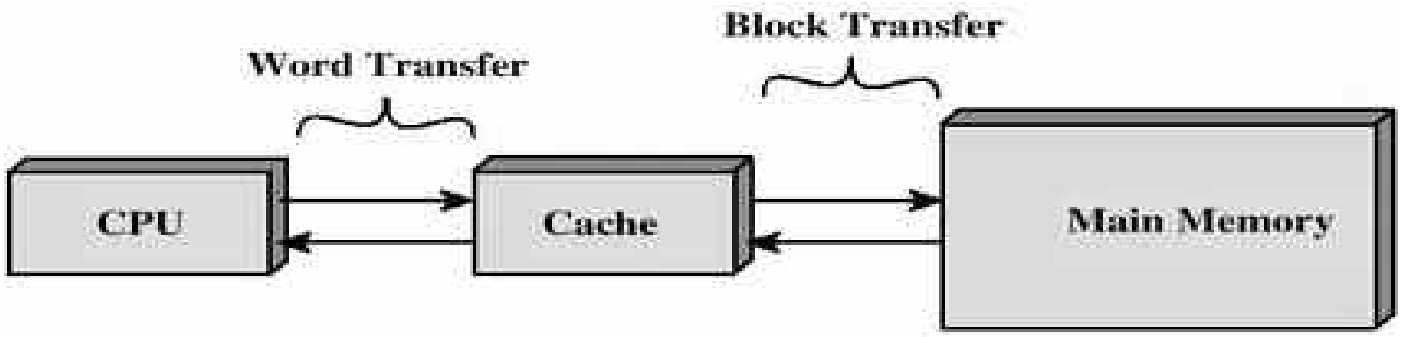
Cache Memory

- The objective of cache memory is to reduce the access time.
- Fundamental Idea :
  - keep the frequently used data and instruction in the cache ,thus reducing the access time.
  - When CPU needs to access memory, the cache is searched first for the data.
  - If the word is found in the cache , it is accessed from the cache.But if the word is not in the cache , then access to main memory is done.
  - And simultaneously word is copied in the cache.

- The transfer of words from the main memory to cache is done in blocks.
- A block consists of many words.
- Since cache has small capacity , exixting item has to be replaced if cache is being full. And the replacement is depends upon the algorithm.



- 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 cache memory

- The cache is the fastest component in the memory hierarchy and approaches the speed of CPU component

Basic Operation on Cache

- When CPU needs to access memory, the cache is examined.
- If the word is found in the cache, it is read from the cache 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.
- If the cache is full, then a block equivalent to the size of the used word is replaced according to the replacement algorithm being used.

Cache Principle

- The cache works on the principle of locality of reference.
- What is locality of reference?
  - In computer science, locality of reference, also known as the principle of locality, is a term for the phenomenon in which the same values, or related storage locations, are frequently accessed.

Cache (Continue)

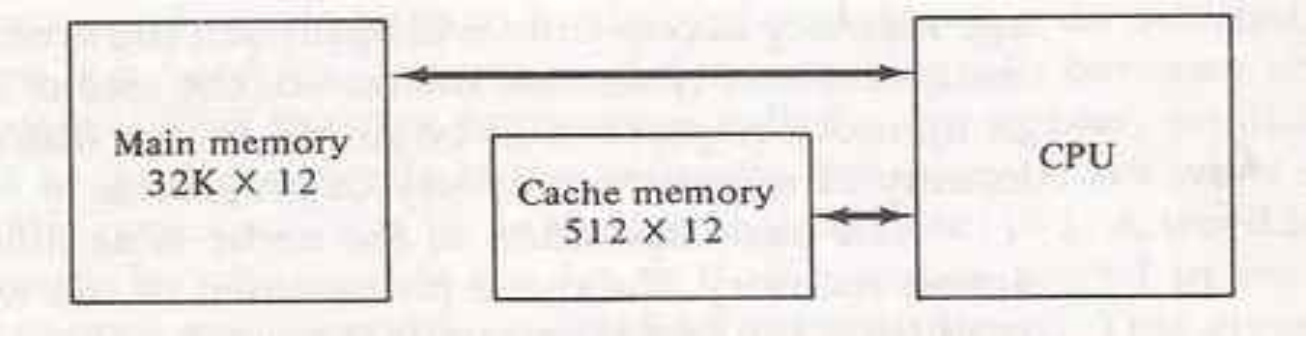
- The main memory is divided into blocks,each consisting of fixed number of locations.
- While a location is accessed,the entire block is brought and stored into the cache memory.
- when the block is stored in the cache memory,it is referred to as line or block frame which is of the same size as the block in the main memory.
- hence any main memory block can be mapped to one of the cache lines.
- The mapping function decides that the block should becopied into which line number

Cache Hit and Cache Miss

- If data is in cache it is termed as "cache hit"
- If data is not present in the cache it is termed as "cache miss".
- The performance of the cache is measured in terms of **hit ratio.**
- Hit ratio is the ratio of thenumber of hits divided by the total number of references to the memory.
  - **Hit ratio = Numbers of hits / Total numbers of memory references**

Mapping

- The transformation of data from the main memory to cache memory is known as mapping.
- There are three types of mapping
  - Direct Mapping
  - Associative Mapping
  - Set-associative Mapping
- To help understand the mapping procedure, we have the following example:



Direct Mapping

- The direct mapped cache is the simplest form of cache and the easiest to check for a hit. Since there is only one possible place that any memory location can be cached, there is nothing to search; the line either contains the memory information we are looking for, or it doesn't.
- Unfortunately, the direct mapped cache also has the worst performance, because again there is only one place that any address can be stored.
- Therfore Direct mapping has disadvantage that the cahce hit ratio reduces sharply if two or more blocks , used alternately , map onto the same block frame in the cache.
- Diagram on page - 209 (fig 6.7)

Associative Mapping

- Any block can go anywhere in cache i.e. Any empty block in cache can be used.
- When request comes for a block , all the entries are compared simultaneously to determine if the requested block is present or not.
- The mapping flexibiltiy permits wide variety of replacement algorithms.
- It encounters longer access time because of associative search, it check all tags to check for a hit.
- It is more flexible and expensive than direct mapping.
- Diagram on page - 210 (fig 6.8)

Set-Associative Mapping

- Combines the simplicity of direct mapping with the flexibility of associative mapping.
- Set associative is an improvement over the direct mapping as here the cache is divided into sets.
- When an address is mapped to a set, the direct mapping scheme is used, and then associative mapping is used within a set.
- When miss occurs in set associative cache and set is full , one of the bolcks has to be replaced according to replacement algorithm .
- Diagram on page - 211 (fig 6.9)

Cache Replacement

- In case of cache miss ,the required block is copied into the cache .
- But if the cache memory is full , then some block from the cache memory needs to be deleted and space should be created for the new block .
- Now which block from the cache memory should be deleted , algorithm are developed

Replacement Algorithms

- Replacement algorithm decide which block frame will be deleted from the cache memory to make space for the new block.
- In case of **direct mapping** , the new block from has to be stored in a specified line or block frame of main memory .
- In **associative and set-associative** mapping , various replacement algorithm are used.

Replacement Algorithm

- Some of the algorithm used are :
  - Random Choice Algorithm:
  - First-in-First-out Algorithm
  - Least Frequently Used Algorithm
  - Least Recently Used Algorithm

Replacement Algorithm

- **Random Choice Algorithm:**

  In this algorithm any block frame from the cache is selected randomly and deleted without any relation to previous page.

- **First-in-First-out Algorithm :**

  This algorithm selects the block frame which has been in the cache memeory for a long time , i.e. the first block which entered the cache is one which has to be deleted.

- **Least Frequently Used Algorithm :**

  This algorithm choosesthe block frame that has been used very frequently by the CPU. The counter value of each block frame gives the details of least frequently used block.

- **Least Recently Used Algorithm :**

  This algorithm chooses the block frame that haas been referenced by the CPU very less number of times from the time it was mapped onto the cache memory.

  The counter value gives the details of how many times the block has been refernced by the CPU.

Cache Write

- When memory write operations are performed, CPU first writes into the cache memory. These modifications made by CPU during a write operations, on the data saved in cache, need to be written back to main memory or to auxiliary memory.
- If the address of the result to be stored is not present in the cache , then the main memory get updated with the result.
- If the address is present in the cache memory,then there are two possibilties: Write-Through & Write-Back
- Therefore there are two popular cache write policies (schemes) are:
  - Write-Through
  - Write-Back

Write-Through

- In a write through cache, the main memory is updated each time the CPU writes into cache.
- The advantage of the write-through cache is that the main memory always contains the same data as the cache contains.
- This characteristic is desirable in a system which uses direct memory access scheme of data transfer. The I/O devices communicating through DMA receive the most recent data.

Write-Back

- In a write back scheme, only the cache memory is updated during a write operation.
- The main memory gets updated only when the corresponding word is to be deleted from the cache memory.
- The updated locations in the cache memory are marked by a flag so that later on, when the word is removed from the cache, it is copied into the main memory.
- The words are removed from the cache time to time to make room for a new block of words.

Need of cache memory:

- Cache memory, also called CPU memory, is high-speed static random access memory (SRAM) that a computer microprocessor can access more quickly than it can access regular random access memory (RAM). This memory is typically integrated directly into the CPU chip or placed on a separate chip that has a separate bus interconnect with the CPU.
- The purpose of cache memory is to store program instructions and data that are used repeatedly in the operation of programs or information that the CPU is likely to need next.
- **Cache memory is** used to reduce the average time to access data from the Main **memory**. The computer processor can access this information quickly from the cache rather than having to get it from computer's main memory. Fast access to these instructions increases the overall speed of the program.

Level of cache memory:

- Cache memory improves the speed of the CPU, but it is expensive.Type of Cache Memory is divided into different level that are L1,L2,L3,L4
- **Level 1 (L1) or Registers:**

  It is a type of memory in which data is stored and accepted that are immediately stored in the CPU. Generally, the L1 cache is the smallest in size and built into the processor chip. In multi-core CPUs, a separate L1 cache is available for each core.

  Examples of L1 cache are accumulator, Program counter and address register,

etc.

- **Level 2 (L2) cache or Secondary Cache**

  There are also L2 caches for larger processors but take longer to access. It is generally part of the CPU, but often a separate chip between the CPU and the RAM.

  L2 is a secondary type of cache memory. L2 cache has more capacity than L1. It is located on a computer microprocessor.

  The processor searches Instructions in the L1 cache, if required data or instructions not found then it searched into L2 cache. The high-speed system bus interconnecting the cache to the microprocessor.

- **Level 3 (L3) or Main Memory**

  The L3 cache is slower than L1 and L2 but larger. In Multi-core processors, each core may have separate L1 and L2, but all cores share a common L3 cache. L3 cache has double speed than the RAM.

  It is a memory on which computer works currently. It is small in size and once power is off data no longer stays in this memory.

- **Level 4 (L4) or Secondary Memory**

  L4 cache is currently uncommon and is external memory which is not as fast as the main memory. Data retains permanently in this memory and is generally on (a form of) DRAM, rather than on static random-access memory (SRAM), on a separate die or chip (exceptionally, the form, eDRAM is used for all levels of cache.

6.6.6 Cache Coherence

In multiprocessor system, all the processors share a common memory. Each processor may have a local memory and part of it or all may be cache. The reason for separate cache is to reduce the average access time in each processor. There is a possibility of same data residing in the caches and main memory. A particular processor can update the data in cache, which makes the data in other caches invalid. This is known as cache coherence or cache consistency problem as shown in Figure 6.8. The multiple copies must be kept identical to ensure the ability of the system to execute memory operations correctly.

Figure 6.10(a) shows a multiprocessor system with three processors, P1, P2 and P3 each having their own local cache. Each processor stores the data Y from the main memory to their local cache. Figure 6.10(b) shows the cache coherence problem in case of write through policy. Here the data Y of processor P2 and P3 is invalid. Figure 6.10(c) shows the cache coherence problem in case of write back policy. Here the data Y of processor P2, P3 and main memory is invalid.

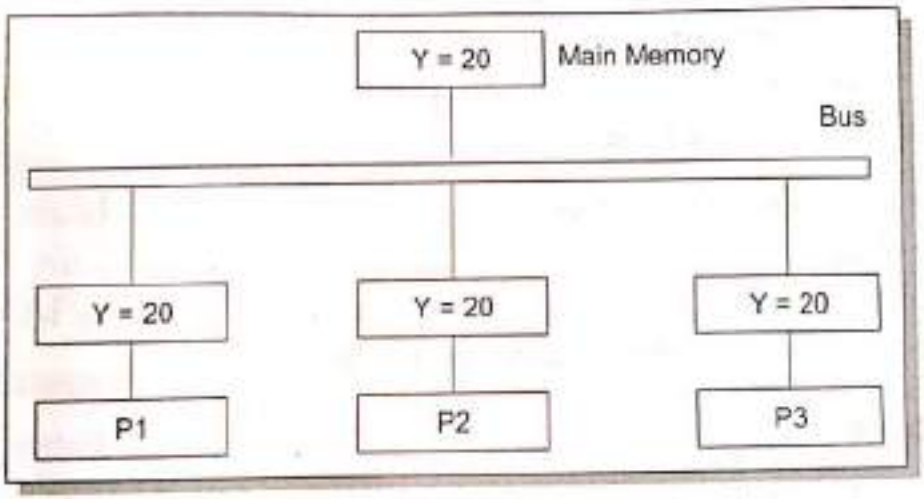


Figure 6.10(a) Caches after loading Y

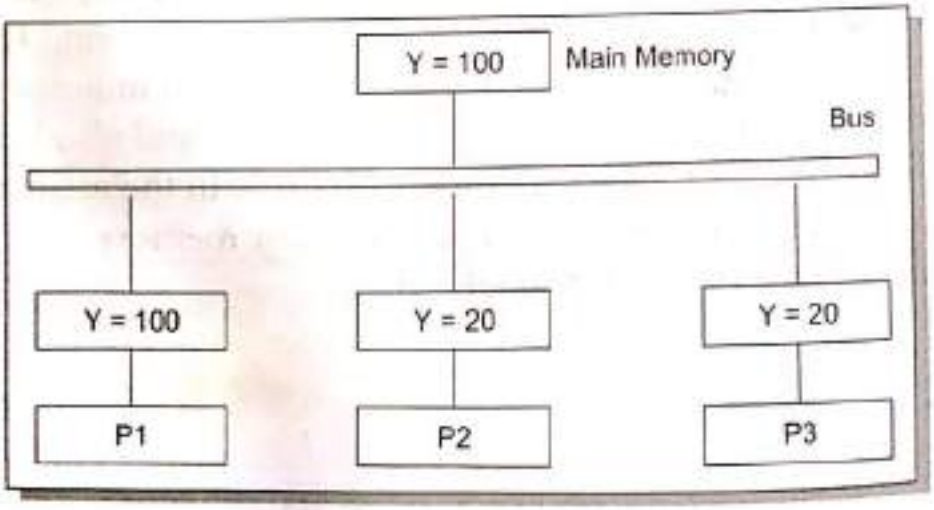


Figure 6.10(b) Cache write through policy

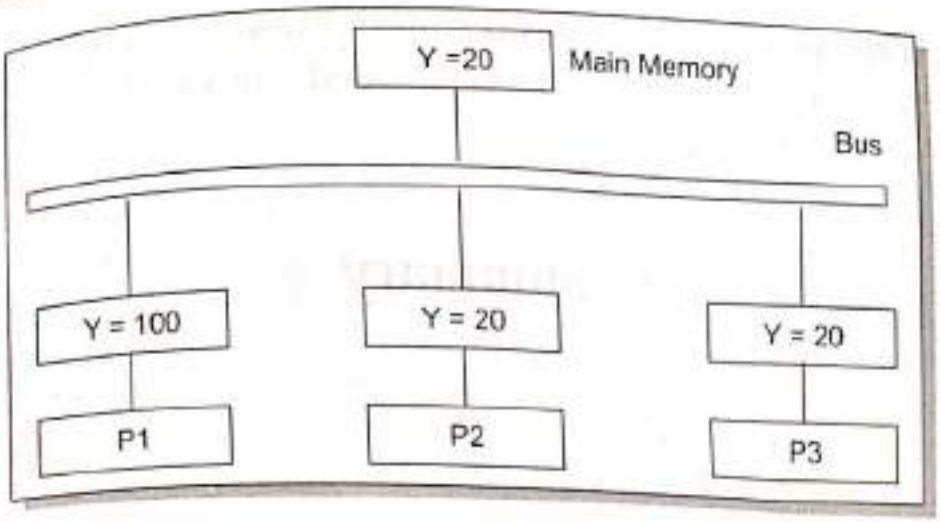


Figure 6.10(c) Cache write back policy

To overcome this problem a simple solution is to have a shared cache memory and disallow the private cache for each processor. Data access is made to the shared cache but it increases the average access time. This scheme solves the problem of cache coherence by avoiding it.

The cache coherence can be solved by two approaches: (a) software and hardware or (b) only hardware.

In software approach, the compiler marks the data that are not valid or noncachable. Only nonshared and read-only data are stored in caches. Such items are cachable. Shared and writeable data are noncachable. The hardware and the operating system prevents noncachable data to be cached. This scheme restricts the data stored in the caches and degrades performance by introducing an extra software overhead.

In hardware approach, all caches attached to the bus, monitor the network constantly for any possible write operation. Depending on the method used, they either update or invalidate the data in their private cache when a match is detected. The hardware unit used for this scheme is known as snoopy cache controller. Snoopy cache controller is designed to keep a watch on the bus for any write operation in the caches. In the snoopy cache controller the simplest method is to adopt write through policy. All the snoopy cache controllers watch the bus for any memory store operation. When a word in a cache is updated, the corresponding location in the main memory is also updated. The snoopy cache controllers check their memory to determine if they have a copy of the word that is updated. If a copy exists, that location is marked

invalid. If the processor accesses the invalid word from the cache, it is equivalent to a cache miss and the updated word is transferred from the main memory.