ECE411 Computer Organization and Design
Mid-Term Exam 1
10/11/2018

NetID:_____

Exam Guidelines:

1. This exam has **6 problems.** Make sure you have a complete exam before you begin **[20 pages + the cover page]**.
2. Write your name on every page in case pages become separated during grading.
3. You will have **3 hours** to complete this exam.
4. 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.
5. This exam is closed-book. You may use one sheet of notes. You may use a calculator.
6. **DO NOT** do anything that might be perceived as cheating. The minimum penalty will be a grade of zero.
7. 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.
8. The exam is meant to test your understanding. Ample time has been provided. So be patient and read the questions/problems carefully before you answer.
9. **Good luck!**

Problem	Points
1. 24 pt	
2. 22 pt	
3. 30 pt	
4. 30 pt	
5. 40 pt	
6. 22 pt	
Total: 168 pt	

Name:_____

Question 1: Processor Pipelining

Given the following 5 stage pipeline with WB->EX & MEM->EX forwarding and register file bypass logic

Fetch stage combinational delay = 4ns
Decode stage combinational delay = 5ns
Execute stage combinational delay = 7ns
Memory stage combinational delay = 5ns
Writeback stage combinational delay = 4ns

Code:

```
ld   r1, DATA0 ; r1 <= M[DATA0]
andi r3, r1, 0 ; r3 <= r1 & 0x0
ld   r2, DATA1 ; r2 <= M[DATA1]
add  r3, r3, r1; r3 <= r3 & r1
add  r3, r3, r2; r3 <= r3 & r2

ld   r4, DATA2
andi r6, r4, 0
ld   r5, DATA3
add  r6, r6, r4
add  r6, r6, r5
```

Assume latch delay is 2ns. Assume the processor operates at maximum possible safe frequency. Assume 100% cache hit rates and branch prediction rates.

A. Calculate how long it will take (in ns) to run the given code. (6 points)

Name:_____

You now run the following code:

```
ld   r3, ADDR
add  r3, r1, r2
addi r5, r3, 0x10
```

B. You discover that this code produces the wrong value for r5 even after producing the correct result for the code at the beginning of the question. What might be a potential hardware bug in the pipeline that causes this error? (10 points)

C. You are asked to evaluate the benefits of adding an additional WB->MEM forwarding path to the pipeline. Write an example piece of assembly code that could benefit from this new forwarding path. (8 points)

Name:_____

Question 2: Caches

Consider an architecture with a 32-bit address range and byte addressable memory. In order to exploit locality, you as the logic designer choose to implement a cache with 64-byte cache lines. 'Cache Type' refers to Direct Mapped, Set-Associative or Fully Associative in this question.

A. Assume each set has 128 bytes of data. There are 47 bits of bookkeeping (tag, valid, dirty and LRU) overhead per set. How many bits are allocated per tag, index and offset? (6 points)

B. Replacement policies are a major contributor to how well each way is utilized and the hit rate of the cache. Least Recently Used is a straightforward replacement policy with two common flavors; Pseudo LRU and True LRU.

How many bits are needed per set for a tree-based pseudo LRU scheme for a 16-way set associative cache? What is the bit storage overhead of using a queue to track True LRU for n-ways where n > 2? (6 points)

C. Prefetching is a scheme devised to predict what data will be useful in the near future and bring it into the caches before the CPU needs to access it. This allows potentially useful memory accesses to be overlapped with execution of non-memory operations.

Assume that we can build a highly accurate prefetcher that will only issue prefetch accesses to data that will be used at some point during program execution. Provide a scenario where performance degrades. (4 points)

D. The program below shows nested for loops loading and operating on some data from the structure[] array. A stride is used to determine the address difference between each subsequent load. Given the following program, answer the listed questions.

```
int structure[...]; // a very large array
uint32_t stride = 128;
for (int j = 0; j < 32; j++) {
   stride = stride * 2;

   for (uint32_t i = 0; i < 16; i ++) {
        // Load an element from some structure
        int cur_element = structure[i*stride];

        // Operate on the cur_element
        ...
   }
}
```

Name:_____

You observe that the memory access times are increasing in the inner loop as the stride increases, which causes a decrease in performance. The memory access time saturates at very large strides. This behavior is caused by a near-zero hit rate in the data cache. What type of cache might be in the processor running the code above? Why? (6 points)

Name:_____

Question 3: Caches and Virtual Memory

Congratulations! you have accepted an offer from a new startup called "NoHope" as a HW architect in their processor design group. They are currently working on their cache design. The team leader explained their current preference - using virtually-indexed, virtually-tagged (VIVT) L1 instruction and data caches. After looking at their cache designs, you realized that they didn't handle any of the known issues associated with VIVT caches.

A. After hearing you explain the issues, your team leader got convinced and asked you to propose 2 software solutions each for a minimum of two VIVT issues. What would be your solutions? (8 points)

B. Unfortunately, the team leader worries about performance degradation and complexity typical of software solutions and asks you to suggest two alternatives (hardware only or hardware supported by software solution). What are you going to propose (caches still need to be VIVT)? (8 points)

Name:_____

C. You also found out that they are using an 8 KB virtually indexed, physically tagged (VIPT) cache as the L2 cache. However, they would now like to double the size of the L2 cache (since new benchmarks have emerged) and decide on the associativity. With the information given in the table below what is the associativity that you recommend and why? (6 points)

Virtual address size	Page size	Size of each page table entry
32 bits	4 KB	8 bytes

D. Frustrated by the poor design choices made by NoHope, you decide to interview with other processor design companies. During an interview at ALittleHope, you have been given the following information:

- A processor with a 32-bit virtual address space.
- A 64KB of physical memory space.
- The size of a page is 1KB.
- TLB is a direct-mapped cache with 4 entries.

You are then asked:

a. How many bits are stored for each TLB tag entry? (4 points)

b. How many bits are stored for each TLB data entry? (4 points)

Name:_____

Question 4: ISA

Consider the following ISAs and instruction latencies for some simple multicycle implementation (like your MP1 design) of each ISA (do not worry about cache misses and hits, assume AMAT is baked into the latency of all instructions that have memory accesses):

RegReg: a register-register ISA (assume 16 general purpose registers named r0-r15) (140MHz)

Assembly Instruction formats	Instruction latency (# of cycles)
load DR, label	8
store SR, label	8
add DR, SR1, SR2	6
sub DR, SR1, SR2	6
mul DR, SR1, SR2	10

Stack: a stack ISA (100MHz)

Assembly Instruction formats	Instruction latency (# of cycles)
push label	8
pop label	8
add	6
sub	6
mul	10

Name:_____

MemMem: a memory-memory ISA (140MHz) (label1 is always the destination)

Assembly Instruction formats	Instruction latency (# of cycles)
add label1, label2, label3	12
sub label1, label2, label3	12
mul label1, label2, label3	16
mov label1, label2	9

A) Write assembly code to execute the following C function in each ISA with no optimization: (12 pts)

```
/* global variables, no need to produce code for this part, just
assume the labels are available to use */
int A;
int B;
int C;
int tmp;
int result;

void foo() {
    result = result + (B*(C-A));
}
```

As always, write any assumptions you make.

RegReg:

Stack:

MemMem:

B) Imagine you want to add an atomic integer addition instruction to one of these ISAs. An atomic operation should write it's result to memory before any other thread has a chance to overwrite the memory operands of the instruction. Assume this is done by simply asserting a lock signal on the bus interface across multiple read/write operations (like the mem_read signal is asserted to do a read). This amendment would be easiest to implement with which ISA and why (6 pts)?

C) What do you think would be the latency of this new instruction in number of cycles and why (6 pts)?

D) Which ISA uses the instruction cache most efficiently, considering that labels take many bits to encode (much more than register operands) and that compilers must emit more instructions for some ISAs than others? Why? (6 pts)

Name:_____

Question 5: MPs

This question contains multiple parts and aims to assess your ability to understand a specification and implement desired functionality using good design practices.

Programmers that optimize their code for high performance often desire information about the utilization of the underlying hardware during the execution of their programs. In order to provide this information, modern processors include blocks which monitor the state of the machine and instructions which allow access to these state registers.

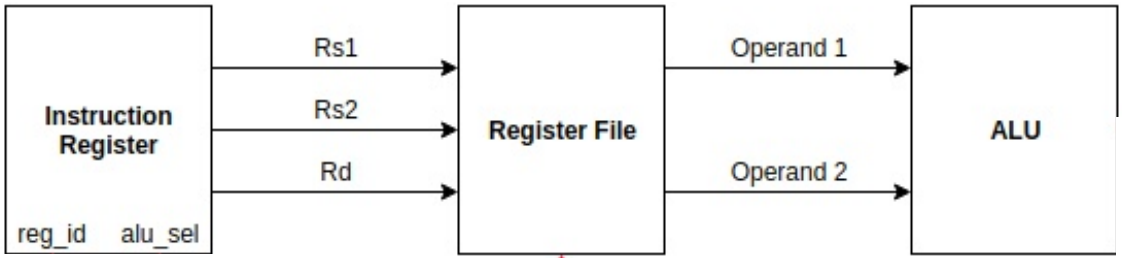
a. You are tasked with writing a module that tracks the number of instructions executed by your processor as well as the number of cycles spent waiting on memory accesses. You should write this code in the provided skeleton below. You are told that the *enable* signal will be high for exactly one cycle during the execution of each unique instruction. The memory signals, *mem_read*, *mem_write*, and *mem_resp* behave identically to the class MPs. The *rst* signal is **active low** and should set your counters to zero. (4 pts)

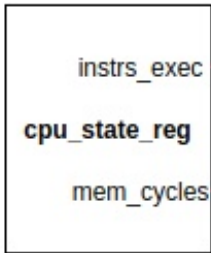
```
module cpu_state_reg (
        input logic clk, rst, enable, mem_read, mem_write, mem_resp,
        output logic[31:0] instrs_exec, mem_cycles
);
// Declare internal signals here


// Combinational logic
always_comb begin

end
// Sequential logic
always_ff @(posedge clk) begin




end
endmodule : cpu_state_reg
```

Name:_____

Now that your design can keep track of its state, you must add support to access these registers and allow a user to see the statistics of their program. The new instruction, *rfsr*, stands for "read from state register" and has the following format: *rfsr Rd, <reg_id>* . This instruction reads from the module you created above and writes the counter value into the destination register, *Rd*. For this problem, *<reg_id>* is a 1-bit signal and can be thought of as the "address" of the counter which should be read. Let *reg_id = 0* correspond to the instruction count and *reg_id = 1* correspond to the memory cycle count. In real processors, there are many different registers to choose from.

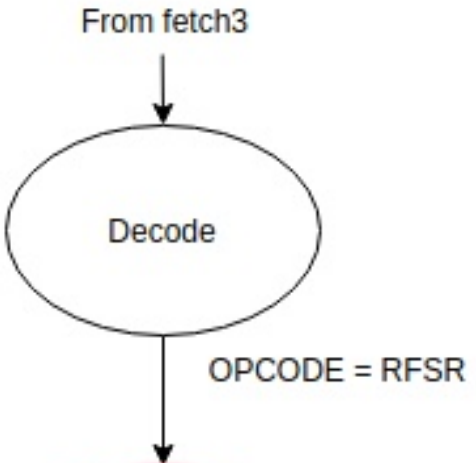
b. Modify the partial datapath below by filling in the missing logic required to support the new *rfsr* instruction. Include labels where needed. Write a sentence or two describing each of the changes you've made to the datapath. If you utilize a block other than logic gates or muxes, you must explain the function of the block. *Alu_sel* is a 1-bit signal which is high when an ALU operation is being performed and low otherwise. (5 pts)





Finally, we must consider the impact of this instruction on the CPU controller. For this machine, there are three fetch states and a single decode state. These states operate identically to those implemented in MP0 and MP1. Following the decode state, you decide to add new state(s) to handle the *rfsr* instruction.

c. Draw the new state(s) needed to support the the *rfsr* instruction. The decode state has been provided for you. For each state you draw, you should write a sentence or two to describe what is being accomplished. Feel free to use signals to describe the functionality of the state(s), just be sure the signal names match your datapath drawn above. (4 pts)



Name:_____

Following the release of the processor designed in the previous three parts, your team discovers that most users read both of the counters back-to-back. To support the same functionality with fewer instructions, your task is to create a new instruction which reads from both counters. You decide to call it *rfbr* or "read from both registers". This instruction has the following format: *rfbr Rd*. Functionally, the instruction will place the value of *instrs_exec* into Rd and will then place the value of *mem_cycles* into the register **immediately following** Rd. For example, *rfbr x1* should write values into *x1* and *x2*.

d. Summarize the differences between the implementation of the *rfsr* instruction and the *rfbr* instruction. At a high level, your answer should address the changes needed within the CPU datapath and the changes needed within the CPU controller. Feel free to describe the changes in English, with diagrams, or a combination of the two. (4 pts)

e. Processor design is primarily about trade-offs. With your previous answers in mind, write one advantage of the *rfbr* instruction and one disadvantage of the *rfbr* instruction when compared to the *rfsr* instruction. Clearly label your responses.(3 points)

Name:_____

Question 6: Potpourri:

a. Consider an ISA with three instructions (Inst1, Inst2, Inst3). Write an instruction sequence with 3 instructions that will have worse performance on a single cycle design compared to multi-cycle design? Why? Assume that inst1, inst2, and inst3 take 1, 2, and 3 cycle respectively on the multi-cycle design. Will this instruction sequence have higher performance on a 3-stage pipeline (vs single-cycle design)? Why? (8 points)

b. If D is dynamic power of a CMOS processor design, S is static power of the design, draw the D/S curve showing how it has changed over last 20 years. Make sure to mark 1 on y-axis. Make sure to mark 2018 on x-axis. (6 points)

c. Here's a comparison between STT-RAM, a memory technology based on magnetic spins and SRAM:

	Area (mm^2)	Read Energy (nJ)	Write Energy (nJ)	Leakage Power at (mW)	Read Latency (ns)	Write latency (ns)	Read @ 2 GHz (cycles)	Write @2 GHz (cycles)
1 MB SRAM	2.61	0.578	0.578	4542	1.012	1.012	2	2
4MB STT-RAM	3.00	1.035		2524	0.998		2	

Where would you put STT-RAM in the memory hierarchy? Why? (6 points)

d. If McDonalds is “I am loving it”, Skittles is “Taste the Rainbow”, Google is “Don’t be Evil”, ECE411 is

Name:_____

Appendix A:

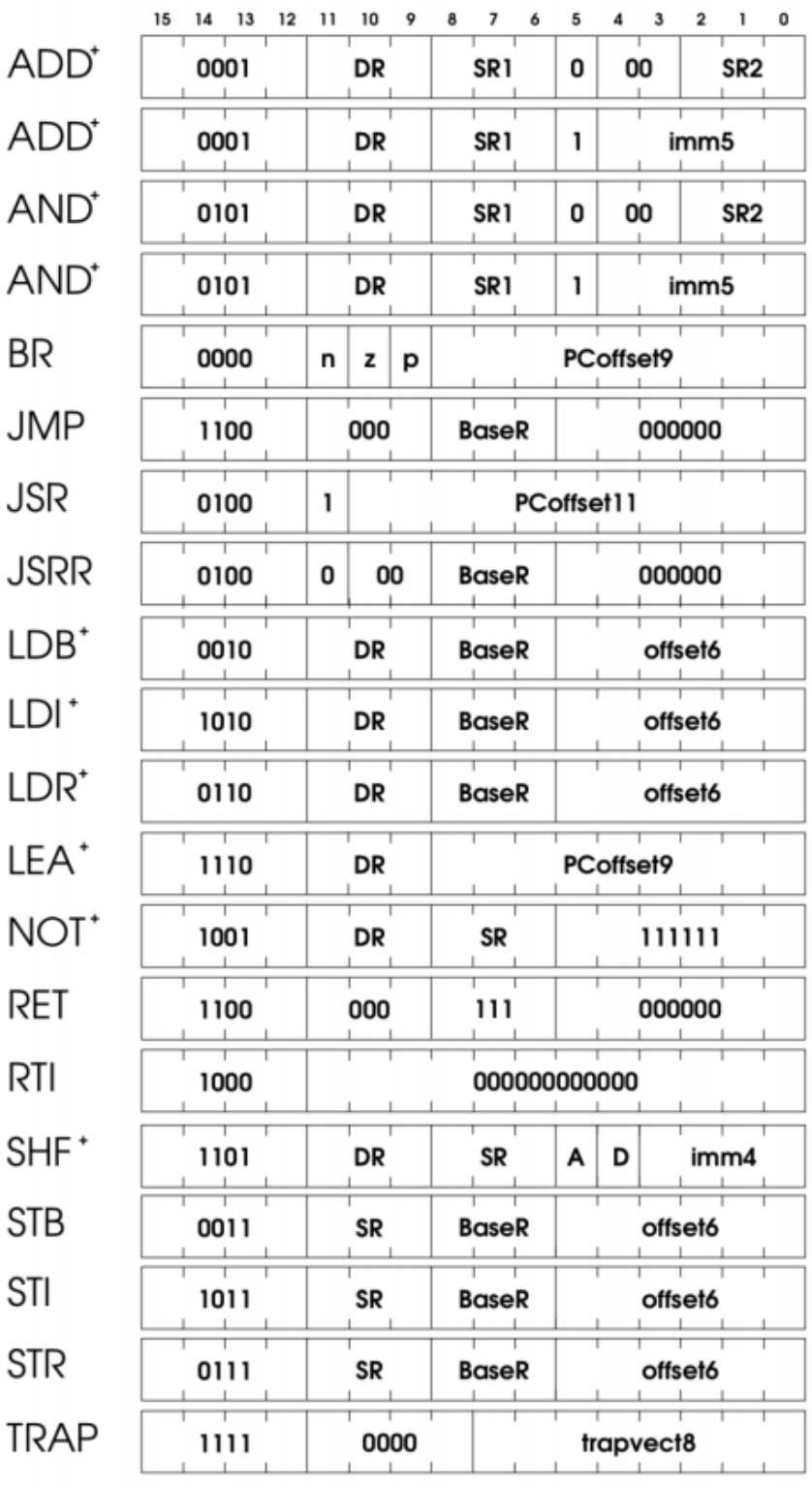


Figure 1.2: LC-3b Instruction Formats. NOTE: + indicates instructions that modify condition codes.

Name:_____

Appendix B:

31 27	26 25	24 20	19 15	14 12	11 7	6 0	
funct7		rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]			rs1	funct3	rd	opcode	I-type
imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]					rd	opcode	U-type
imm[20|10:1|11|19:12]					rd	opcode	J-type

RV32I Base Instruction Set

imm[31:12]			rd	0110111	LUI
imm[31:12]			rd	0010111	AUIPC
imm[20|10:1|11|19:12]			rd	1101111	JAL
imm[11:0]	rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND