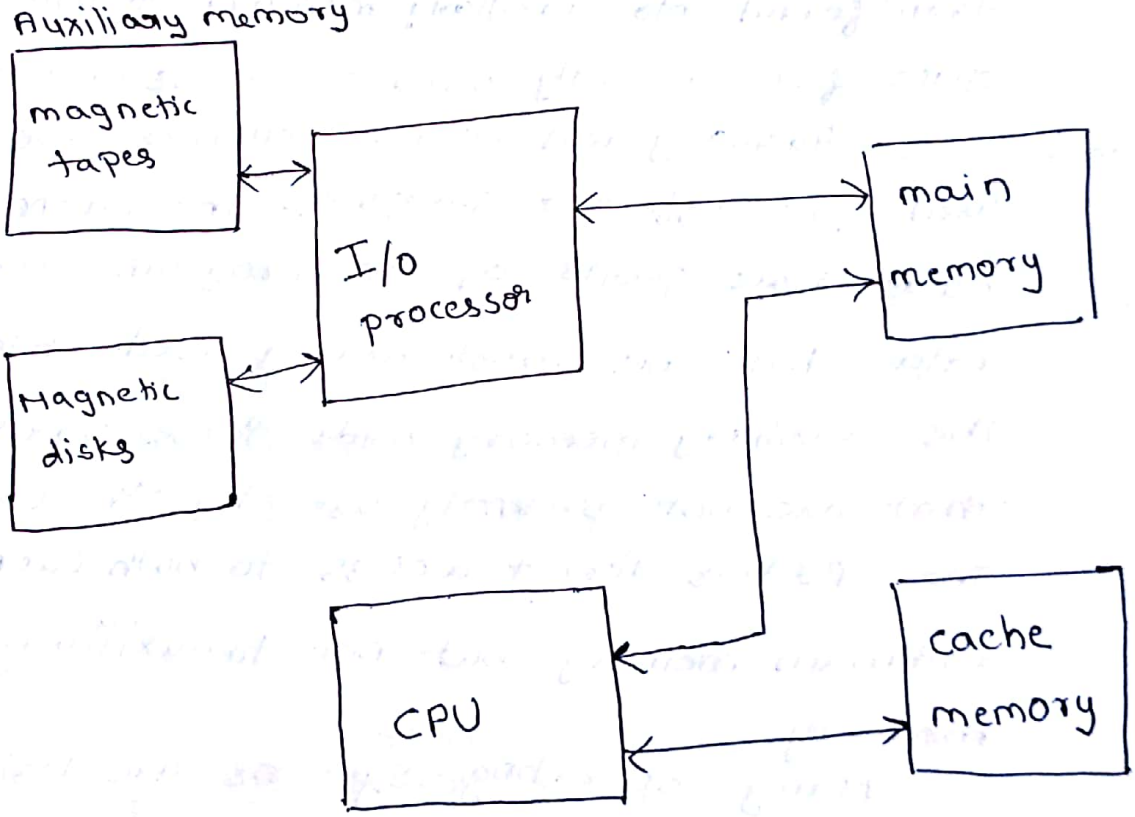
UNIT

① MEMORY ORGANIZATION

The memory unit is an essential component in any computer itsneeded for storing program and data. The memory unit that communicates directly with ~~currently~~ the CPU is called the main memory. Devices that provide backup storage are called auxiliary memory.

The total memory capacity of a computer can be visualized as being a hierarchy components



The memory hierarchy system consists of all storage devices employed in a computer system from the slow but high-capacity auxiliary memory to a relative faster main-memory, to an even smaller and faster cache-memory accessible to the high-speed processing logic

The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory Devices through an I/O processor

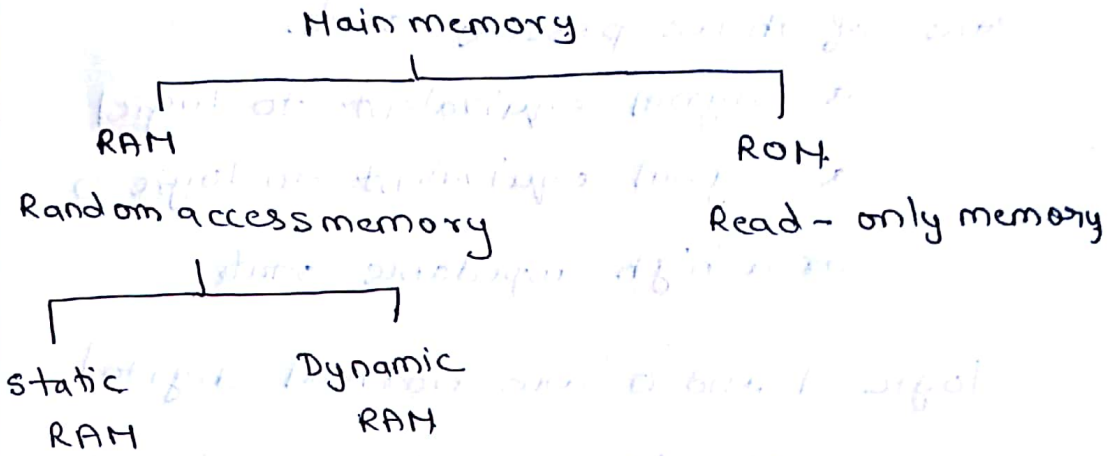
when programs not residing in main memory are needed by the CPU, they are brough in from auxiliary memory. programs not currently needed in main memory are transferred into auxiliary memory to provide space for currently used programs and data.

Auxiliary and cache memories are used for different purposes. The cache holds those parts of the program and data that are most heavily used. while the auxiliary memory holds those parts that are not presently used by the CPU. The CPU has direct access to both cache and main memory but not to auxiliary memory.

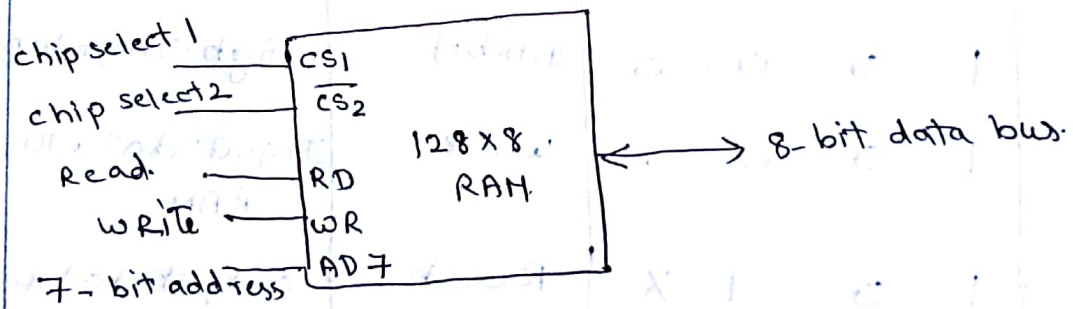
Many operating systems are designed to enable the CPU to process a number of independent program concurrently. This concept, called multiprogramming, refers to the existence of two or more programs in different parts of the memory hierarchy at the same time. In this way it is possible to keep all parts of the computer busy by working with several programs in sequence.

(3) Main memory

Most of the main memory in a general purpose computer is based on semiconductor integrated circuits.



RAM ⇒
RAM is used for storing bulk of program and data that can be subjected to change.
semiconductor memories are available in a wide range of speeds. Their cycle times range from 100ns to less than 10ns. first introduced in 1960.



It has one or more control inputs that select the chip only when needed.

Bidirectional data bus that allows the transfer of data either from memory to CPU during a read operation. or from CPU to memory during a write operation

(4)

A bidirectional bus can be constructed with three-state buffer. A three state buffer output can be placed in one of three possible state.

a signal equivalent to logic 1
a signal equivalent to logic 0
or a high impedance state

logic 1 and 0 are normal digital signal. The high impedance state behaves as an open circuit

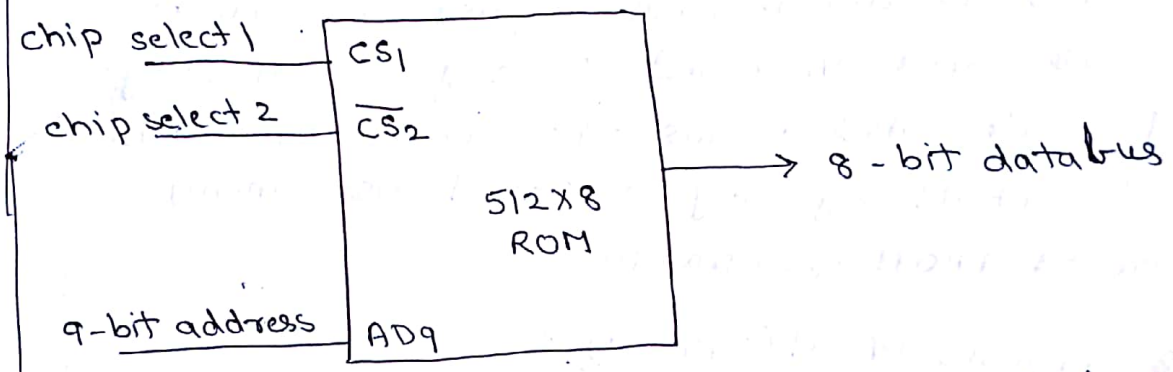
CS1	$\overline{CS2}$	RD	WR	memory Function	State of data bus
0	0	X	X	Inhibit	High-Impedance
0	1	X	X	Inhibit	High-Impedance
1	0	0	0	Inhibit	High-Impedance
1	0	0	1	write	Input data to RAM
1	0	1	X	Read	output data from RAM
1	1	X	X	Inhibit	High-Impedance

unit is in operation when CS1 = 1, $\overline{CS2}$ = 0
The RD and WR signals control the memory operation as well as the bus buffer associated with the bidirectional data bus

⑤ READ ONLY MEMORY (ROM)

It is non-volatile memory, which retains the data even when power is removed from the memory.
Programs and data that can not be altered are stored in ROM.
ROM is used for storing programs that are permanently resident in the COMPUTER
The ROM portion of main memory is needed for storing an initial program called bootstrap loader which is to start the computer operating system when power is turned on.

Typical ROM chip:



since the ROM can only READ, the data bus can only be in output mode.
No need of READ and WRITE control.
same sized RAM and ROM chip, it is possible to have more bits of ROM than of RAM, because the internal binary cells in ROM occupy less space than in RAM.

The required paths in a ROM may be programmed in four different ways

1. Mask programming ⇒

It is done by the company during the fabrication process of the unit. The procedure for fabricating a ROM requires that the customers fills out the truth table he wishes the ROM to satisfy.

2. Programmable Read only memory (PROM).

PROM contain all the fuses intact giving all 1's in the bits of the stored words. A blown fuse defines binary '0' state and an intact fuse give a binary '1' state. This allows the user to program the PROM by using a special instrument called PROM programmer.

3. Erasable PROM (EPROM) –

In a PROM once fixed pattern is permanent and can not be altered. The EPROM can be restructured to the initial state even though it has been programmed previously. When EPROM is placed under a special ultra-violet light for a given period of time all the data are erased. After erasing, the EPROM returns to its initial state and can be programmed to a new set of values

4. Electrically Erasable PROM (EEPROM) :-

It is similar to EPROM except that the previously programmed connections can be erased with an electrical signal instead of ultra violet light. The advantage is that device can be erased without removing it from its socket

Auxiliary memory

Also called as secondary memory, used to store large chunks of data at a lesser cost per byte than a primary memory for backup.

It does not lose the data when the device is powered down. It is non-volatile.

It is not directly accessible by the CPU., they are accessed via the input/output channels.

The most common form of auxiliary memory device used in consumer systems is flash memory optical disc, and magnetic disks, magnetic tapes

Types of Auxiliary memory

Flash memory →

An electronic - non volatile computer storage device that can be electrically erased and reprogrammed, and works without any moving parts. examples of this are USB, flash drives and solid state drives

optical disc →

It is a storage medium from which data is read and to which it is written by lasers. There are three types of optical disks

CD-ROM → (read only

WORM → write once read many

~~WORM~~

EO → erasable optical disks

Magnetic tapes.

A magnetic tape consists of electric, mechanical and electronic components to provide the parts and control mechanism for a magnetic tape unit.

The tape itself is a strip of plastic coated with a magnetic recording medium. Bits are recorded as a magnetic spots on tape along several tracks called Records

Each record on tape has an identification bit pattern as the beggining and end. R/W heads are mounted in each track so that data can be recorded and read as a sequence of characters

can be stopped, started to move forward, or in reverse, or can be rewound but cannot be stopped fast enough between individual characters

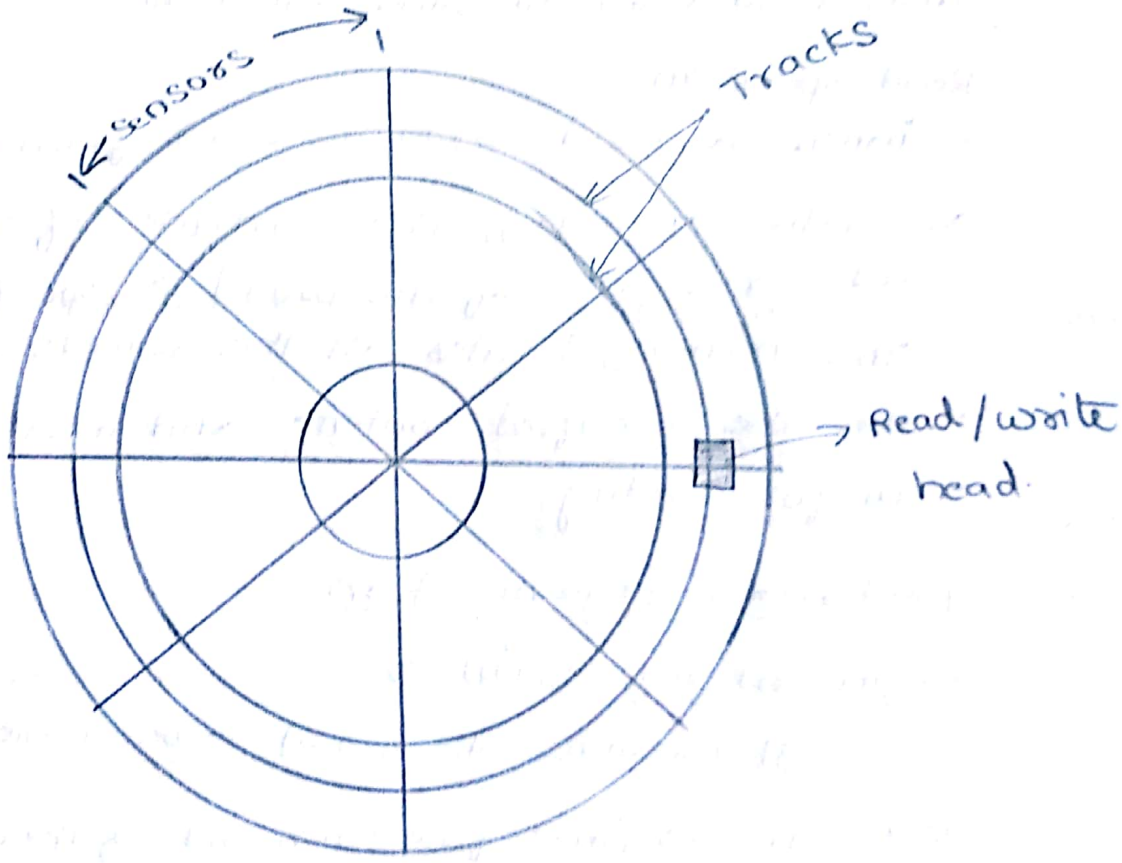
Magnetic Disk

A magnetic disk is a circular plate constructed of metal or plastic coated with magnetized material.

Both sides of the disk are used and several disks may be stacked on one spindle with read/write heads available on each surface.

Bits are stored in magnetized surface in spots along concentric circles called tracks. Tracks are commonly divided into section called sectors

Disks that are permanently attached and cannot be removed by occasional users are called hard disks



magnetic disk

ASSOCIATIVE MEMORY

A memory unit accessed by contents is called an associative memory or content addressable memory (CAM)
This type of memory is accessed simultaneously and in parallel on the basic of data content rather than by specific address or location

Read / write operation in CAM

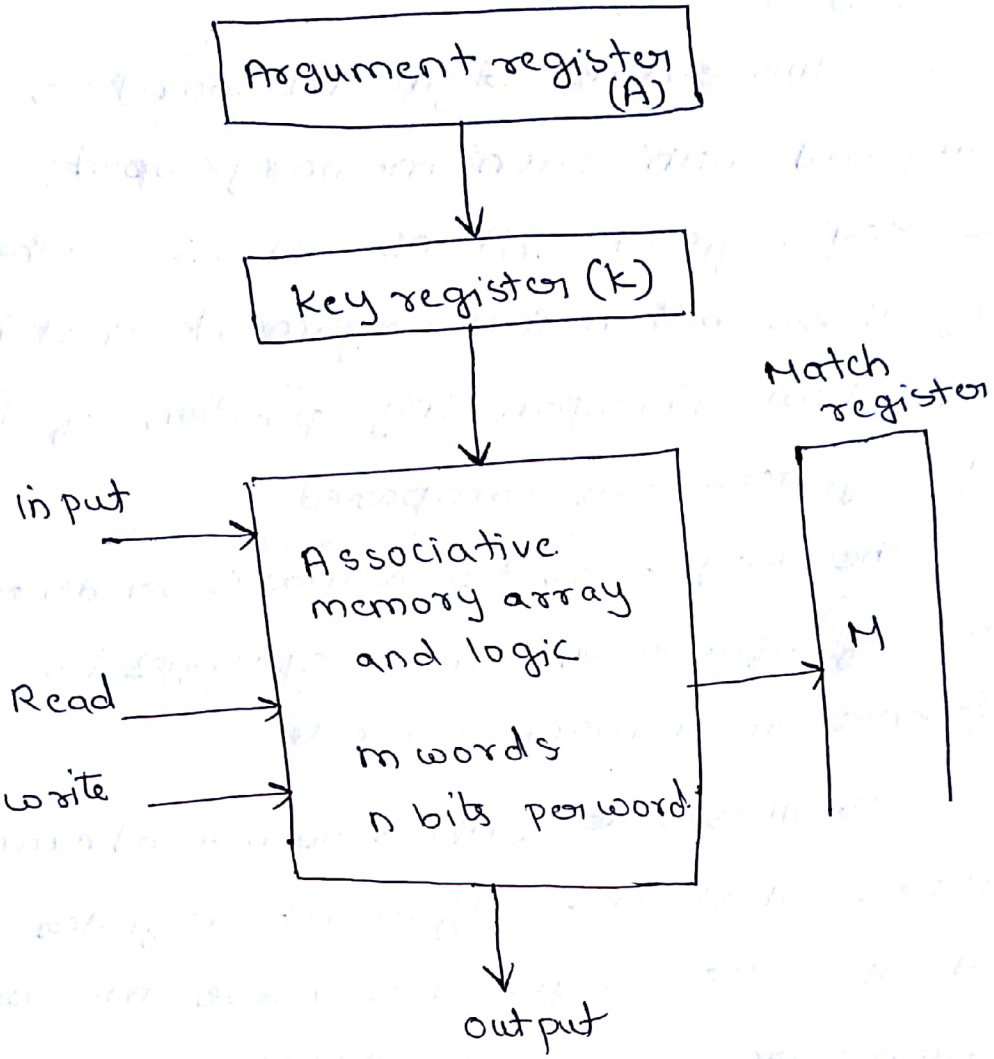
write operation ⇒
when word is written in an associative memory, no address is given
The memory is capable of finding an unused location to store the word.

Read operation.
when a word is to be read from an associative memory, the contents of the word, or a part of the word is specified
The memory locates all the words which match the specified content and marks them for reading.

Hardware organization

Argument register (A) ⇒
gt contains the word to be searched.
gt has n bits (one for each bit of the word)

Key register (K) ⇒
gt provides mask for choosing a particular field or key in the argument word.
gt also has n bits



Associative memory array → It contains the words which are to be compared with the argument word.

Match Register (M) → It has m bits, one bit corresponding to each word in the memory array After the matching process, the bits corresponding to matching words in match register are set to 1.

~~Matching process~~

Matching process

The entire argument word is compared with each memory word, if the key register contains all 1's. otherwise only those bits in the argument that have 1's in their corresponding position of the key registers are compared.

The key provides a mask or identifying piece of information which specifies how the reference to memory is made.

To illustrate with a numerical example suppose that the argument register A and the key register K have the bit configuration as shown below.

only the three left most bits of A are compared with the memory words because K has 1's in these three position only

A	101	111100	
K	111	000000	
word 1	100	111100	no match
word 2	101	000001	match

Disadvantages

An associative memory is more expensive than a random access memory because each cell must have an extra storage capability as well as logic circuits for matching its content with an external argument.

associative memories are used in application where the search time is very critical and must be very short.

(14) CACHE MEMORY

If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced.

Thus reducing the total execution time on the program.

Such a fast small memory is referred to as cache memory.

The cache is the fastest component in the memory hierarchy and approaches the speed of CPU component

When CPU needs to access memory, the cache is examined.

If the word is found in the cache; it is read from the cache memory.

If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word.

A block of words containing the one just accessed is then transferred from main memory to cache memory

If the cache is full, then a block equivalent to the size of the used word is replaced according to the replacement algorithm being used

Hit ratio

When the CPU refers to memory and finds the ~~word~~ word in cache, it is said to produce a hit

otherwise, it is a miss

The performance of cache memory is frequently measured in terms of a quantity called hit ratio

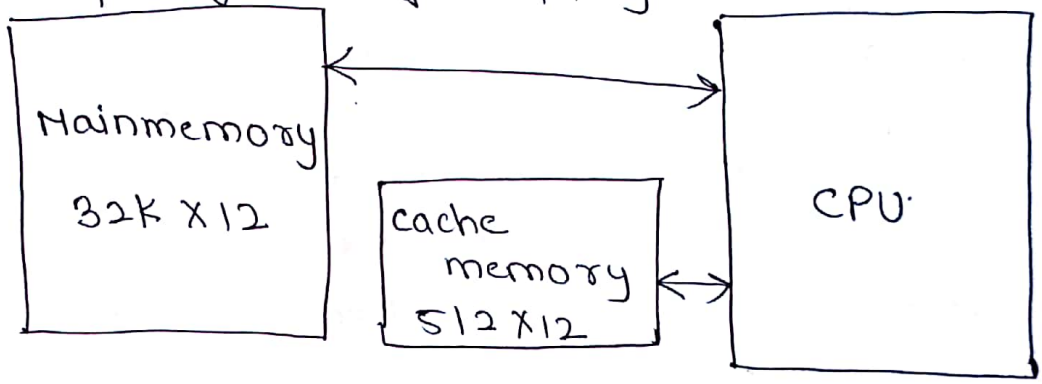
$$\text{Hit ratio} = \text{hit} / (\text{hit} + \text{miss})$$

Mapping process ⇒

The transformation of data from main memory to cache memory is reffered to as a mapping process. there are three types of mapping.

1). Associative mapping
2) Direct mapping
3). set-associative mapping.

example of memory mapping



Associative mapping

CPU address (15 bits)

↓

Argument register

← Address →	← Data →
01000	3450
02777	6710
22345	1234

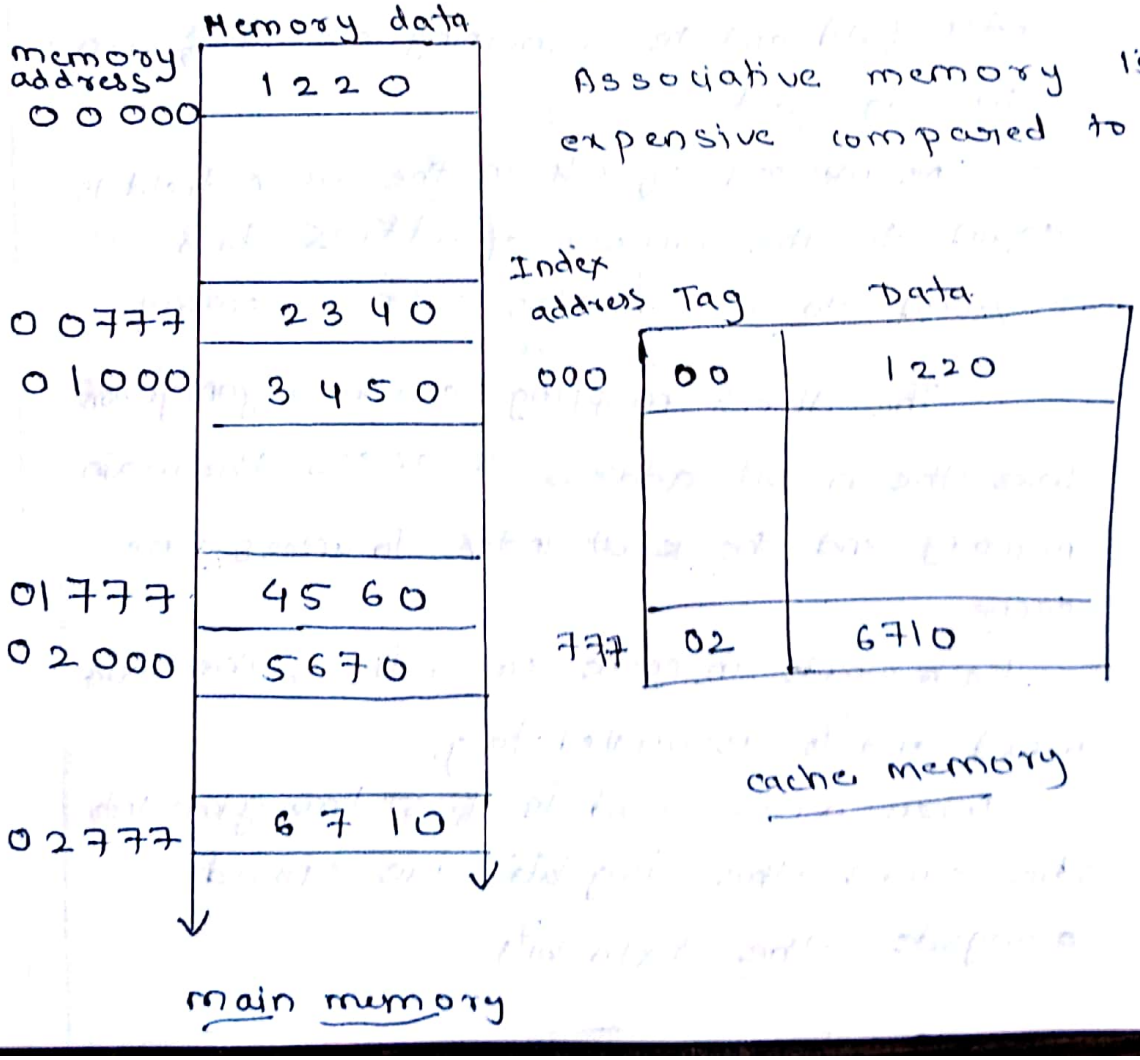
→ The fastest and most flexible cache organization uses an associative memory

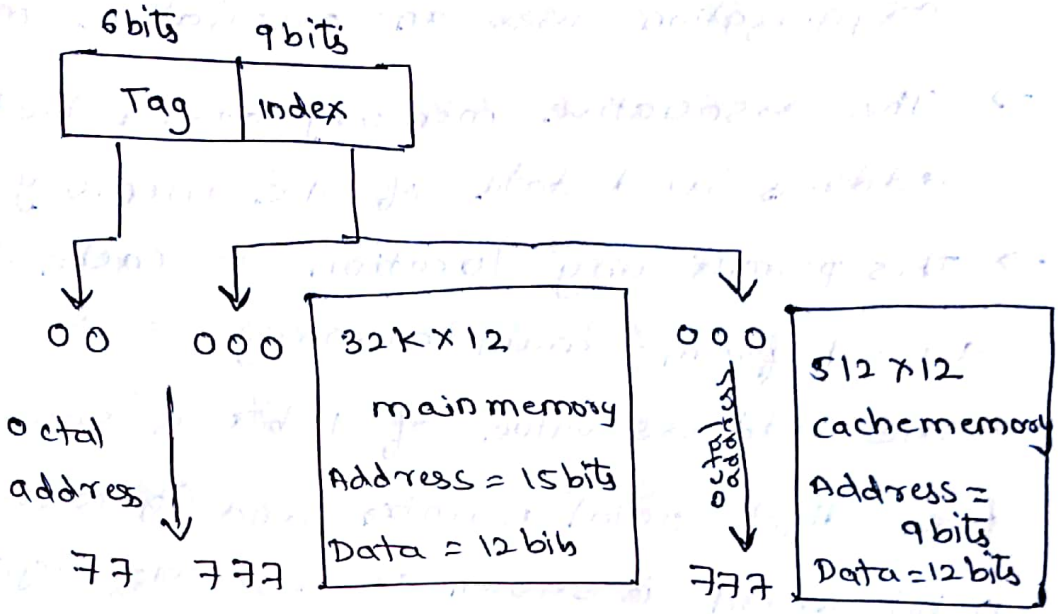
→ The associative memory stores both the address and data of the memory word

→ This permits any location in cache to store a word from main memory

→ The address value of 15 bits is shown as a five-digit octal number and its corresponding 12 bit word is shown as a four-digit octal number

Direct Mapping





The CPU address of 15 bits is divided into two fields.

The nine least significant bits constitute the index field and the remaining six bits form the tag field.

The number of bits in the index field is equal to the number of address bits required to access the cache memory

The direct mapping cache organization uses the n-bit address to access the main memory and the k-bit index to access the cache.

Each words in cache consists of the data word and its associated tag.

When a new word is first brought into the cache the tag bits are stored alongside the data bits.

When the CPU generates a memory request the index field is used for the address to access the cache.

The tag field of the CPU address is compared with the tag in the word read from the cache.

If the two tags match, there is a hit and the desired data word is in cache.

If there is no match, there is a miss and the required word is read from the main memory. It is then stored in the cache together with the new tag replacing the previous value.

~~Set~~ ~~Associative~~ ~~Mapping~~

disadvantage of direct mapping is that two words with the same index in their address but with different tag values cannot reside in cache memory at the same time.

Set- Associative Mapping

Set associative mapping is an improvement over the direct mapping organization

Each word of cache can store two or more words of memory under the same index address

Each data word is stored together with its tag and the number of tag-data items in one word of cache is said to form a set.

index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340

When the CPU generates a memory request, the index value of the address is used to access the cache. The tag

of the field of the CPU address is then compared with both tags in the cache. The tag field of the CPU address is then compared with both tags in the cache to determine if a match occurs.

Replacement algorithm

When a miss occurs in a set-associative cache and the set is full, it is necessary to replace one of the tag-data items with a new value. The most common replacement algorithms are.

1). random replacement.

2) first-in, first-out (FIFO)

3). least recently used (LRU)

Random replacement ⇒ The control chooses one tag-data item for replacement at random

FIFO ⇒ The procedure selects for replacement the items that has been in the set the longest

LRU ⇒ The LRU algorithm selects for replacement the item that has been least recently used by the CPU.

(22) PIPELINE AND VECTOR PROCESSING

Parallel processing

Instead of processing each instruction sequentially, we use a different technique called parallel processing

Parallel processing ⇒ In this technique we perform concurrent data processing to achieve faster execution time

For exg → while an instruction is being executed in the ALU, the next instruction can be read from memory

"The purpose of parallel processing is to speed up the computer processing capabilities"

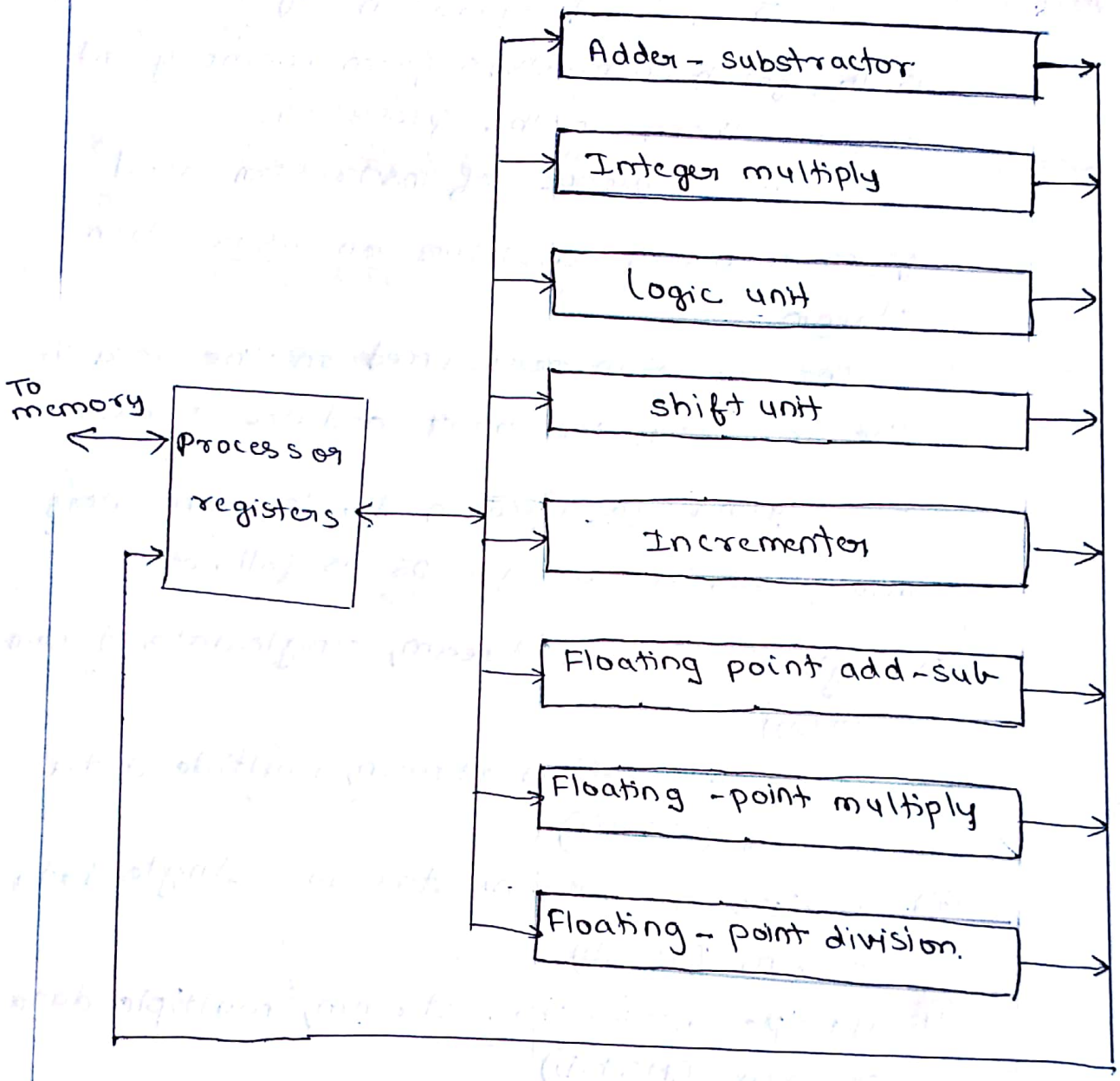
For achieving this the hardware had to be increased

~~Parallel processing~~

multiple Functional unit ⇒

Parallel processing at a higher level of complexity can be achieved by by having a multiple functional unit that performs identical or different operation simultaneously

one possible way of separating the execution unit into eight functional unit operating in parallel



All units are independent of each other. So one number can be shifted while another number is being incremented. A multifunctional organization is usually associated with a complex control unit to coordinate all the activities among the various components.

FLYNN's classification is based on instruction stream and Data stream for parallel computing

The normal operation of a computer is to fetch instructions from memory and execute them in the processor.

The sequence of instruction read from memory constitues an instruction stream.

The operation performed on the data in the processor constitute a data stream

Flynn's classification divides computers into four major groups as follows

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

PIPELINING

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

Suppose that we want to perform the combined multiply and add operation with a stream of number

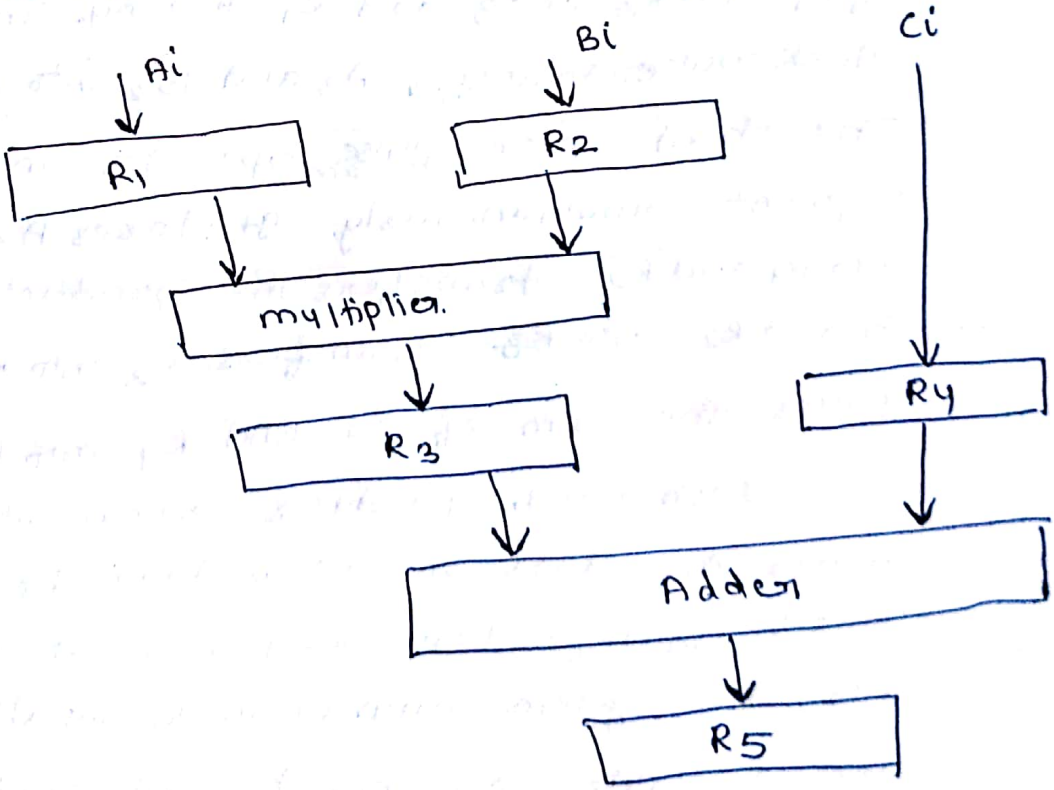
$A_i * B_i + C_i$ for $i = 1, 2, 3 \ldots 7$

The R_1 through R_5 are registers that receive new data with every clock pulse. The suboperation performed are.

$R_1 \leftarrow A_i$, $R_2 \leftarrow B_i$ input A_i and B_i

$R_3 \leftarrow R_1 * R_2$, $R_4 \leftarrow C_i$ multiply and input C_i

$R_5 \leftarrow R_3 + R_4$ Add C_i to product



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	—	—	—	—	$A_7 * B_7 + C_7$

The first clock pulse transfers A_i and B_i into R_1 and R_2

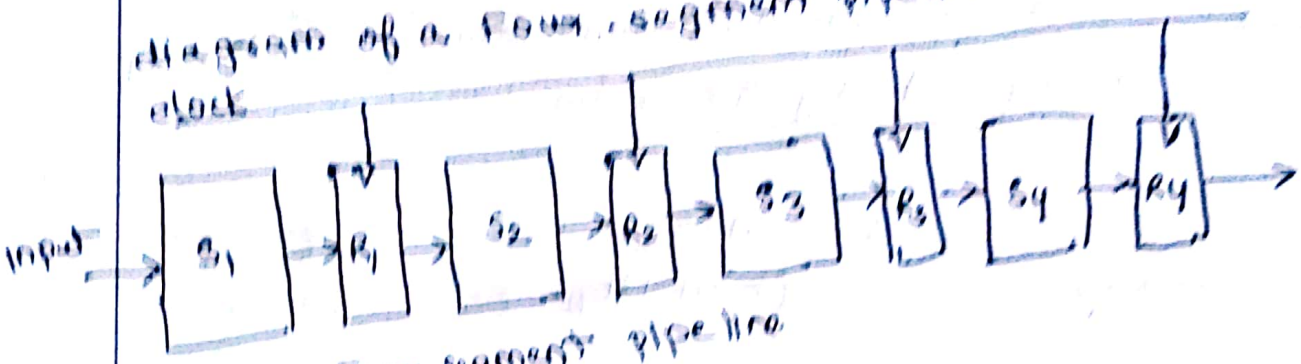
The second clock pulse transfer the product of R_1 and R_2 in R_3 and C_1 into R_4. The same clock pulse transfer A_2 and B_2 into R_1 and R_2.

The third clock pulse operates on all three segment simultaneously. It places A_3 and B_3 into R_1 and R_2, transfers the product of R_1 and R_2 into R_3, transfers C_2 into R_4 and places the sum of R_3 and R_4 into R_5.

each clock produces a new output and moves the data one step down the pipeline This happen as long as new input data flow into the system when no more input data are available, the clock must continue untill the last output emerges out of the pipeline.

(21)

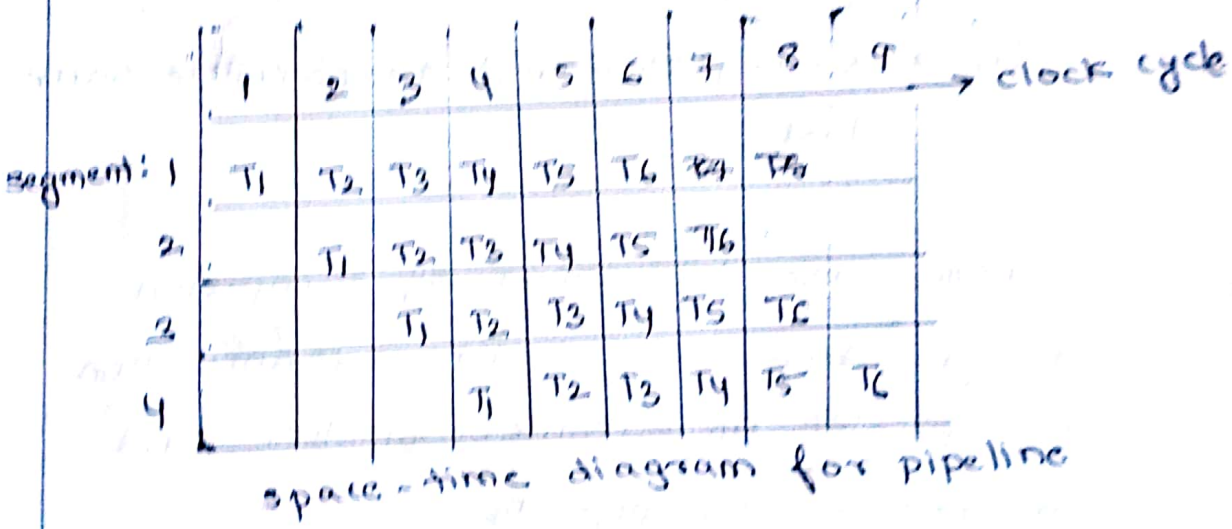
The behavior of a pipeline can be shown with a space-time diagram. The space-time diagram of a four-segment pipeline



Four segment pipeline

S_i : combinational ckt

R_i : Register that holds the result



space-time diagram for pipeline

The horizontal axis displays the time in clock cycle and the vertical axis gives the segment number. The diagram shows tasks T_1 through T_6 executed in four segment. Initially, task T_1 is handled by segment 1. After the first cycle clock, segment 2 is busy with T_1, while segment 1 is busy with task T_2.

The speedup of a pipeline processing over an equivalent non pipeline processing is defined by the ratio

$$S = \frac{nts}{(k+n-1)t_p}$$

k → No of segments

tp → clock cycle time. (pipeline unit)

n → no of task executed

tn → a non pipeline unit perform the same task

There are two areas of computer design where the pipeline organization is applicable. An arithmetic pipeline and an instruction pipeline.

Arithmetic pipeline → divides an arithmetic operation into ~~segment~~ sub-operations for execution in the pipeline segments

Instruction pipeline → operates on a stream of instruction by overlapping the fetch, decode and execute phase of the instruction cycle.

Arithmetic pipeline.

Floating-point operations are easily decomposed into sub operation.

exg ⇒ input to the Floating-point adder pipeline are two normalized floating-point binary numbers

The sub operation that are performed in the four segment are

1. compare the exponents.
2. Align the mantissa.
3. Add or subtract the mantissa.
4. Normalize the result.

$$X = A \times 2^a$$

$$Y = B \times 2^b$$

consider the two normalised floating-point numbers

$$X = 0.9504 \times 10^3$$

$$Y = 0.8200 \times 10^2$$

The larger exponent is chosen so the two exponents are subtracted in <u>first segment</u>

$$X = 0.9504 \times 10^3$$

$$Y = 0.08200 \times 10^3$$

segment-2 ⇒ it shifts the mantissa.

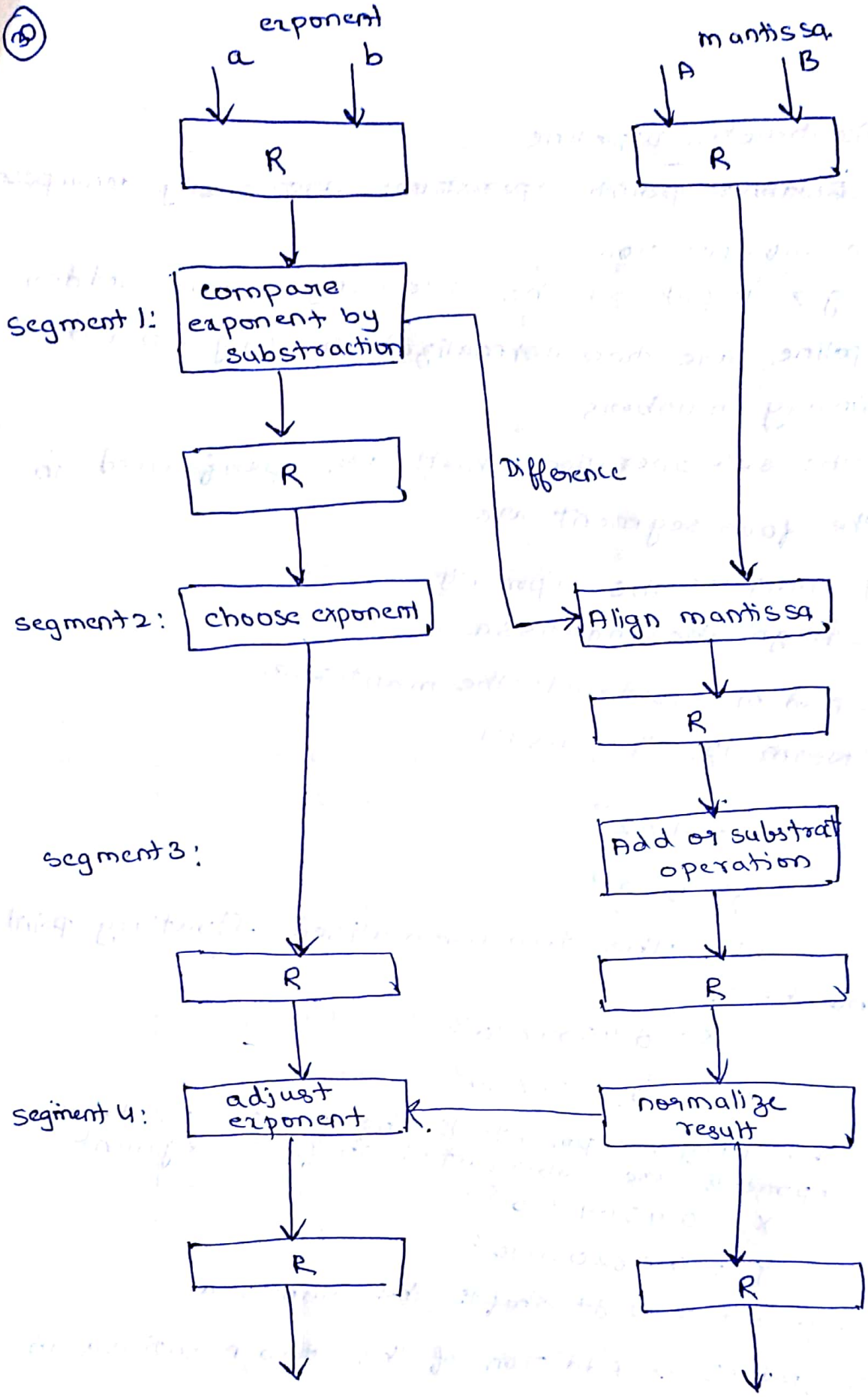
segment 3 ⇒ Addition of the two mantissa in segment 3

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

in decimal number it is normalized by shifting the fraction with a non zero first digit.

$$Z = 0.10324 \times 10^4$$

(20)



Pipeline for floating-point addition and subtraction

(31) INSTRUCTION PIPELINE

In the most general case, the computer needs to process each instruction with the following sequence of steps

1. Fetch the instruction from memory
2. Decode the instruction.
3. calculate the effective address
4. Fetch the operands from memory.
5. Execute the instruction.
6. Store the result in the proper place.

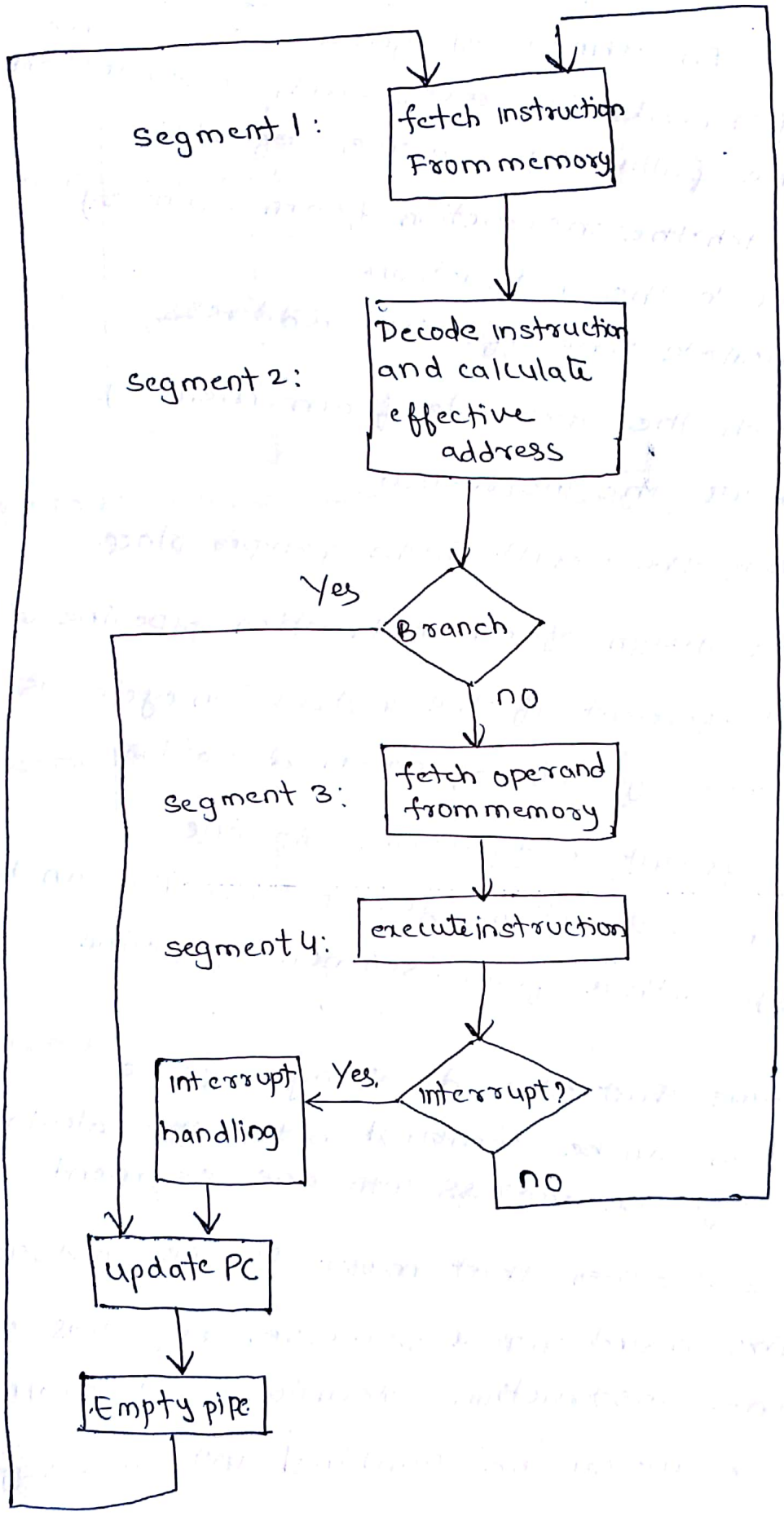
The design of an instruction pipeline will be most efficient if the instruction cycle is divided into segments of equal duration.

Four - segment instruction pipeline.

The instruction cycle in the CPU can be processed with a four - segment pipeline.

→. Assume that the decoding of the instruction can be combined with the calculation of the effective address into one segment.

→. Assume further that most of the instruction places the result into a processor registers so that the instruction execution and storing of the result can be combined into one segment



Four - segment CPU pipeline

The Four segments are represented in the diagram with an abbreviated symbol. The time in the horizontal axis is divided into steps of equal duration

1. FI is the segment that fetches the instruction
2. DA is the segment that decodes the instruction and calculates the effective address
3. FO is the segment that fetches the operand
4. EX is the segment that executes the instruction

STEP		1	2	3	4	5	6	7	8	9	10	11	12	13
INSTRUCTION	1	FI	DA	FO	EX									
	2		FI	DA	FO	EX								
(BRANCH)	3			FI	DA	FO	EX							
	4				FI	-	-	FI	DA	FO	EX			
	5					-	-	-	FI	DA	FO	EX		
	6									FI	DA	FO	EX	
	7										FI	DA	FO	EX

Timing of Instruction pipeline.

→ It is assumed that the processor has separate instruction and data memories so that the operation in FI and FO can proceed at the same time.

→ Thus, in step 4, instruction 1 is being executed in segment EX. the operand for instruction 2 is being fetched in segment FO; instruction 3 is being decoded in segment DA; and instruction 4 is being fetched from memory in segment FI

→ Assume now that instruction 3 is a branch instruction

→ As soon as this instruction is decoded in segment DA in step 4., the transfer from FI to DA of the other instruction is halted until the branch instruction is executed in step 6.

→ If the branch is taken, a new instruction is fetched in setp step 7. If the branch is not taken the instruction Fetched previously in step 4 can be used.

→ The pipeline continues untill a new branch instruction is encountered

Another delay may occur in the pipeline. If the EX segment needs to store the result of the operation in the data memeory while the FO segment needs to fetch an operand In that case, segment FO must wait until segment EX has finished its operation

Pipeline conflicsts

There are three major difficulties that cause the instruction pipeline to deviate from its normal operation

1) Resource conflicts → caused by access to memory by two segments at the same time. Most of these conflicts can be resolved by using separate instruction and data memories

2). Data dependency conflicts arise when an instruction depends on the result of a previous instruction, but this result is not yet available.

3). Branch difficulties arise from branch and other instructions that change the value of PC.

RISC pipeline ⇒

Reduced instruction set computer

RISC has an ability to execute instruction at the rate of one per clock cycle.

Three-segment Instruction pipeline

The instruction cycle can be divided into three suboperations and implemented in three segments

I : Instruction

A : ALU operation

E : Execute instruction.

The I segment fetches the instruction from the program memory.

The instruction is decoded andan ALU operation is performed in the A segment

The ALU is used for three functions

→ it performs an operation for a dat manipulation instruction

→ It evalutes the effective adress

→ calulates the branch address

The E segment direct the o/p of the ALU to one of the three distination depending on the decoded instruction

gt transfers the result of the ALU operation into a destination register in the register file.

gt transfer the effective address to a data memory for loading or storing

gt transfers the branch address to the program counter.

Delayed load

consider now the operation of the following

Four instruction

1. LOAD: $R_1 \leftarrow M[\text{address } 1]$
2. LOAD: $R_2 \leftarrow M[\text{address } 2]$
3. ADD: $R_3 \leftarrow R_1 + R_2$
4. STORE: $M[\text{address } 3] \leftarrow R_3$

If the three-segment pipeline proceeds without interruptions, there will be a data conflict in inst-3 because the operand in R_2 is not yet available in the A segment

The compiler inserts an NO-OP (no operation) instruction, wasting a clock cycle. This concept of delaying the use of the data loaded from memory is referred to as delayed load.

clock cycle	1	2	3	4	5	6
1. Load R1	I	A	E			
2. Load R2		I	A	E		
3. Add R1+R2			I	A	E	
4. Store R3				I	A	E

pipeline timing with data conflict.

clock cycle	1	2	3	4	5	6	7
1. Load R1	I	A	E				
2. Load R2		I	A	E			
3. NO-operation			I	A	E		
4. Add R1+R2				I	A	E	
5. Store R3					I	A	E

Three-stage

Pipeline timing with delayed load

Branch delayed

Let us consider the program having the Following 5 instruction.

Load from memory to R1
Increment R2
Add R3 to R4
substract R5 from R6
Branch to address X

The method used in most RISC Processor is to rely on the compiler to redefine the branches so that they take effect at the proper time in the pipeline.

This method is refferd to as branch delay. The compiler for a processor

to analyze the instruction before and after the branch is designed to analyze the instruction before and after the branch and rearrange. the program sequence by inserting useful instruction in the delay step

clock cycle	1	2	3	4	5	6	7	8	9	10
1 Load	I	A	E							
2. Increment		I	A	E						
3. Add			I	A	E					
4. Substract				I	A	E				
5. Branch to X					I	A	E			
6. No operation						I	A	E		
7. No-operation							I	A	E	
8. instruction in X								I	A	E

using no operation instruction

Clock-cycle.	1	2	3	4	5	6	7	8
1. Load	I	A	E					
2. Increment		I	A	E				
3. Branch to X			I	A	E			
4. Add				I	A	E		
5. substract					I	A	E	
6. Instruction in X						I	A	E

Rearranging the instruction

The compiler inserts two no-op instructions after the branch.

In the Rearranging the program is rearranged by placing the add and substract instruction after the branch instruction.

Vector Processing

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

These are characterized by the fact that they require vast number of computation and it take a conventional computer days or even weeks to complete.

computer with vector processing are able to handle such instruction and they have application in following fields

long range weather forecasting
Petroleum exploration.
Seismic data analysis
medical diagnosis
Aerodynamic and space simulation
Artificial intelligence and expert system
mapping the human genome.
Image processing

Vector operation

A vector v of length n is represented as row vector by

$$V = [V_1 \quad V_2 \quad V_3 \quad \ldots \quad V_n]$$

The element Vi of vector V is written as V(I) and the index I refers to a

to a memory address or register where the number is stored.

Let us consider the program in assembly language that two vectors A and B of length 100 and put the result in C

```
    Initialize I = 0
20  Read A(I)
    Read B(I)
    store C(I) = A(I) + B(I)
    Increment I = I + 1
    if I ≤ 100 go to 20
    continue.
```

A computer capable of vector processing eliminates the overhead associated with the time it takes to fetch and ercute the Instruction in the program loop

gt allows operations to be specified with a single vector instruction of the form

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

Instruction format for vector processing

operation code	Base address source1	Base address source 2	Base address destination	vector length

Matrix multiplication

Let us consider the multiplication of two 3*3 matrix A and B

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

The product matrix C is a 3 x 3 matrix whose elements are related to the elements of A and B by inner product

$$c_{ij} = \sum_{k=1}^{3} a_{ik} + b_{kj}$$

For example the number in the first row and first column of matrix C is calculated by letting $i=1$, $j=1$ we obtain

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

This requires three multiplication and (after initializing c_{11} to 0) three addition. Total number of addition or multiplication required is 3*9

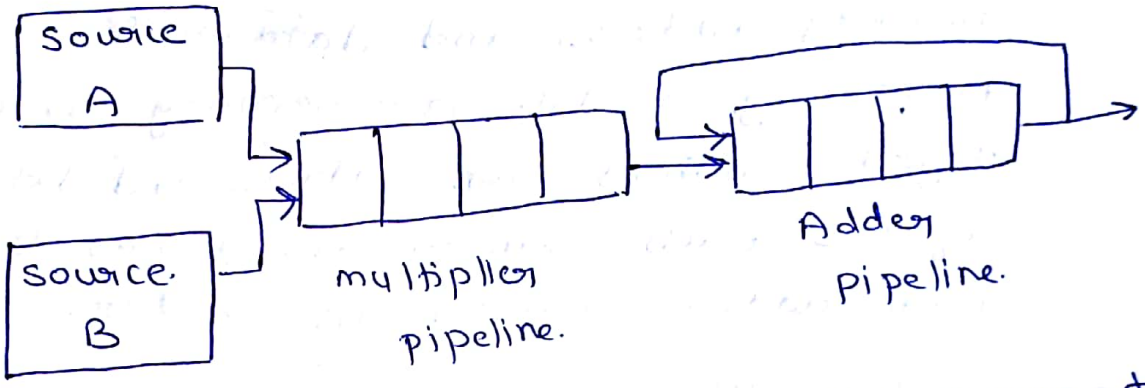
In general inner product consists of the sum of k product term of form

$$C = A_1B_1 + A_2B_2 + A_3B_3 + A_4B_4 \cdots + A_kB_k$$

In typical application value of K may be 100 or even 1000

The inner product calculation on a pipeline vector processor is shown below

Floating point adder and multiplier are assumed to have four segment each



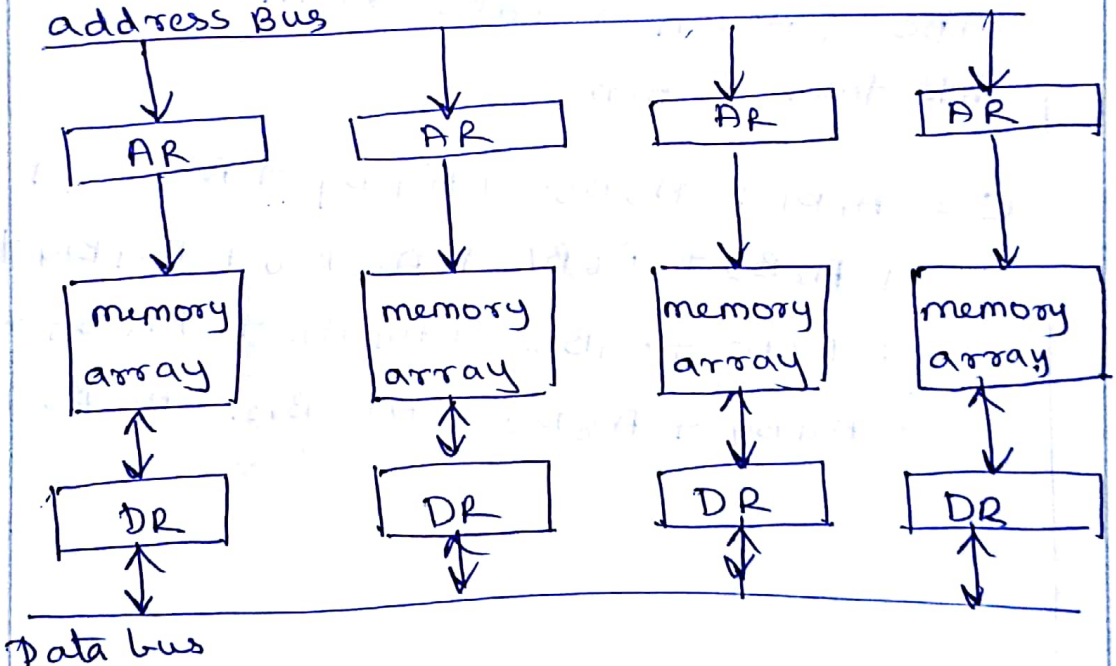
The pattern breaks down the summation into four section.

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

44 Memory Interleaving

Pipeline and vector processor requires simultaneous access to memory from two or more source. Instead of using two memory buses for simultaneous access, the memory can be partitioned into a number of modules connected to a comman memory address and data buses.

A memory module is a memory array together with its own address and data registers. Each memory array has its own address register AR and data register DR.



The advantage of a modular memory is that it allows the use of interleaving In a interleaving memory, different sets of addresses are assigned to different memory modules

Array processor

An array processor is a processor that performs the computations on large arrays of data.

There are two different types of array processor

1). Attached array processor

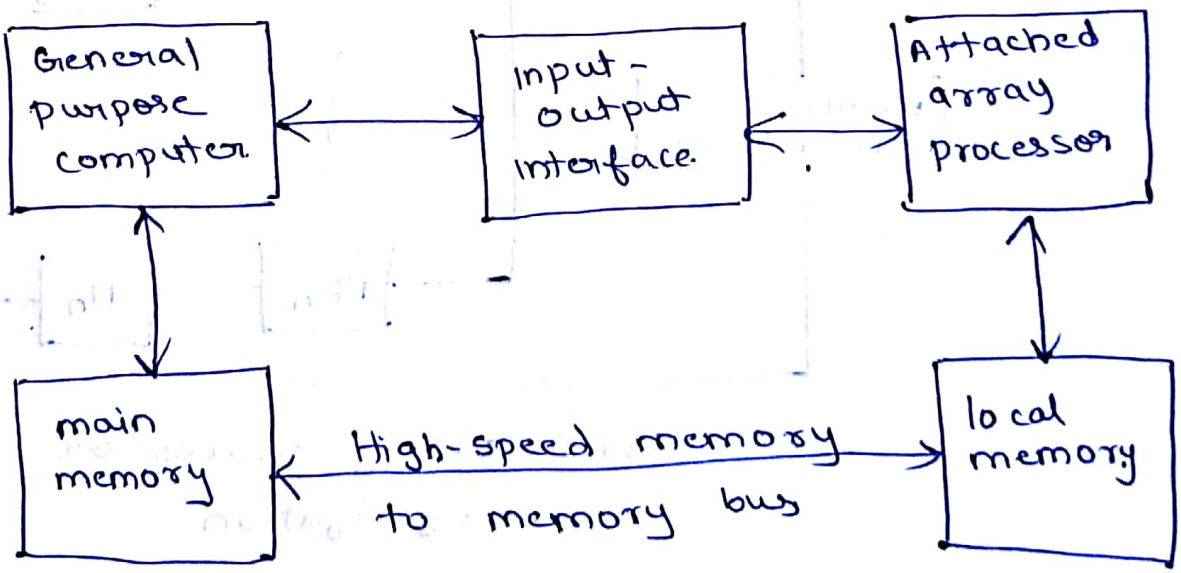
2) SIMD array processor.

Attached array processor.

It is designed as a peripheral for a conventional host computer

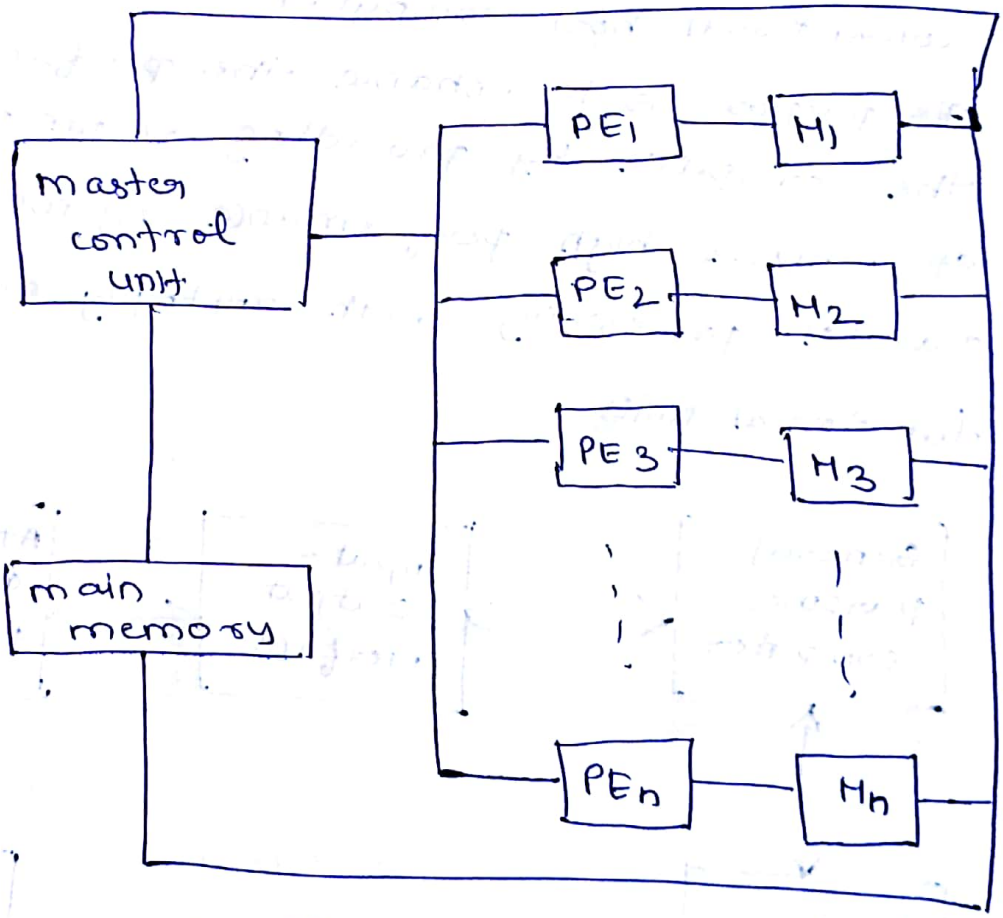
Its purpose is to enhance the performance of the computer by providing vector processing

It achieves high performance by means of parallel processing with ~~multiply~~ multiple functional units.



SIMD Array processor

It is a processor which consists of multiple processing unit operating in parallel.
The processing units are synchronized to perform the same task under control of common control unit.
Each processor elements (PE) includes an ALU, a floating point arithmetic unit and working Register.



SIMD array processor organization

MULTIPROCESSORS

Characteristics of multiprocessors

Multiprocessor system is an interconnection of two or more CPU's with memory and input-output equipment

The components that forms multiprocessor are CPU's, IOP's connected to input-output devices and memory unit thay may be partitioned into a number of separate modules.

multiprocessor are classified as multiple instruction stream, multiple data stream (MIMD) system

→ why choose a multiprocessor.

multiple users

multiple applications

Multi-tasking within an application

Responsiveness and / or throughput

share hardware between CPU's

what are the difference between multiprocessor and multicomputer.

Multiprocessor	multicomputer
→ A multiprocessor system is a computer that has more than one CPU on its motherboard.	A computer made up of several computers. The term generally refers to an architecture in which each processor has its own memory rather than multiple processor with a shared memory
→ Multiprocessor is the use of two or more central processing units (CPU) within a single computer system	

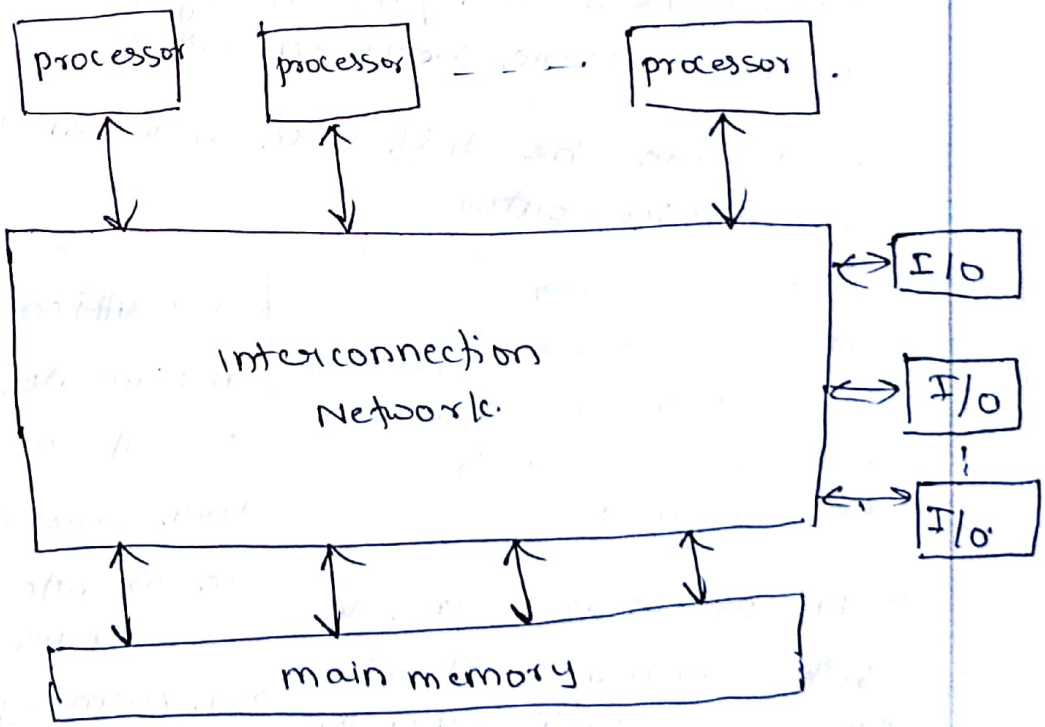
(48) Classification of multiprocessor

multiprocessor are classified by the way their memory is organized. mainly it is classified into two type.

1. Tightly coupled multiprocessor
2. loosely coupled multiprocessor

Tightly coupled multiprocessor

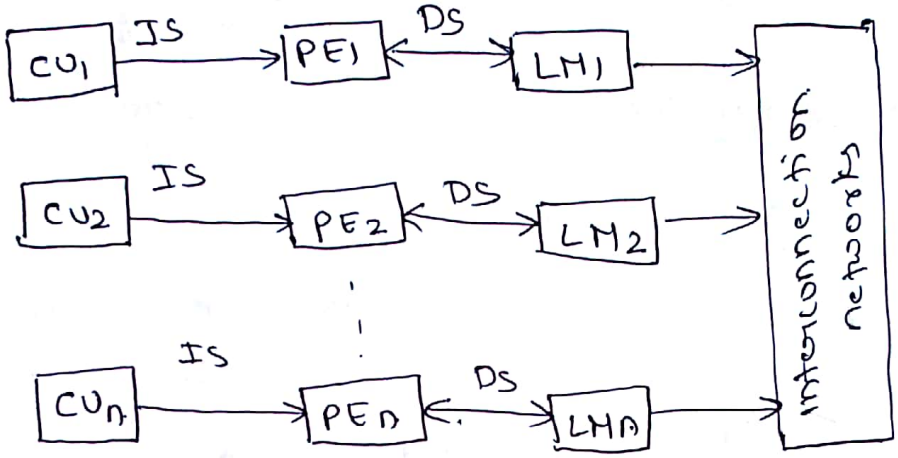
A multiprocessor is a tightly coupled computer system having two or more processing units (multipe processors) each sharing main memory and peripherals; in order to simultaneously process program Tightly coupled multiprocessor is also known as shared memory system



→ Processor share memory
→ communication via that shared memory

Loosely coupled multiprocessor system (often refferred to as clusters) are based on multipe standalone single of dual processor commodity computers interconnected via a high speed communication system

Loosely - coupled multiprocessor is also known as distributed memory



Comparison chart

	Tightly coupled	loosely coupled
Basic.	Processors have shared memory modules	Each processor has its own memory module.
efficient	efficient for high speed or real time processing	efficent when task running on different processors, has minimal interaction
memory conflict	It experiences more memory conflict	It generally, do not encounter memory conflict
interconnections	~~message transfer system~~ (MTS) interconnection network	~~Interco~~ message transfer system
Datarate	high	low
expensive.	more expensive.	less expensive.

Interconnection structures

The components that form a multiprocessor system are CPU's, IOP's connected to input output devices and memory unit

The interconnection between the components can have different physical configuration, depending on the number of transfer path that are available between the processor and memory in a shared memory system.

The physical forms for establishing an interconnection network.

1. Time shared common bus.
2. Multiport memory
3. crossbar switch
4. Multistage switching networks
5. Hypercube system

1. Time shared common bus

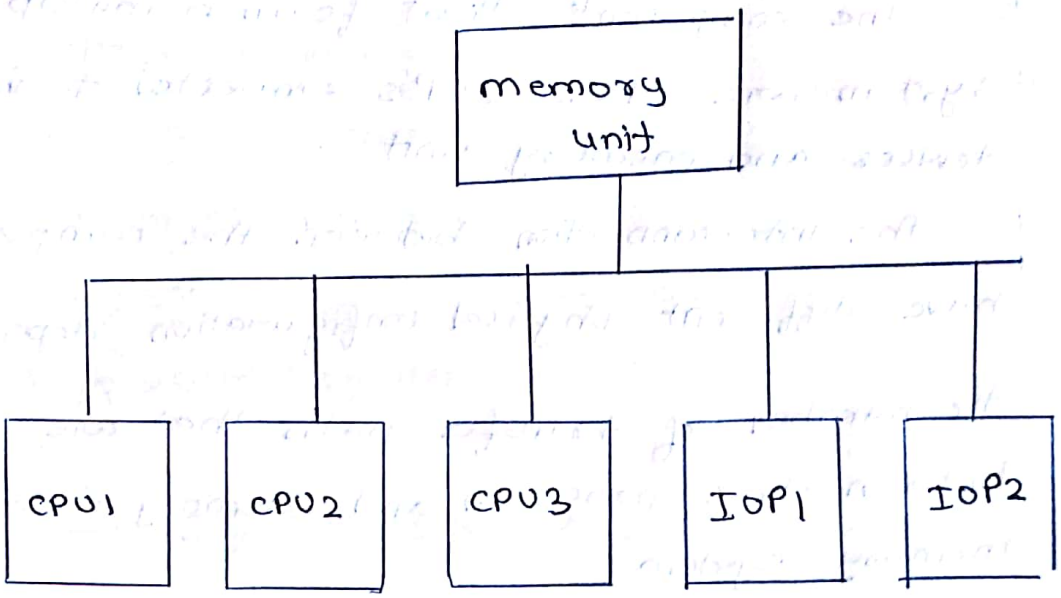
A common-bus multiprocessor system consists of a number of processors connected through a common path to a memory unit

Disadvantage

only one processor can communicate with the memory or another processor at any given time.

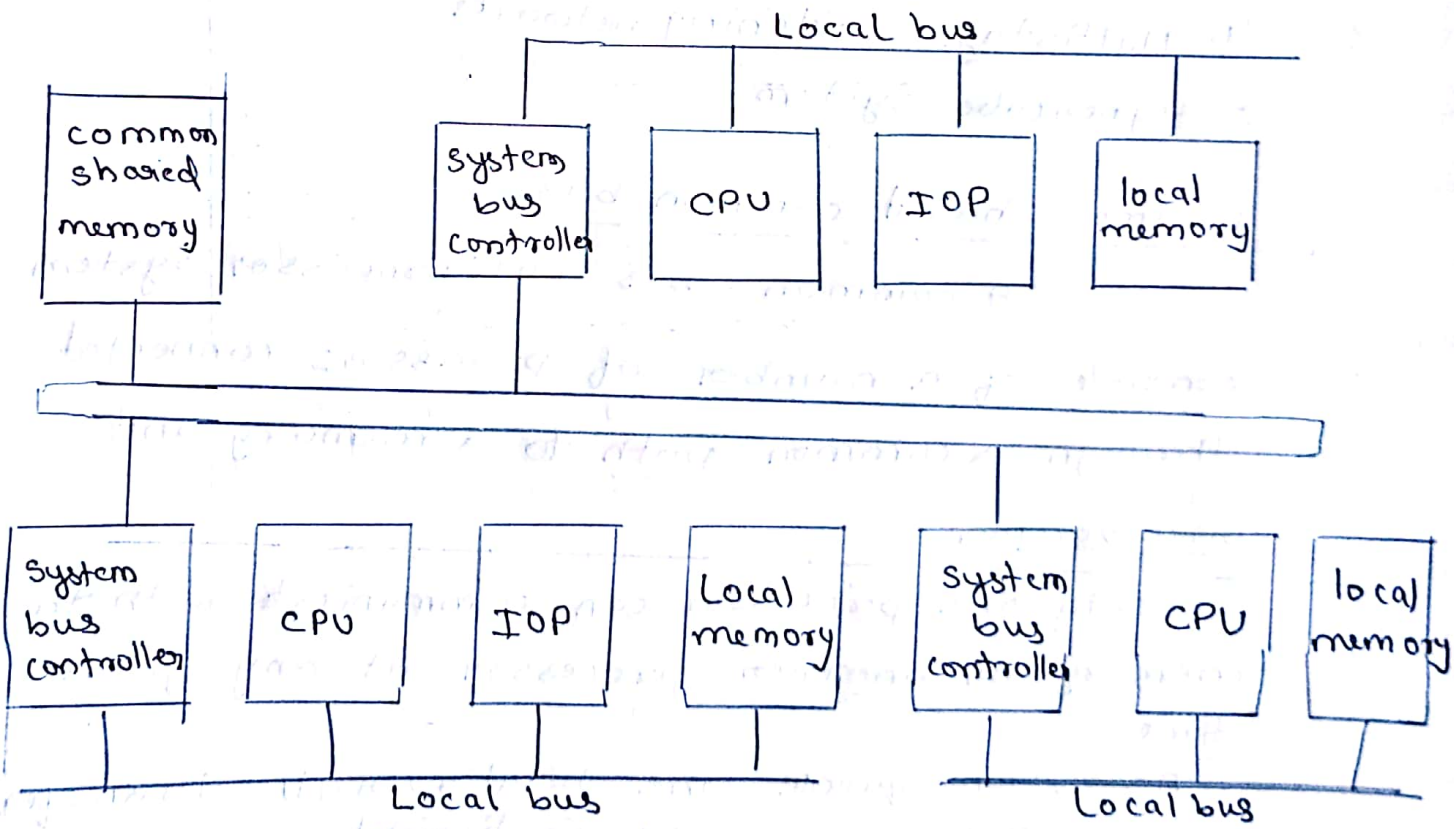
As a consequence, the total overall transfer rate within the system is limited by the speed of the single path

Time shared common bus organization



Below fig is a more economical implementation of a dual bus structure

Part of the local memory may be designed as a cache memory attached to the CPU



System bus structure for multiprocessors

2. Multiport memory

A multiport memory system employs separate buses between each memory module and each CPU.

The module must have internal control logic to determine which port will have access to memory at any given time.

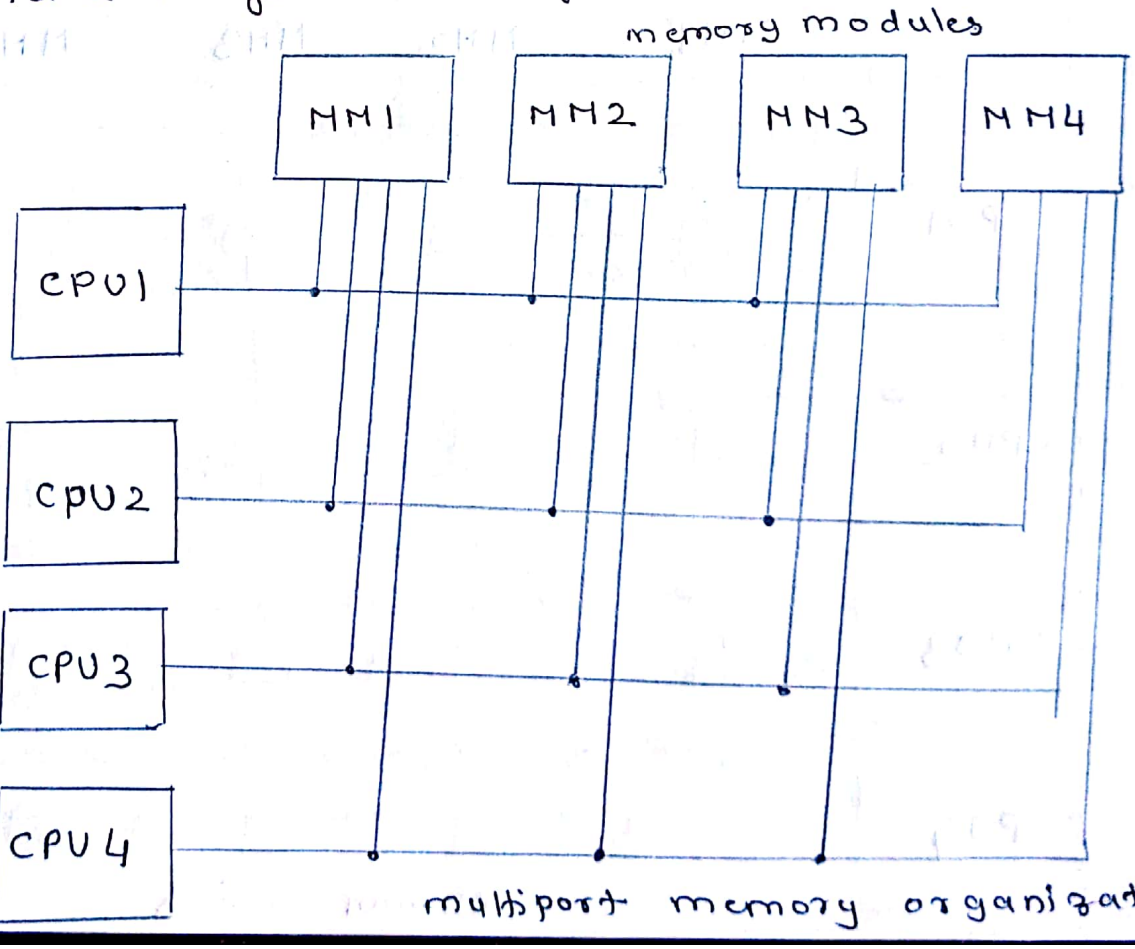
Memory access conflicts are resolved by assigning fixed priorities to each memory port

Adv :-

The high transfer rate can be achieved because of the multiple paths

Disadv ⇒

It requires expensive memory control logic and a large number of cables and connections



multiport memory organization

3. cross bar switch

consists of a number of cross point that are placed at inter sections between processor buses and memory module paths

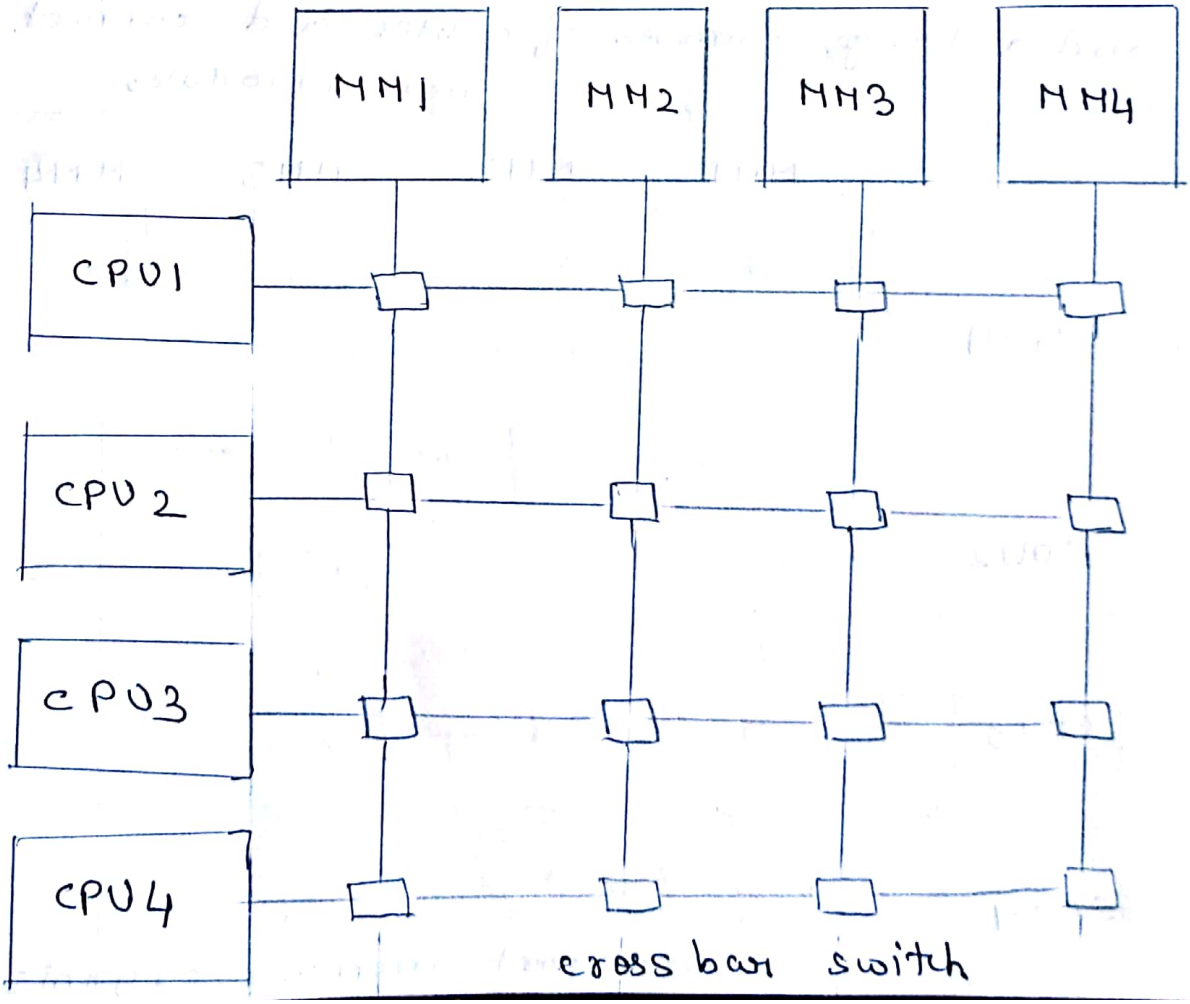
The small square in each crosspoint is a switch that determines the paths from a processor to a memory module.

Advantage →

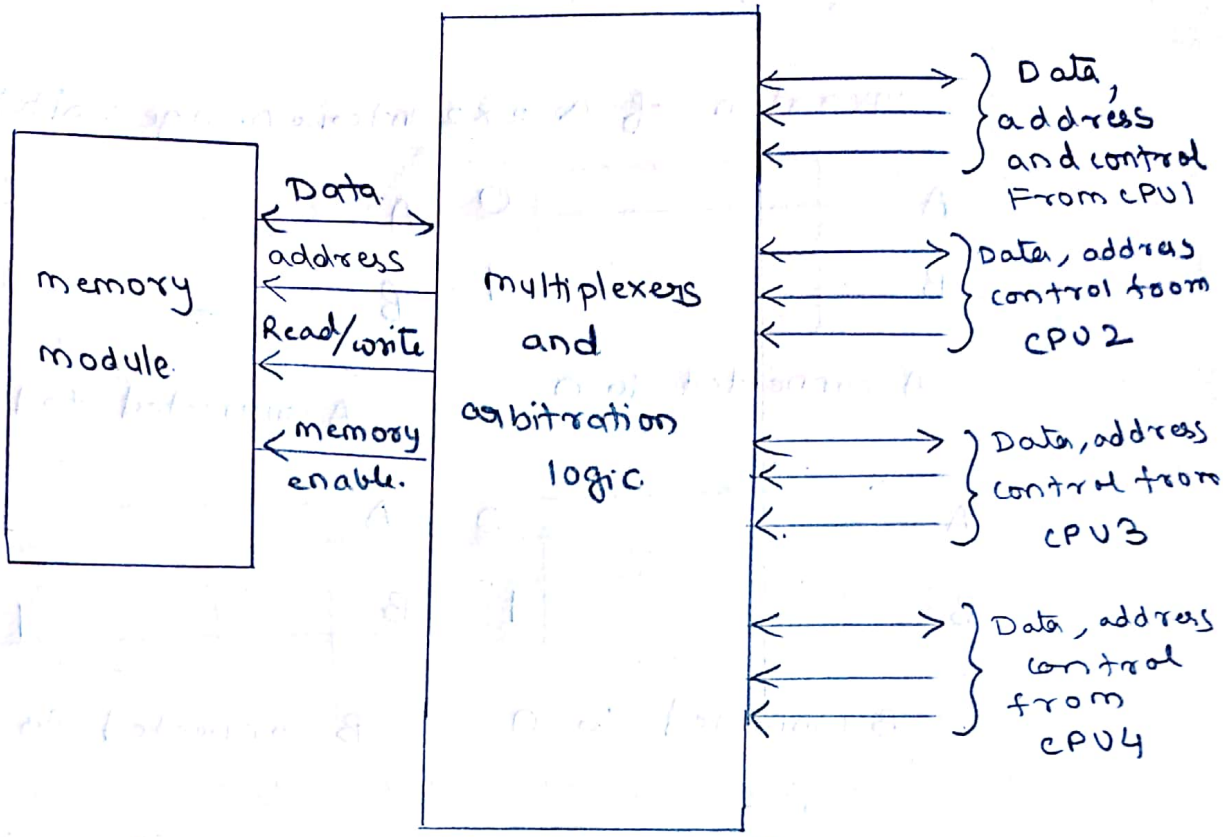
support simultaneous transfers from all memory modules

Disadvantage →

· The hardware required to implement the switch can become quite large and complex



The Functional design of a cross bar switch connected to one memory module



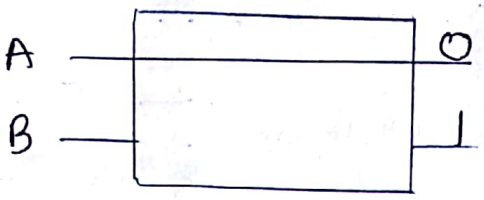
The circuit consists of multiplexers that select the data, address and control from CPU For communication with the memory module. Priority levels are established by the arbitration logic to select one CPU when two or more CPU's attempt to access the same memory. The multiplexers are controlled with the binary code that is generated by a priority encoder within the arbitration logic.

A crossbar switch organization supports simultaneous transfers from all memory module because there is a separate path associated with each module.

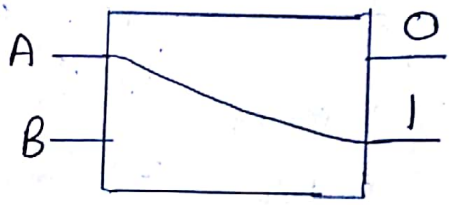
Multistage switching Network

The basic component of a multistage network is a two-input, two-output interchange switch

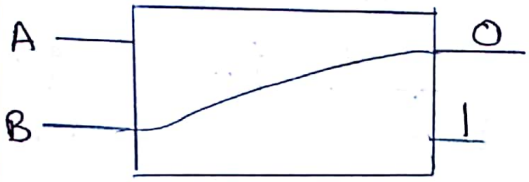
Operation of a 2×2 interchange switch



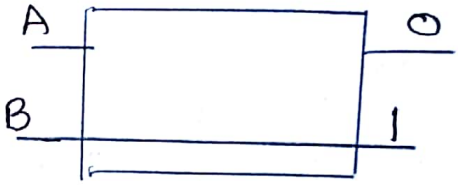
A connected to 0



A connected to 1



B connected to 0



B connected to 1

using the 2×2 switch as a building block, it is possible to build a multistage network to control the communication between a number of source and destination

To see how this is done, consider the binary tree.

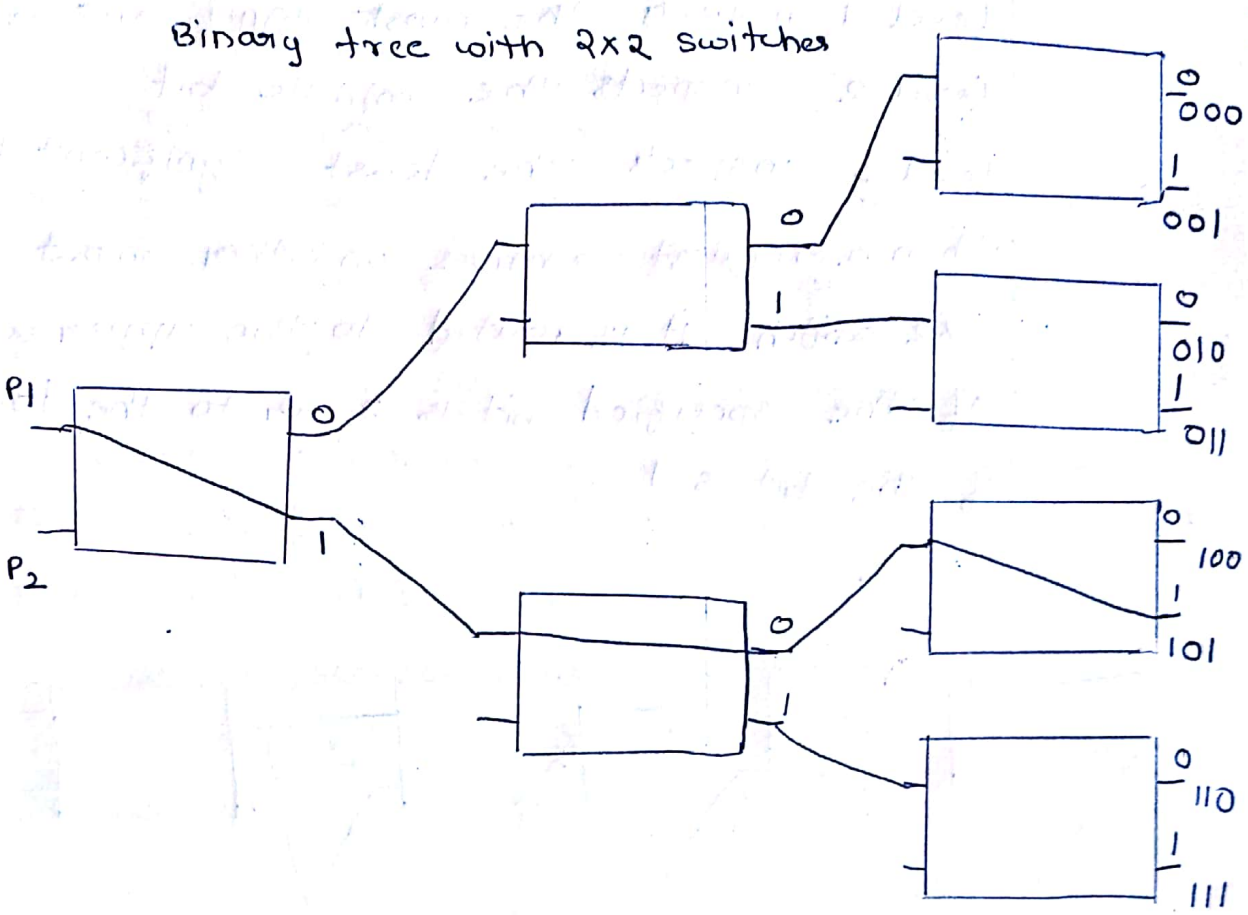
Certain request patterns cannot be satisfied simultaneously .ie, if

$P_1 \rightarrow 000 \sim 011$

$P_2 \rightarrow 100 \sim 111$

The paths from a source to destination is determined from the binary bits of the destination number.

Binary tree with 2x2 switches

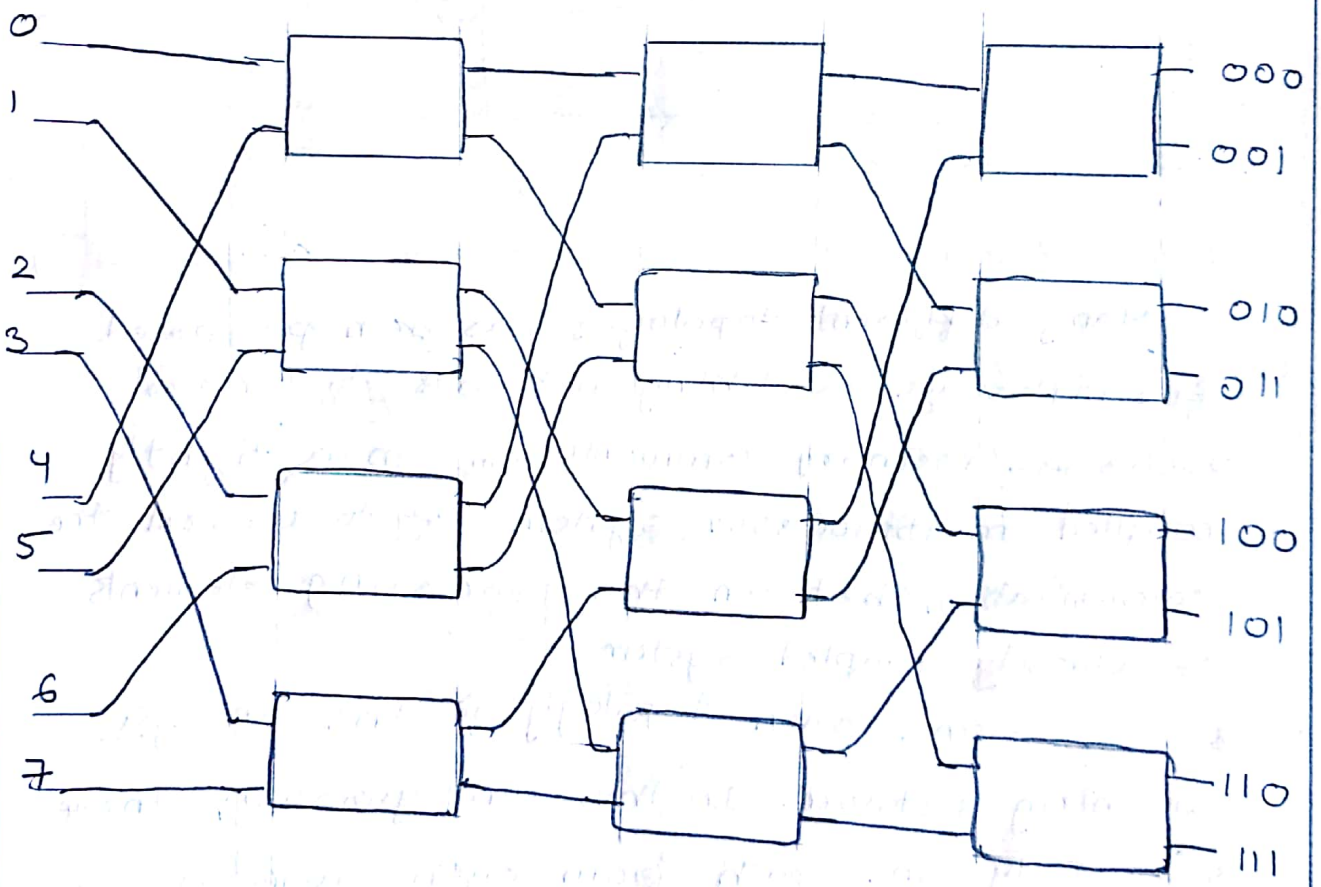


Many different topologies has been proposed for multistage switching networks to control processor-memory communication in a tightly coupled multiprocessor system or to control the communication between the processing elements is aloosely coupled system.

One such topology is the omega switching network. In this configuration, there is exactly one path from each source to any particular destination.

A particular request is initiated in the switching network by the source, which sends a 3-bit pattern represending the destination number. As the binary pattern moves through the network, each level examines a different bit

to determine the 2×2 switch setting.
level 1 inspects the most significant bit
level 2 inspects the middle bit
level 3 inspects the least significant bit.
when a request arrives on either input of the 2×2 switch, it is routed to the upper output of the specified bit is 0 or to the lower if the bit is 1



8×8 omega switching network

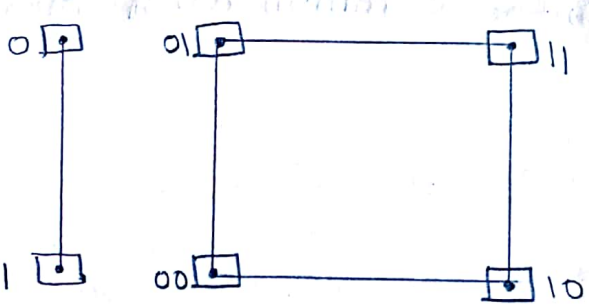
Hyper cube interconnection

Hypercube system

The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of $N = 2^n$ processors interconnected in an n-dimensional binary cube

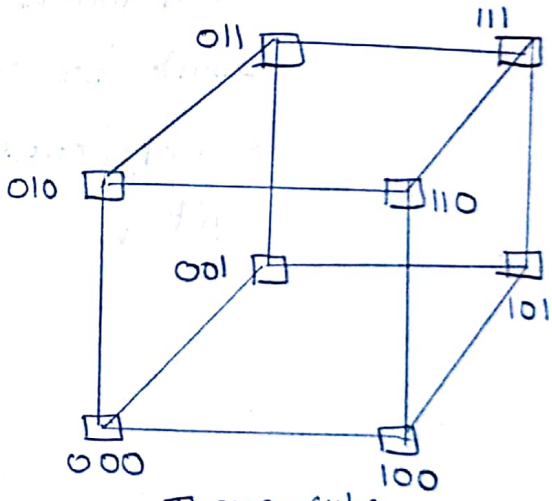
Each processor forms a node of the cube, in effect it contains not only a CPU but also local memory and I/O interface

Each processor address differs from that of each of its n neighbour by exactly one bit position.



one cube

Two - cube.



Three-cube.

the hyper cube structure for n=1, 2 and 3

→ Routing message through a n-cube structure may take from one to n links from a source node to a destination node.

→ A Routing procedure can be developed by computing the exclusive-OR of the source

node address with the destination node address

The message is then sent along any one of the axes that the resulting binary values will have 1 bits corresponding to the axes on which the two nodes differ.

A representative of the hypercube architecture is the intel IPSC computer complex

It consists of 128 (n=7) microcomputers each node consists of a CPU, a floating-point processor, local memory, and serial communication interface unit

Interprocessor Arbitration

Arbitration logic resolves bus conflict

It would be the part of system bus controller

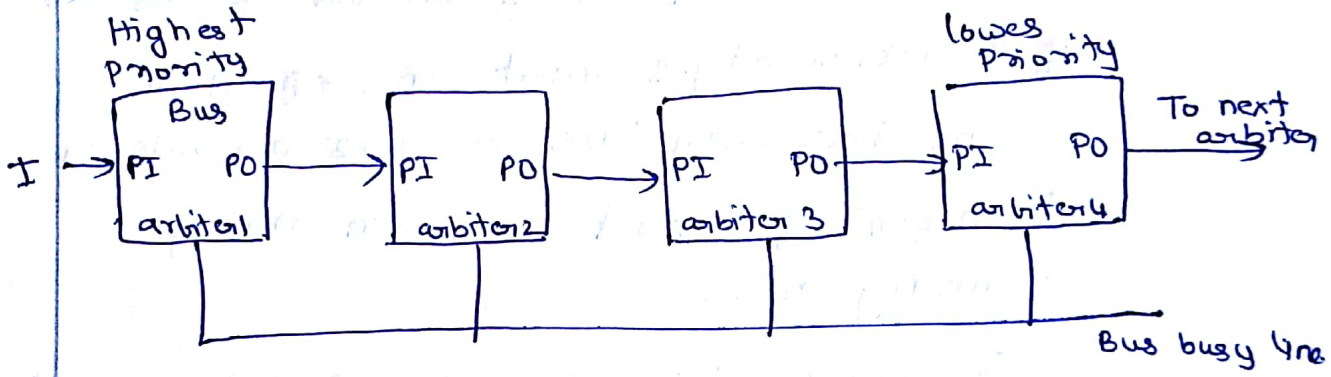
two types

→ serial arbitration procedure.

→ parallel arbitration procedure

Arbitration procedure services all processor request on the basis of established priorities

serial arbitration procedure



serial (daisy-chain) arbitration

Each processor has its own bus arbiter logic with priority-in and priority-out line.

The priority out (PO) of each arbiter is connected to the priority in (PI) of the next-lower priority arbiter.

when arbiter receives PI=1, PO=0 it activates the bus

when arbiter receives PI=0, PO=0 inactive to bus

Parallel arbitration logic

The parallel bus arbitration technique uses an external priority encoder and a decoder.

Each bus arbiter in the parallel scheme has a bus request output line and a bus acknowledge input line

Each arbiter enables the request line when its processor is requesting access to the system bus

The processor takes control of the bus if it acknowledges input line enabled.

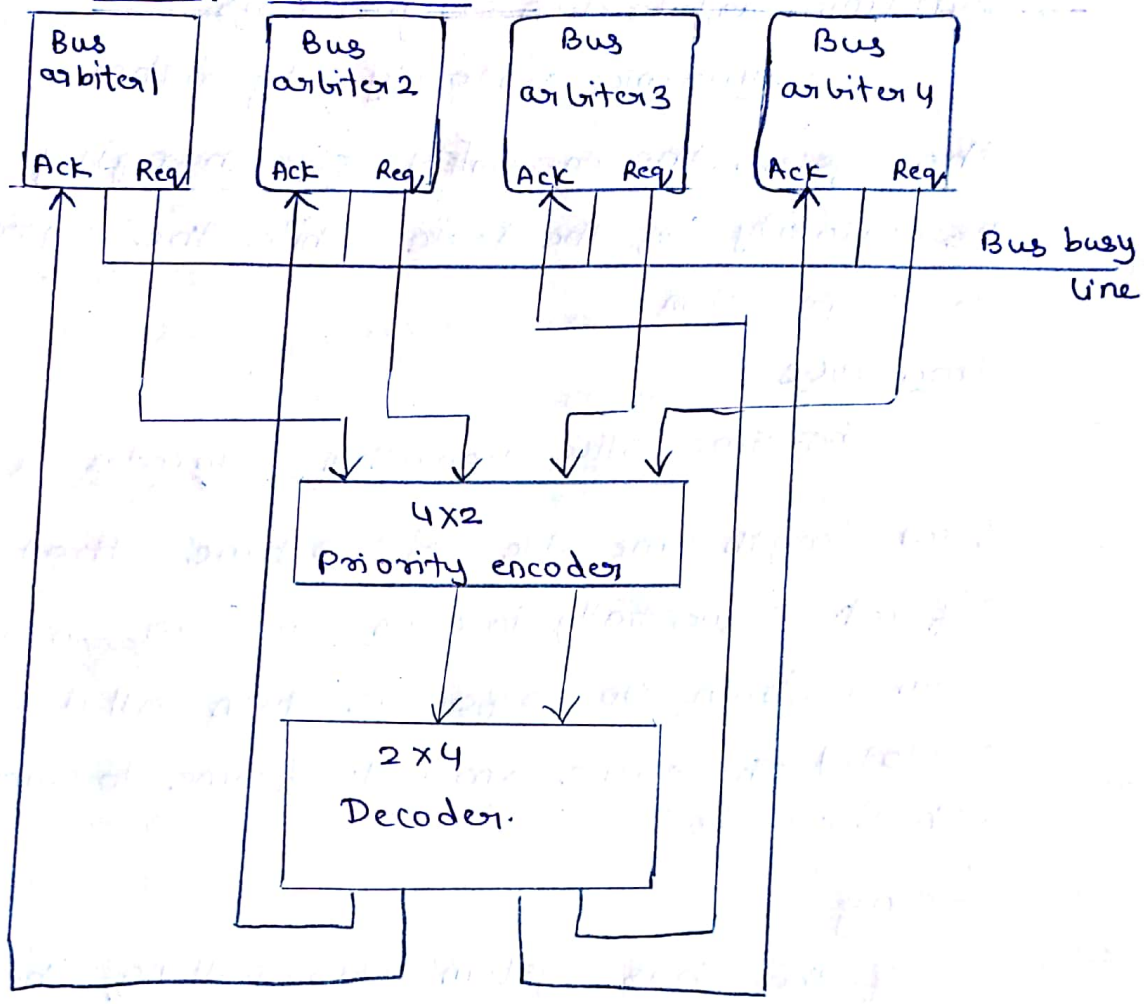
The bus busy line provides an orderly transfer of control as in the Daisy chaining case.

Generally, the figure shows the request line from four arbiters going into a 4x2 priority encoder

The output of the ender generates a 2-bit code which represents the highest-priority unit among those requesting the bus.

The bus priority-in (BPRN) and bus priority-out (BPRO) used for a daisy-chain connection of bus arbitration circuits

Parallel arbitration



The bus busy signal BUSY an open-collector output used to instruct all arbiters when bus is busy conducting a transfer.

The common bus request (CBRQ) is also an open-collector output which serves to instruct the arbiter if there are any other arbiters of lower-priority requesting use of the system.

The signals used to construct a parallel arbitration procedure are bus request (BREQ) and priority-in (BPRN), corresponding to the request and acknowledgement signals.

The bus block (BCLR) used to synchronize all bus arbitration.

Dynamic arbitration Algorithms

Dynamic priority algorithm gives the system the capability for changing the the priority of the device while the system is in operation

Time slice

The time slice algorithm allocates a fixed-length time slice of bus time that is offered sequentially to each processor, in round robin fashion. No preference given all device is alloted the same amount of time to communicate with the bus

Polling

If the bus system uses polling, the bus grant signal is replaced by a set of lines called poll lines which are connected to all units. These lines are used by the bus controller ~~sequence~~ to define an address for each device connected to the bus The polling is programmable, the selection of priority can be changed

LRU

The least recently used (LRU) algorithm gives the highest priority to the requesting device that has not used the bus for the

Longest interval

FIFO

In the first-come, first-serve scheme, request are served in the order received. Bus controller establishes a queue. Each processor must wait for its turn to use the bus ~~requests~~ on a first-in, first-out (FIFO) basis

Rotating daisy-chain

The rotating daisy-chain prodedure is a dynamic extension of the daisy chain algorithm same as serial (daisy-chain) arbitration. Here once an arbiter releases the bus, it has the lowest priority

Interprocessor communication and Synchronization

→ shared memory
→ communication is by common area of shared memory
→ sender alert the receiver by interrupt and data transfer through I/O path

Loosely coupled →
communication is through I/O channels
one calls a procedure which resides in the memory of the destination
operating systems in each node controls the communication

Interprocessor synchronization

The instruction set of a multiprocessor contains basic instruction that are used to implement communication and synchronization between cooperating processes. communication refers to the exchange of data between different process

synchronization is needed to enforce the correct sequence of processes and to ensure mutually exclusive access to shared writable data.

Multiprocessor systems usually include a no. of mechanisms to deal with the synchronization of resources

one of the most popular methods is through the use of a binary semaphore.

Mutual exclusion with a semaphore

A properly functioning multiprocessor system must provide a mechanism that ~~gives~~ will guarantee orderly access to shared memory and other shared resources

This is necessary to protect data from being changed simultaneously be two or more processors

This mechanism has been termed mutual exclusion

Mutual exclusion enables one processor to exclude or lockout access to a shared resource by other processor when it is in a critical section

A critical section is a program sequence, that once begun, must complete execution before another processor accesses to same shared resources.

A binary variable called semaphore is is often used to indicate whether or not a processor is executing a critical section

It is a software controlled flag that is stored in a memory location that all processor can access. when the semaphore is equal to 1, it means that a processor is executing critical program, so that the shared memory is not available to other processor. when the semaphore is equal to 0, the shared memory is available to any requesting processor.

If a processor accesses shared memory and is executing a critical section it sets the semaphore and clears it when it is finished.

Testing and setting the semaphore is itself a critical ~~section~~ operation and must be performed as a single individual operation.

A semaphore can be initialized by means of a test and set instructions in conjuction with a hardware lock mechanism.

The hardware lock is a processor generated signal that

Serves to prevent other processors from using the system bus as long as the signal is active.

The test and set instruction tests and sets a semaphore and activates a lock mechanism during the time that instruction is being executed. This prevents other processors from changing the semaphore between the time that the processor is testing it and the time that it is setting it.

Assume that the semaphore is a memory location symbolized by SEM.

Let the mnemonic TSL designate "the test and set while locked" operation.

The instruction TSL SEM performs

R ← M [SEM] test semaphore

M [SEM] ← 1 set semaphore

The value in R determines what to do next. The last instruction in the program must clear SEM to 0 (zero) to release the shared resource to other processor.