Lecture

- In the lectures in week 14 we will mainly discuss Chapter 8 (Main Memory Management) and Chapter 9 (Virtual Memory).

Exercises

Note, as usual, that you find even more exercises including solutions here :
`http://codex.cs.yale.edu/avi/os-book/OS9/practice-exer-dir/index.html`

Prepare for the Tutorial Sessions in week 15, 2018 all exercises that have not been discussed so far as wall as the following:

8.1 Explain the difference between internal and external fragmentation.

8.2 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?

8.3 Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of 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?

8.4 Most systems allow programs to allocate more memory to its address space during execution. Data allocated in the heap segments of programs are 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

8.5 Compare the main 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

8.6 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?

8.7 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.

8.8 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 towards lower virtual addresses. What is the significance of the above structure for the following schemes:

a. contiguous-memory allocation

b. pure segmentation

c. pure paging

8.9 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

8.10 Consider a logical address space of 32 pages with 1024 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?

8.11 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?

8.12 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.)

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

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

8.15 Consider the following segment table:

segment	base	length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1352	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

8.16 What is the purpose of paging the page tables?

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

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

8.19 Consider the Intel address-translation scheme shown in Figure 8.22 of the course book.

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?