CMS330 – Spr 2019 **Project 4** **Due: 4/8/19 by 9pm to me!**

The following problems cover virtual memory. For all problems, show your work! Answers without work will not receive full credit.

PROBLEM 1: (30pts) Consider a system with a 64 byte virtual address space, 16 byte pages, and 128 bytes of physical memory.

a) How many bits are required to encode each of the 64 virtual addresses? How about the 128 physical addresses?
b) Show how to divide each of the following virtual addresses into a virtual page number (VPN) and offset:
   i. 7
   ii. 12
   iii. 21
   iv. 50
c) Each page table entry has the following format:

```
| 1 valid bit | 3 PFN bits |
```

   Why are there 3 PFN bits?

d) The page table has four entries. For each entry, indicate if it is valid or not and give the associated PFN for each virtual page

Page table entry (in hex)	Valid?	PFN
0xA		
0xF		
0x0		
0xC		

e) Translate the following virtual addresses their corresponding physical addresses
   i. 7
   ii. 12
   iii. 50
   iv. 36

PROBLEM 2: (30pts) For this example, assume a system with an 8K virtual address space, a page size of 1K, and 64K of physical memory.

Suppose we want to execute the following instruction, which is stored at VIRTUAL ADDRESS 4096 (4K)

```
load 2048, r1
```

a) List the VIRTUAL ADDRESSES generated by this instruction.
b) Show how to divide the address into its virtual page number (VPN) and offset.
c) How many bits are required to encode each:
   i. virtual address?
   ii. physical address?
d) Suppose the system is equipped with a hardware-managed linear page table. The page table base register (PTBR) is set to 32K. Each entry is a single byte and has the following format:

```
| 1 valid bit | 6 PFN bits | 1 write-protected bit |
```

   Why are there 6 PFN bits in each page table entry?

e) The contents of the page table are:

```
0x00
0x00
0xA8
0x00
0x8C
0x00
0x00
0x00
```

List all physical memory accesses that are performed when

```
load 2048, r1
```

executes. *Remember that page table lookups also access physical memory!*

PROBLEM 3: (20pts) Label each of the following statements as either True or False AND explain your reasoning.

a) A smaller page size leads to smaller page tables.
b) A smaller page size leads to more TLB misses.
c) A smaller page size leads to fewer page faults.
d) A smaller page size reduces paging I/O throughput.

PROBLEM 4: (20pts) In this exercise, we consider a simple machine with a MMU that implements virtual memory based on segmentation. The main specifications of this machine are as follows:

- The MMU hardware has two pairs of (base, bounds/limit) registers (i.e., a process can at most have two segments).
- Virtual addresses (including the explicit segment ID) are stored in 10 bits and physical addresses are stored in 16 bits.
- The machine is equipped with a capacity of 16 kB of physical memory (RAM).

Now let's consider a process with two segments, for which the MMU configuration is as follows:

- segment 0:
  - base = 0x8400
  - limit (size) = 0x100 bytes
- segment 1:
  - base = 0x0c00
  - limit (size) = 0 bytes

Unfortunately, it appears that the above MMU configuration is incorrect because the values written in some of the MMU registers have been corrupted due to some hardware defects. More precisely, the wrong values have exactly one flipped bit: i.e., compared to the originally written (good) value, there is exactly one bit that has been accidentally modified (from 0 to 1 or vice-versa).

Let us assume that we also know (for sure!) that:

- virtual address 0x300 should cause a segmentation violation (invalid address)
- virtual address 0x2ff is a valid address

a) Which of the segmentation registers (among seg. 0 base, seg. 0 limit, seg. 1 base, seg. 1 limit) determines whether these two addresses are valid or not? Explain your answer.
b) What should the correct value be for the register identified in your previous answer? Explain your answer.
c) Given the correct value identified in the previous answer (and if there is no other error for this segment), what physical address should virtual address 0x2ff translate to and why?