CSE-323
Computer Architecture

Cache Memory

Lecturer Afia Anjum
Military Institute of Science and technology

Characteristics of Memory Systems

Location:

- It refers to whether memory is internal and external to the computer.
- Based on location, there are 2 types of memory system:
  - Internal Memory: Main memory, Cache memory etc.
  - External Memory: USB, CD, Disk etc.

Capacity:

- Memory is typically expressed in terms of bytes/ words/ blocks

Characteristics of Memory Systems

Unit of Transfer:

- It is the number of bits read out of or written into memory at a time.
- For Internal memory, typically it's equal to a word, but not always.
- For External Memory, data are often transferred in much larger units than a word, and these are referred to as blocks.

Characteristics of Memory Systems

Methods of Accessing Units of Data:

Memory is organized into units of data, called records. To access these records, there are mainly 3 methods:

- Sequential Access: Access must be made in a specific linear sequence and consecutive order. Thus, the time to access an arbitrary record is highly variable. Ex: Tape Unit.
- Random Access: Each addressable location in memory has a unique, physically wired-in addressing mechanism. Thus, any location can be selected at random and directly addressed and accessed. The time to access a given location is independent of the sequence of prior accesses and is constant. Ex: Main Memory and some Caches.
- Direct Access: Also called Semi-random Access. Access is accomplished by direct access to reach a general vicinity plus sequential searching, counting, or waiting to reach the final location. Ex: Magnetic Disk

Characteristics of Memory Systems

Performance Parameters:

There are 3 main parameters to measure performance:

- Execution Time/ Access Time: Discussed in Chapter 2!
- Memory Cycle Time: Access Time + time required before a second access can begin.
- Transfer Rate/ Throughput: Discussed in Chapter 2!

Types of Memory (Based on Manufacturing Process)

- Semiconductor Memory: It uses semiconductor based integrated circuits to store data. Ex: RAM
- Magnetic Surface Memory: It uses different patterns of magnetization on a magnetically coated surface to store information. Ex: Magnetic disk.
- Optical Memory: It uses laser beam to write the information on a disk. It uses laser scanner to scan the information.
- Magneto-optical Memory: It combines magnetic and optical recording techniques. The disk is coated with film that initially is uniformly magnetized. A laser beam is used to demagnetize it to write on it. To read the information, the disk is scanned by polarized light.

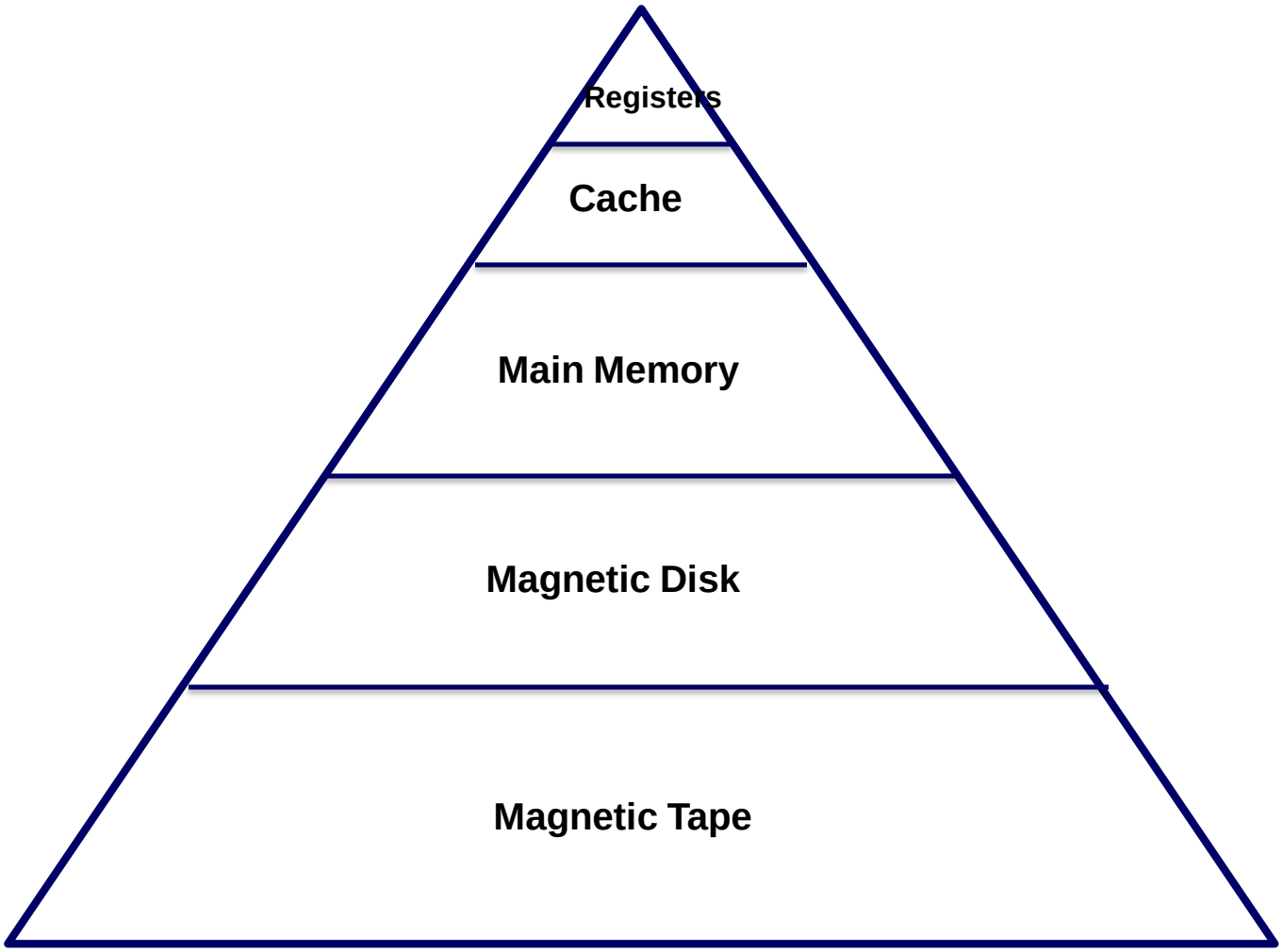
Types of Memory (Based on Data Storage)

- Volatile Memory: Information is lost when electrical power is switched off. Ex: RAM
- Non-volatile Memory: Once written, it never gets lost due to any power off until it is deliberately changed. Ex: ROM
- Erasable Memory: Once written, it can be erased and written again. Ex: Magnetic storage
- Non-erasable Memory: Once written, it can not be erased until the storage is completely destroyed. Ex: ROM

Memory Hierachy

- There is a trade-off among capacity, access time, and cost
  - Faster access time, greater cost per bit
  - Greater capacity, smaller cost per bit
  - Greater capacity, slower access time
- It is obvious that the designer will face a dilemma!!
- The solution is not to use a single memory component, but to employ a ‘Memory Hierarchy’

Memory Hierachy



Locality of Reference

- It is also known as the principle of locality.
- It is a phenomenon describing the same value, or related storage locations, being frequently accessed.
- There are two basic types of reference locality:
  - Temporal Locality: It is based on time. If at one point in time a particular memory location is referenced, then it is likely that the same location will be referenced again in the near future.
  - Spatial Locality: It is based on the location of storage. If a particular memory location is referenced at a particular time, then it is likely that nearby memory locations will be referenced in the near future.

Cache

- Small amount of fast memory.
- Sits between normal main memory and CPU.
- If cache is used, the data are retrieved rapidly from the Cache rather than slowly from the disk.

Cache and Main Memory

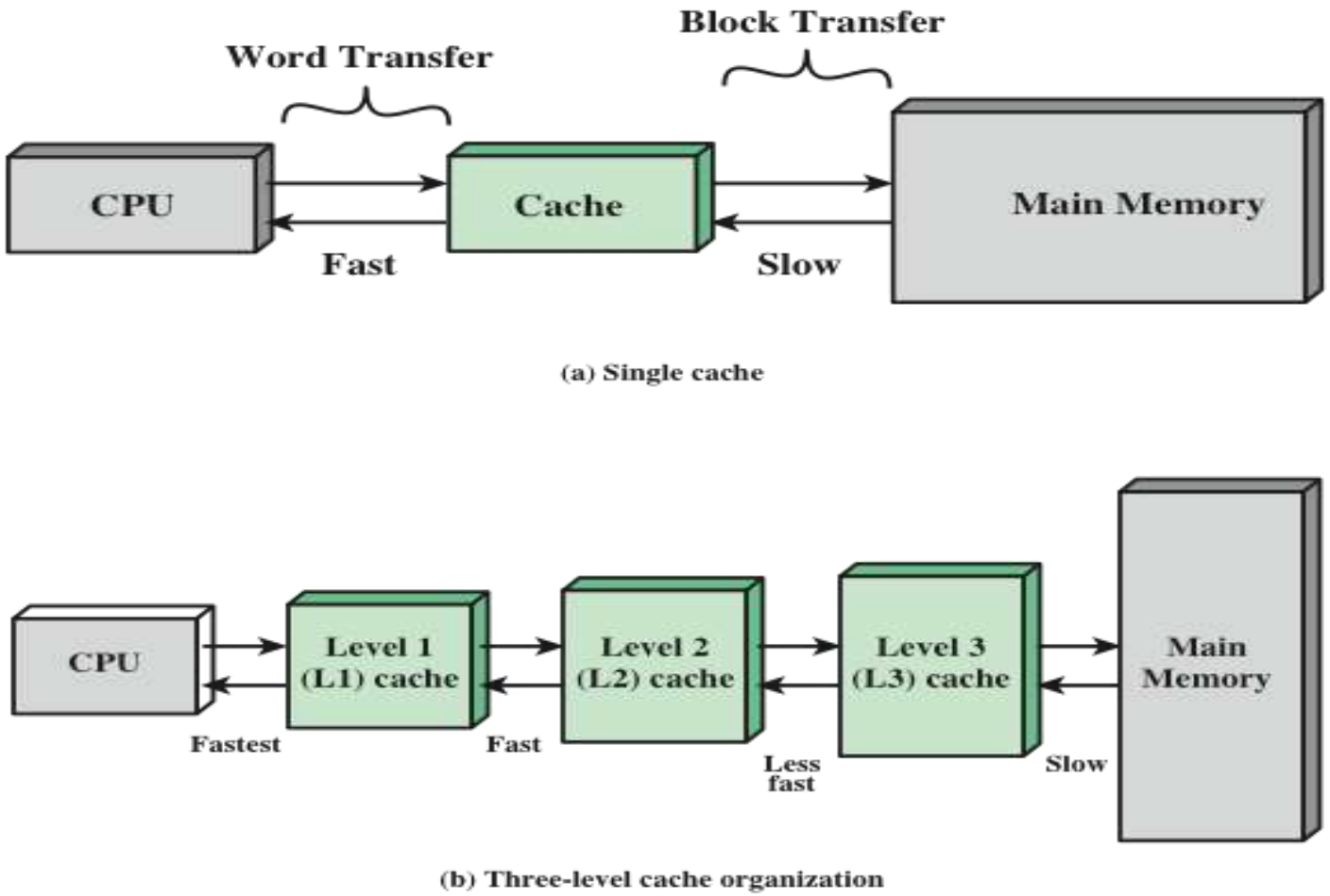


Figure 4.3 Cache and Main Memory

Operation of Two Level Memory

- Let, Cache is the upper-level memory (M1), which is smaller, faster, and more expensive (per bit) than Main Memory, the lower-level memory (M2).
- M1 is used as a temporary store for part of the contents of the larger M2.
- When a memory reference is made, an attempt is made to access the item in M1. If this succeeds, then a quick access is made. If not, then a block of memory locations is copied from M2 to M1 is Temporal Locality is used and the access then takes place via M1. In case of Spatial Locality, blocks of neighbor locations will also be copied to M1.
- Because of locality, once a block and its neighboring blocks are brought into M1, there should be a number of accesses to locations in that block, resulting in fast overall service.

Operation of Two Level Memory

To express the average time to access an item, we must consider not only the speeds of the two levels of memory, but also the probability that a given reference can be found in M1. We have

$$Ts = H \times T1 + (1-H) \times (T1+T2)$$

where

Ts = average (system) access time

T1 = access time of M1 (e.g., cache, disk cache)

T2 = access time of M2 (e.g., main memory, disk)

H = hit ratio (fraction of time reference is found in M1)

Operation of Two Level Memory

Practice problem:

Consider a computer system that has cache memory, main memory (RAM) and an operating system that uses virtual memory. It takes 1 nsec to access a word from the cache, 10 nsec to access a word from the RAM. If the cache hit rate is 95%, what is the average time to access a word?

Answer:

Here,

H = hit ratio = 0.95

T1 = access time of cache = 1 nsec

T2= access time of RAM = 10 nsec

We know, Avg access time = H x T1 +(1-H)x (T1+T2)

= (.95 x 1) + (.05) x (1 + 10) = 1.5 nsec

Cache and Main Memory Structure

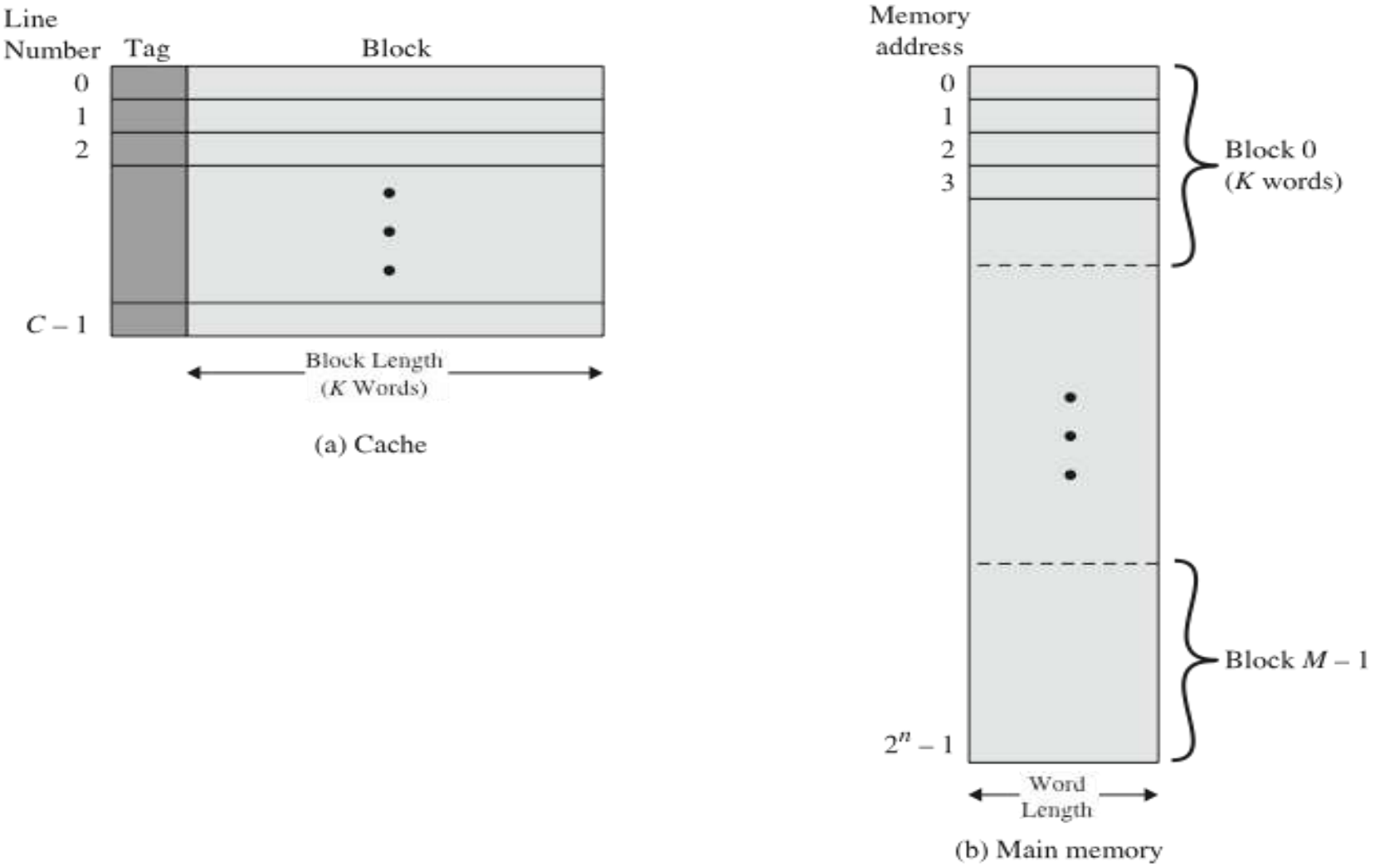


Figure 4.4 Cache/Main-Memory Structure

Cache Read Operation

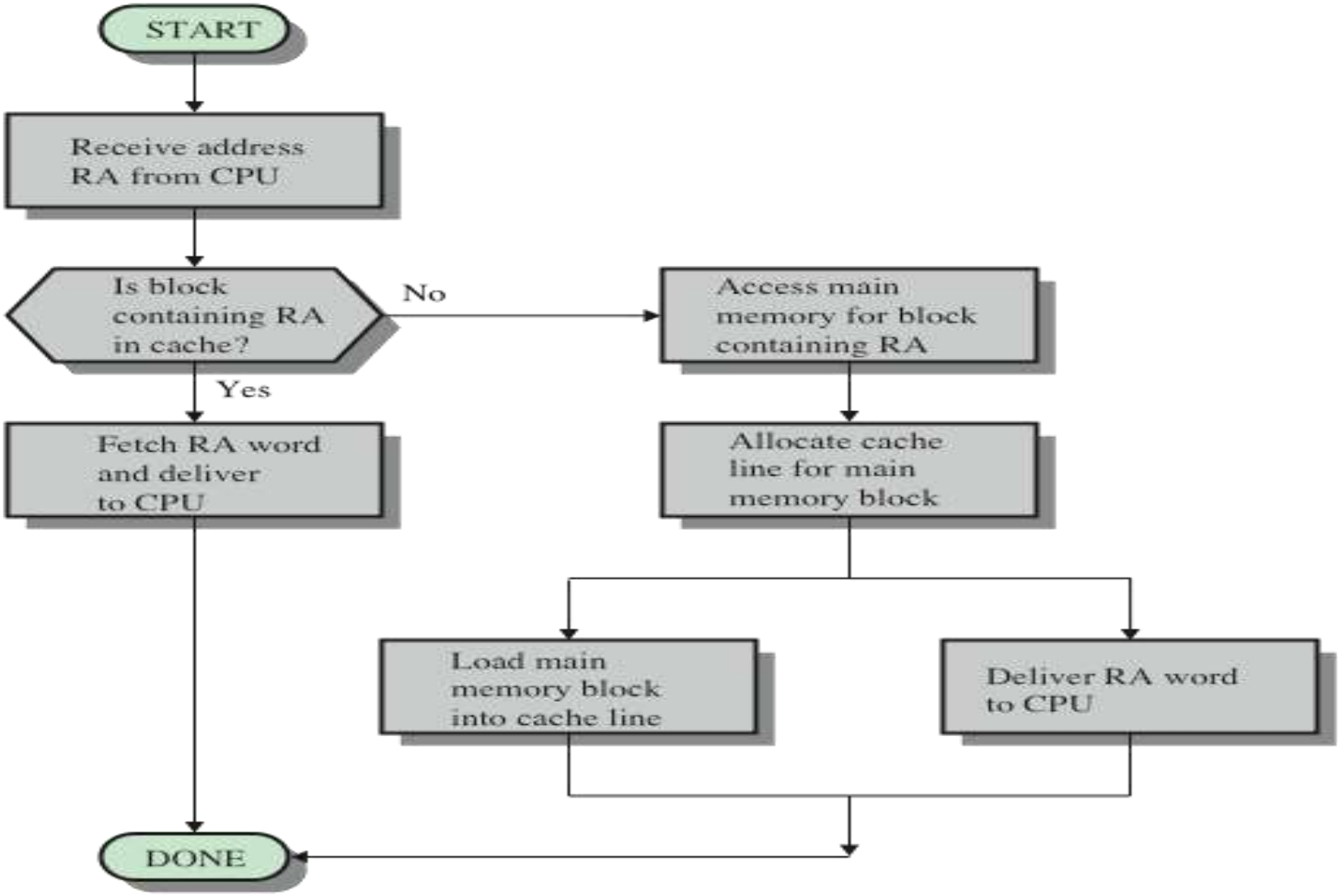


Figure 4.5 Cache Read Operation

Typical Cache Organization 15.4.2

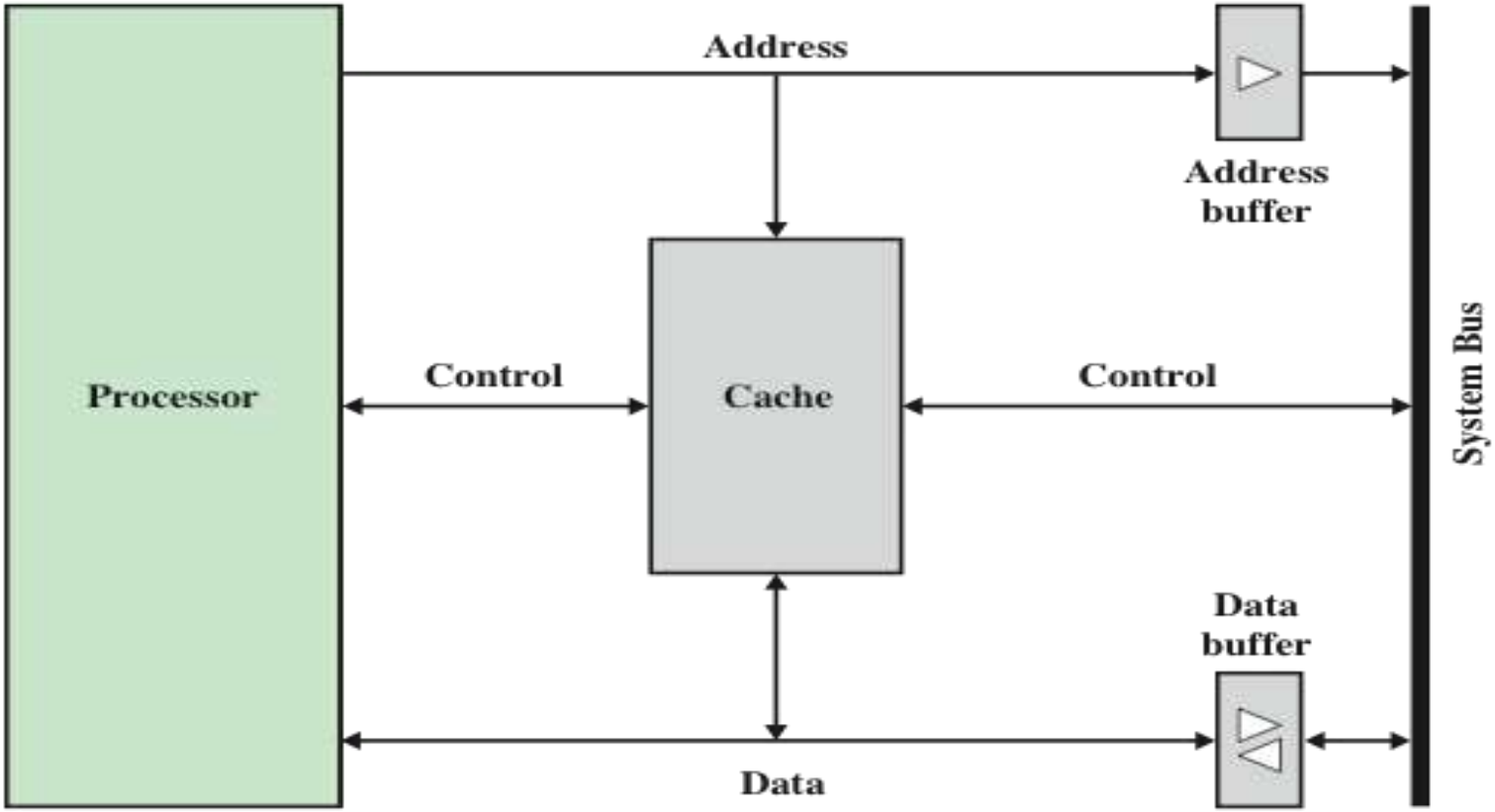


Figure 4.6 Typical Cache Organization

Mapping Function

- Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines.
- Three techniques can be used:
  - Direct Mapping
  - Associative Mapping
  - Set Associative Mapping

Direct Mapping

The simplest technique, known as **direct mapping**, maps each block of main memory into only one possible cache line.

Fully Associated Mapping

K-wayAssociated Mapping

Practice problem

- Assume the size of a main memory in a computer is 1 Mbytes. The block size of the main memory is 16 Bytes. The size of each word is 1Byte. The size of the cache memory is 64KBytes.
- Draw the Main memory format for direct mapping, associative mapping and two way set associative mapping.

Replacement Algorithms

- Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks must be replaced
- For direct mapping there is only one possible line for any particular block and no choice is possible
- For the associative and set-associative techniques a replacement algorithm is needed
- To achieve high speed, an algorithm must be implemented in hardware

The four most common replacement algorithms are:

- Least recently used (LRU)
  - Most effective
  - Replace that block in the set that has been in the cache longest with no reference to it
  - Because of its simplicity of implementation, LRU is the most popular replacement algorithm
- First-in-first-out (FIFO)
  - Replace that block in the set that has been in the cache longest
  - Easily implemented as a round-robin or circular buffer technique
- Least frequently used (LFU)
  - Replace that block in the set that has experienced the fewest references
  - Could be implemented by associating a counter with each line

Write Policy

Write Through

- Simplest technique
- All write operations are made to main memory as well as to the cache
- The main disadvantage of this technique is that it takes a lot of time

Write Back

- Minimizes memory writes
- Updates are made only in the cache
- Portions of main memory are invalid and hence accesses by I/O modules can be allowed only through the cache
- Dirty bits are used to keep track of the updates made in a cache
- This makes complex circuitry

Cache Coherence Problem

- This problem arises during write operation in a multiprocessor system.
- The Cache Coherence Problem is the challenge of keeping multiple local caches synchronized when one of the processors updates its local copy of data which is shared among multiple caches.
- Cache Coherence Problem in Write Through Policy: If a cache of a particular processor updates a data, that data in other caches will become invalid.
- Cache Coherence problem in Write Back Policy: If a cache of a particular processor updates a data, that data in other caches will become invalid. Moreover, before updating in main memory if any other processor tries to access it, it gets the wrong data

Solutions for cache Coherence problem

Write Update

- If a data has been updated in any cache, at the same time propagate that particular update it in all caches (only if the other caches contain that data within themselves).

Write Invalidate

- If a data has been updated in any cache, at the same time propagate a message informing other caches that their data value is invalid (only if the other caches contain that data within themselves).
- Whenever a processor needs a data, it must check whether that data value in its local cache is valid or not.

"Work so hard that one day,
Your signature will be called an autograph"