

## **Chapter 3**

# **Memory Management: Virtual Memory**

### **At a Glance**

#### **Instructor's Manual Table of Contents**

- Overview
- Objectives
- Teaching Tips
- Quick Quizzes
- Class Discussion Topics
- Additional Projects
- Additional Resources
- Key Terms

## Lecture Notes

### Overview

This chapter follows the evolution of virtual memory by covering four memory allocation schemes that first remove the restriction of storing the programs contiguously and then eliminate the requirement that the entire program reside in memory during its execution: paged, demand paging, segmented, and segmented/demand paged allocation. These schemes form the foundation for our current virtual memory methods. The mechanics of paging and page replacement policies are presented with a discussion on how a memory allocation scheme determines which pages should be swapped out of memory. Next, an explanation of the concept of the working set and its use in memory allocation schemes is provided. The chapter also presents an explanation of virtual memory, including an explanation of its impact on multiprogramming. Finally, the discussion of cache memory shows how its use improves the performance of the Memory Manager.

### Learning Objectives

After completing this chapter, the student should be able to describe:

- The basic functionality of the memory allocation methods covered in this chapter: paged, demand paging, segmented, and segmented/demand paged memory allocation
- The influence that these page allocation methods have had on virtual memory
- The difference between a first-in first-out page replacement policy, a least-recently-used page replacement policy, and a clock page replacement policy
- The mechanics of paging and how a memory allocation scheme determines which pages should be swapped out of memory
- The concept of the working set and how it is used in memory allocation schemes
- The impact that virtual memory had on multiprogramming
- Cache memory and its role in improving system response time

### Teaching Tips

#### **Paged Memory Allocation**

1. Provide students with an overview of the paged memory allocation scheme, in which each incoming job is divided into pages of equal size. Explain the terms *disk sector* (a section on a disk) and *page frame* (a section in main memory). Point out that this scheme works well when page size, memory block size (page frames), and size of disk section (sector, block) are all equal. Note that these sizes depend upon the specific operating system, and the disk sector size usually determines the exact number of bytes that may be stored in each of them.

2. Outline the three tasks Memory Manager performs before a program is executed. These include: (1) determining the number of pages in the program, (2) locating enough empty page frames in the main memory, and (3) loading all of the program's pages into them. Be sure to mention that in this scheme, pages do not have to be loaded in adjacent memory blocks.
3. Discuss the advantage of storing programs in noncontiguous locations; for example, main memory is used more efficiently and the compaction scheme for relocatable partitions is eliminated.
4. Discuss the new problem that this scheme introduces of having to keep track of each job's pages. Note that this adds to the size and complexity of the operating system and results in greater overhead.
5. Use Figure 3.1 on page 65 in the text to clarify the concept of this scheme. Point out how this scheme results in limiting the internal fragmentation to the last page only. Note that this scheme offers the advantage of noncontiguous storage; however, the requirement that the entire job be stored in memory during execution remains.
6. Outline the three tables that the Memory Manager requires in order to keep track of the job's pages: the Job Table (JT), Page Map Table (PMT), and Memory Map Table (MMT). Discuss the key information in each of the three tables. Inform students that these tables reside in the part of main memory reserved for the operating system. Use Table 3.1 on page 66 in the text to explain and clarify the Job Table (JT).
7. Explain what displacement is and how displacement or offset of a line is calculated. Refer to the long division example on page 68 of the text and describe the long division calculation that takes place to find the address of a given line of code. Use Figure 3.2 on page 67 in the text to clarify the concept of displacement.
8. Outline all the steps that the operating system (or the hardware) has to perform to calculate the address of a given line in the memory, as listed on pages 69 to 70 in the text. Use an example, as given in the text along with Figure 3.3 on page 70, to clarify this.
9. Describe the concept of address resolution or translating the job space address (a relative address) into its physical address (an absolute address). Note that this processing is overhead that takes time away from jobs waiting to be completed.
10. Discuss the advantages and disadvantages of this scheme. Advantages include the allocation of jobs in noncontiguous memory so that more jobs can fit into main memory. Also, note that internal fragmentation still exists, though only in the last page.
11. Discuss the key success factor in this scheme: the size of the page. A page size that is too small will generate very long PMTs, while a page size that is too large will result in excessive internal fragmentation.

**Teaching  
Tip**

Remind the students that in the examples presented throughout this chapter, page numbering and page frame numbering begin with zero – not one! This is true throughout much of computer science.

**Demand Paging**

1. Explain the scheme of demand paging. Point out that it was the first widely used scheme that removed the restriction of having the entire job in memory from the beginning to the end of its processing. Note that this scheme requires high-speed access to the pages. Explain the fact that program code is written sequentially so that while one section is processed, the other sections are idle. Use the examples on pages 71-72 to clarify this concept.
2. Point out that the most important innovation of demand paging was that it made virtual memory widely available. Note that there is an appearance of almost infinite or nonfinite physical memory. Also, note that jobs run with less main memory than is required in the paged memory allocation scheme.
3. Discuss the hardware requirements to implement a demand paging scheme, such as the need of a high-speed direct access storage device. Be sure to mention that this is vital because pages must be passed quickly from secondary storage to main memory and back again. Note that swapping explains how and when pages are exchanged in memory. It depends upon predefined policies, which are discussed later in the chapter.
4. Outline the three tables that Memory Manager requires in order to keep track of the job's pages: the Job Table (JT), Page Map Table (PMT), and Memory Map Table (MMT). Discuss the key information in each of the three tables. Inform students that these tables reside in the part of main memory reserved for the operating system. Explain the three new fields required for each page in the PMT for the demand paging scheme (referenced, modified, and status). Use Figure 3.5 on page 73 in the text to clarify how the demand paging scheme works.
5. Note that to move in a new page, a resident page in main memory must be swapped back into secondary storage. The swapping process exchanges a resident memory page with a secondary storage page. It involves copying the resident page to the disk (if it was modified), and writing the new page into the empty page frame in main memory. The swapping process depends on predefined policies and requires close interaction between hardware components, software algorithms, and policy schemes.
6. Discuss the purpose of the hardware instruction processing algorithm. That is, it must generate the address of the required page, find the page number, and determine whether it is already in memory. Explain the algorithm on page 74 of the text.

7. Discuss the purpose of the page fault handler. Explain the algorithm to find a page on page 75 of the text. Discuss the difference between hardware instruction processing and the page fault handler. Point out that if all page frames are busy, the page fault handler must decide which page to swap out of main memory. (This decision is directly dependent on the predefined policy for page removal). Make sure students clearly understand the process of swapping and the requirements to implement it.
8. Explain the concept of thrashing, which is an excessive amount of page swapping between main memory and secondary storage that results in increased overhead. Refer the students to the example presented in Figure 3.6 on page 76 of the text to clarify the concept of thrashing.
9. Discuss the advantages and disadvantages of the demand paging scheme. An advantage is that a job is no longer constrained by the size of physical memory. A disadvantage includes the increased overhead needed to maintain tables and handle page interrupts.

### **Page Replacement Policies and Concepts**

1. Explain the concept of page replacement policies (the policy that selects the page to be removed). Identify the two most well known page replacement policies, namely, the First-In First-Out (FIFO) policy, and the least recently used (LRU) policy.

#### **First-In First-Out**

1. Describe the First-In First-Out policy. Illustrate the difference between FIFO and LRU by providing an example of a dresser drawer filled with favorite sweaters, as presented on pages 78–79 in the text.
2. Discuss the efficiency (ratio of number of interrupts to the total number of requests) for FIFO. Explain what factors affect efficiency (limited amount of page frames and order pages are requested).
3. Mention Belady's anomaly, which states there is no guarantee that buying more memory will always result in better performance.
4. Use Figure 3.7 on page 77 of the text and Figure 3.8 on page 78 of the text to explain and clarify the FIFO policy.

#### **Least Recently Used**

1. Describe the Least Recently Used policy. Use Figure 3.9 on page 79 of the text to explain and clarify the LRU policy.

2. Discuss the efficiency of the LRU example provided, and compare the efficiency of the LRU example with that of the FIFO example. Explain the LRU stack algorithm removal policy whereby the LRU policy functions in such a way that increasing main memory will cause either a decrease in or the same number of page interrupts. That is to say, that an increase in memory will never cause an increase in the number of page interrupts for LRU.

**Teaching  
Tip**

Ask your students to discuss whether Belady's anomaly applies to the LRU algorithm. Encourage them to explain why or why not.

3. Explain a variation of the LRU technique, known as the clock page replacement policy, which is paced according to the computer's clock cycle. Refer students to Figure 3.10 on page 80 of the text to clarify this technique.
4. Explain a variation of the LRU technique known as the bit-shifting technique that tracks the usage of each page currently in memory. Refer students to Figure 3.11 on page 81 of the text to clarify this technique. Discuss which of the two pages will be swapped out if both have the same values.

## **Quick Quiz 1**

1. Which of the following is the correct way of calculating the address of the page frame?
  - a. Multiply the page frame number by the page frame size
  - b. Divide the page frame size by the page frame number
  - c. Add the page frame number and the page frame size
  - d. Multiply the page frame number by the displacement
 Answer: a

2. Which of the following are the disadvantages of demand paged memory allocation? (choose all that apply)
  - a. Increased overhead caused by tables and page interrupts
  - b. Thrashing
  - c. Job size limited by memory size
  - d. Does not allow multiprogramming
 Answer: a and b

3. \_\_\_\_\_ is a phenomenon in a virtual memory system where an excessive amount of page swapping back and forth between main memory and secondary storage results in higher overhead and little useful work.  
Answer: Thrashing

## The Mechanics of Paging

1. Note that the memory manager requires specific information about each page before it can determine which page to swap out of main memory. The memory manager obtains this information from the PMT where each bit takes on a value of “0” or “1”. Explain, using Table 3.3 on page 82 of the text, the information in a page map table.
2. Discuss the meaning of the different bits used in a PMT using Table 3.4 presented on page 83 of the text.
3. Point out that the FIFO algorithm uses only the modified and status bits when swapping pages, but the LRU looks at all three (status, referenced, and modified) before deciding which pages to swap out. Discuss possible combinations of modified and referenced bits, as presented in Table 3.5 on page 83 of the text. Discuss the order in which the LRU policy would choose to swap out the page of the four cases presented in Table 3.5.

Of the four cases described in Table 3.5, the pages in Case 1 are ideal candidates for removal because they have been neither modified nor referenced. That means they would not need to be rewritten to secondary storage, and they have not been referenced recently. Therefore, the pages with zeros for these two bits would be the first to be swapped out.

### *Teaching Tip*

Ensure students understand this algorithm by walking through the rest of Table 3.5. Which is the next most likely candidate? The LRU policy would choose Case 3 next because the other two, Case 2 and Case 4, were recently referenced. The bad news is that Case 3 pages have been modified, so it will take more time to swap them out. By process of elimination, we can say that Case 2 is the third choice and Case 4 would be the pages least likely to be removed.

## The Working Set

1. Provide students with an overview of the concept of a working set, which is the set of pages residing in memory that can be accessed directly without incurring a page fault. Note that the working set improves the performance in the demand paging scheme.
2. Discuss the importance of structured programming and the concept of locality of reference.
3. Discuss the importance within time sharing systems. Note that it is time-consuming for the CPU to reload a working set as jobs are moved in and out of memory in such a system. Use Figure 3.12 on page 85 of the text to illustrate the time required to process page faults for a single program.
4. Explain how the working set improves the performance of demand page scheme. Note the disadvantage of increased overhead caused by the table maintenance and page interrupts.

## Segmented Memory Allocation

1. Describe the segmented memory allocation scheme, in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Note that it is based upon the concept of programming modules.
2. Discuss the notion of placing modules in different sized segments. Explain how this concept reduces page faults using the loop with a program example. Note that this is fundamentally different from a paging scheme that uses pages of the same size. Also note that memory is allocated dynamically.
3. Discuss the role of the Segment Map Table (SMT) and the information contained in it, such as segment numbers, their lengths, access rights, status, and (when each is loaded into memory) its location in memory.
4. Use Figure 3.13 to Figure 3.15 on pages 86-88 of the text to clarify a segmented memory allocation scheme.

<b>Teaching Tip</b>	Be sure to point out that in this scheme, segments do not need to be stored contiguously. Also, point out that the addressing scheme is two-dimensional, referencing the segment number and the displacement.
---------------------	---

5. Outline the three tables that Memory Manager uses to track segments in memory: the Job Table, Segment Map Table (SMT), and Memory Map Table.
6. Discuss the advantages and disadvantages of this scheme. An advantage is that it removes internal fragmentation. One disadvantage is that it is difficult to manage variable-length segments in secondary storage. A second disadvantage is that this scheme also causes external fragmentation.

## Segmented/Demand Paged Memory Allocation

1. Describe the segmented/demand paged memory allocation scheme. Note its evolution from the demand paging and segment schemes. Describe how it subdivides segments into pages of equal size, smaller than most segments and more easily manipulated than whole segments. Outline the advantages it offers, such as the logical benefits of segmentation and physical benefits of paging. Point out that the addressing scheme is three-dimensional and requires a segment number, page number within that segment, and displacement within that page.

<b>Teaching Tip</b>	Discuss the reasons why many of the problems found in segmentation are removed in this scheme.
---------------------	--

2. Explain briefly the four tables that this scheme requires: the Job Table, Segment Map Table, Page Map Table, and Memory Map Table. Use the example shown in Figure 3.16 on page 90 in the text to demonstrate the interaction of JT, SMT, PMT, and main memory in a segment/demand paged scheme.
3. Discuss the advantages and disadvantages of this scheme. For example, it offers large virtual memory, and segments are loaded on request. The disadvantages include table handling overhead and the memory requirements for page and segment tables.
4. Provide an overview of the concept of associative memory and explain how it minimizes the number of references, which helps in speeding up the process. Note that two searches begin in parallel when searching for a page. Point out that the only disadvantage of associate memory is the high cost of the complex hardware required to perform parallel searches.

### Virtual Memory

1. Explain the concept of virtual memory, which allows programs to be executed even though they are not stored entirely in memory. Explain the requirements for the use of virtual memory, including that it requires cooperation between the Memory Manager and the processor hardware.
2. Outline virtual memory's important advantages such as a job's size is no longer restricted to the size of main memory, it eliminates external fragmentation and minimizes internal fragmentation, it allows the sharing of code and data, it facilitates dynamic linking of program segments, and an unlimited amount of multiprogramming is possible. Use the bullet points on page 94 in the text to explain these advantages further.
3. Point out that the disadvantages of virtual memory include increased processor hardware costs, increased overhead for handling paging interrupts, and increased software complexity to prevent thrashing.
4. Use Table 3.6 on page 93 of the text to explain the difference between virtual memory with paging and that with segmentation.

#### **Teaching Tip**

Refer to the following Web site to learn more about virtual memory in the Windows XP operating system: [www.aumha.org/win5/a/xpvm.php](http://www.aumha.org/win5/a/xpvm.php)

Refer to the following Web site to learn more about virtual memory in the Linux Red Hat operating system: <http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-isa-en-4/s1-memory-concepts.html>

## Cache Memory

1. Explain what cache memory is, why it is needed, and how it helps to speed up the process. Explain how the movement of data, or instructions, from main memory to cache memory takes place.
2. Use Figure 3.17 on page 95 of the text to clarify the concept of cache memory. Be sure to mention that it is often five to 10 times faster than main memory, matching the speed of the CPU. Refer students to Table 3.7 to see a comparison of access speeds amongst the various components.
3. Outline various factors that one should consider when designing cache memory, such as cache size, block size, the block replacement algorithm, and the rewrite policy. Explain the use of the Level 1 (L1) and Level 2 (L2) caches.
4. Explain the ways to measure the efficiency of a system with cache memory, such as the cache hit ratio calculation and the average memory access time calculation.
5. Remind students about paging algorithms, in which instructions from main memory to cache memory takes place in a similar way. Explain the Main Memory Transfer Algorithm as listed on page 97 in the text.

**Teaching  
Tip**

Refer to the following Web site to learn more about cache memory:  
[www.informit.com/articles/article.asp?p=30422&seqNum=5](http://www.informit.com/articles/article.asp?p=30422&seqNum=5)

## Quick Quiz 2

1. Which of the following memory management schemes solved internal fragmentation?
  - a. Paged memory allocation
  - b. Fixed partition
  - c. Segmented memory allocation
  - d. None of the aboveAnswer: c
2. Which of the following concepts is best at preventing page faults?
  - a. Paging
  - b. The working set
  - c. Hit ratios
  - d. Address location resolutionAnswer: b
3. \_\_\_\_\_ is a small high-speed memory unit that a processor can access more rapidly than main memory.  
Answer: Cache memory

## Class Discussion Topics

1. Have students discuss the pros and cons of memory management schemes presented in this chapter. If given an option, which scheme would they like to implement and why?
2. Have students discuss the concepts of associative memory and cache memory. Ask students to make a list of the differences between the two. Ask them to describe proper uses of these two types of memory.

## Additional Projects

1. Have students research how to manage virtual memory in Windows Vista. They can go to the following link to get started: <http://windowshelp.microsoft.com/Windows/en-US/Help/e5b28b00-dea1-4c65-8a01-018cb231b2151033.mspx>
2. Have students work in groups to write pseudocode for the first-in first-out (FIFO) and least-recently-used (LRU) page replacement policies.

## Additional Resources

1. Microsoft.com:  
[www.microsoft.com](http://www.microsoft.com)
2. IBM.com:  
[www.ibm.com](http://www.ibm.com)
3. A History of Operating Systems:  
[www.osdata.com/kind/history.htm](http://www.osdata.com/kind/history.htm)
4. Windows Products and Technologies History:  
[www.microsoft.com/windows/WinHistoryIntro.mspx](http://www.microsoft.com/windows/WinHistoryIntro.mspx)
5. Ravenbrook – The Memory Management Reference:  
[www.memorymanagement.org](http://www.memorymanagement.org)

## Key Terms

- **Address resolution:** the process of changing the address of an instruction or data item to the address in main memory at which it is to be loaded or relocated.
- **Associative memory:** the name given to several registers, allocated to each active process, whose contents associate several of the process segments and page numbers with their main memory addresses.
- **Belady's anomaly:** see FIFO anomaly.

- **Cache memory:** a small, fast memory used to hold selected data and to provide faster access than would otherwise be possible.
- **Clock cycle:** the elapsed time between two ticks of the computer's system clock.
- **Clock page replacement policy:** a variation of the LRU policy that removes from main memory the pages that show the least amount of activity during recent clock cycles.
- **Demand paging:** a memory allocation scheme that loads a program's page into memory at the time it is needed for processing.
- **Displacement:** in a paged or segmented memory allocation environment, the difference between a page's relative address and the actual machine language address. Also called offset.
- **FIFO anomaly:** an unusual circumstance through which adding more page frames causes an increase in page interrupts when using a FIFO page replacement policy. Also known as Belady's anomaly.
- **First-in first-out (FIFO) policy:** a page replacement policy that removes from main memory the pages that were brought in first.
- **Job Table (JT):** a table in main memory that contains two values for each active job: the size of the job and the memory location where its page map table is stored.
- **Least recently used (LRU) policy:** a page-replacement policy that removes from main memory the pages that show the least amount of recent activity.
- **Locality of reference:** behavior observed in many executing programs in which memory locations recently referenced, and those near them, are likely to be referenced in the near future.
- **Memory Map Table (MMT):** a table in main memory that contains as many entries as there are page frames and lists the location and free/busy status for each one.
- **Offset:** see displacement.
- **Page:** a fixed-size section of a user's job that corresponds in size to page frames in main memory.
- **Page fault:** a type of hardware interrupt caused by a reference to a page not residing in memory. The effect is to move a page out of main memory and into secondary storage so another page can be moved into memory.
- **Page fault handler:** the part of the Memory Manager that determines if there are empty page frames in memory so that the requested page can be immediately copied from secondary storage, or determines which page must be swapped out if all page frames are busy. Also known as a page interrupt handler.
- **Page frame:** an individual section of main memory of uniform size into which a single page may be loaded without causing external fragmentation.
- **Page Map Table (PMT):** a table in main memory with the vital information for each page including the page number and its corresponding page frame memory address.
- **Page replacement policy:** an algorithm used by virtual memory systems to decide which page or segment to remove from main memory when a page frame is needed and memory is full.
- **Page swapping:** the process of moving a page out of main memory and into secondary storage so another page can be moved into memory in its place.
- **Paged memory allocation:** a memory allocation scheme based on the concept of dividing a user's job into sections of equal size to allow for noncontiguous program storage during execution.
- **Reentrant code:** code that can be used by two or more processes at the same time; each shares the same copy of the executable code but has separate data areas.

- **Sector:** a division in a disk's track, sometimes called a "block." The tracks are divided into sectors during the formatting process.
- **Segment:** a variable-size section of a user's job that contains a logical grouping of code.
- **Segment Map Table (SMT):** a table in main memory with the vital information for each segment including the segment number and its corresponding memory address.
- **Segmented/demand paged memory allocation:** a memory allocation scheme based on the concept of dividing a user's job into logical groupings of code and loading them into memory as needed to minimize fragmentation.
- **Segmented memory allocation:** a memory allocation scheme based on the concept of dividing a user's job into logical groupings of code to allow for noncontiguous program storage during execution.
- **Subroutine:** also called a "subprogram," a segment of a program that can perform a specific function. Subroutines can reduce programming time when a specific function is required at more than one point in a program.
- **Thrashing:** a phenomenon in a virtual memory system where an excessive amount of page swapping back and forth between main memory and secondary storage results in higher overhead and little useful work.
- **Virtual memory:** a technique that allows programs to be executed even though they are not stored entirely in memory.
- **Working set:** a collection of pages to be kept in main memory for each active process in a virtual memory environment.