* MAR: Associated with address lines of the system i.e. read & write operation.

* MBR: Associated with data lines of the system i.e. value to be stored in memory (or) last value scanned from memory.

* PC: carries address of next instruction

* IR: holds the last instruction fetched.

* Instruction Cycle: Each & every phase of IC can be broken into micro sequences.

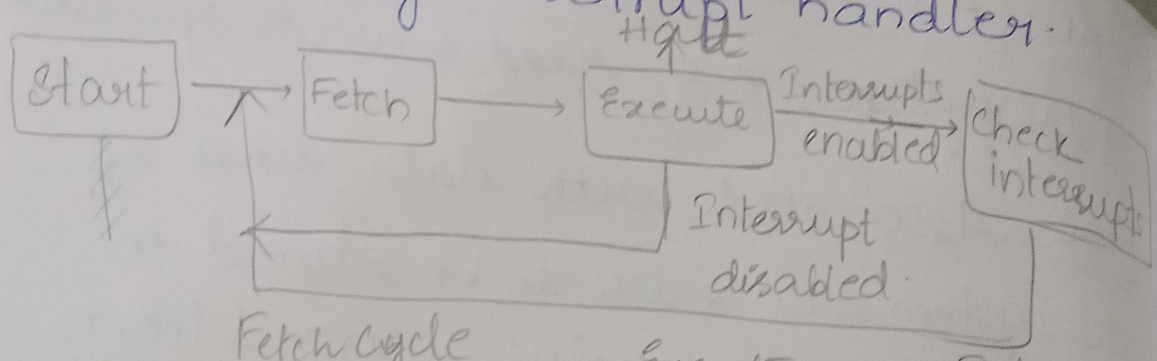
Load: Transfer data from RAM to accumulator.
Store: Transfer data from accumulator to RAM.
Data defn: Variable defn.

* Types of Interrupts:

1) Program
2) Timer
3) I/O
4) Hardware failure.

* A control signal is generated when an interrupt is raised which is

handled by interrupt handler.



→ Sequential Interrupt
→ Nested Interrupt

*Interconnection Structure: Collection of paths with various modules is called inter-connection

→ Memory to processor
→ processor to memory
→ I/o to processor.
→ Processor to I/o
→ I/o to/from Memory → can exchange data directly using direct memory access

*Transfer and control of data.

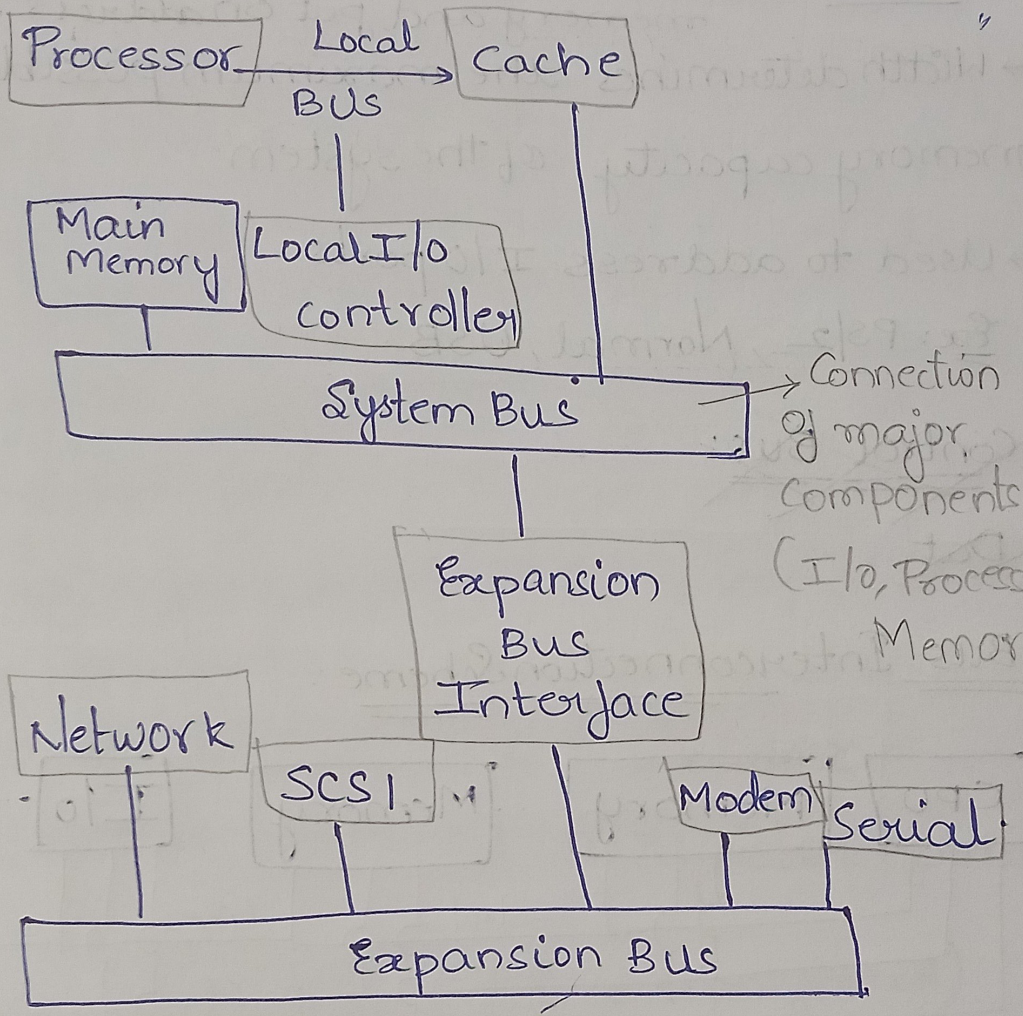
*I/O devices, memory and processor are connected together using bus structures.

*I/O modules are controlled using I/o processors. Exchange of data across diff modules.

*Types of Interconnection Structures:

1) Bus and Multiple Bus Structure
2) Point to Point interconnection.

*Traditional Bus Architecture:



* Address Bus : Unidirection
* I/O : Bidirection

Bus → Address lines → uni-directional
Bus → Control lines, Data lines → Bi-directional.

*→ Address Bus: Used to designate the source or destination of data on the data bus.

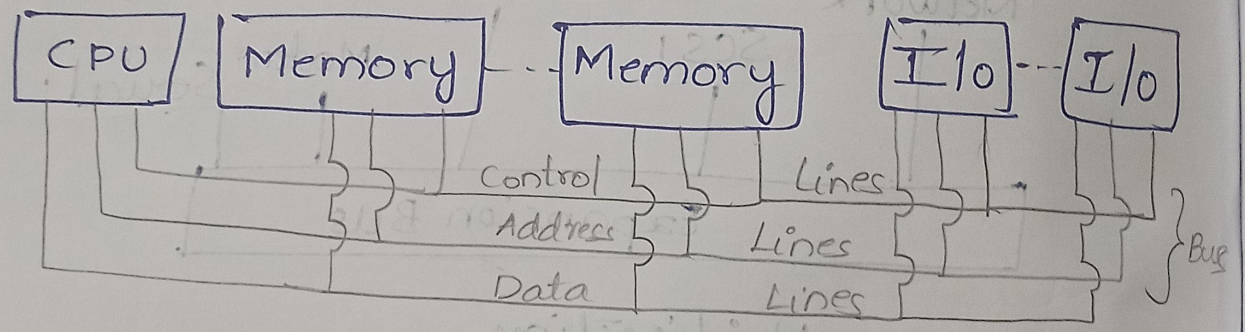
*→ Control Bus: Control the access & the use of data & address lines.

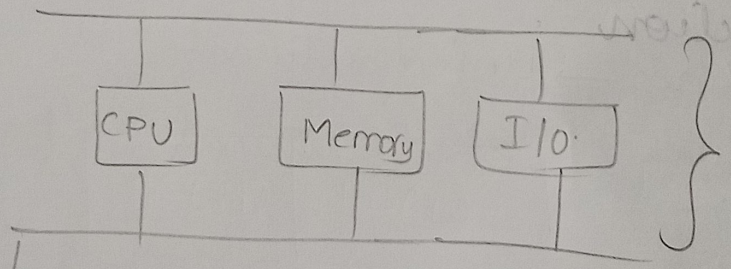
*→ Data Bus: Amount of data to be

* Address Bus: → wishes to read data from memory and put on address lines
→ Width determines the maximum possible memory capacity of the system.
→ Used to address I/O ports.
Ex: PS/2, Normal, USB.

* ~~Control Bus:~~
* ~~Dat~~

* Bus Interconnection Scheme:





↳ Single Bus.

PCI: Peripheral Component Interconnection

core: Speed of processor.

DRAM: Dynamic RAM.

QPI: Quick Path interconnection

Physical → Link → Routing → Protocol

01/11/2022

* Register Transfer Language:

→ Using micro operations; we can construct the code.

→ $R_2 \leftarrow R_1$ → Replacement operator.

↓

In this, the content in R1 is copied to R2 but the content in R1 is not erased. and the value in R1 is same.

→ n-flipflops construct a register holding n-bits.

→ DIGITAL MODULES: (0, 1 → transferred)

→ Decoder
→ Multiplexers
→ Registers.

} These are required to construct RTL code.

Using Microprograms
- Hardwired
- Hardware.

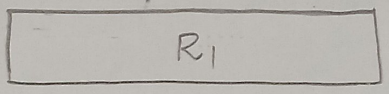
* Micro- Operations:

The operations executed on data stored in

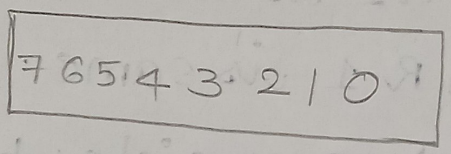
* Register Transfer Language: (RTL)

Symbolic notation used to describe micro operations transfer among registers is called RTL.

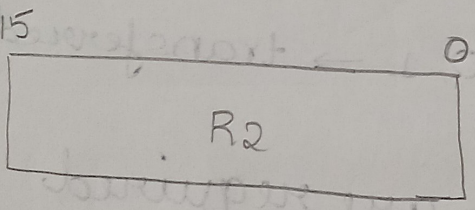
* Block Diagram for register:

a) 

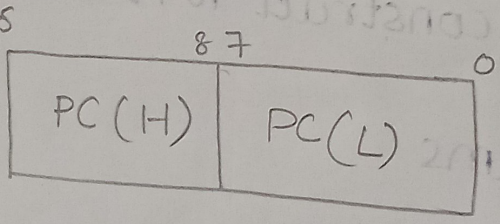
: Register (R)

b) 

: Showing individual bits.

c) 

: Numbering of bits

d) 

: Divide into 2 parts.

* Control Condition:

Ex:

If then statement

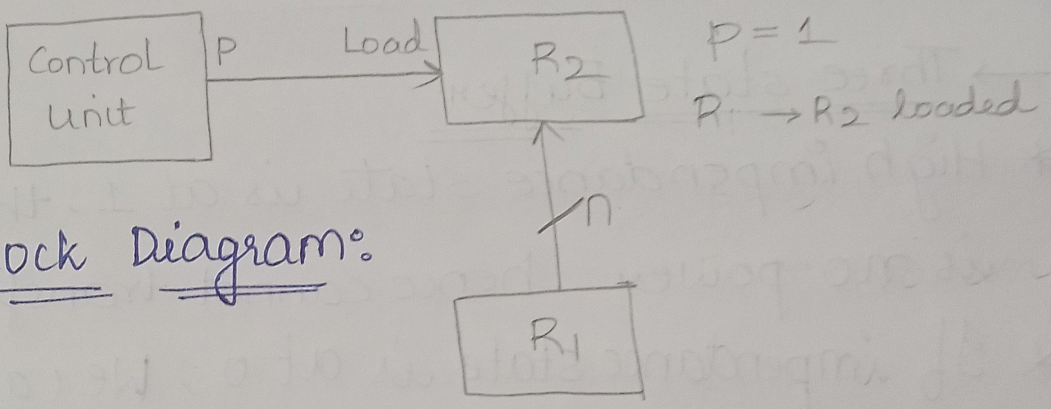
if (P=1) then (R2 ← R1)

Boolean variable is equal to 1/0

$P: R_2 \leftarrow R1$

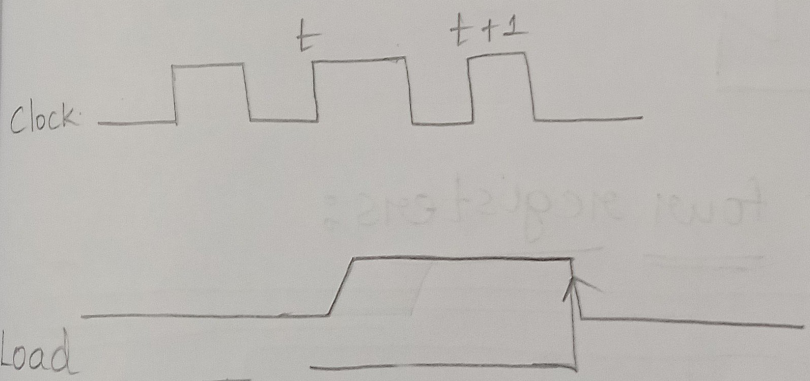
if P=0, comes out of the loop

*→ Implementation of Transfer:



(a) Block Diagram:

(b) Timing Diagram:



05/11/2022

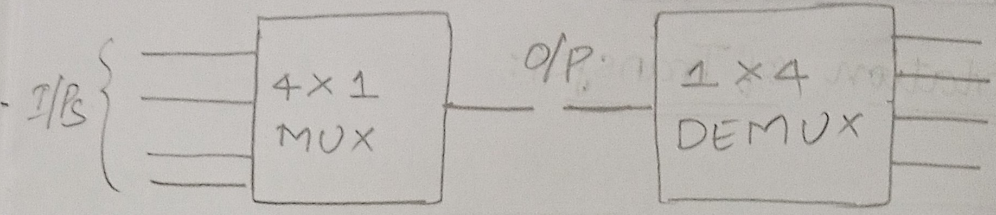
*Bus Memory Transfer:

→ Multiplexer

→ Three-state Buffer

: * Digital circuits :

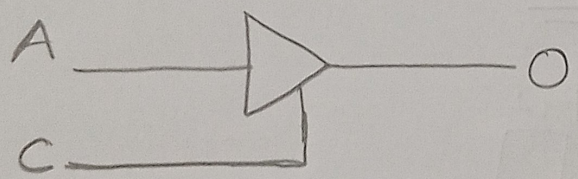
→ Multiplexer:



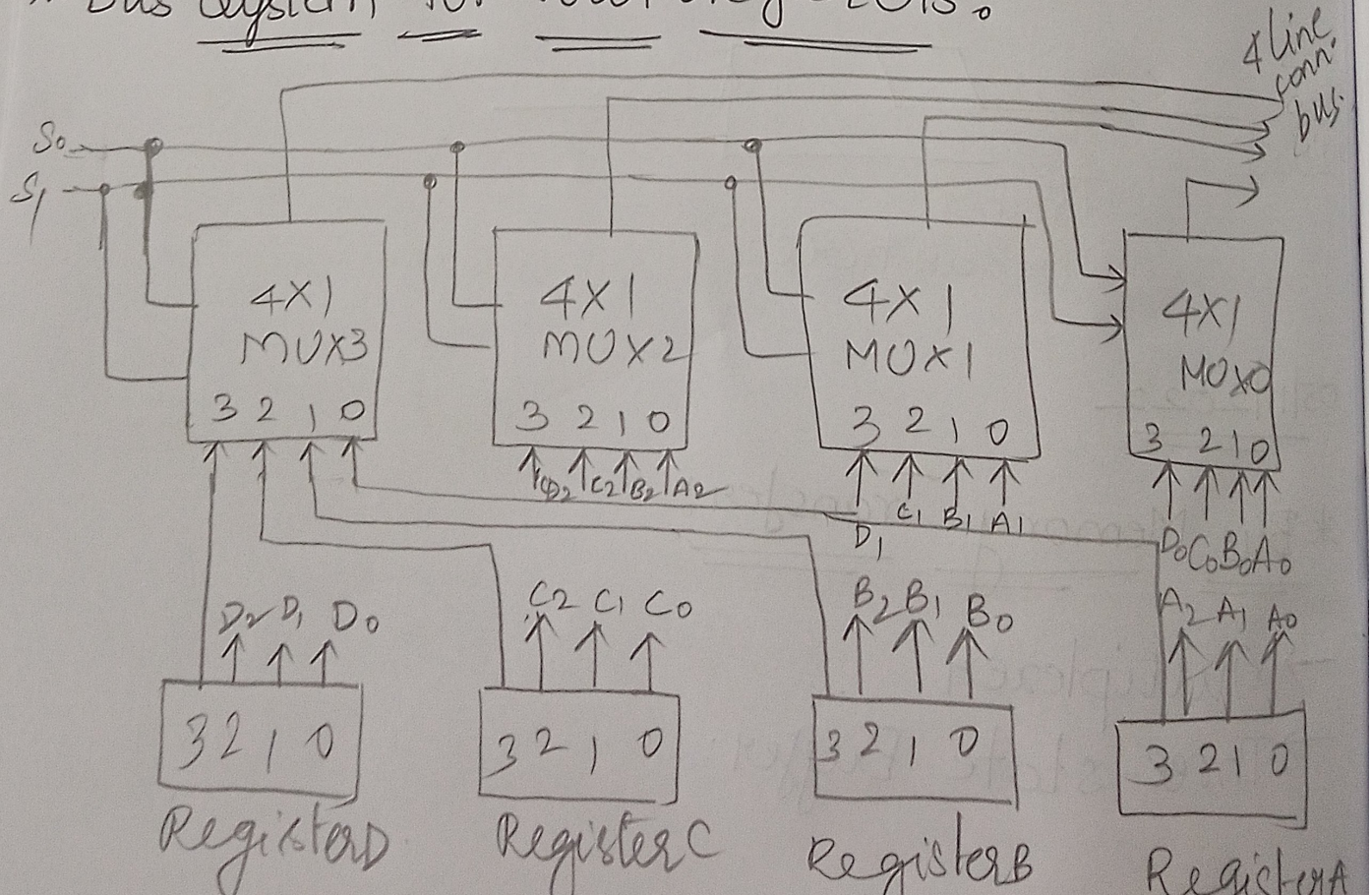
: → Three state Buffer:

. * High impendance state is at 1; there is no power; hence cannot transfer data.

* If impendance state is at 0; We can transfer data.



* Bus System for four registers:



28/10/22

UNIT - I

Overview of Bus Structure

Computer Functions and INTER CONNECTIONS:

→ At top level of computer it consists of

* CPU
* Memory
* I/O components

} interconnected

Computer Components:

→ developed by John von Neumann.

→ 3-concepts of John:-

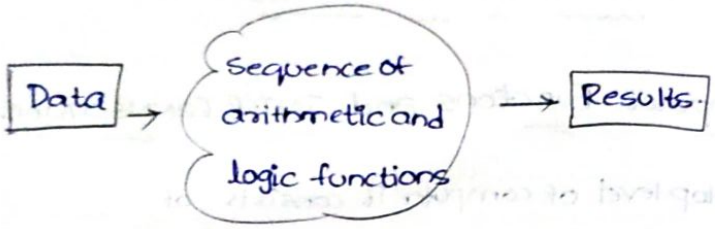
- Data and instructions are stored in a single read-write memory
- The contents of this memory are addressable by location, without regard to the type of data contained there
- Execution occurs in a sequential-fashion (from order) one instruction to the next

a = 5
↓
variable is called as operand.
→ These are stored in registers

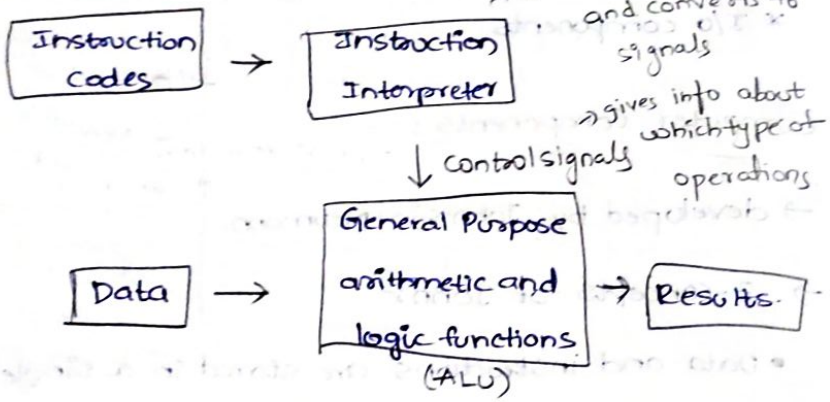
Hardwired Program:

→ The result of connecting (process of) the various components in the desired configuration.

Hardware and Software Approaches:-



(a) Programming in hardware



(b) Programming in software

Software:

→ A sequence of codes or instructions.

→ Part of the hardware interprets each instruction and generates control signals.

→ Provide a new sequence of codes for each new program instead of rewiring the hardware.

Major Components:

* CPU.
* Instruction interpreter
* Module of general-purpose arithmetic and logic funct

I/O Components:-

Input module :-

contains basic components for accepting data and instructions and converting them into an internal form of signals usable by system → (breaks to binary)

output module :-

reports results

To make connection between CPU and memory

Memory (MAR) address Register

- specifies the address in the memory for the next read or write

Memory buffer register. (MBR)

- Contains the data to be written into memory (or) receives the data read from memory

Memory Data register (or)

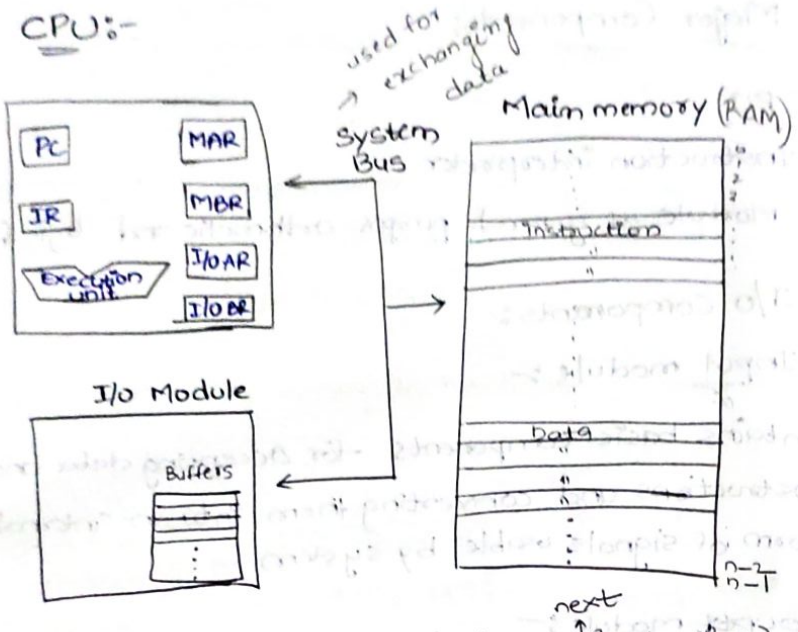
I/O address register (I/OAR)

- Specifies a particular I/O device

I/O buffer register (I/OBR)

- used for the exchange of data between an I/O module and CPU

CPU:-



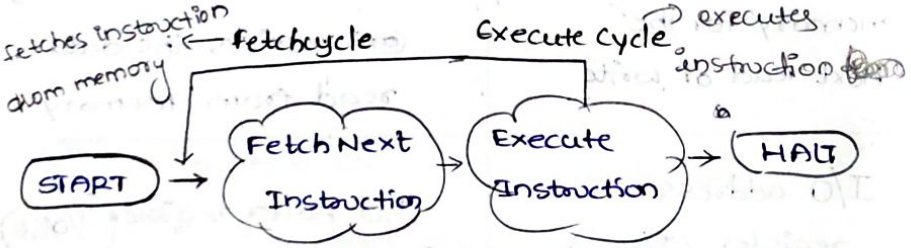
PC = program counter (address of instruction)

PC gives address of next instruction to IR.

IR = Instruction register (instruction is stored here)

Computer Components : Top-Level View.

→ PC will be incremented when one instruction is stored in IR



Basic Instruction Cycle

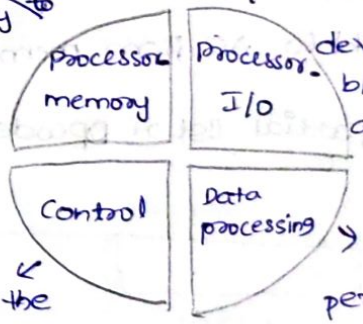
Fetch Cycle :- (reading data)

- At the beginning of each instruction cycle the processor fetches an instruction from memory.
- The PC holds the address of the instruction be fetched next.
- The processor increments the PC after each instruction fetch so that it will fetch the next instruction in sequence
- The fetched instruction is loaded into IR.
- The processor interprets the instruction and performs the required action.

Action Categories :-

Data transfered from processor to memory or from memory to processor

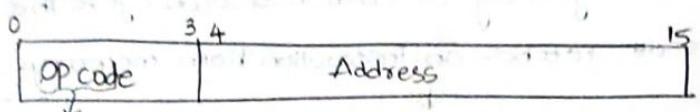
Data transfered to or from a pheripheral device by transfering b/w the processor and an I/O Module



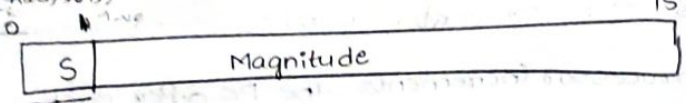
An instruction may specify that the sequence of excution be altered. (ex: in loops (or) if-else statements)

The processor may perform some arithmetic or logic operation on data

Characteristics of a Hypothetical Machine



(a) Instruction format



(b) Integer format

PC = Address of instruction

IR = Instruction being executed

Accumulator (AC) = Temporary storage.

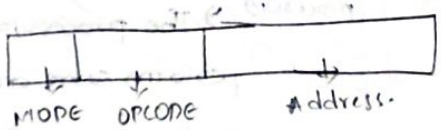
(c) Internal CPU registers

(1) 0001 = Load AC from Memory (LDA)

(2) 0010 = Store AC to memory (STR)

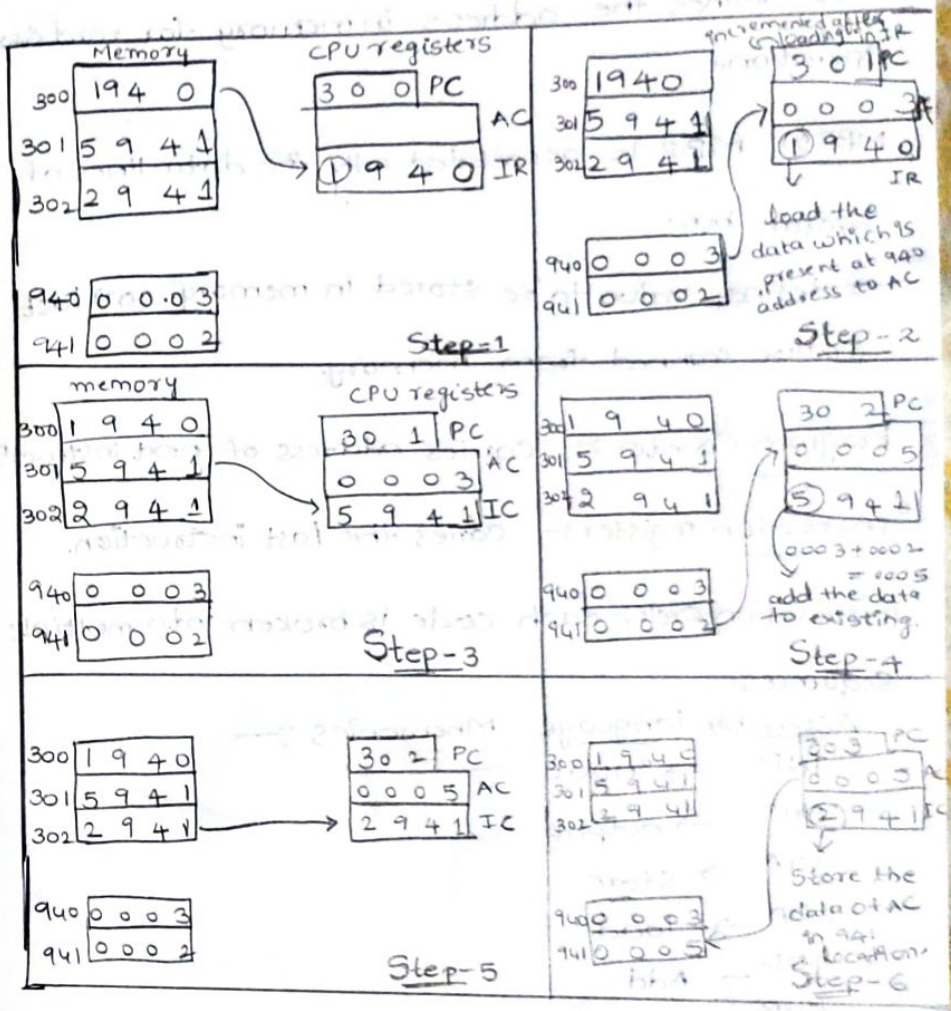
(5) 0101 = Add to AC from memory.

(d) Partial list of Opcodes.



Example of Program Execution.

→ contents of memory and register in hexadecimal.



MAR: Memory address register is associated with address lines of the system bus.

→ It defines the address in memory for read & write operation.

MBR: MBR is associated with the data lines of system bus.

→ defines value to be stored in memory. (or) last value scanned from memory.

Program Counter :- Carries address of next instruction

Instruction register :- Caries the last instruction.

Instruction cycle :- each code is broken into multiple sequences.

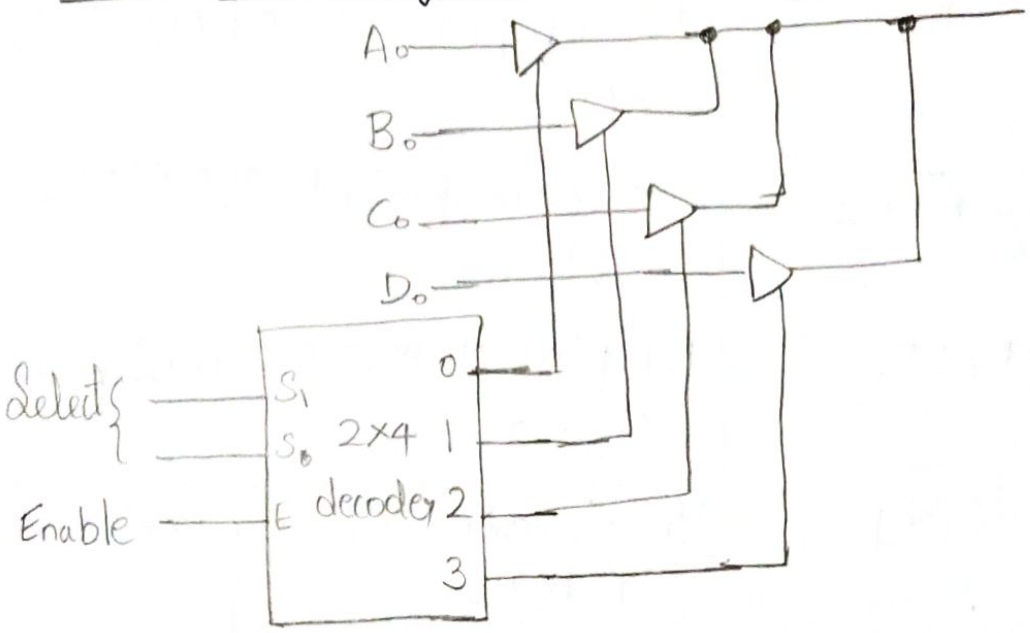
Assembly language Mnemonics :-

INP → input
OUT → output
STA → store
LDA → load
ADD → Add
SUB → subtract
BRA → branch → looping instructions
HLT → Halt/stop/end → stops processor
DAT → Data Defination → Variable defination

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

07/11/2022

*→ Three State Buffers:



To transfer data from one register to another register.

→ Memory transfer: to store data.

* Read Operation: Transfer of data from memory to outside ~~register~~ use.

* Write Operation: Transfer of the data to be stored in memory.

* Arithmetic micro-operation

	System Design	Description
1)	$R_3 \leftarrow R_1 + R_2$	contents of R_1 plus R_2 transferred to R_3
2)	$R_3 \leftarrow R_1 - R_2$	Contents of R_1 minus R_2 transferred to R_3
3)	$R_2 \leftarrow \overline{R_2}$	1's complement of contents of R_2
4)	$R_2 \leftarrow \overline{R_2} + 1$	2's complement of contents of R_2
5)	$R_3 \leftarrow R_1 + \overline{R_2} + 1$	R_1 plus the 2's complement of R_2
6)	$R_1 \leftarrow R_1 + 1$	Increment of contents of R_1 by 1
7)	$R_1 \leftarrow R_1 - 1$	Decrement of contents of R_1 by 1

* Read Operation : $MBR \leftarrow [AR]M$

Transfer of data from the address register into the memory buffer register.

0 → Direct Mode

1 → Indirect Mode

Str → shift right

Stl → shift left

14/11/2022

⇒ MICRO-OPERATION:

* To certain task, the instructions are given in machine language to the CPU.

→ Arithmetic Micro operations (add, sub, Multiply, division)

→ Shift Micro operations → shift micro left, right
↳ circular shift, left, right

→ Logical Micro operations.
↳ (OR, AND, NOT, NOR)

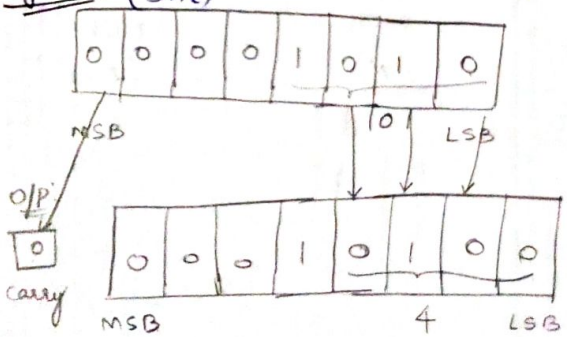
* For logical operations : there are 16 functions ($F_0 - F_{15}$) → Combine

* Shift Micro operations:

→ In Multiplication :- Right to left

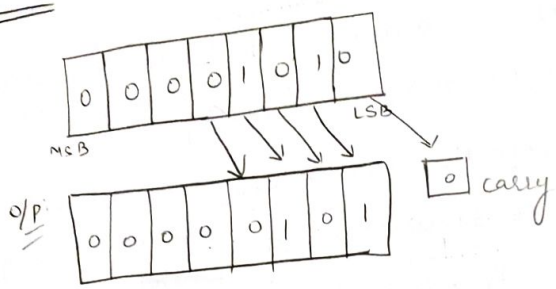
In Division :- left to right

Left Shift (shl)



Left shift is used for multiplication

Right shift :- (for Division) (shr)



Logical Operations:-

OR

$A \vee B$

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

AND

$A \wedge B$

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

XOR

$A \oplus B$

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

NAND

$(A \wedge B)'$

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

NOT

$(A)'$

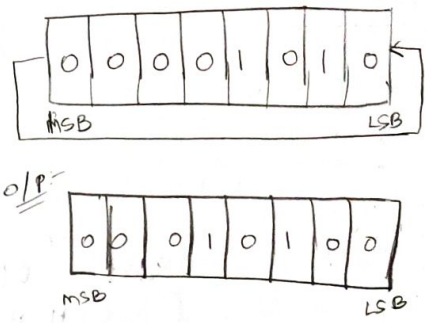
A	A'
0	1
0	1
1	0
1	0

NOR

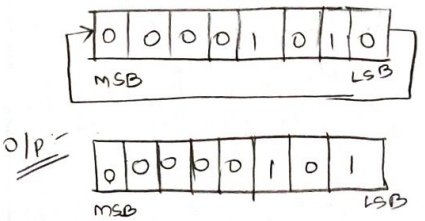
$(A \vee B)'$

A	B	$(A \vee B)'$
0	0	1
0	1	0
1	0	0
1	1	0

Circular left shift (crl)



Circular right Shift (crr)



→ Data Manipulation happens

→ We will not loose a single bit

28/11/2022

* Shift Micro Operations:

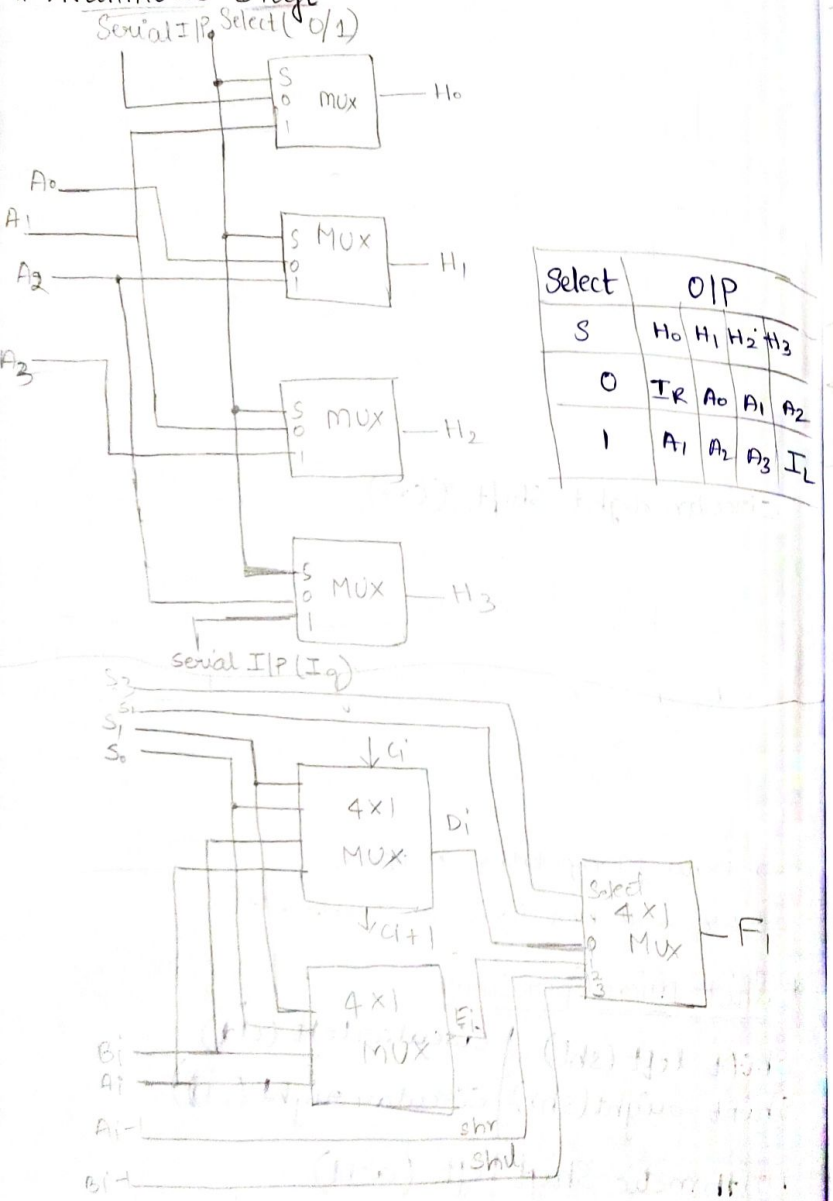
Shift left (shl) | Circular left (cil)

Shift right (shr) | Circular right (cir)

Arithmetic Shift Left (ashl)

Arithmatic Shift Right (ashr)

* Arithmatic Shift.



Select	O/P			
S	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

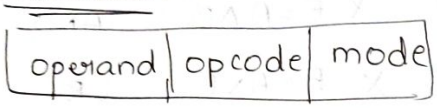
S_3	S_2	S_1	S_0	Cin	Operation	Functions
0	0	0	0	0	F = A	Transfer A
0	0	0	0	1	F = A + 1	Increment A
0	0	0	1	0	F = A+B	Addition
0	0	0	1	1	F = A+B+1	Add with carry
0	0	1	0	0	F = A + $\overline{B}$	Subtract with borrow
0	0	1	0	1	F = A + $\overline{B}$ + 1	Subtraction
0	0	1	1	0	F = A − 1	Decrement
0	0	1	1	1	F = A	Transfer A
0	1	0	0	X	F = A ∧ B	AND
0	1	0	1	X	F = A ∨ B	OR
0	1	1	0	X	F = A ⊕ B	XOR
0	1	1	1	X	F = $\overline{A}$	Complement A
1	0	X	X	X	F = shr A	shift right A into F.
1	1	X	X	X	F = shl A	shift left A into

* BASIC COMPUTER ORGANIZATION AND DESIGN:

Part 1:
→ Instruction codes
→ Computer registers
→ " Instructions
→ Timing control
→ Instruction Cycle
→ memory reference instruction.
→ Input / Output & Interrupt.

* Part -2: Microprocessing.

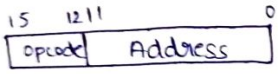
* Instruction:



*→ Instruction Code:

Group of bits that instruct the computer to perform a specific task.

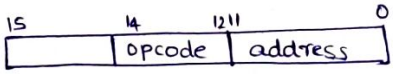
Format of OPCODE:

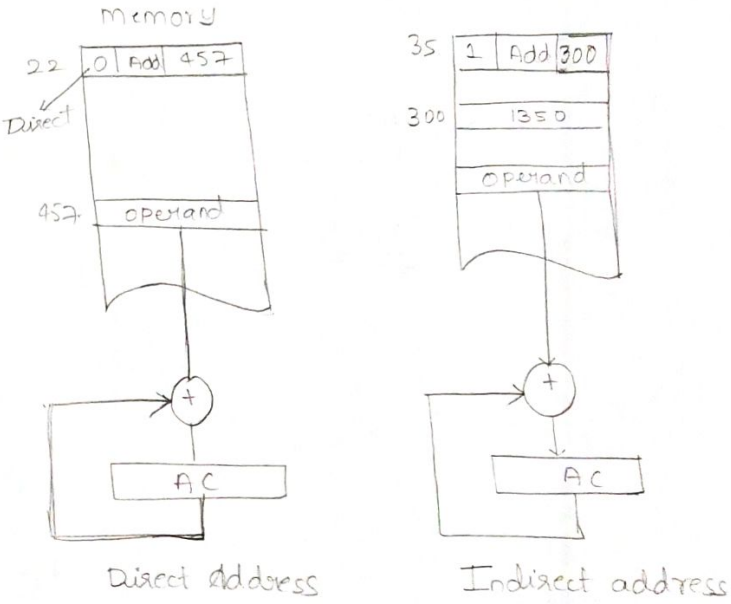


⇒ Instruction format

* AC - Accumulator
Indirect Address
Direct address
Immediate Instruction
Effective address

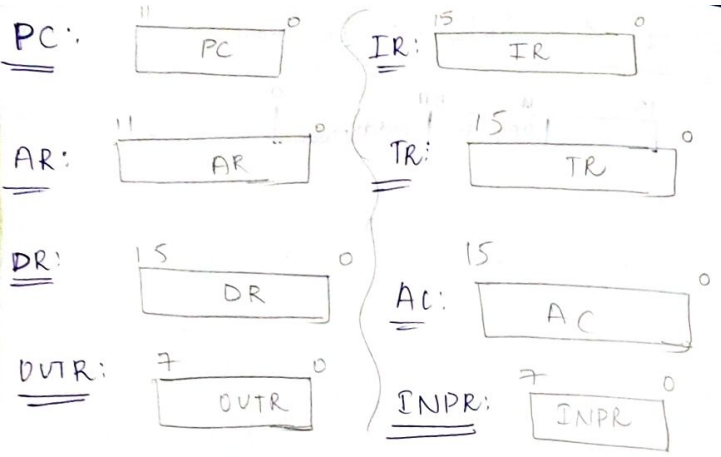
* Direct & Indirect Address:

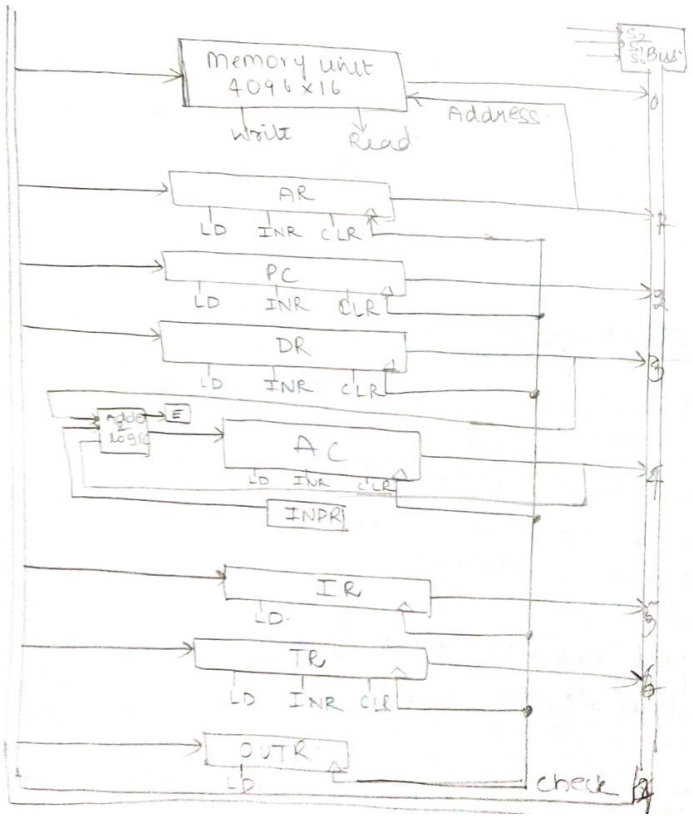




* Computer Registers:

→ Data register (16 bit) (DR)
→ Address Register (12 bit) (AR)
→ Accumulator (16 bit) (AC)
→ Instruction Register (16 bit) (IR)
→ Program counter (12 bit) (PC)
→ Temporary Register (16 bit) (TR)
→ Input Register (8 bit) (INPR)
→ Output Register (8 bit) (OUTR)





1001 LDA $R_1 \rightarrow AR$

1002 LDA $R_2 \rightarrow DR$

1003 ADD $R_1, R_2 \rightarrow IR$

1004 HALT

S_2	S_1	S_0	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Based on this truth table, the computer register is been used.

→ Opcode holds 3 bits of data.

* Remaining 13 bits depends on operation code.

* Memory reference instruction uses 12 bits of data.

* Three Instruction Formats.

→ Memory Reference

→ Register Reference.

→ I/O Instructions.

29/11/2022

* Control signals are generated in the control unit and provides control inputs for the multiplexers and bus system.

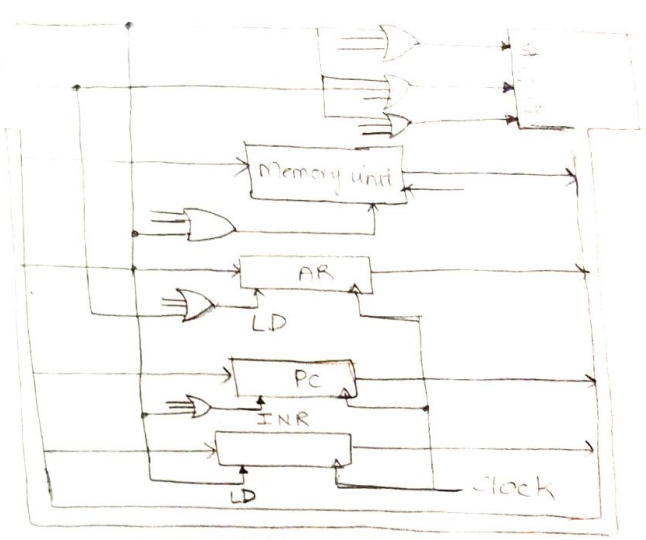
*Instruction Cycle:

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

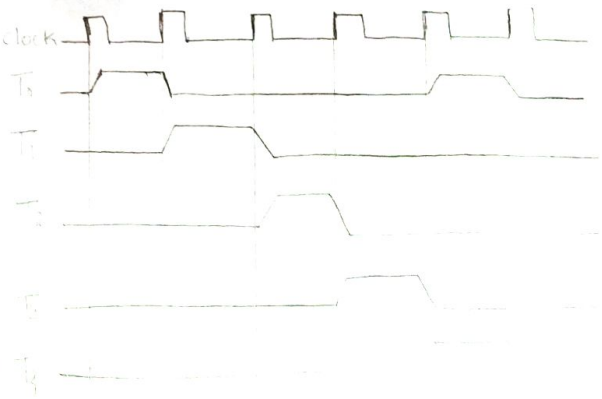
→ Fetch and decode:

1) PC is loaded with address of first instruction in the program.
2) Initially SC = 0, after each clock pulse, SC is incremented by 1.

* For every phase, timing diagram is generated



Example for control timing signals:



01/12/2022

* Timing and Control:

→ 2 Decoders
→ Sequence Counter
→ Logic circuit

} Components of control signals.

* Micro-operations: Logic, Arithmetic, Shift.

* The timing diagram is generated based on no. of instructions given in specific code.

* Memory Reference Instructions are used in generating timing control signals.

* Memory Reference Instructions:

AND
ADD
ICZ
BUN
SHR

} These instructions are used to generate signals.

→ Branch unconditional.

~~* 4001~~

* AC = IR = If partial output is generated, it is held by AC.
PC = DR =
Sq = 0 AR =

* → 4001 LDA R1
4002 LDA R2
4003 MUL R1, R2
4004 – STORE R1
4005 – HALT

R ← M[AR] ; R1 ← BUN[AR]

* Register Reference Instruction:

→ ADD R1, R2
→ AND R1, R2

$R_3 \leftarrow R_1 + R_2$

$R_3 \leftarrow M[R_1 + R_2]$

$R_3 \leftarrow ADD[R_1, R_2]$

* Input and Output Registers:

→ ADD R1, R2
STORE R1
R1 = 05 , R2 = 04
0101 = 0100

MUL

MUL R1 R2