LECTURE -

Content : Introduction to control unit

Control Design: The digital computer generally can be divided into two parts—

1- Data Processing unit: A collection of functional units capable of performing certain operations on data.

2- Control unit: It issues control signals to the data processing unit to perform operations on data.

OR

The control unit decides the sequence in which certain sets of microoperations are to be performed to carry out any particular operation on data.

The two approaches to design control unit are

1- Hardwired control

2. Microprogrammed control

Fundamental Concept :

Register Transfer : The process of data transfer between registers and common bus are shown in figure.

Each register has its individual input and output gating and these gates are controlled by corresponding control signals.

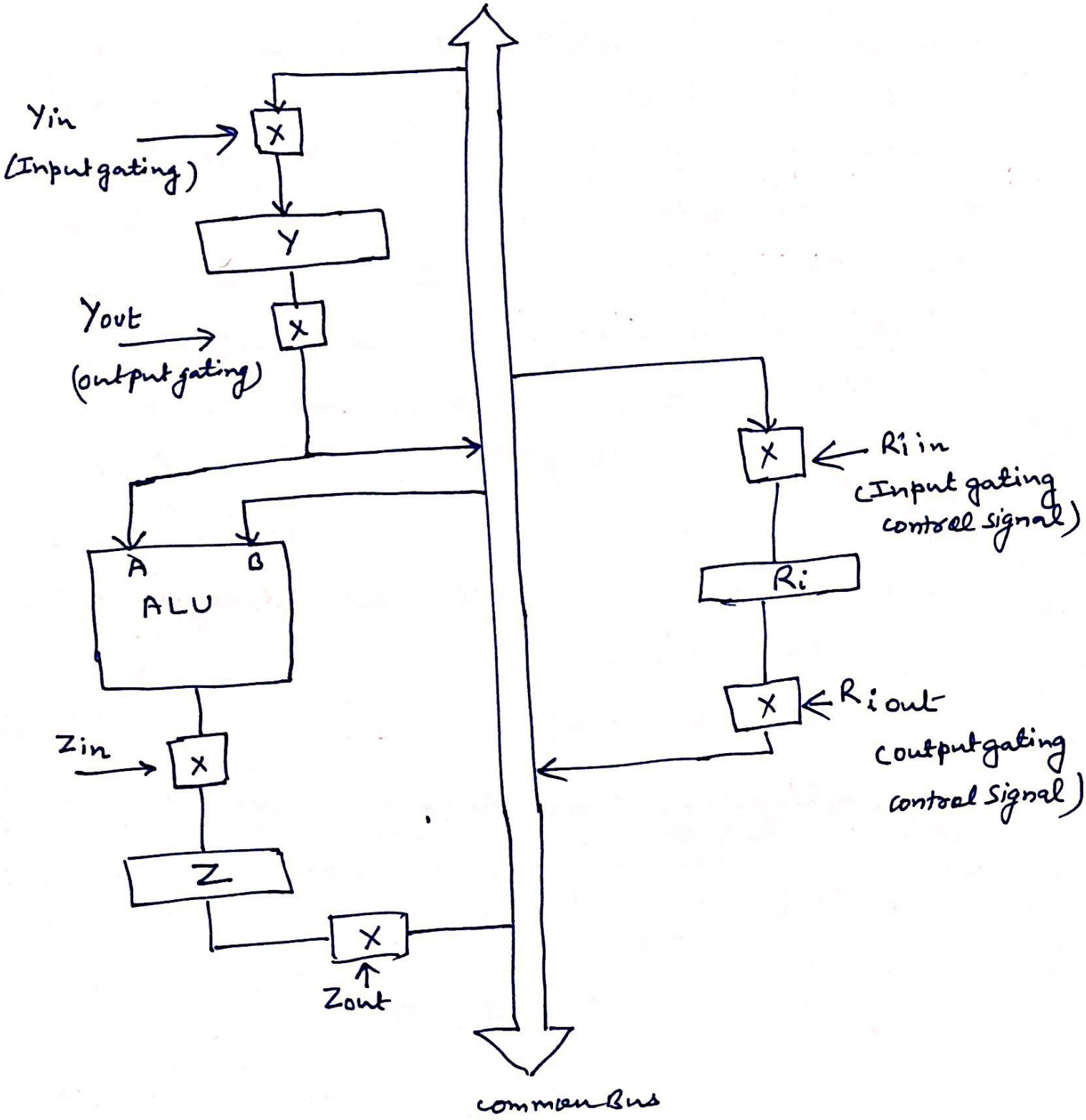
Each register Ri is controlled by two signals :-

Riin :- when it is set to 1, the data available on the common data bus is loaded into register Ri.

Riout :- when it is set to 1, the data available ~~available~~ in register is placed on the common data bus.

- The input and output gates are electronic switches which can be controlled by the control signals. When the signal is

  { 1 - switch makes the connection ON.
  { 0 - switch makes the connection OFF.



Input and output gating for the registers

Performing of Arithmetic and Logic operations

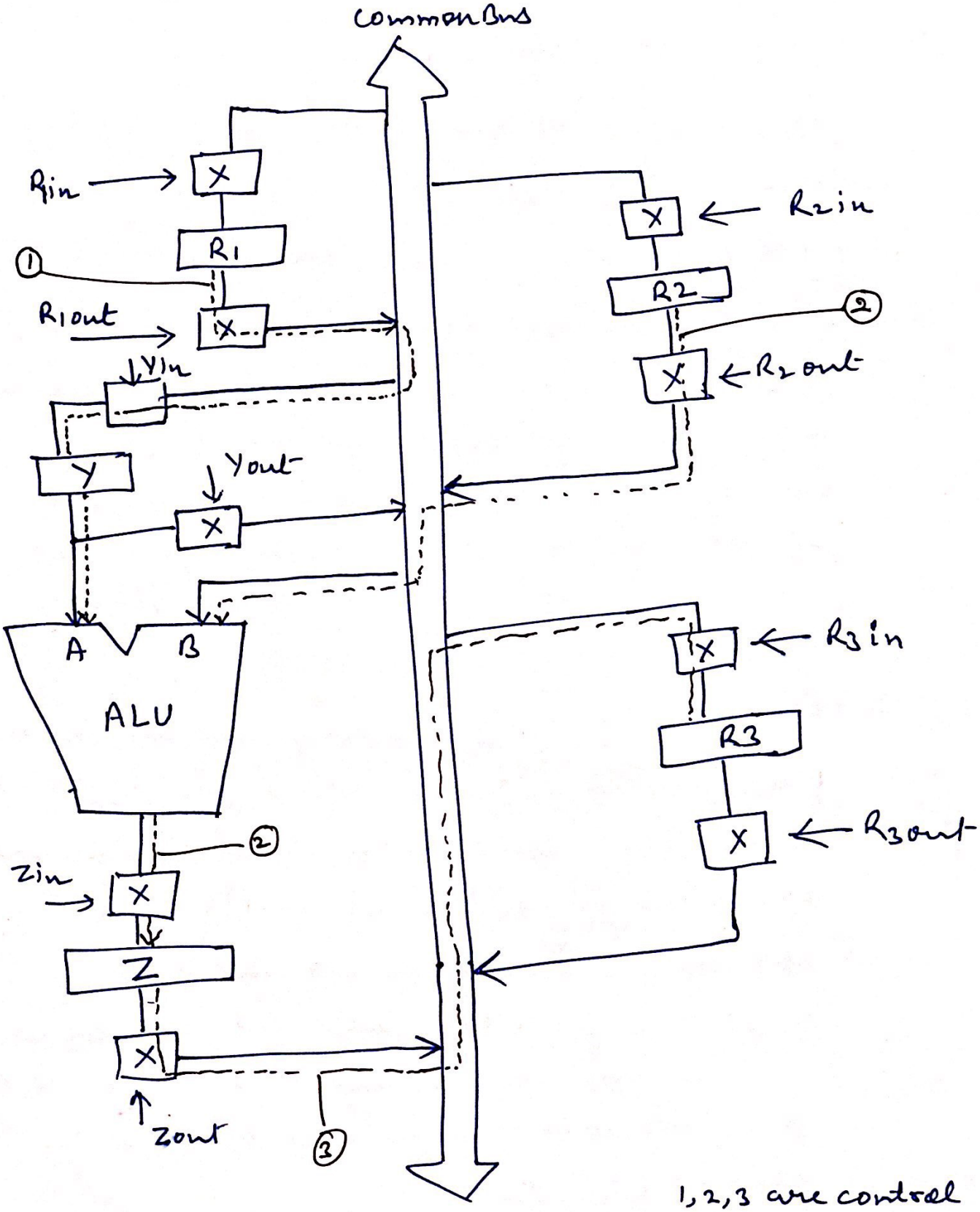
An ALU performs arithmetic and logic operations. It is a combinational circuit that has no internal memory storage. ALU has two inputs and one output comes from register R_1 and the other input comes from Register R_2. To perform addition, the two operands have to be made available at the two inputs of the ALU simultaneously. The result of ALU is loaded temporarily into any one register.

exp: The I/O gating for arithmetic and logic operations for

$$R3 \leftarrow R1 + R2$$

The sequence of operations is as follows:

control step	operation
1	R_1out, Yin
2	R_2out, Zin
3	Zout, R_3in



I/O gating for ALU operation

Fetching a word from memory

To fetch a word from memory, the CPU has to perform

1- opcode fetch cycle : It gives the operation code of fetching a word from memory to the CPU. i.e It specifies the address of memory location where the information is stored. The CPU transfers the address of the required word of information to the address register (AR) which is connected to the address lines of the memory bus.

2- operand fetch cycle : During this the address from AR is transferred to the memory, meanwhile, the CPU activates the read signal of the memory to indicate that a read operation is needed. As a result, memory copies data from the addressed register on the data bus. The CPU then reads this data from the data register and loads it in the specified register.

MFC (Memory functions completed) is used to indicate memory transfer. When it is set to 1 it indicates that the contents of specified location have been read and are available on the data bus.

For Example :

Assume that the address of the memory location to be accessed is in register R1 and that the memory content is to be loaded into register R2 ~~and~~. The following sequence of operation is done —

1- AR ← [R1] - specified address available in R1 is placed in address register.

2- Read — Read required content from memory.

3. Wait for MFC signal - Gives the information about memory transfer completed and required data is placed in the data register.

4. R2 ← DR - content of data register is placed in the register R2.

Storing a word in Memory :

To store a word in memory CPU has to perform

1- opcode fetch cycle : It gives the operation code to the CPU.

2- operand write cycle : As per the operation code, CPU invokes the operand write cycle

after the address is loaded into the address register (AR). The data word to be written is 1st loaded into data register (DR), at the same time the write command is issued to memory from the CPU.

After receiving the write command, the data word is written in memory at the specified location.

For Example:

~~Assume that the address of the memory location to be accessed is in register R1 and that the memory content is to be loaded into register R2. This is done by the following sequence of operation.~~

1- AR ← [R1] : Address is placed in address register from register R1

2. DR ← [R2], write : Data is placed in data register and send the write signal to memory.

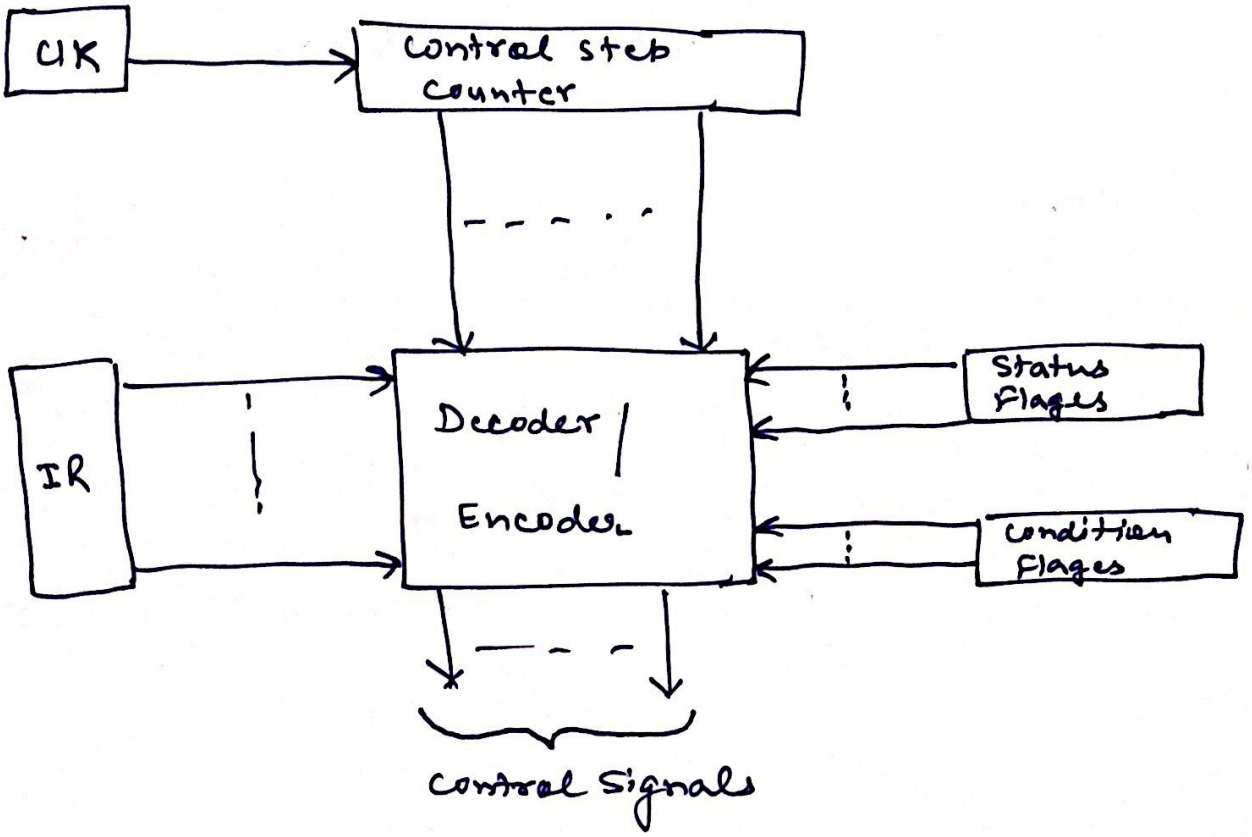
3. Wait for MFC : MFC is set to 1, the data is written to specified address.

Let us assume that the data word to be stored in the memory is in register R2 and that the memory address is in R1, the write operation requires the following sequence :-

LECTURE

Content: Hardwired control unit system.

Hardwired control :- It is a controller as a sequential logic circuit or a finite state machine that generates a sequence of control signals in response to the externally supplied instructions.



· General Block diagram of hardwired control

Methods of Hardwired control design :

There are four simplified and systemetic method for the design of hardwired controllers.

1- State-table method or one-hot method

2- Delay element method

3- Sequence-counter method

4- PLA Method.

Advantage: Fast, No memory used (Economic), uses less area.

Disadvantage: No Flexible, difficult to handle, complex instructions.

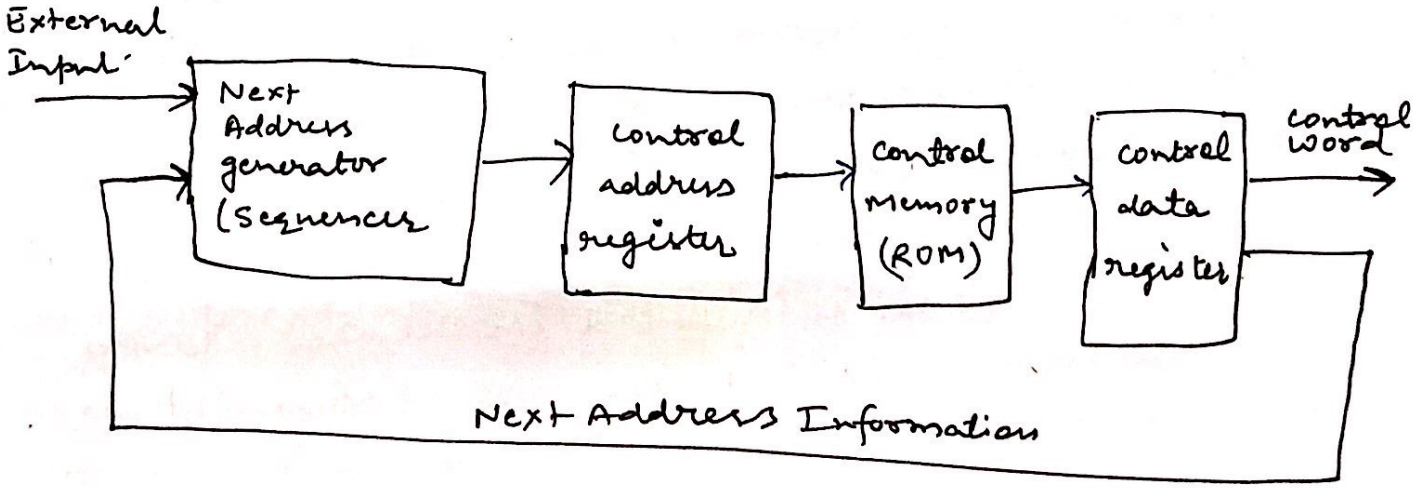
LECTURE

content : Microprogrammed control unit.

Microprogrammed control : A control unit whose binary control variables are stored in the memory (control Memory) is called as microprogrammed control.

control Memory : The memory (RAM or ROM) where control word is stored is called control memory.

control word : The control variables at any given time can be represented by a string of 1's and 0's called as control word. In this word individual bit represents the various control signals.



Block diagram of Microprogrammed control

The general configuration of microprogrammed control unit is demonstrated in the block diagram. The control memory is assumed to be ROM, within which all control information is permanently stored. The control memory address register specifies the address of the microinstruction, and the control data register holds the microinstruction read from memory.

The microinstruction contains a control word that specifies one or more microinstructions for the dataprocessor. Once these operations are executed the control must determine the next address. The location of the next microinstruction may be the one next in sequence, or it may be located somewhere else in the control memory.

For this reason it is necessary to use some bits of the present microinstruction to control the generation of the address of the next microinstruction. The next address may also be a function of external input conditions. While the microoperations are being executed, the next address is computed in the next address generator circuit and then transferred into the control address register to read the next microinstruction. Thus a microinstruction contains bits for initiating microoperations in the data processor part and bits that determine the address sequence for the control memory.

The control variable at any time can be represented by a group of binary bits called control word. control word is stored in control memory which issues the control signals when enabled

Memory control word is written for each micro operation, and these control words are stored In a serial ascending memory location and accessed serially. The output of the control memory provides the required control signals. The storage of control word in ROM is often referred to as firmware.

Advantage of Microprogrammed control :

1- Control unit design becomes simple.

2- Control functions are implemented in software.

3- Design process is orderly and systematic.

4- More flexible.

Disadvantages of Microprogrammed control :

1- Slow in comparison to hardwired control unit - because time required to access the microinstructions from CM.

2- Costly due to the control memory and its access circuitry.

Comparison between Hardwired and Microprogrammed control

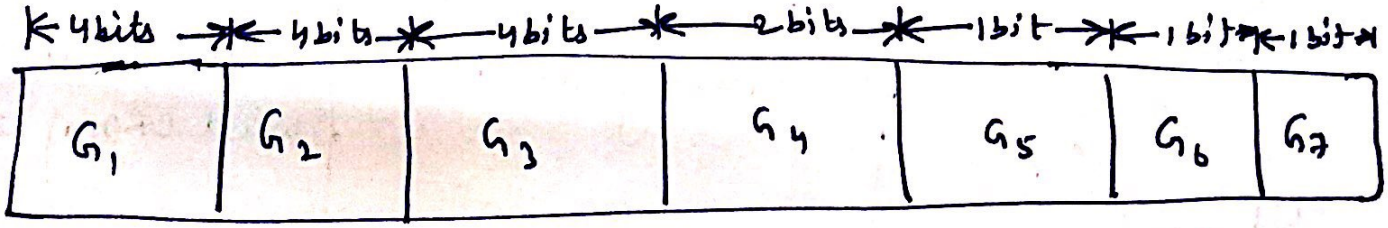
Characteristics	Hardwired control	Microprogrammed control
Speed	Fast	Slow
Implementation	Hardware	Software
Flexibility	No flexibility	More flexibility
Ability to handle large / complex instruction sets	Some what difficult	Easier
Ability to support operating system & diagonastic features	Very difficult	Easy
Design Process	Difficult for more operation	Easy
Memory	No Memory used	Control memory used (RAM / ROM)
Chip area efficiency	Uses least area	Uses more area

Microinstruction : A microinstruction contains the control signals and sequencing information.

Microprogram : A sequence of microinstruction is called a microprogram.

Grouping of control signals : A simple way to structure micro instructions is to ~~design~~ assign one bit position to each control signal that is required in the CPU. However this scheme results a very long microinstruction

- The solution to above problem is to group the control ~~sig~~ signals.
- Grouping Techniques are used to reduce the number of bits in the microinstruction with grouping technique. the 46 control signals can be grouped in 7 different groups.



Partial format for field encoded microinstruction

G1 (4 Bits) : IN Grouping

0000	No transfer
0001	IR_{IN}
0010	PC_{IN}
0011	DR_{IN}
0100	AR_{IN}
0101	A_{IN}
0110	$Temp_{IN}$
0111	R_{IN}
1000	R_{3IN}
1001	R_{4IN}
1010	$R5_{IN}$
1011	SP_{IN}

G2 (4 Bits) : OUT grouping

0000	No transfer
0001	IR_{OUT}
0010	PC_{OUT}
0011	DR_{OUT}
0100	AR_{OUT}
0101	A_{OUT}
0110	$TEMP_{OUT}$
0111	R_{OUT}
1000	SP_{OUT}

G3 (4 bits) : ALU functions

0000	ADD	ALU functions
0001	SUB	
⋮	⋮	
1111	XOR	

G4 (2 bits) : RD/WR control signals

00 : No operation
01 : Read
10 : write

G5 (1 bit) : A register

0 : No operation
1 : Clear A

G6 (1 bit) : carry

0 : carry into ALU
1 : carry into ALU

G7 : (1 bit) : operation

0 : continue operation
1 : End.

LECTURE

Content: Concept of Horizontal and vertical Microprogramming

Grouping Techniques: There are two techniques used to group the control signals:

1- Vertical organization

2- Horizontal organization

1- Vertical Organization: Highly encoded schemes that use compact code to specify only a small number of control functions in each micro instructions are referred to as vertical organization.

- More micro instructions are needed to perform the desired control functions.

2- Horizontal Organization: The minimally encoded scheme in which many resources can be controlled with single micro instructions is called a horizontal organization

- used when
  - Higher operating speed is designed
  - Allows parallel use of resources.

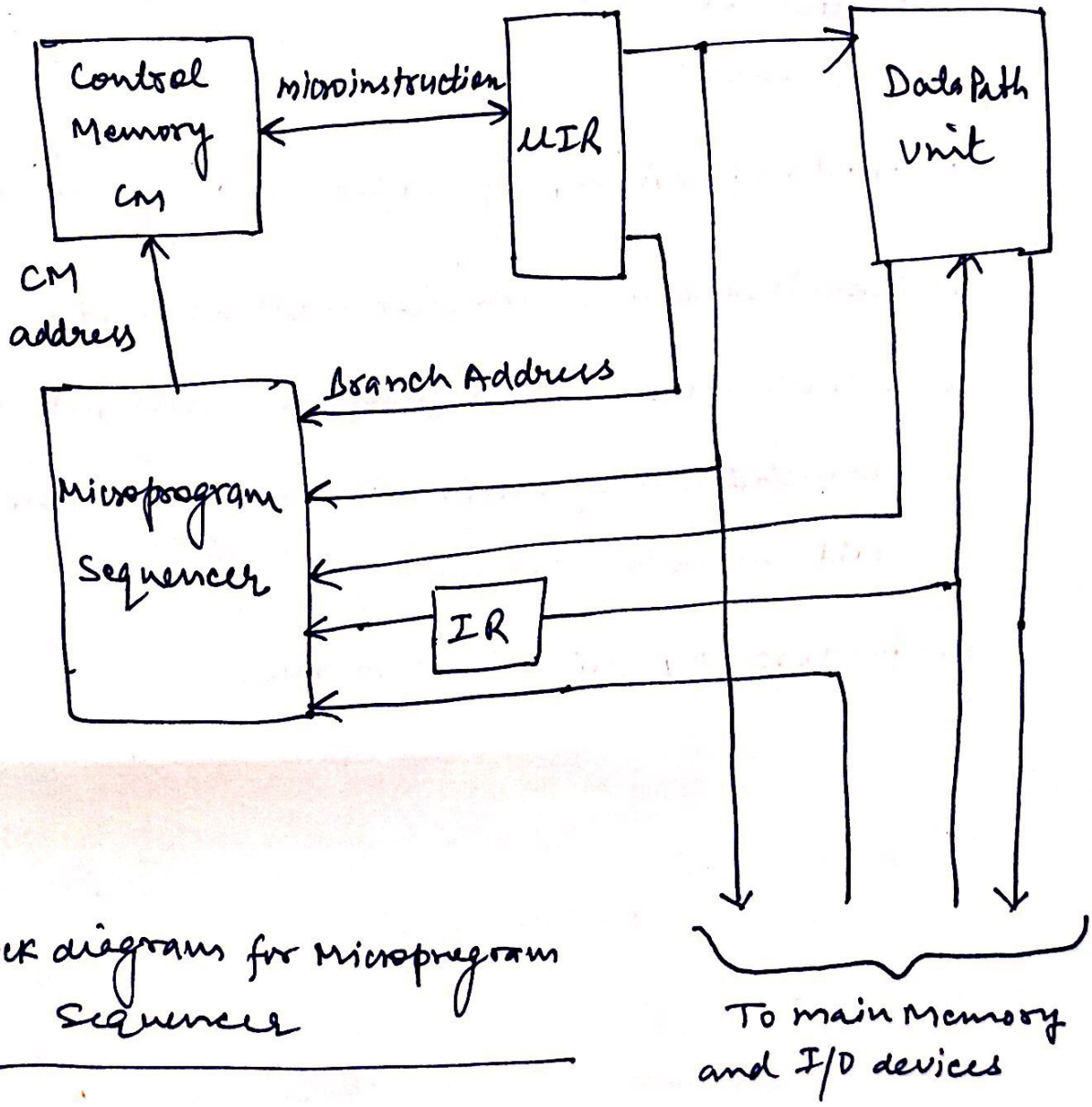
Comparison between Horizontal and vertical organization

Horizontal Organization	vertical organization
1- Lengthy formats	short formats
2. Ability to express a high degree of parallelism	Limited ability to express parallel microoperations
3. considerable encoding of the control information	Highly encoding of the control information
4. Useful when higher operating speed is desired	slower operating speed

LECTURE

Content : Microprogramme Sequencer

Microprogram Sequencer : The basic components of a microprogrammed control unit are the control memory and the circuit that selects the next address. The address selection part is called microprogram sequencer. It is a general purpose building block for microprogram control unit.



Block diagram for Microprogram Sequencer

Routine : Microinstructions 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 ~~microprog~~ microoperations that executes the instruction.

The hardware that controls the address sequencing of the control memory must be capable of sequencing the microinstructions within routine are and be able to branch from one routine to another.

Methods of microprogram Sequencing are —

1- Micro instruction with next address field.

2- Branch address modification using Bit-ORing

3- Branch address modification using condition variables

4- Wide branch addressing.

5- Prefetching microinstruction.

LECTURE -

Content : Introduction to Instruction types & formats

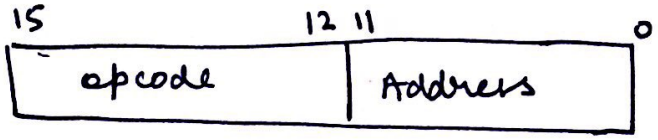
Instruction code :

A Computer instruction is a binary code that specifies a sequence of microoperations for the computer. Instruction codes together with data are stored in memory. The computer reads each instruction from memory and places it in a control register. The control ~~unit~~ interprets the binary code of the instruction and procedes to execute it by issuing a sequence of microoperations.

An instruction code is a group of bits that instruct the computer to perform a specific operation. The instruction code of a computer instruction is divided into parts as follows:

1- **operation code** - it specifies the operation to to be performed.

2- **Address part** - it specifies the ~~operation~~ register or memory word where the operands are to be found.



Instruction format

if 4096 words

$= 2^{12}$ - 12 bits to specify the address part

we need 16 bits
- 12 bits - address
- 4 → opcode → 2^4 = 16 operations distinct

Now the basis of the format -

→ Immediate operand

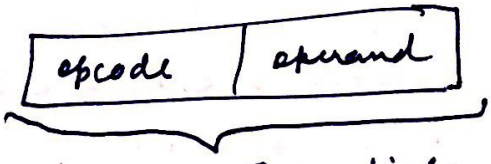
→ Direct address

→ Indirect address

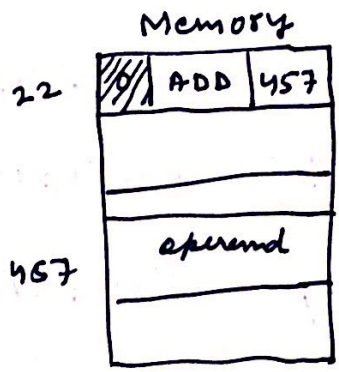
} Effective Address - where operand is present -

This is not necessary that this address denotes the address.

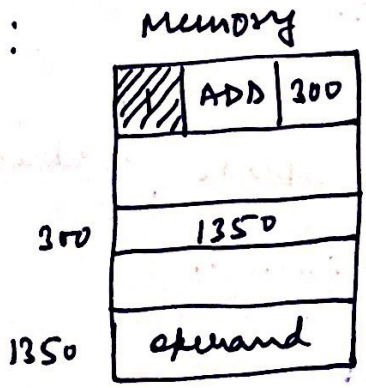
It can act as

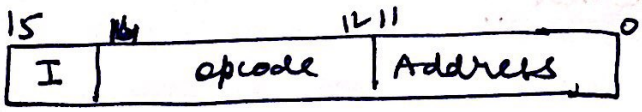


Direct Address :



Indirect address :





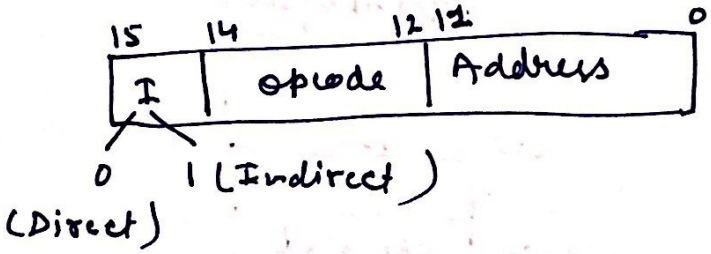
if I = 0 - direct

I = 1 - Indirect

Instruction Type: There are three types of Instruction

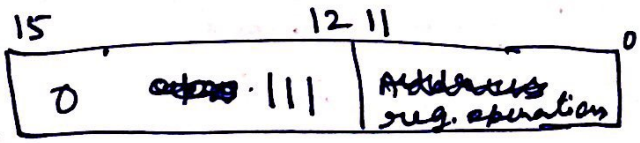
1. Memory-Reference Instruction
2. Register-Reference Instruction
3. Input/output Instruction

Memory-reference:



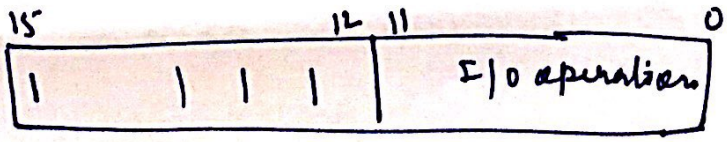
opcode = 000 - 110

Register-reference:



opcode = 111
I = 0

Input/output Instruction



opcode = 111
I = 1

LECTURE-

Content: Instruction cycles & sub cycles

Instruction cycle:

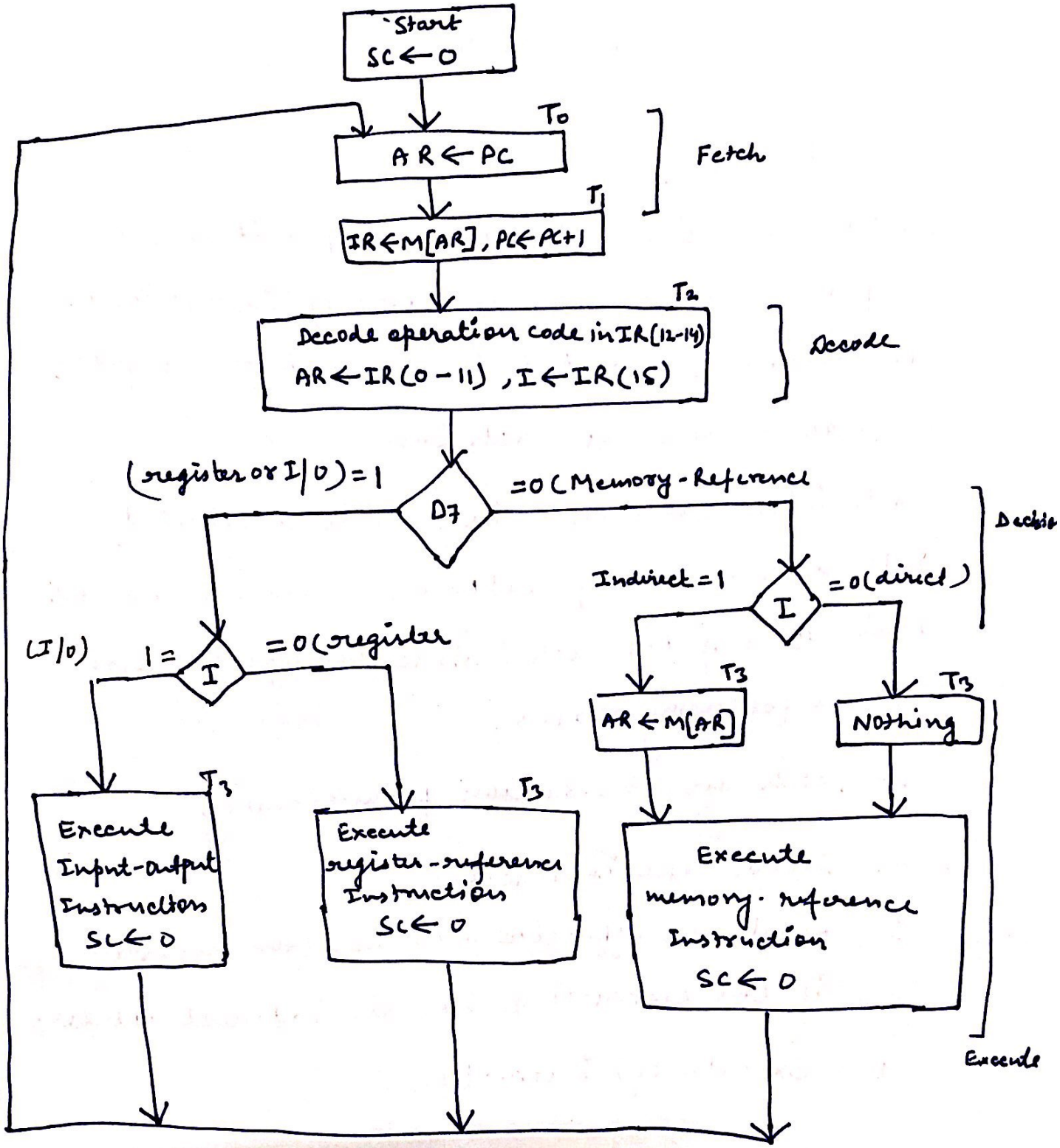
A program residing in the memory unit of the computer consists of a sequence of instructions. The program is executed in the computer by going through a cycle for each instruction.

Each instruction cycle in turn is subdivided into a sequence of subcycles or phases. In the basic computer each instruction cycle consists of the following phases:

1- Fetch an instruction from memory
2- Decode the instruction
3- Read the effective address from memory if the instruction has an indirect address.
4. Execute the instruction.

Upon the completion of step-4, the control goes back to step 1 to fetch, decode, and execute the next instruction.

This process continues indefinitely unless a HALT instruction is encountered.



Flow-chart for Instruction cycle

LECTURE-

Memory-Reference Instruction:

There are seven memory reference instruction (D_i for $i = 0, 1, \ldots 6$) in a computer system. These instruction performs operation on the data stored in Accumulator and the effective address. The effective address is in the address register AR placed there during time signal T_2 when $I = 0$ or T_3 when $I = 1$. The execution of memory reference instruction starts with time signal T_4.

The various memory-reference Instructions are:

① **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 microoperation that execute this instruction are:

$D_0 T_4$: $DR \leftarrow M[AR]$

$D_0 T_5$: $AC \leftarrow AC \wedge DR$, $SC \leftarrow 0$

② ADD to AC: This instruction adds the content of the memory word specified by the effective address to the value of AC. The sum is transferred into AC and the output carry Cout is transferred to the E (extended accumulator) flip-flop. The microoperations needed to execute this instruction are:

$D_1 T_4$: $DR \leftarrow M[AR]$

$D_1 T_5$: $AC \leftarrow AC + DR$, $E \leftarrow Cout$, $SC \leftarrow 0$

① LDA: Load to AC: This instruction transfers the memory word specified by the effective address to AC. The microoperations needed to execute this instruction are:

$D_2 T_4$: $DR \leftarrow M[AR]$

$D_2 T_5$: $AC \leftarrow DR$, $SC \leftarrow 0$

③ STA: Store to AC: This instruction stores the content of AC into memory word specified by the effective address.

$D_3 T_4$: $M[AR] \leftarrow AC$, $SC \leftarrow 0$

⑤ BUN : Branch Unconditionally :

This instruction transfers the program to the instruction specified by the effective address. The BUN instruction allows the programmer to specify an instruction out of sequence and we say that the program branches (or jumps) unconditionally.

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

⑥ BSA : Branch & Save Return Address :

This instruction is useful for branching to a portion of the program called a subroutine or procedure. When executed, the BSA instruction stores the address of the next instruction in sequence into a memory location specified by the effective address.

The effective address plus one is then transferred to PC to serve as the address of the first instruction in the subroutine.

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

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

⑦ ISZ : Increment and Skip if zero :

This instruction increments the word specified by the effective address, and if the incremented value is equal to 0, PC is incremented by one.

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 Instructions:

Register-reference instructions are recognized by the control when $D_7 = 1$ and $I = 0$. These instructions use bits 0 through 11 of the instruction code to specify one of 12 instructions. These 12 bits are available in IR(0-11). They were also transferred to AR during time T2.

$D_7 I' T_3 = r$ (common to all register-reference Inst.)

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

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

Input/output Instructions

Input and output instructions are needed for transferring information to and from AC register.

$D_7IT_3 = p$ (common to all input-output instructions)

$IR(i) = B_i$ (bit in IR(6-11) that specifies the inst.)

	$p: SC \leftarrow 0$	clear SC
INP	$pB_{11}: AC(0-7) \leftarrow INPR,$ $FGI \leftarrow 0$	Input character
OUT	$pB_{10}: OUTR \leftarrow AC(0-7),$ $FGO \leftarrow 0$	output character
SKI	pB_9 : If (FGI = 1) then (PC ← PC+1)	skip on input flag
SKO	pB_8 : if (FGO = 1) then (PC ← PC+1)	skip on output flag
ION	pB_7 : IEN ← 1	Interrupt enable on
IOF	pB_6 : IEN ← 0	Interrupt enable off

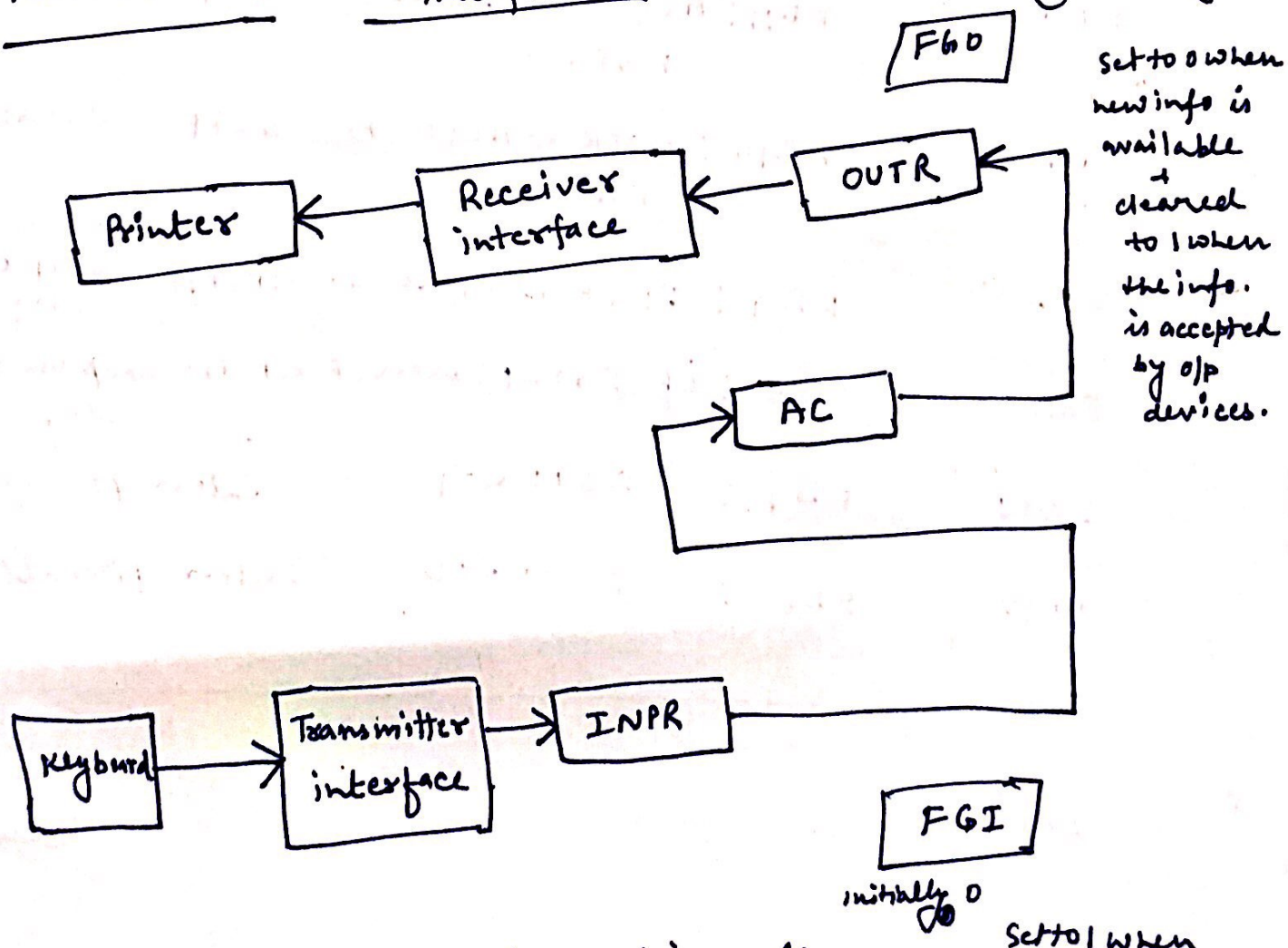
Input/Output Configuration :

The terminal sends and receives serial information. Each quantity of information has eight bits of an alphanumeric code. The serial information from the keyboard is shifted into the input register INPR. The serial information for the printer is stored in the output register OUTR.

Input-output terminal

Serial communication interface

computer register & Flip flops



Input-output configuration