# CSSE 332 – Operating Systems
Rose-Hulman Institute of Technology
Computer Science and Software Engineering Department

## Exam 2 — Paper Part

Name: Solution Section:_____ CM:_____

**Instructions**: The paper part of the exam is closed book, open notes limited to one double-sided sheet of hand written notes, but **no computer or electronic devices** other than a calculator. Write all of your answers in the spaces provided.

You have one period to complete this exam.

When you complete the paper part of the exam, turn it in to an exam proctor.

Please begin by putting your name on the first page and your initials on every page of the exam. We encourage you to skim the entire exam before answering any questions and show all your work to receive partial credit.

| Page | Points available | Your Points |
|---|---|---|
| 1 | 8 | |
| 2 | 10 | |
| 3 | 6 | |
| 4 | 8 | |
| 5 | 4 | |
| 6 | 4 | |
| Total | 40 | |

**Problem 1** (8 points) Suppose the UNIX file system is modified so that each i-node contains 10 direct block pointers, 8 single-indirect pointers, 4 double-indirect pointers. Assume that each 32-bit pointer identifies an 8KB ($2^{13}$ bytes) block containing file data or addresses.

Use the information above to answer the questions below for the modified system. Show your work in the space provided to receive partial credit.

Use powers of 2 in your answer to the extent that you can.

(a) How many block pointers can fit in a block?

$$\frac{8KB}{32b} = \frac{2^{13}B}{4B} = \frac{2^{13}}{2^2} = 2^{11}$$

$2^{11}$ pointers

(b) How many data blocks can be accessed using only single-indirect pointers?

$$8 * \frac{8KB}{32b} = 2^3 \cdot 2^{11} = 2^{14}$$

$2^{14}$ data blocks

(c) How many data blocks can be accessed using only double-indirect pointers?

$$4 * \left(\frac{8KB}{32b}\right)^2 = 2^2 \cdot 2^{22} = 2^{24}$$

$2^{24}$ data blocks

(d) What is the maximum file size in blocks supported by this modified system?

$(10 + 2^{14} + 2^{24}) \cdot$ ~~8KB~~

data blocks

**Problem 2** (10 points) Consider a simple paging virtual memory system with the following parameters.

*2 GB (i.e., $2^{31}$ Bytes) of physical memory,*
*page size of 4 KB (i.e., $2^{12}$ Bytes),*
*and $2^{24}$ pages of logical address space.*

(a) (2 points) How many bits encode a logical address?

~~24+1~~

$\log_2$(# of pages) + $\log_2$(size of a page)

24 + 12 = 36

36 Bits

(b) (2 points) How many bytes are in a frame?

Same as page → 4K

4K Bytes

(c) (2 points) How many bits of a physical address specify the frame number?

$\log_2(\#\text{ of frames}) = \log_2\left(\frac{2GB}{4KB}\right) = \log_2\left(\frac{2^{31}B}{2^{12}B}\right) = 19$

19 Bits

(d) (2 points) What is the maximum number of entries in the page table?

Same as the number of pages

$2^{24}$

$2^{24}$ Entries

(e) (2 points) Assuming each page table entry includes a valid/invalid bit, how many bits are in each page table entry?

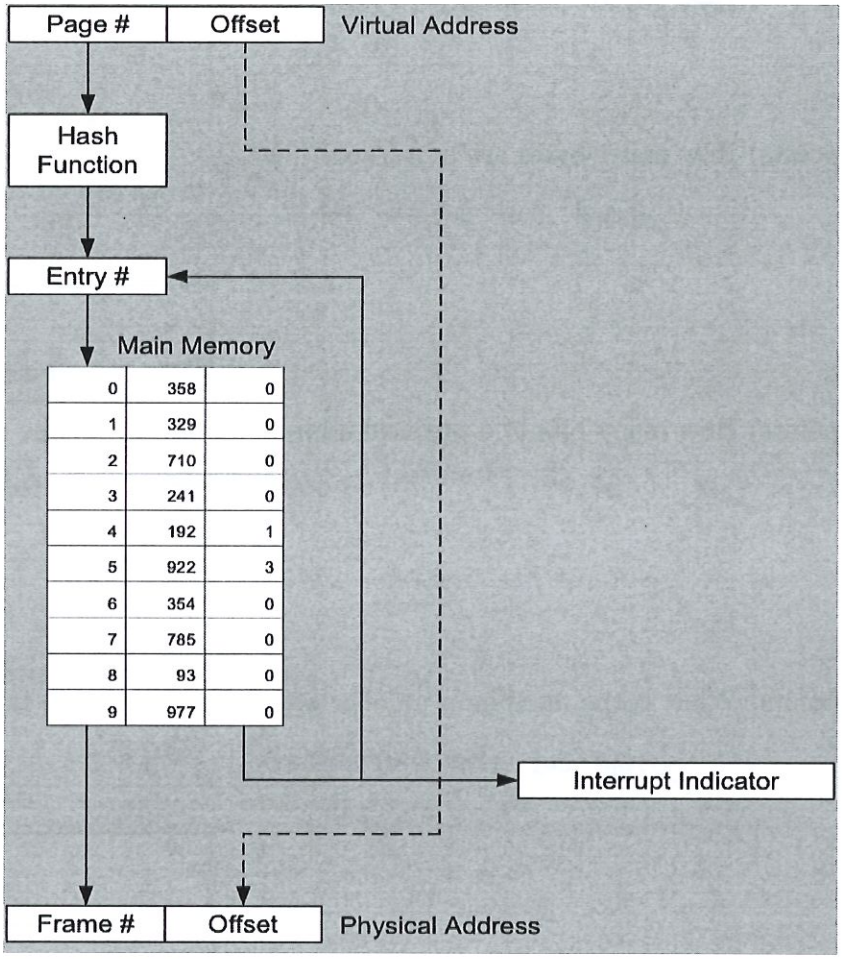
valid bit (1) + frame number (19)

20 Bits

**Problem 3** (6 points) Use the inverted page table below and hash function

$$h(x) = (2 + x)\%10$$

to determine the result of the address translation for each of the following virtual addresses from a process. Note that the presence of a page in memory is no guarantee that such page belongs to the process. Only pages in frames given by the use of the inverted page table and given hash function belong to the process. Each page table entry contains a page table index, a page number, and chain pointer value. If a page fault occurs say so, and do not complete that address translation.




| | | |
|---|---|---|
| 0 | 358 | 0 |
| 1 | 329 | 0 |
| 2 | 710 | 0 |
| 3 | 241 | 0 |
| 4 | 192 | 1 |
| 5 | 922 | 3 |
| 6 | 354 | 0 |
| 7 | 785 | 0 |
| 8 | 93 | 0 |
| 9 | 977 | 0 |

(a) (2 points) 358 329

0/329

(b) (2 points) 922 358

Page Fault

5/358

(c) (2 points) 93 192

8 | 192

**Problem 4** (8 points) Provide short, concise answers to the following questions about your Bare Metal Operating System Project.

(a) In milestone 5, you occasionally need to call the function setKernelDataSegment. What problem does this solve and how does it solve it?

The process table and other global varibles ~~eee~~ in kernel are stored in the data segment of kernel. In order to use them, we need to set the register to point to the kernel's data segment.

(b) When a new file is created in your OS, a free sector must be found to contain it. How can you tell if a particular sector is free?

Check the bitmap in the first sector. If 0xFF, then the corresponding sector is occupied.

(c) As part of your team's Makefile or script, you likely have a command like this:
`dd if=kernel of=floppya.img bs=512 conv=notrunc seek=3`
What does this line do?

Put "kernel" program on the third sector of the disk.

(d) How does your OS preempt the currently running process? If your OS is non–preemptive, just write "the OS was non–preemptive".

Using timer interrupt in Milestone 5.

**Problem 5** (4 points) For each of the following disk scheduling policies and the set of pending requests for disk tracks, determine the sequence of tracks accessed and the number of tracks traversed. The requests are arranged in the order of arrival. There are 200 tracks (ranging from 0 to 199) and the disk head is currently at track 100 and moving toward the outside of the disk (i.e. track 199).

2, 171, 20, 68, 189, 150

Fill in the columns of each table appropriately.

(a) SCAN

| Next track accessed | Tracks traversed |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| Total | |

(b) CIRCULAR SCAN

| Next track accessed | Tracks traversed |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| Total | |

No Longer Needed.

**Problem 6** (4 points) Consider the least recently used (LRU) page replacement policy discussed in class. Use the following page reference stream in a system with 4 frames to complete the table below. Determine the number of page faults in each case. Assume that a page fault occurs whenever a requested page is not in a memory frame.

Page reference stream: 1, 2, 3, 0, 3, 2, 4, 0, 3, 1, 0, 7, 1

| LRU | Page Reference | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 0 | 3 | 2 | 4 | 0 | 3 | 1 | 0 | 7 | 1 |
| Frame 1 | 1 | 1 | 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 | 7 | 7 |
| Frame 2 | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| Frame 3 | | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| Frame 4 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Page Fault? | F | F | F | F | | | F | | | F | | F | |