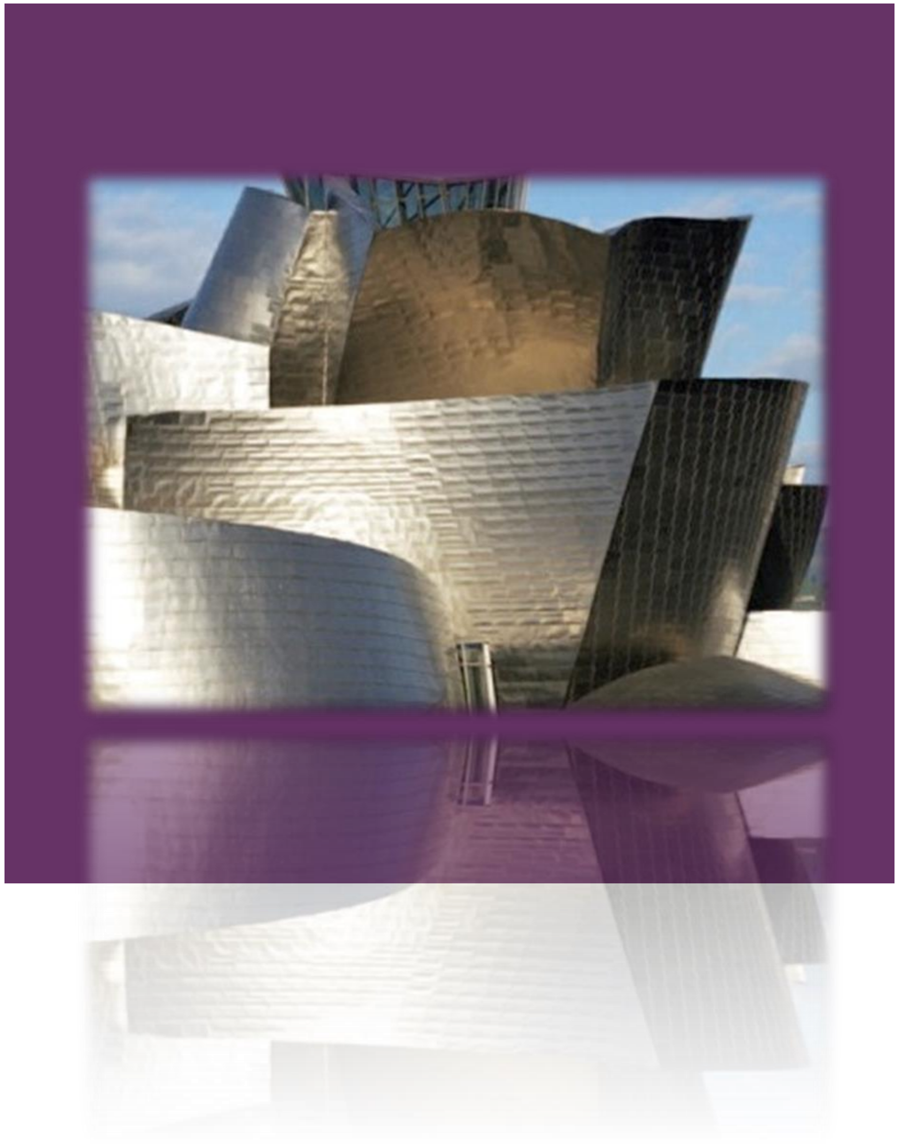


SECR2033 Computer Organization and Architecture

Lecture slides prepared by “Computer Organization and Architecture”, 9/e, by William Stallings, 2013.

Module 6
Memory

Objectives:

- To master the concepts of hierarchical memory organization.
- To understand how each level of memory contributes to system performance, and how the performance is measured.
- To master the concepts behind cache memory and understands the basic concept of virtual memory.

Module 6

Memory

Module 6
Memory

- ❑ Overview
- ❑ Type of Memory
- ❑ The Memory Hierarchy

Overview

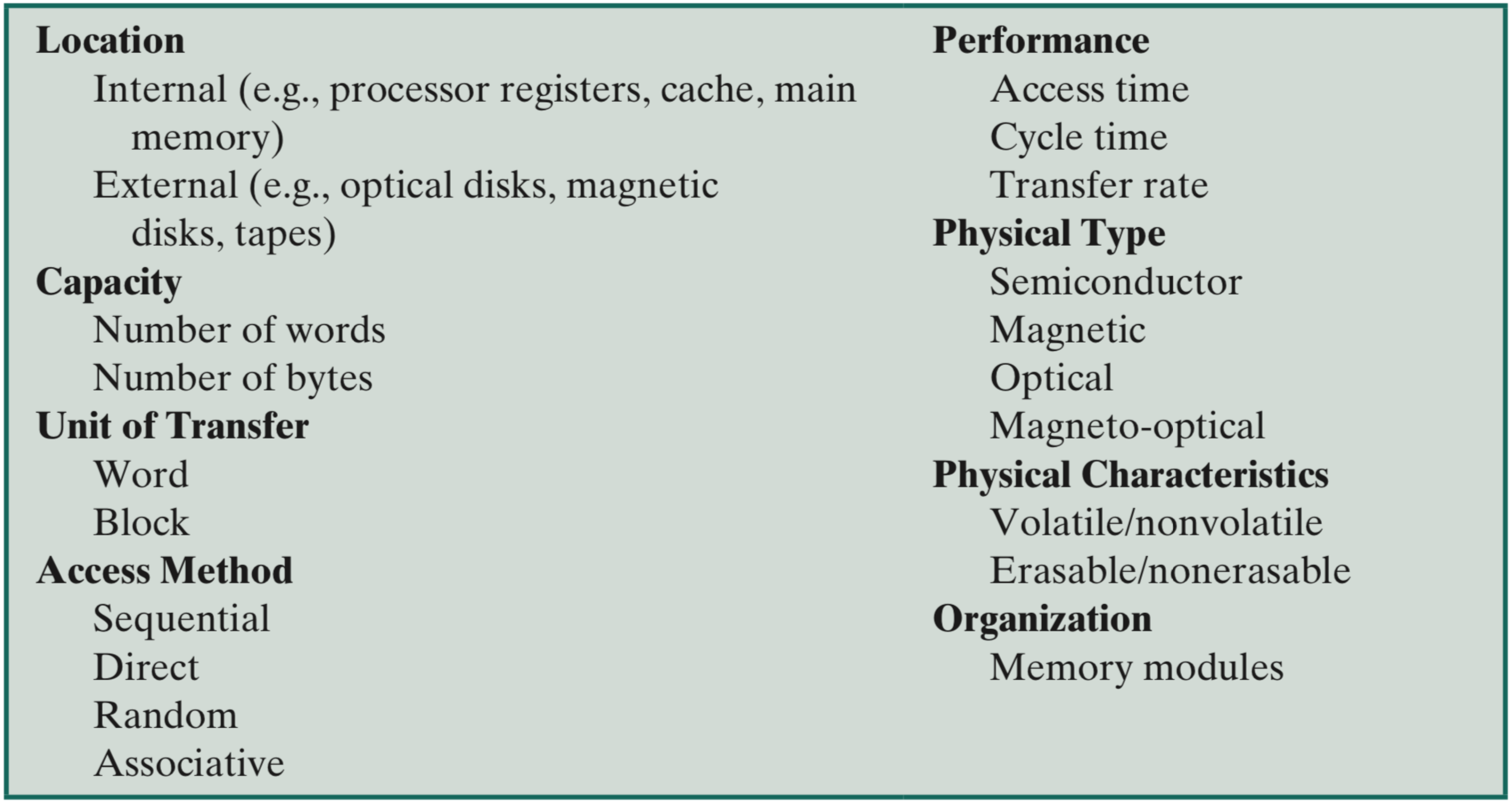
- Most computers are built using the *Von Neumann* model, which is centered on memory.
- The *programs* that perform the processing are stored in memory.
- The memory unit that communicates directly with the CPU is called *main memory**.
- Devices that provide backup storage are called *auxiliary memory*.*

Only programs and data currently needed by the processor reside in main memory.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.233.
* Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.445.

- Internal memory is often equated with main memory, but there are other forms of internal memory:
  - The processor requires its own local memory → registers.
  - The control unit portion of the processor may also require its own internal memory → control memory.
  - Cache is another form of internal memory.
- The complex subject of computer memory is made more manageable if we classify memory systems according to their key characteristics (see next table).

Table: Key Characteristics of Computer Memory Systems.



Location
- Internal (e.g., processor registers, cache, main memory)
- External (e.g., optical disks, magnetic disks, tapes)

Capacity
- Number of words
- Number of bytes

Unit of Transfer
- Word
- Block

Access Method
- Sequential
- Direct
- Random
- Associative

Performance
- Access time
- Cycle time
- Transfer rate

Physical Type
- Semiconductor
- Magnetic
- Optical
- Magneto-optical

Physical Characteristics
- Volatile/nonvolatile
- Erasable/nonerasable

Organization
- Memory modules

Type of Memory

Why are there so many different types of computer memory?

- The answer → new technologies continue to be introduced in an attempt to match the improvements in CPU design.
- Even though a large number of memory technologies exist, there are only two basic types of memory:

RAM (Random Access Memory)

ROM (Read-Only Memory)

(a) RAM *(Random Access Memory)*

- RAM is also the "main memory" or primary memory.
- Used to <u>store programs</u> and <u>data</u> that the computer needs when executing programs.

However, RAM is volatile, and loses this information once the power is turned off.

- Two general types of chips used to build the bulk of RAM memory in today's computers:
  - ❑ SRAM (*Static Random Access Memory*).
  - ❑ DRAM (*Dynamic Random Access Memory*).

*The **organization** of RAM is a key design issue → refers to the physical arrangement of bits to form words*[*].

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.234
* William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.124

(b) ROM *(Read-Only Memory)*

- Stores critical information necessary to operate the system, such as the program necessary to boot the computer.

  - ROM is not volatile and always retains its data.
  - Maintain information when the power is shut off.

- Some different types of ROM:
  - ❑ PROM *(Programmable ROM)*
  - ❑ EPROM *(Erasable PROM)*
  - ❑ EEPROM *(Electrically Erasable PROM)*
  - ❑ Flash memory → essentially EEPROM with the added benefit .

The Memory Hierarchy

- 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.

Closest to the processor.

Memories closer to the top tend to be smaller in size, better performance.

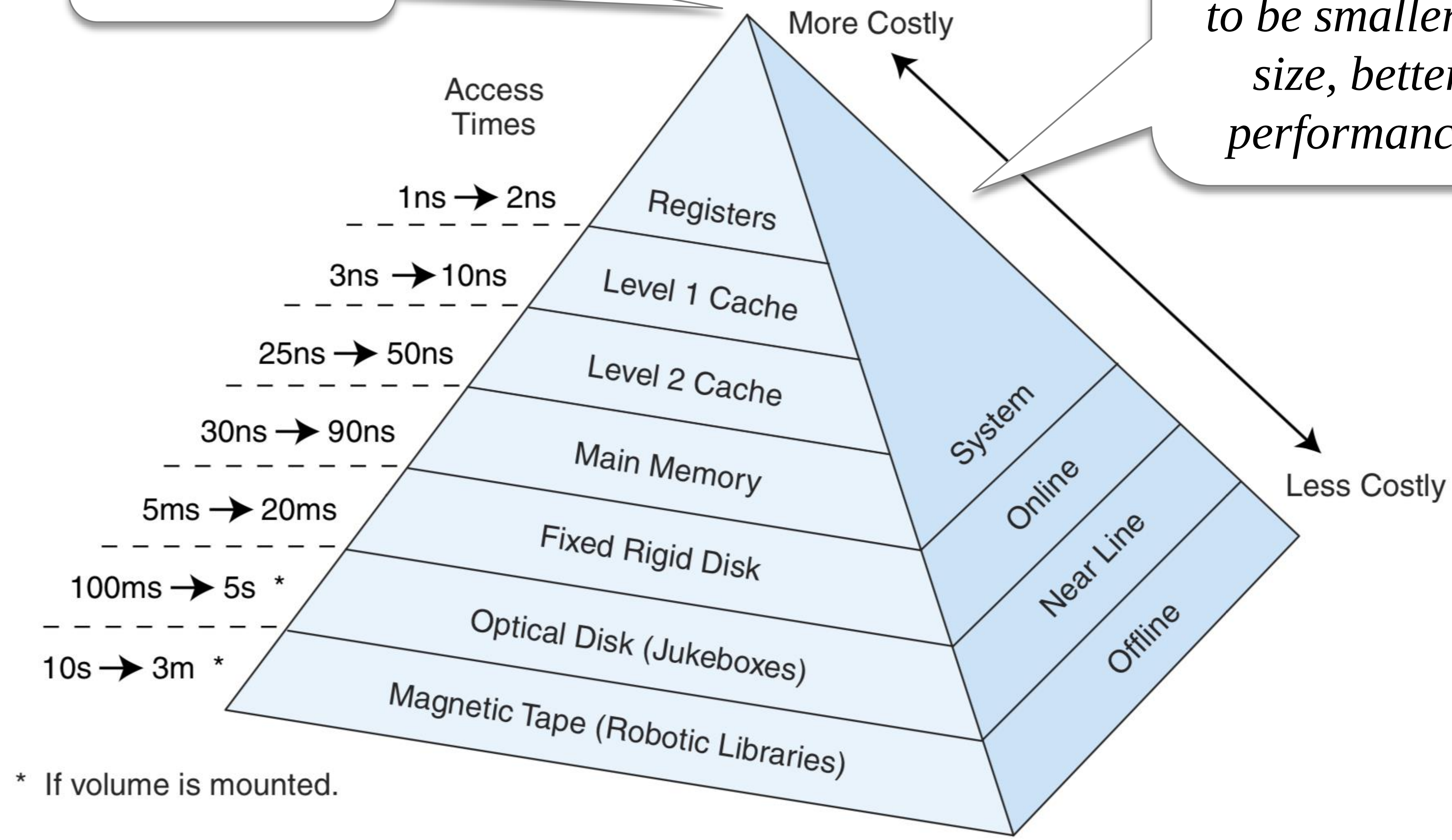
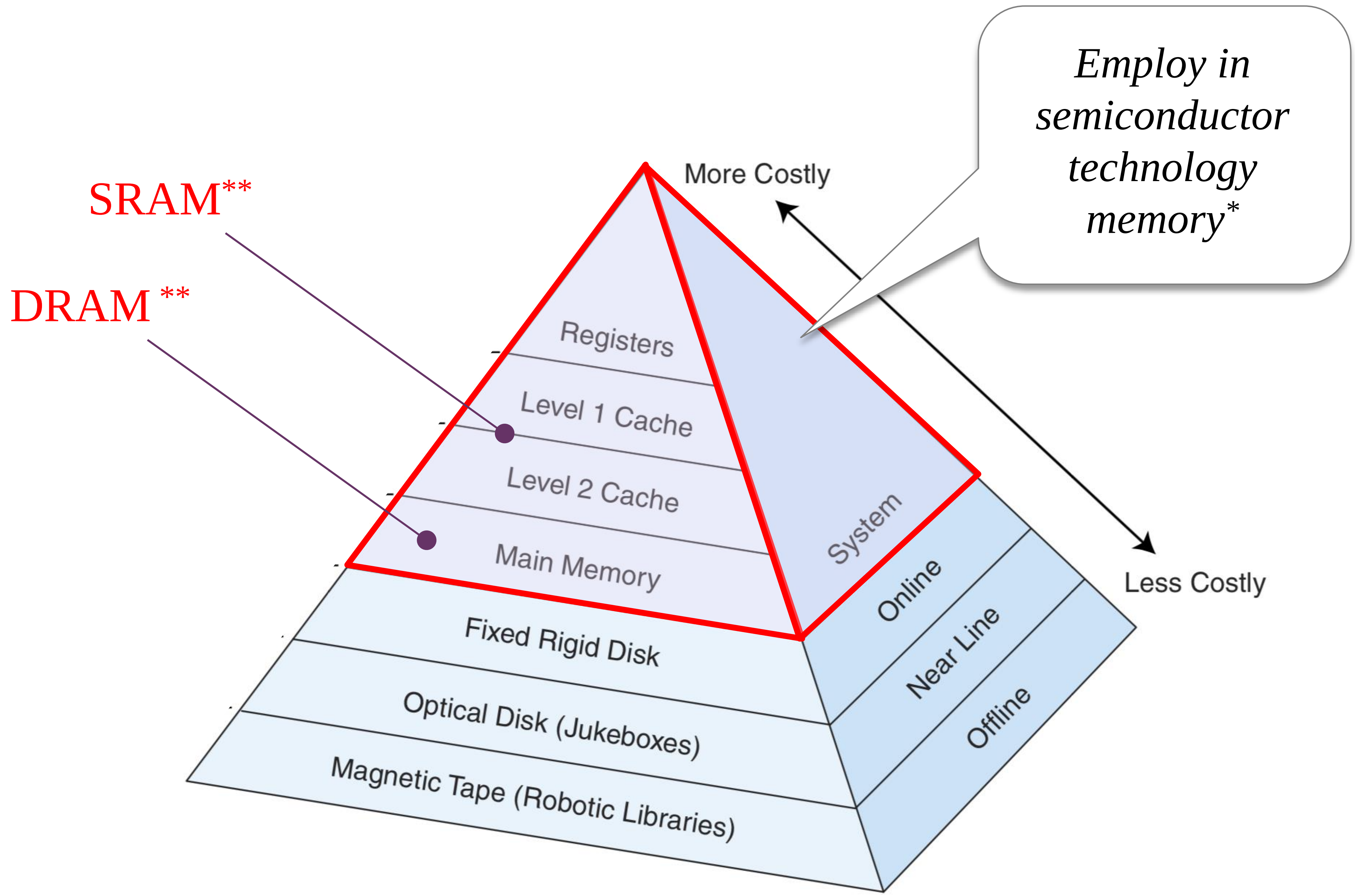
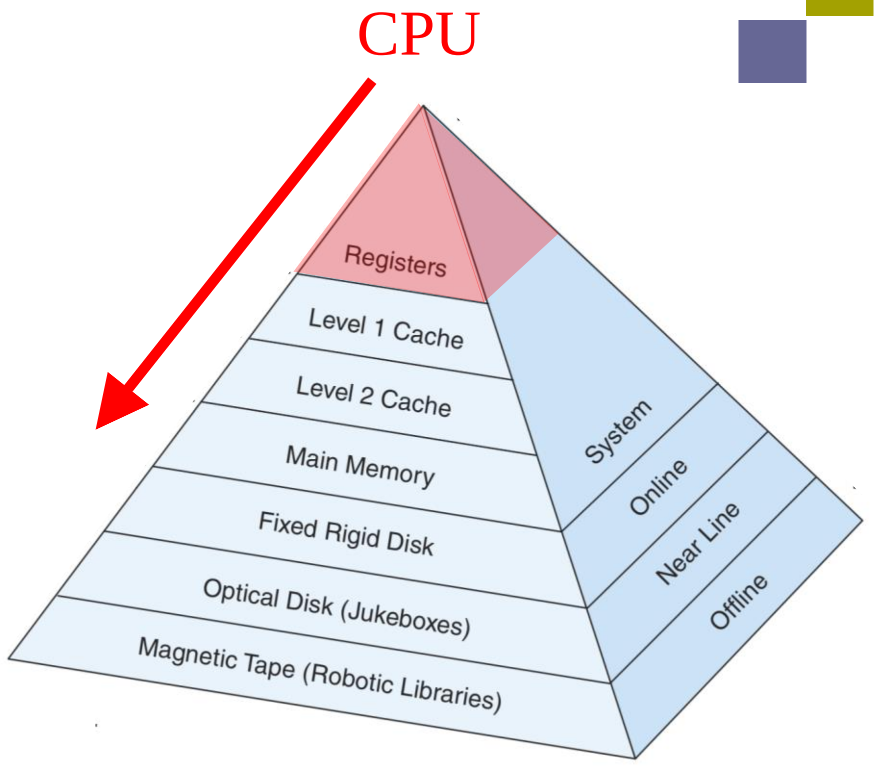


Figure: The memory hierarchy.



Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.236
* William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.127
** Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.378

- 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.

Table: The following terminology is used when referring to the memory hierarchy.

Terminology	Definition
Hit	The data is <u>found</u> at a given memory level.
Miss	The data is <u>not found</u> at a given memory level.
Hit Rate	The percentage (%) of memory accesses <u>found</u> in a given level of memory.
Miss Rate	The percentage (%) of memory accesses <u>not found</u> in a given level of memory → (1 – *hit rate*)
Hit Time	The time required to access data at a given memory level.
Miss Penalty	The time required to process a *miss*, including the <u>time that it takes to replace</u> a block of memory plus the <u>time it takes to deliver</u> the data to the processor.

Locality of Reference

- An entire blocks of data is copied after a *hit* because the principle of *locality* tells us that once a byte is accessed, it is likely that a nearby data element will be needed soon.
- There are three basic forms of *locality*:

1) *Temporal locality*—Recently accessed items tend to be accessed again in the near future.
2) *Spatial locality*—Accesses tend to be clustered in the address space.
3) *Sequential locality*—Instructions tend to be accessed sequentially.

Review Questions

6.1.1. What are the advantages of using DRAM for main memory?

6.1.2 Explain the concept of a memory hierarchy.

Module 6
Memory

- ❑ Overview
- ❑ Memory Organization
- ❑ Memory Capacity
- ❑ Memory Interleaving

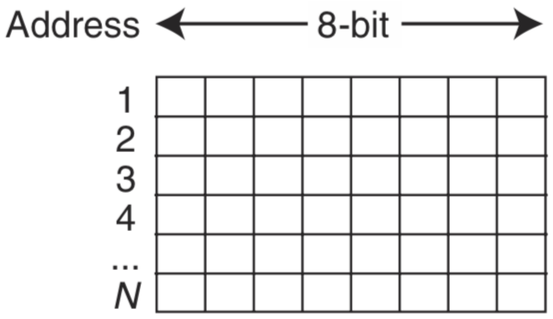
Overview

- The main memory is the central storage unit in a computer system.
- The principle technology used for the main memory is based on semiconductor integrated circuits (RAM) either *static* or *dynamic* operating modes.

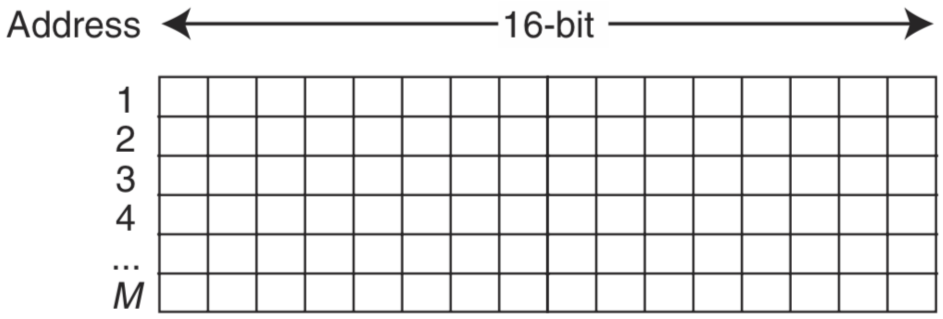
- Most of the main memory in a general-purpose computer is made up of RAM integrated circuit, but a portion of the memory may be constructed with ROM.

Memory Organization

- We can envision memory as a matrix of bits.
- Each row, implemented by a register, has a length typically equivalent to the *word size* of the machine.
- Each register (more commonly referred to as a *memory location*) has a unique address; memory addresses usually start at zero (0) and progress upward.



N 8-Bit Memory Locations



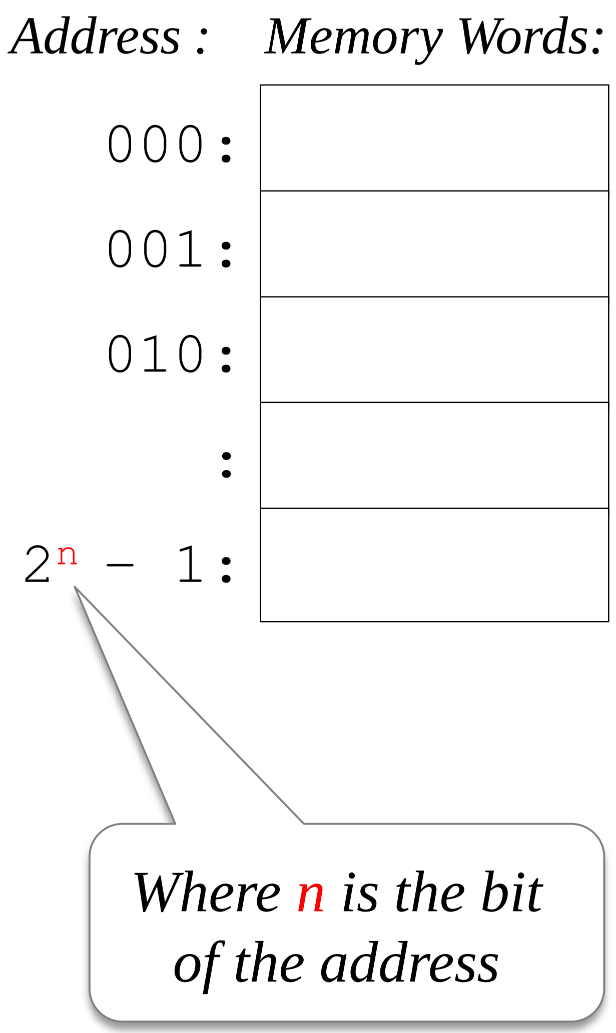
M 16-Bit Memory Locations

Memory Location

- Each part of memory has a separate memory location, which can be referred to using a memory address.
- Number of bits for an address uniquely access to a memory location.

$$No\ of\ bits = \frac{\log(memory\ capacity)}{\log 2}$$

$$No\ of\ locations = 2^{(No.of\ bit\ in\ the\ address)}$$

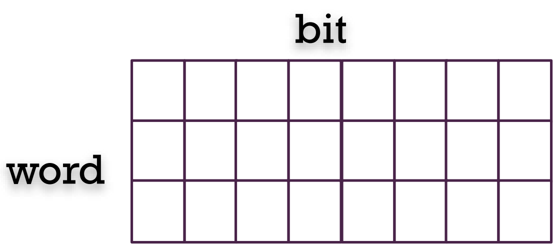


Memory Capacity

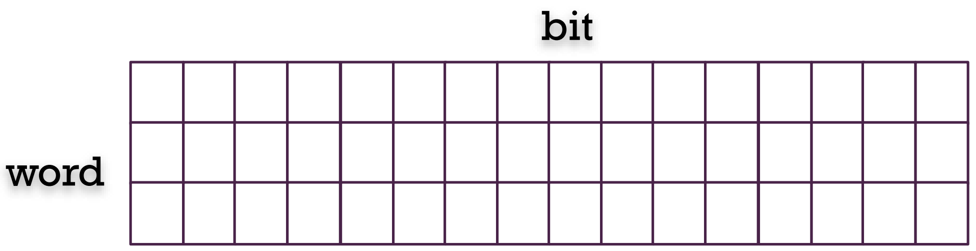
- Memory capacity usually measured in bits:

Total no. of memory locations (words) × size of memory word

- **Examples**: 3 words (locations/rows)



(a)



(b)

Size = 3 words x 8 bits
= 24 bits

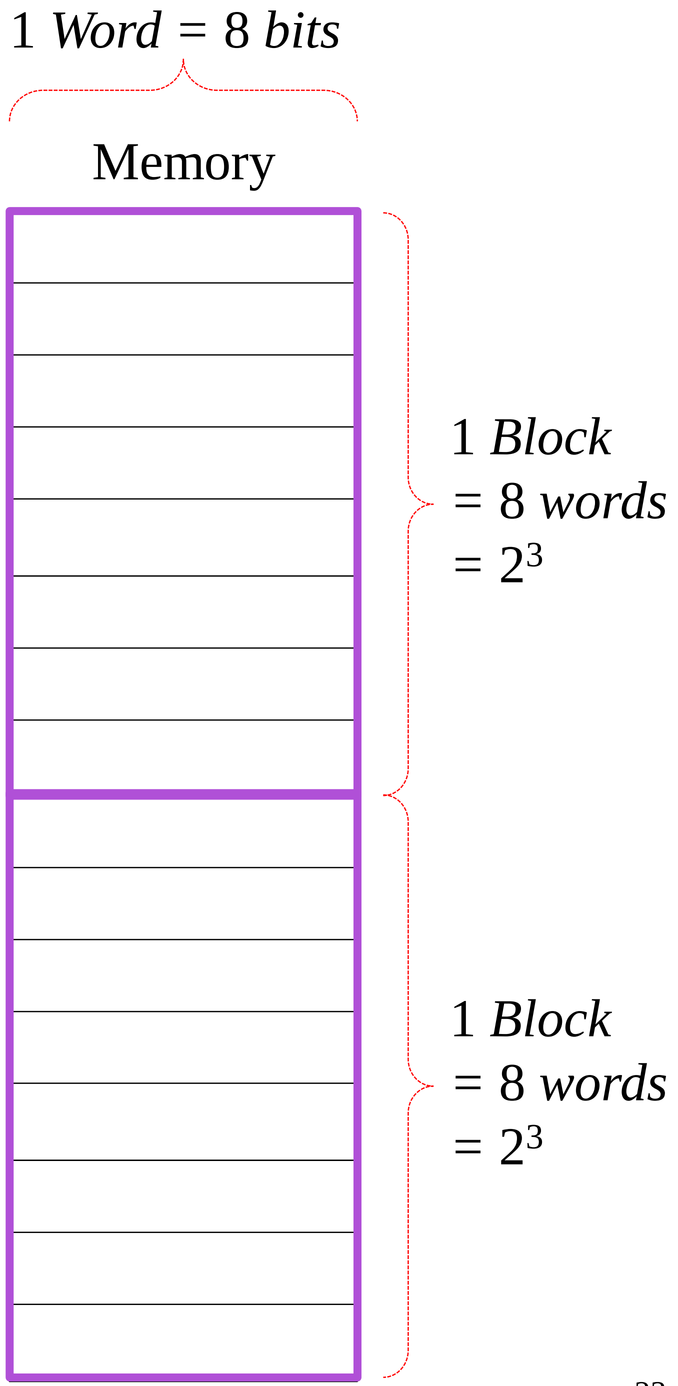
Size = 3 words x 16 bits
= 48 bits

Example 1:

Main memory is divided into blocks. The memory word is 8 bit and the size of a block is 8 words.

(a) What is the capacity of the main memory, if the total number of blocks in the memory is 128?

(b) How many blocks in the main memory if the memory capacity is 32 Kbit?



1Kbits = 1024bits = 2^{10}

Solution :

$$(a)\ Memory\ Capacity = Blocks \times Words \times Bits$$
$$= 128 \times 8 \times 8$$
$$= 2^{7} \times 2^{3} \times 8$$
$$= 2^{10} \times 8$$
$$= 1Kbit \times 8 = 8Kbits$$

$$(b)\ Memory\ Capacity = Blocks \times Words \times Bits$$
$$32Kbit = Blocks \times 8 \times 8$$
$$Blocks = \frac{2^{5} \times 2^{10}}{2^{3} \times 2^{3}}$$
$$= 2^{15-6} = 2^{9}$$
$$Blocks = 512$$

Memory Interleaving

- A single shared memory module causes sequentialization of access.
- *Memory interleaving* → splits memory across multiple memory modules (or memory banks).

A number of memory chips can be grouped together to form a memory bank

Low-Order Interleaving (LOI)	*High-Order Interleaving (HOI)*
The low-order bits of the address are used to select the bank. *e.g.* → 00000101	The high-order bits of the address are used to select the bank. *e.g.* → 10100000

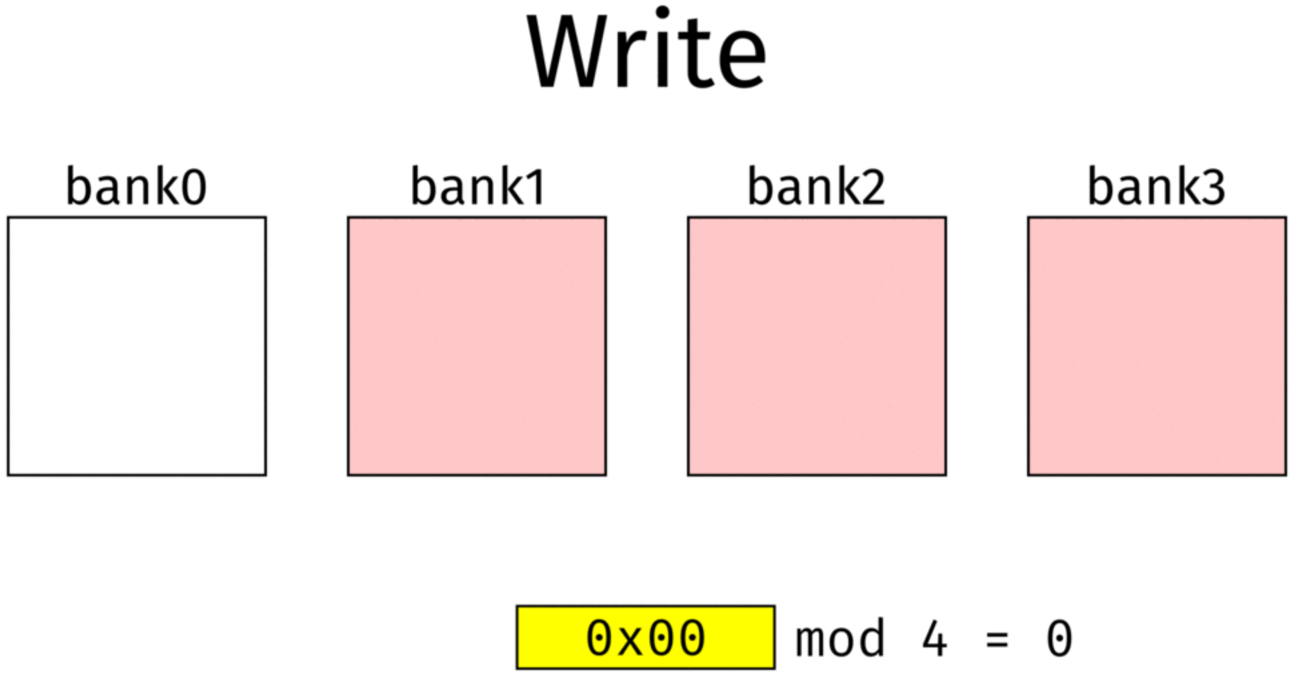


Figure: Memory interleaving (LOI) example with 4 banks. Red banks are refreshing and can't be used.

Memory modules = Memory banks

Example 2: Memory interleaving on 32 addresses.

Module 1	Module 2	Module 3	Module 4	Module 5	Module 6	Module 7	Module 8
0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

Figure: High-Order Interleaving (HOI).

Module 1	Module 2	Module 3	Module 4	Module 5	Module 6	Module 7	Module 8
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31

Figure: Low-Order Interleaving (LOI).

Address format:

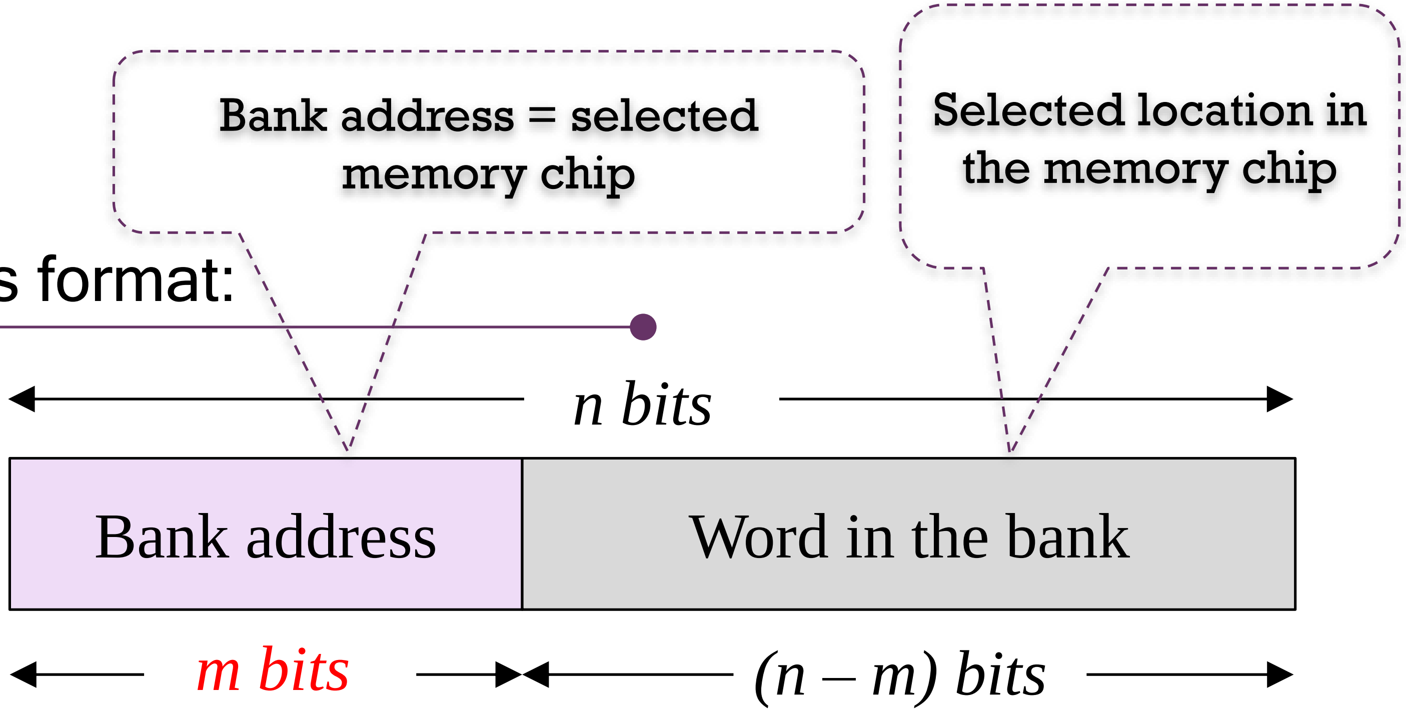

Bank address = selected memory chip
Selected location in the memory chip
n bits
Bank address
Word in the bank
m bits
(n – m) bits

Figure: High-Order Interleaving (HOI).

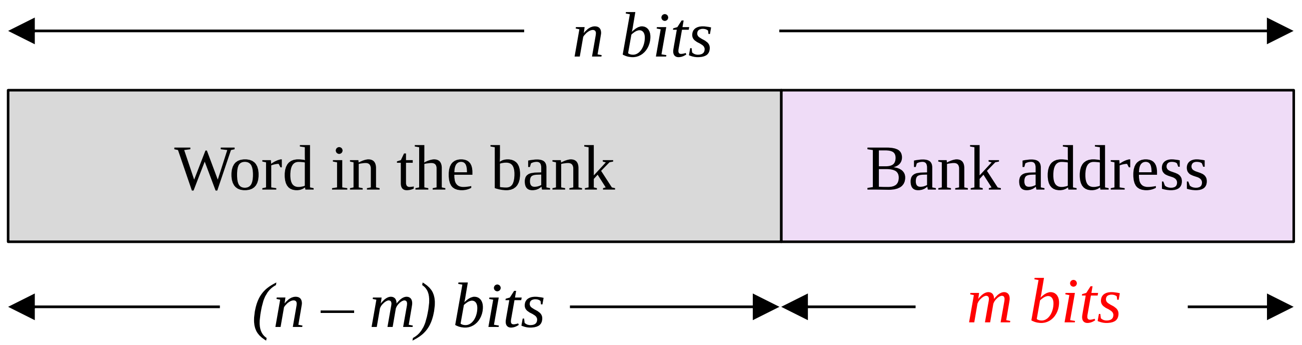


Figure: Low-Order Interleaving (LOI).

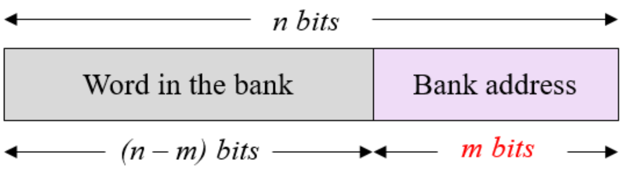
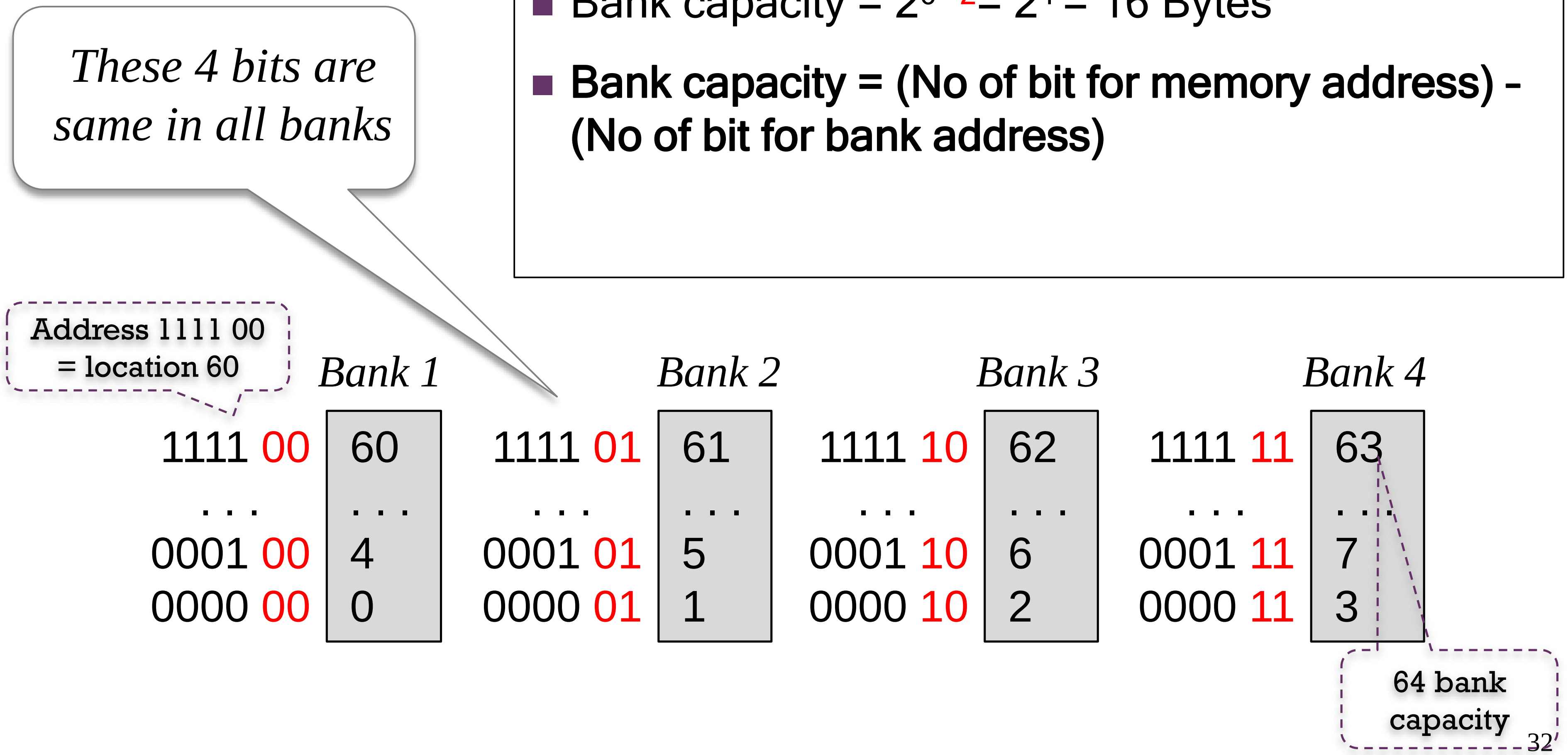


Figure: Low-Order Interleaving (LOI).

Example 3: LOI

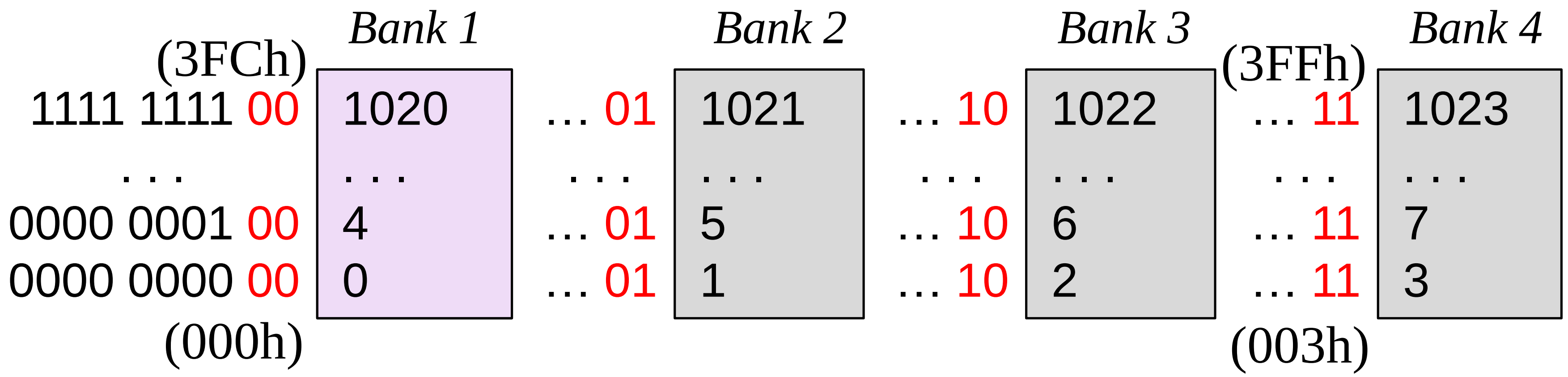
- Memory capacity is 64 Bytes
  → $2^6 = 64$ (Number of bit for memory address = 6)
- Total bank is 4
  → $2^2 = 4$ (Number of bit for bank address = 2)
- Bank capacity = $2^{6-2} = 2^4 = 16$ Bytes
- **Bank capacity = (No of bit for memory address) - (No of bit for bank address)**



Activity 1

Example 4: LOI

Given a memory address as 29Ch (10 bits) and there are 4 memory banks. Determine the memory bank address and the address of the word in the bank using LOI.



LOI: Advantages and Disadvantages

It produces memory interference.

A failure of any single bank would be catastrophic to the whole system.

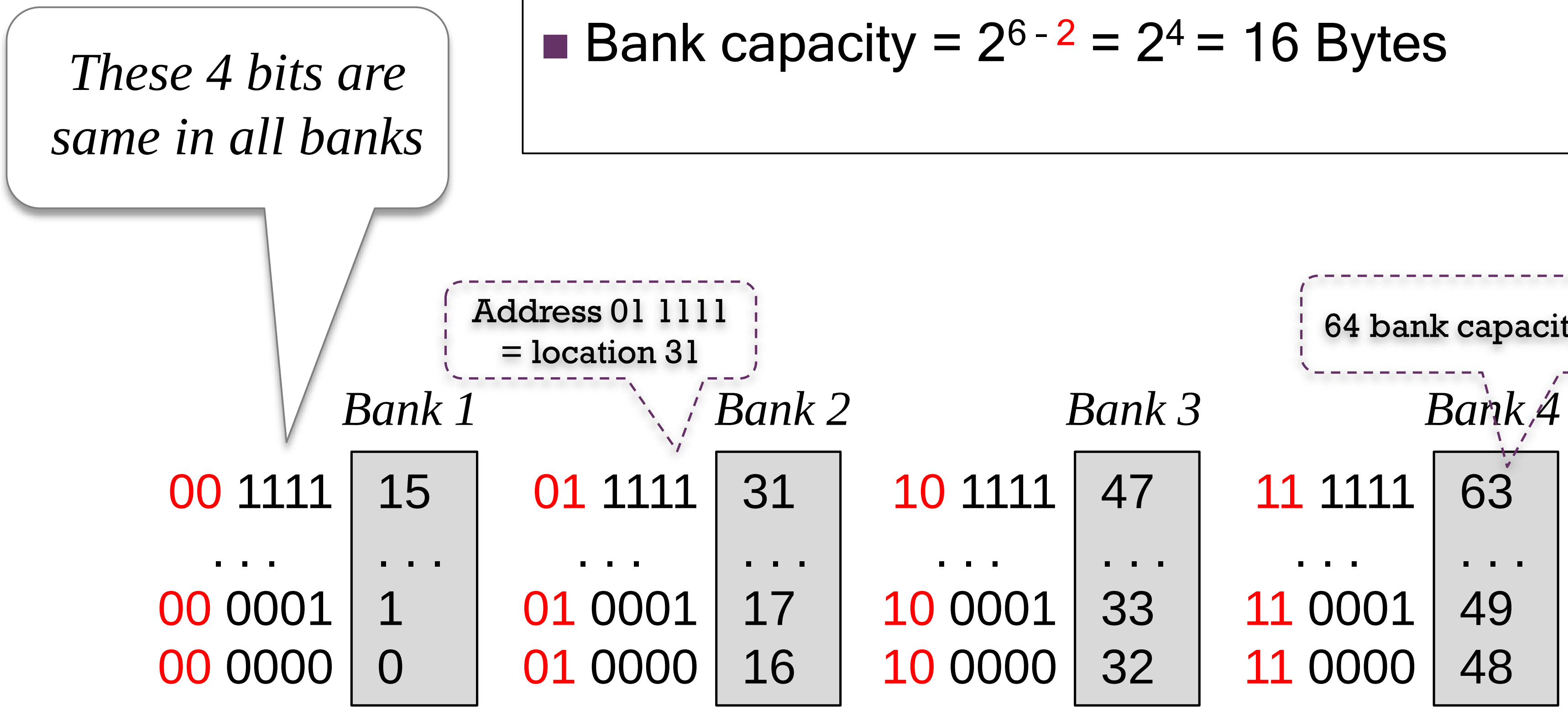
Because the data are stored sequentially across all memory banks

Example 5: HOI

- Memory capacity is 64 Bytes

  → 2^6 (Number of bit for memory address = 6)
- Total bank is 4

  → 2^2 (Number of bit for bank address = 2)
- Bank capacity = $2^{6-2} = 2^4$ = 16 Bytes

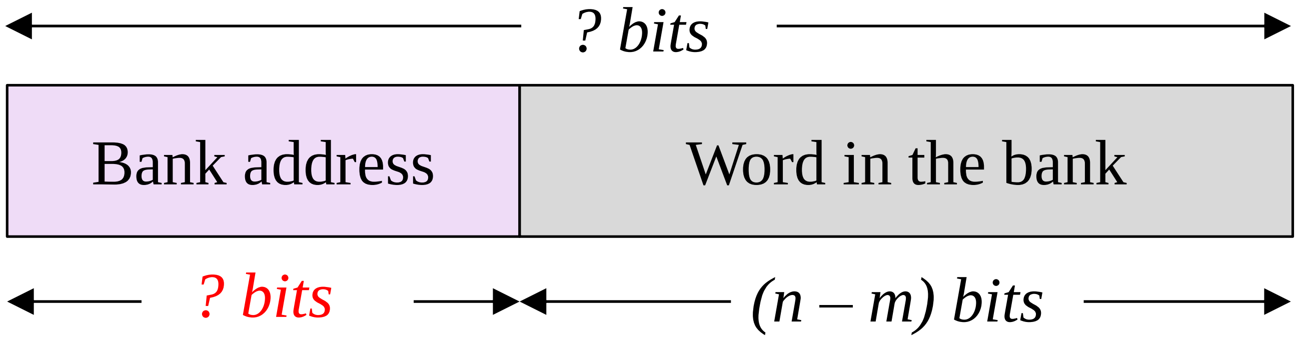


Activity 2

Example 6: HOI

Given a main memory has 32 *Mwords* with the size of each word is 1 *Byte*, and there are 16 memory banks. Draw the modular memory address format if the system is implemented with HOI.

Solution:



HOI: Advantages

- Easy memory extension by the addition of one or more memory modules to a maximum of $M - 1$.
- Provides better reliability, since a failed module affects only a localized area of the address space.
- This scheme would be used w/o conflict problems in multiprocessors if the modules are partitioned according to disjoint or non-interleaving processes (programs should be disjoint for its success).

HOI: Disadvantages

- Scheme will cause memory conflicts in case of pipelined, vector processors due to the sequentially placement of instructions and data in the same module/memory bank. Since memory cycle time is much greater than pipelined clock time, a previous memory request would not have completed its access before the arrival of the next request, thereby resulting in a delay.
- Process interacting and sharing instructions and data in multiprocessor system will encounter considerable conflicts.
- This technique is useful only in one single user system / single user multitasking system.

Review Questions

6.2.1. What is the difference between a byte and a word? What distinguishes each?

6.2.2. List and explain the two types of memory interleaving and the differences between them.

Module 6

Memory

Overview

- [*] *Cache memory* is designed to combine:
  - the memory access time of expensive, high-speed memory with
  - the large memory size of less expensive, lower-speed memory.
- [*] The cache contains a copy of portions 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*.

* William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.128

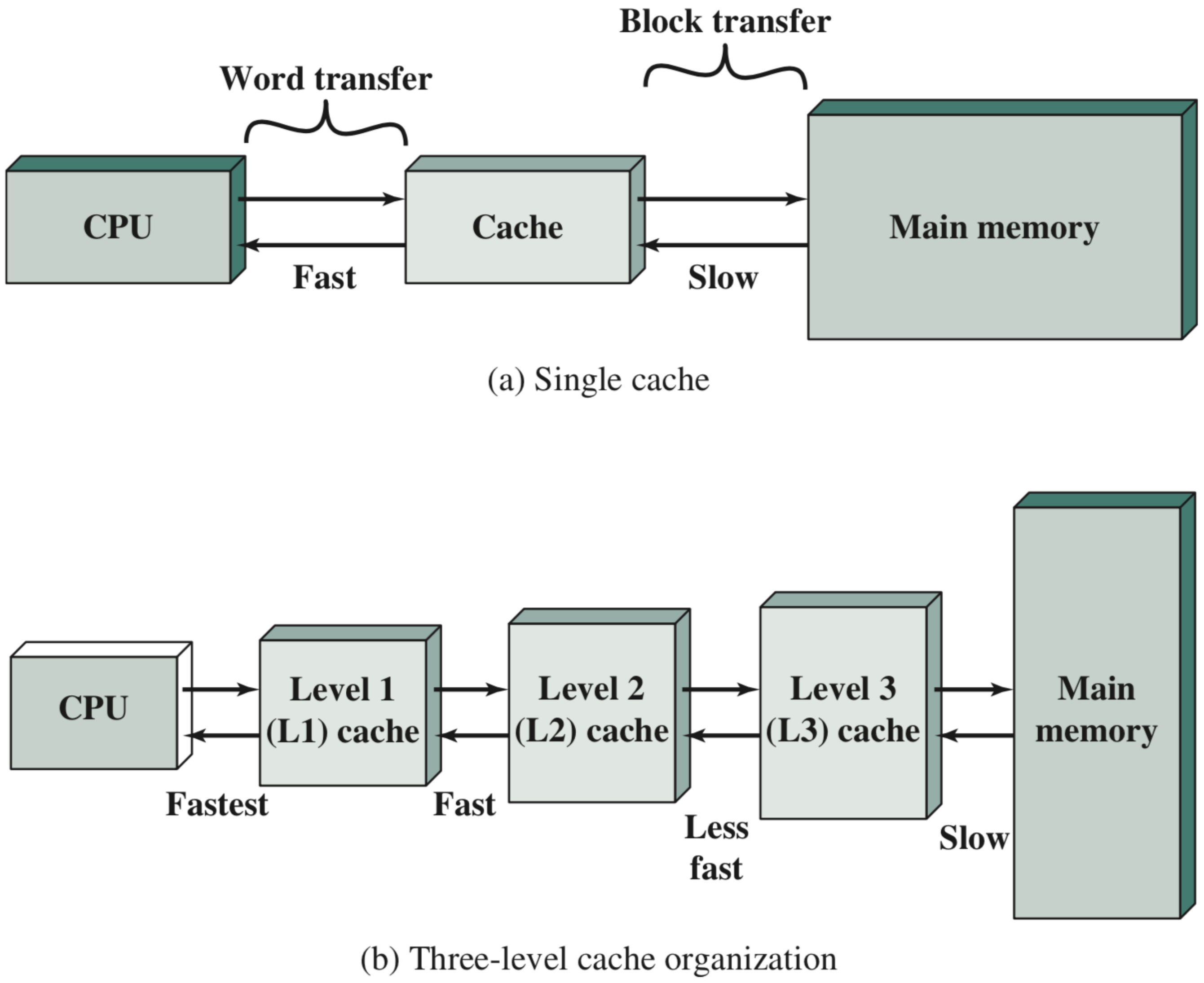


Figure: Cache and Main Memory

Cache Mapping Schemes

- The "content" that is addressed in *addressable cache memory* is a subset of the bits of a main memory address called a *field*.
- The fields into which a memory address is divided provide a many-to-one mapping between larger main memory and the smaller cache memory.
- Many *blocks* of main memory map to a single *block* of cache.
- A *tag* field in the cache block distinguishes one cached memory block from another.

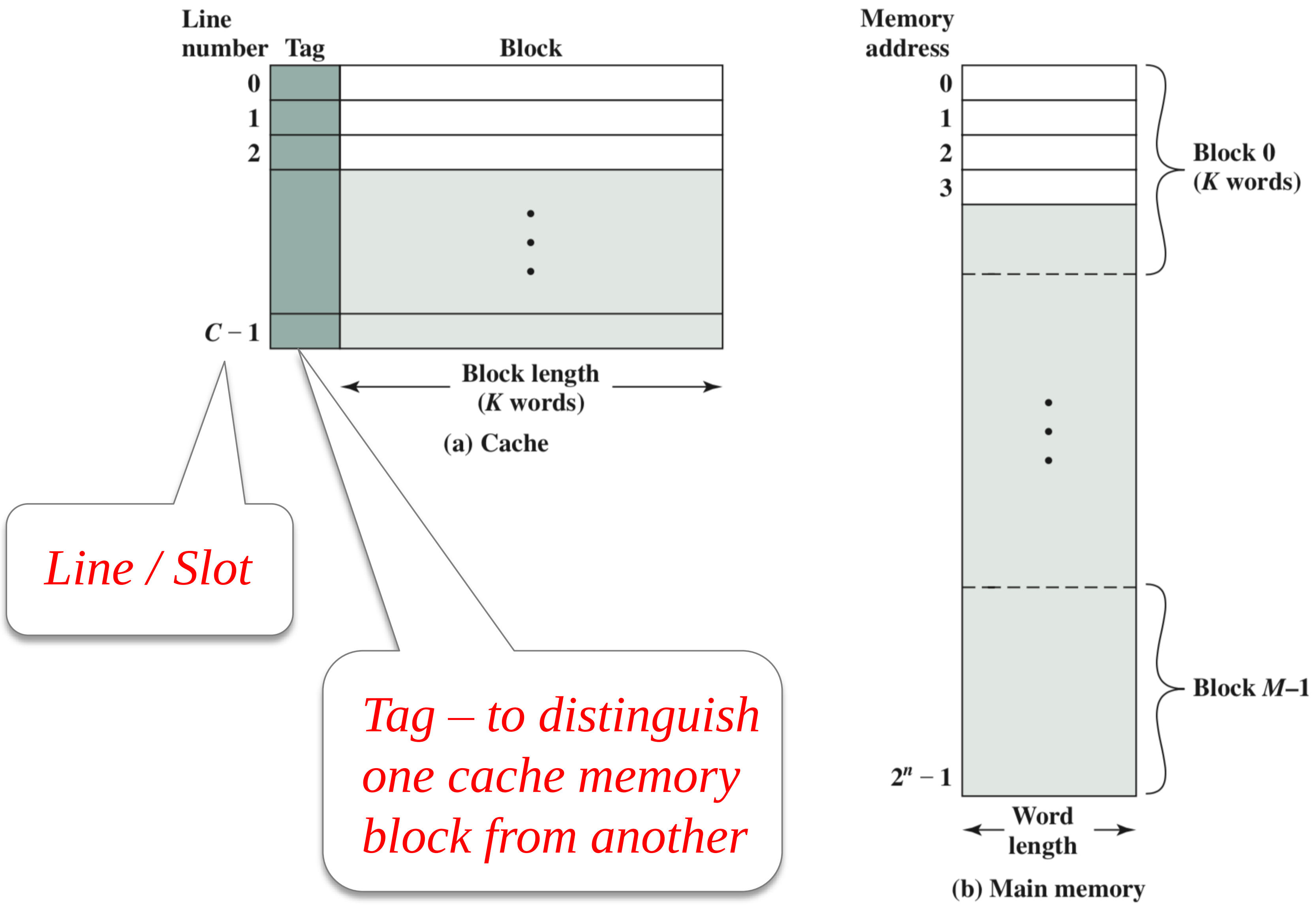


Figure: Cache/Main Memory Structure

How to map main memory address to cache memory address?

- Depending on the mapping scheme, cache may has two or three fields.
- These fields depends on the particular mapping scheme being used to determine where the data is placed when it is originally copied into cache.

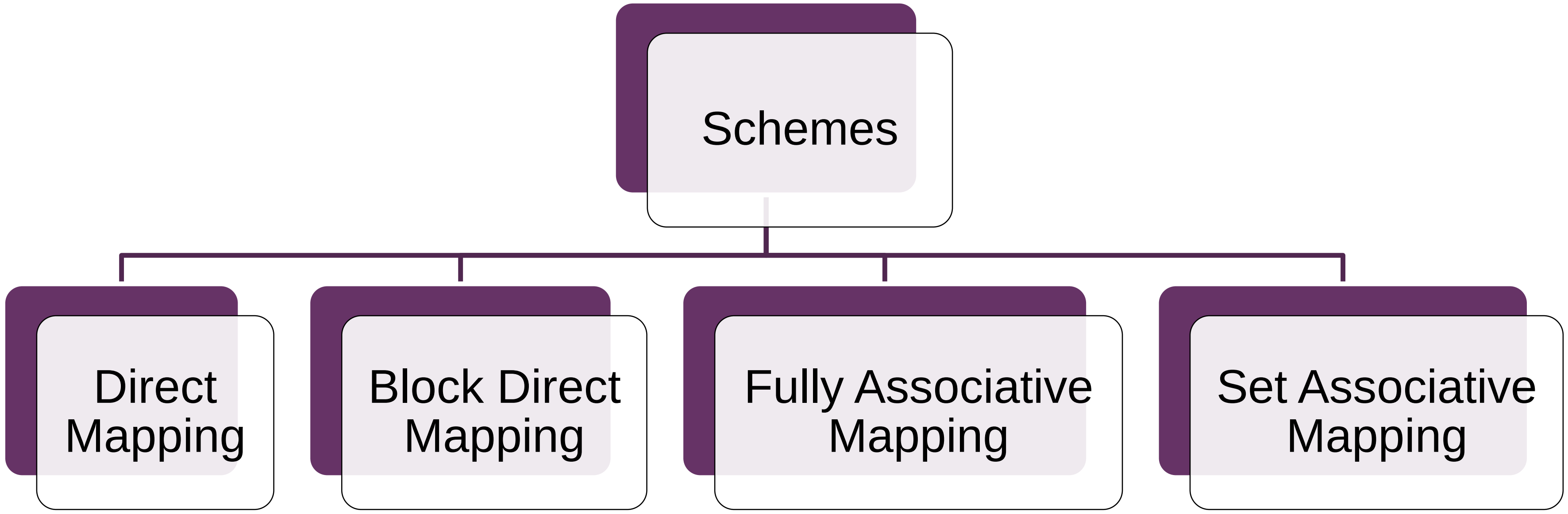
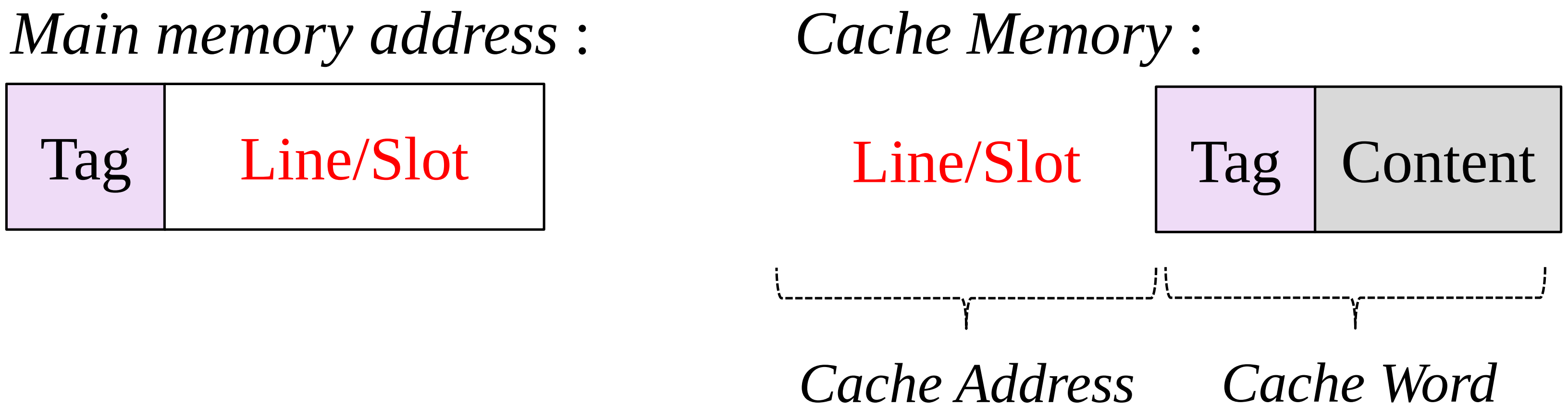


Figure: Cache mapping schemes

(a) Direct Mapping

- The simplest technique.
- [*]**A cache structure in which each memory location is mapped to exactly one location in the cache**.
- Address formats:



[*]Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.384

Example 7: Direct Mapping.

A main memory contains 8 words while the cache has only 4 words. Using direct address mapping, identify the fields of the main memory address.

Solution :

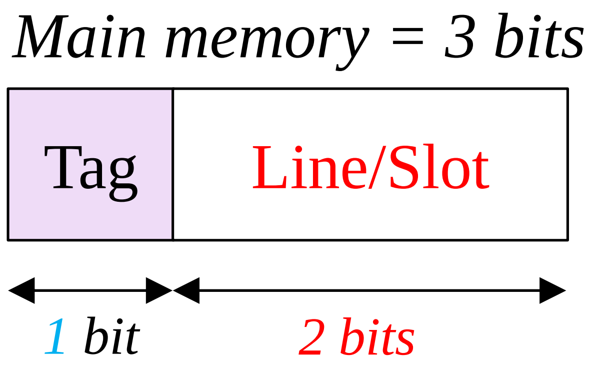
- Total memory words = 8 = 2^3

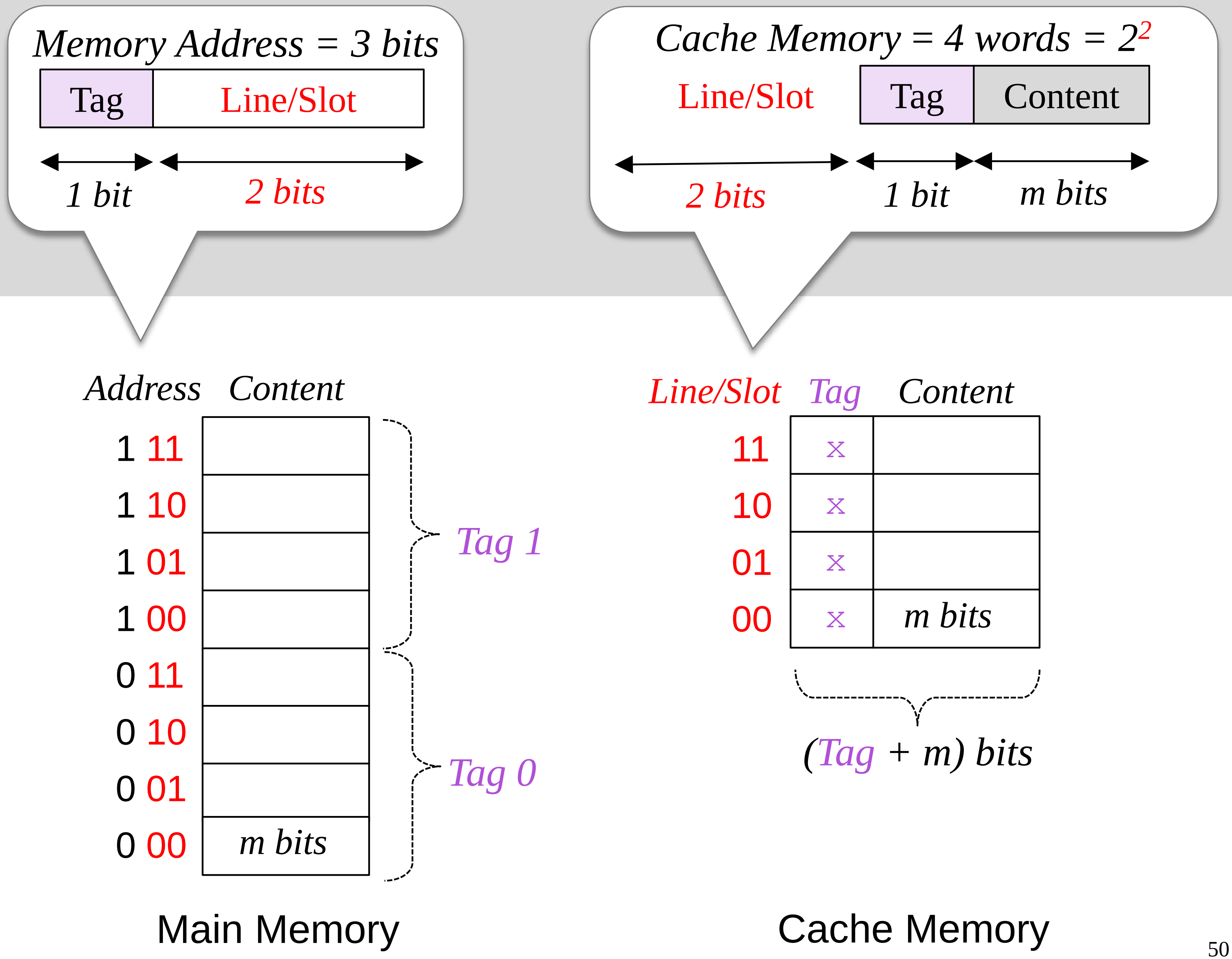
  → Require 3 bits for main memory address.

- Total cache words = 4 = 2^2

  → Require 2 bits for cache address (*Line/Slot*)

- Tag size = 3 – 2 = 1 bit




Memory Address = 3 bits
Tag
Line/Slot
1 bit
2 bits
Cache Memory = 4 words = 2^2
Line/Slot
Tag
Content
2 bits
1 bit
m bits
Address
Content
1 11
1 10
1 01
1 00
0 11
0 10
0 01
0 00
m bits
Tag 1
Tag 0
Line/Slot
Tag
Content
11
10
01
00
x
x
x
x
m bits
(Tag + m) bits
Main Memory
Cache Memory

Example 8: Direct Mapping.

A main memory contains 32 words while the cache has only 8 words. Using direct address mapping, identify the fields of the main memory address.

Solution :

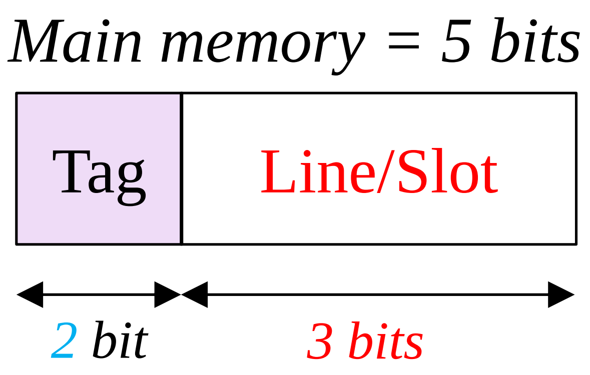
- Total memory words = 32 = 2^5

  → Require 5 bits for main memory address.

- Total cache words = 8 = 2^3

  → Require 3 bits for cache address (*Line/Slot*)

- Tag size = 5 – 3 = 2 bits



Example 9: Direct Mapping.

Size of cache memory is 64*Kword* and the size of main memory is 64*M* × 8 *bit* word. Determine the word size of main memory, cache and the main memory address format.

Solution :

- Total words in main memory = 64*M* = $2^6 \times 2^{20} = 2^{26}$

  → Require 26 bits for main memory address.

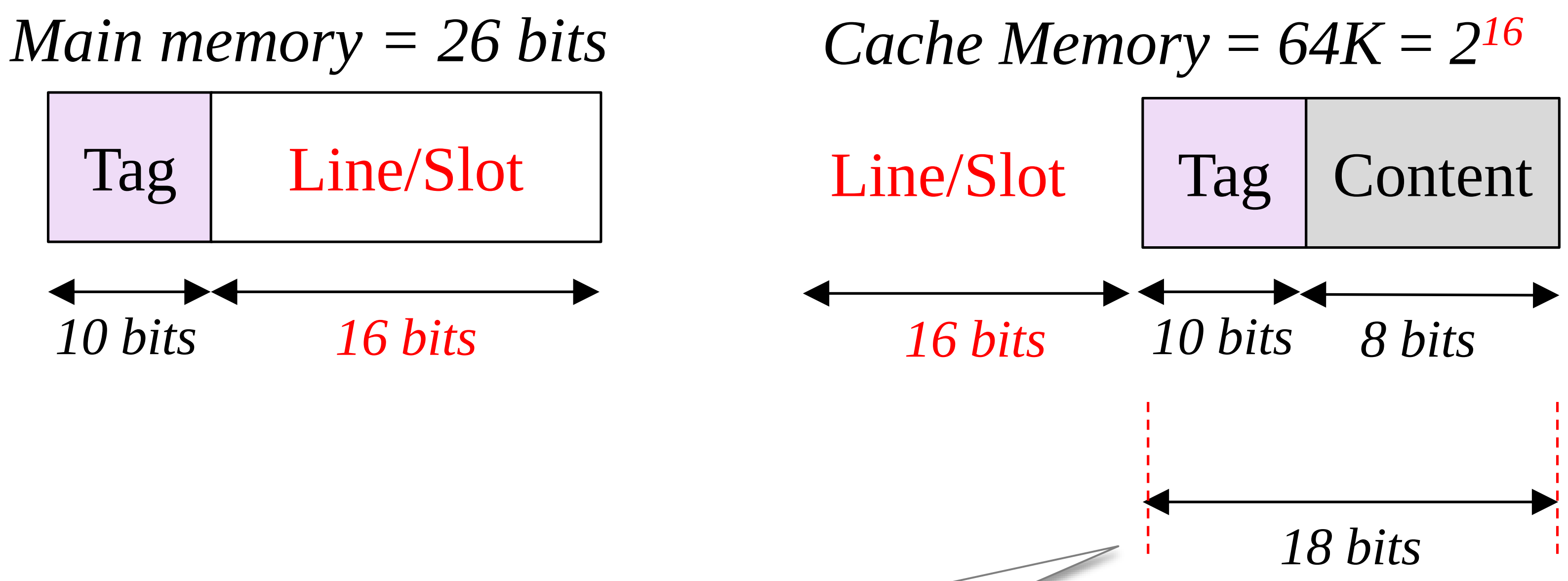
- Total cache words = 64*K* = $2^6 \times 2^{10} = 2^{16}$ → 16 bits for *Line/Slot*

- Tag size = 26 – 16 = 10 bits

- Size of main memory word = 8 *bits*

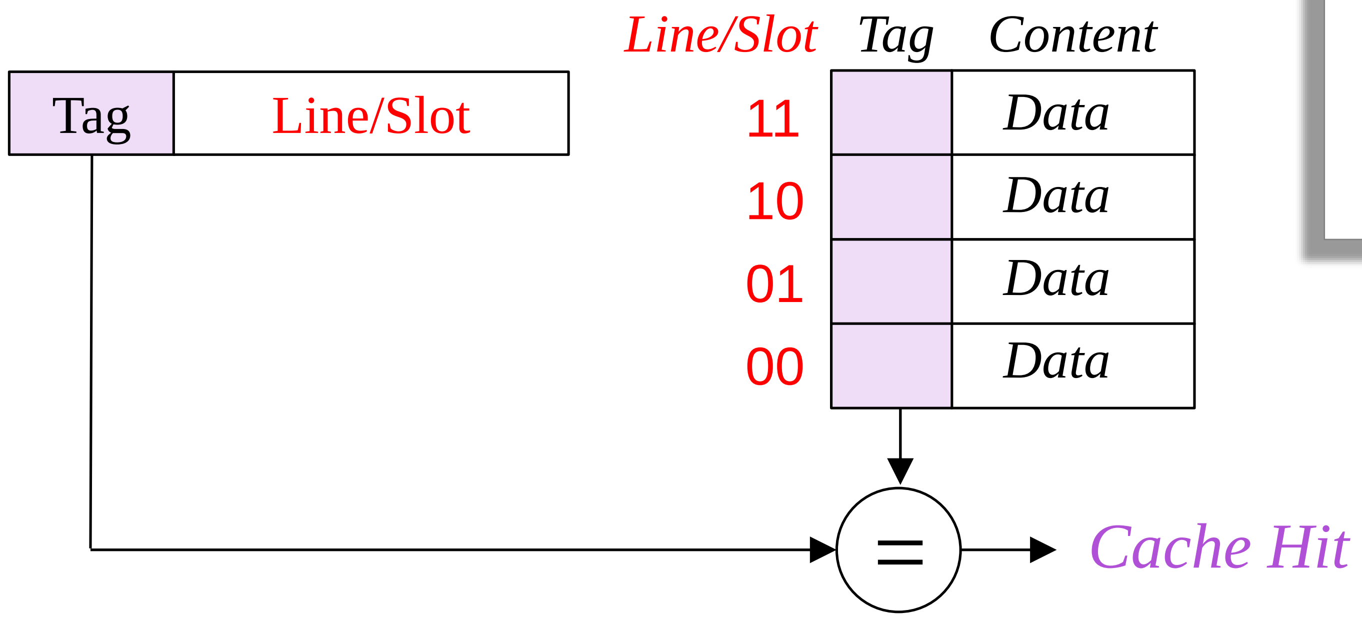
  → Size of cache word = Tag + (No. words in cache × size)

  = 10 + (1 × 8) = 18 *bits*



Cache word
= (Tag + size of content)
= 10 + 8 = 18 bits

Cache Hit and Miss



Main Memory

Cache Memory

*Cache Miss

A request for data from the cache that cannot be filled because the data is not present in the cache.

*Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.392

Example 10:

Cache hit and miss.

A main memory contains 16 words while the cache has only 4 words. Using direct address mapping, identify the fields of the main memory address.

Solution :

- Total memory words = 16 = 2^4

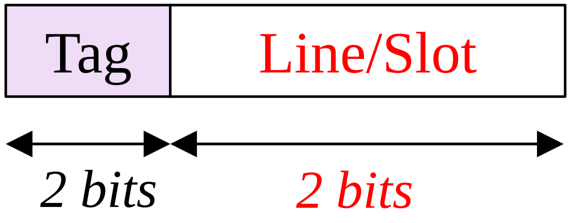
  → Require 4 bits for main memory address.

- Total cache words = 4 = 2^2

  → Require 2 bits for cache address (*Line/Slot*)

- Tag size = 4 – 2 = 2 bits

Main memory = 4 bits



Dec/Hex.	*Line/Slot*	*Tag*	*Content*
3	11	xx	
2	10	xx	
1	01	xx	
0	00	xx	

Cache Memory

Memory address:
→ (Tag | Line/Slot)

Based on previous example, consider the following main memory with contents.

Dec/Hex.	*Line/Slot*	*Tag*	*Content*
3	11	xx	xx
2	10	xx	xx
1	01	xx	xx
0	00	xx	xx

Cache Memory

Hex.	*Address*	*Content (Hex)*
F	11 11	11
E	11 10	10
D	11 01	01
C	11 00	A1
B	10 11	F5
A	10 10	F4
9	10 01	F3
8	10 00	F1
7	01 11	FF
6	01 10	91
5	01 01	10
4	01 00	19
3	00 11	13
2	00 10	02
1	00 01	01
0	00 00	EE

Main Memory

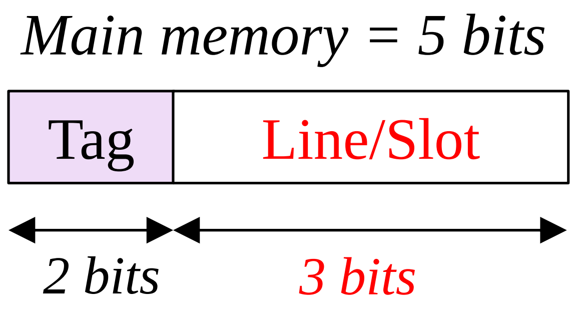
Example 11: Cache hit and miss operation.

(Read Process)

A main memory contains 32 words while the cache has only 8 words. Using direct address mapping, identify the fields of the main memory address.

Solution :

- Total memory words = 32 = 2^5 → 5 bits
- Total cache words = 8 = 2^3 → 3 bits (*Line/Slot*)
- Tag size = 5 – 3 = 2 bits



Memory address:
→ (Tag | *Line/Slot*)

Dec/Hex.	*Line/Slot*	*Tag*	*Content*
7	111	xx	xx
6	110	xx	xx
5	101	xx	xx
4	100	xx	xx
3	011	xx	xx
2	010	xx	xx
1	001	xx	xx
0	000	xx	xx

Cache Memory

Hex.	*/Dec*	*Address*	*Content (Hex)*
1F	31	11 111	11
1E	30	11 110	10
1D	29	11 101	01
1C	28	11 100	A1
1B	27	11 011	F5
1A	26	11 010	F4
19	25	11 001	F3
18	24	11 000	F1
17	23	10 111	FF
16	22	10 110	91
15	21	10 101	10
14	20	10 100	19
13	19	10 011	13
12	18	10 010	02
11	17	10 001	01
10	16	10 000	EE
…	…	…	…
3	3	00 011	A3
2	2	00 010	A1
1	1	00 001	AB
0	0	00 000	01

Main Memory

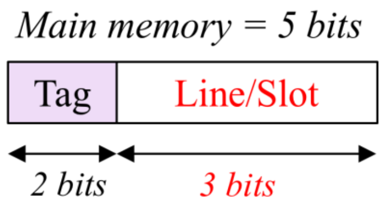
Based on the example, show the contents of the cache as it responds to a series of request (decimal address):

22, 26, 22, 26, 16, 3, 16, 18

Generated Address			Format Address	
Address		Content	Tag	Line/Slot
(Dec)	*(Binary)*	*(Hex)*		
22	10110	91	10	110
26	11010	F4	11	010
22	10110	91	10	110
26	11010	F4	11	010
16	10000	EE	10	000
3	00011	A3	00	011
16	10000	EE	10	000
18	10010	02	10	010

Main Memory

Complete the table by referring to previous slide



Based on the example, show the contents of the cache as it responds to a series of request (decimal address):

22, 26, 22, 26, 16, 3, 16, 18

Format Address		
Line/Slot	Tag	Content
		(Hex)
111	xx	xx
110	10	91
101	xx	xx
100	xx	xx
011	00	A3
010	~~11~~ 10	~~F4~~ 02
001	xx	xx
000	10	EE

Cache Memory

Format Address		Read/Write operation cache				
Tag	Line/Slot	Hit	Miss	Update cache	Read	Write
10	110		✓	✓		✓
11	01					✓
10	110				✓	
11	010				✓	
10	000		✓	✓		✓
00	011		✓	✓		✓
10	000	✓			✓	
10	010		✓	✓		✓

Main Memory

Cache memory after updating

Activity 3

Exercise 6.1:

Base on previous example, complete the following tables as they respond to a series of request (hexadecimal address):

1D, 14, 1E, 1D, 14, 17, 3, 1C

Format Address		
Line/Slot	Tag	Content (*Hex*)
111		
110		
101		
100		
011		
010		
001		
000		

Cache Memory

Format Address	
Tag	Line/Slot

Main Memory

Read/Write operation cache				
Hit	Miss	Update cache	Read	Write