CS232 Midterm Exam 3
April 26, 2006

Name: ______________________________

Section: ______________________________

- This exam has 5 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	55	
2	45	
Total	100	

Question 1: Cache Organization and Performance (55 points)

For the following questions, consider a write-allocate, write-back, direct-mapped cache that holds 32KB of data in 32-byte blocks.

Address	Tag	Index	Block offset

Part (a)
How large would the index, tag, and block offset fields be for 32-bit addresses? (5 points)

With 32-byte blocks the block offset has to be 5 bits ($2^5 = 32$).
There are 1024 blocks (32KB/(32B/block). Because it is direct-mapped, there are also 1024 sets.
With 1024 sets the index is 10 bits ($2^{10} = 1024$).
Those bits that are not part of the index or block offset must be part of the tag: (32 - 10 - 5) = 17 bits.

Part (b)
Compute how many bits of storage are required to implement the cache for a machine with 32-bit addresses (include all fields: data, tag, etc.). You can leave your answer as an expression. Show work for partial credit. (10 points)

Store requirements = (# of blocks) * (bits per block) = 1024 * (bits per block)
Bits per block = (data/block) + tag + valid bit + dirty bit # dirty bit since it is write-back
= 32*8b + 17b + 1b + 1b

Part (c)
Assuming the cache was initially empty, which of the following byte accesses would hit in the cache? *Note: We've only shown the bottom 20 bits of the addresses; the top bits are all zeros.* (10 points)

Type	Address (hex)	Address (binary)	Hit/Miss
Store	`0x24325`	0 0 1 0 0 **1 0 0 0 0 1 1 0 0 1** 0 0 1 0 1	Miss
Load	`0x439af`	0 1 0 0 0 **0 1 1 1 0 0 1 1 0 1** 0 1 1 1 1	Miss
Store	`0x34328`	0 0 1 1 0 **1 0 0 0 0 1 1 0 0 1** 0 1 0 0 0	**Miss/Alloc (evicts 1st block)**
Load	`0x24330`	0 0 1 0 0 **1 0 0 0 0 1 1 0 0 1** 1 0 0 0 0	**Miss**
Store	`0x439bf`	0 1 0 0 0 **0 1 1 1 0 0 1 1 0 1** 1 1 1 1 1	**Hit**
Store	`0x24327`	0 0 1 0 0 **1 0 0 0 0 1 1 0 0 1** 0 0 1 1 1	**Hit**
Load	`0x34330`	0 1 1 0 1 **0 0 0 0 1 1 0 0 1 1** 0 0 0 0 0	**Miss**

Question 1, continued

```
void compute(float A[], float B[], float C[], float k, int length) {
   for (int i = 0 ; i < length ; ++i) {
      C[i] = k*A[i] + B[i];
   }
}
```

Part (d)
Still considering the cache described on the previous page, compute the *average* number of caches misses per iteration of the loop. Assume that the base addresses of arrays `A`, `B` and `C` all map to the initial byte of *distinct* cache blocks. Explain and state any assumptions. (20 points)

Assume that floats are 4B each. Assume that "i", "k", and "length" are register allocated.

There are 3 cache accesses per iteration of the loop, for "A[i]", "B[i]", and "C[i]". We are striding through this code accessing sequential word addresses. Eight 4-byte floats will fit in every cache line so we are going to miss once every eight requests to bring in a new cache block. Because A, B, and C map initially to different cache blocks, and we're walking them at the same speed, they will continue to map to different cache blocks, so we need not consider conflict misses.

Thus, each iteration we will have 3*(1/8) misses or 3/8 misses.

Part (e)
We could alternatively use the most-significant bits (MSBs) of the address as the index, as shown below. Why would you expect, in general, a cache with such an ordering would not perform as well as one that used the MSBs as the tag? (10 points)

Address	Index	Tag	Block offset

If we use the MSBs for the index, then two consecutive cache blocks (e.g., the block at address A and the block at address A+32, for 32-byte blocks) will have the same index, and therefore conflict in the cache. Given that spatial locality is common, we'd expect to frequently access neighboring blocks (e.g., for the stack). By using the MSBs for the tag, these blocks can co-exist in the cache, likely resulting in a lower miss rate for almost all workloads.

Question 2: Pipelining (45 points)

Part (a)
Give a non-computing example of pipelining not involving laundry. (5 points)

Part (b)
Assume you have 10 items to process. If you can pipeline the processing into 5 steps (perfectly balancing the pipeline stages), how much faster does pipelining enable you to complete the process? You can leave your answer as an expression. (10 points)

Question 2, continued

Consider the following MIPS code:

```
loop: sub     $t3,    $t3,    $t0
      add     $t0,    $t0,    $t0
      addi    $t1,    $t0,    5
      lw      $t2,    0($t0)
      add     $t2,    $t2,    1
      sw      $t2,    0($t1)
      bgt     $t3,    $zero,  loop
```

Part (c) Label all dependences within one iteration of this code (not just the ones that will require forwarding). One iteration is defined as the instructions between the `sub` and `bgt` *inclusive.* (10 points)

Part (d) The above code produces the pipeline diagram shown below when run on a 5-stage MIPS pipeline with stages IF (fetch), ID (decode), EX (execute), MEM (memory) and WB (write-back).

Note: stalls are indicated by –, and registers can be read in the same cycle in which they are written.

Inst	iter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
sub	N	IF	ID	EX	MEM	WB												
add	N		IF	ID	EX	MEM	WB											
addi	N			IF	ID	-	EX	MEM	WB									
lw	N				IF	-	ID	EX	MEM	WB								
add	N						IF	ID	-	EX	MEM	WB						
sw	N							IF	-	ID	EX	MEM	WB					
bgt	N								-	IF	ID	EX	MEM	WB				
sub	N+1										-	-	IF	ID	EX	MEM	WB	

For each of the following questions, justify your answer or explain why it cannot be answered using the given information. *Note: full credit requires explanations*. (20 points)

Is forwarding from EX/MEM to the EX stage implemented? Explain.

Is forwarding from MEM/WB to the EX stage implemented? Explain.

Note: The branch target is computed in the ID stage. In which stage are branches resolved? Explain.

The branch target is computed in the ID stage. What is the branch prediction scheme? Explain.