Midterm

EECS 370 Winter 2025: Introduction to Computer Organization

You are to abide by the University of Michigan College of Engineering Honor Code. Please sign below to signify that you have kept the honor code pledge:
I have neither given nor received aid on this exam, nor have I concealed any violations of the Honor Code.

Signature: _____

Name: _____

Uniqname: _____

Uniqname of person sitting to your ***Right***
(Write ⊥ if you are at the end of the row) _____

Uniqname of person sitting to your ***Left***
(Write ⊥ if you are at the end of the row) _____

Exam Directions:

- You have **120 minutes** to complete the exam. There are **9** questions in the exam on **14** pages (double-sided). **Please flip through your exam to ensure you have all the pages. Your answers must appear in the space provided for your answer. Answers on the blank pages will not be graded.**
- You must show your work to be eligible for partial credit!
- Write legibly and dark enough for the scanners to read your answers.
- **Write your uniqname at the bottom of each page.**

Exam Materials:

- You are allotted **one 8.5 x 11 double-sided** note sheet to bring into the exam room.
- You are allowed to use calculators that do not have an internet connection. All other electronic devices, such as cell phones or *anything* with an internet connection, are strictly forbidden.

This page intentionally left blank.

Do not put answers on this page.

This page intentionally left blank

1) Multiple Choice 16 points

Completely shade in the box with the best answer. Select only 1 answer per question. **[2 points each]**

1. The "gap" between representations of a 32-bit IEEE floating point scheme varies most directly with:
   - ☐ The computer manufacturer.
   - ☐ The value of the exponent.
   - ☐ The value of the mantissa.
   - ☐ The high-level language used.
   - ☐ Nothing: the gap does not vary.

2. The "gap" between representations of a 32-bit signed integer varies most directly with:
   - ☐ The computer manufacturer.
   - ☐ The value of the integer.
   - ☐ The sign of the integer.
   - ☐ The high-level language used.
   - ☐ Nothing: the gap does not vary.

3. Executing which of the following lines of LC2K assembly sets every bit of register 1 to 0, assuming register 0 is a zero, register 1 has an initial value of 5, and memory address 0 has a value of 0x3.
   - ☐ `nor   1 1 1`
   - ☐ `nor   0 0 1`
   - ☐ `lw    0 1 0`
   - ☐ `.fill 1`
   - ☐ `add   1 0 0`

4. If a global variable is declared in a source file named file1.c, do accesses to it in file1.c need to go in the relocation table when linking?
   - ☐ **No**, because the global variable's location is declared in this file, there is no need to relocate it.
   - ☐ **No**, because the global variable will be in the symbol table for this file so relocation can happen before linking.
   - ☐ **Yes**, because even though the global variable is defined in the same file as the reference, we're not sure where the global variable will be actually placed in memory until linking is done.
   - ☐ **Yes**, because we will need to create a list of files in which the global variable is used, even if it's the one where the global variable is declared.
   - ☐ **Maybe**, because the reference to a global variable can be done with PC-relative addressing in which case it need not be in the relocation table.

5. Consider a Moore-type state machine with 4 bits of input, 3 bits of output and 3 states. How large would your ROM need to be?
   - ☐ 128 bits
   - ☐ 256 bits
   - ☐ 300 bits
   - ☐ 320 bits
   - ☐ 512 bits

6. Which of the following is true?
   I. Dennard scaling is the notion that as transistors shrink, the power to use them will shrink at about the same rate.
   II. Moore's law is the notion that power density is expected to stay constant as transistors shrink every 18 months or so.
   III. Modern computers have heat problems because Moore's law has not held for at least the last decade, maybe longer.
   - ☐ Only I
   - ☐ Only III
   - ☐ Only I and II
   - ☐ Only II and III
   - ☐ I, II, and III

7. The range of representation of an 8-bit 2's complement number is:
   - ☐ -128 to 128
   - ☐ -256 to 256
   - ☐ -255 to 256
   - ☐ -256 to 255
   - ☐ -128 to 127

8. Which of the following is true?
   I. All else being equal, a single-cycle processor is expected to have a higher clock period than a multi-cycle processor.
   II. All else being equal, a single-cycle processor is expected to have a higher clock period than a pipelined processor.
   III. All else being equal, a multi-cycle processor is expected to have a higher CPI than a single-cycle processor.
   - ☐ Only I
   - ☐ Only III
   - ☐ Only I and II
   - ☐ Only II and III
   - ☐ I, II, and III

2) Short Answer 12 points

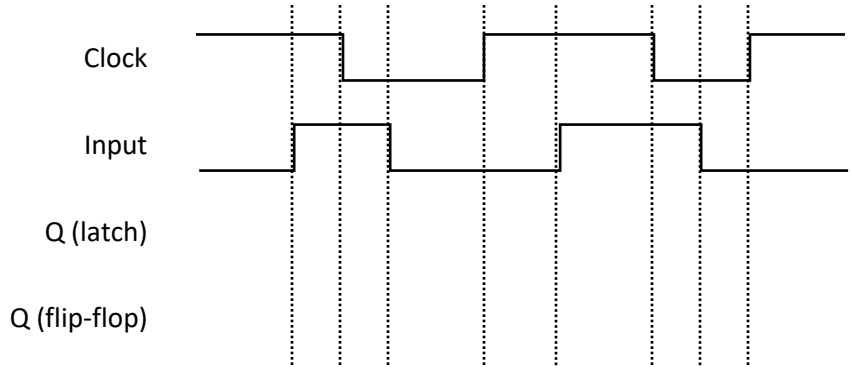
Answer the following questions.

1. Convert each of the following numbers into 8-bit two's complement. **[4]**

   i. -12 (decimal): _____

   ii. 33 (decimal): _____

2. Complete the timing diagrams for the D latch and D flip flop. The flip flop is rising edge triggered. The "input" is the D input and the clock is the gate for the latch (which is fairly common terminology). You are to assume both devices have an initial value of "0". **[4]**



3. Say you have a 4-bit *signed* number X[3:0] (so X3 is the most significant bit, X2 is the next most significant, etc.). Using only standard gates (AND, OR, NOT), and constant 0s and 1s, multiply X[3:0] by 4 and output it as a 7-bit *signed* number Y[6:0]. Use as few gates as possible, even if it means you use no gates. **[4]**

3) Slightly longer questions 15 points

1. Show the final value of the memory and registers listed after the following LEGv8 code runs. You may assume that all registers and any memory value not shown is initialized to be zero. Assume we are in little endian mode. **Put all answers in hex. [8]**

```
LDUR        X3, [X5, #80]
STURB       X3, [X5, #81]
STURH       X3, [X5, #82]
LDURSW      X4, [X5, #80]
```

X3: _____

X4: _____

Memory location	Initial value	Final value (if changed)
80	0x49	
81	0x82	
82	0xEE	
83	0x55	
84	0x44	
85	0x33	
86	0xAA	
87	0xBB	

2. For the following structure, the starting address is 1000 (***decimal***). Write the corresponding range for each element of the struct as it would be laid out in memory. This code is being compiled for a **64-bit operating system.** We've completed a small part of this for you. **[7]**

1	`struct game {`		
2	`  char euro;`	1000 - 1000	
3	`  char *gameName;`	_____ - _____	
4	`  struct {`		
5	`  char ameri;`	_____ - _____	
6	`  short id;`	_____ - _____	
7	`  double bggRating;`	_____ - _____	
8	`  char name [13];`	_____ - _____	
9	`  } Game [10];`	_____ - _____	Game[0]
10		_____ - _____	Full array of Game
11			
12	`  int total;`	_____ - _____	
13	`  char new;`	_____ - _____	
14	`};`	1000 - _____	Full struct game

4) LC2K object files 8 points

For project 2a you were to generate an object file for LC2K. Below is an assembly program and a partially complete object file. You are to fill in all the missing lines (or partial lines) of the object file (don't forget the first line) in a way that meets the project 2a specification. It is possible that one or more lines should remain blank, be sure you only have blank lines where intended.

Main.as	Main.obj
`Main lw 0 3 Add`	**0x00830006**
`bob lw 0 2 bob`	**0x00820001**
`add 2 3 4`	**0x00130004**
`beq 4 0 Main`	**0x0120_____ ← Finish this line**
`sw 0 3 AAA`	
`halt`	**0x01800000**
`Add .fill 7`	**0x00000007**
`bet .fill Main`	**0x00000000**
`Dull .fill bob`	**0x00000001**
	Main T 0
	0 lw Add

5) Taking All Factors into Account 15 points

Consider the recursive and iterative versions of a factorial function as well as the assembly programs “Code A” and “Code B”. The questions and space for your answers are on the following page.

```
int factorial(int n) {
    if (n == 0 || n == 1)
        return 1;

    return n*factorial(n-1);
}
```

```
int factorial(int n) {
    if (n == 0 || n == 1)
        return 1;
    int ret = 1;
    while (n > 0) {
        ret = n*ret;
        n--;
    }
    return ret;
}
```

Code A

```
factorial:
        MOV     X3, X0
        ADDI    X0, XZR, #1
        CMPI    X3, #1
        B.LE    BOB
TOM:    MUL     X0, X3, X0
        SUBS    X3, X3, #1
        B.NE    TOM
BOB:    BR      LR
```

Code B

```
factorial:
        CMPI    X0, #1
        B.LE    MARY
        PUSH    {X20, LR}
        MOV     X20, X0
        SUBS    X0, X0, #1
        BL      factorial
        MUL     X0, X20, X0
        POP     {X20, PC}
MARY:   ADDI    X0, XZR, #1
        BR      LR
```

Note the following information about the assembly code provided:

1. The assembly language programs provided are in ARMv8. They use a few extra instructions in addition to those in the LEGv8 subset. Specific information about these instructions is provided below.
2. The `PUSH {<rx>, <ry>}` instruction pushes the register identified by `<ry>` to the stack, and then pushes the register identified by `<rx>` to the stack. It also modifies the stack pointer register to point to the top of the new stack.
3. The `POP {<rx>, <ry>}` instruction pops the 64 bits at the top of the stack into the register identified by `<rx>`, and then pops the next 64 bits on the stack into the register identified by `<ry>`. It also modifies the stack pointer register to point to the top of the new stack.
4. The `MUL <rx>, <ry>, <rz>` instruction multiplies `<ry>` and `<rz>` and puts the result into `<rx>`.
5. The `PC` register identifier corresponds to the PC.
6. The `LR` register identifier is a synonym for register X30.
7. Note that the `POP` instruction can modify the PC.

1. Which of “Code A” or “Code B” corresponds to the recursive factorial program? **[1]**

   ☐ Code A ☐ Code B

2. For Code A, is the argument to the factorial function being passed via memory or a register? If you select register, indicate which register is being used in the blank. **[1]**

   ☐ Memory ☐ Register _____

3. For Code B, is the return value from function being passed via memory or a register? If you select register, indicate which register is being used in the blank. **[1]**

   ☐ Memory ☐ Register _____

4. In Code B, is X20 being used in the program as a caller or callee save register? How do you know? Your answer should be based on how the register is used in the program. **[3]**

5. If computing factorial(100), which version of the assembly code will execute more loads from memory? You must correctly justify your answer to receive credit. **[3]**

6. If we assume that the CPI and clock period are the same for both programs, which version of the assembly code will run faster if computing `factorial(100)`? You must correctly justify your answer to receive credit. **[3]**

7. Why does code B need to push the link register onto the stack near the beginning of the code and pop it into the PC at the end of the code? **[3]**

6) Multi-cycle performance 6 points

You are designing a new processor and are considering whether or not to include the MAC instruction. With the MAC instruction, your customer's workload has the following mix of instructions:

Instruction	Percent of Workload
R-type	40%
lw	20%
sw	10%
beq	15%
MAC	15%

Your competitor's processor does not have the MAC instruction. Instead, they emulate the MAC instruction using three instructions (a lw, a sw, and an add (R-type)). The following table specifies how many cycles each instruction takes on the two processors:

Instruction	Number of cycles on your processor	Number of cycles on the Competitor's Processor
R-type	2	2
lw	5	5
sw	3	3
beq	4	4
MAC	5	Not implemented

If your competitor's clock period is 10ns, what clock period would your implementation have to achieve to achieve the same performance on the customer's benchmark? Place your answer (rounding to exactly two digits after any decimal point) in the blank and clearly show your work.

_____**ns**

7) Pipelining basics 5 points

For this problem consider the LC2K code segment below. Notice that there are no hazards. Assume the add instruction is in the fetch stage in cycle 0.

```
add     3     2     3
lw      0     7     5
nor     2     4     1
sw      0     5     5
halt
.fill   10
```

Register	Value
0	0
1	-1
2	3
3	10
4	2
5	0
6	10
7	12

Fill in the contents of the pipeline registers at the point in time between cycle 4 and 5. Place an "??" if the value is **unknown**, otherwise put the correct value.

Cycle 4-5

IF/ID	
Opcode:	
PC Plus 1:	

ID/EX	
Opcode:	
PC Plus 1:	
regA val:	
regB val:	
offset:	

EX/MEM	
Opcode:	
aluResult:	
regB val:	

MEM/WB	
Opcode:	
writeData:	

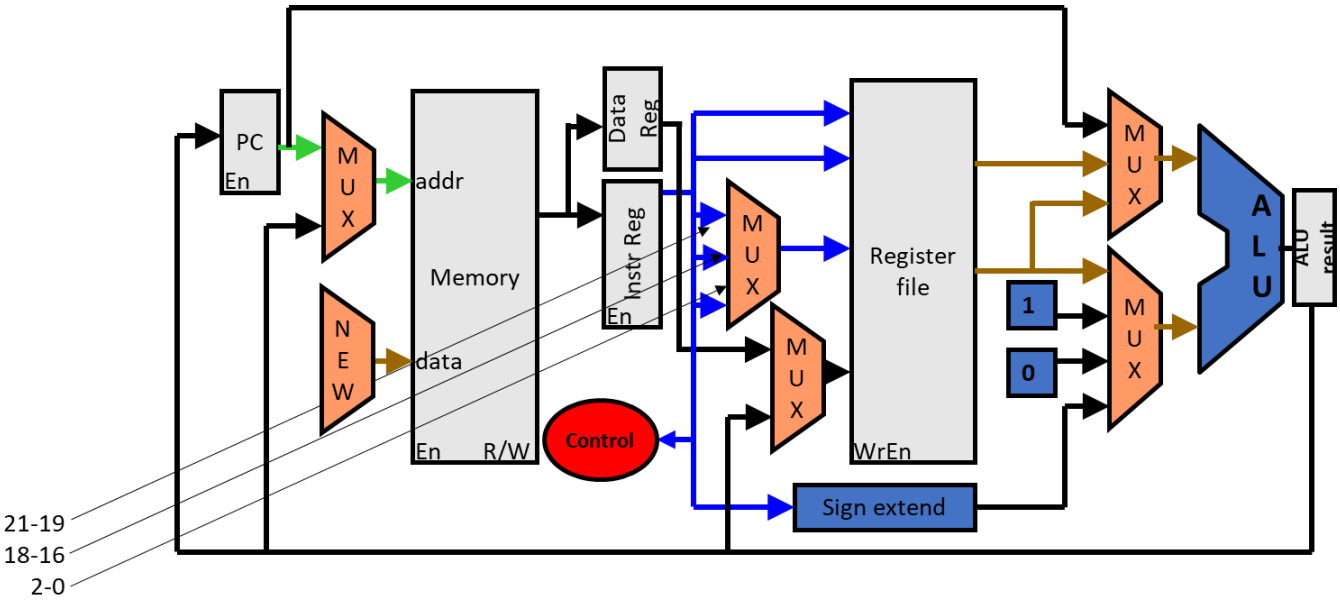
8) Multi-cycle design 15 points

We want to provide hardware support for a new instruction named “copy”: copy performs the following operation: **MEM[RegA+offset]=MEM[RegB+offset].** So if r1=4 and r2=7 this instruction:

```
copy 1 2 4
```

would cause address 11’s value to be copied to address 8.

1. Now, consider the data path below. Notice it is somewhat different than the data path discussed in class. One change has been the addition of a MUX named “NEW”. Connect signals to NEW as needed to enable the copy instruction to be executed while still allowing the old instructions to work correctly. **[3]**



2. Give a cycle by cycle description of the LC-2K operation when executing the new instruction. For each cycle, give the following information:
    - Single-sentence description of what the cycle is about.
    - Register updates.

    Use as few cycles as possible given your hardware. To get you going, we have provided the first 2 cycles. **[12]**

Cycle 1	**Fetch instruction.** Instruction Register ← MEM[PC] ALU Result ← PC + 1
Cycle 2	**Decode instruction and read registers** PC ← ALU Result
Cycle 3	
Cycle 4	
Cycle 5	
Cycle 6	
Cycle 7	

9) Datapath 8 points

Consider the multi-cycle data path below (the same one that was used in HW2). Say you wish to do the following:

Cycle 1:

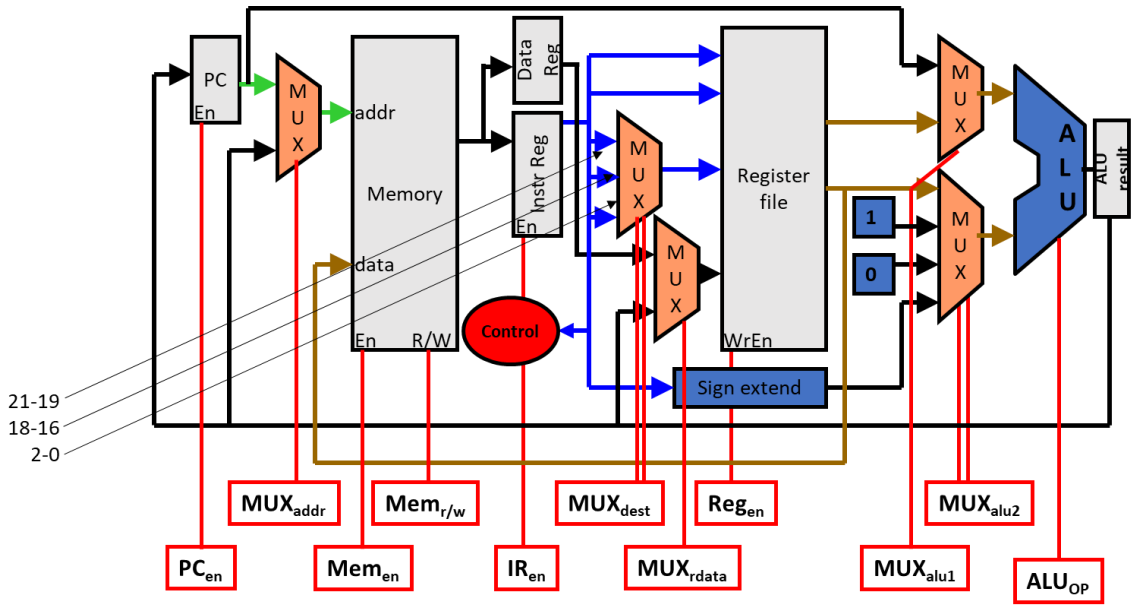
Data_register ← MEM[ALU_result]

Reg[RegA] ← Data_register

Cycle 2:

MEM[ALU_Result] ← Reg[RegB]

ALU_Result ← PC



Show what values you would need for each control signal in each cycle. If a value doesn't matter, you must indicate that with an X. Give your answers in binary.

	PC_{en}	MUX_{addr}	Mem_{en}	$Mem_{r/w}$	IR_{en}	MUX_{dest}	Reg_{en}	MUX_{rdata}	MUX_{alu1}	MUX_{alu2}	ALU_{OP}
Cycle 1											
Cycle 2											