UNIT-V

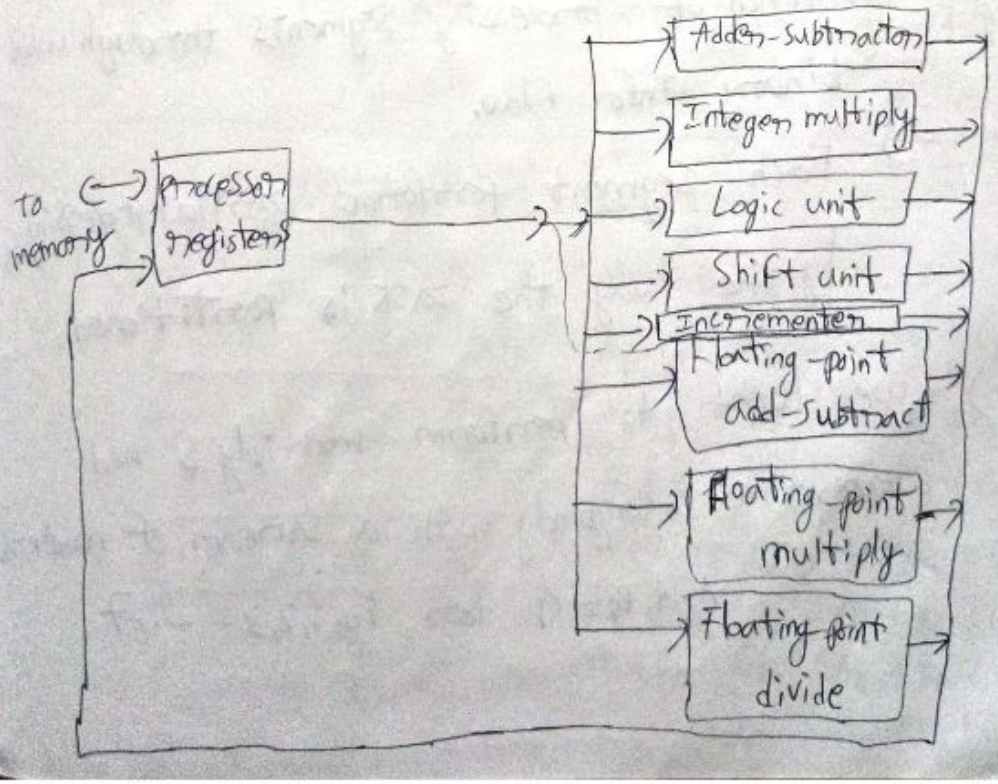
Pipeline & Vector processing multiprocessors, multi computers

1) parallel processing:-

→ It 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.

→ Instruction is being executed in the ALU, next instruction can be read from memory. The system may have 2/more ALU's & be able to execute 2/more instructions at the same time.

→ Further more, the system may have 2/more processors operating concurrently.



→ Single Instruction stream & single data stream (SISD)

→ single Instruction stream & multiple data stream (SIMD)

→ multiple Instruction streams & single data stream (MISD)

→ multiple Instruction stream & multiple data stream (MIMD)

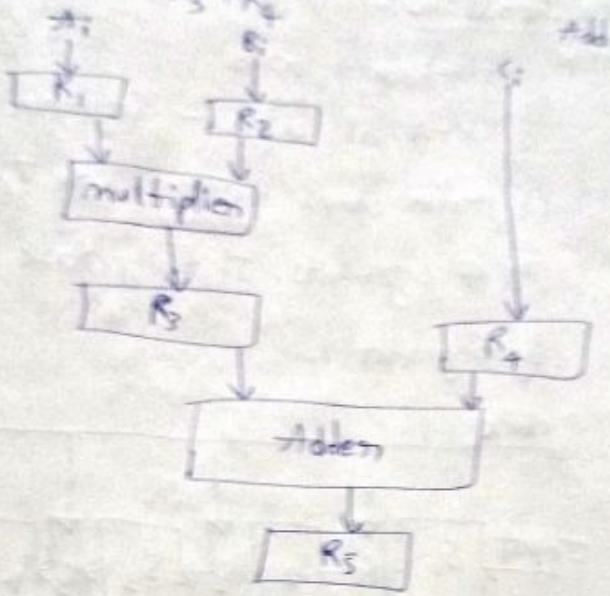
⇒ pipelining:-

→ A technique of decomposing a sequential process into sub operations, with each sub process being executed in a special dedicated segment that operates concurrently with all other segments. A pipeline can be visualized as a collection of processing segments through which binary Info flow.

→ Each segment performs partial processing dictated by the way the task is partitioned.

→ we want to perform multiply & add operations combinely with a stream of numbers

$A_i * B_i + C_i$ for $i = 1, 2, 3 - - - , 7$



Example of pipeline processing

Clock pulse number	Segment 1		Segment 2		Segment 3
	R_1	R_2	R_3	R_4	R_5
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	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9	—	—	—	—	$A_7 * B_7 + C_7$

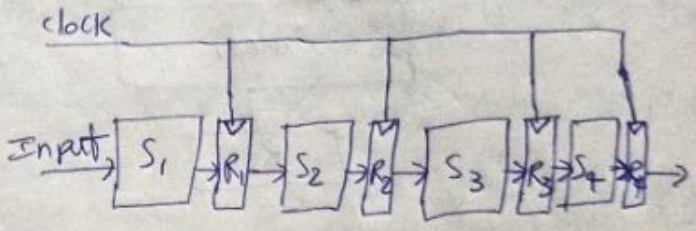
Contents of Registers in pipeline Example

⇒ General considerations:-

Horizontal axis — Clock cycles,

Vertical axis — Segment number,

Diagram shows tasks T_1 to T_6 executed in 4 segments. Initially, task T_1 is handled by Segment 1.



Four-segment pipeline

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

Space-time diagram for pipeline

Segment	1	2	3	4	5	6	7	8	9 → Clock cycle
1	T_1	T_2	T_3	T_4	T_5	T_6			
2		T_1	T_2	T_3	T_4	T_5	T_6		
3			T_1	T_2	T_3	T_4	T_5	T_6	
4				T_1	T_2	T_3	T_4	T_5	T_6

As no. of tasks ↑, n becomes much larger than K-1, & K+n-1 approaches the value of n

ine Example

k cycles,
nt number,
T_6 executed in
t_i is handled

line

s_cycle

ger
n.

Under this condition, the speed up becomes

$$S = \frac{(k+n-1)t_n}{(k+n-1)t_p} = \frac{t_n}{t_p} \Big/ S = \frac{t_n}{t_p}$$

If we assume that time it takes to process a task is the same in the pipeline & non-pipeline circuits. we will have $t_n = kt_p$. Including this assumption, the speed up reduces to

$$S = \frac{kt_p}{t_p} = k.$$

*) Instruction pipeline:-

→ An instruction pipeline reads consecutive instruction read from memory while previous instructions are being executed in other segments.

→ It causes the instruction fetch & Execute phases to overlap & perform simultaneous operations.

To process each instruction with the following Sequence of steps:-

a) Fetch the instruction from memory.

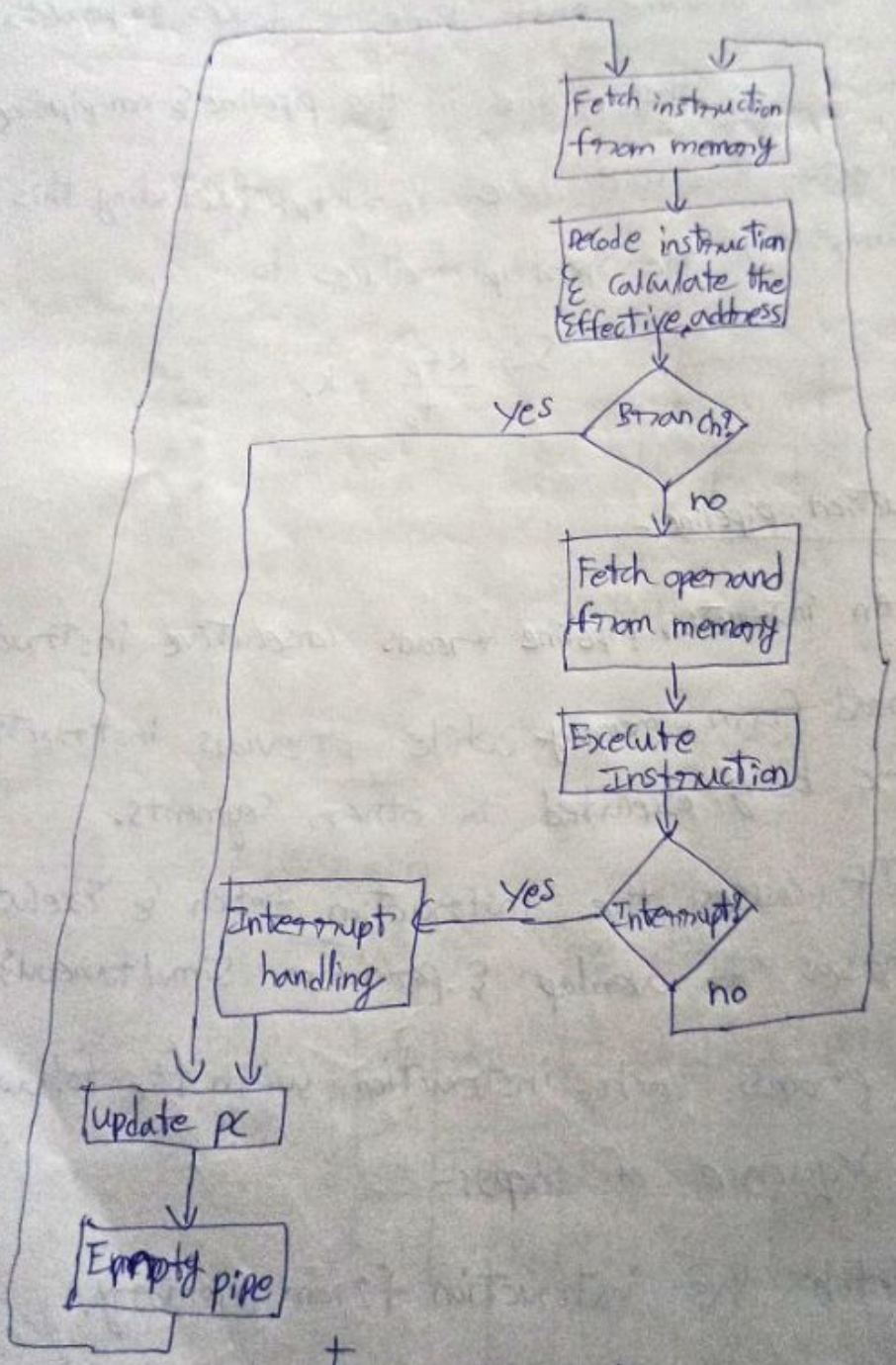
b) Decode the instruction.

c) Calculate the effective address.

d) Fetch the operands (more) from memory.

e) Execute the Instruction.

f) Store the result in the proper place.



Four-Segment CPU pipeline

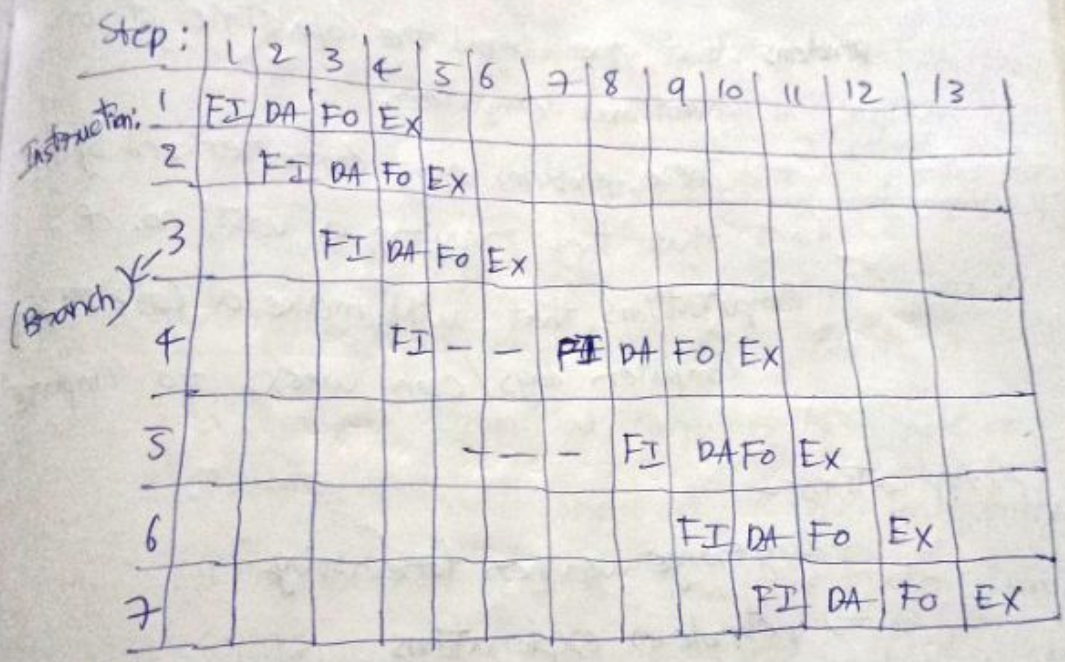
Segments

a) FI → fetches an Instruction.

b) DA → Decodes the Instruction & calculates the effective Address.

c) Fo → fetches the operand.

d) EX → Executes the Instruction.



we can't fetch any other Instruction until branch is executed.

(True) | If 3 - Branch then Fo & FI can't done at same time in steps.

Another delay may occur in pipeline, if EX Segment needs to store the result of operation in data memory while Fo segment needs to

Fetch an operand.

→ In that case, Fo must wait, until Ex finished it's operation.

=> Vector processing:-

→ There is a class of computational problems that are beyond the capabilities of a conventional computer.

→ These problems are characterized by the fact that they require a vast no. of computations that will make a conventional computer days/even weeks to compute.

=> Applications:-

→ Long range weather forecasting
→ petroleum explorations
→ Seismic data analysis
→ medical diagnosis
→ Aerodynamics & Space fight Simulations.
→ AI & expert Systems.
→ mapping the human genome
→ Image processing

=> Vector operat

→ many scie
on large
usually
floatin

→ A ve
arra

→ I

⇒ vector operations

→ many scientific problems require arithmetic operations on large arrays of numbers. These numbers are usually formulated as vectors & matrices of floating-point numbers.

→ A vector is an ordered set of a one-dimensional array of data items. A vector V of length n is represented as a row vector by $V = [V_1\ V_2\ V_3 \cdots V_n]$.

→ It may be represented as a column vector if the data items are listed in a column. A conventional sequential computer is capable of processing operands one at a time. Consequently, operations on vectors must be broken down into single computations with subscripted variables.

```
      DO (20) I = 1, 100
(20)  C(I) = B(I) + A(I)
```

This is a program for adding 2 vectors A & B of length 100 to produce a vector C. This is implemented in machine language by the following sequence of operations.

Initialize I = 0

20 (20) Read A(I)
Read B(I)
Store C(I) = B(I) + A(I)
Increment I = I + 1
If I ≤ loop go to (20) 20
Continue

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

⇒ matrix multiplication:-

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

⇒ mem

The product C is 3×3, elements (A & B) by the inner product.

$$C_{ij} = \sum_{i=1}^{3} a_{ik} * b_{kj}$$

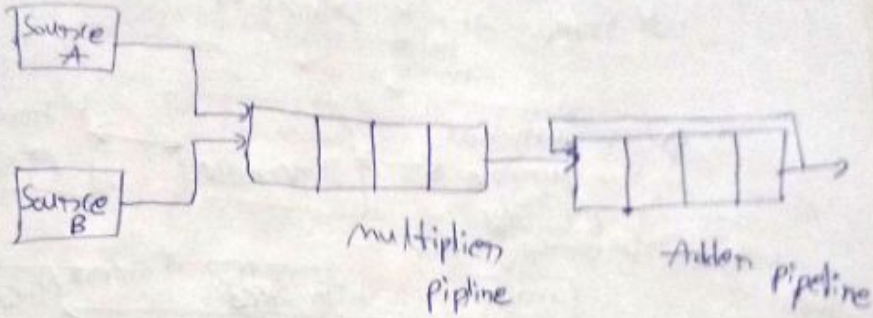
For 'x':- no. in the first row & first colum of matrix C is calculated by letting i=1, j=1, to obtain

$$C_{11} = a_{11}b_1 + a_{12}b_{21} + a_{13}b_{31}$$

Operation Code	Base address Source 1	Base address Source 2	Base address destination	Vector length

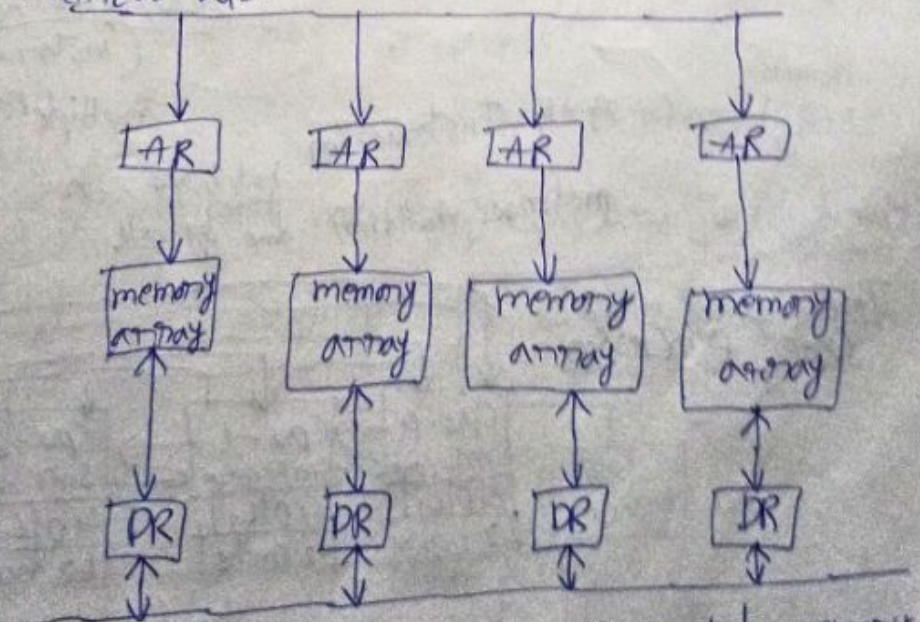
Instruction format for vector processor

$C = A_1B_1 + A_5B_5 + A_9B_9 + A_{13}B_{13} +$
$+ A_2B_2 + A_6B_6 + A_{10}B_{10} + A_{14}B_{14} +$
$+ A_3B_3 + A_7B_7 + A_{11}B_{11} + A_{15}B_{15} +$
$+ A_4B_4 + A_8B_8 + A_{12}B_{12} + A_{16}B_{16} + \ldots$



⇒ memory Interleaving:-

→ Instead of using 2 memory buses for simultaneous access, memory can be partitioned into a no. of modules connected to a common memory address & data buses.



multiple module memory organisation

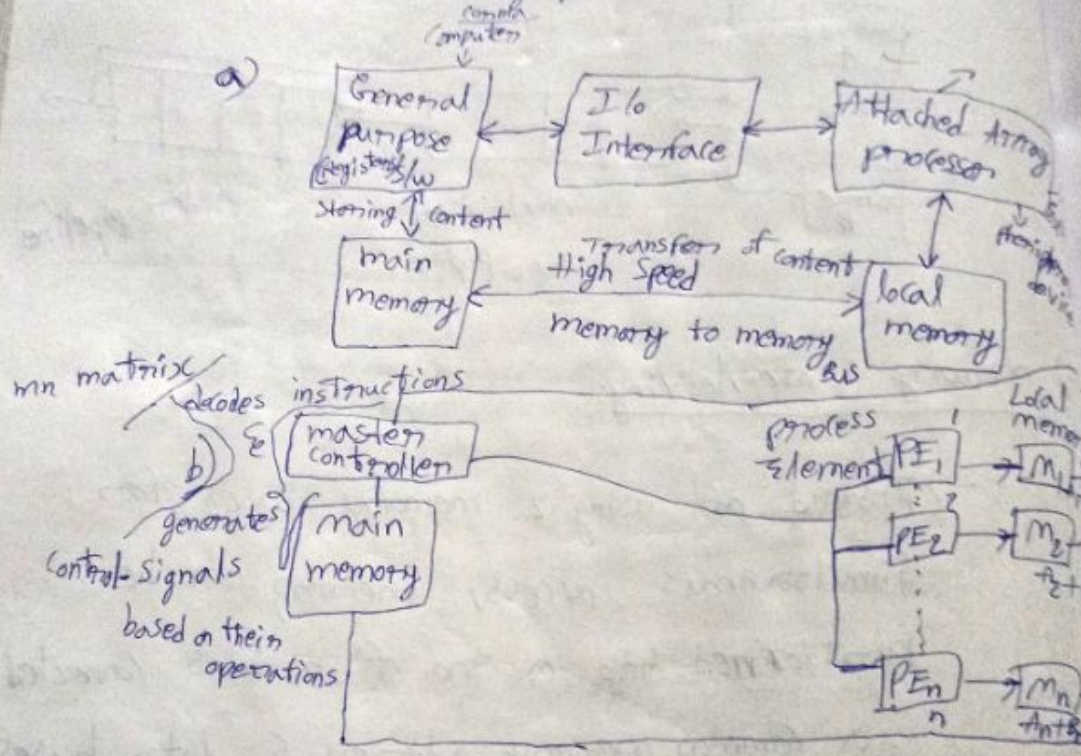
⇒ Array processor:-

Single processor

To improve, desired performance of the computer.

a) Attached Array processor
b) SIMD Array processor

a)



b)

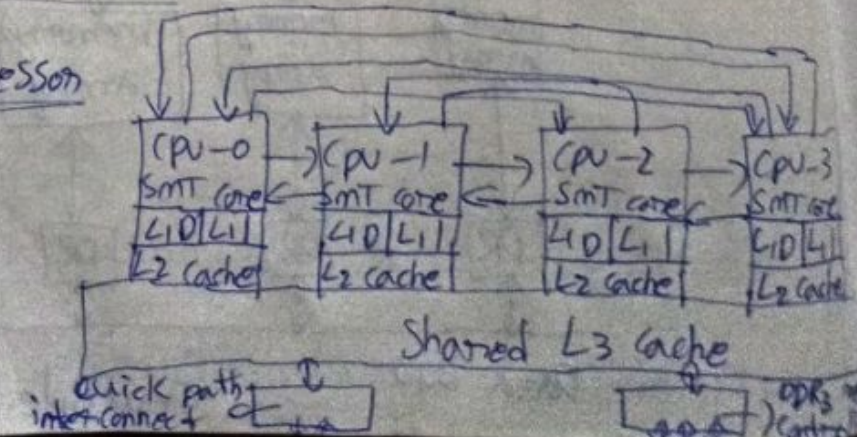
Registers
ALU, Pipeline
(instructions)
multiple data

$C = A_i + B_i$ Instruction

$i = 1, 2, 3, 4 \ldots n$

(Matrix Addition) done by all.

Intel i7 processor



characteristics of multiprocessors:-

Inter connection of 2/more CPUs with memory and input-output equipment.

Tightly coupled, Loosely coupled

Shared memory / Distributed memory

MIMD → multiple Instruction multiple Data stream.

computation can proceed in parallel in one of 2 ways.

→ multiple Independent Jobs can be made to operate in parallel.

→ A Single Job can be partitioned into multiple parallel tasks.

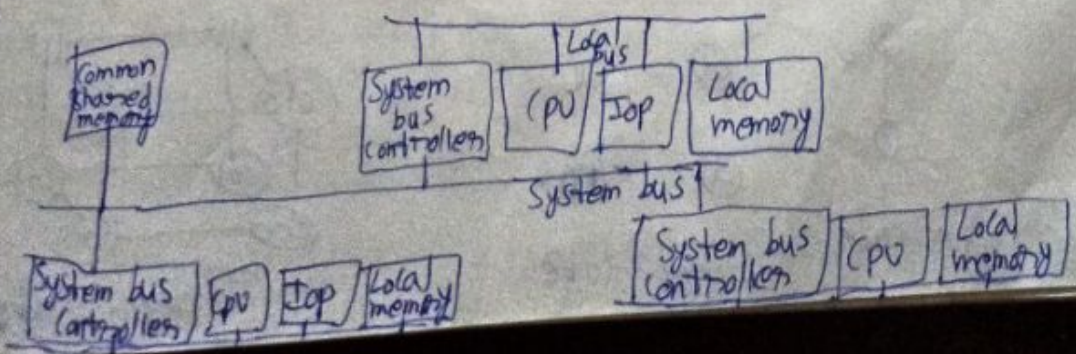
Inter connection Structure:-

It has components;-

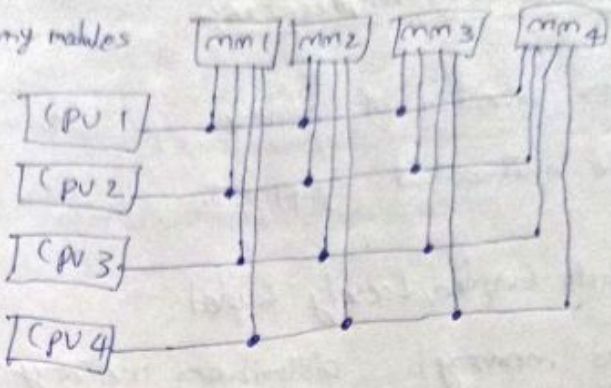
a) Time shared common Bus/

b) multiport memory / c) Cross bar Switch / d) multistage Switching network / e) HyperCube System time Shared common bus.

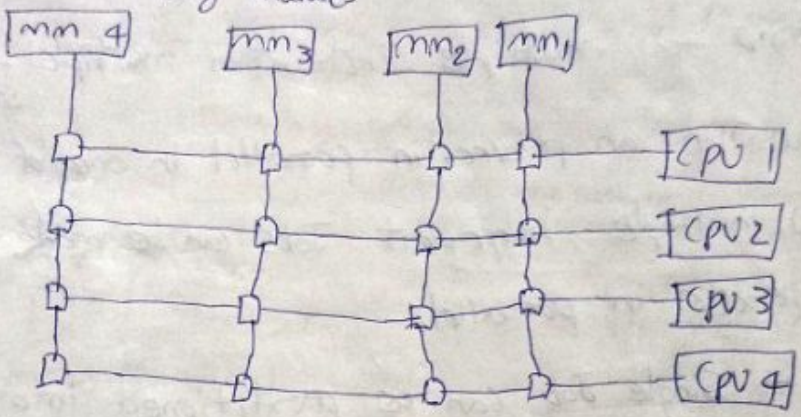
a)



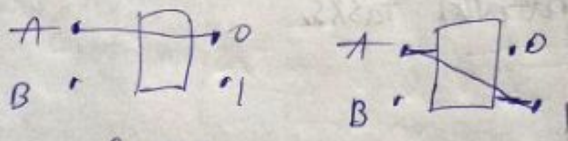
b)



c)



d)

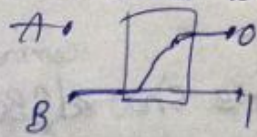


$C_A = 0$

A Connected to 0

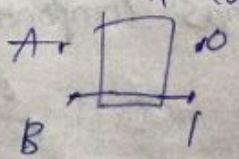
$C_A = 1$

A Connected to 1



$C_B = 0$

B Connected to 0

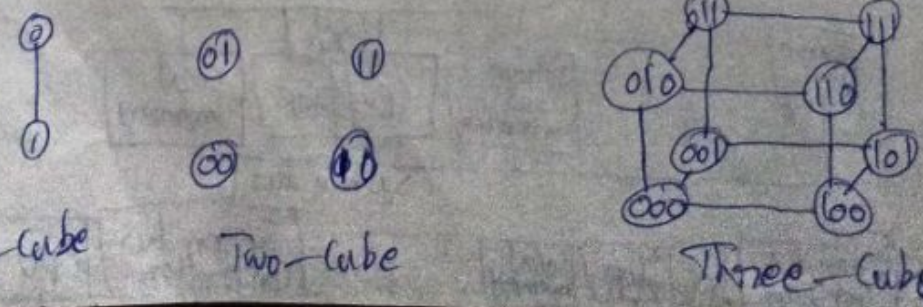


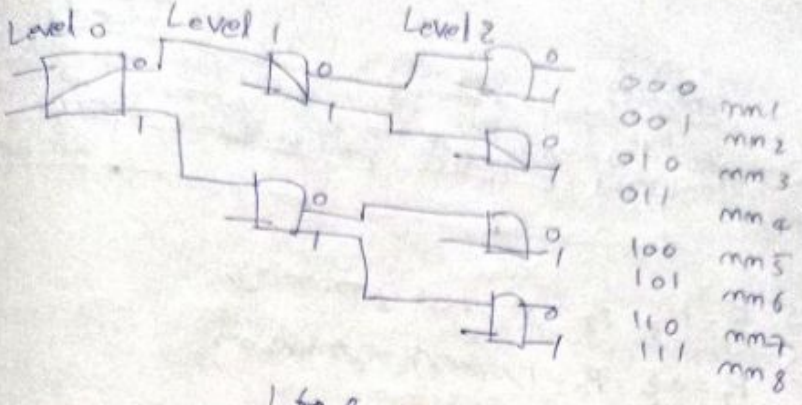
$C_B = 1$

B Connected to 1

2*2 Crossbar Switch

e)





1 to 8 way switch using 2*2 switch

⇒ Inter Connection Arbitration:-

A bus that connects major components in a multi-processor system, such as CPUs, IOPs & memory, is called a system bus.

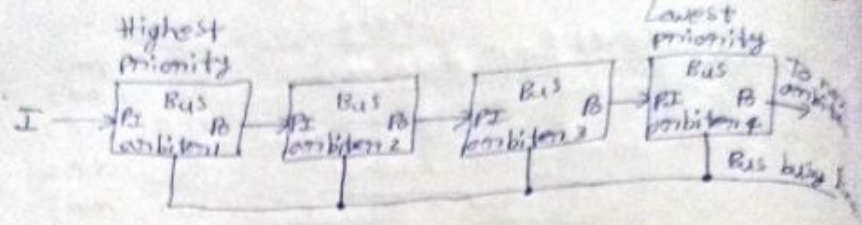
The processor in a shared memory (There will be) multiprocessor System request access to common (System) memory / other common resources through the system bus. If other processor is using the system bus, it must wait for some time. If many more, than Arbitration Logic must be performed / part of system bus controller placed b/w Local bus & System bus.

⇒ Static Arbitration:-

a) Daisy chain / Serial Arbitration.
b) Parallel Arbitration

a) Daisy chain Arbitration:-

Based on serial mechanism, it



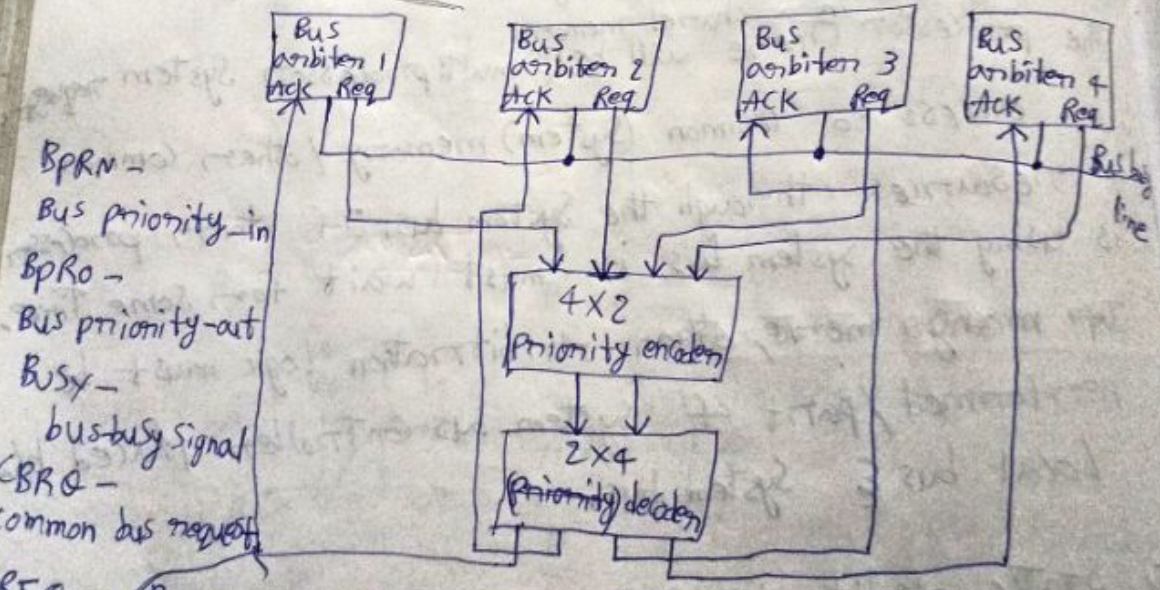
→ $P_I = 1$ & $P_I = 0$, has priority.

→ $P_I = 0$ & $P_O = 1$, Next processor

→ $P_I = 0$ & $P_O = 0$, Lowest priority

When arbiter receives control of bus, it examines the busy line. If it is inactive, no other processor is using the bus. It (arbiter) activates the busy line & it's processor takes control of the bus.

b) Parallel arbitration:-



BREQ – Bus Request
BCLK – Bus clock

Requests from 4 arbiters into encoder & 2 bit code will generated then decoder which enables proper acknowledge line to give to highest priority unit.

c) LRU:-

d) FIFO rotating daisy chain

=> dynamic arbit

Time slice:-

to all sy

communicat

b) Polling:-

which

each

access

it

a d

=> th

=) Dynamic Arbitration Algorithm:-

→ Time slice, → polling, → LRU, → FIFO (rotating daisy chain)

a) Time slice:-

No priority, in Round Robin fashion, Given to all System with time limit, processor will communicate with bus.

b) Polling:-

In this, bus grant signal is replaced by poll lines, which is connected to all units. Define address to each device, connected to the bus. If processor requires access, it recognizes it's address & in a prescribed manner, it activates the bus busy line. Polling done by choosing a different processor, Under program control.

c) LRU:-

(LRU] Least recently used processor that uses bus will have the highest priority.

d) FIFO:- rotating daisy chain

The first processor will use's the bus as

→ first in first out principle.

Round bus connections as daisy chain & bus releasing arbiter will have the least priority.

=) Hardware performance issues:-

→ microprocessors have seen an exponential increase in performance

- Improved organization
- Increased clock frequency

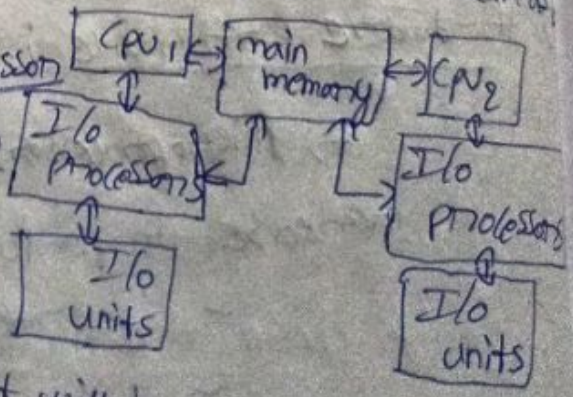
-> Increase in parallelism
  - pipelining, - superscalars (multi-issue),
  - simultaneous multi-threading (SMT)
-> Diminishing returns
  - more complexity requires more logic
  - Increasing chip area for coordinating & signal transfer logic
  - Harder to design, make & debug

=> Software performance issues:-
-> performance benefits dependent on effective exploitation of parallel resources (obviously).
-> Even small portion impacts performance.
-> many overheads of MPSoC; - comms, - distribution of work, cache coherence
-> Some APPs effectively exploit multicore processor

=> multicore organization:-
An arch. in which a single physical processor incorporates the core logic of more than one processor. A single integrated circuit is used to package/hold these processors. multicore arch. places multiprocessor cores & bundles them as a single physical processor.

=> Intel core i7 990X:-

multiprocessor organization



Used for computer processors it makes high end desktop systems, fastest processors, they think it will be used to build most powerful consumer devices.