

Q 1) A digital computer has a common bus system for 16 registers of 32 bits each. The bus is constructed with multiplexers.

(a) How many selection lines are there in each multiplexers?

(b) What size of MUX are needed?

(c) How many multiplexers are there in the bus?

solⁿ

Given,

No of register = 16 registers

Each register has 32 bits.

(a) No of selection lines in each register = $\log_2(16)$
 $= \log_2(2^4)$
 $= 4$

(b) size of multiplexers $\Rightarrow$ No of register $\times$ No of i/p to MUX
 $16 = \text{No of i/p to MUX}$

Hence, 16:1 MUX are required.

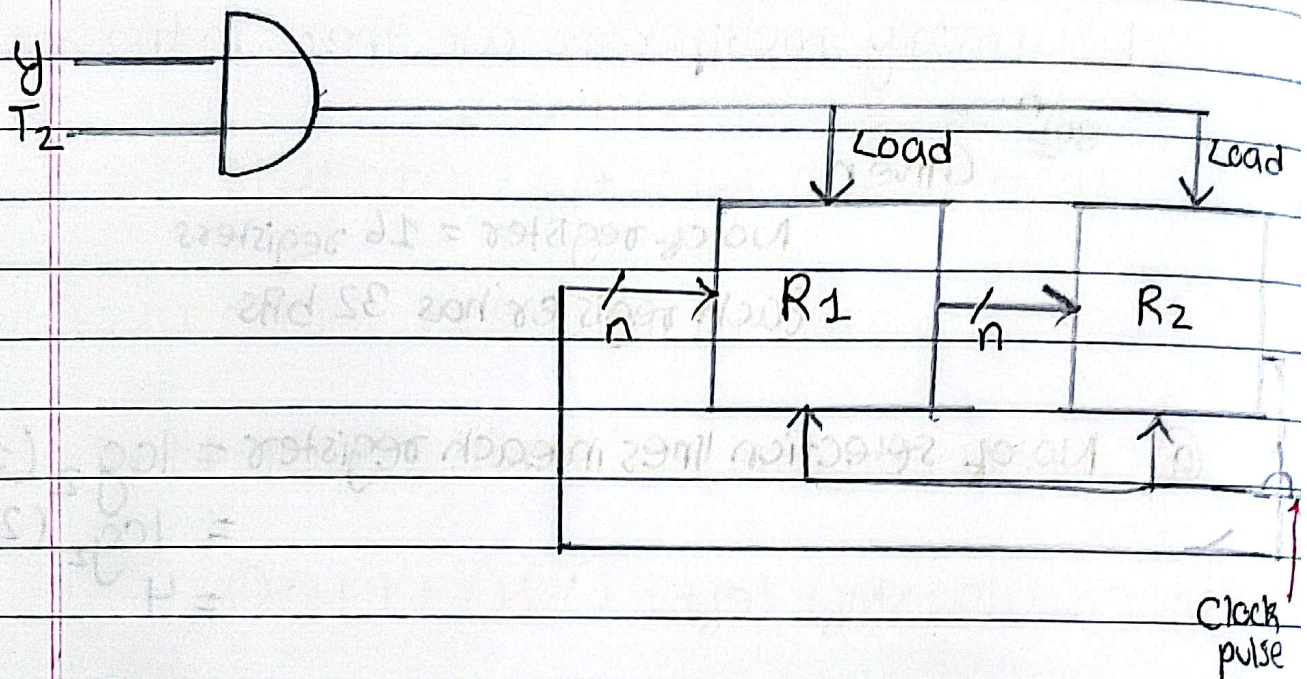
(c) For number of multiplexers required:

No of MUX = No of bits in register.
 $= 32\text{-bit}$

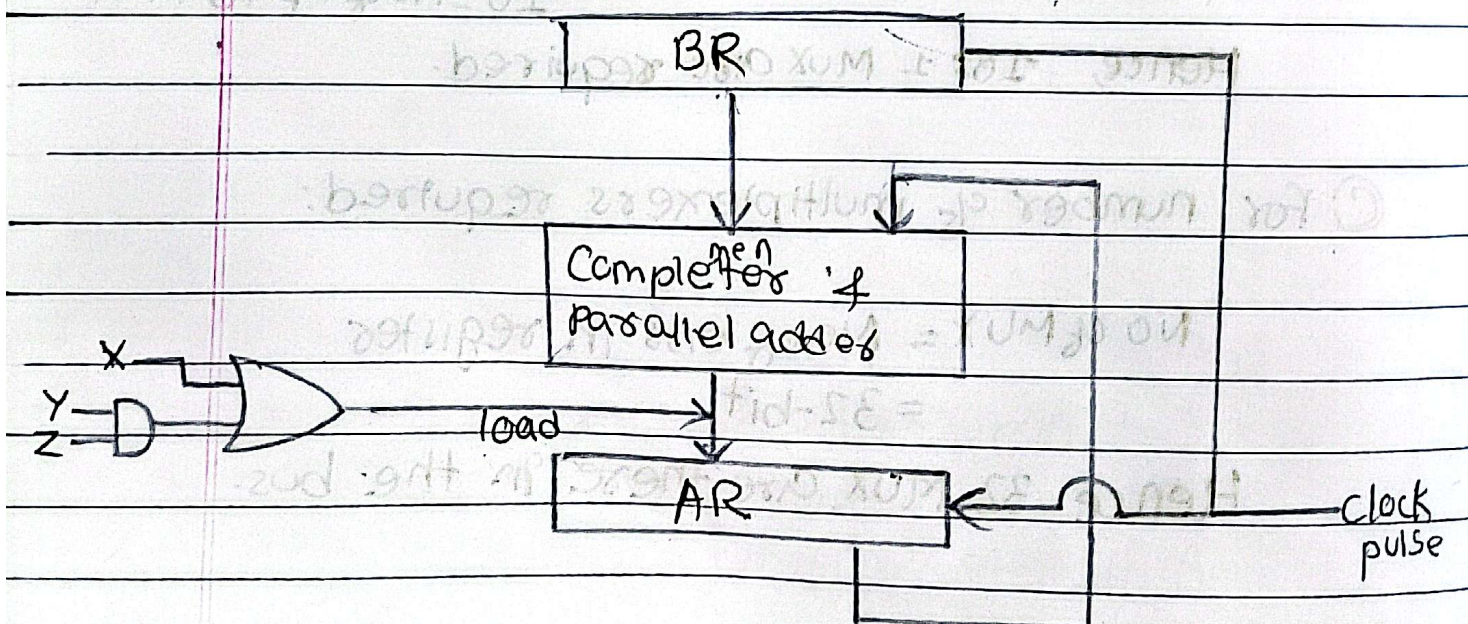
Hence 32 MUX are there in the bus.

2 (2) Show the block diagram of the hardware that implements the following register transfer statement.

(a) $yT_2: R_2 \leftarrow R_1, R_1 \leftarrow R_2$



(b) $x+yz: AR \leftarrow AR + BR$



- 3
5 ③ A 8-bit registers contain the binary value the $R = 10011101$, determine the sequence of binary values in R after logical shift left, followed by a circular shift-right, followed by a logical shift-right and circular shift-left.

soln

Here, $R = 10011101$

- ① Logical shift left: 0100111010 (LSHL) ^{Multiply by 2} ~~Divide by 2~~
- ② Circular shift right (CSR): 00011101
- ③ Logical shift right (SHR): 00001110
- ④ Circular shift left: 00011100

- 7
④ 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 after an arithmetic shift left, and state whether there is an overflow.

soln

Binary value of register = 11011100

Arithmetic shift Right (ASHR) = 11101110 ^{or 1's}

Arithmetic shift left (ASHL) = 10111000

To check overflow:

Arithmetic Shift Right (ASHR) = 11101110

A B A/B

$$V = D_7 \oplus D_6$$

$$= 1 \oplus 1$$

$$= 0; \text{ No overflow}$$

So, there is no overflow.

5) A basic computer has an instruction format as follows:

- Determine the memory size and its data bits for the computer.
- How many registers does the computer has?

I	Opcode = 3 bits	Registers = 4 bit	Address = 16 bit
---	-----------------	-------------------	------------------

Solⁿ

Given,

I	Opcode	Register	Address
1	3	4	16

$$\text{Memory size} = 16 \text{ bits}$$

$$= 2^6 \times 2^{10}$$

$$= 2^6 \times 1024$$

$$= 2^6 \text{ K Word}$$

$$= 64 \text{ KW}$$

Data bits = $16 + 4 + 3 + 1 = 24$ bits
Total no. of register = $2^4 = 16$ bits

⑥ How do you detect overflow while adding 2's complement data?

→ Overflow occurs when:-

- ① Two (-ve) no. are added and an answer comes (+ve)
- ② Two (+ve) no. are added and an answer comes as

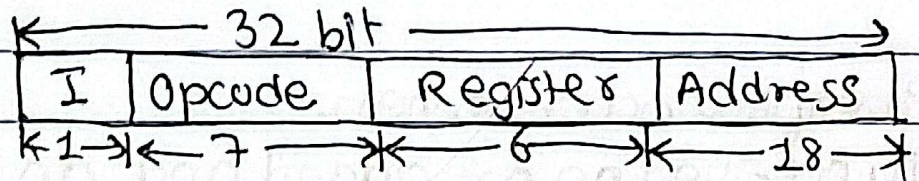
So, overflow can be detected by checking MSB of two operand and answer. But instead of using 3-bit comparator, overflow can also be detected using 2-bit comparators just by checking Carry-in (C-in) and Carryout (C-out) from MSB.

⑦ A computer uses a memory unit with 256KW of 32 bit each. A binary instruction code is stored in one memory. The instruction has 4 fields, an indirect bit, an operational code, a register code field to specify one of 64 registers and an address part?

① How many bits are there in the operational code, the register code and the address field?

- 7
- (b) Draw the instruction word format and indicate the no. of bits in each part.
- (c) How many bits are ~~bl~~ there in the data and address input of the memory.

Solⁿ



$$\begin{aligned}
 \text{Memory size} &= 256 \text{ KW} \\
 &= 256 \times 1024 \\
 &= 2^8 \times 2^{10} \\
 &= 2^{18}
 \end{aligned}$$

$$\begin{aligned}
 \text{No of address bit} &= \log_2 (\text{M/M size}) \\
 &= \log_2 (2^{18}) \\
 &= 18 \text{ bits.}
 \end{aligned}$$

$$\text{Total no. of Registers} = 64$$

$$\begin{aligned}
 \text{Now, no of bits access 64 register} &= \log_2 (64) \\
 &= \log_2 (2^6) \\
 &= 6 \text{ bits}
 \end{aligned}$$

$$\text{Opcode} = 32 - 18 - 6 - 1 = 7 \text{ bits.}$$

(c) data = 32 bit
Address = 18 bit

8. Given a register $R_1 = 11100011$, determine the value of R_1 after execution of following microoperation

(a) $R_1 \leftarrow CIL(R_1)$

(b) $R_1 \leftarrow ASH(R_1)$

(c) $R_1 \leftarrow SHL(R_1)$

Soln
(a) $R_1 \leftarrow CIL(R_1); R_1 = 11100011$
 $CIL = 11000111$

(b) $R_1 = 11100011$
 $SHR = 01110001$

(c) $R_1 = 11100011 \leftarrow 0$
 $SHL = 11000110$

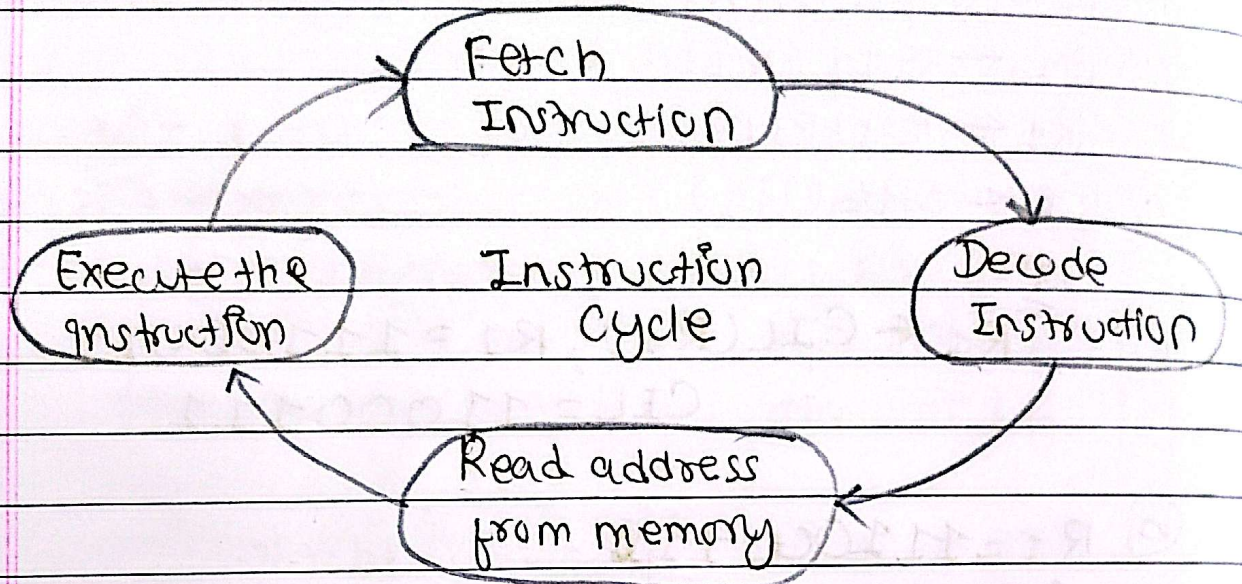
9. Discuss instruction cycle of a basic computer

→ A program residing in the memory unit of a computer consists of a sequence of instructions. These instructions are executed by the processor by going through a cycle for each instruction. Each instruction consists of the following phases:-

(a) Fetch instruction from memory

(b) Decode the instruction

- ③ Read the effective address from memory.
- ④ Execute the instruction.



① Fetch instruction:

Each instruction executed in CPU uses the fetch instruction. During this phase, CPU sends the PC to MAR and then read instruction to a control bus. After sending read instruction on the data bus, the memory returns the instruction that was stored at that exact address in the memory. The CPU then copies the data from the data bus into MBR which is then copied to registers. The pointer is incremented to the next memory location, allowing the next instruction to be fetched from memory.

Date _____
Page _____

(b) Decoding instruction:-

The second phase is instruction decoding. During this step, the CPU determines which instruction should be fetched from the instruction and what ^{short} action ^{be} taken on the instruction. The instruction's opcode is also retrieved from memory and it decodes that related operation that must be performed for the instruction.

(c) Read of an effective address.

The third phase is the reading of an effective address. The operation decision is made during this phase. Any memory type operation are used. Direct memory instruction and indirect memory instruction are the two types of memory instruction available.

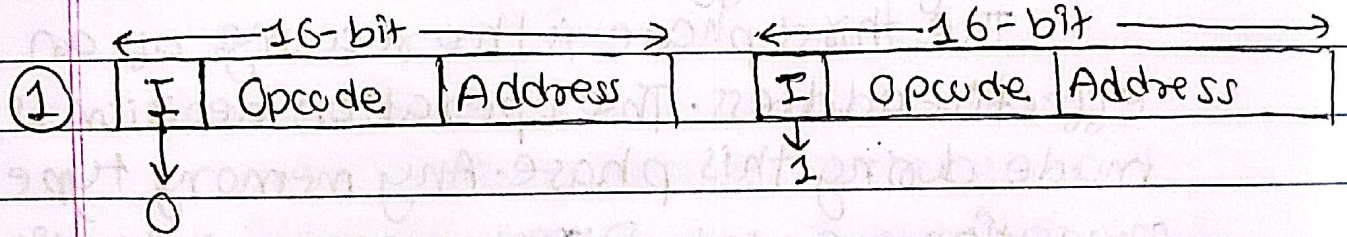
(d) Execution of the instruction:-

The last step is to carry the instructions. The instruction is finally carried out at this stage. The instruction is carried out and the result is saved in the register. The CPU gets prepared for the execution of the next instruction after the completion of each instruction. The execution time of each

instruction is calculated, and this information is used to determine the processor processing speed.

10 Q) What is the difference between a direct and an indirect address instruction? How many ~~the~~ references to memory are needed for each type of instruction to bring an operand into a processor register.

Direct Address Instruction Indirect Address Instruction



② The address bit itself is the effective address

The content of the address field is the Effective address.

③ It required two M/M reference i.e.

- Read Instruction
- Read Operand.

It required three M/M reference i.e.

- Read Instruction
- Compute EA
- Read Operand.

④ It is fast

It is slow.

⑤ It require small M/M size.

It required large M/M size.

→ A direct address instruction needs two references to memory i.e. (a) Read instruction
(b) Read operand.

→ An indirect address instruction needs three references to memory i.e. (a) Read instruction
(b) Read Effective address.
(c) Read operand.

11 Q) What is stored program organization?

→ The simplest way to organize a computer is to have one processor register and an instruction code format with two parts.

→ The first part specifies the operation to be performed and the second specifies an address.

→ The memory address tells the control where to find an operand in memory.

→ This operand is read from memory and used as the data to be ~~sp~~ operated on together with the data stored in the processor register.

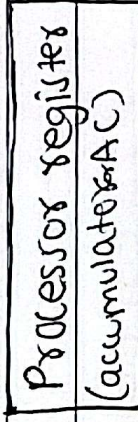
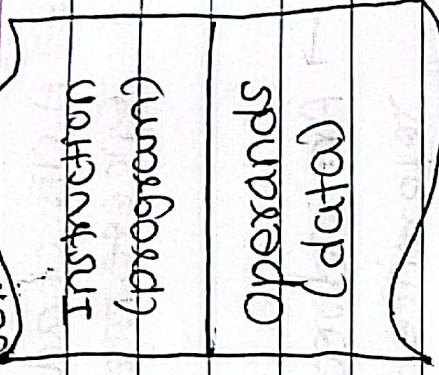
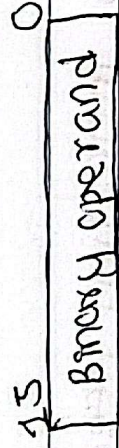
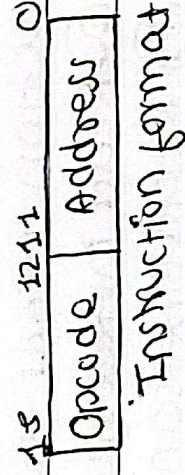


Fig:- Stored program organization.

→ Figure above depicts this type of organization. Instruction are stored in one section of memory and data in another. For a memory unit with 4096 words. We need 12 bits to specify an address since $2^{12} = 4096$. If we store each instruction code in one 16-bit memory word, we have available 64K bits for the operation code to specify one out of 18 possible operation and 12 bits to specify the address of an operand.

→ The control ~~reads~~ reads a 16 bit instruction from the program portion of memory. It uses the 12 bit address of the instruction to read a 16-bit operand from

the data ^{position} of memory.

- It then executes the operation specified by the operation code.
- Computers that have a single processor register usually assign to it the name accumulator and label it AC. The operation is performed with the memory operand and the content of AC.
- If an operation in an instruction code doesn't need an operand from memory, the rest of the bits in the instruction can be used for other purposes.
for eg: Operation such as clear AC, complement AC and increment AC operate on data stored in the AC register.
- If an operation in an instruction code doesn't need an operand from memory.
- They do not need an operand from memory. For these types of operations, the second part of the instruction code (bits 0 through 11) is not needed for specifying a memory address and can be used to specify other operations for the computer.

Q) Evaluate the arithmetic statement:
 $X = A + C(D - F)$ using-

G

(a) 1-address (Accumulator)

Mnemonic	Operand	Microoperation
LOAD	D	$AC \leftarrow M[D]$
SUB	F	$AC \leftarrow AC - M[F]$
MUL	C	$AC \leftarrow AC * M[C]$
DIV	G	$AC \leftarrow AC \div M[G]$
ADD	A	$AC \leftarrow AC + M[A]$
STORE	X	$M[X] \leftarrow AC$

(b) 2-address

Mnemonic	Operand	Microoperation
MOV	R_1, D	$R_1 \leftarrow M[D]$
SUB	R_1, F	$R_1 \leftarrow R_1 - M[F]$
MUL	R_1, C	$R_1 \leftarrow R_1 * M[C]$
DIV	R_1, G	$R_1 \leftarrow R_1 \div M[G]$
MOV	R_2, A	$R_2 \leftarrow M[A]$
ADD	R_1, R_2	$R_1 \leftarrow R_1 + R_2$
MOV	X, R_1	$X \leftarrow R_1$

③. 3-address,

SUB R_1, D, F $R_1 \leftarrow M[D] - M[F]$
 MUL R_2, R_1, C $R_2 \leftarrow R_1 \times M[C]$
 DIV R_3, R_2, G $R_3 \leftarrow R_2 / M[G]$
 ADD X, A, R_3 $M[X] \leftarrow M[A] + R_3$

④ Zero Address.

Mnemonic	Operand	Microoperation
PUSH	A	$TOS \leftarrow A$
PUSH	G	$TOS \leftarrow G$
PUSH	C	$TOS \leftarrow C$
PUSH	D	$TOS \leftarrow D$
PUSH	F	$TOS \leftarrow F$
SUB		$TOS \leftarrow D - F$
MUL		$TOS \leftarrow C \times (D - F)$
DIV		$TOS \leftarrow C \times (D - F)$
ADD		$TOS \leftarrow A + C \times (D - F)$
POP	X	$X \leftarrow TOS$

1) (Q) Explain General Register Organization of CPU.

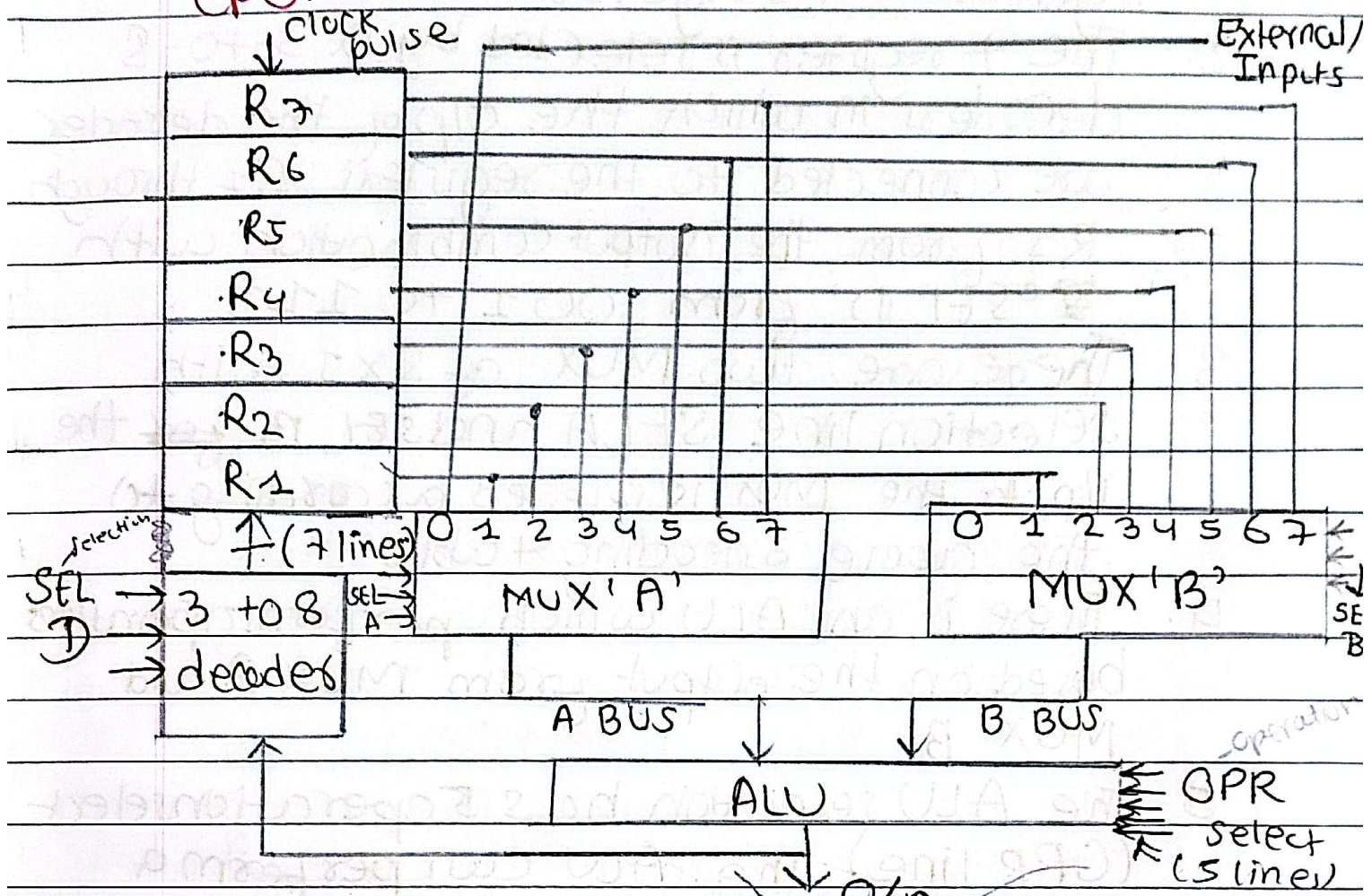


Fig:- Register organization of CPU.

Binary Value	SEL A External I/P	SEL B External I/P	SELD
0 0 0	External I/P	External I/P	—
0 0 1	R ₁	R ₁	R ₁
0 1 0	R ₂	R ₂	R ₂
0 1 1	R ₃	R ₃	R ₃
1 0 0	R ₄	R ₄	R ₄
1 0 1	R ₅	R ₅	R ₅
1 1 0	R ₆	R ₆	R ₆
1 1 1	R ₇	R ₇	R ₇

Point

1. In the register organization, there are 7 general CPU registers.
2. The 7 registers is selected by a 3-to-8 decoders in which the output of the decoder are connected to the registers R7 through R0 from the output combination with 'SEL' from 001 to 111.
3. There are two MUX of 8×1 with selection line SEL A and SEL B for the input to the MUX is selected according to the above encoding table.
4. There is an ALU which perform operations based on the output from MUX A and MUX B.
5. The ALU selection has 5 operations select (OPR line). The ALU can perform a maximum of $2^5 = 32$ ALU operation. Some of which are listed below.

OPR Select	Operation	Symbol
00000	Transfer (A)	; TSP A
00001	Increment (A)	; INC A
00010	Add (A+B)	; ADD
11000	Shift left (A)	; SHL A

15 (8) An instruction is stored at location 300 with Address field at location 301. The address field has the value 400. A processor register R_1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is.

- (a) Direct (b) Immediate (c) Relative
(d) Register indirect (e) Index with R_1 as the index register

300	Opcode	Mode
301	Address = 400	
302		
	⋮	
	⋮	

$$R_1 = 200$$

- (a) Direct, $EA = 400$
 (b) Immediate; $EA = 301$
 ; ~~(c) Register Direct; $EA = \text{None}$~~
 (c) Register Relative: $EA = \text{Address field} + PC$

$$= 400 + 302$$

$$= 702$$

 (d) Register Indirect: $EA = R_1 = 200$
 (e) Indexed: $EA = \text{Address field} + X_R$

$$= 400 + R_1$$

$$= 400 + 200$$

$$= 600$$

16. Consider a direct mapped cache of size 16 KB with block size 256 Bytes. The size of main memory is 128 KB. Find.

- Number of bits in tag
- Tag directory size.

Soln

= Given,

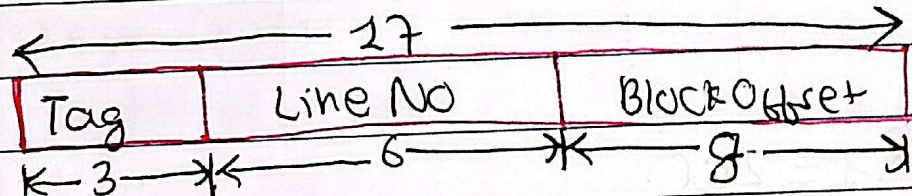
Size of Main memory = 128 k byte.

$$= 128 \times 1024$$

Block size = 256 Byte.

Cache memory = 16 kByte.

In cache :-



$$\begin{aligned} \text{No. of address bit} &= \log_2 (\text{M/M size}) \\ &= \log_2 (128 \times 1024) \\ &= \log_2 (2^{17}) \\ &= 17 \text{ bits} \end{aligned}$$

*size of main
No. of address bit in
main mem = No. of
address bit in
cache??*

$$\begin{aligned} \text{No. of block offset} &= \log_2 (\text{Block size}) \\ &= \log_2 (256) \\ &= \log_2 (2^8) \\ &= 8 \text{ bits} \end{aligned}$$

$$\frac{192}{8} = 24$$

$$\text{size of tag directory} = 2^7 = 128 \text{ i.e. } 2^{\text{no of tag bit}}$$

Date _____
Page _____

$$\begin{aligned} \text{No of bits in line No} &= \frac{\text{cache size}}{\text{Block size}} \\ &= \frac{16 \times 1024}{256} \\ &= 2^6 \\ &= 6 \text{ bits} \end{aligned}$$

$$\therefore \text{No of bits in tag} = 17 - 8 - 6 = 3 \text{ bits}$$

$$\begin{aligned} \text{(b) Tag directory size} &= \text{No of tags} \times \text{tag size} \\ &= \text{No of line in cache} \times \text{No of bits in tag} \\ &= 2^6 \times 3 \\ &= 192 \text{ bits} = 24 \text{ bytes} \end{aligned}$$

$$\therefore \text{Size of tag directory} = 24 \text{ bytes}$$

V-IMP

1st (Q) Consider a 4-stage pipeline processor. The numbers of cycles needed by the four instructions I_1, I_2, I_3 & I_4 in stage S_1, S_2, S_3 and ~~instruction~~ S_4 is shown below:

Instruction/Segment	S_1	S_2	S_3	S_4
I_1	2	1	1	1
I_2	1	3	2	2
I_3	2	1	1	3
I_4	1	2	2	2

What is the number of cycle needed to execute the following loop?

for $I = 1$ to 2

{

$I_1;$

$I_2;$

$I_3;$

$I_4;$

}

⇒

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S_1	I_1	I_2	I_2	I_3	I_3	I_4	I_1	I_2	I_2	I_3	I_3	I_4									
S_2			I_1	I_2	I_2	I_3	I_3	I_4	I_1	I_2	I_2	I_3	I_4	I_1	I_2	I_2	I_3	I_4			
S_3				I_1			I_2	I_2	I_3	I_4	I_1	I_2	I_2	I_3	I_4	I_1	I_2	I_2	I_3	I_4	
S_4					I_1				I_2	I_2	I_3	I_3	I_4	I_1	I_2	I_2	I_3	I_4	I_1	I_2	I_2

Pipeline (steal)

Q. A non-pipelined system takes 50ns to process a task. The same task can be processed in a six segment pipeline with a clock cycle of 20ns. Determine the speedup of the pipeline for 100 tasks. What is the maximum speedup that can be achieved?

soln

No of segment (k) = 6

No of task (n) = 100

$t_n = 50\text{ns}$

$t_p = 20\text{ns}$

$$\therefore \text{speedup ratio } (S) = \frac{n + n}{(k + n - 2) + p}$$

$$= \frac{100 \times 500}{(8 + 100 - 1) \times 20}$$

$$= 4.7619.$$

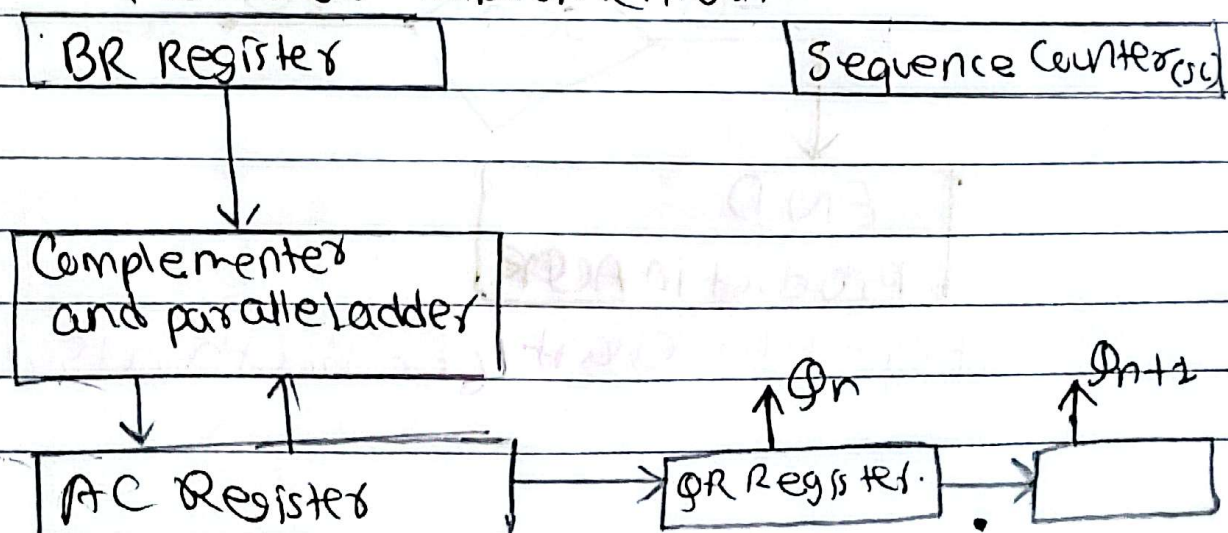
$$\text{Max. speedup } (S) = \frac{tn}{tp} = \frac{500}{100} = 5.$$

22 23 19. (Q) Explain Booth Multiplication Algorithm with hardware implementation diagram, multiply $(-4) \times (-5)$ using Booth multiplication algorithm.

I₄ I₄

→ Booth multiplication algorithm is a multiplication algorithm that allows us to multiply the two signed binary integers in 2's complement. It is also used to speed up the performance of the multiplication process.

Hardware implementation



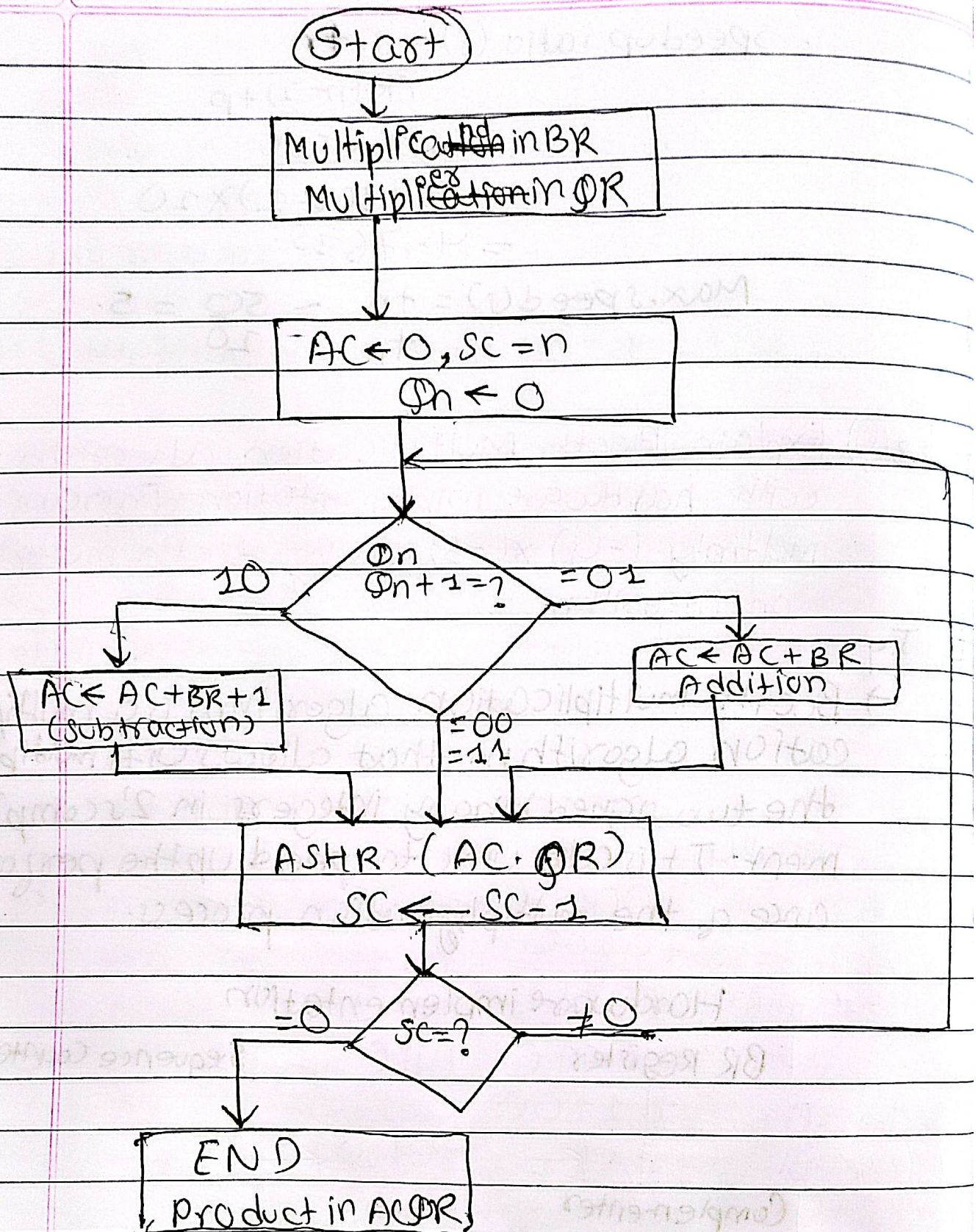


Fig:- Flowchart for booth algorithm

where,

Φ_n = LSB of ΦR Register

Φ_{n+1} = Shifted version of Φ_n

SC = Sequence counter

n = No of bits in ΦR register including sign bit.

ASHR $\rightarrow$ Arithmetic shift Right

-4 BR
x -3 ΦR .

SC = n = 4.

$10^3 0$

2's complement = 011

+ 1

100

011

2's complement = 100

+ 1

101

-4 $\Rightarrow$ BR $\Rightarrow$ 1100 $\Rightarrow$ BR + 1 = 0100

Initially

AC

$\Phi R \leftarrow \Phi_n$

Φ_{n+1}

SC

0000

1101

0

4

$\Phi_n \Phi_{n+1} = 10 + 0100$

AC $\leftarrow$ AC + BR + 1

0100 $\rightarrow$ 1101

ASHR(AC ΦR)

0010

0110

1

3

SC $\leftarrow$ SC - 1

$\Phi_n \Phi_{n+1} = 01 + 1100$

+ 1100

AC $\leftarrow$ AC + BR

1110 $\rightarrow$ 10110

ASHR(AC ΦR)

1111

0011

0

2

SC $\leftarrow$ SC - 1

$$\Phi_n \Phi_{n+1} = 10 \quad + 0100$$

$$AC \leftarrow AC + BR_1$$

$$\text{neglect } 00011$$

$$0011 \rightarrow$$

$$ASHR(AC, R)$$

$$SC \leftarrow SC - 1$$

$$0001$$

$$1001$$

$$1$$

$$1$$

$$\Phi_n \Phi_{n+1} = 11$$

$$0001$$

$$1001$$

$$ASHR(AC, R)$$

$$SC \leftarrow SC - 1$$

$$0000$$

$$1100$$

$$1$$

$$0$$

$$\therefore (00001100)_2 = (12)_{10}$$

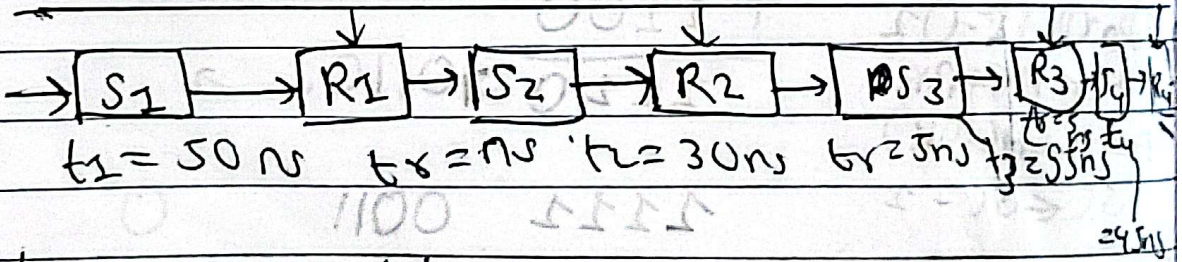
(20) The time delay for the four segments in the pipeline are as follows:

$$t_1 = 50 \text{ ns}, t_2 = 30 \text{ ns}, t_3 = 95 \text{ ns} \text{ \& } t_4 = 45 \text{ ns}$$

The interface registers delay ~~is~~ time $t_x = 5 \text{ ns}$

- (a) How long would it take to add -100 pair of number in the pipeline?
- (b) Determine the speed up factor for the system. Also determine the max speed up factor.

$\Rightarrow$ Clock pulse



$$t_p = \max \text{ of } (t_1, t_2, t_3, t_4) + t_x$$

$$= t_3 + t_8 = 95ns + 5ns$$

$$= 100ns$$

$$t_n = t_1 + t_2 + t_3 + t_4 + t_8$$

$$= 50 + 30 + 95 + 45 + 5$$

$$= 225$$

$$\Rightarrow (k + n - 1) \times t_p = (4 + 100 - 1) \times 100$$

$$= 10300ns$$

$$k = 4$$

$$n = 100$$

$$t_n = 225$$

$$t_p = 100$$

$$S = \frac{n \cdot t_n}{(k + n - 1) t_p} = \frac{100 \times 225}{(4 + 100 - 1) \times 100}$$

$$= 2.1875$$

$$\text{Max. speedup factor (S)} = \frac{t_n}{t_p} = \frac{225}{100}$$

$$= 2.25$$

(21) How effective Address is computed in different addressing mode?

$$t_8 = 5ns$$

→ The way of specifying the operand in the instruction is called Addressing mode.

Different addressing mode are:-

(i) Immediate Addressing mode:

→ In this addressing mode, the operand is a part of the instruction itself.

Effective address: Not applicable because there is no address to compute; the operand is directly provided.

Eg:- 'MOV AX, 5' (where 5 is the immediate operand)

(ii) Direct addressing mode:-

→ In this addressing mode the address of the operand is given explicitly in the instruction.

Effective address calculation: $EA = \text{address specified in the instruction}$.

Eg:- 'MOV AX, [1234]' (where 1234 is the address)

(iii) Indirect Addressing Mode:-

→ In this The instruction specifies a register or memory location that contains the address of the operand.

Effective address: $EA = \text{Address or register specified in the instruction}$.

Auto decrement: In this addressing mode, the EA is decremented by 1 before the execution of instruction.

~ (i) What is meant by pipelining? List the different conflicts that will arise in pipelining.

→ Pipeline is a technique of decomposing a sequential task with no. of sub-task that are executed in different segment (stages). The purpose of pipeline is to increase the processing speed of the system.

Different conflicts that will arise in pipelining are:-

(i) Resource conflict: The resource conflict ~~has~~ arise by the excess to the memory by two or more segment at the same time. Most of these conflict can be resolved by using separate data and address bus.

(ii) Data dependency: The data dependency

can also arise when an instruction depends on the result of previous instruction but the result is not yet available.

(iii) Branching difficulty: It arises due to a branch of the instruction which causes random changes in the value of program counter (PC).

(iv) In certain scientific computation it is necessary to perform the arithmetic operation $(A_i + B_i) (C_i + D_i)$ with stream of numbers. Specify the pipeline configuration to carry out this task. List the contents of all registers in the pipeline for $i = 1$ to 6.

→ Segment 1: $R_1 \leftarrow A_i, R_2 \leftarrow B_i, R_3 \leftarrow C_i, R_4 \leftarrow D_i$
Segment 2: $R_5 \leftarrow A_i + B_i, R_6 \leftarrow C_i + D_i$
Segment 3: $R_7 \leftarrow R_5 + R_6$

The pipeline configuration of the segments can be represented as:

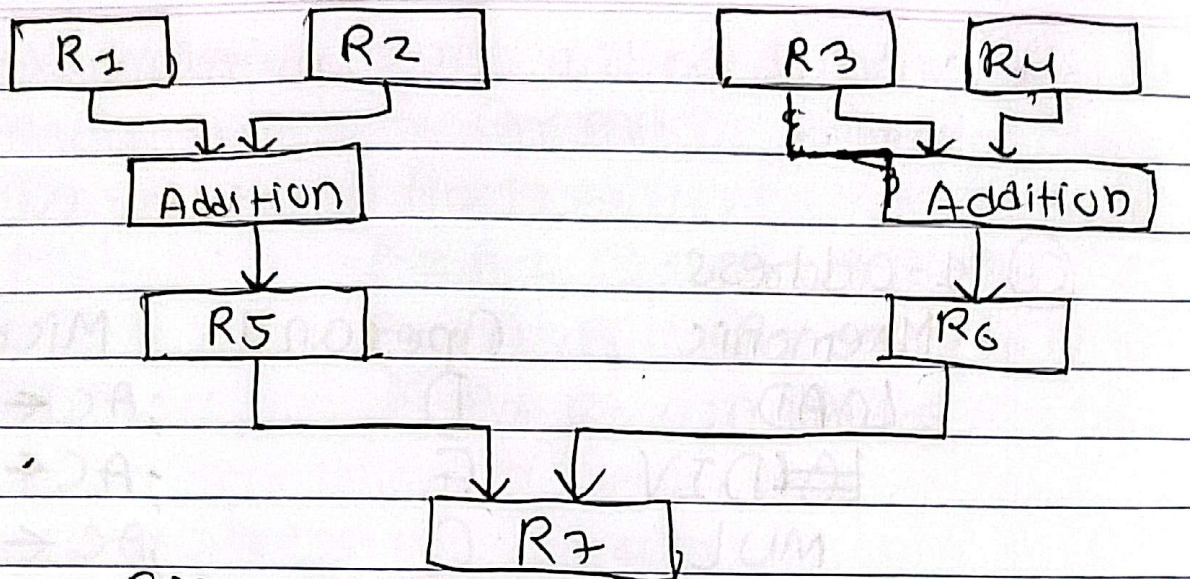


Fig:- Pipeline Configuration

The different segment with respect to clock pulse can be shown as:-

	Segment 1				Segment 2		Segment 3
	R ₁	R ₂	R ₃	R ₄	R ₅	R ₆	R ₇
1	A ₁	B ₁	C ₁	D ₁	-	-	-
2	A ₂	B ₂	C ₂	D ₂	(A ₁ +B ₁)	(C ₁ +D ₁)	-
3	A ₃	B ₃	C ₃	D ₃	(A ₂ +B ₂)	(C ₂ +D ₂)	(A ₁ +B ₁)(C ₁ +D ₁)
4	A ₄	B ₄	C ₄	D ₄	(A ₃ +B ₃)	(C ₃ +D ₃)	(A ₂ +B ₂)(C ₂ +D ₂)
5	A ₅	B ₅	C ₅	D ₅	(A ₄ +B ₄)	(C ₄ +D ₄)	(A ₃ +B ₃)(C ₃ +D ₃)
6	-	-	-	-	(A ₅ +B ₅)	(C ₅ +D ₅)	(A ₄ +B ₄)(C ₄ +D ₄)
7	-	-	-	-	-	-	(A ₅ +B ₅)(C ₅ +D ₅)

In general, n = no of tasks has to be completed
 k = no of segment.

$$\text{Total clock pulse} = n + k - 1$$

$$= 5 + 3 - 1 = 7$$

21 (36) Evaluate arithmetic expression; $X = \frac{A}{B} + C * D$
Using: E

(a) 1-address:

Mnemonic	Operand	Microoperation
LOAD	D	$; AC \leftarrow M[D]$
LOAD DIV	E	$; AC \leftarrow AC / M[E]$
MUL	C	$; AC \leftarrow AC * C$
STORE	P	$; M[P] \leftarrow AC$
LOAD	A	$; AC \leftarrow M[A]$
DIV	B	$; AC \leftarrow AC / M[B]$
ADD	P	$; AC \leftarrow AC + M[P]$
STORE	X	$; M[X] \leftarrow AC$

(b) 2-address:

Mnemonic	Operand	Microoperation
MOV	R1, D	$; R1 \leftarrow M[D]$
DIV	R1, E	$; R1 \leftarrow M[D] / M[E]$
MUL	R1, C	$; R1 \leftarrow R1 * M[C]$
MOV	R2, A	$; R2 \leftarrow M[A]$
LOAD DIV	R2, B	$; R2 \leftarrow M[A] / M[B]$
ADD	R1, R2	$; R1 \leftarrow R1 + R2$
MOV	X, R1	$; M[X] \leftarrow R1$

3-address:

Mnemonic	Operand	Microoperation
DIV	R_1, D, E	$R_1 \leftarrow M[D] / M[E]$
MUL	R_2, R_1, C	$R_1 \leftarrow R_2 * M[C]$
DIV	R_3, A, B	$R_3 \leftarrow M[A] / M[B]$
ADD	X, R_3, R_2	$M[X] \leftarrow R_3 + R_2$

Zero address:

Mnemonic	Operand	Microoperation
PUSH	C	$TOS \leftarrow C$
POSH	D	$TOS \leftarrow D$
POSH	E	$TOS \leftarrow E$
DIV		$TOS \leftarrow D / E$
MUL		$TOS \leftarrow D * E$
PUSH	A	$TOS \leftarrow A$
PUSH	B	$TOS \leftarrow B$
PUSH DIV		$TOS \leftarrow A / B$
ADD		$TOS \leftarrow \frac{A}{B} + \frac{C * D}{E}$
POP	X	$M[X] \leftarrow TOS$

25 (P) Define micro operation. Explain the different types of microoperation. The following ~~the~~ transfer statement specify. Explain the memory operation in each case.

~~Ans~~
~~(a)~~ $R_2 \leftarrow M[AR]$
 (b) $M[AR] \leftarrow R_3$

// with RTL code programmer includes.

→ Microoperation is a functional operation in a computer's central processing unit (CPU) that typically involves the transfer manipulation or storage of data within the CPU or between the CPU and other components of the computer system.

Types of microoperation :-

- ① Data Transfer Microoperation
- ② Arithmetic microoperation
- ③ Logical microoperation
- ④ Shifting microoperation.

① Data Transfer Microoperation:

→ Here, data is transfer from source to the destination without modifying this content. A transfer may be between the processor registers themselves or between registers and memory or between registers and I/O device.

Eg:- $R_1 \leftarrow R_2$; it transfer value of R_2 into R_1

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

$\bar{P}: R_1 \leftarrow R_2; IF (P=0)$ then ($R_1 \leftarrow R_2$)

P 4:13 I can do all things through Christ
who strengthens me

TECHNICALSNAPEBOOKA

Date _____
Page _____

Q1a Differentiate between computer organization and architecture with example

Computer Organization Computer Architecture

P The physical aspects of a computer, including circuits and hardware components. The conceptual design of a computer system

H It describes how computers do the work. It describes what the computer does the work

D Deals with the low-level design issues (logic and circuit design) and high level design issues (system and computer)

S It explains Structural Relationship- Functional Behaviour

S Computer Organization is Computer Architecture the second into approach is the first into approach while designing a computer

I It involves circuit sets, It involves instruction sets,

design, signals, ALU,
CPU and memory.

ISA - Instruction set Architecture
addressing modes and
datatype.

E Example studies: Studies
on microprocessors
chips, physical memory
layout, and interface
design

Studies on pipelini-
ng techniques, and
computational models

PH DSS IE

PH DSS IE

2018 Fall.

Q 1a Explain the different stages of computer
evolution with reference of generation
of computer.

History of Computer / Miles stones in
Computer ~~Org~~ Organization:

Mechanical Era (1623-1945):

→ The idea of using machines to solve
mathematical problems can be traced
back to 17th century where mathemati-
cian designed and implemented calculators
that were capable of addition, subtrac-
tion, multiplication and division.

- Date _____
Page _____
- The first multipurpose programmable computing device was Charles Babbage's Differential and Analytical engine.
 - First commercial use of mechanical computer was in US Census Bureau by Herman Hollerith.

Drawback:

- Speed of operation limited by inertia of moving parts.
- Cumbersome, unreliable and expensive

Electrical Era:

First Generation (1945-1956)

- Vacuum tubes for circuitry
- Magnetic drum for memory.
- Enormous high power
- ENIAC (Enhanced Numeric Integrator and Computer)

Second generation: (1956-1963)

- Transistor
- Smaller, faster, cheap, more energy efficient, more reliable.
- More complex ALU, control units (CU)

Third generation (1964-1971)

- Integrated circuit,
SSI (Small scale Integration) - 1 million transistors

2nd LSI (Large scale Integration) - 10 million transistors

3rd VLSI (Very large scale Integration) - 10 billion transistors.

- Third generation computer through keyboard and monitors and interface with an Operating system, which allowed the device to run many different applications at one time with central program that monitored the memory.

Fourth Generation (1971-present)

- Used microprocessor, fourth generation of computer as thousand of integrated circuits were built on a single silicon chip.
- IBM introduced first computer, 1984 Apple Macintosh.

Fifth Generation (1974-present).

- Based on Artificial Intelligence, still development phases.
- Voice ~~recognized~~ recognition, self decision,

nanotechnology, self organization

1 ^{Small} SSI (Scale Integration): 1 million transistors which refer to electronic circuit where the number of transistors is low enough that entire circuit can be built on a single semiconductor chip.

(class) ^{Medium Scale Integration} LSI (Large Scale Integration): which refer to circuits with a large number of transistors, allowing for complex functionality 10 million transistors.

VLSI (Very Large Scale Integration), which refer to circuits with a very large number of transistors, enabling even more complex and powerful electronic device. 10 billion transistors.

Often used to describe the level of integration and complexity of electronic circuit.

PBRNYKKNMHSKNMILTNOCKA

Q10 Explain the functional view of the computer system with reference to each component.

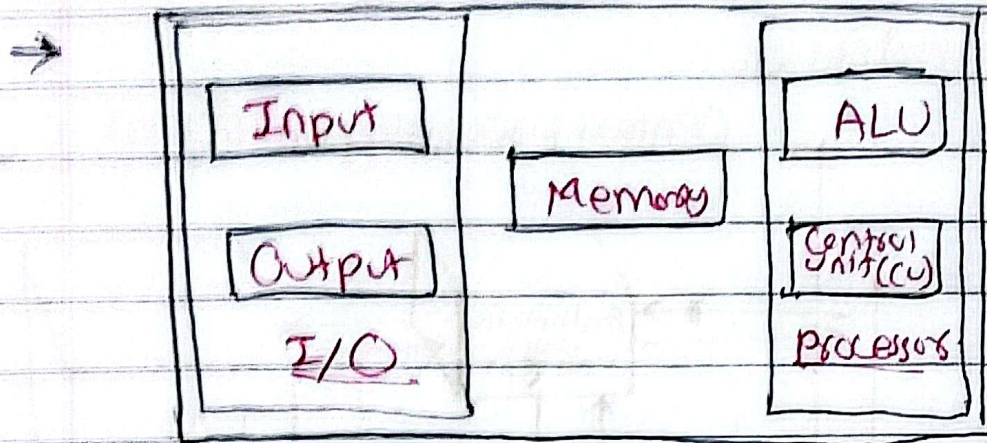


Fig:- Functional unit of computer system.

Input: gives i/p to the system
Keyboard, joystick

Output: Display on the screen monitor,
mobile screen, speakers.

Memory: It performs operation like read,
write, RAM, ROM, hard disk, etc.

ALU: → (Arithmetic Logic Unit): Operates
on binary data.

Control Unit: Interprets instruction in
memory and causes them to be exe-
cuted.

16) What is register in computer system?

Explain different types of registers used in computer system

- Register is small amount of memory to store data.
- Used to by the CPU during execution of program/Instruction
- Also known as processor register

What can be stored?

→ 10 Data 11 Address 12 Instructions

Types:

- Input Register: 8 bits - Holds the user I/P
- Output Register: 8 bits - Holds the output
- Address Register: 12 bits - Holds address of memory location
- Program Counter: 12 bits - Holds address of next instruction which is to be executed

D → Data Register: 16 bits - Holds the data of memory location i.e. it holds an operand → variable

A → Accumulator: 16 bits - General purpose Register (can be used for any purpose)

T → Temporary Register: 16 bits - Holds temporary data. (Intermediate result in between the input and output will be stored)

Size of register - 8 bit - 1 byte

16 bit - 2 byte

Register is sequence of bits or sequence of 1s & 0s

VAN NEUMAN MACHINE / IAS computer

It was designed by the Princeton Institute for advanced studies (IAS)

It consists of:

Central Processing Unit (CPU)

Resistant memory

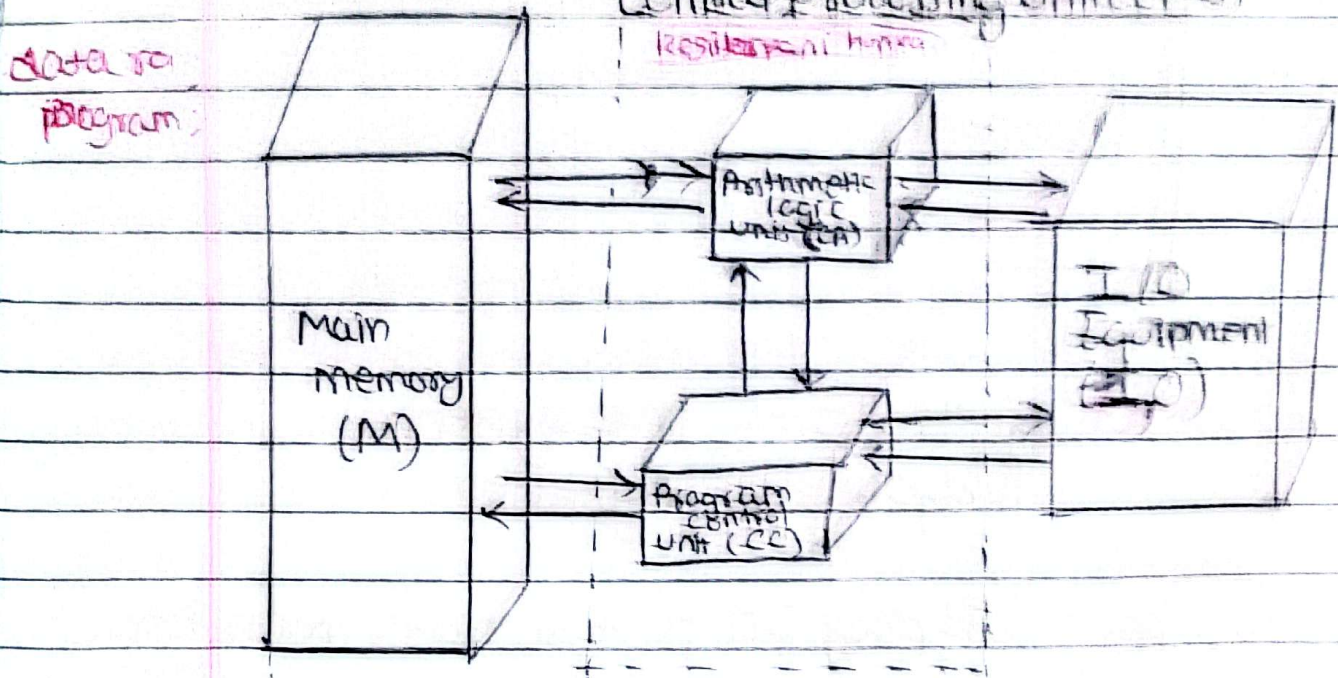


Fig:- Structure of IAS computer

- Main memory: which stores ~~the~~ both data and instruction.
- Arithmetic and logic unit: Capable of operating on binary data. And, $+$, $-$, $*$, $/$.
- A control unit which interprets the instruction in memory and causes them to be executed. It provides timing signal & control signal.
- Input and output equipment operated by control unit.
- Register: It has high speed, to match with speed of ALU rather than Main memory because it is slower & complex.

DAPATIO

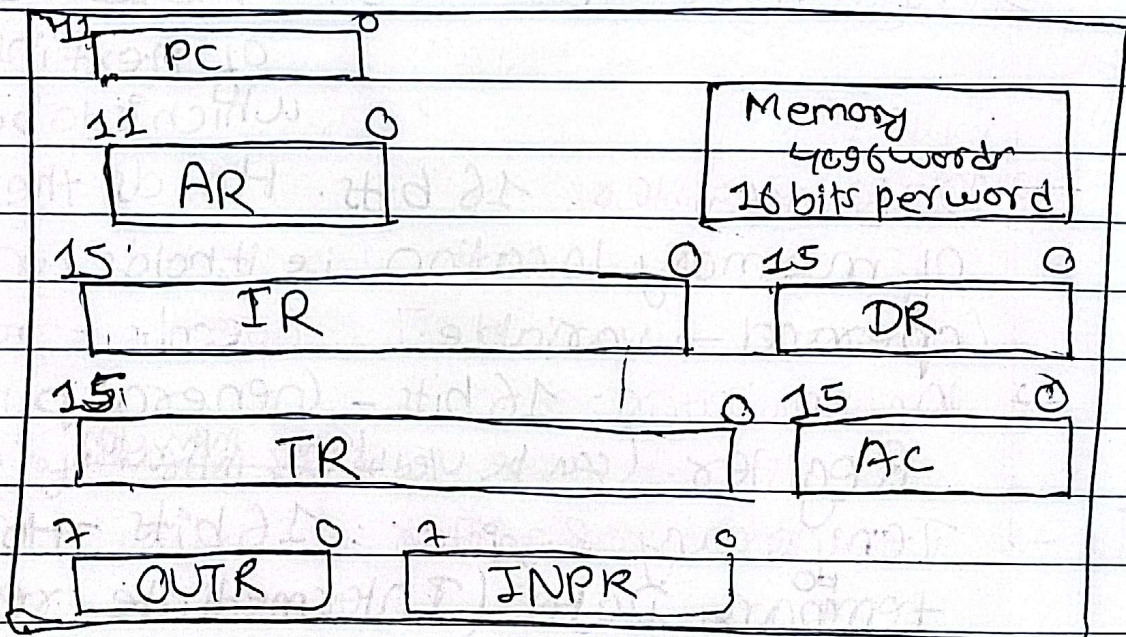
Date: Same
Page: 1
DIPAA TOPI

→ Instruction Register: 16 bits: Holds instructions read from memory

Memory
4096x16
No of words

2¹²
12 bits.

Register	Symbol	Number of bits	Function.
✓ Data register	DR	16	Holds memory operand
✓ Accumulator	AC	16	Processor register
✓ Program Counter	PC	12	Holds address of instruction
✓ Address register	AR	12	Holds address for the memory
✓ Instruction register	IR	16	Holds instruction code
✓ Temporary register	TR	16	Holds temporary data
✓ Input register	INPR	8	Carries input character
✓ Output register	OUTR	8	Carries output character



S₁ S₀
0 0
0 1
1 0
1 1

A₀ (Zero Input)

1(b) Define RTL.

1(b) Define RTL. Explain about BUS and Memory transfer in RTL.

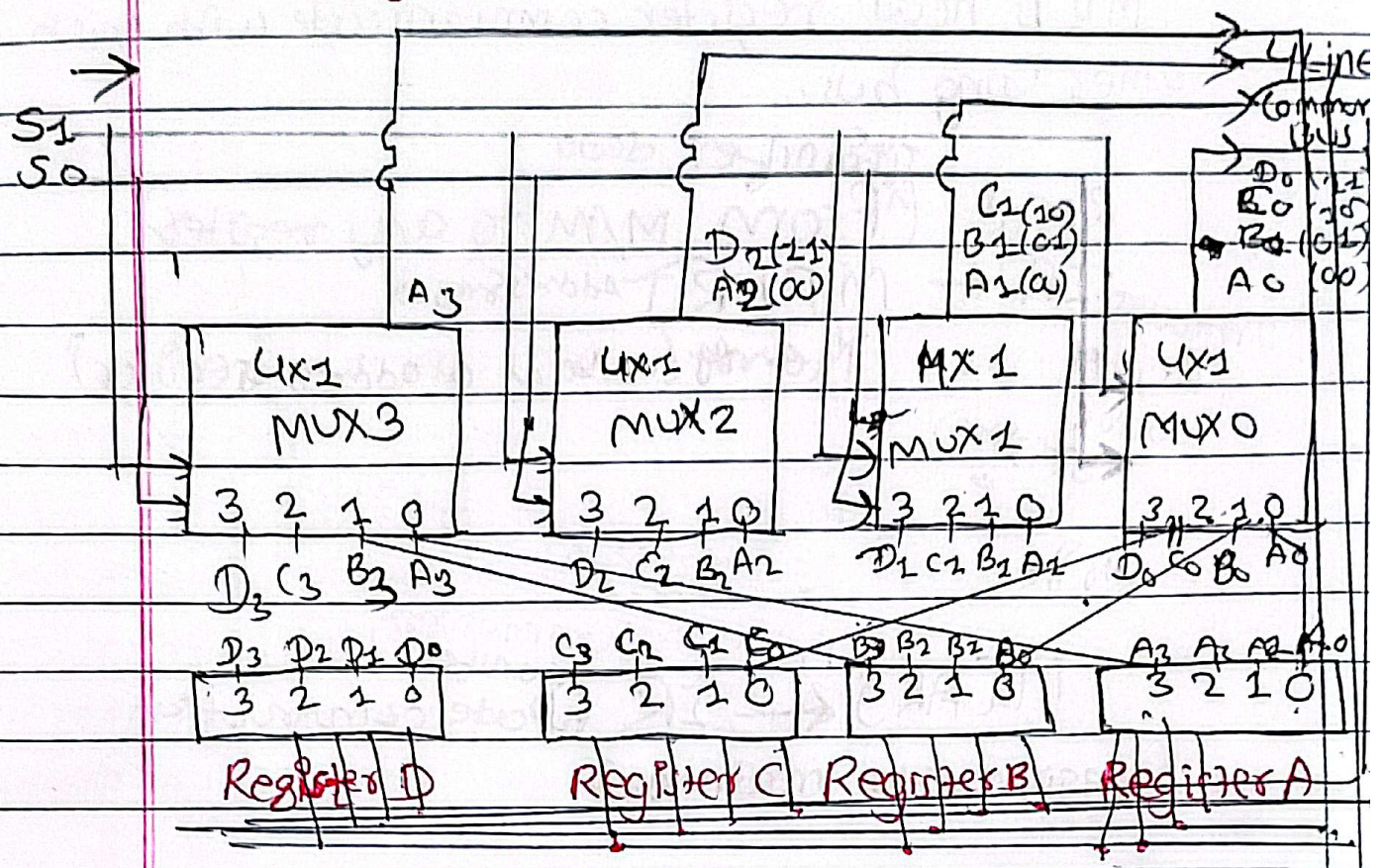


Fig:- Bus Architecture.

R₁ ← C

For this
B₀ ← C
R₁ ← B₀

By Addressing (10)

No of MUX = No of registers.

Size of register = 4 bit so 4 x 1 MUX.

No of input in MUX = No of register

R₁ ← C
For this
B₀ ← C
R₁ ← B₀

Now we have to pick data from A and send it to B. so,

→ Set select line '00' It will load data of A₀ in bus. Now we have to pick data register

Technique used
Bus helps to transfer information from one register to another register in a less expensive manner.

to B.

→ Each register has its load ^{bit} ~~board~~. So, the control unit ~~value~~ needs to make the value of load signal of B active i.e 1. This is how register communicate with each other using bus.

Read ^{the transfer data} (From M/M to any register)
 $IR \leftarrow M[AR]$
Instruction Register ← Memory (value at address register)
(Can be any register)

Write.

$M[AR] \leftarrow IR$ ^{from register to memory transfer} (Data of instruction register to memory).

Memory transfer.

2 Memory operation Read
Write.

→ Transfer of information from memory to outside environment is called read operation.

→ Transfer of new information to be stored into the memory is called write operation.

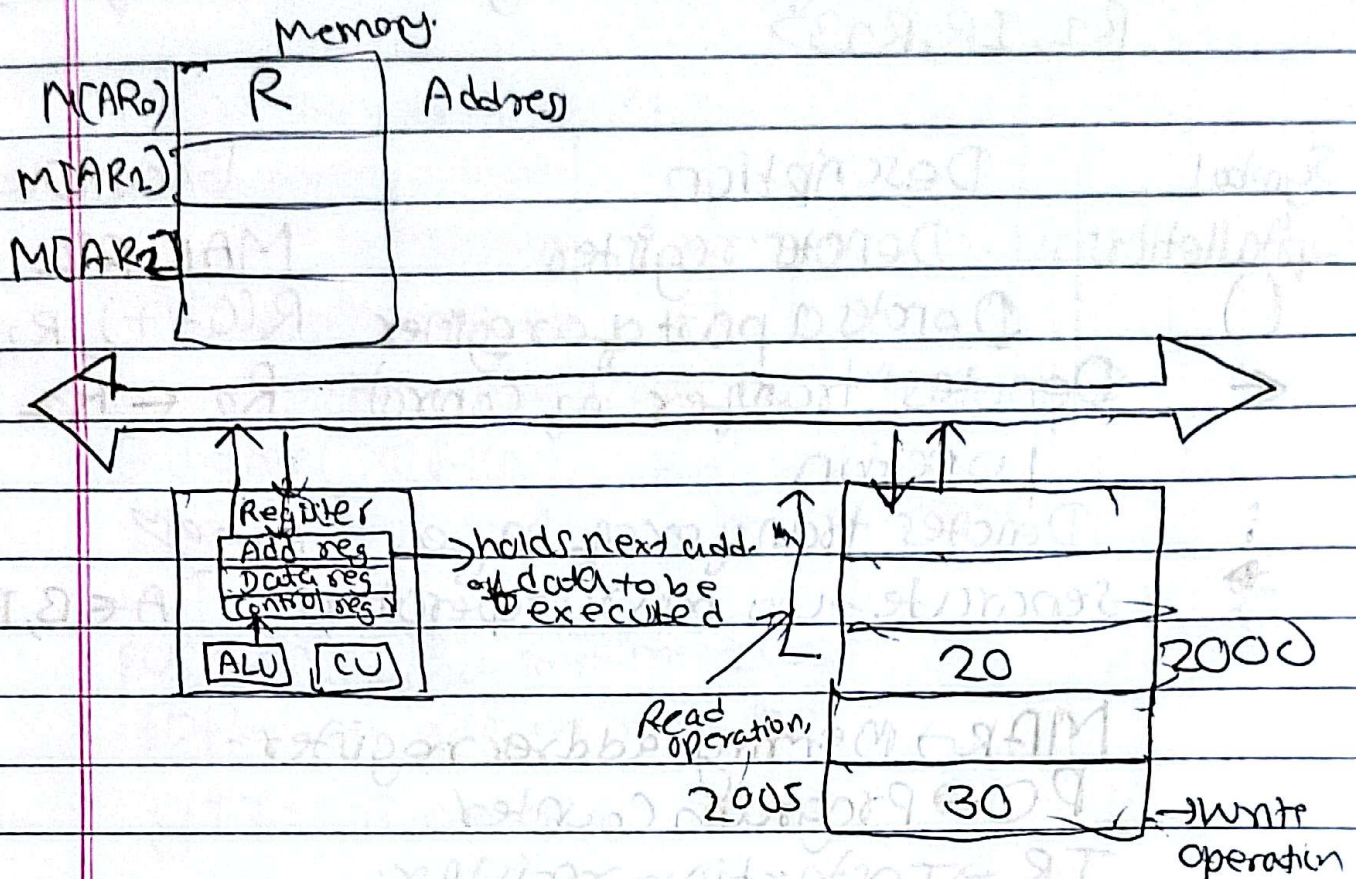
Read and Write operation can be extended as:

Read :-

$$DR \leftarrow M[AR]$$

Write :-

$$M[AR] \leftarrow R$$



Read operation :-

$$DR \leftarrow M[AR]$$

Write operation,

$$M[AR] \leftarrow R$$

Register Transfer Language :-

- Information transfer from one register to another register is called Register Transfer
- It is the symbolic representation of data

notations used to specify the sequence of micro-operation.

→ registers are designated by capital letters, sometimes followed by number (eg. -A, R₁, IR, R₂₃)

Symbol	Description	Example
Capital letters	Denote registers	MAR, R ₂
()	Denote a part of a register	R ₍₀₋₇₎ , R ₂₍₁₎
←	Denotes transfer of control	R ₂ ← R ₁
!	Denotes transfer of info	
→	Denotes two micro-operations	A ← B, B ← A

MAR → Memory address register -

PC → Program Counter

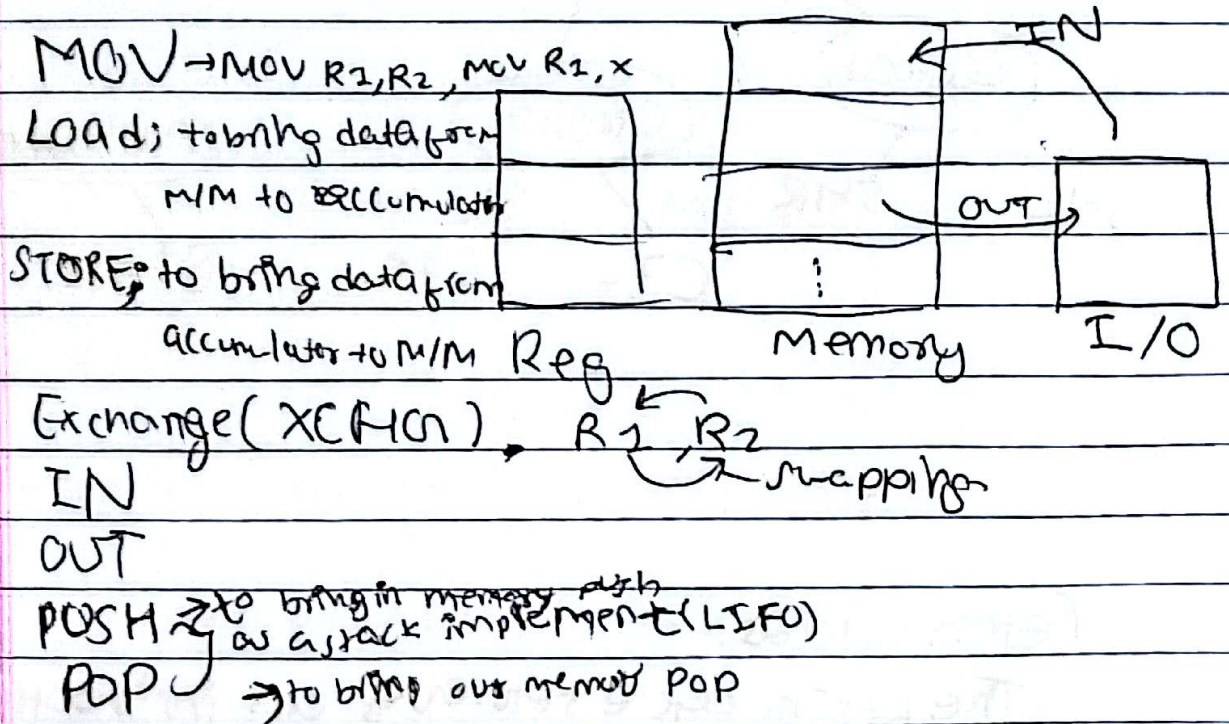
IR → Instruction register.

Register Transfer Language: RTL

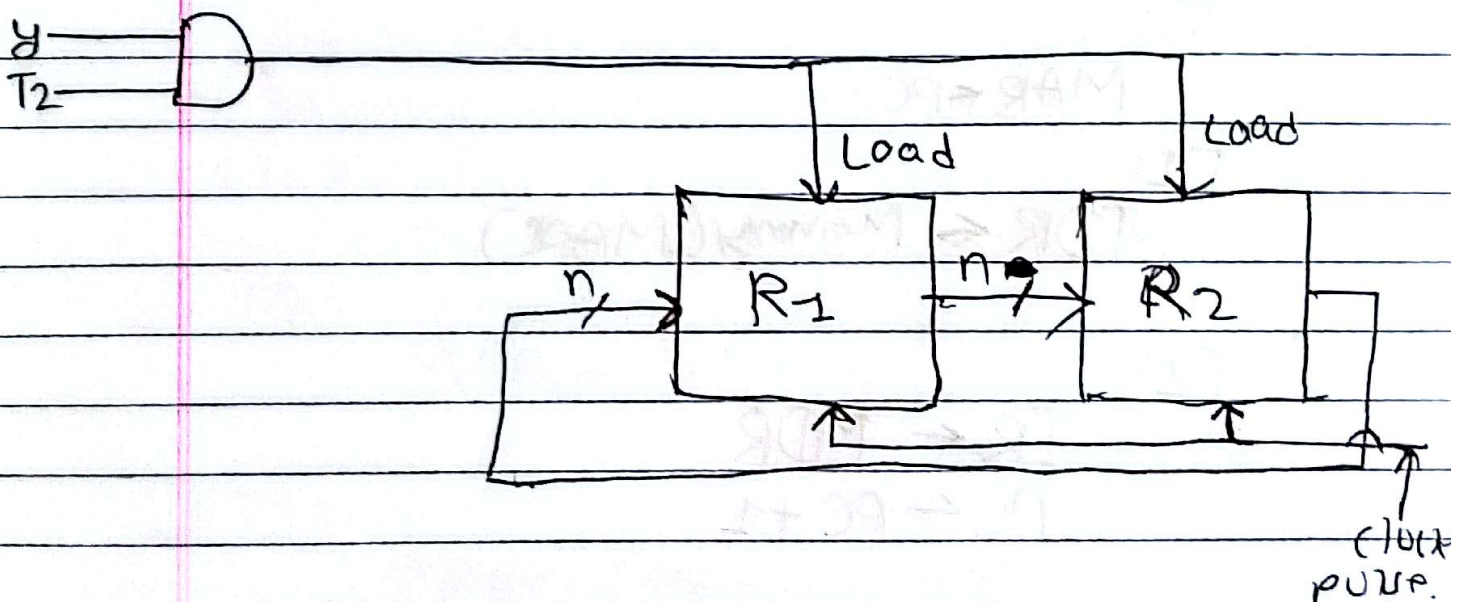
The symbolic notation used to describe the microoperation transfers amongst registers.

Q) How can data be transferred between registers? Explain in detail.

→ Using Data Transfer Instruction



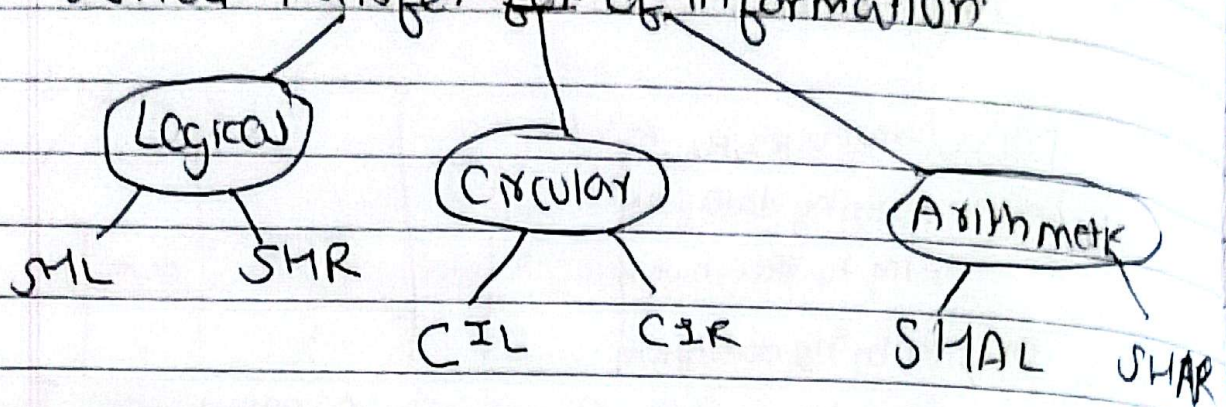
Eg:- Y12: $R_1 \leftarrow R_2, R_2 \leftarrow R_1$



2013 Fall

1(a) Describe different type of shift micro-operation.

→ Shift Microoperation are used for serial transfer of information.



2016 Spring:

1(b) Write RTL for fetch, indirect and interrupt cycles.

Fetch cycles:

The fetch cycle retrieves an instruction from memory.

① T₀:

$$MAR \leftarrow PC$$

② T₁:

$$MDR \leftarrow \text{Memory}[MAR]$$

③ T₂:

$$IR \leftarrow MDR$$

$$PC \leftarrow PC + 1$$

Indirect Cycle:

The Indirect cycle retrieves the effective address from memory when the instruction specifies an indirect addressing mode.

① T0: $MAR \leftarrow IR(\text{Address})$

② T1: $MDR \leftarrow \text{Memory}[MAR]$

③ T2: $IR(\text{Address}) \leftarrow MDR$

Interrupt Cycle:

The Interrupt cycle handles the interrupt by saving the state of the CPU and jumping to the interrupt service routine (ISR):

① T0: $MBR \leftarrow PC$

② T1: $MAR \leftarrow \text{Stack Pointer}$
 $\text{Memory}[MAR] \leftarrow MBR$

③ T2:

Stack Pointer $\leftarrow$ Stack Pointer - 1

④ T3

PC $\leftarrow$ ISR Address.

Micro-operation: The elementary operation performed by a computer is called micro-operation. The microoperations are recognized by the symbols as mention below:-

Symbol	Meaning
$\leftarrow$ (Arrow)	Indicate transfer
, (comma)	Separation between micro-operation
; semicolon	For comments.
: colon	for label

VHDL

- VHSIC : Stand for Very high speed Integrated circuit.
- Used to describe the behaviour and structure of electronic systems, particularly digital circuit.
- They are standardized by IEEE (Institute

of Electrical and Electronics Engineers
→ Application : Commonly used for designing and simulating FPGA (Field Programmable Gate Array) and ASIC (Application-specific Integrated Circuit) circuits.

~~VHDL~~ VHDL in Computer Architecture:

① Design of Microoperation Microprocessors and CPUs:

- **Behavioral Modeling:** VHDL can describe the behaviour of a CPU at a high level, such as the instruction set architecture (ISA).
- **Structural Modeling:** It allows for the detailed structural modeling of the CPU, including registers, ALU (Arithmetic Logic Units), control units and data paths.

② Simulation and Testing:

- **Simulation:** VHDL is used to simulate the behaviour of the CPU before physical implementation, ensuring that the design works as intended.

HDL (General) in Computer Architecture

Hardware Description Language

② Design Abstraction:

- High-level Description: HDLs can describe high-level architectural components, such as modules and their interconnections.
- Low-level Description: They can also describe low-level gate and transistor level details.

② Component Design:

- ALU, Registers and Memory: HDLs are used to design critical components like arithmetic units, registers, memory controllers and caches.
- Control Logic: The control logic of CPU, including finite state machines and pipeline control can be efficiently described using HDLs.

Memory Access
MR: Memory address register that holds the address for memory unit.
MDR: Memory buffer register store the data and data sent received from the memory and sent from the memory.