CS232 Midterm Exam 3
April 18, 2007

Name: _____

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

- This exam has 4 pages.
- 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	40	
2	30	
3	30	
Total	100	

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

Question 1: Cache Organization and Performance (40 points)

Part (a)

Address	Tag	Index	Block offset

Consider a 2-way set-associative cache that uses a 10-bit cache index and has 32-byte cache blocks. If a machine uses 32-bit physical addresses, compute: (5 points each, show work for partial credit)

The number of blocks in the cache: **a 10-bit index means there are 2^{10} (or 1024) sets. Because it is 2-way set-associative, each set has two blocks. Thus, there are 2x1024 or 2048 blocks.**

The size of the block offset: **With 32-byte blocks, we need a 5-bit block offset to specifiy which of the 32 = 2^5 bytes in the block we are referencing (as the first byte of a load or store).**

The size of the tag: **For the tag we must store all of the bits of the (physical) address that are not part of the index or the block offset. Thus, 32 - 10 - 5 = 17 bits for the tag.**

Part (b)

Given a two-way set-associative (using LRU) cache with a total of 4 blocks of 4 bytes each, which of the following byte accesses hit? (15 points)

Address (binary)	Hit/Miss
110001	Miss
100111	**Miss**
001111	**Miss**
001100	**Hit**
010001	**Miss**
110010	**Hit**
100101	**Hit**
001110	**Hit**
100001	**Miss**
110001	**Hit**

Part (c)

Consider a program with the instruction mix shown at the right. What will be the CPI of this program if it executes on an ideally-pipelined machine with no hazards, has no instruction cache misses, a 98% data cache hit rate, and a miss penalty of 100 cycles. All misses cause stalls. (10 points)

Instruction Type	Frequency
ALU	40%
Loads	25%
Branches	20%
Stores	15%

The CPI of the ideally-pipelined machine with a perfect cache would be 1. To compute the full CPI we need to add the number of memory stall cycles per instruction. Unfortunately, there was an error on the cover sheet suggesting that only loads caused stall cycles; this is actually complex issue, but the problem clearly states that all misses cause stalls. Thus our intended answer was:

CPI = baseCPI + (fraction loads + fraction stores) x miss_rate x miss_penalty
= 1 + (.25 + .15) x (1.0 - .98) x 100 cycles = 1 + (.4 x .02 x 100) = 1 + .8 = 1.8 cycles

Question 2: Disks (30 points)

A web cache intercepts http requests and tries to satisfy them from locally-stored data. If the cache has the desired file, it can be returned to the requestor without accessing the internet, reducing latency and saving network bandwidth. If the cache doesn't have the file, the request is placed on the network and the file is buffered by the cache when it is returned.

Web caches are typically implemented by storing some files in memory and some files on disk. Assume a hard drive that has a 12,000 rpm rotational speed, a 3ms average seek time, and a negligible amount of overhead. The disk track has 200 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)
Many html files are small and could be stored in a single sector. How long does it take to read a single sector on average? You may leave your answer as an expression. (10 points)

Access time = seek time + rot. delay + transfer time + overhead
Rot. Delay = (.5 rotations)/12000rpm = 2.5msec
Transfer time = (full rotation time)/(# sectors/rotation) = 5ms/200 = .025msec
Access time = 3ms + 2.5ms + .025ms + 0 = 5.525ms

Part (b)
Many non-html files (*e.g.,* .gif, .jpeg, .pdf, .doc) requested over the web are substantially larger than a single sector. How long would it take to read a 400kB file on average, if the file was laid out contiguously on disk? You may leave your answer as an expression. (10 points)

Since they are laid out contiguously on a track, we can assume that only one seek and one rotational delay needs to be paid for the whole file. Need to compute the new transfer time.

 # sectors = 400kB / 8kB = 50 sectors **Transfer time = 50 x .025ms = 1.25ms**

Access time = 3ms + 2.5ms + 1.25ms + 0 = 6.75ms

Part (c)
In light of your answers for part (a) and (b), if you had a collection of html and non-html files that were equally likely to be accessed, suggest how they should be allocated to memory and disk so as to optimize performance. (hint: think about caching and average access time in an AMAT sense). (10 points)

For a server we want to minimize the number of files served and AMAT is useful, in that if the average access time is longer, we are in general going to have lower throughput.

AMAT = hit_time + miss_rate x miss_penalty

The access time for all of the files is going to be roughly the same whether theyare html files or non-html files, so our opportunity is in trying to reduce the miss rate. If all the files are equally likely to be accessed, then we'll have the best hit rate when the largest number are in memory. Thus we should sort the files by size and store the smallest in memory and the largest on disk.

Question 3: Conceptual Questions (30 points)

Part (a)
Define the two types of locality. (10 points)

Temporal locality: a variable that is accessed is likely to be accessed again soon.

Spatial locality: when a variable is accessed, variables at nearby addresses are likely to be accessed soon.

Part (b)
How is parity implemented? (5 points)

We add an additional bit to each "word" of data, (for even parity) setting the value of this extra bit to 1 if there are an odd number of bits set in the "word" and 0 if there are an even number of ones. As a result, the number of 1 bits in the "word" plus the parity bit should be an even number.

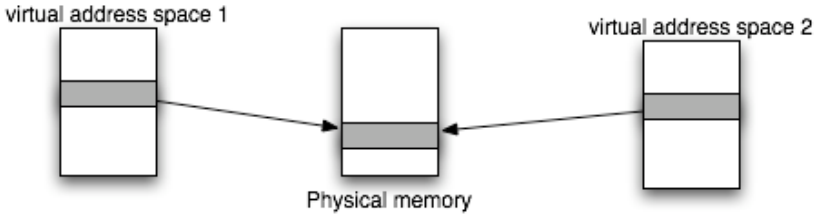
When the word is read back, we re-compute what the parity bit should be, and compare that to what is stored. If they don't match, we have detected an error.

Part (c)
What Hamming distance does parity achieve? Explain. (5 points, 3 of which are for the explanation)

Parity achieves a Hamming distance of two, because between every valid even parity encoding, there is an invalid odd parity encoding. Thus, to move between any two valid even parity code words, it takes two steps, as shown below. Each step add one to the Hamming distance.

Part (d)
How does virtual memory enable controlled sharing of memory between processes? (5 points)

The operating system can map the same physical page into two address spaces, as shown.



Part (e)
(**True**/False) Assuming a direct-mapped cache of a given block size, a cache with 2N blocks is guaranteed to hit on every access that hits in a cache with N blocks. Explain. (5 points, 3 of which for the explanation.)

When going to 2N blocks, the addresses that mapped to a given block with 1N blocks, now map to two blocks. In the 1N case, the stream of addresses that compute a given index will only result in hits, if the previous access in that stream had the same tag. When this stream maps to two blocks, when two accesses have the same tag, there will still be a hit. (In addition, there will be hits when there are two accesses in the stream with the same tag and the intervening accesses in the stream differ in the last bit of the tag (as those get mapped to the other block)).