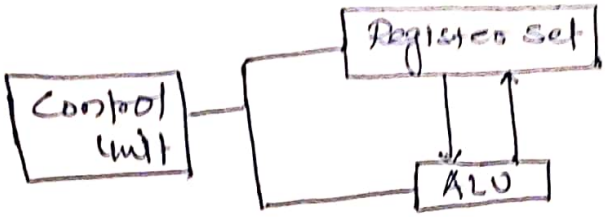
UNIT-III

Central Processing Unit

CPU - Part of computer that performs the bulk of data processing operations is called CPU.

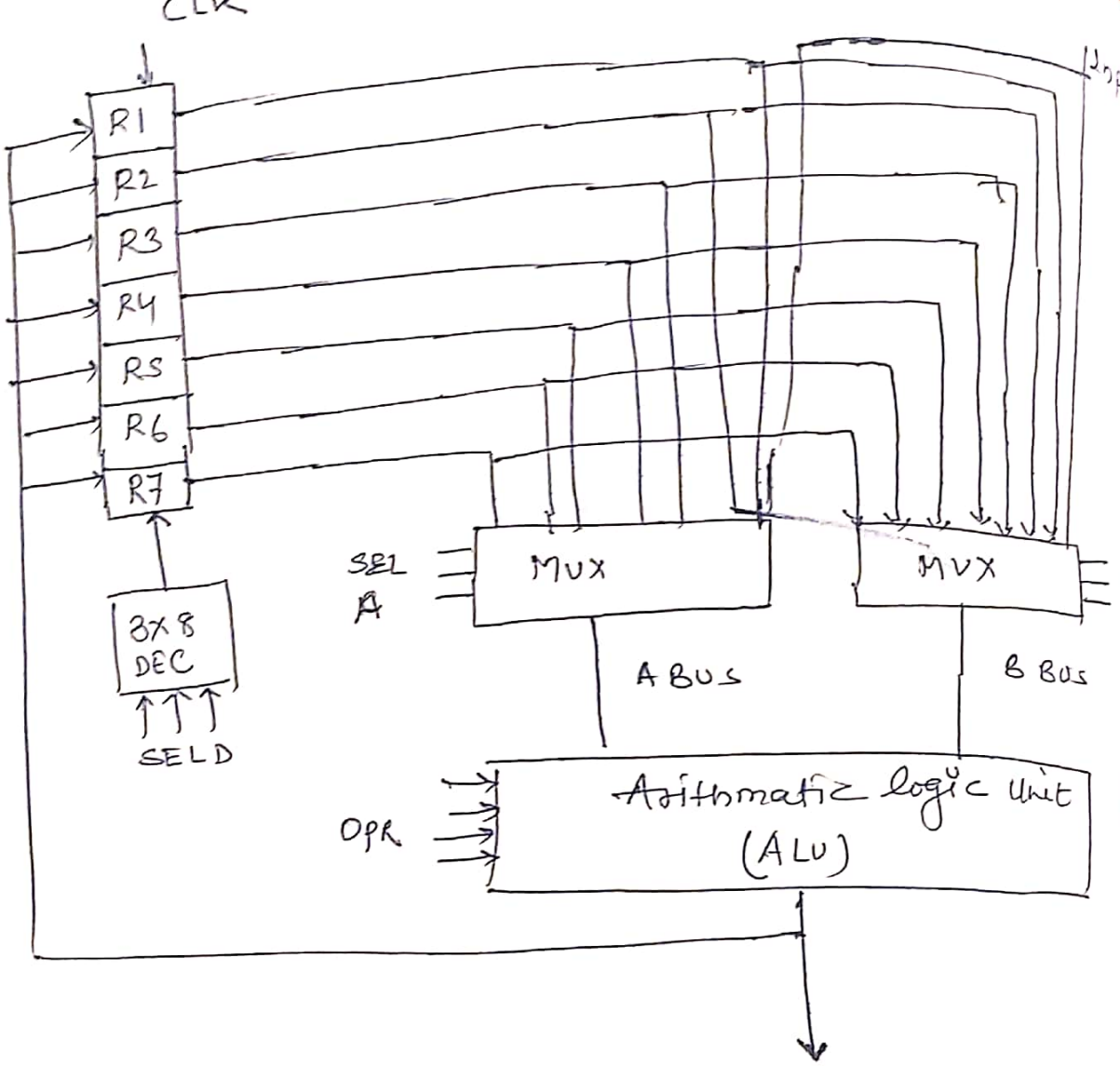
Three parts

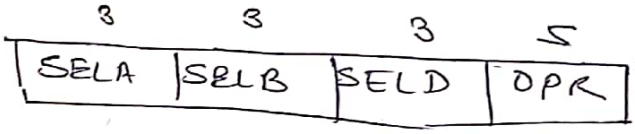
1. ALU
2. Registers
3. CU.



General Register organisation

- When large number of registers are included in the CPU. It is most efficient to connect them through a common bus system
- The Registers communicate with each other not only for direct data transfer but also while performing various micro operations.





→ Control word.

R3 + R2 → R1

1. MUX A selector - to select the content of R2 into BUS A
2. MUX B selector (SELB) to place the content of R3 into BUS B.
3. ALU operation selector: to provide the arithmatic addition
4. Decoder destination selector (SELD) to transfer the content of the output bus into R1

Control word —

Encoding of Register Selection fields.

Binary code	SELA Input	SELB Input	SELD None.
000	R1	R1	R1
001	R2	R2	R2
010	R3	R3	R3
011	R4	R4	R4
100	R5	R5	R5
101	R6	R6	R6
110	R7	R7	R7
111			

OPR Select	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A+B	ADD
00101	Subtract A-B	SUB.

R2 - R3 → R1

Control word.

STACK organization

→ STACK is LAST in first out

→ SP(stack pointer) - holds the address for the top of stack

→ Two operations of stack are Insertation & Deletion. (Push &, POP)

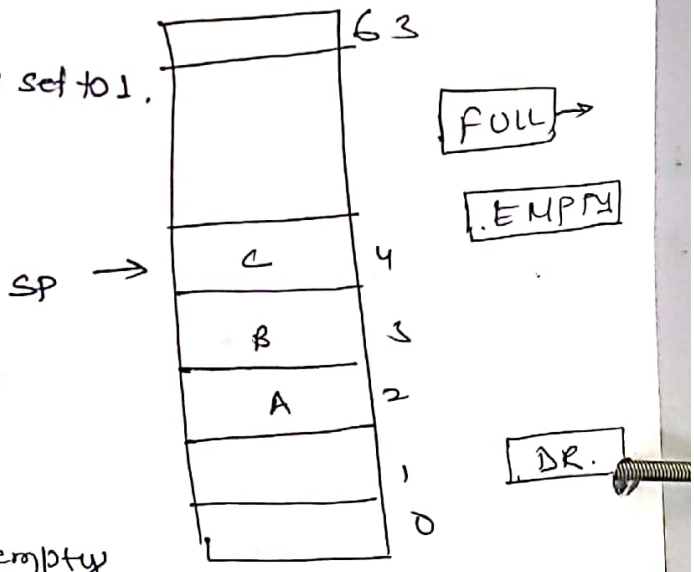
→

Register STACK

A stack can be placed in a portion of a large memory or it can be organised as a collection of finite no. of memory words. or Register.

⇒ SP cleared to 0 EMPTY set to 1.

⇒ full is cleared to zero.



SP ← SP+1

M[SP] ← DR

if (SP = 0) then (full ← 1)

Empty ← 0

Mark the stack not empty

check if stack is full

DR ← [SP]

SP ← SP-1

if (SP=0) than (Empty ← 1) → Check stack is empty

full ← 0

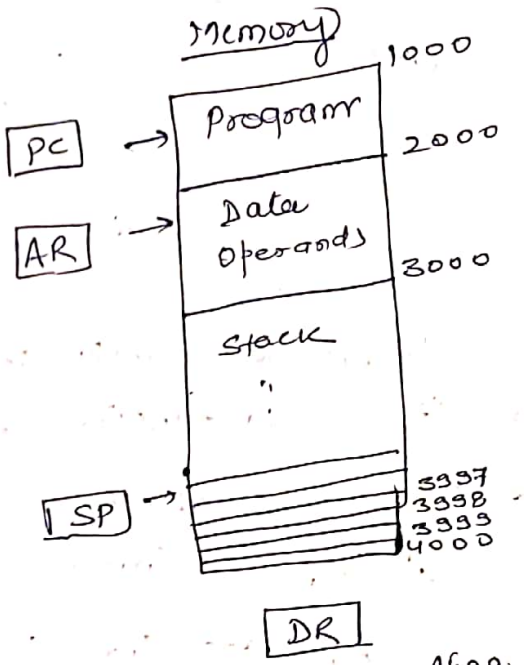
→ Mark the stack not full

MEMORY STACK

Assigning a portion of memory to stack operations and using a processor register as a stack pointer

PUSH SP ← SP-1
M[SP] ← DR

POP DR ← M[SP]
SP ← SP+1



Reverse polish Notation:→

Stack organization is very effective for evaluating arithmatic operations.

Infix notation A * B + C * D ⇒ AB * CD * +

Pre fix notation + AB

Post fix notation AB+ → RPN (reverse polish notation)

Conversion to RPN

$(A+B) * [C*(D+E)+F]$.

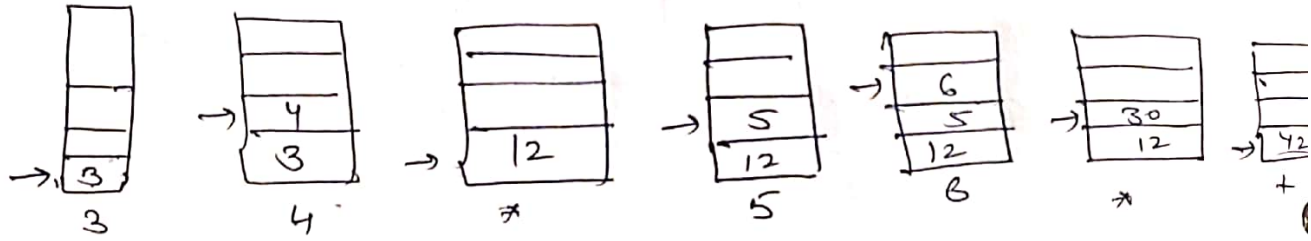
$\Rightarrow (AB+) * [C*(DE+)+F]$

$= (AB+) * [\ DE+\ C*F+]$

$= \underline{AB+\ DE+\ C*F+*}$

Evaluation

$3*4+\ 5*6.$



Instruction format :→

⇒ Computer has variety of instruction code formats.

⇒ It is the function of control unit within the CPU to interp

⇒ The bits of the instruction are divide into groups calle field. The common fields are

- An operation code field that specifies the operati to be performed
- An address field that designates a memory Address or a Processor Register
- A Mode field that specified the way the operand or the effective address is determine

Computers may have instruction of several different lengths containing varing number of addresses. The number of address fields in the instruction format of a computer depends on the internal organisation of it's registers.

① Single Accumulator organisation
Add X

② general Register organisation
ADD R1, R2, R3.

③ STACK organisation
PUSH X

Three Address Instruction

```
ADD R1, A, B
ADD R2, C, D
MUL X, R1, R2
```

Two Address Instruction

```
MOV R1, A
ADD R1, B
MOV R2, C
```

One Address Instruction

```
LOAD A
ADD B
STORE T
```

Zero Address

```
PUSH A
PUSH B
```

Addressing Mode :-

The way of choosing operands during program execution is dependent on the addressing mode.

The Control Unit of a Computer is designed to go through an instruction cycle that is divided into three major phases.

1. fetch the instruction from memory
2. Decode the instruction
3. execute the instruction.

① Implied Mode

Ex CMA

Operands are specified implicitly in the definition of instruction.

② Immediate Add. Mode

In this operands are actual data rather than address field.

Ex MVI A, 80H. A ← 80H

③ Register mode : Operands are register within CPU

Add R1, R2 R1 + R2 → R1

④ Direct Add Mode : The operand resides in memory and it's address is given directly by the address field of the instruction.

Ex LXI B, 2000H

⑤ Indirect Add Mode - In this mode the address field of the instruction gives the Address where effective address is stored in the Memory

Effective Add = add Part of Instruction + Content of CPU Register.

⑥ Relative Address Mode :→
In this mode the content of the Program Counter is added to the address part of the instruction in order to obtain the effective address.

⑦ Indexed Addressing Mode
Content of the index register is added to the Address of the instruction to obtain the effective Address.

⑧ Base Register Addressing Mode :→
Content of Base Register is Added to the address Part of the instruction to obtain the effective address

⑨ Auto Increment / Decrement
This is similar to the register in direct mode except that the register is incremented or decremented after it's value is used to Access memory.

Data

Data Transfer & Manupulation

Computer provides an extensive set of instructions to give the user the flexibility to carry out various computational task.

classified into three categories

(1) Data Transfer (2) Data manupulation

(3) Program Control instruction.

(1) Data Transfer

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCHG
Input	IN
Output	OUT
Push	PUSH
Pop	POP

Mode	Assembly Conversion	Register Transfer
Direct	LD ADR	AC ← M[ADR]
Indirect	LD @ADR	AC ← M[M[ADR]]
Relative	LD $ ADR	AC ← M[PC+ADR]
Immediate	LD # NBR	AC ← NBR
Register	LD R1	AC ← R1
Register Indirect	LD (R1)	AC ← M[R1]
Auto Increment	LD (R1)+	AC ← M[R1], R1 ← R1+1

Data Manupulation Instructions —

(1) Arithmatic Instruction

(2) Logical & bit manipulation instructions.

(3) Shift Instructions.

(1)

Arithmatic Instruction

Increment	INC
Decrement	DEC
Add	ADD
Substract	SUB
multiply	MUL
Divide	DIV
Add with carry	ADDC
substract with borrow	SUBB
Negate (2's Complement	NEG.

① Logical & Bit manupulation

clear	– CLR
Complement	– COM
AND	– AND
OR	– OR
Exclusive-OR	– X-OR
clear carry	– CLRC
set carry	– SETC
Complement Carry	– COMC

Enable Interrupt	EI
Disable Interrupt	DI

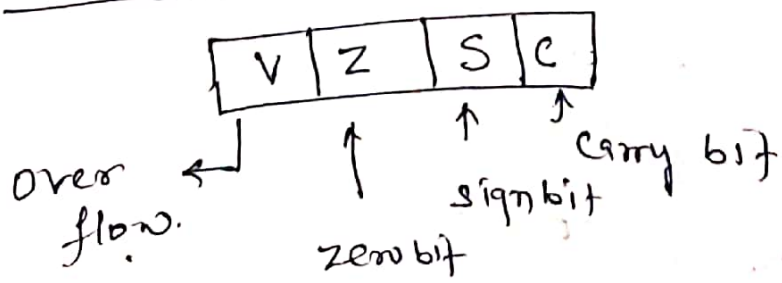
③ Shift operations:

Logical shift Right	SHR
Logical shift Left	SHL
Arithmatic shift right	SHRA
Arithmatic shift left	SHLA
Rotate Right	ROR
Rotate Left	ROL
Rotate Right Through Carry	RORC
Rotate Left Through Carry	ROLC

: Program control

Branch	— BR
Jump	— JMP
skip	— SKP
call	— CALL
Return	— RET
Compare	— CMP
Test (by ANDing)	— TST

Status Bits



Conditional Branch Instruction

BZ	Z = 1
BNZ	Z = 0
BC	C = 1
BNC	C = 0
BP	S = 0
BM	S = 1
BV	V = 1
BNV	V = 0

Un signed	Compare	Condition
BHI	A > B	higher.
BHE	A ≥ B	higher or equal
BLO	A < B	lower than
BLOE	A ≤ B	lower or equal
BE	A = B	equal
BNE	A ≠ B	not equal

Signed	Compare Condition
BGT	grather than
BGE	grater or equal
BLT	less than
BLE	less or equal
BE	equal
BNE	not equal,

SUB routine

1. Call
2. Return.

SP ← SP − 1

M[SP] ← PC

PC ← effective Address.

Interrupts -

Program interrupt refers to the transfer of program control from a currently running program to another service program as a result of external or internal generated request

⇒ PSW: collection of all status bits in the CPU is sometimes called a program word or PSW.

Types

① External interrupt - from timing device, input.output device, from power supply or from any other external source.

② Internal interrupts. arise from illegal or erroneous use of an instruction or data Internal ~~not~~ interrupts are also called traps.

The difference b/w internal & external interrupt is that the internal interrupt is initiated by some exceptional condition caused by the program itself rather than by an external event.

③ Software interrupt: it is a special call instruction that behaves like an interrupt rather than a subroutine call. It can be used by the programmer to initiate an interrupt procedure at any desired point in the program.

& Supervisor call instruction.

Reduced Instruction set Computer

The concept of Risc architecture invoks an attempt to reduce execution time by simplying the instruction set of the computer. The major characteristics of RISC processor are.

1. Relatively few instructions
2. Few Addressing Modes
3. Memory Access limited to Load & Store instr.
4. All operations are done withinthe registers of the CPU
5. fixed Length, easily decoded instruction format
6. Single cycle instruction execution
7. Hardware rather than microprogrammed control.

Complex Instruction set computer

(1) Large number of instructions - typically from 100 to 250 instructions.
(2) Some instructions that perform specialised task and are used frequently
(3) A large variety of Addressing modes.
(4) variable Length instruction format
(5) Instruction set that manipulates operands in memory.

Pipe line and Vector processing

Parallel Processing is a term used to denote a large class of techniques that are used to provide simultaneous data processing tasks for the purpose of increasing the computational speed of a computer system.

→ The purpose of PP is to speed up the computing process and increase in throughput that is the amount of processing that can be accomplished during a given interval of time.

M. J. flynn classification –

The normal operation of a computer is to fetch instructions from the memory & execute them in processor. The sequence of instruction read from memory constitutes an instruction stream. The operations performed on the data in the processor constitutes an data stream. Parallel processing may occure in instruction stream, Data stream or both.

1. Single Instruction, Single data Stream (SISD)
2. Single Instruction, multiple data Stream (SIMD)
3. multiple instruction, multiple data Stream (MIMD)
4. multiple instruction, single data Stream (MISD)

SISD – single computer containing a control unit, a processor unit and a memory unit.
- Parallel processing is achieved by pipelining.

SIMD – many processors under the supervision of a common control unit
- All processor receive same instruction but different data items.

MIMD – organisation refers to a computer system capable of processing several programs at the same time.

MISD – No Pratical system only Theoritical Concept

Three Types of Parallel Processing
1. Pipe line processing
2. Vector processing
3. Array processors.

Pipelining —

Pipelining is a Technique of decomposing a sequential process into suboperations with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments.

$A_i * B_i + C_i$ for : $i = 1, 2, 3 \ldots 7$

Segment 1 $R1 \leftarrow A_i$ $R2 \leftarrow B_i$ Input A_i & B_i

Segment 2 $R3 \leftarrow R1 * R2$, $R4 \leftarrow C_i$ multiply & input C_i

segment 3 $R5 \leftarrow R3 + R4$ Add C_i to product

Clock Pulse	Segment 1 R1	R2	Segment 2 R3	R4	Segment 3
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 B_2$	C_2	$A_1 * B_1 + C_1$

The speed up ratio due to pipelining is

$$S = \frac{n t_n}{(K+n-1) t_p}$$

$n t_n$ → total time required for n tasks to complete n tasks with K pipeline is $(K+n-1) t_K$

$t_n = K t_p$

if n is large

$$S = \frac{t_n}{t_p}$$

$$S = \frac{K t_p}{t_p}$$

$$\boxed{S = K}$$

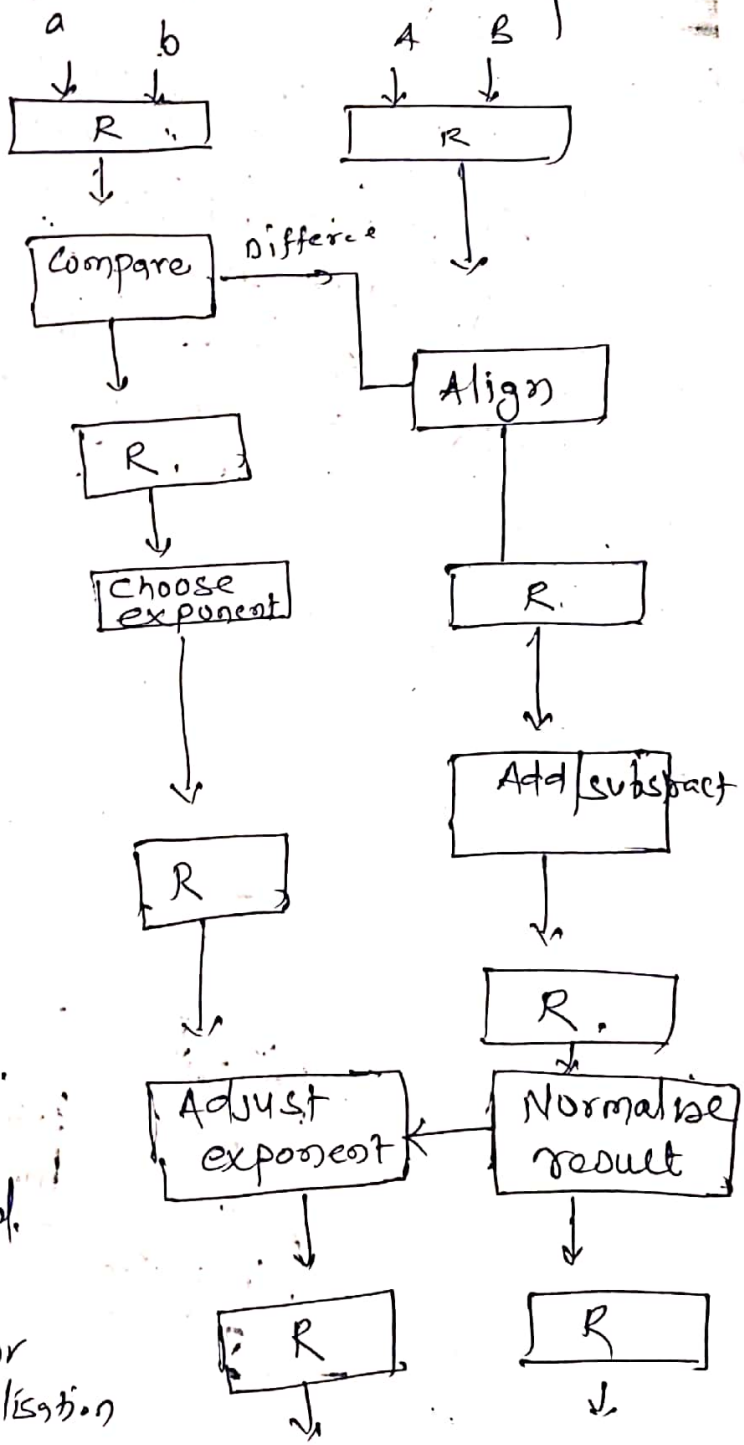
Arithmatic pipe line ⇒

- used in very high speed computers for Arithmatic operations of floating point.

$X = A \times 2^a$

$Y = B \times 2^b$

1. Compare exponent
2. Align the Mantissa.
3. Add or Substract the Mantissa.
4. Normalise the Result



$X = 0.9504 \times 10^3$

$Y = 0.8200 \times 10^2$ 3-2 ⇒ 1

$X = 0.9504 \times 10^3$ → Align.

$= 0.0820 \times 10^3$

$Z = 1.0324 \times 10^3$ Add.

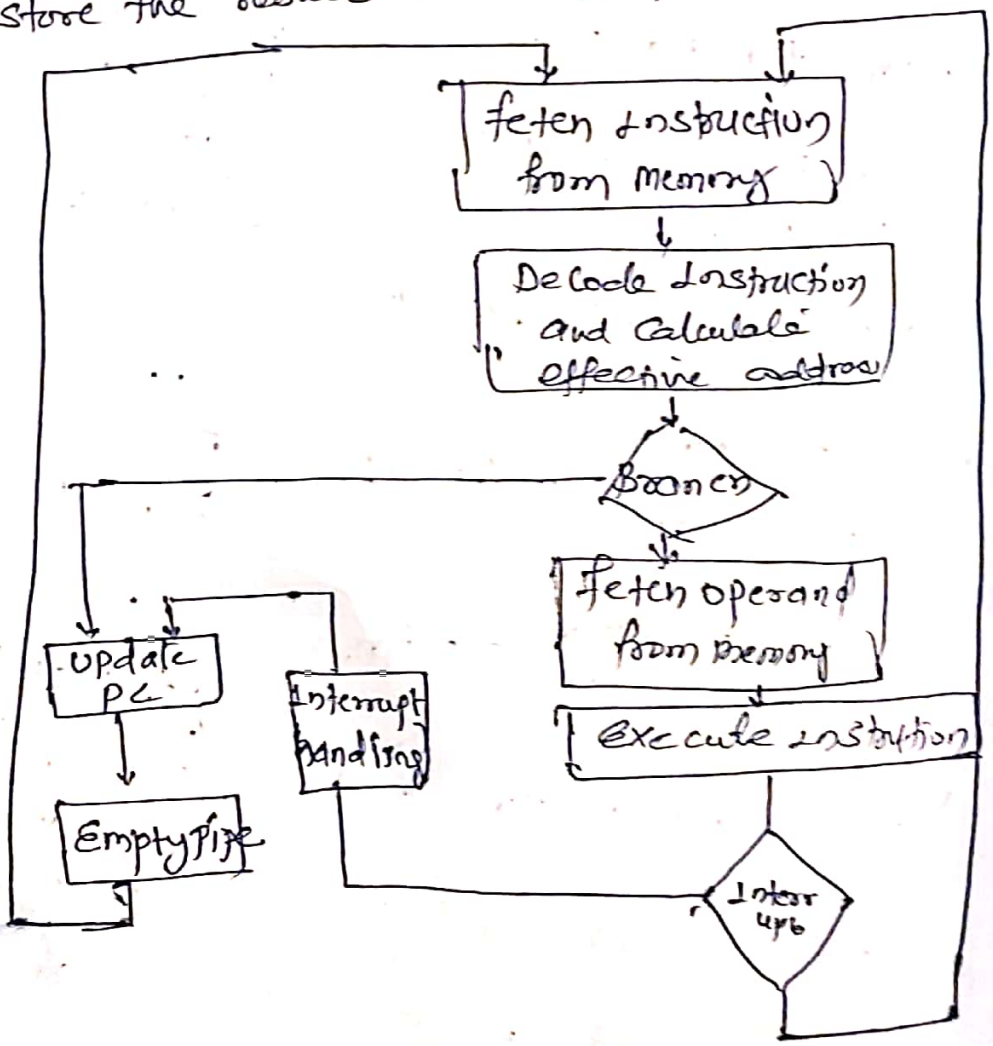
$Z = 0.10324 \times 10^4$ Normalisation

Instruction pipe line

- Pipeline processing can occur in Instruction Stream aswell
- An instruction pipeline reads consecutive instruction from memory while previous instructions are being executed in other segment.
- A CISC require other than feten & Decode process of a instruction completely

— Some are

1. feten the instruction from memory
2. Decode the instruction
3. calculate the effective address
4. fetch the operands from memory
5. Executive the instruction
6. store the result in the proper place.



RISC Pipeline

RISC is it's ability to use an efficient pipelining because of fixed Length Instruction format, the decoding of operation can occure at the same time as the register selection.

- All data Manupulation Instruction have the register to register operation, so need to calculate effective address or fetching of operand from memory.
- Therefore Two or Three segment Instruction Pipeline can be used.
- One segment fetches the Instruction from memory, Second executes the Instruction in ALU and third May be store of result into memory

Three segment Instruction Pipeline

I - Instruction feter
A - ALU operation
E - Execute Instruction

1. LOAD R1 ← M[Add1]
2. LOAD R2 ← M[Add2]
3. ADD R3 ← R1+R2
4. STORE M[Add3] ← R3.

	1	2	3	4	5	6
	I	A	E			
		I	A	E		
			I	A	E	
				I	A	E

PAGE NO.11

FI → Fetch instruction
DA → Decode and Calculate effective address
FO → Fetch the operand
EX → Execute the instruction.

Steps	1	2	3	4	5	6	7	8	9	10	11	12
1	FI	DA	FO	EX								
2		FI	DA	FO	EX							
3.			FI	DA	FO	EX						
4.				FI			FI	DA	FO	EX		
								FI	DA	FO	EX	
									FI	DA	FO	EX

In general three Major difficulties in instruction pipeline

1. Resource conflicts.
2. Data dependency
3. Branch difficulties.

Vector Processing: —

- Many scientific problem require arithmatic operations on large arrays of numbers.
- These numbers are usualy formulated as vectors and matrics of floating point numbers.

Ex — Initilise I = 0

Level: Read A(I)
Read B(I)
Store C(I) = A(I) + B(I)
Increment I = I + 1
if I ≤ 100 goto 'Level'
Continue.

- A computer capable of vector processing eliminates the overhead associated with the time it takes to fetch & executes the instruction in the Program loop.
- To constitutes a program loop that reads a pair of operands form arrays A & B and performs a floating Point operation / Addition. The loop control variable is then updated and the step repeat 100 times.

$$C(1:100) = A(1:1W) + B(1:1W)$$

Matrix Multiplication

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} * \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ c_{32} & c_{32} & c_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

$$C_{ij} = \sum_{k=1}^{3} a_{ik} \times b_{kj}$$

Applications

- Long range weather fore casting
- Petroleum explorations
- Seismic data Analysis
- Medical diagonosis
- Aerodynamics and space flight Simulations.
- Artificial Intelligence & Expert system
- mapping the human genome
- Image Processing

Array Processors –

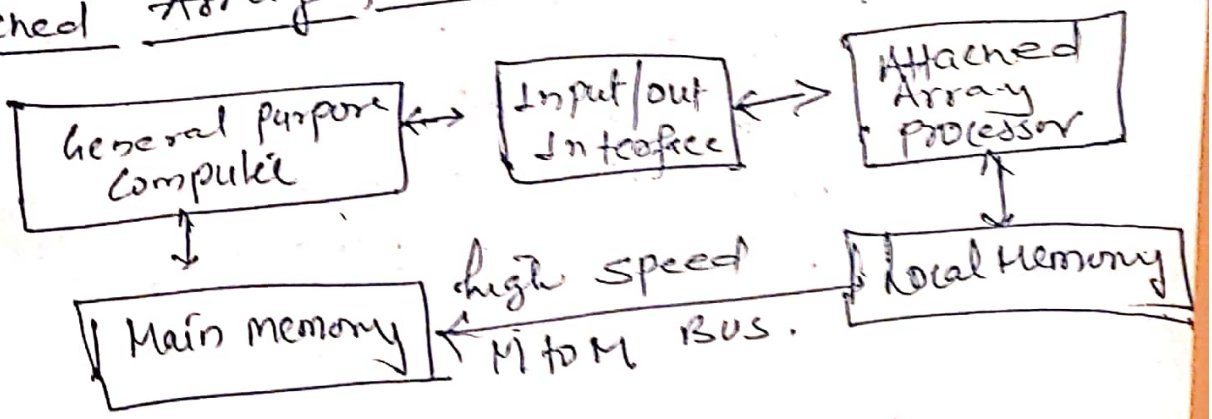
PAGE NO

– Performo Computation of large array of data

Types

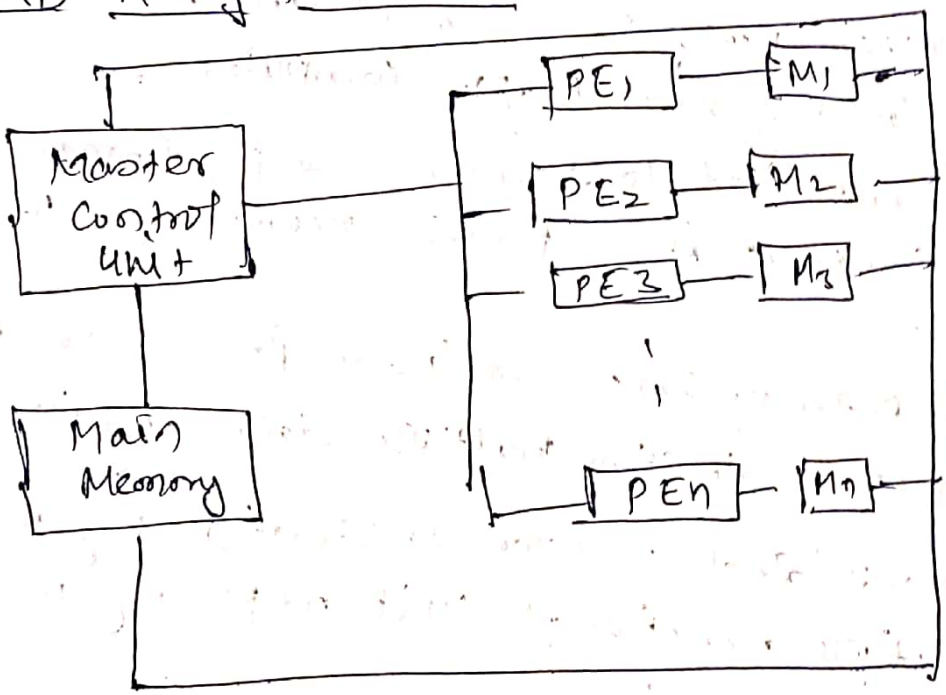
- An Attached Array processor is an Auxiliary proces attached to a general purpose computers.
- It is intended to improve the performance of host Computer in specific numerical computation tasks.
- An SIMD array processor is a processor that has a single instruction multiple data organization
- It Manipulated vector instruction by means of multiple functional unit responding to a common instruction.

Attached Array processor



- The host computer is general purpose computer and attached processor is a back end machine driven by the host computer
- The array processor is attached through an input/OP controller to the computer and treat it like an external interface.
- The data from main memory to local memory is transferred through high speed bus for attached processor.

SIMD Array Processor



The processing units are synchronized to perform the same operation under the control of a control unit thus providing a single instruction stream, multiple data stream organization.

PE → Processing elements

M → local Memory

END