Unit-2 Basic Computer Organization & Design

Instruction Codes:-

Group of bits that instruct Computer to perform given instruction.

Instruction Codes together with data (operand) are stored in Memory.

The processor fetches the Instr from "Memory" & places it in a Control Register. This fetching Operation is done by Control unit of CPU.

The Control then interprets the binary pattern of instruction & proceeds to execute it by issuing a sequence of Micro operations.

After Computation the result is sent to output unit. This Result is also stored in Memory for future Reference.

Thus the ability to store & Execute instructions is said to be stored program Concept.

This Concept was developed by John Von Neumann.

The instruction is divided into fields op-code & operand Address.

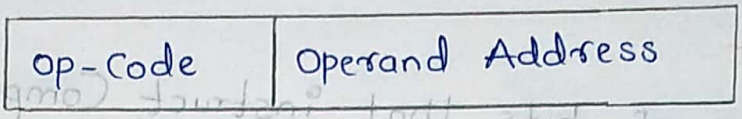
What is Op-Code?

Op-Code Specifies the type of operation to be performed.

Eg:- For 256 distinct operations We need 2^7 i.e 7 bits as op-code i.e op-code bits

op-code is called as Macro operation

Instruction Code Format :-



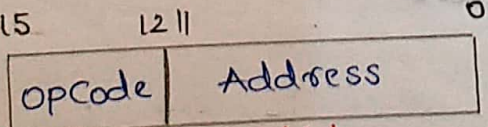
Eg :- A + B
operands
op-code

Stored program Organization

What are processor Register ?

1) INPR = Input Register
2) OUTR = output Register
3) PC = program Counter
4) IR = Instruction Register
5) TR = Temporary Register
6) MAR/AR = Address Register
7) MDR/DR = Data Register
8) AC = Accumulator

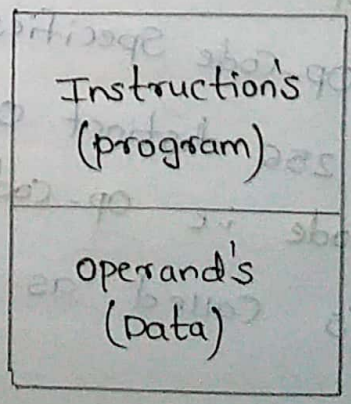
Easy Way to organize a Computer is to have One processor Register & Inst^{r} Code Format With Two Fields



⇒ 16-bit

Binary operand

Memory 4096 × 16



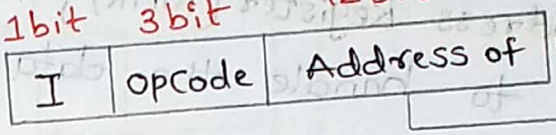
Memory $4096 \times 16 \Rightarrow 2^{12} \times 2^{4}$

12 address lines — 4 Op-code bits i.e 2^4 distinct operations

4096 Words of 16 bit Each (or) 4096 Location's Where Each Location Contains 16 bit Inst^{n} (or) Data

Addressing Modes

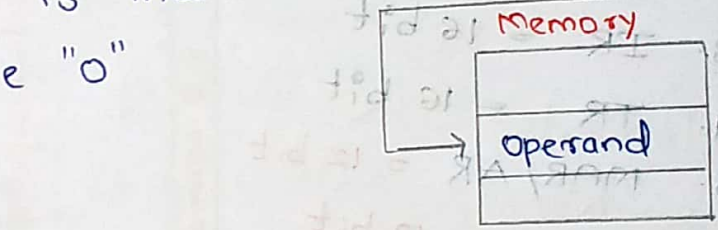
1. Direct Addressing:-



Here I = 0 for Direct Addressing

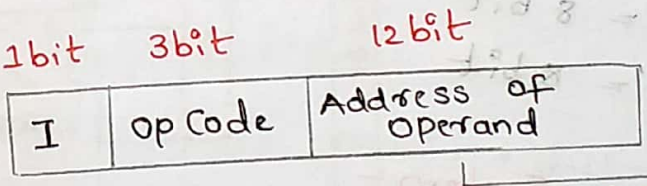
It uses direct Address of operand. This address is a Memory address Where Data is stored.

This Direct addressing is indicated to the processor by clearing I bit i.e "0"

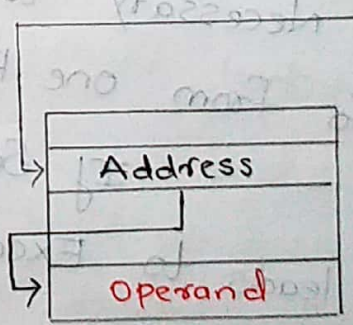


Eg:- Mov Ax, Address

2. In-direct Addressing



Here in 2nd part of instruction Code Specifies another Memory Location Where the desired data is Located. This is indicated to the processor by Setting I bit i.e 1



Eg:- Mov Ax, [Address]

Computer Registers :-

Computer Executes Instructions which are stored sequentially in a Computer Memory. These instructions are Fetched → Decoded → Executed in a proper sequence.

To perform the task of Fetching the instruction Computer uses program Counter (PC); Instr Register (IR); Memory Address Register (MAR); Memory Data Register (MDR) are used to handle the data transfer blw main Memory & processor.

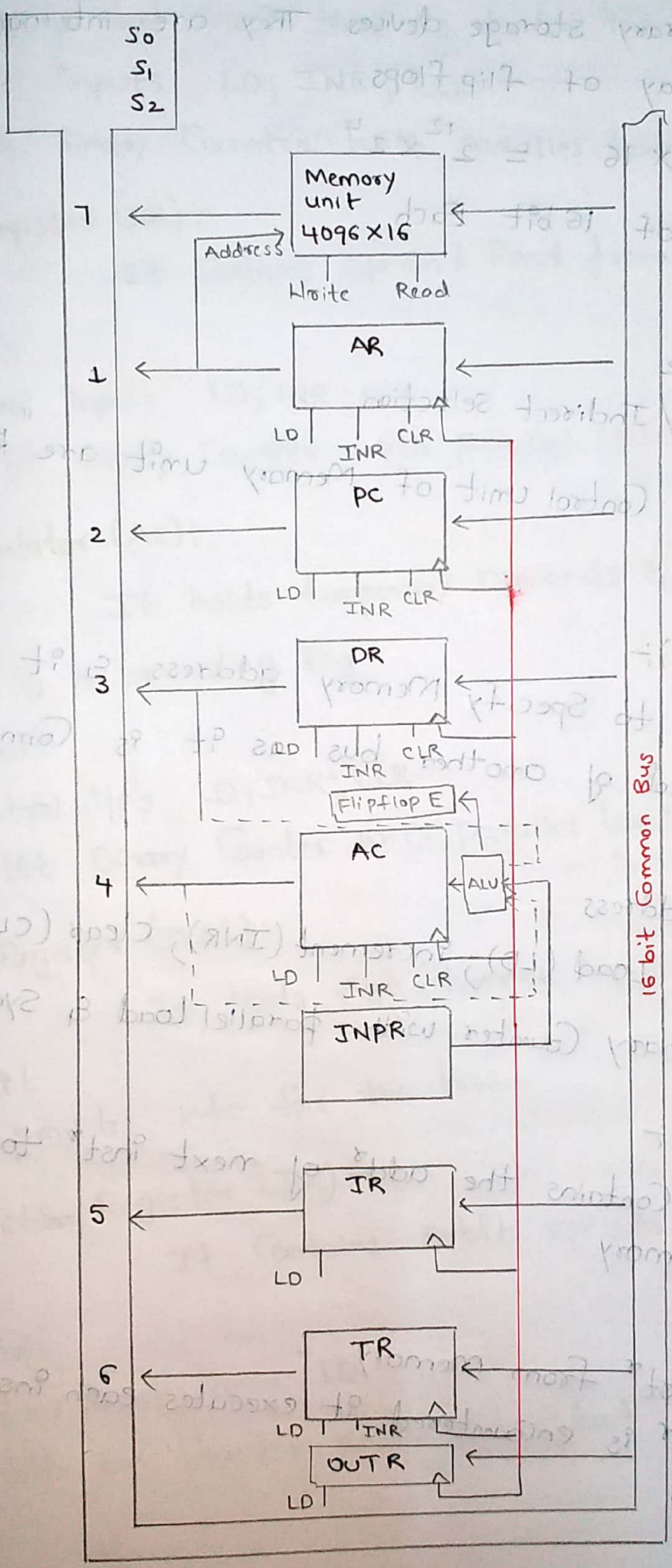
1. MDR/DR = 16 bit
2. AC = 16 bit
3. IR = 16 bit
4. TR = 16 bit
5. MAR/AR = 12 bit
6. PC = 12 bit
7. INPR = 8 bit
8. OUTR = 8 bit

Common Bus System :-

The Basic Computer has Eight Registers & it is Necessary to provide Data path blw them to transfer info from one Register to Another.

If Separate Wires are used blw Each Register it leads to Excess Number of Wires & makes the Ckt Complex.

To avoid Such problem We are using Common Bus System.



Registers are Temporary storage devices. They are internally made up of an array of flip flops.

Memory unit :- $4096 \times 16 = 2^{12} \times 2^4$

4096 Words of 16 bit Each

12 bit address
3 bit Op-Code
1 bit Direct / Indirect Selection

The Control unit of Memory unit are "Read" & "Write"

Address Register (AR) :-

used to Specify Memory address & it eliminates the need of another bus as it is Connected to Memory address

* 12 bit Memory address
* Control inputs Load (LD); increment (INR); Clear (CLR)
* it is 4-bit Binary Counter with parallel Load & Sync Clear.

Program Counter (PC) :-

It Contains the add^r of next inst^r to be fetched from Memory.

* 12 bit
* used to Read inst^r from Memory
* When branch inst^r is encountered it executes each inst^r Word Sequentially

* It holds Next inst^{n} that is to be incremented by "1"
* Control inputs LD; INR; CLR
* 4 bit Binary Counter with parallel Load Sync Clear

Data Register (DR):-

It Contains Operand Read from Memory

* 16 bit
* Control inputs LD; INR; CLR
* 4 bit Binary Counter with parallel Load Sync Clear

Accumulator (AC):-

It holds temporary operands & Results of ALU

* It is a processing Reg
* 16 bit
* Control i/p's LD; INR; CLR
* 4 bit Binary Counter with parallel Load Synch Clear

Input Register (INPR):-

It holds Alphanumeric input info

* 8 bit
* It provides info for the bus

Instruction Register (IR):-

It Contains 8-bit opcode inst^{n} being executed

* 16 bit
* Control input only "LD"
* 4 bit Bin Counter with parallel load

Temporary Register (TR) :-
It Contains Temporary Data

* While processing it holds Temp Data
* 16 bit
* LD; INR; CLR are Control inputs
* 4 bit Binary Counter with parallel Load & Sync Clear

Output Register (OUTR) :-
It accepts the Coded info

* it Contains output Character
* 8 bit
* Control input (LD)
* Receives info from bus

So; S1; S2 are Selection Variables

ALU (Adder & Logic ckt) :- Contains 3-sets of input

* 1st set of 16-bit i/p from AC
* 2nd set of 16-bit i/p from DR
* 3rd set of 8-bit i/p from INPR

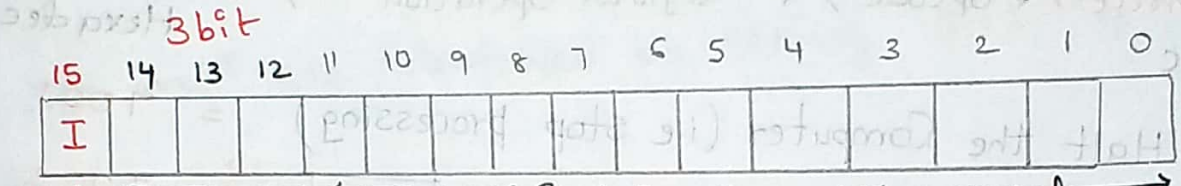
Flip Flop (E) :-
Holds End Carry out of all arithmetic operations performed.

Computer Instructions:-

A Basic Computer has 3 types of instructions Each of which are 16-bit Long

a) Memory Reference Instructions:-

This Instr requires the access of memory either to store (or) Load data for Execution.



When I = 0

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

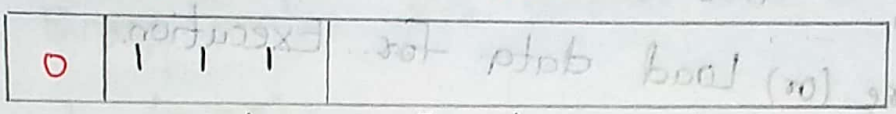
When I = 1

1 000 = 8
1 001 = 9
1 010 = 10
1 011 = 11
1 100 = 12
1 101 = 13
1 110 = 14

AND:- AND's the Content of Specified Memory Location to AC
ADD:- ADD's the Content of Specified Memory Location to AC
LDA:- Loads Accumulator with Specified memory Location
STA:- Stores the Content of AC to Specified Mem Location
BUN:- unConditional Branch
BSA:- Branch to Specified Mem Location & Save Return address
ISZ:- Increment the Content of Specified mem location & skip if Zero

b) Register Reference Instruction's :-

It does not Require access to Memory for Execution instead it Specifies the test (or) operation to be performed on Accumulator.

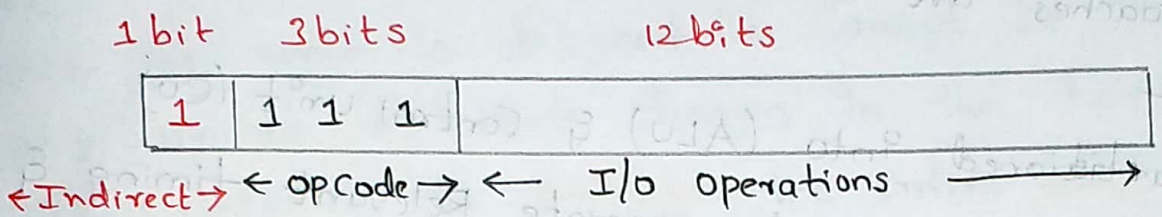


←Direct→ ←opcode→ ← Register operation →

Mnemonic		Hexa dec Code
1) HLT	= Halt the Computer (i.e stop processing)	= 7001
2) SZE	= Skip Next inst^{n} if Carry bit E=0	= 7002
3) SZA	= Skip Next inst^{n} if AC is 0	= 7004
4) SNA	= Skip Next inst^{n} if AC is -ve	= 7008
5) SPA	= Skip Next inst^{n} if AC is +ve	= 7010
6) INC	= Increment AC	= 7020
7) CIL	= Circulate Left accumulator & Carry bit E	= 7040
8) CLA	= Clear the Content of AC	= 7800
9) CIR	= Circulate Right AC & Carry bit E	= 7080
10) CLE	= Clear the Carry bit E=0	
11) CME	= Complement the Content of AC E	= 7100
12) CLE	= Clear E	= 7400
13) CMA	= Complement AC	= 7200

c) Input output Instruction's :-

They access the i/p / o/p ports through Registers for Execution purposes.



I/o inst^{r} are identified by setting I = 1 & 3 bits of opcode to 111

Mnemonic

1) INP = Input Character to AC = F 800
2) OUTP = Output Character from AC = F 400
3) SKI = Skip on input Flag = F 200
4) SKO = Skip on output Flag = F 100
5) ION = Interrupt ON = F 080
6) IOF = Interrupt OFF = F 040

Instruction Completeness

1) Data Transfer
2) Arithmetic
3) Logical
4) program Control Inst^{r} = These Inst^{r} changes the sequence of program Execution.
5) I/o Inst^{r} = These inst^{r} transfer info blw processor (or) its main memory & External I/o devices

Timing & Control :-

There must be some means for generating the Control signals in proper sequence to Execute instructions, Two general approaches are available to design the Control unit

Control unit :-

* CPU is partitioned into (ALU) & Control unit (CU)
* The function of C.U is to generate Relevant timing & Control signals to all operations in the Computer
* It Controls the flow of data blw the processor & memory & peripheral

Functions of Control unit :-

* The Control unit directs the entire Computer System to Carry out stored program instructions
* The C.U must Communicate with both ALU & main memory
* The C.U instructs the ALU that which Logical or arithmetic operation is to be performed
* The Control unit Co-ordinates the activities of other two units as well as all peripherals & auxillary storage devices linked to the Computer

Design of Control unit :-

Control unit generates Control Signals using one of the two organizations :-

a) Hardwired Control unit
b) Micro-programmed Control unit

Hard Wired Control unit:-

In this organization Control unit is designed with "Logic Gates"; "Flip Flops"; "Decoders" & other digital ckts.

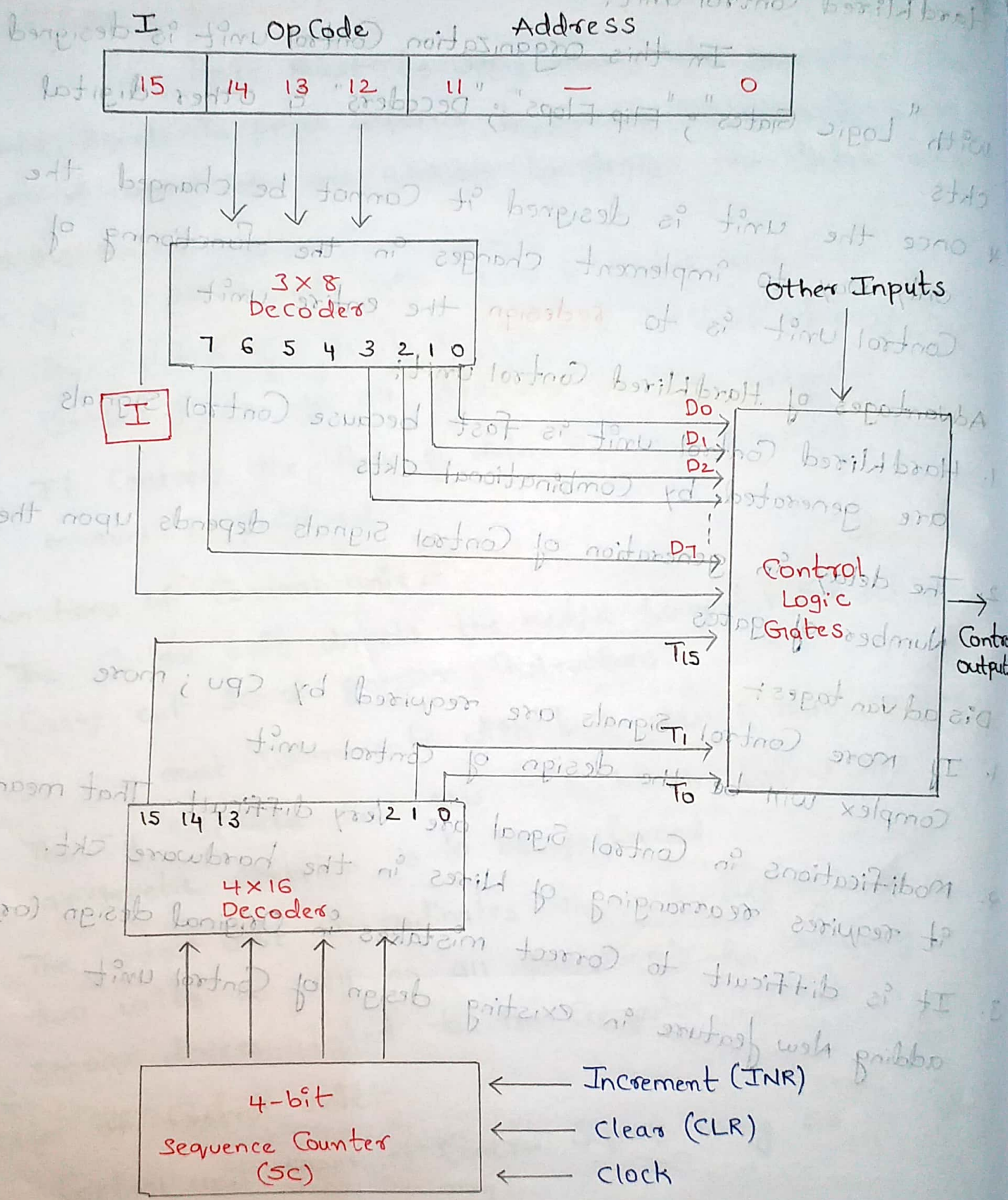
* once the unit is designed it Cannot be changed the only way to implement changes in the functioning of Control unit is to redesign the entire unit

Advantages of Hard Wired Control unit:-

1. Hard Wired Control unit is fast because Control Signals are generated by Combinational ckts
2. The delay in generation of Control Signals depends upon the Number of gates

Disadvantages:-

1. If more Control Signals are required by CPU; more Complex will be the design of Control unit
2. Modifications in Control Signal are very difficult. That means it requires rearranging of Wires in the hardware ckt.
3. It is difficult to Correct mistakes in Original design (or) adding New feature in existing design of Control unit



Hard Wired Control unit :-

Control unit Consists of following

a) Instruction Register
b) Number of Control Logic gates
c) Two Decoders
d) 4-bit Sequence Counter

An Instruction Read from Memory is placed in the Instruction Register (IR)

The IR is divided into three parts "Ibit"; "op-code"; "address"

* First 12-bits (0-11) are applied to Control Logic gates
* The op-Code (12-14) are decoded with 3x8 Decoder
* The eight outputs (D_0 through D_7) from a decoder goes to the Control Logic gates to perform Specific operation
* Last bit i.e 15^{th} is transferred to I-FF designated by symbol- I

The 4-bit SC can Count in binary from '0' through '15'

The Counter output is decoded into "16" timing pulses $T_0 \rightarrow T_{15}$

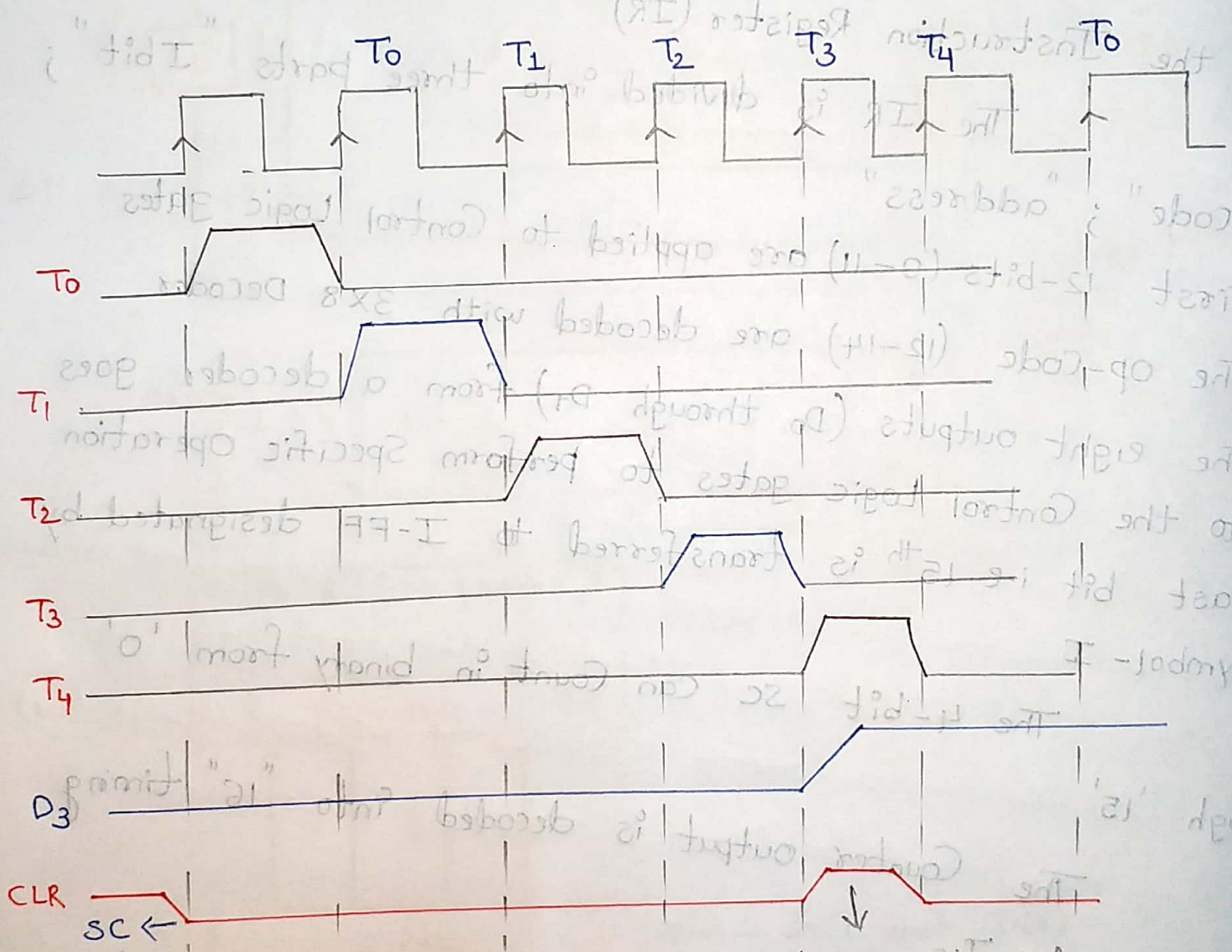
The Sequence Counter (SC) Can be incremented by (INR) input (or) Clear by (CLR) input Synchronously.

For Example:-

Consider the Case where SC is incremented to provide timing Signals T_0 T_1 T_2 T_3 & T_4 in Sequence. At Time T_4 SC is cleared to 0 if decoder output D_3 is active.

This Can be Expressed Symbolically as

$D_3 T_4 : SC \leftarrow 0$



When SC = 0000 i.e if it is cleared using CLR input Timing Signal T_0 gets activated out of decoder.

* At Timing Signal T_0 the Control Signals that are Connected to T_0 will trigger the Corresponding register

$$T_0 : AR \leftarrow PC$$

i.e the Contents of "PC" are transferred to "AR" at time "T_0"

* Next "SC" is incremented by "1" as long as "CLR" input of "SC" is active i.e Timing Sequences Continues as $T_0\ T_1\ T_2\ T_3\ T_4 \ldots$ till T_{15}

* In Order to stop Timing Signal at T_4 and start at T_0 SC must be cleared to "0"

i.e $D_3T_4 : SC \leftarrow 0$

Micro programmed Control Unit :-

In this Organization the Control Signals information is stored in a Control memory. The Control memory is programmed to initiate the required sequence of Micro Operations. unlike in hardwired in this approach the C.U Can be Modified by updating Microprogram in Control memory.

Instruction Cycle :-

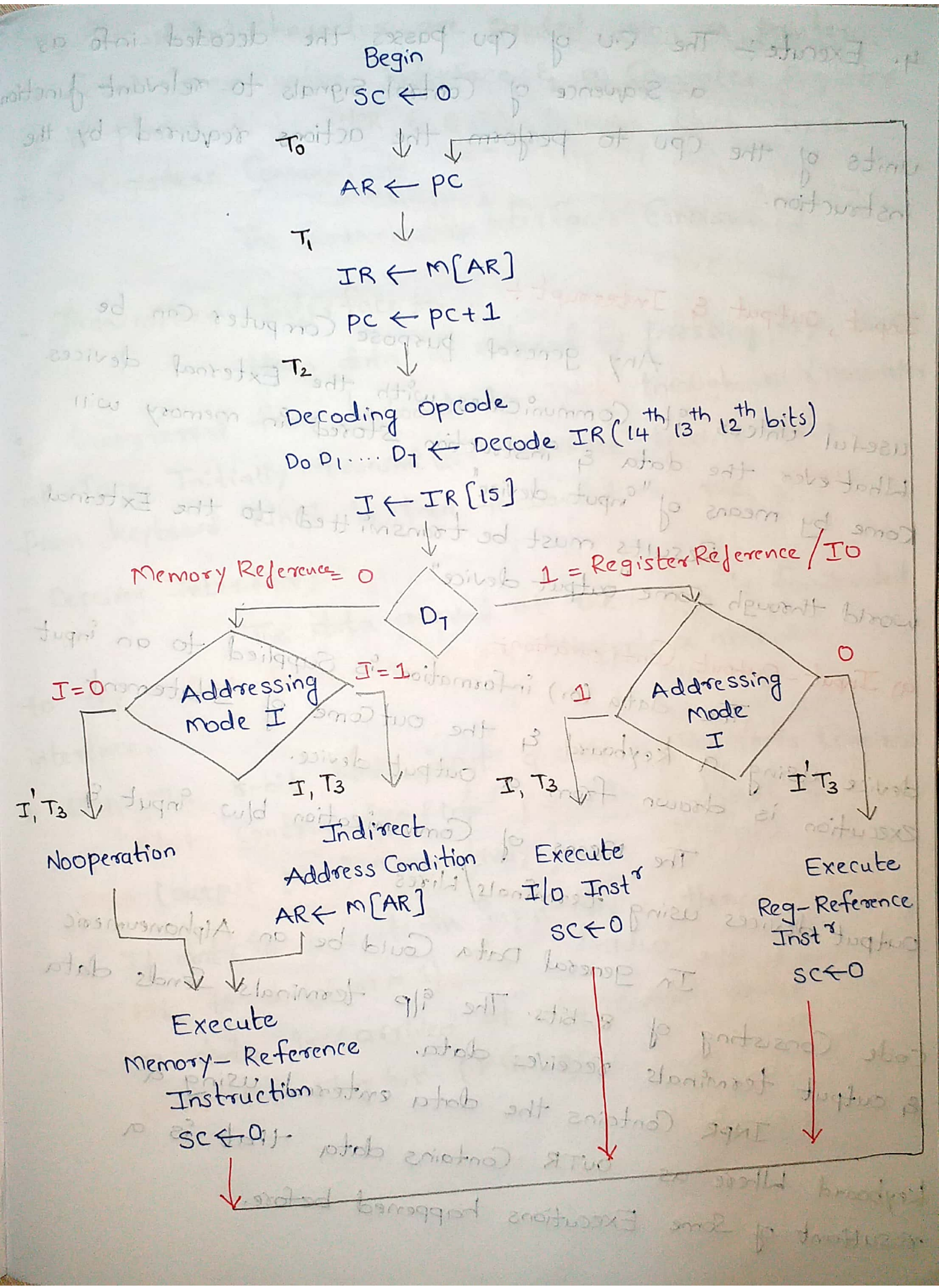
1. An Inst^{r} cycle (Sometimes Called fetch decode cycle)
2. It is basic operational process of Computer in which a Comp retreives a program instruction from its memory that determines what actions the instruction dictates & Carries out those action

Components of Instruction Cycle

1. PC
2. MAR
3. MDR
4. IR
5. CU
6. ALU

Steps of Inst^{r} Cycle

1. Fetch the Instruction :- The Next Inst^{r} is fetched from the Mem add^{r} that is Currently Stored in (PC) & stored in (IR)
2. Decode :- During this Cycle the Encoded Inst^{r} present in (IR) is interpreted by decoder
3. Read the Effective Address :-
   a) Direct Mem Inst^{r} :- Nothing is being done in clock pulse
   b) Indirect :- The EA is being read from Mem



4. Execute :- The C.U of CPU passes the decoded info as a sequence of Control Signals to relevant function units of the CPU to perform the actions required by the instruction.

Input, Output & Interrupt :-

Any general purpose Computer can be useful unless it Communicates with the External devices. Whatever the data & instruction stored in memory will come by means of "input device".

Results must be transmitted to the External world through some "output device"

a) Input - Output Configuration :-

The data (or) information is supplied to an input device using a keyboard & the outcome of statement execution is drawn from a output device.

The Means of Communication b/w input & output devices using terminals/Wires

In general Data Could be an Alphaneumeric Code Consisting of 8-bits. The i/p terminals Sends data & output terminals receives data.

INPR Contains the data entered using a keyboard Where as OUTR Contains data that is a resultant of some Executions happened before.

The Result (or) character are printed using a printer. A Serial Communication interface & a Computer Register AC is placed b/w INPR & OUTR through which these two registers Communicate.

The Communicatean interfaces Consists of

- Transmitter Interface :-

When data is entered by pressing keys, data is transferred from keyboard to INPR through a transmiter interface. Initially transmitter interface receives data from keyboard which is then forwarded to the INPR.

- Receiver Interface :-

The data arrived at the OUTR is forwarded to the output device i.e printer through a receiver interface.

* Both the 8-bit wide input & output Registers Consists of 1bit Control flipflop FG1 (Input Flag) & FG0 (output Flag) respectively.

- If data is present in input device then Set Flag bit to 1 (FG1 = 1)
- If data is arrived at the output device then clear the Flag bit (FG1 = 0)

* The data is transferred from keyboard to INPR in following manner

- The input Flag (FGI=1) is set to "1" so that the data received from keyboard Can be transmitted to INPR
- The Data in the INPR remains Same till the input Flag (FGI=1)

Even though a user presses any key data remains unchanged.

- The data gets transferred from INPR to Ac When the Computer Encounters flag bit value as 1
- Once the transfer of data is Complete the flag bit is cleared to "0"
- finally the INPR starts Receiving data upon Clearing the flag bit.

* The data is transferred from OUTR to printer

- Set the output Flag to 1 (FGO=1)
- The data present in Ac is transferred to OUTR When Computer Encounters Flag bit to (FGO=1)

Input - output Instructions :-

INP = $AC(0-7) \leftarrow INPR$; $FGI \leftarrow 0$

The "INP" performs the transfer of input information from INPR in to Lower 8-bits of AC & then input Flag (FGI=0) is cleared.

It is cleared since INPR has No data

OUT = $OUTR \leftarrow AC(0-7)$;

SKI = If (FGI=1) then $PC \leftarrow PC+1$

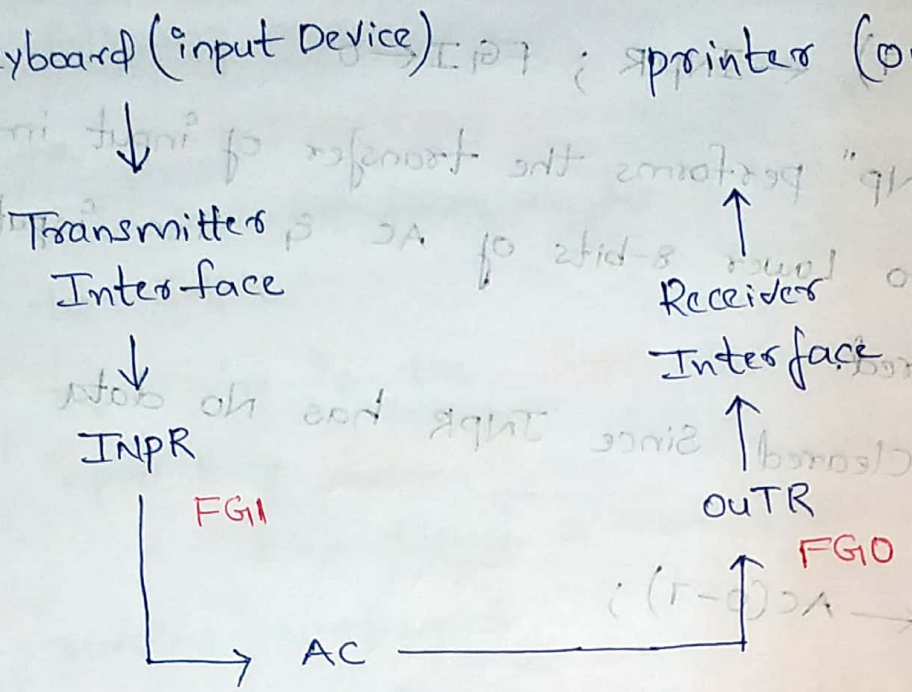
SKO = If (FGO=1) then $PC \leftarrow PC+1$

Here (FGI=1) means (INPR) has Some data that is to be Read.

When (FGO=1) the OUTR has Some data that is to be printed.

ION = $IEN \leftarrow 1$ (Interrupt ON)

IOF = $IEN \leftarrow 0$ (Interrupt OFF)



FGI = 1 When INPR has Content

FGO = 1 When OUTR has Content

Program Interrupt:-

The Rate of information transfer blw Computer & input-output device makes the system inefficient if the information flow among the devices are different

i.e Computer has to check the flag bit several times inorder to decide whether to transfer (or) Not the info.

Thus the time required to check the "flag bit" is wasted Resulting inefficiency as no other useful task can be performed at the same time.

To Eliminate Consumption of time required to check flag bit each time an External device of a Computer should be assigned with a decision making task.

This decision making device acts as a program interrupt which decides whether information should be transferred or not.

When IEN = 1 Suppose if Computer is running a program ; program interrupt informs Computer to halt the program for a while & take care of input-output transfer.

Once the transfer is Complete the Computer gets back to its position & Continues with its work.

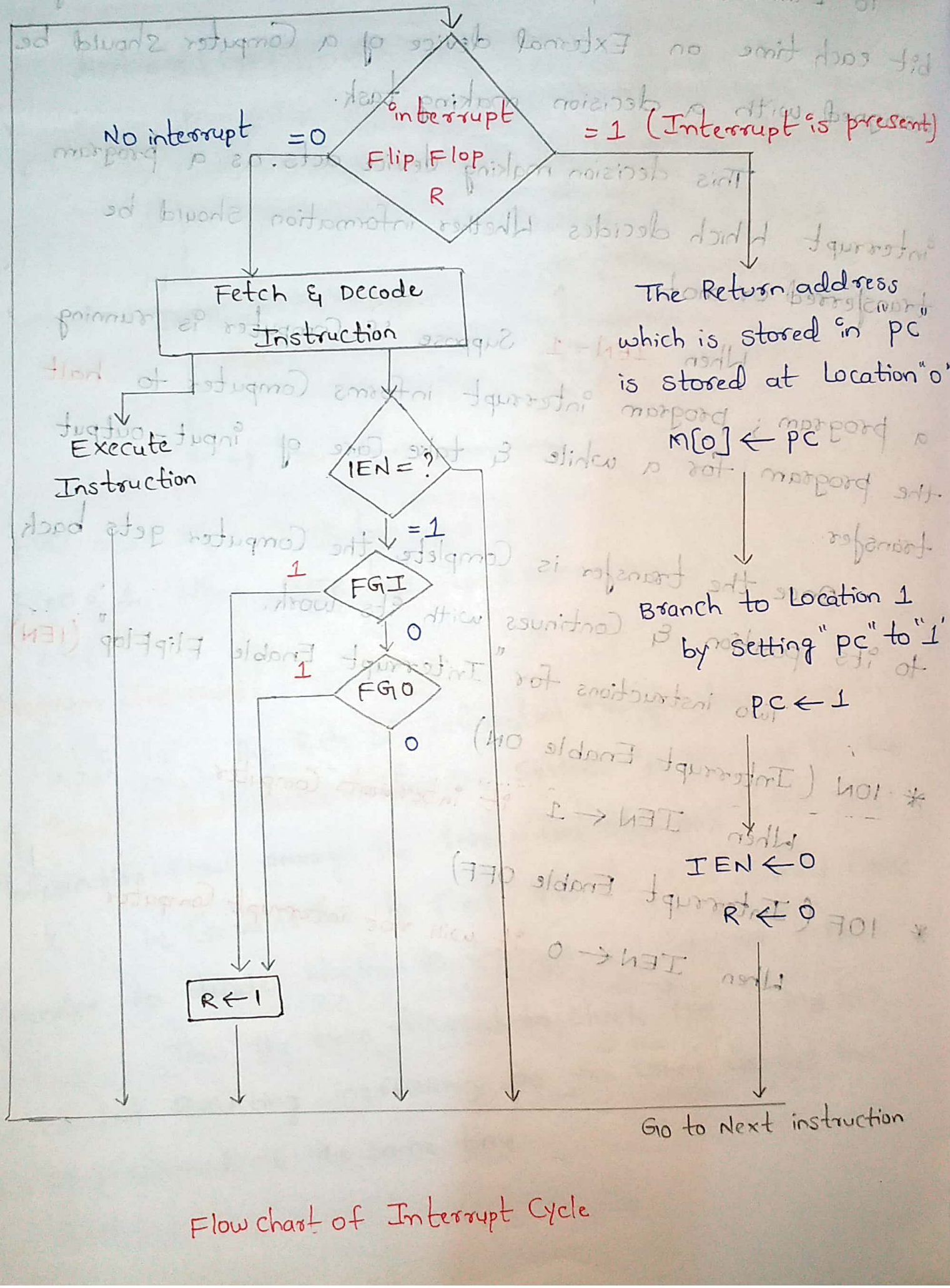
Two instructions for "Interrupt Enable FlipFlop" (IEN)

* ION (Interrupt Enable ON)

  When IEN ← 1 it interrupts Computer

* IOF (Interrupt Enable OFF)

  When IEN ← 0 it will not interrupt Computer



Flow chart of Interrupt Cycle

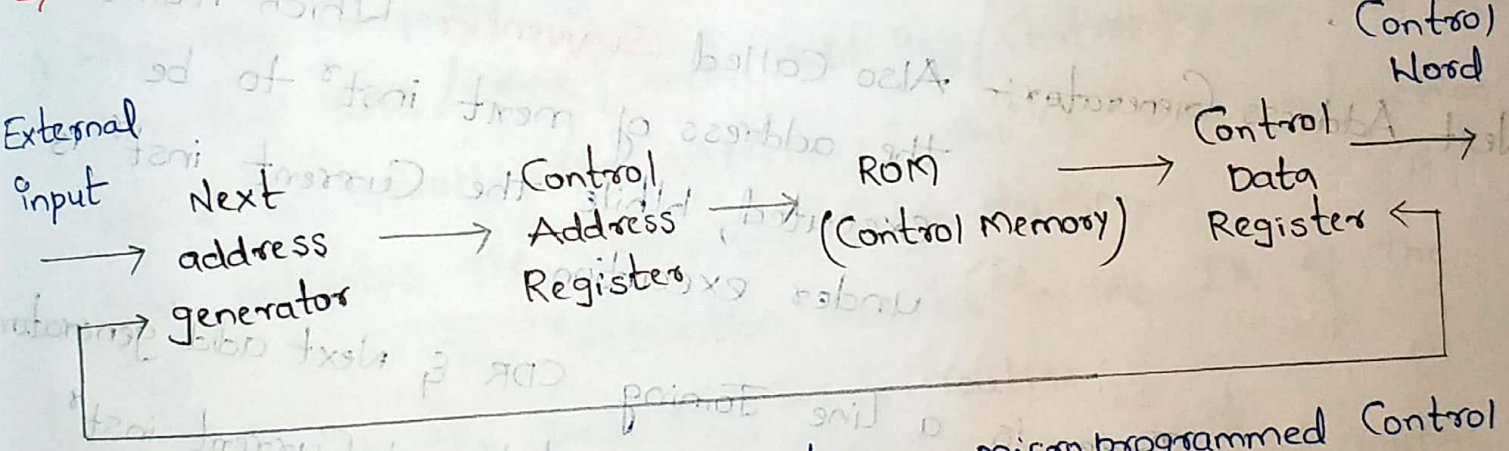
⇒ Micro programmed Control unit

The Disadvantage of Hard Wired Control unit is that once the C.U is designed further modifications can't be done.

Now in this Method "Control Memory" is used for Controlling the "Micro Operation Sequences" in a digital Computer

In this Method the C.U design can be modified by updating the "Microprogram" stored in "Control memory"

⇒ Micro programmed Control Organization



Any System maintaining micro programmed Control unit utilizes the Services of two types of memories

1) Control memory / ROM
2) Main Memory / RAM

Control Memory :- It is Called "ROM" Which stores Special type of programs referred as Microprograms.

The "microprograms" are the Combination of finite number of microinstructions. When ever these "microinstructions" are executed

it results in a Series of timing Signals which are responsible for Controlling Various "micro operations"

Main Memory/RAM:- The data stored in these maintains user related data. The data can be Read or rewritten on Convenience of user

Control Address Register (CAR):- The "CAR" is responsible to store the addresses of micro inst^{r} that are stored in "memories"

Control Data Register (CDR):- Stores the "micro instructions" fetched from the memory.

Next Address Generator:- Also Called Sequencer which maintains the address of next inst^{r} to be executed, while the Current inst^{r} is under execution.

There is a Line Joining CDR & Next add^{r} generator Here "Next add^{r} generator" fetches few bits of Current inst^{r} that is under execution So as to generate Next Consecutive instruction

Here Next add^{r} is generated by using these bits + External inputs So it is Called as "Sequencer"

As the add^{r} of Next inst^{r} is fetched while Current inst^{r} is under execution in Control Data Reg

∴ (CAR) is also Called "pipeline Register"

⇒ Address Sequencing:-

Micro instructions are stored in Control Memory in groups with each group specifying a routine.

Each Computer Instruction has its own Microprogram routine in Control memory to generate the micro operations that execute the instruction.

The Sequence of steps that the "Control unit" must undergo during the execution of a Single Computer instruction inorder to appreciate the address Sequencing in a "Micro programmed Control unit" are

1) When the Computer is turned ON an initial address is Loaded into the CAR / CMAR [Control Add^r Reg / Control Memory Add^r Reg] Which is usually the address of the 1st Micro instruction that activates the instruction fetch routine.

The instruction will be in the IR of the Computer at the end of fetch routine.

2) Next the Control Memory must go through the fetch routine so that the Effective address of the Operand Can be determined.

After Computing the Effective add^r of operand Now the operand add^r is available in Memory address Register.

3) Effective address / OFFset address Computation depends upon mode bit "I"

if I=1 it is In-direct Addressing mode if it is in this mode then "Effective addr / offset addr" has to be Computed

4) The Micro Operations that Execute the inst fetch from memory has to be generated.

The steps involved in generation of Micro operation depends upon the op-code of instruction that is to be executed.

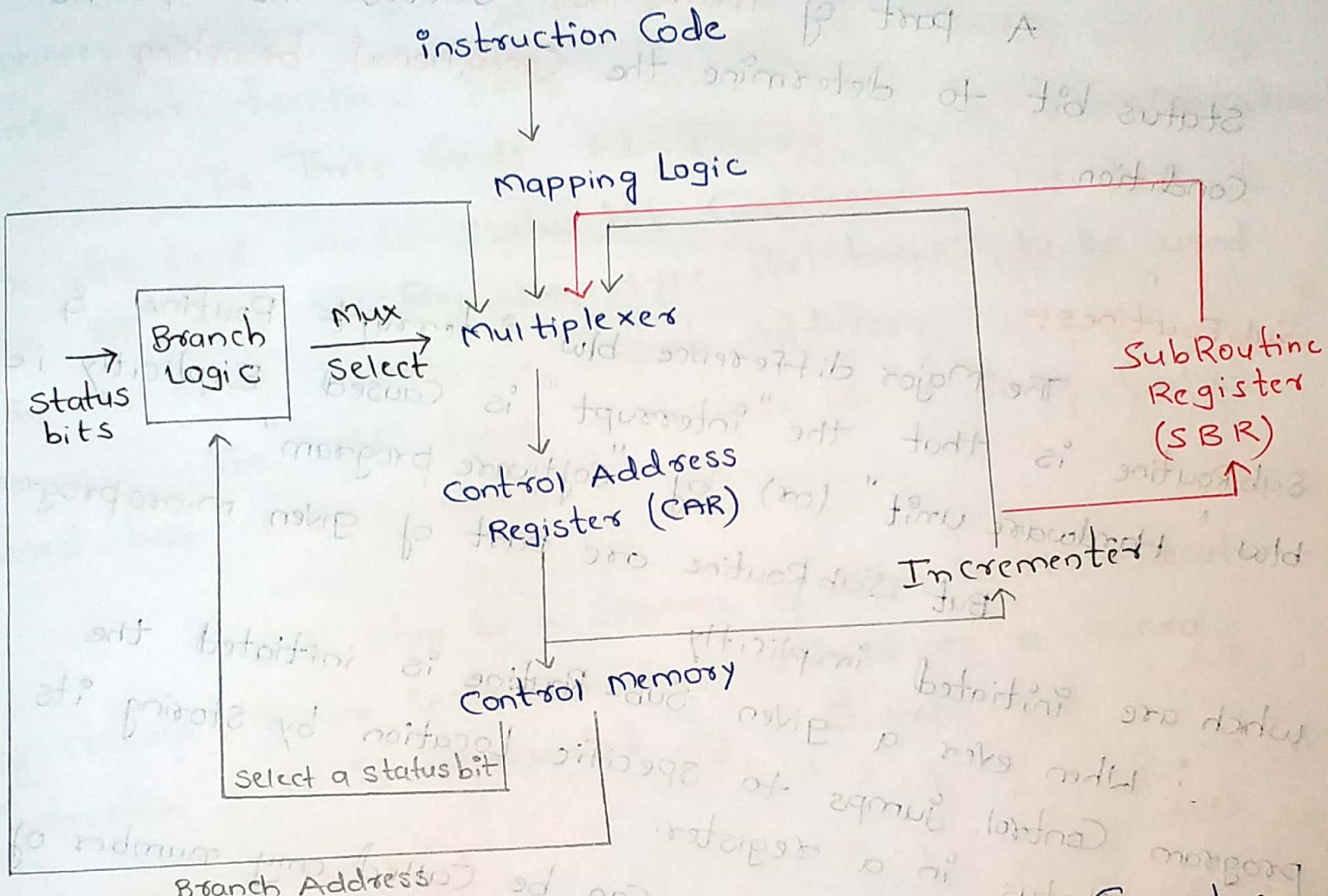
Each inst has its own Microprogram routine stored in a given Location of the Control Memory.

Thus the op-code must be Mapped into a Micro inst addr i.e the transformation from the opCode to an addr in Control Memory where the routine is Located is referred as Mapping process.

After Execution of this inst the addr in CAR is incremented.

5. Finally Control must return to the 1st addr of fetch routine as soon as the Execution of inst is Completed.

Selection of Address for Control Memory



1. CAR Loads the initial address as soon as Computer is turned ON
2. Incrementer is used to increment the CAR inorder to select the micro instr in sequence.
3. Mapping Logic tranforms the instruction Code into a Control Memory addr
4. SBR is used to store the return address for a Subroutine Whose value is then used when the micro-program wishes to return from the Subroutine.

⑤ Branch Logic is used to acheive branching by specifying the branch addr in one of the fields of Micro-instr

A part of micro inst selects the specific status bit to determine the Conditional branching condition.

SubRoutines:-

The Major difference blw interrupt Routine & SubRoutine is that the "interrupt" is caused Explicitly i.e blw "Hardware unit" (or) by "Software program"

But SubRoutine are part of given Microprogram which are initiated implicitly

∴ When ever a given Sub-Routine is initiated the program Control jumps to specific Location by storing its Current Status in a register.

A Single SubRoutine Can be Called any number of times in a given "microprogram"

⇒ Micro Instruction Format:-

The Micro Instruction Format for the Control memory is shown. The 20bits of micro instruction are divided into four functional parts.

The Three fields F1; F2; F3 Specify "Micro Operation"

The CD field Selects Status bit Conditions.

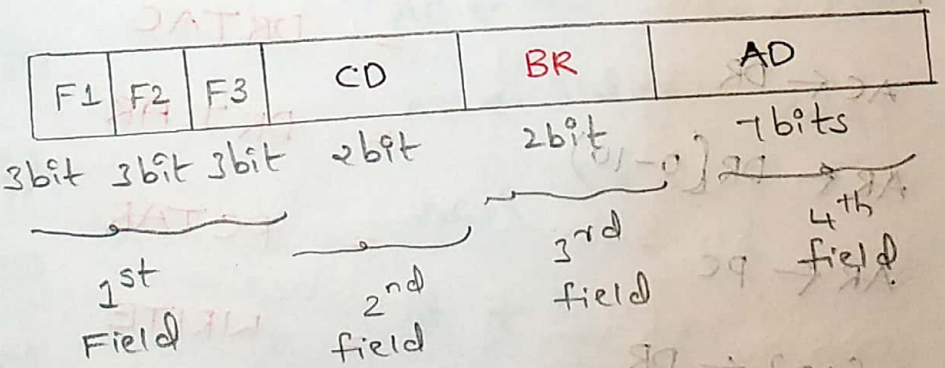
The BR field Specifies the type (or) branch to be used

The AD field Contains a branch address

The addr field is 7 bits Wide Since, Control word has $128 = 2^7$ Words.

Each step in a Sequence of steps in the Execution of certain Machine instruction is Considered as a micro instruction & it is represented by a Control word

Eg:- ADD R1, X



1st field (Micro-Operation Field):-

The micro operation field is Subdivided in to "3" fields (F1 F2 F3) of three bit each

The "3" bits in each field are encoded to Specify "7" distinct micro operations + "1" No-operation

This gives total of 21 micro operations.

* A micro instruction includes 3 different micro operations one for each field

So No more than 3 micro operations can be selected for a micro instruction"

* If micro inst^{r} needs micro operations Less than three; one or more of the micro operation fields should be filled by a binary code 000 for No operation

000 indicates No-Operation

F1			
1) 000	=	No-operation	= NOP
2) 001	=	AC ← AC + DR	= ADD
3) 010	=	AC ← 0	= CLRAC
4) 011	=	AC ← AC + 1	= INCAC
5) 100	=	AC ← DR	= DRTAC
6) 101	=	AR ← DR[0-10)	= DRTAR
7) 110	=	AR ← PC	= PCTAR
8) 111	=	M[AR] ← DR	= WRITE

F2

1)	000	= No-operation	= NOP
2)	001	= AC ← AC − DR	= SUB
3)	010	= AC ← AC ∨ DR	= OR
4)	011	= AC ← AC ∧ DR	= AND
5)	100	= DR ← M[AR]	= READ
6)	101	= DR ← AC	= ACTDR
7)	110	= DR ← DR + 1	= INCDR
8)	111	= DR(0-10) ← PC	= PCTDR

F3

1)	000	= No-operation	= NOP
2)	001	= AC ← AC ⊕ DR	= XOR
3)	010	= AC ← $\overline{AC}$	= COM
4)	011	= AC ← shl AC	= SHL
5)	100	= AC ← shr AC	= SHR
6)	101	= PC ← PC + 1	= INCPC
7)	110	= PC ← AR	= ARTPC
8)	111	= Not Specified / Reserved	

CD

	Symbol used	Equivalent Condition	Description
1) 00	U	it's value is always 1	Represent's an Unconditional Branch
2) 01	I	DR(15)	Indirect address bit
3) 10	S	AC(15)	Refer's to Sign bit of Accumulator
4) 11	I	AC=0	store a value Zero in the Accumulator. Which reflects that the accumulator is empty

BR

1) 00 = JMP = CAR ← AD if Condition = 1
CAR ← CAR+1 if Condition = 0

2) 01 = CALL = CAR ← AD, SBR ← CAR+1 if Condition = 1
CAR ← CAR+1 if Condition = 0

3) 10 = RET = CAR ← SBR (Return from Subroutine)

4) 11 = MAP = CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0