ECE 411 Spring 2012, Exam 1

- This exam has 6 problems. Make sure you have a complete exam before you begin.
- Write your name on every page in case pages become separated during grading.
- You will have three hours to complete this exam.
- Write all of your answers on the exam itself. If you need more space to answer a given problem, continue on the back of the page, but clearly indicate that you have done so.
- This exam is closed-book. You may use a calculator.
- DO NOT do anything that might be perceived as cheating. The minimum penalty will be a grade of zero.
- Show all of your work on all problems. Correct answers that do not include work demonstrating how they were generated may not receive full credit, and answers that show no work cannot receive partial credit.

- The exam is meant to test your understanding. Ample time has been provided. So, be patient. Read the questions/problems carefully before you answer.

- **Good luck!**

Problem 1(6): ___________
Problem 2(6): ___________
Problem 3(8): ___________
Problem 4(12): ___________
Problem 5(10): ___________
Problem 6(10): ___________
Total: (52): ___________
1 MP1 (6 points)

a) Consider the multicycle design specified in MP1. With no changes to the datapath, could states ‘IF3’ and ‘Decode’ in the control unit be combined (correctness must be maintained)? Why, or why not? (2 points)

Solution: No, because IR needs to get its new value before moving onto decode state. If combined, decode will use the old value of IR from the last instruction.

b) Use LC3b instructions to implement the Fibonacci function: \( f(n) = f(n-1) + f(n-2) \), where \( f(0) = 0, f(1) = 1 \). Your program must be iterative. You should specify the \( n \) as a number (e.g., 5) in the data segment such that it can be easily extended to arbitrary number. (2 points)

Solution: Assume that all the registers’ values are set to zero initially.

```
LDR R1, R0, num       ; any arbitrary number n
LDR R2, R0, negone    ; for decrement
ADD R3, R1, R2        ; R3 = num - 1
BRnz STORE            ; for n = 0 or 1
LDR R4, R0, zero      ; f(0) = 0 -> not necessary
LDR R5, R0, one       ; f(1) = 1

LOOP: ADD R1, R5, R4   ; f(n) = f(n-1) + f(n-2)
ADD R3, R3, R2        ; decrementing counter
BRz STORE             ; finished computation
ADD R4, R5, R0        ; f(n-2) <- f(n-1)
ADD R5, R1, R0        ; f(n-1) <- f(n)
BRnzp LOOP            ; iterate

STORE: STR R1, R0, result    ; store the result

HALT: BRnzp HALT  ; EOP - infinite loop
```

zero: DATA2 4x0000
one: DATA2 4x0001
negone: DATA2 4xFFFF
num: DATA2 4x0005 ; any arbitrary number
result: DATA2 ?
c) The below datapath and control state diagrams are from MP1 design. On the datapath, highlight all of the paths excited when a BR is executed (from state IF1 until just before the next instruction is fetched). Calculate the critical path of the BR instruction. (2 points)

Delay values
Clock cycle - 20ns
Mux (2 input) - 2ns
Regfile - 8ns
Reg16 - 4ns
IR - 5ns
ALU - 4ns
All other components - 1ns

Solution: Critical path = ADJ9 + BRadd + PCMux + PC(Reg16) = 1ns + 1ns + 2ns + 4ns = 8ns
Note that in this problem, critical path is the longest path within the one clock cycle which would be one state in the control state machine.
2 MP2.1 (6 points)

Outtel, the world’s largest semiconductor chip manufacturer, is hiring you to create the LC3b-86, a complex and expensive variant of your MP2.1 design. They would like you to add support a PUSH instruction:

**Syntax:**

```
PUSH SR
```

**Instruction format:**

```
[ opcode ] [ SR ] [xxxxxxxxx]
```

**Instruction operation:**

```
MEM[R6] <= SR;
R6 <= R6 + 2;
```

The stack pointer is always assumed to be in R6.

a) What changes must be made to the MP1 datapath in order to support the PUSH instruction? Be specific. (3 points)

**Solution:**

- Add constant ‘6’ as possible input to RFA port of regfile (expand mux)
- Add constant ‘6’ as possible input to DEST port of regfile (add mux)
  - *These could be combined by adding the mux on DEST between DEST and StoreMUX.*
- Add constant ‘2’ as possible input to ALU input B (expand mux).

b) Show the state machine for PUSH, starting at the DECODE state. For each state, show all necessary control signals (state actions) and transition conditions. (3 points)

![State Machine Diagram]

- **PUSH 1**
  - StoreMuxSel <= ‘10’ -- select R6
  - ALUop <= pass -- send R6 thru ALU
  - MARMuxSel <= ‘1’ -- R6 to MAR
  - LdMAR <= ‘1’ -- Save MAR

- **PUSH 2**
  - DestMuxSel <= ‘0’ -- Select ‘dest’ = SR
  - StoreMuxSel <= ‘00’ -- select RFA = SR
  - ALUop <= pass -- send R6 thru ALU
  - MDRAMuxSel <= ‘1’ -- R6 to MAR
  - LdMDR <= ‘1’ -- Save MAR

- **PUSH 3**
  - MWRITE <= ‘0’ -- MEM[R6] <= SR

- **PUSH 4**
  - DestMuxSel <= ‘1’ -- Select ‘dest’ = R6
  - StoreMuxSel <= ‘10’ -- select RFA = R6
  - ALUop <= op_add -- R6+= 2
  - ALUOpSel <= ‘10’ -- sel ‘2’ at RFBmux
  - RFMuxSel <= ‘1’ -- Sel ALU to RF
  - RegWrite <= ‘1’ -- save into R6

MRESP_H = ‘1’
3 Performance (8 points)

a) Two compilers generate instructions of several classes:

A - 1 cycle  
B - 2 cycles  
C - 3 cycles  
D - 5 cycles  
E - 10 cycles

Compiler one generates machine code for a program using:
- 5 million A, 7 million B, 4 million C, 1 million D, and none of class E.

Compiler two generates machine code for a program using:
- 3 million A, 2 million B, 1 million C, 1 million D, and 2 million of class E.

i) Which takes longer in terms of execution time? (2 points)

Solution: Compiler 1 takes 36 million cycles  
 Compiler 2 takes 35 million cycles  
 Therefore, Compiler 1 takes longer

ii) Which executes faster according to MIPS? Assume 1GHz clock speed. (1 point)

Solution: Compiler 1 executes \((5+7+4+1)/36 \times 10^{-9} = 470\) MIPS  
 Compiler 2 executes \((3+2+1+1+2)/35 \times 10^{-9} = 257\) MIPS  
 Therefore, compiler 1 is faster according to MIPS

b) Consider a multicycle design with 5 stages (fetch, decode, memory, execute, writeback). Each instruction must pass through each stage; there is no cache. A stage with no memory access takes 1 cycle (decode, execute, writeback); a stage with a memory access takes 10 cycles (fetch, memory). The memory access delay is avoided in the memory stage if the instruction is not a load or store. Assume 25% of instructions are loads and stores.

i) What is the CPI of this machine? (3 points)

Solution: \(CPI = 0.75(10+3+1) + 0.25(10+3+10) = 16.25\)

ii) A cache with a 90% hit rate (1-cycle hit) is added to the machine (services both fetch and memory stages). What is the speedup over (i)? (2 points)

Solution: \(0.25(0.9*5+0.1*23) + 0.75(0.9*5+0.1*14) = 6.125\)  
\(\text{Speedup} = \frac{\text{old\_CPI}}{\text{new\_CPI}} = \frac{16.25}{6.125} = 2.65\)
4 Single-Cycle vs Multi-Cycle (12 points)

Consider a single-cycle processor with a 5-instruction ISA. One instruction requires 6ns of work to complete; the other four instructions require only 3ns of work to complete. Assume that these 5 instructions are uniformly distributed in code executing on this processor. Additionally, assume a 0.1ns setup and 0.1ns hold time.

i) What is its clock speed (single cycle machine)? (2 points)

Solution: 6.2ns

ii) Suppose the single cycle processor is divided into a multi-cycle processor with six one-cycle stages, each performing 1ns of work. What is the clock speed of this processor? Hint: One instruction requires 6 cycles to complete; the other 4 instructions complete in 3 cycles. (2 points)

Solution: 1.2ns

iii) What is the CPI for the single cycle processor? (2 points)

Solution: 1

iv) What is the CPI for the multi cycle processor? (2 points)

Solution: 6*1/5 + 3*4/5 = 18/5 = 3.6 CPI

v) Which is faster? (1 points)

Solution: Single cycle takes 1*6.2ns = 6.2ns
Multicycle takes 3.6 * 1.2 = 4.32ns

vi) When does the setup+hold overhead cause speed of the multicycle design to be equal to the speed of the single cycle design? (3 points)

Solution: 1*(6+x) = 3.6*(1+x)
2.4 = 2.6x
x = 2.4/2.6 = 0.923
Therefore, Setup + hold = 0.923ns
5 Memory Hierarchy (10 points)

a) You are building a computer system with in-order execution that runs at 1GHz and has a CPI
of 1, with no memory accesses.

The memory system is split L1 cache. Both the I-cache and the D-cache are direct mapped
and hold 32KB each with block size 64 bytes. The I-cache has a 2% miss rate, and the D-
cache is a write-through with 5% miss rate. The hit time for both the I-cache and the D-cache
is 1ns.

The L2 cache is a unified write-back with a total size of 512KB and a block size of 64
bytes. The hit time of the L2 cache is 15ns. L2 write miss takes 15ns. The local hit rate of the
L2 cache is 80%. The 64-bit wide main memory has an access latency of 20ns, after which
any number of bus words may be transferred at the rate of one bus word (64-bit) per bus
cycle on the 64-bit wide 100MHz main memory bus.

Compute the CPI considering memory accesses. (6 points)

Solution: Assume a L2-miss will cause the memory transfers to both of L1 and L2.
Cycle time to transfer one bus word (64b) = 1/100MHz = 10ns
Time to transfer memory block to cache way (64B): 64B/64b * 10ns = 80ns

For Instruction:
\[
AMAT = 98\% \times 1\text{ns} + 2\% \times (80\% \times 15\text{ns}) + 20\% \times (20\text{ns} + 80\text{ns})
\]
\[= 1.62\text{ ns}\]

For data:
\[
AMAT = 95\% \times 1\text{ns} + 5\% \times (80\% \times 15\text{ns}) + 20\% \times (15\text{ns} + 20\text{ns} + 80\text{ns})
\]
\[= 2.7\text{ ns}\]

Overall AMAT = 1.62 + 2.7 = 4.32ns

Mistakes that should be avoided:
- Some students take 64b as 64B
- Some students write 2% as 0.2
b) A program is run on the computer containing the cache that performs computations on an 8KB array of integers (32-bit values). After the program is run, it is found that a total of 64 cache misses occurred in the data cache. Explain what caused those cache misses. If changes to the cache’s configuration could reduce the number of misses that occur when the program is run, suggest them. (2 points)

Solution: Assume we are using cache in part (a).
Note that we need 8KB/64B = 128 compulsory misses to load the data into the cache. However, there are only 64 misses occurred. Since the 512KB cache is direct-mapped and the 8KB array is contiguous, the cache is capable to hold the whole array and these misses cannot be conflict misses. Hence, there is no way to avoid them unless the block size is increased.

c) A second program is run on the computer containing the cache. This program performs computations on 8 data structures, each 1KB in length. After the program is run, it is found that 1,327,485 cache misses occurred in the data cache. What does this large number of misses imply about the organization of the data cache? How could the program be changed to reduce the number of misses seen when it executes? (2 points)

Solution: There are a total of 8KB data, while there are a huge number of misses. We already know that the compulsory misses should not be so many. This implies the misses are most likely to be conflict misses. This is due to the direct-mapped cache organization as given in part (a). Possible changes include:

- Improve the spatial locality by reorganizing the data, such as putting them in a contiguous array.
- Improve the temporal locality by reordering the data access sequence, such that blocks of data can be processed one before another.
6 Anatomy of Caches (10 points)

a) Consider a 4-way set associative, byte addressable cache. Addresses 0x0000000, 0x0004000, and increasing multiples of 0x0004000 (0x0008000, …) are the only locations that map to set 0. If the line size is 128B, what is the total size of the cache? (2 points)

Solution:
- 7 bits for index, 128 bytes cache line, 4 ways.
- Therefore Size = 4ways * 2^7 lines * 128 bytes/line = 64 KB

b) A byte-addressable computer has a small data cache capable of holding four 32-bit words. When a given program is executed, the processor reads data from the following sequence of hex addresses: 200, 204, 208, 20C, 2F4, 2F0, 200, 204, 218, 21C, 24C, 2F4. This pattern is repeated four times. (6 points)

i) Show the contents of the cache at the end of each pass throughout this loop if a direct-mapped cache is used. Compute the hit rate for this example. Assume that the cache is initially empty.

Solution:

<table>
<thead>
<tr>
<th>Index 0</th>
<th>Pass 1</th>
<th>Pass 2</th>
<th>Pass 3</th>
<th>Pass 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index 1</td>
<td>x200</td>
<td>x200</td>
<td>x200</td>
<td>x200</td>
</tr>
<tr>
<td>Index 2</td>
<td>x2F4</td>
<td>x2F4</td>
<td>x2F4</td>
<td>x2F4</td>
</tr>
<tr>
<td>Index 3</td>
<td>x218</td>
<td>x218</td>
<td>x218</td>
<td>x218</td>
</tr>
<tr>
<td>Index 4</td>
<td>x2FC</td>
<td>x2FC</td>
<td>x2FC</td>
<td>x2FC</td>
</tr>
</tbody>
</table>

First pass no hits. Second, third and fourth pass have hit on 200 only once/pass. Therefore, hit ratio 3 / (12*4) = 3/48 = 1/16

ii) Repeat part i) for a four-way set-associative cache that uses the LRU replacement algorithm.

Note: cache size here doesn’t change. Therefore cache becomes 4-way set associative.

Solution: No hits. Hit ratio = 0
<table>
<thead>
<tr>
<th></th>
<th>Way 0</th>
<th>Way 1</th>
<th>Way 2</th>
<th>Way 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass 1</td>
<td>x218</td>
<td>x21C</td>
<td>x24C</td>
<td>x2F4</td>
</tr>
<tr>
<td>Pass 2</td>
<td>x218</td>
<td>x21C</td>
<td>x24C</td>
<td>x2F4</td>
</tr>
<tr>
<td>Pass 3</td>
<td>x218</td>
<td>x21C</td>
<td>x24C</td>
<td>x2F4</td>
</tr>
<tr>
<td>Pass 4</td>
<td>x218</td>
<td>x21C</td>
<td>x24C</td>
<td>x2F4</td>
</tr>
</tbody>
</table>

(c) What is one advantage of a physically tagged, virtually indexed cache over a virtually tagged, virtually indexed cache? How about over a physically tagged, physically indexed cache? (2 points)

Solution: Faster than physically tagged, physically index because no address translation needed. More secure than virtually tagged, virtually indexed because physical tag is used.