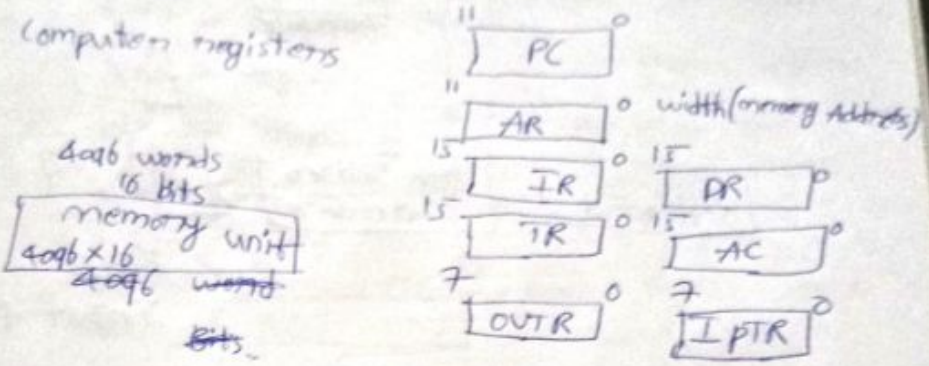
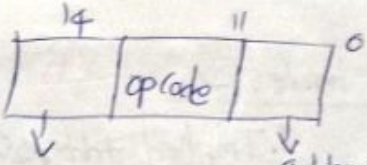
Logic shift unit

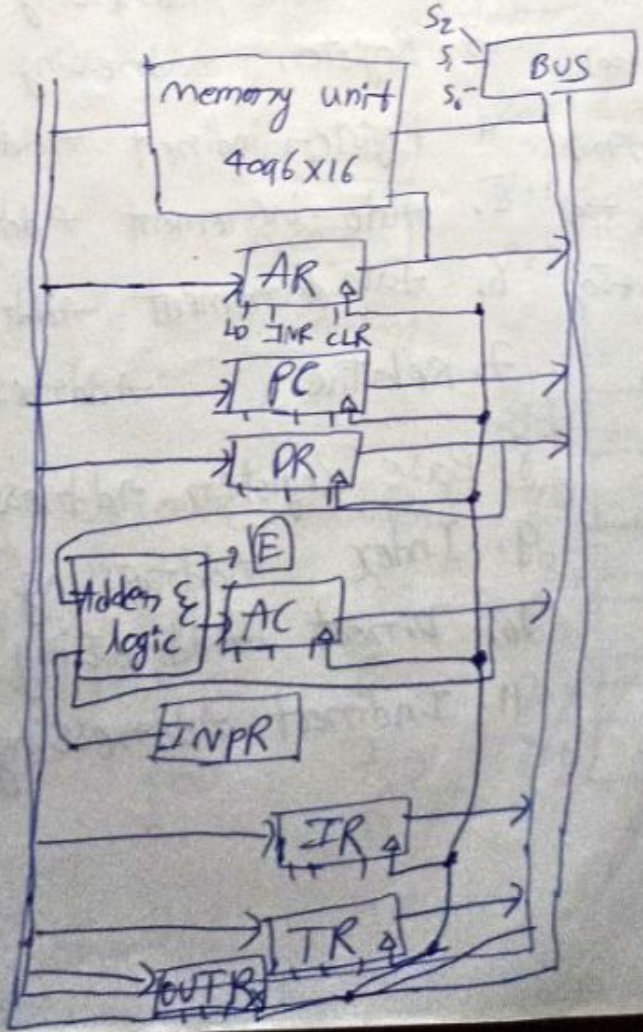
UNIT-II

Basic computer organisation & Design

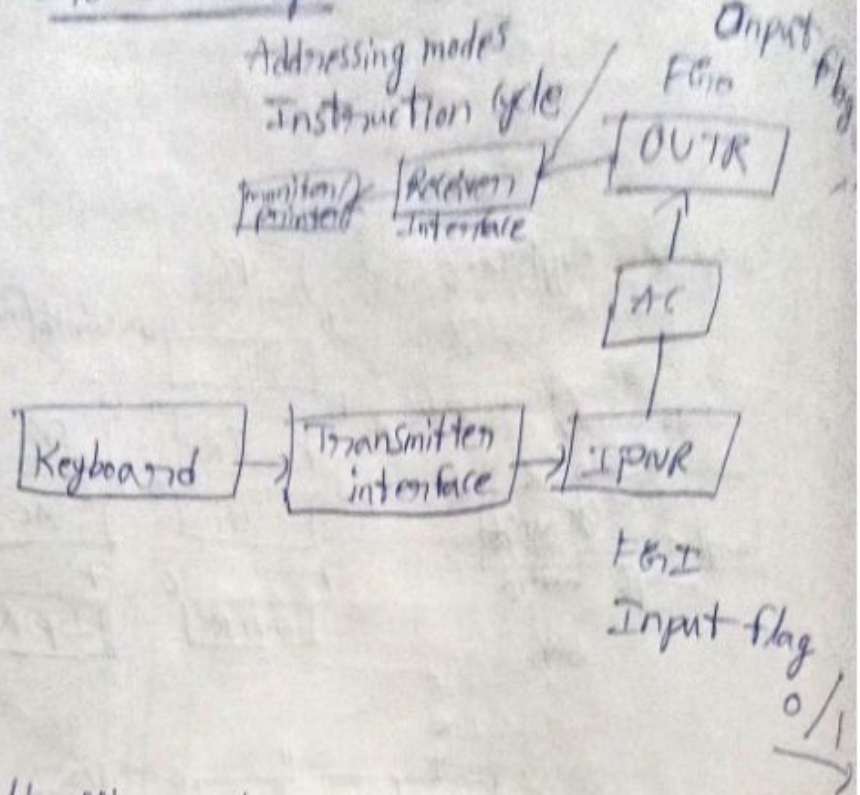
Computer registers







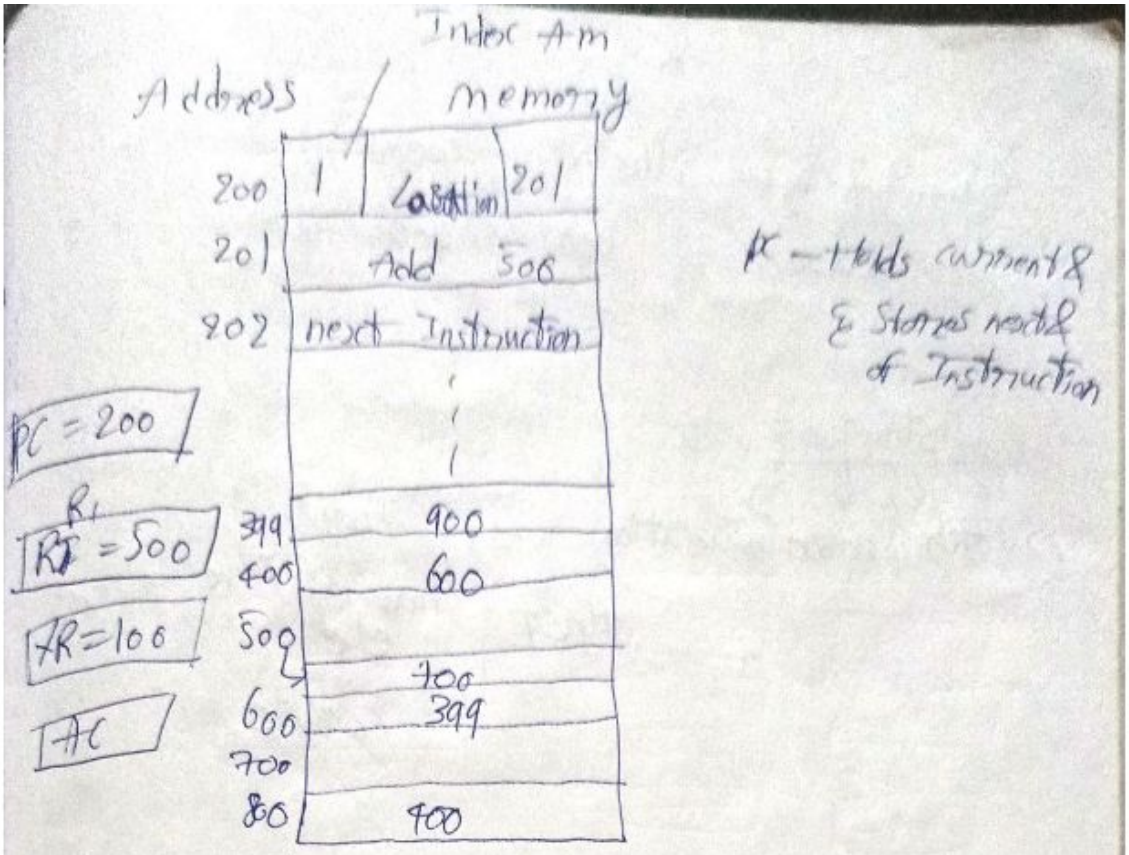
=) I/O Interrupt:-



=) Addressing modes:-

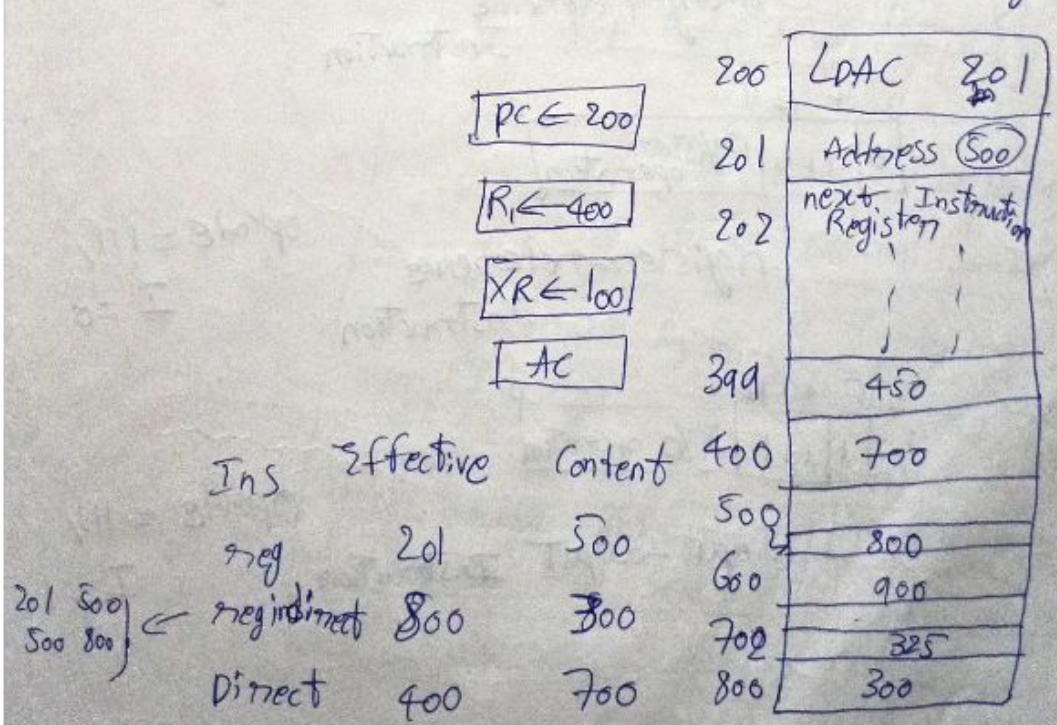
1. Implied Addressing mode
2. Immediate Addressing mode
3. Register Addressing mode
4. Register Indirect Addressing mode
5. Auto increment Addressing mode
6. Auto decrement Addressing mode
7. Relative Addressing mode
8. Base Register Addressing mode
9. Index Addressing mode
10. Direct Addressing mode
11. Indirect Addressing mode

(20,500)
(-,400)
(400,700)
(400,700)
(399,450)
(702,325)
(600,400)
(500,800)
(800,300)



1. no need to specify the operand.

2. $R_1 \leftarrow (400)$ I.A.m, Ex:- mvi B 45



Ins	Effective	Content
reg	201	500
reg indirect	800	300
Direct	400	700
Indirect	800	300
Relative	702	325

201 500
500 800
201 500
500 800
(201+1)
202 + 500 => 702

XR
100+ 500

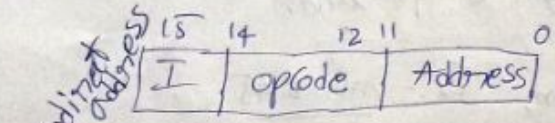
Ins	Effective	Content
Base	600	900

Instruction → Fetch → Decode → Execution

If Indirect → Effective Address
Read
else
Execute

HALT

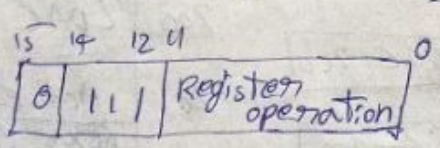
Computer Instruction formats (3 bits performs 8 operations, i.e.)



I - Indirect address

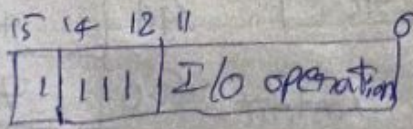
Opcode = 000 through 110

Memory reference Instruction



Register-reference Instruction

Opcode = 111, I = 0



Input-output Instruction

Opcode = 111, I = 1

LDA - Load Accu
BUN - Uncondit
BSA - Subpro
ISZ - Conditi
LDA - Load
STA - Stor

	I=0
AND	0XXX
ADD	1XXX
LDA	2XXX
STA	3XXX
BUN	4XXX
BSA	5XXX
ISZ	6XXX

LDA - Load Accumulator

BUN - Unconditional Branches

BSA - Subprogram calls

ISZ - Conditional branches

LDA - Load Accumulator

STA - Store Accumulator

CLA - Clear AC

CLE - Clear E

CMA - Complement AC

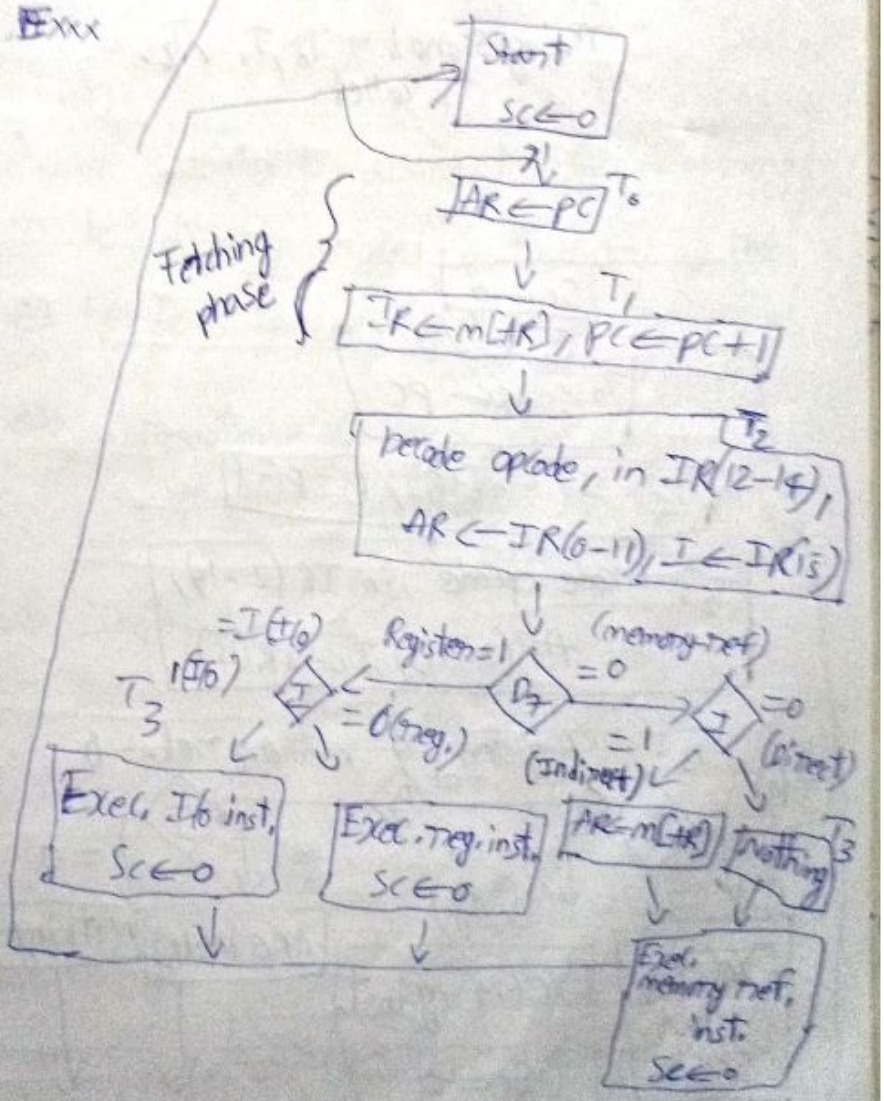
CME - Complement E

CIR - Circular Right AC & E

CIL - Circular left AC & E

INC - Increment AC

	I=0	I=1
AND	0XXX	8XXX
ADD	1XXX	9XXX
LDA	2XXX	AXXX
STA	3XXX	BXXX
BUN	4XXX	CXXX
BSA	5XXX	DXXX
ISZ	6XXX	EXXX



:) Instruction Cycle:

→ Program → memory

↓

Set of Instructions

Program Counter →
Contains the address of next Instruction.

1) Fetch
2) Decode
3) Read the Effective Address
4) Execute
HALT

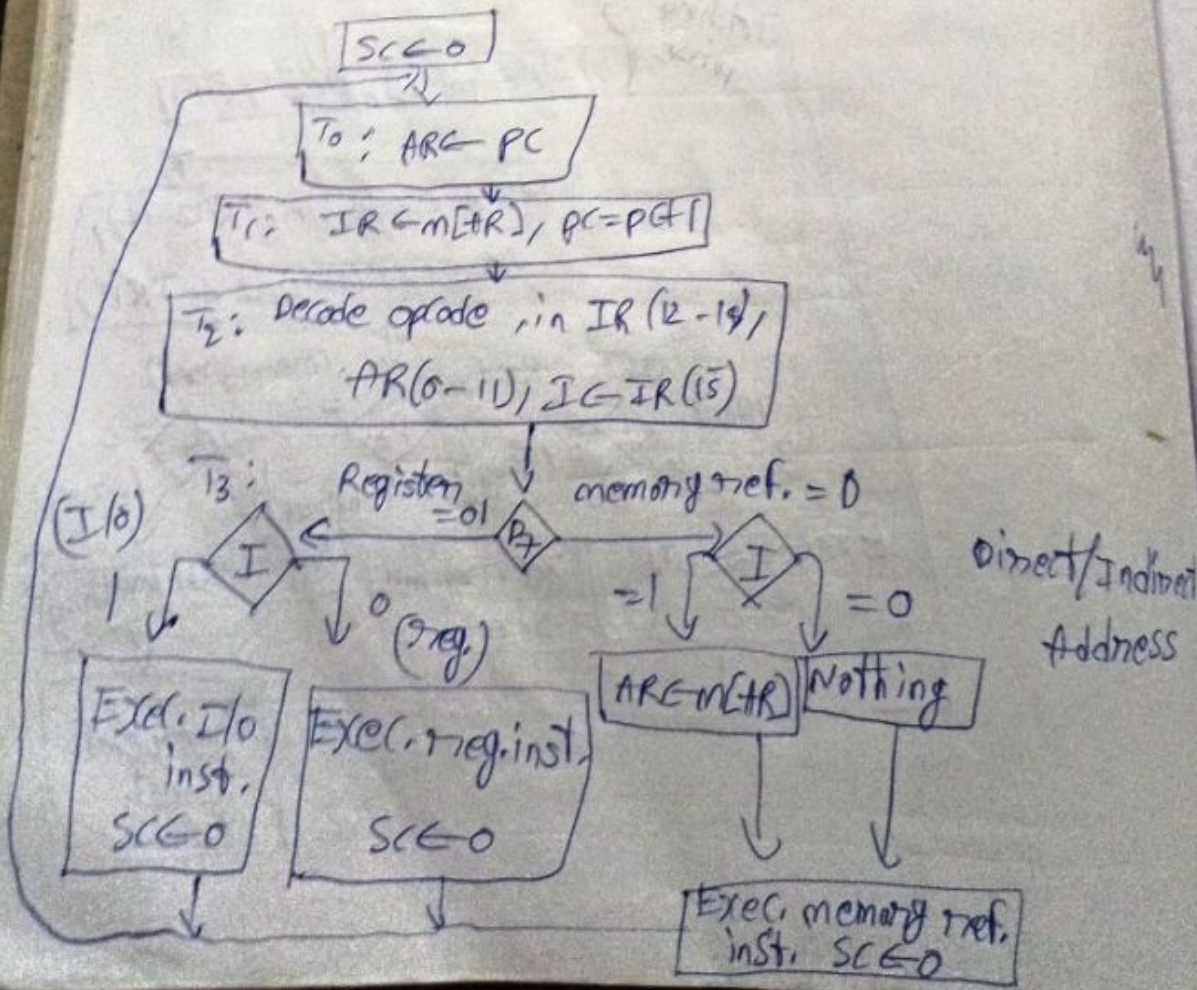
Clock pulse →

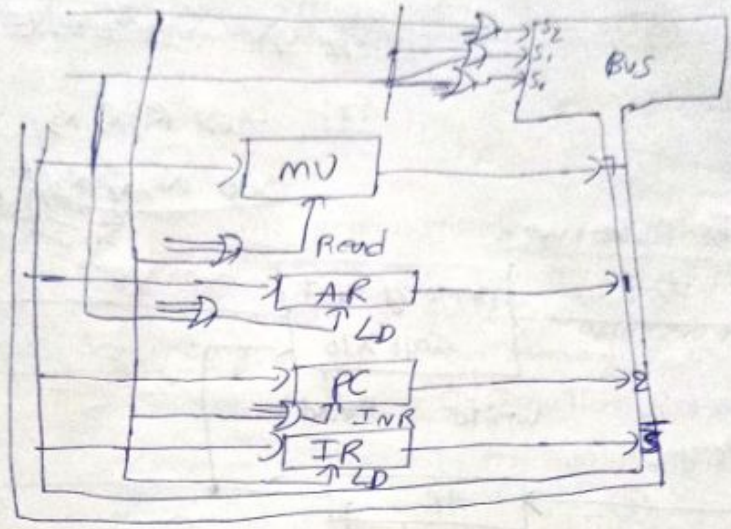
SC PC

↓ ↓

Timing Signal T_0, T_1, T_2. is called

$D_0, D_1, \ldots D_7$





Instruction format:-

Standard machine instruction format that can be directly decoded & executed by the central processing unit (CPU).

Zero Address Instruction:-

| mode | opcode |

Ex:- CMA
CME

One Address Instruction:-

| mode | opcode | operand |

Ex:- ADD 06H
LDA 20H

Two Address Instruction:-

| mode | opcode | operand1 | operand2 |

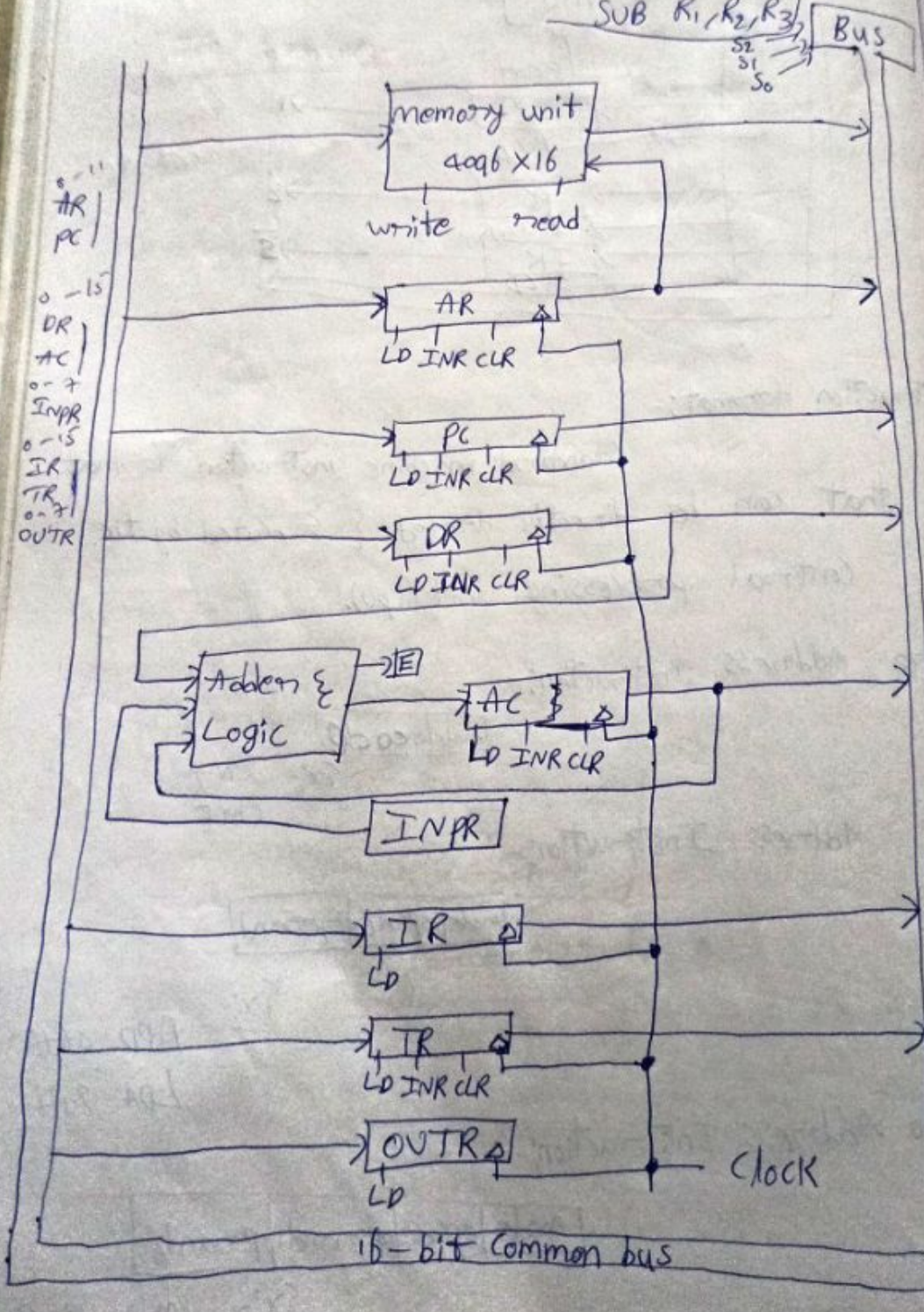
Ex:- MOV R_1, R_2
ADD AX, BX

Three address Instruction:-

mode	opcode	operand1	operand2	operand3

Ex:- ADD R_1, R_2, R_3

SUB R_1, R_2, R_3



UNIT-

=> Control

=> Address

Op-code

mapping

Control

Control

Incre

Contr

Uncond

depend

Address

next

Instruc

Next

Address

Generate

④