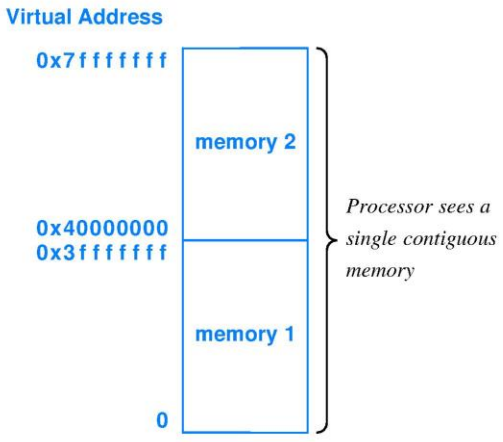
CS 250 Spring 2017 Homework 10
Due 11:58pm Wednesday, April 12, 2017
Submit your typewritten file in PDF format to Blackboard.

1. Consider the circuit for a 4 KB, 2-way set associative write-back cache using LRU replacement and having 16-byte blocks that is for a computer using 32-bit addresses that address 32-bit words in main memory.
   a. What are the sizes of the tag, index, and block offset fields in bits?
      To find the offset, set offset = O in the equation: 2^O = 16 bytes
      We get O, or offset, is equal to 4.
      We are given that 4KB make up the entire cache, with 2 cache blocks per set this means a set contains 32 bytes. 4KB / 32B = 256 = 2^**8**
      Therefore, we know the index is 8.
      The tag remains, so we subtract our offset and index from 32.
      32 – 8 – 4 = 20

   b. How many blocks are there in the cache?
      We calculate the amount of blocks in a cache from 2^(index bits). Therefore in our case where index = 4…
      There are 2^8 = 256 blocks in the cache.

   c. How many bytes of overhead are there in the cache? What percentage of all memory bits in the cache are overhead bits?
      There needs to be 12,032 bytes of overhead SRAM in order to accomadate for the 2^8 cache lines. The percentage would be (tag bits)/(address size). Therefore using the given information we get 20/32 as **62.5%**.

2. Given the following virtual address space



and the C program

```
char c;
char *p;
p = (char *)1073741826;
c = *p;
```

Which memory module will be referenced, and where will the referenced byte be located with the module?.
Memory module 2 will be referenced. The referenced byte (p in above code) will be mapped to 40000002 as a memory address.

3. Assuming a page size of 8 Kbytes what is the page number in hexadecimal and offset in hexadecimal for addresses 0x00000FFF, 0x00001FFF, 0x00002002, and 0xFFFFDFFD? Pad as needed with leading zeros to obtain hexadecimal representations for your answers.

8Kb = 8192 bytes

Address	Page Number	Offset
0x00000FFF	0x00000000	0x00000FFF
0x00001FFF	0x00000000	0x00001FFF
0x00002002	0x00002000	0x00000002
0xFFFFDFFD	0xFFFFC000	0x00001FFD

4. Assume a virtual memory with a 13-bit address space organized into 8 pages. Physical DRAM memory implements addresses 0x000 through 0xFFF.

Page number	Page frame
0	3
1	1
2	Not present in DRAM memory
3	Not present in DRAM memory
4	2
5	Not present in DRAM memory
6	0
7	Not present in DRAM memory

In the table below. for each listed virtual address, fill in the corresponding physical address or note if a page fault would occur.

Virtual address (13 bits shown in hex with leading zero padding)	Corresponding physical address or Page fault
0x0000	0xC00
0x0E90	Page Fault
0x03FF	0xFFF
0x0400	0x400

5. In Question 4 we assumed that the given page table did not change as the CPU tried to access each of the 4 virtual addresses. An actual page table would dynamically update per its design as the CPU accessed successive virtual addresses.
So, let's assume that the CPU has accessed the four virtual addresses in the table of Question 4 in the order shown, that is, in the order of reading down the column of the table. Further, let's assume that any page faults that you identified in answering Question 4 are the only page faults that occurred when the CPU accessed these four addresses in order (depending on

the replacement algorithm this need not be the case). Finally, let's assume that we do not know the page replacement algorithm used by this virtual memory system design nor do we know which accesses in Question 4 were loads and which were stores. Finally, a complete page table must at least have bits for each page to indicate Valid and Dirty (other bits may be needed to support access permissions and the page replacement algorithm, but let's not worry about access permission and we do not know the replacement algorithm, so we will ignore those bit(s) as well).

Fill in the empty page table below to show its contents after the fourth access. If there is more than one possibility for a translation entry, list all possibilities. Use the following definitions for the Valid and Dirty bits: 0 = characteristic not true, 1= characteristic is true, X = table entry is correct with either a 0 or 1 for this bit, and ? = "given the limited degree of knowledge about this virtual memory system design, the value of this bit must be only one of 0 or 1 but the actual value cannot be determined."

Page	Starts at frame physical address	Present	Dirty
0	0x0C00	1	?
1	0x0800	1	?
2	0x0C00 (Not present in DRAM memory)	0	X
3	0x0C00, 0x0400, 0x0800, 0x000	1	?
4	0x0400	1	X
5	0x0C00 (Not present in DRAM memory)	0	X
6	0x000	1	X
7	0x0C00 (Not present in DRAM memory)	0	X