COMPUTER ORGANISATION:-

Introduction: A computer is defined as a fast electronic calculating machine that accepts the (data) digitalized input information, process it as per the list of internally stored instructions and produces the resulting information.

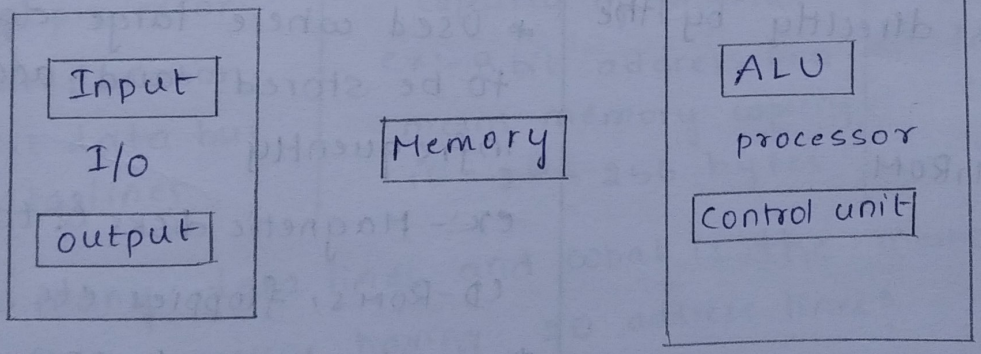
- List of instructions are called as programs and internal storage is called Computer memory.

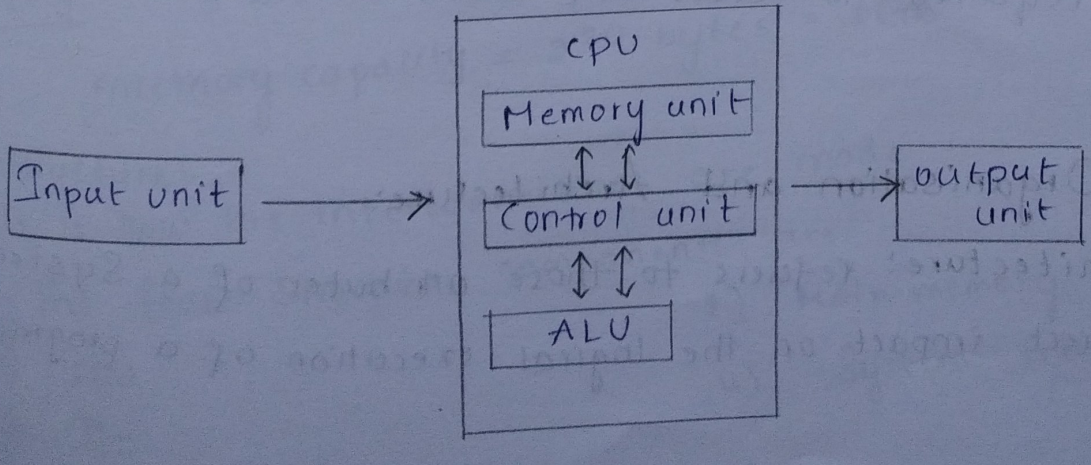
Different types of computer:-

1) personal
2) Note book – pc, Notepad etc.
3) Work stations
4) Enterprise systems
5) Super computers – for large calculations etc.

Functional units:-

5 functionally independent main parts – input, memory, ALU, output and control unit.





Central processing Unit (CPU):- Brain of the computer

3 components — ALU, control unit, Memory Unit.

Arthemetic Logic Unit (ALU):- Most of the computer operations are executed in ALU of processor like addition, subtraction, division, multiplication, comparing etc.

→ The operands are brought into the ALU from memory and stored in high speed storage elements called registers.

→ ALU — 64 bit → 64 bit processor

→ performs operations and result is stored in ALU itself & when an instruction is given, the result gets an advice to store or display etc.

Control unit: this controls the operations of all parts of Computer but doesn't carry out actual data processing operations.

- program is set of instructions.

Primary memory	Secondary memory
* Accessed directly by the CPU ex:- RAM, ROM	* Used where large data have to be stored and accessed infrequently ex:- Magnetic disks & tapes, CD-ROM's, floppies etc * it is external to CPU.

→ The time required to access one word is called memory access time.

Organisation and Architecture:-

Computer Architecture: referers to those attributes of a System that have direct impact on the logical execution of a program.

ex:- • instruction set
• no. of bits used to represent various data types.
• I/O mechanisms.
• memory addressing techniques.

Computer organization: refers to the operational units and their interconnections that realize the architectural specifications.

ex:- • control signals
• Interfaces blw computer and peripherals.
• The memory technology being used.

Structure and Function:
* structure is the way in which components relate to each other.
* function is the operation of the individual components as a part of structure.

Computer functions are: Data processing, data storage; data movement and control.

System bus:

1) Data bus	2) Address bus	3) Control bus.
Defined as the bit length of processor ex:- 8 bit processor have 8 bit data bus and 8 data lines	Defines max memory capacity of processor ex:- 8 bit address lines means memory capacity is $= 2^8 = 256$ bytes	physical connections that carry control information blw cpu & other devices with the computer

ex:- How many data lines and what is the memory capacity of a 16-bit processor having 20 address lines?

Ans:- 16 bit → 16 data lines

memory capacity $= 2^{20}$ bytes $=$ 1GB

Structure:-
It is how the interconnections are made.

The four structural components are.

1) cpu
2) Main memory
3) I/O
4) System interconnections.

CPU structural components:-

1) Control Unit 2) ALU 3) Registers

4) Internal CPU interconnections

* Registers store temporary data & some address. for every processor, register set will be different.

The structural components of CU (Control Unit).

1) Sequencing Logic → logic signals - read operation, every signal
2) Control unit registers & decoders
3) Control memory → to store the signals - we write a logic that is stored here.

Input/output Subsystems:-

controlls all I/O devices

Basic fxns include:

- Issuing commands to devices
- Handling interrupts from devices
- Responding errors from devices.

Temporary storage of code and results are needed.

- Main memory (RAM)

Register:-

It is a very small amount of very fast memory i.e built into the CPU.

* they store data temporarily during execution of a program
* Registers are normally measured by no. of bits they can hold.

ex:- 8-bit registers means it can store 8 bits of data.
32-bit registers means it can store 32 bits of data

* Since they store data temporarily, later we can move the data to another location to avoid loss of data
* combination of flip flops is register
* IFF can hold only 1 bit

Computer Registors:-

- **Accumulator (AC):** The processor register AC consists of 16-bits. here intermediate arthemetic & logical signals are stored.
- **Data register (DR):** The data register DR Consists of 16-bits and is used to hold memory operands (data)
  - → Register writing into memory ⇒ Write operation.
  - → Register reading from memory ⇒ Read operation.
- **Temporary register (TR):** TR have 16-bits and it provides tempo-rary storage of variables & results.
- **Instruction register (IR):** IR consists of 16-bits. It holds a copy of Instruction which processor is to execute. IR holds instruction code which is read from memory.
- **Address register (AR):** It specifies the address in memory for next read or write operation. The AR consists of 12-bits.
- **Program counter (PC):** has 12 bits and holds the address of next instruction to be read from memory after the current execution is executed.
- **Input Register (INPR):** has 8-bits. INPR recieves a character from an input device and delivers it to the AC.
- **output Register (OUTPR):** has 8-bits. It recives information from Ac and transfers it to o/p device.

ex:- ADD (AC → destination), (DR → source)

- AC = AC + DR → this instruction adds the contents of Ac and DR and stores the result in accumulator.

AC = 1234 H
DR = 2222 H
After execution AC = 3456H

Instruction:
A computer instruction is a binary code that Specify a Sequence of micro operations for the computer.

An Instruction set:- Is a group of bits that instruct Computer to perform a specific operation.

The basic computer have 3 instruction code formats, each of 16-bit.

There are 3 types of instruction.

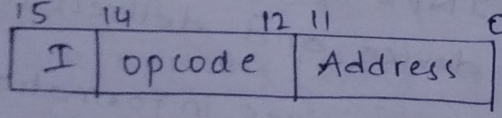
① Memory reference instruction: Instruction that has 1 or more of its operand address referring to a location in memory.

② Register reference instruction: Specifies an operation on or a test of the AC register.

An operand from memory is not needed, so other 12-bits are used to specify operation or test to be executed.

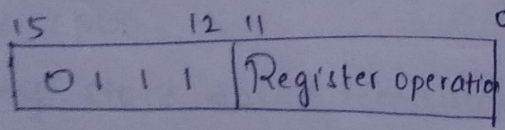
③ Input-output Instruction: Doesn't require a refererence to memory and it is recognised by opcode 111 & I=1

Various Instruction format:



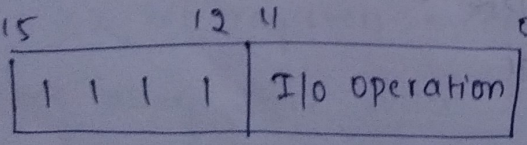
(opcode = 000 to 110)

a) Memory-reference Instruction (I=0 direct addressing mode, I=1 indirect addressing mode)



(opcode = 111, I = 0)

b) Register-reference Instruction



(opcode = 111, I = 1)

c) Input-output Instruction

Instruction cycle:-

- The time period during which one instruction is fetched from memory and execute when a computer gives an instruction in machine language.

phases of Instruction cycle:-

1. Fetch the instruction — (from memory)
2. Decode the instruction — from our readable lang to machine language
3. Read the effective Address from memory if the instruction has an indirect address.
4. execute the instruction.

→ The program counter PC is loaded with address of the 1st instruction in the program.

→ The sequence counter SC is cleared to 0, providing a decoded timining signal T_0.

→ After each clock pulse, SC is incremented by 1, so that the timing signals go through a sequence T_0, T_1, T_2 and so on.

→ The micro operations from the fetch and decode phases can be specified by the following register transfer statements.

T_0 : AR ← PC

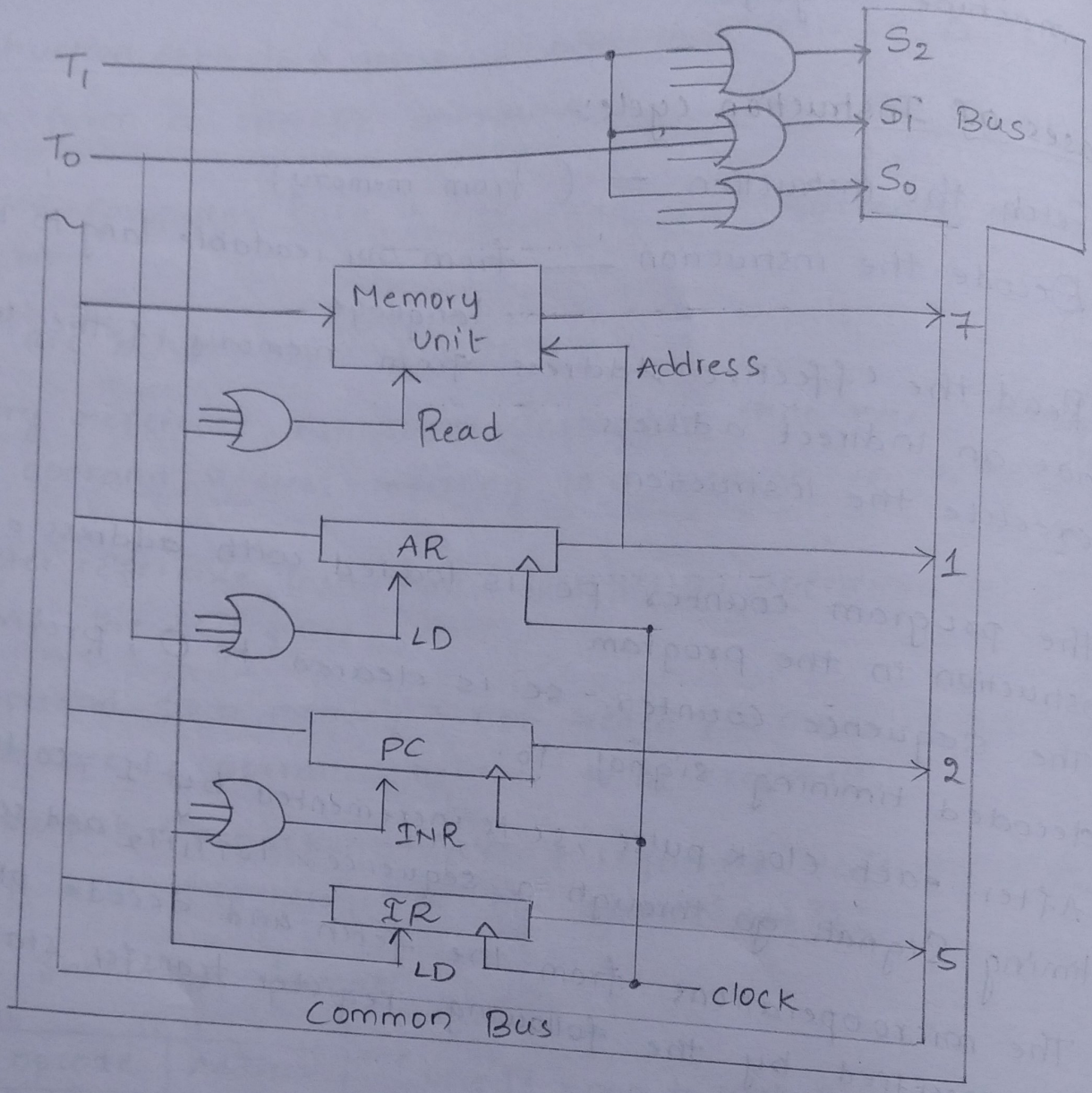
T_1 : IR ← M[AR], PC ← PC+1

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

Here IR ← M[AR] means the contents of AR are moved into IR.

* In memory-reference Instruction, in direct addressing mode, it will directly go to register having address and access its contents.
* In indirect addressing mode, the address is present in some other register. so it should first go to that register & read that address & go to that address then reads its contents.

Register transfer for fetch phase:-



From flow chart

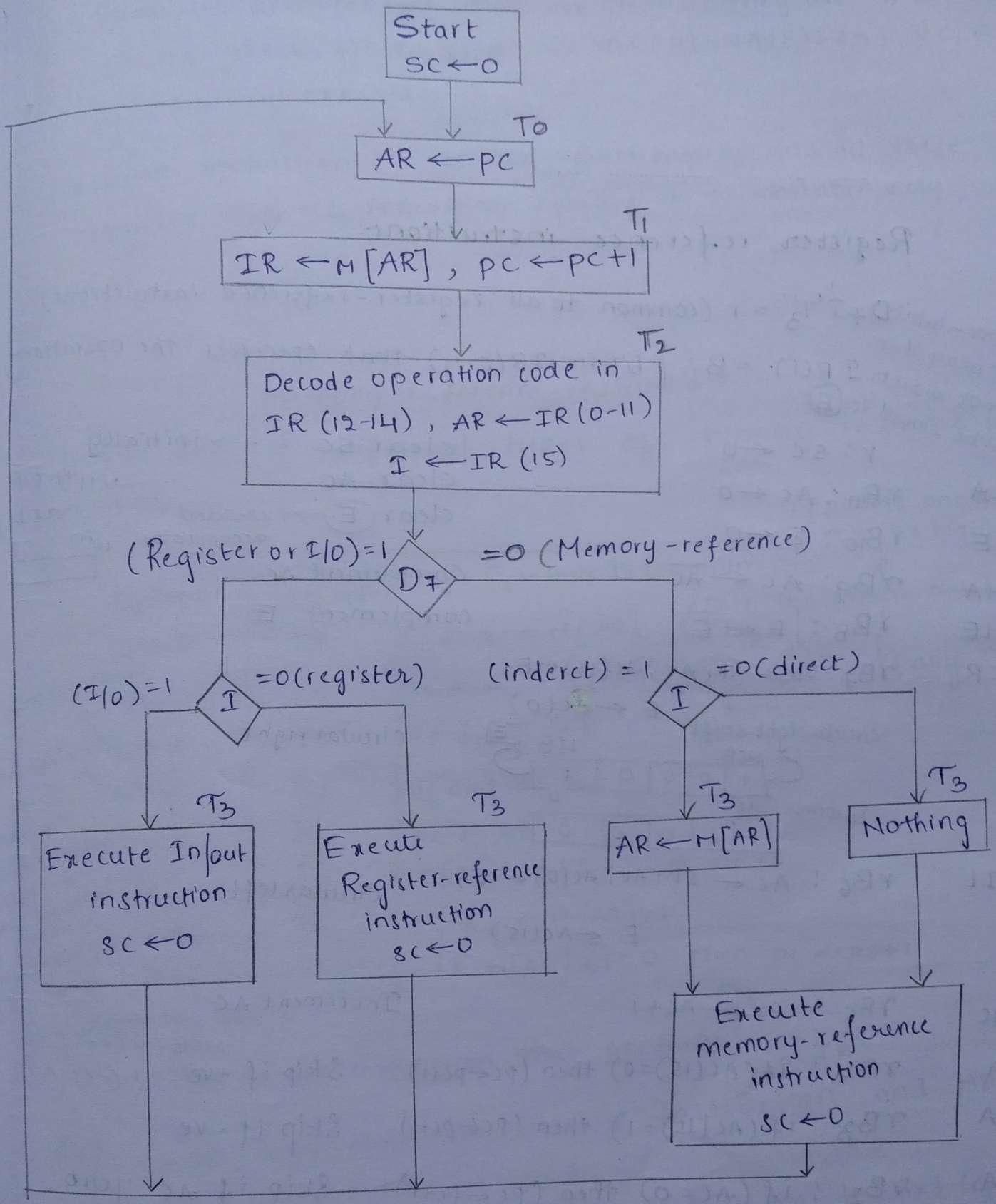
$D_7'IT_3$: AR ← M[AR] (indirect addressing mode)

$D_7'I'T_3$: Nothing (direct addressing mode)

$D_7I'T_3$: Execute a register-reference instruction

D_7IT_3 : Execute a I/O instruction

Flow chart for instruction cycle:-



Definition of Register reference Instructions:-

Register reference instructions:-

Accumulator - 16 bit $D_7 I' T_3 = r$ (common to all register-reference instructions)

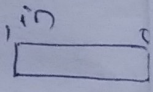
we get carry then 16 bit is not enough those will be stored in (E)

$IR(i) = B_i$ [bit in IR(0-11) that specifies the operation]

r : SC ← 0 — clear SC → initially

CLA rB_{11} : AC ← 0 — clear AC → 11th Bit as 1

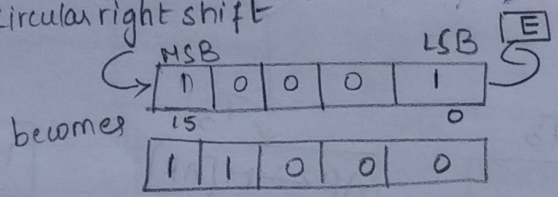
CLE rB_{10} : E ← 0 — clear (E) → extended accumulator



CMA rB_9 : AC ← $\overline{AC}$ — Complement AC

CME rB_8 : E ← $\overline{E}$ — complement E

CIR rB_7 : AC ← shr AC, AC(15) ← E, E ← AC(0) — circular right



CIL rB_6 : AC ← shl AC, AC(0) ← E, E ← AC(15) — Circular left

INC rB_5 : AC ← AC + 1 — Increment AC

SPA rB_4 : if (AC(15) = 0) then (PC ← PC + 1) — Skip if +ve (MSB = 0 (+ve no), = 1 (-ve no))

SNA rB_3 : if (AC(15) = 1) then (PC ← PC + 1) — Skip if -ve

SZA rB_2 : if (AC = 0) then (PC ← PC + 1) — Skip if AC zero

SZE rB_1 : if (E = 0) then (PC ← PC + 1) — Skip if E zero

HLT rB_0 : S ← 0 (S is a strat-stop flip flop) — Halting computer

- There are a total 12 register reference instructions in computer architecture that are in use they are 1) CLA, 2) CMA, 3) CIL, 4) CIR, 5) INC, 6) SPA, 7) SNA, 8) SZA, 9) CLE, 10) CME 11) SZE 12) HLT
- Each instruction has its own functionality and all these register reference instructions operate on accumulator only.

Memory reference instructions.

There are 7 different types of Memory reference instructions.

Symbol	operation decoder	Symbolic description.
AND	D0	AC ← AC ∧ M[AR] (Both the inputs must be high)
ADD	D1	AC ← AC + M[AR], E ← Cout → if any carry is present (atleast one input must be high)
LDA (Load accumulator)	D2	AC ← M[AR]
STA (Store accumulator)	D3	M[AR] ← AC
BUN (Branch unconditionally)	D4	PC ← AR
BSA (Branch & Save the return address)	D5	M[AR] ← PC, PC ← AR+1
ISZ (increment & Skip if 0)	D6	M[AR] ← M[AR]+1 if M[AR]+1 = 0 then PC ← PC+1

AND to AC:

This is an instruction that performs nxt page (IR) and ADD

Let AND instruction be current instruction be nxt instruction → (PC)

BSA → saves written instruction and Jumps to nxt instruction

↓ PC ← AR+1

Branching:- M[AR] ← PC

↳ to break the current flow and executing some another set of instructions where label is present

AND to AC.

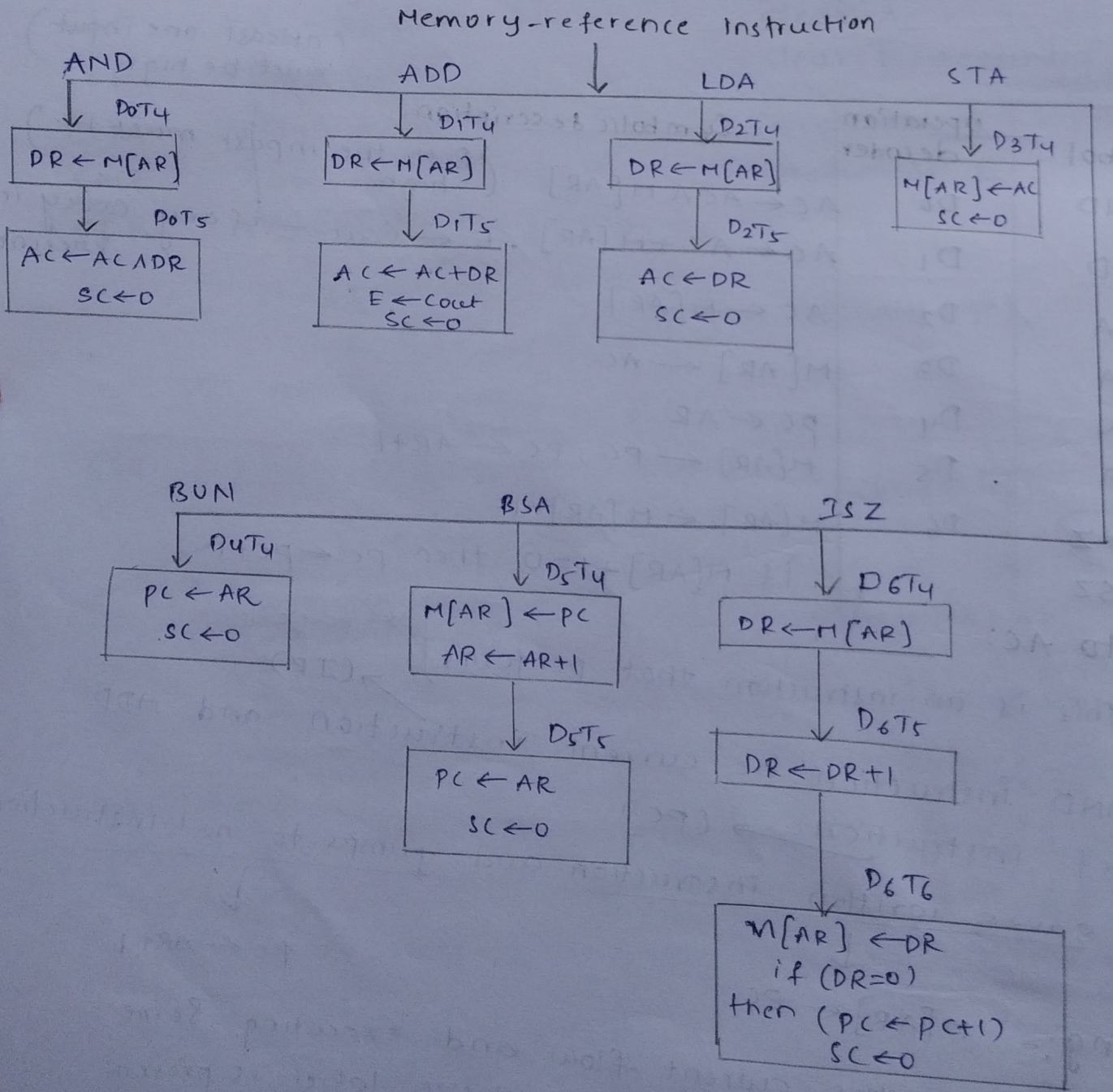
This is an instruction that performs the AND logic operation on pairs of bits in AC and the memory word specified by the effective address. The result of the operation is transferred to AC.

The microoperations involved are:

D_0T_4: $DR \leftarrow M[AR]$

D_0T_5: $AC \leftarrow AC \wedge DR \quad SC \leftarrow 0$

FLOW CHART FOR MEMORY REFERENCE INSTRUCTIONS:



Input and output Instructions:

$D_7 I T_3 = P$ (common to all input-output instructions)

$IR(i) = B_i$ [in IR (6-11) that specifies the instruction]

1 bit - which gives the status of the result
ex:- result is 0, -ve, +ve, odd, even etc.

$P: SC \leftarrow 0$ (clear SC)

INP	$PB_{11}: AC(0-7) \leftarrow INPR, FGI \leftarrow 0$ (I/p register, Input flag)	Input character
OUT	$PB_{10}: OUTR \leftarrow AC(0-7), FGO \leftarrow 0$ (O/p Register, Output flg)	output character
SKI	$PB_9: If(FGI=1)$ then $(PC \leftarrow PC+1)$	Skip on input flag
SKO	$PB_8: If(FGO=1)$ then $(PC \leftarrow PC+1)$	Skip on output flag
ION	$PB_7: IEN \leftarrow 1$	Interrupt enable on
IOF	$PB_6: IEN \leftarrow 0$	Interrup enable off

Addressing modes:-

→ The way of accessing the operands

→ The operation field of an ~~of~~ 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 choosen during program execution is dependent on the addressing

Types of addressing modes:-

1) Implied mode
2) Immediate mode
3) Register mode
4) Register Indirect mode
5) Autoincrement or Autodecrement mode
6) Direct Address mode
7) Indirect Address mode
8) Relative Address mode
9) Indexed Addressing mode
10) Base Register addressing mode

Implied addressing mode:

- In this mode the operands are specified implicitly in the definition of the instruction

Example:- all register reference instructions (CMA, CLA, CLE e means we are not referring any register (or) memory to access the operand.

Immediate mode:-

- operands are part of the instruction. in other words, an imediate-mode instruction has an operand filed rather than an address field.
- these instructions are useful for initilizing registers to a Const value.

Eg:- ADD A, 1234H

A = 2222H

- After execution A=3456H

Register Mode:-

- In this mode the operands are in registers that reside within the CPU.
- A particular register is selected from a register field in the instruction.

Eg:- Add A & (R) → data register

↓ ↓

1111H + 2222H

= 3333H

Register indirect mode:-

DR ← address

Contents ← address

Add ← Contents

- the selected registeer contains the address of the operand rather than the operand itself.
- Advantage is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.

Eg:- ADD A1 [AR] → indicates address.
- AR = 5000H
- 5000H = 1234H

(or)
Add A1 [5000H]
occupies less memory

Autoincrement or Autodecrement Mode:-
→ Similar to register indirect mode except that the register is incremented (or) decremented after (or before) its value is used to access memory

Ex:- MOVE A1 [AR] → 5000
INC AR → [5001]
LOOP dec AR → 4999

Direct Address Mode:-
- In this mode the effective adress = address part of the instruction. The operand resides in memory & its address is given directly by the address field of instruction.
- In a branch-type instruction the address field specifies the actual branch address

Eg:- ADD A1 [5000H]

→ eg:- BUN 2000H
nxt instruction address

Indirect Addressing Mode:
- Register → Address → Again Address → operand.
↓
effective address

Relative Address Mode:

- In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.
- either it can add address to the program counter (+ve) or it can subtract address to the PC to get the effective address whose position in memory is relative to the address of nxt instruction

Indexed Addressing Mode:-

- In this mode the content of an index register is added to the address part of the instruction to obtain effective address
- index register → Special CPU register → contains index value
- The address field of the instruction defines the beginning address of a data array in memory

Base Register Addressing Mode:-

- This is Similar to the indexed addressing mode except that the Register is now called the base register insted of an index register

Index register:
holds index number that is relative to the address part of the instruction.

Base Register:
holds a base address and the address field of the instruction

Numeric example for Addressing modes

Addressing Mode	Effective Address	Content of AC

Instruction set:

1) Data transfer instructions
2) Data manipulation instructions
3) program control Instructions

① → Data transfer instructions
operands are chainging from one location to another with out changing the binary information

② → Data manipulation instructions.
These instructions performs arthemetic, logic & shift operations.

Data transfer instructions:

Name	Mnemonic
Load	LD → AC will be the destination
Store	ST
Move	MOV → MOV AC R1 (R1 → source, AC → destination)
Exchange	XCH → XCH AC,R1
Input	IN
Output	OUT
Push	PUSH
POP	POP

Before
AC = 1234H, R1 = 300H
After
AC = 300H, R1 = 1234H ← Exchange

if R1 = 1234H
AC = 1234H
R1 = 1234H only still

Addressing modes for load instruction:

Mode	Assembly Convention	Register Transfer
Direct address	LD ADR	AC ← M[ADR]
Indirect address	LD @ADR	AC ← M[M[ADR]]
Relative address	LD $ADR	AC ← M[PC+ADR]
Immediate operand	LD #NBR	AC ← NBR
Index addressing	LD ADR(X)	AC ← M[ADR+XR]
Register	LD R1	AC ← R1
Register indirect	LD(R1)	AC ← M[R1]
Auto increment	LD(R1)+	AC ← M[R1], R1 ← R1+1

(2) → Data manipulation instructions performs operations on data & provide the computational capabilities of the computer.

i, Arithemetic instructions
ii, logical & bit manipulation instructions
iii, Shift instructions.

Arithemetic Instructions: (operates on Bytes or word)

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate (2's comp)	NEG

Logical & Bit Manipulation Instructions: (Operates on single bits)

Name	Mnemonic	
Clear	CLR	
Complement (1's comp)	COM	
AND	AND	
OR	OR	
Exclusive-OR	XOR	
clear carry	CLRC	→ c is both
Set carry	SETC	
Complement carry	COMC	
Enable interrupt	EI	interupt flag = 1 (processor accepts any interpults)
Disable interrupt	DI	interupt flag = 0

SHIFT INSTRUCTIONS

Name	Mnemonic
Logical shift right	SHR
logical shift left	SHL
Arthemetic shift right	SHRA
Arthemetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC

Program control:-

provides descion-making capabilities and change the path taken by the program when executed in computer

ex:- Branching (BUN, --)

- The instruction set of a particular Computer determines the register transfer operations and control decisions that are available to the user

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
call	CALL
Return	RET
Compare (by sub)	CMP
Test (by ANDing)	TST