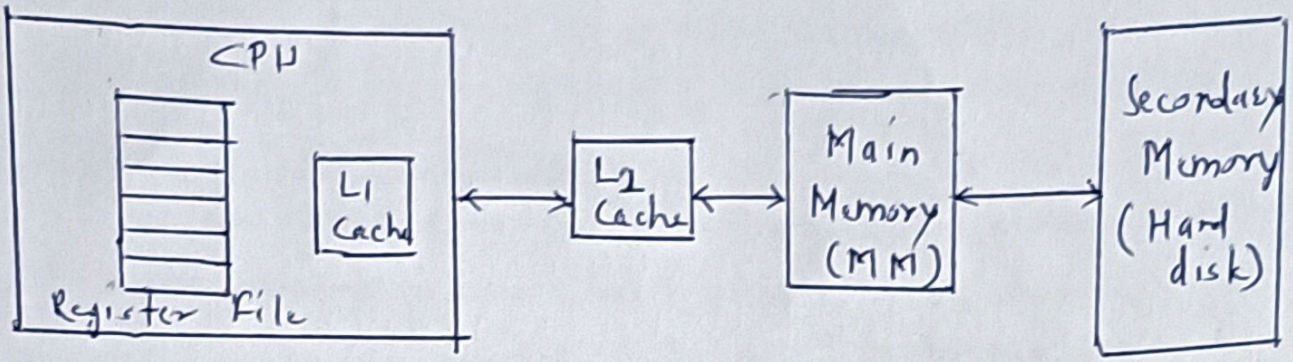
Hierarchical Memory Organisation in Computer System :

Inside the computer system there are variety of memory elements. All these elements are organised or placed in hierarchical manner as shown in the diagram



1] Register file / Internal Register

Each processor contains a set of internal registers known as register file. These register used for temporary storage & hence they act as internal memory of computer.
It is always better to have large number of registers with larger size.
They are the closest memory element for microprocessor. i.e when up needs something he will approch first to internal registers.

2] Internal Cache / L1 Cache

Apart of from internal registers now a days each processor has inbuilt or internal cache which is known as L_1 Cache.

3] External Cache / L_2 Cache : Apart from internal cache there is external Cache which is a high speed SRAM placed between up & main memory

4] Main Memory

It is made up with DRAM technology & size of main memory is decided by number of add. lines.

5 Secondary Memory : It is much more larger in size & generally made up with magnetic technology i.e hard disk

Cache Memory

It is a high speed static RAM (SRAM) placed between Processor & the main memory.

Contents of Cache memory

The cache memory contains frequently used instructions and frequently used data items from main memory.

The instructions and data items which are required again & again by the processor are copied into cache memory.

Size of the Cache Memory

As compared to main memory size (storage capacity) of cache memory is much more smaller than memory. Also the cache memory is made up with SRAM technology while main memory is made up with DRAM technology.

Difference between SRAM & DRAM

SRAM	DRAM
1) It is used in Cache memory	1) It is used in main memory
2) It is faster	2) It is slower
3) It is costlier	3) It is cheaper
4) storage capacity is smaller	4) storage capacity is more
5) Physically size is much more larger	5) Physical size is smaller.
6) It consumes more power.	6) It consumes less power
7) Flip flop is used for information storage	7) Capacitors are used for information storage
8) Refreshing circuitary is not required	8) Refreshing circuitary is required.

Principle of Operation of Cache Memory

It is observed that same information from concecutive memory location is required again & again in order to executive program loop. Most of the program that run on PC concist of such loop. Characteristics of such programs can be explained with a term called "Locality of Reference" It has two terms —

1] Temporal Locality
2] Spatial Locality

Classification of Cache Memory

1] Internal & External Cache
2] Unified & split up cache

External Cache: If cache memory is connected externally (outside) to processor then it is called as external or Level 2 (L_2) cache.

Internal Cache: If cache memory is present within the processor (inbuilt) then it is called as internal or inbuilt or Level 1 (L_1) cache

Unified Cache: If the cache memory is used to store both instructions as well as data items, then it is known as unified cache memory.

Split up Cache: If two seperate cache memories are used, one for storing only instructions & one for storing only data then it is known as split up cache.

I Cache: The cache memory which contains or stores only instructions is called Instruction or I Cache

D Cache: The cache memory which contains or stores only data is called Data or D Cache.

* 80386 is the first processor in which cache memory is used. In this processor cache was connected externally (L_2) & it was unified cache.
* 80486: is the first processor in which both external (L_2) as well as internal (L_1) cache is used. Both the cache memories were unified.
* Pentium: is the first processor in which both external (L_2) and internal (L_1) cache is used. But both the cache memories were in split up form.

Performance of Cache Memory:

The performance of cache memory is measured in terms of "Hit Ratio"

$$\text{Hit Ratio} = \frac{\text{No. of Hits}}{\text{No. of Hits} + \text{No. of Miss}}$$

OR

$$\text{Hit Ratio} = \frac{\text{No. of Hits}}{\text{Total No. of Memory references made by CPU}}$$

The maximum possible value of hit ratio = 1
Hit Ratio is generally expressed in percentage & max. possible value is 100%. But practically 100% hit ratio is never acchived. It is always better to have higher/larger value of Hit Ratio

Hit : If processor makes refereftce of a desired thing & if it is found in cache memory, then it is called as Cache Hit

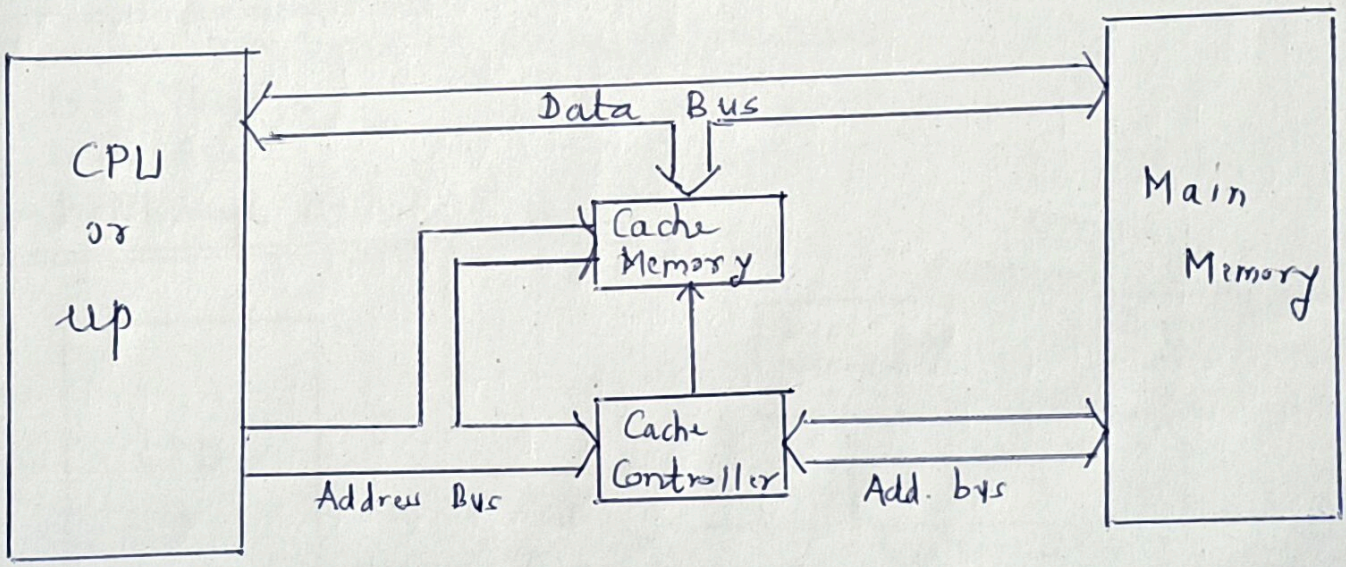
Miss : If processor makes reference of a desired thing & if it is not found in cache then in that case processor has to go to main memory, it is called as Cache Miss.

Example: Processor need 100 things, 80 times he got the desired things in cache & only 20 times he went to main memory.

∴ No of hits = 80
∴ No. of miss = 20

$$\therefore \text{Hit Ratio} = \frac{80}{80+20} = \frac{80}{100} = 0.8 \text{ or } 80\%$$

Working of Cache Memory



* Cache memory is always connected between CPU & main memory
* Cache memory always comes with Cache Controller
* When Microprocessor or CPU will start read operation, the Cache controller will check whether the desired information is present in cache or not.
* If the desired information is present in cache then it is given to CPU by using data bus & it is called as Read hit

If the cache controller determines that the desired thing is not present in cache then it is obtained from main memory (misses) The desired information from main memory is given to CPU by using data bus and also copied into cache memory.

Factors on which Hit Ratio / Performance Depends.

1] Cache memory size
2] Cache Architecture
3] Cache memory Organisation / memory mapping
4] Cache update / Write policy

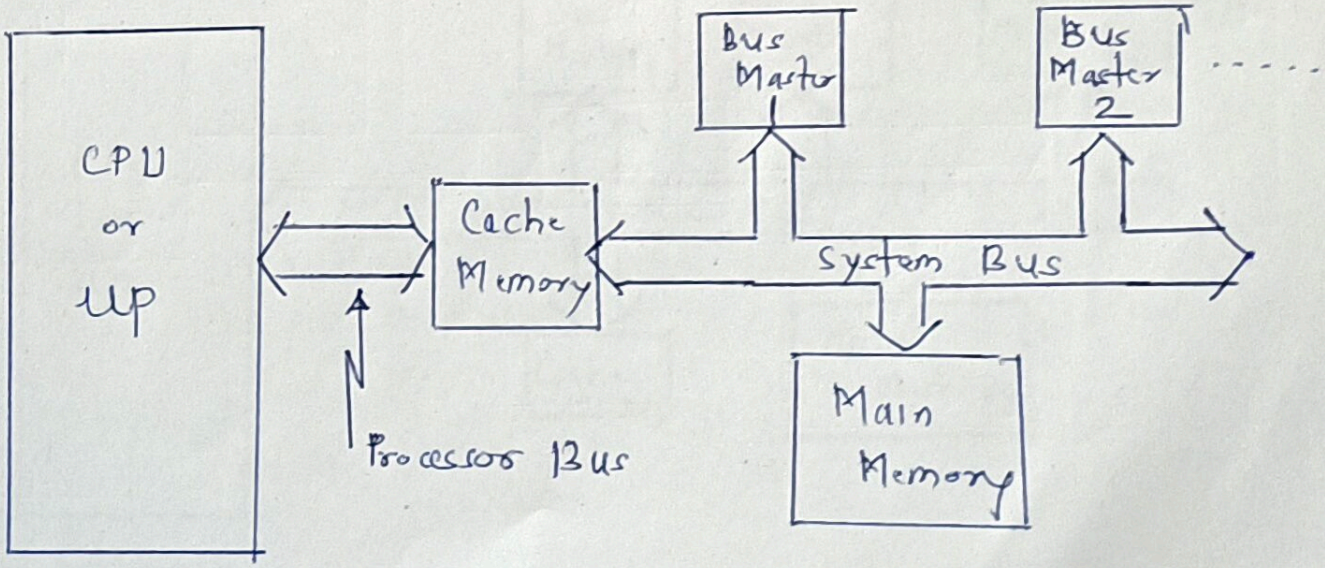
Cache Memory Size:

Hit Ratio depends on size (storage capacity) of cache memory Larger is the size of cache more will be the hit ratio. Because when cache size will increases the chances of getting the desired thing will increase. That means no. of hits will increase & ultimately hit ratio will increase. But we can increase the size of cache memory much more because of the physical size & cost. Hence a suitable combination of smaller size of Cache & larger size of main memory is to be used.

Cache Architecture

There are two types of Cache Architectures
1] Look Through
2] Look Aside

Look Through Architecture



* In this architecture two seperate buses are used. i.e processor bus & system bus.
* When CPU needs something, he will go the cache memory by using Processor bus & if the requested information is present in cache memory it will be immediately given to CPU by using data bus
* When CPU is accessing cache memory it uses processor bus & and at that time system bus is totally free.
* As shown in diagram all other bus masters are connected to main memory by using system bus & finally main memory is connected to Cache memory.
* When CPU is accessing Cache memory by using Processor bus at that time any other bus master can access Main memory. Thus concurrent operations are possible in this architecture which will improve the performance.

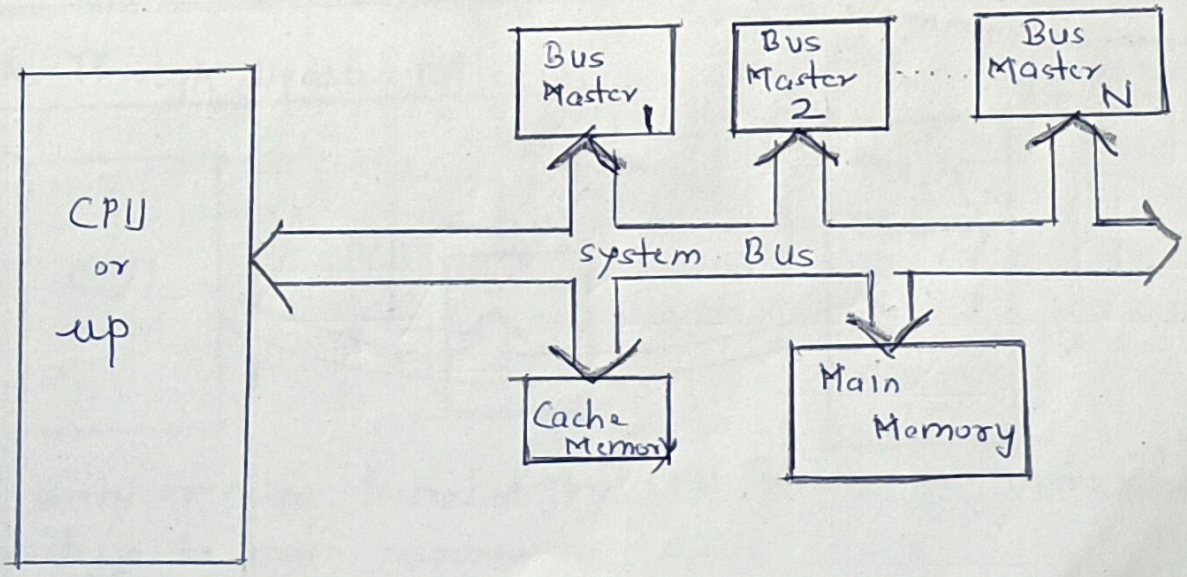
Advantages

1] Concurrent operations are possible

Disadvantages

1) Because of two seperate buses design is complex & costlier
2] In case of cache miss this method exhibits Look up Penalty i.e extra time component to check cache memory as well as main memory (one by one) in case of cache miss.

Look Aside Cache Architecture



In this method Cache memory main memory & all other bus masters are connected to CPU by using a common bus i.e system bus.

The cache controller (not shown in fig) monitors each memory request to see if cache memory contains copy of required information. The cache controller sits aside for this monitoring & hence the name is given as look-aside architecture.

When processor needs something, he will check in cache mem & if it is found in cache (cache hit), he will get it from cache. If it is not found in cache (cache miss), it will be obtained from main memory.

In case of Cache hit the main memory will start the access which is actually not required. Therefore there is unnecessary memory precharge cycle. This means that any other bus master needing access to memory must wait until the precharge cycle has completed.

Advantage

1] Simple design & hence less costly

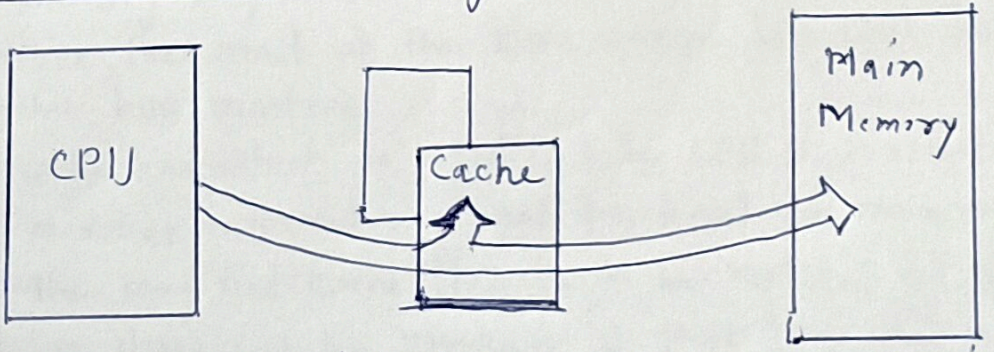
2] No Look up penalty i.e better response time in case of cache miss

Disadvantage

No concurrent operations are possible as all bus masters and up share a common system bus.

Types of update/Write Policies

1] Write Through Update Policy

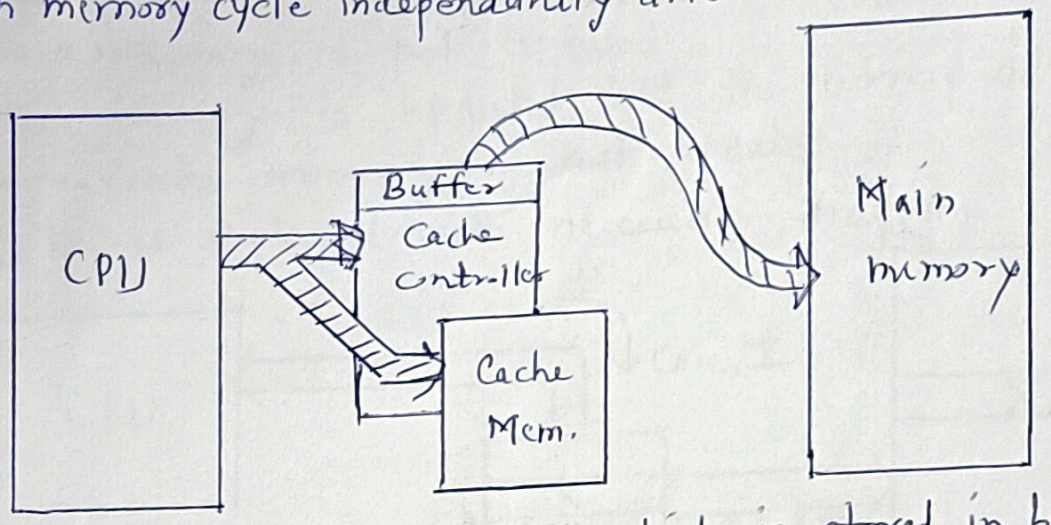


Each write operation initiated by CPU passes the information immediately to main memory as shown.

In short, main memory is immediately updated, therefore it

ontains always fresh & valid data
This implementation is very simple & effective but results in poor performance since each write operation must access slower main memory.

* Buffered Write Through / Posted Write
* When a write operation occurs in buffered write through method the cache controller stores the entire write operation in buffer while putting it into cache
* Therefore, there is no wait state. The controller completes the main memory cycle independantly after some time



* The updated information in cache which is stored in buffer is transferred to main main memory through system bus when any other bus master is not using it (Assuming look through Archi)
* The other bus masters connected to system bus are not allowed to access those main memory locations whose modifications are stored in buffer.

* Write Back Policy:
* This method improves overall system performance by updating the main memory only when it is necessary to update
* Therefore, for most of the time system bus will be free for use by other bus masters.
* The cache line which is modified by CPU is marked as modified/dirty
* This modified information is not transferred to main mem. immediately
* When the modified cache line is to be replaced by main memory line, which is demanded by processor at that time, that modication in cache line is transfered to corresponding location of main memory.
* This policy may encounter cache inconsistancy problem when main memory location is updated by one of the bus master which is already modified in cache by CPU.

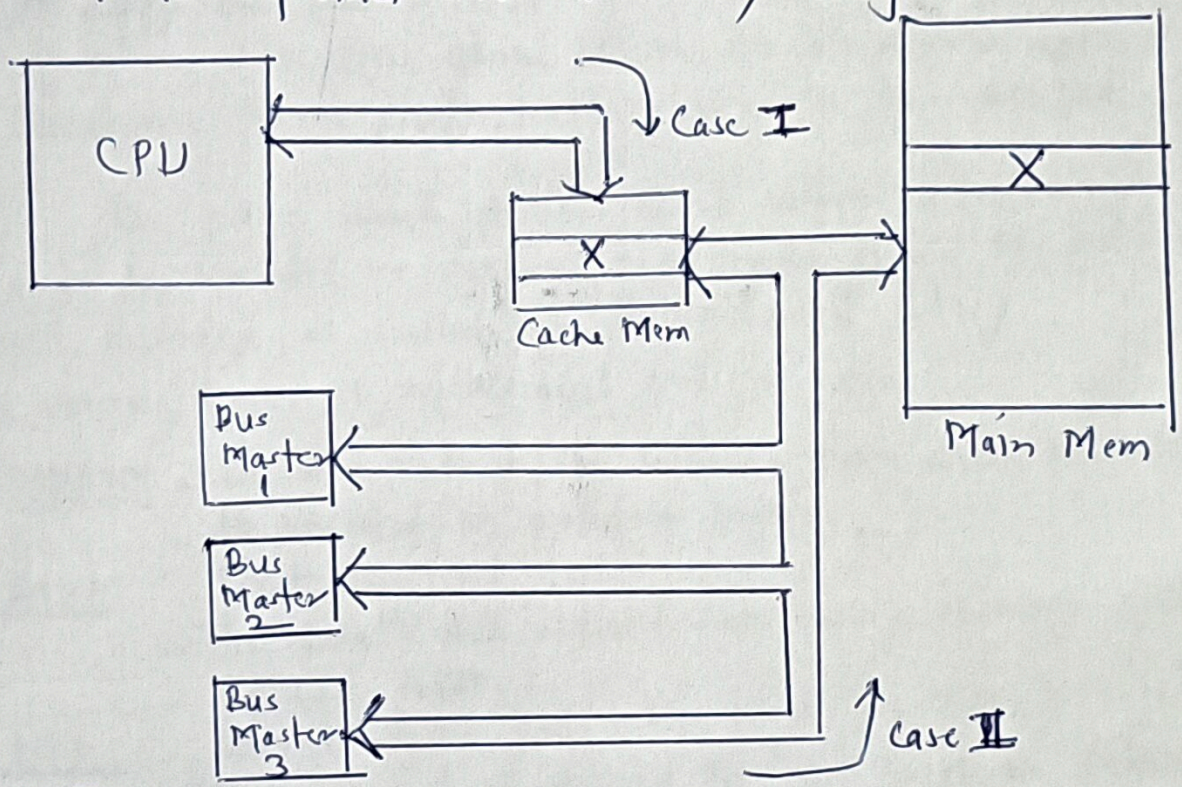
Cache Consistency / Coherency

If we want proper operation of cache subsystem, then contents of cache and main memory must be exact duplicate. There are several instances in which locations stored in cache memory are update but the corresponding location of main is memory is not updated & it contains old or stale information.

There are two possible conditions for this cache inconsistancy problem

1) Cache memory is updated but the corresponding location from main memory is not updated.
2) Main memory is updated but the corresponding location from cache memory is not updated.

It is explained with necessary diagram



Case I

Line X from Cache memory is updated by CPU while the same line from MM is not updated. This line X from MM migh be reffered by any other bus master connected to system bus (This information is old or stale)

Case II

Line X from main memory is updated by one of the bus master connected to system bus (MM is updated), while the same location which is mapped into CM is not accordingly updated (cache contains old or stale information)

In either of these 2 cases either MM or CM is updated while its duplicate contains state information. This problem is called inconsister

Replacement Policies used in Cache memory

* Whenever CPU needs something from memory, it will first check for into cache memory.
* If it is found in cache memory (cache hit), it will be directly provided to CPU
* But if it is not found in cache memory (cache miss), then it is obtained from main memory.
* The desired data from main memory is not only given to CPU but at the same time it is copied into cache mem.
* If there is space available in cache memory, then this data is copied into vacant locations of cache memory.

* But if cache memory is full i.e. no space is available to copy the data from main memory, in that case question will arise about where to store the data or which locations from cache memory are to be vacated.

The policy ~~which~~ that decides about which cache memory locations are to be vacated or replaced to provide space for data from main memory, is called as Replacement Policy

The various types of replacement policies are

1] RANDOM : In this method the locations from cache memory will be vacated in random manner.

2] FIFO : First In First Out
The data which entered into cache memory will be replaced first

3] LIFO : Last in first Out (Also known as FILO)
The data which entered in last will be taken out first

4] LRU : Least Recently Used
The data from cache memory which is least recently used or data which is not used for long time from cache memory will be replaced first because chances of using that in future will be very very less

LRU is the best Replacement Policy.

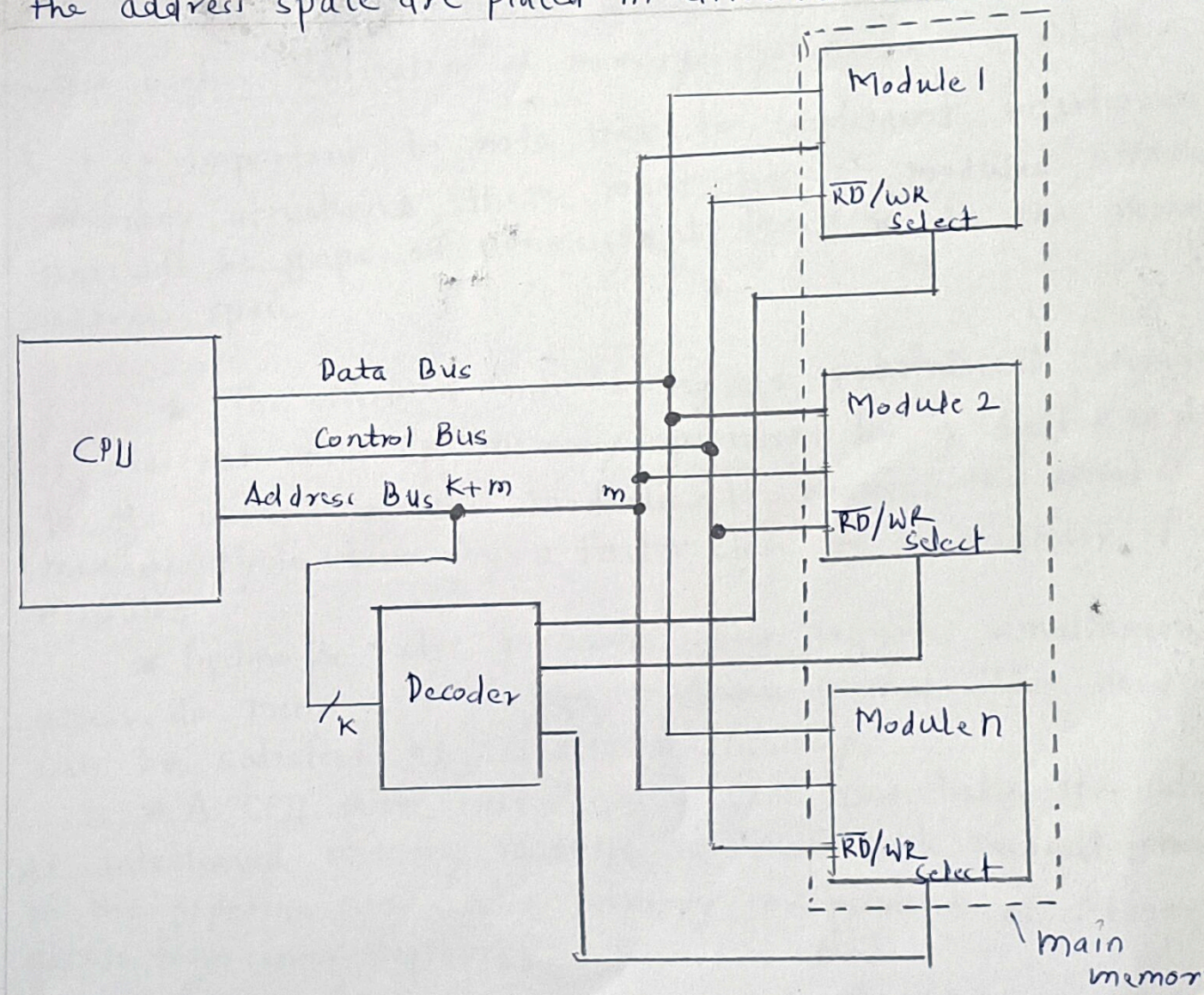
Interleaved Memory / Memory Interleaving

* We know that performance of memory is measured in terms of Access Time. Less is the access time faster is the memory.

* But more access time (slower memories) is one of the biggest problem in improving overall system performance

* One way to reduce the access time is to use cache memory

An alternative technique to reduce memory access time is memory interleaving. In this technique, the main memory is divided into a number of memory modules and the addresses are arranged such that the successive words in the address space are placed in different modules as shown



* Most of the time CPU accesses consecutive memory locations. In such situations address will be to different memory modules.

* Since these modules can be accessed in parallel, the average access time of fetching word from the main memory can be reduced.

* The lower order 'k' bit of memory address are generally used to select a module and the higher order 'm' bits are used to access a perticular location within the selected module.

* In this way consecutive addresses are located in successive modules. Thus any component of the system that generates requests for access to consecutive memory locations can keep several modules busy at any one time.

* This results in both faster access to a block of data and higher utilization of memory system as a whole.

* It is important to note that to implement interleaved memory structured, there must be 2^k modules, otherwise there will be gaps of nonexistant locations in the memory address space

* The effect of interleaving is substantial. However it does not speed up memory operation by a factor equal to the number of the module. It reduces the effect memory cycle time by a factor close to the number of modules.

* Pipeline & vector processors often require simultaneous access to memory from two or more sources. This need can be satisfied by interleaved memory.

* A CPU with inst. pipeline can also take the advantage of interleaved memory modules so that each segment (stage) in the pipeline can access memory, independent of memory access from other segments.