OPEN BOOK, OPEN NOTES. OPEN COMPUTER IS OK. NOT SOLVING PROBLEMS DIRECTLY USING CALCULATORS or GOOGLE, or SIMILAR ELECTRONIC COMPUTATION.

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.

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 two's 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 _____________   (10 points)  (ABET #4)
Question 2 _____________   (15 points)  (ABET #2, #3)
Question 3 _____________   (5 points)    (ABET #6)
Question 4 _____________   (10 points)  (ABET #5, #1)
Question 5 _____________   (10 points)  (ABET #7)
Question 6 _____________   (15 points)  (ABET #5, #7)
Question 7 _____________   (10 points)  (ABET #8)
Question 8 _____________   (15 points)  (ABET #1)

TOTAL    ______________
0. What’s your plan(s) after you graduate?

1a. IEEE Floating point number representation. Make sure your answer has the correct number of digits. Represent -65.375 in 32 bit IEEE single precision floating-point format. Show your answer in hexadecimal.

1b. Multiply the integer -10110 by -2510 using 2's complement. Use 8 bits. You may express your answer in binary, but it must have 8 bits. Show work for reporting/checking for overflow.

2. Consider the following instruction through the implementation on page 9 of this test.

<table>
<thead>
<tr>
<th>Address of instr.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF040 0000</td>
<td>slti $t3, $t1, -56</td>
</tr>
</tbody>
</table>

After the instruction is decoded:

- What are Instruction bits 25-21 __ __ __ __ __
- What are Instruction bits 20-16 __ __ __ __ __
- What is ALUSrc __ (0, 1, or X < don't care)
- What is MemtoReg __ (0, 1, or X < don't care)
- What is RegDst __ (0, 1, or X < don't care)
- What is Branch __ (0, 1, or X < don't care)

Consider the following instruction through the implementation on page 7 of this test.

<table>
<thead>
<tr>
<th>Address of instr.</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF040 0000</td>
<td>slti $t3, $t1, -56</td>
</tr>
</tbody>
</table>

- What is ALUSrcA during cycle 1 __ (0, 1, or X < don't care)
- What is ALUSrcA during cycle 2 __ (0, 1, or X < don't care)
- What is ALUSrcA during cycle 3 __ (0, 1, or X < don't care)
3. 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.

```
| IF | ID | EX | MA | WB |
```

lw $3, 64($4)

beq $3, $4, OUT

add $7, $3, $3

jr $7

4. The CPI of a machine with a perfect cache is 1.8 for a certain benchmark.

Now, assume a L1 instruction cache miss rate of 8% and a L1 data cache miss rate of 21%. Assume the miss penalty (for either L1 caches) is 8 cycles. The hit time is 2 cycle (L1 cache). Assume a L2 instruction cache miss rate of 4% and a L2 data cache miss rate of 9%. Assume the miss penalty (for either L2 caches) is 800 cycles. The hit time is 20 cycles (L2 cache).

The frequency of data access instructions in this benchmark is 20%.

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.
5. Assume we have a 32 KB cache fully-associative cache. (KB = 2^10 bytes)
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.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

6.

CACHE DYNAMICS: 16 bytes (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>CACHE CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Address</td>
</tr>
<tr>
<td>Cache Index</td>
</tr>
<tr>
<td>HIT/MISS</td>
</tr>
<tr>
<td>DIRTY? 0 1 DIRTY?</td>
</tr>
<tr>
<td>2 3</td>
</tr>
<tr>
<td>IME</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7. Assume a 48-bit virtual address and a 32-bit physical address. The page size is 8 KB. How many entries are there in the page table? Express your answer in powers of 2.
Show your work for this problem. (KB = 2^10 bytes)

8. You have 1 Billion inverters on your microprocessor (assume 2 transistors/inverter, or 2B transistors/die). Each inverter has an input capacitance of 1fF (1e-15 F), a leakage energy of I(leak)=10nA (1e-8 A), and all inverters are switched every cycle. (This is a simplification, because clearly we want to do logic beyond just inverter switching. However, assume just inverters switching every cycle). Assume that Vdd=1V, activity factor is 100% (since all inverters are switching every cycle), and clock frequency is f=1GHz.

A) What is the power consumption, for both 100% switching (fully active) and 0% switching (idle mode)? (Note: switching factor \( \alpha = \frac{1}{2} \) due to all inverters switching every cycle in fully active).

B) You have written 10e9 lines of code, that are 100% parallelizable. Assume you can now break up your 1B inverters into 1000 parallel cores (aka CPUs) of 1M inverters, running at Vdd=0.5V with clock frequency of f=1MHz. What is the total energy dissipated from the battery for the nominal 1-core CPU (f=1GHz, Vdd=1V, cores=1) and the 1000-core multicore CPU (f=1MHz, Vdd=0.5V, cores=1000)?
What is the fraction improvement in energy savings for doing the multicore approach?

Note: Ignore leakage for this portion, and assume %100 parallelizable code.
Energy-Consumed = Power * Time; V = I*R; Power(static) = I*V; Power(dynamic) = lecture notes