CSCI 3323 (Principles of Operating Systems), Fall 2021

Homework 3a

Credit: 40 points.

1 Reading

Be sure you have read, or at least skimmed, Chapters 12 through 24 of the textbook.

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 turn in a PDF or plain text file. (No links to shared files on Google Drive please, and no word-processor files.) Turn it in by putting it in your course "TurnIn" folder on Google Drive. Please be sure to include your name somewhere in the file, so when I print it for grading I know whose work it is. (In the pledge is fine.)
(*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. Consider a computer system with 8K bytes of memory whose MMU uses the simple base/bounds scheme described in Chapter 15 of the textbook, and suppose memory is currently allocated as follows (all numbers are base 16):

   - Locations `0--0x1fff` are reserved for use by the operating system.
   - Process A occupies locations `0x2000--0x3fff`.
   - Process B occupies locations `0x6000--0x7fff`.
   - Other locations are free.

   Answer the following questions about this system.

   (a) (2.5 points)
   What value would need to be loaded into the base register if we performed a context switch to restart process A?

   (b) (2.5 points)
   What memory locations would correspond to the following virtual addresses in process A? (All addresses are base-16.)

   - `0x0100`
   - `0x5000`

2. 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) (2.5 points)
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 address?

(b) (5 points)
What memory locations would correspond to the following virtual addresses for process *A*? (Addresses are given in base-16, and your answers should be too.)

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

3. 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) (2.5 points)
What is the maximum size in bytes of a process's address space on this system?

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.)

(b) (5 points)
How much space is required for each process's page table, if:

- Page size is 4K (2^{12}).
- Each page table entry consists of a page frame number and four additional bits (present/absent, referenced, modified, and read-only).

(You should express the size of each page table entry in bytes, not bits, assuming 8 bits per byte and rounding up if necessary.)

(c) (5 points)
Suppose instead the system uses a single inverted page table (as described briefly in Chapter 20 of the textbook), in which:

- Page size is the same as in the previous problem.

- Each entry consists of a page number, a process ID, and four additional bits (free/in-use, referenced, modified, and read-only).
- At most 64 processes are allowed.

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. (2.5 points)
Page sizes all seem to be powers of 2. Is that unavoidable, or is it possible (not practical, maybe, but possible) to build hardware with a page size expressed in some round power of 10 that might appeal more to humans? Why is it not done? *Hint:* Consider what the bit-fiddling involved in address translation is actually accomplishing.

5. (5 points)
How long it takes to access all elements of a large data structure can depend on the order in which they're accessed. 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 possible explanations for this phenomenon, based on this section of the textbook. (If you're up on hardware you may be able to think of a third possible explanation as well; mention it for extra credit.)

6. (7.5 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 swap space. 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 swap 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 swap 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 swap disk.

3 Essay and pledge

Include with your assignment the following information.

For programming assignments, please put it a separate file. (I strongly prefer plain text, but if you insist you can put it in a PDF — just no word-processor documents or Google Drive links please.) For written assignments, please put it in your main document.

3.1 Pledge

This should 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). Text *in italics* is explanatory or something for you to fill in; you don't need to repeat it!

- I did not get outside help *aside from course materials, including starter code, readings, sample programs, the instructor.*
- 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, 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.)*

3.2 Essay

This should be 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 interesting, difficult, or otherwise noteworthy.