OPEN BOOK, OPEN NOTES. You can use material from previous exams, the book, etc. You CANNOT use other people for help.

SHOW YOUR WORK. ANSWERS ONLY will be treated as if cheating may have occurred, ESPECIALLY SINCE THIS IS A TAKEHOME EXAM.

OUT: Nov 21, 2011, 5PM-PST
IN: Nov 28, 2011, 10AM-PST

Your signature is your promise that you have not cheated and will not cheat on this exam, nor will you help others to cheat on this exam. NOTE: If exam is not signed, it will not be scored.

Signature: _____________________________________

ABET Requirements:
1. Use various metrics to calculate the performance of a computer system
2. Identify the addressing mode of instructions
3. Determine which hardware blocks and control lines are used for specific instructions
4. Demonstrate how to add and multiply integers and floating-point numbers using twos complement and IEEE floating point representation
5. Analyze clock periods, performance, and instruction throughput of single-cycle, multi-cycle, and pipelined implementations of a simple instruction set
6. Detect pipeline hazards and identify possible solutions to those hazards
7. Show how cache design parameters affect cache hit rate
8. Map a virtual address into a physical address

Question 0______________ (10 points)
Question 1 _____________ (15 points) (ABET #2, #3)
Question 2 _____________ (10 points) (ABET #6)
Question 3 _____________ (20 points) (ABET #5, #1)
Question 4 _____________ (20 points) (ABET #5, 7)
Question 5 _____________ (10 points) (ABET #8)
Question 6 _____________ (15 points) (ABET #1)

TOTAL ______________
1. (15 points) Assume that you have a pipelined implementation of MIPS with a perfect cache. Draw the pipelining diagram for the code along the left side. I have started the diagram for you with the first statement. **Indicate all data dependencies by circling the registers in the code.** Indicate all necessary forwarding from the output of one functional unit to the input of another functional unit. Indicate all necessary stalls with a nop. **Keep everything that happens in the same clock cycle in a straight column.** You do not have to shade the active units. Assume BRANCH NOT TAKEN.

**TWO MORE DIFFICULT THINGS ADDED to this problem:**

a) Assume the register file **CANNOT** do a WRITE and READ in the SAME cycle

b) The branch decision has been moved from the EX stage into the ID stage

Add $4, $1, $2

lw $2, 64($4)

beq $2, $4, NEXT1

beq $3, $2, NEXT2
2. (10 points) The CPI of a machine with a perfect cache is 1.0 for a certain benchmark.

Now, assume a L1 instruction cache miss rate of 3.1% and a L1 data cache miss rate of 21%.
Assume the miss penalty (for either L1 caches) is 5 cycles. The hit time is 2 cycle (L1 cache).
Assume a L2 instruction cache miss rate of 1% and a L2 data cache miss rate of 5%.
Assume the miss penalty (for either L2 caches) is 400 cycles. The hit time is 50 cycles (L2 cache).

The frequency of data access instructions in this benchmark is 17%.

How much faster is the machine with the perfect cache than the one with the imperfect L1 and L2 caches? Otherwise the machines are identical and running the same code. Express your answer either as a fraction or as a decimal rounded to the hundredths place. (X.XX) Your answer should be > 1.

SHOW ALL WORK.
3a. (5 points) Assume we have a 16 KB cache, direct-mapped cache. (KB = 2^10 bytes)
The block size is 1 word.
Given a 32-bit physical address, divide up all the bits and indicate what they are used for to find and access the requested word in the cache. Do NOT draw the cache entries, the mux, the AND gate, ...
Just indicate exactly how many bits and which bits of the address are used for each purpose. Label your groups of bits with their purpose.

3b. (5 points) Assume we have a 16 KB cache, 4-way set associative cache. (KB = 2^10 bytes)
The block size is 16 words.
Given a 32-bit physical address, divide up all the bits and indicate what they are used for to find and access the requested word in the cache. Do NOT draw the cache entries, the mux, the AND gate, ...
Just indicate exactly how many bits and which bits of the address are used for each purpose. Label your groups of bits with their purpose.

3c. (10 points) DRAW the cache entries, the mux, the AND gate, the valid and dirty bits, LRU, for BOTH designs below (if they are required. You will be penalized if they are NOT required). Also include the decode logic symbolically, so that I know that you understand how to build an address decoder.
4. (20 points) CACHE DYNAMICS: 16 bytes total (4 words); block size = 2 words

Direct-Mapped; Writeback Cache; Assume cache is EMPTY at t=0.

INSTRUCTION SEQUENCE:

$0 = register with all 0’s

lw $1, 0($0)
lw $2, 4($0)
sw $10, 16($0)
lw $3, 8($0)
lw $4, 19($0)
lw $11, 8($0)

<table>
<thead>
<tr>
<th>Block Address</th>
<th>Cache Index</th>
<th>HIT/MISS</th>
<th>DIRTY?</th>
<th>0</th>
<th>1</th>
<th>DIRTY?</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


5. (10 points) A) Write a loop in MIPS that first multiplies each INT in an array size of 4 by the constant ‘4’, and then adds it to a running sum. Use my registers ($t0, $t1, $t2) as declared in the comments. Use only MIPS core instructions (and a new instruction ‘mult’). Feel free to use registers $t1 - $t7.

# $t0 holds the address of the first int in the array

LP: 

UNROLLED:

B) Unroll your loop (just the code that starts at LP) so that it executes 1 time and has no data hazards. Write your code to the right of your code from part A).
6) (15 points) Assume you have two processors: one that operates on the normal sequential loop code in problem #6, and another that operates on the loop-unrolled version of problem #5, and can execute on four parallel instructions simultaneously.

**PROCESSOR-1 (1-core/die):**
- VDD=1.0V
- f = 1GHz
- Capacitance(C) = 1e-9 F (for entire PROCESSOR)

**PROCESSOR-2 (4-cores/die):**
- VDD=0.6V
- f=200MHz
- Capacitance(C) = 0.3e-9 F (for each core)

**PROCESSOR-2** can execute on 4 arithmetic operations simultaneously, if they are parallel. It CANNOT operate on a LW and a MULT simultaneously, but it can do a MULT and another MULT, if they are independent MULTs, for example.

A) What is the average power consumption, of PROCESSOR-1 and PROCESSOR-2, over this entire code sequence? Assume that an idle core is burning the same power as an active core.

B) What is the energy-consumed for each PROCESSOR (both 1 and 2) in executing the two different code sequences for Problem 6? Assume the energy consumed for executing one instruction is CV^2. For example, for four instructions in parallel, the energy will be 4*CV^2. For PROCESSOR-2, if only one core is performing multiplication while the other 3 are idle, the energy is still 4*CV^2.

C) What is the execution time for each PROCESSOR, using both these code sequences from Problem 5?