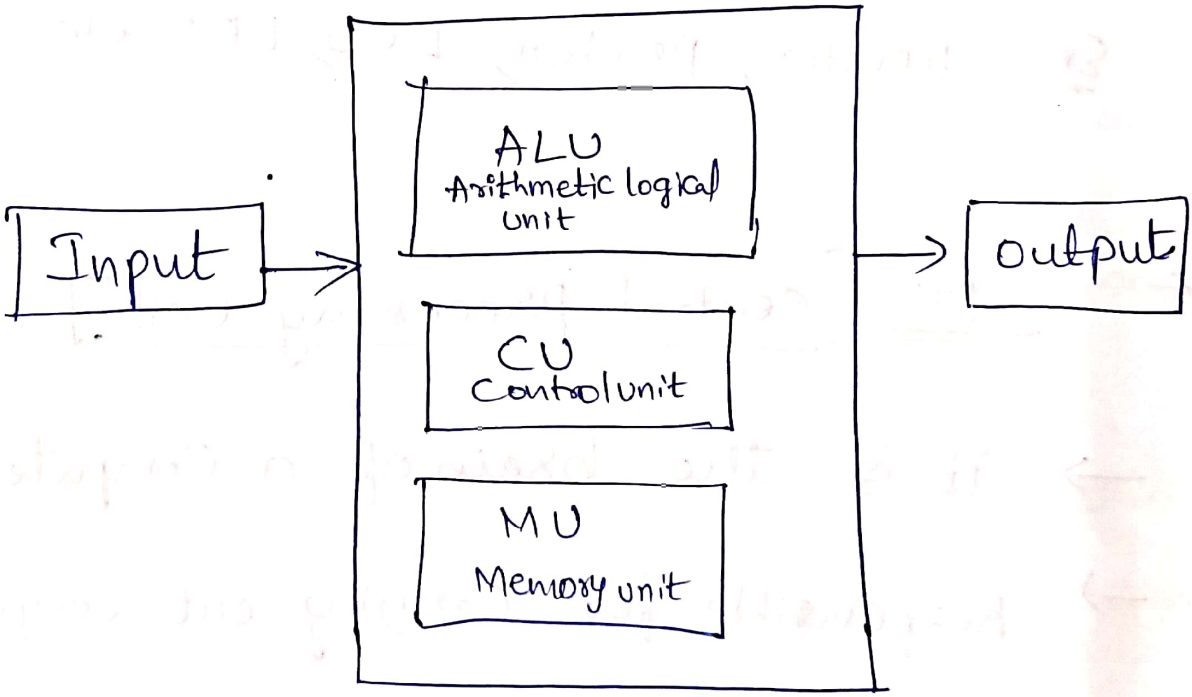
UNIT-I

Functional Blocks of A Computer

1. Input Unit
2. Output unit
3. Central processing unit (ALU + Control unit)
4. Memory
5. Bus structure



* Functional Blocks of a Computer

* Input Unit: Input unit Converts the External World data to a binary format which can be understood by CPU

Ex: Keyboard, Mouse, Joystick etc

* Output Unit: Output unit Converts the binary format data to a format that a common man can understand.

Ex: monitor, printer, LCD, LED etc.

* CPU [central processing unit]

→ It is the brain of a Computer

→ Responsible for Carrying out Computational task

→ It Contains ALU and CU, Registers

→ ALU performs Arithmatic and Logical

operations.

→ CU provides Control signals in Accordance with some timings which in turn Control the Execution process

→ Registers Stores data and result and Speeds up the operation.

* Example Add R1,R2

T_1 ⟶ Enable R1

T_2 ⟶ Enable R2

T_3 ⟶ Enable ALU for addition operation

T_4 ⟶ Enable output of ALU to store result of the operation

Memory

→ Memory stores the data, Results and Programs.

→ Two class of storage

i) primary ii) secondary

↓ ↓

RAM (Random Access Memory)

ROM (Read Only memory)

Registers

Register's are fast stand-Alone storage locations that holds the data temporarily

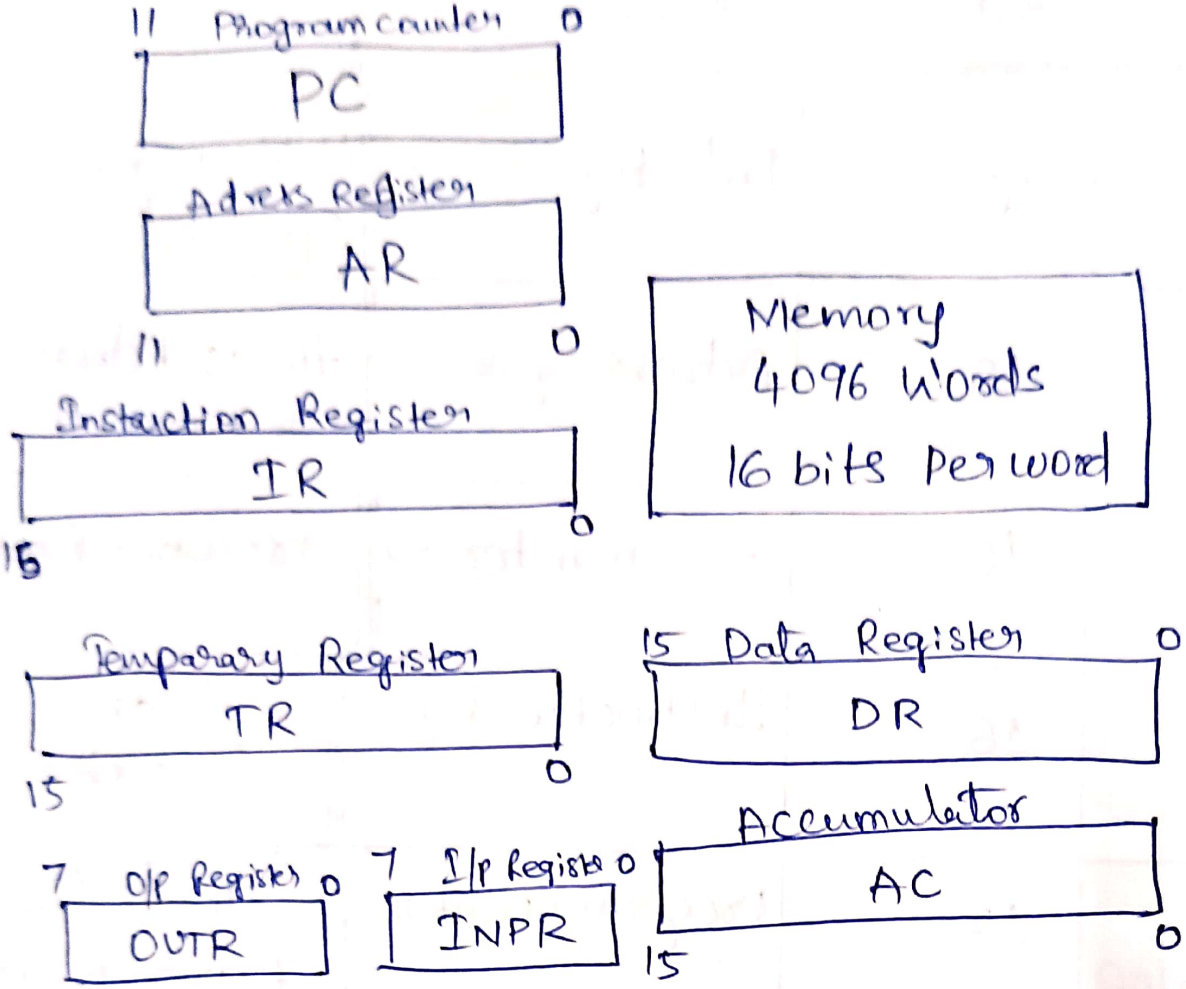
→ Multiple Register's are Needed to facilitate the Operation of CPU

* A Register is a group of flip-flops with Each flip-flop Capable of Storing one bit of information.

S.NO	Register Symbol	No. of Bits	Register Name	Function.
1)	DR	16	Data Register →	Holds memory Operand
2)	AR	12	Adress register →	Holds Adress for memory
3)	AC	16	Accumulator →	Processor Regis
4)	IR	16	Instruction Register →	Holds Instruction cod
5)	PC	12	Program Counter →	Holds the Adress of an Instruction
6)	TR	16	Temporary Register →	Holds tempory data
7)	INPR	8	Input Register →	Holds input Character
8)	OUTR	8	output Register →	Holds output Character

→ MAR : memory Adress Register's

→ MDR : Memory Data Registers



i) Basic Computer Registers and memory.

i) Program Counter.

The program Counter is a Register that manages the memory adress of the instruction to be Executed Next.

→ The Adress specified by PC will be "+n"

(i.e) +1 for 1 word instruction

(or) +2 for 2 word instruction.

ii) Adress Register:

The Adress Register is a Register that stores the Adress of the data to be processed.

iii) Instruction Register:

The IR is used to store the instruction word.

→ When the CPU fetches an instruction from memory it is temporarily stored in the IR.

iii) Temporary Register (TR)

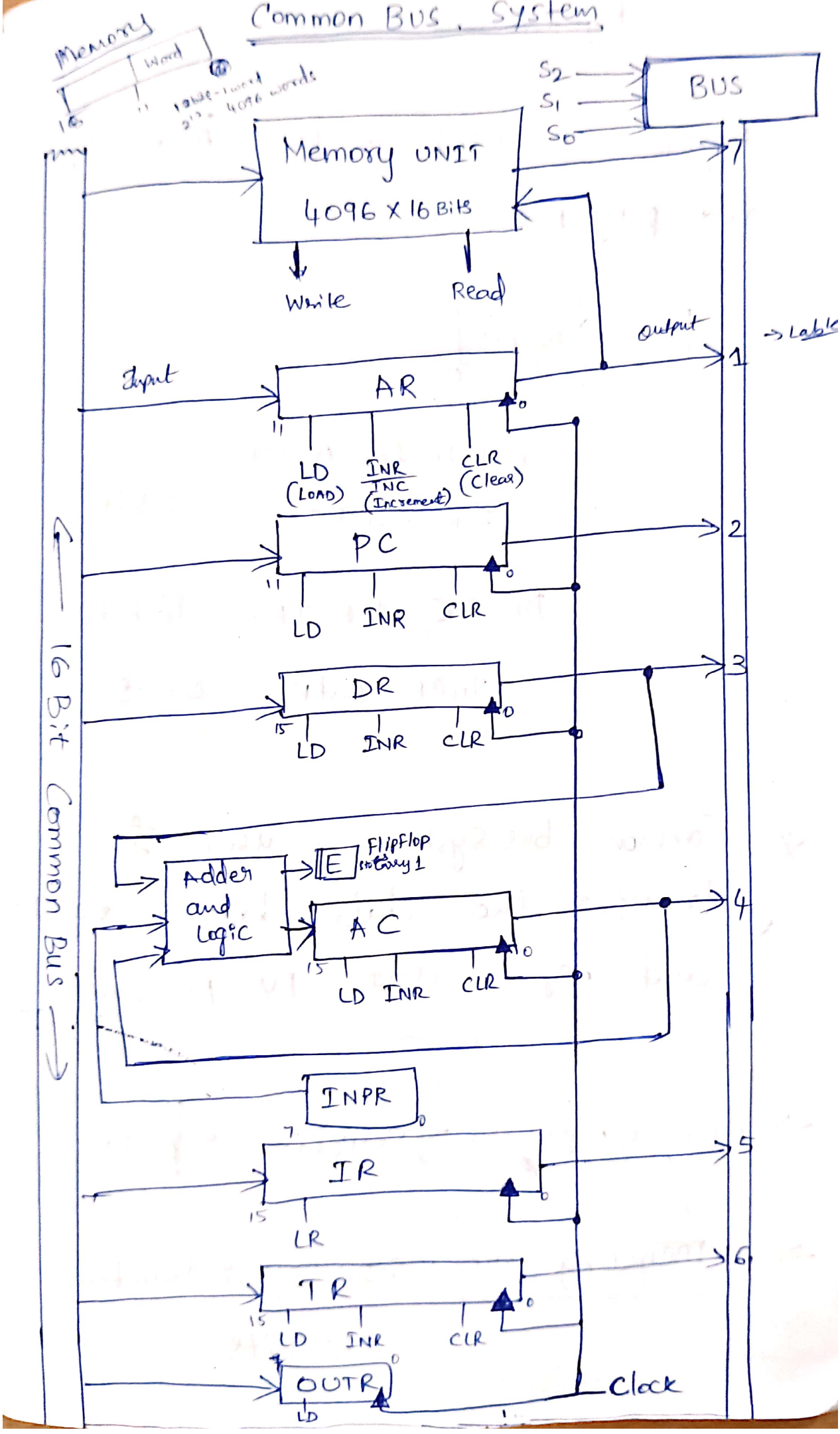
A Temporary Register is the only register that can be read and written more than once in a single instruction.

V) Data Register: Holds the data (RAM)

VI) Accumulator:

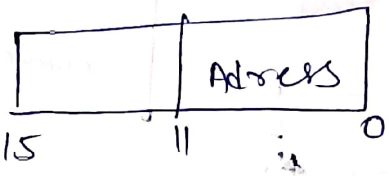
An Accumulator is a type of Register for short term, intermediate storage of arithmetic and logic data in a computer's Central processing unit (CPU).

Common Bus System



Registers and Common Bus System

→ Registers are used to fast Access of data by processor



12 bits for Adress

AR and PC → 12 bits

DR, AR, IR, TR → 16 bits

INPR, OutR → 8 bits

* Common Bus System is used to transfer the data between registers and register and CPU (Processor)

→ S0, S1, S2 → selection inputs

→ memory unit : size of instruction 16 bits

first 12 bits Adress

↳ 2^{12} words

$2^{10} = 1024$

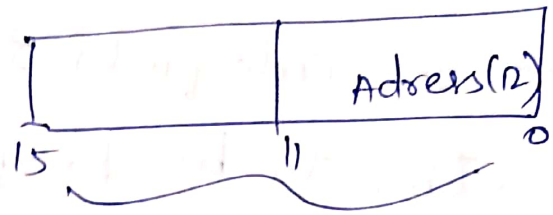
$2^{11} = 2048$

$2^{12} = 4096$ words

We can Store in memory

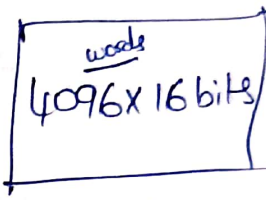
and size of Instruction is 16 bits

(ie)



Instruction(16)

(i,e)

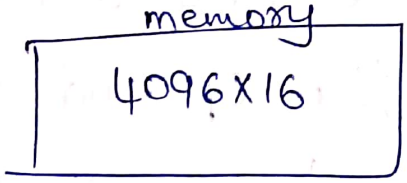


4096 words (16 bit Instruction)

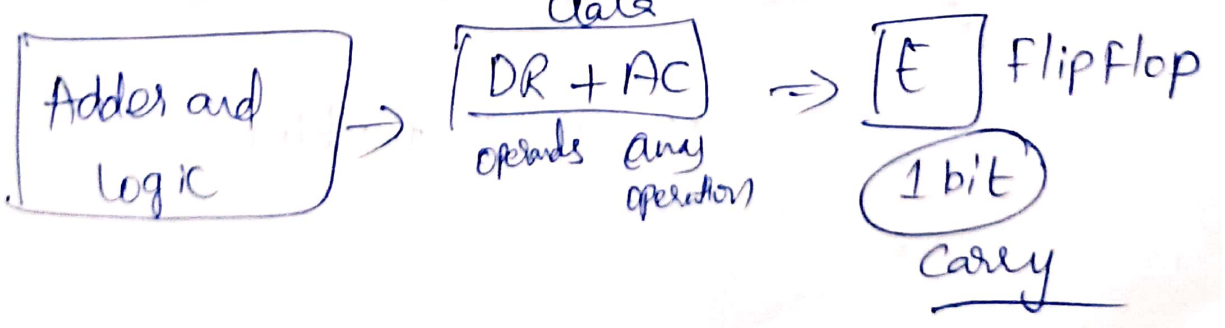
1 word = 16 bits (16)

Memory can store 2^{12} word's

(i,e) 4096 words of 16 bit memory



if $S_0 - 1$, $S_1 - 1$, $S_2 - 1$ } (T) table memory can be Accessed.



Computer Instruction

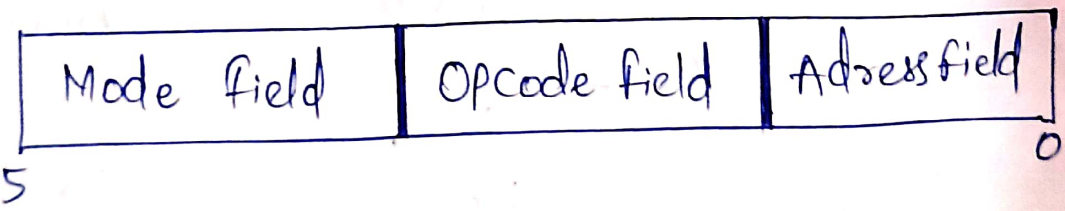
* A Computer Instruction set is the set of groups called fields

→ The most common fields in instruction are:

1) A Operation code field that specifies the operation to be performed.

2) An Adress field that designates a memory address (or) a processor register

3) A mode field that specifies the way the operand (or) the Effective Adress is determined.



16 Bit instruction

* The basic Computer has three Instruction code formats, Each of 16 Bits.

→ The Operation code (opcode) part of instruction Contains 3 bits and the meaning of 13 bits depends on the Operation Code Encountered.

① Memory Reference Instruction

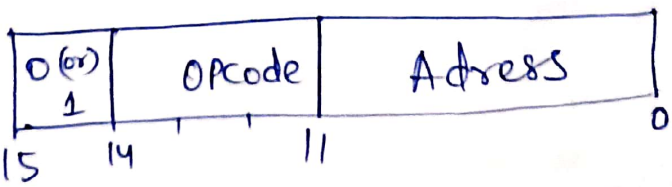
② Register - Reference Instruction

③ Input - Output Instruction.

① A memory Reference: (opcode = 000 - 110)

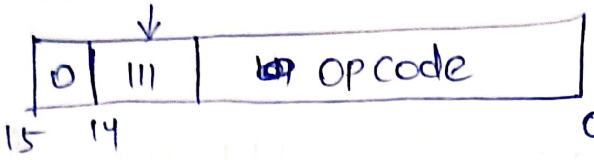
(I = 0, 1)

→ It uses 12 bits to specify an address and one bit to specify the addressing mode and 3 bits to opcode



(2) Register - Reference : (opcode = 111) (I = 0)

Ex



(3) I/O op Instruction : (opcode = 111, I = 1)

INSTRUCTION CYCLE

* A program residing in the memory Unit of the Computer Consists of a sequence of instructions.

* A program is Executed in the Computer by going through a cycle for Each instruction. Each instruction cycle in turn is Sub divided into a sequence of Sub cycles (or) phases.

1) Fetch an instruction from memory.

2) Decode the instruction.

3) Read the Effective Adress from memory if the instruction has an indirect address.

(4) Execute the instruction.

FETCH AND DECODE

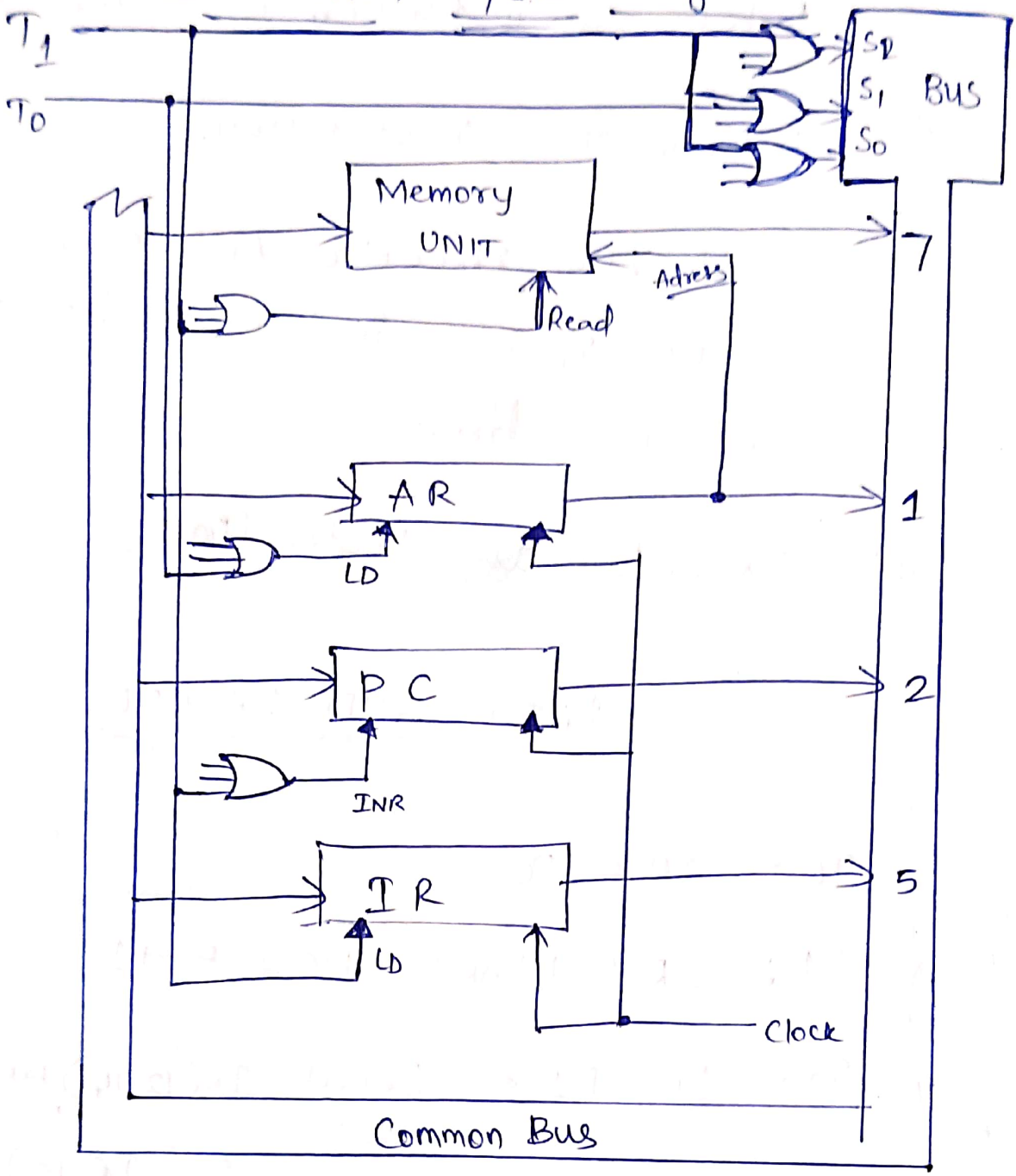
T0 : AR ← PC

T1 : IR ← M[AR] , PC ← PC+1

T2 : D0 ... D1 ← Decode IR(12-14), AR ← IR(1-11)

I ← IR(15)

Instruction Cycle Diagram



FETCH & DECODE

Step-1 T0: AR ← PC

Step-2 T1: IR ← M[AR], PC ← PC+1

Instruction Cycle

The Above figure shows how the first two registers are transfer statements are Implemented in Bus System.

→ To provide the data path for the transfer of PC to AR we must Apply timing signal T0 to Achieve the following Connection.

1) place the Content of PC onto the bus by making the bus selection inputs $S_2 S_1 S_0 = 010$. (i.e) (G2)

2) Transfer the Content of the bus to AR by Enabling the LD input AR

3) The Next Clock transition initiates the transfer from PC to AR since $T_0 = 1$

→ In order to Implement the second Statement

(i.e) T_1: $IR \leftarrow M[AR]$, $PC \leftarrow PC + 1$

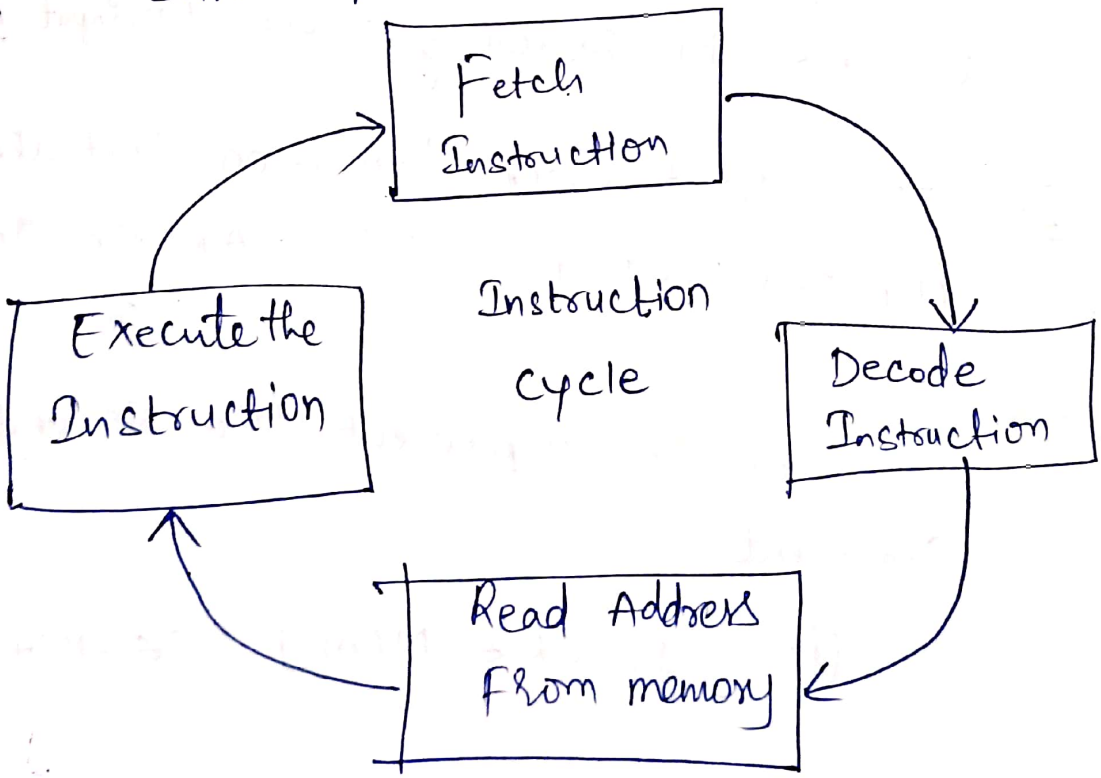
* It is necessary to use timing signal T1 to provide the following Connections in the bus systems

1) Enable the Read input of the memory.

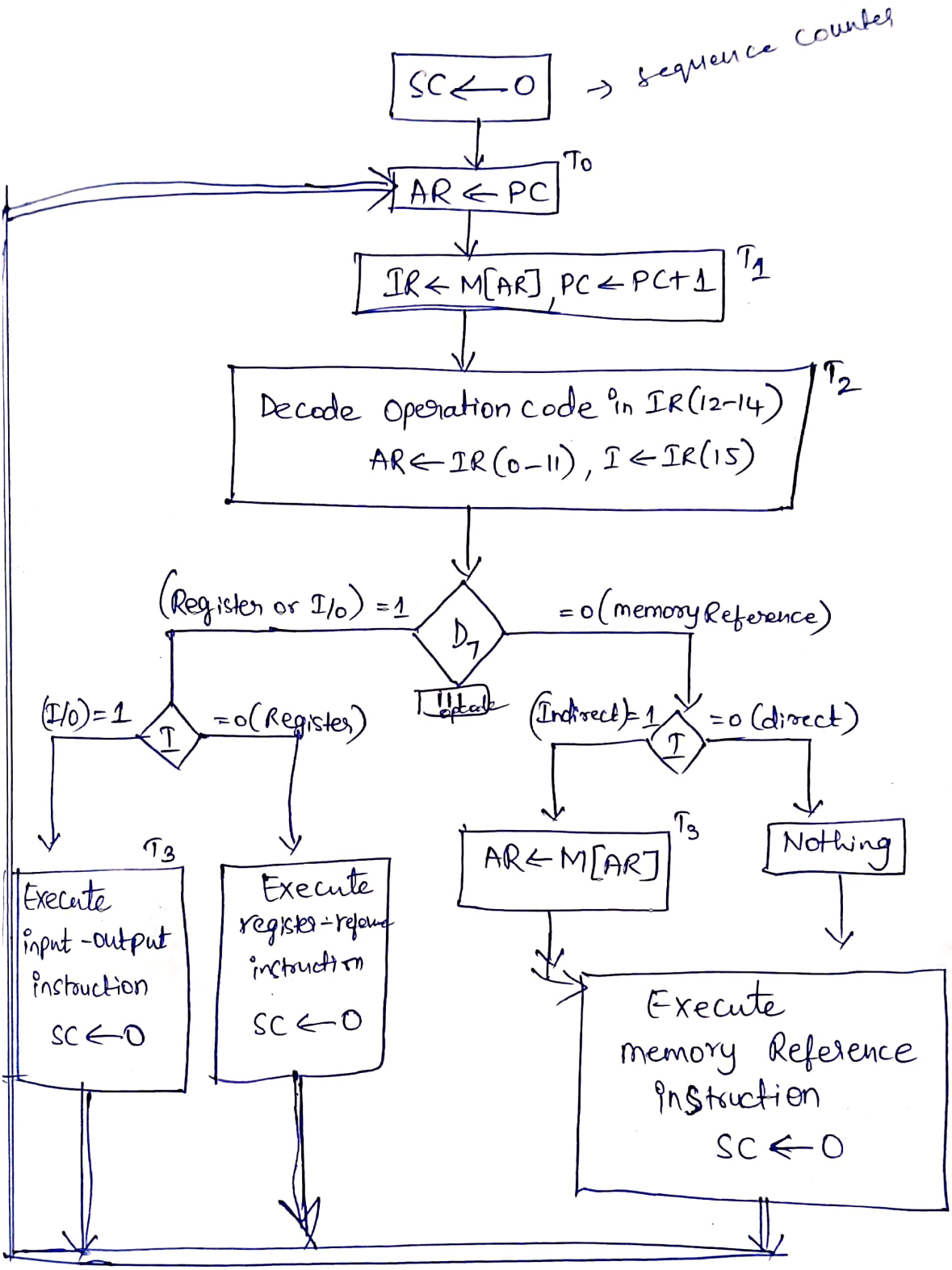
2) place the Content of the memory onto the bus by making $S_2S_1S_0 = 111$

3) Transfer the contents of the bus to IR by Enabling the LD input of IR

4) Increment PC by Enabling the INR input to PC



Flow Chart for Instruction Cycle



ADDRESSING MODES

→ The operation field of an instruction specifies the operation to be performed

→ This operation must be Executed on some data stored in Computer registers (or) Memory words.

→ The way the operands are chosen during program Execution is dependent on the Addressing mode of Instruction.

* Computers use addressing mode techniques are

i) It provides
   1. pointers to memory
   2. Counters to loop
   3. Indexing of data
   4. program Relocation.

ii) To Reduce the No. of bits in the addressing field of the instruction.

iii) More Efficient, with respect to Number of instructions and Execution Time

Types of Addressing modes

1) Implied Addressing Mode
2) Immediate Addressing mode
3) Register Addressing Mode
4) Register Indirect Addressing mode
5) Auto-Increment (or) Auto-decrement (Addressing mode)
6) Direct Addressing Mode
7) Indirect Addressing Mode
8) Relative Addressing Mode
9) Index Addressing Mode
10) Base Register Addressing Mode.

① Implied Addressing Mode

Operand is specified implicitly in the difinition of instruction

→ Size of instruction is Reduced

Ex

No Address Needed directly give data in Instruction

INC A

AC ← AC+1

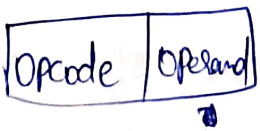
② Immediate Addressing mode (Constant data)

→ Operand is directly provided as Constant

→ No computation required to Calculate Effective Address.

Ex Add R1, #3

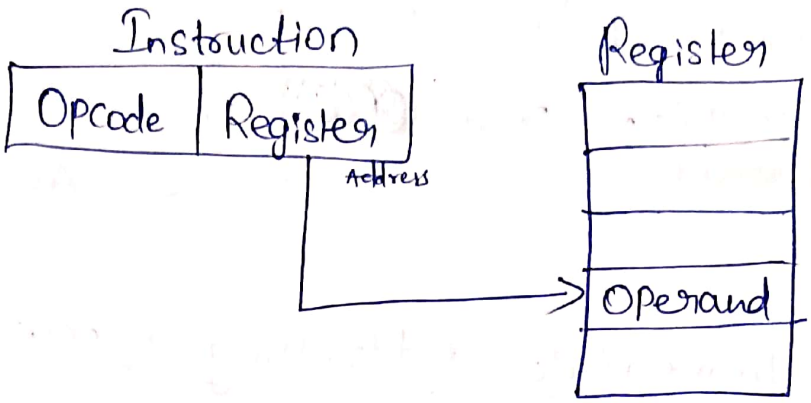
R1 ← R1+3



③ Register Addressing Mode

In this mode the Operands are in Registers that Reside within the CPU

Ex: MOV A,B



→ Effective Address (EA) = R

→ Advantage: No Memory Reference

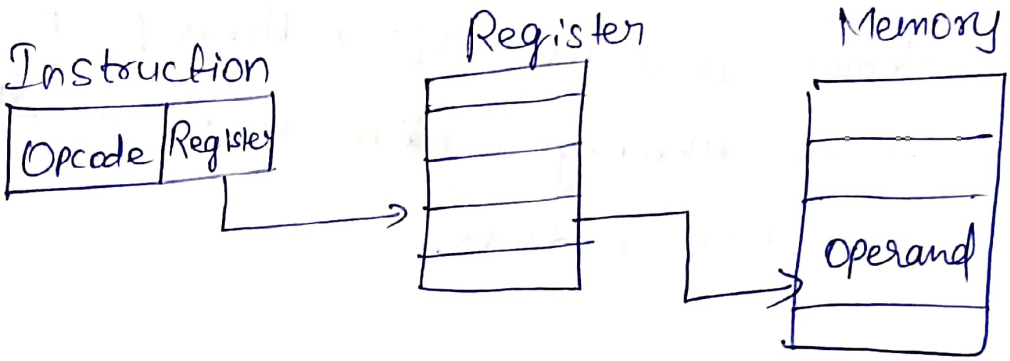
→ DisAdvantage: limited address space.

④ Auto - Increment (or) Auto decrement

→ When the Address Stored in the register refers to a table of data in Memory, it is necessary to increment (or) decrement the register after Every Access of the table.

(5) Register Indirect Addressing Mode

* In this mode the instruction specifies a register in the CPU whose contents give the Address of the Operand in the memory



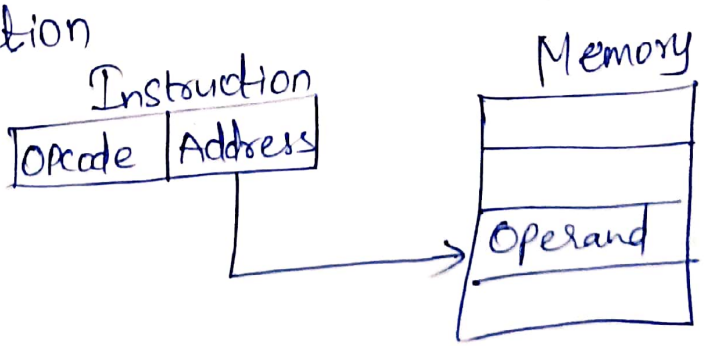
→ Effective Address = (EA) = R

→ Advantage : Large Address space

→ disadvantage : Extra memory reference

(6) Direct Addressing Mode

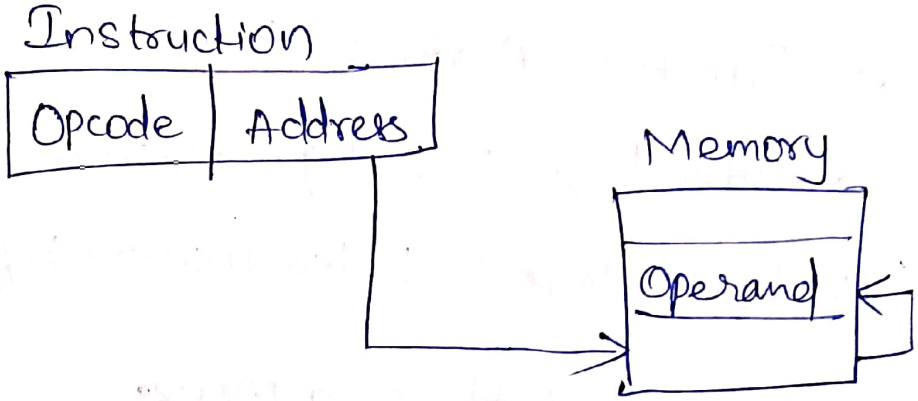
In this Mode the Effective address is Equal to the address part of the instruction The Operand resides in memory and its address is given directly by the address field of the instruction



(7) Indirect Addressing Mode

* In this mode the address field of the instruction gives the Address where the Effective address is stored in memory.

* Control fetches the instruction from Memory and uses its Address part to Access memory again to Read the Effective. Address.



Effective Address = Address part of Instruction

+

Content of CPU register

(6) Relative Addressing mode

→ In this mode the Content of the program Counter (PC) is added to the Address part of Instruction in order to obtain the Effective Address.

→ The Address part of the Instruction is usually a signed Number +ve (or) -ve

→ When the Number is Added to the Content of the program Counter, the Result produces an Effective Addr

* Whose position in memory is relative to the Address of the Next Instruction

Effective Address = address part of Instruction
+
Content of the PC.

(9) Indexed Addressing mode

Effective Address (EA) = Content of Indexed Register (XR) + Address part of the instruction

* In this mode the content of an index Register (XR) is added to the address part of the instruction to obtain the Effective Address.

* Index register is a special CPU register that contains an index value.

(10) Base Register Addressing mode

Same as Above

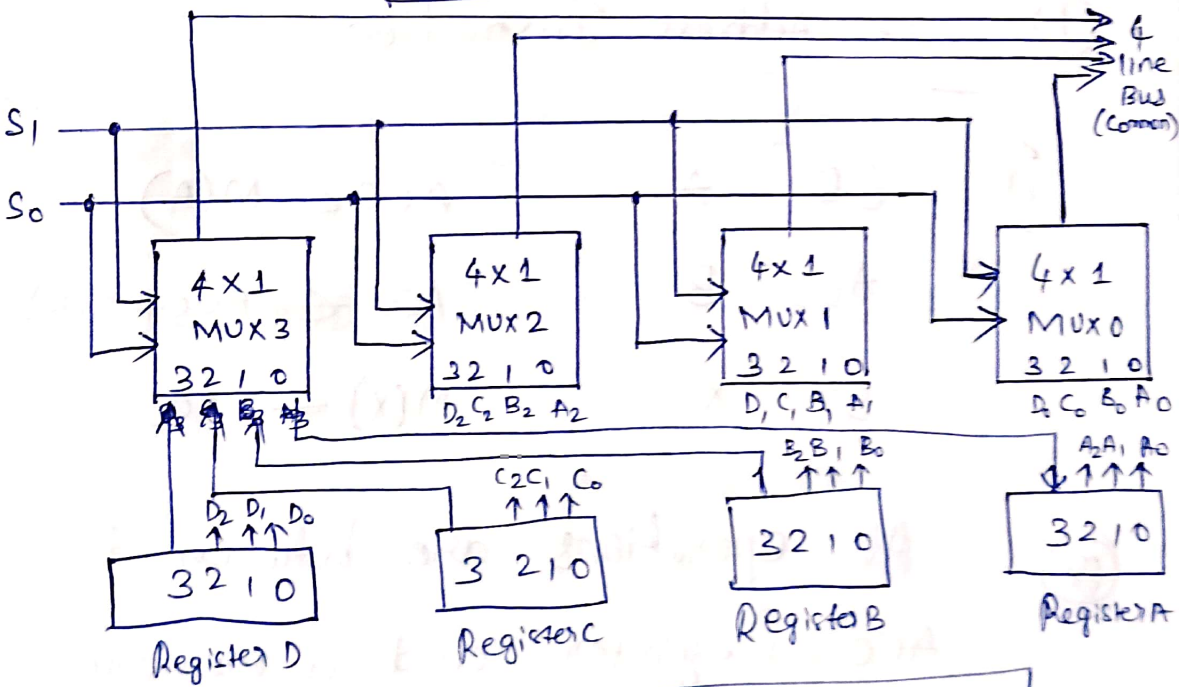
EA = Content of Base Register + Address Part of the Instruction

Basic Symbols for Register Transfers

Register Transfer Language (RTL)

Symbol	Description	Examples
1) Letters & Numbers	Denotes a Register's	MAR, R_2
2) Parenthesis ( )	Denotes part of Register's	$R_2(0-7)$, $R_2(L)$
3) Arrow ←	Denotes transfer of information	$R_2 \leftarrow R_1$
4) Comma ,	Separates two micro operations	$R_2 \leftarrow R_1, R_1 \leftarrow R_2$

Bus System for Four Registers



Function Table for Bus

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

Instruction field

Instruction set formats

The most common fields are

1) Operation field which specifies the operation to be performed like addition.

2) Address field which Contain the location of Operand (i.e) Register (or) Memory location

3) mode field which specifies how operand is to be founded.

(1) One Address Instruction

Ex

LD	A	ACC ← M(A)
ADD	B	ACC ← ACC + M(B)
ST	X	M[X] ← ACC

(or) All operations are between the ACC Register and a memory Operand

ACC → Short term (or) Intermediate Storage of arithmatic and logic data

② Two Address Instructions:

Ex

MOVE T_1, A	$M[T1] \leftarrow M[A]$
ADD T_1, B	$M[T1] \leftarrow M[T1] + M[B]$
MOVE X, C	$M[X] \leftarrow M[C]$

③ Three Address Instruction:

Ex

ADD T_1, A, B	$M[T1] \leftarrow m[A] + m[B]$
ADD T_2, C, D	$M[T_2] \leftarrow M[C] + M[D]$

④ Zero Address Instructions:

Ex

ADD	$TOS \leftarrow TOS + TOS_{-1}$
PUSH X	$TOS \leftarrow M[X]$
POP X	$M[X] \leftarrow TOS$