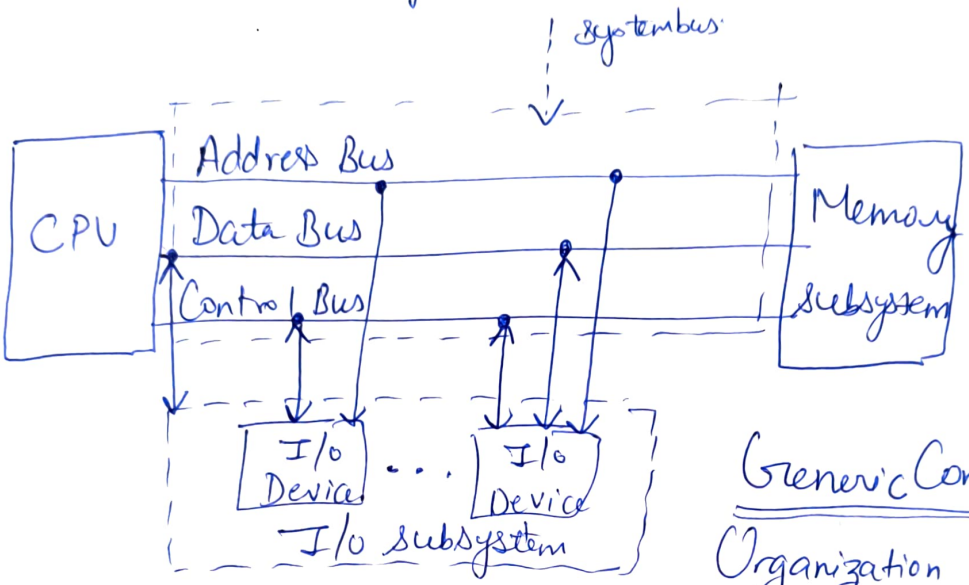
Basic Computer Organization

3 Main Components

1. CPU (Central processing unit)
2. Memory subsystem
3. I/O subsystem.



Generic Computer Organization

- System Bus : [Bus is a set of wires]
  - Address : when CPU reads data or instructions from memory or write data to memory, it must specify the address of the memory location it wishes to access.
  - Data Bus : Buses used to transfer the data from Memory to I/O devices or Vice-Versa.

⇒ I/O to Memory : 1. CPU places the address of the location (add. bus) where memory will store the data recieved from I/O.

2. I/O places the data on data bus, memory reads this data.

3. Memory places the data on Add. assigned by CPU

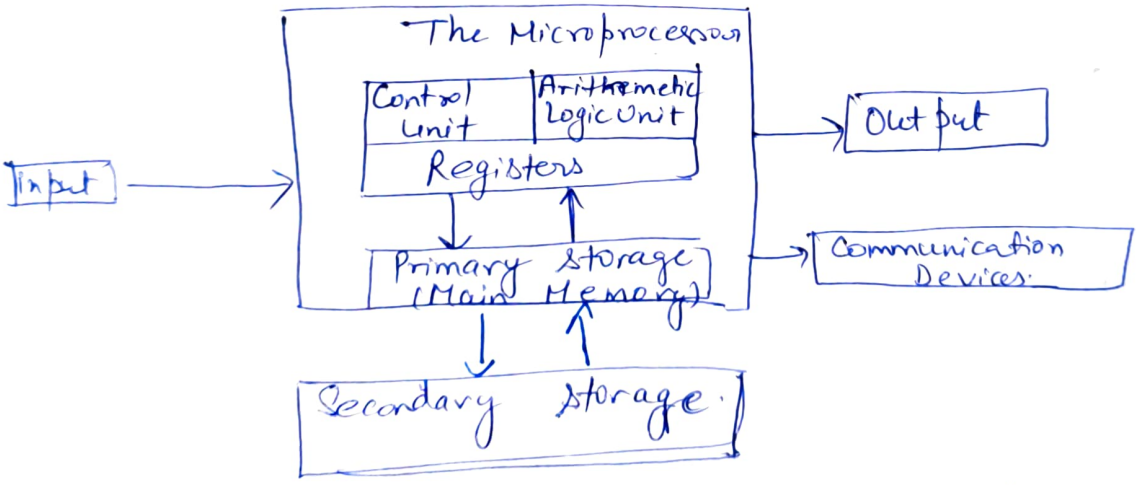
⇒ Memory to I/O: 1. CPU places the add. of the I/O device to which data is to send.

2. Memory places data on data bus. I/O device reads.

3. I/O could ~~store~~ the data, only if CPU issue ~~read~~ write signal to I/O device

- Control bus: Set of signals to control
  - → Memory Read / write
  - → I/O Read / write

- CPU Organization



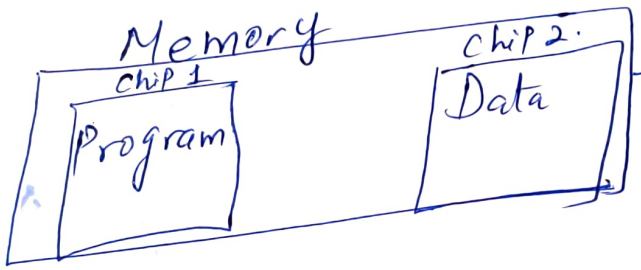
- Arithmetic Logic Unit (ALU) Performs Arithmetic and logical functions in ~~AA~~ microprocessor (μP).
- Registers: Used by μP to perform ALU operations. Some registers are not accessible by programmer.

~~Stored Program Concept :~~

Harvard Vs Von-Neumann Architecture.

Harvard : The Harvard Architecture is a term used for a computer using two separate areas for commands (program/instructions) and data.

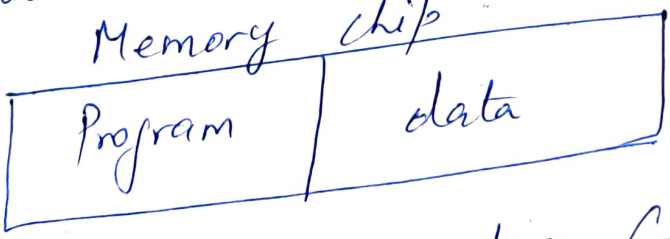
- CPU can do two task simultaneously.
  1. fetch the instruction from program section
  2. Load or store data in data section.



→ Harvard Architecture.

- Von-Neumann Architecture :

Architecture for a stored program computer where same memory chip is used to have program and data section on shared basis.



- Being used by all modern Computers.

Difference

Harvard	Von-Neumann
Separate buses for data and program	Program and data are stored in the same memo
Both program(or code) and data sections can be handled simultaneously due by CPU due to different hardware	CPU can handle only 1 section, program or data due to single hardware is present to access these secti
An instruction can be completed in minimum 1 timing cycle	An instruction requires minimum 2 cycles to complet 1. 1 cycle for fetching & decod 2. To execute
High cost due to complex Architecture	Cost is lower
Primarily used in microcontrollers and digital signal processing (DSP)	Used in every machine from desktop computers, laptops, notebooks, high performance computers and ~~workstations.~~ workstations.

Instruction Code:

Computer: Work acc. to instruction given by users

Program: Set of Instructions.

Memory: holds data and ~~address~~ Program for a μP.

Peripheral: for users: To input data [keyboard, Mouse, touch-Pad].
: To see results (Monitor, Printer)

- Assembly to Machine Language Conversion:

Machine Language is basically binary code Language, where every information is provided to the μP in binary code.

How instructions in Assembly Languages are converted into Codes?

An Instruction in Assem. Language is divided into two parts viz.

1. Mnemonic / Operation Code / Opcode
2. Operands.

1. Mnemonic / opcode: Tells the μp about type of ~~Operations~~ operations. e.g.

ADD : Addition
SUB : Subtraction
MUL : Multiplication.

it is always given in the form of binary code to the μP.

* Binary code: A code consists of 1's and 0's that can be presented in Hex numbers also.

e.g.

ADD: 1111, 1011 → F B H

F B

8-Bit code

- The number of bits in the opcode depends upon no. of operations performed by μP.

e.g. for a μP if 16 different operations are there than length or no. of bits in binary code is 4 as $[2^4 = 16]$.

As opcode specify the operation to be performed.
But on which this operation is to be performed?
Ans: Data

Where is the data?
in μP Registers
or
in memory

- Therefore, an Instruction Code must also specify register or memory word that holds data (or operands)

→ Memory word can be specified in instruction codes by their address.

→ Processor Registers can by specified using k-bit binary code for 2^k registers.

Stored Program Organization (Instruction-format)

The simplest way to organize a computer is to have.

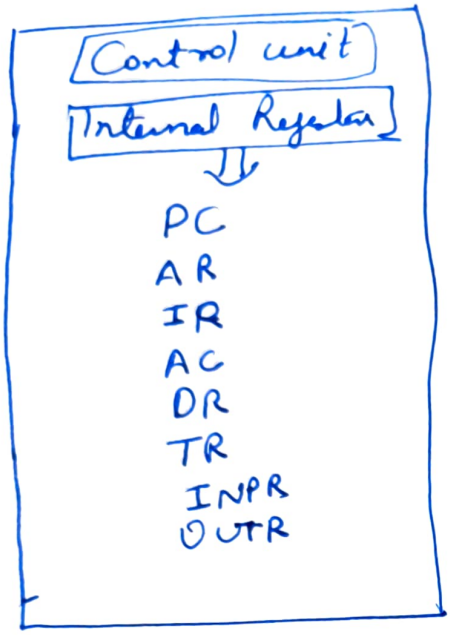
1. One processor Register
2. An instruction code with 2 parts.

first part : opcode
second part : an address : that specify operand in memory. This operand is ~~Sequence~~ ~~stored~~ read from memory and is used as the data, together with the data stored in processor Register.

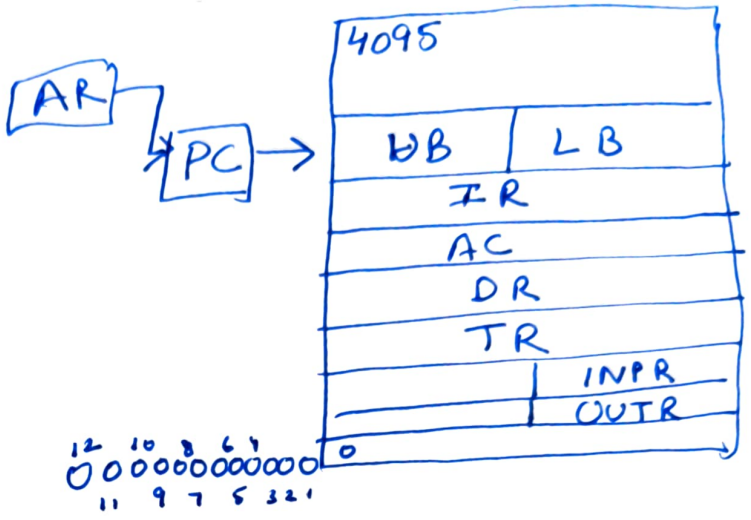
2^{12}

4096×16

CPU

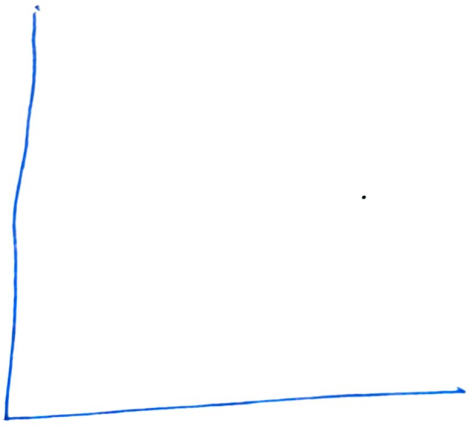


Memory chip/unit

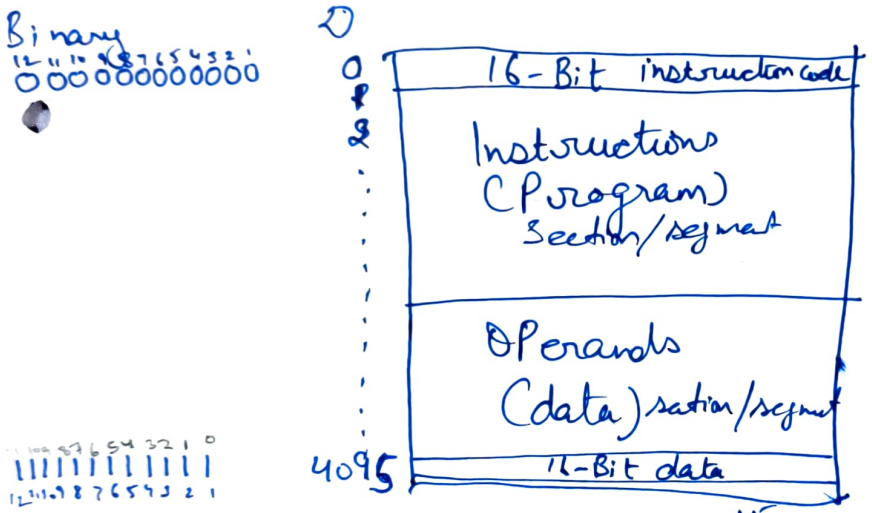


4096×16 → Memory chip = 4096 words

→ Each word length = 16 Bit



Operands : for simplifications, it is assumed operands are always stored in memory. In the computer instruction only add. of operand is given to µp. µP itself get to that particular add. of operand & find the data in data segment. (data segment ?).

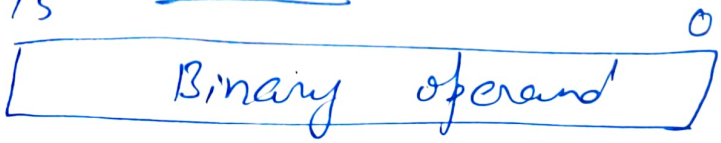


16-Bit instruction code

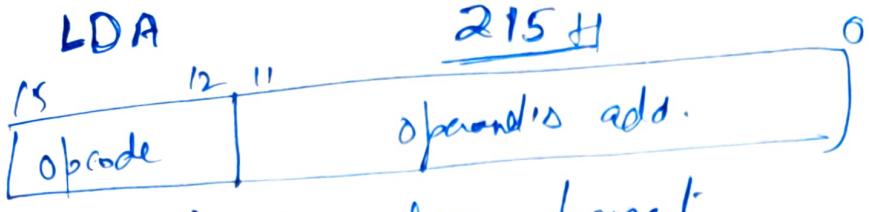
For 16 operations. It requires only 4-Bit i.e. bit 12, bit 13, bit 14, bit 15
Because $2^{4\text{-Bit}} = 16$ operations

Rest of the bits out of total 16 will be used to provide operand's add. i.e. bit 0 – bit 11 total 12-Bit. This implies operand will have 12-Bit add.

Note: Add. of the operand given in program segment is of 12-Bit. However operand itself lying in data segment is of 16-Bit

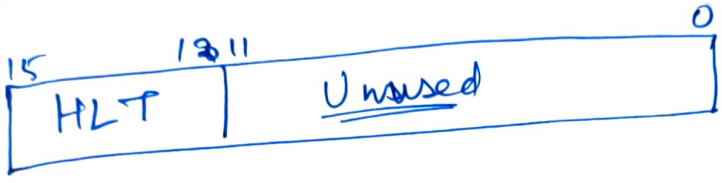


LDA 215H



instruction format

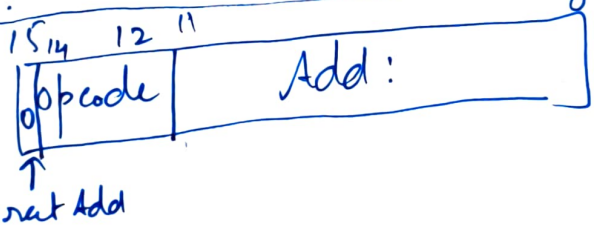
* There are certain instructions where operands are not required. like HLT [to terminate the program].



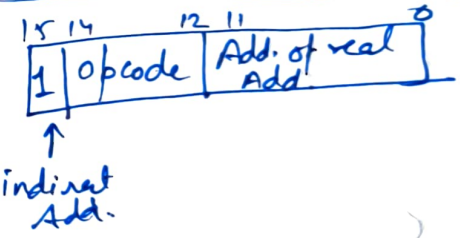
* Immediate Add:

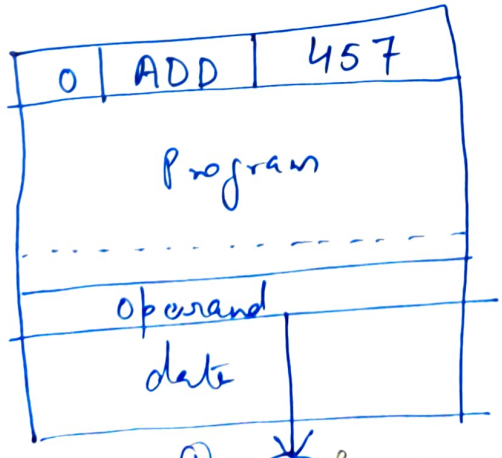
e.g. MVI A, 10 H

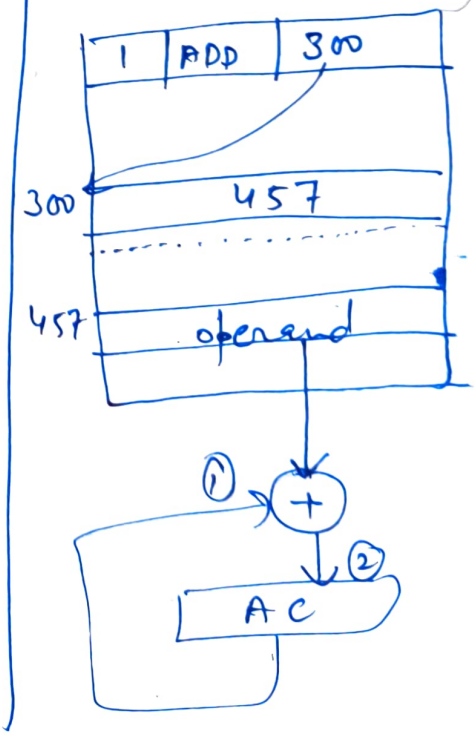
* direct Add:



* Indirect Add

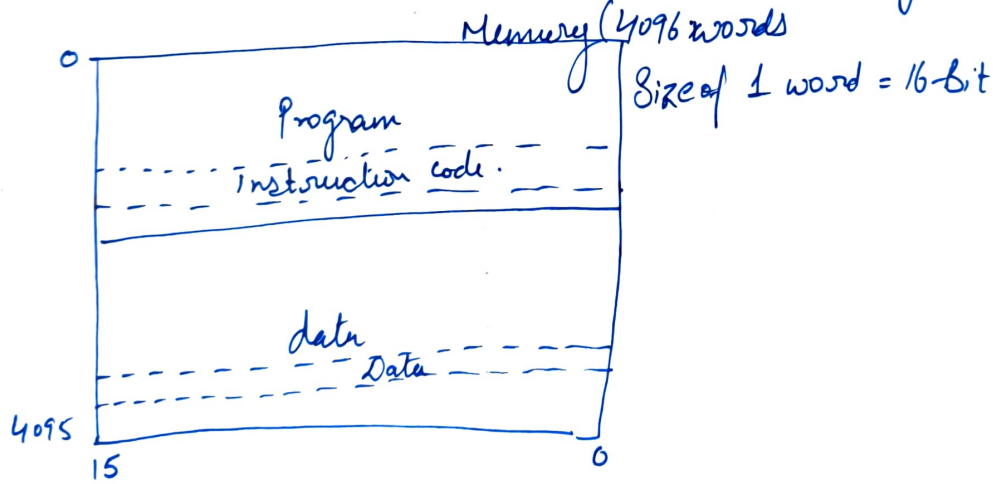




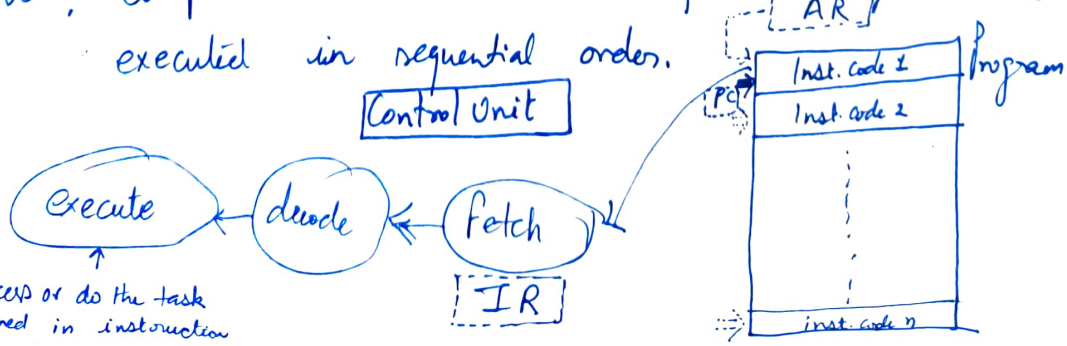


8 imputer Registers (DA_APIT_IO)

	Register Symbol.	Number of bits	Name
memory	DR	16	Data Reg.
	AR	12	Add. Reg.
µP	AC	16	Accumulator
instructions	IR	16	Instruction Reg.
	PC	12	Program Counter
	TR	16	Temporary Reg.
peripheral	INPR	8	Input Reg.
	OUTR	8	Output Reg.



* Program: Computer instructions in consecutive memory locations, executed in sequential order.



Note: Both AR and PC contains Address.

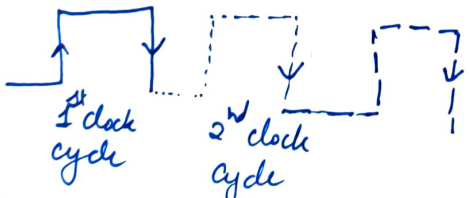
AR → works for memory; present add. of the instruction that is under execution.

PC → Works for program seq. track in memory. & holds add. of next instruction to be executed

Common Bus System [Not included]

1. Content of any Register can be placed/Applied onto the bus & an operation can be performed in Adder & logic unit during the same clock cycle.

2. Clock transition at the end of cycle → transfer the



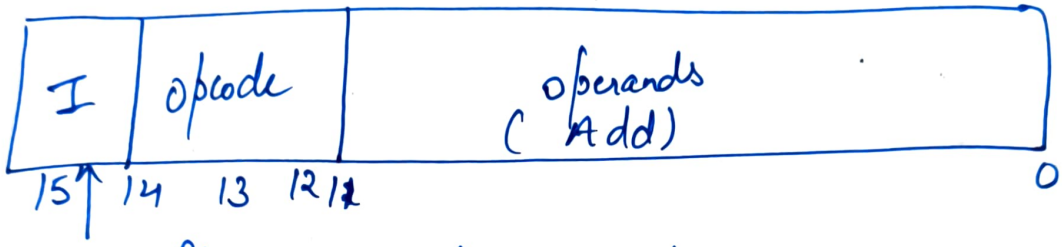
Content of the bus into the designated destination register and the output of the ALU into AC.

Basic Computer Instructions

Basic Computer Instruction format

1. Memory-Ref Instruction.

Instruction are hold by Instruction Register (IR) whose size for a basic computer (with memory 4096x16) is 16 bit i.e. 0-15



Bit 15: flag I : Values 0 or 1.

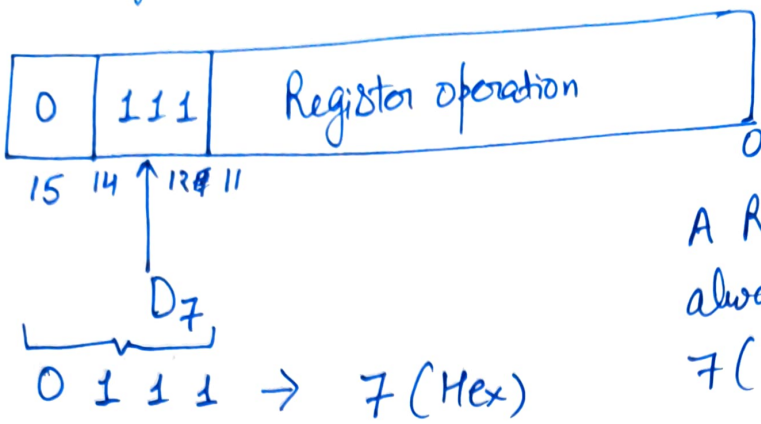
Bit 12-14: Opcode.

Bit 0-11: Add. of operand (I=0: direct ; I=1: Indirect)

I(1)	opcode(3)	Hex
0	000 (D_0)	0
0	001 (D_1)	1
0	010 (D_2)	2
0	011 (D_3)	3
0	100 (D_4)	4
0	101 (D_5)	5
0	110 (D_6)	6
0	111 (D_7)	7 → Reserved for Register & I/O Reference.
1	000 (D_0)	8
1	001 (D_1)	9
1	010 (D_2)	A
1	011 (D_3)	B
1	100 (D_4)	C
1	101 (D_5)	D
1	110 (D_6)	E
1	111	F → Reserved for Registers & I/O Reference

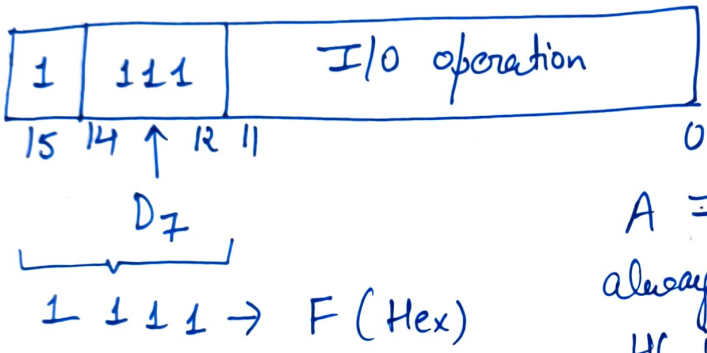
D_7 is reserved

2 Register - Reference Instructions.



A Register - Ref. instruction always starts with 7 (in Hex)

3 I/O - Reference Instructions



A I/O Reference Instruction always starts with H(in Hex).

■ Basic Computer Instructions

• Memory Ref.

Symbol	I=0	I=1	Description
AND	0	8	AND Memory Word to AC
ADD	1	9	ADD " " " "
LDA	2	A	Load AC from Memory
STA	3	B	Store " to "
BUN	4	C	Branch Unconditionally
BSA	5	D	Branch & Store Return Address
ISZ	6	E	Increment and Skip if Zero

Reference

Symbol	Hex Code	Description
CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Compliment AC
CME	7100	Compliment E
CIR	7080	Circulate Right AC & E
CIL	7040	Circulate left AC & E
INC	7020	Increment AC
SPA	7010	Skip if AC +ve (bit 15 of AC=0)
SNA	7008	Skip if AC -ve (bit 15 of AC=1)
SZE	7002	Skip if E is Zero
HLT	7001	Halt computer
SZA	7004	Skip if AC=0

I/O Reference

Symbol	Hex Code	Description
INP	F800	Input char. to AC
OUT	F400	Out " from AC
SKI	F200	Skip On input flag
SKO	F100	" " Output "
ION	F080	Interrupt ON
IOF	F040	Interrupt Off

Instruction Set Completeness

Set of instructions required by the user for designing, writing any machine Language program to evaluate any Computable function. For this given Instructions set must Contains following category of instructions.

- Instruction Type
  - Functional Instructions.
    - Arithmetic, logic, and shif instructions
    - ADD, AND, CMA, CLA, CIR, CIL
  - Transfer Instructions
    - Data transfers b/w the main memory and processor Registers
    - LDA, STA
  - Control Instructions
    - Program sequencing and control
    - BUN, BSA, ISZ
  - Input / Outpid Instructions
    - Input and output
    - INP, OUT

1. Timing And Control

Timing: Timings of all registers in basic computers is controlled by clock pulses. These clock pulses are applied to all registers and flip-flops in the system and control unit. However, clock pulses cannot change state of Registers untill & unless Control Signal is Enabled.

Control Signal: These are generated in control unit and provide Control inputs for Multiplexers in Common Bus, & Control signals (LD, CLR) in processor Registers.

There are 2 major types of Control Organization.

1. Hardwired Control Unit
2. Microprogrammed Control Unit.

Hardwired CU	Microprogrammed CU
⇒ Hardware based Control	⇒ Software based Control
⇒ Implemented with logic gates, decoders and other digital circuits	⇒ Implemented with microprogram stored in Control memory.
⇒ Can be easily designed	⇒ Complex in designed
⇒ Complex to modify	⇒ Easy to modify

Hardwired Control unit of Basic Computer

Hardware Required.

1. 3x8 decoder
2. 4x16 decoder
3. 4-Bit Sequence Counter (SC)
4. Combinational Control Logic
5. 1-bit flag (flip-flop)

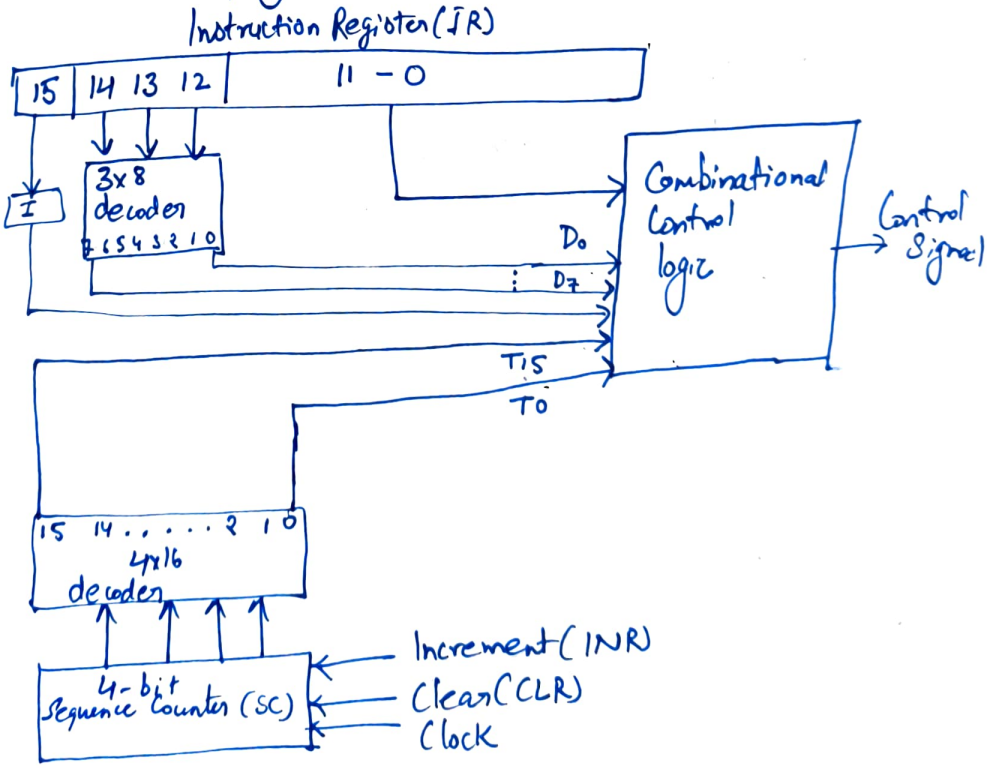
Why these Components ? See next in diagram

An Instruction Register Contains Instruction Code of 16-Bit.

0-11 field contains data information [Component 4 works].

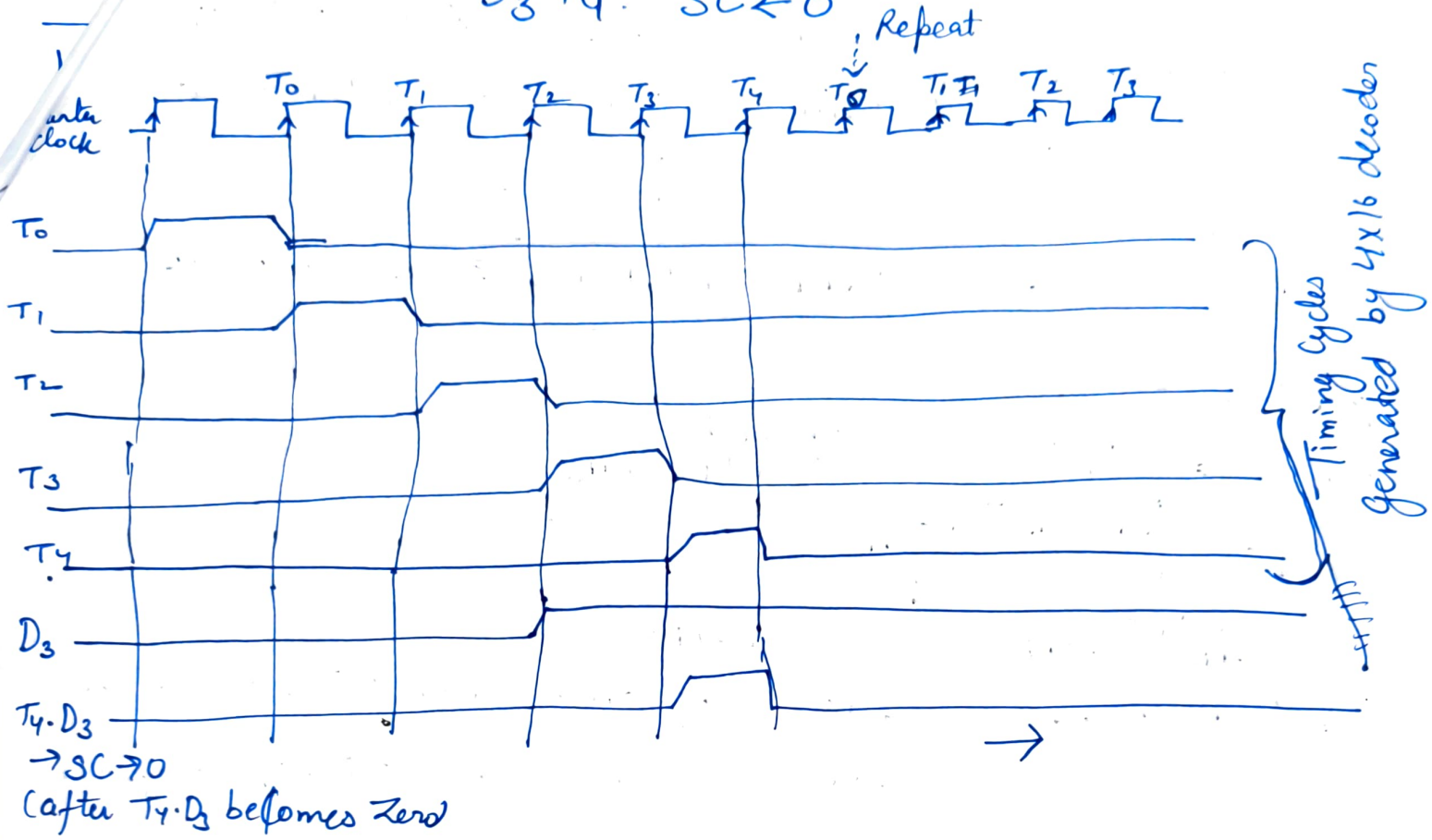
12-14 field Contains opcode [3x8 decoder works].

15 field contains flag bit [flip-flop required].



Timing Signals

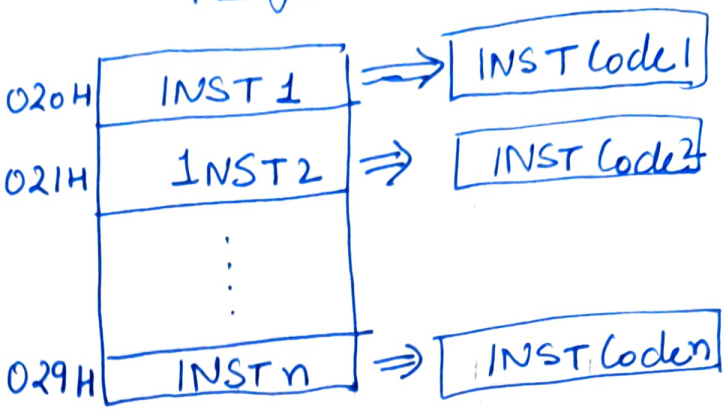
$D_3 T_4$: SC ← 0



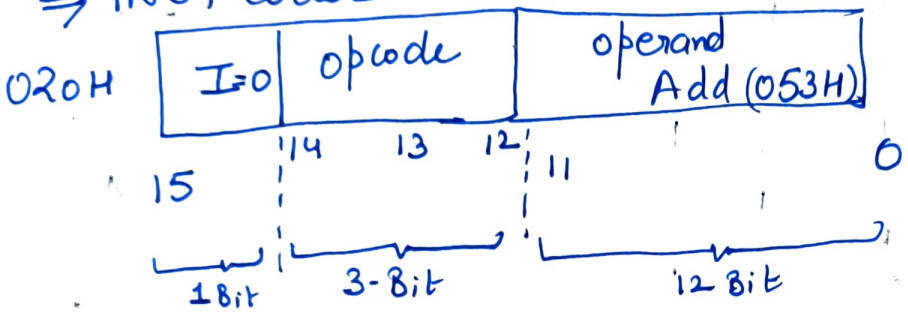
→ SC → 0
(after $T_4 \cdot D_3$ becomes Zero)

Memory-chip
4096 x 16

Program Section



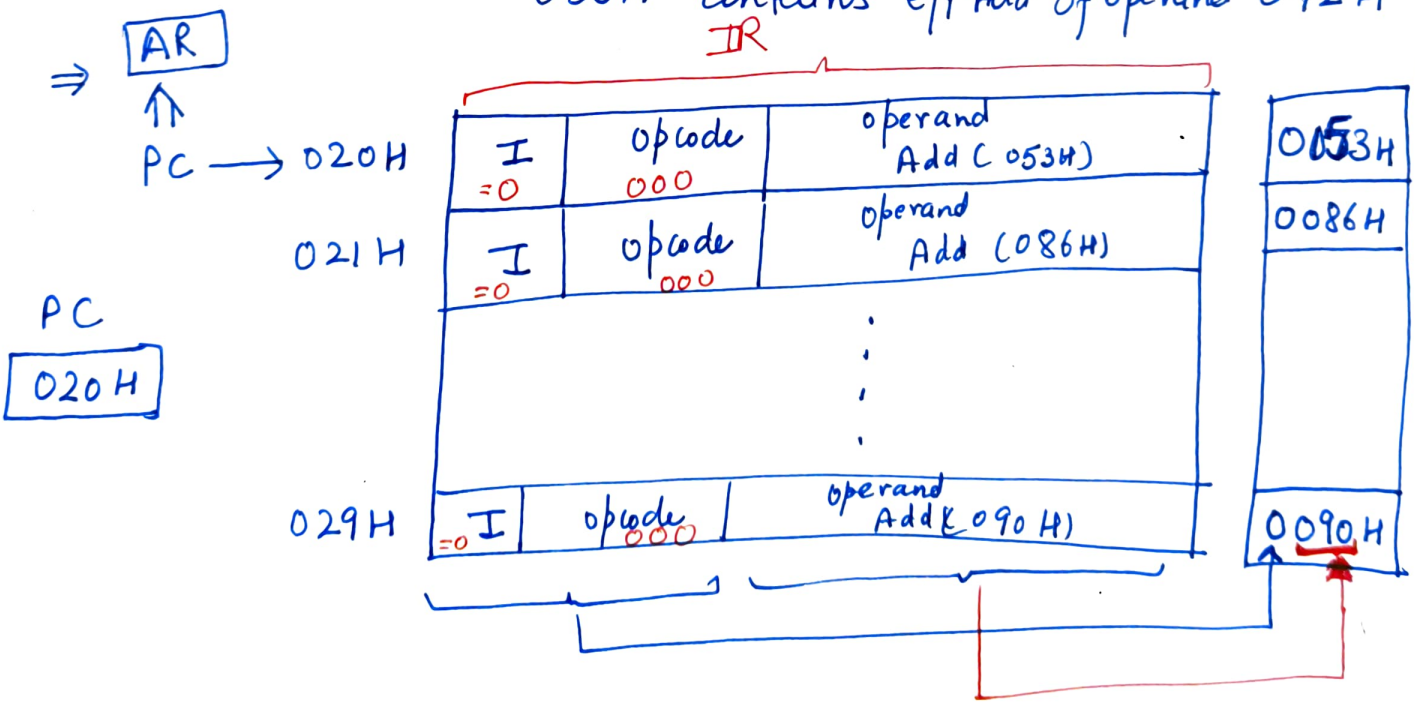
⇒ INST Code 1



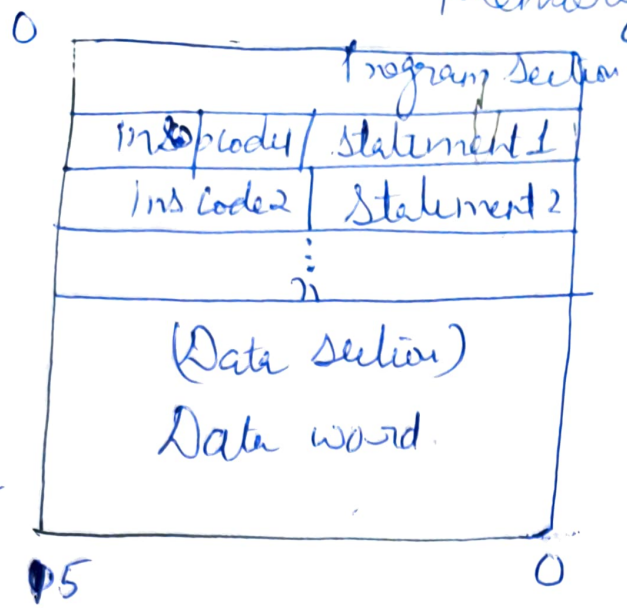
INST Code 1 (I = 0 operand Add (053H = eff. Add)

I = 1 " " (053H = Not eff. Add)

053H Contains eff. Add of operand 072H



Memory



Instruction Cycle

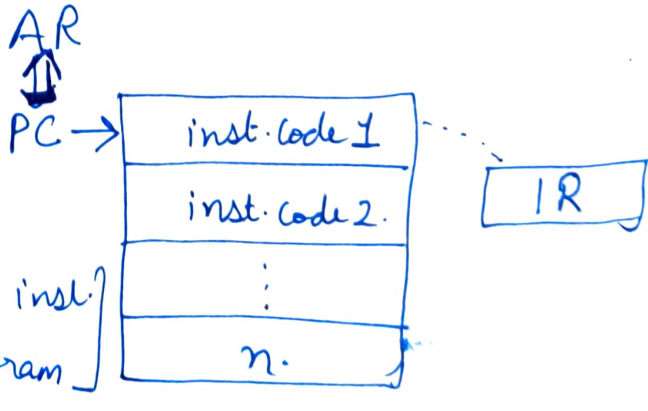
1. Fetch the instruction Code
2. Decode the opcode part.
3. Find the data word from the effective add. in memory ~~if inst. has indirect data~~
4. Execute

✓ Fetch phase [T_0]

✓ PC ⇨ AR

[Shifted the PC to keep track of the program sequence]

[Add. of Ist. inst. of the program]



1. Place the contents of PC on the common bus by selecting PC on MUX.
2. Activate the load control signal of AR. So that it can recieve the PC content available on bus.

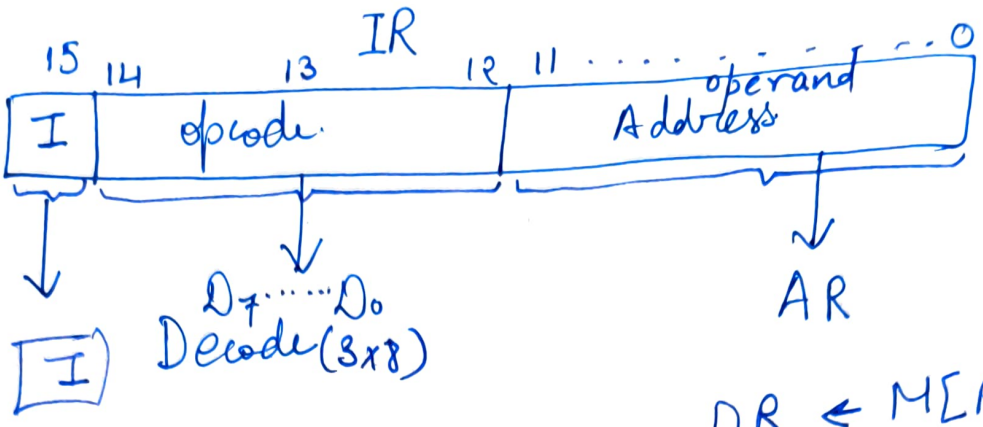
Fetch Phase (T_1)

~~Decode Phase~~

$IR \leftarrow M[AR], \quad PC \leftarrow PC+1$

1. Enable the read input of memory
2. ~~Select the~~ Place the contents of memory ~~so that~~ by selecting it. i.e. $S_2S_1S_0 = 111$
3. Transfer the contents of the bus to IR by enabling its LD input
4. Increment PC by 1 by enabling its INR input.

Decode Phase (T2)

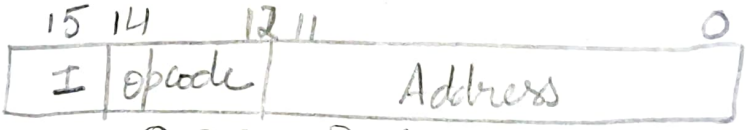


$DR \leftarrow M[AR]$

$T_2: D_0 \ldots D_7 \leftarrow$ Decode IR(12-14), $AR \leftarrow IR(0\text{-}11)$, $I \leftarrow IR(15)$

Basic Computer Instruction format

Memory-Reference Instruction (op-code = 000~110)



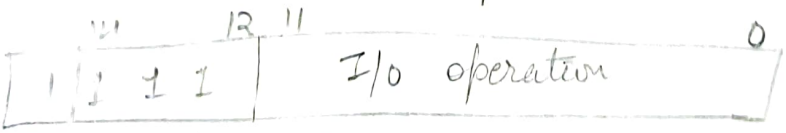
000 – D_0
⋮
110 – D_6

D_7 = Not active

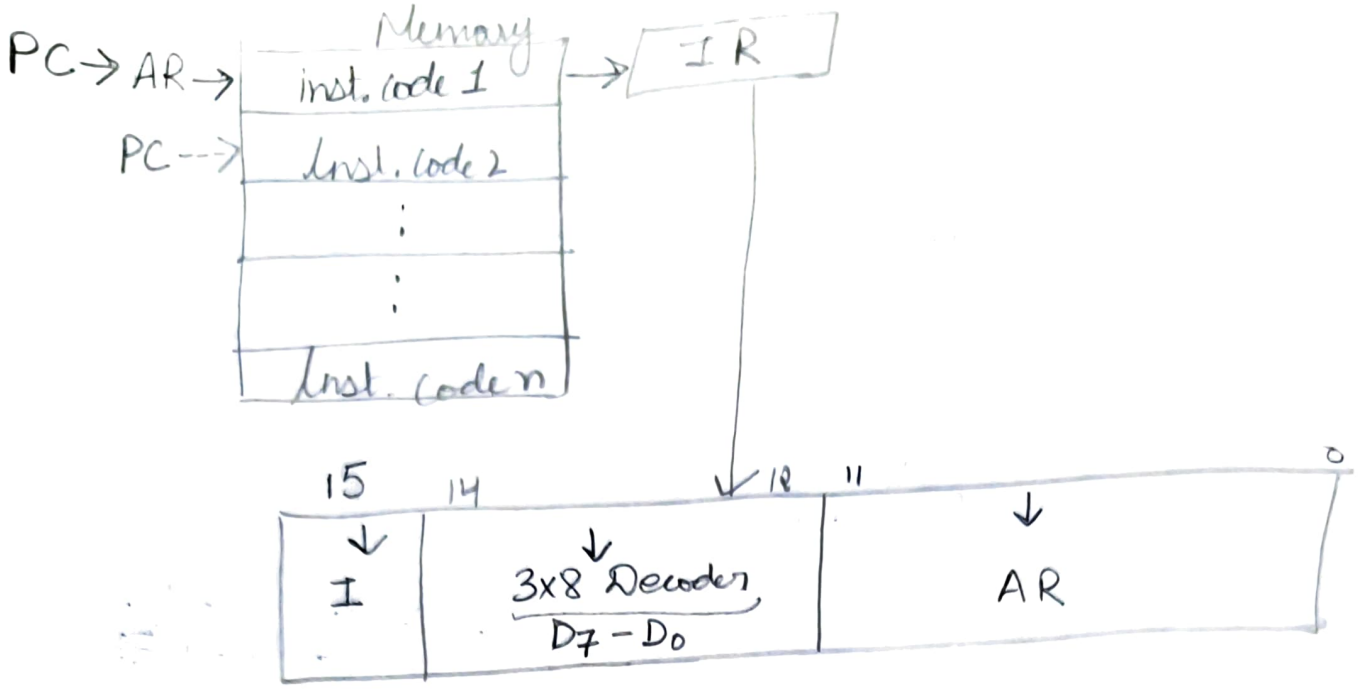
Register-Reference (op-code = 111, I = 0)

I=0 1 1 1 → D_7

Input-Output Instructions (op-code = 111, I = 1)

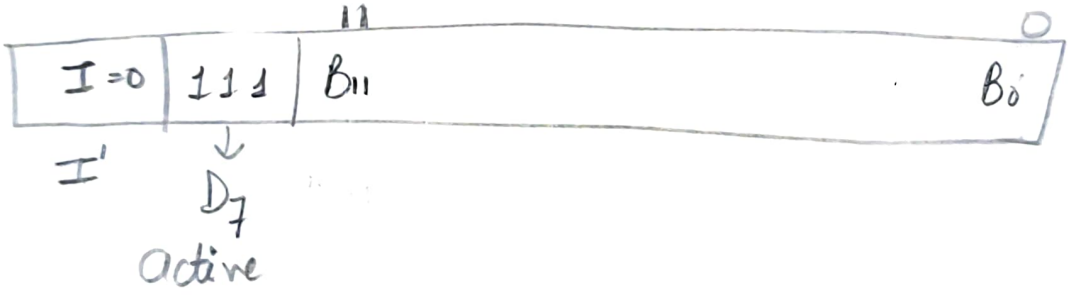


I=1 1 1 1 → D_7



Register Reference Instructions

$D_7 = 1$, $I = 0$



$r = D_7 I' T_3 =$ Register Reference Instruction

⇒	CLA	rB_{11}	$AC \leftarrow 0$ } clear
⇒	CLE	rB_{10}	$E \leftarrow 0$ } clear
⇒	CMA	rB_9	$AC \leftarrow \overline{AC}$ } compliment
⇒	CME	rB_8	$E \leftarrow \overline{E}$ } compliment
⇒	INC		$AC \leftarrow AC + 1$ } increment
⇒	HLT		$S \leftarrow 0$ (Start-stop FF)

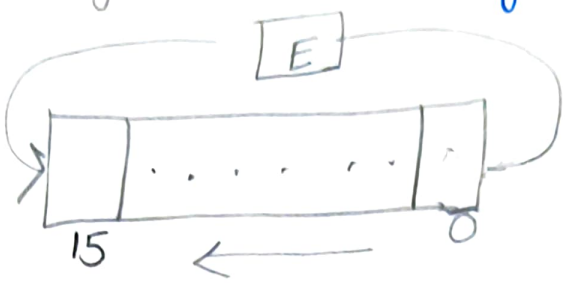
(in these instructions, AC and extented flag/bit is considered in Basic computer.

Int

Int CIR (Circulate Right)

e.g. (10) 1010 (8) 1000
(5) 0101 (4) 0100

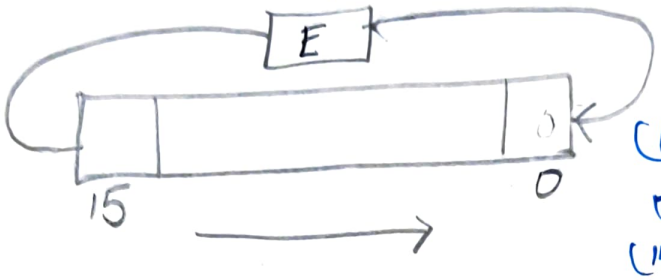
Division



⇒ CIL (Circulate left)

Multiply

(5) 0101 (6) 0110
(10) 1010 (12) 1100
(7) 0111
(14) 1110



⇒ Statement 1

Skip : Statement 2 ← PC

Statement 3 ← PC+1

SPA : skip if AC(15) = 0 ; PC ← PC+1

[skip if +ve]

SNA : if AC(15) = 1

SZA : if AC = 0

SZE : if E = 0