CSCI 3323 (Principles of Operating Systems), Fall 2020

Homework 4a

Credit: 50 points.

1 Reading

Be sure you have read, or at least skimmed, Chapter 3.

2 Problems

Answer the following questions. You may write out your answers by hand and scan them, or you may use a word processor or other program, but please submit a PDF or plain text via e-mail to my TMail address. (No links to shared files on Google Drive please.) Please use a subject line that mentions the course and the assignment (e.g., "csci 3323 hw 4a" or "O/S hw 4a").
(*Note:* In all of the following I assume that addresses refer to bytes, as opposed to words or some other unit. As far as I know, byte addressability is so much the norm these days that it almost goes without saying.)

1. (5 points) Consider a computer system with 10,000 bytes of memory whose MMU uses the simple base register / limit register scheme described in section 3.2 of the textbook, and suppose memory is currently allocated as follows:

   - Locations 0–1999 are reserved for use by the operating system.
   - Process A occupies locations 5000–6999.
   - Process B occupies locations 7000–8999.
   - Other locations are free.

   Answer the following questions about this system.

   (a) What value would need to be loaded into the base register if we performed a context switch to restart process A?

   (b) What memory locations would correspond to the following virtual (program) addresses in process A?

   - 100
   - 4000

2. (10 points) Consider a computer system using paging to manage memory; suppose it has 64K (2^{16}) bytes of memory and a page size of 4K bytes, and suppose the page table for some process (call it process A) looks like the following.

Page number	Present/absent bit	Page frame number
0	1	5
1	1	6
2	1	2
3	0	?
4	0	?
5	1	7
6	0	?
...	0	?
15	0	?

Answer the following questions about this system.

(a) How many bits are required to represent a physical address (memory location) on this system? If each process has a maximum address space of 64K bytes, how many bits are required to represent a virtual (program) address?

(b) What memory locations would correspond to the following virtual (program) addresses for process *A*? (Here, the addresses will be given in hexadecimal, i.e., base 16, to make the needed calculations simpler. Your answers should also be in hexadecimal. Note that if you find yourself converting between decimal and hexadecimal, *you are doing the problem the hard way.* Stop and think whether there is an easier way!)

- 0x1420
- 0x2ff0
- 0x4008
- 0x0010

3. (15 points) Now consider a bigger computer system, one in which addresses (both physical and virtual) are 32 bits and the system has 2^{32} bytes of memory. Answer the following questions about this system. (You can express your answers in terms of powers of 2, if that is convenient.)

(a) What is the maximum size in bytes of a process's address space on this system?

(b) Is there a logical limit to how much main memory this system can make use of? That is, could we buy and install as much more memory as we like, assuming no hardware constraints? (Assume that the sizes of physical and virtual addresses don't change.)

(c) If page size is 4K (2^{12}) and each page table entry consists of a page frame number and four additional bits (present/absent, referenced, modified, and read-only), how much space is required for each process's page table? (You should express the size of each page table entry in bytes, not bits, assuming 8 bits per byte and rounding up if necessary.)

(d) Suppose instead the system uses a single inverted page table (as described in section 3.3.4 of the textbook), in which each entry consists of a page number, a process ID, and four additional bits (free/in-use, referenced, modified, and read-only), and at most 64 processes are allowed. (Page size is the same as in the previous problem.) How much space is needed for this inverted page table? (You should express the size of each page table entry in bytes, not bits, assuming 8 bits per byte and rounding up if necessary.) How does this compare to the amount of space needed for 64 regular page tables?

4. (10 points) How long it takes to access all elements of a large data structure can depend on whether they're accessed in contiguous order (i.e., one after another in the order in which they're stored in memory), or in some other order. The classic example is a 2D array, in which performance of nested loops such as

```
for (int r = 0; r < ROWS; ++r)
  for (int c = 0; c < COLS; ++c)
    array[r][c] = foo(r,c);
```

   can change drastically for a large array if the order of the loops is reversed. Give two explanations for this phenomenon. (*Hint:* The likeliest explanation these days involves the memory hierarchy as discussed many weeks ago (registers, caches, RAM, etc. — 9/09 lecture). Another explanation, more likely when computers had less memory but still possible, involves something from the current chapter on memory management.)

5. (10 points) A computer at Acme Company used as a compute server (i.e., to run non-interactive jobs) is observed to be running slowly (turnaround times longer than expected). The system uses demand paging, and there is a separate disk used exclusively for paging. The sysadmins are puzzled by the poor performance, so they decide to monitor the system. It is discovered that the CPU is in use about 20% of the time, the paging disk is in use about 98% of the time, and other disks are in use about 5% of the time. They are particularly puzzled by the CPU utilization (percentage of time the CPU is in use), since they believe most of the jobs are compute-bound (i.e., much more computation than I/O). First give your best explanation of why CPU utilization is so low, and then for each of the following, say whether it would be likely to increase it and why.

   (a) Installing a faster CPU.

   (b) Installing a larger paging disk.

   (c) Increasing the number of processes ("degree of multiprogramming").

   (d) Decreasing the number of processes ("degree of multiprogramming").

   (e) Installing more main memory.

   (f) Installing a faster paging disk.

3 Pledge

Include the Honor Code pledge or just the word "pledged", plus *at least one of the following* about collaboration and help (as many as apply).[1] Text *in italics* is explanatory or something for you to fill in. For programming assignments, this should go in the body of the e-mail or in a plain-text file `pledge.txt` (no word-processor files please).

- This assignment is entirely my own work. *(Here, "entirely my own work" means that it's your own work except for anything you got from the assignment itself — some programming assignments include "starter code", for example — or from the course Web site. In particular, for programming assignments you can copy freely from anything on the "sample programs page".)*

[1] Credit where credit is due: I based the wording of this list on a posting to a SIGCSE mailing list. SIGCSE is the ACM's Special Interest Group on CS Education.

- I worked with *names of other students* on this assignment.
- I got help with this assignment from *source of help — ACM tutoring, another student in the course, the instructor, etc. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

4 Essay

Include a brief essay (a sentence or two is fine, though you can write as much as you like) telling me what if anything you think you learned from the assignment, and what if anything you found found interesting, difficult, or otherwise noteworthy. For programming assignments, it should go in the body of the e-mail or in a plain-text file `essay.txt` (no word-processor files please).