UNIVERSITY OF CALIFORNIA
College of Engineering
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

CS 162 Prof. Alan J. Smith

Problem Set 2

Please place the work for this assignment online, and submit in the usual manner.

1. Suppose a system has a virtual address containing 28 bits. Suppose that the page size is 1024 bytes, that physical addresses contain 24 bits, and that the system uses a segmented-and-paged scheme with 256 segments and a segment size of one megabyte.

(a) Describe the address translation mechanism, using a picture to show the relevant tables and operations. Don't worry about a TLB. Be sure to show the exact size of each table (how many entries, how many bits wide).

(b) Suppose that a byte of physical memory is to be shared by two processes. Describe how the tables could be arranged to permit this.

(c) Suppose that a particular segment in this system contains 4800 bytes. How much memory will the segment cause to be wasted in internal and external fragmentation? Consider only the memory allocated to the segment itself; do not worry about memory used in the mapping tables.

(d) Now suppose that the owner of the 4800-byte segment wishes to expand it by 200 bytes. What must the operating system do in order to increase its size? Compare this to a system based on pure segmentation.

(e) Suppose that the segment is expanded again, from 5000 bytes to 5500 bytes. What must the operating system do to accomplish this?

2. Consider a program that generates the following set of page references:

A A A A B B C A C B B D D A E B C E A E D E

Assuming that the program is given 3 page frames, how many page faults are generated using the following replacement algorithms:

(a) FIFO

(b) MIN

(c) LRU

(d) Working Set. Tau = 6 (if six references have been made without accessing a page then it is no longer in the working set). For this case, the program may need to use more than 3 page frames. Indicate how large the working set is at each point in time. Assume that pages are thrown out of memory as soon as they leave the working set.

3. Consider the clock algorithm for LRU page replacement (also called "FINUFO" (first in, not used, first out), also called "first chance" replacement). Suppose that there are P pages of physical memory in the system and that over a particular interval of time F page faults have occurred. What are the minimum and

maximum number of times that the clock hand could possibly have been advanced during the time interval? Give your answer in terms of P and F.

4. Consider a demand paging system. Pages that are not in main memory are stored on the "paging device", which may be a portion of a hard disk, an electronic disk, or something else. The size of the paging device is its capacity. Measured utilizations (in terms of time, not space) are:

center; l r. CPU utilization 20% Paging device 99.7% Other I/O devices 5%

Which of the following, if any, will probably improve the CPU utilization? Why or why not?

(a) Get a faster CPU.

(b) Get a bigger paging device.

(c) Increase the degree of multiprogramming.

(d) Decrease the degree of multiprogramming.

(e) Get faster other I/O devices.

5. This question concerns translation lookaside buffers (TLBs).

(a) In almost all computers with translation lookaside buffers, the TLB must be flushed (all the entries must be invalidated) during each context switch. Why?

(b) Are large TLBs better than small ones from a performance standpoint?

(c) A new computer has 32-bit virtual addresses, but uses 30-bit values as the input to the TLB. The high-order 8 bits are a process identifier that is unique for each process, and the low-order 22 bits are the virtual page being referenced (the high-order 22 bits of the virtual address; pages are 1024 bytes long). What is the purpose of inputting the process identifier to the TLB?

(d) Does this use of process IDs simplify or complicate the translation lookaside buffer?

6. In this problem, all numbers are given in decimal. Suppose three object files, A, B, and C, are to be linked together. Each of the object files contains two segments, code and data, and the output of the linker will also contain two segments. A contains 1400 bytes of code and 600 bytes of data. B contains 420 bytes of code and 20 bytes of data. C contains 2600 bytes of code and 112 bytes of data. The file A defines one external symbol, X, which is at location 760 in A. One external symbol is defined in B: its name is Y and its location is 430. C defines 2 external symbols: Z is at 2216 and W is at 2704. Suppose the linker processes the files in the order A, B, C, so that A's segments end up before B's which end up before C's.

(a) Suppose that file A has an address at location 134 that is supposed eventually to refer to symbol X. What will be the contents of location 134 in A? What will its contents be after linking, and where will the contents be in the output file?

(b) Suppose that file A has an address at location 136 that is supposed eventually to refer to symbol Y. What will be the contents of location 136 in A? What will its contents be after linking, and where will the contents be in the output file?

(c) Suppose that file B has an address at location 430 that is supposed eventually to refer to symbol Z. What will be the contents of location 430 in B? What will its contents be after linking, and where will the contents be in the output file?