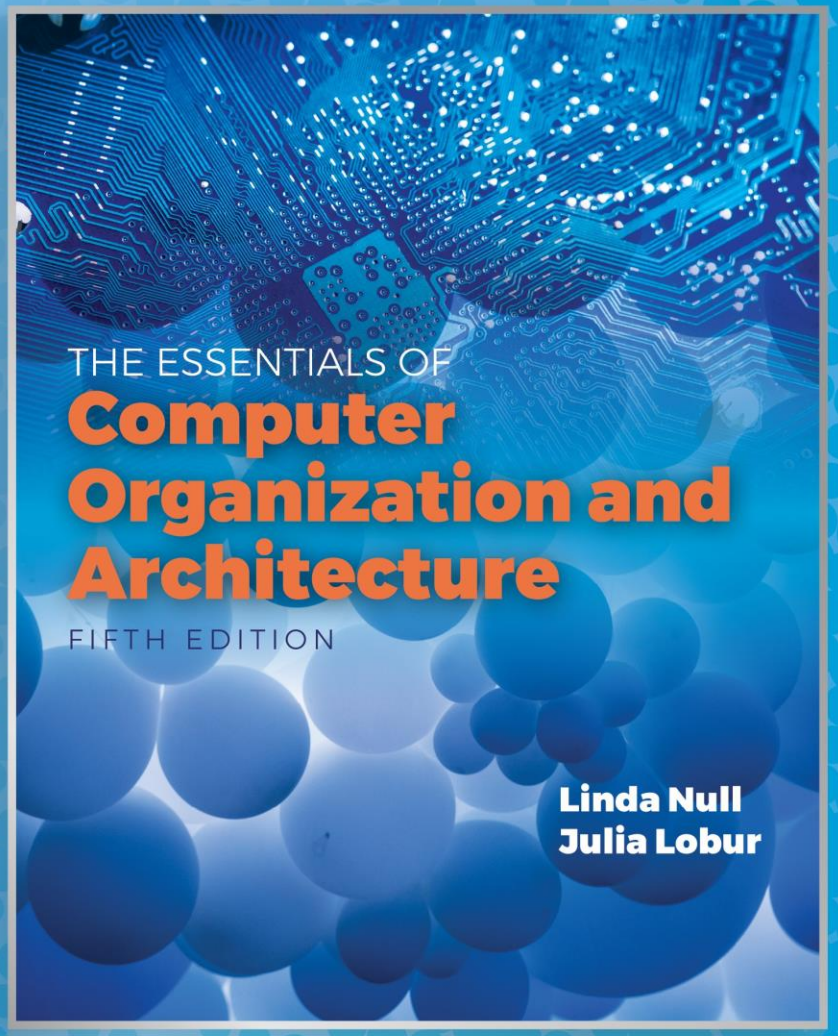
This is the first lecture of Chapter 6

Chapter 6

Memory (A)



Objectives

- Master the concepts of hierarchical memory organization.
- Understand how each level of memory contributes to system performance, and how the performance is measured.
- Master the concepts behind cache memory, virtual memory, memory segmentation, paging, and address translation.

6.1 Introduction

- Memory lies at the heart of the stored-program computer.
- In previous chapters, we studied the components from which memory is built and the ways in which memory is accessed by various ISAs.
- In this chapter, we focus on memory organization. A clear understanding of these ideas is essential for the analysis of system performance.

6.2 Types of Memory (1 of 2)

- There are two kinds of main memory: random access memory (RAM) and read-only-memory (ROM).
- There are two types of RAM: dynamic RAM (DRAM) and static RAM (SRAM).
- DRAM consists of capacitors that slowly leak their charge over time. Thus, they must be refreshed every few milliseconds to prevent data loss.
- DRAM is “cheap” memory owing to its simple design.

6.2 Types of Memory (2 of 2)

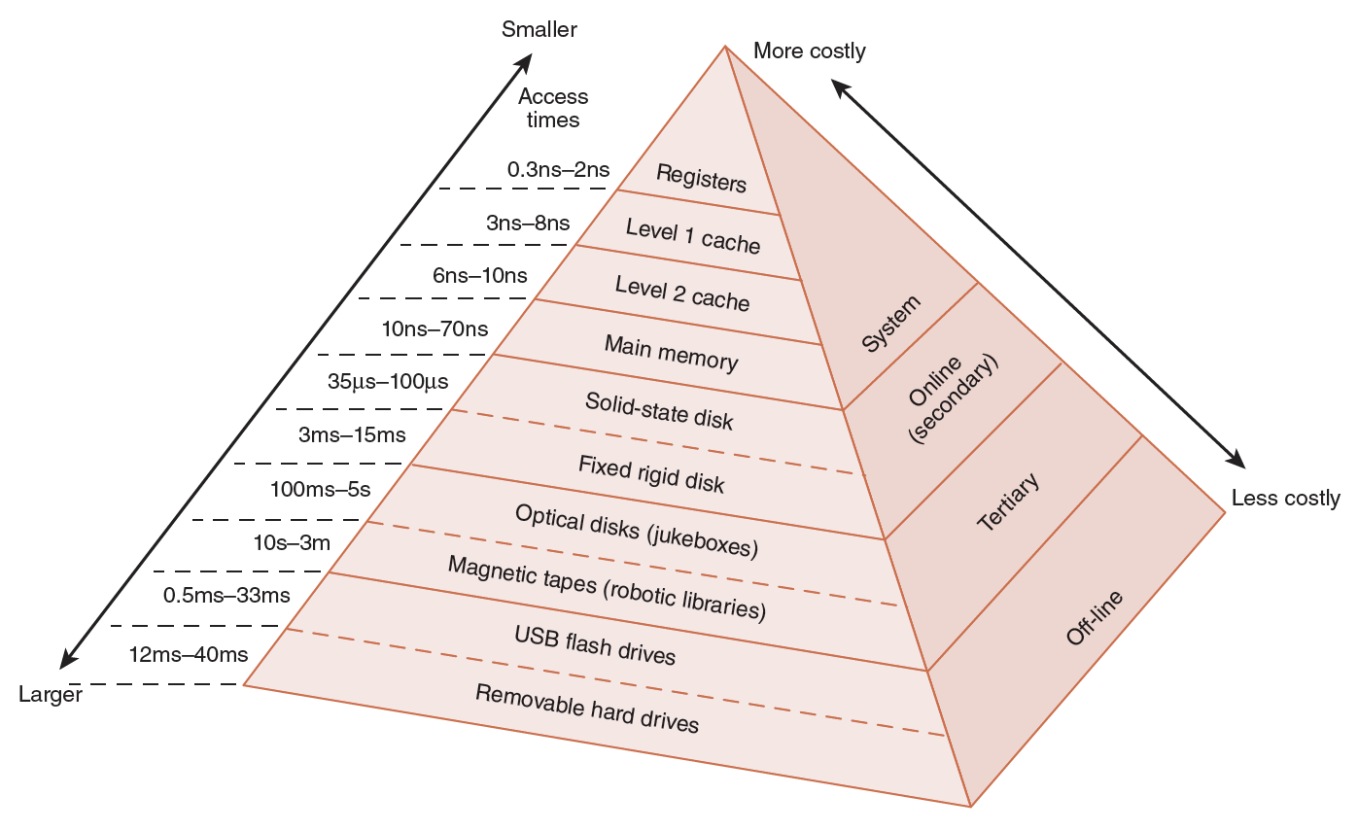
- SRAM consists of circuits similar to the D flip-flop that we studied in Chapter 3.
- SRAM is very fast memory and it doesn't need to be refreshed like DRAM does. It is used to build cache memory, which we will discuss in detail later.
- ROM also does not need to be refreshed, either. In fact, it needs very little charge to retain its memory.
- ROM is used to store permanent, or semi-permanent data that persists even while the system is turned off.

6.3 The Memory Hierarchy (1 of 6)

- Generally speaking, faster memory is more expensive than slower memory.
- To provide the best performance at the lowest cost, memory is organized in a hierarchical fashion.
- Small, fast storage elements are kept in the CPU, larger, slower main memory is accessed through the data bus.
- Larger, (almost) permanent storage in the form of disk and tape drives is still further from the CPU.

6.3 The Memory Hierarchy (2 of 6)

- This storage organization can be thought of as a pyramid:



6.3 The Memory Hierarchy (3 of 6)

- We are most interested in the memory hierarchy that involves registers, cache, main memory, and virtual memory.
- Registers are storage locations available on the processor itself.
- Virtual memory is typically implemented using a hard drive; it extends the address space from RAM to the hard drive.
- Virtual memory provides more space: Cache memory provides speed.

6.3 The Memory Hierarchy (4 of 6)

- To access a particular piece of data, the CPU first sends a request to its nearest memory, usually cache.
- If the data is not in cache, then main memory is queried. If the data is not in main memory, then the request goes to disk.
- Once the data is located, then the data and a number of its nearby data elements are fetched into cache memory.

6.3 The Memory Hierarchy (5 of 6)

- This leads us to some definitions.
  - A *hit* is when data is found at a given memory level.
  - A *miss* is when it is not found.
  - The *hit rate* is the percentage of time data is found at a given memory level.
  - The *miss rate* is the percentage of time it is not.
  - Miss rate = 1 – hit rate.
  - The *hit time* is the time required to access data at a given memory level.
  - The *miss penalty* is the time required to process a miss, including the time that it takes to replace a block of memory plus the time it takes to deliver the data to the processor.

6.3 The Memory Hierarchy (6 of 6)

- An entire block of data is copied after a hit because the *principle of locality* tells us that once a byte is accessed, it's likely that a nearby data element will be needed soon.
- There are three forms of locality:
  - *Temporal locality*: Recently-accessed data elements tend to be accessed again.
  - *Spatial locality*: Accesses tend to cluster.
  - *Sequential locality*: Instructions tend to be accessed sequentially.

6.4 Cache Memory (1 of 45)

- The purpose of cache memory is to speed up accesses by storing recently used data closer to the CPU, instead of storing it in main memory.
- Although cache is much smaller than main memory, its access time is a fraction of that of main memory.
- Unlike main memory, which is accessed by address, cache is typically accessed by content; hence, it is often called *content addressable memory*.
- Because of this, a single large cache memory isn't always desirable—it takes longer to search.

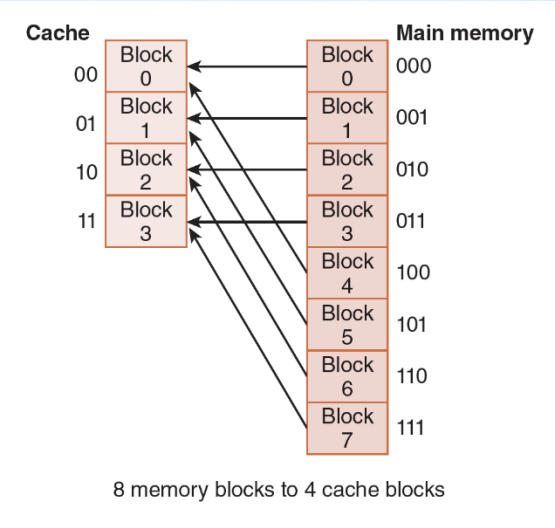
6.4 Cache Memory (2 of 45)

- The simplest cache mapping scheme is direct mapped cache.
- In a direct mapped cache consisting of *N* blocks of cache, block *X* of main memory maps to cache block $Y = X \bmod N$.
- Thus, if we have 10 blocks of cache, block 7 of cache may hold blocks 7, 17, 27, 37, . . . of main memory.

The next slide illustrates this mapping concept.

6.4 Cache Memory (3 of 45)

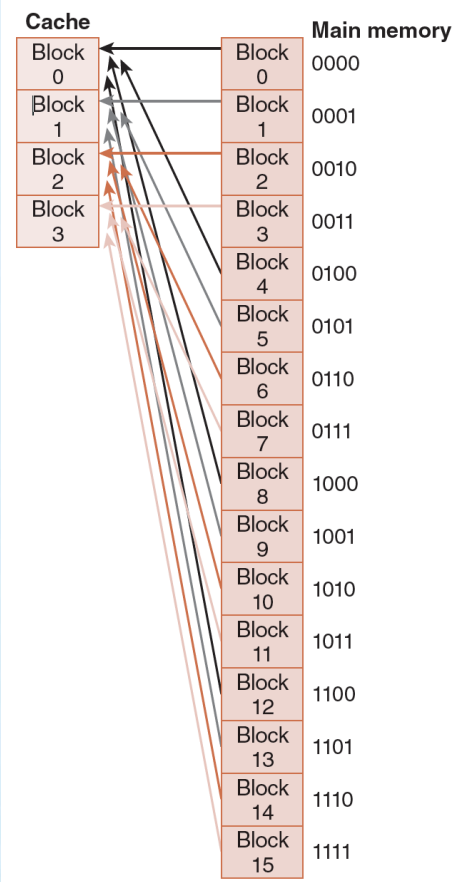
- With direct mapped cache consisting of 4 blocks of cache, block *X* of main memory maps to cache block *Y* = *X* mod 4.



8 memory blocks to 4 cache blocks

6.4 Cache Memory (4 of 45)

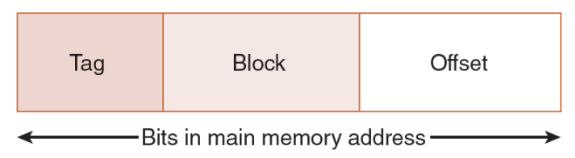
- A larger example.



16 memory blocks to 4 cache blocks

6.4 Cache Memory (5 of 45)

- To perform direct mapping, the binary main memory address is partitioned into the fields shown below.
  - The *offset field* uniquely identifies an address within a specific block.
  - The *block field* selects a unique block of cache.
  - The *tag field* is whatever is left over.



  - The sizes of these fields are determined by characteristics of both memory and cache.

6.4 Cache Memory (6 of 45)

- Example 6.1: Consider a byte-addressable main memory consisting of 4 blocks, and a cache with 2 blocks, where each block is 4 bytes.
- This means Block 0 and 2 of main memory map to Block 0 of cache, and Blocks 1 and 3 of main memory map to Block 1 of cache.
- Using the tag, block, and offset fields, we can see how main memory maps to cache as follows.