Syllabus for IA2 COA

- IEEE 754 format & sum
- Floating point Arithmetic unit
- Hardwired control unit
- Flynn's classification
- Booth's algorithm & sum
- Restoring division algorithm & sum
- Instruction pipelining
- I/O module
- memory characteristics
- RISC Vs CISC
- Instruction cycle state diagram.

Scanned by CamScanner

Q.8 Explain in brief the IEEE 754 format – for floating point number representation. Hence, convert the number as directed.

(i) 15 : 4 to IEEE 754 format.

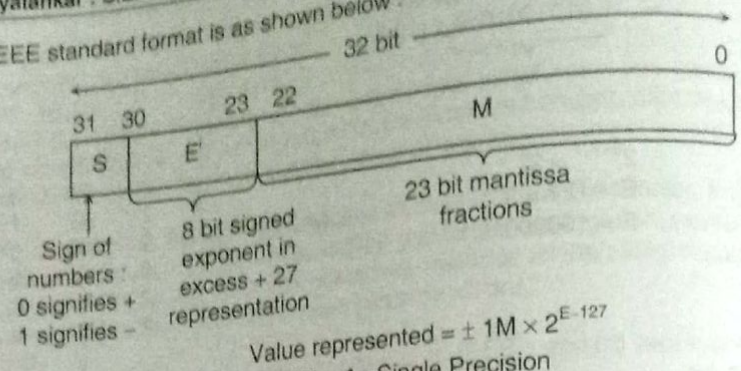
(ii) $(BFC00000)_{16}$ to floating point real number.

Ans.: IEEE 754 format :

The standards for representing floating point numbers in 32-bits and 64-bits have been developed by the Institute of Electrical and Electronic Engineering (IEEE) referred as IEEE standards.

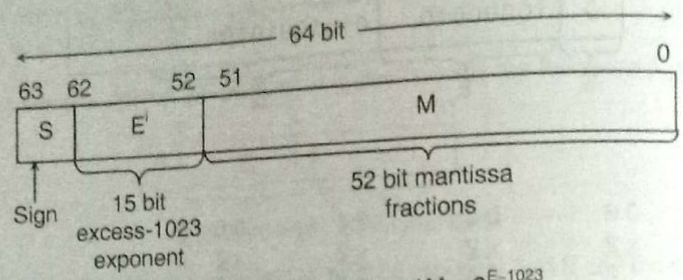
Scanned by CamScanner

IEEE standard format is as shown below :



Value represented = $\pm 1M \times 2^{E-127}$

Fig. 1 : Single Precision



Value represented = $\pm 1M \times 2^{E-1023}$

Fig. 2 : Double Precision

The 32-bit standard representation shown in Fig. 1 is called a single – precision representation because it occupies a single 32-bit word.

The 32-bits are divided into three fields as shown below :

- (field 1) sign ← 1- bit
- (field 2) Exponent ← 8 – bit
- (field 3) Mantissa ← 23 – bits

The sign of the number is given in the first bit, followed by a representation for the exponent (to the base 2) of the scale factor. Instead of the signed experiment E, the value actually stored in the exponent field is F′ = E + bias. In the 32-bit floating point system (single precision) base is 127. This is representation of exponent is also called as excess - 127 format. The end values of E′, namely 0 and 255 are used to indicate the floating point values of exact zero and infinity respectively in single precision. Thus name of E′ for normal values in single precision is a < E′ < 255. This means that for 32-bit representation the actual exponent E is in the range $-126 \le F \le 127$.

The 64-bit standard representation shown in Fig. 2 is called a double – precision representation because it occupies two 32-bit words. The 64-bits are divided into three fields as shown below :

- (field 1) Sign ← 1 – bit
- (field 2) Exponent ← 11 – bits
- (field 3) Mantissa ← 52 – bits

In the double precision format value actually stored in the exponent field is given as,

$$E' = E + 1023$$

Hence, bias value is 1023 and hence, it is also called as excess – 1023 format. The end values of E′ viz. 0 and 2047, are used to indicate the floating point exact values of exact zero and infinity, respectively. Thus, the range of E′ for normal values in double precision is 0 < E′ < 2047. This means that for 64-bit representation the actual exponent E is in one range $-1022 \le E \le 1023$.

(i) 15.4 :

Binary = $(1111 . 0110011)_2$

Normalized the number = 1.1110110011×2^3

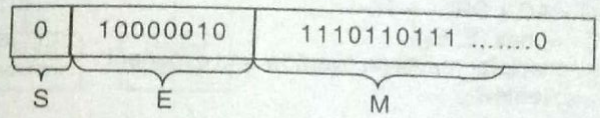
for single precisions

S = 0, E = 3 and M = 1110110011

E′ = E + 127 = 3 + 127 = 130

= $(10000010)_2$

Number in 32-bit format is given as,



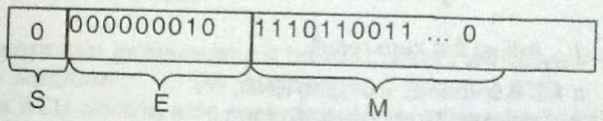
Double precision

S = 0, E = 3 and M = 1110110011

E′ = E + 1023 = 3 + 1023 = 1026

= $(10000000010)_2$

In double precision format



(ii) $(B\ F\ C\ 00000)_H$

Given number

0 1011 1111 1100 0000 0

Exponent = 191

E′ = E + 127 = 191 – 127

= 64

Q.9 Give the algorithm for adding two floating point numbers to handle check for the overflow, underflow errors.

Ans.: **Declare** : Register AC $[n_M - 1]$, DR $[n_M - 1]$
$E[n_E - 1]$, $E_1[n_E - 1]$, $E_2[n_E - 1]$
AC – Overflow, Error

Begin :

Load : // initialize exponent and mantissa of two numbers.

Scanned by CamScanner

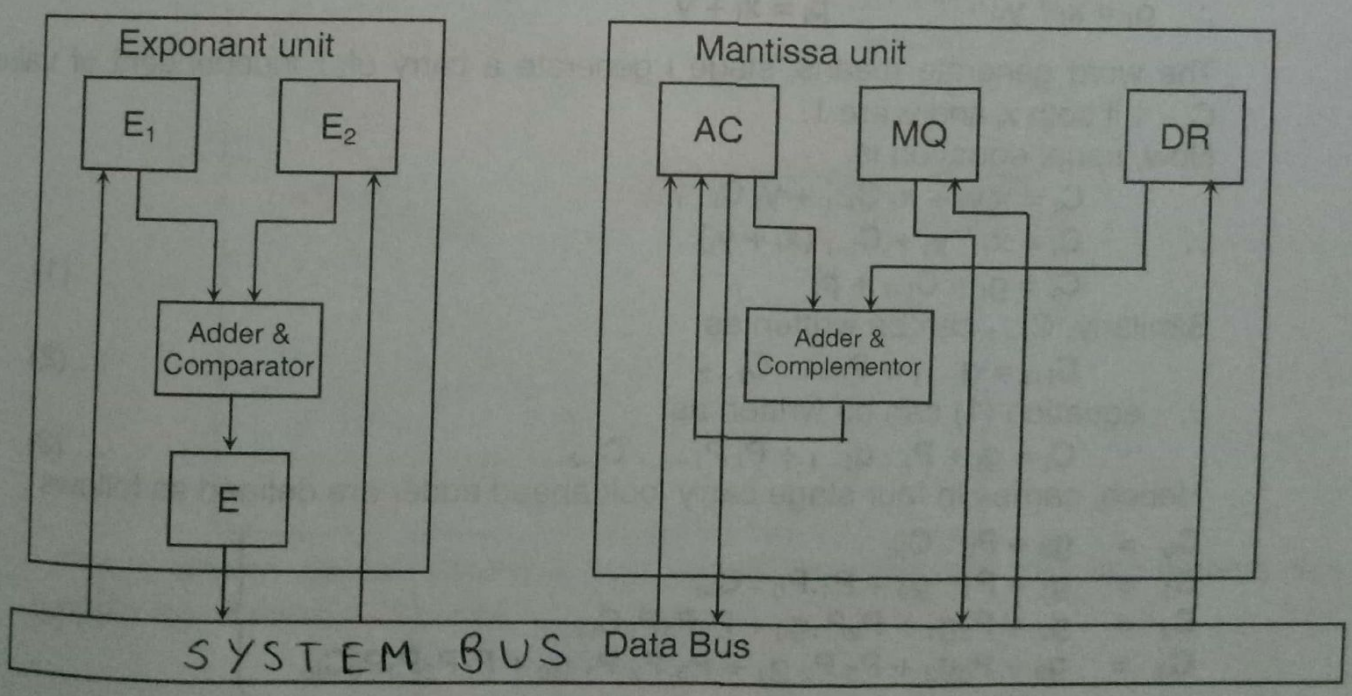
Q.18 Write detailed note on Floating point arithmetic unit.

Ans.: The floating point arithmetic unit can be implemented by two loosely connected fixed point data parts circuits, an exponent unit and Mantissa unit.

The Mantissa unit performs all 4 basic operations on Mantissa parts. Hence, its structure is similar to fixed–point arithmetic unit.

A similar circuit that performs adding, subtracting and comparing of exponent is suffers to exponent unit. Exponent comparisons can be done by comparator or by the subtracting the exponent.

The output of floating point arithmetic unit :



Scanned by CamScanner

The exponents of input operands are put in the E_1 and E_2 registers and these are given to adder to calculate $E_1 + E_2$. The exponent comparison required for addition and subtraction is made by computing $E_1 - E_2$. The large exponent is identified by sign of F.

The shifting of mantissa before mantissa addition or subtraction can occur is controlled by E. The magnitude of F is sequentially decremented by shifting mantissa by one digit position at each time until E is decremented to zero. Once mantissa have been aligned, they are processed in usual manner. Exponent addition i.e., exponent of result is stored in E.

All computers with floating–point instruction also have fixed point instructions, so it is sometimes desirable to design a single ALU to execute both fixed point and floating point instructions. This design takes a term of fixed point arithmetic unit in which register and adder can be partitioned into exponent and mantissa parts (i.e., floating point unit) when floating point operations are being performed.

Now days, all computers have two separate units, a fixed or integer point unit and floating point unit. This separation makes it possible for fixed point, floating point instructions to be executed in parallel.

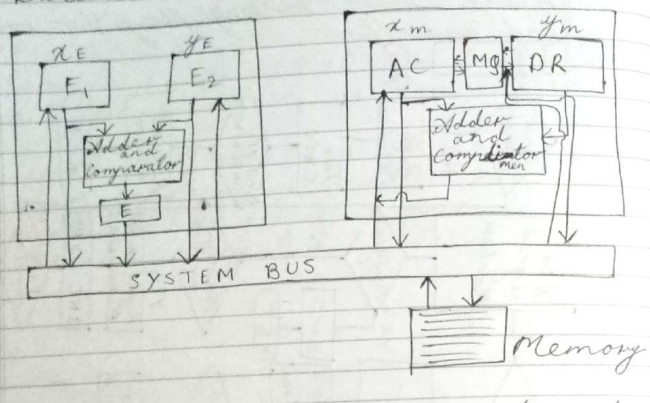
9 Write short note on Floating point ALU.

A.: Refer Answer of Q.18

Scanned by CamScanner

Short Note: **Floating Point Arithmetic Unit (FPAU)**

External Unit | Mantissa Unit



Note: The FPAU consists of 2 independant units:

1. Exponent unit: Its E_1 and E_2 reg. are initialized with x_E and y_E. On these exponents it performs following 3 operations: $x_E + y_E$, $x_E - y_E$, comparison of exponents.
   The result of comparison is held by reg. E and then made available by system bus so that mantissa unit can understand which exponent is smaller and M of smaller exponent is loaded in AC reg.
2. Mantissa unit: It is generic sequential arithmetic unit, that performs following 7 operations on the mantissa of FP nos.

i. $x_m \times y_m$ ii. $x_m \div y_m$ iii. $x_m + y_m$
iv. $x_m - y_m$ v. Right shifting of x_m during expo. equalization
vi. Left shifting of Mantissa for normalization vii. Result Rounding.

The rounded result of AC and MQ is made available on system bus so that it can be stored in external memory

New Module **Processor Organization**

Instruction cycle state diagram:

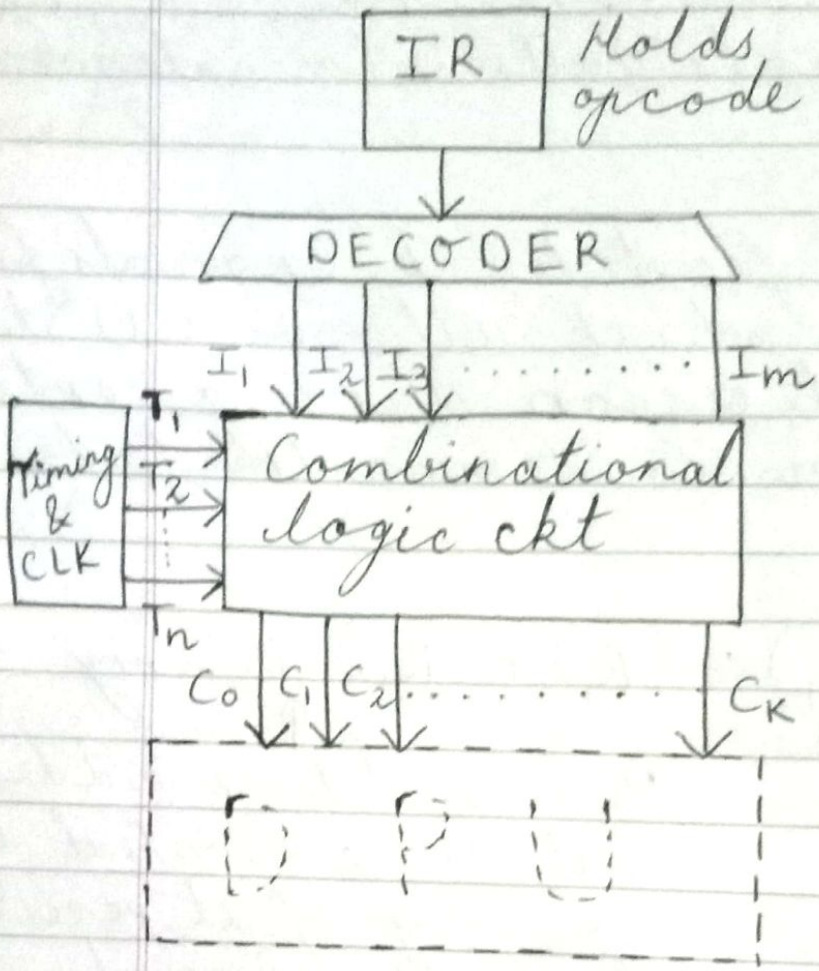
Let the instructions under execution are:

ADD R_1, R_2 ; $(R_1) \leftarrow (R_1) + (R_2)$
ADD $R_1, [R_2]$; $(R_1) \leftarrow (R_1) + M[R_2]$
ADD $[R_1], R_2$; $M[R_1] + (R_2)$

Def^n of instruction: Total time requ for an instruction's execution t the processor is called instructi cycle which consists of fetch and execute cycle.
The IC state diagram consists 5 internal states and 3 externa states as shown below:

Scanned by CamScanner

Hardwired C.U.



Operating Principle:

1. Opcode in I_R is immediately decoded to initiate inst. codes I_1 to I_m which are immediately processed by combinational logic circuits under supervision of timing and clock to generate control

slg C_0 to C_K which directly operates on DPU to carry out inst's execution.

2. Its combinational logic ckt is designed using low cost logic gates & flip-flop when have very small propagation delay and hence its response speed is very fast.

3. Due to combinational logic circuit which is complex and random in structure it does not allow to make any changes in the behaviour of hardwired CU.

4. Due to small cost and fast speed these are found in playing toys or video game machines.

Scanned by CamScanner

Q. Explain the 3 Hardwired CU implementation Methods.

A. The various methods used for implementation of HCU:

1. State table method
2. Delay element "
3. Sequence counter method.

1. State table method: This method is suitable for implementation of those CU whose behaviour is similar to finite state sequential machine, for this method the original description about CU behaviour is required in the form of 'STATE TABLE'. In this table foll 2 types of entries are expected.

'S_{ij}': Next state to be triggered.

'Z_{ij}': indicates control signal to be generated when the ctrl unit is in state S_i & the decoder activates inst code 'I_j'.

The standard format of this state table will be as shown below: In this table we have represented execution of instruction like ADD R_1, R_2.

STATE	Instruction Code Combination			
	I_1	I_2		I_m
S_1	S_{11}, Z_{11}	S_{12}, Z_{12}		S_{1M}, Z_{1M}
S_2	S_{21}, Z_{21}	S_{22}, Z_{22}		S_{2M}, Z_{2M}
⋮				
S_n	S_{n1}, Z_{n1}	S_{n2}, Z_{n2}		S_{nm}, Z_{nm}

ER1 ER2 EADD

The above state table consists of entries in the form of 0's and 1's. On this state table designer can apply std. table minimization techniques. Eg: K-map. Logical implementation of table is then possible using very few no. of logic gates and flip-flops so that they will have very small propagation delay and hence the response speed will be very fast.

Advantages:

1. Easy implementation
2. Very low cost
3. Fast speed.

Disadv:

1. The CU designed using state table method is non-reprodurable this is bcoz the original state table and CU implementation using minimized

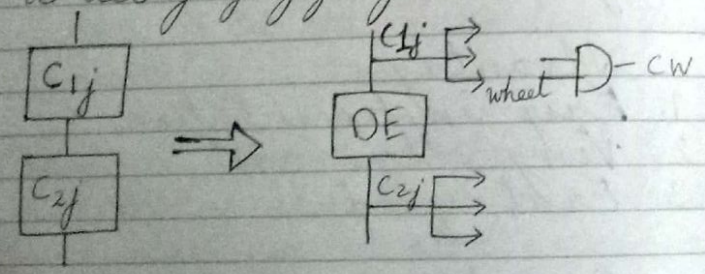
Scanned by CamScanner

table doesn't have one-to-one mapping. In short it conceals info. about CU behaviour and is undesirable.

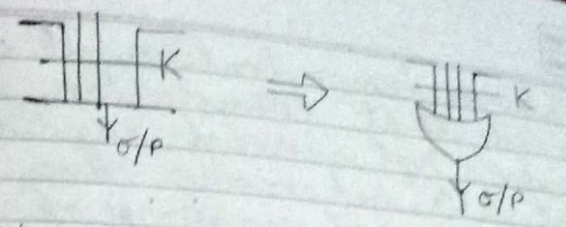
To overcome this drawback: "Delay element method" is suggested.

2 <u>Delay Element Method</u>: It is suitable for the implementation of those CU whose behaviour consists of complex decisions and repeated loops & patterns. The original description about CU behaviour is reg. in the form of HDL (Hardware Description Lang) flowchart. On this HDL, designer must apply following 3 basic rules:

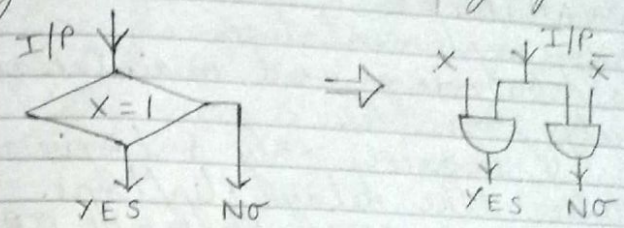
<u>RULE 1</u> It defines successive cont.^{n} sigs need to be provided with necessary tm separation using delay element (i.e a delay flipflop) as shown:



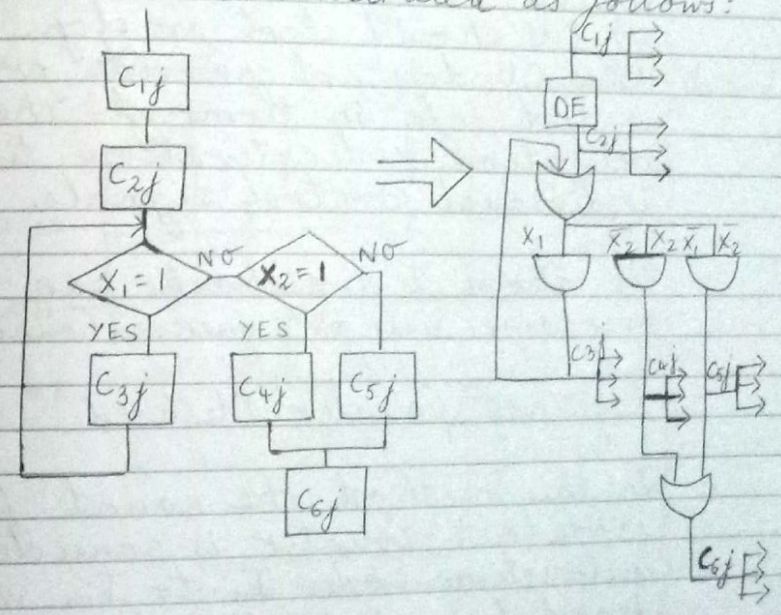
<u>RULE 2</u> If no of cont^{n} lines are merging together in HDL flowchart then it can be implemented using multi-input OR-gate as shown



<u>RULE 3</u>: The decision box in the HDL can be implemented using pair of AND gate.



The translation of given HDL into its delay element representation can be demonstrated as follows:



In above demonstration it can be observed that translation of given HDL does not conceal any info. about CU's behaviour due to this reason of repairing of these CU, expert designer isn't required.

ADVANTAGES:

1. Easy implementation
2. Its debugging and maintenance is easily possible.
3. This CU requires only 3 logical entities like delay flipflops. (Multi-input AND gate and OR gate)

DISADVANTAGES:

1. Does not provide supervisory control signals to define when the operation of CU should start or stop.
2. This CU does not provide precise control info. in terms of the activation & deactivation time of individual control signals.

The above 2 drawbacks can be overcome using Sequence Counter Method.

3. Sequence Counter Method.

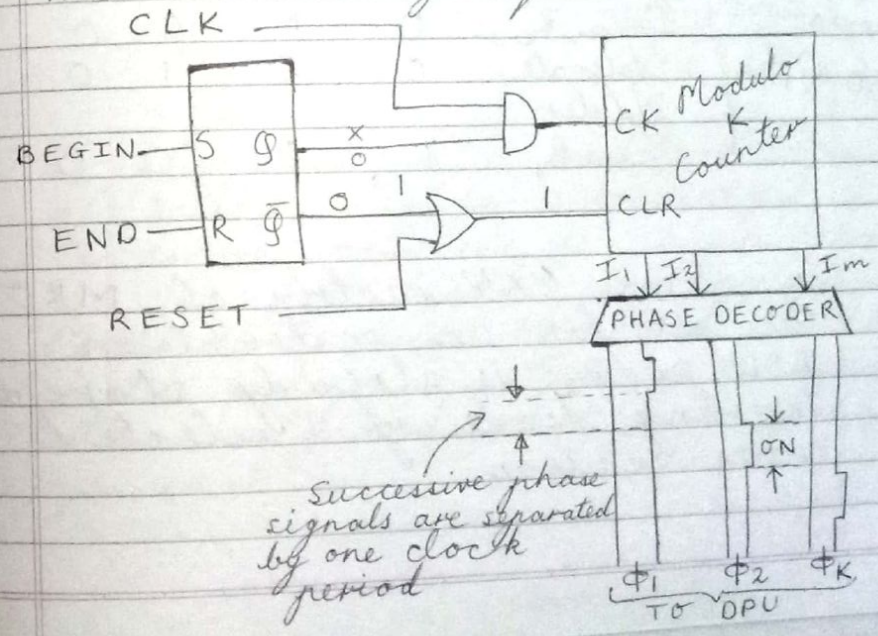
In this method the count from modulo K-counter is considered as instruction code I_1 to I_m which is decoded by specially designed

Phase ~~Haze~~ decoder' to initiate control signals $\Phi_1, \Phi_2, \ldots \Phi_K$.

The 'ON' time of individual phase signal defines exact duration for which the control signal should remain in its enable condition and hence it provides precise control info. This method also provides supervisory control signals - BEGIN and END to indicate or define when operation of CU should start or stop.

NOTE: SEQUENCE COUNTER METHOD is suitable for the implementation of CU which should generate same set of control signals again & again.

The structure of Sequence Counter Method:



Scanned by CamScanner

Q.1 Explain the Flynn's Classification in detail?

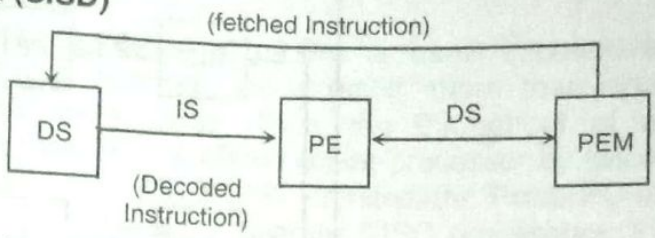
[CMPN : D-15,16, M-16; INFT : M-16, D-16]

Ans.: Flynn's classification of Computer Organisation :

The classification suggested by Flynn, is based on multiplicity of instruction stream (IS) and data stream (DS).

1. **Single Instruction Single Data (SISD)**

This is normal single processor system (UNI processor) which contains single decoded instruction stream (IS) which operates on a single data stream (DS).

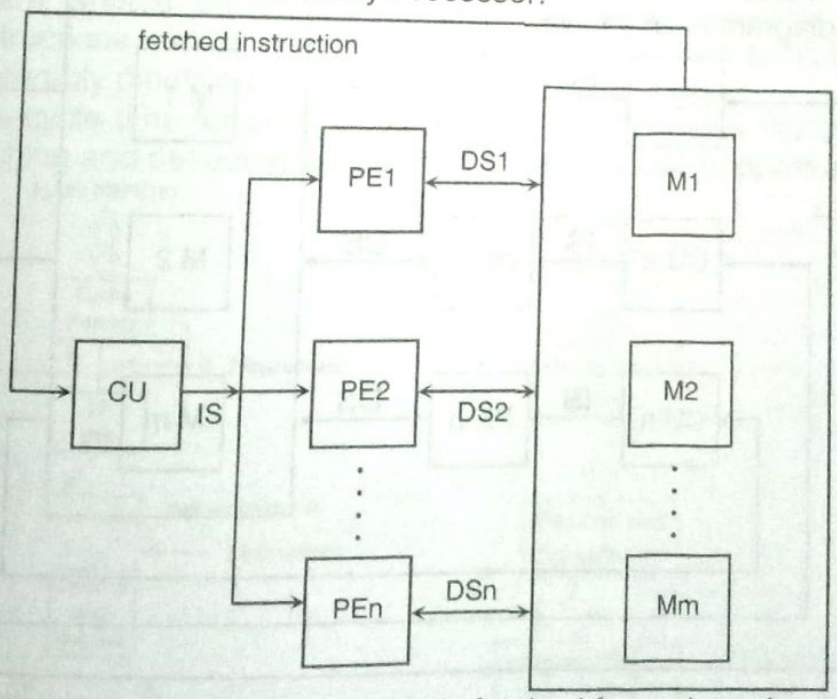


- Here control unit (CU) fetches instruction stream from processing element memory (PEM), which is decoded to generate single decoded instruction stream (IS). This is executed by processing element (PE) using single data stream (DS).
- Once the result data is generated by PE, it is stored back to the PEM.

Example : of SISD architecture are INTEL, 8085, 8086, 80386.

2. **Single Instruction Multiple Data (SIMD)**

- This is also referred as Array Processor.



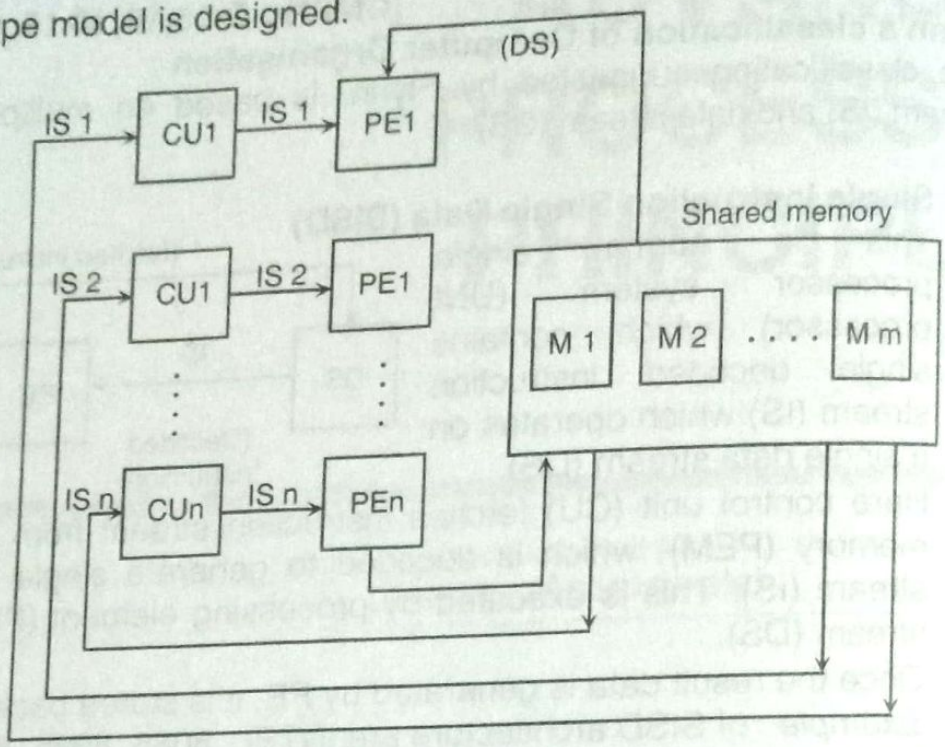
Here the single instruction stream is fetched from shared memory modules or taken from external front end system. It is decoded by a single control unit (CU), to generate decoded instruction steak (IS). This instruction is broadcasted to multiple processing elements (PES), which will operate on different data sets. Hence the name given (SIMD). The result generated by processing elements (PEs) are stored back into memory modules.

3. **Multiple Instruction Single Data (MISD)**

Here multiple instruction streams are fetched form shared memory modules by multiple control units which in turn generates multiple decoded instructions stream (IS). These are operated on single data streak (DS) taken

Scanned by CamScanner

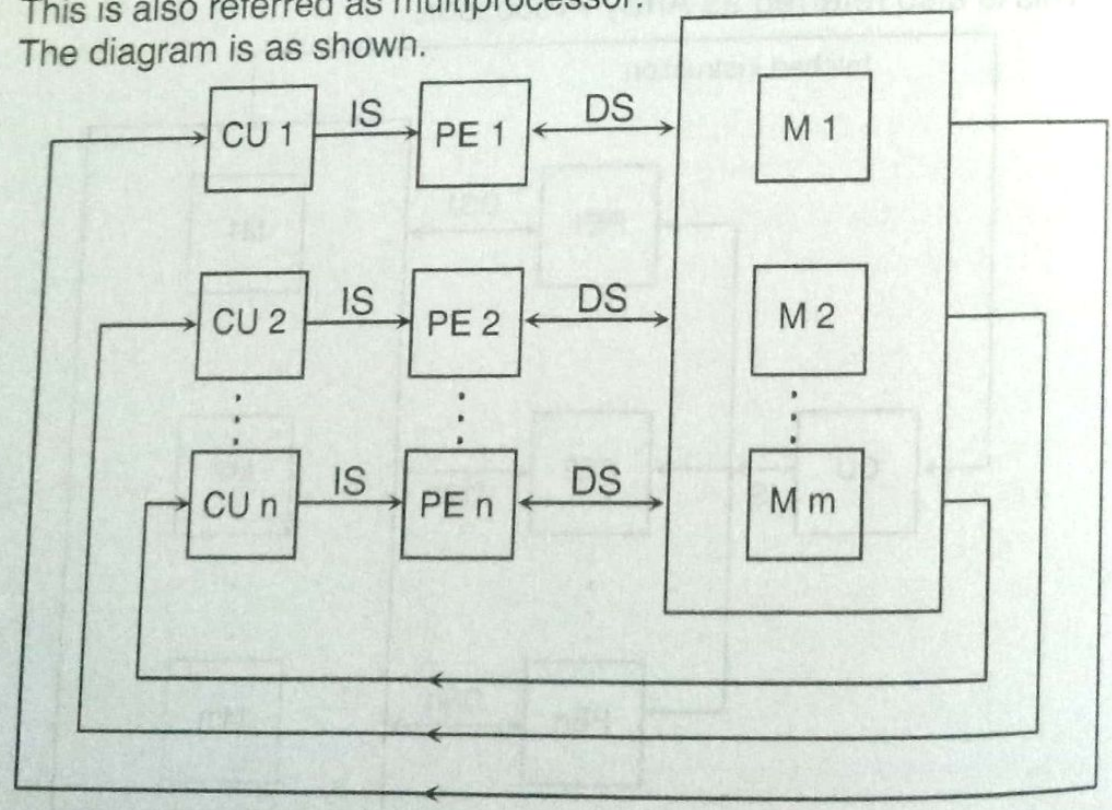
from shared memory modules. MISD architecture is not realized but just a prototype model is designed.



4. Multiple Instruction Multiple Data (MIMD)

This is also referred as multiprocessor.

The diagram is as shown.



Here, multiple instruction streams are fetch by control units. These instruction stream are decoded to get multiple decoded instruction stream (IS), which operates on multiple data stream (DS) taken from shared memory modules. Here, each processing element executes one instruction stream and operate exactly on one data stream.

Scanned by CamScanner

	A – Register	Q – Register
Initially	0 0 0 0 0	1 0 0 1
shift	0 0 0 0 0	0 0 1
Subtract	1 1 1 1 0	0 0 1 0
Set q_0	1 1 1 1 1	0 1 0
Shift Add	1 1 1 1 0	0 1 0 1
Set q_0	0 0 0 1 0	1 0 1
Shift	0 0 0 0 0	1 0 1 0
Subtract	1 1 1 1 0	0 1 0
Set q_0	1 1 1 1 1	0 1 0 0
Shift Add	1 1 1 0 1	0 1 0
Set q_0	0 0 0 1 0	
	1 1 1 1 1	Quotient
Add	0 0 0 1 0	
	0 0 0 0 1	
	Remainder	

Q.30 Explain and solve the following problem using by non-restoring division algorithm. Hence divide $(10)_{10}$ with $(3)_{10}$.

Ans.: *Refer Answer of Q.29*

Q.31 Explain Booth's algorithm.

Ans.: Booth's algorithm is depicted as in following figure and can be described as follows :

1) Multiplier and Multiplicand are placed in Q and M registers respectively.
2) There is also a 1 bit register placed logically to the right of the least significant bit (Q_0) of the Q register and designed Q_{-1}.
3) The result of the multiplication will appear in the A and Q register. A and Q_{-1} are initialized to zero.
4) Control logic scans the bits of the multiplier one at a time. Now, as each bit is examined, the bit to its right is also examined.
5) If the two bits are the same (1 – 1 or 0 – 0), then all of the bits of the A, Q and Q_{-1} registers are shifted to the right 1 bit. If two bits differ, then the multiplicand is added to or subtracted from the A register, according as the two bits are 0–1 or 1–0.
6) Following the addition or subtraction, the right shift occurs. In either case, the right shift is such that the leftmost bit of A, namely A_{n-1}, not only shifted into A_{n-2}, but also remains in A_{n-1}. This is required to preserve the sign of the number of A and Q.

   It is known as an arithmetic shift, since it preserves the sign bit.

e.g. M = 0111 Q = 0011

A	Q	Q_{-1}		
0000	0011	0	Initial	
1001	0011	0	A ← A – 1	First cycle
1100	1001	1	shift	
1110	0100	1	shift	
0101	0100	1	A ← A + M	Second cycle
0010	1010	0	shift	
				Third cycle
0001	0101	0	shift	Fourth cycle (product in A, Q)

Result = $(00010101)_2 = (143)_{10}$

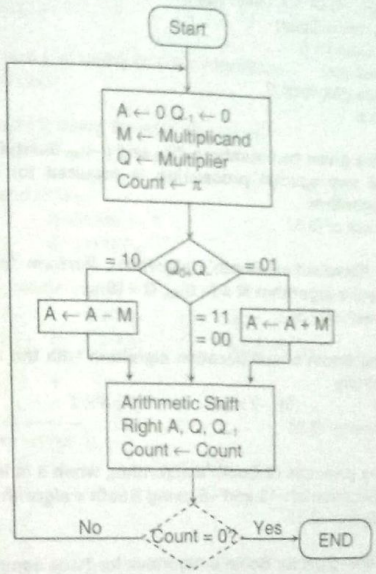


Fig : Booth's algorithm for two's complement multiplication

Algorithm

Case – I : If $Q_i = 0$, $Q_{i-1} = 1$
then add M to partial product

Case – II : If $Q_i = 1$, $Q_{i-1} = 0$
then subtract M from partial product

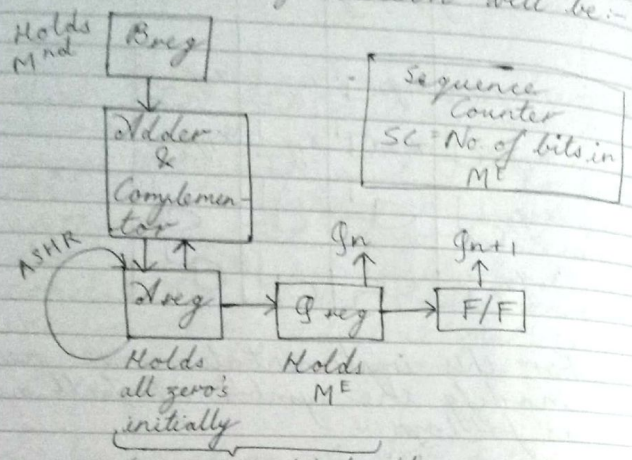
Case – III : If $Q_i = 0$, $Q_{i-1} = 0$ OR $Q_i = 1$, $Q_{i-1} = 1$
then, Rshift

Declare : A(7 : 0) Q(7 : 0), M(7 : 0), Count (3 : 0)

Scanned by CamScanner

Booth's Algorithm for signed multiplication

The Hardware required for this Booth's Multiplication will be:-



It will hold double length signed product

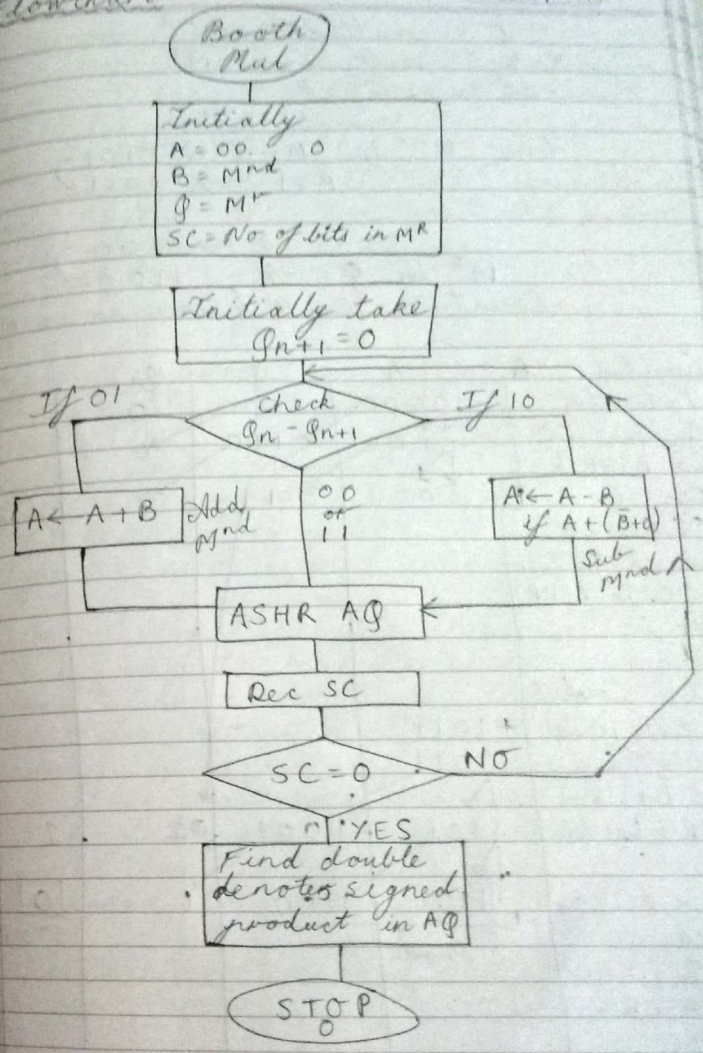
Diff^n b/w ASHR & LSHR
ASHR ↳ Arithmetic Shift Right
LSHR ↳ Logical Shift Right

Let

AQ ⇒	1010	0110	Q_{n+1}
ASHR ⇒	1101	0011	0
ASHR ⇒	1110	1001	1
LSHR	0101	0011	0
LSHR	0010	1001	1

Q Explain Booth's algo with the help of flowchart and perform (5)x(-4) using Booths algorithm.

ANS: Flowchart:



Scanned by CamScanner

5 ← multiplier M^{nd}
× −4 ← multiplicant M^{E}
−20 ← product

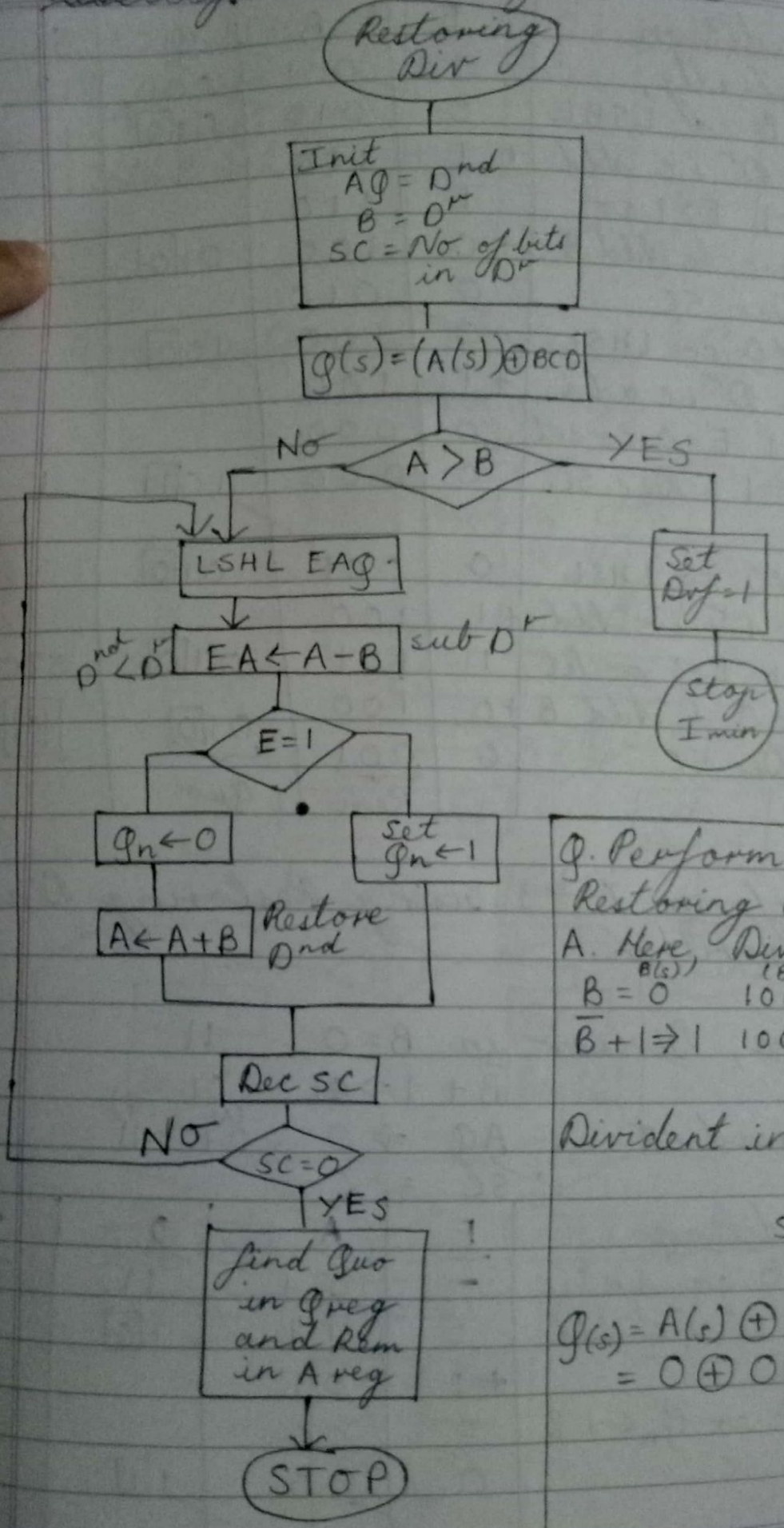
Here, M^{nd} in $B = 5 \Rightarrow 0101$
$B+1 \rightarrow 1011$

$4 \Rightarrow 0100$
M^{E} in $Q = -4$ 1100
Count in SC = 4 (since no in ME is 4)

Condition	A	A	Q	Q_{n+1}	SC
Initially, $Q_n Q_{n+1} = 00$ so ASHR & Dec SC.		0000	1100	0	4
		0000	0110	0	3
		0000	0011	0	2
SC ≠ 0 so check $Q_n Q_{n+1} = 00$. So ASHR Dec SC					
Set 0 to check $Q_n Q_{n+1} = 10$ so sub M^{nd} i.e add $\bar{B}+1$ & ASHR & Dec SC		+1011 1011			
		1101	1001	1	1
Set SC ≠ 0 so check $Q_n Q_{n+1} = 11$ so ASHR & Dec SC		1110	1100	1	0 STOP
		Product			

Scanned by CamScanner

Restoring Division Algorithm



Q. Perform 9 ÷ 4 using Restoring Div Algo.

A. Here, Divisor in

B(s) (B)

$\underline{B} = 0$ 100 2's complement

$\overline{B} + 1 \Rightarrow 1$ 100

Divident in AQ ⇒ A(s) 0

001001

SC = 3 (no. of bits in D^r)

$Q(s) = A(s) \oplus B(s)$

$= 0 \oplus 0 = 0$

Scanned by CamScanner

Condition	E	A	Q	SC
Initially,	–	001	001	3
A<B so LSHL	0	010	01[0] Q_n	
sub D^r i.e Add	+1	100		
$\bar{B}+1$ E=1 so	1	110		
$Q_n \leftarrow 0$ & Add B	+0	100	01[0]	2
& Dec SC	0	010		
SC≠0 so LHSL	0	100	10[0]	
sub D^r i.e add	+1	100		
$\bar{B}+1$ E=0 so set	0	000		
$Q_n \leftarrow 1$ & Dec SC	0	000	10[1]	1
SC≠0 so LHSL	0	001	01[0]	
sub D^r, i.e Add	+1	100		
$\bar{B}+1$ E=1 so let	1	101	01[0]	STOP
$Q_n \leftarrow 0$ & Add B	+0	100	01[0]	[0]
& Dec SC	0	001 (Rem)	(Quo)	

Q. Perform 7 ÷ 3 using Restoring Division Algorithm.

A. Here, Divisor in B = 0 11

$\bar{B}+1 \Rightarrow 01$ 01

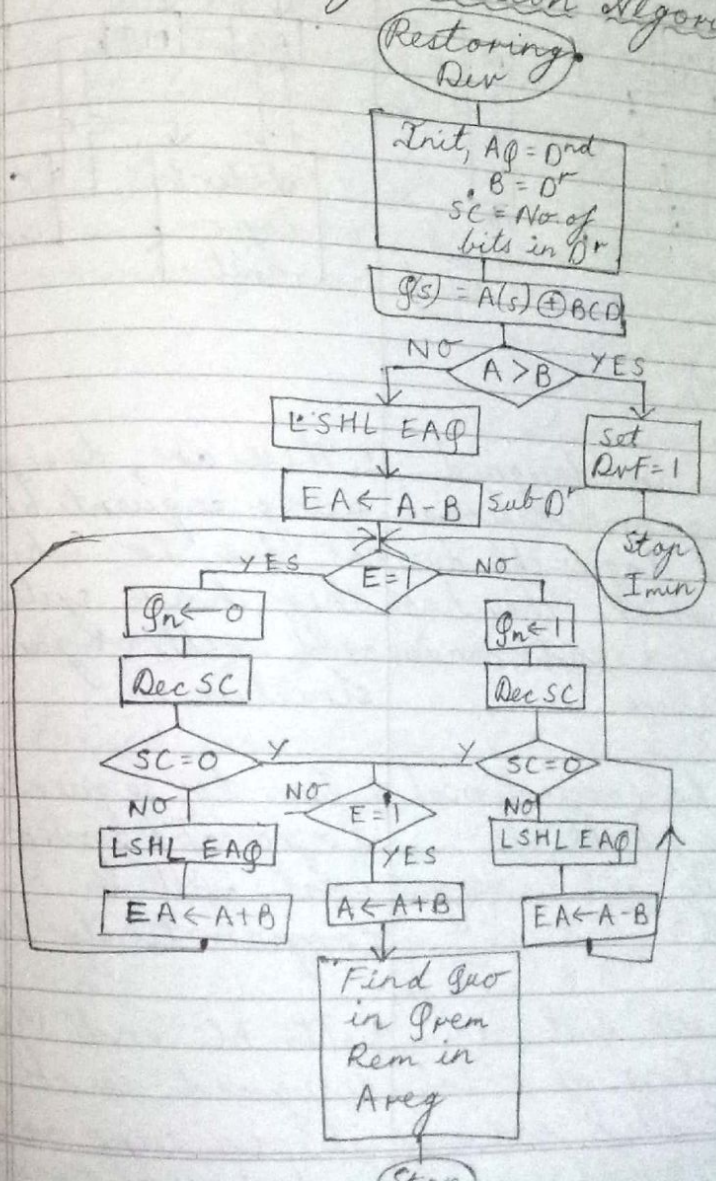
Divident in AQ ⇒ 0 (A) 01 (Q) 11

SC = 2

Condition	E	A	Q	SC
A<B so LHSL	–	01	11	2
Sub D^r i.e Add	0	11	1[0]	
$\bar{B}+1$	+1	01		
E=0 so $Q_n \leftarrow 1$ &	0	00		
Dec SC	0	00	1[1]	1

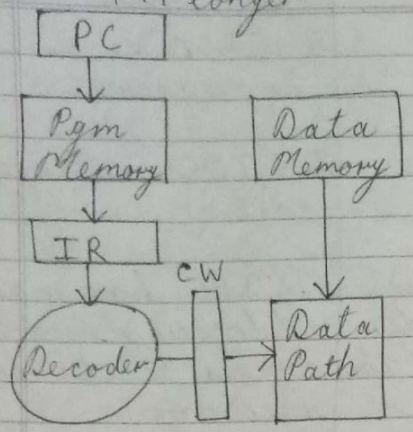
SC≠0 so LSHL sub	0	01	1[0]	
D^r i.e add $\bar{B}+1$	+1	01		1
E=1 so set $Q_n \leftarrow 0$	1	10		
& Add B & Dec SC	+0	11		
	0	01 (Rem)	1[0] (Quo)	[0]

Non-Restoring Division Algorithm

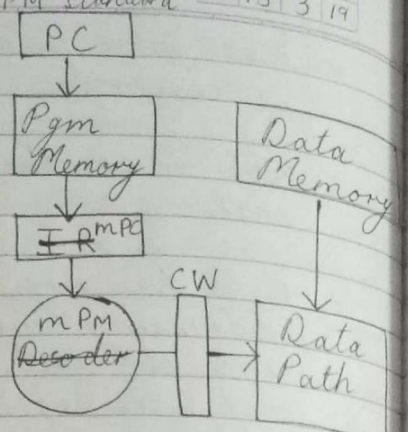


Scanned by CamScanner

Simple instⁿ
1 instⁿ = 1 CW
PM longer



Complex instⁿ
1 instⁿ - several CW
PM standard



Pipelining Concept: It defines overlapping operations during an inst. execution to inc. overall performance of the processor. The standard processor's could be implemented by 2 pipelining:

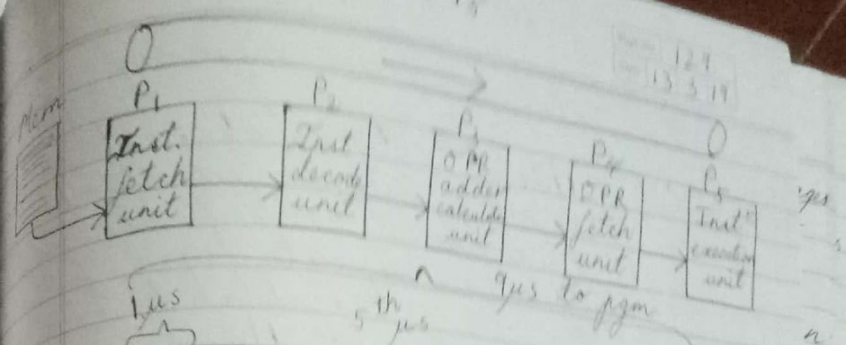
1. Instruction pipelining. (IP)
2. Data path " " (DPP)

1. Inst. pipelining:

Q. Draw and explain 5 stage IP.

A. The term IP defines the way in which various operations involved in the individual inst. execution can be overlapped to inc. prog. execution speed of pro. To implement IP pro's architecture is divided into no. of processing units as shown in the following diagram:

For 5 pages add P_6
" 4 " " remove P_5



	1μs			5th μs				
P_1	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9
P_2	-	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8
P_3	-	-	I_1	I_2	I_3	I_4	I_5	I_6	I_7
P_4	-	-	-	I_1	I_2	I_3	I_4	I_5	I_6
P_5	-	-	-	-	I_1	I_2	I_3	I_4	I_5

Prog. of 5 pages.

Timeslot →

In above IP individual inst. of the pgm. enters in the processor through P_1 and it is finally executed in P_5. On every inst. P_2-P_5 will perform certain predefined operation and this has resemblance of pipelined operation and hence called

In above time analysis diagram it can be observed that in the 5th time-slot due to inst. pipelining following operations are overlapped:

P_5 executes I_1
P_4 fetches operand for I_2
P_3 calculates address of operand for I_3
P_2 decodes I_4

Scanned by CamScanner

P_1 fetches I_5 from memory.

After this 5th timeslot each inst of the prog. will be executed in just 1 μ secs so it will takes only 9 μ secs for inst to execute pgm of 5 inst. On, the other hand for non-pipelined pro. the time req. for a prog. of 5 inst will be 25 μ secs.

Q. What is inst. hazard?

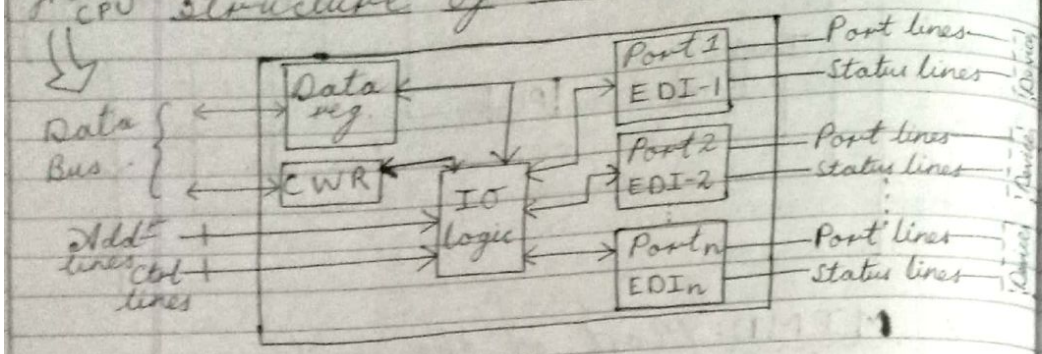
A. In the IP if the branch inst. is encountered then the whole processor's inst. pipeline need to be flushed which is a kind of penalty need to be paid by the pro. and it may degrade performance of IP. This problem is known as Inst. hazard. To minimize this problem from analysis point of view they have suggested no. of processing units in the IP to be restricted to only 5 units.

Scanned by CamScanner

I/O ORGANIZATION

I/O Module and its Responsibilities.

Structure of I/O Module



Responsibilities of I/O Module.

1. CPU communication.
2. Device "
3. Data buffering to match slow data transfer rate of I/O devices with fast data transfer rate of CPU.
4. Ctrl and timing of data transaction by deciding when to collect data from i/p devices like keyboard and when it is transferred to CPU.
5. Organization of I/O module: F^n block diagram of I/O module as shown in above fig. consists of foll. units:

5.1. Data register - It interfaces I/O module with data bus of CPU.

5.2. Ctrl Word Reg - It has been programmed

by appropriate ctrl word by CPU to configure individual port either as I/P or O/P port depending on type of device to the port.

5.3 I/O logic - Its address lines are used to select individual port one at a time with which CPU can perform data read or data write operation. Its ctrl. lines used to define data transfer direction.

5.4 EDI: (External device interface): Also known as port which is physically consisting of a latch and buffer to provide necessary port lines and status lines.

Types of I/O Modules:

1. I/O controller: These I/O modules provide primary ctrl. on data transfer activity of I/O devices with the help of CPU. [Intel 8255 PPI] is an eg. of I/O controller found mostly in personal computer or desktop.

2. I/O channel: These I/O module can provide fairly complete ctrl on standard I/O devices without any interference from CPU. [Intel 8089 IOP] (Independant I/O processor) which runs its own pgm and handle data transfer with I/O devices independantly.

Scanned by CamScanner

Memory Organization.

Q. Explain various Memory characteristics.

A. The characteristics of memory systems are described using following 6 parameters:

1. Location: - CPU - If the mem. is located inside the CPU then it is called cache memory.
- Main mem. - Mem. unit directly accessible to C.P.U. It is dynamic (D-RAM)
- AUX mem. - Mem. unit not directly accessible to C.P.U. but provided in large size due to its small cost to provide back-up storage. (FD/HD/CD).

2. Capacity: $\{(2^n \times m \text{ bit})\}$ - Word Size: (m-bit). Indicates no. of data pins mem. system can have. If word size is m-bit then it can have $(D_0 - D_{m-1})$ data bits.

- No. of words (2^n): It indicates no. of addr. pins that mem. systems have which is n bit $(A_0$ to $A_{n-1})$ if the mem. systems have 2^n words storage capacity

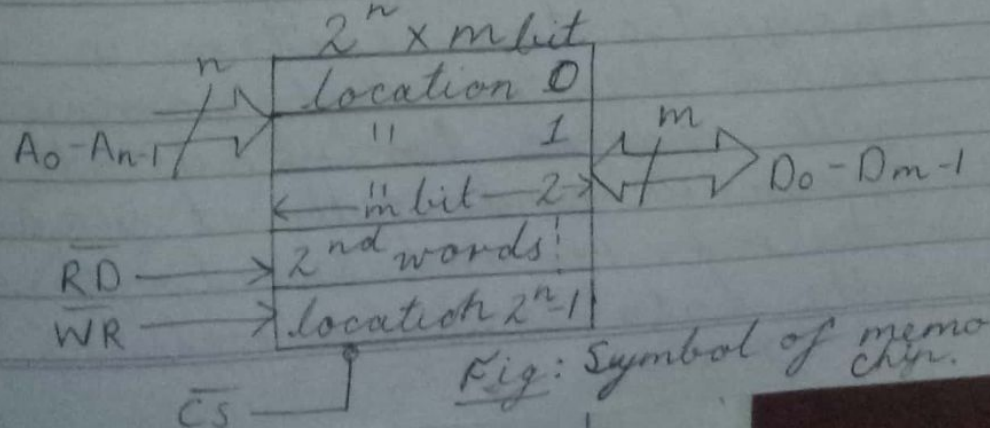
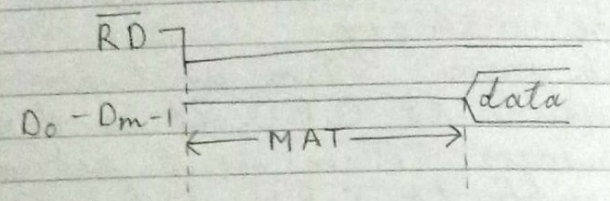


Fig: Symbol of memory chip.

Scanned by CamScanner

3. Performance of mem. system calculated using 3 parameters:

- MAT: The time gap b/w activation of read slg and the time at which data placed on the data pins $D_0 - D_{m-1}$ is called MAT



- MCT: Mem. cycle time - Time that must be elapsed b/w 2 successive read/write operation of the mem. so that mem system can respond properly is called MCT.

- Data Transfer Rate: Inversely prop. to MAT and MCT.

4. Access Methods: Diff. mem. systems will support various accessing methods as follows:

- Sequential access: Supported by magnetic tape system in which to read the record, computer system will require to read all the prev. records till it reaches desired record. Due to this reason the data transfer rate of this is very slow.

- Direct access: Supported by (FD/HD/CD). It is the access in which read/write number is located on the desired track no. & sector no. directly. To read the record for computers & is comparatively faster.

- Random access: Supported by all the semi-conductor memories in which the memory locations are selected randomly to provide data for the CPU/pro. In this access, mem. locations are identified by their add$^{\underline{n}}$ so that b/w 2 successive read/write operations significant amt of pro. time as well as memory time will be wasted. This is bcoz after reading data from mem the pro will req. some time to generate next addr to carry that add$^{\underline{n}}$ towards mem & to select next location of a mem. Due to this data transfer rate of random access mem is very slow. To overcome this associative access

- Associative Access: Supported by CAM (content addressable mem) which is mostly used in google search where the mem. locations are identified not by their addresses but by their content. i.e. locations are identified by the data associated with the users request & hence called Associative access.

5. Physical Types:

- Magnetic: Stores binary bit in terms of direction of magnetization of tiny particles of iron oxide.

Eg. If the magnetization is from

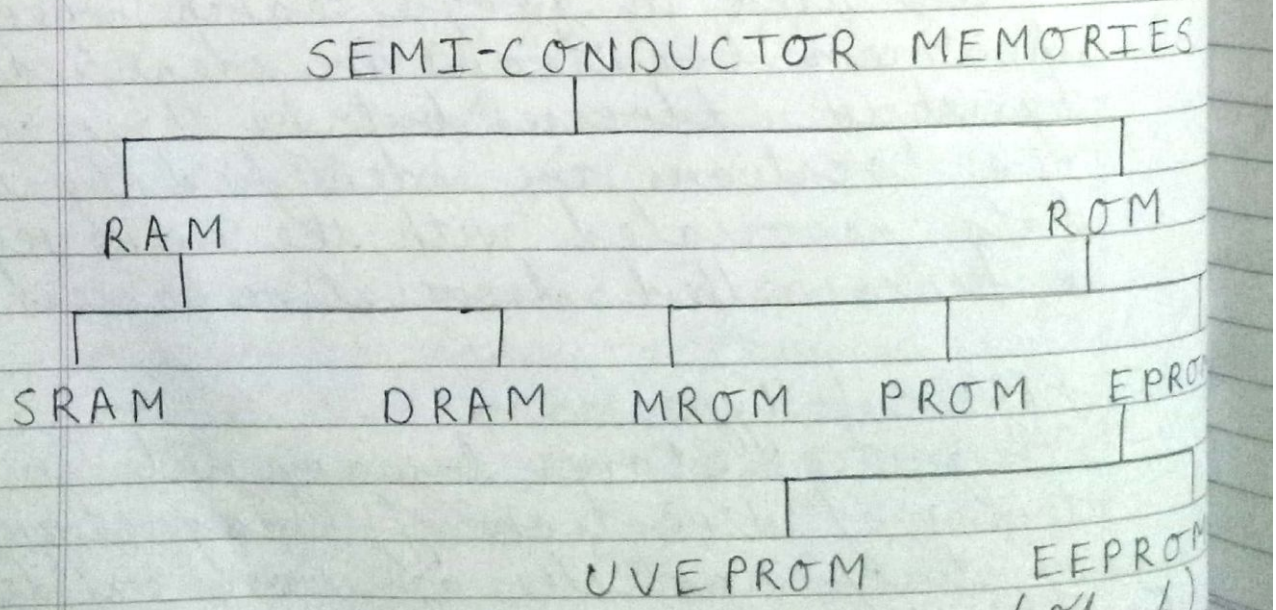
Scanned by CamScanner

south to north it represents logic 1 and vice-versa.

- Semi-conductor: It stores binary bit in terms of voltage or charge on capacitor. Eg: +5 volts represents logic '1' & 0 (zero) volts represents logic '0' similarly the capacitor with fully charged represents logic '1' & vice-versa.

- Optical Memory: It stores binary bits in terms of pits and lands which are created by the burning of the CD where pit represents logic 1 and lands represent logic 0.

6. Physical Characteristics:
i. Volatile / Non-volatile.
ii Erasable / Non-erasable.



Scanned by CamScanner

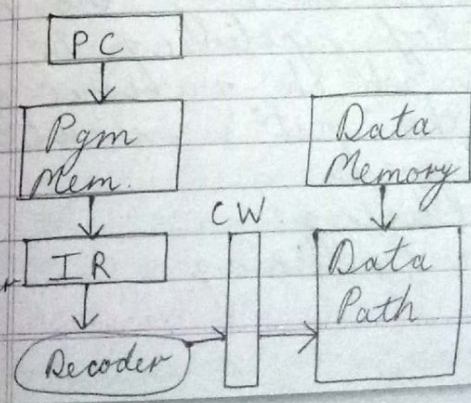
inst^{n} is known as RISC [Reduction Inst^{n} Set Computer]	
6. Due to minimal encoding of ctrl field it is easily decoded by using hard-wired CU as it is small in cost and fast in speed	6. Due to complex encoding of the ctrl field it needs to be decoded using complex μ pgm. of μ programmed CU as it is slow in speed and large in cost.

9. Differentiate b/w CISC and RISC.

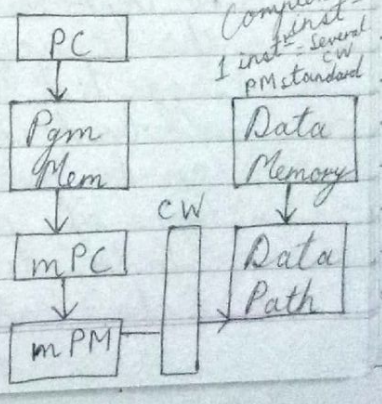
RISC	CISC
1. It uses horizontal μ instruction	1. It uses vertical μ instruction.
2. It has Hardwired CU.	2. It has micro-programmed CU
3. It is fast in speed	3. It is slow in speed.
4. It is small in cost.	4. It is large in cost.
5. It has very few addressing mode available since the most of the inst^{n} uses reg. addressing.	5. It has many addressing modes available to refer the memory since its very few inst^{n} uses reg. addressing.

6. It has multiple reg. sets available to support its reg. addressing.	6. It has only 1 reg. set and it is sufficient because the most of inst^{n} refers memory for their data operand
7. For complex addressing (if required) it needs s/w synthesization	7. The CISC architecture supports complex addressing hence does not require synthesization.
8. Its compiler designing is difficult.	8. Its compiler designing will be easier since its processor itself supports complex addressing
9. Requires single cycle for execution due to its simple inst^{n} format	9. Takes multiple cycles for execution due to its complex inst^{n} format.

10. RISC architecture:



10. CISC architecture:



Scanned by CamScanner

11. RISC processors are highly pipelined processors as its horizontal μ instⁿ. allows to express high degree of parallelism.	11. CISC processors are mostly non-pipelined processor due to complex encoding of their ctrl field.

Scanned by CamScanner

Instruction Cycle: (IC)

Def^n: Total tm req. for an inst^r execution is called IC. Which consists of fetch cycle + execute cycle.

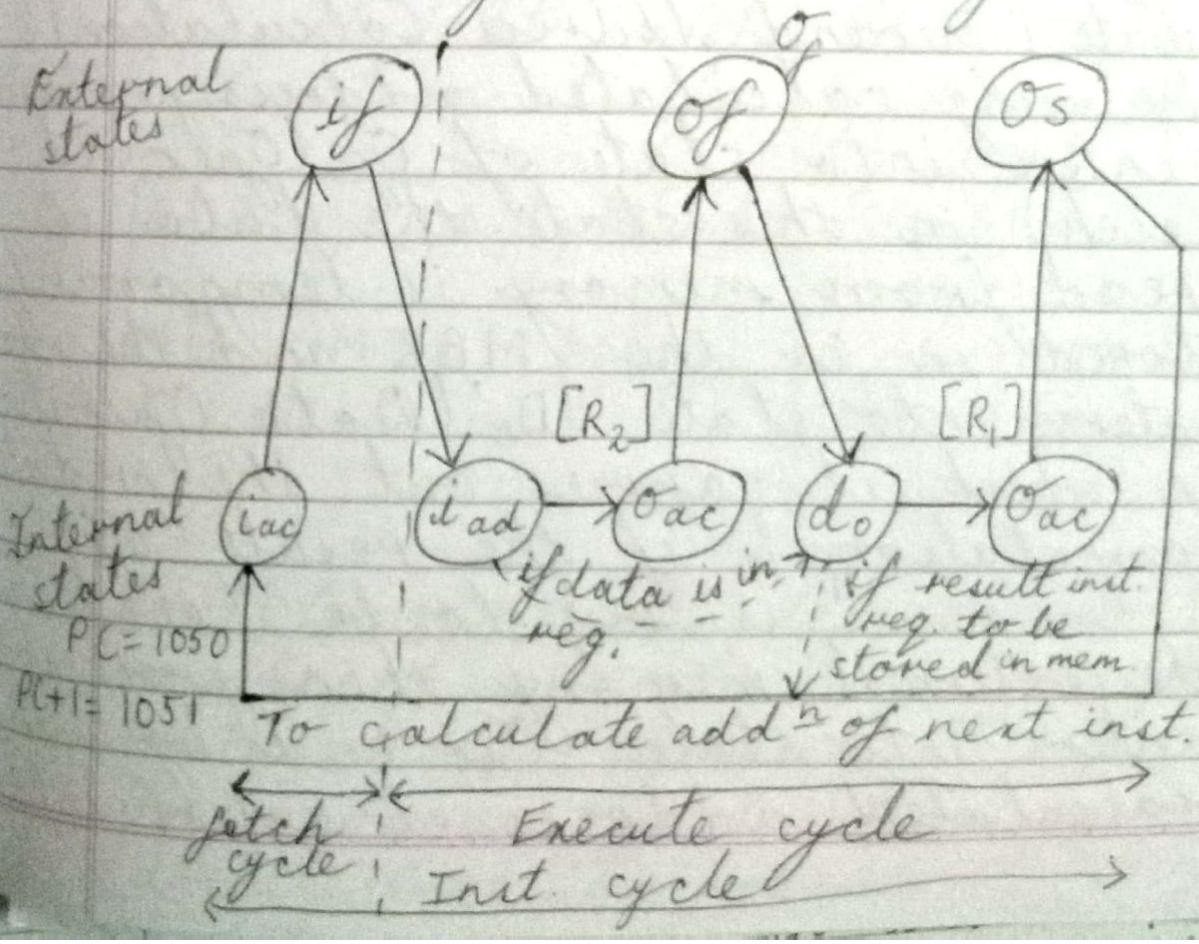
IC state diagram: It consists of 3 external states and 5 internal states as shown below; the following IC state diagram is representing various instructions:

tions.

ADD R_1, R_2; $(R_1) \leftarrow (R_1) + (R_2)$
ADD $R_1, [R_2]$; $(R_1) \leftarrow (R_1) + M[R_2]$
1050 ADD $[R_1], R_2$; $M[R_1] \leftarrow M[R_1] + (R_2)$

Q. Draw instruction cycle state diagram and explain inst^r execution in detail.

Inst. cycle state diagram



Scanned by CamScanner

To initiate execution of an inst the processor enters into state i_{ac} (instruction address calculation) in which PC (prog. counter) is prepared with appropriate address Eg PC=1050. and then processor enters into state i_f (inst. fetch). In this state the opcode of inst. is read from memory and temporarily stored in the i_R (inst. reg) and then processor enter into state i_{op} (inst operation decode) In which the processor carries out inst. interpretation to understand which computation the processor should perform. Moreover the processor also understands whether the data for this inst. is in the processor's reg or it needs to be read from the memory & if so, then processor enters into 'O_{ac}' state [Operand Address Calculation] & by using calculated address processor enters into state of O_f (Operand fetch) in this state the data is read from memory is temporarily stored in in the MBR and the processor enters into state 'D_o' (Data Operation) in which it carries out the predicted computation & if the result of " " is reg to be stored in the external memory then processor enters into state O_{ac} and by using calculated address it enters into

state O_s (Operand store) in which the result of inst computation will be stored in memory and the processor resumes back to i_{ac} where PC = PC+1 will be resumed carried out to calculate address and next inst as shown in fig.

Sequencing:

Instruction: It defines the method by which the processor can calculate address of next inst. during its program's execution the various methods of inst. sequencing are as listed below:

Method 1	Method 2
Inclusion of address of next inst. in the instruction itself	Use of PC
Eg: ADD R_1, R_2, A_3 ; $M[A_1] \leftarrow M[A_1] + M[A_2]$ A_3: Next inst	PC ← PC + K where K is length of inst. Eg: let PC → 1050 ADD R_1, R_2 PC+1 → 1051 DIV R_1, R_3 PC+2 → 1053 ADD R_1, R_3 PC+1 → 1054 Next inst.

Akash_aish_aishu

Instruction format: It defines no. of argument reg. to be specified in instruction's syntax.
The various instruction formats (architecture) are as follows:

Scanned by CamScanner