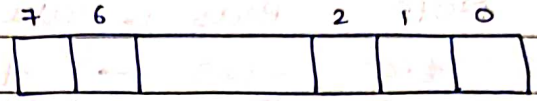
Comp. Architecture

__/__/__

Signed & Unsigned number

8 bits : 0 to 256 (256 different no's)
$\because 2^8 = 256$

- Signed bits : Most significant bit is used to represent +ve or -ve.



↳ significant bit

Range -1 to -128
0 to 127

- Other way is to take 2's complement of a no to get it's -ve counterpart.

E.g +5 : 0 0 0 0 0 1 0 1
1 1 1 1 1 0 1 1

→ Go form right to left & start changing 0 to 1 & 1 to 0 after the first 'one'.

* IEEE Standard : 754 floating Point no representation
- Fixed point : 25.75
- Floating point : $\pm 25.75 * 10^{\pm 15}$ (for small & very large value)

* Storage Layout :

IEEE floating point no's have 3 basic components. The sign, the exponent & the mantissa.

sign ← ± 1.25 × 10^4 → exponent
↳ mantissa

The Mantissa is composed of a fraction & implicit leading digit.

1.25
leading point fraction

__/__/____

	Sign	Exponent	Fraction	Bias
32 bit Single Precision	1[31]	8[30-23]	23[22-00]	127
64 bit Double Precision	1[63]	11[62-52]	52[51-00]	1023

* Note : Bias is always added in exponent.

e.g $-123_{10} \rightarrow -123+127 = 4_{10}$

* Note : In fraction, the leading bit is always '1' thus it is not stored in the system. All the 23/52 bits go to decimal part.

e.g 1.1101×10^4 $1.000101001 \times 10^{15}$ .....

Q. Convert to IEEE 754 standard

→ $(37.5)_{10}$

S1 convert to binary

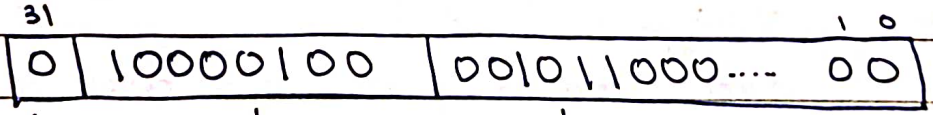
$(100101.1)_2 \Rightarrow 1.001011 \times 2^5$

S3 sign : 0

Exponent : $5 + 127 = 132 = (10000100)_2$

fraction : 001011

S4



→ -78.25

S1 $78 \rightarrow 1001110 \rightarrow 1.00111001 \times 2^6$

$0.25 \rightarrow 0.01$

__/__/____

Step-3 sign : 1

Exponent: $6 + 127 = 133 = (10000101)_2$

Fraction : 00111001

31		1 0
1	10000101	00111001 00 00

* Register Transfer & Micro Operations :-

- A digital system is an interconnection of digital H/w modules. The modules are constructed from digital components such as Registers, decoders, arithmetic elements & control logic.

- Various modules are interconnected with common data & control paths to & from a digital comp. system.

* Micro Operations :-

- Registers contain the data.
- The operations executed on data stored in registers are called micro operations.
- The result of operations made, replace the previous binary info. of register or may be transferred to another register.
- Example:
  - Shift , - count , - clear , - load

* Register Transfer :-

- Different types of Register in basic Computer.

① MAR : Memory Address Register
② PC : Program Counter
↳ stores the address of next instance to be executed
③ IR : Instruction Register
↳ stores instructions for operation

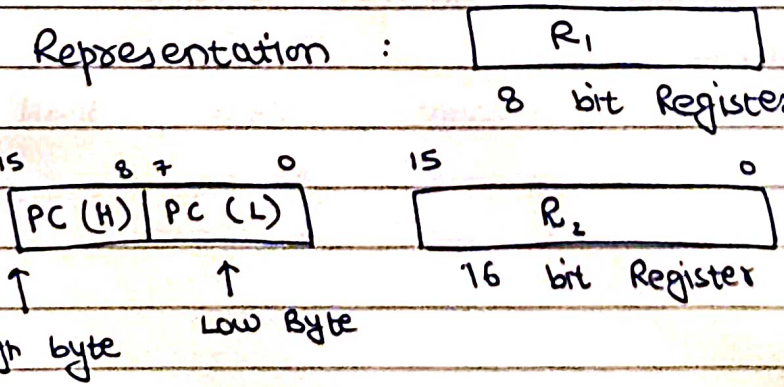
④ R_1, R_2, R_3 .... : Registers to store basic information (Processor Register)

- The information transferred from one register to another, designated in symbolic form by replacement operator.

- It denotes a transfer of the content of register R_1 to R_2.

e.g $R_2 \leftarrow R_1$

- Representation :



__/__/____

* Error Detection Codes
  - detect any change in binary representation / transmission of data.
  - Detect only change in single bit.
  - An Error ~~ction~~ detection detection code that detects digital error during the transmission
  - The Error is occured due to external noise during transmission that could change bits from $0 \rightarrow 1$ or $1 \rightarrow 0$

* Parity Bit

A parity bit is an extra bit include with a binary message to make total number of 1's in binary value = 0's in binary value. It is most common EDC

A message of 3 bit and 2 posibble parity bits is shown in table | Even & odd

Message	P(odd)	P(even)
0 0 0	1	0
0 0 1	0	1
0 1 0	0	1
0 1 1	1	0
1 0 0	0	1
1 0 1	1	0
1 1 0	1	0
1 1 1	0	1

Parity (odd) make no of one (1) in binary representation odd

__/__/__

* **Parity Generator**

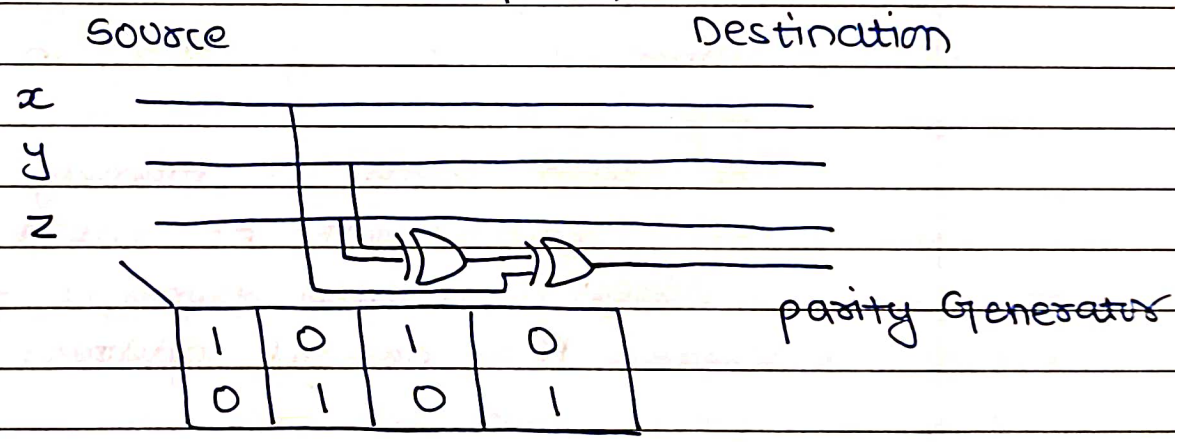
At the sending end, A message is applied to parity generator where parity bit is ~~generated~~ generated.

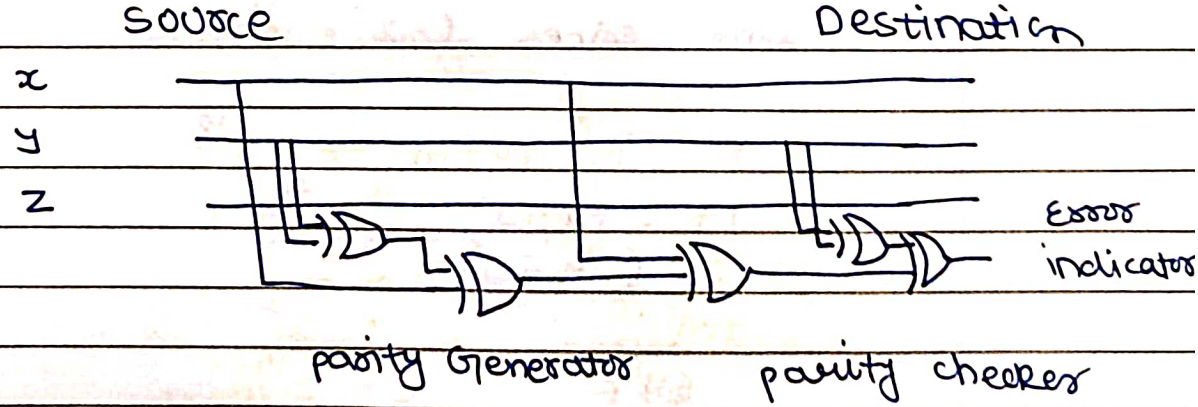
* **Parity Checker**

At the recieving end, All the incoming bits are applied to a parity checker that checks the proper parity adopted.

Ex. Consider a 3 bit message to be transmitted with an odd parity bit, the circuit diagram is shown

(odd parity)





__/__/____

NOTE

Parity checker value = 1 when error occurs

* MEMORY UNIT

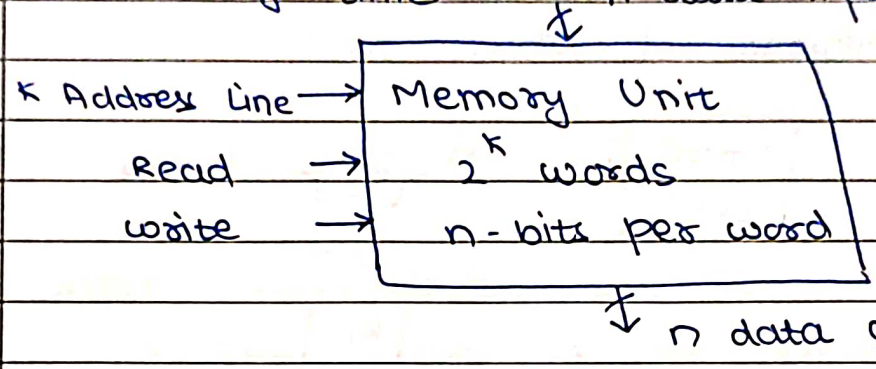
- A memory unit is a collection of storage cells that store one bit together with associated circuit needed to invoke the transfer info. in and out of storage
- The memory stores binary info. in group of bits called word
- A Group of 8 bit is called byte
- The internal structure of memory unit is specified by the no of words it contains & no of bits in each word.
- The each word in memory is assigned an identification no called an memory address started from 0 to 2^{K-1} where K = no of address line
- 1024 words memory require 10 address line slices ($2^{10} = 1024$)

K is equal to 2^{10}
M (Mega) = 2^{20}
G (Giga) = 2^{30}
2M = 2^{21}
64K = 2^{16}
4G = 2^{32}

__/__/____

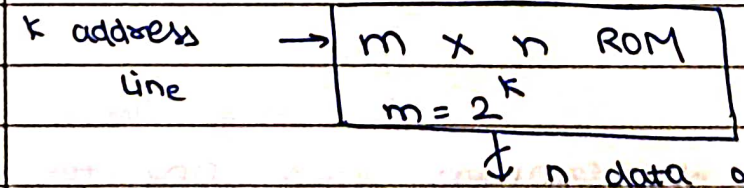
→ 2 Major types of Memory

- RAM - In Ram, memory cells can be accessed for the information transfer from any desired random location. The process of locating a word in the memory is the same and required equal amount of time



1 word = 16 bits
2^k word = $2^4 * 2^k$ bits } Memory needed

- ROM - A ROM is a memory unit that perform the read operations only this implies that the binary info. stored in the ~~rom~~ is made ~~perf~~ permanent during hardware produced. during manufacturing



- ROM does not need "Read", "write" control line

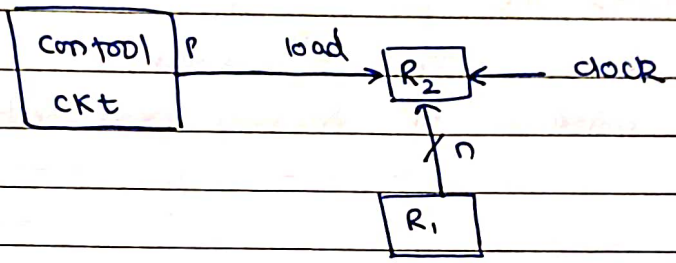
__/__/____

* Control Function

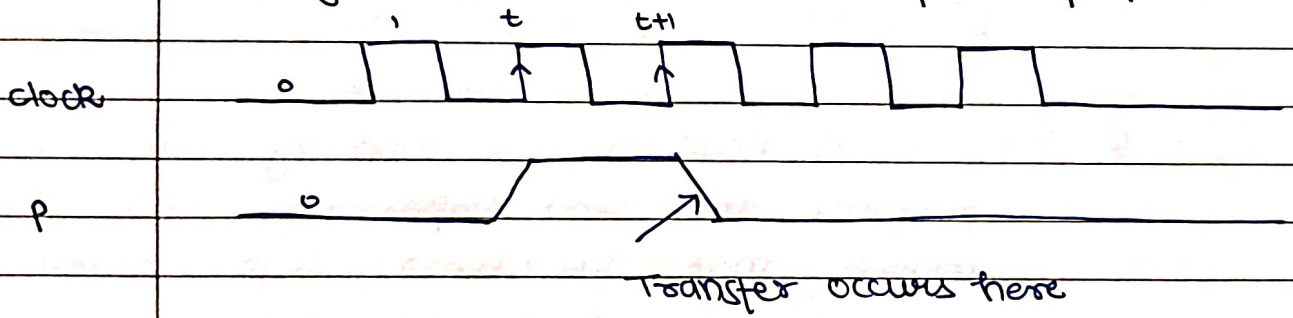
- A control function is a boolean variable i.e. equal to 0 or 1.
- The control funcn is included in the stmt as follow

$$P : R_2 \leftarrow R_1$$

- Transfer operation is executed by the H/w only if $P=1$
- Block diagram of control funcn :



- Timing Diagram - when an opⁿ is performed

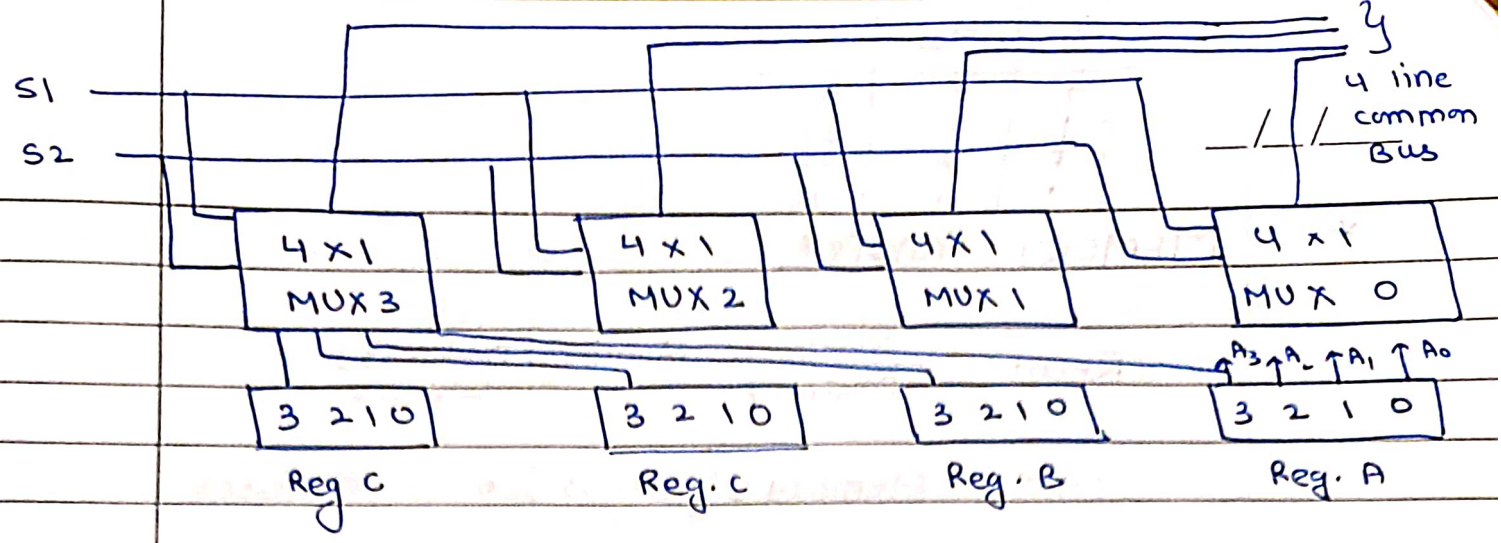


~~The~~ ~~o/p~~

Resistor R_2 has low i/p that activated by the control variable P.

* BUS

Bus consist of common lines. One for each bit of the resistor to which binary info. is transferred one at a time.



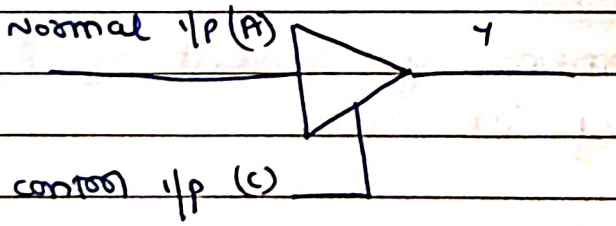
S_2	S_1	Data on bus
0	0	A_3 A_2 A_1 A_0
0	1	B_3 B_2 B_1 B_0
1	0	C_3 C_2 C_1 C_0
1	1	D_3 D_2 D_1 D_0

* **Three State Bus Buffer :-**

- A Bus system can be constructed with three state gate instead of MUX.

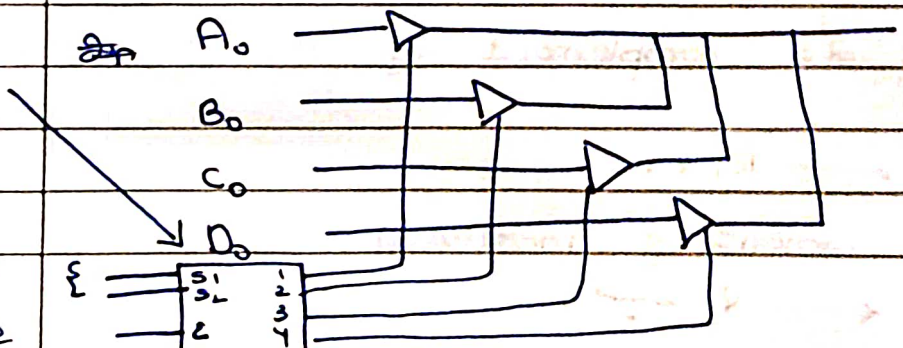
- A Three state gate is a digital circuit that executes 3 states.

- Two states are 0 or 1.
- 3rd state is high impedence.
- A high impedence state we have like open circuit.



output Y = A if C = 1
high impedence if C = 0

2x4 Decoder



Bus line for Bit 0

S_1	S_2	
0	0	A_0
0	1	B_0
1	0	C_0
1	1	D_0

__/__/____

* MEMORY TRASFER :-

Read : $\leftarrow$ Data Register — Memory[Address] Register

write : Memory[Address] Register $\leftarrow$ Register

* TYPES OF MicroOperations (MO)

- Register Transfer Micro operations.
- Arithmetic MO
- Logic MO
- shift MO

The Basic Arithmetic MO are :-

- Addition
- subtraction
- Increment
- Decrement

symbolic designation	Desc
① $R_3 \leftarrow R_1 + R_2$	The content of R_1 plus R_2 transfer to R_3

② $R_3 \leftarrow R_1 - R_2$

The content of R_1 minus R_2 transfer to R_3

③ $R_2 \leftarrow \overline{R_2}$

The 1's complement of content of R_2 is replaced by its own.

④ $R_2 \leftarrow \overline{R_2} + 1$

The 2's complement of " " " ....

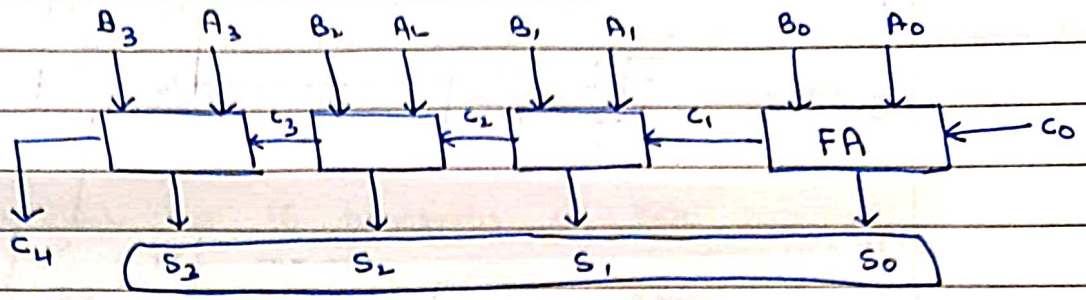
⑤ $R_1 \leftarrow R_1 + 1$

The content of increased by 1

⑥ $R_2 \leftarrow R_2 - 1$

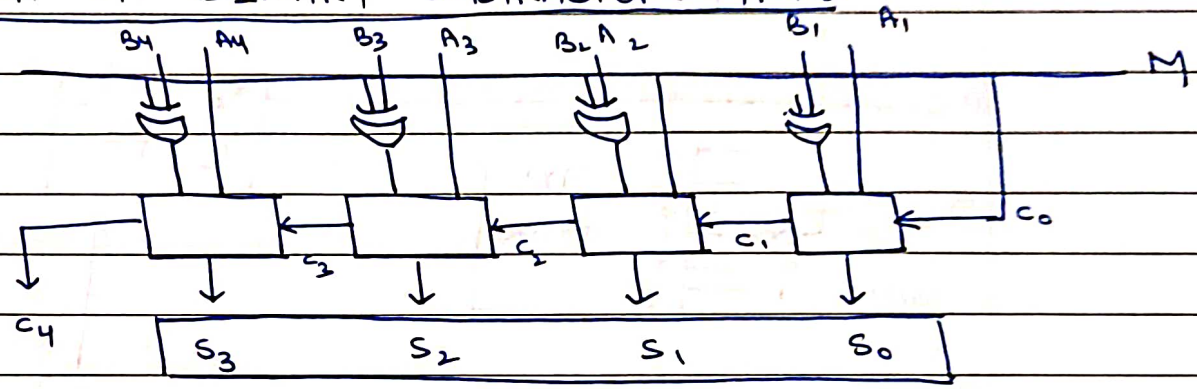
__/__/____

* FOUR BIT BINARY ADDER



FA : Full Adder

* FOUR BIT BINARY SUBTRACTOR - ADDER



When $M = 0$ CKT = Adder

$M = 1$ CKT = Subtractor

$$S_0 = A_0 + (\overline{B_0} + 1) = A_0 - B_0$$

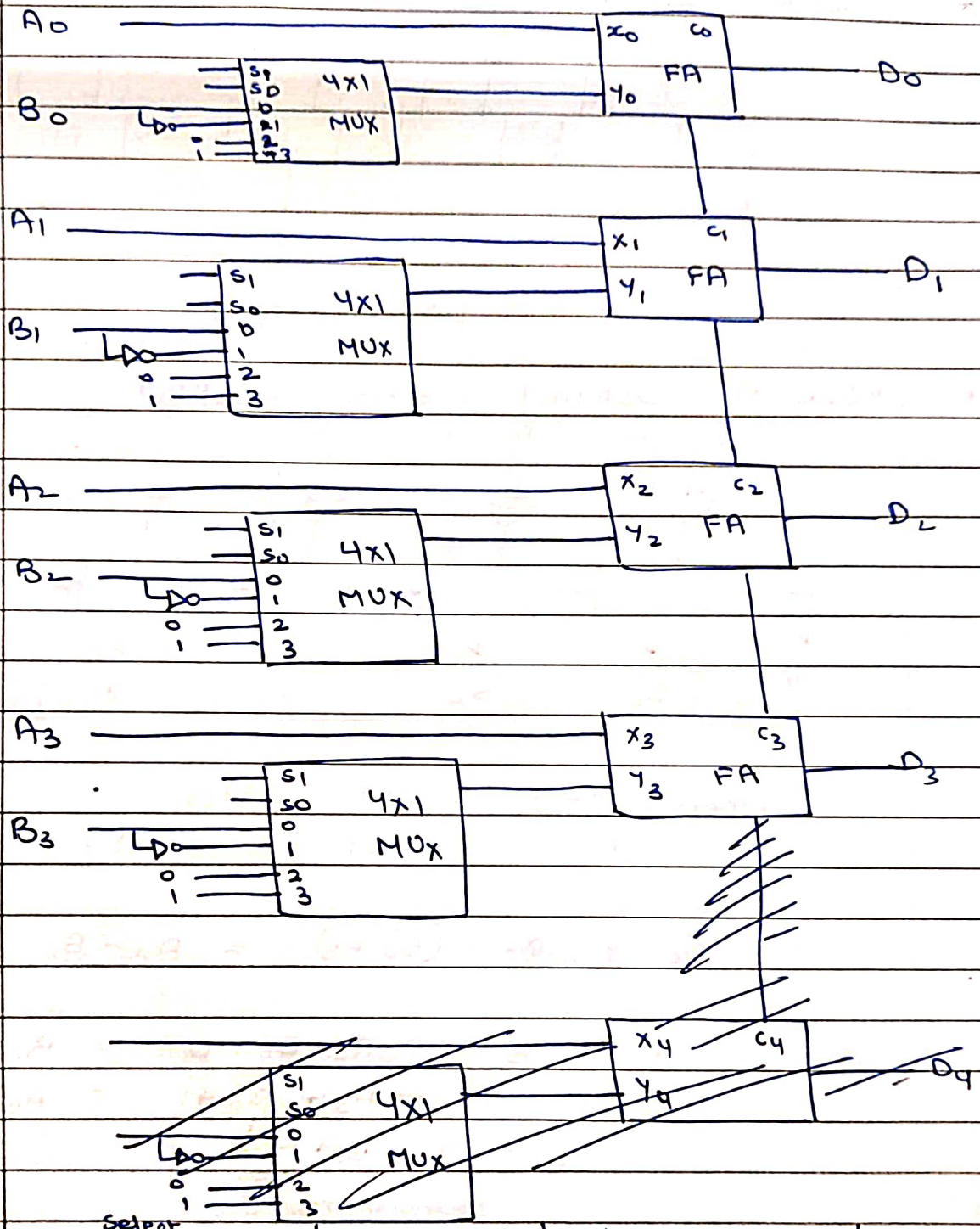
$$B \oplus 0 = B \rightarrow A_0 + B_0 + 0 = A_0 + B_0$$

$$B \oplus 1 = \overline{B} \rightarrow A_0 + B_0 + 1 = A_0 - B_0$$

$$\boxed{S = A_0 + B_0 + M}$$

*

__/__/____



Select			Input (Y)	O/P D = A+Y+Cin	Mic. opⁿ
S_1	S_0	cin			
0	0	0	B	D = A+B	Addⁿ
0	0	1	B	D = A+B+1	Add with carry
0	1	0	$\overline{B}$	D = A+$\overline{B}$	Subⁿ with Borrow
0	1	1	$\overline{B}$	D = A+($\overline{B}$+1)=A-B	Subⁿ
1	0	0	0	D = A+0+0	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

if All Bit == 1 then value is [-1]

__/__/____

* LOGIC MICRO-OPERATION

Logic MO specify Binary operat^n's ~~for four for~~ string of the bits stored in Register.

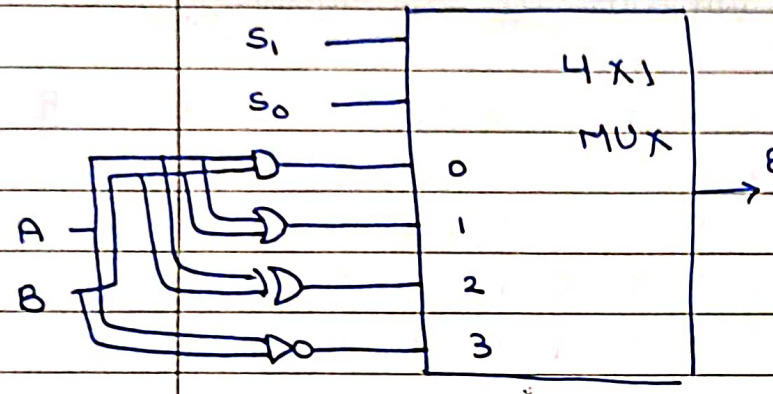
Truth Table for 16 functions of two variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean Function	Micro opt^n	Name
$F_0 = 0$	$F \leftarrow 0$	clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \overline{B}$	—
$F_3 = xy' + xy = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow \overline{A} \wedge B$	
$F_5 = x'y + xy = y$	$F \leftarrow B$	Transfer B
$F_6 = x'y + xy' = x$ XOR y	$F \leftarrow A \oplus B$	XOR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = \overline{x} \cdot \overline{y} = \overline{x+y}$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = \overline{x \oplus y}$	$F \leftarrow \overline{A \oplus B}$	EX-NOR
$F_{10} = \overline{y}$	$F \leftarrow \overline{B}$	
$F_{11} = x + y'$	$F \leftarrow A \vee \overline{B}$	
$F_{12} = x'$	$F \leftarrow \overline{A}$	
$F_{13} = x' + y$	$F \leftarrow \overline{A} \vee B$	
$F_{14} = \overline{xy}$	$F \leftarrow \overline{A \wedge B}$	
$F_{15} = 1$		

* Hardware Implementation of Logic MO__/__/___

- There are 16 MO. Most computers use only four AND, OR, XOR, complement. from which all can be derived.



S_1	S_0	Ei O/P	opn
0	0	$A \wedge B$	AND
0	1	$A \vee B$	OR
1	0	$A \oplus B$	XOR
1	1	$\overline{A}$	complement

* Shift Micro operations :-

1.) Logical shift - A logical shift is one that transfer 0 ~~to the se~~ through the serial I/p

x = | 1 | 0 | 1 | 1 |

Left shift = x<<1 = $x * 2^1$

Right shift = x>>1

x<<1 | 0 | 1 | 1 | 0 |

x>>1 | 0 | 1 | 0 | 1 |

The most significant is lost in (<<)

2. Circular shift

It circulates the bits of the register around the two ends w/o loss of ~~the~~ info.

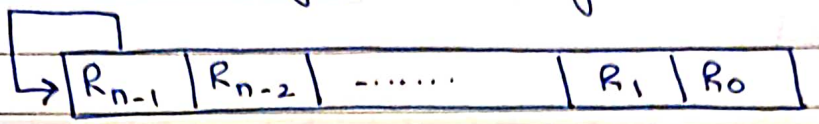
(<<) | 1 | 0 | 1 | 1 | → | 0 | 1 | 1 | 1 |

(>>) | 1 | 0 | 1 | 1 | → | 1 | 1 | 0 | 1 |

__/__/____

3. Arithmetic shift :-

- It is a micro operation that shift a signed binary no to the left or right.

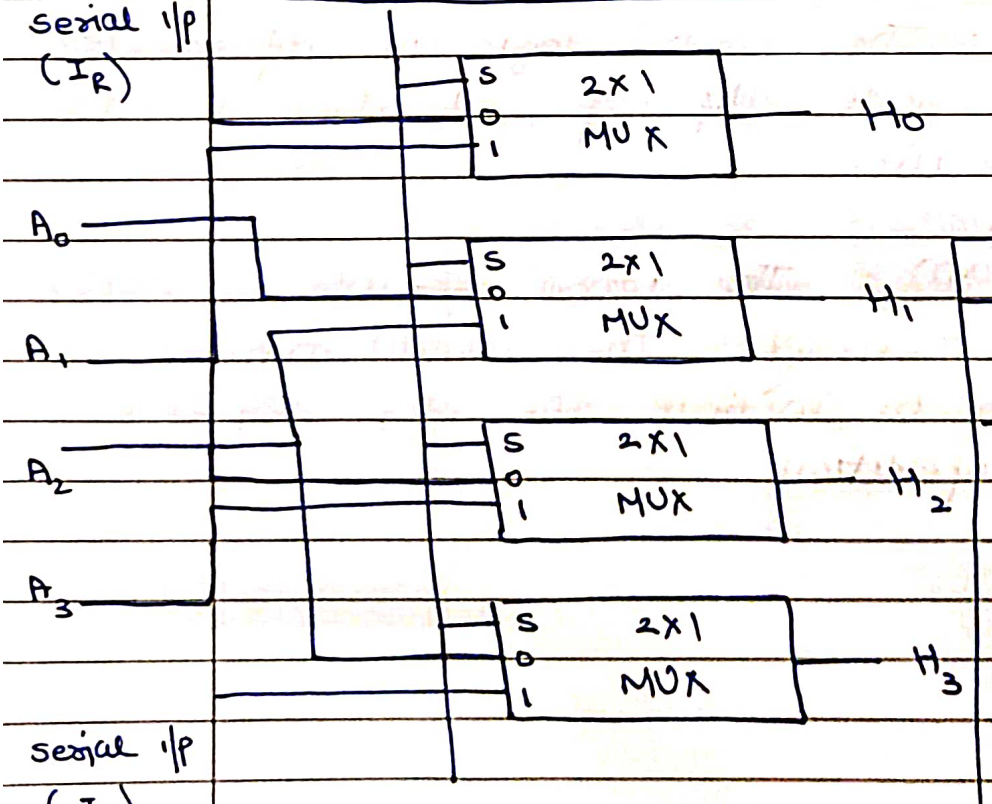


↑ sign bit (does not move)

* R_{n-2} to R_0 hold the number

Symbolic Designation	Description
$R \leftarrow$ shlR	shift left Register R
$R \leftarrow$ shrR	shift right Register R
$R \leftarrow$ cilR	circular shift left Register R
$R \leftarrow$ cirR	" " right " "
$R \leftarrow$ ashlR	arithmetic shift left Register R
$R \leftarrow$ ashrR	" " right " "

* 4 Bit Combinational CKT shifter :-



select-0 for shift right

select-1 for shift left

Function Table

select (S)	output H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

__/__/___

(4) Timing and Control

The timing of all registers are on the basis computer is controlled by Master clock generator.

The clock pulse are applied to all flip flops and registers in the system.

The control signals are generated in controlled unit & provides control i/p for multiplexer in common bus.

control i/p in processor registers and micro operations for accumulator.

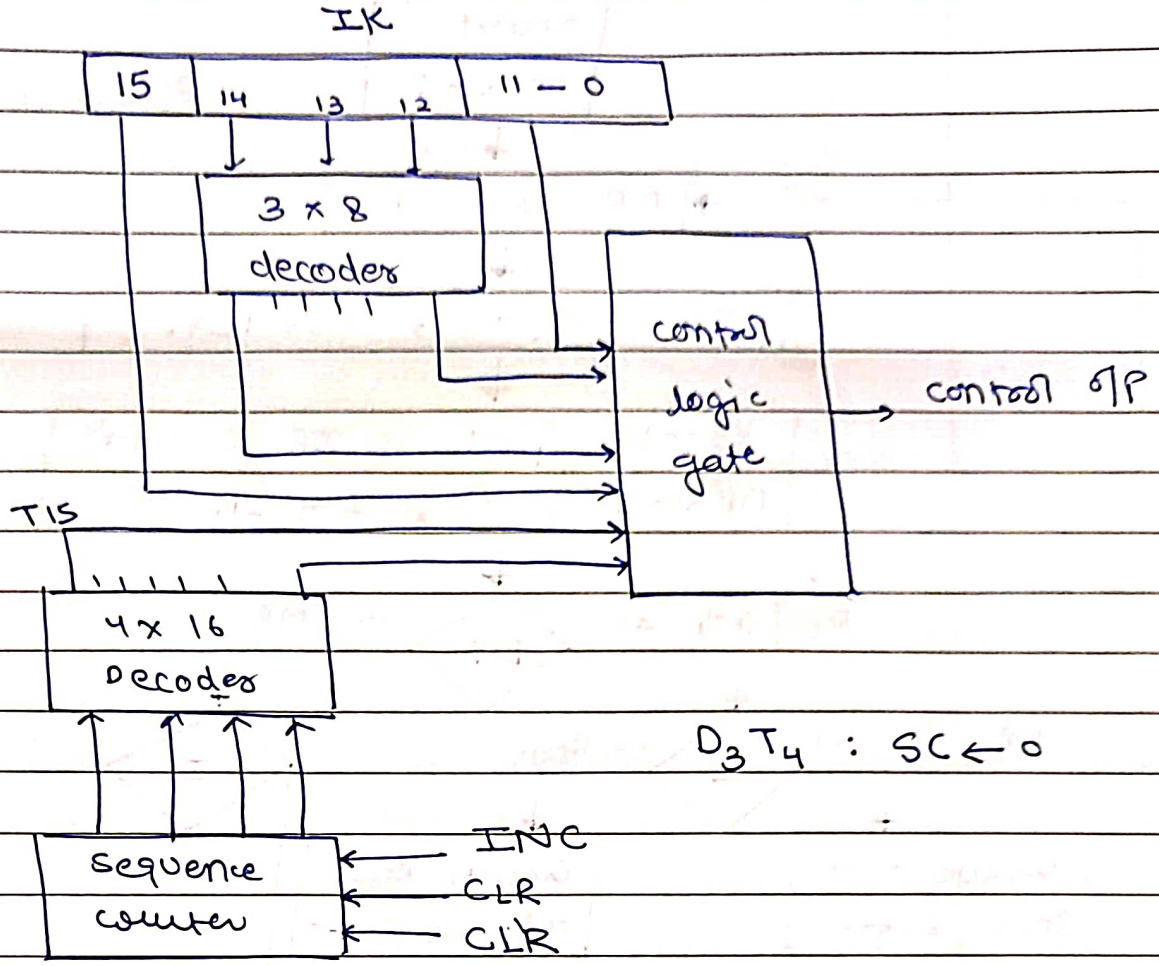
2 Major control org

→ Hard wired control

→ Micro programmed control

1.) Hard wired : The control logic is implemented with logic gate, flip flop, decoders & other digital circuits

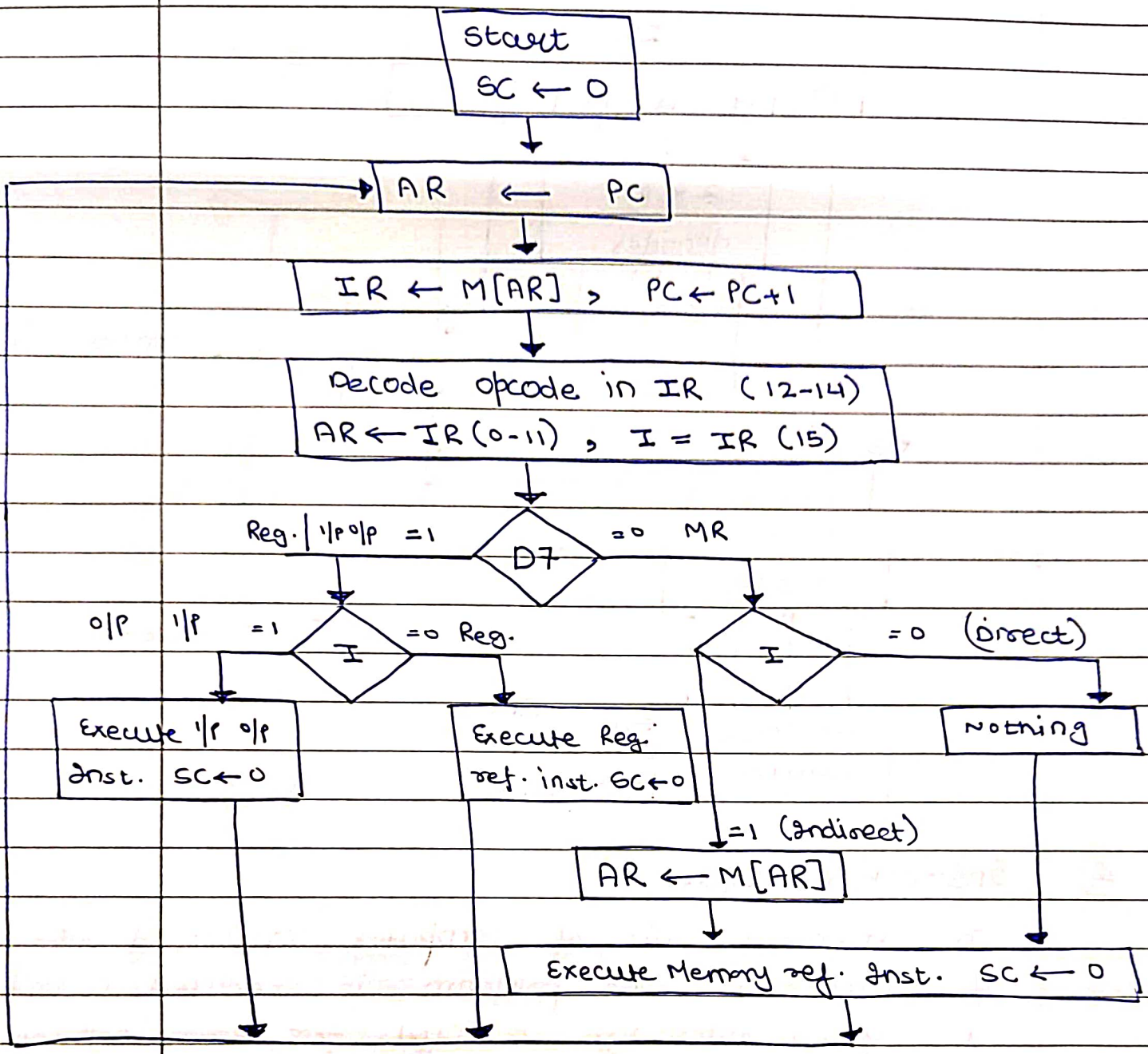
2.) Micro Program : The control ~~i/p~~ info is stored in control memory. The control memory is a program to initiate the req. sequence of micro operations.



Instruction Cycle

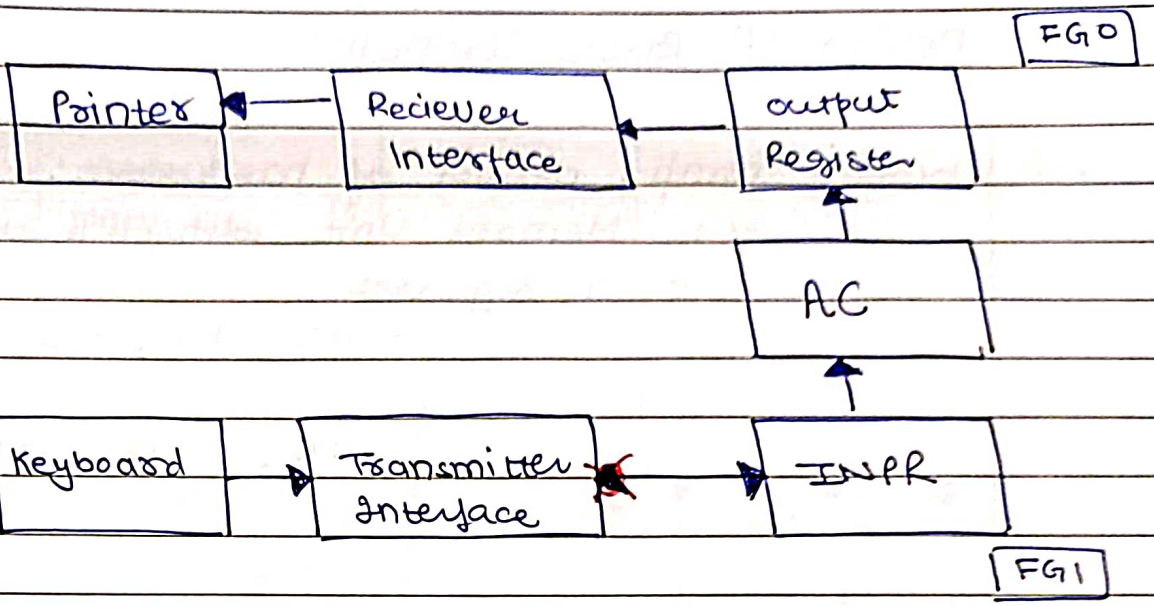
The memory unit of computer consist of seq. of instructions. The program is executed in comp. by going through a cycle for each instruction

1. Fetch an instruction from memory
2. Decode the instructions
3. Fetch operand
4. Execution of instruction



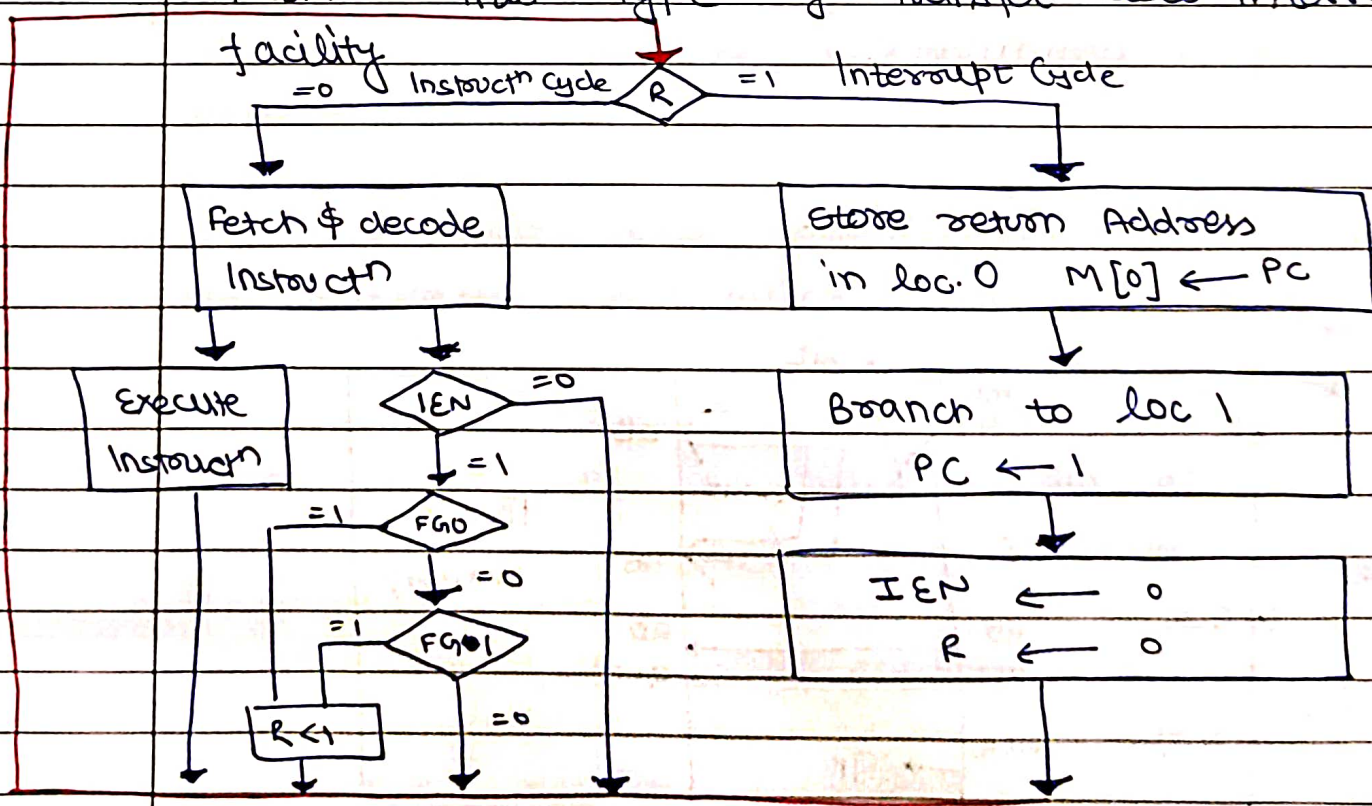
I/O Configuration

The I/P device is the keyboard and O/P device is printer. They send & recieve the data consecutively. The data is alphanumeric & 8 bit in size.



⊛ INTERRUPT

The external device inform the computer when it is ready for the transfer. In the mean time, the computer can be busy with other task. This type of transfer uses interrupt facility



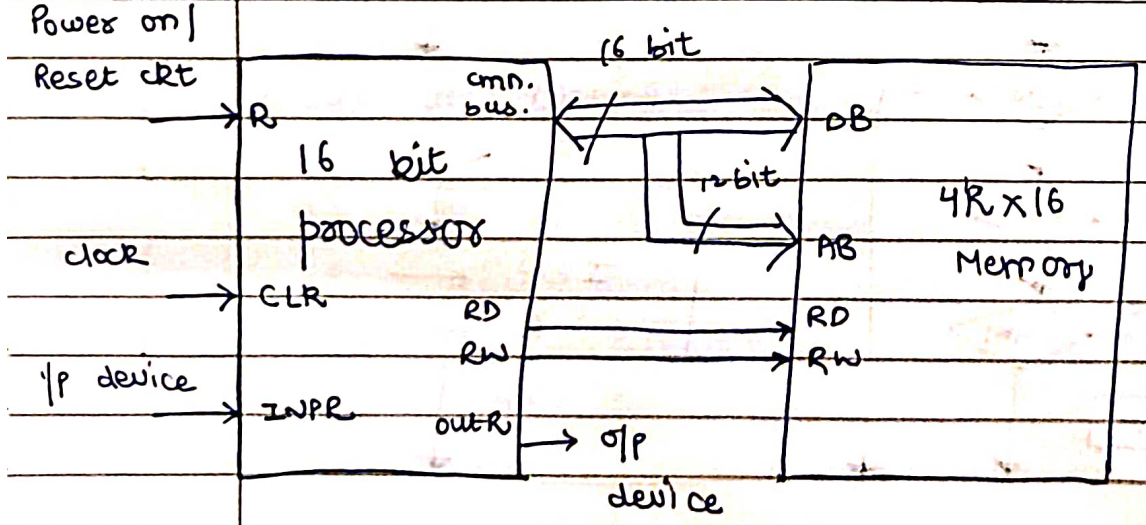
__/__/____

Design of Basic Computer

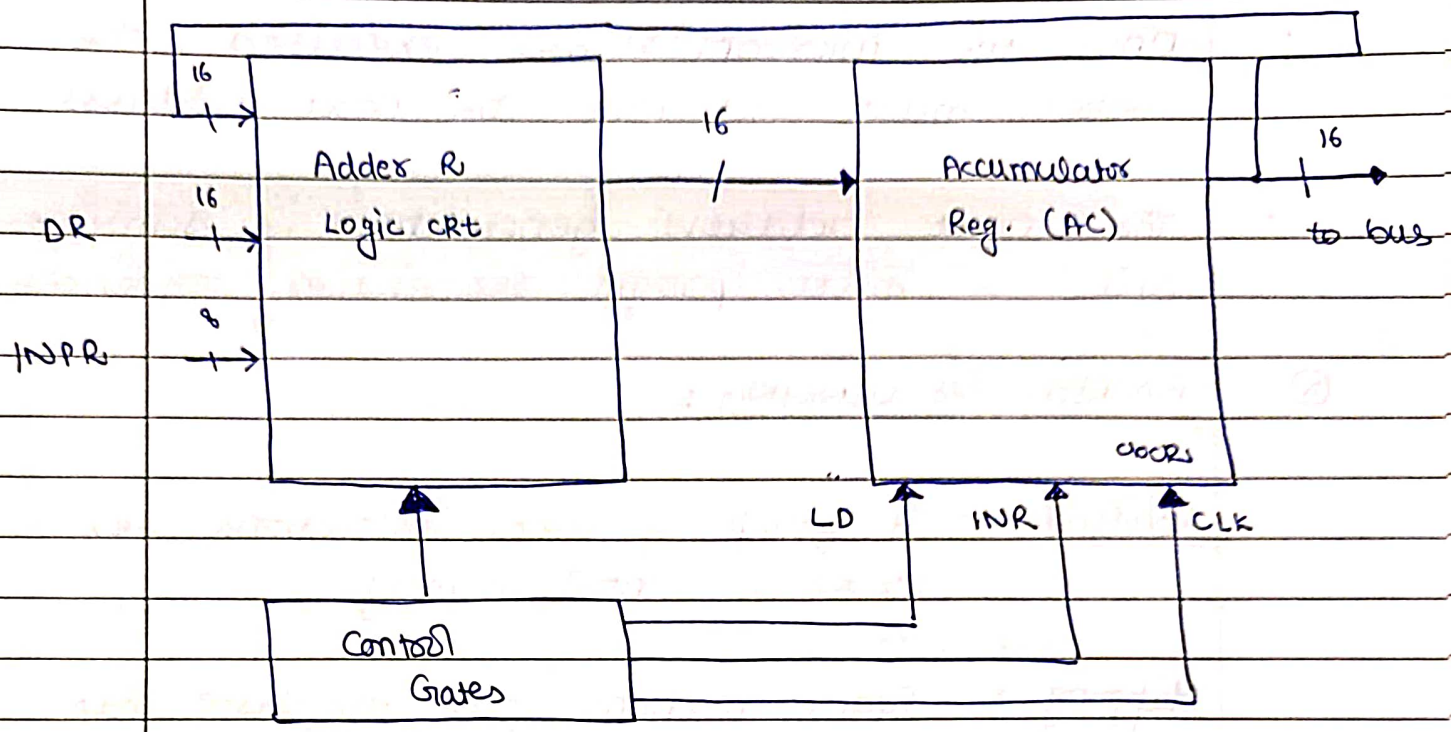
basic comp. consist of hardware:

1. Memory Unit with 4096 words of 16 bit ea
2. 9 Registers → AR, PC, AC, IR, TR, OUTR, INPR, SC, DR
3. Seven flip flops: I, S, E, R, IEN, FGI, FGO,
4. Two decoders: 3 x 8 operation decoder
   4 x 16 timing decoder
5. A 16 bit common bus:
6. control logic Gate
7. Adder & logic ckt connected to the i/p of accumulator.

Block Diagram:



__/__/____



⊛ MICRO-PROGRAM-CONTROL-UNIT

- The function of control unit in a digital system is to initiate sequences of the micro-opn's. It may be ~~hardware or~~ hardwired or micro-program

- A control unit whose control signals are generated by hardware is called hardwired control unit

- If the CU whose binary control variable are stored in memory is called micro program CU.

- A memory that is part of the CU is referred to as a control memory

- The control memory address Reg. specifies the address of micro instruct^n & the control data register hold the micro instruct^{n}s read from memory.

__/__/____

Once the microopt^n's are executed, the control must determine the next address.

The next address generator is sometimes call a micro progm ~~sequential~~ sequencer.

⊗ ADDRESS SEQUENCING :

Routine : A group of micro instructions i.e, stored in control memory

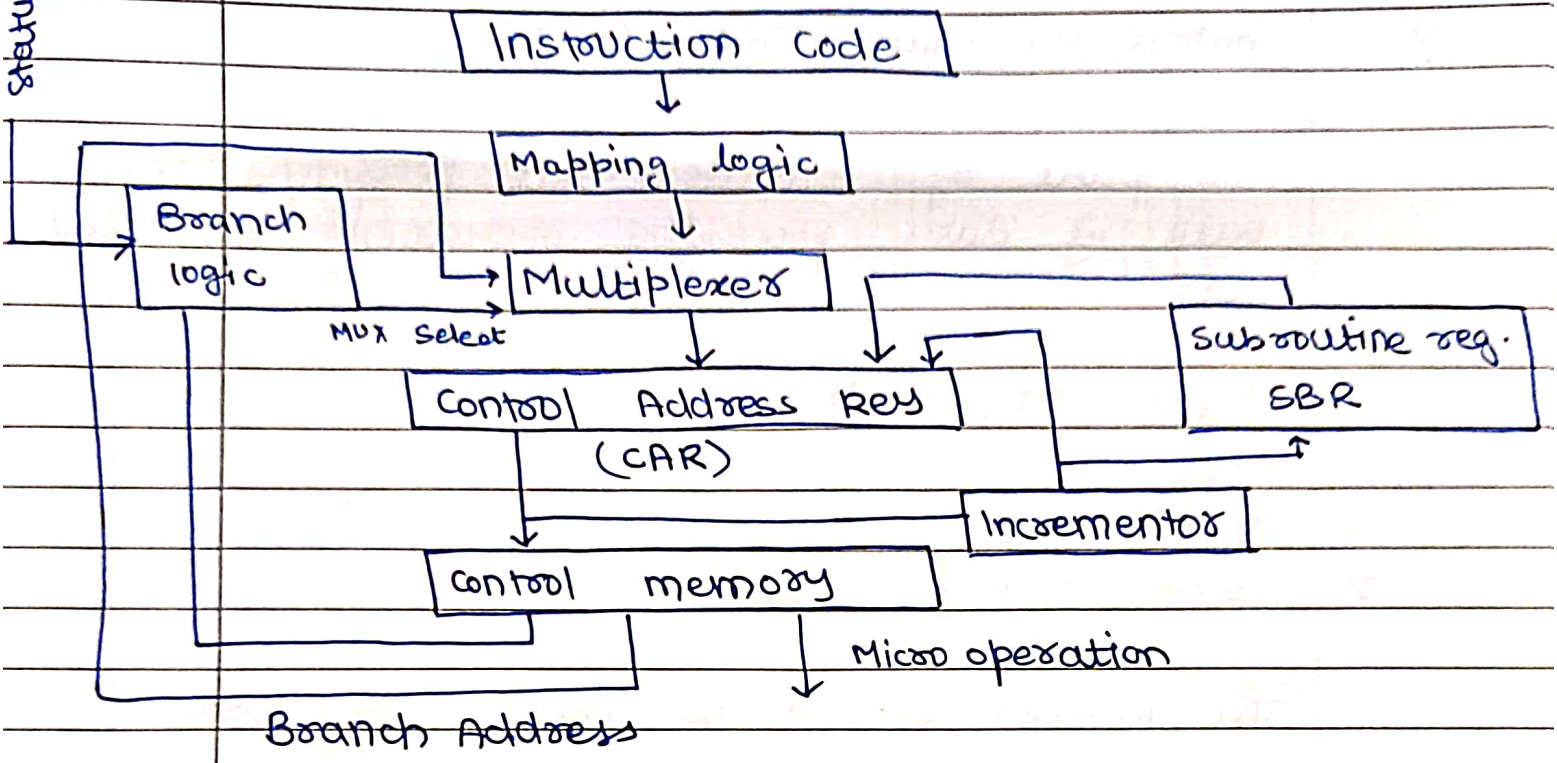
Mapping : Transformations from the instr code bits to an address in control memory where routine is located is referred to as mapping

⊛ The address sequencing capabilities required in a control memory are:

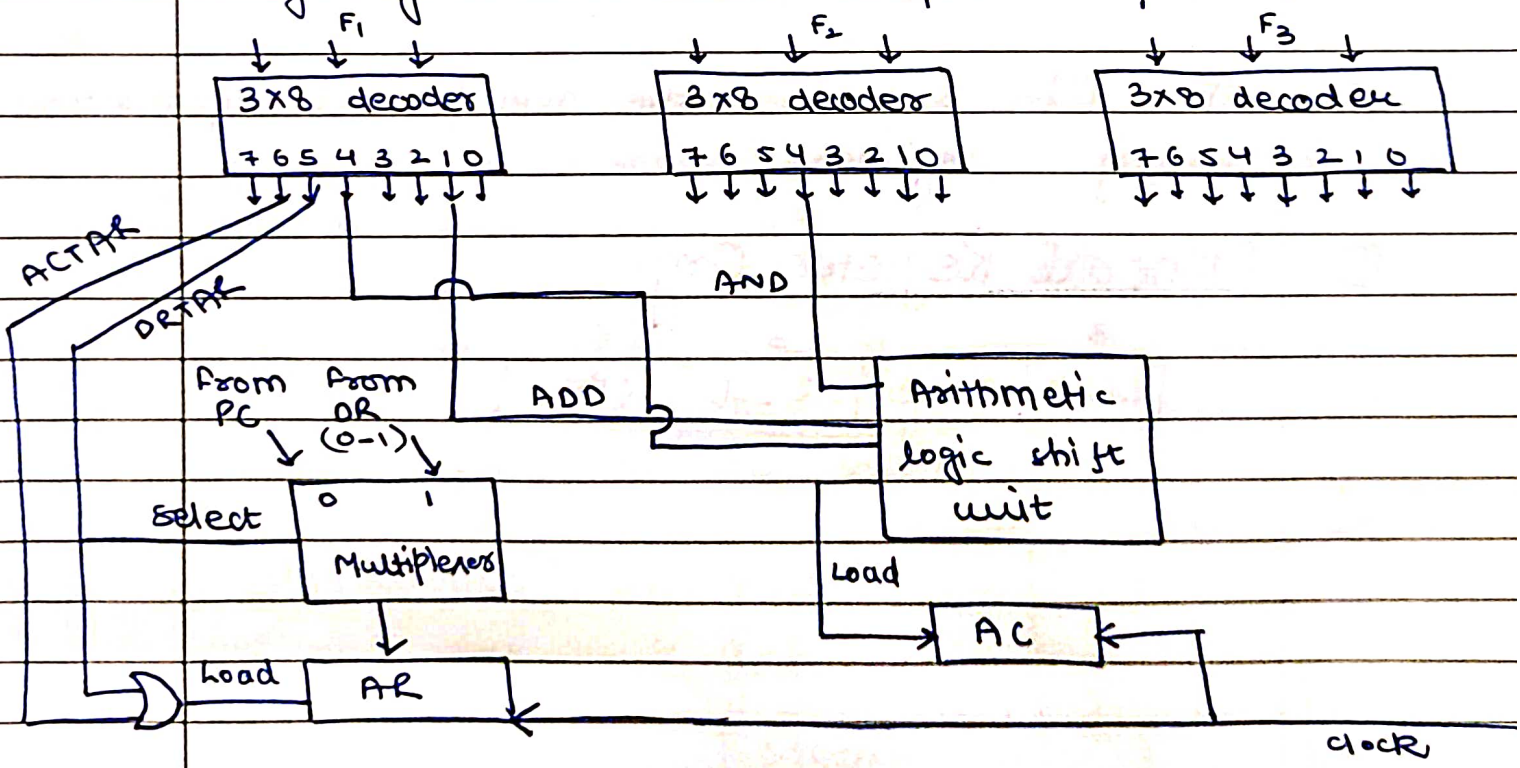
1. Incrementing ~~Increasing~~ of the control address register.
2. Unconditional / conditional branch depending on status bit conditions.
3. A memory process from the bits of the Instruct^n to an address from control memory.
4. Facility for subroutine and written or called
5. A block diag of a control memory & associated hardware needed for selecting next micro instruct^n address.

status bit

__/__/____



The bits of the micro instruction are usually divided into fields with each field defining a distinct separate function.



__/__/____

⊛ Central Processing Unit (CPU)

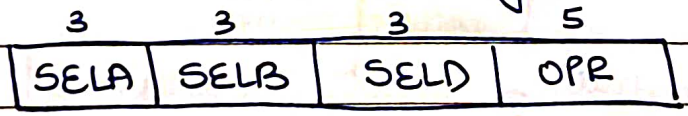
The part of computer that performs the bulk of data processing operations is called CPU

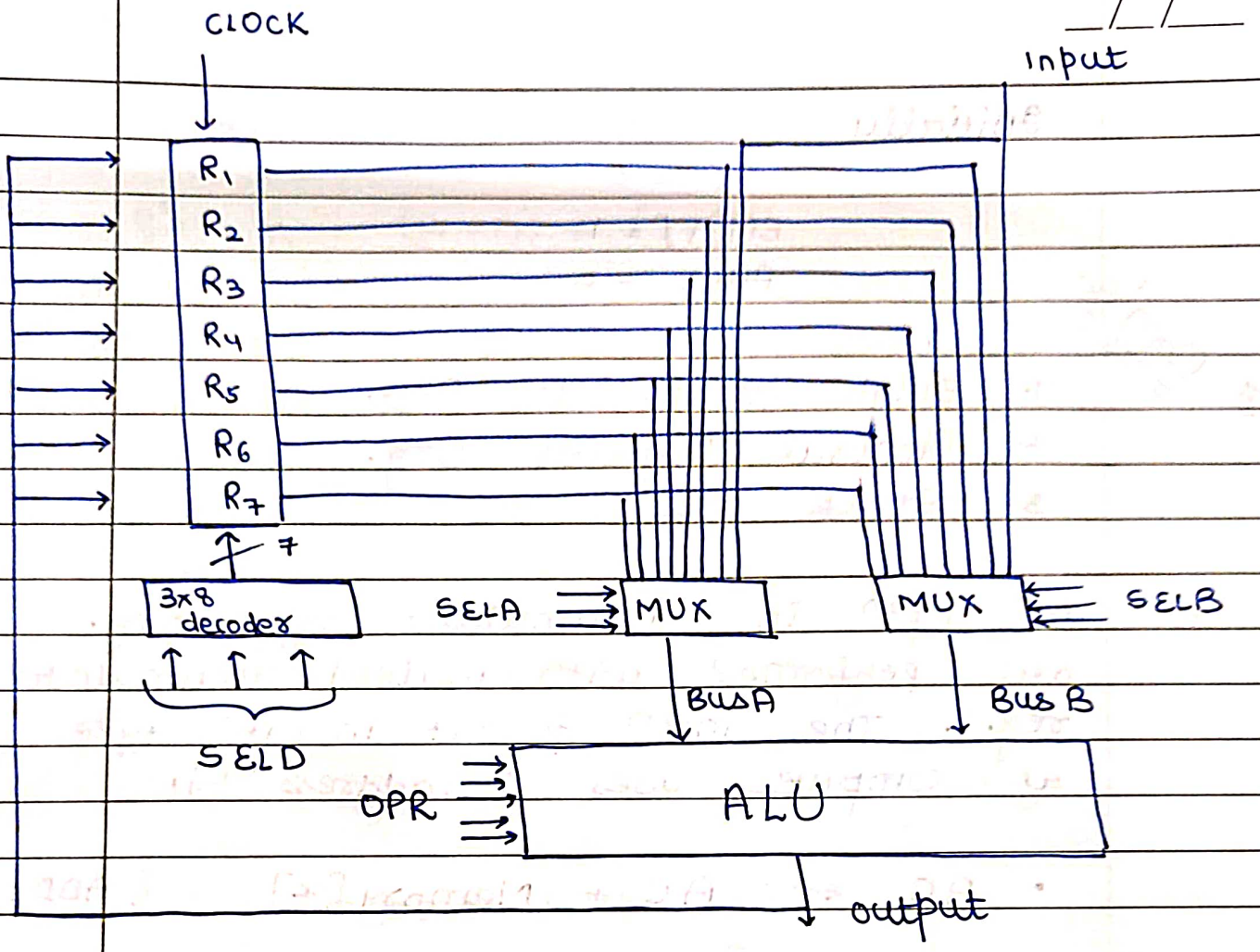
Major Components

1. ALU
2. Register set
3. Control Unit

- The Register set stores intermediate data used during the execution of Instruct^{n}.
- The ALU performs the required micro operations for executing the Instruct^{n}.
- The CU supervise the transfer of informations among registers

① General Register Org.



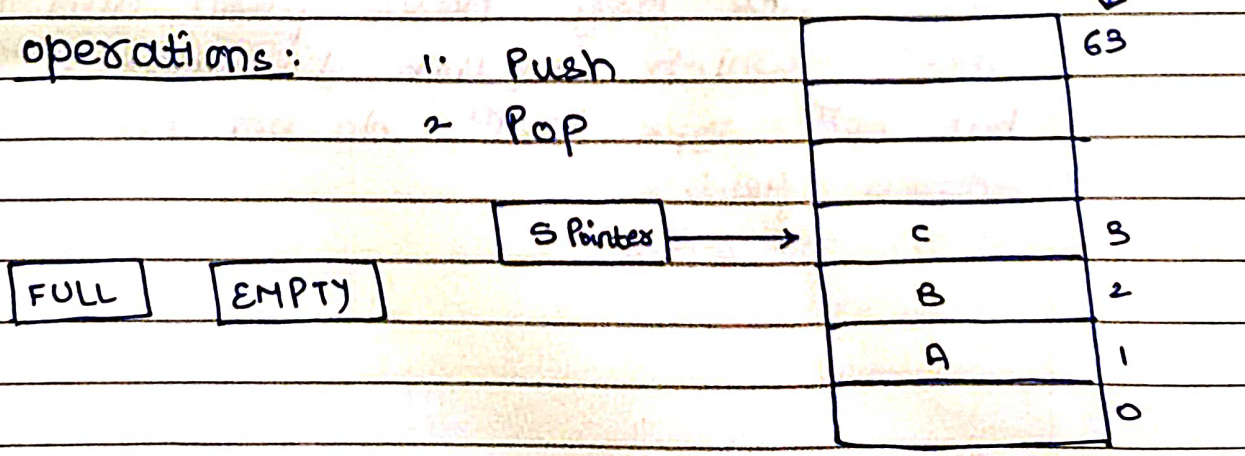


* Stack Org.

Stack is a storage device that stores infor. in such a manner that the item stored last is the first out (LIFO)

The Register that hold the address for stack is called stack pointer. It always point to pmost point of the stack.

operations: 1. Push
2. Pop



__/__/____

Initially

SP = 0

EMPTY = 1

FULL = 0

3 ~~0&9~~ 1's

1. Single accumulator Org.
2. General Register Org.
3. Stack Org.

1.) All opⁿ in accumulator type org. are performed with inclined accumulator reg. The instⁿ format in this type of computer uses 1 address bit

- AC ← AC + Memory[x] {ADD: x}

2.) The Instⁿ format in general register type org. needs three address register fields.

- $R_1 \leftarrow R_L + R_3$

  {ADD, R_1, R_2, R_3}

- ADD, R_1, R_2 { $R_1 \leftarrow R_2 + R_1$ }

3. The Stack org. have push and pop instⁿ which require 1 address field but opⁿ type instⁿ do not need an address field.

__/__/____

Three Address Instⁿ

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

General Register org:

```
ADD  R1, A, B
ADD  R2, C, D
MUL  x, R1, R2
```

Two Address Instruction

eg

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

```
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
MOV  x, R1    :  X ← R1
```

One Address Instruction :

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

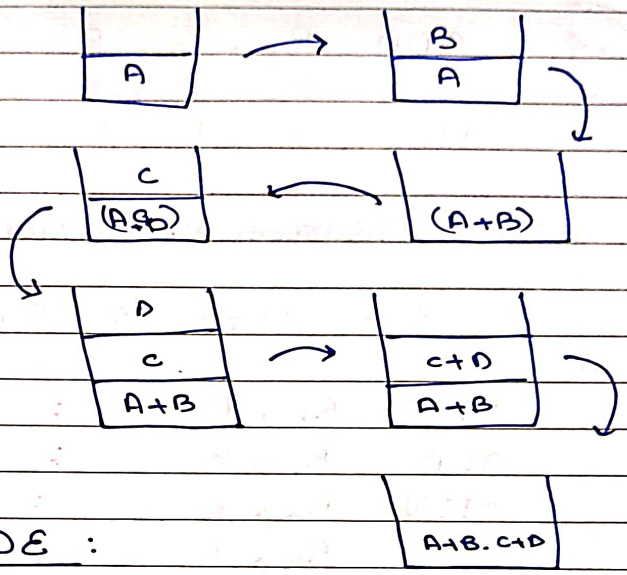
```
LOAD   A   :  AC ← M[A]
ADD    B   :  AC ← AC + M[B]
STORE  T   :  M[T] ← AC
LOAD   C   :  AC ← M[C]
ADD    D   :  AC ← AC + M[D]
MUL    T   :  AC ← AC * M[T]
```

__/__/____

Zero Address Instruction

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

```
PUSH A
PUSH B
ADD
PUSH C
PUSH D
ADD
MUL
```



ADDRESS MODE :

The addressing mode specifies the rule for interpreting or modifying the address field of the Instn before the operand is actually referred.

The Instn format with mode field is

OPCODE	MODE	ADDRESS

Diff. Addressing Mode

1. Implied mode : The operands are specified implicitly in the definition of the Instn (zero addressing field)

E.g. Complement of AC

__/__/____

2. **Immediate Mode**

OPCODE	MODE	OPERAND

In this, the operand is specified in the Instn itself. In other word, an Immediate mode Instn has an operand field rather than an address field.

3. **Register Mode :**

In this, the operands are in registers that ~~are in CPU~~ reside within a CPU.

4. **Register Indirect Mode :**

In this, Reg Instn specify a register in CPU whose contents the address of operands in memory.

5. **Direct Address Mode**

In this, the effective address is equal to the address part of Instn

6. **Indirect address mode**

In this, the address field of Instn gives the address where the effective address is stored in Memory.

7. **Relative Address mode**

In this, the content of program counter is added to the address part of Instn in order to obtain effective address.

__/__/____

8. **Index Address Mode**

In this, the content of index reg. is added to the address part of instⁿ to obtain effective address.

9. **Base Register Addressing Mode**

In this mode, the content of base register add the to the address part of the instruction to obtain effective address.

Address	Memory
200	Load to AC | Mode
201	Address = 500
202	
	⋮
399	4500
400	700
	⋮
570	800
	⋮
600	900
	⋮
702	325
	⋮
800	300

PC [200]
R1 [400]
XR [100]

[AC]

__/__/____

	Address Mode	Effective Add.	Content of AC
1.	Direct Address	500	800
2.	Immediate Operand	201	500
3.	Indirect Address	800	300
4.	Relative Address	702	325
5.	Indirect Address	600	900
6.	Register	-	400
7.	Reg. Indirect	400	700