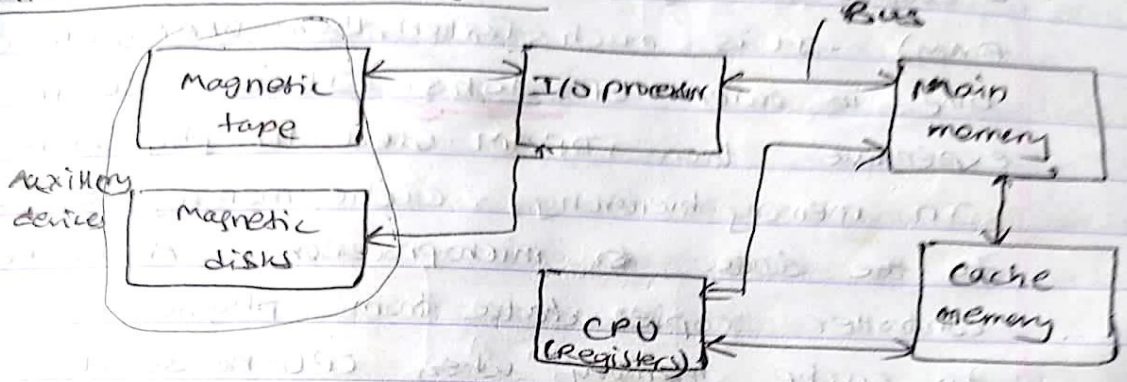
Chapter 07: Memory Organization

Chapter 7

Memory organisation

Hierarchial memory system:



(fig: Memory hierarchy in computer system)

The memory unit that communicates directly with CPU is called main memory (physical memory → DRAM). The devices that provides backup storage are auxilary memory. (magnetic tapes & disks)

The memory hierarchy system consists of all storage devices employed from the slow high capacity auxillary memory to a relatively faster main memory to an even smaller & faster cache memory accessible to high speed processing logic.

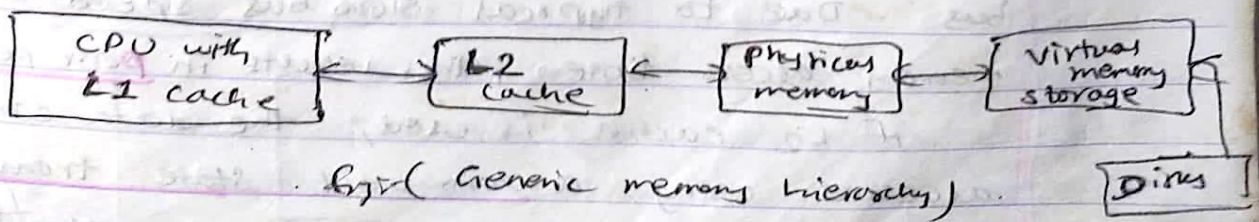


fig: (Generic memory hierarchy)

Cache memory:

- Cache memory is constructed with SRAM (static RAM). It is much faster than DRAM., with access time in order of 10ns. It is also much more expensive than DRAM used for physical memory

In memory hierarchy, cache memory is closest to the microprocessor. A cache controller copies data from physical memory to cache memory when CPU needs it.

Cache memory consists of two levels: level 1 (L_1) cache and level (L_2) cache
(internal cache) (external cache)
(outside CPU)

The reasons for including L_2 (Level 2) cache are :-

if there is no L_2 cache & the processor makes an access requests for a memory location not in L_1 cache, then procesor must acces DRAM physical memory across the bus. Due to typical slow bus speed & slow memory access time, this results in poor performance

if L_2 cache is used, the data can be accessed using a zero wait state transaction. And use of separate bus rather than system bus.

Virtual memory uses auxillary storages such as disk to expand the memory space available to the processor. It is much less costly than adding physical memory to a system & does not significantly degrade system performance. A memory management unit maps the logical addresses issued by processor to their corresponding physical addresses

* Associative memory (Content addressable memory)

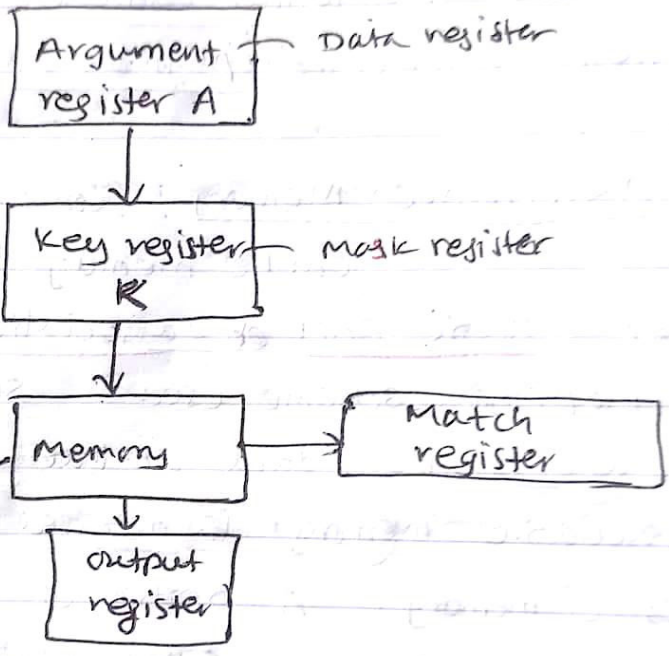
(cache memory is constructed using either static RAM or associative memory, depending on mapping scheme used. SRAM receives an address and access the data at that address.

Associative memory is accessed differently than other types of memory.) A portion of data is specified to access data in associative memory (The associative memory searches all of its locations in parallel & marks the locations that match the specified data input. The matching data values are then read out sequentially.)

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

For this reason, associative memories are used in application where search time is very critical & short.

Consider a ~~small~~ simple associative memory consist of 8 words, each with 16 bits.



	Data (memory words)				V
word 1	0000	1111	0000	1111	1
word 2	0000	1011	1000	0000	0
word 3	1000	1000	0011	1101	1
	1111	1111	0100	1001	1
	1000	1000	0011	1101	0
	0011	0000	1010	0000	1
	1010	1101	0000	0111	1
	1010	0000	0000	0000	0

for given exam
1
1

Each word has one additional bit labeled V. This is known as valid bit.

So, V=1, indicates that the word contains valid data & V=0 indicates that data is not valid.

Now, to read a value from associative memory, CPU must specify data value to be matched & bits of this data value that are to be checked. The first value is argument or data the second is mask or key. if the CPU wants to access data in associative memory that has 1010 as its four high order bits. The CPU would load

1111 0000 0000 0000 into mask register (key register)

The CPU also loads value -

1010 xxxx xxxx xxxx on data register

The 4 leading ~~octets~~ bits are value to be matched & remaining 12 bits can have any value since they will not be checked.

Now, the associative memory checks each location in parallel

A match occurs if 1 for every bit position that has a value of 1 in mask

register, a location bits are same as those in data register

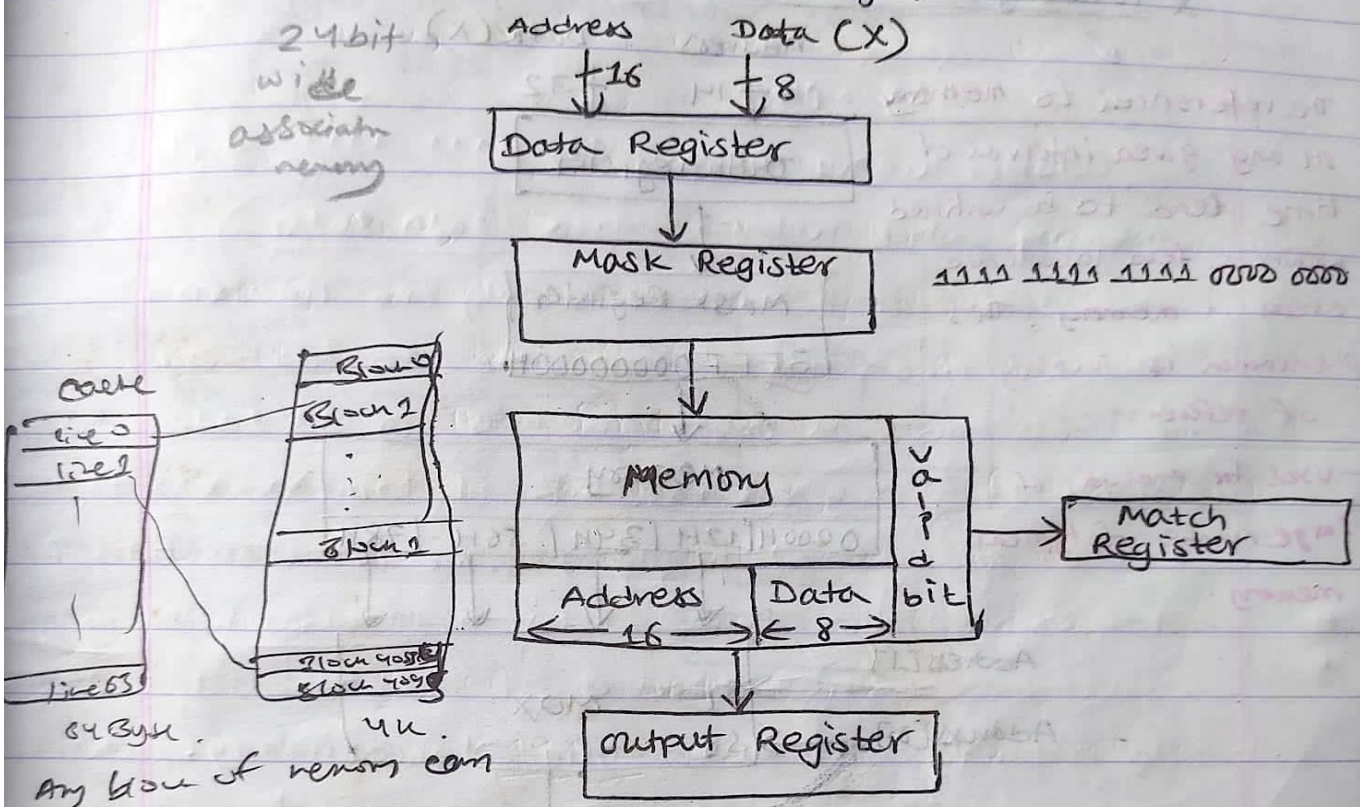
ii) the location bits are same valid bit is (V) set to 1.

⇒ Each memory location in associative memory contains hardware to perform this operation. The associative memory contains a match register which has one bit for each location in associative memory. If a location generates a match, its bit in match register is set to 1 otherwise it is set to 0.

* Cache memory with associative mapping :—

- Associative cache
- cache with associative memory.
- Let the associative memory, of 24 bits wide.
- The First 16 bits of each location consist memory address & last 8 bits hold data at that address
- The processor outputs the address it wishes to access to data register

- Also output the value 1111 1111 1111 1111 0000 0000 to its mask register.

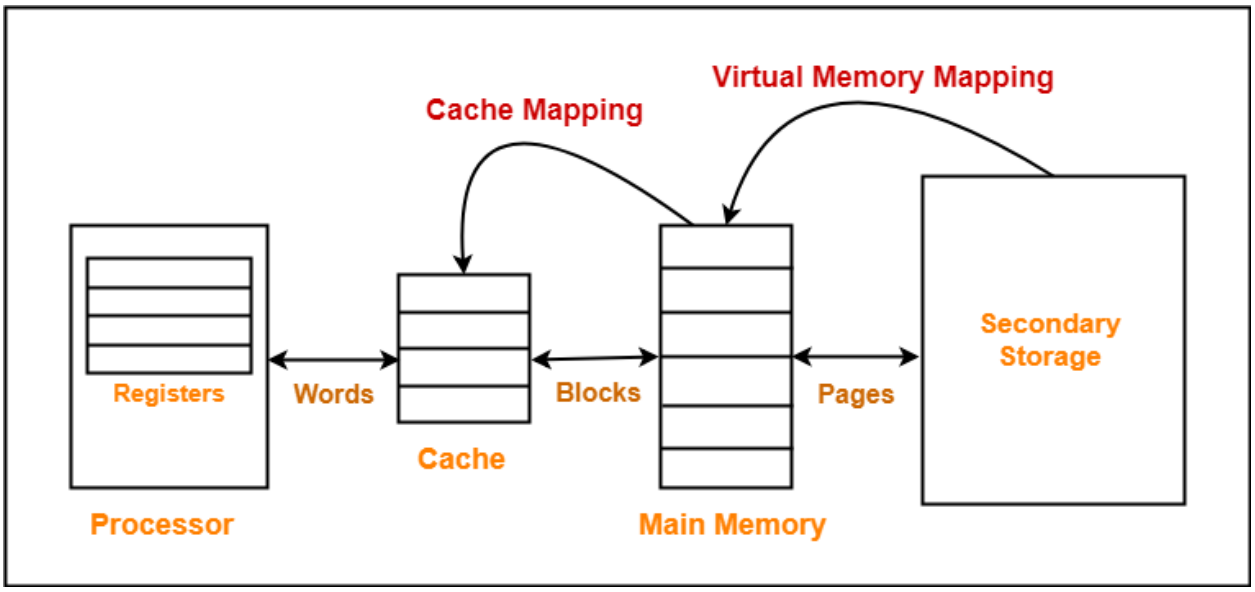


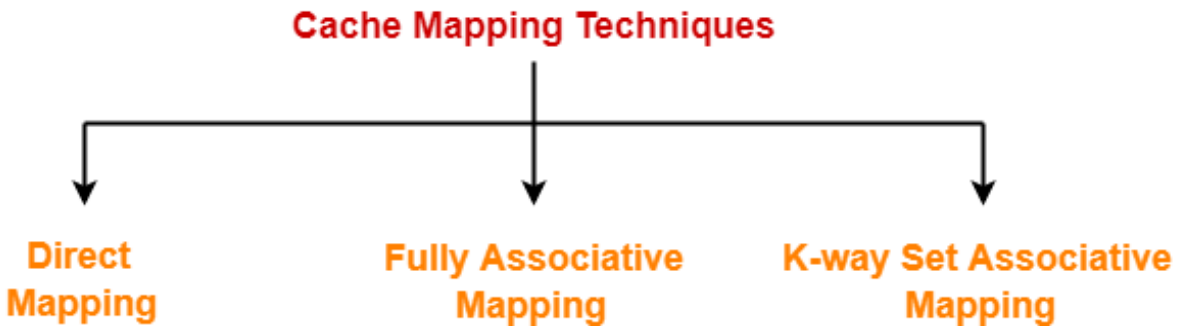
In this configuration, the instruction is read from physical memory & copied into cache. ~~(location 0)~~ when instructed is executed, CPU request the instruction at location 1, which is not in cache. This instruction also need to fetched from physical memory & copied into cache. The same happens for instruction at location 2 & 3.

Cache Mapping-

Cache mapping defines how a block from the main memory is mapped to the cache memory in case of a cache miss.

Cache mapping is a technique by which the contents of main memory are brought into the cache memory.



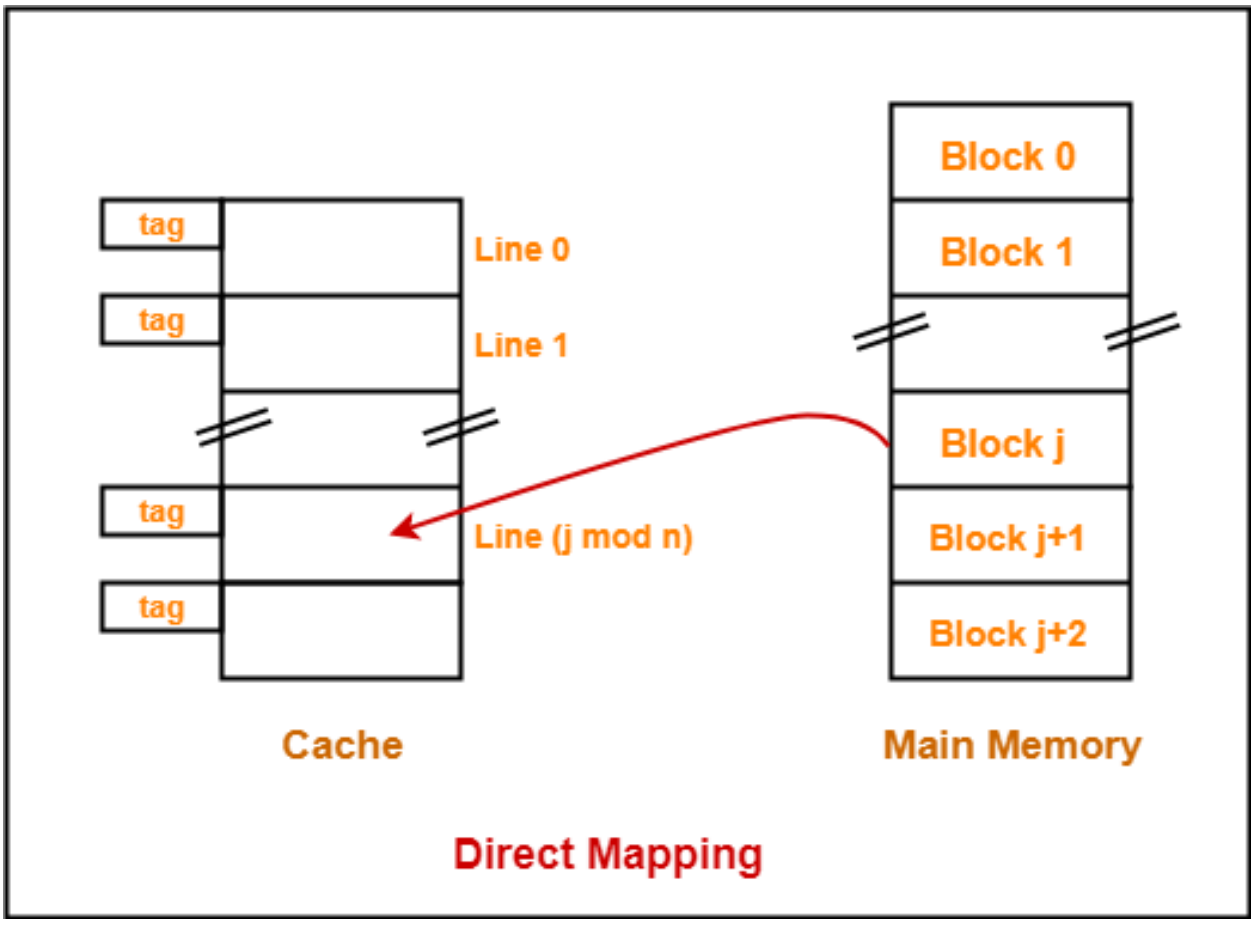


1. Direct Mapping
2. Fully Associative Mapping
3. K-way Set Associative Mapping

In direct mapping,

- A particular block of main memory can map only to a particular line of the cache.

Consider cache memory is divided into ‘n’ number of lines.
Then, block ‘j’ of main memory can map to line number (j mod n) only of the cache.



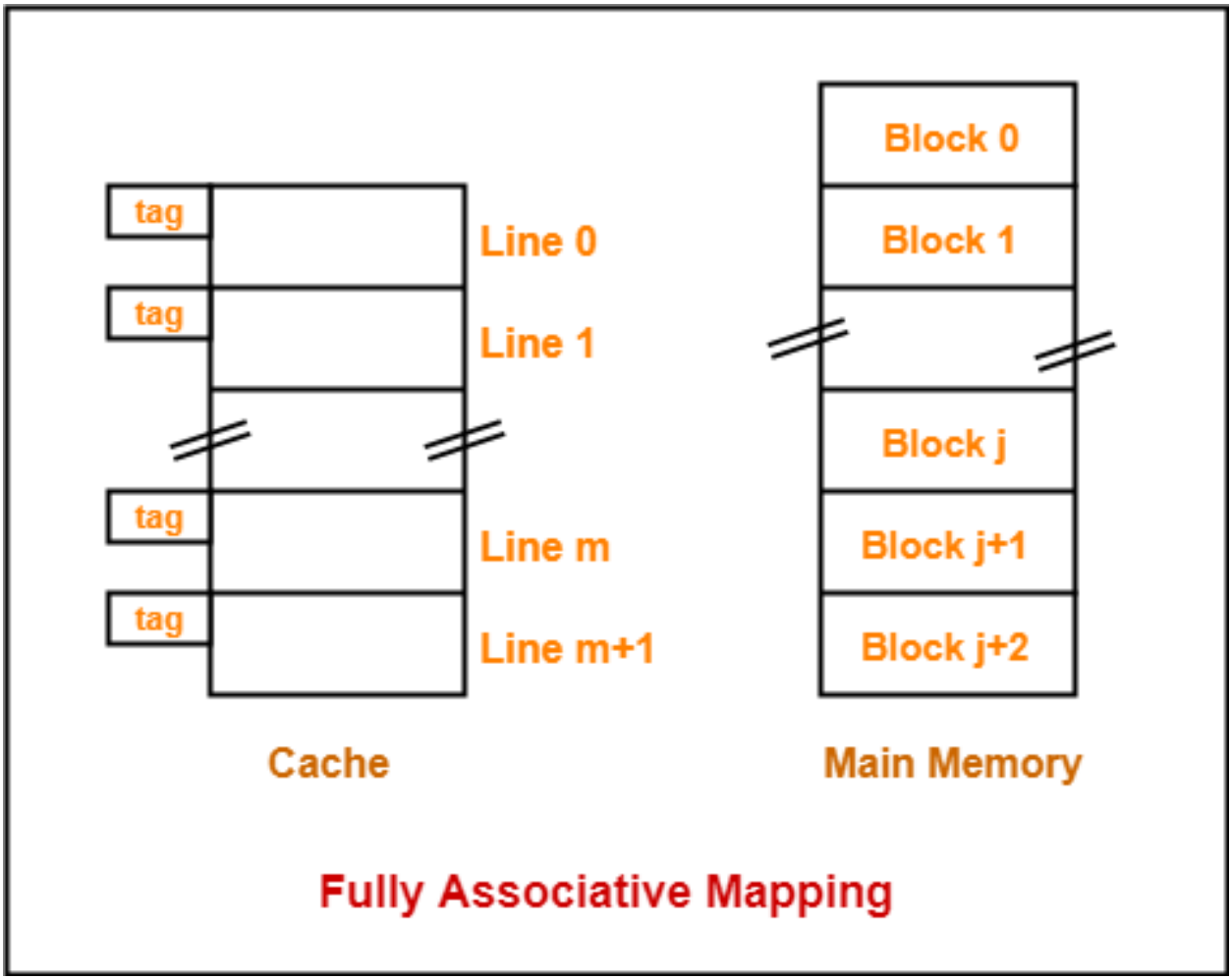
In direct mapping,

- There is no need of any replacement algorithm.
- This is because a main memory block can map only to a particular line of the cache.
- Thus, the new incoming block will always replace the existing block (if any) in that particular line.

2. Fully Associative Mapping

In fully associative mapping,

- A block of main memory can map to any line of the cache that is freely available at that moment.
- This makes fully associative mapping more flexible than direct mapping.



Here,

- All the lines of cache are freely available.
- Thus, any block of main memory can map to any line of the cache.
- Had all the cache lines been occupied, then one of the existing blocks will have to be replaced.

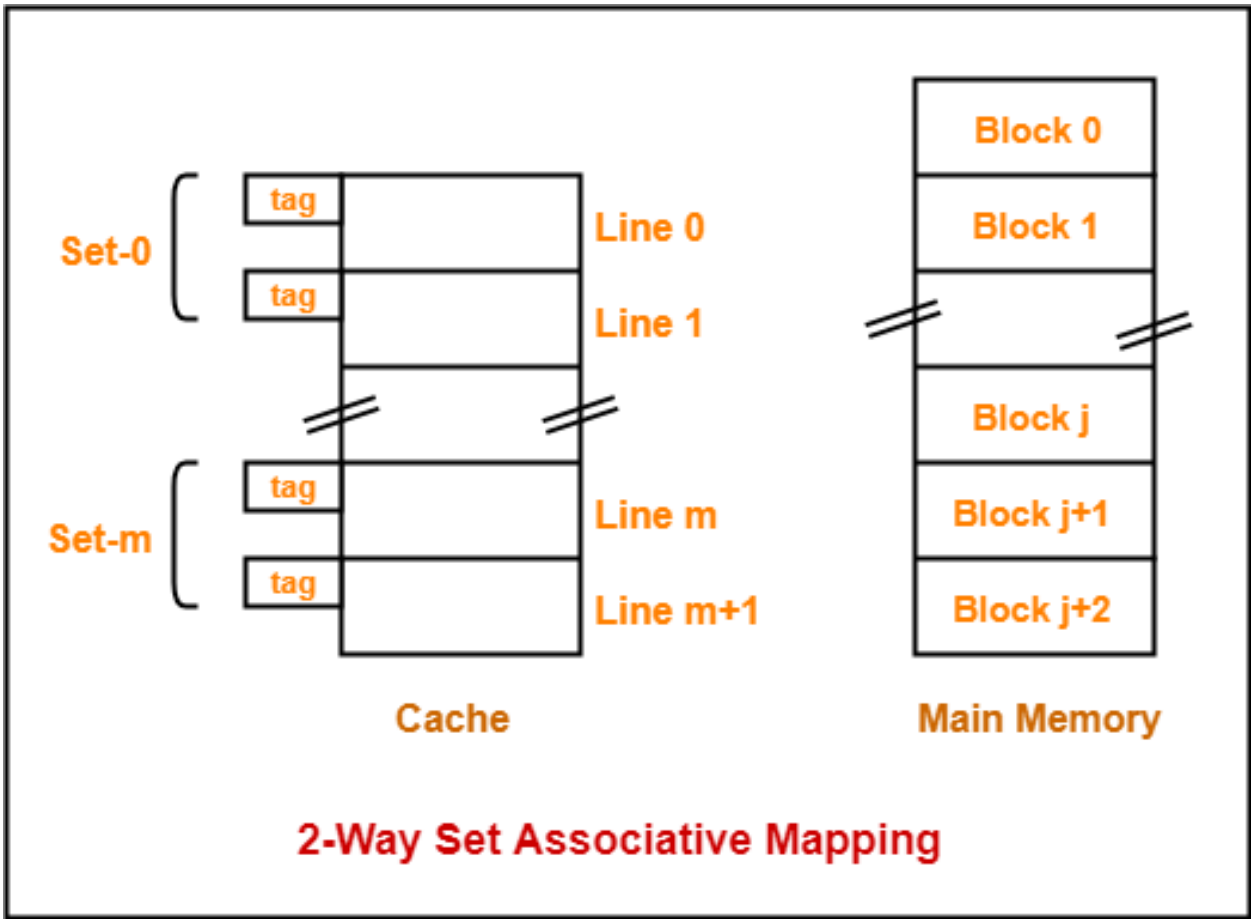
In fully associative mapping,

- A replacement algorithm is required.
- Replacement algorithm suggests the block to be replaced if all the cache lines are occupied.
- Thus, replacement algorithm like FCFS Algorithm, LRU Algorithm etc is employed.

3. K-way Set Associative Mapping-

In k-way set associative mapping,

- Cache lines are grouped into sets where each set contains k number of lines.
- A particular block of main memory can map to only one particular set of the cache.
- However, within that set, the memory block can map any cache line that is freely available.



Here,

- k = 2 suggests that each set contains two cache lines.
- Since cache contains 6 lines, so number of sets in the cache = 6 / 2 = 3 sets.

- Block ‘j’ of main memory can map to set number (j mod 3) only of the cache.
- Within that set, block ‘j’ can map to any cache line that is freely available at that moment.
- If all the cache lines are occupied, then one of the existing blocks will have to be replaced.

- Set associative mapping is a combination of direct mapping and fully associative mapping.
- It uses fully associative mapping within each set.
- Thus, set associative mapping requires a replacement algorithm.

If k = 1, then k-way set associative mapping becomes direct mapping

Define Hits & Miss?

⇒ The performance of cache memory is frequently measured in terms o... called hit ratio. when CPU refers to memory & finds the word in cach... said to produce a hit. if the word is not found in cache, then it is ... main memory it counts as miss.

Design issues of cache memory:

a) **Cache size:** cache size should be small so that average cost per bit is close to that of main memory.
The larger the cache, it is slower.

b) **Mapping techniques:** It specifies how the cache is organised. Direct, Associative, set associative.

c) **Replacement algorithms:** when cache is filled up, & a new block is brought into cache, one of existing block must be replaced. Replacement algorithm is needed for associative & set associative mapping. To achieve high speed, such algorithm must be implemented in hardware.
The 4 most common replacement algorithms are

i) **LRU:**
- Least recently used.
- most effective.
- replace that block in the set that has been in cache longest with no reference to it.
- easily implemented in two way associative.

ii) **FIFO:**
- First IN First out.
- replace that block in a set that has been in cache longest.
- FIFO is easily implemented as Round-R... or circular buffer technique.

iii) LFU :- least frequently used.
- replace that block in the set that has experiences fewest references.
- LFU can be implemented by associating a counter with each line.

iv) Random replacement :
- pick a line at random from among the candidate lines.
- It has low performance.

d) write policy :

e) line size : For HPC systems, 64 & 128 byte cache line sizes are mostly used.

f) Number of caches.
L_1 & L_2 (level 1 & level 2) cache.

Virtual Memory

Virtual memory makes it appear to CPU that there is more physical memory than it is actually present. Virtual memory is a concept used in some large computers system that permits user to construct programs as though a large memory space were available, equal to the totality of auxillary memory.

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

working of virtual memory :-

A MMU (Memory management unit) moves data between physical Memory & storage device. Retrieving data from physical memory is much faster than accessing data from swap disk or swap file (storage device) A mechanism is needed to move data from swap disk to physical memory & viceversa.

Two methods :

a) Paging : The basic components of MMU are :-

a) A facility for dynamic storage relocation that maps logical memory references into physical memory addre

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

c) Protection of information against unauthorized access between users & preventing users from changing operating system functions

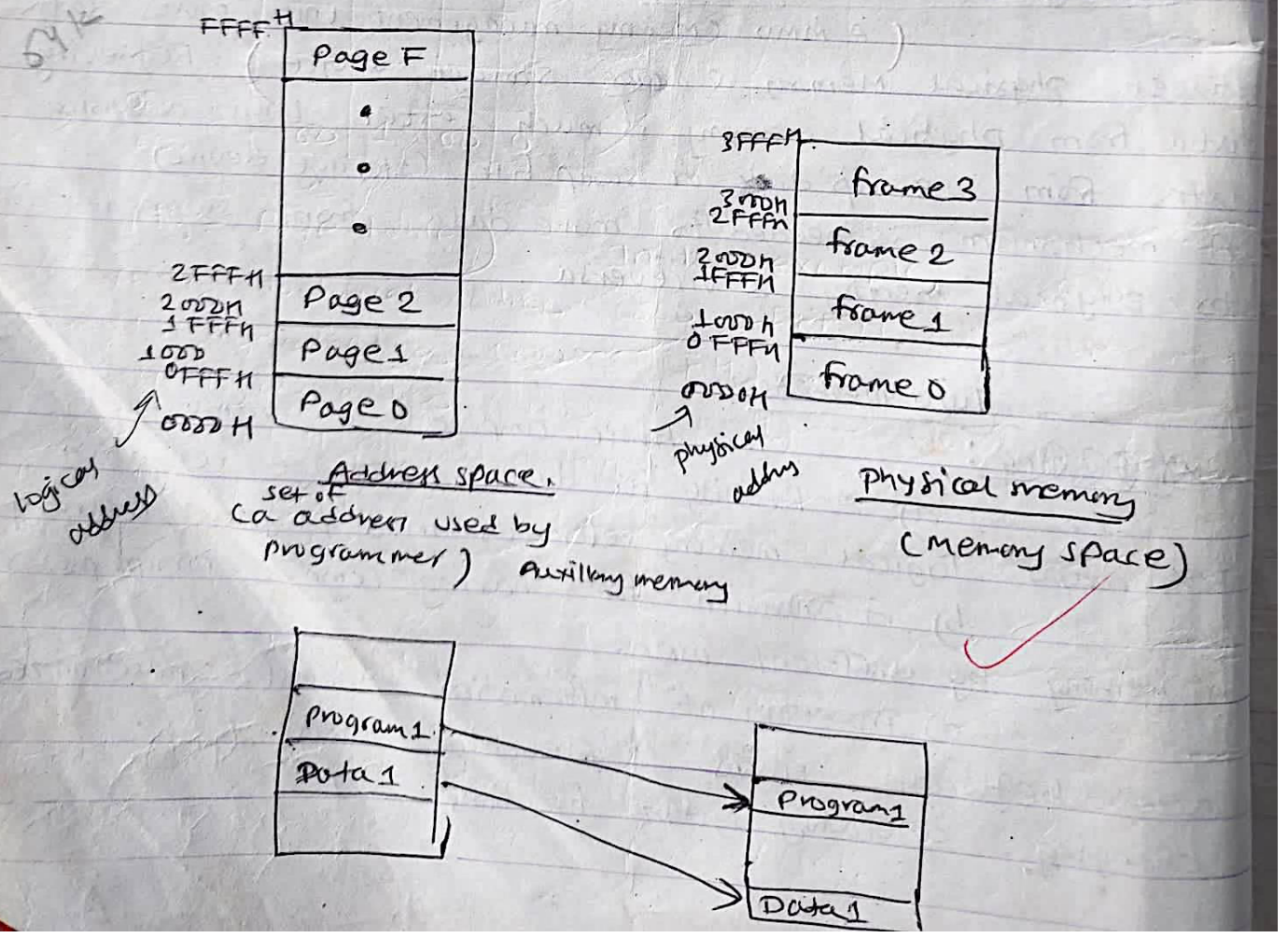
Two methods :-

a) Paging
b) Segmentation

a) Paging :-

The logical addresses is divided into contiguous block called pages. All pages are of same size & each logical address resides in exactly one page. Page will contain either instruction or data.

Physical memory is divided into non overlapping frames. The size of a frame is same as size of a page.



Paging moves pages from swap disk (auxillary storage) to frames of physical memory so data can be accessed by processor.

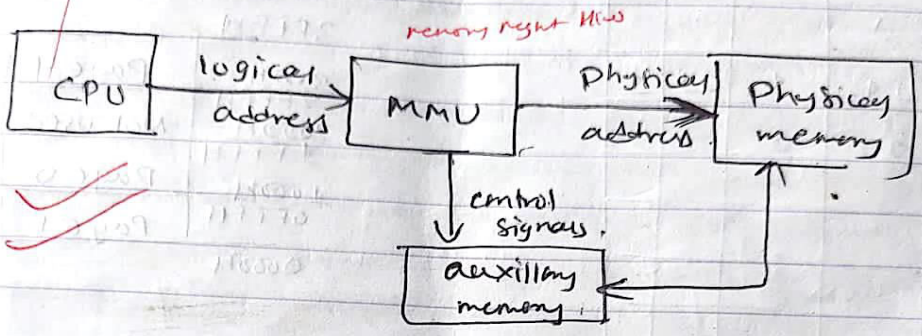


fig:- (MMU configuration within memory hierarchy)

Page table:

Page table is used by MMU to keep track which pages are stored in which frames.

Here, the page table contains 16 locations, one for each page in the logical address space. The columns of page table are :- frame, valid, count, & Dirty.

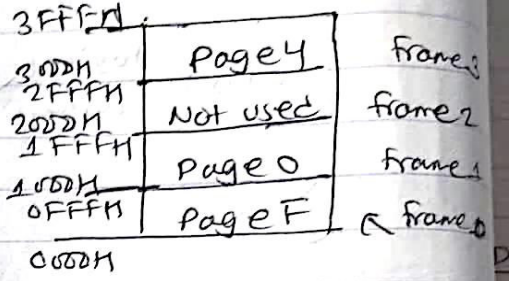
The frame field is of 2bit. It denotes the frame in which that page is stored.

The valid field is of 1 bit. It indicate the value represents valid data or not.

The other fields are count bits and dirty bit which are used when moving pages out of physical memory.

Page	Frame	Valid	count	Dirty
0	01	1	10	0
1				
2				
3				
4	11	1	01	1
5	00	0	00	0
6				
7				
8				
9				
A				
B				
C				
D				
E				
F	00	1	00	0

(fig:- Page table)



(fig:- physical memory)

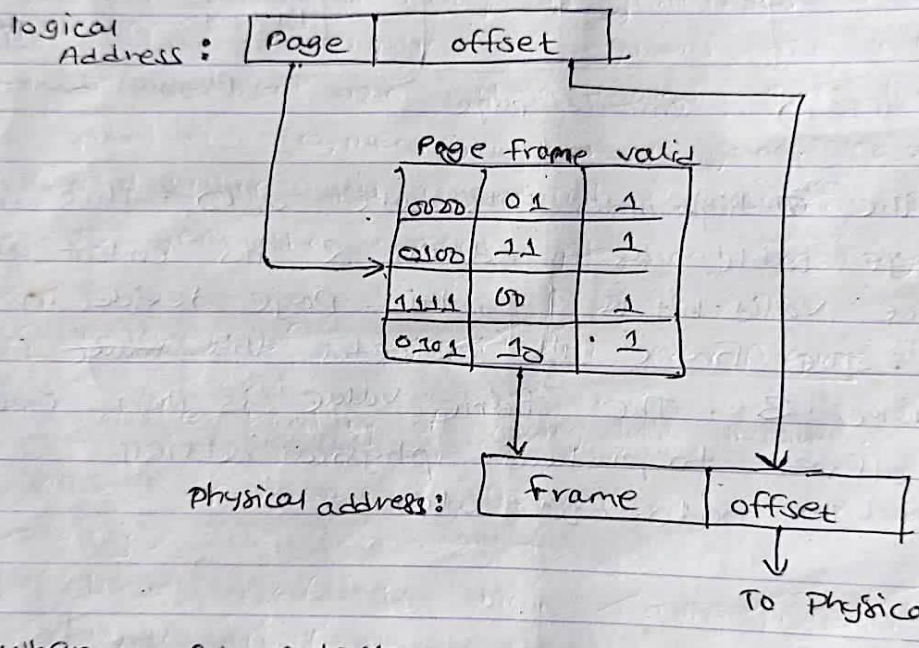
In this example, page 0 corresponding to logical address 0000 - 0FFFH is stored in frame 1. (physical address 1000 - 1FFFH).
Page 4 logical address (4000 - 4FFFH) is stored in frame 3 3000 - 3FFFH physical address.
Page 15 (F000H - FFFFH) is stored in frame 0 (0000 - 0FFF
Frame 2 is not used.

Now, lets see how page table is used to calculate the required physical address.
First of all, MMU treats logical addresses as two parts, a page number & an offset.

* Translation lookaside buffer : (TLB)

It is similar & perform same function as page table i.e generating the frame value. But it is faster & improve system performance.

The conversion of logical address to physical address using TLB :



⇒ when CPU outputs a logical address, the MMU forwards it to both its page table & TLB simultaneously. if the entry is stored in TLB, it will output the frame number, concatenate with the offset & send this address to cache & physical memory.

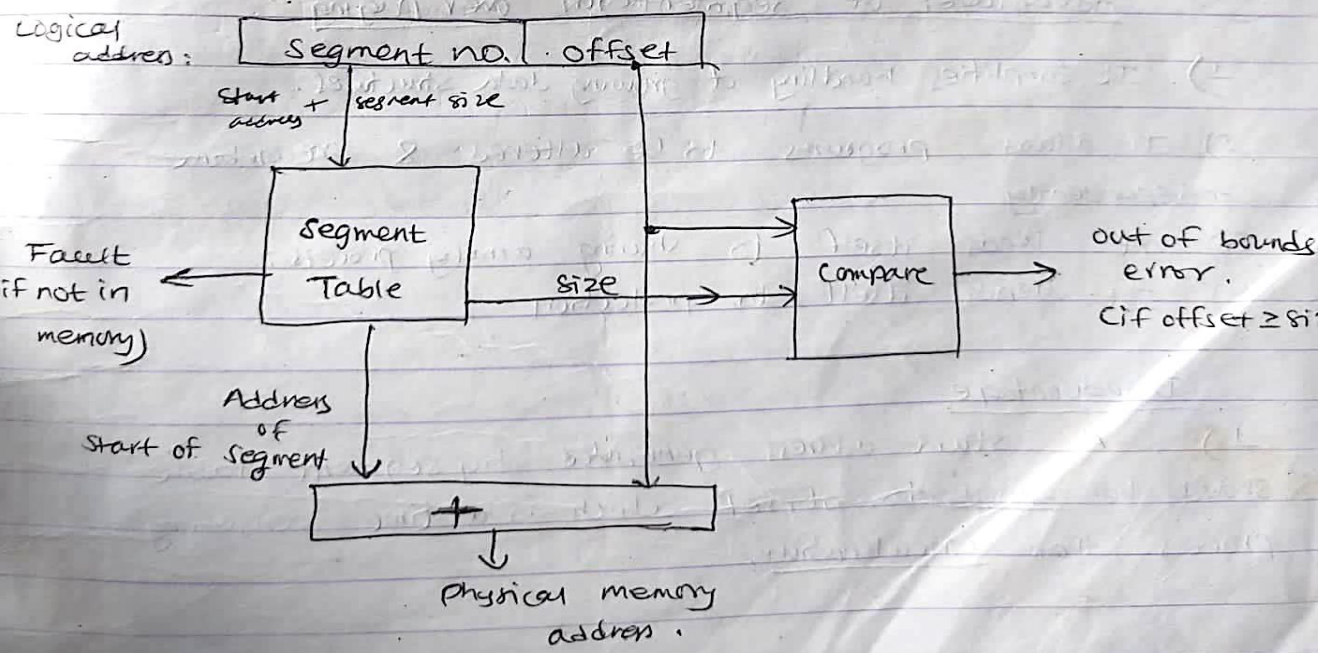
if there is no match, the address translation ~~process for~~ proceeds as before.

It can speed up the address translation process for recently access pages. Thus it can takes

advantage of locality of reference to improve system performance.
TLB does not have count & dirty bits.

b) **Segmentation:** Segmentation is the other method of allocating memory. It is a way to subdivide the addressable memory. (i.e Program is divided into several segments each of which is a self contained unit such as subroutine or data structure.

Paging is invisible to programmer & provide larger address space whereas segmentation is visible to programmer which organise programs & data. segments are of dynamic size. (vary in size).



A segment table keep tracks of segments resident in memory. Each segment table entry must include start address & segment size.

The logical address is partitioned into a segment number & an offset. The segment number is input to the segment table. if the segment is located in physical memory, outputs the address of start of segment & the segment size. if not, it generates a segment fault.

The offset is compare to segment size. if offset ≥ segment size it indicates that location is not a part of segment, error is generated. if offset has a valid value, it is added to start of segment address to generate correct physical memory address.

Advantages of segmentation over paging

1) It simplifies handling of growing data structures.
2) It allows programs to be altered & recombine independently
3) It lends itself to sharing among process.
4) It lends itself to protection

Disadvantage

1) A start address generated by segment table should be added to offset which is a time consuming process than concatenation.