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

6.1 Introduction

- Overview
- Type of Memory
- The Memory Hierarchy

6

Overview

- Most computers are built using the *Von Neumann* model, which is centered on memory.
- The *programs* that perform the processing are stored in _____.
- The memory unit that communicates directly with the CPU is called _____ *.
- 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.
5

6

- Internal memory is often equated with main memory, but there are other forms of internal memory:
  - The processor requires its own local memory → _____.
  - The control unit portion of the processor may also require its own internal 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).

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.121.
6

6

Table: Key Characteristics of Computer Memory Systems.

Location	**Performance**
Internal (e.g., processor registers, cache, main memory)	Access time
External (e.g., optical disks, magnetic disks, tapes)	Cycle time
Capacity	Transfer rate
Number of words	**Physical Type**
Number of bytes	Semiconductor
Unit of Transfer	Magnetic
Word	Optical
Block	Magneto-optical
Access Method	**Physical Characteristics**
Sequential	Volatile/nonvolatile
Direct	Erasable/nonerasable
Random	**Organization**
Associative	Memory modules

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.121. 7

6

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)

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.233 8

(a) RAM *(Random Access Memory)*

6

- RAM is also the "main memory" or _____.
- Used to store programs and data 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*).
  - _____.

*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 9

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

6

- 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*)
  - _____
  - Flash memory → essentially EEPROM with the added benefit .

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.234 10

6 The Memory Hierarchy

- Generally speaking, faster memory is _____ 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.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.235 11

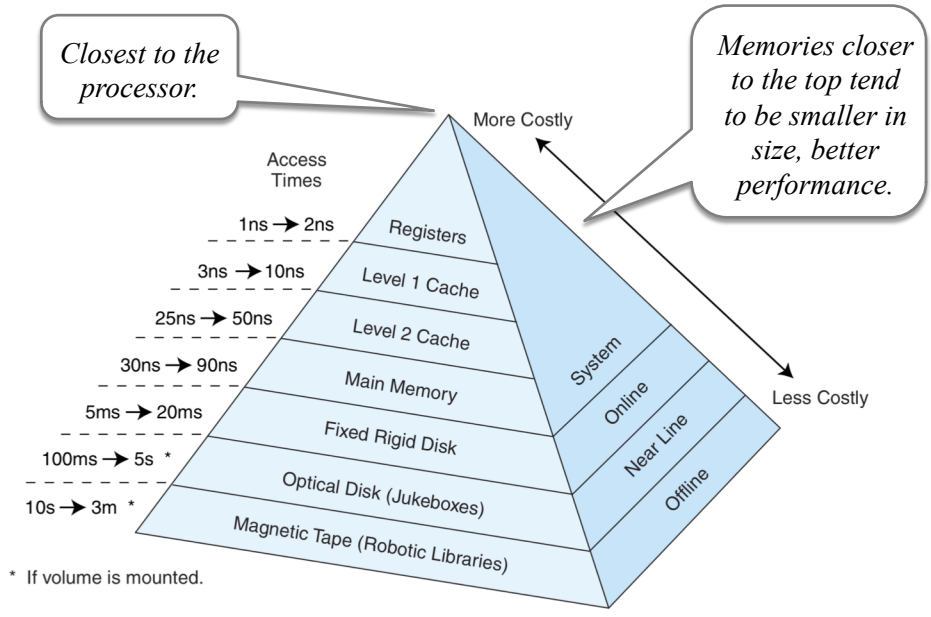
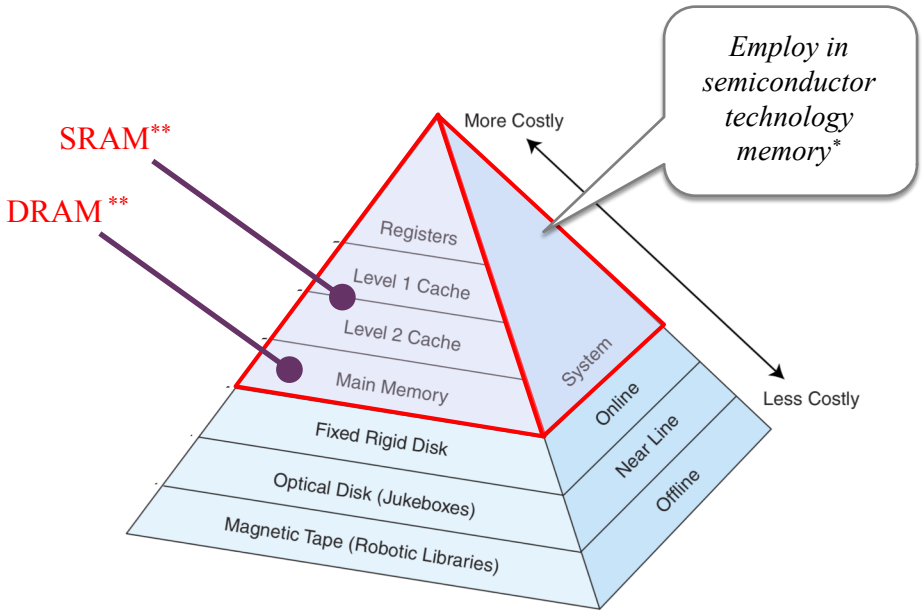


Figure: The memory hierarchy.

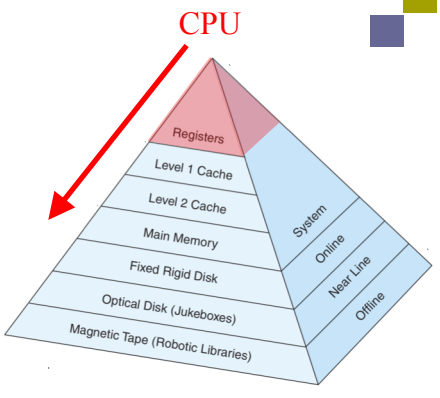
Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.236 12



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 13

6

- To access a particular piece of data, the CPU first sends a request to its nearest memory, usually _____.
  - 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 _____.

14

6

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

Terminology	Definition
Hit	The data is found at a given memory level.
Miss	The data is not found at a given memory level.
_____ *Rate*	The percentage (%) of memory accesses found in a given level of memory.
_____ *Rate*	The percentage (%) of memory accesses not found 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 time that it takes to replace a block of memory plus the time it takes to deliver the data to the processor.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.235 15

6

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

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.237 16

Module 6
Memory

17

6.1 Introduction
6.2 Main Memory
6.3 Cache Memory
6.4 Virtual Memory
6.6 Summary

- Overview
- Memory Organization
- Memory Capacity
- Memory Interleaving

RAM (Read-Access Memory)
ROM (Read-Only Memory)

6

Overview

- The _____ 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.

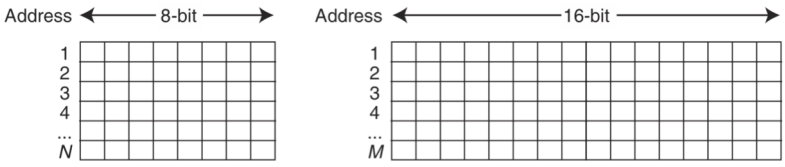
Mano, Morris M. (1993). *Computer System Architecture* (3rd Edition). p.448

18

6

Memory Organization

- We can envision memory as a matrix of bits.
- Each row, implemented by a _____, 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

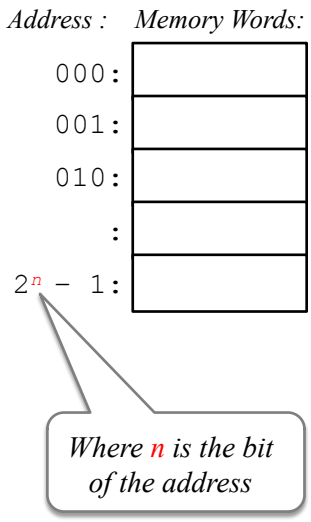
Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.153 19

Memory Location

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

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

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



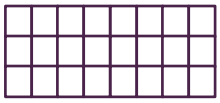
20

6 Memory Capacity

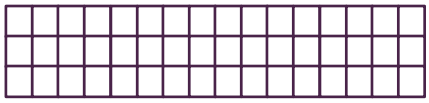
- Memory capacity usually measured in bits:

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

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



(a)



(b)

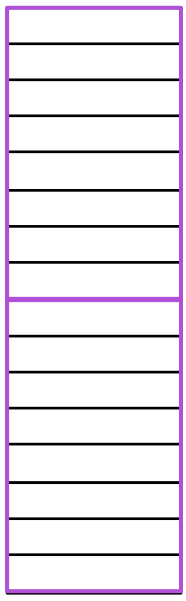
Size = 3 words x 8 bits
= 24 bits

Size =
=

21

1 *Word* = 8 *bits*

Memory



1 *Block* = 8 *words* = 2^3

1 *Block* = 8 *words* = 2^3

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?

22

6

Solution :

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

$$(b)\ Memory\ Capacity = Blocks \times Words \times Bits$$
$$32Kbit = Blocks \times 8 \times 8$$

23

6

Memory Interleaving

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

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

Low-Order Interleaving (LOI)	
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

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.155

24

LOI (Low-Order Interleaving)

6

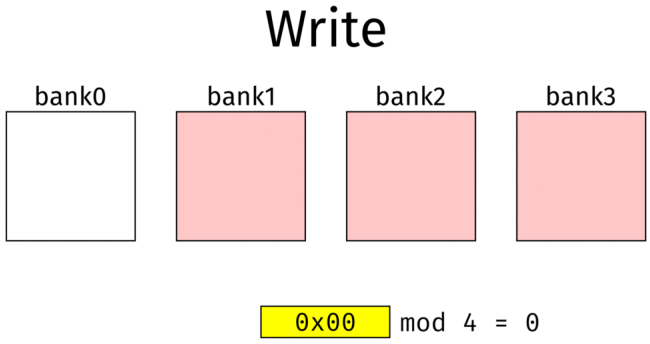


Figure: Memory interleaving (LOI) example with 4 banks. Red banks are refreshing and can't be used.

https://en.wikipedia.org/wiki/Interleaved_memory#/media/File:Interleaving.gif (11 Apr 2019) 25

6

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	
1	5	9	13	17	21	25	
2	6	10	14	18	22	26	
3	7	11	15	19	23	27	

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		7
8	9	10	11	12	13		15
16	17	18	19	20	21		23
24	25	26	27	28	29		31

Figure: Low-Order Interleaving (LOI).

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.156 26

6

Address format:

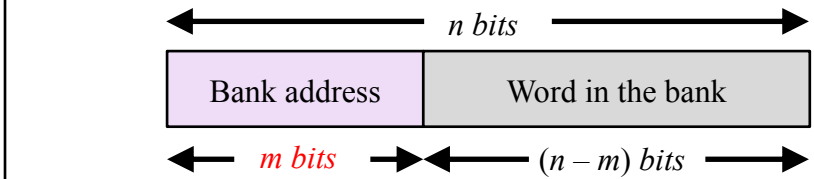


Figure: High-Order Interleaving (HOI).

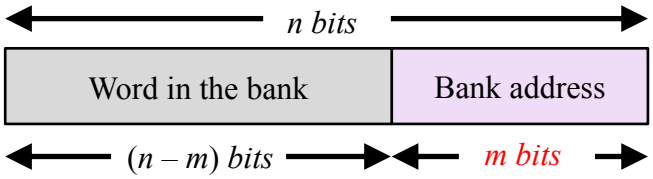


Figure: Low-Order Interleaving (LOI).

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.156 27

Example 3: LOI

- 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

These 4 bits are same in all banks

Bank 1		Bank 2		Bank 3		Bank 4	
1111 00	60	1111 01	61	1111 10	62	1111 11	63
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
0001 00	4	0001 01	5	0001 10	6	0001 11	7
0000 00	0	0000 01	1	0000 10	2	0000 11	3

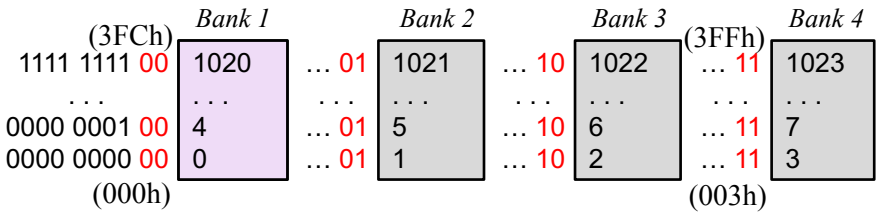
28

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.

Solution:

- 4 bank → 2^2 (Number of bit for bank address = 2)
- Bank capacity = $2^{10-2} = 2^8 = 256$ Bytes
- Memory address = 29Ch = 10 1001 1100



29

LOI: Advantages and Disadvantages

6

It produces memory interference.

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

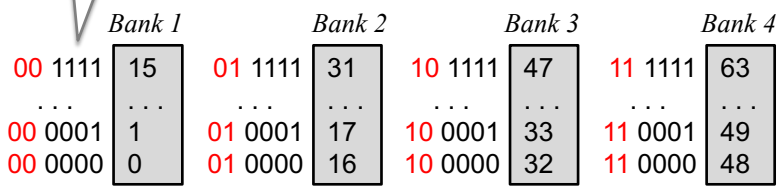
30

Example 5: HOI

(Same as Example 3)

- 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

These 4 bits are same in all banks



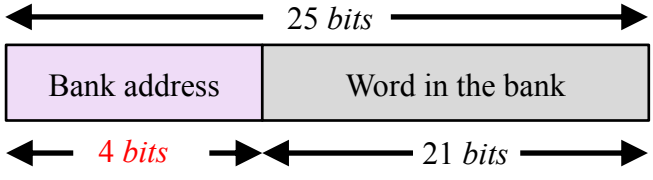
31

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:

- Main memory → 32 *Mwords* = 2^5 *Mwords* = $2^5 \times 2^{20}$ *bits* = 2^{25}
  (Number of bit for memory address = 25)
- 16 memory bank → 2^4 (Number of bit for bank address = 4)
- Word in the bank = 25 – 4 = 21 bits



32

HOI: Advantages

6

- 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).

33

HOI: Disadvantages

6

- Scheme will cause memory conflicts in case of pipelined, vector processors. The sequentially of instructions and data to be placed in the same module. 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.

34

Module 6
Memory

6

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 _____, 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

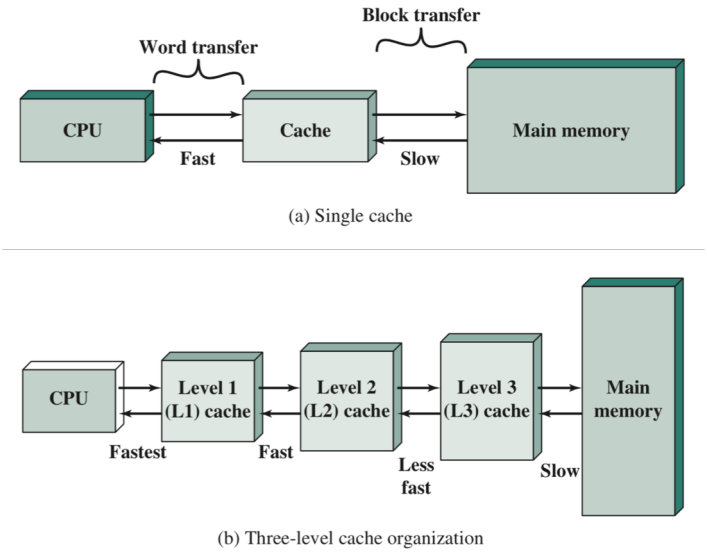


Figure: Cache and Main Memory

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.128 37

6 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 _____.
- 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 _____ field in the cache block distinguishes one cached memory block from another.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture*. United States: Jones and Bartlett Publishers. p.239 38

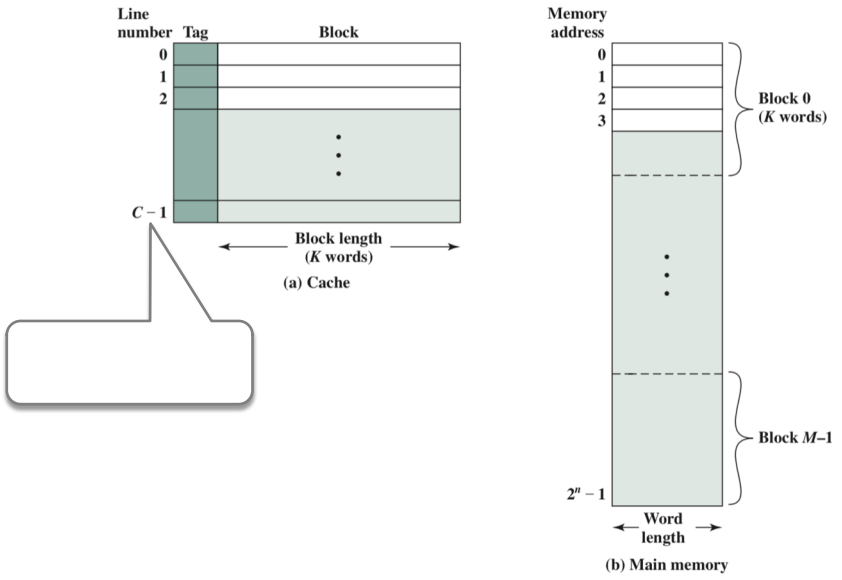


Figure: Cache/Main Memory Structure

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.129 39

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

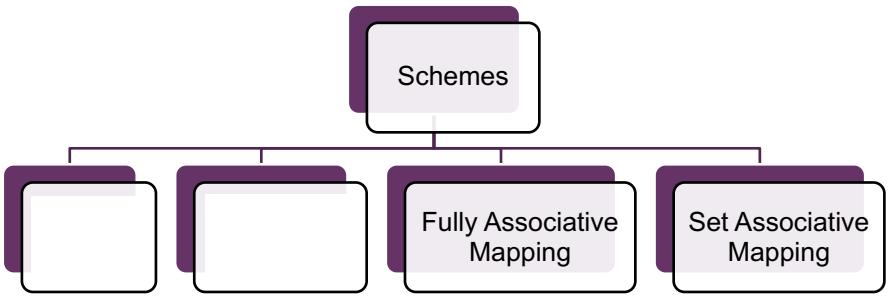


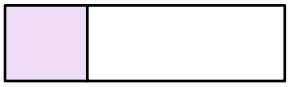
Figure: Cache mapping schemes

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.240 40

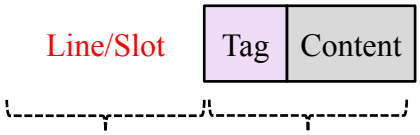
(a) Direct Mapping

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

Main memory address :



Cache Memory :



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

41

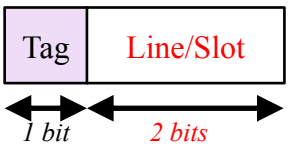
Example 7: Direct Mapping.

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.

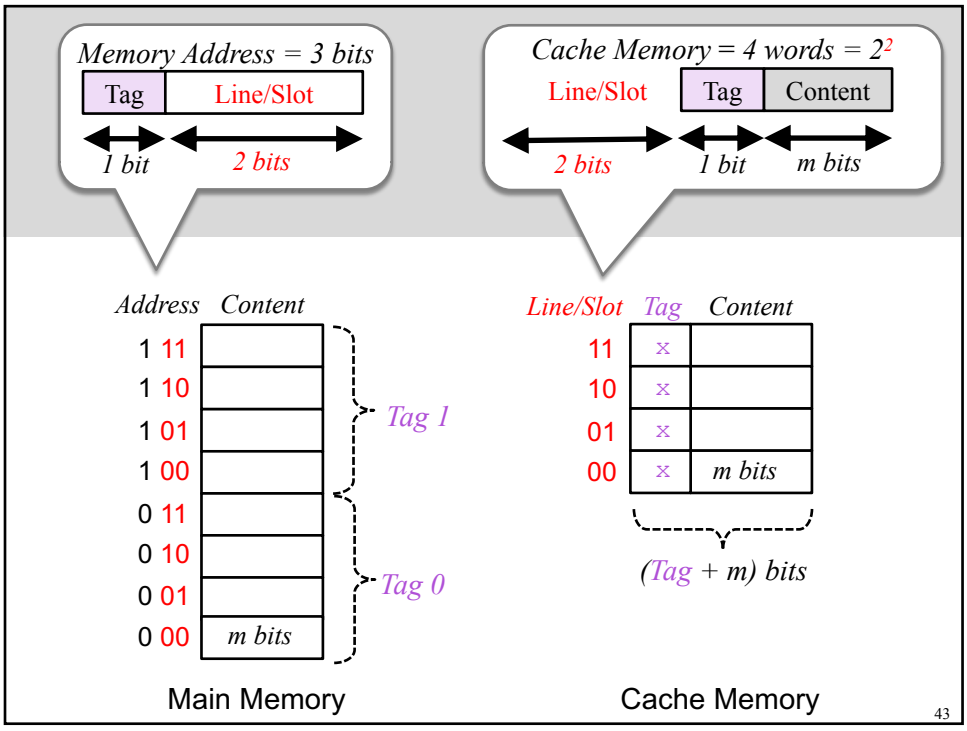
Solution :

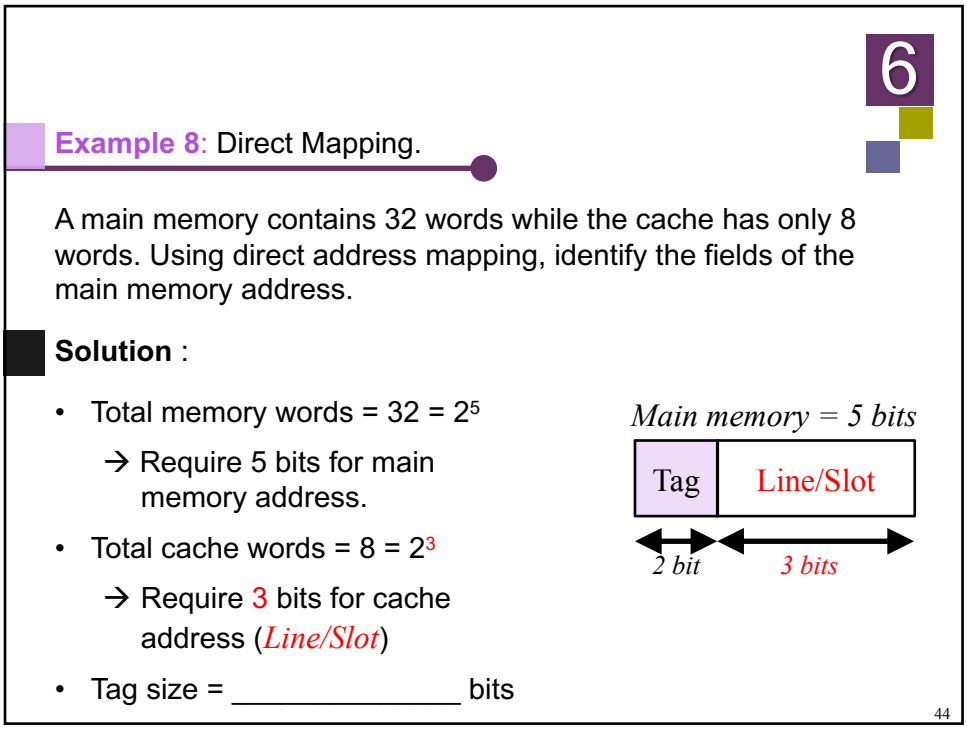
- Total memory words = 8 = 2^3
  → Require 3 bits for main memory address.
- Total cache words = 4 = 2^2
  → Require 2 bits for cache address (*Line/Slot*)
- Tag size = 3 – 2 = 1 bit

Main memory = 3 bits



42


Memory Address = 3 bits
Tag
Line/Slot
1 bit
2 bits
Cache Memory = 4 words = 2²
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
m bits
(Tag + m) bits
Main Memory
Cache Memory
43


6
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 = 2⁵
→ Require 5 bits for main memory address.
• Total cache words = 8 = 2³
→ Require 3 bits for cache address (Line/Slot)
• Tag size = _____ bits
Main memory = 5 bits
Tag
Line/Slot
2 bit
3 bits
44

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* = $2^6 \times 2^{20} = 2^{26}$

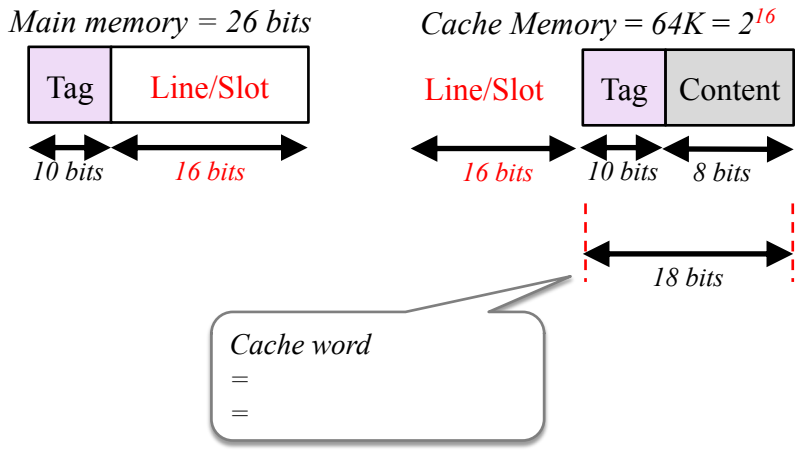
  → Require 26 bits for main memory address.
- Total cache words = 64*K* = $2^6 \times 2^{10} = 2^{16}$ → 16 bits for *Line/Slot*
- Tag size = 26 – 16 = 10 bits
- Size of main memory word = 8 *bits*

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

  = _____ *bits*

45

6

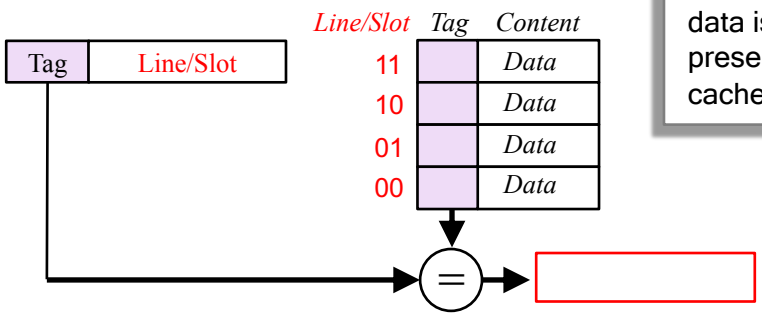


46

Cache Hit and Miss

Cache Miss

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



Main Memory Cache Memory

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

47

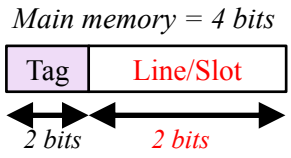
Example 10:

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



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

Cache Memory

48

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

Memory address:
→

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

(File: Module 6 – CacheHitMiss.doc) – Example 1 49

Example 11: Cache hit and miss operation.

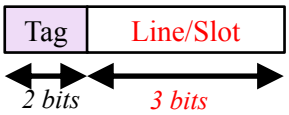
(Read Process)

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

Main memory = 5 bits



(File: Module 6 – CacheHitMiss.doc) – Example 2 50

Memory address:
→

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

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

(File: Module 6 – CacheHitMiss.doc) – Example 2 51

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		
26	11010	F4		
22	10110	91		
26	11010	F4		
16	10000	EE		
3	00011	A3		
16	10000	EE		
18	10010	02		

Main Memory

Complete the table

Main memory = 5 bits

Tag	Line/Slot

2 bits 3 bits

(File: Module 6 – CacheHitMiss.doc) – Example 2 52

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			Format Address	
Line/Slot	Tag	Content (Hex)	Tag	Line/Slot
111	xx	xx	10	110
110	10	91	11	010
101	xx	xx	10	110
100	xx	xx	11	010
011	00	A3	10	000
010	~~11~~ 10	~~E4~~ 02	00	011
001	xx	xx	10	000
000	10	EE	10	010

Cache Memory Main Memory

Read/Write operation cache				
Hit	Miss	Update cache	Read	Write

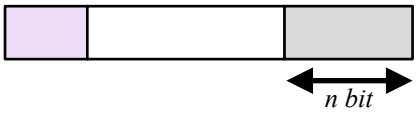
(File: Module 6 – CacheHitMiss.doc) – Example 2 53

(b) Block Direct Mapping

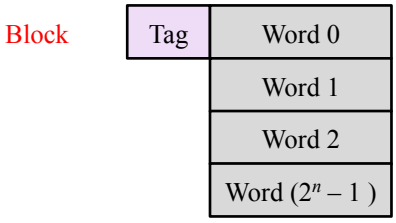
- The simplest technique of direct mapping can maps each block of main memory into only one possible cache line.
- Address formats: (2^n words, *e.g.* n = 2)

$= 2^n - 1$
$= 2^2 - 1$
$= 4 - 1 = 3$

Main memory address :



Cache Memory :



Cache Address *Cache Word*

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.134 54

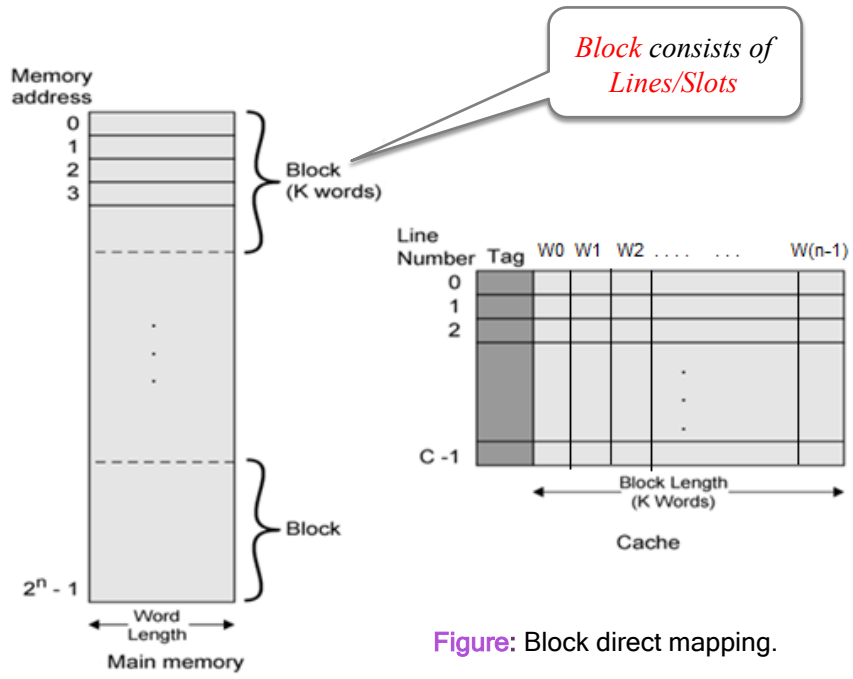
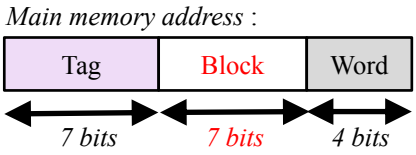


Figure: Block direct mapping.

55



Example 12:

Given the main memory address format using block direct mapping, if each main memory word is 8 bits, calculate:

a) The main memory capacity.

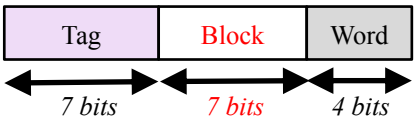
b) Total cache words.

c) The size of cache words.

56

Solution :

Main memory address :



a) Total main memory bits = ____

→ Total memory words = 2^{18} = 256*Kword*

→ Total memory capacity = 256*Kword x* 8 *bits* = 2*Mbit*

b) Block = 7 bits

→ Total cache words = 2^7 → 128 words (*Block*)

c) Total words = 2^4 = 16 words

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

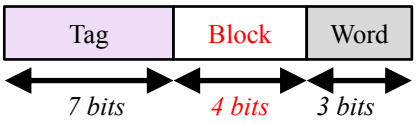
=

=

57

Example 13:

Main memory address :



Consider the main memory address.

Suppose a program generates the address 1AA.

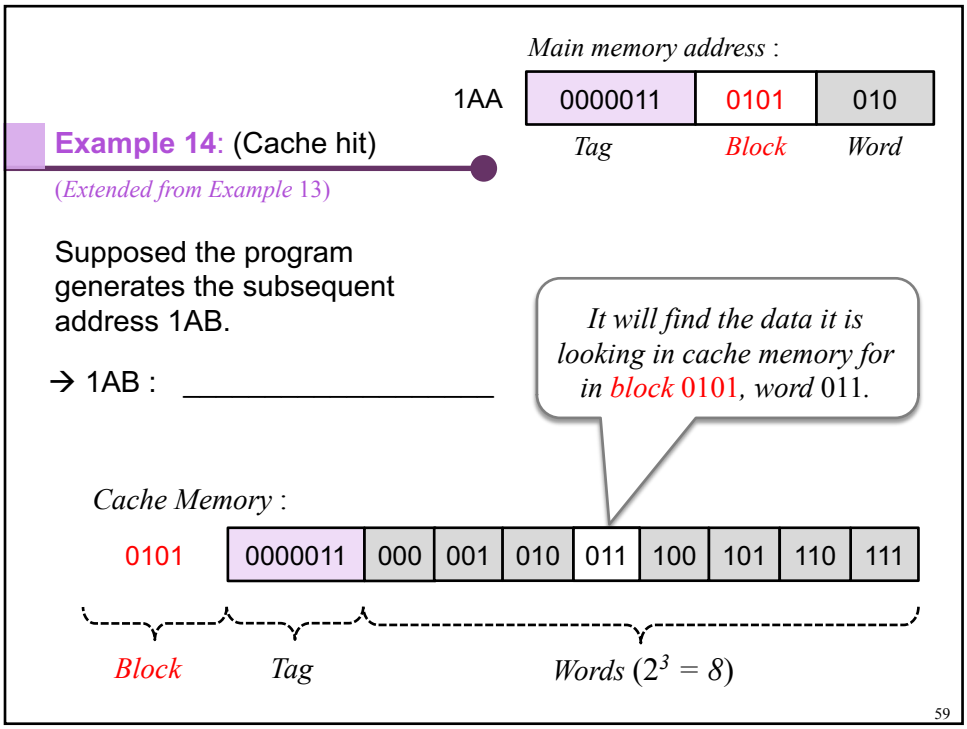
→ In 14-bit binary, this number is: 00 0001 1010 1010

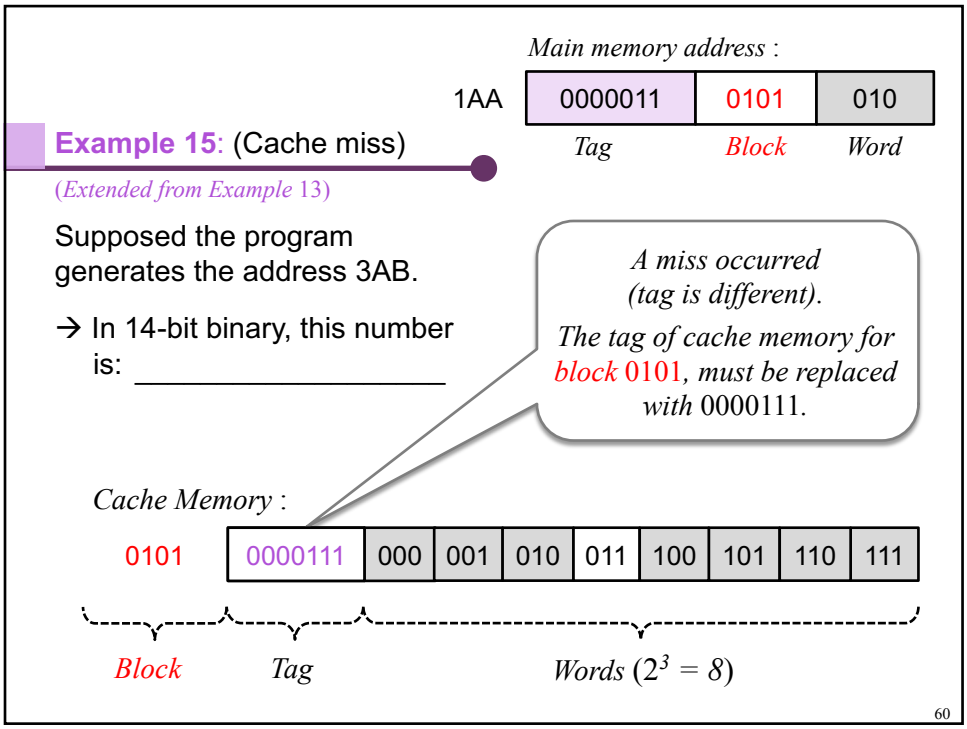
Padding with bit 0 at MSB to complete 14-bit

- ✓ The first 7 bits of the address go in the tag field.
- ✓ The next 4 bits go in the block field.
- ✓ The final 3 bits indicate the word within the block.

Main memory address :

58


Main memory address :
1AA
0000011
0101
010
Tag
Block
Word
Example 14: (Cache hit)
(Extended from Example 13)
Supposed the program generates the subsequent address 1AB.
→ 1AB : _____
It will find the data it is looking in cache memory for in block 0101, word 011.
Cache Memory :
0101
0000011
000
001
010
011
100
101
110
111
Block
Tag
Words ($2^3 = 8$)
59


Main memory address :
1AA
0000011
0101
010
Tag
Block
Word
Example 15: (Cache miss)
(Extended from Example 13)
Supposed the program generates the address 3AB.
→ In 14-bit binary, this number is: _____
A miss occurred (tag is different).
The tag of cache memory for block 0101, must be replaced with 0000111.
Cache Memory :
0101
0000111
000
001
010
011
100
101
110
111
Block
Tag
Words ($2^3 = 8$)
60

Block Direct Mapping:

Advantages and Disadvantages

Other cache mapping schemes are designed to prevent this kind of thrashing.

The technique is simple and inexpensive to implement.

- A fixed cache location for any given block. If a program accesses 2 line/slot/block that map to same line repeatedly, cache hit ratio will be low (Known as *thrashing*).
- Least effective in its utilization - that is, it may leave some cache lines unused because a given line/slot has fixed location.

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.138 61

6

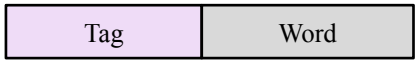
62

(c) Fully Associative Mapping

6

- Overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any location (line/slot/block) of the cache.
- A memory address is partitioned into only two fields:

Main memory address :



- With associative mapping, there is flexibility as to which block to replace when a new block is read into the cache.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.245
William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.138
*Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.403 63

6

- When the cache is searched for a specific main memory block, the tag field of the main memory address is compared to all the valid tag fields in cache;

  - ✓ if a match is found, _____.
  - ✓ If there is no match, we have a _____ and the block must be transferred from main memory.

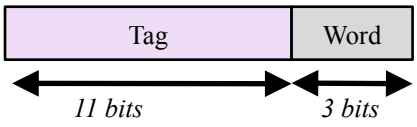
Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.245 64

Example 16:

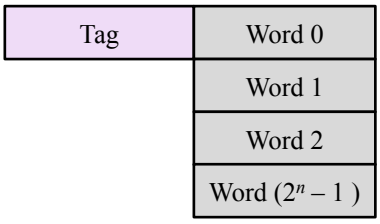
Consider a memory configuration with 2^{14} words, a cache with 16 blocks, and blocks of 8 words.

- Each block = 8 words = 2^3 → 3 bits (*Word*)
- Tag size = 14 – 3 = 11 bits

Main memory address = 14 bits



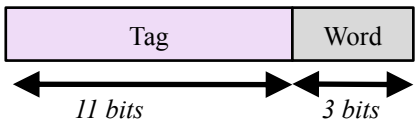
Cache Memory :



Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.245 65

Example 17:

Main memory address :

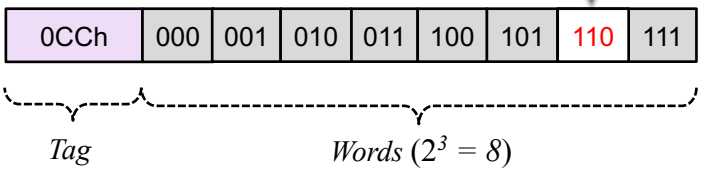


Supposed the program generates the memory address 666h.

→ In 14-bit binary, this number is: _____

It will find the data it is looking in cache memory for in tag 0CCh, *word* 110.

Cache Memory :

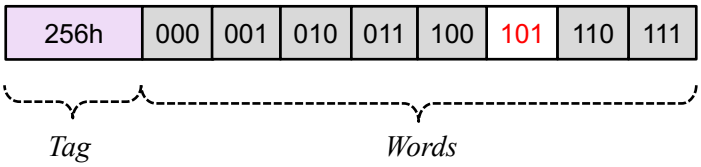


66

Example 18:

Consider a memory configuration with 2^{14} words, and blocks of 8 words. For data in cache as follows, what is the memory address (hexadecimal).

Cache Memory : (*Data*)



→ In 14-bit binary, the number 256h is: 0010 0101 0110 101

→ The memory address = 001 0010 1011 0101

= _____ h

67

6

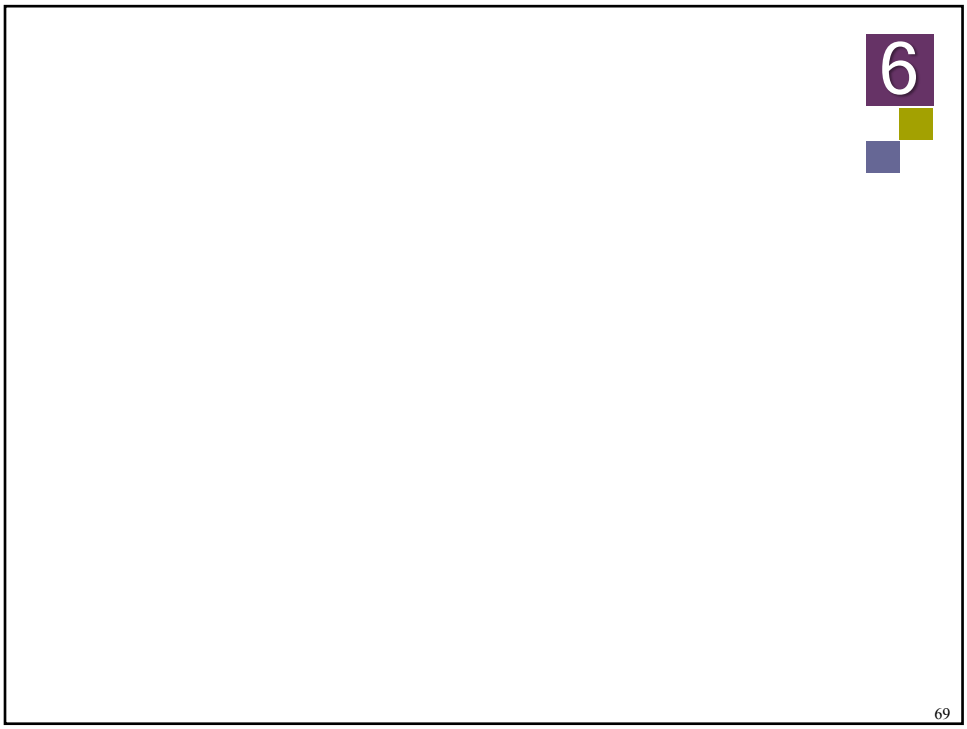
Fully Associative Mapping:

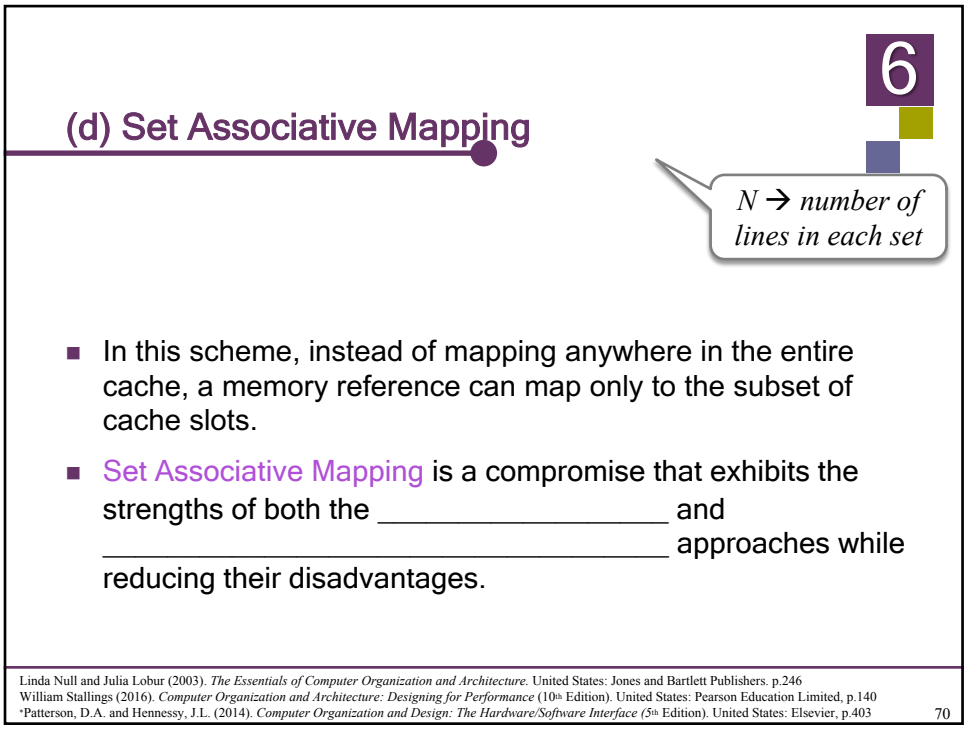
Advantages and Disadvantages

<u>Flexibility</u> scheme in term of which block to be replaced when a new block is read into the cache.

Every line's tag is examined for a match in fully associative cache (associative mapping), cache searching gets expensive.

William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.138 68


6
69


(d) Set Associative Mapping
6
N → number of lines in each set
In this scheme, instead of mapping anywhere in the entire cache, a memory reference can map only to the subset of cache slots.
Set Associative Mapping is a compromise that exhibits the strengths of both the _____ and _____ approaches while reducing their disadvantages.
Linda Null and Julia Lobur (2003). The Essentials of Computer Organization and Architecture. United States: Jones and Bartlett Publishers. p.246
William Stallings (2016). Computer Organization and Architecture: Designing for Performance (10th Edition). United States: Pearson Education Limited, p.140
*Patterson, D.A. and Hennessy, J.L. (2014). Computer Organization and Design: The Hardware/Software Interface (5th Edition). United States: Elsevier, p.403
70

6

- A set-associative cache with N locations for a block is called an N-way set-associative cache.
- An N-way set-associative cache consists of a number of sets, each of which consists of N blocks.
- A block is directly mapped into a set, and then all the blocks in the set are searched for a match.

- In set-associative cache mapping, the main memory address is partitioned into three pieces:

Main memory address :

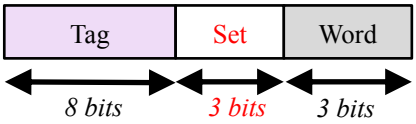
Cache memory : 1-*way*

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.246
Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.403

71

Example 19: 2-way

Main memory address :



- Suppose we have a main memory of 2^{14} bytes.
- It is mapped to a 2-way set associative cache having 16 blocks where each block contains 8 words.
- Since this is a 2-way cache, each set consists of 2 blocks, and there are 8 sets.
- Thus, we need 3 bits for the set, 3 bits for the word, giving 8 leftover bits for the tag.

72

Cache memory (Hex.):

Set	Tag	Word	Tag	Word
7				
7				
7				
7				
6	FF	7	FF	6
6	FF	5	FF	4
6	FF	3	FF	2
6	FF	1	FF	0

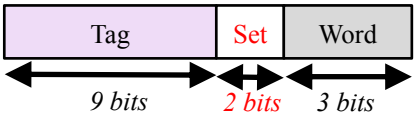
(Hex.) Memory Address:

	Tag | Set | Word
3FFF	11111111111111
3FFE	11111111111110
3FFD	11111111111101
3FFC	11111111111100
3FFB	11111111111011
3FFA	11111111111010
3FF9	11111111111001
3FF8	11111111111000
3FF7	11111111110111
3FF6	11111111110110
3FF5	11111111110101
3FF4	11111111110100
3FF3	11111111110011
3FF2	11111111110010
3FF1	11111111110001
3FF0	11111111110000

73

Example 20: 4-way

Main memory address :



- Suppose we have a main memory of 2^{14} bytes.
- It is mapped to a 4-way set associative cache having 16 blocks where each block contains 8 words.
- Since this is a 4-way cache, each set consists of 4 blocks, and there are 4 sets.
- Thus, we need 2 bits for the set, 3 bits for the word, giving 9 leftover bits for the tag.

74

(Hex.)	Memory Address:	(Hex.)	Memory Address:
3FFF	11111111111111	3FF7	11111111110111
3FFE	11111111111110	3FF6	11111111110110
3FFD	11111111111101	3FF5	11111111110101
3FFC	11111111111100	3FF4	11111111110100
3FFB	11111111111011	3FF3	11111111110011
3FFA	11111111111010	3FF2	11111111110010
3FF9	11111111111001	3FF1	11111111110001
3FF8	11111111111000	3FF0	11111111110000

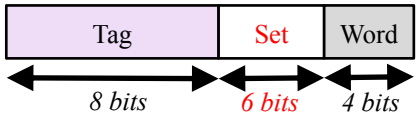
Cache memory (Hex.):

Set	Tag	Word	Tag	Word	Tag	Word	Tag	Word
3								
3								
2	FF	7	FF	6	FF	5	FF	4
2	FF	3	FF	2	FF	1	FF	0

75

Example 21:

Main memory address :



Suppose we want to read or write a word at the memory address 357A.

→ In 18-bit binary of the main memory, this number is:
00 0011 0101 0111 1010

Tag: 13 Set: 23 Word: 10

Under set-associative mapping, this translates to (dec).

→ So we search only the two tags in cache set 23 to see if either one matches tag 13. If so, we have a _____.

76

Figure: An eight-block cache configured as direct mapped, two-way set associative, four-way set associative, and fully associative.

One-way set associative (direct mapped)

Block	Tag	Data
0		000
1		001
2		010
3		011
4		100
5		101
6		110
7		111

Two-way set associative

Set	Tag	Data	Tag	Data
0		000		001
1		010		011
2		100		101
3		110		111

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0		000		001		010		011
1		100		101		110		111

Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data
	000		001		010		011		100		101		110		111

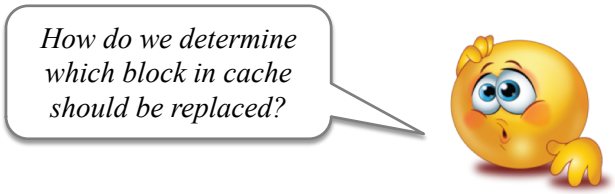
Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.404 77

78

6

Replacement Policy

- *Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks must be replaced.



- The algorithm for determining replacement is called the *replacement policy*.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture*. United States: Jones and Bartlett Publishers. p.247
* William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.145 79

6

- A replacement policy is invoked when it becomes necessary to evict a line/slot/block from cache.
- *There are several popular replacement policies:
  - → One that is not practical but that can be used as a benchmark by which to measure all others is the optimal algorithm.

- Although it is impossible to implement an optimal replacement algorithm, it is instructive to use it as a _____ for assessing the efficiency of any other scheme.

* Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture*. United States: Jones and Bartlett Publishers. p.247
William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.145 80

6

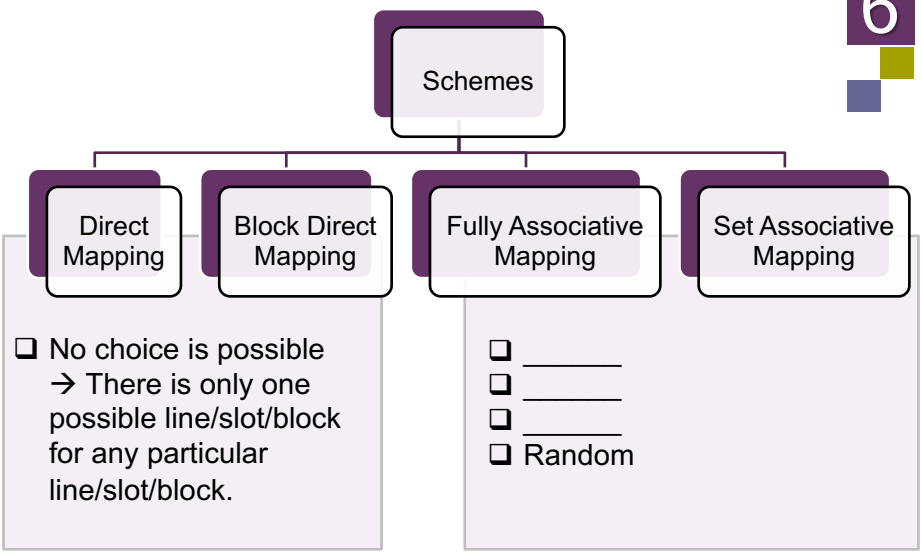


Figure: The replacement algorithms for cache memory.

81

6

Table: The replacement algorithms for cache memory.

Algorithms	Operational
FIFO *(First In First Out)*	Replace block that has been in cache longest.
LRU *(Least Recently Used)*	Keeps track of the last time that a block was assessed and evicts block that has been unused for the longest period of time.
LFU *(Least Frequently Used)*	Replace block which has had fewest hits.
Random	Picks a block at random and replaces it with a new block.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.248
William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.145

82

6

Which replacement algorithm is the best?

- ✓ The algorithm selected often depends on how the system will be used.
- ✓ No single (practical) algorithm is best for all scenarios.
- ✓ For that reason, designers use algorithms that perform well under a wide variety of circumstances.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.248 83

6

Cache Write Policies

- In addition to determining which victim to select for replacement, designers must also decide what to do with so-called _____ of cache, or blocks that have been modified in cache.
- Dirty blocks must be written back to memory.
- A write policy determines how this will be done.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.249
*William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.145 84

6

- *There are two basic write policies:

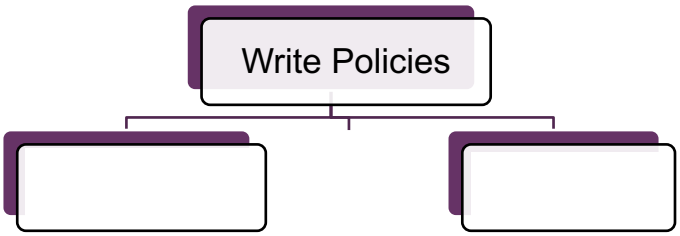


Figure: Cache write policy techniques

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.249
*William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.145
Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface* (5th Edition). United States: Elsevier, p.393 85

6

Cache Write Policies:
(a) Write Through

- A write-through policy updates both the _____ and the _____ simultaneously on every write.

Every write to the cache requires a main memory access, essentially slows the system down to main memory speed.

- However, in real applications, the majority of accesses are reads so this slow-down is negligible.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.249 86

Cache Write Policies: (b) Write Back

- A write-back policy (also called _____) only updates blocks in main memory when the cache block is selected as a victim and must be removed from cache.

- Normally faster than *write-through* because time is not wasted writing information to memory on each write to cache.
- Memory traffic is also reduced.

- Main memory & cache may not contain the same value at a given instant of time.
- Data in cache may be lost, if a process terminates (crashes) before the write to main memory is done.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.249 87

6 Cache Performances

- To improve the performance of cache, one must increase the _____ by using :
  - a better mapping algorithm (up to roughly a 20% increase),
  - better strategies for write operations (potentially a 15% increase),
  - better replacement algorithms (up to a 10% increase), and
  - better coding practices (up to a 30% increase in hit ratio).
- Simply increasing the size of cache may improve the hit ratio by roughly 1– 4%, but is not guaranteed to do so.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.249 88

Average Memory Access Time (AMAT)

6

- To capture the fact that the time to access data for both hits and misses affects performance, designers sometime use AMAT as a way to examine alternative cache designs.

$$\text{AMAT} = \textit{Time for a hit} + (\textit{Miss rate} \times \textit{Miss penalty})$$

- AMAT is the average time to access memory considering both _____ and _____ and the frequency of different accesses.

Patterson, D.A. and Hennessy, J.L. (2014). *Computer Organization and Design: The Hardware/Software Interface (5th Edition)*. United States: Elsevier, p.402 89

Example 22:

Assume that 33% of the instructions in a program are data accesses. The cache hit ratio is 97% and the hit time is one cycle, but the miss penalty is 20 cycles.

$$\text{AMAT} = \textit{Time for a hit} + (\textit{Miss rate} \times \textit{Miss penalty})$$
$$=$$
$$=$$
$$=$$

If the cache was perfect and never missed, the AMAT would be one cycle. But even with just a 3% miss rate, the AMAT here increases 1.6 times!

90

Effective Access Time (EAT)

6

- The performance of a hierarchical memory is measured by its EAT, or the average time per access.
- EAT is a weighted average that uses the hit ratio and the relative access times of the successive levels of the hierarchy.

$$EAT = (H \times Access_C) + ((1 - H) \times Access_{MM})$$

H : Cache hit rate
$Access_C$: Access time for cache
$Access_{MM}$: Access time for main memory

This formula can be extended to apply to three- or even four-level memories

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.248
William Stallings (2016). *Computer Organization and Architecture: Designing for Performance* (10th Edition). United States: Pearson Education Limited, p.147 91

Example 23:

Suppose the cache access time is 10ns, main memory access time is 200ns, and the cache hit rate is 99%.

The average time for the processor to access an item in this two-level memory would then be:

$$EAT = (H \times Access_C) + ((1 - H) \times Access_{MM})$$
$$=$$
$$=$$
$$=$$

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.248 92

Module 6
Memory

93

6.1 Introduction
6.2 Main Memory
6.3 Cache Memory
6.4 Virtual Memory
6.6 Summary

- Overview
- A Real-World Example: Pentium

6

Overview

- We now know that caching allows a computer to access frequently used data from a smaller but faster cache memory. (Cache is found near the top of our memory hierarchy).
- Another important concept inherent in the hierarchy is _____.

The purpose of virtual memory is to use the hard disk as an extension of RAM, thus increasing the available address space a process can use.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.250 94

- Virtual memory can be implemented with different techniques:

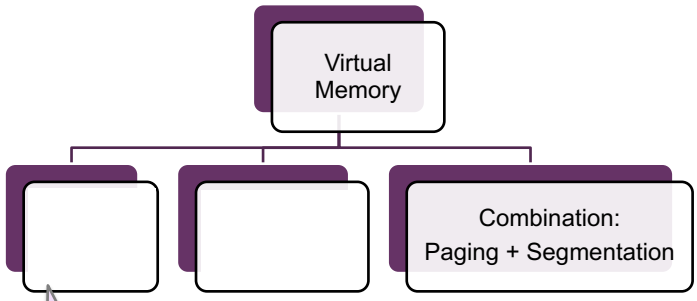
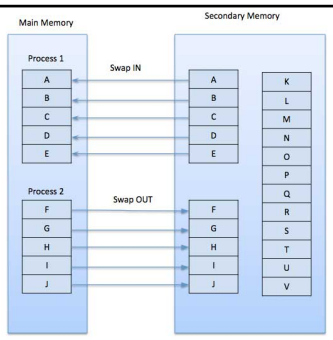


Figure: Virtual memory techniques

The most popular techniques !

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.251 95

Paging



- The most common way to implement virtual memory is by using *paging*.
- _____ is divided into fixed-size blocks and programs are divided into the same size blocks.
- Typically, chunks of the program are brought into memory as needed.
- It is not necessary to store contiguous chunks of the program in contiguous chunks of main memory.
- Every virtual address for a program that generated by CPU must be translated into a _____.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.250 96

Table: Some frequently used terms for virtual memory implemented through paging.

Terms	Description
Virtual address	The logical or program address generated by CPU.
Physical address	The real address in physical memory.
Mapping	Mechanism by which virtual addresses are translated into physical memory.
.....	The equal-size chunks in main memory.
Pages	The equal-size chunks in virtual memory.
Paging	The process of copying a virtual page from disk to a page frame in main memory.
.....	Memory that becomes unusable.
.....	An event when a requested page is not in main memory and must be copied into memory from disk.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.250-251 97

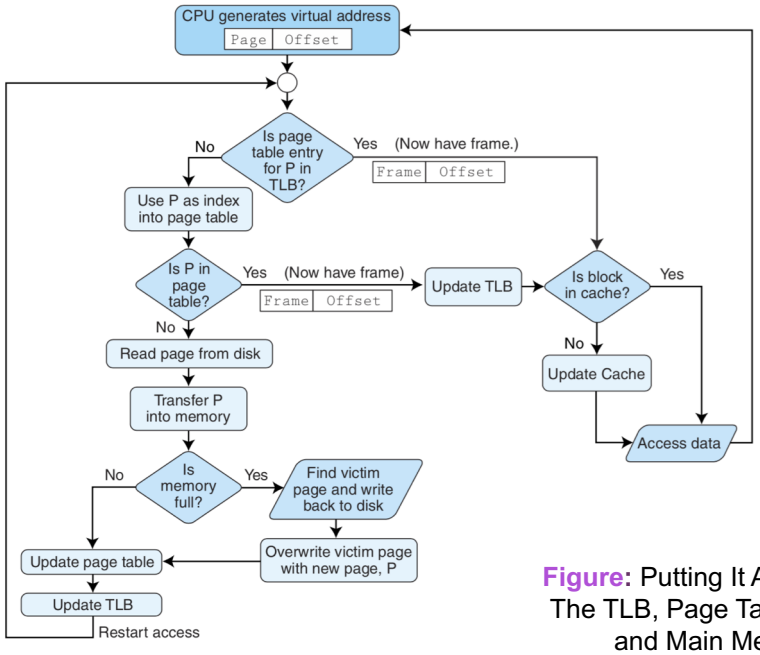


Figure: Putting It All Together: The TLB, Page Table, Cache and Main Memory

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.261 98

6

Example 25:

Suppose a main memory access requires 200ns and that the page fault rate is 1% (99% of the time with the pages needed are in memory). Assume it costs about 10ms to access a page not in memory (this time of 10ms includes the time necessary to transfer the page into memory, update the page table, and access the data).

The effective access time for a memory access is now:

$$EAT = 0.99\,(200ns + 200ns) + 0.01\,(10ms)$$
$$=$$
$$=$$
$$=$$

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.258 99

Example 26:

Based on previous example, even if 100% of the pages were in main memory, the effective access time would be:

$$EAT = 100\%\,(200ns + 200ns) + 0$$
$$=$$
$$=$$

which is double the access time of memory.

Accessing the page table costs an additional memory access because the page table itself is stored in main memory.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.258 100

6 A Real-World Example: Pentium

- The Pentium architecture exhibits fairly characteristic traits of modern memory management.
  - 32-bit virtual addresses and 32-bit physical addresses.
  - Uses either 4KB or 4MB page sizes, when using paging in different combinations of paging and segmentation.
  - Two caches, L1 and L2, both utilizing a 32-byte block size.
  - Both L1 caches (*I-cache* and *D-cache*) utilize an LRU bit for dealing with block replacement.
  - Each L1 cache has a TLB (*Translation Lookaside Buffer*).
  - The L2 cache can be from 512KB up to 1MB.

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.263 101

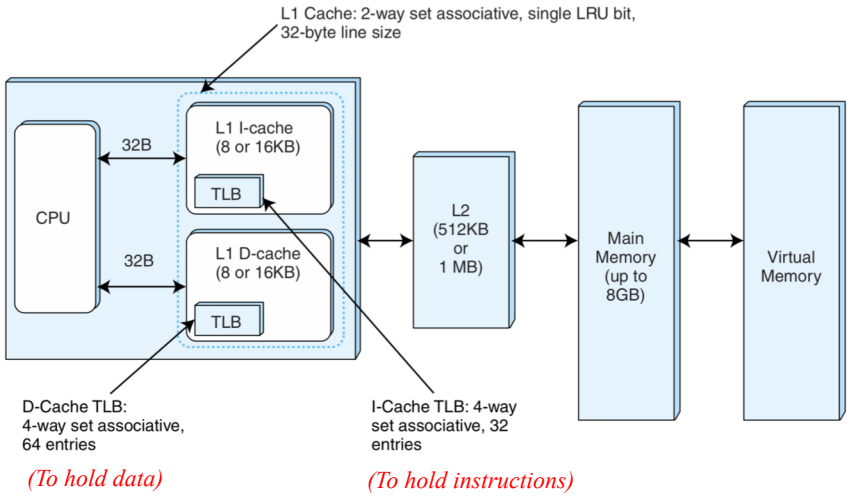


Figure: Pentium memory hierarchy

Linda Null and Julia Lobur (2003). *The Essentials of Computer Organization and Architecture.* United States: Jones and Bartlett Publishers. p.264 102

6.5 Summary

6

- Computer memory is organized in a hierarchy, with the smallest, fastest memory at the top and the largest, slowest memory at the bottom.
- Cache memory gives faster access to main memory, while virtual memory uses disk storage to give the illusion of having a large main memory.
- Cache maps blocks of main memory to blocks of cache memory. Virtual memory maps page frames to virtual pages.
- There are three general types of cache: Direct mapped, fully associative and set associative.

103

6

- With fully associative and set associative cache, as well as with virtual memory, replacement policies must be established.
- Replacement policies include LRU, FIFO, or LFU. These policies must also take into account what to do with dirty blocks.
- All virtual memory must deal with fragmentation, internal for paged memory, external for segmented memory.

104