University of Waterloo
Midterm Examination #2 Model Solution

Spring, 2009

1. (11 total marks)

Consider a virtual memory system that uses multi-level paging for address translation. Virtual addresses and physical addresses are 64 bits long. The page size is 1 MB (2^{20} bytes). The size of a page table entry is 16 (2^4) bytes. Each individual page table, at each level, must fit in a single frame.

a. (2 marks) How many bits of each virtual address are needed to represent the page offset?

> 20 bits (because the page size is 2^{20} bytes).

b.(2 marks) What is the maximum number of entries in an individual page table?

$$\frac{2^{20}}{2^4} = 2^{16}\text{entries}$$

c. (4 marks) What is the minimum number of levels of page tables that will be required for virtual-to-physical translation in this system? Show your work for possible partial credit.

> 20 of 64 bits are used for the offset, leaving 44 bits for page numbers. Each level of page tables will require a 16 bit page number (because 2^{16} is the maximum size of each page table). Thus, a total of 3 levels of page tables will be required.

d. (3 marks) Suppose that a particular process uses only 128 MB (2^{27} bytes) of virtual memory, with a virtual address range from 0 to $2^{27} - 1$. How many individual page tables, *at each level*, will be required to translate this process' address space? Your answer should be of this form:

```
Level 1:  X page tables
Level 2:  Y page tables
...
Level N:  Z page tables
```

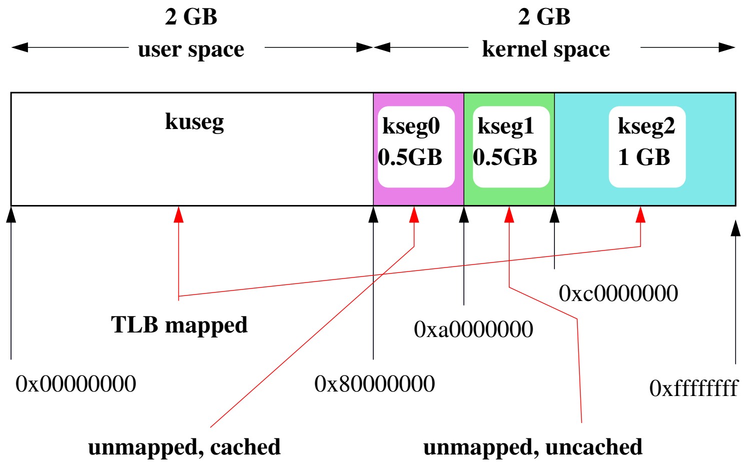
Show your work for possible partial credit.

> The address space for this process has only $2^{27}/2^{20} = 2^7$ pages. A single level 3 page table can have up to 2^{16} entries, so only a single page table will be needed at level 3. Only a single page table entry will be needed at levels 1 and 2 to index the level 3 page table.

```
Level 1: 1 page table
Level 2: 1 page table
Level 3: 1 page table
```

2. (12 marks)

The following diagram, from the lecture notes, illustrates the virtual address space of a MIPS process.



Suppose that the TLB in the MIPS MMU contains the following entries. For each entry, only the virtual page number and physical frame number are shown. Assume that all of the entries are valid.

entry number	virtual page number	frame number
0	`0x10000`	`0x08343`
1	`0x10004`	`0x08344`
2	`0x00400`	`0x0100a`
3	`0x7ffff`	`0x01112`
4	`0x7fffe`	`0x01113`
5	`0x00402`	`0x0600b`
6	`0x00404`	`0x01114`

For each of the following *physical* addresses, indicate which virtual address (or addresses) will translate to that physical address. If there is more than one virtual address that translates to the given physical addresses, you should list all of the virtual addresses. If there are no virtual addresses that translate to the given physical address, you should write "No Virtual Addresses".

a. (3 marks) `0x01113fa0`

`0x7fffefa0`, `0x81113fa0`, `0xa1113fa0`

b. (3 marks) `0x0100a088`

`0x00400088`, `0x8100a088`, `0xa100a088`

c. (3 marks) `0x0100b014`

`0x8100b014`, `0xa100b014`

d. (3 marks) `0x08343ffc`

`0x10000ffc`, `0x88343ffc`, `0xa8343ffc`

3. (12 total marks)

Consider a small system in which there are two processes, P_A and P_B. Virtual and physical addresses in this system are only 16 bits long. The page size is 256 (2^8) bytes. The diagram below shows the the current page tables for the two processes. In addition, the diagram also shows the kernel's current *core map*, which lists the contents of each physical frame.

Page Table for Process P_A

Page Number	Frame Number	Valid	Use	Dirty
0	1	1	1	0
1	3	0	0	0
2	6	0	0	0
3	4	1	0	1
4	5	1	1	0
5	7	0	0	0

Core Map

Frame Number	Process ID	Page Number	
0	P_B	2	
1	P_A	0	← clock pointer
2	P_B	0	
3	P_B	3	
4	P_A	3	
5	P_A	4	
6	P_B	4	
7	P_B	1	

Page Table for Process P_B

Page Number	Frame Number	Valid	Use	Dirty
0	2	1	1	0
1	7	1	0	0
2	0	1	0	0
3	3	1	1	0
4	6	1	0	0
5	1	0	0	0
6	0	0	0	0

Suppose that the kernel uses a global clock replacement algorithm. The current position of the clock algorithm's victim pointer is shown in the core map. The pointer moves down (towards larger frame numbers).

Suppose that P_A is the currently running process. The value of the program counter is `0x0144`, and the instruction at that address is a "store" instruction which will write the contents of a register to memory at virtual address `0x057c`.

a. (4 marks)
Will any page faults occur when the "store" instruction is fetched and executed by the processor? If so, specify how many page faults will occur, and specify the virtual page(s) that the processor will attempt to use that will result in the page fault(s). (For each page, be sure to specify *both* the process ID and the page number.) If there are no page faults, write "NO PAGE FAULTS".

> Two page faults will occur. Fetching the "store" instruction will cause a page fault for P_A's page `0x01`, and storing the register contents will cause a page fault for P_A's page `0x05`.

b. (2 marks)
Will any pages be written from memory to secondary storage as a result of these page fault(s)? If so, indicate which page(s). (For each page, be sure to specify *both* the process ID and the page number.) If there are no page faults, or if no pages are written, write "NO PAGES WRITTEN".

> The page fault for P_A's page `0x01` will result in the eviction of P_A's page $0x3$. Since that page is dirty, it will have to be written to secondary storage. The page fault for P_A's page `0x05` will result in the eviction of P_B's page `0x04`. Since that page is not dirty, this page will not be written to secondary storage.

c. (6 marks)
Show what that core map will look like after the processor has successfully executed the "store" instruction, i.e., after any page fault(s) have occurred. Be sure to indicate the position of the clock pointer.

Core Map

Frame Number	Process ID	Page Number	
0	P_B	2	
1	P_A	0	
2	P_B	0	
3	P_B	3	
4	P_A	1	
5	P_A	4	
6	P_A	5	
7	P_B	1	← clock pointer

4. (15 marks)

a. (5 marks)
Indicate (by circling them) which of the following pieces of information can be found in the ELF file.

- $\boxed{\text{the number of pages in the text (code) segment}}$
- the number of pages in the stack segment
- $\boxed{\text{the starting virtual address of the text (code) segment}}$
- $\boxed{\text{the starting virtual address of the data segment}}$
- the starting physical address of the data segment

b. (4 marks)
Indicate (by circling them) which of the following events will cause OS/161 to create a trap frame

- $\boxed{\text{a system call}}$
- $\boxed{\text{a timer interrupt}}$
- a call to `thread_exit()`
- returning from a system call

c. (6 marks)
Please indicate whether each of the following statements is true or false by writing "TRUE" or "FALSE" next to the statement.

- Once a thread has acquired a lock (using `lock_acquire()`), it cannot be preempted by the scheduler until it has released the lock by calling `lock_release` $\boxed{\text{FALSE}}$
- Peterson's algorithm can be used to enforce mutual exclusion on a multiprocessor. $\boxed{\text{TRUE}}$
- A test-and-set instruction can be used to enforce mutual exclusion on a multiprocessor. $\boxed{\text{TRUE}}$
- It is possible to have deadlock in a system in which there is only a single (single-threaded) process. $\boxed{\text{TRUE}}$
- "Belady's anomaly" refers to a situation in which a process with a smaller virtual address space experiences more page faults than a process with a larger virtual address space. $\boxed{\text{FALSE}}$
- Exceptions cause the processor to switch to privileged execution mode. $\boxed{\text{TRUE}}$