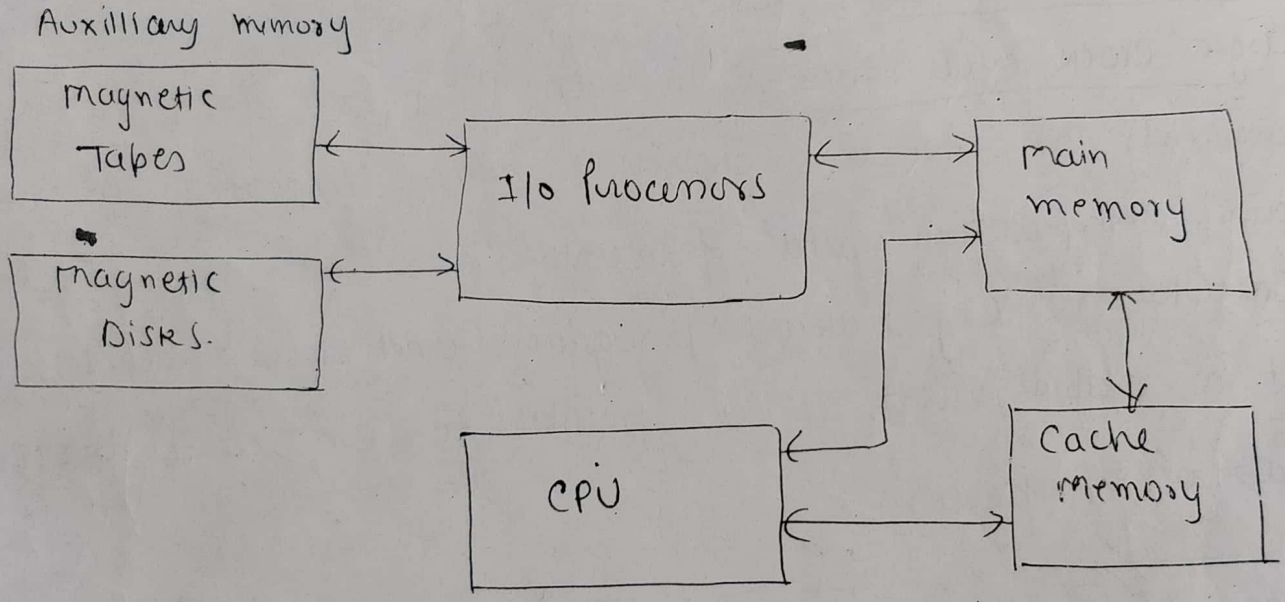
MODULE - IV Memory & Intra System Comm. & I/O Organization

Memory Hierarchy

Memory unit is essential component of computer since it is needed to storing data and programs. The ~~main~~ memory unit that communicate directly with CPU is called the main memory. Devices that provide backup storage are called auxilliary memory. Most common auxillary memory devices used in computer systems are magnetic disks and tapes.



[Memory Hierachy in Computer System].

Total memory capacity of computer can be consists of all Storage devices employ in computer system from slow but hight speed auxilliary memory to a relatively faster main memory, to an even smaller and faster cache memory accenible to high speed procaning logic.

Cache memory → A special, very-high speed memory cache is used to increase speed of procuring by making current programs and data available to CPU at rapid rate.

CPU logic is usually faster than main memory accen time, with the result that procuring speed is limited primarily by speed of main memory. A technique used to compensate for mismatch in operating speed is to employ an extremly fast, small cache between the CPU & main memory whose accen time is closer to procumer logic clock cycle time. Cache is used to for storing segments of programs currently being executed in CPU and temporary data frequently needed in present-calculation. By making programs and data available at a rapid rate, it is possible to increase performance rate of computer.

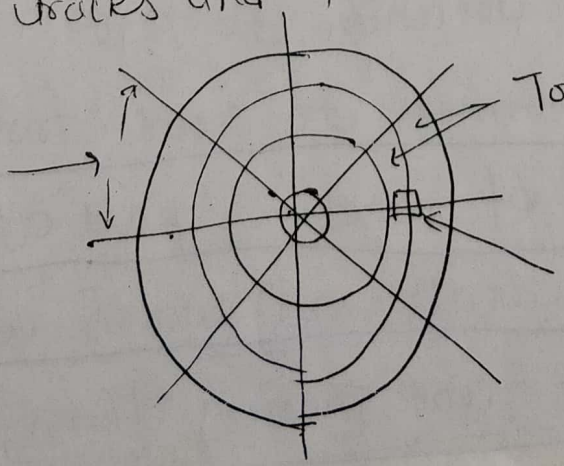
Auxillarity memory → : Most common auxillary memory devices used in computer system are magnetic tapes and disks. Important characteristics of these devices are checked by its accen time, accen mode, transfer rate capacity and cost.

The average time required to reach a storage location in memory and obtain its contents is called accen time.

In electromechanical devices with moving parts such as disks and tapes, the access time consists of seek time required to position of read-write head to a location & a transfer time required to transfer data to or from the device. Because seek time is usually much longer than transfer time, auxiliary storage is organized in records or blocks.

(1) Magnetic Disks:

A magnetic disk is a circular plate constructed of metal or plastic coated with magnetic material. Often both sides of disks are used and several disks may be stacked on one spindle with read/write heads available on each surface. All disks ~~the~~ rotate together at high speed and are not stopped or started for access purpose. Bits are stored in magnetic surface in spots along concentric ~~Sectors~~. ~~Circle~~ circles called tracks. The tracks are commonly divided into sections called sectors. In most systems, minimum quantity of information which can be transferred is a sector. Subdivision of one disk surface into tracks and sectors is shown as:-



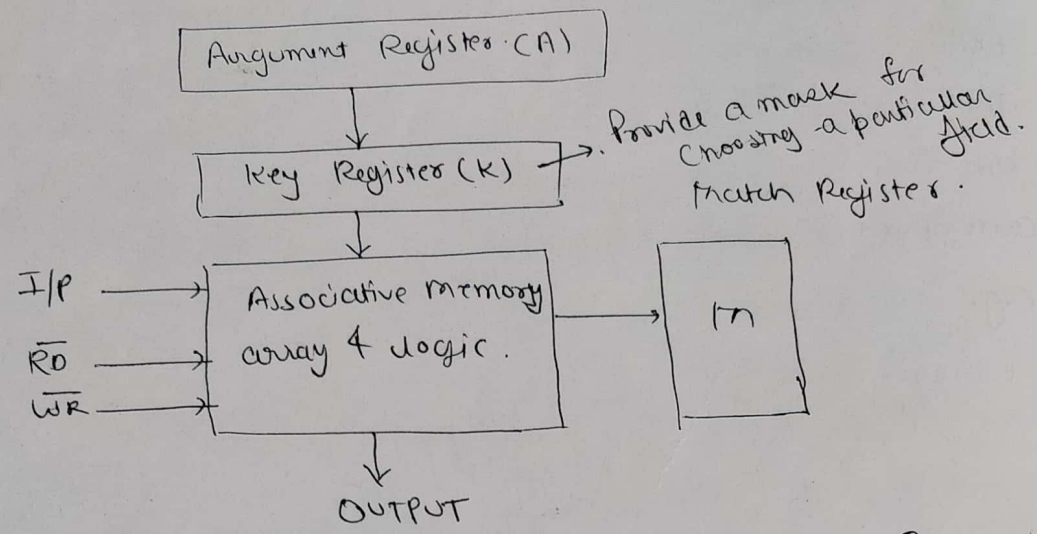
(2) Magnetic Tapes →

A magnetic tape transport consists of the electrical, machanical 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 magnetic spots on the tape along several tracks. Usually, seven or nine bits are recorded simultaneously to form a character together with parity bit. Read/write heads are mounted one in each track so that data can be recorded and read as sequence of characters.

— × — × — × — × — × —

Associative Memory →: A memory unit accessed by content is called an associative memory or content addressable memory (CAM). This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location. When a record is written in an associative memory, no address is given. The memory is capable of finding an empty unused location to store the word. When a word is ~~written~~ to read from an associative memory, the content of word or part of word is specified. Memory locates all words which match the specified content and marks them for reading.

Hardware organization of Associative Memory →



[Block Diagram of Associative Memory]

// Block diagram of associative memory is shown above. It consits of a memory array and logic for ~~m words and~~ n-words with n-bits per words. The argument register (A) and key register (K) each have n-bits, one for each bit of a word. The match register has m-bits, one for each memory word.

Each word in memory is compared in parallel with the content of argument register. The words that match the bits of the argument register set a corresponding bit in a match register. After the matching proces, those bits in the match register that have been set indicate the fact that their corresponding words have been matched.

The key register provide a mask for choosing a particular field or key in the argument ~~field~~ word. The entire argument 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 key register are compared.

e.g. Suppose we have following values of A, K and words:-

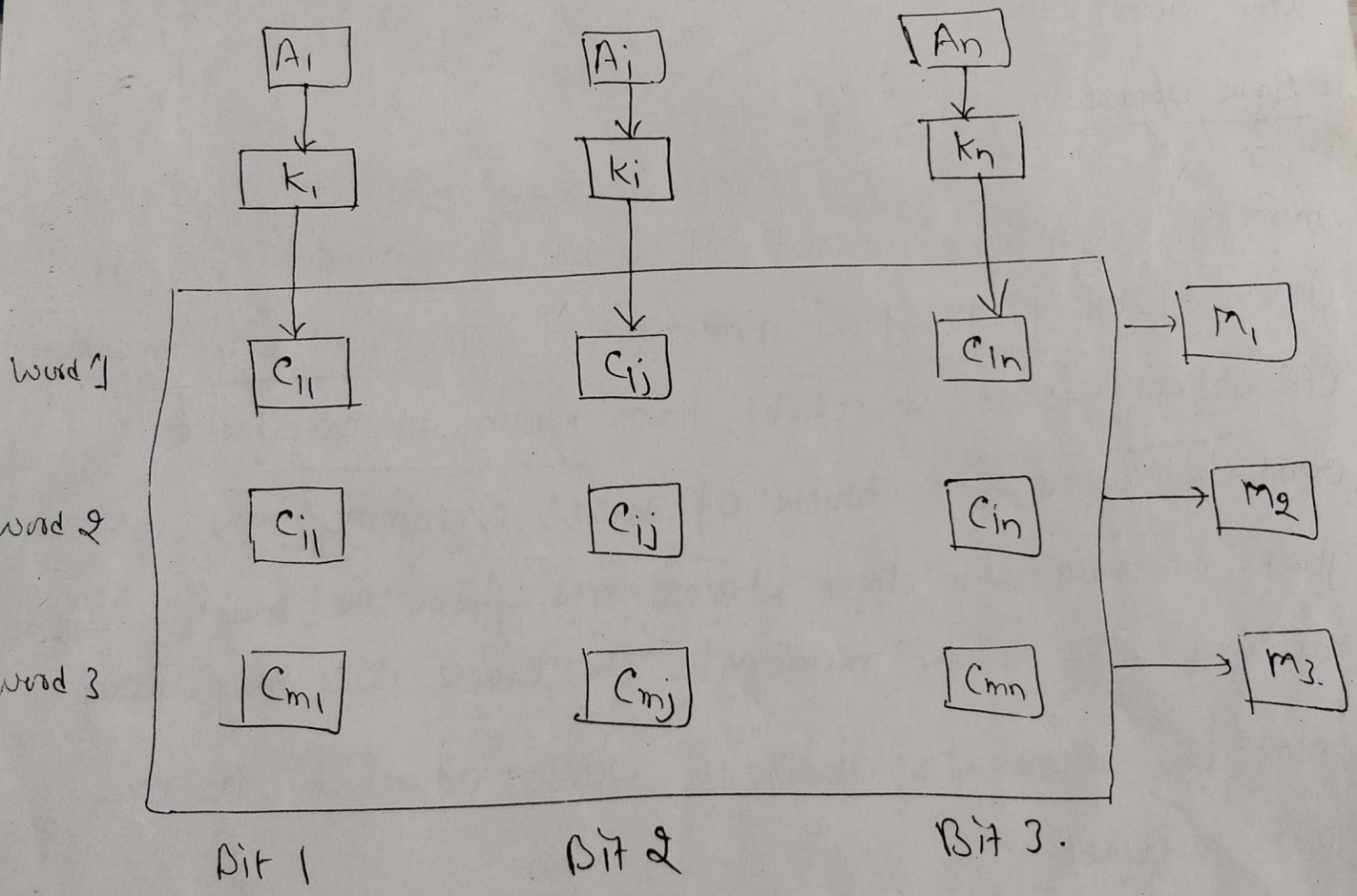
A	101	111100
K	111	000000 → for masking of bits.
Word 1	100	111100 → No match.
Word 2	101	000001 → match.

Only the ~~ther~~ three leftmost bits of A are compared with memory words because K has 1's in these 3 positions. Word 2 matches the unmask argument field because the three leftmost bits of argument & word are equal.

Relation between memory array and External Registers:

Relation between memory array and external registers in an associative memory is shown in fig (a). The cells in array are marked by letter 'C' with two subscripts. The first subscript gives the word number and second specify the bit position in word. Thus cell Cij

is the cell for bit j in word i. A bit Aj in the argument register is compared with all the bits in column j of the array provided that $K_j = 1$. This is done for all columns $j = 1, 2, \ldots, n$. If a match occurs between all the unmasked bits of argument and the bits in word i, the corresponding bit M_i in the match register is set to 1. If one or more unmasked bits of argument and word do not match, M_i is cleared to 0.



Cache Memory:

When active portion of program and data are placed in a fast small memory, the average

memory access time can be reduced, thus reducing the total execution time of program. Such a fast small memory is referred to as cache memory. The cache memory access time is less than access time of main memory by factor of 5 to 10.

The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in fast cache memory, the average memory access time will approach the access time of cache.

Basic operation on cache

When the CPU needs to access memory, cache is examined. If word is found in cache, it is read from fast memory. If the word addressed by CPU is not found in cache, main memory is accessed to read the word. A block of words containing the one just accessed is then transferred from the main memory to cache memory. The block size may vary from one word to about 16-words adjacent to one just accessed.

Performance of cache memory

Performance of cache memory is frequently measured in terms of quantity called hit

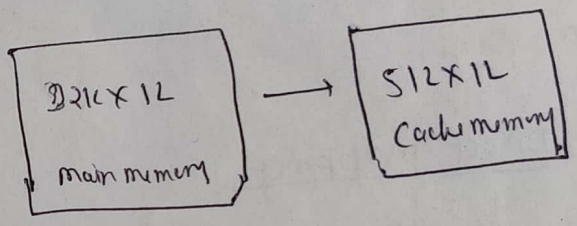
ratio. When the CPU refers to memory and finds the word in main memory cache, it is said to be hit. If the word is not found in cache, it is in main memory and it counts as a miss.

"The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the hit ratio."

Mapping Process: The transformation of data from main memory to cache memory is refferred to as a mapping process.

Following are three mapping process:→

1. Associative mapping.
2. Direct mapping
3. Set-associative mapping.



① **Associative ~~memory~~ mapping:** Associative mapping stores both the address and data of memory word. This permits any location in cache to store any word from main memory.

CPU address - 15 bit.

Argument Reg.

← address →	← Data →
01000	3450
021007	6710
01234	1234
!	!

(Associative mapping cache, all number in Octal)

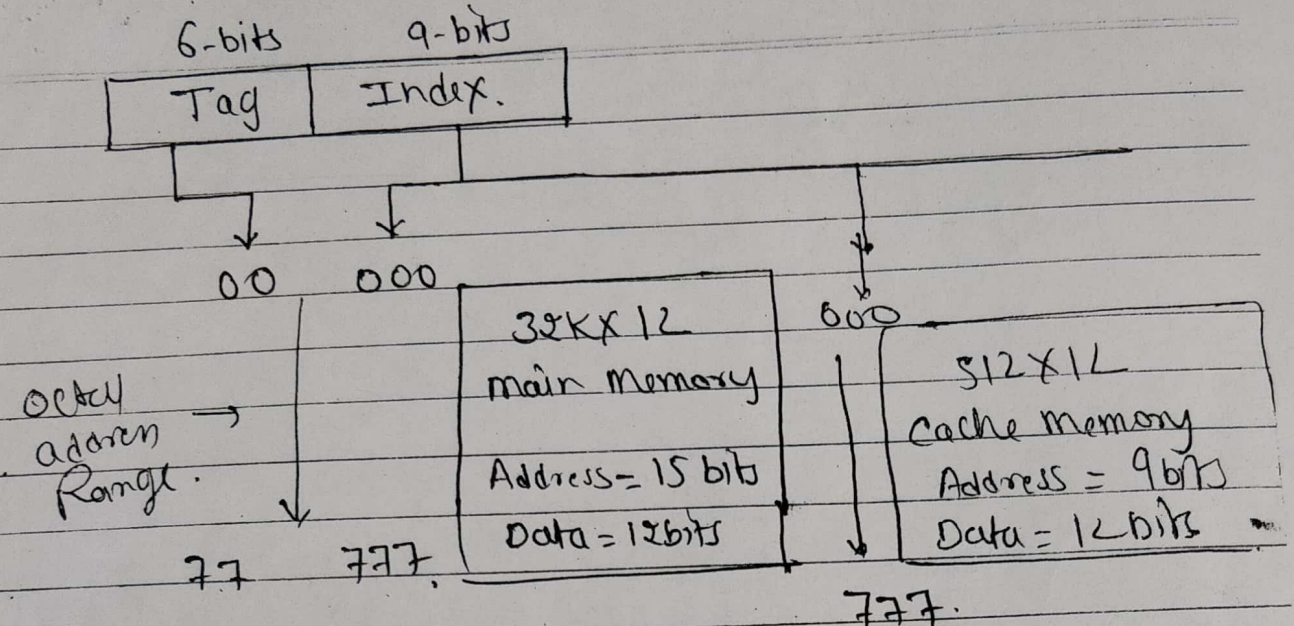
Diagram shows the three words stored in cache. The address value of 15-bit CPU address is placed in the argument register and ~~associative memory and the~~ associative memory is searched for ~~the~~ matching address. If address is found, corresponding 12-bit data is read and sent to CPU. If no word ~~is~~ found, main memory is accessed for the word. Address - data pair is then transferred to associative ~~memory~~ cache memory.

If cache is full, ~~the~~ an address - data pair must be ~~placed~~ displaced to make room for a pair that is need and not presently in cache.

(b) Direct mapping:-

In direct mapping, CPU address of 15-bits is divided into two fields. The 9-least significant bits ~~consists of~~ constitute the index field. and remaining six bits form the tag field.

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



(Addressing relationship between main & cache memory)

Internal organization of words in cache memory is shown as:-

address	Memory Data.
00000	1220
00777	2340
01000	3450
	⋮
02000	5670
02777	6710

(a) main memory.

Index address	Tag	Data
000	00	1220
777	02	6710

(b) cache memory.

When a new word is first brought into the cache, the tag bits are stored alongside the data bits. When CPU generate a memory request, the index field is used for address to access the cache. The tag field of the CPU address is compared with the tag in the word read from cache. If two tags are match there is hit, and desired data word is in cache. If there is no match, desired word is then accessed from main memory.

→ Disadvantage of direct mapping is that the hit ratio can drop considerably if two or more words

whose addresses have the same index but different tags are accessed repeatedly.

Example, see in last figure. Suppose the word at address zero is presently stored in cache (index=000 tag=00, data=1220) Suppose that the CPU now wants to access the word at address 02000. Index address is 000, so it is used to access the cache. The two tags are then compared. Cache tags is 00 and addrss tag is 02., which does not match. Therefore main memory is accessed and data word 5670 is transferred to CPU. Cache word at index address 000 is then replaced with tag of 02 and data of 5670.

Set associative memory:

This is improvement over the direct mapping. In this mapping, ~~cache can store~~ each word of the cache can store two or more words of memory under same index value.

Each tag requires = 6 bits

Each data word = 12 bits.

So word length = 2(6+12) = 36 bits.

An index address of 9-bits can accomodate = 512 words.

So, total size of cache memory = 512 x 36 = 18432 bits.

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

[Two-way set associative mapping cache).

Disadvantages 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. So the third type of cache organization, called the Set-Associative mapping, is improvement over the direct mapping organization is that each word of cache can store two or more words of the memory under the same index address.

Each index address in the figure refers to two data words and their associated tags. Each tag requires six bits and each data word has 12-bits, so the word length is 2(6+12) = 36 bits. An index value of 9-bits can accommodate 512 words. Thus the size of cache memory is 512×36.

The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000. Similarly, the words at addresses 02777 and 00777 are stored in cache at index address 777.

When the CPU generate the memory request, the index value of the address is used to accen the cache. The tag field of the CPU address is then compared with both tags in cache to determine if a ~~cache~~ match occurs.

The hit ratio will improve as the set size increases because more words with the same index but different tag can reside in cache. However, an increase in set size increases the number of bits of words of cache and requires more complex comparision logic.

Virtual memory:

Virtual memory is a concept that permits the user to construct programs as though a large memory space are available, equal to the totality of the auxilarty memory. Each address that is referenced by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory...

"Virtual memory is used to give programers the illusion that they have a very large memory ~~used the give~~ at their disposal even though the computer actually has a relatively small main memory".

Memory management Hardware:

"A memory management system is a collection of hardware and software procedure for managing the various programs residing in memory".

The basic components of a memory management unit are:-

(1) A facility for dynamic storage relocation that maps logical memory references into physical memory addresses.

(2) A provision for sharing common programs stored in memory by different users.

(3) Protection of information against unauthorized accen between users and preventing users from changing operating system functions.

- Dynamic storage relocation hardware is a mapping procedure from logical state to physical state. It is more convenient to divide programs and data into logical parts called the segments. A segment is a set of logically related instructions or data elements associated with a given name". Segments can be generated by the user or can be by the operating system. Examples of ~~set~~ Segments are subroutines, an array of data, a table of symbols or a user's program.

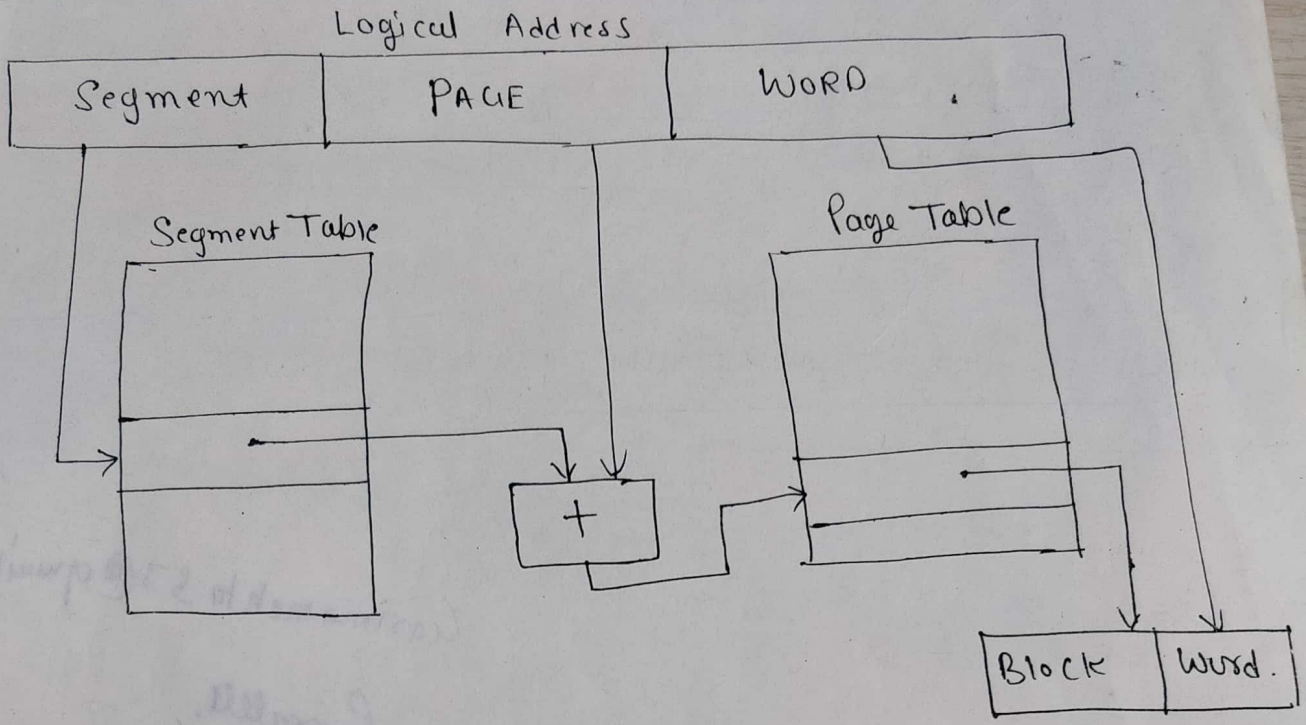
* Sharing of common programs is an integral part of multiprogramming system. For example several users wishing to compile their fortran programs should be able to share a single copy of compiler rather than each user having a separate copy ~~of~~ in memory.

* Third issue in multiprogramming is protecting one program from unwanted interaction with another. An example is one user's unauthorized coping of another user's program.

LOGICAL TO PHYSICAL ADDRESS MAPPING (Segmented-Page mapping)

The address generated by a segmented program is called the logical address.

[illegible] figure shows how the logical to physical address mapping is being performed:-



[Logical to Physical Address mapping]

Logical address is partioned into three fields. The Segment field specify the segment number, page field specify the page within the segment and word field gives the word within the page. The segment number may be anociated with just one page or with more than one page.

mapping of the logical address into the physical address is done by means of two tables. The

(1) Segment number of the logical addren specifies the address for the segment table. The entry in the

Segment table is a pointer address for a page table base. The page table base is added to page number given in the logical address. The sum produces a pointer address to an entry in page table. The value found in the page table provides a block number in the physical memory.

Combinaly both block field and word field produces the final physical address.

Garimamehta53@gmail.com

Parallel.

Flynn classification

Parallel processing can be classified as:-

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 procen into suboperations, with each subprocen being executed in special dedicated Segment that operates concurrently with all other segments.

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

Suboperations are:-

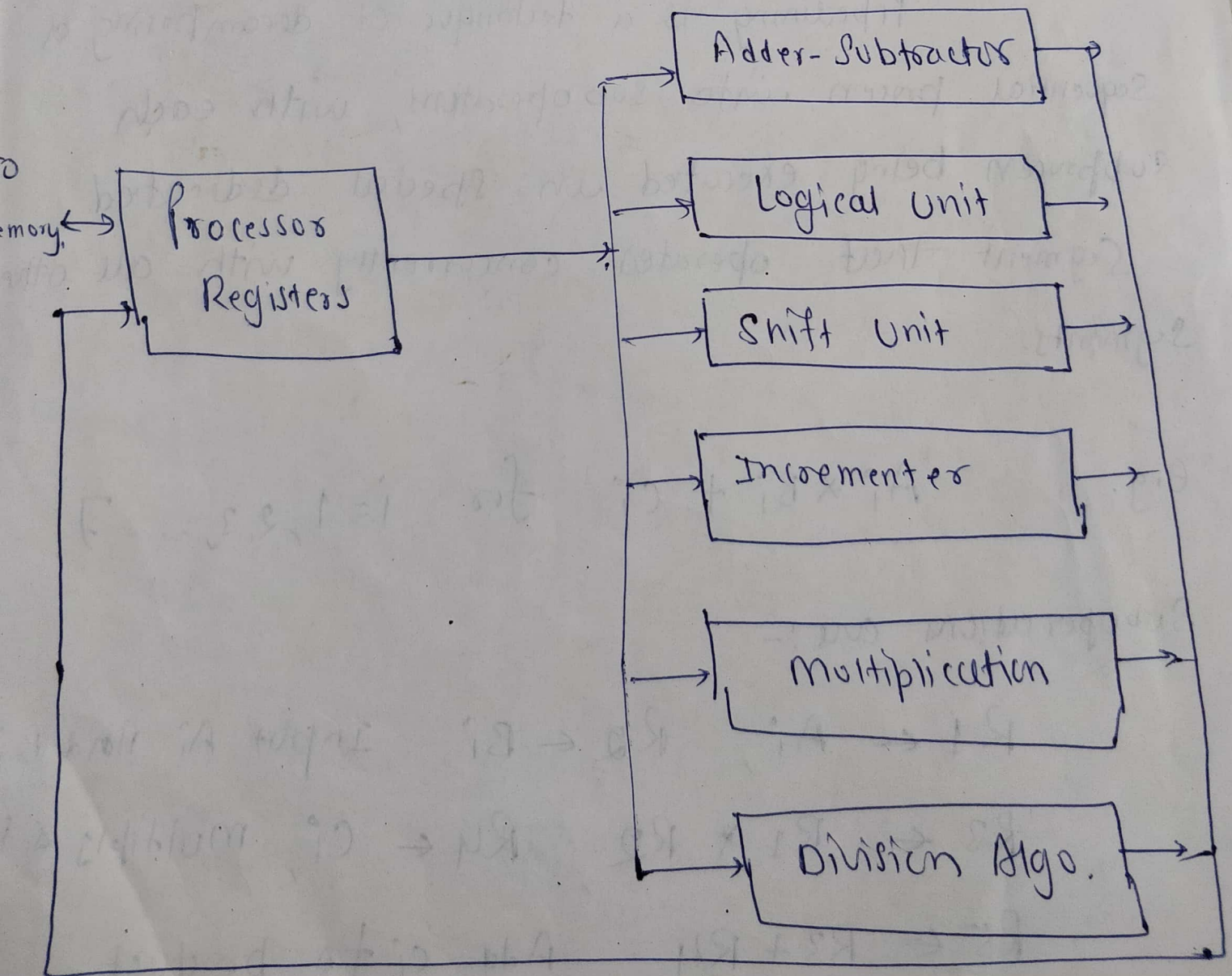
$R1 \leftarrow A_i$, $R2 \leftarrow B_i$ Input A_i And B_i.

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

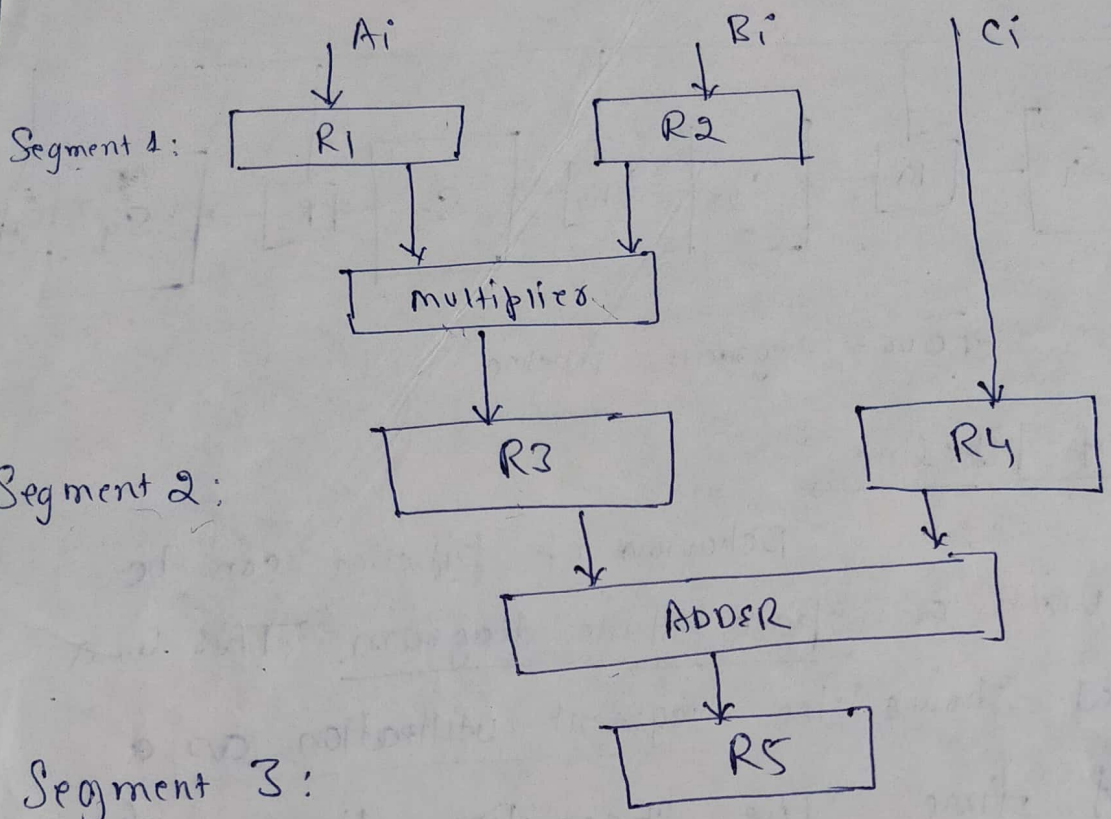
$R5 \leftarrow R3 + R4$, Add C_i to product.

Parallel Processing:

Parallel processing is a term used to denote a large class of techniques that are used to provide simultaneous data processing tasks for purpose of increasing the computational speed of computer system. Parallel processing also increased the throughput i.e. amount of processing that can be accomplished during a given interval of time.



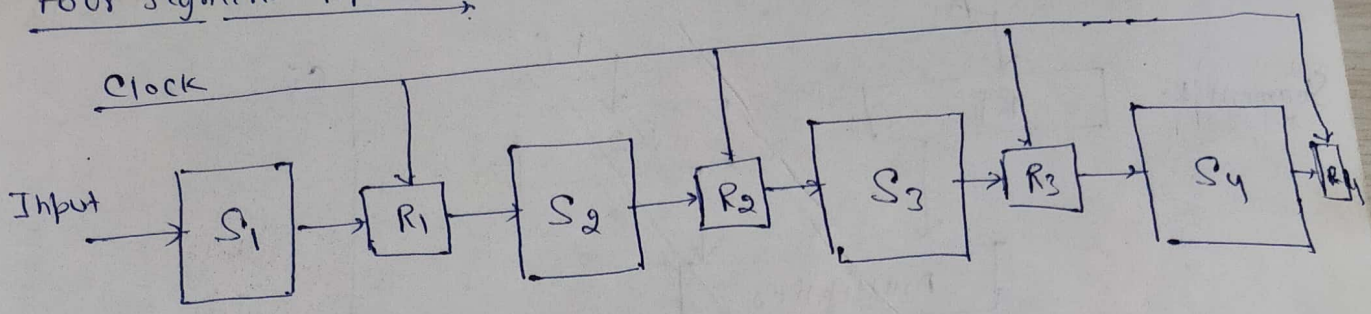
14(ii)



Content of register in pipeline

clock pulse Number	Segment 1		Segment 2		Segment 3
	R1	R2	R3	R4	R5
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.	—	—	—		$A_7 * B_7 + C_7$

Four segment Pipeline



Four- Segment Pipeline

Behaviour of pipeline

Behaviour of pipeline can be illustrated with a space-time diagram. This is a diagram that shows the segment utilization as a function of time. The space time diagram for 4- segment pipeline is as below:-

Segment ↓ / Clock cycles →	1	2	3	4	5	6	7	8	9
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

they will be completed after a time = $(n-1)tp$

and for n- tasks using le-segments we require.

Total no. of clock cycle = $k+(n-1)$

So, in above example,

total segments are, $k=4$.

no. of tasks. $n=6$.

time require to complete all operation = $k+(n-1)$

Clock cycles required = $= 4+(6-1)$

$= 9$ clock cycle.

* Now lets consider the non-pipeline unit that perform the same operation, and takes a time equal to tn to complete each task. Total time required for n-tasks is ~~ntp~~. ntn. so, speed up of pipeline procesing over an equivalent non-pipeline procesing is defined as:-

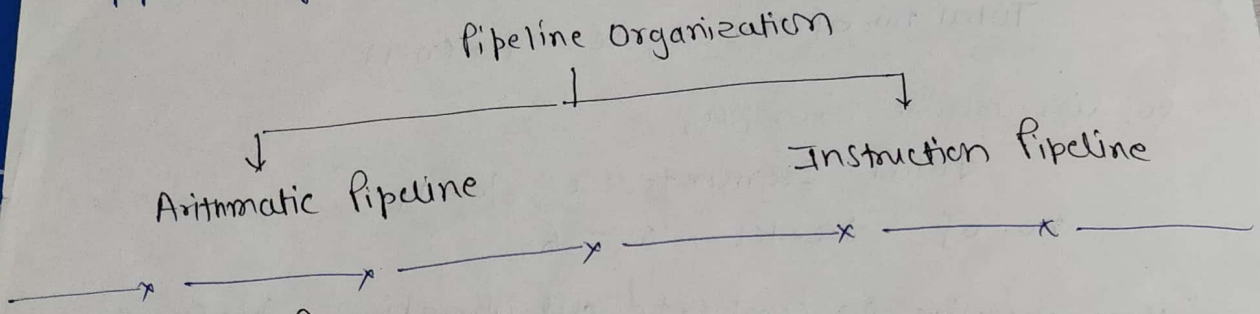
$$\text{Speed-up } S = \frac{ntn}{[k+(n-1)]tp}$$

(There are two areas of computer design where the pipeline organization is applicable.

(1) An arithmatic pipeline → divide arithmatic operation into sub-operation for execution in pipeline segments.

(2) Instruction pipeline →: operates on stream of instructio by overlaping the fetch, decode & executed phases.)

There are two areas of computer design where the pipeline organization is applicable.



Arithmatic Pipeline:- [Arithmatic pipeline, ~~is a very high speed~~ units are usually found in very high speed computers. They are used to implement floating point ~~no~~ operations, multiplication of fixed-point numbers and etc.

* Floating point numbers are usually decomposed into suboperations.

Exp. Suppose we have 2- normalized floating point binary numbers:

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

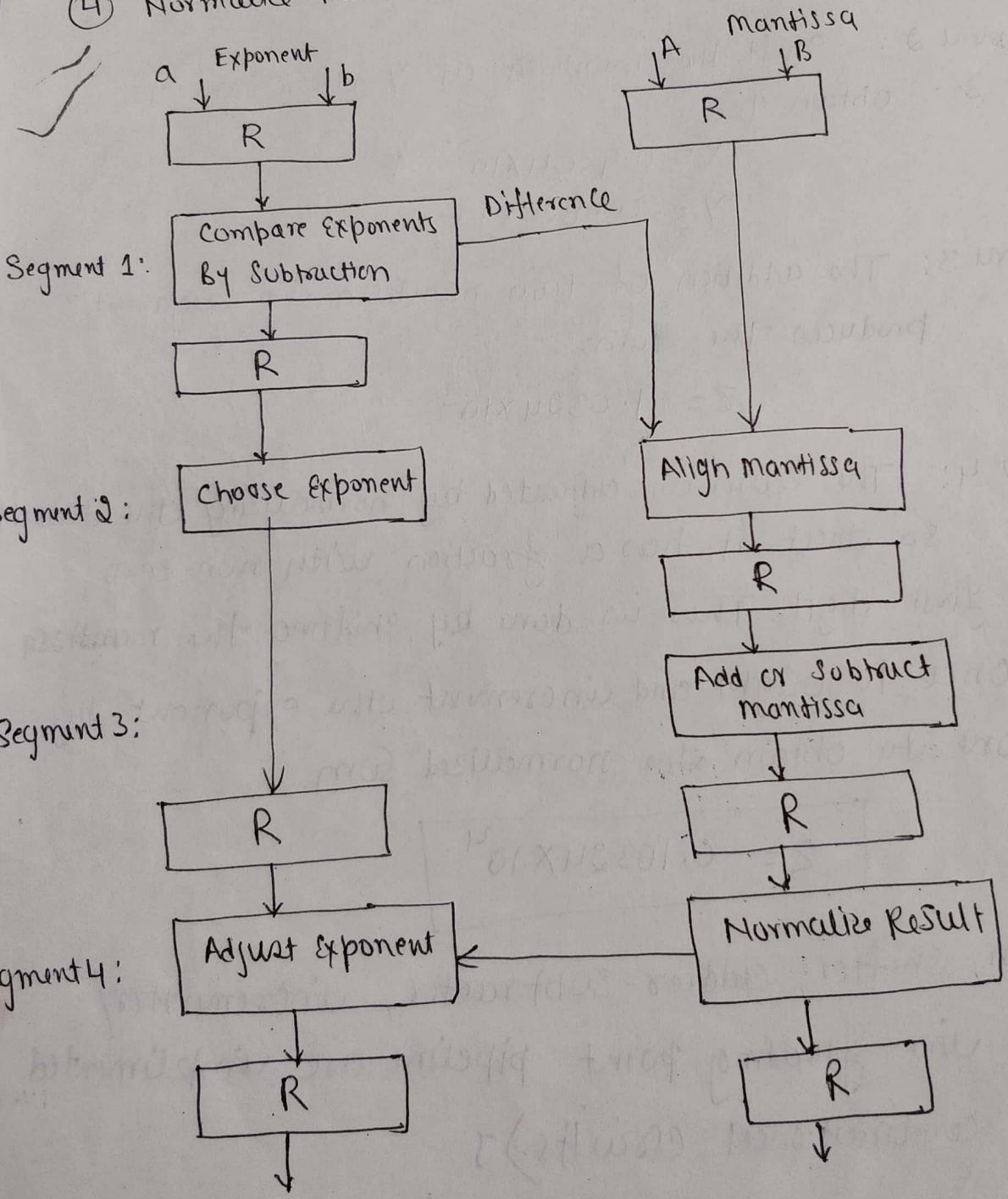
$$Y = B \times 2^b$$

where 'A' and 'B' are the mantissa part and 'a' and 'b' are the exponent part.

The floating point addition and subtraction can be performed in 4-segments. The register 'R' are placed between the segments to store the intermediate result.)]

The suboperation that are performed in the 4-segments are as follows:-

1. Compare the exponent
2. Align the mantissa
3. Add or Subtract the mantissa
4. Normalize the result.



[Pipeline for floating point addition & Subtraction]

Example:-

$X = 0.9504 \times 10^3$ &

$Y = 0.8200 \times 10^2$.

Segment 1: Two exponents are subtracted in first segment to obtain $3-2=1$. The larger exponent 3 is chosen as exponent of the result.

Segment 2: Shift the mantissa of Y to right to obtain the

$X = 0.9504 \times 10^3$ &

$Y = 0.0820 \times 10^3$.

Segment 3: The addition of two mantissa in segment 3 produces the sum:

$Z = 1.0324 \times 10^3$

Segment 4: The sum is adjusted by normalizing the result so that it has a fraction with non-zero first digit. This is done by shifting the mantissa once to right and increment the exponent by one to obtain the normalized sum.

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

Comparator, shifter, adder-subtractor, incrementer/decrementer in floating point pipeline are implimented with the combinational circuits.)

Instruction pipeline

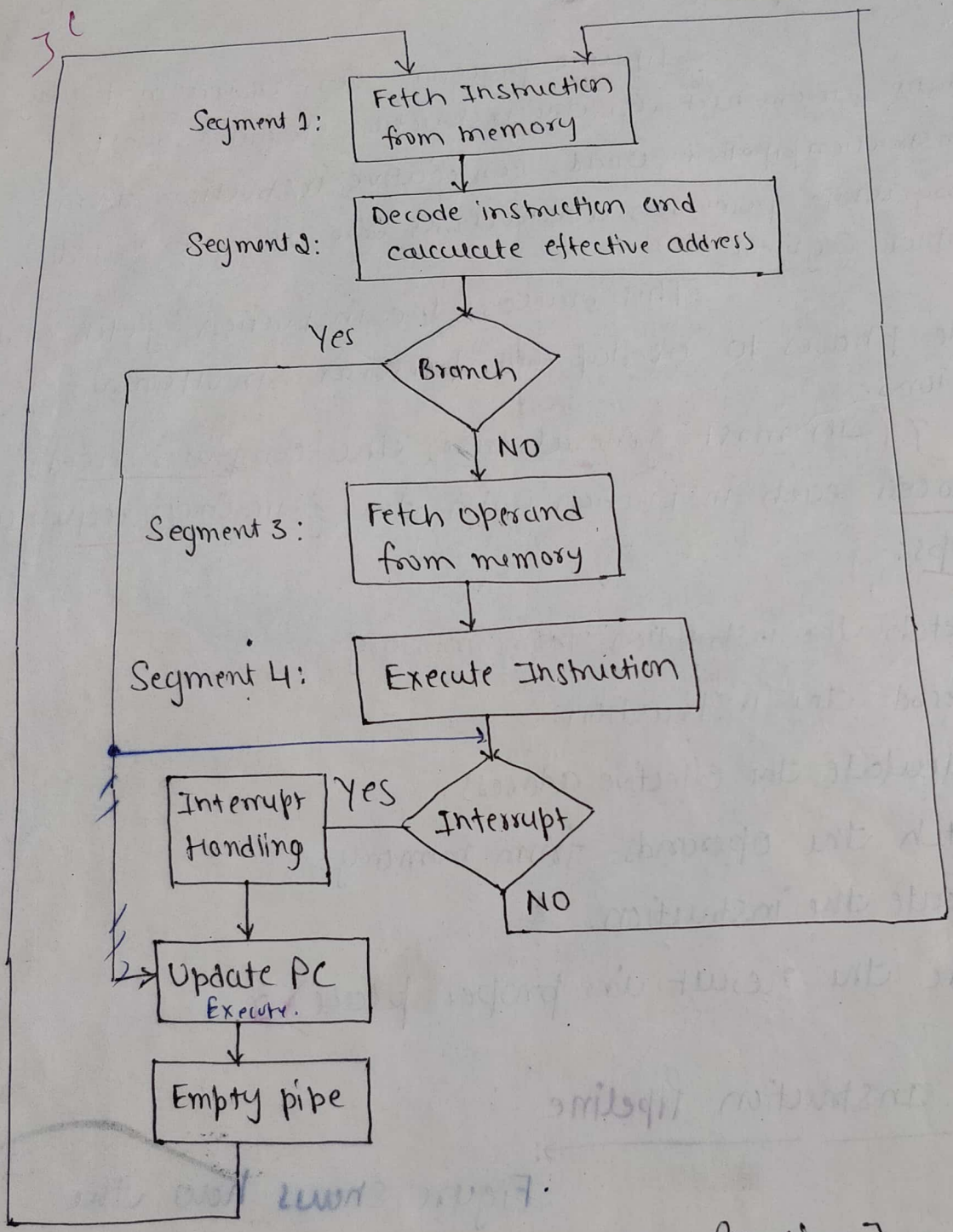
Pipeline processing can occure not only in data stream but in the instruction stream as well. "An instruction pipeline reads consecutive instruction from memory while previous instructions are being executed in other segments". This causes the instruction fetch and execute phases to overlap and perform simultanous operations.

In 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 proper place.

4-segment Instruction pipeline

Figure shows how the instruction cycle in the CPU can be processed with a 4-segment pipeline. while an instruction is being executed in segment 4, the next instruction in sequence is busy fetching an operand from memory in segment 3.



[Four- Segment CPU Pipeline]

The effective address may be calculated in separate arithmatic circuit for third instruction and whenever the memory is available, the fourth and all subsequent

instructions can be fetched and placed in an instruction FIFO. Thus upto four suboperations in instruction cycle can overlap and upto four different instructions can be in progress of being procured at the same time.

Following figure shows the operation of the instruction pipeline:-

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							
Interrupt 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]

(1) FI = FI is segment that fetches the instruction.

(2) DA = is a segment that decodes the instruction & calculates the effective address

(3) FO is segment that fetches the operand

(4) EX is the segment that executes the instruction.

It is assumed that the processor has separate instruction and data memories so that operation in FI and FO can perform at same time. In absence of branch instruction, each segment operates on different instructions. Thus in step 4, instruction 1 is being executed in segment EX; operand for instruction 2 is being fetch in segment FO; instruction 3 is being decoded in segment DA; and instruction 4 is being fetch from memory in segment FI.

Now assume 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 other instructions is halted until branch instruction is executed in step 6.

In general 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 same time. most of these conflic can be resolved by using separate Instruction & data memories.

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

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

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 number of computations that will take the conventional computers days or even weeks to complete. And following are the applications

Computers with vector processing capabilities are in demand in specialized applications. Following are the representative application areas where vector processing is of utmost importance :-

(1) Long-range weather forecasting.
(2) Petroleum explorations.
(3) Medical Diagnosis.
(4) Space flight simulations.
(5) Artificial intelligence & expert systems.
(6) Image processing.

Without sophisticated computers many of the required computations can not be completed within a reasonable amount of time. To achieve the required level of

high performance it is necessary to utilized the fastest and most reliable h/w and apply innovative procedures for from vector and parallel procuring techniques.

Vector Processing Definition :-

"A vector is an ordered set of a one-dimentional array of data items".

A vector 'V' of length 'n' is represented as a row vector by:

$$V = [V_1 V_2 V_3 \text{ ---- } V_n].$$

Also can be represented by the column vector.

* A vector processor is an ensemble of hardware resources including:- vector registers, functional pipelines, processing elements and register counters for performing vector operations.

* Vector processing occurs when arithmetic or logical operations are performed to vectors. It is distinguish from scalar processing which operates on one-or more pair of data.

Example :

To examine the difference between a conventional scalar processor and a vector processor consider following fortran program;-

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

This is a program for adding two vectors 'A' and 'B' of length 100 to produce a vector 'C'. This is implemented in machine language by following sequence of operations:-

```
        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
```

This constitutes a program loop that reads a pair of operands from array A and B and perform a floating point addition. Loop control variable is then updated and step repeat 100 times.

A computer capable of vector processing eliminates the overhead associated with the time it take to fetch and execute the instruction in program loop; it allows operations to be specified with a single vector instruction of the form.

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

The vector instruction include the initial address of operand, the length of vector and operation

to be performed, all in one composite instruction.

Operation code	Base Address Source 1	Base address Source 2	Base address Destination	Vector Length

(Instruction format for vector processor)

For succenful vector procening one need to makee improvements in vector hardware, vectorieing complex and programming skills, Specially target at vectors terms.

Array Processor

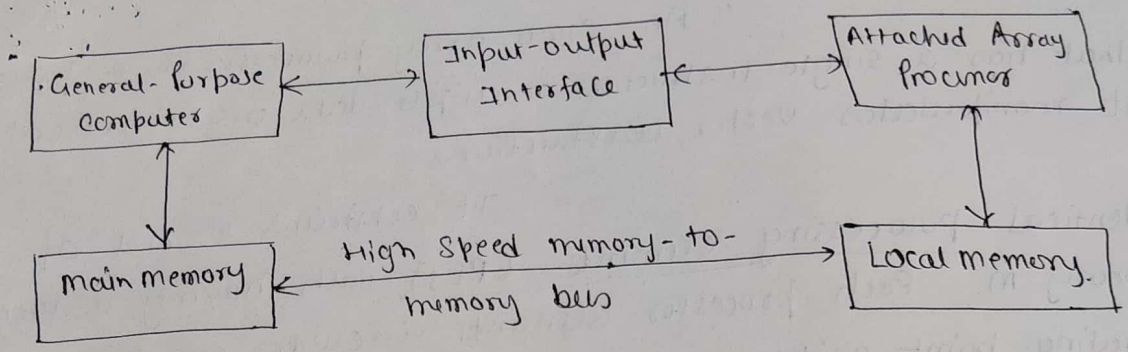
("An array processor is a procenor that performs computational on large arrays of data."

There are two types of array procenors:-

1. Attached array procenor
2. SIMD array procenor.

1. Attached array procenor

An attached array procenor is an auxiliary procenor attached to a general purpose computer. It is intended to improve the performance of the host computer in specific numerical computation tasks, by providing the vector procening.



[Attached array processor with host computer)

* The host computer is a general-purpose commercial computer and attached processor is a back-end machine driven by host computers.
* The array processor is connected through an Input output controller to the computer.
* The data for the attached processor are transferred from main memory to a local memory through a high speed bus.
* The system with the attached processor satisfies the needs for complex arithmatic applications.

Objective of the attached Processors →:

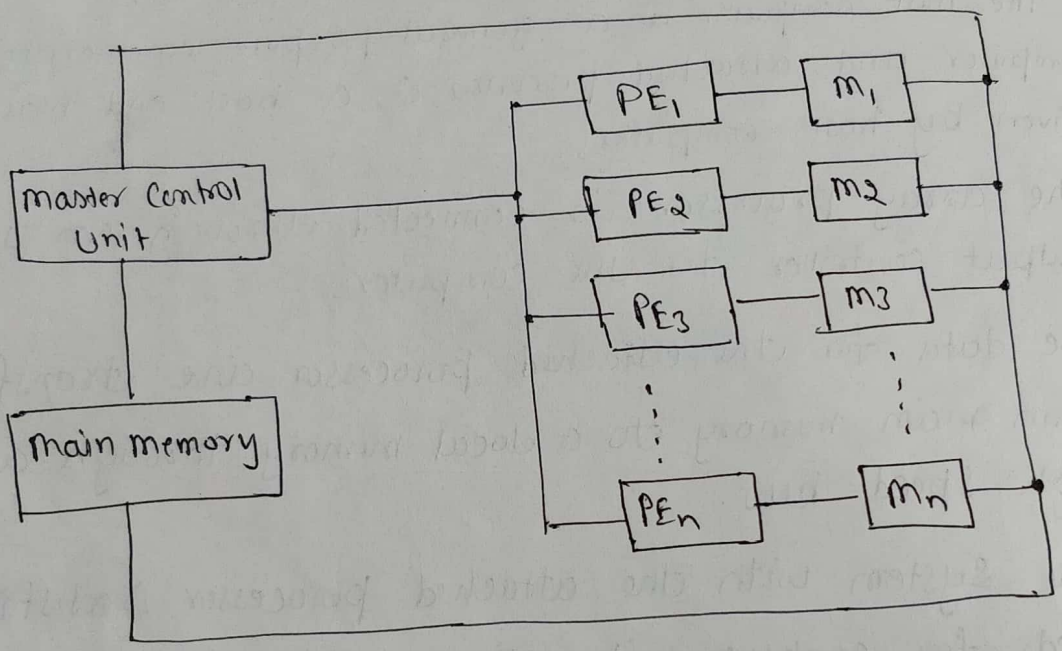
Objective of the attached processor is to provide vector manipulation capabilities to a conventional computer at a fraction of the cost of supercomputers.

P.T.O.

② SIMD Array Processor

"An SIMD array processor is a processor that has a single instruction multiple data organization". It manipulates vector instructions.

It contains a set of identical processing elements (PEs) each having a local memory M. Each processor elements include an ALU, floating point arithmatic unit & working registers.



[SIMD Array Processor Organization]

The master control unit controls the operations in the processor elements. & the main memory is used for storing the programs.

Function of master control Unit

Function of master control Unit is to decode the instruction and determine how the instruction to be executed.

Scalar and program control instructions are directly executed within the master control unit, and vector instructions are broadcast to all PEs simultaneously. Each PEs users operands stored in its local memory. Vector operands are distributed to the local memories prior to the parallel execution of the instruction.

Example:

Consider vector addition :-

$$C = A + B$$

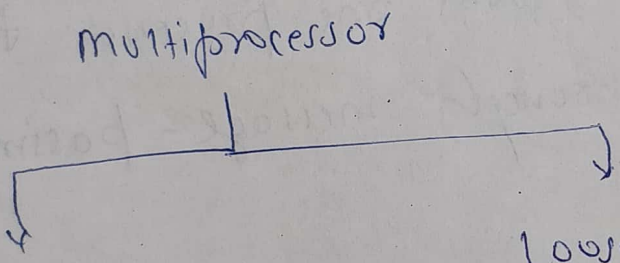
Master control unit first store the ith component ai and bi of A and B in local memories mi for $i = 1, 2, 3, \ldots n$. It then broadcast the floating point add instruction $Ci = ai + bi$ for all PEs causing the addition to take place simultaneously. The component of ci are stored in fixed length in each of local memory. This produce the desired vector sum in one add cycle.)]

MODULE-5

MULTIPROCESSOR

"A multiprocessor system is an inter-connection of two or more CPUs with memory and input-output equipment".

The term "processor" in multiprocessor can mean either a central processing unit (CPU) or an input-output processor (IOP).



1) Tightly coupled multiprocessor

multiprocessor are classified by the way their memory is organized. A multiprocessor system with common shared memory is classified as shared memory or tightly coupled multiprocessor. This does not preclude each processor from having its own local memory. In fact, most commercial