Unit-1

HPC :- It is the ability to proces data and perform complex calculations at high speed.

* Used for solving large problems
* Used to perform research activities through modeling simulation and analysis.

* Super-Computing and HPC are often used interchangabely.

Technologies based on High Peformance Computing:
→ IoT
→ Cloud
→ 3-D
→ Research activities

HPC act as a foundation for scientific and industrial advancement.

Stored Program Computer Architecture :-

Earlier storage was not there to store the result of computations. Stored program Computer architecture then emerges in which instructions are numbered and is stored as data in memory. Instruction was read and executed by control Unit. ALU was responsible for all computations. I/O provides Communication between the devices.

Control and arithematic units together with the appropiate interface to memory and I/O are called CPU.

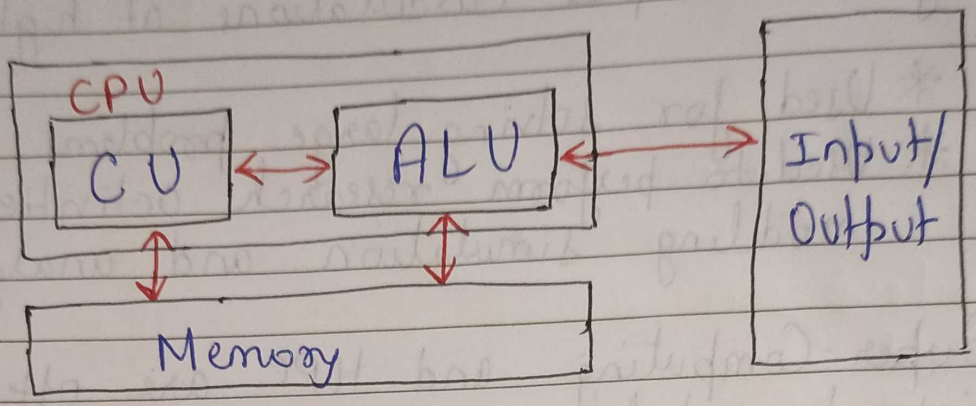


fig. Stored program Computer Architecture Block Diagram

General Purpose Cache based Computer Architecture :-

Microprocessor are probably the most complicated machinery that man has ever created. However they all implement the stored program digital computer concept. The block diagram of a modern cache based general purpose preprocessor is shown in fig. 2.

Components that actually do work for running application are the arithmetic unit for floating-point (FP) and integer (INT) operations. It makes up the very small portion of the chip area. The rest consist of administrative logic that helps to feed those unit with operands.

Load and store units handle instructions that transfer data to and from registers. Instructions are stored into several queues waiting to be executed. Finally Cache holds data and instructions to be frequently used. The major part of the chip area is usually occupied by cache.

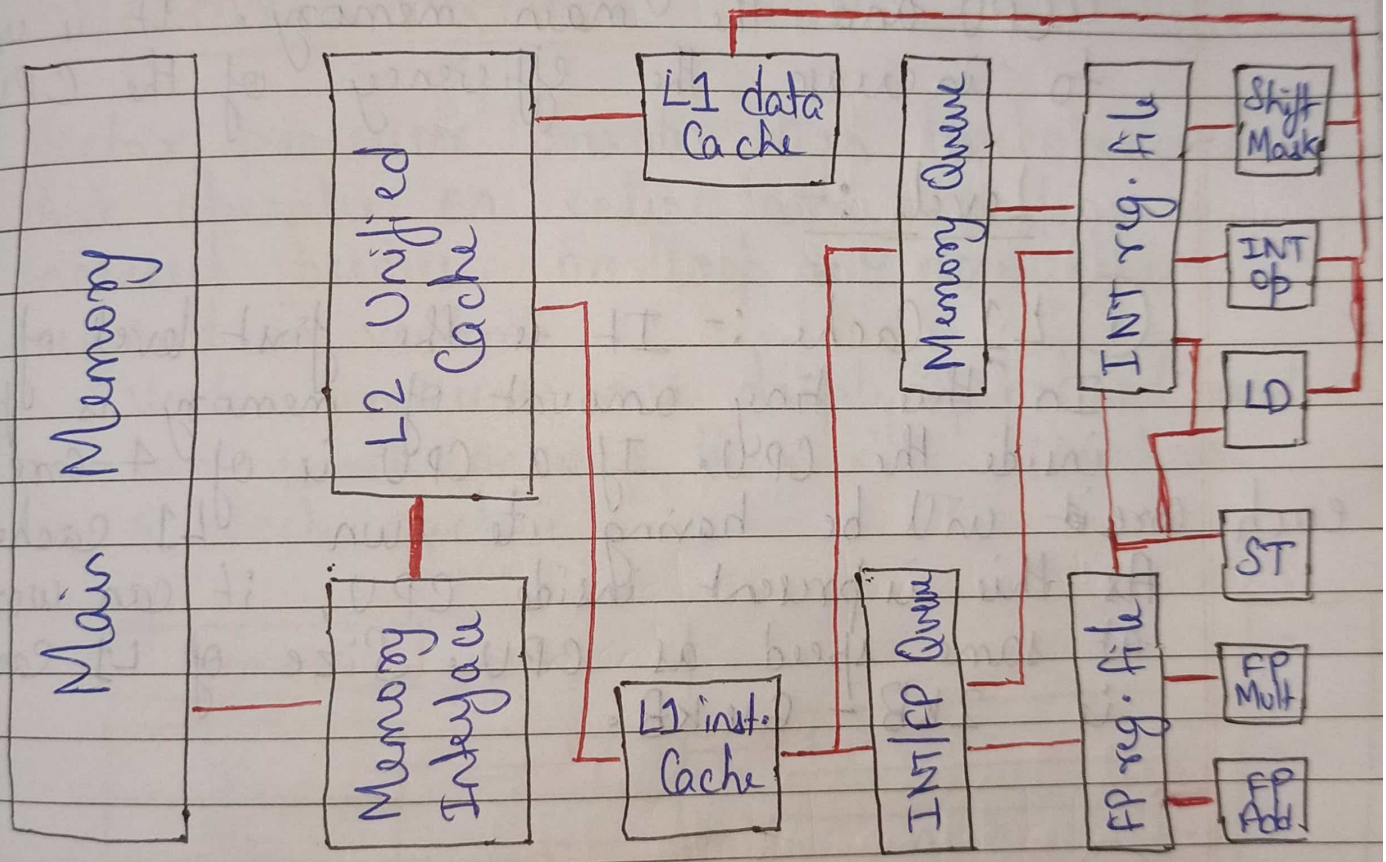
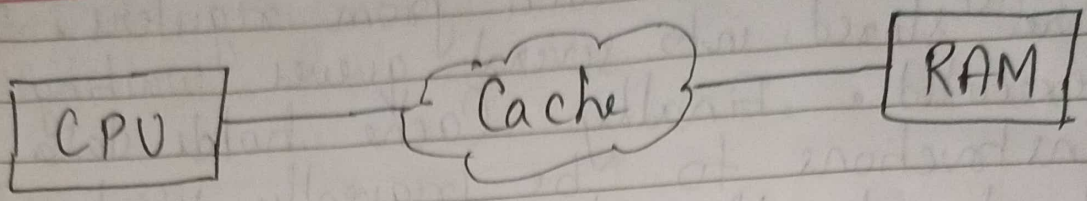


fig.2 General-purpose Cache based microprocessor

* HDD → 80 - 150 Mbps
* RAM → 4000 - 8000 MHz
* DRAM → 16000 - 21300 MHz
* CPU → 1 - 4 GHz

Cache :-



Cache is a temporary memory used to store the most recent items. It is a high-speed memory which lies between the CPU and the main memory. It is used to increase the efficiency of the CPU.

Levels :-

(i) L1 Cache :- It is the first level of cache. In this tiny amount of memory is present inside the CPU. If a CPU is of 4-core, then each core will be having its own L1 cache. As this is present inside CPU; it can work at same speed as CPU. Size of L1 cache is 2kB - 64 kB.

(ii) L2 Cache :- This cache may be inside the CPU or outside the CPU. All the cores of a CPU can have their own separate level 2 cache or they can share one L2 cache among themselves. Size of L2 Cache is 256 KB - 512 KB. Slower than L1 but faster than L3. If outside, then connected to CPU by high-speed bus.

(iii) L3 cache :- It is not present in all CPU but it is present in some modern processors. It is present outside the CPU and is used to enhance the performance of L1 and L2 cache. Size of this cache is 1 MB - 8 MB. It is expensive.

Vector Processor :-

Vector processor provides high level instructions that operates on entire array. Efficiency increases because no loop override is required in vector.

Vector processors are deeply pipelined and we can work on more than one instruction in a clock cycle.

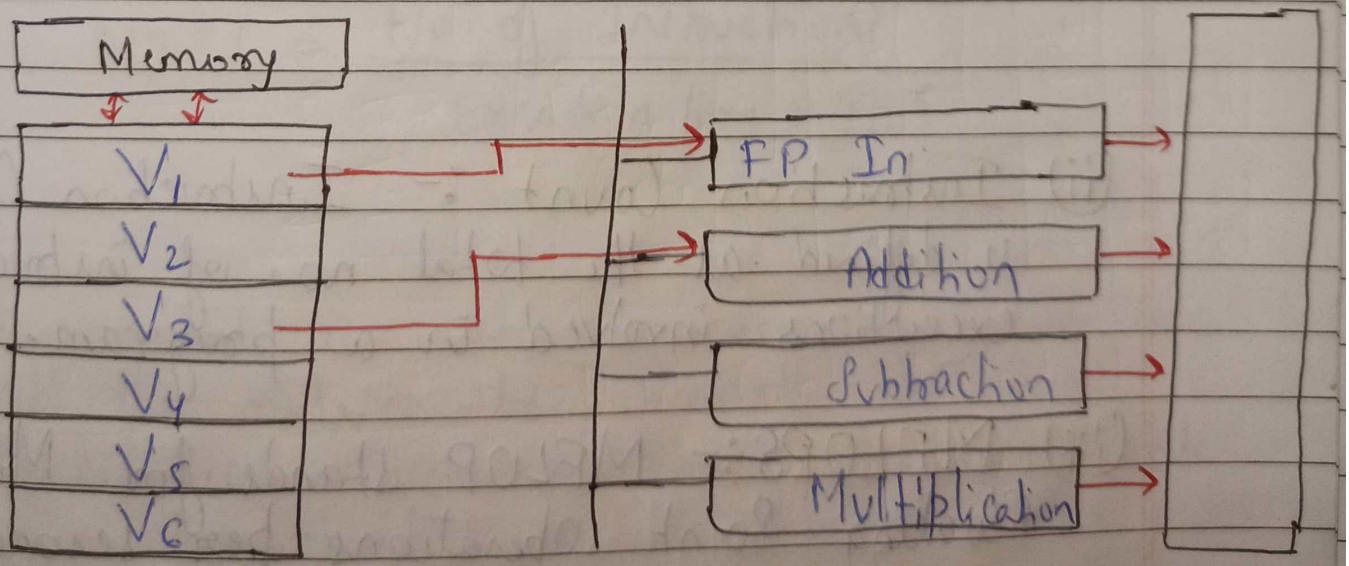


fig. Vector Processor

BenchMark :-

It is the test which measures the performance of a system or a subsystem.
or
It is a method of comparing performance of 2 different computer architecture.

We use various parameter like speed, cycles, etc for this.

Various performance Metrics :-

(i) Cycle per Instruction (CPI) :-
It is defined as the total CPU clock cycle divided by Instruction Count.

$$CPI = \frac{\text{total CPU clock cycle for program}}{\text{Instruction Count}}$$

(ii) Instruction Count :- Instruction Count (IC) is defined as the total no. of instruction executions involved in a program.

(iii) MFLOPS :- MFLOP stands for Millions Floating Point Operations per second.

$$MFLOP = \frac{\text{No. of floating point operation in program}}{\text{Execution time} \times 10^6}$$

(iv) Execution time :- It is defined as the total amount of time that is required to execute the program.

$$\text{Execution time} = CC \times CT$$

CC → Cycle Count

CT → Cycle time

(v) Clock time / Cycle time :- It is the period of the clock that synchronizes the circuits in a processor. It is the reciprocal of frequency.

$$\text{Clock time} = \frac{1}{f}.$$

(vi) MIPS :- MIPS stands for Million instruction per second.

$$\text{MIPS} = \frac{\text{No. of Instructions}}{\text{Execution time} \times 10^6}$$

Note → It is not necessary that we use all the metrics to compare the two architecture. We use those metrics which can be counted for both architecture.

Moore's Law :-

In 1965, Gordon Moore discussed about transistor density on IC. According to Moore's law,

The no. of transistor per square inch in the IC will double after every 2 years.

Is Moore's Law still holding?

This law was remarkably accurate from 1965 to 2013. After 2013, the trend has changed slightly with transistor density now doubling every 3 year instead of 2 year.

Advantages :-

Moore's law has had a direct impact on the progress of computing power. It results in rise in reduction of IC chips (i.e. Robustness).

Pipelining :-

Pipelining is the process of accumulating instructions from the processor through a pipeline.
It is a technique where multiple instructions are overlapped during execution.
It improves the performance of CPU.

We introduce faster circuit to improve hardware. In pipelining we arrange hardware in order to perform more than one operation at a time.

Difference b/w pipelined or non-pipelined :-
In non-pipelined system the no. of cycles required to execute an operation is more compared to a pipelined system where no. of cycles required is less.

Stages of pipelining :-

(i) Fetch :- In this, the CPU reads the instructions from the address in the memory whose value is present in program count.

(ii) Decode :- In this, instruction is decoded and the register file is accessed to get the value from the register used in instructions.

(iii) Execute :- In this stage, ALU operations are performed.

(iv) Memory :- In this, memory operands are read and written to/from the memory that is present in instruction.

(v) Write back :- In this, computed / fetched value is written back to register present in the instruction.

Inst. No. \ Clock Cycle	1	2	3	4	5	6	7
1.	IF	ID	EX	MEM	WB		
2.		IF	ID	EX	MEM	WB	
3.			IF	ID	EX	MEM	WB
4.				IF	ID	EX	MEM
5.					IF	ID	EX

Benchmark :-

1. Speed-up :- It gives idea of "how much faster" the pipelined execution is as compared to non-pipelined execution.

$$\text{Speed-up} = \frac{\text{Non-pipelined execution time}}{\text{pipelined execution time}}$$

2. Efficiency :-

$$\text{Efficiency} = \frac{\text{Speed-up}}{\text{No. of stage in a pipeline}}$$

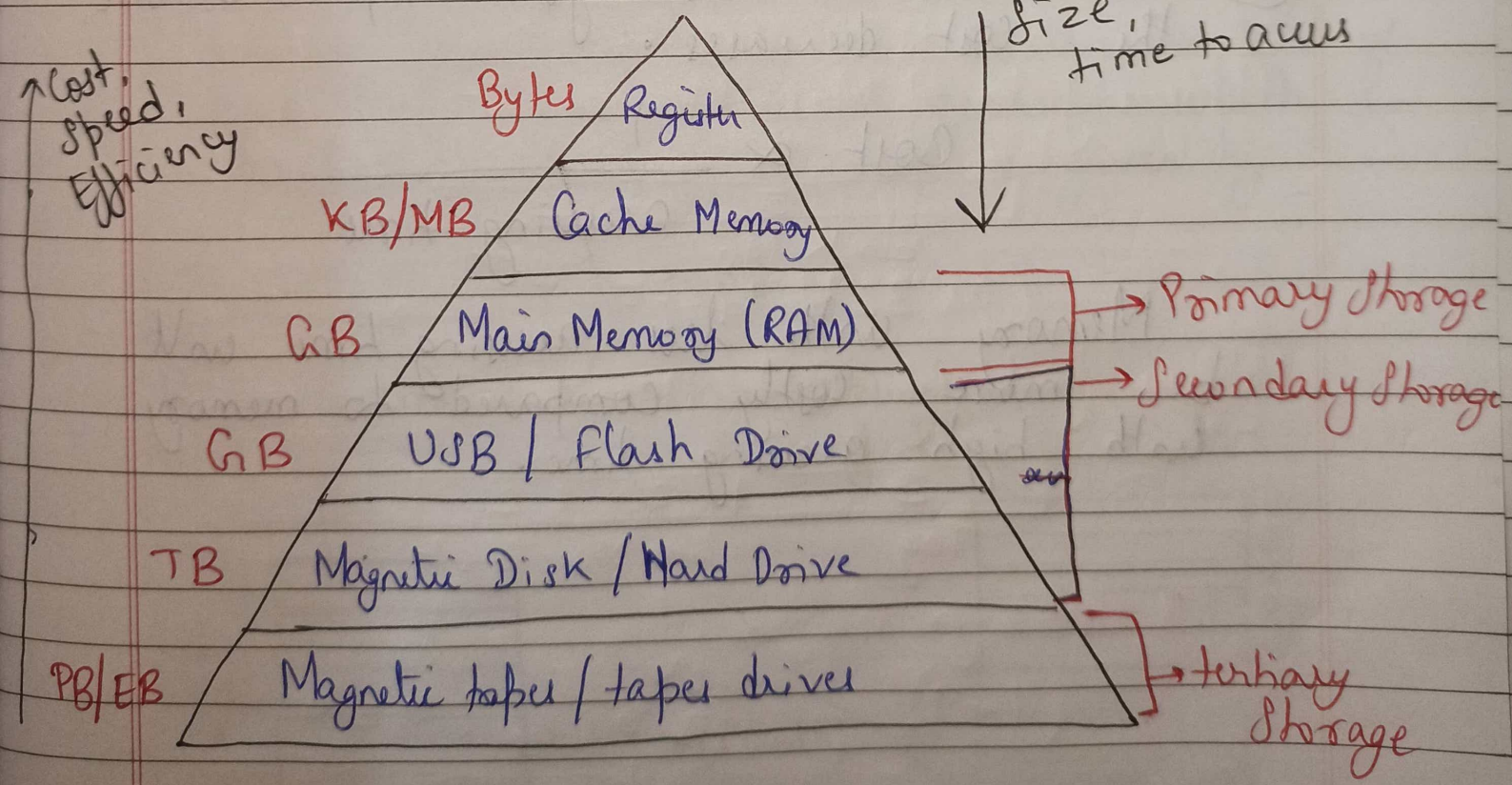
Throughput :- Throughput is defined as no. of instructions executed per unit time.

$$\text{throughput} = \frac{\text{No. of instruction executed}}{\text{Total Time taken}}$$

Memory Heirarchy :-

In computer System design, memory hierarchy is an enhancement to organize the memory such that it can minimize the access time.

It was developed based on program behaviour.



We can infer following characteristics from the above diagram :-

i) Capacity :- As we move from top to bottom the capacity increases.

(ii) Acess time :- It is the time interval between the read/writ request and the availability of data. As we move from top to bottom the acess time increases.

(iii) Performance :-
As we move from bottom to the top, the performance increases.

(iv) Cost :- As we move from top to bottom the cost decreases.

$$\text{Cost} \propto \frac{1}{\text{accessing time}}$$

Memory with less accessing time will be more costly compared to memory with high accessing time.

Flynn's Classification :-

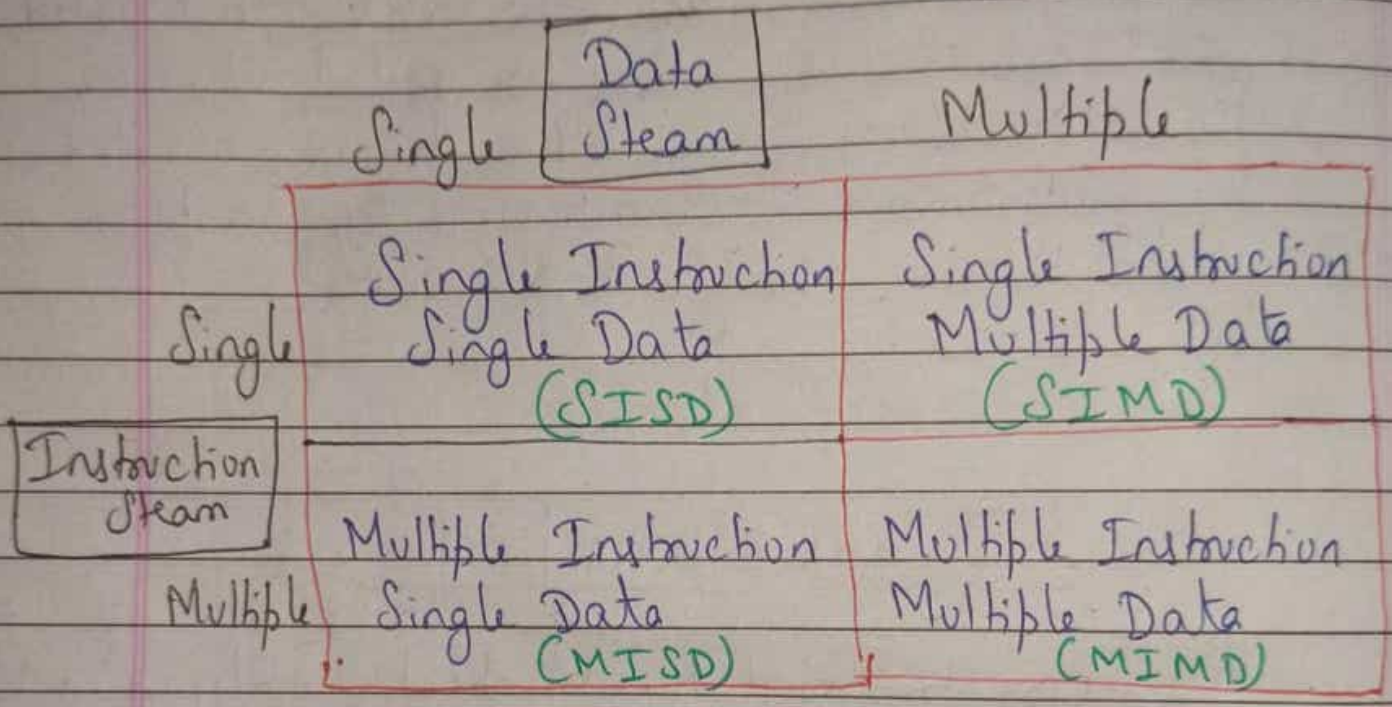
It was proposed by Micheal J. Flynn in 1966. In this classification computer are classified by whether it processes a single instruction at a time or multiple instruction simultaneously and whether it operate on one or multiple data sets.

This taxonomy distinguishes multiprocessor computer architectures according two independent dimension.

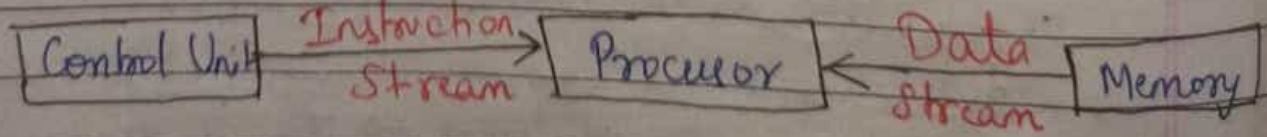
(a) Instruction steam → It is sequence of instruction executed by machine

(b) Data Steam → It is sequence of data including input, partial or temporary results used by instruction steam.

Each of these dimension can have only one of two possible states :- Single or Multiple.

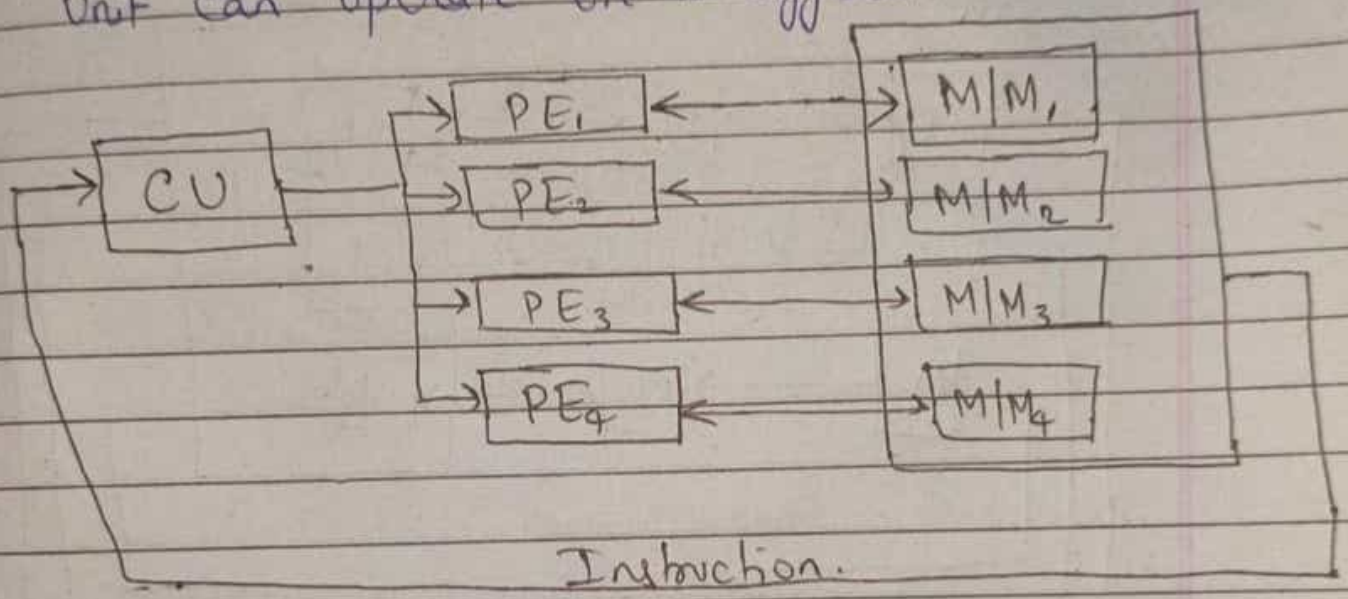


(a) SISD :- They are also called scalar processor i.e. one instruction at a time and each instruction have only one set of operands.



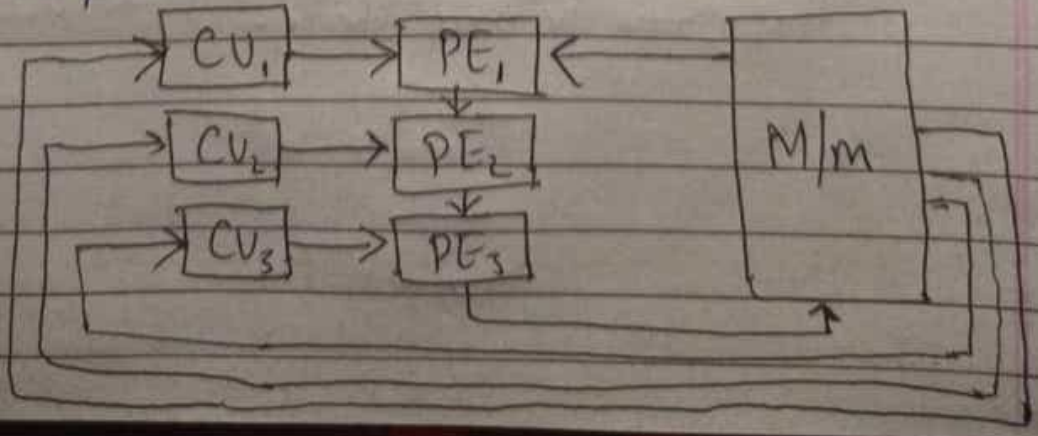
SISD have one control unit, one processor unit and single memory unit. In this instructions are executed sequentially.

(b) SIMD :- A type of parallel computer. Single Instruction is executed by different processing unit on different set of data. In this, all processing units execute the same instruction issued by control unit at any given clock cycle, also each processing unit can operate on a different data element.

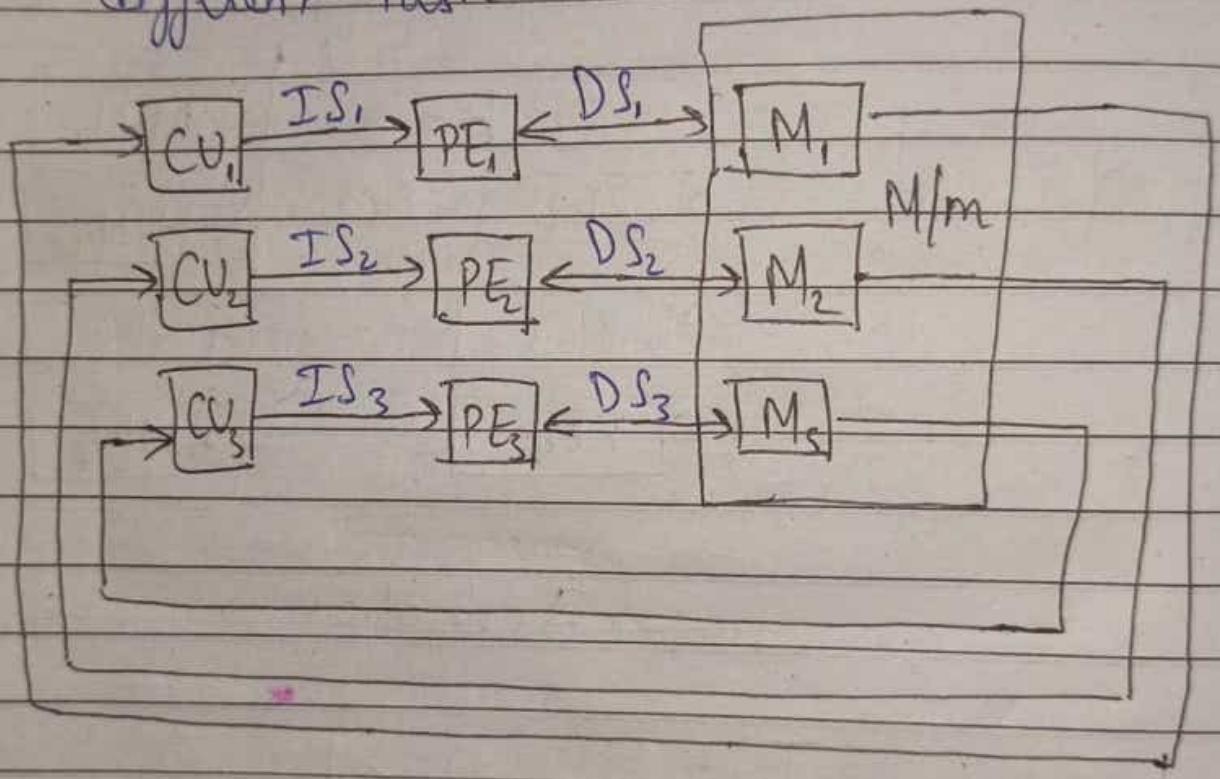


$I_s = 1 \quad D_s > 1$

(c) MISD :- A single data steam is fed into multiple processor unit. Each processing unit operates on data independently via independent instruction.



(d) MIMD :- In this, every processor may be executing a different instruction steam and different data steam too. Execution can be synchronous or asynchronous. Different processor are there with each of them processing different task.



Thread :- It is an instruction steam with state (Register / Memory). The Thread state is called as thread context. It may belong to same process or different processes. Threads on same program share the same address space.

Cache Mapping :-

It defines how content of main memory are brought into cache (i.e how a block from a main memory is mapped to the cache memory).

* In cache memory, data is stored in lines, unlike main memory where data is stored in blocks.

Cache Mapping Techniques :-

(i) Direct Mapping :-

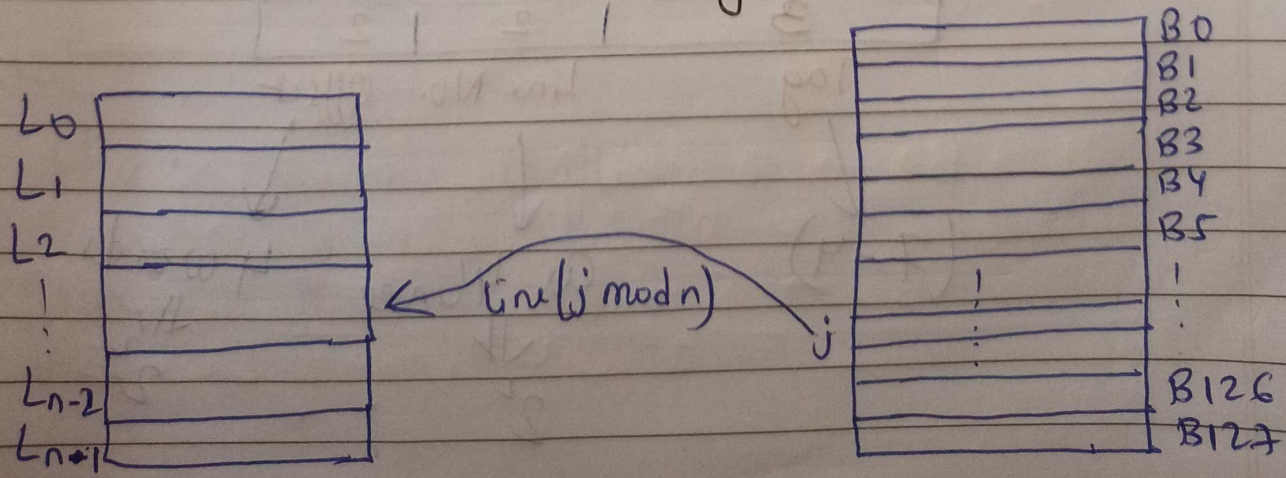
In direct mapping, a particular block of main memory can map only to a particular line of cache. The line number of cache to which a particular block can map is given by

$$K \bmod n$$

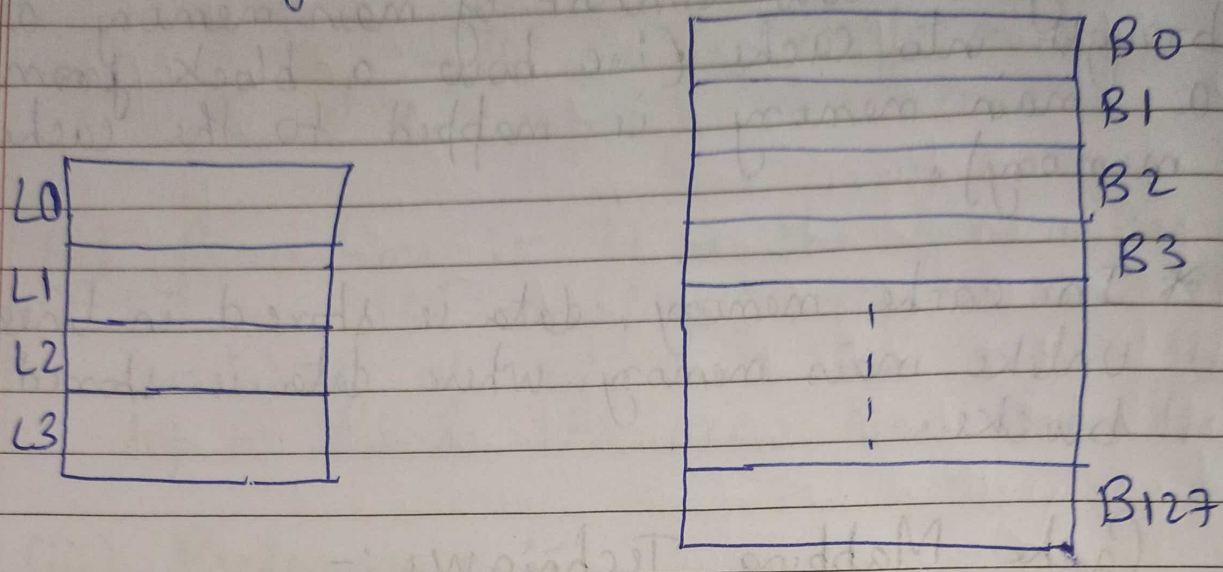
↓

where K → Block Number

n → No. of lines in Cache.

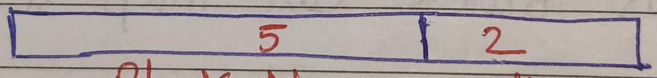


For a main memory with 128 blocks and cache of 4 lines.



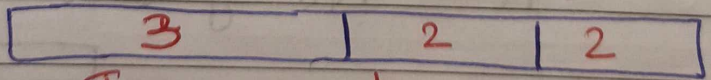
Addrss field consist of 7 bits [as $2^7 = 128$ ← No. of blocks]

Addrss field of Main Memory :-



(as 32 blocks $= 2^5$) → (as 4 words $= 2^2$)

Addrss field of Cache Memory :-



(7-4)

as 4 lines ⇓ 2^2

4 words ⇓ 2^2

Advantage of Direct Mapping :-

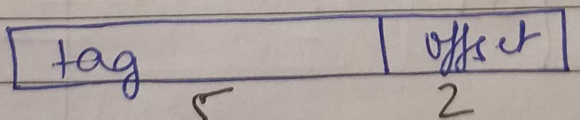
→ Simple
→ No need to search each line, just search tag no.
→ Search time is less
→ less expensive than associative

Disadvantage :-
It gives low performance because of replacement for data-tag value which is done for conflict miss.

(ii) Fully Associative :-

In this, a block of main memory can map to any line of cache that is freely available at that ~~or~~ moment. This makes it more flexible than direct mapping.

So, in addres field of cache Line no is not there



Adv :- Overcome Conflict miss

Disadv :- Searching hard

(iii) Set-Associative (K-way Associative) :-

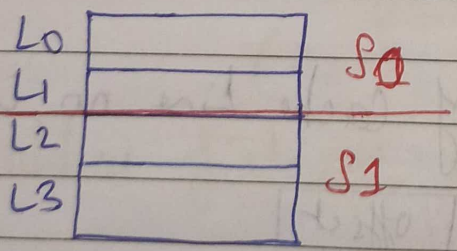
It is a combination of both direct mapping and ~~k-way~~ associative mapping.

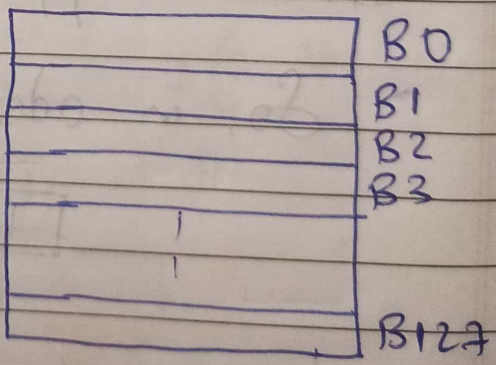
In this, Cache lines are grouped into sets where each set contains K numbers of lines. A particular block of main memory can map with one particular set of the Cache. However within that set, the memory block can map with any cache line that is freely available.

The set of cache to which a particular block of memory can map is given by

Block No modulo No. of Sets

Example for 2-way Associative :-

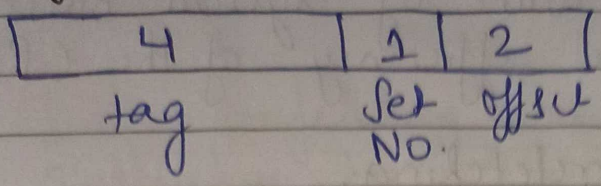




B0 → 0 mod 2
= 0th Set (any of L0 & L1)

B5 → 5 mod 2
= 1st Set (any of L2 & L3)

Address field :-



* If K=1, then K-way set associative mapping becomes direct mapping. i.e

1-way Set Associative ≡ ~~Fully Associative~~ Direct

* If K=n then K-way set associative mapping becomes fully associative mapping.

Prefetching :-

Prefetching is the loading of a resource before it is required to decrease the time waiting for that resource.

Latency problem can be solved by prefetching. It supplies the cache with data ahead of the actual requirement of the application.

Latency basically means inactive of data in memory(either

continous cache miss occur). Long latency lead to the long wait of system buses which degrades the efficiency of the CPU.

for prefetching,

$$T = T_L + \frac{L_c}{B}$$

where $T_L \rightarrow$ Latency
$B \rightarrow$ Bandwidth
$L_c \rightarrow$ Whole Line of Length

→ If T_L is the latency & B is bandwidth the transfer of whole line of length L_c (in bytes), then time taken is derived by this formula.

$$\text{prefetch} = \frac{\text{Total time } (T)}{\left(\frac{L_c}{B}\right)}$$

$$= \frac{T_L + \left(\frac{L_c}{B}\right)}{\left(\frac{L_c}{B}\right)}$$

$$\boxed{\text{prefetch} = 1 + \frac{T_L}{\left(L_c/B\right)}}$$

In Computer architecture, there are following types of prefetching :-

(i) Data prefetching
(ii) Instruction prefetching
(iii) Hardware prefetching
(iv) Software prefetching

Super-Scalar :-

It refers to a machine that is designed to improve the performance of the execution of the scalar instruction. A super scalar processor implements a form of parallelism (Instruction level parallelism) within a single processor. It is commonly used in RISC architecture.

Basically, in this, we equip the processor with multiple processing units to handle several instructions in parallel in each processing stage.

Superscalar are much faster than the scalar processor because it increases the throughput as the CPU can execute multiple instruction per clock cycle.

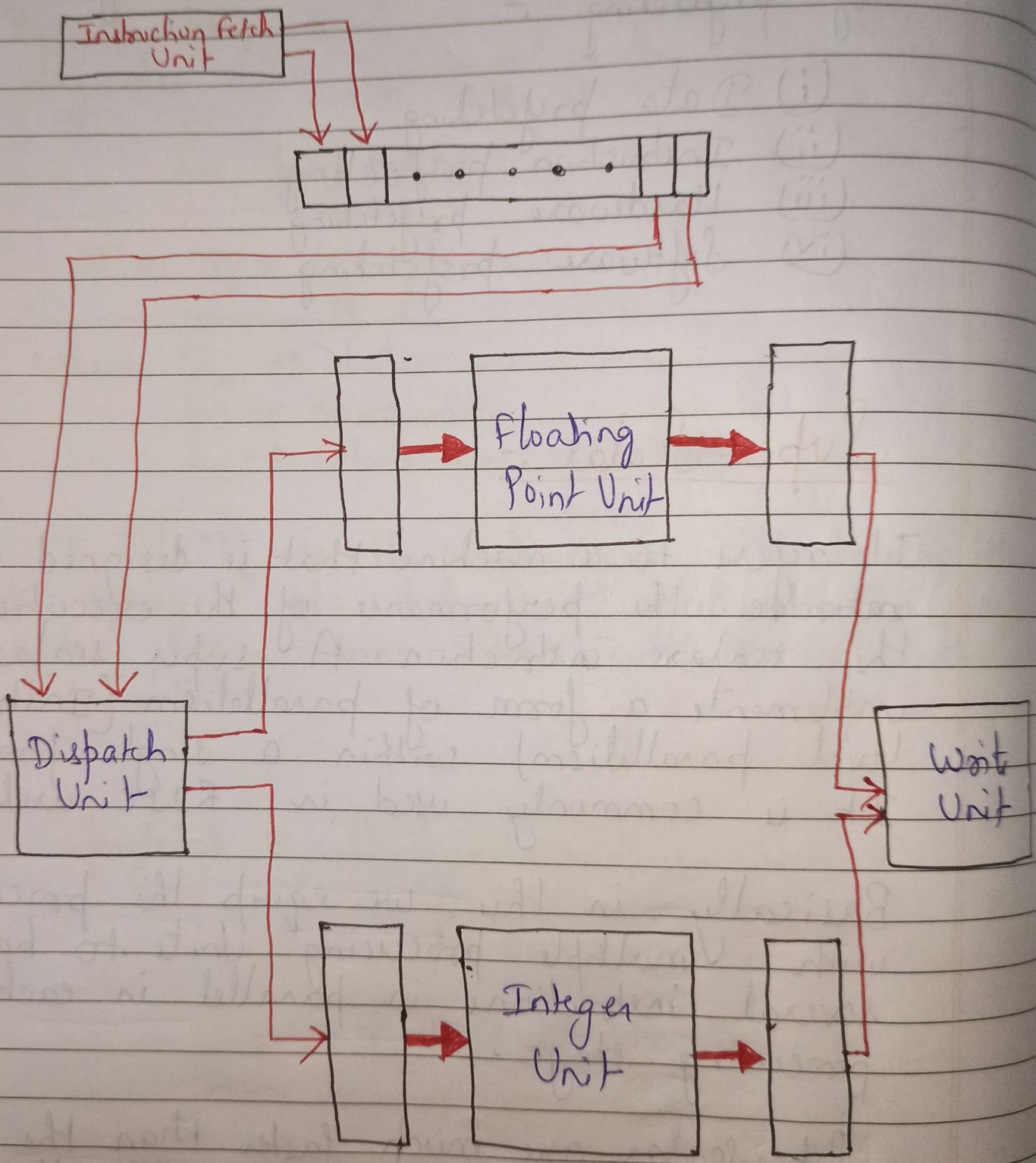


fig. An example of 2 Execution Unit

Advantange :-

1) Functional Unit will not be ideal for long. So efficiency of CPU increases.
2) More than 1 issue can be solved.

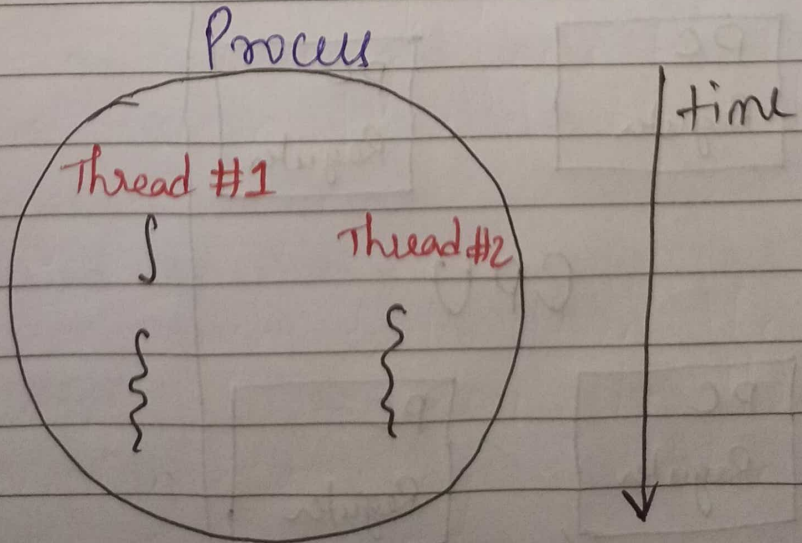
Disadvantage :-

1) Hard to execute Complex instructions
2) We can't efficiently execute properly if the ~~inst~~ instruction is dependent on one another.

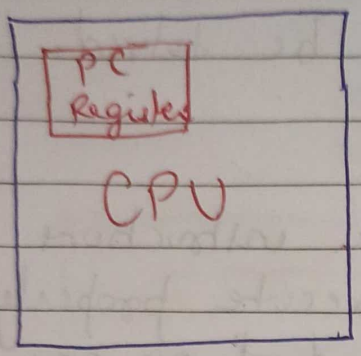
Multithreading :-

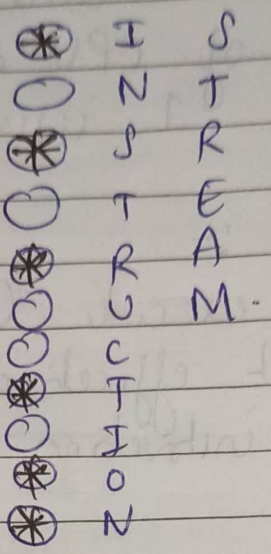
Thread is basically a control flow of execution. A thread is also known as lightweigth process.

In Computer architecture, multithreading is the ability of a central Processing Unit to provide multiple threads of execution concurrently.



Single-Thread System vs Multithread System





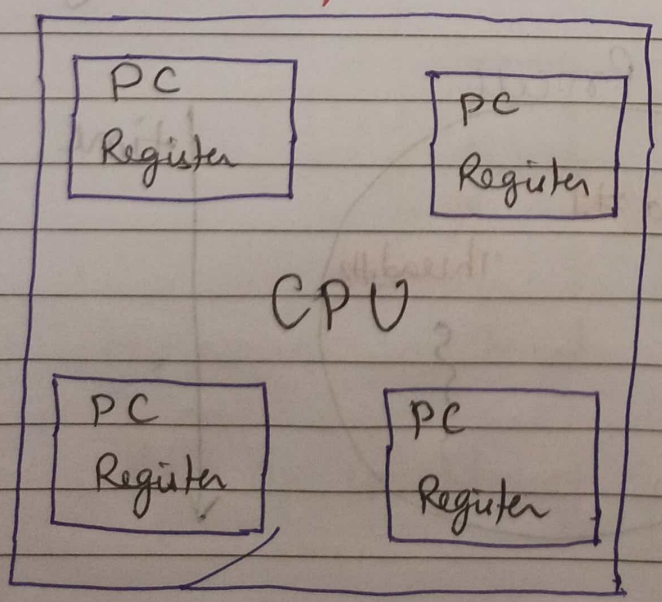
Single thread

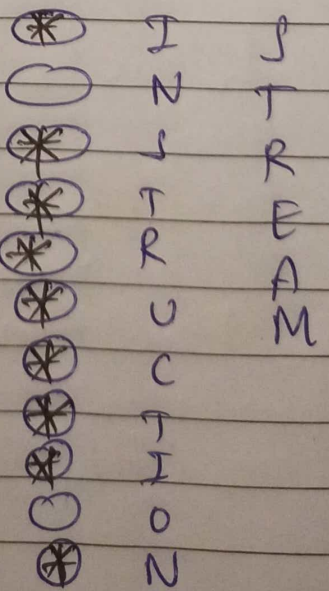
○ → Not utilized instruction

⊛ → Utilized instruction.

* In single thread no. of non-utilized instruction is more compared to a multithread system.

→ Multi-Thread



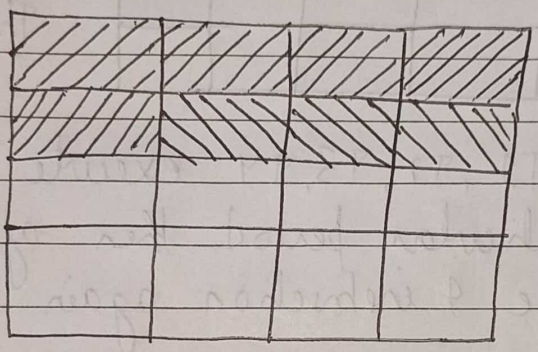


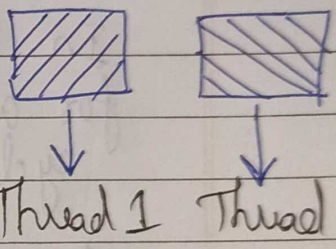
3 Categories of Multithreading :-

(i) Course Grain Multithreading (CGMT) :-

In this ~~long~~ when long latency occurs, one thread stops and automatically switch to next thread.

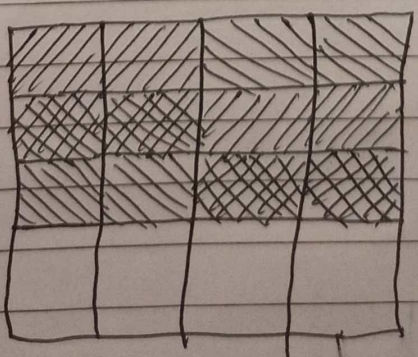
It is not suitable for out of order instructions.

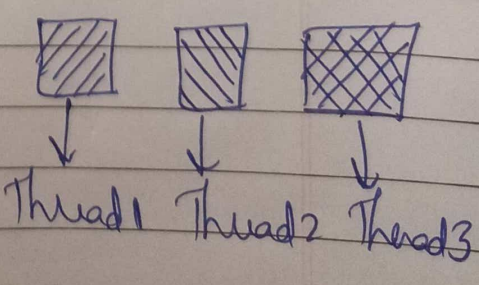




(ii) Fine Grain Multithreading (FGMT) :-

It is based on Round Robin algorithm. Each thread ~~is~~ is executed once for a particular period of time and that thread is again get executed after every thread is executed for that particular period of time (i.e after one cycle).



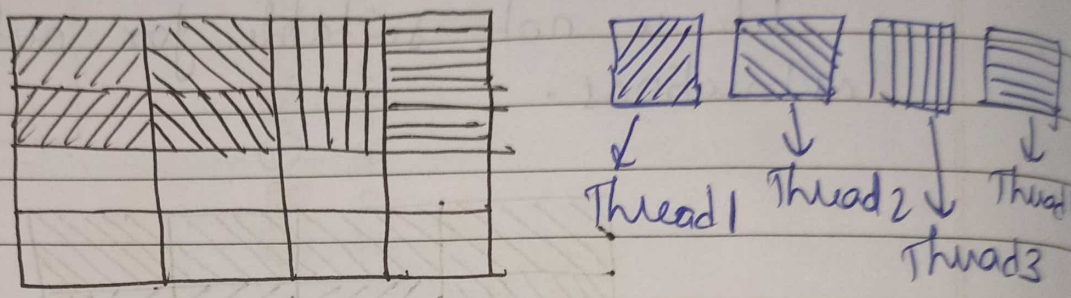


We switch to next thread in every cycle.

(iii) Simultaneous Multithreading :-

It is a combination of both fine grain and superscalar.

In this if functional units are available then they work on more than one instructions.



In this T1, T2, T3, T4 execute together ~~and~~ for a particular period then after 1 round cycle these 4 instruction again execute together.

This works well in out of order instructions.

Multicore :-

→ A multicore processor is a microprocessor on a single integrated chip with two or more seperate processing units called cores, each of which reads and execute program instructions.

On the other hand single-core processor is a ~~multipro~~ microprocessor with a single core on its die. It performs the fetch-decode-execute cycle once per cycle as it only runs on one thread.

Multiprocessor :- A system with two or more CPU's that allows simultaneous processing of programs.

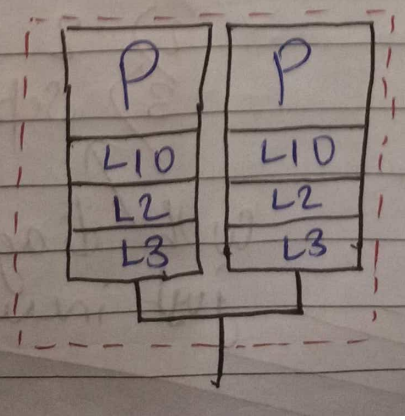
Difference b/w Multicore & Multiprocessor :-

MultiCore	Multiprocessor
→ A single CPU or processor with two or more independent processing units called cores that are capable of reading and executing program instructions.	→ A system ~~that~~ with two or more CPU's that allows simultaneous processing of programs.
→ It execute single program faster.	→ It executes multiple program faster.
→ Not as reliable as multiprocessor.	→ More reliable since failure in one CPU will not affect other.
→ It has less traffic	→ It has more traffic
→ It does not need to be configured	→ It need little complex configuration
→ It is cheaper	→ It is Expensive.

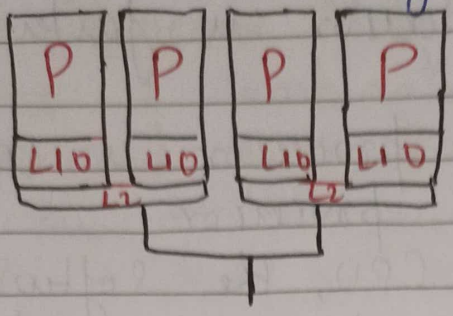
→ Multicore processor is faster than single core processor.
→ ~~Micro~~ Multicore processor is costly as compared to a single-core processor.
→ In multicore CPU, the software is always assigned to different cores. When one piece of software fails, the other remain unaffected. Whenever a defect arises, it affects only one core. As a result, multi-core CPU are better ~~than~~ to resist faults.
→ An operating system can use a multicore CPU to run two or more processor at a same time, even if many programmes may be executed at the same time.
→ In comparision to a single-core processor, multicore is capable of processing large amounts of data.

Variations of Multicore :-

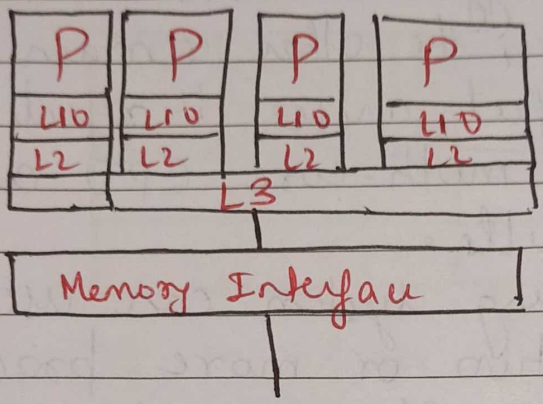
(i) Dual-Core :- A dual core processor for a computer is a central processing unit (CPU) that has two separate cores on the same die, each with its own cache.



(ii) Quad-Core :- having 4 Cores.

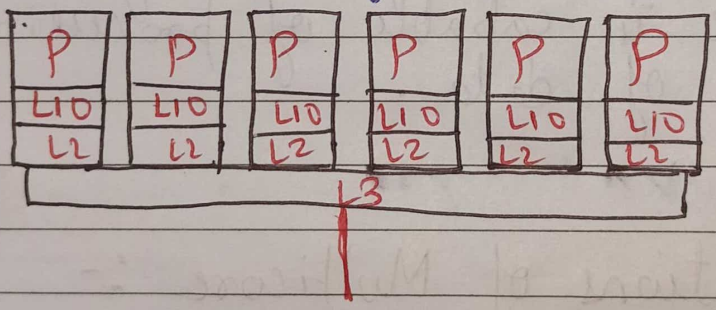


← 2 dual core variant



← Each core with own cache.

(iii) Hexa-Core :- having 6 Cores



(iv) Octa Core :- having 8 cores :-

↳ variant :- 1) 4 dual-core
2) 2 Quad-core
3) Separate one

With diagram similar to above. just increase cores.