address generator is sometimes called sequencer as it determines the address sequence that is read from control memory.
- The control data register holds the present microinstruction while the next address is computed & read from memory. So sometimes known as Pipeline Register.

① St
② Ex
③ Tr
④ Co

⟹ Ge

001 R1
010 R2
:
111 R7

R3 ← R1 + R2
↓ MUX A
decoder decides destination

16/8/18

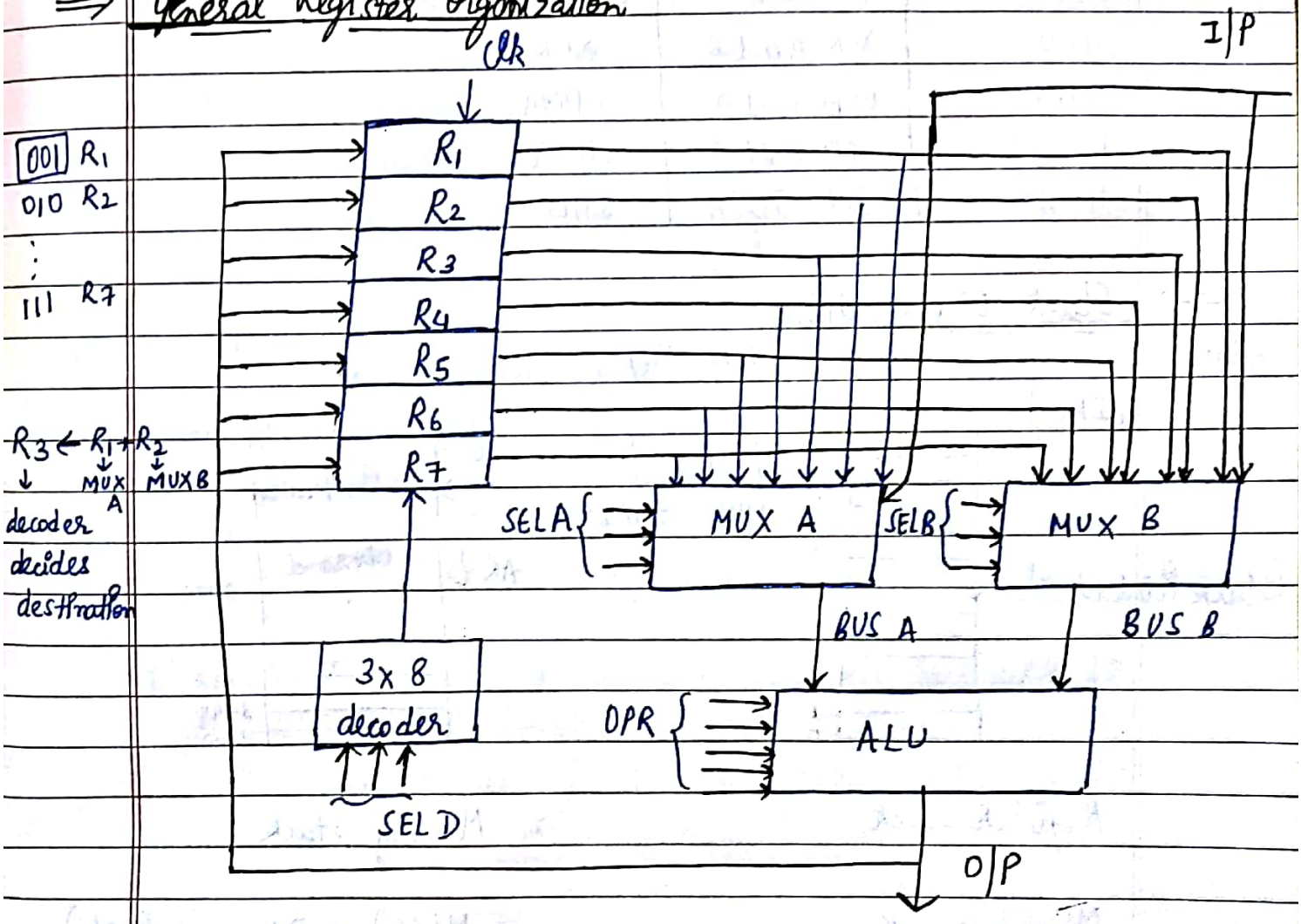
Date: / /
Page No:

MODULE - 3

CPU

1. Storage Component :- Registers, flip flops
2. Execution Component :- ALU
3. Transfer Component :- Bus
4. Control component :- Control Unit

⇒ General Register Organization



$R_3 \leftarrow R_1 + R_2$

SELA	SELB	SEL D	OPR
001	010	011	00010

SARAA

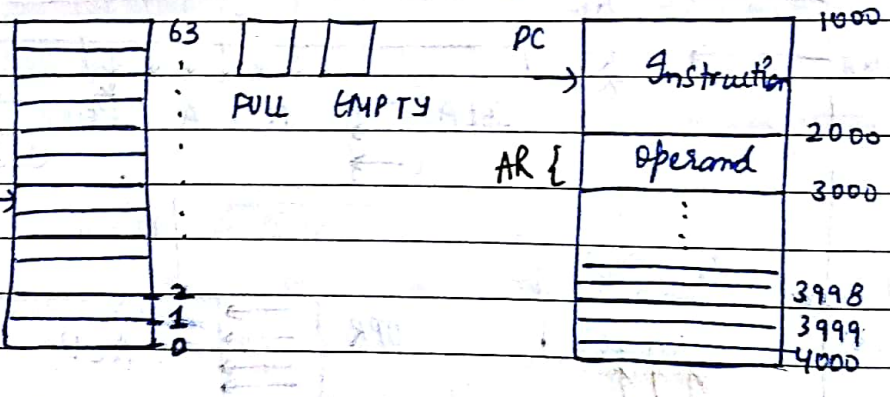
Table

OPR Selector	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A+B	ADD
00101	Subtract A-B	SUB
00110	Decrement A	DEC A
01000	AND A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA
10000	Shift right A	SHRA
11000	Shift left A	SHLA

⇒ Stack Organization

6 Bits — 2 Flip Flop

IFO



Register Stack

M[SP] ← DR

SP ← SP + 1

Memory Stack

M[SP] ← DR (Push)

SP ← SP - 1

When FULL is 1, we have to pop stack is full.

When EMPTY is 1, we have empty stack.

SARAA

MODL

Date : / /
Page No :

Q. Explain why each of the microoperation cannot be executed during a single clock pulse in the system. Specify a sequence of microoperation that will perform the operation

a) $IR \leftarrow M[PC]$

b) $AC \leftarrow AC + TR$

c) $DR \leftarrow DR + AC$ (AC not changing)

Q. Draw a timing diagram similar to before fig. assuming that SC is cleared to zero at time T_3. If the control system C_7 is active:

$$C_7 T_3 : \quad SC \leftarrow 0$$

C_7 is activated with a positive clock transition associated with T_1.

A. a) PC cannot provide address to the memory, so transfer to AC first.

$$AR \leftarrow PC$$
$$IR \leftarrow M[AR]$$

b) $DR \leftarrow TR$ Add operation must be done with DR. Transfer TR to DR first.

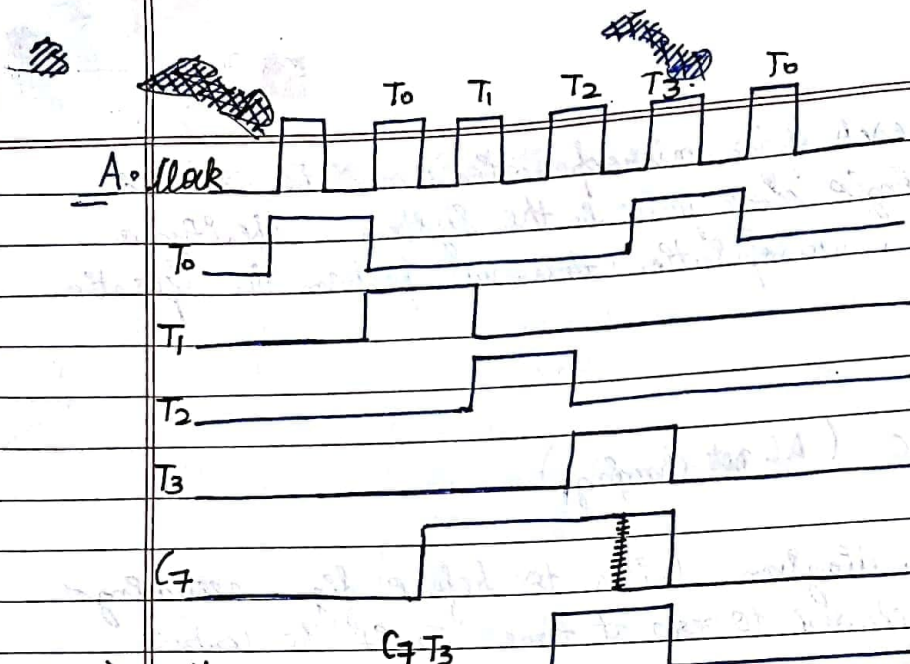
$$AC \leftarrow AC + DR$$

c) $AC \leftarrow DR$, $DR \leftarrow AC$

$$AC \leftarrow AC + DR$$

$$AC \leftarrow DR, \quad DR \leftarrow AC$$

Date: / /
Page No:



⇒ **Half Subtractor**

A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Difference = $A\bar{B} + \bar{A}B = A \oplus B$ ⇒

Borrow = $\bar{A}B$ ⇒

⇒ **Full Subtractor**

A	B	C	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Difference = $\bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

= [illegible]

= A ⇒

⇒

* Sub o ⇒

(i)
(ii)
(iii)

(i)

(ii)

Using 3
Add
Instructio

SARAA

Date: / /
Page No:

* Sup 0

⇒ Instruction format

CPU organization can be of 3 types -

(i) Single accumulator organization
(ii) General Register organization
(iii) Stack Organization

(i) ADD A

$AC \leftarrow AC + M[A]$

(ii) Add R_1, R_2, R_3 (Destination) — Add R_1, R_3

$R_1 \leftarrow R_2 + R_3$ — $R_1 \leftarrow R_1 + R_3$

⇒ In single accumulator organization the Instruction format in this type of computer uses 1 address field.

⇒ • In general register type the computer needs 3 register address field ex- Add R_1, R_2, R_3
• In this the no. of address fields can be reduced from 3 to 2 if the destination register is one of the same as source register. Ex- Add R_1, R_3
• Add R_1, A — MOV R_1, R_2

$R_1 \leftarrow R_1 + M[A]$ — $R_1 \leftarrow R_2$

⇒ In stack organization, the Instruction will be like
PUSH X store in stack (Implemented stack)

⇒ 3-address Instruction or Registers

$(A+B) * (D+E)$

Using 3 Address Instruction

ADD R_1, A, B — $R_1 \leftarrow M[A] + M[B]$

ADD R_2, D, E — $R_2 \leftarrow M[D] + M[E]$

MUL X, R_1, R_2 — $X \leftarrow R_1 * R_2$

SARAA

Date: / /
Page No:

In the Instruction format we will use ADD, SUB, MUL, DIV for operation. MOV for transfer type operation & LOAD and STORE for transfer to & from memory & AC register.

Using 2 Add. Inst.

```
MOV  R1, A      R1 ← M[A]
ADD  R1, B      R1 ← R1 + M[B]

MOV  R2, D      R2 ← M[D]
ADD  R2, E      R2 ← R2 + M[E]

MUL  R1, R2     R1 ← R1 * R2

MOV  X, R1      X ← R1
```

The advantage of 3 address format is that it results in short program when evaluating arithmetic operation. The disadvantage is that the binary coded instruction require too many bits to specify 3 addresses.

Using 1

```
LOAD   A
ADD    B
STORE  T
LOAD   D
ADD    E
MUL    T
STORE  X
```

Zero address Organization

SARAA

01
a)
b)
c)
d)
e)

02
a)
b)
c)
d)
e)

03
a)
b)
c)
d)

04
a)
b)

SELA	..B	..D	OPR

Date: / /
Page No:

Mod 3

Q1 Specify the control word that must be applied to the processor of fig to implement the following microoperations-

a) $R_1 \leftarrow R_2 + R_3$
b) $R_4 \leftarrow \overline{R_4}$
c) $R_5 \leftarrow R_5 - 1$
d) $R_6 \leftarrow shl\ R_1$
e) $R_7 \leftarrow I/P$

Q2. Determine the microoperations that will be executed in the processor of fig (same), when the following 14 bit control word are applied -

a) 001 010 011 00101
b) 000 000 000 00000
c) 010 010 010 011 00
d) 000 001 000 0001 0
e) 111 100 011 10000

Q3. Convert the following from Infix to Reverse Polish notation :-

a) $A * B + C * D + E * F$
b) $A * B + A * (B * D + C * E)$
c) $A + B * [C * D + E * (F + G)]$
d) $\dfrac{A * [B + C * (D + E)]}{F * (G + H)}$

Q4. Convert the following arithmetic expressions from Rev. Polish to infix :-

a) ABCDE + * - /
b) ABCDE * / - +
c) ABC * / D - EF / +
d) ABCDEFG + * + * + *

SARAA

Date : / /
Page No :

3=D

Q5. Convert the following arithmetic exp. into reverse polish notation & show the stack operations for evaluating the results :- A3.

a) $(3+4)[10(2+6)+8]$

Imp
* Q6. WAP to evaluate the arithmetic statement :-

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

a) Using a general register computer with 3 address instruction
b) . . . . . 2 address . . .
c) Using an AC type computer with 1 address . . .
d) Using a stack organized operation with 0 address . . .

Solutions

	SELA	SELB	SELD	OPR
A1. a)	010	011	001	00010
b)	100	XXX	100	01110
c)	101	XXX	101	00110
d)	001	XXX	110	11000
e)	000	XXX	111	00000

A2. a) $R3 \leftarrow R1 - R2$
b) Out. $\leftarrow$ Inp
c) $R2 \leftarrow R2 \oplus R2$
d) Out. $\leftarrow$ Inp + R1
e) $R3 \leftarrow shr\ R7$

SARAA

Date : / /
Page No :

A3. a) $A * B + C * D + E * F$

$= AB* + CD* + EF*$

$= AB* + CD* EF* +$

$= AB* CD* EF* +$

b) $A * B + A * (B * D + C * E)$

$= A * B + A * BD* + CE*$

$= A * B + A * BD* CE* +$

$= A * B + ABD* CE* + *$

$= A * B \; ABD * CE * + * +$

$= AB * ABD * CE * + * +$

c) $A + B * [C * D + E * (F + G)]$

$= A + B * [CD* + EFG+*]$

$= A + B * [CD* EFG+*+]$

$= A + BCD * EFG+*+*$

$= ABCD * EFG+*+*+$

d) $\dfrac{A * [B + C * (D+E)]}{F * (G+H)}$

$= \dfrac{A * [B + C * DE+]}{F * GH+}$

$= \dfrac{A * [B + CDE+*]}{FGH+*}$

$= \dfrac{A * [BCDE+*+]}{FGH+*}$

$= ABCDE+*+* FGH+*/$

SARAA

Date : / /
Page No :

A4. a) ABCDE + * - /

= A B C D+E * - /

= A B C * (D+E) - /

= A B - C * (D+E) /

= $\frac{A}{[B - C * (D+E)]}$

A5.

b) ABCDE * / - +

= A B C D*E / - +

= A B $\underset{(D*E)}{C}$ - +

= A B - $\underset{(D*E)}{C}$ +

= A + B - $\underset{(D*E)}{C}$

A6.

c) ABC * / D - EF / +

= A B * C / D - $\frac{E}{F}$ +

= $\frac{A}{B*C}$ D - $\frac{E}{F}$ +

= $\frac{A}{B*C}$ - D $\frac{E}{F}$ +

= $\frac{A}{B*C}$ - D + $\frac{E}{F}$

d) ABCDEFG + * + * + *

= ABCD E (F+G) * + * + *

= ABCD(E * (F+G)) + * + *

= ABC(D + E * (F+G))) * + *

= A B(C * (D + E * (F+G))) + *

= A B + C * (D + E * (F+G))) *

SARAA = A * (B + C * (D + E * (F+G)))

Date: / /
Page No:

A5. $(3+4)\ [10\ (2+6)+8] = 616$

				6		10		8			
	4		2	2	8	8	80	80	88		
3	3	7	7	7	7	7	7	7	7	616	= 616
3	4	+	2	6	+	10	*	8	+	*	

A6. $$X = \frac{A - B + C * (D * E - F)}{G + H * K}$$

a)

MUL R1, D, E	SUB R5, A, R4
SUB R2, R1, F	MUL R6, H, K
MUL R3, R2, C	ADD R7, R6, G
ADD R4, R3, B	DIV X, R5, R7

b)

MOV R1, D	SUB R2, R1
MUL R1, E	MOV R3, H
SUB R1, F	MUL R3, K
MUL R1, C	ADD R3, G
ADD R1, B	DIV R2, R3
~~SUB A R1~~	MOV X, R2
MOV R2, A	

c)

LOAD D	LOAD H
MUL E	MUL K
SUB F	ADD G
~~ADD~~ MUL C	STORE T3
ADD B	LOAD T2
STORE T1	DIV T3
LOAD A	STORE X
SUB T1	
STORE T2	

SARAA

Info

Date: / /
Page No:

10)

Num Q.

⇒ ADDRESSING MODES

1) **Implied Mode** - In this mode the operands are specified implicitly in the def. of the Instruction. For Ex- Complement, Increment Accumulator.

2) **Immediate Mode** - In the Immediate mode, the operand is specified in the Instruction itself.

3) **Register Mode** - In this mode the operands are in registers that reside within the CPU

4) **Register Indirect mode** - In this mode the Instructions specifies a register in the CPU whose content give the address of the operand in the memory.

5) **Auto Increment / Decrement Mode** - In auto increment mode the content of register is incremented after the execution of the Instruction.
In Auto decrement mode the content of register is decrement ~~of~~ before the execution of the Instruction.

6) **Direct Address mode** - In this mode the effective add. is equal to the address part of the Instruction.

7) **Indirect Address mode** - In this mode the address field of the Instruction gives the address where the effective address is stored in the memory.

8) **Relative Address mode** - PC + Address = Effective address

9) **Indexed Address mode** - IR + Address = Effective address
Indexed Register of Instruction

SARAA

Date : / /
Page No :

10) Base Register Addressing mode - BR + Address part of Instruction = Effective address
(BR = Base)

Num.
Q.

200	load to AC	Mode
201	Address = 500	
202	Next Instruction	
399	450	
400	700	
500	800	
600	900	
~~702~~	~~325~~	
~~800~~	300	

PC: 200

R_1: 400

XR (Index): 100

AC

BR: 250

Mode	Effective Address	Operand [AC]
Immediate	201	500
Direct	500	800
Indirect	800	300
Relative	702	325
Indexed	600	900
Auto Inc.	400	700
Auto Dec.	399	450
Register D	–	400
Register I	400	700
Base Register	750	479

SARAA

Date: / /
Page No:

Q. An instruction is stored at location 300 with its address field at 301. The address field has the value 400. A processor register R_1 contains 200. Evaluate the effective address if addressing mode is

(i)	Direct	400
(ii)	Immediate	301
(iii)	Relative	702
(iv)	Register Indirect	200
(v)	Index with R_1 as IR	600

⟹ Data Transfer & Manipulation

Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP

- The LOAD instruction is used to designate a transfer from memory to a processor register.
- The STORE inst. is from processor register into memory
- MOVE to transfer from 1 register to another.
- EXCHANGE swaps data b/w 2 registers or register and a memory word.
- Input output Inst. transfers data in a processor register & I/O terminals.
- PUSH & POP transfer data b/w processor register & memory stack.

(i)

INC
DEC
ADD
SUB
MUL
DIV
ADDC
SUBB
NEG

SARAA

Date : / /
Page No :

Load Inst. using different modes

Direct Address	LD ADR	AC ← M[ADR]
Indirect	LD @ ADR	AC ← M[M[ADR]]
Relative	LD $ ADR	AC ← M[PC+ ADR]
Immediate Operand	LD #	AC ← NBR (data)
Index Address	LD ADR (X)	AC ← M[ADR + R_1]
Register	LD R_1	AC ← R_1
Register Indirect	LD(R_1)	AC ← M[R_1]
Auto Increment	LD (R_1)+	AC ← M[R_1], R_1 ← R_1+1

Data Manipulation

Data manipulation Instruction performs operations on the data. It can be divided into 3 basic types:-

(i) Arithmetic (ii) Logical & Bit Manipulation (iii) Shift Instructions

INC	Increment
DEC	Decrement
ADD	ADD
SUB	Subtract
MUL	Multiply
DIV	Divide
ADDC	Add with carry
SUBB	Sub. with borrow
NEG	Negate (2's compl.)

SARAA

Logical & Bit Manipulation

CLR	Clear
COM	Complement
AND	AND
OR	OR
XOR	XOR
CLRC	Clear carry
SETC	Set Carry
COMC	Complement Carry
EI	Enable Interrupt
DI	Disable Interrupt

Q. A 2-word Instruction is stored in memory at an address designated by symbol W. The address field of the Instruction stored at W+1 is designated by Y. The operand used during execution of the Instr. is stored at an address symbolized by Z & Index register contains value X. State how Z is calculated from other addresses, if the addressing mode is

1) Direct — Y
2) Indirect — M[Y]
3) Relative — W+2+Y
4) Indexed — X+Y

Q. WAP to evaluate the arithmetic Statement

$$X = A * (B + C) + D / E * F + G$$

Using a general register comp. with 3 operand Instructions, 2, 1 operand (AC type computer with 1 operand Inst.)
Using a Stack organized computer with 0 add. Instructions.
Do not modify the values of A, B, C, D, E, F, G. Use a temporary locating T to store intermediate result if necessary

SARAA

Q. Explain various types of logical operations available. Starting from an initial value of R = 11011101. Determine the sequence of binary values in R after each operation in the sequence.
(i) A logical shift left followed by (ii) Arith. shift right (iii) followed by another Arit. shift right iv) Cir shift left Show all your work. Repeat the same for R = 00111100.

Q. A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers.
a) How many selection lines (Inputs) for each MUX.
b) What size of multiplexers are needed.
b) How many multiplexers are there in the bus.

Q. What are various addressing modes? Explain any 5 with examples.

Q. Draw flowchart for Instruction cycle of computer & explain it.

Q. Convert from Infix to Reverse polish:

(i) A * B + (C * D)

(ii) A * (B * D) + C * E)

Q. What are registers? Mention various types of register & explain any 2.

Q. What is the diff. b/w logical, Circular & Arithmetic shift

Q. Diff. b/w Hardwired vs Microprogrammed control unit.
RISC & CISC process

Q. Draw a common bus system using Multiplexers as well as 3 state buffers

SARAA

Date : / /
Page No :

~~Qo~~ ~~An Instruction is stored at location 200.~~

Qo Design a 4-bit combitional circuit using 4-bit full adder.

Qo Define :- i) Microoperations
ii) Microinstructions
iii) Microprogram

Qo What is the diff. b/w direct & Indirect Address Instruction? How many references to memory are needed for each type of instruction to bring an operand into a process register?

⟹ PROGRAM CONTROL INSTRUCTIONS

(Direct) (Indirect Add.)

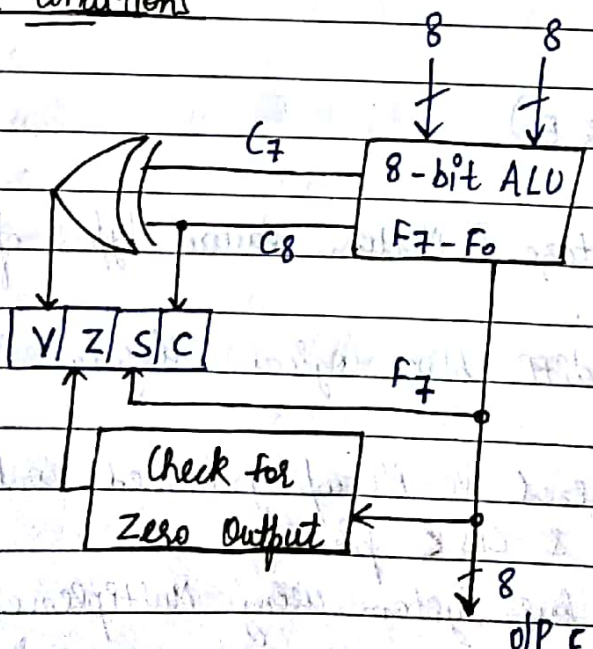
Branch	BR	} Conditional, unconditional
Jump	JMP	
Skip	SKP	⇒ zero Address
Call	CALL	} Used in
Return	RET	Subroutines
Compare	CMP	(by subtraction)
Test	TST	(by ANDing)

Status Bit Conditions

or

Conditional Code or

Flag Bits



Carry
Signed Bit
Zero output
Overflow

SARAA

Date : / /
Page No :

Status Register is used in the CPU where status bit conditions can be stored for further analysis.

1) Bit C - It is set to 1, if the end carry C_8 is 1, it is clear to 0 if C_8 is 0.
2) Bit S - It is set to 1, if the highest order bit F_7 is 1
3) Bit Z - It is set to 1, if the output of ALU contains all zeros.
4) Bit V - It is set to 1, if the XOR of the last 2 carries is equal to 1 & clears to 0 otherwise.

for 8 bit ALU, V = 1 if the output is greater than +127 or less than -128.

C_0 C_1 C_2 C_3 ... C_7

Examples: 0. A = 11110000
B = 0001 0100

A - B

$\bar{B}$ = 1110 1011
1
11101100
C_7 00
A 11110000
$\bar{B}+1$ 11101100
1 11101.11,00
C_8

Here $C_7 = 1$
$C_8 = 1$

∴ Values V = 0
Z = 0
$F_7 = 1$ S = 1
C = 1

SARAA

Scanned by CamScanner

Date : / /
Page No :

⇒ Subroutine Call & Return

Memory Stack is used.

SP ← SP - 1
M[SP] ← PC
PC ← effective address sub.
PC ← M[SP]
SP ← SP + 1

★ Program Interrupt

Program Interrupt refers to the transfer of from a currently running program to another service program, as a result of an external or internal generated request.
The Interrupt procedure is quite similar to a Subroutine call except for 3 variations. 1) The Interrupt is usually initiated by an Internal or external signal rather than from the execution of the Instruction (except for the Software Interrupt)
2) The address of the Interrupt Service program is determined by the hardware rather than from the address field of an Instruction.
3) An Interrupt procedure usually stores all the information necessary to define the state of CPU rather than storing only the program counter.

Types of Interrupts

1) Internal Interrupts are also known as traps. Register overflow, stack overflow, divide by zero etc.

SARAA

Date: / /
Page No:

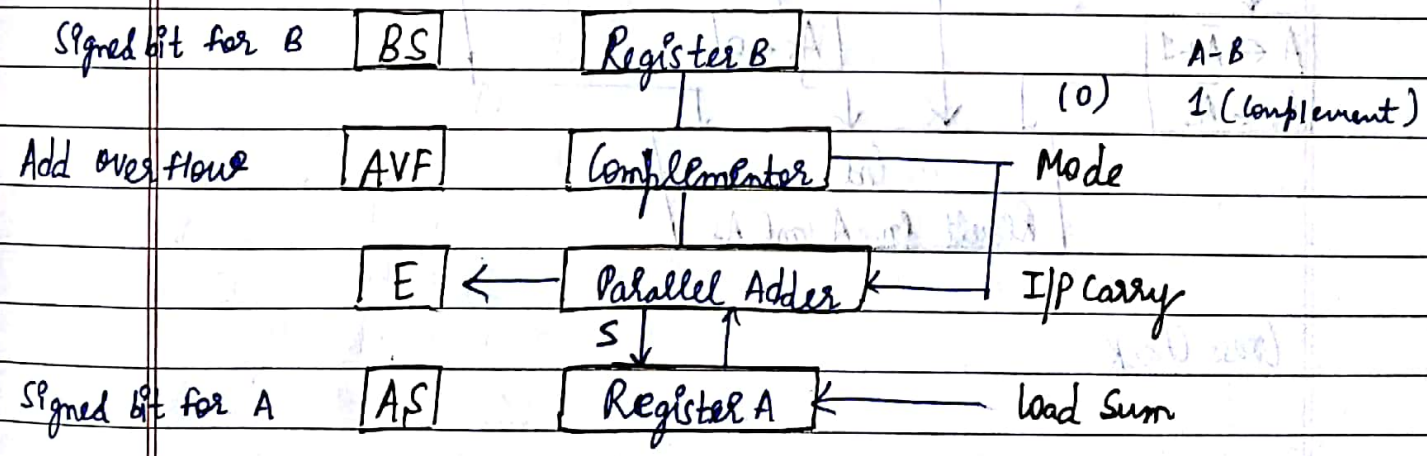
2) External Interrupt → power supply & (I/O)
3) Software Interrupt

⇒ **Computer Arithmetic**

Addition & Subtraction of Signed magnitude numbers (Sub)

	ADD	A > B	A < B	A = B
(+A) + (+B)	+(A+B)			
(+A) + (-B)		+(A-B)	-(B-A)	+(A-B)
(-A) + (+B)		-(A-B)	+(B-A)	+(A-B)
(-A) + (-B)	-(A+B)			
(+A) - (+B)		+(A-B)	-(B-A)	+(A-B)
(+A) - (-B)	+(A+B)			
(-A) - (+B)	-(A+B)			
(-A) - (-B)		-(A-B)	+(B-A)	+(A-B)

Hardware Implementation

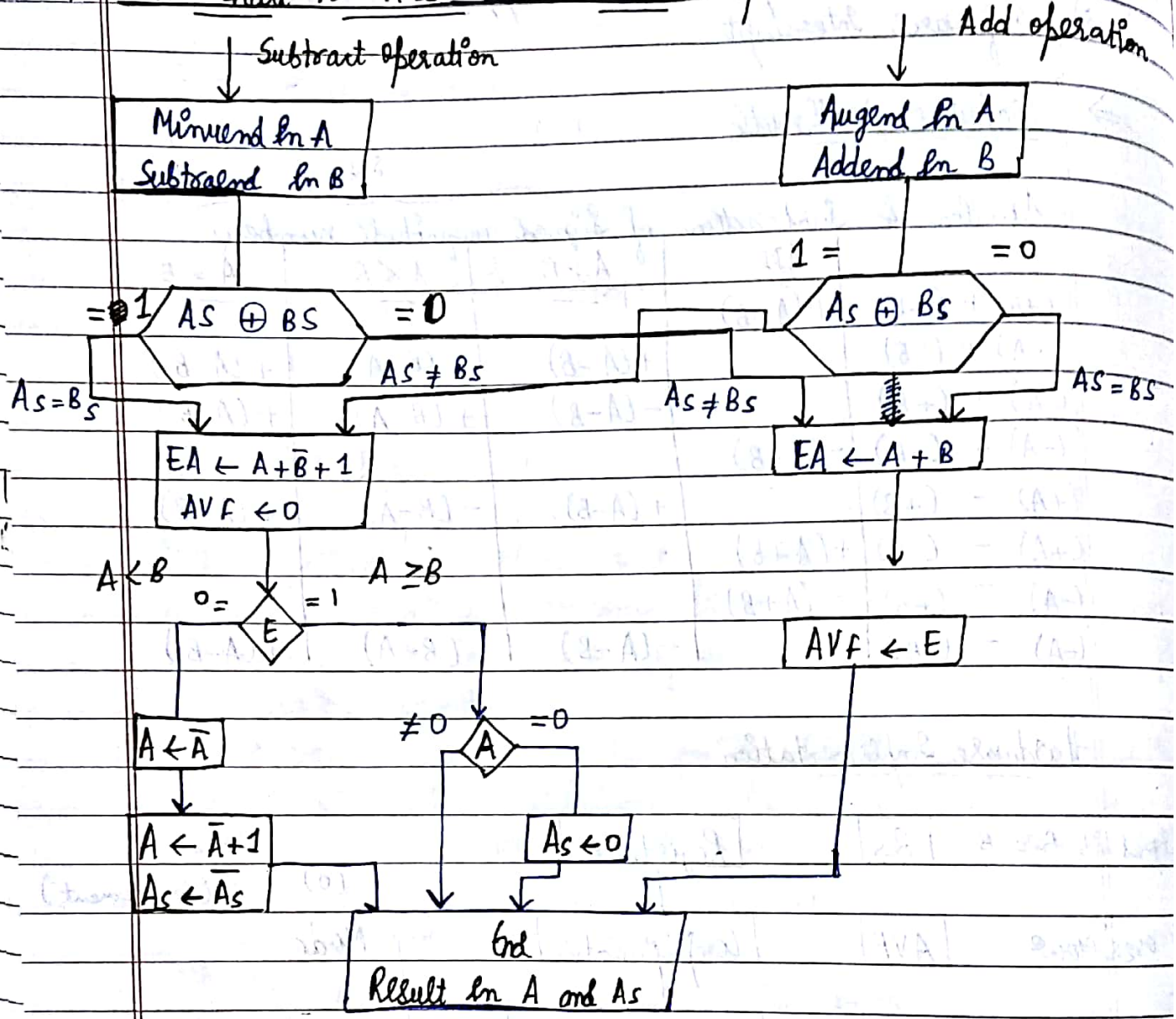


SARAA

3-0

Date: / /
Page No:

⇒ Flow Chart for Add & Subtract Microoperation



Cross Check

SARAA

Similarly for All

Do till 18 only

Date: / /

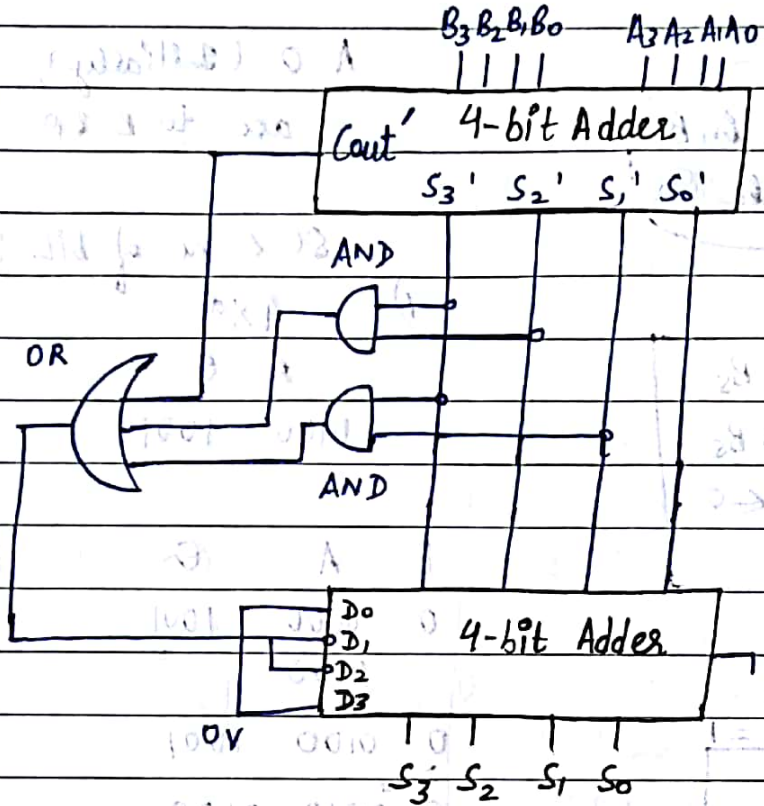
Page No:

⇒ BCD Adder

0-9 → 0000 - 1001

10 → 10000

11 → 10001



9+8

1001

1000

10001

0001

0110

10111

7

3 → Multiplicand → B
× 9 → Multiplier → Q

Date: / /
Page No:

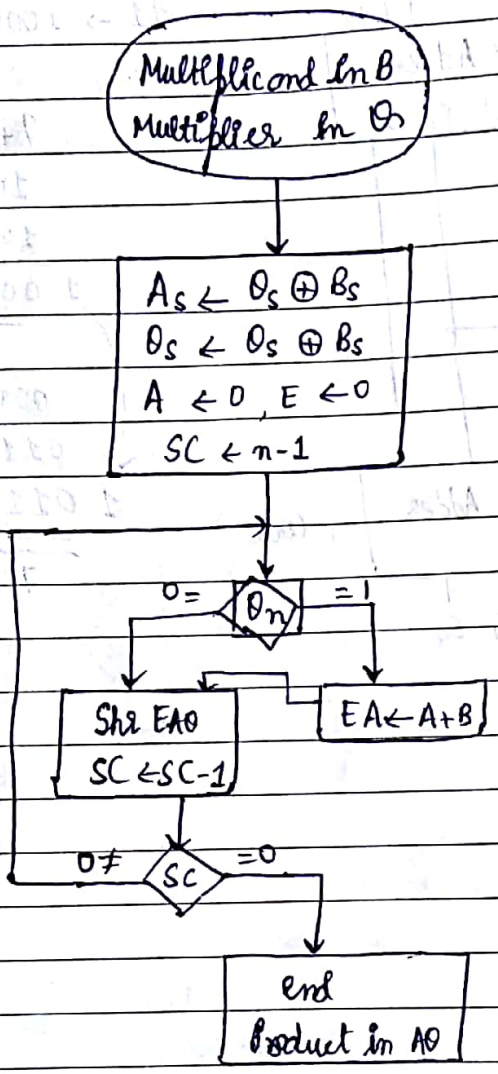
25

⇒ Multiplication Algorithm

E = 0 (1 bit flip flop)

A = 0 (Initially), padded acc. to B & Q ∴ A = 0000

SC ← no. of bits Here 4



1) 4 × 9

B Q

0100 1001

E	A	Q	SC
0	0000	1001	4
	0100		
0	0100	1001	
0	0010	0100	
0	0001	0010	3
0	0000	1001	2
	0100		1
0	0100	1001	
0	0010	0100	
	36		0

2) 23 × 19

B Q

10111 10011

E	A	Q	SC
0	00000	10011	5
	10111		
0	10111	10011	
0	01011	11001	
	10111		4
~~1~~	~~00010~~	~~11001~~	
0	11110	11001	
0	01111	01100	
0	00111	10110	3
0	00011	11011	2

SARAA

Date : / /
Page No :

0 00011 11011
10111
0

26 X 29
11010 X 11101
B Q

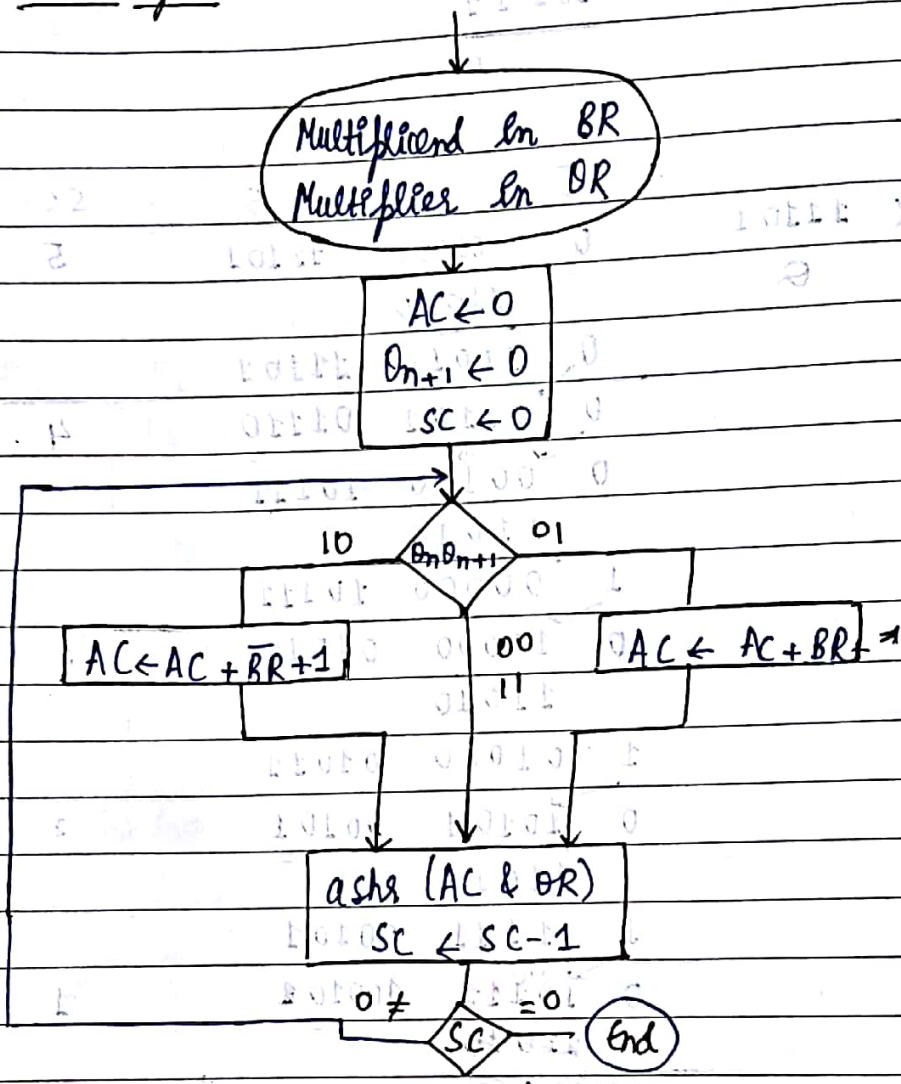
	E	A	Q	SC
	0	00000	11101	5
		11010		
	0	11010	11101	
	0	01101	01110	4
	0	00110	10111	
		11010		
	1	00000	10111	
	0	10000	01011	3
		11010		
	1	01010	01011	
	0	10101	00101	2
		11010		
	1	01111	00101	
	0	10111	10010	1
	~~1~~	~~10001~~	~~1001~~	
	0	01011	10010	0

BIT 5 4 3 2 1 0

SARAA

Ques (Multiplication of signed No.)

⇒ Booth's Algorithm



Example - +9, +2 → 01001 (BR), 00010 (QR)

AC	QR	Q_{n+1}	SC
00000	00010	0	5
(10) 00000	00001	0	
			4

BR ⇒ 01001

10110

1

10111 ← AC + BR

10111	00001	1	
(01) 11011	10000	1	

SARAA

Date : / /
Page No :

	AC	OR	θ_{n+1}	SC	
		01001			
		11011			
		100100			
(00)	00100	10000	0	3	
	00010	01000	0	2	
	00001	00100	0	1	
	00000	10010 (18)	0	0	
(ii)	$-2, +5$				
	BR OR				
	$-2 \Rightarrow$ 2's Complement of $+2 \Rightarrow 0010 \Rightarrow 1101 + 1 \Rightarrow 1110$				
	$BR = 1110$, $OR = 0101$ $\overline{BR}+1 = 0010$				
	AC	OR	θ_{n+1}	SC	
(10)	0000	0101	0	4	
	0010	0101	0	4	
(01)	0001	0010	1	3	
	AC 0001				
	BR 1110				
	1111				
	1111	0010	1	3	
(10)	1111	1001	0	2	
	AC 1111				
	$\overline{BR}+1$ 0010				
	0001				
	0001	1001	0	2	
(01)	0000	1100	1	1	AC 0000
	1110	1100	1	1	BR 1110
	SARAA				1110
	1111	0110	0	0	

Date: / /
Page No:

Q1

= (iii) (-2), (-5)

BR OR

BR ⇒ 0010 ⇒ 1101
1
1110

OR ⇒ 0101 ⇒ 1010
1
1011

BR ⇒ 1110 OR ⇒ 1011 $\overline{BR}+1 = 0010$

	AC	OR	θ_{n+1}	SC
	0000	1011	0	4
(10)	0010	1011	0	4
(11)	0001	0101	1	3
(01)	0000	1010	1	2

AC 0000
BR 1110
1110

	1110	1010	1	2
(10)	1111	0101	0	1

AC 1111
$\overline{BR}+1$ 0010
0001

(1(	0001	0101	0	1
	0000	1010	1	0

1010 ⇒ +10

Hence verified A1

To

SARAA

Scanned by CamScanner