in memory, however, user memory accesses can be degraded substantially. A TLB can reduce the performance degradation to an acceptable level.

- **Fragmentation**. A multiprogrammed system will generally perform more efficiently if it has a higher level of multiprogramming. For a given set of processes, we can increase the multiprogramming level only by packing more processes into memory. To accomplish this task, we must reduce memory waste, or fragmentation. Systems with fixed-sized allocation units, such as the single-partition scheme and paging, suffer from internal fragmentation. Systems with variable-sized allocation units, such as the multiple-partition scheme and segmentation, suffer from external fragmentation.

- **Relocation**. One solution to the external-fragmentation problem is compaction. Compaction involves shifting a program in memory in such a way that the program does not notice the change. This consideration requires that logical addresses be relocated dynamically, at execution time. If addresses are relocated only at load time, we cannot compact storage.

- **Swapping**. Swapping can be added to any algorithm. At intervals determined by the operating system, usually dictated by CPU-scheduling policies, processes are copied from main memory to a backing store and later are copied back to main memory. This scheme allows more processes to be run than can be fit into memory at one time.

- **Sharing**. Another means of increasing the multiprogramming level is to share code and data among different users. Sharing generally requires that either paging or segmentation be used to provide small packets of information (pages or segments) that can be shared. Sharing is a means of running many processes with a limited amount of memory, but shared programs and data must be designed carefully.

- **Protection**. If paging or segmentation is provided, different sections of a user program can be declared execute-only, read-only, or read–write. This restriction is necessary with shared code or data and is generally useful in any case to provide simple run-time checks for common programming errors.

Practice Exercises

8.1 Name two differences between logical and physical addresses.

8.2 Consider a system in which a program can be separated into two parts: code and data. The CPU knows whether it wants an instruction (instruction fetch) or data (data fetch or store). Therefore, two base–limit register pairs are provided: one for instructions and one for data. The instruction base–limit register pair is automatically read-only, so programs can be shared among different users. Discuss the advantages and disadvantages of this scheme.

8.3 Why are page sizes always powers of 2?

8.4 Consider a logical address space of 64 pages of 1,024 words each, mapped onto a physical memory of 32 frames.

a. How many bits are there in the logical address?

b. How many bits are there in the physical address?

8.5 What is the effect of allowing two entries in a page table to point to the same page frame in memory? Explain how this effect could be used to decrease the amount of time needed to copy a large amount of memory from one place to another. What effect would updating some byte on the one page have on the other page?

8.6 Describe a mechanism by which one segment could belong to the address space of two different processes.

8.7 Sharing segments among processes without requiring that they have the same segment number is possible in a dynamically linked segmentation system.

a. Define a system that allows static linking and sharing of segments without requiring that the segment numbers be the same.

b. Describe a paging scheme that allows pages to be shared without requiring that the page numbers be the same.

8.8 In the IBM/370, memory protection is provided through the use of *keys*. A key is a 4-bit quantity. Each 2-K block of memory has a key (the storage key) associated with it. The CPU also has a key (the protection key) associated with it. A store operation is allowed only if both keys are equal or if either is zero. Which of the following memory-management schemes could be used successfully with this hardware?

a. Bare machine

b. Single-user system

c. Multiprogramming with a fixed number of processes

d. Multiprogramming with a variable number of processes

e. Paging

f. Segmentation

Exercises

8.9 Explain the difference between internal and external fragmentation.

8.10 Consider the following process for generating binaries. A compiler is used to generate the object code for individual modules, and a linkage editor is used to combine multiple object modules into a single program binary. How does the linkage editor change the binding of instructions and data to memory addresses? What information needs to be passed from the compiler to the linkage editor to facilitate the memory-binding tasks of the linkage editor?

8.11 Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Which algorithm makes the most efficient use of memory?

8.12 Most systems allow a program to allocate more memory to its address space during execution. Allocation of data in the heap segments of programs is an example of such allocated memory. What is required to support dynamic memory allocation in the following schemes?

a. Contiguous memory allocation

b. Pure segmentation

c. Pure paging

8.13 Compare the memory organization schemes of contiguous memory allocation, pure segmentation, and pure paging with respect to the following issues:

a. External fragmentation

b. Internal fragmentation

c. Ability to share code across processes

8.14 On a system with paging, a process cannot access memory that it does not own. Why? How could the operating system allow access to other memory? Why should it or should it not?

8.15 Compare paging with segmentation with respect to the amount of memory required by the address translation structures in order to convert virtual addresses to physical addresses.

8.16 Program binaries in many systems are typically structured as follows. Code is stored starting with a small, fixed virtual address, such as 0. The code segment is followed by the data segment that is used for storing the program variables. When the program starts executing, the stack is allocated at the other end of the virtual address space and is allowed to grow toward lower virtual addresses. What is the significance of this structure for the following schemes?

a. Contiguous memory allocation

b. Pure segmentation

c. Pure paging

8.17 Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

a. 2375

b. 19366

c. 30000

d. 256

e. 16385

8.18 Consider a logical address space of 32 pages with 1,024 words per page, mapped onto a physical memory of 16 frames.

a. How many bits are required in the logical address?

b. How many bits are required in the physical address?

8.19 Consider a computer system with a 32-bit logical address and 4-KB page size. The system supports up to 512 MB of physical memory. How many entries are there in each of the following?

a. A conventional single-level page table

b. An inverted page table

8.20 Consider a paging system with the page table stored in memory.

a. If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

b. If we add TLBs, and 75 percent of all page-table references are found in the TLBs, what is the effective memory reference time? (Assume that finding a page-table entry in the TLBs takes zero time, if the entry is there.)

8.21 Why are segmentation and paging sometimes combined into one scheme?

8.22 Explain why sharing a reentrant module is easier when segmentation is used than when pure paging is used.

8.23 Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a. 0,430

b. 1,10

c. 2,500

d. 3,400

e. 4,112

8.24 What is the purpose of paging the page tables?

8.25 Consider the hierarchical paging scheme used by the VAX architecture. How many memory operations are performed when a user program executes a memory-load operation?

8.26 Compare the segmented paging scheme with the hashed page table scheme for handling large address spaces. Under what circumstances is one scheme preferable to the other?

8.27 Consider the Intel address-translation scheme shown in Figure 8.22.

a. Describe all the steps taken by the Intel Pentium in translating a logical address into a physical address.

b. What are the advantages to the operating system of hardware that provides such complicated memory translation?

c. Are there any disadvantages to this address-translation system? If so, what are they? If not, why is this scheme not used by every manufacturer?

Programming Problems

8.28 Assume that a system has a 32-bit virtual address with a 4-KB page size. Write a C program that is passed a virtual address (in decimal) on the command line and have it output the page number and offset for the given address. As an example, your program would run as follows:

```
./a.out 19986
```

Your program would output:

```
The address 19986 contains:
page number = 4
offset = 3602
```

Writing this program will require using the appropriate data type to store 32 bits. We encourage you to use `unsigned` data types as well.

Wiley Plus

Visit Wiley Plus for

- Source code
- Solutions to practice exercises
- Additional programming problems and exercises
- Labs using an operating system simulator

Bibliographical Notes

Dynamic storage allocation was discussed by Knuth [1973] (Section 2.5), who found through simulation results that first fit is generally superior to best fit. Knuth [1973] also discussed the 50-percent rule.

The concept of paging can be credited to the designers of the Atlas system, which has been described by Kilburn et al. [1961] and by Howarth et al.

[1961]. The concept of segmentation was first discussed by Dennis [1965]. Paged segmentation was first supported in the GE 645, on which MULTICS was originally implemented (Organick [1972] and Daley and Dennis [1967]).

Inverted page tables are discussed in an article about the IBM RT storage manager by Chang and Mergen [1988].

Address translation in software is covered in Jacob and Mudge [1997].

Hennessy and Patterson [2002] explains the hardware aspects of TLBs, caches, and MMUs. Talluri et al. [1995] discusses page tables for 64-bit address spaces. Alternative approaches to enforcing memory protection are proposed and studied in Wahbe et al. [1993a], Chase et al. [1994], Bershad et al. [1995], and Thorn [1997]. Dougan et al. [1999] and Jacob and Mudge [2001] discuss techniques for managing the TLB. Fang et al. [2001] evaluate support for large pages.

Tanenbaum [2001] discusses Intel 80386 paging. Memory management for several architectures—such as the Pentium II, PowerPC, and UltraSPARC—are described by Jacob and Mudge [1998a]. Segmentation on Linux systems is presented in Bovet and Cesati [2002].