CS232 Midterm Exam 3
April 16, 2008

Name: ***Solution***

Section: 10am noon 2pm 3pm 4pm (circle one)

- This exam has 4 pages; some equations are provided on the cover for your reference.
- You have 50 minutes, so budget your time carefully!
- No written references or calculators are allowed.
- To make sure you receive credit, please write clearly and show your work.
- We will not answer questions regarding course material.

Question	Maximum	Your Score
1	35	
2	30	
3	35	
Total	100	

CPI = Cycles Per Instruction
AMAT = hit_time + (miss_rate * miss_penalty)
Memory Stall Cycles = # accesses * miss_rate * miss_penalty
1KB = 1024B, 1M = 1024KB

Question 1: Cache Construction and Behavior (35 points)

For a 64KB 2-way set-associative cache with 128B blocks on a machine with 32-bit address spaces (both virtual and physical), answer the following questions in part (a) and (b). State any assumptions.

Part (a)
How big are the block offset, index and tag fields? (10 points)

128 bytes -> 2^7 block size -> 7 offset bits
64KB cache -> 2^16 cache size / 2^7 block size -> 2^9 blocks.
2^9 blocks / 2 blocks/set (because it is 2-way associative) = 2^8 sets -> 8 index bits
Size of address in bits = tag size + index size + offset size -> tag size = 32 – 8 – 7 = 17 -> 17 bits as tag

Part (b)
How many bits of storage are required to implement the cache including all data, tag, valid, dirty and LRU state. **Make your derivation clear and state any assumptions.** You may leave your answer as an expression. (15 points)

(you needed to make an assumption about write back vs. write through; dirty bit = 1 if write back, dirty bit = 0 if write through; I'll assume write back for the answer, but either would be fine.)

Total size = # sets * size of set
Size of set = (# blocks/set * size of block) + size of LRU state # LRU state is per set, not per block
Size of block = data size + tag size + valid bit + dirty bit = 128B * 8b/1B + 17b + 1b + 1b = 1043b/block
Size of LRU state = 1 bit # 1 bit is sufficient to determine the LRU in a set of two things
Size of set = (2 * 1043bit) + 1 = 2087 bits
Total size = 256 * 2087 bits = 534,272 bits -> 66,784 bytes

Part (c)
Consider a fully-associative cache with four blocks where the block offset is 4 bits long. Identify which of the following accesses are hits and misses. (10 points)

Block 1	1000			1000					0111				1111
Block 2		0111					1001			1001			
Block 3			1111					1111				1000	
Block 4					0000	0000					0000		

Address (binary)	Hit/Miss
10000000	Miss
01110100	Miss
11111000	Miss
10001000	***Hit 1***
00000100	***miss***
00001110	***Hit 4***
10010100	***Miss – evicts 2***
11111110	***Hit 3***
01110100	***Miss – evicts 1***
10010000	***Hit 2***
00000000	***Hit 4***
10001000	***Miss – evicts 3***
11110000	***Miss – evicts 1***

Question 2: Disks (30 points)

Consider the following disk: Assume a hard drive that has a 10,000 rpm rotational speed, a 3ms average seek time, and a negligible amount of overhead. The disk track has 600 sectors, and each sector holds 8 kB of data. The drive can read or write data as fast as it rotates. To make the computations easier, you may assume that 1 kB = 1,000 bytes and 1 MB = 1,000,000 bytes. State any assumptions.

Part (a)
How long does it take to read a random single sector on average? You may leave your answer as an expression. (15 points)

Rotational delay = time for half rotation = 0.5 rotations / 10000 rpm * 60000 ms/minute = 3ms
transfer 1 sector = spin across 1/600 of disk = 1/600 rotation / 10000 rpm * 60000 ms/minute = 0.01 ms

Total read time for random sector = rotation + seek + overhead + transfer = 3+3+0+0.01 = 6.01 ms

Part (b)
Consider now a large file that is over a GB. Assume that its sectors are laid out contiguously and across neighboring tracks and that the seek time to a neighboring track is 1ms. Assume also that there is sufficient buffering in the hard drive to read/write the sectors in any order. Estimate the read/write bandwidth for the file given these assumptions and provide units for your answer. (10 points)

Bandwidth is a rate (bytes per unit time), so we can determine this by assuming any size transfer; I'll use a single track (using the assumptions above).

Time to read a single track = seek time + rotational delay + transfer time + overhead
seek time is given as 1ms; there is no rotational delay because tracks can be read in any order
transfer time is 6ms for one full rotation to read all 600 sectors; no overhead
= 1ms + 0 + 6ms + 0 = 7ms
A single track has 600 sectors, which is 600 * 8kB or 4.8MB

Thus, we can read/write at 4.8MB every 7ms or 685MB/s

Part (c)
If you have two identical disks, like the one described above, installed in your computer, and you were copying a highly-fragmented 1GB file from one disk to the other disk where you could lay it out sequentially. Discuss how the data moves through the computer and whether you would likely be able to read from one disk while you write to the other. (5 points)

Data will take one of two paths: 1) (at least it used to be for IDE/ATA drives) the O/S would do two series of DMAs; moving the data from the hard drive, through the I/O bus, across the north bridge, and into memory and the reverse from memory back to the other disk, or 2) (SCSI drives could always do this and perhaps newer ATA controllers can also) moving data from disk-to-disk just at the controller, causing the data to move from the hard disk, across the I/O bus to the other disk.

In either case, reading data from a highly fragmented disk would produce data at a rate much lower than any of the buses/connections could handle. By pipelining, we could be writing sector I to the second disk while we are reading sector i+1 from the first.

Question 3: Conceptual Questions (35 points)

Part (a) Give examples of both temporal and spatial locality in programs, making it clear that you know the difference. (5 points)

Temporal – accessing same data repeatedly, e.g., an induction variable in a loop.
Spatial – accessing nearby data, e.g., sequential accesses to an array.

Part (b)
Why do we build cache hierarchies? What trade-off are we managing? (5 points)

We are managing the "hit rate" vs. "access time" trade-off.

Generally, as we make caches larger and more associative, both the hit rate and the access time (the number of picoseconds it takes to read a value from the cache) go up. (The former is good, the latter is bad). A cache hierarchy addresses this trade-off by including a small first-level cache that is optimized for access time, followed by larger second-level cache that optimizes for hit rate. In this way, we can closely approximate the memory latency of a single cache with the access time of the L1 and the hit rate of the L2.

Part (c)
Explain how indirection enables virtual memory to prevent one application from reading/writing the memory of another application. What hardware support is necessary to maintain this security? (10 points)

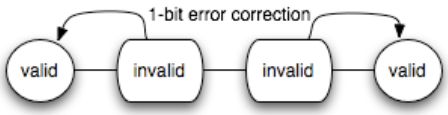
By mapping (remember indirection) the virtual address spaces of two applications to non-overlapping regions of physical memory, each application is prevented from reading and writing the memory of the other.

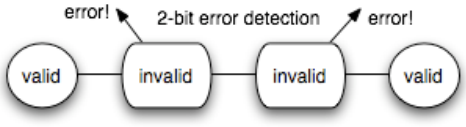
Maintaining this property requires that applications cannot change their own virtual-to-physical address mappings. This is enforced by preventing applications (running in user mode) from writing the processor's page table base pointer register and not placing the page tables in application accessible regions (i.e., not giving the application virt.-to-phys. mappings to its own page tables (or that of other applications for that matter).)

Part (d)
In a certain coding scheme, the binary words 01100011 and 01000110 are as close in Hamming distance as any pair of code words. What Hamming distance and degree of error detection or correction does this code achieve? (10 points)

If you identify the bits that you need to flip to change 01100011 and 01000110 (which corresponds to a bit-wise XOR), you get 00100101, which has 3 bits set. Thus there is a Hamming distance of 3 between the numbers. This translates to either: 2-bit error detection OR 1-bit error correction.

Part (e)





In a computer that performs translation from virtual to physical addresses before cache access, how does page size affect the cache hit rate? Explain. *Note: we are not asking about TLB hit rate.* (5 points)

As long as the page size is a multiple of the size of a cache block (i.e., larger and non-overlapping, which is common) then the actual page size has no impact on the spatial locality seen by the cache and, therefore, no impact on cache hit rate.