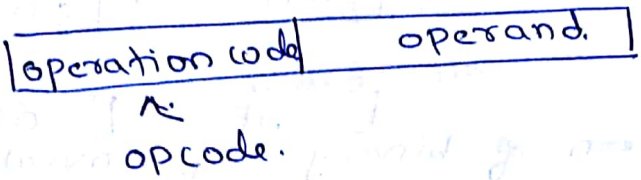
UNIT-1

BASIC COMPUTER ORGANIZATION AND DESIGN

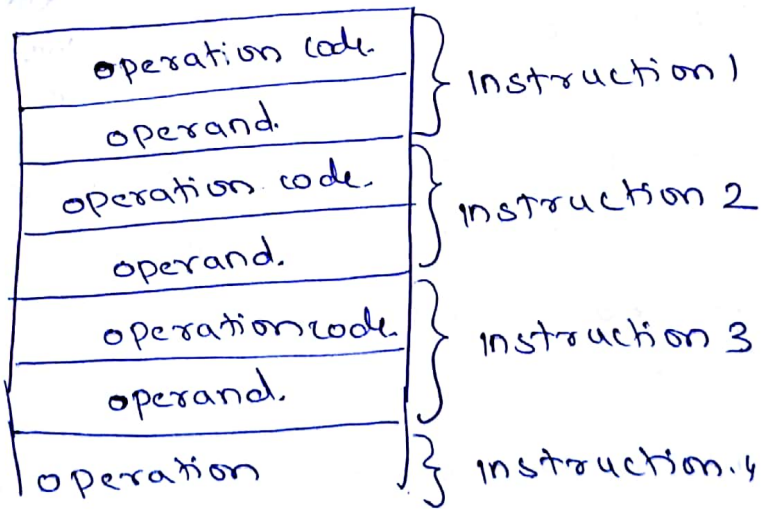
A machine code instruction usually consists of an operation code and operand.



All the operation code and the operands are binary codes

It is the binary that controls the computer hardware.

A machine code program consists of a sequence of opcodes and operands stored in the computer memory (eg RAM)



eg → ~~LAA~~ LD A, 01H

an Assembly language program

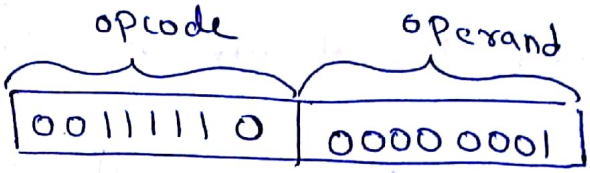
LD stand for load. A stands for register a which is inside the central processing unit. 01 stands for the.

number that is going to be loaded into register a

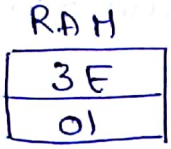
H means hexadecimal it is actually not part of the number but

assembler ⇒ converter into machine language (binary)

eg LDA, 01H → Assemble →

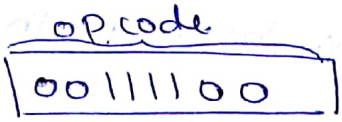


long pattern of binary are converted into hexadecimal.

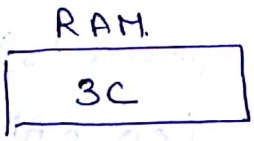


— memory location address

eg INC A → assembly →



immediate addressing
the data to be manipulated by the instruction comes immediately after the opcode



implied addressing
as it is implied that the data to manipulated by the instruction is already in place.

BASIC COMPUTER ORGANIZATION AND DESIGN

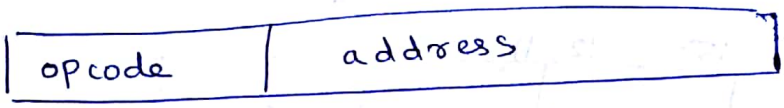
Instruction code.

An instruction code is a group of bits ~~instruction~~ that instruct the computer to perform a specific operation.

It is usually divided into two parts.

1). operation code.

2). address



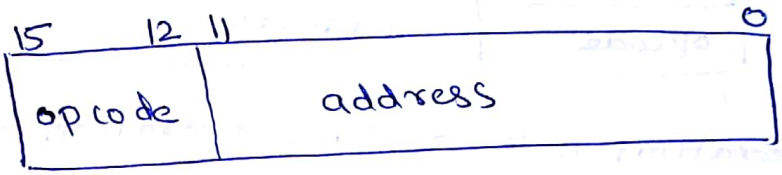
operation code → The operation code of an instruction is a group of bits that define operations as add, substract, multiply, shift and complement

The operation code must consist of at least n bits for a given 2^n distinct operation.

Address → operation must be performed on some data stored in processor register or in memory. An instruction code must specify not only the operation but also the registers or the memory words where the operands are to be found, as well as the register or memory word where the result is to be stored. memory words can be specified in instruction codes by their address.

Instruction code formats are conceived by computer designers who specify the architecture of the computer

exg → For a memory unit with 4096 words if we store each instruction code in one 16-bit memory word, we have available four bits for the opcode.



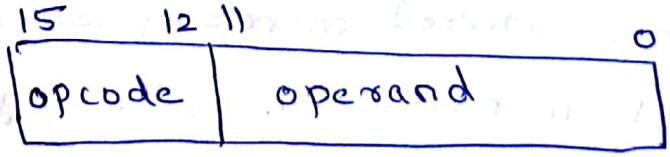
Different address format

1). Immediate operand

2). effective address — Direct address / Indirect address

Immediate operand →

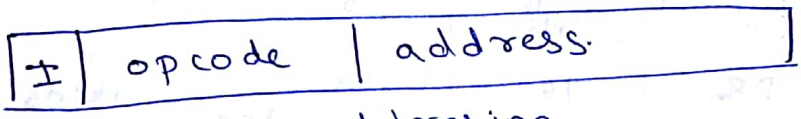
The address bits of an instruction code is not an address but an actual operand. ie the second part of the instruction code specifies an operand. The instruction is Immediate operand.



12 bits for operand and 4 bits for opcode

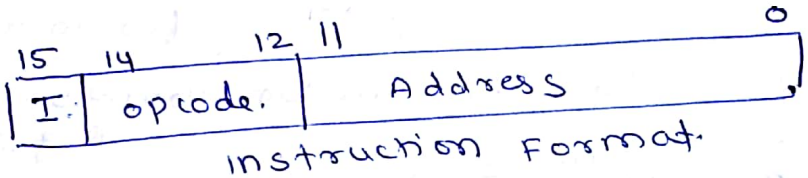
EFFECTIVE ADDRESS ⇒ The instruction does not give the operand or its address Instead it provides information from which the memory address of the operand can be determined.

A single bit is used for representing if the address mode is direct or indirect.



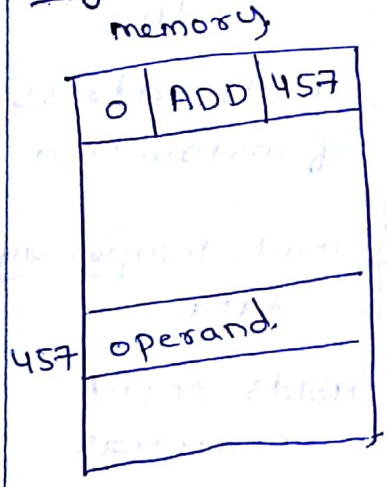
I = 0 direct addresing

I = 1 → Indirect addressing.

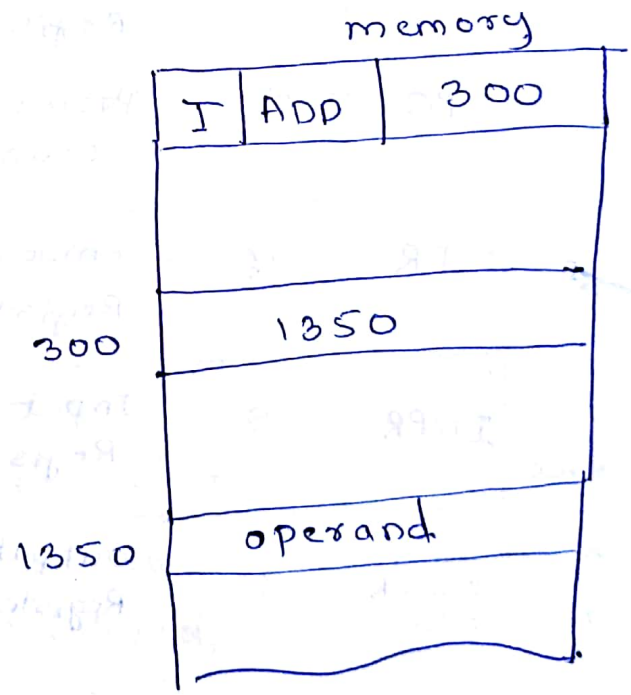


Instruction Format.

ex →



Direct address



Indirect address

COMPUTER REGISTERS

when operands are brought into the Processor, they are stored in high-speed storage elements called registers

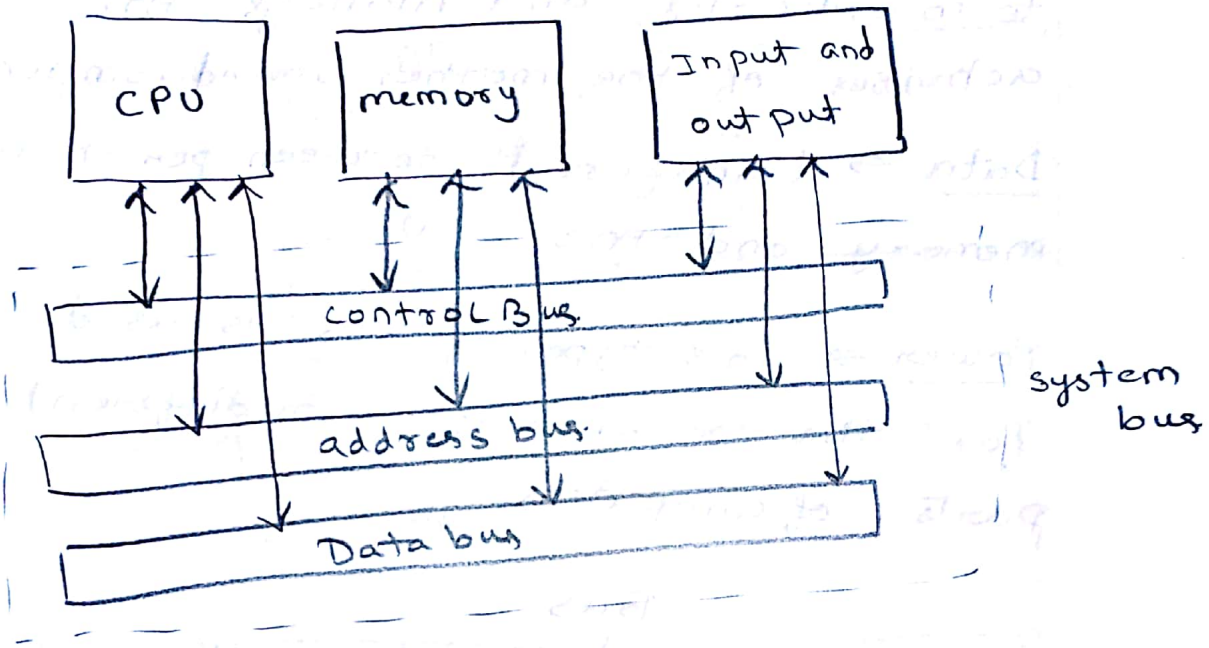
List of Registers for the basic computers

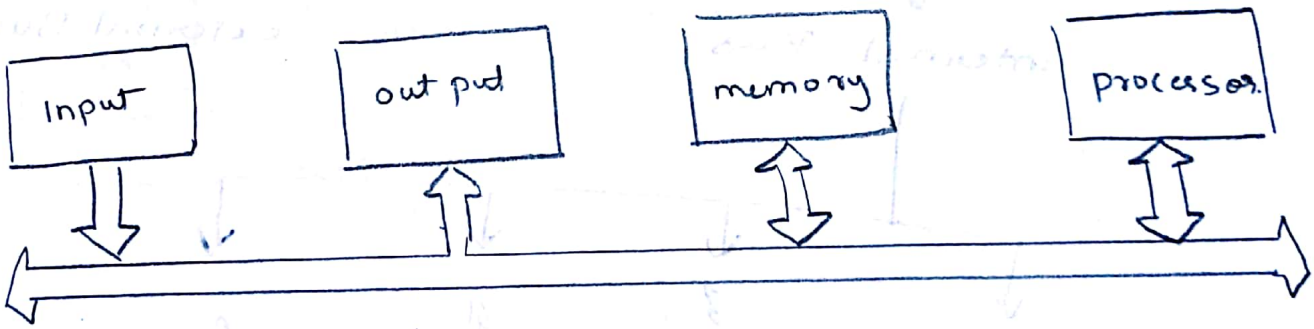
Register symbol	Number of bits	Register name	Function
DR	16	Data Register	Holds memory operands
AR	12	Address register	Holds address for memory
AC	16	Accumulator	Processor Register.
IR	16	Instruction Register	Holds Instruction code.
PC	12	Program counter	Holds address of instruction
TR	16	Temporary Register	Holds temporary data.
INPR	8	Input Register	Holds Input characters
OUTR	8	output Register	Holds output character.

Bus structures

A group of lines that serves as a connecting path for several devices is called a bus.

system bus





single-bus structure.

mainly computer BUS can be divided into two parts:

→ internal bus → carries data with in the mother-boa

→ external bus → A bus that connects a computer to peripheral devices.

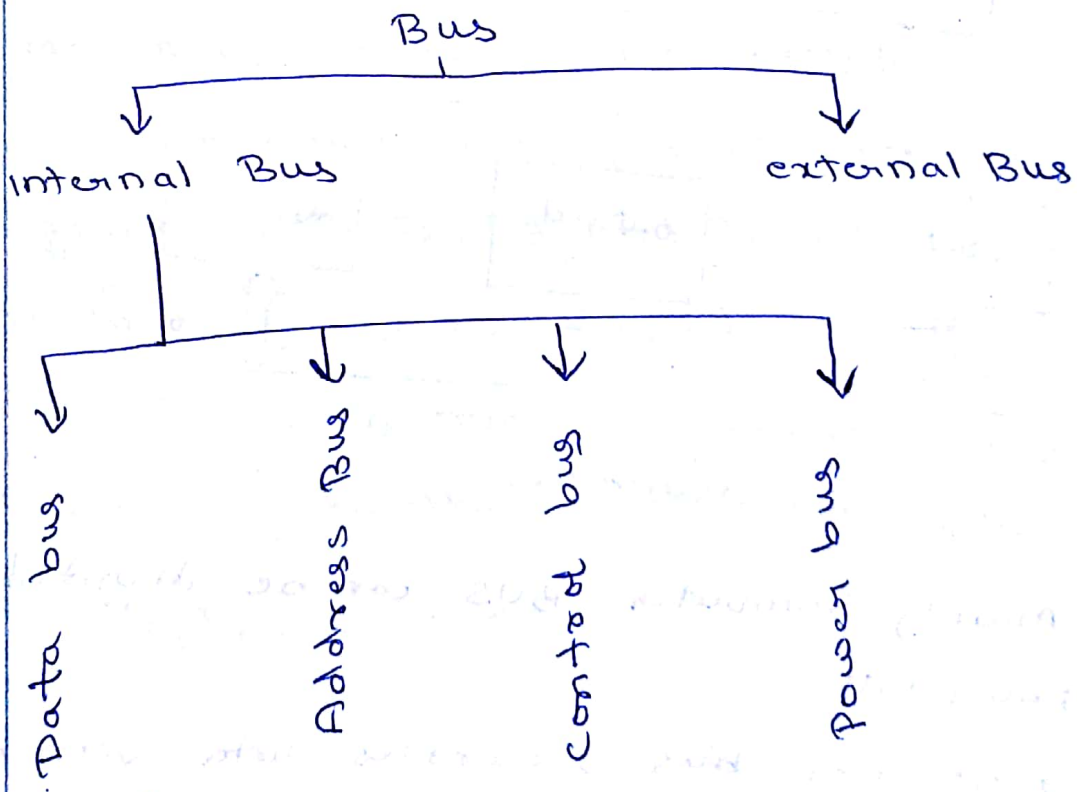
The lines or pins of a bus are of three bus types.

Address ⇒ The components pass memory addresses to one another over the address bus

control ⇒ used to send out signals to co-ordinates and manage the activities of the mother board components

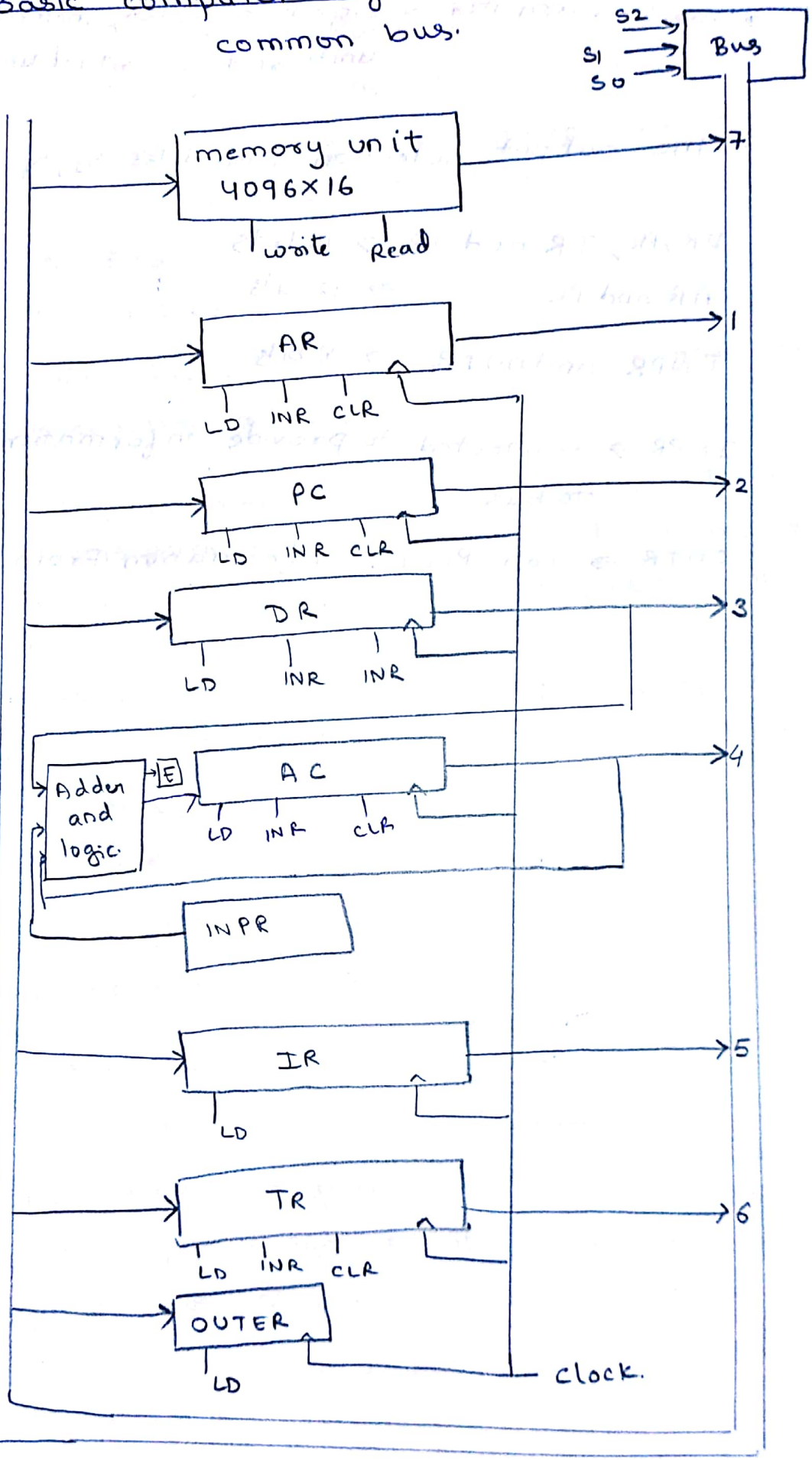
Data ⇒ transferred between peripherals memory and the CPU

Power ⇒ This type of BUS is used for the power supply to different parts of computers



9

Basic computer registers connected to a common bus.



→ Basic computer → eight register, memory unit and a control unit

→ The output selection variables S_2, S_1, S_0

DR, AC, IR and TR → 16 bits

AR and PC → 12 bits

INPR and OUTR → 8 bits

INPR → connected to provide information to bus

OUTR → can Receive information From bus

COMPUTER INSTRUCTIONS.

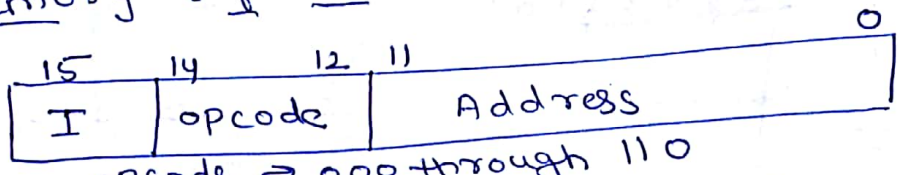
The basic computer has three instruction code format.

a). Memory reference Instruction

b). Register - reference Instruction

c). Input - output Instruction

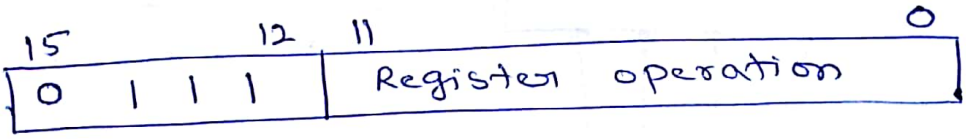
a). Memory reference Instruction



opcode → 000 through 110

A memory-reference instruction uses 12 bits to specify an address and one bit to specify the addressing mode I. I is equal to 0 for direct address and 1 for indirect address

b). Register - reference instruction

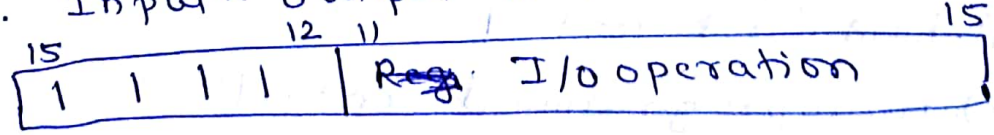


opcode → 111

I → 0

The register reference instructions are recognized by the operation code 111 with a 0 in the left most bit (15) of the instruction.

c). Input - output instruction

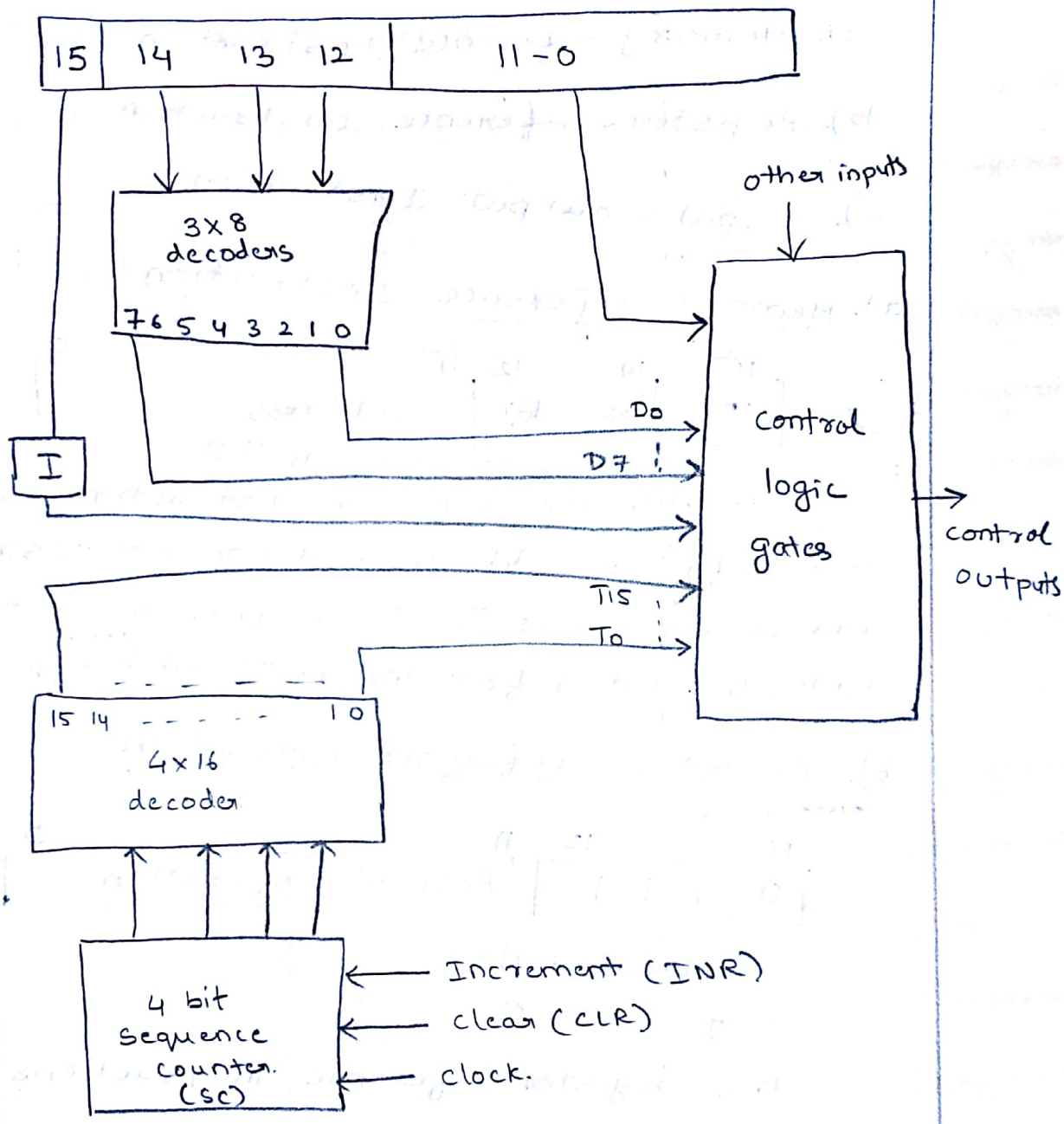


opcode = 111
I = 1

12 bits are used to specify the type of input-output operation or test performed.

TIMING AND CONTROL.

control unit of basic computer.



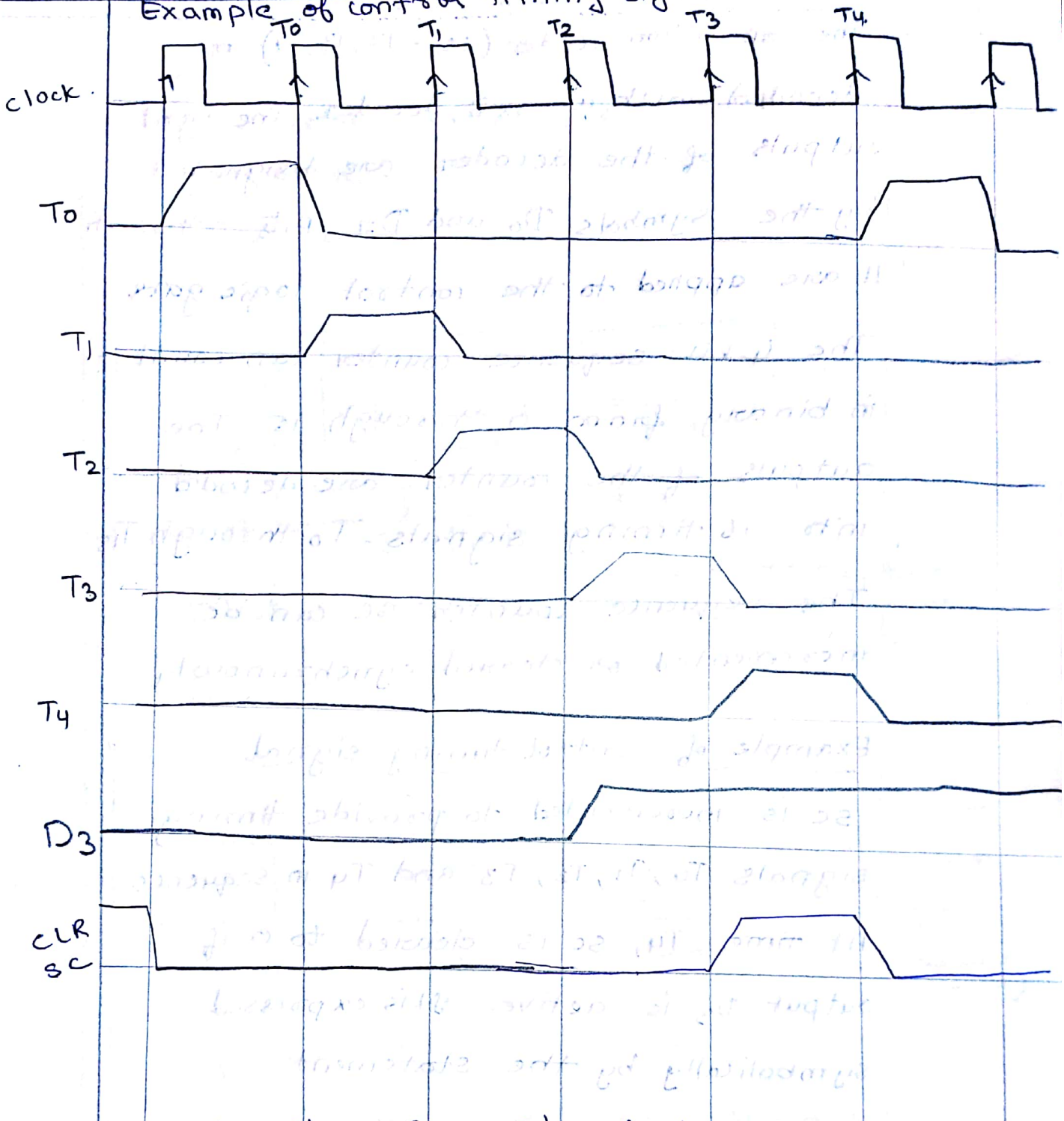
There are two types of control organization

1). hard wired control.

2). Microprogrammed control.

13.

Example of control timing signal.



control unit ⇒ It consists of two decoders, a sequence counter, and a number of control logic gates. An Instruction Read from memory is placed in the Instruction Register (IR)

IR is divided into 3 parts. The I bit, the operation code, and bits 0 through 11.

The operation code (ie - 12, 13, 14) are decoded with a 3x8 decoder. The eight outputs of the decoder are designated by the symbols D_0 and D_7. Bits 0 through 11 are applied to the control logic gates. The 4-bit sequence counter can count in binary from 0 through 15. The outputs of the counter are decoded into 16 timing signals T_0 through T_{15}. The sequence counter SC can be incremented or cleared synchronously.

Example of control timing signal

SC is incremented to provide timing signals T_0, T_1, T_2, T_3 and T_4 in sequence. At time T_4, SC is cleared to 0 if output D_3 is active. It is expressed symbolically by the statement

$$D_3T_4 : SC \leftarrow 0$$

INSTRUCTION CYCLE

In the basic computer each instruction cycle consists of the following phase.

1. Fetch an instruction from memory
2. Decode the instruction.
3. Read.
4. execute.

upon the completion of step 4, the control goes back to step1

Program has many instruction to be executed
A program has to be executed means every instruction present in that program has to be executed.
every instruction goes through a cycle.
The cycle is divided into Four phases

1) Fetch
2) Decode.
3). Decision
4). execute

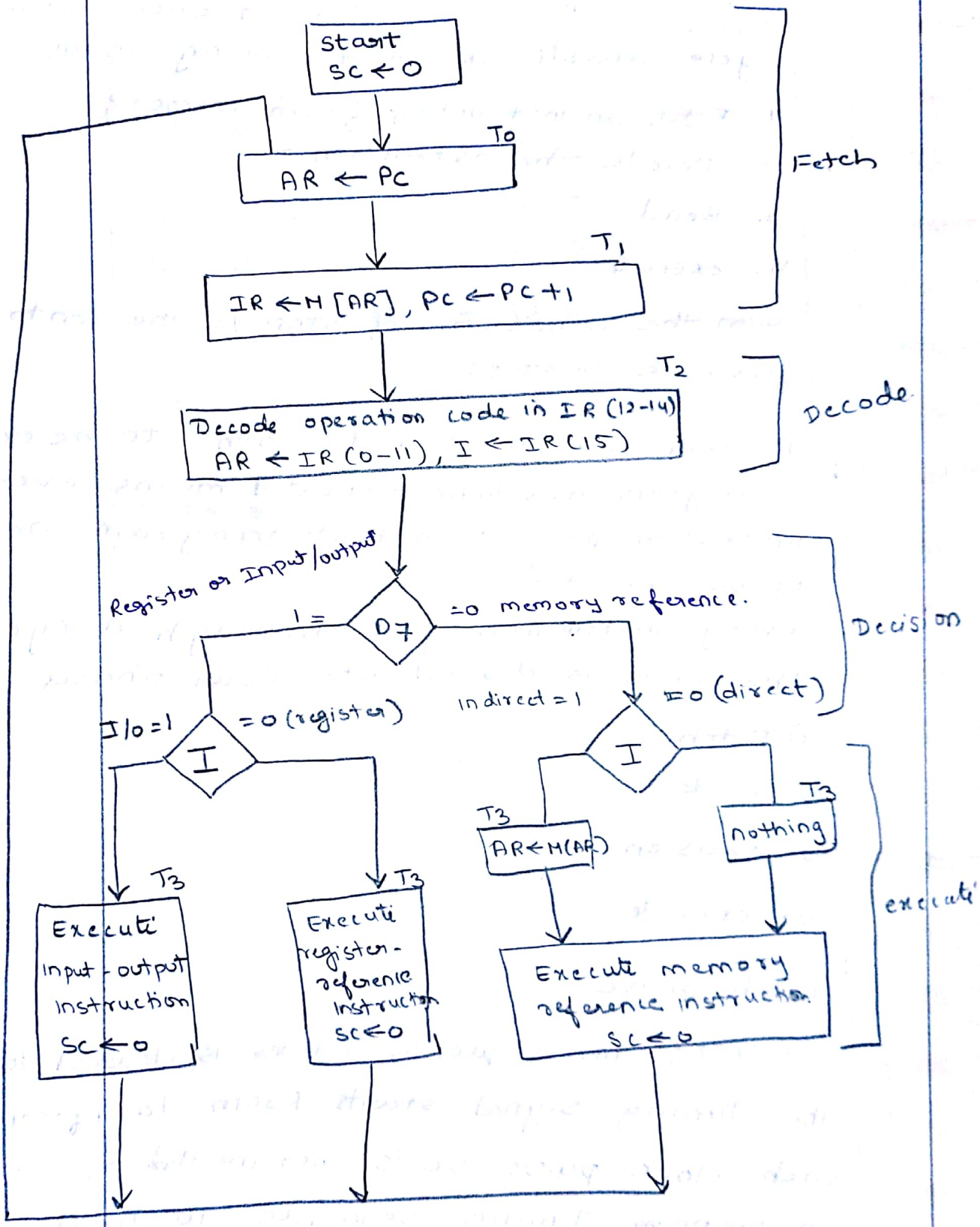
Fetch phase

initially the sequence counter is cleared to 0, The Timing signal starts From To. After each clock pulse sc is incremented by one. so we have Timing sequence T_0, T_1, T_2 ---

T_0: $AR \leftarrow PC$

during T_0 we have to Transfer the address from PC to AR

Flow chart for instruction cycle.



T_1 : $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

On timing signal T_1, the instruction Read from memory is placed in IR. At the same time, PC is incremented by one to prepare it for the address of the next instruction in the program.

Decode phase

T_2 : $D_0, \dots D_7 \leftarrow$ Decode IR (12-14)

$AR \leftarrow IR(0-11)$, $I \leftarrow IR(15)$

The operation code in IR is decoded the indirect bit is transferred to flip-flop I and address part of the instruction is transferred to AR.

Decision phase

During time T_3, the control unit determines the type of instruction that was just read from memory.

$D_7 \rightarrow$ Desision making → ~~binary code !!!~~

$gf\,0 \rightarrow$ then memory-reference instruction

$gf\,1 \rightarrow$ then Register or I/O instruction

$gf\ D_7 = 0$

then opcodes have any value from 000 to 110.

$D_7 = 0$
memory reference instruction

If $D_7 = 0$ and $I = 1$
memory reference instruction with an indirect address. It is then necessary to read the effective address from memory

$AR \leftarrow M[AR]$

If $D_7 = 0$ and $I = 0$
not necessary to do anything since the effective address is already in AR.

If $D_7 = 1$ and $I = 0$
Register reference instruction.

If $D_7 = 1$ and $I = 1$
input and output instruction.

$D_7' I T_3$: $AR \leftarrow M[AR]$

$D_7' I' T_3$: nothing

$D_7 I' T_3$: execute a register-reference instructions

$D_7 I T_3$: execute an input-output instruction.

memory reference instruction

when $D_7 = 0$ and $I = 0$.

~~12 bits 0 to 11 specify the instruction code~~

ie D_0 to D_6 → are opcode.

D_0 → AND to AC

There is no direct path from bus into accumulator first the logic receive info from the data register and then transfer. first the contents will be stored in data Register and then AND operation will be performed between the bits of the accumulator and that particular data

D_0T_4: $DR \leftarrow M[AR]$

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

(2). D_1 → ADD to AC

$DR \leftarrow M[AR]$

$AC \leftarrow AC + DR$, $SC \leftarrow 0$ $E \leftarrow C_{out}$

(3) D_2 → LDA : load to AC , $DR \leftarrow M[AR]$

$AC \leftarrow DR$, $SC \leftarrow 0$

(4) D_3 STA: store AC.

$M[AR] \leftarrow AC$, $SC \leftarrow 0$

(5) D_4. BUN → Branch unconditionally

$PC \leftarrow AR$

It allow the programmer to specify an instruction out of sequence and when that program Jumps to that particular instruction that means branches unconditionally.

D5 → BSA: Branch and save Return address

useful in branching to a portion of a program. which is called a subroutine. or procedure.

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

$PC \leftarrow AR$, $SC \leftarrow 0$

D6 → ISZ : Increment and skip if zero
we increment the word that is specified by effective address and then, check if that incremented value is zero

D6 ← ISZ : Increment and skip if zero

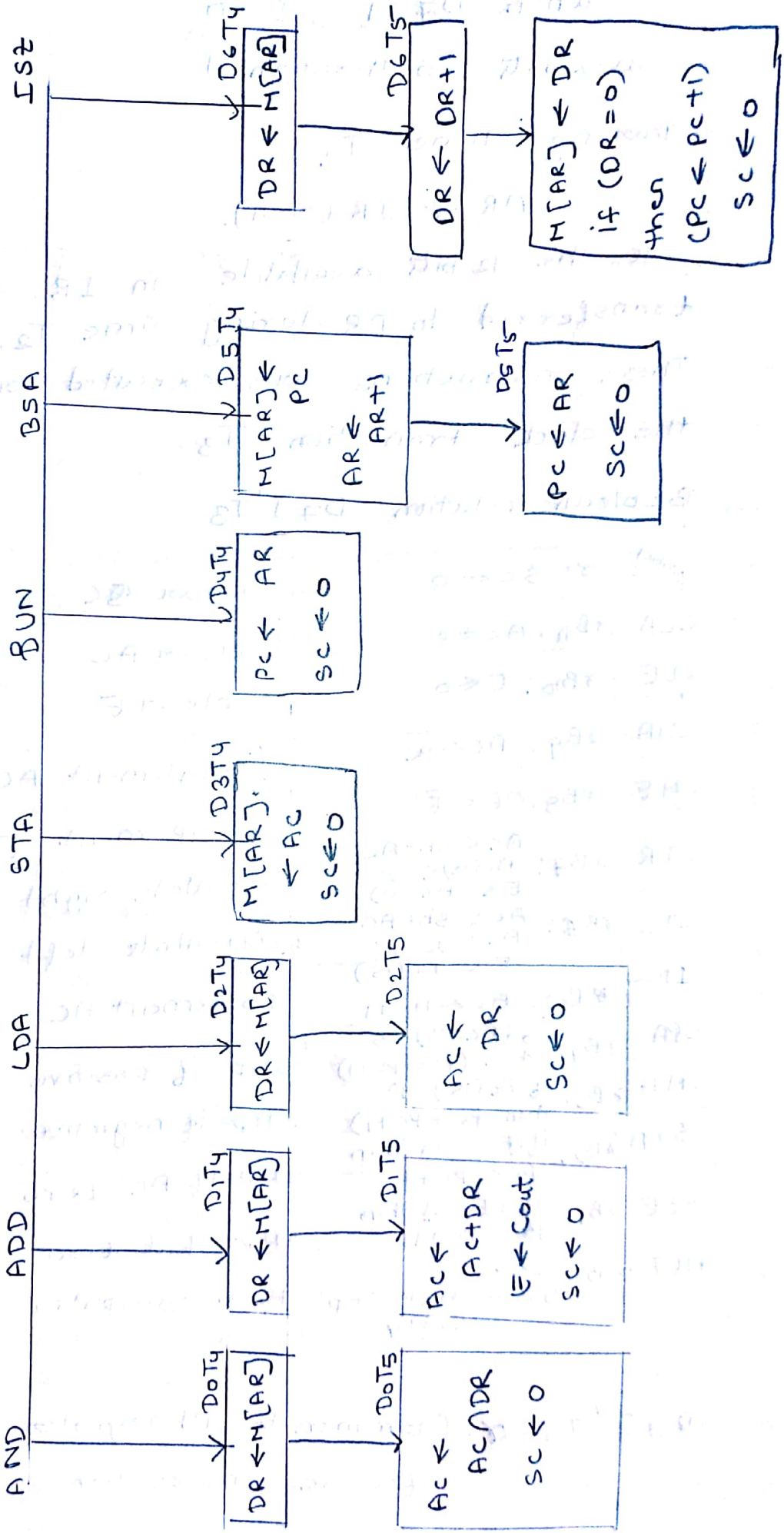
$DR \leftarrow M[AR]$

$DR \leftarrow DR+1$

$M[AR] \leftarrow DR$, if $(DR = 0)$
then $(PC \leftarrow PC+1)$,
$SC \leftarrow 0$

contents at the effective address are transfered to DR , then DR is incremented by 1 ; if the incremented part is 0 then PC is incremented by 1

Memory - reference Instruction



Register - Reference instruction

when $D_7 = 1$, $I = 0$

uses bits 0 through 11

During Time T_2

$AR \leftarrow IR(0-11)$.

ie the 12 bits available in IR are transferred to AR during Time T_2.

These instructions are executed with the clock transition T_3.

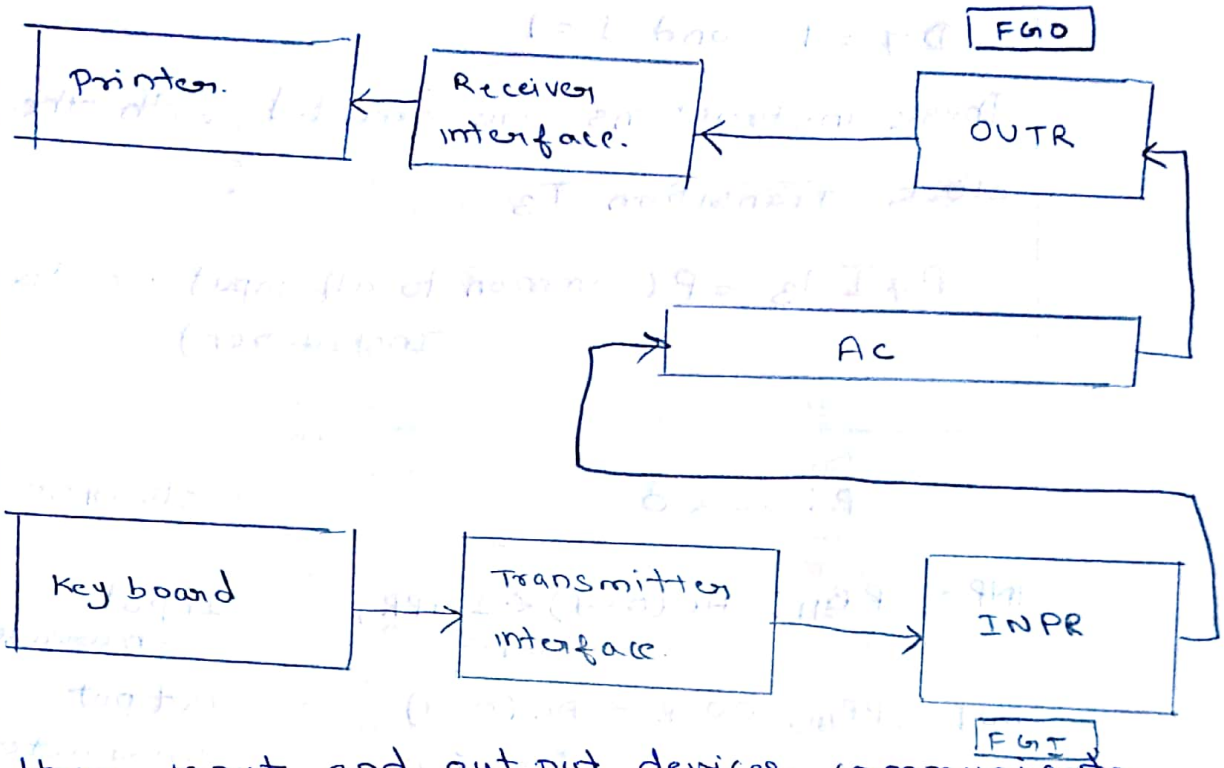
Boolean relation $D_7 I' T_3$

Symbol	r: Sc ← 0	clear SC
CLA	rB_{11}: Ac ← 0	clear AC
CLE	rB_{10}: E ← 0	clear E
CMA	rB_9: $Ac \leftarrow \overline{Ac}$	complement AC
CME	rB_8: $E \leftarrow \overline{E}$	complement E
CIR	rB_7: AC ← shr AC AC(15) ← E E ← AC(0)	circulate right
CIL	rB_6: AC ← shl AC AC(0) ← E E ← AC(15)	Circulate left
INC	rB_5: AC ← AC + 1	Increment AC
SPA	rB_4: if (AC(15) = 0 then (PC ← PC + 1)	SKip if positive.
SNA	rB_3: if AC(15) = 1 then PC ← PC + 1)	SKip if negative
SZA	rB_2: if (AC = 0) then PC ← PC + 1	SKP if AC tero
SZE	rB_1: if (E = 0) then PC ← PC + 1	SKP if E tero
HLT	rB_0 S ← 0 S is a start - stop flip flop	Halt computer

$D_7 I' T_3 = r$ (common to all register reference Instruction

Input - output and Interrupt.

input output configuration



How input and output devices communicate. with each other using certain interfaces, flip-flops and registers

Printer, keyboard → input - output terminals

Receiver interface, Transmitter interface → serial comm interface.

OUTR, INPR → computer registers.

FGO, FGI → flags or flip-flop

The INPR and OUTR are eight bits

The 1 bit input flag FGI is a control flip-flop

The flag bit is set to 1 when new information is available in the input device and is cleared to 0 when the information is accepted by the computer

If FGO is set to 1 the computer checks the flag bit ie information from AC to OUTR and FGO is cleared to 0.

Input - output instruction

Input output instruction have an operation code 1111

$D_7 = 1$ and $I = 1$

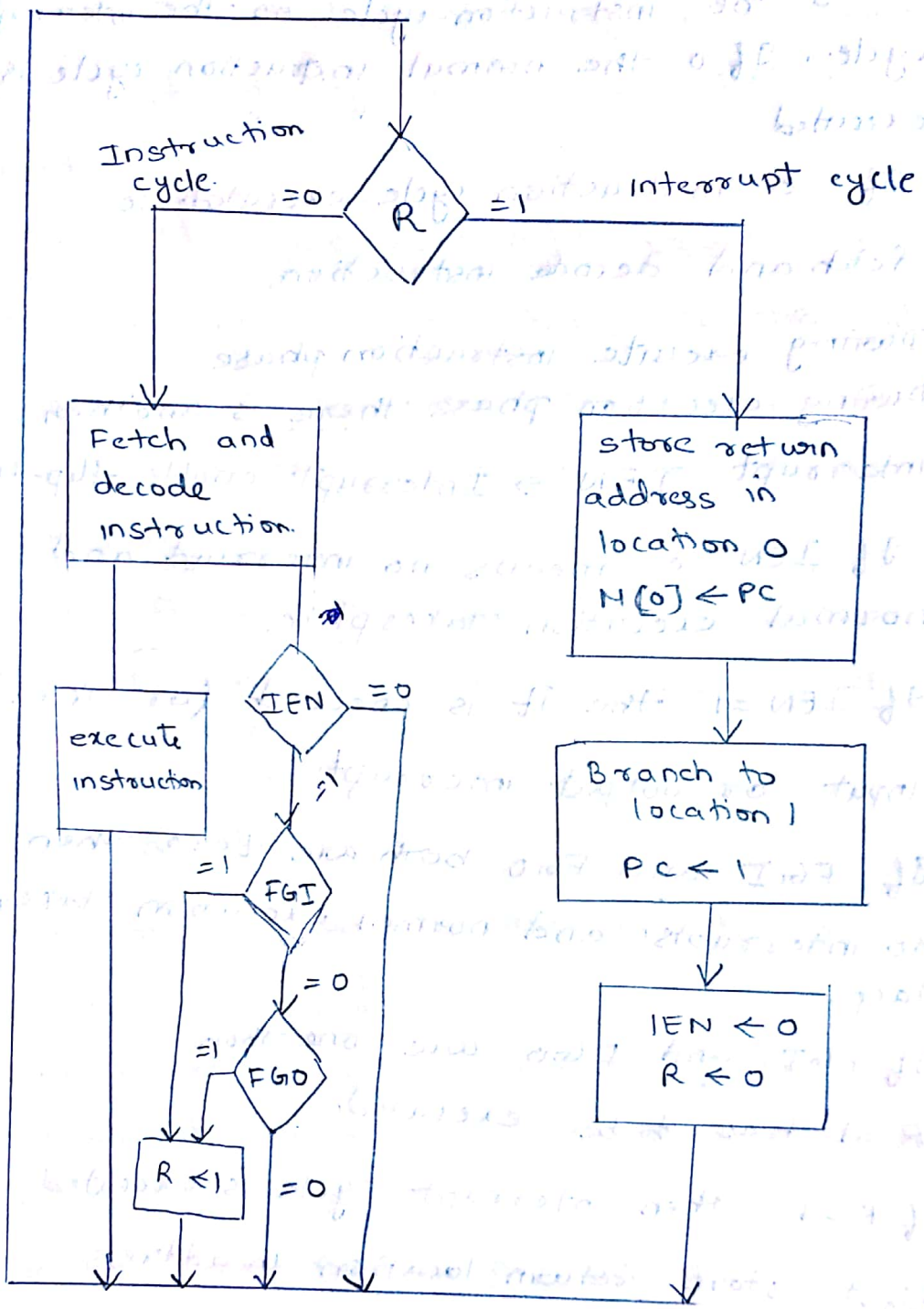
These instructions are executed with the clock transition T_3

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

	$P: SC \leftarrow 0$	clear SC
INP	PB_{11}: $AC(0-7) \leftarrow INPR$, $FGI \leftarrow 0$	Input character
OUT	PB_{10}: $OUTR \leftarrow AC(0-7)$, $FGO \leftarrow 0$	out put 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: $IEN \leftarrow 0$	Interrupt enable off

Program Interrupt

During an instruction cycle how an interrupt is handled.



Flow chart of interrupt cycle.

IEN ⇒ Interrupt enable flipflop

R ⇒ interrupt flip-flop

FGO ⇒ 1 bit output flag

FGI ⇒ 1 bit input flag

The value of R determines wheather it enters the instruction cycle or the interrupt cycle.. If 0 the normal instruction cycle is executed

If 0 instruction cycle executed. ie fetch and decode instruction.

~~During execute instruction phase~~
During execution phase there is another interrupt IEN → Interrupt enable flip-flop

If IEN = 0 means no interrupt and normal execution takes place.

If IEN = 1 the it is checked for the input or output interrupt

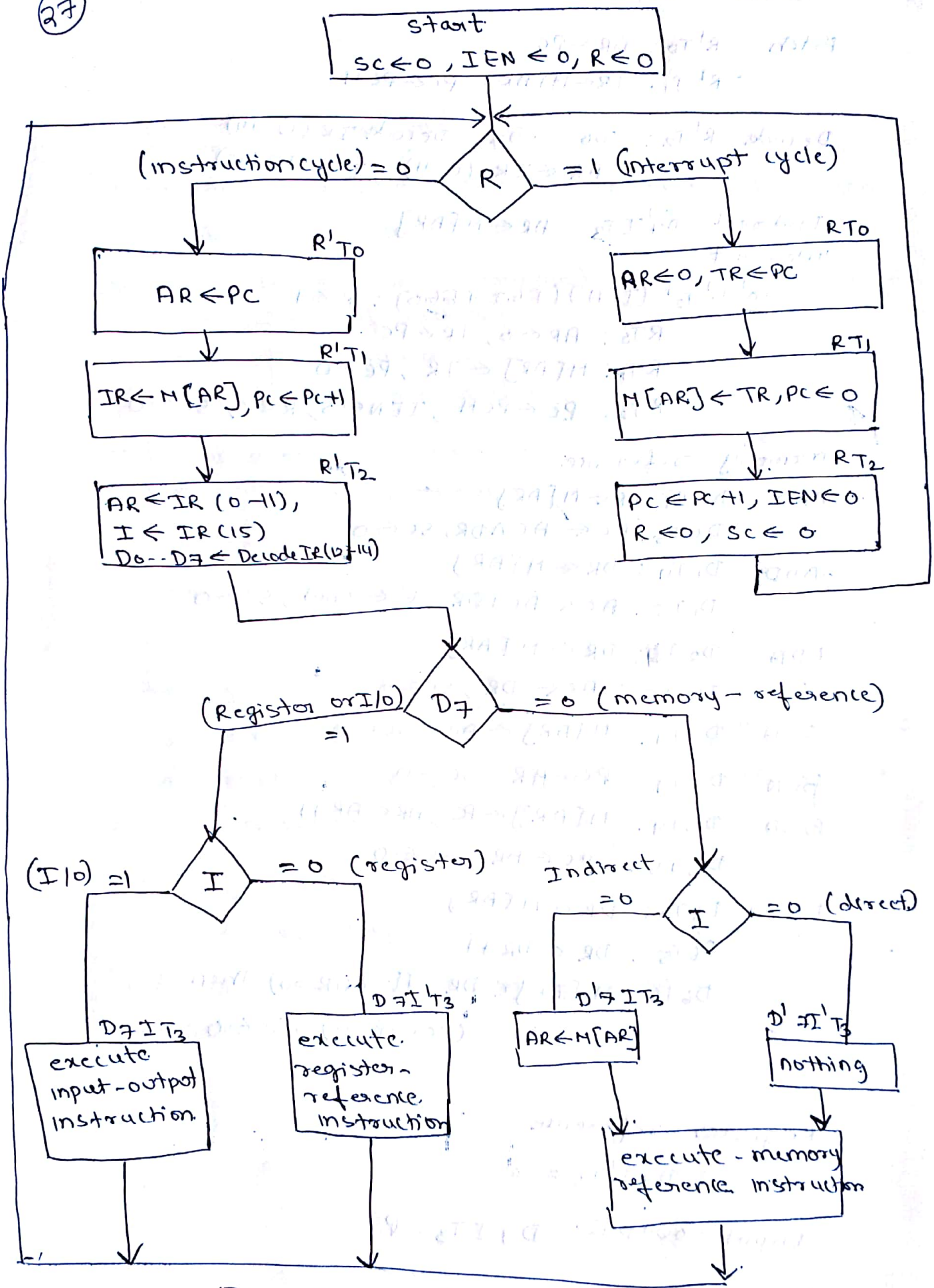
If FGI and FGO both are zero then no interrupts and normal execution taker place.

If FGI and FGO are one then R = 1 has to be executed.

If R = 1 then interrupt cycle is executed.

first store return location ~~it~~ address then branch ~~the~~ to Location and then both value IEN ← 0, R ← 0 set to zero

complete computer Description



Flow chart for computer operation

control Functions and microoperations for the Basic computer

Fetch $R'T_0$: $AR \leftarrow PC$

$R'T_1$: $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$

Decode $R'T_2$: $D_0 \cdots D_7 \leftarrow$ Decode $IR(12-14)$

$AR \leftarrow IR(0-11)$, $I \leftarrow IR(15)$

Indirect $D_7'IT_3$ $AR \leftarrow M[AR]$

interrupt

$T_0'T_1'T_2'(IEN)(FGI + FGO)$: $R \leftarrow 1$

RT_0 : $AR \leftarrow 0$, $TR \leftarrow PC$

RT_1 : $M[AR] \leftarrow TR$, $PC \leftarrow 0$

RT_2 : $PC \leftarrow PC+1$, $IEN \leftarrow 0$, $R \leftarrow 0$, $SC \leftarrow 0$

Memory-reference.

AND D_0T_4 : $DR \leftarrow M[AR]$

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

ADD D_1T_4 : $DR \leftarrow M[AR]$

D_1T_5 : $AC \leftarrow AC + DR$, $E \leftarrow$ cout, $SC \leftarrow 0$

LDA D_2T_4 : $DR \leftarrow M[AR]$

D_2T_5 : $AC \leftarrow DR$, $SC \leftarrow 0$

STA D_3T_4 : $M[AR] \leftarrow AC$, $SC \leftarrow 0$

BUN D_4T_4 : $PC \leftarrow AR$, $SC \leftarrow 0$

BSA D_5T_4 : $M[AR] \leftarrow PC$, $AR \leftarrow AR+1$

D_5T_5 : $PC \leftarrow AR$, $SC \leftarrow 0$

ISZ D_6T_4 : $DR \leftarrow M[AR]$

D_6T_5 : $DR \leftarrow DR+1$

D_6T_6 : $M[AR] \leftarrow DR$, if $(DR=0)$ then $(PC \leftarrow PC+1)$, $SC \leftarrow 0$

Register-reference

$D_7I'T_3 = r$

Input-output $D_7IT_3 = p$

Microprogrammed CONTROL.

CONTROL MEMORY.

The control variables can be represented by a string of 1's and 0's called a control word.

The control words can be programmed to perform various operations on the system

A control unit whose binary control variables are stored in memory is called a microprogrammed control unit.

A computer that employs a ~~micropro~~ microprogrammed control unit will have two separate memories

1) A main memory.
2) A control memory

Main memory ⇒ it is available to the user for storing the programs. The contents of main memory may alter when the data are manipulated. and every time that the program is changed. The user's program in main memory consists of machine instructions and data.

control memory ⇒ The control memory holds a fixed microprogram that cannot be altered by the occasional user. The microprogram consists of microinstruction that specify

various internal control signals for execution of register microoperations

Each machine instruction initiates a series of microinstructions in control memory. These microinstructions generate the microoperations to fetch the instruction from main memory; to evaluate the effective address, to execute the operation specified by the instruction, and to ~~repeat~~ return control to the fetch phase in order to repeat the cycle for the next instruction

CONTROL signals

Group of bits used to select path in multiplexers, decoders, arithmetic logic units.

control variable.

Binary variable that specify micro-operation.

control word

It is a string of control variables (0's and 1's) occupying a word in control memory.

ie memory contains control word.

micro instructions

control word stored in control memory.

specify control signals for execution of micro-operation

It contains a control word and a sequencing word.

control word → It contain all the control information required for one clock cycle.

sequencing word → It contains information needed to decide the next microinstruction address

control memory

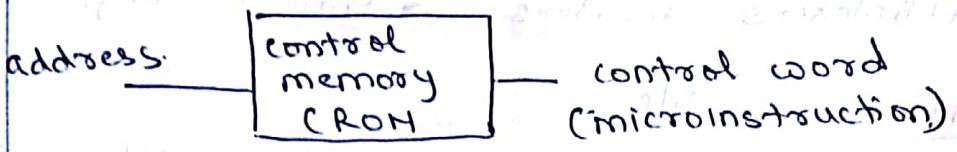
microprogram

A program stored in control memory that generates all the control signals required to execute the instruction set correctly

It consists of micro instruction

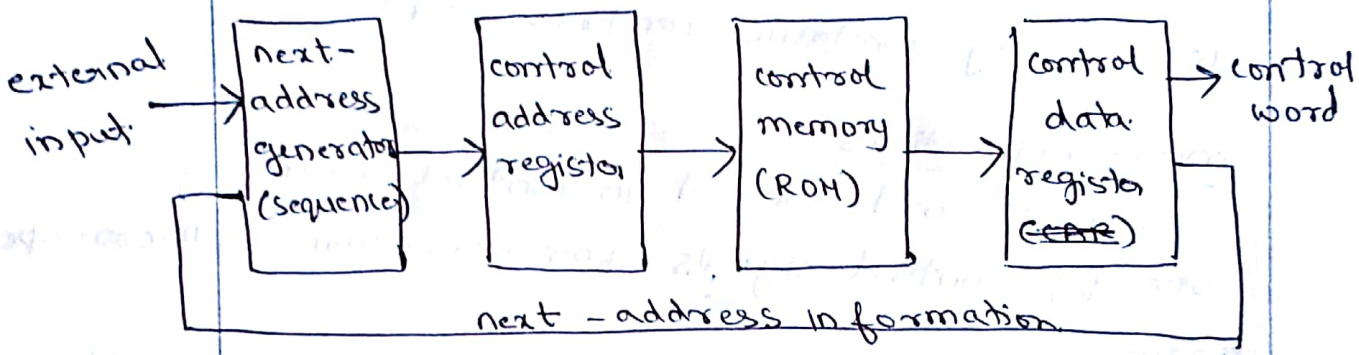
CONTROL memory

Read only memory.



contains word in ROM at given address

Each computer instruction initiates series of microinstruction (microprogram) in control memory



control memory →

- contains microprograms (sets of microinstruction)
- microinstruction contains
  * Bits initiate microoperations
  * Bits determine address of next microinstruction

control address Register →

specifies address of next microinstruction

next address generator (microprogram sequencer)

→ Determines address sequencers for control memory microprogram sequencer.

function.

→ increments CAR by one.

→ Transfer external address into CAR.

→ load initial address into CAR to start control operation.

control data Register

Holds microinstructions read from control memory. Allows execution of micro-operations specified by control memory.

~~simultaneously generates next microinstruction~~

Address sequencing

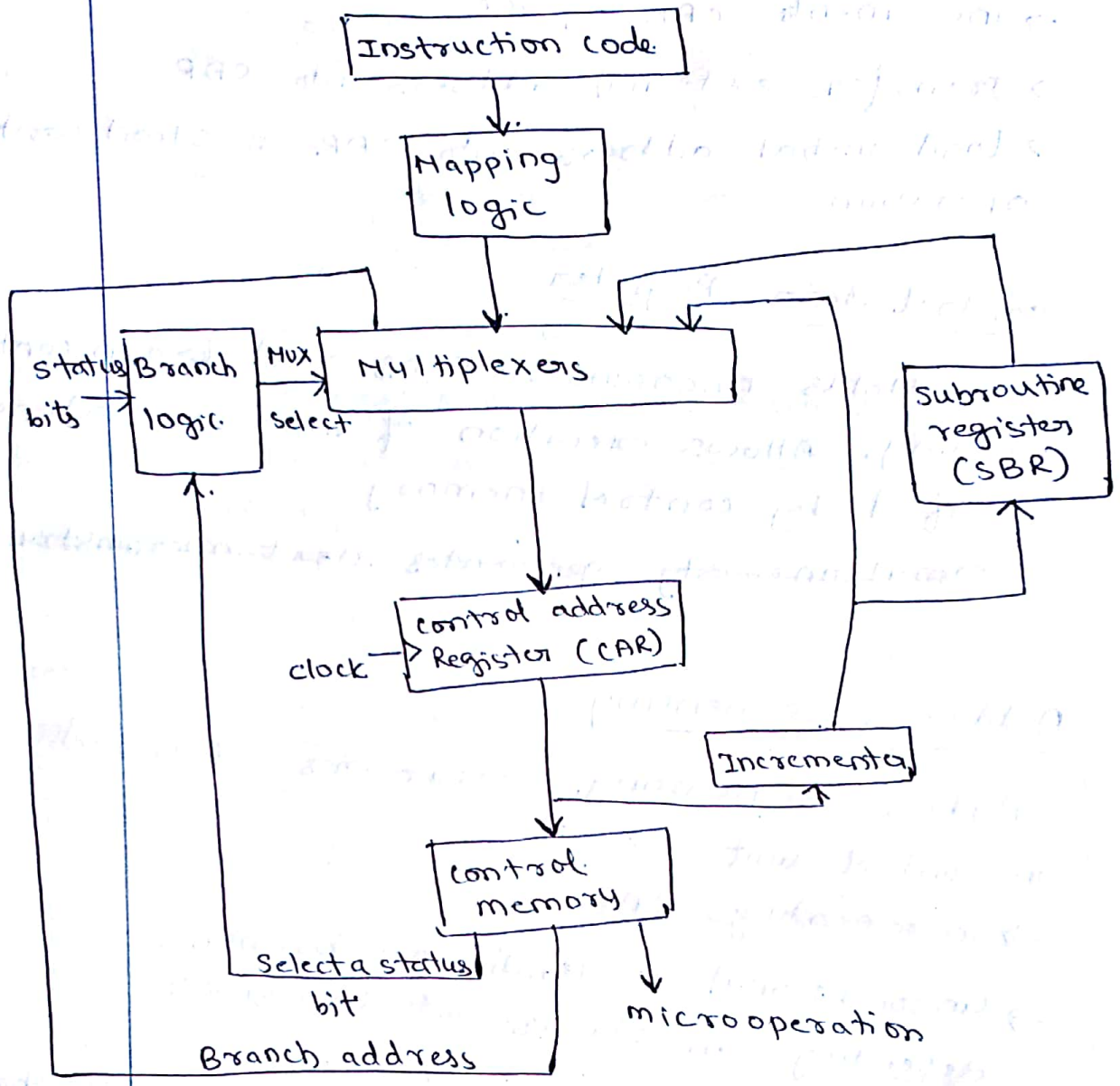
Address sequencing capabilities required in control unit

→ incrementing CAR

→ unconditional or conditional branch, depending on status bit conditions

→ A mapping process from the bits of the instruction to an address for control memory

→ A facility for subroutine call and return



Selection of address for control memory

conditional branching

Branching from one routine to another depends on status bit conditions.

status bits provide parameters info such as
- carry-out of adder
- sign bit of number.
- mode bits of instruction.

info in status bits can be tested and actions initiated based on their conditions 1 or 0

unconditional branch
- fix value of status bit to 1

Mapping of Instruction

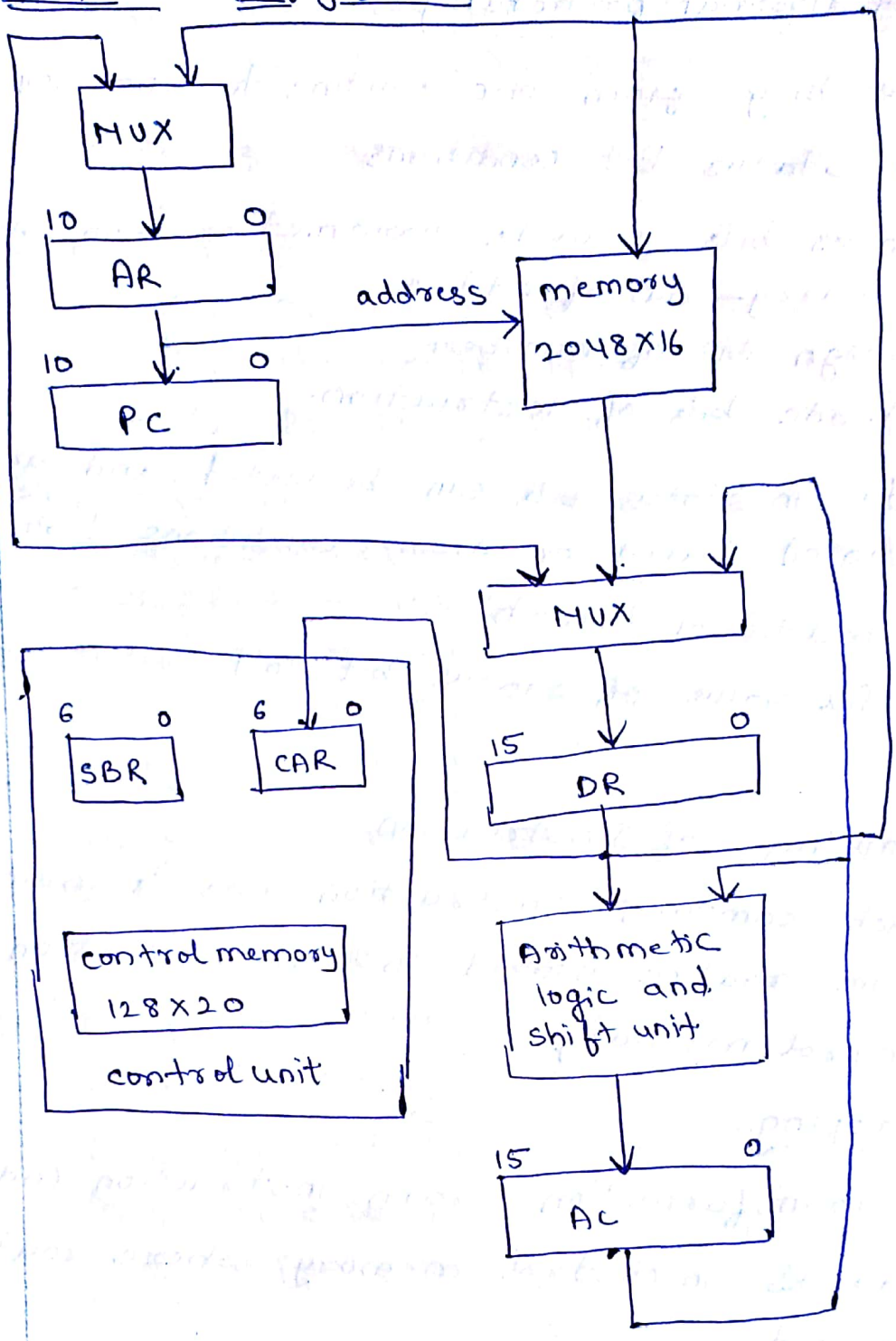
Each computer Instruction has its own microprogram routine stored in a given location of the control memory.

mapping

Transformation from instruction code bits to address in control memory where routine is located.

Microprogram Example

computer configuration



consiste of two memory unit

main memory → storing instruction and data

control memory → storing the microprogram

Total six Register.

Four associated with processor.

PC, AR, DR, AC

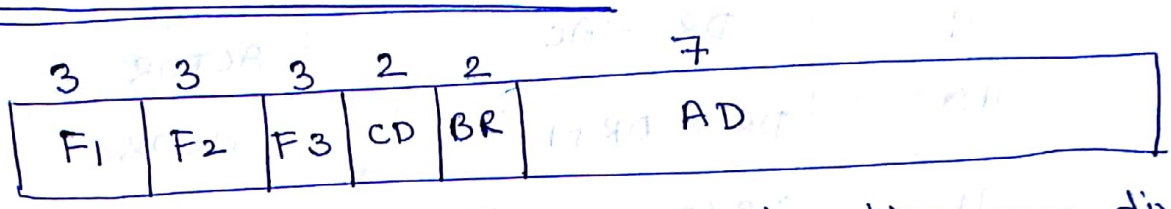
Two registers with control unit.

CAR → control address register

SBR → subroutine register

Transfer of information is done through multiplexer

Microinstruction Format



The 20 bits of the microinstruction are divided into four functional parts

F_1, F_2, F_3 → microoperation fields

CD → condition for branching

BR → Branch field.

AD → Address field

F_1	microoperation	symbol
000	None	NOP
001	$AC \leftarrow AC + DR$	ADD
010	$AC \leftarrow 0$	CLRAC
011	$AC \leftarrow AC + 1$	INC AC
100	$AC \leftarrow DR$	DRTAC
101	$AR \leftarrow DR(0-10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE

F2	Microoperation	symbol
000	None	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR + 1$	INCDR
111	$DR(0-10) \leftarrow PC$	PCTDR

F3	Microoperation	symbol
000	None	NOP
001	$AC \leftarrow AC \oplus DR$	XOR
010	$AC \leftarrow \overline{AC}$	COM
011	$AC \leftarrow shl\,AC$	SHL
100	$AC \leftarrow shr\,AC$	SHR
101	$PC \leftarrow PC + 1$	INCPC
110	$PC \leftarrow AR$	ARTPC
111	Reserved	

CD (condition) field.

CD	condition	symbol	comments
00	Always = 1	U	unconditional branch
01	DR(15)	I	Indirect address Bit
10	AC(15)	S	sign bit of AC
11	AC = 0	Z	Zero value in AC

BR (Branch) field used in conjuction with the address field AD, to choose the address of the next micro instruction

BR	symbol	Function
00	JMP	CAR ← AD if condition = 1 CAR ← CAR+1 if condition = 0
01	CALL	CAR ← AD, SBR ← CAR+1 if condition = 1 CAR ← CAR +1 if condition = 0
10	RET	CAR ← SBR (Return from subroutine)
11	MAP	CAR (2-5) ← DR (11-14), CAR (0,1,6) ← 0

~~Design of control unit~~

Symbolic Microinstructions

sample format

label: micro-ops CD BR AD

→ label may be empty or may specify symbolic address terminated with colon

→ micro-ops consists of 1, 2, or 3 symbols separated by commas.

→ CD one of {U, I, S, Z}

U: Unconditional Branch

I: Indirect address Bit

S: sign of AC

Z: zero value in AC.

→ BR one of {JMP, CALL, RET, MAP}

→ AD one of {Symbolic address, NEXT, empty}

41 Fetch ROUTINE

→ Fetch routine.

→ Read Instruction from memory.
Decode instruction and update PC

Microinstruction for Fetch routine.

$AR \leftarrow PC$

$DR \leftarrow M[AR], PC \leftarrow PC+1$

$AR \leftarrow DR(0-10), CAR(2-5) \leftarrow DR(11-14),$
$CAR(0,1,6) \leftarrow 0$

Symbolic microprogram for fetch routine.

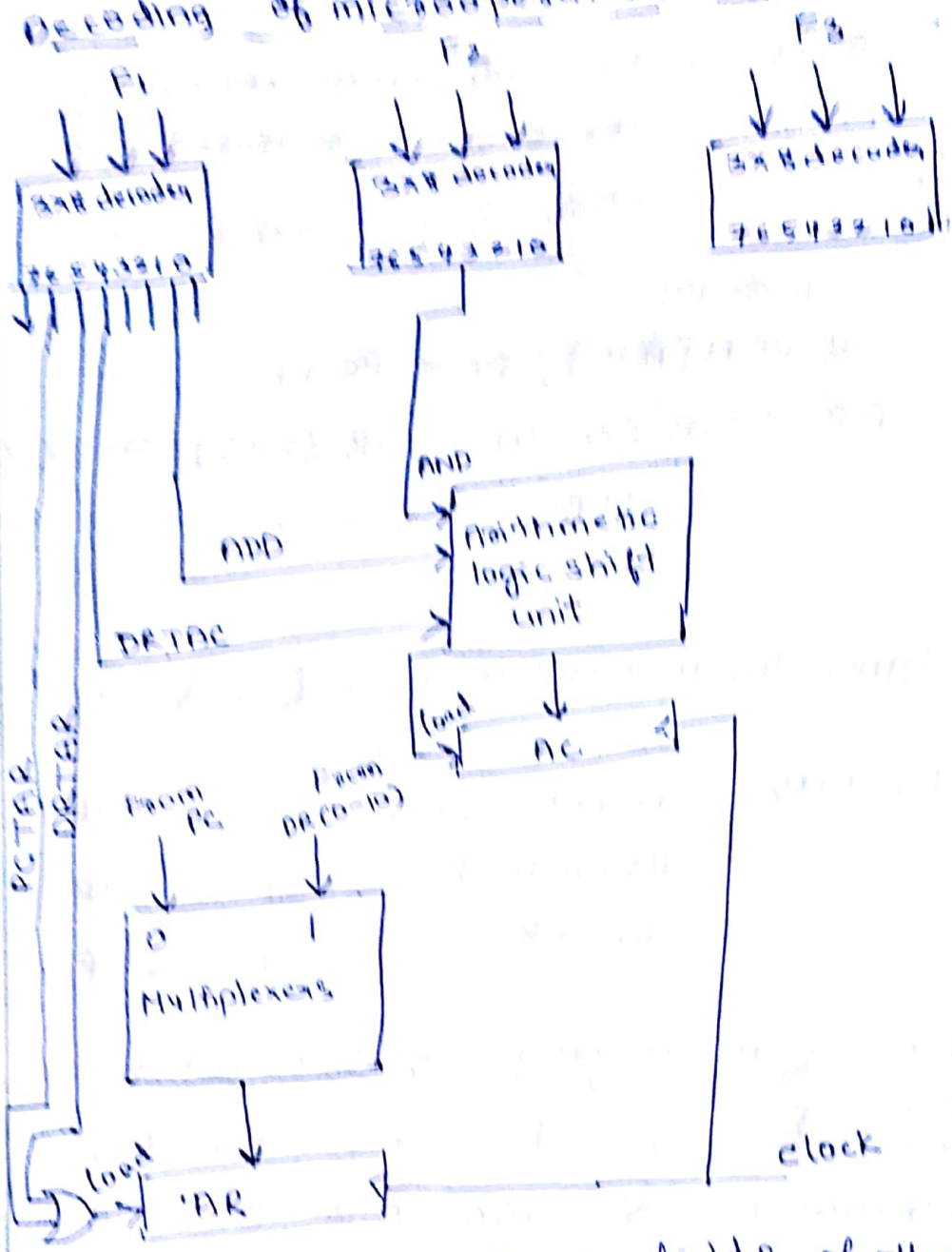
```
        ORG 64
FETCH:  PCTAR          U   JMP   NEXT
        READ, INCPC    U   JMP   NEXT
        DRTAR          U   JMP
```

Binary microprogram for fetch Routine.

Binary address	F_1	F_2	F_3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	000	000	00	11	0000000

Design of control unit

Decoding of microoperation fields.



Each of the three three fields of the microinstruction are decoded with a 3x8 decoder to provide eight outputs. Each of these output must be connected to the proper circuits to initiate the corresponding microoperation. The o/p of the decoders that are associated with an AC operation must be connected to arithmetic logic unit).