EECE.4810/EECE.5730: Operating Systems

Spring 2019

Exam 3
May 11, 2019

Name: ____

Section: **EECE.4810 (undergraduate)** **EECE.5730 (graduate)**

For this exam, you may use one 8.5" x 11" double-sided page of notes. All electronic devices (e.g., cell phones, calculators, laptops) are prohibited. If you have a cell phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 4 sections for a total of 100 points. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

Please note that students enrolled in EECE.5730 must complete two extra problems, which are worth a total of 15 points:

- Question 2d, on page 8
- Question 3d, on page 10

You will have three hours to complete this exam.

S1: General memory management	/ 38
S2: Paging	/ 42 + 9
S3: File systems	/ 20 + 6
TOTAL SCORE	/ 100 + 15

1. (38 points) ***General memory management***

a. (20 points) Write the function with the prototype below, which could be used in base & bounds or segmented address translation to return the starting address of the hole that best fits a memory request of size `reqSize`:

```
unsigned bestFit(int reqSize);
```

The function uses this structure for each node in the free space (hole) list:

```
typedef struct FSN {
  unsigned addr;         // Starting address of hole
  int size;              // Hole size
  struct FSN *next;      // Address of next node in list
} FreeSpaceNode;
```

Assume you have a global pointer (`FreeSpaceNode *first`) to the first node in the free space list. Since the pointer is a global variable, it does not need to be passed to the function. Assume there is at least one node in the free space list.

This function should use best fit allocation to choose a hole in the free space list, update the node representing that hole to reflect the amount of space used, and return the address of the hole. For example, if you request 20 bytes from a 100-byte hole that starts at address 0x1000, the hole's new starting address is 0x1014 (0x14 = 20) and its size is 80 bytes; the function returns 0x1000.

Your function should also account for the special case that the requested space completely fills the hole, in which case that node should be removed from the list, not modified. Assume you have a function with the prototype `remove(FreeSpaceNode *n)` to remove a node from the list. So, for example, if `ptr` points to a node to remove, calling `remove(ptr);` modifies the free space list to remove that node. You only need to recognize the conditions under which you call `remove()` and call it appropriately—don't worry about the details of that function.

The comments in the function outline how it should behave. Space to solve this problem is on the next page.

1a (continued) The FreeSpaceNode structure definition is shown again for your reference:

```
typedef struct FSN {
  unsigned addr;          // Starting address of hole
  int size;               // Hole size
  struct FSN *next;       // Address of next node in list
} FreeSpaceNode;

unsigned bestFit(int reqSize) {
  // Variable declarations/initialization






  // Traverse free space list and find hole with best fit









  // If no fit found, return 0



  // Otherwise, update node accordingly







  // One special case—if hole is completely filled, remove node
  //  (but make sure you've saved your return address first!)





  // Return address of hole


}
```

1 (continued)

b. (10 points) For each of the three types of address translation we discussed (base & bounds, segmentation, paging), explain what type(s) of fragmentation (internal or external) could occur in each and explain how.

c. (4 points) Describe one benefit of segmentation over base and bounds as an address translation scheme.

d. (4 points) Describe one benefit of segmentation over paging as an address translation scheme.

2. (42 **+ 9** points) ***Paging***

a. (18 points) Explain what conditions would create the fastest possible translation for each of the page table organizations we discussed. (For example, if a tree-based page table existed, the fastest possible translation would happen if the necessary translation was at the root of the tree and could therefore be found in a constant amount of time.)

You should not give a numeric answer, but may provide an example to explain your answer.

i. Two-level page table

ii. Hashed page table using chaining

iii. Inverted page table

2 (continued)

b. (12 points) This problem involves a process using the page table below:

Virtual page #	Valid bit	Reference bit	Dirty bit	Frame #
0	1	1	1	3
1	0	0	0	--
2	1	0	1	4
3	0	0	0	--
4	1	1	0	0
5	1	1	0	1

Assume the system uses 16-bit addresses and 8 KB pages. The process accesses four addresses: 0x1234, 0x4755, 0x6A13, and 0xB0A9.

Determine (i) which address would cause a page to be evicted if there were no free physical frames, (ii) which one would mark a previously clean page as modified, if the access were a write, and (iii) which one accesses a page that has not been referenced for many cycles. **For full credit, show all work.**

2 (continued)

c. (12 points) Say the currently running process has 16 active pages, P0-P15. P1, P3, P5, and P7 all have their reference bits set to 0, while all other pages have their reference bits set to 1. If the operating system uses the clock algorithm for page replacement, the pages are ordered numerically around the "clock" (P0 is first, P1 is second, etc.), and the "clock hand" currently points to P2, how many page faults will occur before P4 is replaced, assuming none of the currently active pages are referenced before P4 is replaced? **Explain your answer for full credit.**

2 (continued)

d. (9 points, ***EECE.5730 only***) Assume you have a system using 48-bit virtual addresses, 32-bit physical addresses, 4 KB pages, and a multilevel page table. Assume a page table entry in each lowest level table (in a 2-level page table, for example, the second level would be the lowest level) uses 4 bytes, and each entry in the upper level table(s) contains a single physical address.

If you want each table (1st level table, each 2nd level table, and so on) to be the same size, and that size should be small enough to fit inside a single physical frame, what is the minimum number of levels the multilevel table must have? **Show all of your work for full credit.**

3. (20 <u>+ **6**</u> points) ***File systems***

a. (8 points) Consider two of the file systems we discussed, FFS and NTFS. Describe a case in which the NTFS index scheme would allow you to access file data more quickly than FFS would for a comparably sized file, and describe a case in which FFS would allow you to access file data more quickly for a comparably sized file.

b. (6 points) We discussed two schemes for free space management in file systems, bitmaps and linked lists. Explain one benefit of each of these schemes.

3 (continued)

c. (6 points) Explain what a transaction is, and also explain how the two main methods we discussed for implementing transactions provide reliability in a file system.

d. (6 points, ***EECE.5730 only***) Explain how, in some cases, one volume can span multiple disks, and how, in other cases, one disk can contain multiple volumes.