interleaved memory

- Main memory divided into two or more sections.
- The CPU can access alternate sections immediately, without waiting for memory to catch up (through wait states).
- Interleaved memory is one technique for compensating for the relatively slow speed of dynamic RAM (DRAM).

INTERLEAVED MEMORY

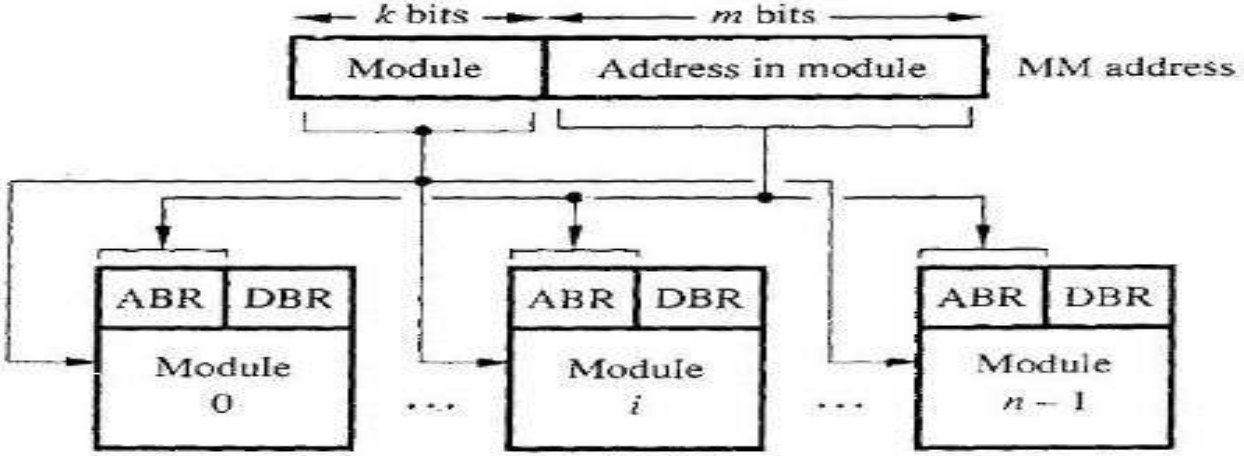
- Memory interleaving increases bandwidth by allowing simultaneous access to more than one chunk of memory.
- This improves performance because the processor can transfer more information to/from memory in the same amount of time.

INTERLEAVED MEMORY

- Interleaving works by dividing the system memory into multiple blocks.
- The most common numbers are two or four, called two-way or four-way interleaving, respectively.
- Each block of memory is accessed using different sets of control lines, which are merged together on the memory bus.
- When a read or write is begun to one block, a read or write to other blocks can be overlapped with the first one.
- The more blocks, the more that overlapping can be done.
- As an analogy, consider eating a plate of food with a fork. Two-way interleaving would mean dividing the food onto two plates and eating with both hands, using two forks. (Four-way interleaving would require two more hands. :^)) Remember that here the processor is doing the "eating" and it is much faster than the forks (memory) "feeding" it (unlike a person, whose hands are generally faster.)

INTERLEAVED MEMORY

- each with its own address buffer register (ABR) and data buffer register (DBR), memory access operations may proceed in more than one module at the same time.
- Two methods of address layout shown



(a) Consecutive words in a module

INTERLEAVED MEMORY

- In the first case, the memory address generated by the CPU is decoded as shown in part a of the figure
- The high order k bits name one of *n modules, and the low-order m bits name a particular word* in that module.
- When consecutive locations are accessed, as happens when a block of data is transferred to a cache, only one module is involved

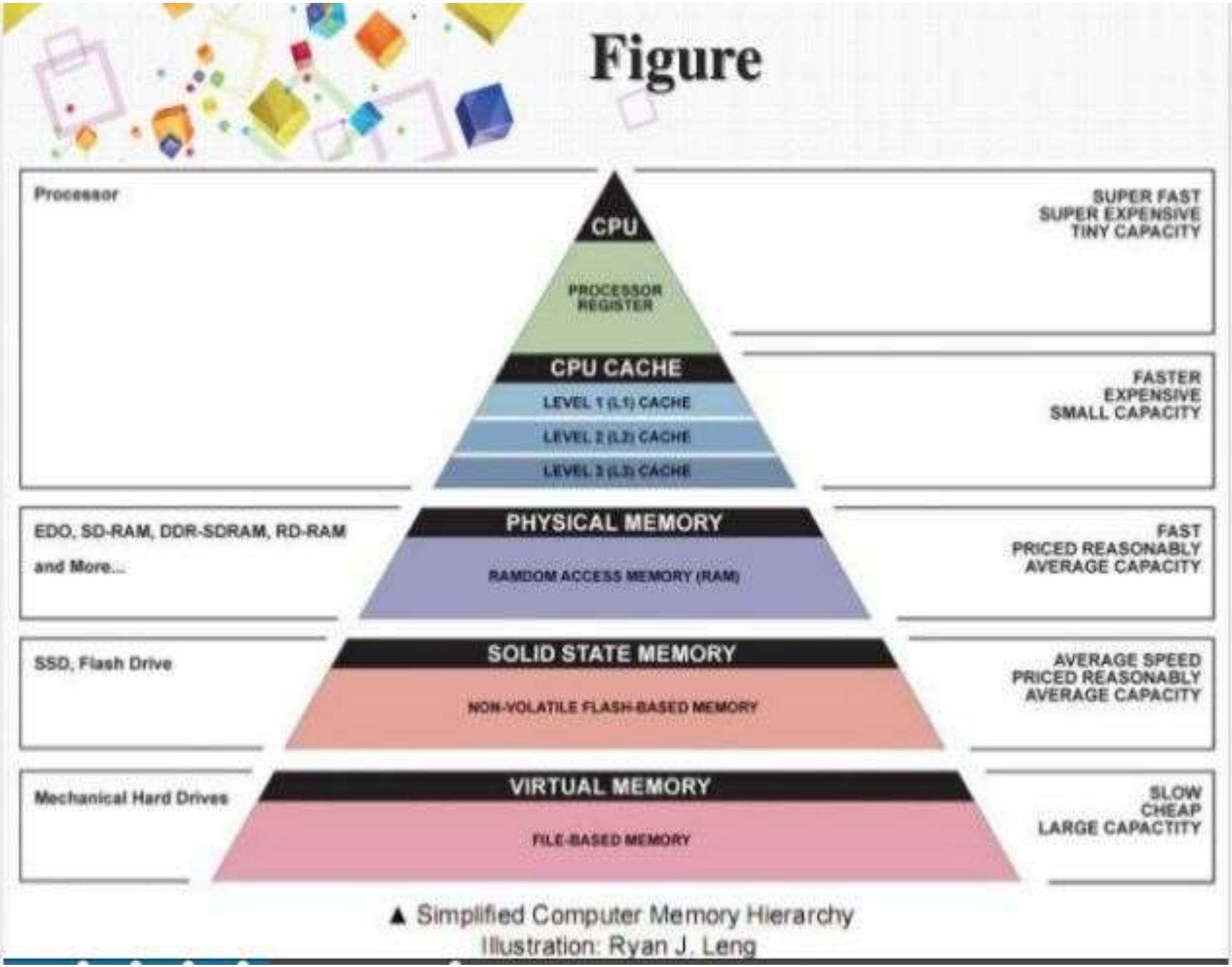
Memory Hierarchy

- The memory unit is an essential component in any digital computer since it is needed for storing programs and data
- Not all accumulated information is needed by the CPU at the same time
- Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by CPU

Memory Hierarchy

- Computer Memory Hierarchy is a pyramid structure that is commonly used to illustrate the significant differences among memory types.
- The memory unit that directly communicate with CPU is called the *main memory*
- Devices that provide backup storage are called *auxiliary memory*
- The memory hierarchy system consists of all storage devices employed in a computer system from the slow by high-capacity auxiliary memory to a relatively faster main memory, to an even smaller and faster cache memory

Figure



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

Main memory

- Most of the main memory in a general purpose computer is made up of RAM integrated circuits chips, but a portion of the memory may be constructed with ROM chips

- RAM– Random Access memory
  - Integrated RAM are available in two possible operating modes, *Static and Dynamic*
- ROM– Read Only memory

RAM

- A RAM chip is better suited for communication with the CPU if it has one or more control inputs that select the chip when needed
- Read/write memory, that initially doesn't contain any data
- The computing system that it is used in usually stores data at various locations to retrieve it latter from these locations
- Its data pins are bidirectional (data can flow into or out of the chip via these pins), as opposite to those of ROM that are output only
- It loses its data once the power is removed, so it is a volatile memory
- It has a directional select signal R/W'; When R/W'=1, the chip outputs data to the rest of the circuit; when R/W' = 0 it inputs data from the rest of the circuit

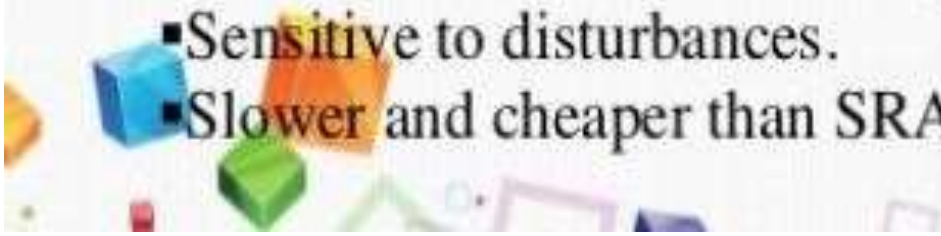
Random-Access Memory Types

Static RAM (SRAM)

- Each cell stores bit with a six-transistor (Diode) circuit.
- Retains value indefinitely, as long as it is kept powered.
- Relatively insensitive to disturbances such as electrical noise.
- Faster and more expensive than DRAM.

Dynamic RAM (DRAM)

- Each cell stores bit with a capacitor and transistor.
- Value must be refreshed every 10-100 ms.
- Sensitive to disturbances.
- Slower and cheaper than SRAM.



ROM

- ROM is used for storing programs that are **PERMENTLY** resident in the computer and for tables of constants that do not change in value once the production of the computer is completed.

- The ROM portion of main memory is needed for storing an initial program called *bootstrap loader,* witch is to start the computer software operating when power is turned off

ROM Types

- **Masked ROM** – programmed with its data when the chip is fabricated
- **PROM** – programmable ROM, by the user using a standard PROM programmer, by burning some special type of fuses. Once programmed will not be possible to program it again
- **EPROM** – erasable ROM; the chip can be erased and chip reprogrammed; programming process consists in charging some internal capacitors; the UV light (method of erase) makes those capacitors to leak their charge, thus resetting the chip
- **EEPROM** – Electrically Erasable PROM; it is possible to modify individual locations of the memory, leaving others unchanged; one common use of the EEPROM is in BIOS of personal computers

Auxiliary Memory

- The main memory construction is costly. Therefore, it has to be limited in size. The main memory is used to store only those instructions and data which are to be used immediately. However, a computer has to store a large amount of information. The bulk of information is stored in the auxiliary memory. This is also called backing storage or secondary storage. They include hard disk, floppy disks, CD-ROM, USB flash drives, etc.

- When the electricity supply to the computer is off, all data stored in the primary storage is destroyed. On the other hand, this is not true for secondary storage. The data stored in secondary storage can be stored for the desired time.

CACHE MEMORY

- If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program.
- **Such a fast small memory is referred to as a cache memory**.
- It is placed between the CPU and main memory as illustrated in Fig. 12-1 .
- The cache memory access time is less than the access time of main memory by a factor of 5 to 10. The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components.

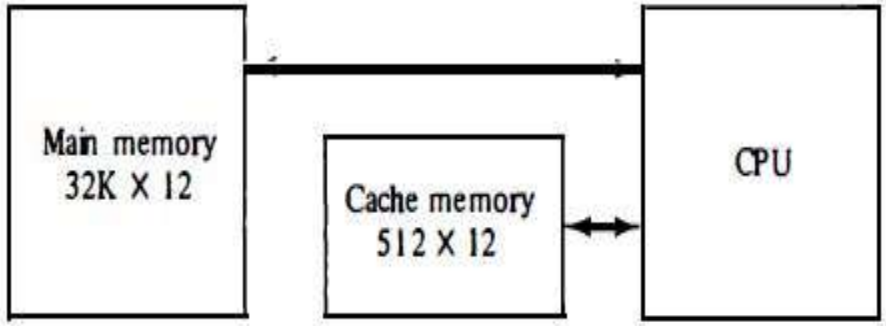


Figure 12-10 Example of cache memory.

CACHE MEMORY

- The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in the fast cache memory, the average memory access time will approach the access time of the cache.
- Although the cache is only a small fraction of the size of main memory, a large fraction of memory requests will be found in the fast cache memory because of the locality of reference property of programs.
- The basic operation of the cache is as follows. When the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory.
- If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word.
- A block of words containing the one just accessed is then transferred from main memory to cache memory.

CACHE MEMORY

- The performance of cache memory is frequently measured in terms of a quantity called hit ratio . When the CPU refers to memory and finds the word in cache, it is said to produce a 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 FUNCTIONS

- The basic characteristic of cache memory is its fast access time.
- Therefore,very little or no time must be wasted when searching for words in the cache.
- The transformation of data from main memory to cach memory is referred to as a **mapping process**.
- Three types of mapping procedures are of practical interest when considering the organization of cache memory:

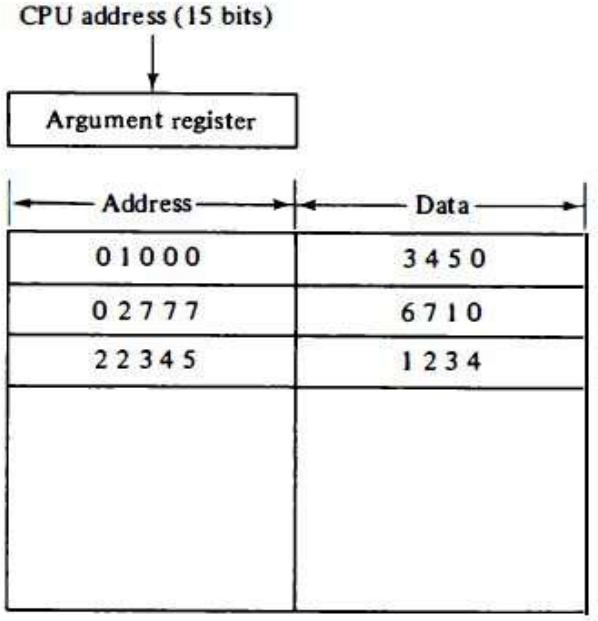
    1. Associative mapping

    2. Direct mapping

    3. Set-associative mapping

Associative Mapping

- The fastest and most flexible cache organization uses an associative memory. This organization is illustrated in Fig. 12-1 1 .
- The associative memory stores both the address and content (data) of the memory word.
- This permits any location in cache to store any word from main memory.
- The diagram shows three words presently stored in the cache.



Associative Mapping

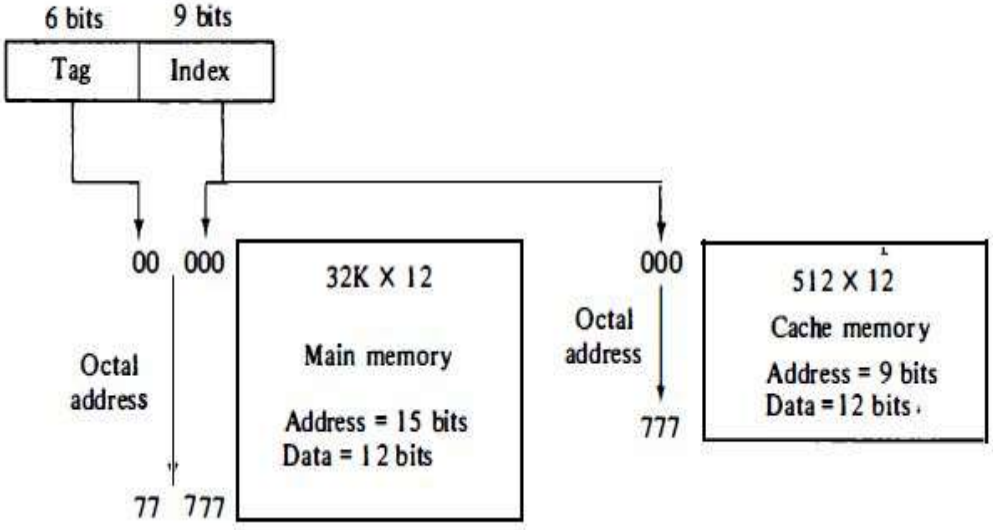
- The address value of 15 bits is shown as a five-digit octal number and its corresponding 12 - bit word is shown as a four-digit octal number.
- A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address.
- If the address is found, the corresponding 12-bit data is read and sent to the CPU.
- If no match occurs, the main memory is accessed for the word.

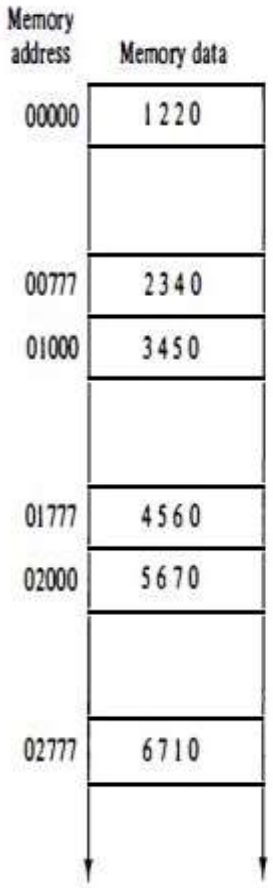
Direct Mapping

- Associative memories are expensive compared to random-access memories because of the added logic associated with each cell.
- The possibility of using a random-access memory for the cache is investigated in Fig.
- The CPU address of 15 bits is divided into two fields.
- The nine least significant bits constitute the index field and the remaining six bits form the tag field.
- The figure shows that main memory needs an address that includes both the tag and the index bits.
- The number of bits in the index field is equal to the number of address bits required to access the cache memory.

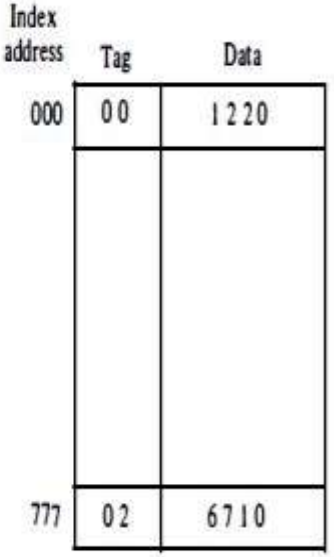
Direct Mapping

Figure 12-12 Addressing relationships between main and cache memories.





(a) Main memory



(b) Cache memory

Figure 12-13 Direct mapping cache organization.

Direct Mapping

- In the general case, there are 2' words in cache memory and 2" words in main memory.
- The n-bit memory address is divided into two fields: k bits for the index field and n - k bits for the tag field.
- The direct mapping cache organization uses the n-bit address to access the main memory and the k-bit index to access the cache.

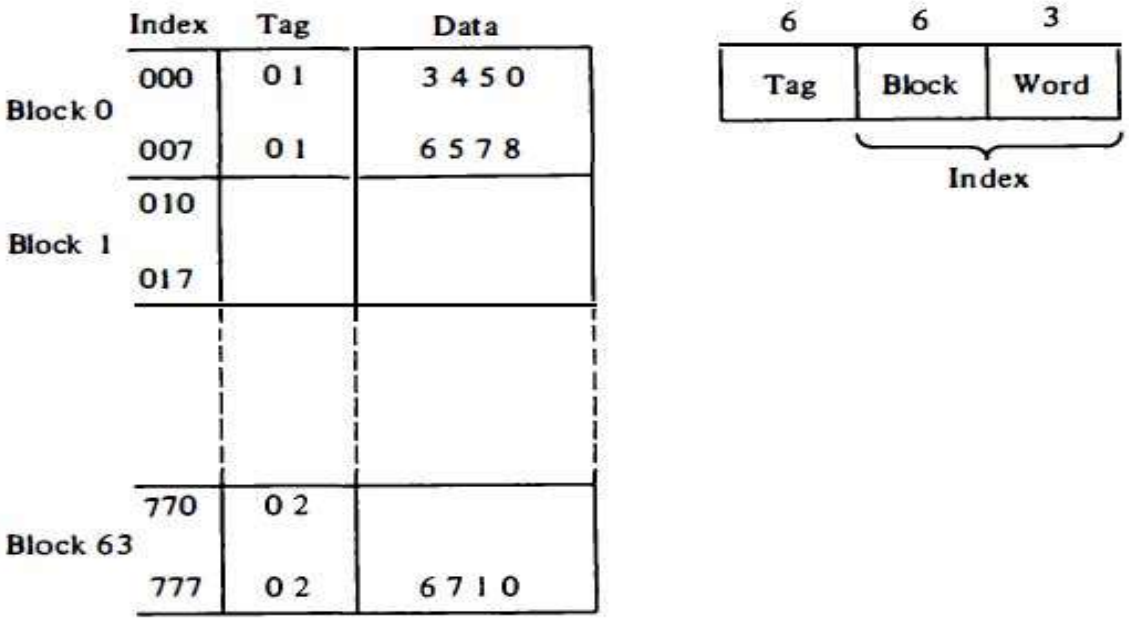


Figure 12-14 Direct mapping cache with block size of 8 words.

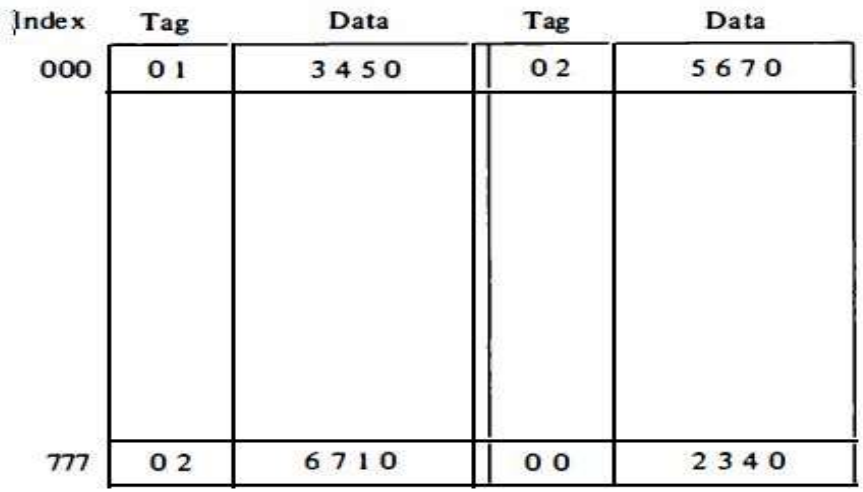
Set-Associative Mapping

- It was mentioned previously that the disadvantage of direct mapping is that two words with the same index in their address but with different tag values cannot reside in cache memory at the same time.
- A third type of cache organization, called set-associative mapping, is an improvement over the direct-mapping organization in that each word of cache can store two or more words of memory under the same index address.
- Each data word is stored together with its tag and the number of tag-data items in one word of cache is said to form a set.
- An example of a set-associative cache organization for a set size of two is shown in Fig. 12-15.

Set-Associative Mapping

- Each index address 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 address of nine bits can accommodate 512 words. Thus the size of cache memory is 512 x 36. It can accommodate 1024 words of main memory since each word of cache contains two data words.
- In general, a set-associative cache of set size k will accommodate k words of main memory in each

word of cache.



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

Figure 12-15 Two-way set-associative mapping cache.

Replacement Algorithms

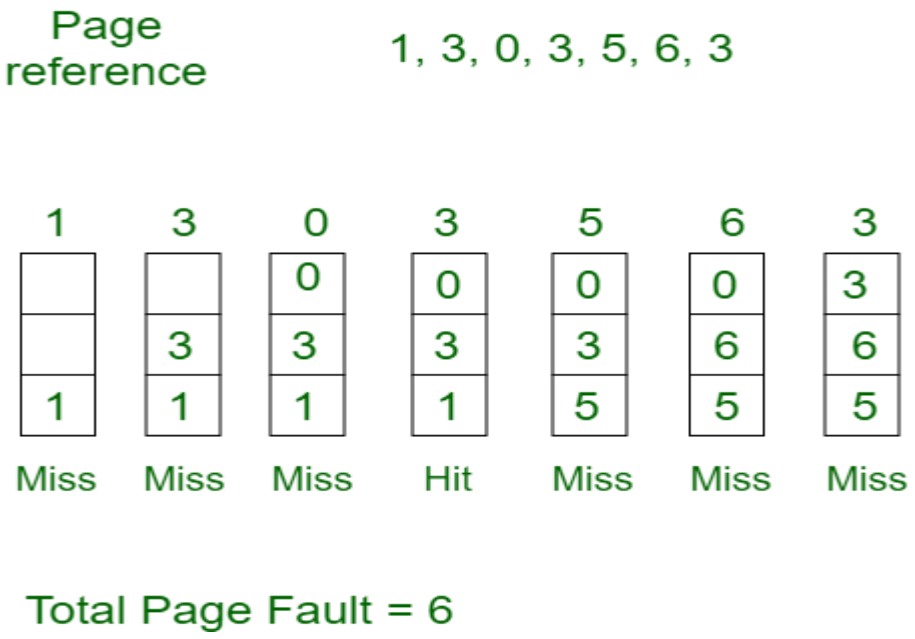
1. **First In First Out (FIFO)**
2. **Optimal Page replacement**
3. **Least Recently Used**

1.First In First Out (FIFO):

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

1.First In First Out (FIFO)

- **Example-1**Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames.Find number of page faults.

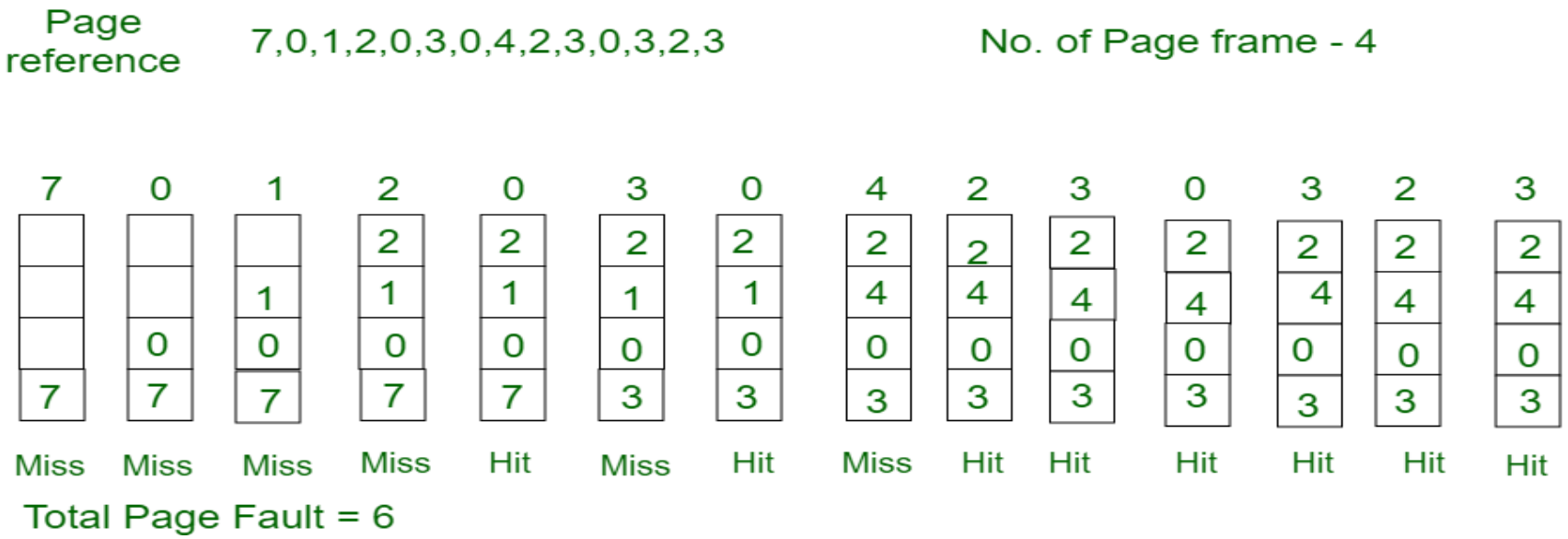


1.First In First Out (FIFO)

- Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**
- when 3 comes, it is already in memory so —> **0 Page Faults.**
- Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.**
- 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.**
- Finally when 3 come it is not avilable so it replaces 0 **1 page fault**

2.Optimal Page replacement

- In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.
- **Example-2:**Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame. Find number of page fault.



2.Optimal Page replacement

- Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
- 0 is already there so —> **0 Page fault.**
- when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.**
- 0 is already there so —> **0 Page fault.**
- 4 will takes place of 1 —> **1 Page Fault.**
- Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

- Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests.
- The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

3.Least Recently Used

- In this algorithm page will be replaced which is least recently used.
- **Example-3**Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames Find number of page faults.

Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3 No. of Page frame - 4

Total Page Fault = 6

Here LRU has same number of page fault as optimal but it may differ according to question.

3.Least Recently Used

- Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
- 0 is already their so —> **0 Page fault.**
  when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**
- 0 is already in memory so —> **0 Page fault**.
- 4 will takes place of 1 —> **1 Page Fault**
- Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Write Through and Write Back:

- **Write Through:**

  In write through, data is **simultaneously updated to cache and memory**. This process is simpler and more reliable. This is used when there are no frequent writes to the cache.

- **Write Back:**

  The data is updated only in the cache and updated into the memory in later time. Data is updated in the memory only when the cache line is ready to replaced (cache line replacement is done using Belady's Anomaly, Least Recently Used Algorithm, FIFO, LIFO and others depending on the application).
  Write Back is also known as Write Deferred.