

Assignment - L:

1. Given a register $R_1 = 11100011$, determine the value of R_1 after execution of following micro operations.

- $R_1 \leftarrow CIL(R_1)$
- $R_1 \leftarrow ASHR(R_1)$
- $R_1 \leftarrow SHL(R_1)$

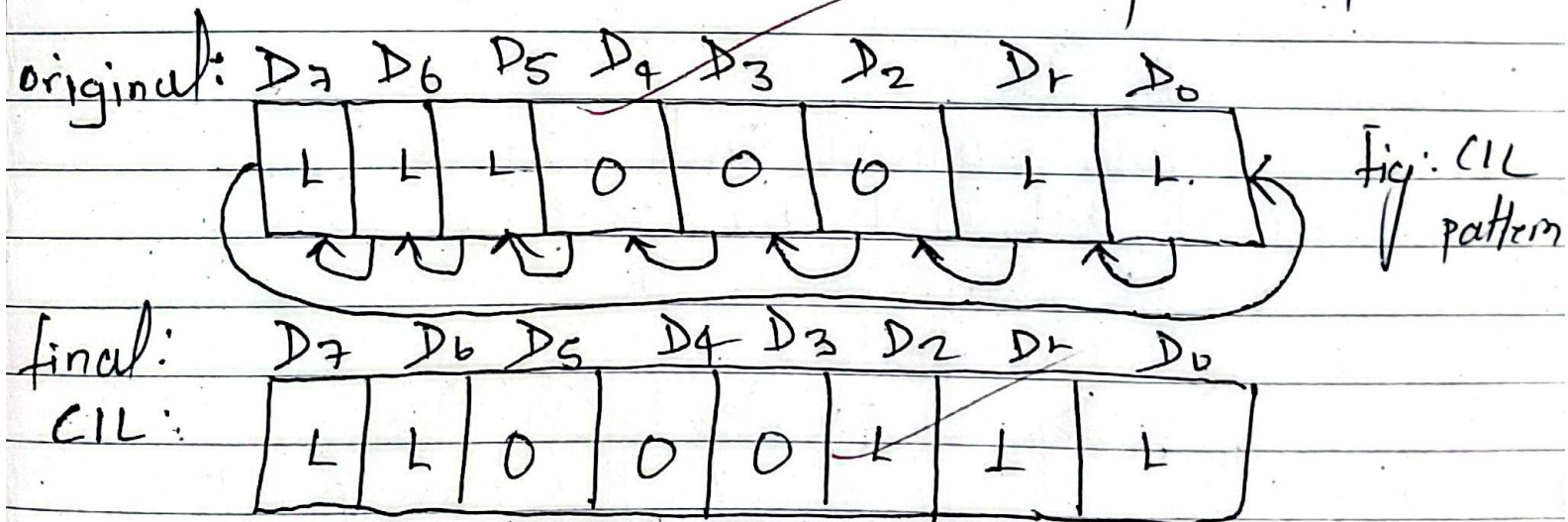
$\Rightarrow$ Here,

Given,

register(R_1) = 11100011 (8-bits)

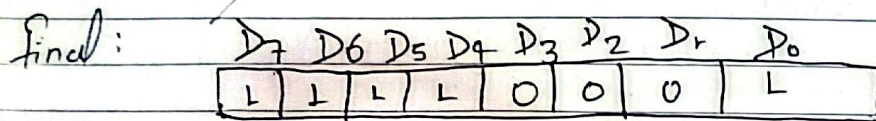
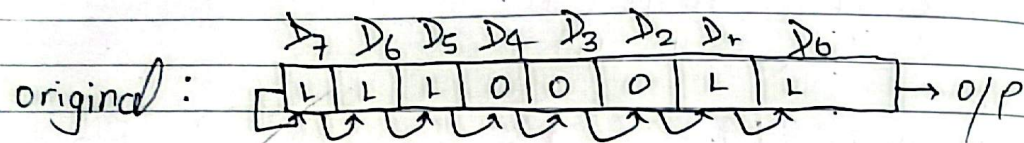
a. $R_1 \leftarrow CIL(R_1)$

$\rightarrow$ CIL means Circular Shift Left
Here bits are shifted left by one position.



$\therefore$ result after CIL : 11000111

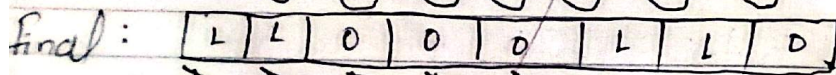
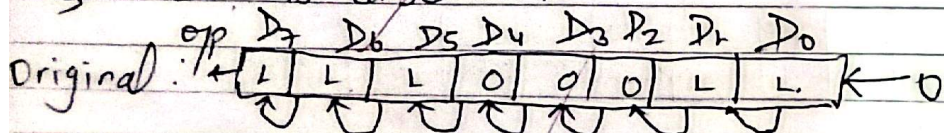
- b. $R_1 \leftarrow \text{ASHF}(R_1)$
- ASHF means Arithmetic shift Right
 - Right shift all bits by 1.
 - The sign bit (MSB) is preserved.



∴ result after ASHF is 1111001

c. $R_1 \leftarrow \text{SHL}(R_1)$

- SHL means logical shift Left.
- Shift all bits left by one.
- insert 0 into LSB.
- MSB is discarded.



result is $(R_1) = 11000110$

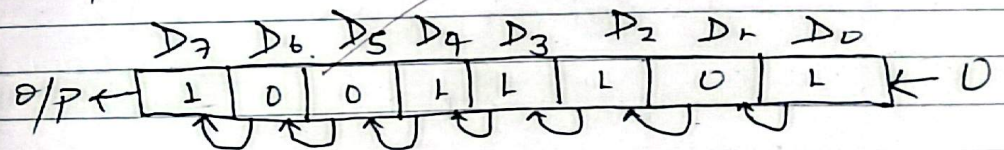
2. An 8-bit register contains the binary value $R = 10011101$, determine the sequence of binary values in R after logical shift-left, followed by a circular shift right, followed by logical shift right and a circular shift left.

⇒ Here.

Given,
register $(R) = 10011101$ (8-bit)

i) Logical shift Left (LSL):

→ A logical shift left shifts all bits to the left by 1, inserting 0 at the right/most position. Leftmost bit is discarded.

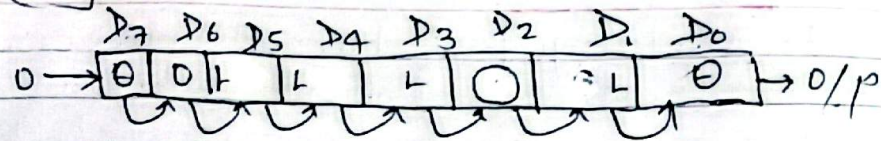


Then the result is 00111010

ii) Circular shift-right (Rotate Right)

→ Rotate all bits to right by 1. The rightmost bits moves to the left position.

→ Input (original) (P) = 00111010

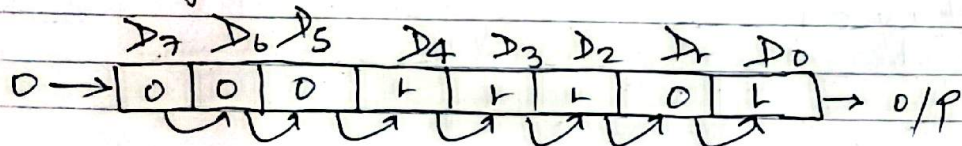


then result is 00011101

1) Logical Shift Right (LSR):

→ Input (original) (P) = 00011101

→ Shifts all bits to the right by 1, inserting 0 at the leftmost position.
→ rightmost bits is discarded.

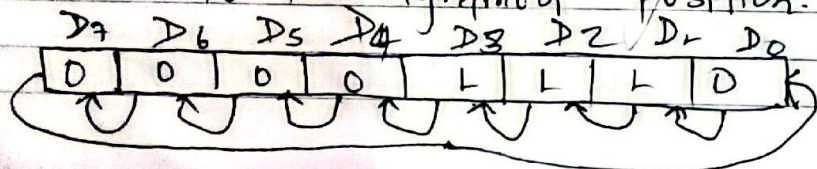


Then -

the result is 00001110

IV). Circular shift left (Rotate Left):

→ Here input bits are 00001110
→ Rotate all bits to left by 1. The leftmost bit moves to the rightmost position.

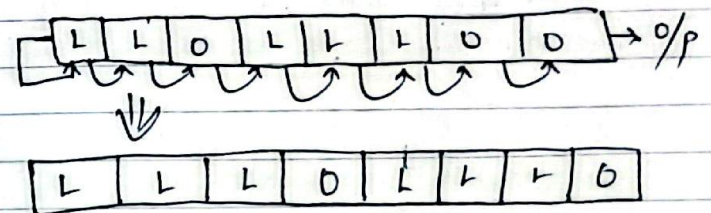


then, the result is 00011100

3. An 8-bit register contains the binary value 11011100. What is the register value after an arithmetic shift right? Starting from the initial number 11011100; determine the register value an arithmetic shift left, and state whether there is an overflow.

⇒ Here, First - Part,
initially P = 11011100

After arithmetic shift right:



∴ P = 11101110

Second - Part,
Starting from P = 11011100
performing arithmetic shift left (ASHL):

5. A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers.

- How many selection lines are there in each multiplexer?
- What size of multiplexers are needed?
- How many multiplexers are there in the bus?

⇒ The given digital computer register is 16 bits = 32 bit.

a) The no. of selection lines used in each multiplexer is 4. Because every multiplexer will get 16 different inputs from 16 different registers. Hence, 2^4 i/p in multiplexer will need 4 selection line.

b) The size of multiplexer needed is 16×2 . Because 16 registers will generate 16 different i/p for each multiplexer.

c) There will 32 mux used in the bus system. Because the system has 16 registers

($P_0, P_2, \dots, P_{15}$) in which each of them will generate 32 bits. To select multiplexed i/p among 32-bits, we need 32 multiplexers.

In conclusion,

No. of mux used = No. bits present in each register.
= 32

No. of i/p to mux = No. of registers used.
= 16

##

6.] Define micro-operation. Explain the different types of micro operation. The following transfer statement specify a memory. Explain the memory operation in each case.

a. $R_2 \leftarrow M[AR]$

b. $M[AR] \leftarrow R_3$

⇒ The elementary operation / task performed by a computer is called micro-operation.

* Classification of micro-operation.

i) Data transfer micro-operation: In this micro-operation data is transferred from source to destination without changing the content. The transfer may be betⁿ register to register, memory to register or betⁿ register to i/p.
The data transfer is represented by ($\leftarrow$)
Eg. $R_1 \leftarrow R_3$; (R_3) is transferred to R_1 .

ii) Arithmetic micro-operation: In this micro-operation arithmetic operation such as addition, subtraction, multiplication, Division are performed.

Some Arithmetic operation are,

ADD (Addition)
SUB (Subtraction)
MUL (multiplication)
DIV (Division)

Eg. $R_1 \leftarrow R_1 + R_2$;

iii) Logical micro-operation: In this micro-operation, logical operations like AND, NOT, OR, NAND, NOR, Ex-OR, Ex-NOR are formed.

Eg. $R_1 \leftarrow R_1 \wedge R_2$; (AND)
 $R_1 \leftarrow R_1 \vee R_2$; OR

iv) Shift operation & operation:

a) Logical shift

1) LSL (Logical Shift Left)

↳ Every bit is transferred towards left by one bit position and LSB will be zero and MSB goes to o/p.

Eg.

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

o/p ←

↓

D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	0
----------------	----------------	----------------	----------------	----------------	----------------	----------------	---

2) LSR (Logical Shift Right)

↳ Every bit is shifted towards right by one bit position and MSB will be zero and LSB goes to o/p

Eg.

0	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
---	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

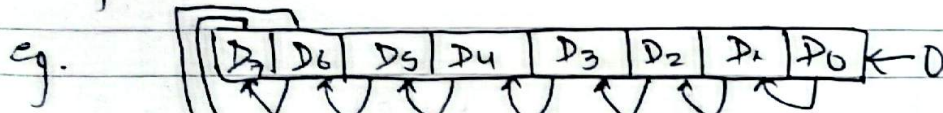
o/p →

0	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
---	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

b. Arithmetic Shifting:

i) ASHL (Arithmetic Shift Left):

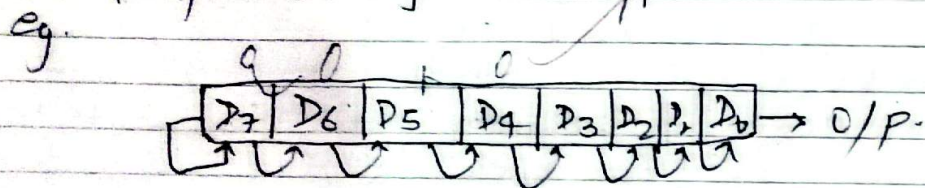
↳ Every bit is shifted towards left by one bit position, LSB will be zero and two of the MSB to Ex-OR get to check for overflow.



↓ (overflow bit): 0: No overflow
1: overflow.

ii) ASHR (Arithmetic Shift Right):

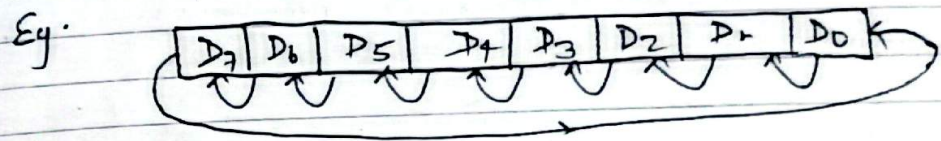
↳ Every bit is shifted towards right by one bit position. MSB will be repeat itself. LSB goes to o/p.



c.) circular shifting:

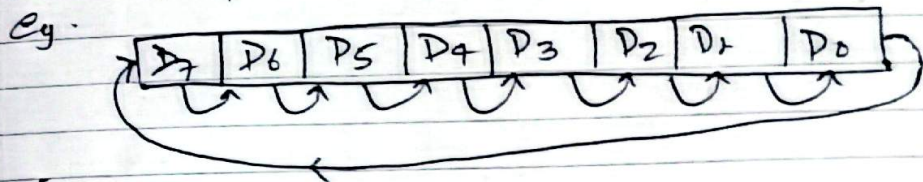
i) CIL (Circular Shift Left):

↳ Every bit is shifted towards left by one bit position and MSB will go to LSB.



ii) CIR (Circular Shift Right):

↳ Every bit is shifted towards right by one bit position and LSB will go to MSB.



Given.

$$a. R_2 \leftarrow M[AR]$$

Here, the content of memory location whose address is given in AR-register, have to be transferred from that location to R_2 -register. So, the required memory operation are,

M/M RD; for address read.

M/M RD; for data read.

$$b. M[AR] \leftarrow R_3$$

Here, The content of the register (R_3) have

to be transferred for the memory location whose address is given in AF register. Required M/M opt. are

M/M PD, for address read.
M/M PD, for data write.

Q.2.7] The 8-bits register AF, BF, CF and DF initially have following value:

$$AF = 11110010$$

$$BF = 11111111$$

$$CF = 10111001$$

$$DF = 11101010$$

Determine the 8-bit values in each registers after the execution of the following sequence of micro-operations

$$AR \leftarrow AR + BF$$

$$CF \leftarrow CR \wedge DF, BF \leftarrow BF + 1$$

$$AR \leftarrow AR - CR$$

⇒ Performing the given sequence of micro-operations

$$\begin{array}{r} AR \leftarrow AR + BF \Rightarrow \\ 11110010 \\ + 11111111 \\ \hline 11110001 \\ CY = 1 \end{array}$$

$$\therefore AF = 11110001$$

$$\begin{array}{r} CF \leftarrow CF \wedge DF \Rightarrow \\ 10111001 \\ \wedge 11101010 \\ \hline 10101000 \\ \therefore CR = 10101000 \end{array}$$

$$\begin{array}{r} BF \leftarrow BF + 1 \Rightarrow \\ 11111111 \\ + 1 \\ \hline 10000000 \\ CY = 1 \\ \therefore BF = 00000000 \end{array}$$

$$\begin{array}{r} AR \leftarrow AR - CF \Rightarrow (11110001) - (10101000) \\ = 11110001 \\ 01010111 \\ + 1 \\ \hline 01001001 \\ \therefore CY = 1 \end{array}$$

$$AR = 01001001$$

Hence, find values in registers AR, BF, CF & DF after completion of given sequence of instructions.

$$\begin{array}{ll} AR = 01001001 & CR = 10101000 \\ BF = 00000000 & DF = 11101010 \end{array}$$

8. Design a digital circuit that performs the four logic operations of exclusive-OR, ex-NOR, NOR and NAND. Use two selection variable. show the logic diagram of one typical state.

⇒ We have to design a logical circuit, that can perform any one of four operation ex-OR, Ex-NOR, NOR, NAND at any typical stage.

truth table: let s_0 and s_1 be the two selection line.

s_1	s_0	O/P (y)
0	0	Ex-OR
0	1	Ex-NOR
1	0	NOR
1	1	NAND

Circuit: let A and B be two I/p on which A B operation have to performed.

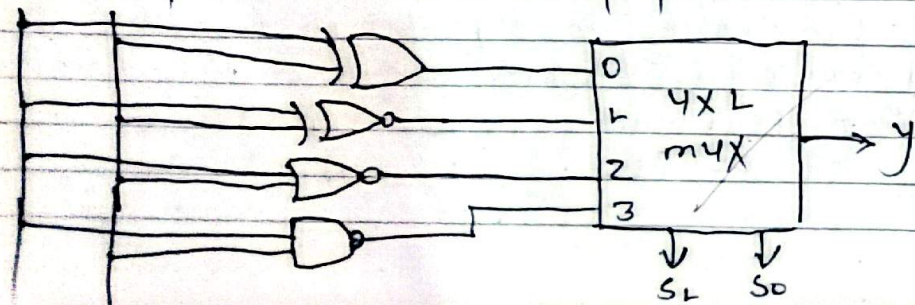


fig: logical circuit to performed the given operation.

The above logical circuit for NAND operation for. NAND.

$$s_1 = 1$$

$$s_0 = 1$$

$$(11)_2 = (3)_{10}$$

3rd I/p will be selected as o/p of mux.

$$i.e. y = \overline{A \cdot B}$$

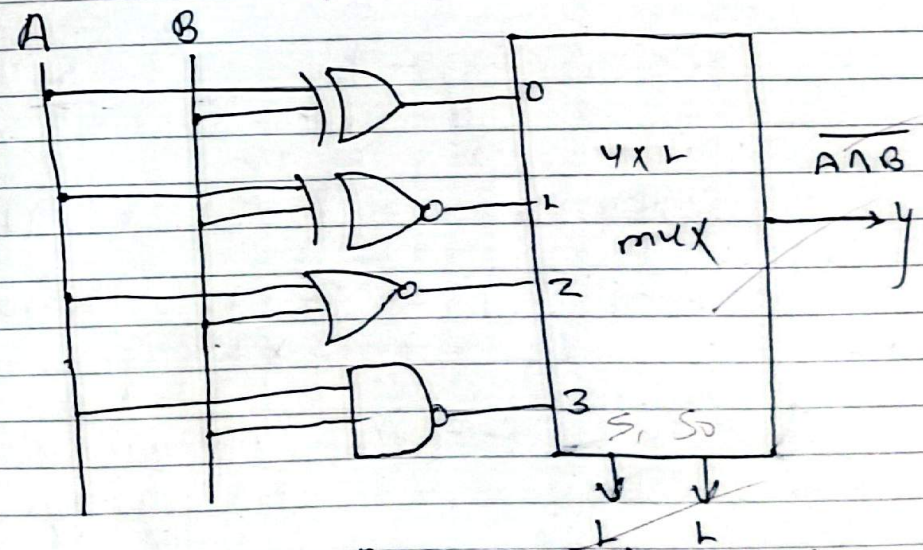


fig: logical circuit to perform NAND among multiple operation using mux.

9]. A digital computer has a common bus system for 8 registers 16-bits each. How many multiplexers are required to implement common bus system? What size of multiplexers is required?

⇒ Here, Given,

The common bus system has.

8- registers

16-bit in each register.

The no. of mux required = No. of bits present in each registers.

$$= 16$$

The size of mux required = No. of registers $\times 1$

$$= 8 \times 1$$
$$= 8$$

Hence, 3. Selection line needed

✓

Introduction:

Unit-1

Note:-

Introduction and difference between Computer Architecture and Computer Organization:

Computer architecture refers to those attributes of a computer which are visible to programmers and have direct impact on a logical execution of a program.

Computer Architecture includes the instruction set, the number of bits used to represent various instructions and data types, input-output mechanism and technique for addressing memory.

Computer organization refers to those attributes of a computer which are related with operational unit and their interconnection that realize the architectural specification.

Computer organization include those attribute i.e hardware details transparent to the programmer such as control signals, interface between peripherals and computers and memory technology.

Example: It is an architectural design that issues whether a computer will have multiple instruction. It is an organization that issues whether that instruction will be implemented by using a special multiply unit or by a mechanism that uses successive addition.

Some Difference in table form (2024's)

Computer Architecture

Computer Organization

→ It focus on functional behavior (what it does).

→ It focus on structural behavior (how it does it).

→ Concerned with ISA, instruction, addressing mode, data paths, control units.

→ It is high level (Software view).

→ It is low level (hardware view).

→ Audience are the programmers.

→ Audience are the hardware engineers.

→ Example: 64-bit processors, RISC vs CISC.

→ Example: Pipeline implementation, cache design.

~~S.M.F~~

2018 LE

2019 LS

Evolution or Generation of Computer System.
Generation refers to the involvement of technique towards technology. Computer generation is drastically changing from historical period. The following are the generation of computer:

(i) First Generation [1946 - 1957 A.D.]

All the computer which uses vacuum tubes as an electronic component are referred to as first generation of computer.

Some major characteristics of this generation are as follows.

- Very big size, unreliable and slower.
- Produce a large amount of heat.
- Operating speed was in terms of milliseconds.
- Machine level programming.
- High cost and not for commercial use.

Example: ENIAC, EDVAC, Mark-II etc.

(ii) Second Generation [1958 - 1971 A.D.]

The computers of this generation used transistor as an electronic component.

The major characteristics of second generation are as follows.

- Smaller in size than previous generation.
- 1 transistor replace 100 vacuum tubes
- faster processing speed in terms of microsecond.
- Little bit transfer I/O devices were used.
- Used assembly language.

Example: Mark-II, LEO, PDP, J etc.

(iii) Third Generation [1965 - 1971 A.D.]

In this generation the computers used IC (Integrated Circuit) as a main component. Some major characteristics of this

generation are:

- Smaller in size, more reliable.
- Generates less heat, & consumes less power
- processing speed increased to nanosecond
- Computer were portable.
- Uses high level language.
- Magnetic disk were used as secondary memory.

Example: IBM 360 series, UNIVAC-1100 etc.

(iv) Fourth Generation

The computer of this generation uses microprocessors as main electronic component. The characteristics of this generation are as follows.

- Highly accurate and totally reliable.
- operating speed in terms of picoseconds or microprocessor (MIPS)
- Magnetic and optical storage.
- multi-programming, multimedia, graphics & distributed computing is possible.
- Advancement in software package.

Example: IBM-PC, Pentium-PC etc.

(v) Fifth Generation (not yet developed)

The computers which will use AI and VLSI chip as main electronic circuit and "Bio chips" as main component are known as fifth G.C.

The features of computer of this

generation will be:

- Ability to understand natural language.
- Equivalent with multiprocessing and parallel processing.

- PLOLON and LISP as programming language.
- voice recognition & vision system will be provided.

Organization.

2022 [F].

Imp

Structure of IAS Computer

Von-Neumann Architecture

2019 (F)

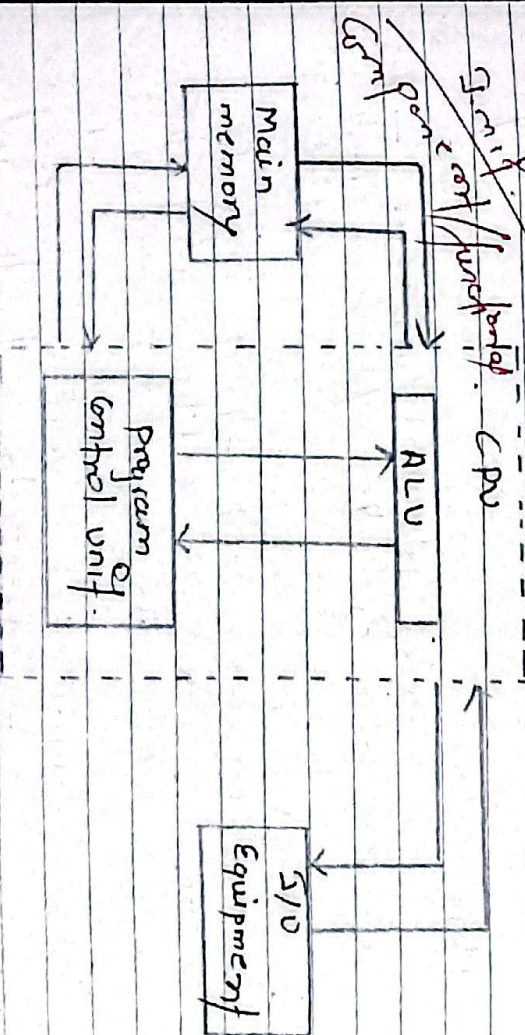


fig: General structure of IAS Computer.

It is the stored Program concept developed by von-Neumann in 1940's.

development process computers so many new concepts are invented and used one of which is stored program concept.

In this, the instruction and data are stored in memory before executing by the processor.

In 1946 A.D. von Neumann and his friends designed stored program computer called SACS (Institute for advanced study).

(i) Main memory :

It stores data and instruction before and after execution of the instruction.

(ii) ALU

It performs arithmetic & logical operation such as addition, subtraction, multiplication, AND, OR, NOT etc.

(iii) Program control unit

It interprets instruction in memory and sends them to be executed.

(iv) I/O Equipment

It performs input output operations.

Memory and instruction format of von-Neumann machine. fig below.

SACS machine consists of thousands memory location 1 location is called word and it has 40 bits. Both data and instruction are stored in these location.

The memory format are given above. A number word contain 1 bit sign bit and remaining 39 bits is the magnitude bit.

The instruction word contain two 8-bit for opcode and two 12-bits for address. The address must be started

from 0 to 999.

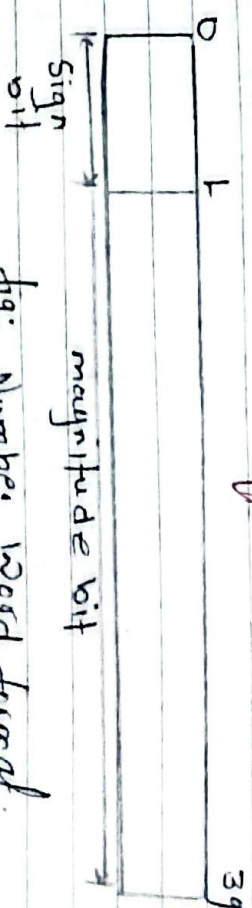


fig: Number word format.

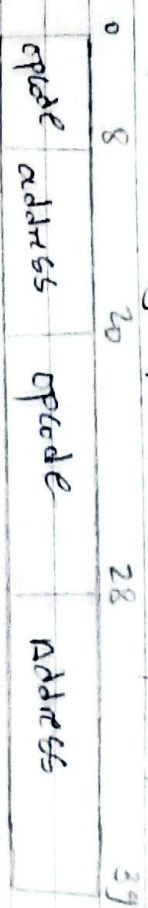


fig: Instruction word format.

Q. 1. (5)

Q. 1. Future trends in Computer:

Computer trends is drastically changing from historical period to present future period and going on the future planning. present trend is not sufficient for scientific purpose. So, Scientists and programmer are just day to day research to obtain the following future trends in the computer.

- Bio Chips
- Natural language processing.
- Artificial Intelligence (AI).
- Voice recognition system.
- Vision recognition system.
- Cloud computing.
- I/P Clouding.
- Maximum utilization of server.
- massive use of natural language processing etc.

Q. 2. Instruction, instruction set and instruction format.

Commands given to a computer system to perform specific task is called instruction. instruction set is a collection of all the instruction which can be executed by CPU. If we define all the instruction by computer, we can say that we define instruction set. Instruction means machine instruction or computer instruction.

OpCode	Reference to source code	Reference to destination operand
4-bit	6-bit	6-bit

Q. 3. Elements of instruction set.

(i) Operation code (OpCode):

It specifies the operation to perform. The operation is specified by binary code.

Example: ADD, SUB, MUL, DIV etc.

(ii) Source-operand reference!

It is a reference to operand on which data processing is performed.

(iii) destination / result operand reference.

A reference to operand which stores the result after data processing.

(iv) Next instruction reference.

A reference to which computer's alert to fetch next instruction.

Instruction format in MIPS

1) R-type format (Register-type)

operation	rs	rt	rd	shamt	function
6-bit	5-bit	5-bit	5-bit	5-bit	6-bit

Where,

operation $\rightarrow$ opcode
rs $\rightarrow$ first source operand.

rt $\rightarrow$ second source operand
rd $\rightarrow$ result operand.
shamt $\rightarrow$ shift.
function $\rightarrow$ ALU/shift specifier.

2) J-type format (Instruction-type)

operation	rs	rt	immediate
6-bit	5-bit	5-bit	16-bit

Where,

operation $\rightarrow$ opcode
rs $\rightarrow$ source operand specifier.
rt $\rightarrow$ result operand specifier.
immediate $\rightarrow$ immediate, branch or address displacement.

3) I-type format (Jump-type)

Operation	Target
6-bit	26-bit

Where, operation $\rightarrow$ opcode
Target $\rightarrow$ destination condition
(True/False)

Addressing modes:

- The method of specifying the operand (data) to be operated by the instruction is called -
- 1) Immediate Addressing mode:
- In immediate addressing mode the source operand is always data.
 - If the data is 8 bit, then instruction will be of 2 bytes, if the data is of 16-bit, then the instruction will be of 3-byte.

→ Example: MVI, LXI, ADI etc.

Assignment - 2: (CPU organization)

1. A binary computer has an instruction format as follows:
- a. Determine the memory size and the data size for the computer.
- b. How many registers does the computer have?

7	Address = 4 bits	Register code = 4 bits	Address = 16 bits
---	------------------	------------------------	-------------------

Address bit = 16 bits
 Register code = 4 bits
 Operate = 4 bits

- a) No. of address bit = $\log_2 (\text{memory size})$
- b) $16 = \log_2 (\text{memory size})$
- memory size = 256
- data size = 4 bits
- data size = 4 bits

- b) No. of register bit = $\log_2 (\text{Total Register})$
- $4 = \log_2 (\text{Total register})$
- $\therefore \text{Total register} = 2^4$
- $= 16 \text{ registers}$

2. A computer with a memory and with data word of 32 bits each. A binary instruction is stored in one memory. The instruction has four parts: an address bit, an operation code, a register code and a specify size of 178 register code part and an address part.

- a) How many bits are there in the register code, the register code and address part?
- b) Draw the instruction word format and indicate the number of bits in each part.
- c) How many bits are there in the data and address part of the memory?

Answer:
 memory size = 512 K word
 (word size) data word = 32 bits

total register = 128 register.

a) memory size = 512 K word
 $= 2^9 \times 1024 \text{ word}$

$$= 2^9 \times 2^{10}$$

$$= 2^{19}$$

$\therefore$ No. of address bit = $\log_2 (\text{m/m size})$

$$= \log_2 2^{19}$$

$$= 19 \text{ bit}$$

$\therefore$ Indirect bit = 1 bit

$\therefore$ No. of register bit = $\log_2 (\text{Total register})$

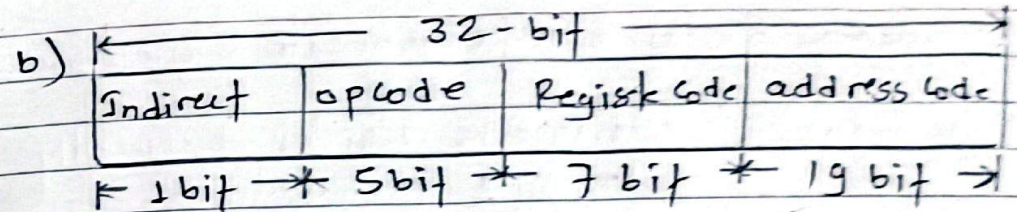
$$= \log_2 128$$

$$= \log_2 2^7$$

$$= 7 \text{ bit}$$

$\therefore$ Data bit = 32 bit

$\therefore$ opcode = $32 - (19 + 1 + 7) = 5 \text{ bits}$



c) Here,

the data input bits = 32 bits

Address input bits = 19 bits.

3] Compute the following table.

CPU organization	Instruction Address format
Single Accumulator	one address
General Register	Two address
Stack	Zero address.

4] Compute the following table.

Instruction	CPU Organization format
MOV R ₁ , R ₂	General register
ADD B	Single Accumulator
ADD	Stack
PUSH X	Stack
ADD X, R ₁ , R ₂	General register
LOAD X	Single Accumulator
STORE X	Single Accumulator.

mode of instruction	Effective Address
a) Direct	400
b) Immediate	301
c) Relative	702
d) Register Indirect	200
e) index with R ₁ as index register	600

6] How effective address is computed in different addressing modes.

⇒ Here. Solve this question, take an example, such as.

5] An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R₁ contain the number 200. Evaluate the effective address if the address mode of the instruction is

- a) Direct b) Immediate c) Relative
d) Register Indirect e) index with R₁ as index register.

⇒ Here.

Address	memory
300	instruction/Mode
301	address = 400
302	Next instruction

R₁ = 200

Given.

R₁ = 98
X_R = 3

PC = next instruction = 52.

50	load/store/mode
51	Address = 100
52	Next instruction
...	...
97	450
98	700
99	800
100	105
101	35
102	45
103	65
104	75
105	500

Now.

a) Direct addressing mode

→ The effective address is specified directly in the instruction.

→ $E.A = \text{Address field of the instruction.}$

→ Example, $100 = E.A.$

b) Immediate addressing mode.

→ The operand itself is part of the instruction, not a memory address.

→ $E.A$ is not used, operand = value given directly in the instruction.

→ Example, $E.A = 52$

c) Indirect addressing mode

→ The instruction gives a memory that contains the effective address.

→ $E.A = \text{Content of the memory location specified in the instruction.}$

→ Example; $E.A = 105$

d) Register indirect Addressing mode

→ A register holds the memory address of the operand

→ $E.A = \text{Content of register.}$

→ Example, $E.A = 98$

e) Index Addressing mode.

→ The address is calculated by adding a constant (base register) to the contents of an index register.

→ $E.A = \text{Base register (Xr) + address field.}$

→ Example, $E.A = 3 + 100 = 103$

f) Relative Addressing mode

→ The $E.A$ is calculated by adding a constant value (displacement) in the instruction to the PC.

→ $E.A = PC + \text{Address field.}$

→ Example, $E.A = 52 + 100 = 152$

7] Evaluate the arithmetic statement

$$X = A + \frac{C(D-F)}{G}$$

- a. 1-address b. 2-address
c. 3-address d. 0-address.

⇒ a) 1-address (Accumulator-Based)

Mnemonic	Operand
LOAD	D
SUB	F
MUL	C
DIV	G
ADD	A
STORE	X

b) 2-address

Mnemonic	Operand
MOV	R ₁ , D
SUB	R ₁ , F
MUL	R ₁ , C
DIV	R ₁ , G
ADD	R ₁ , A
MOV	X, R ₁

c) 3-address

Mnemonic	Operand
SUB	R ₁ , D, F
MUL	R ₂ , C, R ₁
DIV	R ₃ , R ₂ , G
ADD	X, A, R ₃

d) 0-address

Mnemonics	Operand
PUSH	D
PUSH	F
SUB	
PUSH	C
MUL	
PUSH	G
DIV	
PUSH	A
ADD	
POP	X

8] Evaluate arithmetic expression;

$$X = \frac{A}{B} + C * \frac{D}{E}$$
 using.

- a) 1-address c) 3-address
b) 2-address d) 0-address.

⇒ a) 1-address

Mnemonic	operand.
LOAD	A
DEU	B
STORE	A
LOAD	C
MUL	D
DIV	E
ADD	Z
STORE	X

b) 2-address.

Mnemonics	operand
MOV	R ₁ , A
DIV	R ₁ , B
MOV	R ₂ , C
MUL	R ₂ , D
DIV	R ₂ , E
ADD	R ₁ , R ₂
MOV	X, R ₁

c) 3-address.

Mnemonic	operand
DEU	R ₁ , A, B
MUL	R ₂ , C, D
DEU	T ₃ , R ₂ , E
ADD	X, R ₃ , R ₂

d) 0-address

Mnemonic	operands.
PUSH	A
PUSH	B
DIV	; A/B
PUSH	C
PUSH	D
MUL	; C * D
PUSH	E
DIV	; C * D / E
ADD	; (A/B) + (C * D / E)
POP	X

g) Write a program to evaluate the arithmetic statement with zero, one, two, three address instruction.

$$X = \frac{A - B + C}{G + H * K}$$

⇒ 0-address (stack based)

Mnemonic	operand
PUSH	A
PUSH	B
SUB	
PUSH	C

ADD	
PUSH	G
PUSH	H
PUSH	K
MUL	.
ADD	
DEV	
POP	X

b) (1) one-address

Mnemonic	operand.
LOAD	A
SUB	B
ADD	C
STORE	Z
LOAD	H
MUL	K
ADD	G
STORE	W
LOAD	Z
DEV	W
STORE	X

c) 2-address

Mnemonics	operands.
MOV	R ₁ , A
SUB	R ₁ , B
ADD	R ₁ , C
MOV	R ₂ , H
MUL	R ₂ , K
ADD	R ₂ , G
DEV	R ₁ , R ₂
MOV	X, R ₂

d) 3-address

Mnemonic	operand
SUB	R ₁ , A, B
ADD	R ₂ , R ₁ , C
MUL	R ₃ , H, K
ADD	R ₄ , R ₃ , G
DEV	X, R ₂ , R ₄

Q1 & Q2 ✓

RISC	CISC
→ Instruction set is simple and limited.	→ Instruction set is complex, and many instructions.
→ Instruction length fixed	→ Instruction length variable
→ Execution takes one instruction per cycle	→ Execution takes multiple cycle per instruction.
→ Pipelining easy to implement.	→ Pipelining hard to implement.
→ Memory usages is more instruction, more RAM	→ Memory usages is fewer instruction, less RAM
→ Compiler dependency is high	→ Compiler dependency is low
→ Hardware is simple	→ hardware is complex
→ Uses, phones, Tablets etc.	→ Uses PCs, laptops etc.

11. Consider the following memory and the instruction LDA 250,

250	511
432	225
511	432

R = 250
PC = 325

Write the value loaded into AC when the addressing mode is
a) Indirect b) Register Indirect c) Immediate
d) Direct.

⇒ Here, Given table,

Addressing mode	E. A	AC
a) Indirect	(432)	225
b) Register Indirect	250	511
c) Immediate	250	511
d) Direct	511	432

12. Consider the following table

$P = 300$

$R = 500$

What is the value in AC if the instruction is LDA 300 if the mode are.

i) Direct addressing.

ii) Immediate

iii) Indirect addressing

iv) Register indirect if LDA(R) is used.

Address	memory
300	450
...	...
450	200
500	100

⇒ Here,

Addressing Mode	E.A	AC
i) Direct	300	450
ii) Immediate	300	450
iii) Indirect	450	200
iv) Register indirect if LDA(R)	500	100

13] A two word instruction is stored in memory at an address designated by the symbol W.

The address field of the instruction (stored at W+1) is designated by the symbol Y. The operand used during the execution of the instruction is stored at an address symbolized by Z. An index register contains the value X. State how Z is calculated from the other address if the addressing mode of the instruction is

- direct
- indirect
- relative
- indexed.

⇒ Here, This is conceptual question.

$$X_P = X,$$

Address	Memory
W	instruction
W+1	address = Y

a) direct A.M.

$$\rightarrow E.A(Z) = Y$$

b) indirect A.M

$$\rightarrow E.A(Z) = (Y)$$

i.e. Content of Y.

c) Relative A.M
 value of z can be calculated as
 $z: y + (w + z)$
 Where $(w + z)$ address of next instruf.

d) Indexed
 $z = y + x$
 Where, x value of index register

14] An instruction is stored at memory location 400 with its address field at location 401. The address field has value 500. A processor register R_1 contain the number 250. Evaluate the effective address if the addressing mode of instruction is.

a) Direct (b) Immediate (c) Register indirect
 d) Index with R_1 as index register.

⇒ Here,

$R_1 = 250$	Address	Memory
	400	Instruction
	401	address = 500
		⋮

a) Direct
 → $E.A = 500$

b) Immediate
 → $E.A = 401$

(c) Register indirect
 → $E.A = 250$

d) Index with R_1 as index R
 → $E.A = 500 + 250$
 $= 750$

16] A two word instruction with opcode and mode field is stored at location 500 and address field at location 501. The address field has value of 700. A processor register R_1 contain the value 300. Calculate the Effective address if addressing mod is

a) Direct b) Indirect c) Immediate
 d) Register indirect.

⇒ Here. $R_1 = 300$.

a) Direct
 → $E.A = 700$

b) Indirect
 → $E.A = (300)$

i.e. The value of 700

c) Immediate
 → $E.A = 501$

d) Register indirect
 → $E.A = 300$

Address	Memory
500	opcode/mode
501	Address = 700
	⋮

⇒ An 8-bit computer has a register R.
 Determine the value of status bit C,
 S, Z and V after each of the following
 instructions. The initial value of register
 R in each case is hexadecimal 72.

The numbers below also in hexadecimal

- Add immediate operand C6 to R.
- Add immediate operand 1E to R.
- Subtract immediate operand 9A from R.
- AND immediate operand 3D to R.
- EX-OR R with R.

⇒ Given, $R = (72)_{16} = (01110010)_2$

Flag:



a) $(C6)_{16} = (11000110)_2$

$$\begin{array}{r} 01110010 \\ + 11000110 \\ \hline 10011000 \end{array}$$

Status bit

$$C = 1$$

$$S = 0$$

$$Z = 0$$

$$V = 0 \oplus 0 = 0$$

b) $(1E)_{16} = (00011110)_2$

$$\begin{array}{r} 01110010 \\ + 00011110 \\ \hline 10010000 \end{array}$$

Status bit

$$C = 0$$

$$S = 1$$

$$Z = 0$$

$$V = 1 \oplus 0 = 1$$

c) $(9A)_{16} = (10011010)_2$

$$\begin{array}{r} 01110010 \\ - 10011010 \\ \hline 01110010 \\ + 01100101 \\ \hline 11011000 \end{array}$$

Status bit

$$C = 0$$

$$S = 1$$

$$Z = 0$$

$$V = 0 (1 \oplus 1)$$

$$d) (8D)_{16} = 10001101$$

$$\text{AND } R, \quad \begin{array}{r} 01110010 \\ \wedge 10001101 \\ \hline 00000000 \end{array}$$

$$C = 0$$

$$S = 0$$

$$Z = 0$$

$$V = 0 \oplus 0 = 0$$

$$e) X-OF, R,$$

$$\begin{array}{r} 01110010 \\ \oplus 01110010 \\ \hline 00000000 \end{array}$$

$$\text{Status bit : } C = 0$$

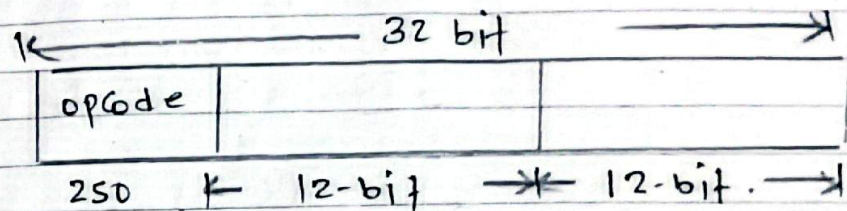
$$S = 0$$

$$V = 0$$

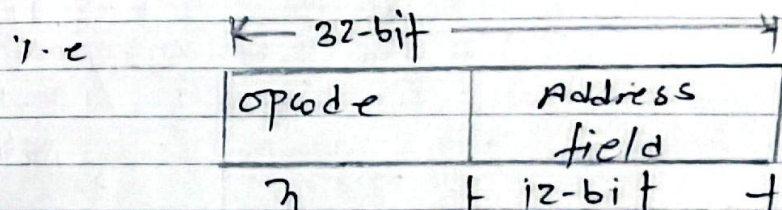
$$Z = 0$$

18] A Computer has 32-bit instruction and 12-bit addresses. If there are 250 two address instructions, how many one address instruction can be formed?

⇒ Here, Given.



Total no. of instruction in the Comp^r = 2^{32}
 total No of two address instruction = $250 \times 2^{12} \times 2^{12}$
 n-number of one address instruction are.



Now, Total instruction in the Computer
 = total 2-address instrⁿ +
 total n-address instrⁿ
 $2^{32} = 250 \times 2^{24} + n \times 2^{12}$

$$2^{32} = 4194304000 = n \times 2^{12}$$

$$\frac{4194304000}{2^{12}} = n$$

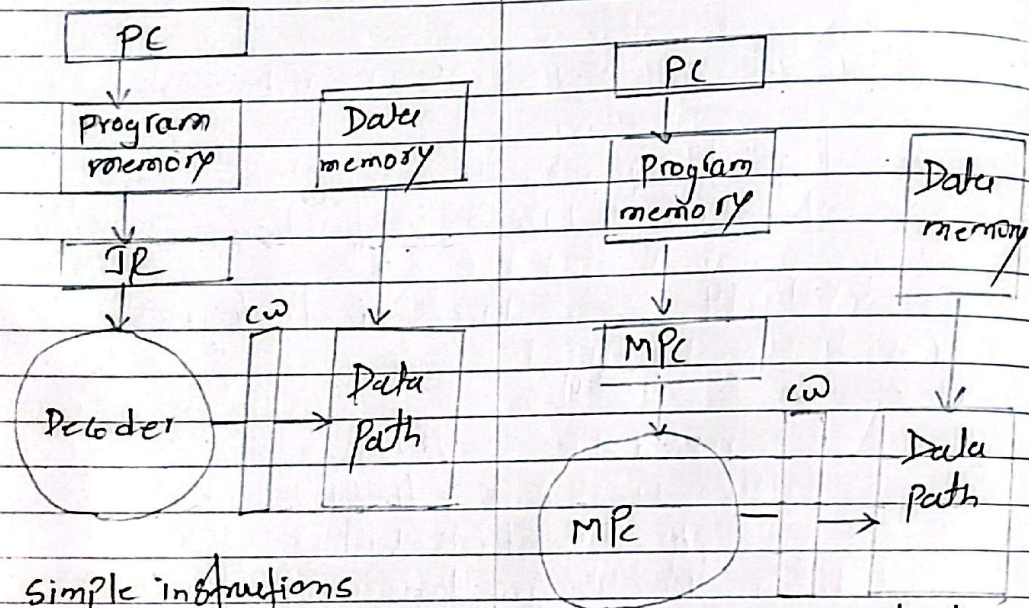
$$\therefore n = 24576$$

Q1 & Q2

Note: fig: Architecture of μ ISC and CISC.

μ ISC:

CISC:



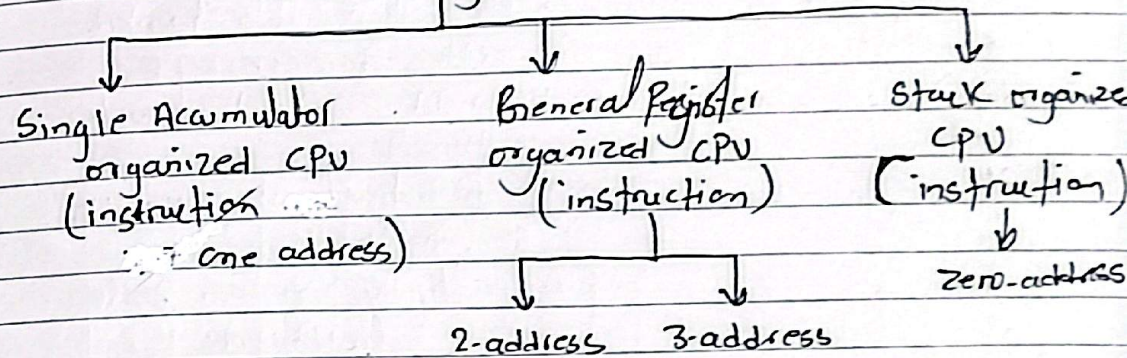
Simple instructions

one instruction = one 1w

complex instruction
one instruction = several mw

CPU organization/structure ^{the they are}

CPU organization:



CPU organization define as how the CPU can be structured within a Computer system. The CPU can be made two operate in one of the following organization.

1) Single Accumulator organized CPU

In this CPU organization all the operation are performed based on AC. All the instruction in this CPU organization are called one (1) address instruction. It means that the instruction has only one address field in the operand field. Some of the one address instruction are listed below.

Mnemonics	meaning
→ LOAD (LDA)	→ To load data from memory into AC.
→ STORE (STA)	→ To store data from 'AC' into memory
→ ADD	→ To add content of m/m with AC.
→ SUB	→ To Subtract " " " with AC.
→ MUL	→ To multiply " " " with AC
→ DIV	→ To Divide content of AC by content of memory.

2. **General Register organized CPU** → 2-address operation
 ↳ In this CPU organization all the operation are based on CPU register R_1 through R_7 .

↳ All the operation in this CPU organization are formed either by using two address or by using three address instruction formed. It means that the operand field has either two address field or three address field.

Two address Instruction.		
Mnemonics	operand	Micro operation
→ MOV	R_1, R_2	; $R_1 \leftarrow R_2$
→ MOV	R_3, A	; $R_3 \leftarrow M[A]$
→ ADD	R_1, R_2	; $R_1 \leftarrow R_1 + R_2$
→ SUB	R_1, B	; $R_1 \leftarrow R_1 - M[B]$
→ MUL	R_1, R_2	; $R_1 \leftarrow R_1 * R_2$
→ DIV	R_1, R_2	; $R_1 \leftarrow R_1 / R_2$

Three address instruction.		
Mnemonics	operand	micro operation.
→ ADD	R_1, R_2, R_3	; $R_1 \leftarrow R_2 + R_3$
→ SUB	R_1, A, B	; $R_1 \leftarrow M[A] - M[B]$ etc.

3. **Stack organized CPU**

In this CPU organization, all the operation are based on memory operation. All the instruction in this CPU organization are called zero-address instruction. As operation based instruction doesn't operand (address field). The instruction Push and POP takes the operand field based on the address. The instruction Push is used for the data entry within the stack while

instruction POP is used for retrieval of data from the stack. The operation based instruction such as ADD, SUB, MUL, DIV etc. perform operation on the top of stack. The stack operation is based on LIFO sequence.

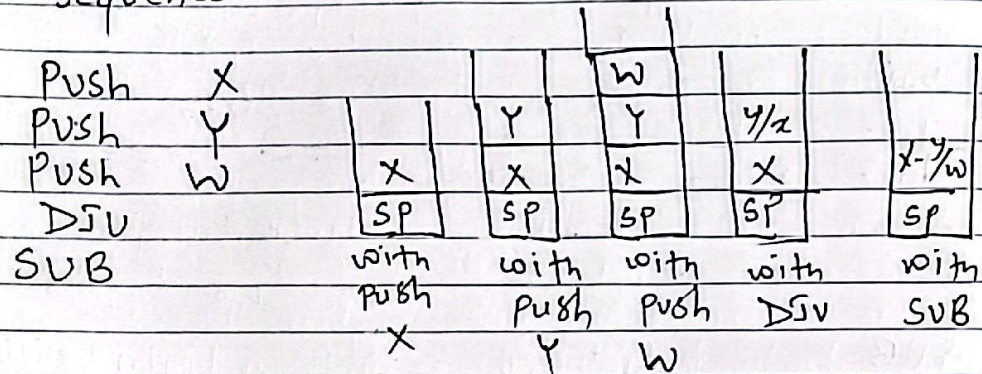


fig: Stack operation with diagrammatic representation. (-9/20/21)

Im.

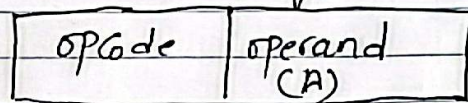
2022(F), 2024(S)

Addressing Modes:-

The method of specifying the operand (data) to be operated by the instruction is called addressing modes.

Types:-

1) Immediate addressing mode:



Instruction

Here,

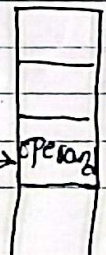
$EA = \text{operand} = A$

It is simplest addressing mode in which the operand is actually present in the instruction. This mode can be used to define and initialize value of variable. Example: `MOV A, 5000`

2) Direct addressing mode:



instruction



memory

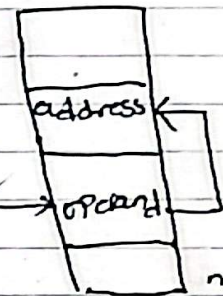
Here, $EA = A = \text{operand}$

In direct addressing mode, the address field contain effective address of the operand. It is also very simple form of addressing. Example: `MOV A, 5000` etc.

3) Indirect addressing mode:



`mov, A, (R)`



memory

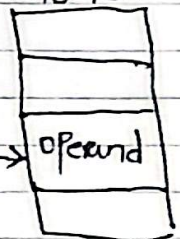
Here, $EA = A = \text{operand}$

In this addressing mode, the address field refers to the address of a word in memory which contain the full length address of the operand. Example: `MOV A, (R)` etc.

4) Register Direct addressing mode:



instruction

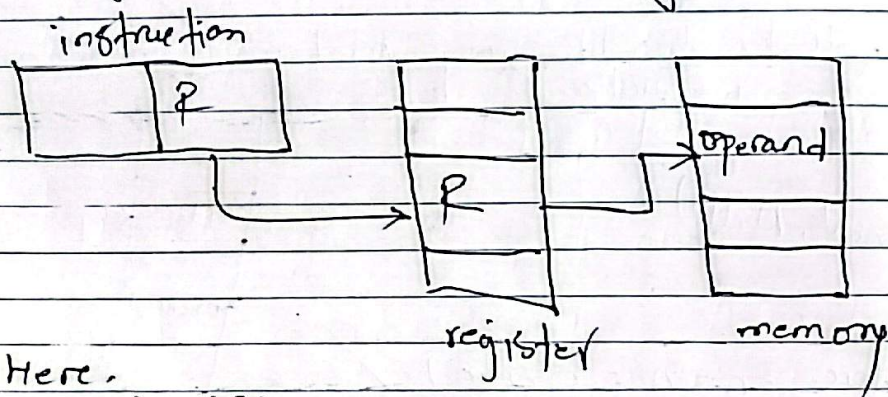


register

Example: `MOV, A, B` etc.

It is similar to direct addressing mode. The address field of the instruction refers to the register which has actual operand.

5) Register Indirect addressing mode.



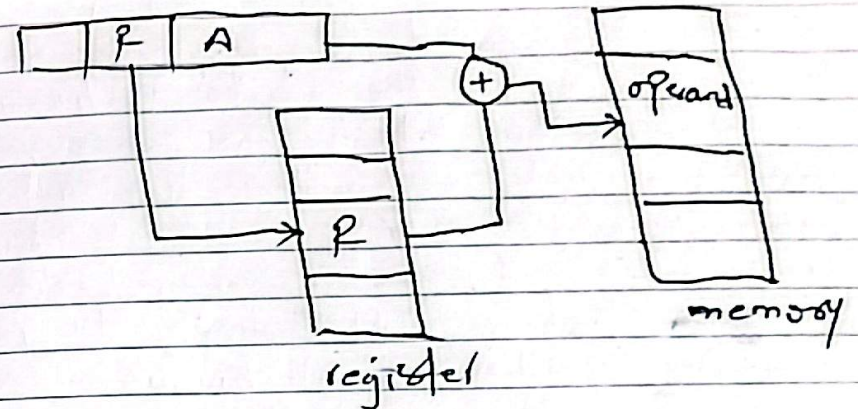
Here,

$$EA = (R) = \text{operand}$$

In this addressing mode, the address field of instruction refers to a register and this register has full length address of operand.

Example: `MOV A, (R)` etc

6) Displacement addressing mode:



$$\text{Here, } EA = (R) + A = \text{operand}$$

It's types :-

a) Relative addressing mode:

In this addressing mode, the effective address is computed by adding the content of PC with address field.

→ Ex. `JMP +5` etc.

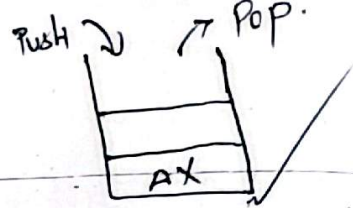
b) Index addressing mode:

In this addressing mode, the value of index register (X_p) is added with the value of address field.

→ Ex. `MOV A, 100(R)`

c) Base addressing mode:

In this addressing mode, the value of base register is added with the value of address field. Example: `MOV A, (EX)`



7) Stack addressing mode:

The stack is linearly located memory in which we can reference only from the top of that linear memory.

It is also referred to as push down list or LIFO queue. It is a temporary addressing technique.

Example: $\downarrow$ Push AX,
Pop AX, etc.

Characteristics of Computer:

- Speed: can process millions of instructions per second.
- Accuracy: performs operations with high precision.
- Automation: Executes tasks automatically after instructions are provided.
- Versatility: can perform a wide range of tasks.
- Storage: can store large amount of data for long duration.

→ multitasking: can handle multiple tasks simultaneously.

JUMP +5.

Example MOV A, 100(P1).

MOV A, (BX).

good mr

Unit - 2

2.1] Register Transfer Language (RTL):

- A digital system is a discrete system having interconnection of digital hardware modules that accomplish a specific information processing task.
- Digital systems vary in size and complexity from a few integrated circuits to a digital computer.
- The modules are constructed from the such digital components such as control logic, arithmetic elements, registers and decoders etc.
- The symbolic notation that is used to describe the micro operation transfer among the registers is called register transfer (RT).
- The term register transfer implies the availability of hardware logic circuit that can perform specific micro operation and transfer the result of operation to same or other registers.

→ A programming language used to executes the process, it is called register transfer programming language or register transfer language (RTL).

J.M.I

Q. Write the structure of VHDL programming and write the VHDL code for full adder using components.

⇒ A basic VHDL program consists of several key sections. Here's the general structure.

a). Library Declarations.

```
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;
```

b) Entity Declaration:

(5/0)
Defines the interface of the module

entity <entity_name> is
port (


```

input1 : in STD_LOGIC;
input2 : in STD_LOGIC;
output1 : out STD_LOGIC;
);
end <entity_name>;

```

C.) Architecture Body

Defines the internal working of the entity.

```

architecture <arch_name> of <entity_name> is

```

Comment: -- Component declaration signals.

```

begin
    -- Internal logic // Comment
end <arch_name>;

```

Full adder using Component in VHDL.

Code:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

```

```

entity HalfAdder is
port (
    A : in STD_LOGIC;
    B : in STD_LOGIC;
    Sum : out STD_LOGIC;
    Carry : out STD_LOGIC;
);
end HalfAdder;

```

architecture Behavioral_HA of HalfAdder is
begin

```

    Sum <= A XOR B;
    Carry <= A AND B;

```

end Behavioral_HA;

```

entity FullAdder is
port (

```

```

    A, B, Cin : in STD_LOGIC;
    Sum : out STD_LOGIC;
    Cout : out STD_LOGIC;
);
end FullAdder;

```

architecture Structural_FA of FullAdder is

Component HalfAdder

port (

A : in STD_LOGIC;

B : in " "

Sum : out " "

Carry : out " "

);

end Component;

Signal S1, C1, C2 : STD_LOGIC;

begin

HA1 : HalfAdder Port map (

A => A,

B => B

Sum => S1,

Carry => C1

);

HA2 : HalfAdder Port map (

A => S1,

B => C1,

Sum => Sum,

Carry => C2

);

-- Final Carry out

Cout <= C1 OR C2;

end Structural_FA;

4. Vhdl - Mux:

1. Vhdl code for 2-to-1 mux (Behavioral)

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity Mux2to1 is

port (

A, B : in STD_LOGIC;

Sel : in STD_LOGIC;

Y : out STD_LOGIC;

);

end Mux2to1;

Architecture Behavioral of Mux2to1 is

begin

Y <= A when Sel = '0' else B;

end Behavioral;

Of. Structural: only replace by.

architecture structural of Mux2 to 1 is
signal nsel, and1, and2: STD_logic;

```
begin
  nsel <= NOT sel;
  and1 <= A AND nsel;
  and2 <= B AND nsel;
  Y <= and1 OR and2;
```

end structural;

Similarly as,

4:1 mux (2²:1 2 selection line
input bit
input-bit.
line
3 selection line
input-bit.)

~~4 bit~~ 4 bit ALU code:
operation: A+B, A-B, A AND B, A OR B,
NOT A, A XOR B, A <=, A >=

sel (2²:0) - (3 bit) 000, 001 . .

Micro operation:

→ The elementary operation performed by a computer is called micro operation. it is classified into 4-types.

1] Data transfer micro operation:

→ In this micro operation the data is transfer from source to destination without changing the content.

→ The transfer may be between the processor, register them self, between the register to memory or between the register to I/O devices.

→ The data transfer is represented by arrowhead (←).

→ Example:

$R_1 \leftarrow R_2$; The value of register R_2 is transfer into register R_1 .

P: $R_1 \leftarrow R_2$; if (P=1) then ($R_1 \leftarrow R_2$)

Where P = Control variable (Binary variable)

or. $P: R_1 \leftarrow R_2 ; \text{if } (P=0) \text{ then } (R_1 \leftarrow R_2).$

2) Arithmetic micro operation:

→ In this microoperation, arithmetic operation such as addition (ADD), subtraction (SUB), multiplication (MUL) and division (DIV) are performed.

→ Example:

$R_1 \leftarrow R_1 + R_2$; Add R_1 and R_2 and the sum is store in R_1 .

$P: R_1 \leftarrow R_1 + R_2 ; \text{if } (P=1) \text{ then } (R_1 \leftarrow R_1 + R_2).$

3) Logical micro operation:

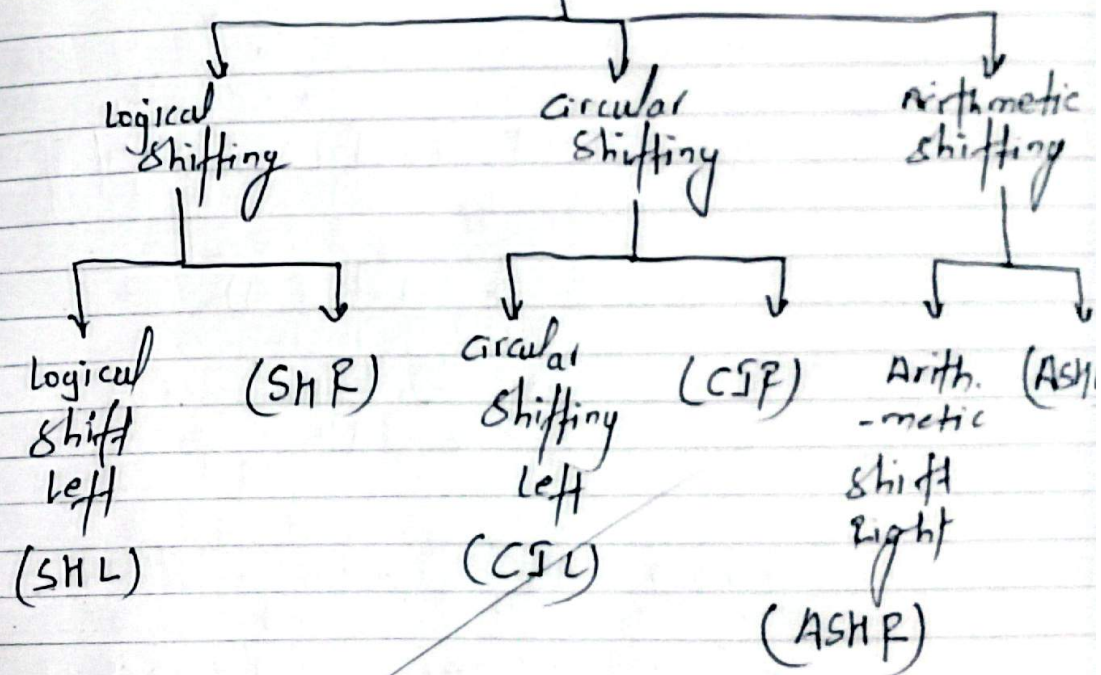
→ In this microoperation, the logic gate such as AND, OR, NOT, Ex-OR, Ex-NOR and ~~NAND~~ NAND etc are used to performed logical operation.

→ Example:

$R_1 \leftarrow R_1 \wedge R_2$; logically AND R_1 and R_2 and result transferred

$P: R_1 \leftarrow R_1 \vee R_2 ; \text{if } (P=1) \text{ then } (R_1 \leftarrow R_1 \vee R_2).$

4) Shift micro operation.



Exm. 1

formula:

to decide Bit size.

$$n = \lceil \log_2(x) \rceil + 2$$

$$= \log_2(17) + 2$$

$$= 6$$

max-value.

$$\text{max value} = \text{max}(|\text{Multiplicand}|, |\text{multiplier}|)$$

Booth Algorithm.

Q. 1] 17×4 2] $(12) \times (-9)$

3] -13×7 4] $9 \times (-3)$

5] $(-5) \times (-3)$

Q. unsigned binary multiplication method.

$(14)_{10}$ and $(3)_{10}$

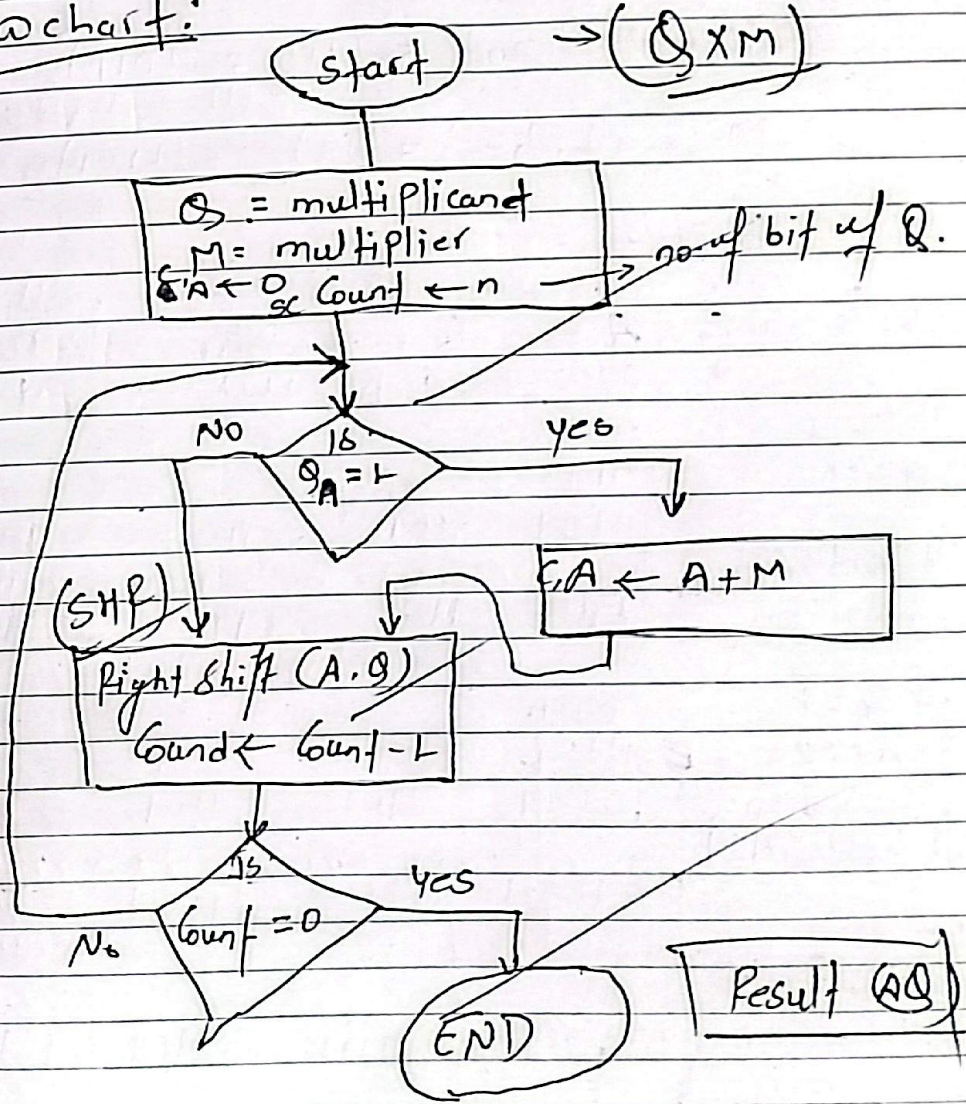
→ Signed multiplication - $(16)_{10} \times (-9)_{10}$

Q. use Signed magnitude data $(9) \times (-2)$.
↓ 2's complement method $(7) \times (-3)$

unit-5

5.4] Unsigned Binary multiplication Algorithm

flowchart:



Example (a)

$$(11)_{10} \times (13)_{10}$$

⇒ Here.

$$Q(\text{multiplicand}) = (11)_{10} = (1011)_2$$

$$M(\text{multiplier}) = (13)_{10} = (1101)_2$$

$$\text{Count}(n) = 4$$

initial	C	A	Q	M	n
	0	0000	1011	1101	4
check $Q_0 = 1$ $A \leftarrow A + M$	0	0000 +1101 1101	1011	1101	4
Shift Right (C, A, Q) $n \leftarrow n - 1$	0	0110	1101	1101	3
check $Q_0 = 1$ $A \leftarrow A + M$	0	0110 +1101 0011	1101	1101	3
Shift Right (C, A, Q) $n \leftarrow n - 1$	0	1001	1110	1101	2
check $Q_0 = 0$ (No)					
Shift Right (C, A, Q) $n \leftarrow n - 1$	0	0100	1111	1101	1

check $Q_0 = 1$ $C, A \leftarrow A + M$	1	0100 +1101 0001	1111	1101	1
Shift Right (C, A, Q) $n \leftarrow n - 1$	0	1000	1111	1101	0

Now.

$$(11)_{10} \times (13)_{10} = (1011)_2 \times (1101)_2$$

$$= (AC)_2$$

$$= (10001111)_2$$

$$= (143)_{10}$$

good

Ex. (b) $(14)_{10} \times (3)_{10}$

⇒ Here, $Q(\text{multiplicand}) = (14)_{10} = (1110)_2$

$$M(\text{multiplier}) = (3)_{10} = (0011)_2$$

$$\text{Count}(n) = 4$$

$$C, A \leftarrow 0$$

initial	C	A	Q	M	n
	0	0000	1110	0011	4
check $Q_0 = 0$ (No)					
Shift Right (C, A, Q)	0	0000	0111	0011	3
$n \leftarrow n-1$					
check $Q_0 = 1$ (Yes)		0000			
C, A $\leftarrow$ A + M	0	+0011			
		0011	0111	0011	3
Shift Right (C, A, Q)			1011		
$n \leftarrow n-1$	0	0001	0111	0011	2
check $Q_0 = 1$		0001			
C, A $\leftarrow$ A + M	0	+0011			
		0100	1011	0011	2
Shift Right (C, A, Q)	0	0010	0101	0011	1
$n \leftarrow n-1$					
check $Q_0 = 1$		0010			
C, A $\leftarrow$ A + M	0	+0011			
		01001	0101	0011	1
Shift Right (C, A, Q)	0	0010	1010	0011	0
$n \leftarrow n-1$					

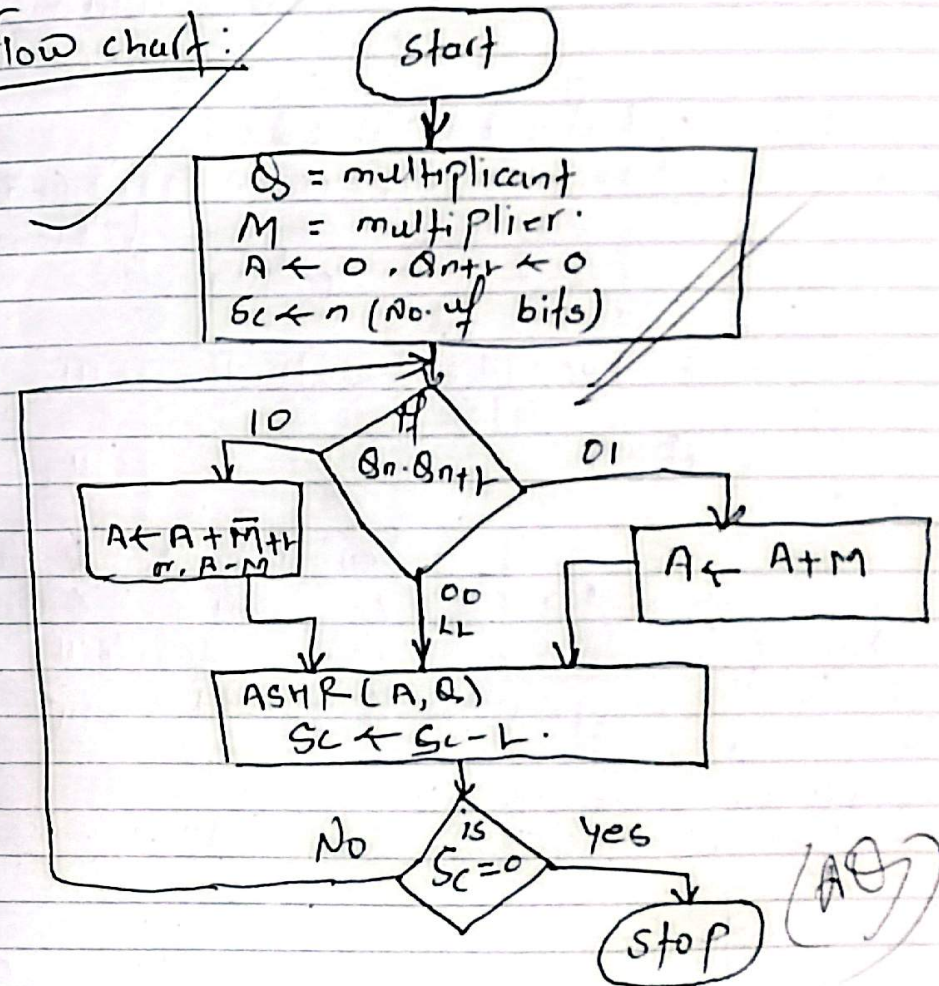
$$\text{Now, } (14)_{10} \times (3)_{10} = (AQ)_2 = (00101010)_2$$

$$= (42)_{10} \checkmark \#$$

S.S J.

Booth multiplication. (2's comp method)
 or,
 Signed multiplication.
 or,
 Signed magnitude data.

flow chart:



Example: $(-12) \times (-9)$

here. $Q(\text{multiplicand}) = (-12) = 1100$

$$\begin{array}{r} 1100 \\ + 0001 \\ \hline 0100 \end{array}$$

1's complement

$$M(\text{multiplier}) = (-9) = 1001$$

$$\begin{array}{r} 0110 \\ + 1001 \\ \hline 0111 \end{array}$$

1's complement
2's complement

$$\bar{M} + 1 = 1000 + 1 = 1001$$

$$S_c \leftarrow n = 4$$

initial	A	Q	Q _{n+1}	M	S _c
	0000	0100	0	0111	4
check Q _n Q _{n+1} = 00					
ASHF(A, Q, Q _{n+1})	0000	0010	0	0111	3
S _c ← S _c - 1					
check Q _n Q _{n+1} = 00					
ASHF(A, Q, Q _{n+1})	0000	0001	0	0111	2
S _c ← S _c - 1					
check Q _n Q _{n+1} = 10	0000 + 1001 1001				
A ← A + $\bar{M}$ + 1		0001	0	0111	2

(106)

$$\begin{array}{r} \text{ASHF}(A, Q, Q_{n+1}) \\ S_c \leftarrow S_c - 1 \\ \text{check } Q_n Q_{n+1} = 01 \\ A \leftarrow A + M \end{array}$$

$$\begin{array}{r} 1100 \\ + 0111 \\ \hline 0011 \end{array}$$

1000 1 0111 1

$$\begin{array}{r} \text{ASHF}(A, Q, Q_{n+1}) \\ S_c \leftarrow S_c - 1 \end{array}$$

0001 1100 0 0111 0

Now, Here, result should be 2's complement form, so.

$$\begin{array}{r} (AQ) = 00011100 \\ = 11100011 \\ + 1 \\ \hline 11100100 \end{array}$$

1's complement
2's complement

here. $Q(\text{multiplicand}) = (-12) = 01100$

$$\begin{array}{r} = 10011 \\ + 1 \\ \hline 10100 \end{array}$$

2's complement

$$M(\text{multiplier}) = (-9) = 01001$$

$$\begin{array}{r} = 10110 \\ + 1 \\ \hline 10111 \end{array}$$

1's complement

$$\bar{M} + 1 = 01000 + 1 = 01001$$

$$SC = n = 5$$

initial	A	Q	Q _{n+1}	M	SC
	00000	10100	0	10111	5
check Q _n .Q _{n+1} = 00					
ASHF(A, Q, Q _{n+1})	00000	01010	0	10111	4
SC ← SC - 1					
check Q _n .Q _{n+1} = 00					
ASHF(A, Q, Q _{n+1})	00000	00101	0	10111	3
SC ← SC - 1					
check Q _n .Q _{n+1} = 10	00000 +01001				
A ← A + M + L	01001	00101	0	10111	3
ASHF(A, Q, Q _{n+1})	00100	10010	1	10111	2
SC ← SC - 1					
check Q _n .Q _{n+1} = 01	00100 +10111				
A ← A + M	11011	10010	1	10111	2
ASHF(A, Q, Q _{n+1})	11101	11001	0	10111	1
SC ← SC - 1					
check Q _n .Q _{n+1} = 10	11101 +01001				
A ← A + M + L	00110	11001	0	10111	2
ASHF(A, Q, Q _{n+1})	00011	01100	1	10111	0
SC ← SC - 1					

$$\text{Ans. } (AQ) = (0001101100)$$

$$= (108) \quad \text{good}$$

Ex.(b) 17x4

⇒ here

$$Q = 17 = 10001$$

$$M = 4 = 00100$$

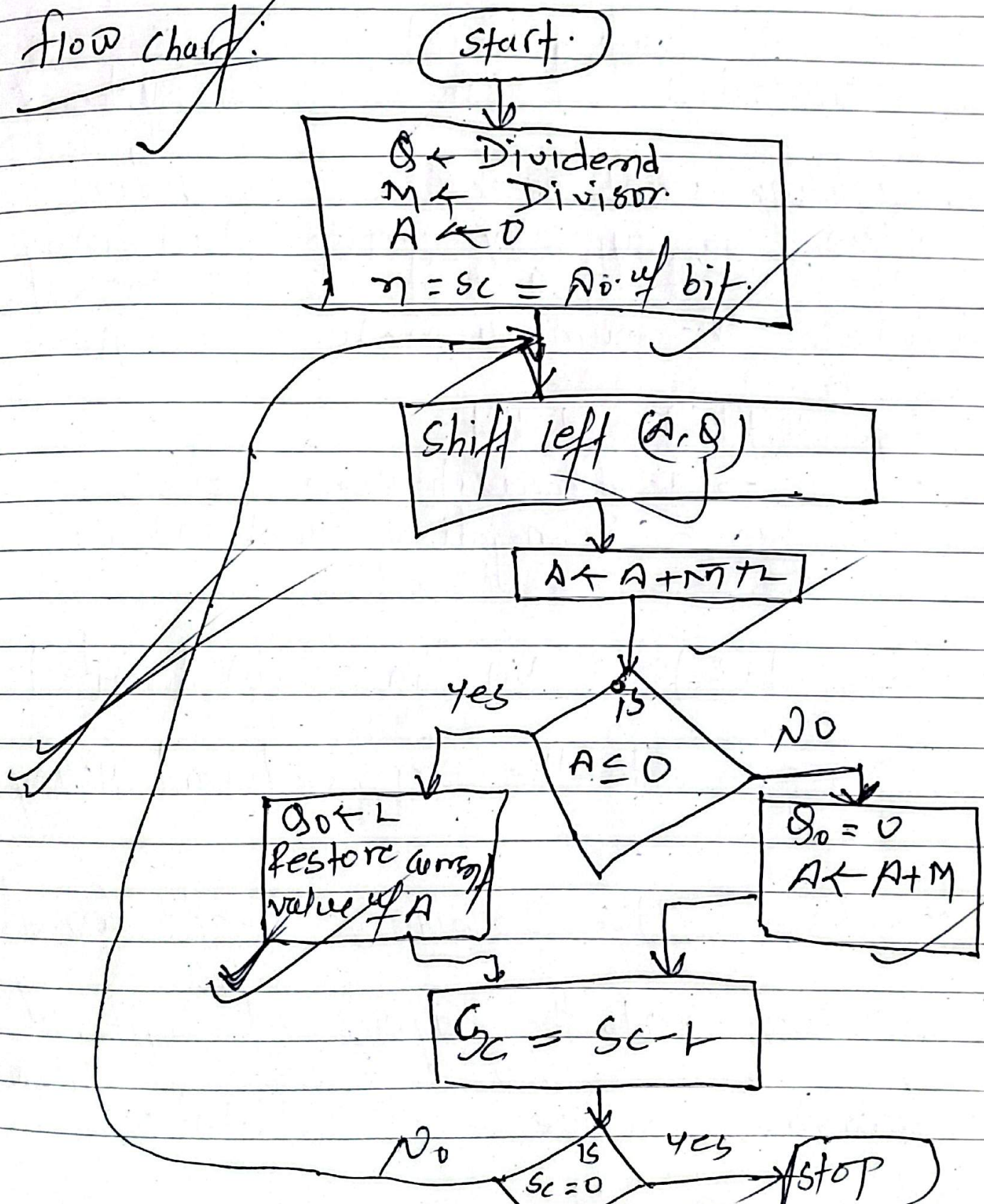
$$SC = n = 5$$

$$\bar{M} + L = 11011 + L = 11100$$

initial	A	Q	Q _{n+1}	M	SC
	00000	10001	0	00100	5
check Q _n .Q _{n+1} = 10	00000 +11100				
A ← A + M + L	11100	10001	0	00100	5
ASHF(A, Q, Q _{n+1})	11110	01000	1	00100	4
SC ← SC - 1					
check Q _n .Q _{n+1} = 01	11110 +00100				
A ← A + M	00010	01000	1	00100	4
ASHF(A, Q, Q _{n+1})	00001	00100	0	00100	3
SC ← SC - 1					

5.6] Unsigned Binary Division Algorithm.

flow chart:



5.7 Floating point Representation:

Q. How a floating point number is represented in Computer System?

→ A floating point number in a Computer System is typically represented using the IEEE 754 Standard, which divides the number into three types.

a) Sign bit (S): 1 bit
 → 0 for positive numbers.
 → 1 for negative numbers.

b) Exponent (E): 8-bit [in 32-bit format]

→ represents the exponent part using bias.

c) Mantissa (M): 23 bit (in 32-bit format)

→ Represents the significant digits of the number.
 formula: value = $(-1)^S \times (1.M) \times 2^{(E-Bias)}$

Q. 17×4

⇒ Here, $S(\text{multiplicand}) = 17 = 010001$
 $M(\text{multiplier}) = 4 = 000100$

$$S_c = n = 6$$

$$\bar{M} + L = 111011$$

$$\begin{array}{r} +L \\ 111100 \end{array}$$

initial	A	S	S_{n+L}	M	S_c
	000000	010001	0	000100	6
check $S_n \cdot S_{n+L} = 10$	000000				
$A \leftarrow A + \bar{M} + L$	+111100				
	111100	010001	0	000100	6
$ASHF(A, S, S_{n+L})$	111110	001000	1	000100	5
$S_c \leftarrow S_c - 1$					
check $S_n \cdot S_{n+L} = 01$	111110				
$A \leftarrow A + M$	+000100				
	000010	001000	1	000100	5
$ASHF(A, S, S_{n+L})$	000001	000100	0	000100	4
$S_c \leftarrow S_c - 1$					
check $S_n \cdot S_{n+L} = 00$	000000	100010	0	000100	3
$ASHF(A, S, S_{n+L})$					
$S_c \leftarrow S_c - 1$					
check $S_n \cdot S_{n+L} = 00$	000000	010001	0	000100	2

(68)

check $S_n, S_{n+1} = 10$ $A \leftarrow A + \bar{M} + L$	000000 + 111100 ----- 111100	010001	0 000100	2
ASHF(A, S, S _{n+1}) $SC \leftarrow SC - L$	111110	001000	1 000100	L
check $S_n, S_{n+1} = 01$ $A \leftarrow A + M$	111110 + 000100 ----- 000010	001000	1 000100	L
ASHF(A, S, S _{n+1}) $SC \leftarrow SC - L$	000001	000100	0 000100	0

Now, $17 \times 4 = (010001) \times (000100)$

$\Rightarrow (AQ) = 000001000100$

$= (68)$

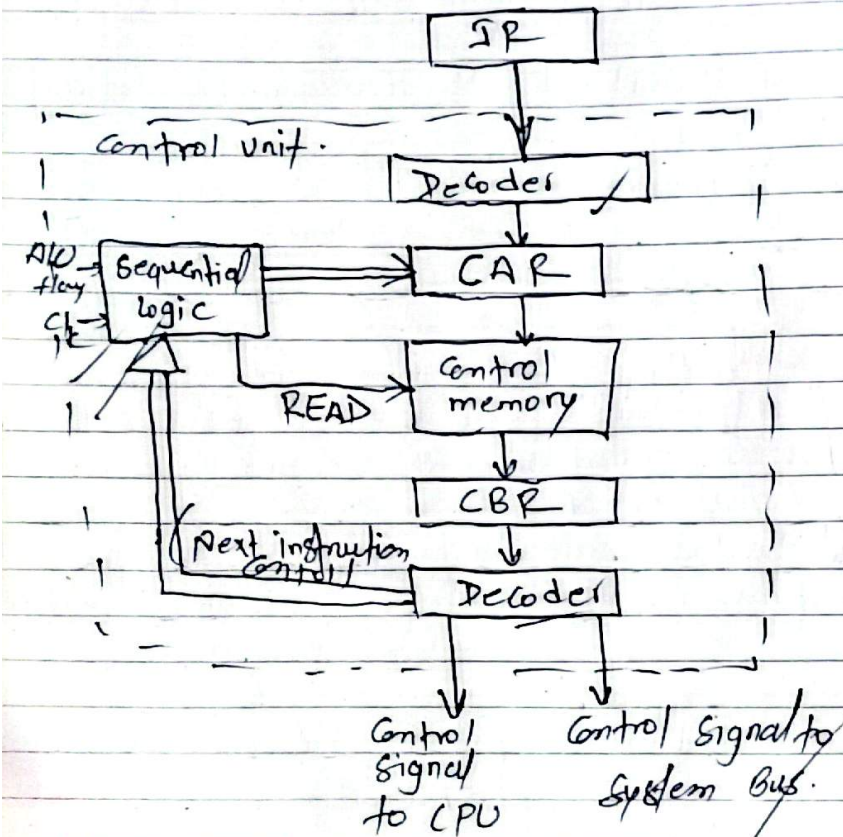
good

Unit-4

Difference: Hardwired Control unit & Micro-programmed Control unit:

Hardwired Control unit	Micro-programmed Control unit
→ It implies hardware Control unit	→ It is a midway between hardware and software.
→ on the basis of input signal, output signal is generated.	→ Micro operation generates sequence of micro instruction on the control line.
→ It is difficult to design, test and implement.	It is easy to design, test, less error and implementation.
→ It is faster mode of operation	→ It is slower mode of operation.
→ It is used in RISC processor	→ It is used in CISC process.
→ It is inflexible to modify.	→ It is flexible to modify.

Architecture of μ -programmed
Control unit
4.5] or, function of micro-programmed.



4.2] functioning of Hardwired Control unit.

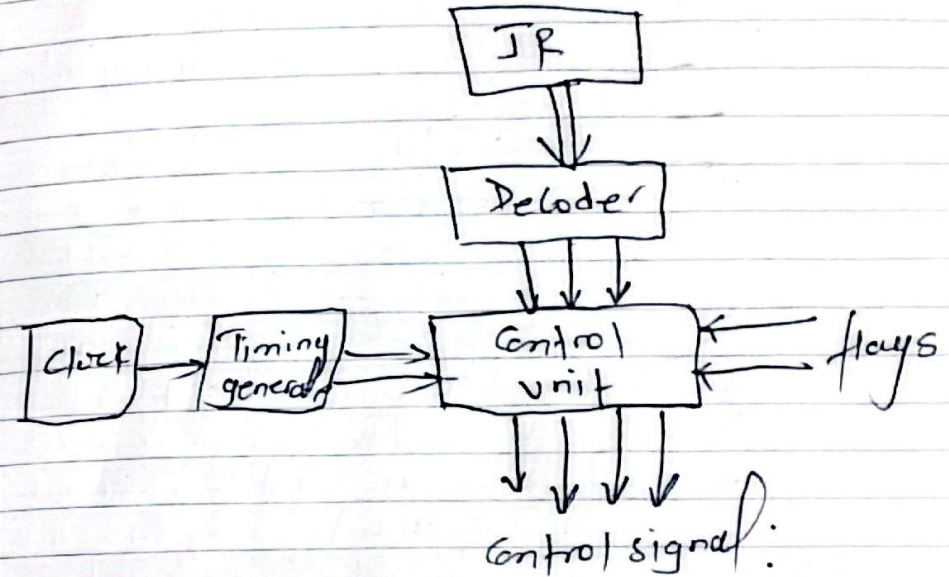


fig:

4.3 Microinstruction format:

→ It is a symbolic notation that is used in micro programming language to describe the microoperation.

Normally, μ -instruction is also known as μ -program or sequence of instruction.

The format of microinstruction must include the following field.

- μ -instruction address
- Jump Condition (unconditional branch, zero and overflow).
- System bus control signals.
- Internal CPU control signals.

The μ -instruction format are classified into two categories.

- Vertical μ -instruction format
- Horizontal μ -instruction format.

A) Vertical μ -instruction format:

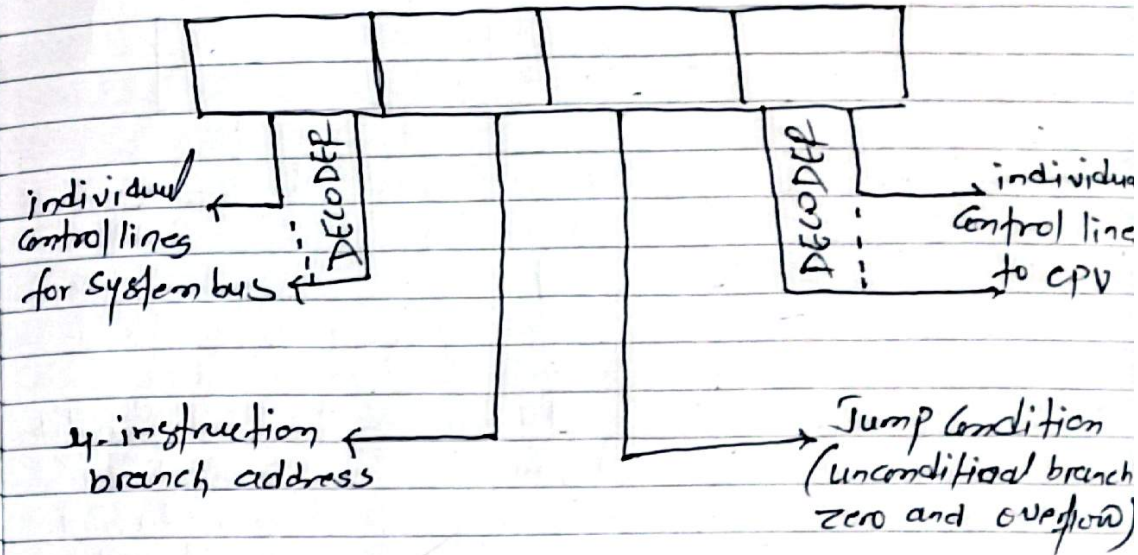
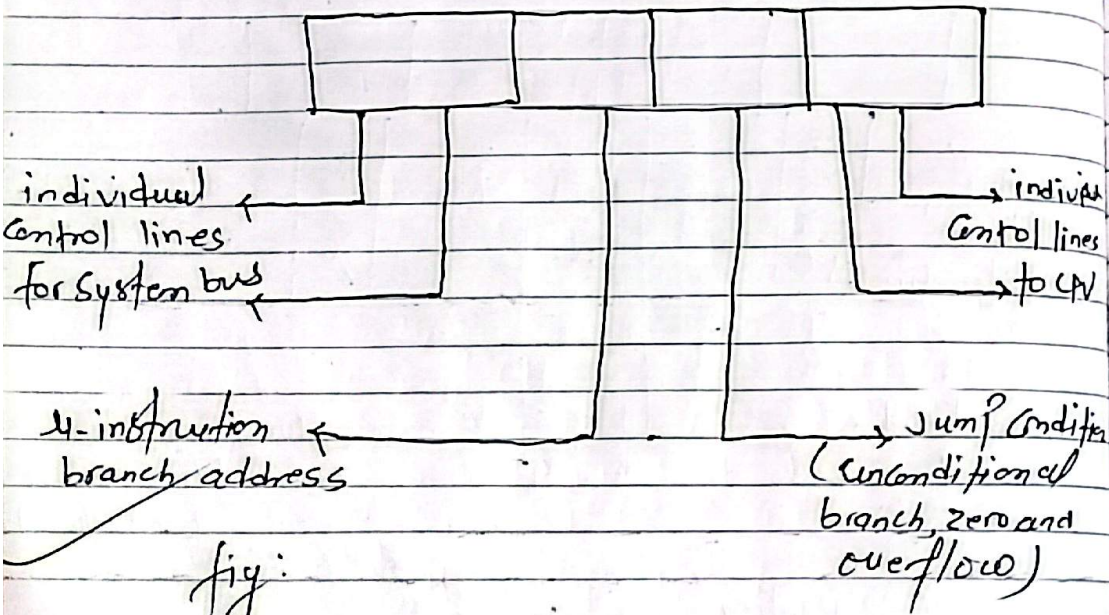


fig: vertical
→ In

BT Horizontal 4-instruction format:



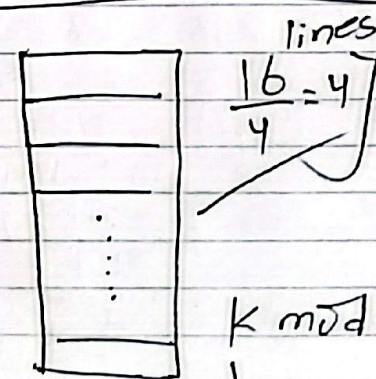
7

7.6.] Cache Replacement Algorithm

FIFO (first in first out)

Q. Consider a 4-way set associative cache with total 16 cache blocks/lines. main memory block requests are:

0, 255, 1, 4, 3, 8, 133, 159, 216, 129, 63, 48, 32, 73, 92, 155.



S ₀	0, 4, 8, 216, 48, 32, 92
S ₁	1, 133, 229, 73
S ₂	
S ₃	255, 3, 159, 63

155

$k \bmod n$

no. of hit = 1

block number

no. of set

block/lines

~~unit~~

least frequently used.

LRU

Q. same:

formula:

S0	10, 4, 8, 2, 16, 8, 48, 32, 92
S1	1, 183, 129, 73
S2	
S3	285, 8, 159, 63, 155

Q. Consider a fully associative cache with 8 cache blocks/lines (0-7) and the following sequence of memory block requests. M M M M M M H H
4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45
22, 8, 3, 16, 25, 7
if LRU replacement policy is used. Which cache block will have memory block 7?

⇒

0	4	45
1	3	22
2	25	
3	8	
4	19	3
5	6	7
6	16	
7	35	

No. of hit = 5

lines/block now 5

cache memory

LFU [least frequently used]

Q. reference strip: 4, 3, 2, 1, 4, 3, 5, 4, 3, 2, 1, 5
free frame = 3.

4	1	3	1	2	1	1	4	2	3	2
		3	1	3	1	2	1	1	4	2
		4	2	3	1	2	2	1	1	1

fault.

3	2	4	3	3	3	3	3
4	2	3	2	4	3	4	3
5	1	5	1	5	1	2	1

X X
hit hit

3
4
5

No. of page fault/mis = 10.

✓ (amt)

Optimal Paye Replacement (replace the paye which is not used in logest Dimension of time in future).

Frame size = 7

Future check. →

adj. String: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0,
3, 2, 1, 0, 2, 1, 0, 1, 7, 0, 1

[illegible]

feud. hit

2/ No. Demand
then change
costly.

No. of paye hit = 12

No. of Page fault/miss

$$\text{radio bit} = \frac{12}{20} \times 100$$

Unit: 10

10.1] Hardware Performance issues:

- Increase Cost
- Increase Parallelism
- Power Consumption.
- Alternative Chips organization.
- Extra hardware Component is required.

Software Performance issue.)

- Compatibility with software.
- Software on multicore processor must be specified.
- Platform dependent.
- Unsuitable environment may affect on a software quality.

10.5] Power Efficient processor (PEP):

10.3] multicore Organization:

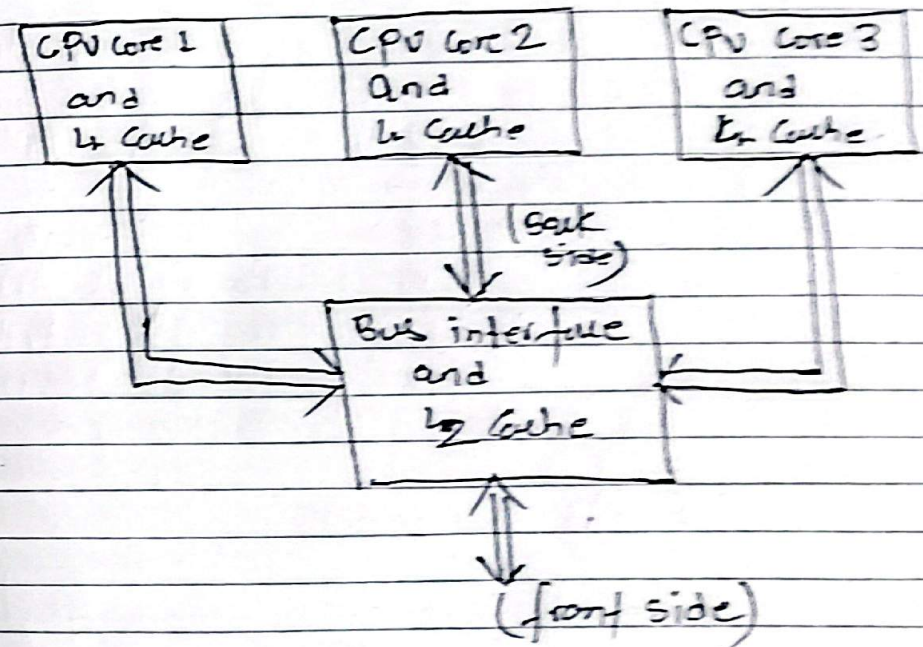


fig: multicore organization.

Structure of VHDL :-

a) Library Declaration:

```
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;
```

b) Entity Declaration:

Defines the interface I/O of the module.

```
entity <entity_name> is  
    port (  
        input 1 : in STD_LOGIC;  
        input 2 : in STD_LOGIC;  
        output 3 : out STD_LOGIC;  
    );  
end <entity_name>;
```

c) Architecture Body

Defines the internal working of the entity.

```
architecture <arch_name> of <entity_name>  
    is  
begin
```

-- ~~Logic~~ Internal logic

```
end <arch_name>;
```

Full adder using Component in VHDL

```
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity HalfAdder is  
    port (  
        A, B : in STD_LOGIC;  
        Sum, Carry : out STD_LOGIC;  
    );  
end HalfAdder;
```

Architectural Behavioral HA of HalfAdder is
begin

```
Sum <= A XOR B;  
Carry <= A AND B;
```

end Behavioral HA;

```
entity fullAdder is  
    port (  
        A, B, Cin : in STD_LOGIC;  
        Sum, Cout : out STD_LOGIC;
```



```

    A, B, cin : in STD_LOGIC;
    Sum, Carry : out STD_LOGIC;
);
end FullAdder;

```

architecture Structural_FA of fullAdder is
~~begin~~ Component HalfAdder
 port (

```

        A, B : in STD_LOGIC;
        Sum, Carry : out STD_LOGIC;
    );
end Component;
Signal Sr, C1, C2 : STD_LOGIC;

```

begin

H1 : HalfAdder port map (

```

    A => A,
    B => B,
    Sum => Sr,
    Carry => C1
);

```

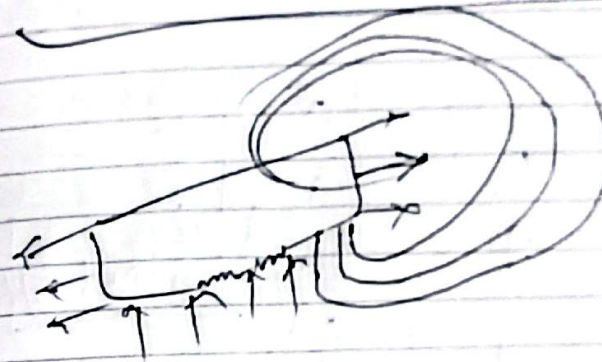
H2 : HalfAdder port map (

```

    A => Sr,
    B => C1,
    Sum => Sum,
    Carry => C2
);

```

Car1 <= C1 OR C2
end Structural_FA;



Structure of ~~ISA~~ IAS Computer:

Von. neumann Architecture

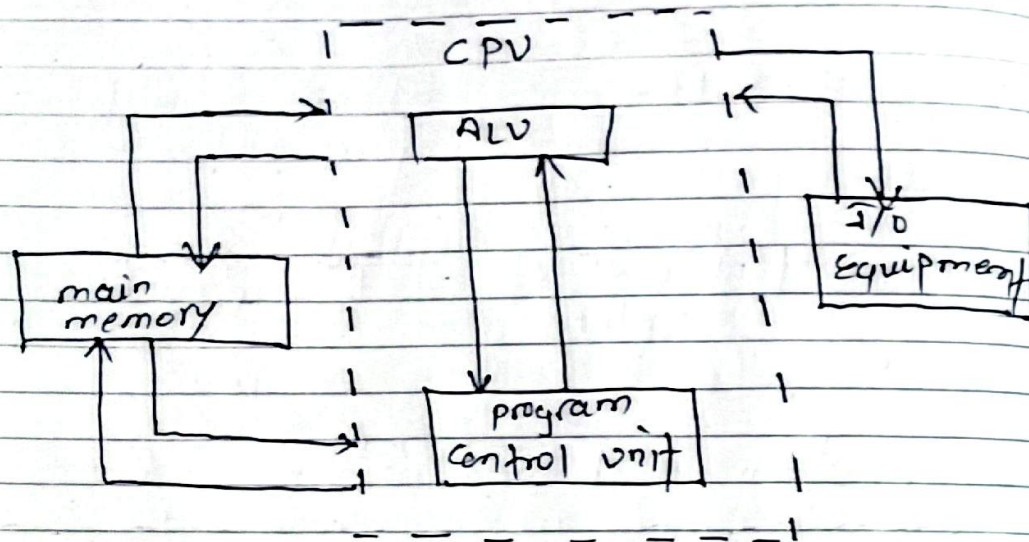


fig: Structure of IAS Computer

GPU

→ Graphics processing unit

→ Very flexible - can handle graphics, simulation AI, video processing and more.

→ Fast, but not as fast as TPV's for specific deep learning workload

→ High power usage, especially high GPUs

→ widely available from companies like NVIDIA, AMD, Intel

→ Example: video rendering, 3D animation etc.

TPV

→ Tensor processing unit

→ less flexible - best for machine learning, not for general purpose computing

→ faster than GPUs for large scale deep learning training and inference

→ Generally more power efficient for ML tasks

→ mostly available through google cloud.

→ Example: Deep learning training and inference

Associative memory

Cache memory:

- search data by content
- it is content based addressing.
- Extremely fast for searches
- It is small in size
- Example of use: Cache for matching routing table.
- Store frequently accessed data for fast retrieval
- Address based.
- very fast for stored data
- it is smaller than RAM but bigger than Associative memory
- CPU instruction/data storage for quick access.

main memory

Auxiliary memory:

- it is primary memory
- It stores data and instruction currently in use by the CPU
- it is volatile
- It is very fast
- It cost per GB is high
- Example: RAM, cache memory.
- Directly accessed by CPU.
- Secondary storage
- Store data and program permanently or long term.
- it is non-volatile.
- much slower as compared to main memory
- low
- Example: Hard drive, CD, DVD, pen drive
- Accessed indirectly via main memory