Lecture 1

Monday — Date 15/07/19 — Page

Introduction to Computer Architecture and Organization:

Computer Architecture	Computer Organization
(1) A bit higher level	(1) Often called microarchitecture (low level)
(2) Programmer view (i.e. Programmer has to be aware of which instruction set used)	(2) Transparent from programmer (ex. a programmer does not worry much how addition is implemented in hardware)
(3) Logical components (Instruction set, Addressing modes, Data types, Cache Optimization)	(3) Physical components (circuit design, Adders, Signals, Peripherals)
(4) What to do? (Instruction set)	(4) How to do? (implementation of the architecture)

* The architecture indicates its hardware whereas the organization reveals its performance.
* For designing a computer, its architecture is fixed first and then its organization is decided.

Architecture → (1) interface between hardware and software.
(2) Abstract model and is programmer's view in terms of instructions, addressing modes and registers.
(3) Describe what computer does
(4) While describing computer system architecture is considered first.
(5) It deals with high level design issues
e.g. is there a multiplication instruction??

Organization → (1) deals with components of connection in a system.
(2) enchrons the realization of architecture.

3. describes how computer does a task.
4. organization is done on the basis of architecture.
5. deals with low level design issues.

e.g. Is there a multiplication unit or is it done by repeated addition?

Evolution of Computer from 1st Generation to Pentium and PowerPC:

↳ ENIAC was the first computer system designed in the 1940s.

1st Generation:
Size large with 18,000 buzzing electronic switches called vacuum tubes.

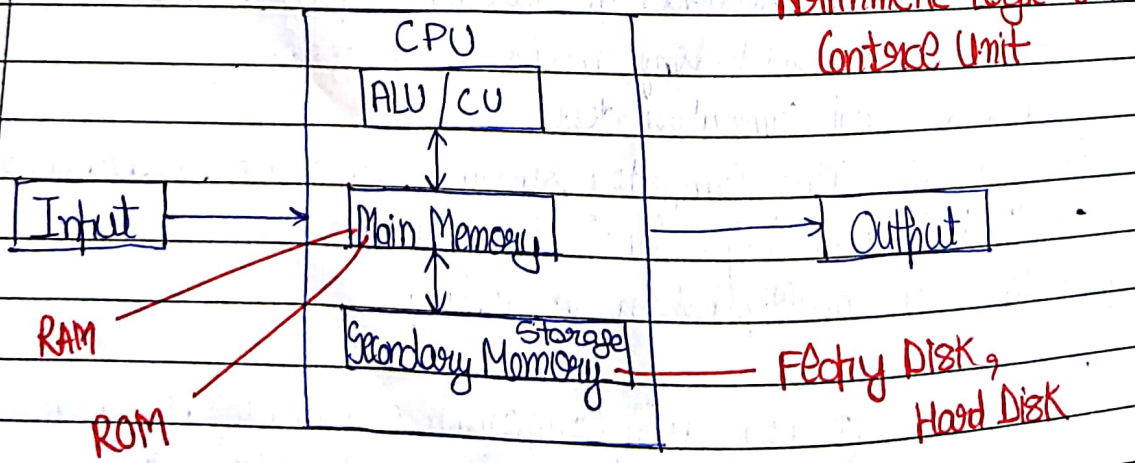
Second Generation:
In 1947, transistors, which were the fraction the size of the vacuum tubes and consumed less power, but still the complex circuits were not easy to handle.

Third Generation:
In 1958, Integrated Circuits, small size

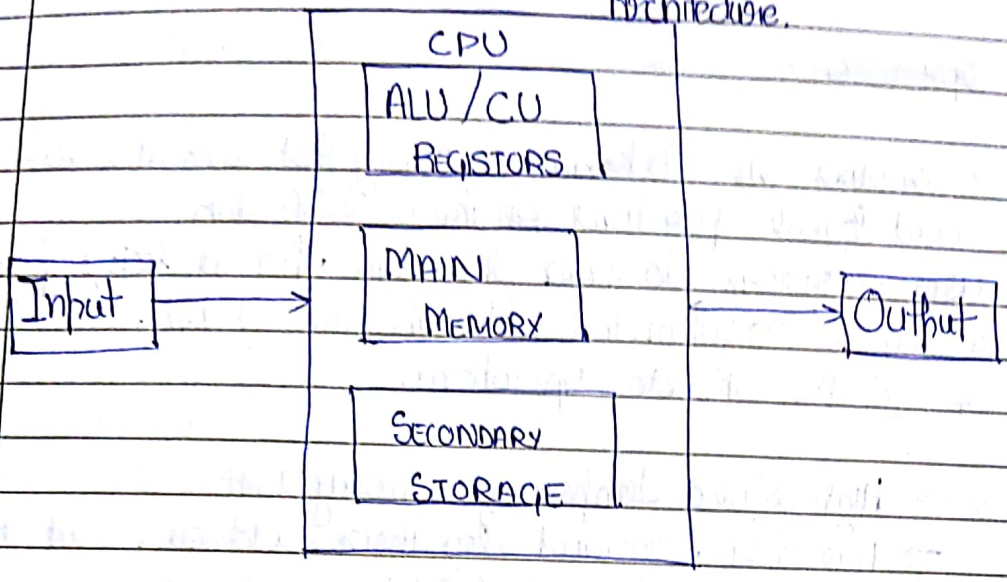
Fourth Generation:
Microprocessor e.g. Intel, Dual Core, Pentinum IV etc.

Fifth Generation:
AI (Artificial Intelligence) Based e.g. Alexa.

Basic Components of Computer:



VON-NEUMAN ARCHITECTURE :- Today's Computer Based on Von-Neuman Architecture.



* John- Von Neuman in 1945.
* This Computer Architecture design consist of Control Unit (CU), Arithmetic Logic Unit (ALU), Registors, Main Memory and Secondary Storage and Inputs/Outputs.

Input Units → Computer accepts coded information through input unit by user. It is used to give computer important info. e.g. Keyboard, Mouse etc.

Output Units → The output sends the processed results to the user e.g. on monitor, printer etc.

CPU (Control Processing Unit) → Together interpret and execute instructions in assembly language.

1. CPU transfers information (instructions) and input data from main memory to registors i.e. internal memory.
2. The CPU execute the program in the stored sequence.
3. When necessary, CPU transfers output data from registors to main memory.

* CPU → brain of Computer.
  ↙ ↘
  IC Microprocessor

* ALU → Contains the electronic circuitry that executes all arithmetic and logical operations on the available data.
  ALU uses registers to hold the data that is being processed.
  Used to calculate arithmetic (add, sub, mul & div) and logical (>, <, +, AND, OR, etc) operations.

* Registers → High speed temporary memory unit.
  → Are not referenced by their address, but directly accessed and manipulated by the CPU during execution.
  → Used for memory allocation., to store data.

* CU → It organises the processing of data and instructions.
  → To fetch the instruction stored in the devices involved in it and acc. generate control signals.

Microprocessor → PCB (Printed Circuit Board) used in calculator, digital system etc.

* Memory Unit → It holds data and instructions,
  ↙ ↘
  Primary Secondary → used to store data and instructions permanently i.g. harddisk, CDs, DVDs etc.
  ↓
  Store data and instructions during execution
  ↙ ↘
  RAM ROM

RAM (Random Access Memory): volatile memory. It provides temporary storage for data and instructions. It directly provides the required information to the processor.

1. Static RAM
2. Dynamic RAM

ROM (Read - Only Memory): (non-volatile) It is used for storing standard processing programs that permanently reside in the computer. It can only read not written.

1. Programmable ROM (PROM)
2. Erasable Programmable ROM (EPROM).
3. Electrically Erasable Programmable ROM (EEPROM).

There are two types of computer:

1. Fixed Program Computer → These function is very specific and they cannot or could not be programmed.
   e.g. Calculator
2. Stored Program Computer → These can be program to carry out many diff. task, application are stored on them. that is why they are named so.

Types of Registers:

1. MBR (Memory Buffer Registers): It holds the data that is being transfered to and from memory. It uses in Input/Output of Data.
   → data (Address) → load to MAR and give data.
2. MAR (Memory Address Registers): It holds the memory location of the data that needs to be access
3. Accumulator (AC): Where intermediate (Properly known as data) Arithmetic & logical results are stored.

Date 16/07/19
Page

Tuesday

4. Programme Counter → It contains the address of next instruction to be executed.

Lecture - 2

BUSES

Address Bus	Data Bus	Control Bus
(b/w Processor and Main Memory)	(b/w I/O Devices and Processor)	(Control signals from CPU to all the all the parts of Computer)

Buses are the means by which data is transmitted from one part of the computer to another, connecting all major internal components to the CPU and memory.

1. Address Bus carries the address of the data between Processor and memory.
2. Data Bus carries the data between the processor, memory unit and Input/Output devices.
3. Control Bus carries the control signals/commands from the CPU in order to control and coordinate all the activities within the computer.

Flynn's Classification of Computers

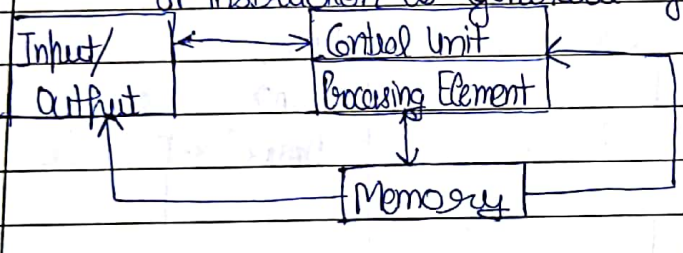
INSTRUCTION STREAM	DATA STREAM

1. Instruction Stream: It is the sequence of instructions read from memory, constitutes an instruction stream.
(Read)

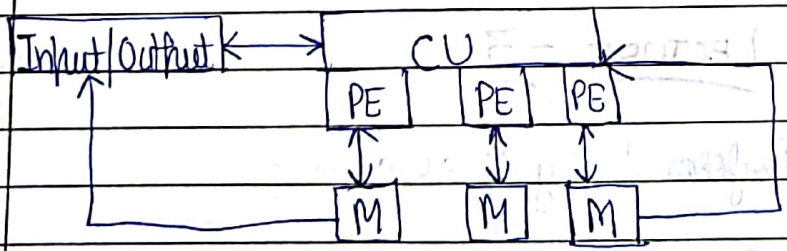
2. Data Stream: Its operation perform on data in the processor, constitutes a data stream.
(Process)

- SISD (Single Instruction Single Data)
- SIMD (Single Instruction Multiple Data)
- MISD (Multiple Instruction Single Data)
- MIMD (Multiple Instructions Multiple Data)

1. SISD: A single Processor Computer (Uni-Processor) in which single stream of instruction is generated from program.

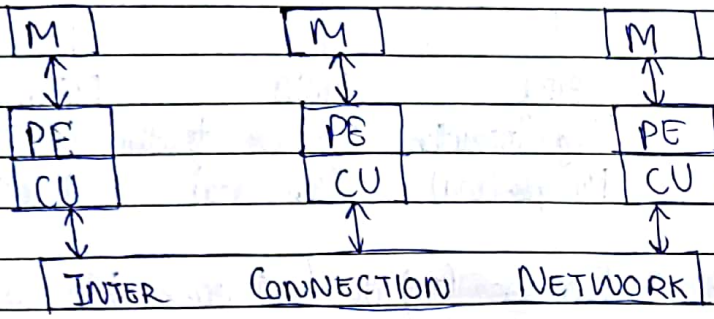


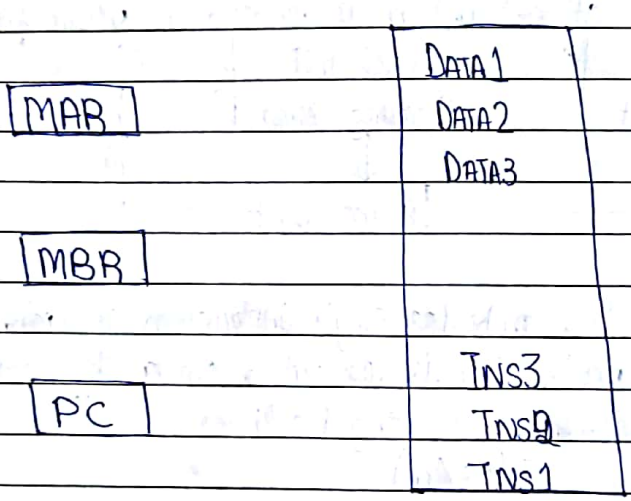
2. SIMD: It stands for single instruction stream, multiple data stream. Each instruction is executed on a diff. set data, with diff. process.
e.g. Array Processing Machines
(n x m)



3. MISD: → Each Processor executes on diff. sequence of instructions.
→ In case of MISD computers, multiple processing units operate on one single Data Stream.
↳ Practically, this kind of Organisation have never been used.

4. MIMD: → Each Processor has a seperate program.
→ Each Instruction operates on diff. data.
e.g. Multiprocessor, Multicomputers.
Data in Processor Executes.
* Used in Modern Computers.





17/07/19

LECTURE -3

Wednesday

Types of Actions Performed by Instruction:-

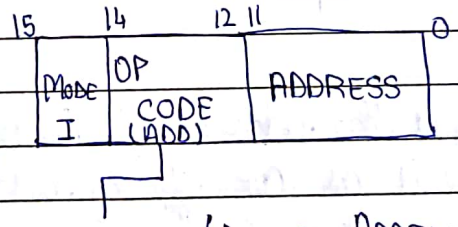
1. PROCESSOR - MEMORY: transfer data from Processor to Memory or vice versa.
2. PROCESSOR - I/O: I/O to CPU to I/O, Data transfer from Input device to CPU or from CPU to output devices.

3. Data Processing: At ALU with help of Instructions, includes arithmetic or logical actions performed by CPU.
4. Control: Instructions that alter the sequential execution of instructions.

Instruction Types

1. Data Processing Ins: for ALU, Arithmetic and logical instructions.
2. Data Storage Ins: Register to Memory, M to R to ALU Deals with the movement of data between Memory to Memory, Register to Register or Memory to Register.
3. Input/Output Ins: User Input to ALU Processor with help of memory to Output., It takes user Input and gives to the System. → System generated output and used to the user.
4. Control Ins: Test Instructions (POS).

Instruction Format

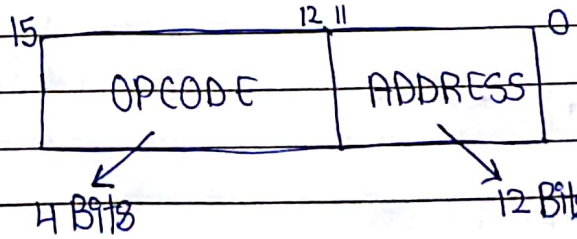


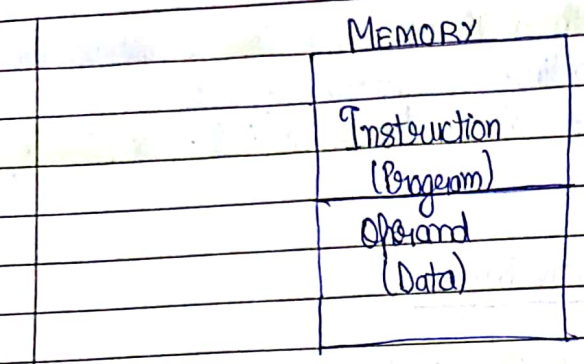
I=0 (Direct Address)
I=1 (Indirect Address)

18/07/19 — Thursday

Lecture - 4

Instruction Code In Computer Architecture





4096 X 16

$2^{12} \times 2^{4}$ → OPCODE

address of Instructions of data

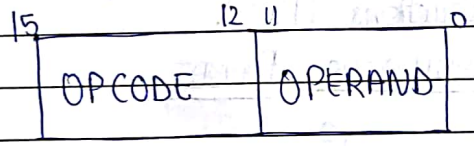
INSTRUCTION CODE: It is a group of bits (0101) that tells the processor to perform certain task.

It can be organised into two parts:

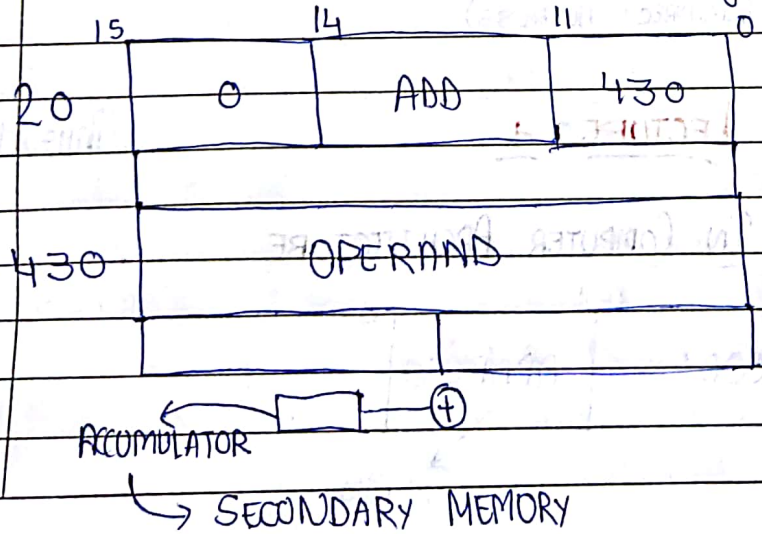
① OPCODE

② ADDRESS

Based on the Instruction code, processor understands whether data comes from imediate, direct or indirect address.

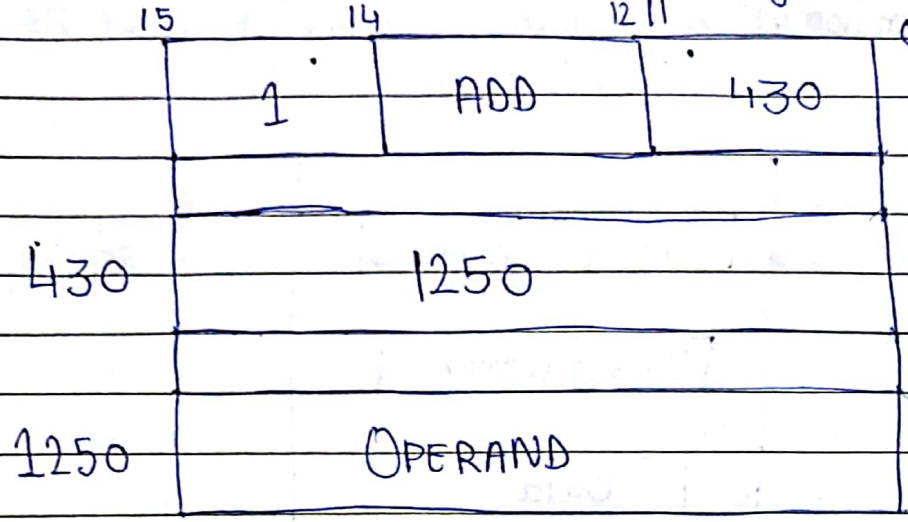


1. DIRECT ADDRESS: It specifies the address of the operand, so we call it as an effective address. So, direct address holds the address of the operand.

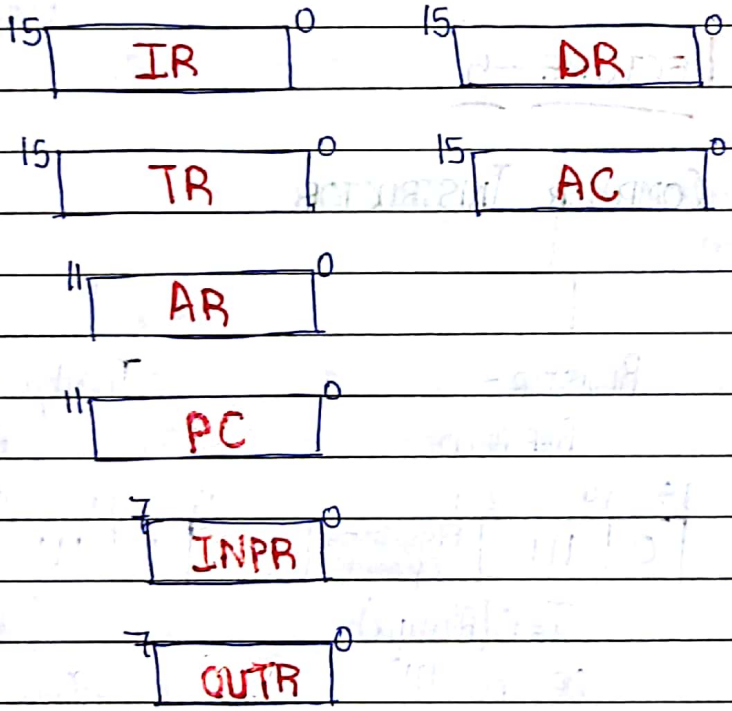


→ SECONDARY MEMORY

2. INDIRECT ADDRESS: It specifies the address of memory word in which address of operand is store.

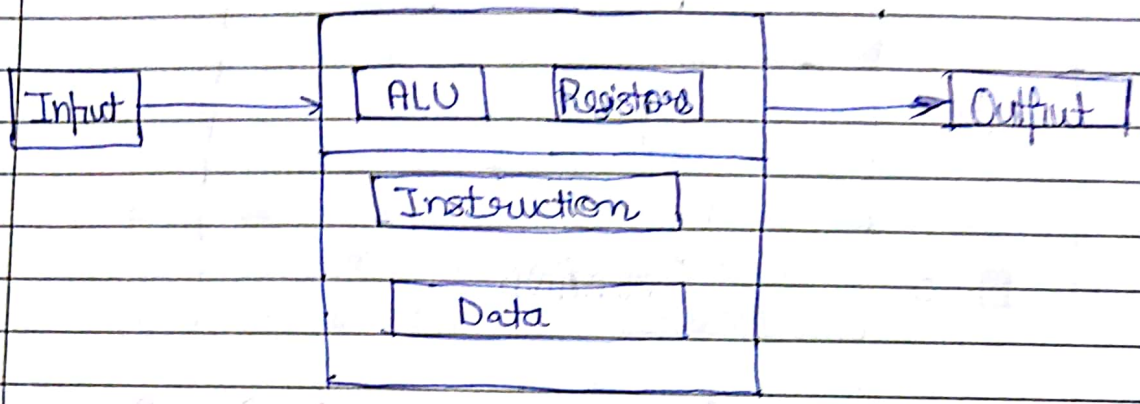


COMPUTER REGISTORS: (Processor take data from Registors)



1. IR (Instructions Registors): It holds the instruction code.
2. TR (Temporary Registors): It holds the temporary code.
3. AR (Address Registors): It holds the address for memory.
4. PC (Program Counter): It holds the address of next instructions.
5. INPR (Input Registors): It holds the Input Character.

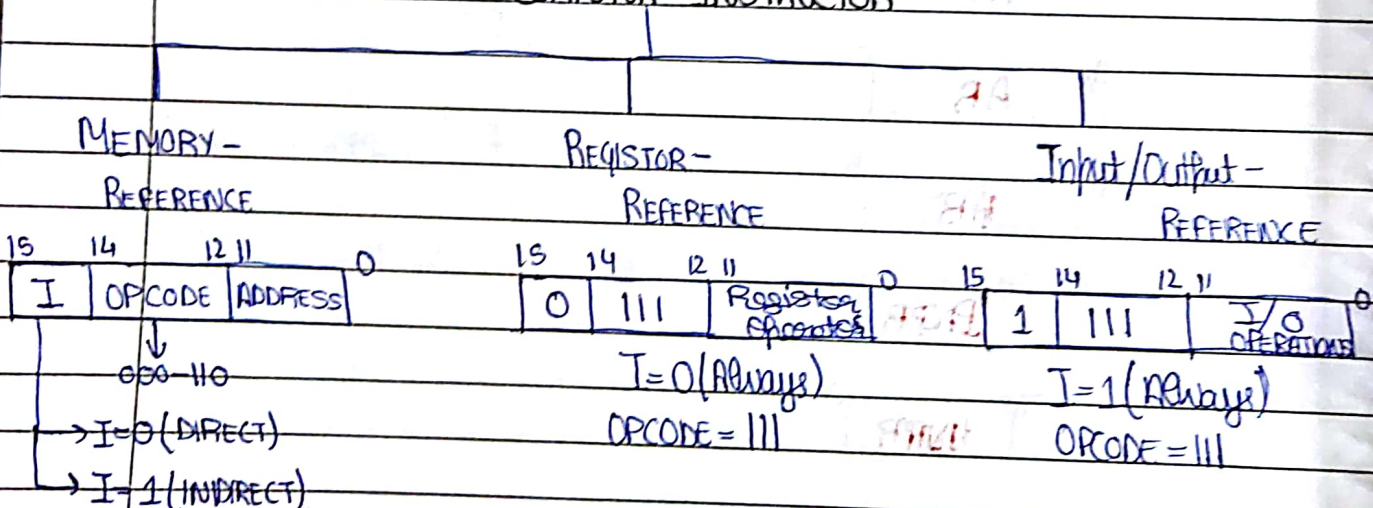
6. OUTR (Output Register): It holds the output character.
7. DR (Data Register): It holds the memory operand.
8. AC (Accumulator): Intermediate between CPU and its units. (ALU) Arithmetic Logic Unit.



19/07/19 Friday

LECTURE -5

COMPUTER INSTRUCTOR



1. Memory-Reference Instruction: uses 12 bits to specify an address and 1-bit to specify the addressing mode.
2. Register-Reference Instruction: are recognized by the operation code 111 with 0 in the left most bit.

The 12 bits are used to specify the operation or test to be executed.

3. Input/Output - Reference: It doesn't need a reference to memory and recognized by Opcode 111 with 1 in the left most bit. The 12 bits are used to specify the type of I/O operations.

Symbol	Hexadecimal Code I=0 (Direct)	I=1 (Indirect)	Description
AND	0XXX	8XXX	AND memory word to AC.
ADD	1XXX	9XXX	ADD " " " "
LDA	2XXX	AXXX	Load " " " "
STA	3XXX	BXXX	Store content of AC in memory
BUN	4XXX	CXXX	Branch Unconditionally
BSA	5XXX	DXXX	Branch & save return address
ISZ	6XXX	EXXX	Increment & if zero Skip

MEMORY REFERENCE

CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Complement AC
CME	7100	Complement E
CIR	7080	Circular right AC & E
CIL	7040	" left " "
INC	7020	Increment AC
SPA	7010	Skip next instruction & AC is +
SMA	7008	" " " " " " -
SZA	7004	" " " " " " 0
SZE	7002	" " " " E is 0
HLT	7001	Halt Complete

REGISTOR REFERENCE

INP	F800	Input character to AC
OUT	F400	Output character from AC
SKI	F200	Skip on Ip flag
SLO	F100	Skip on Op flag
ION	F080	Interrupt ON
IOF	F040	Interrupt OFF

I/O REFERENCE

LECTURE - 6

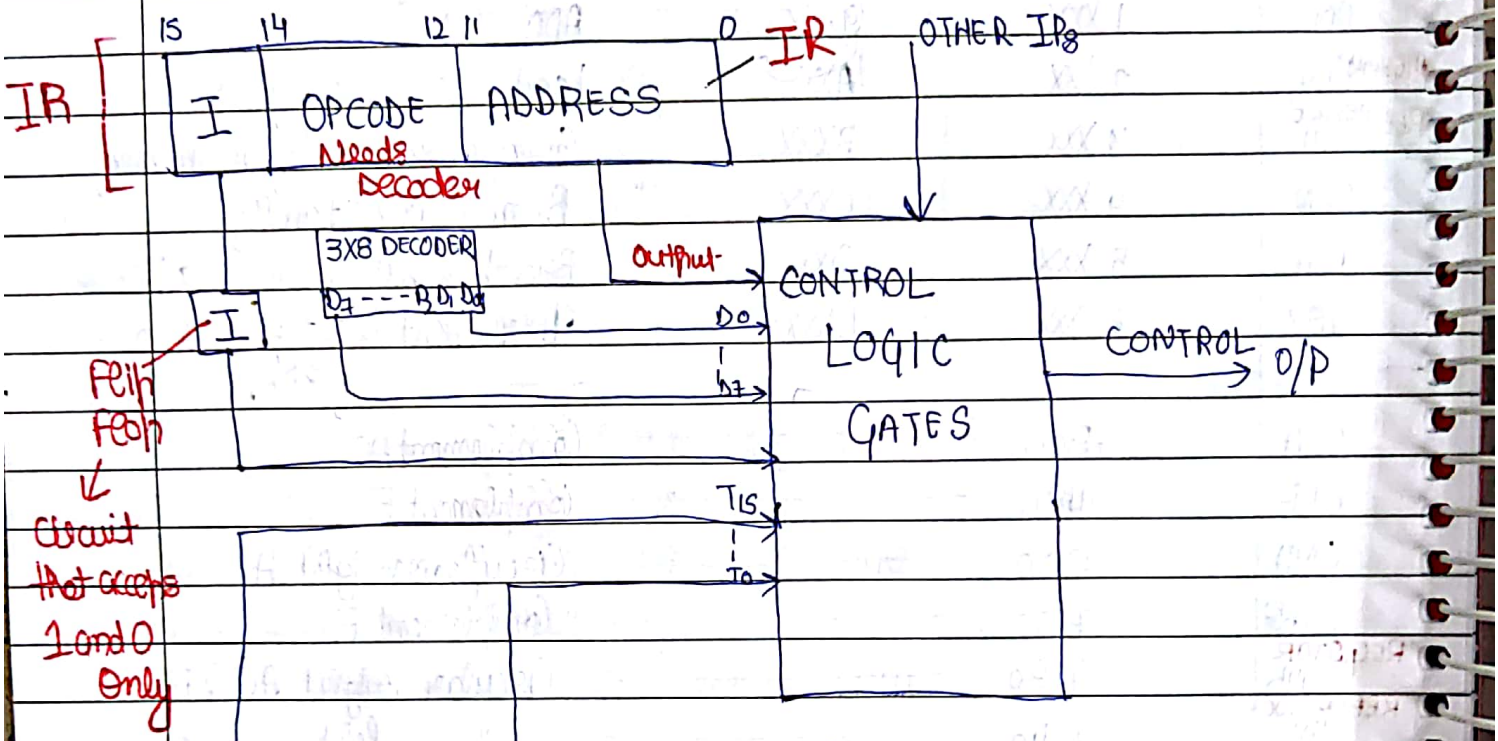
Date 22/07/19
Page

Monday

TIMING AND CONTROL

1. Design of CU → HARDWIRED CU (Mostly Followed)
   → MICROPROGRAMMED CU

2. Components of HWCU: Flip Flop, Logic Gates, Registers using Hardware

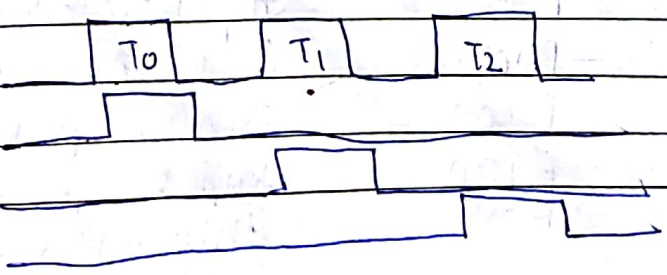


Benefit → Hardwired is very fast

Disadvantage → One change, make other change

CU → Memory → Processor → to all units

Timing needs for bits to work



Control timing and control signals through architecture are relevant

* The function of the control unit is to generate timing and control signals to all operations in the computer.
* Design of CU → CU generates control signals using one of the two organisation:

1. Hardwired CU → It is implemented as logic circuits (Flip flops, gates, Decoders) in the hardware.
→ Hardwired CU is fast because control signals are generated by combination signals.
→ Modification in control signals is very difficult.
→ Components of Hardware CU:

1. Instructor Register
2. No. of control logic gates
3. Two Decoder → Instruction Decode
   Timing Signal Generate
4. 4-bit sequence Counter
5. 1- flip flop (0 and 1) (ON & OFF).

An Instruction read from the memory is placed in instructor Register. IR is divided into three parts:
→ I-bit → OPCODE → ADDRESS

a) First 12 bits (0-11) are applied to CLG.
b) The operation OPCODE (12-14) are decoded bit into 3x8 Decoder.
c) The 8 outputs ($D_0 - D_7$) from decoders goes to CLG to performs specific function.
d) Last bit 15 is transferred to I-flip flop designated by I Symbol.
e) The 4-bit sequence counter can count in binary in 0-15. The counter output is decoded into 16 timing pulses $T_0 - T_{15}$.
f) The sequence counter can be incremented by INR input or clear by (CLR).

Simple View	Complex View
→ Contains only one single base table or is created from one single base table.	→ Contains more than one base tables or is created from more than one tables.
→ We cannot use group functions like Max, Count etc.	→ We can use group functions.
→ Doesn't contain groups of data.	→ It can contain groups of data.
→ DML operations could be performed though a simple view	→ Could not always be through a simple view

23/07/19

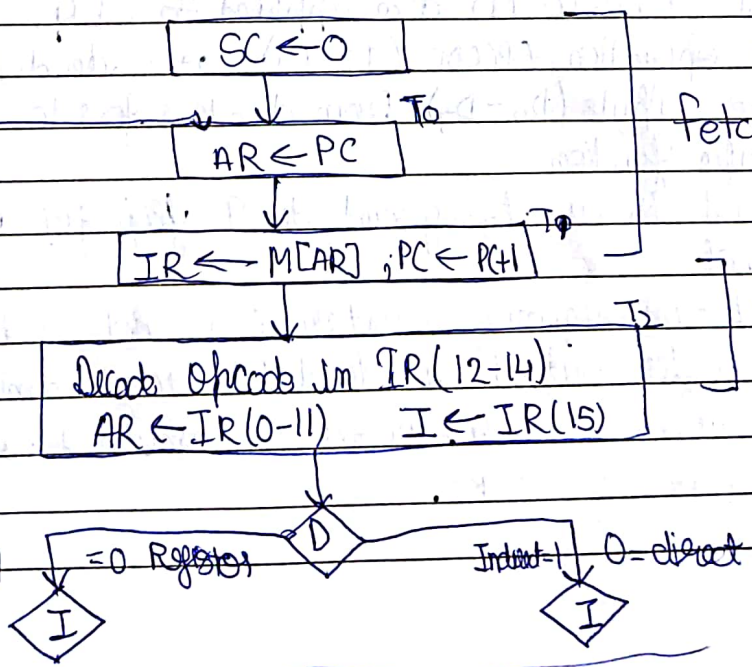
LECTURE - 7

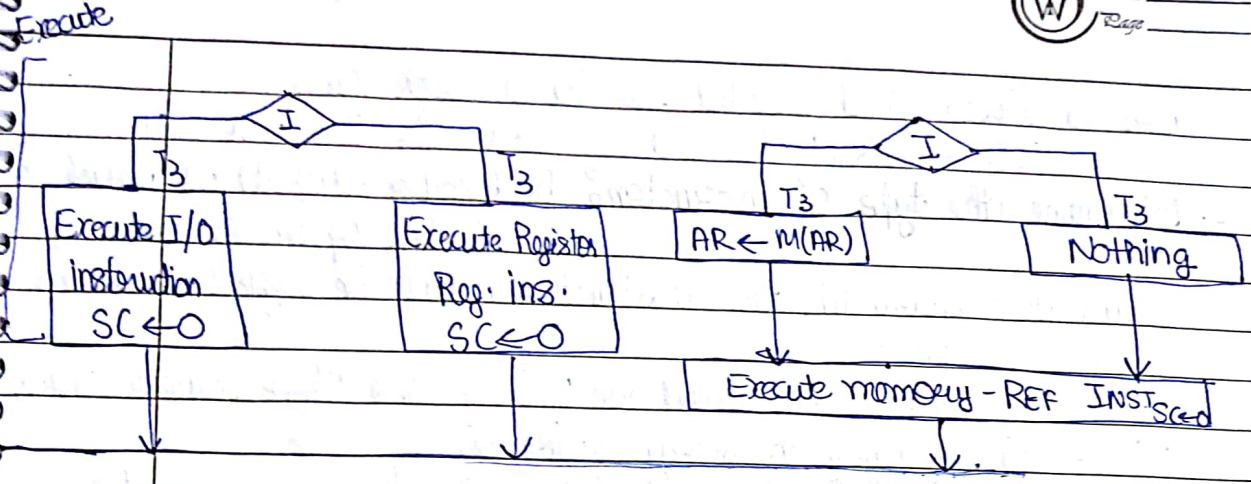
Tuesday

INSTRUCTION CYCLE

1. Fetch an instruction from memory.
2. decode the instruction
3. Read effective address from memory
4. Execute the instruction

SC → Sequence Counter
PC → Program Counter
AR → Address Register
IR → Instruction Register
PC++ → Program Counter Increases





If you want the processor to stop processing instructions, the command used is "HALT".

⇒ A program residing in the memory unit of computer consists of a sequence of instruction.

⇒ The program is executed in the computer by going through a cycle for each instruction. Each instruction cycle in turn is sub divided into subcycles:

(a) fetch (b) Decode (c) Read effective address (d) Execute

⇒ Upon the completion of step4 (d), the control goes back to step1 to fetch, decode and execute the next instruction.

⇒ This process continues unless a 'HALT' instruction is encountered.

1. Fetch: Initially program counter (PC) is loaded with the address of the first instruction in the program.

To AR ← PC

T_1 IR ← M[AR], PC ← PC+1

→ SC is initialized to zero.

→ The address of the first instruction from PC is loaded into address Register during the first clock cycle.

2. Decode: → The instruction is decoded by the Instructor Decoder of processor.

→ All the bits of the instruction under execution stored in IR

are analysed and decoded in third clock cycle.
T_2: DECODE ← IR [12-14]; AR ← IR (0-11), I ← I.R (15).

3. Determine the type of instruction: D (Decoder Output) is equal to 1. If the Operation code is equal to binary 111. The instruction must be register reference or I/O reference.
→ If D = 0, the OPCODE must be one of the other seven values (000 - 110) specifying memory reference.

4. Execute Cycle: → In the last phase the processor executes the instruction.
→ After instruction is executed, SC is cleared to 0. Control returns to the fetch phase.

25/07/19 Thursday

LECTURE : 8

INTRODUCTION TO MEMORY - REFERENCE INSTRUCTION

	SYMBOL → $(T_4 X T_6)$	OPERATION DECODER
1.	AND	D_0
2.	ADD	D_1
3.	LDA	D_2
4.	STA	D_3
5.	BUN	D_4
6.	BSA	D_5
7.	ISZ	D_6

000 - 110

I → 1/0

AND:

(1) Performs the AND Logic Operation on pairs of bits in accumulator and the Memory word specified by the effective address

(2) Two Timing signals are needed $D_0 T_4$ $DR \leftarrow M[AR]$ (effective address)

In T_4, Transferring operand from memory into DR.

In T_5, Transferring result of AND Logic Operation b/w the contents of DR and AR. $D_0 T_5$ $AC \leftarrow AC \wedge DR, SC \leftarrow 0$

ADD:

(1) Add the contents of memory word specified by the effective address to the value of accumulator (AC).

(2) Some is transfer to AC and output Carry (C_{out}) is transferred to E (Extended AC). (Operand Address)

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

$D_2 T_5$, $AC \leftarrow AC + DR$ $E \leftarrow C_{out}$ $SC \leftarrow 0$

LDA: Load to AC

(1) Transfers the Memory word specifies by the effective address to AC. $D_2 T_4$: $DR \leftarrow M[AR]$

(2) $D_2 T_5$: $AC \leftarrow DR$

STA: Store AC (result)

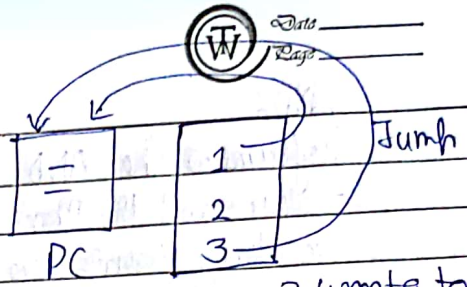
(1) It stores the content of AC into the memory word specify by the effective address.

$D_3 T_4$: $M[AR] \leftarrow AC$

BUN: Branch Unconditionally

(1) BUN instruction allows the programmer to specify the instruction out of sequence and we say that the Programme Branches (jumps and program).

$D_4 T_4$: PC ← AR



3 wants to execute first

BSA: Branch and Save return address.

(1) Its useful for Branching to the routine of program.

(2) When executed it stores the address of the next instruction in sequence (available in PC) into a memory location specified by effective address.

$D_5 T_4$: MBAR ← (PC), AR ← AR+1

$D_5 T_5$: PC ← AR

ISZ: Increment & Skip if zero

↳ This instruction increments the word specified by the effective address and if increment value 0 then PC is increment by 1.

$D_6 T_4$: DR ← M[AR]

$D_6 T_5$: DR ← DR+1

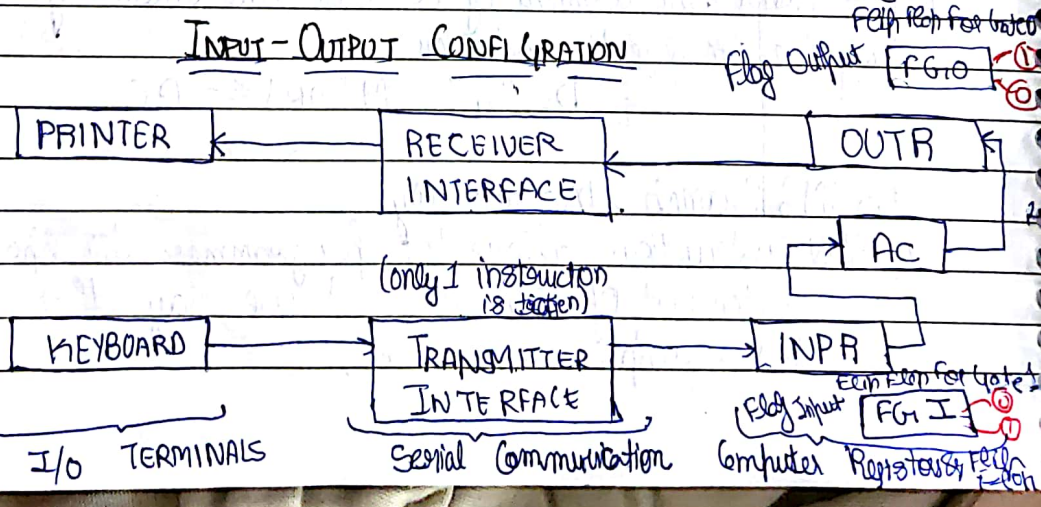
$D_6 T_6$: M[AR] ← DR, if (DR=0), then PC ← PC+1

26/07/19

LECTURE: 9

Friday

INPUT-OUTPUT CONFIGURATION



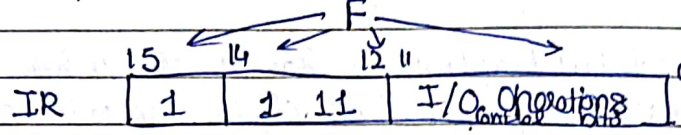
1. The INPR consist of 8 bits and holds an alpha-numeric input information.
2. The 1-bit input FLAG (FGI) is a control Flip Flop. The Flag bit is said to 1 when new information is available into Input device and it clears to zero when info. is accepted by computer. (AC).
3. The Flag is needed to synchronize the timing rate difference between the Input device and the computer.
4. Initially,
   (i) The Input Flag (FGI) is clear to zero, when any key is struck in the keyboard, 8-bit alpha numeric code is shifted into INPR, and Input Flag (FGI) is said to one.
   (ii) The computer checks the Flag bit, if it is One, the info. from INPR is transferred in parallel into AC and FGI is clr to zero. Once the flag is clr, new information can be shifted into INPR, by striking another key.
   (iii) The Output Register (OUTR), works similarly but direction of information flow is reverse, the output Flag initially said to 1, the Comp. Check the Flag bit, if it is 1, the info. from AC is transferred in parallel to OUTR and FGO is clr to 0. The Output device accepts the coding info., prints the corresponding character and when the info is complete it sets FGO → 0.

10-7-19 Thursday

LECTURE : 10

INPUT - OUTPUT INSTRUCTIONS

* I/P & O/P instructions are needed for transferring information to & from AC register for checking the flag bits & for controlling the interrupt.
* Used for data communicating data b/w CPU & I/O Peripheral devices.



* I/O instruction have OPCODE '111' and are recgnized by control $D_7 = 1$ $I = 1$
* These instructions are executed with clock transaction having signal T_3.
* Each control function needs a Boolean relation $D_7 I T_3$ which we designate for our convenience by the signal 'P'.
* $D_7 I T_3 = p$ (Common to all I/O instructions)
  $IR(i) = B_i$ [bit in IR(6-11) that specifies the instruction]

FGI (Flag Input)

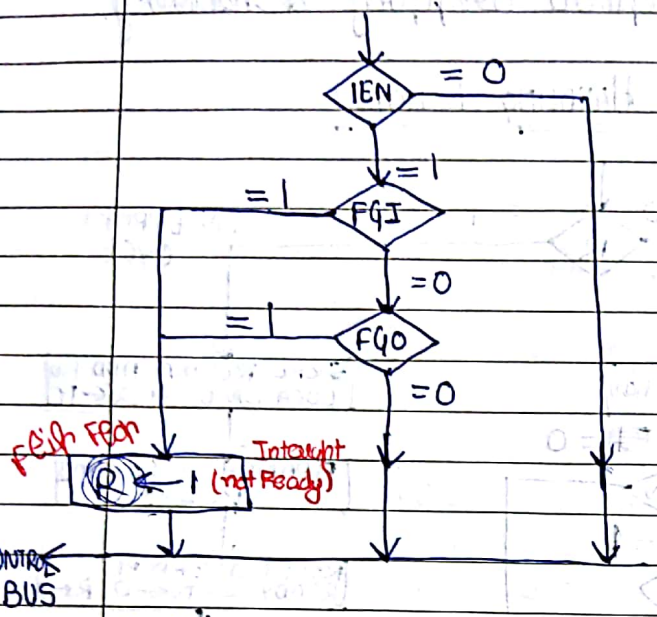
	p		
INP	PB_{11}	: $AC(0-7) \leftarrow INPR$, $FGI \leftarrow 0$	Input Character
OUT	PB_{10}	: $OUTR \leftarrow AC(0-7)$, $FGO \leftarrow 0$	Output Character
SKI	PB_9	: If $(FGI = 1)$ Then $(PC \leftarrow PC+1)$	Skip on I/P Flag
SKO	PB_8	: If $(FGO = 1)$ Then $(PC \leftarrow PC+1)$	SKip on O/P Flag
ION	PB_7	: $IEN \leftarrow 1$	Interrupt enable on
IOF	PB_6	: $IEN \leftarrow 0$	Interrupt enable off

FGO ↳ Flag Output

02/08/19 Thursday

LECTURE : 11

INTERRUPTS



IEN
FGI
FGO
R

on load Flag value is 0. (Booting Process)

1. I/O interaction with electro-mechanical Peripheral devices require huge processing time compare with CPU processing time.
   - ↳ i.e. I/O (milliseconds) Vs CPU (nano/microsec)
2. Interrupts permit other CPU instructions to execute while waiting for I/O to complete
   - ↳ need an additional 1 bit 'IEN' Flip Flop to store the interrupts status (0/1).

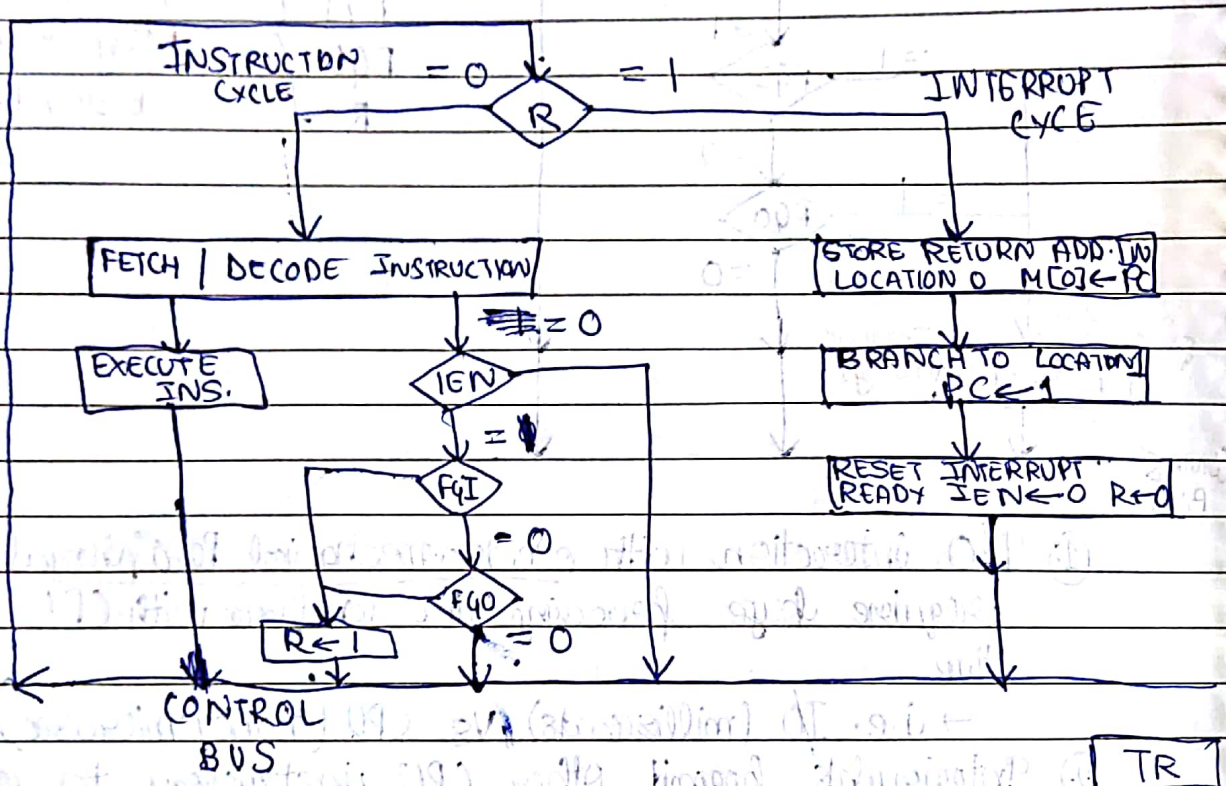
OPCODE	Mnemonic (Assembly Language Code)	Meaning
F080	ION	IEN ← 1
F040	IOFF	IEN ← 0

3. In this approach, Interrupts are used only with I/O handling.
4. In addition to a flip flop to store the interrupt

enable state, one more flip flop(R) is needed to store the I/O status (Ready / Not Ready).

(5) In general, interrupts may be used with arbitary instructions for exception trapping & handling.

INTERRUPT HANDLING FLOWCHART



(1) An Interrupt flip flop (R) is included in the computer. (For Interrupt Handling) (2) When R=0, the computer goes through an instruction cycle.

(3) During execution phase Ins. cycle, IEN is checked by control.

↳ If it is 0, it indicates that the Programmer does not want use the interrupt. So control continues with next instruction cycle.

↳ If IEN is 1, control checks the flag bits. If both flags are 0, it indicates that neither I/P nor O/P

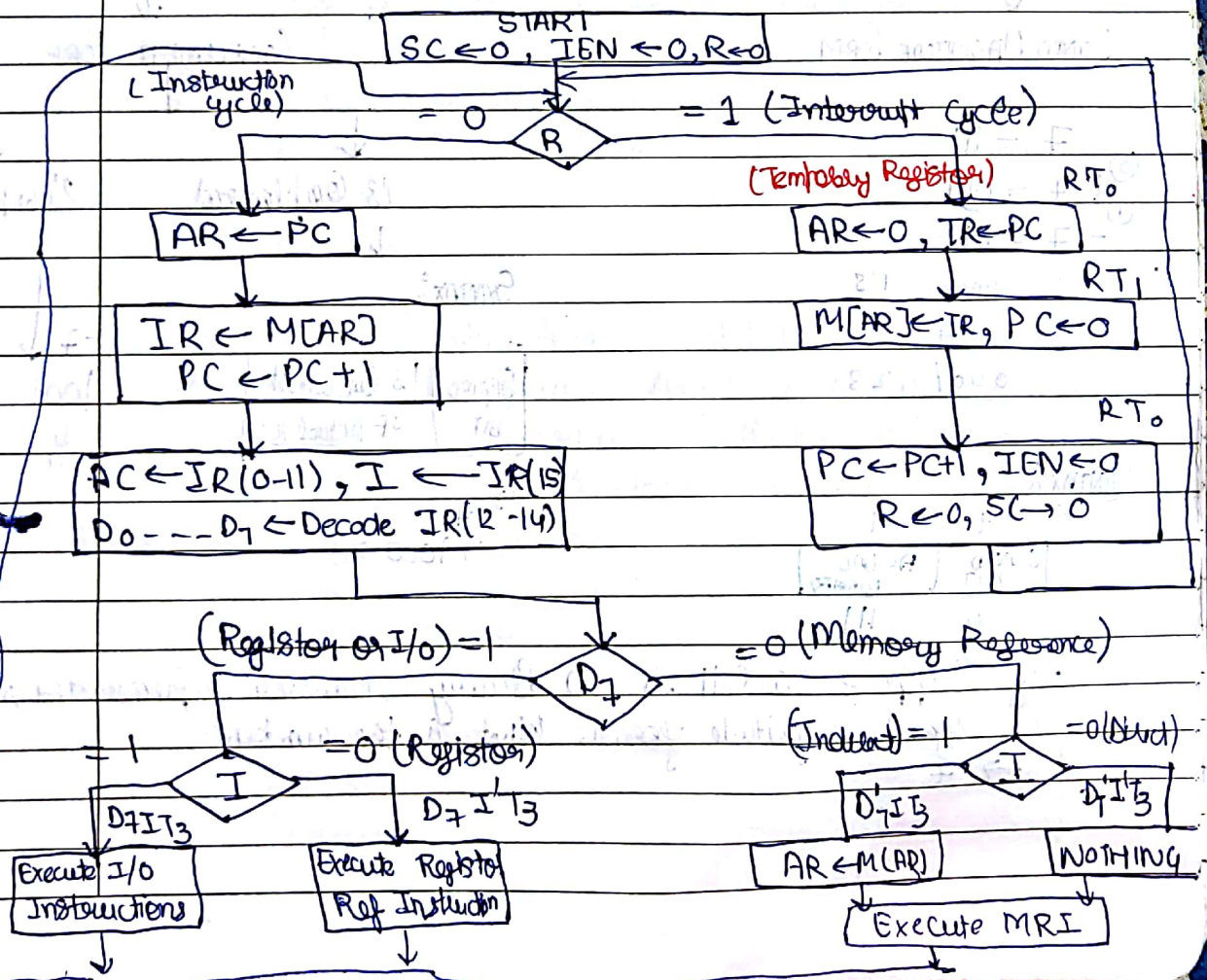
registers are ready for transfer of info. In this case control continues with next inst. cycle.

If either flag is set to 1 while IEN=1, flip flop R is set to 1. At the end of execute phase, control checks the value of R & if it is not equal to 1, it goes to an interrupt cycle instead of an instruction cycle.

02/08/19 Friday

LECTURE : 12

Complete Computer Instruction



① If R=1, Computer goes through in Interupt cycle.
② If R=0, Computer goes through a instruction cycle.
③ If the Instruction is one of the memory Reference inst., the Computer first checks if there is an indirect address and then continues to execute the decoded instruction acc. to the flow chart.

06/08/19

LECTURE: 13

Tuesday

REPRESENTATION OF SIGNED NO's

- SIGNED MAGNITUDE FORM
- COMPLEMENT FORM
  - 1's Complement
  - 2's Complement

+7 = 111
+7 = 0111
-7 = 1111

0000 1's
+1
0001 2's

SYNTAX:

+	+
SIGNED BIT	ACTUAL BINARY
0	111

SYNTAX:

SIGNED BIT	1's Complement of Actual BIT

-7
↳ 1000 → 8

-7
1001
↓
9

e.g. Suppose a 0111 is a Binary number represented in signed magnitude form. What is the number?
→ 7

BINARY AIRTHMETIC

1. Airthmetic Instructions in digital computer manipulate data to produce results necessary for solutions of computational problems.
2. The four Basic Airthmetic Operation are:
   a) Addition
   b) Subtraction
   c) Multiplication
   d) Division
3. An Airthmetic Processor is the part of processor unit that executes airthmetic Operation.
   a) An Airthmetic Instruction may specify binary or decimal data and in each case the data may be fix point or floating point form.
   b) Negative numbers may be signed magnitude or signed complement Representation.

Algorithm :

↳ Solution to any Problem that is stated by infinite no. of well-defined procedural steps is called Algorithm.
↳ Here we develop, the various Airthmetic algos and show the procedure for implementing with digital hardware.
↳ We consider add, sub, mul, div for following type of data.
   a) Fix-point binary data in signed magnitude representation
   b) Fix-point binary data in signed 2's complement representation
   c) Floating-point binary data.

Thursday
8 Aug 2019

LECTURE: 14

ADDITION & SUBTRACTION WITH SIGNED-MAGNITUDE DATA

Algo. for addition and subtraction:

Addition Algo:

1. When the signs of a and b are identical (same) add the two magnitudes and attach sign of (a) to the result.
2. When the signs of a and b are different, compare the magnitudes and choose the sign of the result to be same as (a), if a > b.

Subtraction Algo:

1. When the signs of a and b are different, add to the magnitudes and attach sign of (a) to the result.
2. When the signs of a and b are indentical, compare the magnitude, choose the compliment of sign of a, if a < b.
3. If the two magnitudes are equal, subtract (b-a) & make the sign of the result positive.

OPERATION	ADD MAGNITUDE	SUBTRACT MAGNITUDE when A > B (Sign A)	A < B (Sign B)	A = B
addition (+A) + (+B)	+(A+B)			
(+A) + (-B)		+(A-B)	⊖(B-A)	+(A-B)
(-A) + (+B)		-(A-B)	⊕(B-A)	+(A-B)
(-A) + (-B)	-(A+B)			
Subtraction (+A) - (+B)		+(A-B)	-(B-A) (compliment)	+(A+B)
(+A) - (-B)	+(A+B)			
(-A) - (+B)	-(A+B)			
(-A) - (-B)		-(A-B)	+(B-A)	+(A-B)

Working:

(1) We designated the magnitude of two numbers by a and b.

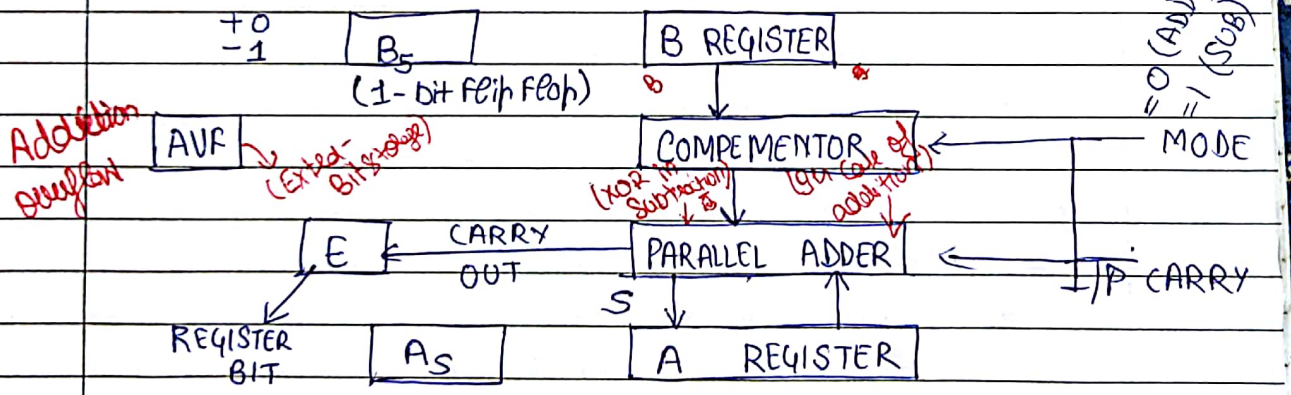
(2) When the signed numbers are added or subtracted, we find that there are 8 diff. conditions. To Consider, depending on the sign of numbers and the operation perform, these conditions are listed in the 1st column of table below.

(3) The other columns in the table shows the actual operation to be perform with magnitude of numbers.

(4) The last column is needed to prevent negative zero. In other words, when to equal numbers are subtract, the result should be plus and not zero.

09 Aug 2019 — LECTURE : 15 — Friday

HARDWARE IMPLEMENTATION



(1) To Implement the ~~the~~ two arithmetic operations with hardware, it is first necessary that the 2 no's to be stored in registers.

(2) Let a and B_S ~~be~~ be two registers that hold the magnitudes of the numbers and A_S and B_S be two flip flops that hold the corresponding sign.

(5) The Results of the Operation may be transfer into A_S and A_I, thus A and A_S together form an accumulator register.

Working:

Consider Now the Hardware Implementation of the Algorithm

(1) Parelled Adder is needed to Perform the micro-Operation. (A+B).

(2) Comparator circuit is needed to establish (if a>b), a-b Or (a<b). (BACKEND)

(3) Two Parelled Subtractor circuits are needed to perform the micro-Operation, (a-b) & (b-a).

(4) The Sign relationship can be determined by from an Exclusive-OR Gate with A_S & B_S as input.

(5) The Output Carry is transferred to Flip Flop E.

(6) The Complementor consists of Exclusive OR Gates and the Parelled Adder consists of (Full-Adder Circuit).