CS 351 Exam 3, Fall 2013

Your name: ____

Rules

- You may use one handwritten 8.5 x 11" cheat sheet (front and back). This is the only resource you may consult during this exam.
- Include explanations and comments in your answers in order to maximize your partial credit. However, you will be penalized for giving extraneous incorrect information.
- You may use the backs of these pages if you need more space, but make it clear where to find your answer to each question.
- Unless otherwise specified, you do not need to work out the arithmetic on math problems. Just do enough algebra to set up an answer of the form: Answer = [arithmetic expression] [units]

Grade (instructor use only)

	Your Score	**Max Score**
Problem 1: Short answer		20
Problem 2: AMAT		15
Problem 3: Cache sizing		20
Problem 4: Cache mappings		15
Problem 5: Page table sizing		15
Problem 6: Virtual memory mappings		15
Total		100

Problem 1: Short-answer (20 points)

a) [1 point + 1 bonus point if everyone gets it right] How many bits are in a byte?

b) [3 points] Which can store more data, a system's L2 cache or its main memory? Explain.

c) [4 points] What write policy (write-back or write-through) is used for pages in physical memory, and why?

d) [4 points] What kind of access (e.g. write/read, hit/miss) triggers a write to memory in…

- a write-back cache?
- a write-through cache?

Be as specific as possible.

e) [4 points] The x86 MOV instruction can be used to copy the value of the page table base register into a general-purpose register. Assuming that this instruction does not trap when executed in user mode, is it safe to let the guest OS directly execute it? Specifically, why or why not?

f) [4 points] What does *physical memory* refer to in a virtualized environment? Is this different from *physical memory* in a non-virtualized environment?

Problem 2: Average memory access time (15 points)

Assume that a program is executed on a system with the following memory hierarchy and hit rates.

- L1 cache: 95% hit rate, 1-cycle access time
- L2 cache: 90% hit rate, 5-cycle access time
- Main memory: 100% hit rate, 80-cycle access time
- A write buffer eliminates all of the stalls from writing back to memory.

The program has the following characteristics:
- 25% of instructions are loads
- 10% of instructions are stores

A. [5 points] What is the average memory access time of this program?

B. [5 points] The base CPI of this hardware is 1. This number assumes that all memory accesses are L1 hits. Given the actual memory access patterns above, what is the true CPI for this program on this hardware?

C. [5 points] Consider the L2 hit rate. Instead of fixing it at 90%, let's call it *H*. So now we have:
   - L1 cache: 95% hit rate, 1-cycle access time
   - L2 cache: *H* hit rate, 5-cycle access time
   - Main memory: 100% hit rate, 80-cycle access time
   - A write buffer eliminates all of the stalls from writing back to memory.

If *H* is 0%, it's easy to show that having this L2 cache is worse than not having one at all. If *H* is 100%, then having an L2 cache is extremely helpful.

Show an expression that is true if and only if having an L2 cache with hit rate *H* improves the average memory access time for this system.
(For example: $0.4H > 0.2 + 0.1/5$)

Problem 3: Cache sizing (20 points)

A byte-addressable machine with 64-bit memory addresses has a cache with the following properties:

- 16-byte cache blocks
- 16KB of data in the cache
- 2-way set-associative
- Write-back

A. [3 points] How many cache blocks are there?

B. [3 points] How many cache sets are there?

C. [3 points] Given a memory address, how many cache blocks could it possibly map to?

D. [3 points] How many memory addresses could possibly map to a given byte in the cache?

E. [5 points] How many bits of metadata are required for each cache block? Explain what each is for.

F. [3 points] How many ***bits*** are needed to implement the cache (data and metadata)?

Problem 4: Cache mappings (15 points)

For each of the following caches, show the cache contents after memory references to addresses 01110110 and 10001010. Include any necessary metadata. The main memory has 8-bit addresses.

Be sure you show at LEAST the valid bit and tag of each block.

A. [5 points] The cache has:

- 8B of data (total)
- 1-byte blocks
- Direct mapping

B. [5 points] The cache has:
   - 8B of data (total)
   - 4-byte blocks
   - Direct mapping

C. [5 points] Which of these two caches would perform better for a program with lots of spatial locality but low temporal locality? Explain.

Problem 5: Page table sizing (15 points)

A byte-addressable memory system has 64-bit virtual addresses and 33-bit physical addresses, with 16 KB pages.

A. How many virtual pages does each process have?

B. How many physical pages does the system have?

C. If we were to use flat, per-process page tables, how many page table entries would each process need to store?

D. How many bits (data and metadata) are needed at minimum for each page table entry?

E. How many bits (data and metadata) are needed at minimum for each TLB entry? Assume that the TLB gets flushed on a context switch.

Problem 6: Virtual memory mapping (15 points)

A certain system has an unrealistically tiny physical memory with 8 pages. The OS has mapped these pages to processes as follows:

- Physical page 0 is reserved for the OS
- Physical pages 1, 2, 4, and 6 are used by Process A
- Physical pages 3, 5 and 7 are used by Process B

A. [8 points] If processes have 4 virtual pages, show plausible page tables for processes A and B.

B. [7 points] If the page size is 32B, how would you translate a request from Process A for virtual address 1001010? Explain.

What about Process B?