CS/ECE 552: Introduction to Computer Architecture
Solution for Mock Midterm 2022

CLOSED BOOK
ONE 8.5"x11" SHEET OF NOTES (TWO-SIDED) ALLOWED

NAME: _____

DO NOT OPEN EXAM UNTIL TOLD TO DO SO!

Read over the entire exam before beginning. You should verify that your exam includes all of the problems listed in the table below as well as two spare pages for scratch work. Budget your time according to the weight of each question and your ability to answer them. I have provided a large amount of space for you to write your answers, but if that space is not sufficient, please write on the BACK of the SAME sheet. WRITE YOUR NAME ON EACH SHEET. You will turn in your cheat sheet with your exam, so please make sure your name is written on it.

Upon announcement of the end of the exam, stop writing on the exam paper immediately. Pass the exam to the head of the tables to be picked up by the proctor (the TAs and me). The instructor will announce when to leave the room. Failure to follow instructions may result in forfeiture of your exam and will be handled according to the UW Academic Misconduct policy.

Problem	Possible Points	Points
Problem 1	10	
Problem 2	10	
Problem 3	15	
Problem 4	30	
Problem 5	10	
Total	75	

Name: _____

Problem 1 [10 points]

Part A [5 point]

Given a program *P*, and a processor *Q*, you are given two choices for running P on Q (assume Q has support for both of these options):

- Option 1: Run *P* at 500 KHz, which provides a CPI of 2.25
- Option 2: Run *P* at 200 KHz, which provides a CPI of 1.5

Given these options, which is the better choice? To receive credit you must justify your answer (NOTE: you may assume K = 1000 or 1024 for your answer)

Solution:

We can apply the Iron Law of Performance here (note that the instructions/program is constant, since it's the same program):

$$Execution\ Time = \frac{instructions}{program} * \frac{cycles}{instruction} * \frac{seconds}{cycle}$$

$$Execution\ Time_{Option\ 1} = P * 2.25 * \frac{1}{5 * 1024 * 10^6} = P * 0.44\ ns$$

$$Execution\ Time_{Option\ 2} = P * 1.25 * \frac{1}{2 * 1024 * 10^6} = P * 0.61\ ns$$

Part B [5 points]

If a MIPS design has a buggy sign extension unit, name five instructions that will produce incorrect outcomes? Justify your answer in one line for each instruction. [4 points]

addi, andi, lui, slti, lbu, **etc.**

Name: _____

Problem 2 [10 points]

Part A [5 points]

Convert the following C code to its equivalent MIPS assembly code:

```
i= 0;
do {
        Z[i] = (A + X[i]) ^ Y[i];
        ++i;
} while (i < N);
```

Assume that $s0 initially holds X[0] (base address of X), $s1 initially holds Y[0], $s2 initially holds Z[0], $s3 initially holds i, $s4 initially holds N, and $s5 initially holds A. Moreover, assume that i has been initialized to 0, and that the arrays are all arrays of integers (4B per element) and have been properly allocated before this code snippet.

```
   XOR $s3, $s3, $s3;  // optional; initialize i = 0;
.Loop:
     0: LW   $t1, 0($s0);    // load X[i]
     1: ADD  $t2, $t1, $s5;  // X[i] + A
     2: LW   $t3, 0($s1);    // load Y[i]
     3: XOR  $t4, $t3, $t2;  // (A + X[i]) ^ Y[i]
     4: SW   $t4, 0($s2)     // store to Z[i]
     // now update the loop counters
     5: ADDI $s3, $s3, 1;    // ++i
     6: ADDI $s0, $s0, 4;    // update X[i] by 4B since array of ints
     7: ADDI $s1, $s1, 4;    // update Y[i] by 4B since array of ints
     8: ADDI $s2, $s2, 4;    // update Z[i] by 4B since array of ints
     9: BNE $s3, $s4, .Loop; // go back top of loop if i < N
```

Name: _____

Part B [5 points]

Given the following four 32-bit binary numbers, what is the corresponding MIPS assembly code?

```
0000 0010 0001 0001 1000 0000 0010 0000

0000 0000 0001 0000 1000 1000 0010 0010

0001 0110 0001 0001 0000 0000 0000 0100

0010 0010 0001 0000 1111 1111 1111 1100
```

Solution:

```
add $s0, $s0, $s1
sub $s1, $zero, $s0
bne $s0, $s1, 4 // branches past ADDI
addi $s0, $s0, -4
```

Name: _____

Problem 3 [15 points]

Part A [10 points]

In terms of gate delay, *quantitatively* calculate, compare, and contrast the performance for a 16-bit carry lookahead adder (CLA) **without ripple** and a 16-bit CLA **that accidentally ripples between every bit**. To receive credit you must explain your answer.

The baseline equation for a CLA-4 is:

$$c_{i+1} = g_i + p_i * c_i$$

If we don't unroll this, then we have the following:

$$c_3 = g_3 + p_3 * c_3$$
$$c_2 = g_2 + p_2 * c_2$$
$$c_1 = g_1 + p_1 * c_1$$
$$c_0 = g_0 + p_0 * c_0$$

Where each equation relies on the previous equation to complete before it can be calculated. Thus, each carry equation is a 2-OR with a 2-AND input to that 2-OR. Moreover, since the carry's are rippling into one another, that means our critical path is **8 + 1 = 9 gates** (from p_0 * c_0 → the final OR for c_4, with the "+1" being for the gates that calculate the g_i's and p_i's). More generally this should look familiar to the ripple carry adder where the delay is linear with the number of bits (4*2 + 1 = 9 gates).

Likewise, we can apply the same logic to the CLA-16 with ripple. Now we have:

$$c_{i+4} = G_{i,i+3} + P_{i,i+3} * c_i$$

Which becomes:

$$c_{16} = G_{12,15} + P_{12,15} * c_{12}$$
$$c_{12} = G_{8,11} + P_{8,11} * c_8$$
$$c_8 = G_{4,7} + P_{4,7} * c_4$$
$$c_4 = G_{0,3} + P_{0,3} * c_0$$

If you were to draw the gates for this – calculating c4, feeding it into the c8 calculation, and so on, it would again be a long series of 2-input AND/OR gates feeding into one another. Here, there are **8+2 = 10 gates** (8 gates for $P_{3,0}$ * c_0 → the final OR for c_{16}; 2 gates for the G and P calculations). Again, this should look familiar to the ripple carry adder calculations, except for the additional gates for the G and P calculations.

Non-Rippling CLA:

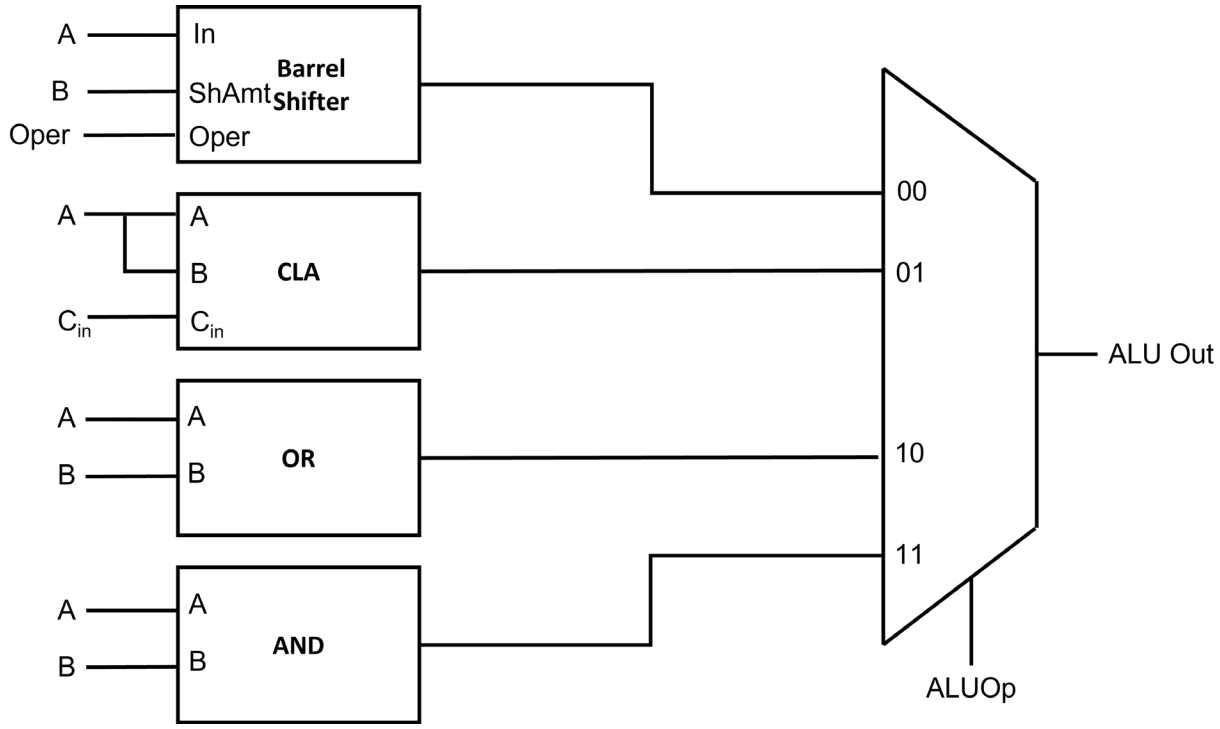
This should follow directly from our discussion in class, where the critical path delay for a non-rippling CLA is $log_2(\#\ bits)$ + 1, since we build the CLA-16 our of CLA-4's (which are built out of 1-bit full adders). So that means the critical path delay in terms of gates is log2(16) + 1 = **4+1 = 5**

gates. The key idea here is we are performing all of the calculations for carry bits in parallel by unrolling the equations above, which adds more logic, but since the logic can be done in parallel (ignoring that 5-input gates may be more expensive than 2-input gates), the critical path is much shorter.

Name: _____

Part B [5 points]

Imagine you are the designer of a single-cycle RISC processor similar to ones discussed in class. However there is a bug in your ALU as shown below:



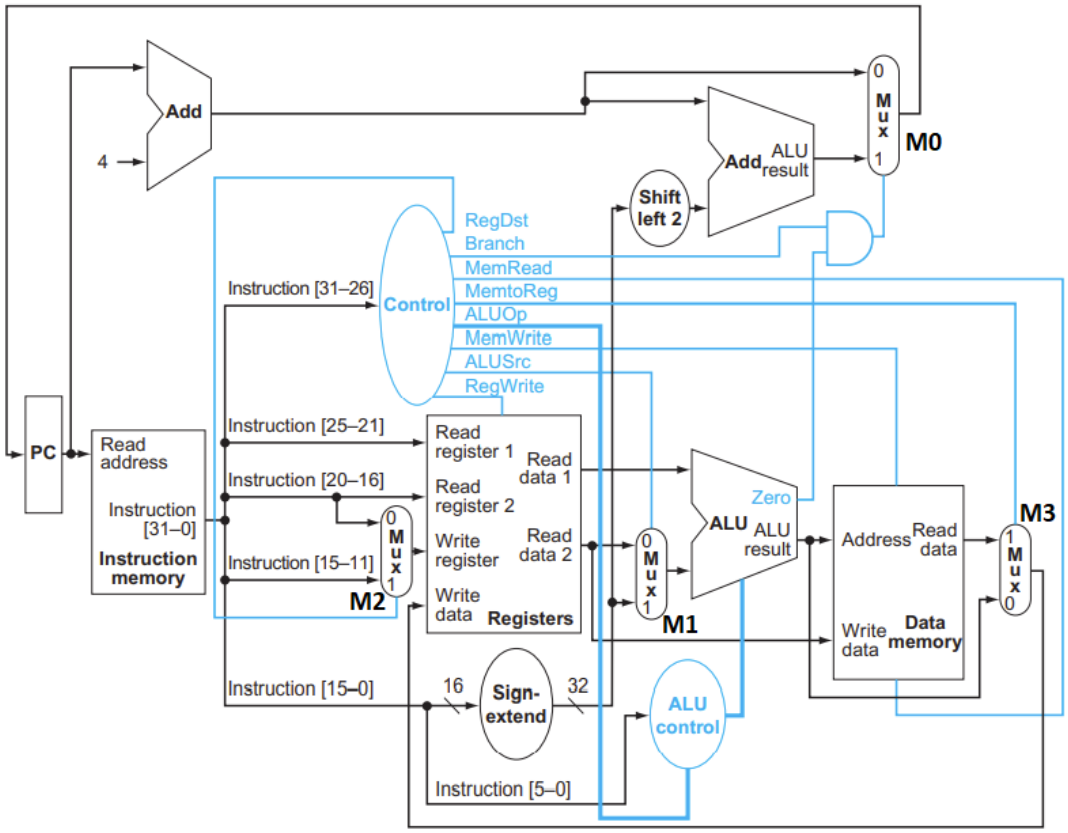
Design a simple assembly test that will expose this bug (i.e., a sequence of MIPS assembly instructions that will get the wrong answer because of this bug). To receive credit, you must justify why this test exposes the problem with your ALU. You should not assume anything about the initial values of the registers (i.e., you should set the register values to be what you want in the assembly code).

CLA input B is broken, any test that where A != B for CLA should expose it. For example:

```
ORI   $1, $0, 1
ORI   $2, $0, 2
ADD   $3, $1, $2 // should get 1 + 2 = 3, but get 1 + 1 = 2
```

Problem 4 [30 points]

Single Cycle Datapath and Control Path



For this question, we will refer to the above-given single-cycle datapath and control signals. For convenience, muxes are named M0 to M3 so they are easier to reference while answering the questions.

Additionally following are the delays for each unit:

I-Mem	Add	Mux	ALU	Registers	D-Mem	Sign extend	Shift left 2
225 ps	65 ps	20 ps	150 ps	100 ps	250 ps	20 ps	15 ps

Assume delays for all other elements are 0 ps.

Part A [10 points]

1. Suppose the processor only supports 1 instruction which is:

- beq rs, rt, offset: if(rs == rt) branch to offset relative to pc

What are the control signal values for "RegDst, Branch, MemRead, MemtoReg, MemWrite, ALUSrc & RegWrite"? (For each signal specific 1, 0, don't care)

What will be the minimum cycle time for the processor?

Control Signals are as follows:

RegDst = Don't Care

Branch = 1

MemRead = 0 (or Don't Care?)

MemtoReg = Don't Care

MemWrite = 0

ALUSrc = 0

RegWrite = 0

Minimum Cycle Time:

2 steps in parallel, first calculate zero to decide whether to branch or not. And second, calculate PC+4 and further values.

First,

Calculate zero = 225(I-Mem) + 100(Registers) + 20 (Mux M1) + 150(ALU) = 495 ps

Second,

Calculate PC further values = 225(I-Mem) + 20 (SignExt) + 15 (Shift Left 2) + 65 (Add) = 325 ps

So by the time we calculate zero, we would be ready with all the signals to MUX M0. So the next delay is only due to MUX M0.

So total cycle time = 495 + 20 = 515 ps

Part B [10 points]

2. Now let's add support for 2 more instructions which are as follows: **[10 pts]**

- add rd, rs, rt: rd ← rs + rt
- lw rt, offset(rs): rt ← memory[rs + signExt(offset)]

What will be the minimum cycle time of the processor? Write the timings for both add and lw to explain your answer.

add cycle time:

Cycle time = 225(I-Mem) + 100(Register) + 20(Mux M1) + 150(ALU) + 20(Mux M3) = 515 ps

Note: We are not adding a delay of MUX M2 as the value is needed only before the next clock edge and can be done in parallel of register reading with any issues.

lw cycle time:

cycle time = 225(I-Mem) + 100(Register) + 150(ALU) + 250(D-Mem) + 20(Mux M3) = 745 ps

Note: We are not adding delays for Sign Extend because it will happen in parallel to Register File. Also, we are not adding delay due to Mux M1 because the signal to M1 will select sign extend and both together will happen in parallel of register file reading even for r2 is out. (You don't care about the r2 value if M1 is not selecting r2).

So cycle time = 745 ps.

Part C [10 points]

3. Lets define a new instruction called:

- jmi rs, offset: pc = memory[rs + signExt(offset)]

For this new instruction:

i.) Which of the already existing blocks can be used?

ii.) What are the new blocks needed to support the instruction and where? Also, specify what are the control block changes? If any.

i.) The already existing blocks in use will be Register File, ALU, sign Extend, Mux M1, D-Mem, and I-Mem.

ii.) We will need to add one more mux just after M0. We will pass 2 values to it first the M0 output and second the Read data from Memory. And lastly, we will need to add a control signal for that new MUX.

Problem 5 [10 points]

(5 points) We want to build low-power core for a smartwatch type product. Our design achieves low power but reduces the performance of 20% of the benchmark program running on the processor by a factor of 4. On the other hand, architects have come up with an optimization that improves the performance of another 12% of the benchmark by a factor of 6. Assume the two enhancements are independent and affect different parts of the program (there is no overlap between the 20% and 12% of the program). What is the overall effect on the performance of the program?

By Amdahl's Law

$$\text{Overall Speedup} = \frac{1}{fraction\ unchanged + fraction\ changed}$$

$$= \frac{1}{0.68 + [(0.2 \times 4) + \left(\frac{0.12}{6}\right)]}$$

$$= 0.666$$

NOTE: 0.2 fraction (20%) of the benchmark program is *slowed down* by a factor of 4.

The improvement in 12% of the program is not sufficient to compensate for the performance degrade in the other 20% and still results in an overall slowdown.

(5 points) Explain how jump instruction in MIPS works with an example – where PC address for the jump instruction is 0x FFFF 0000 and 26-bit label address is 0000 0000 0000 0000 1111 0000 11.

Name: _____ (Write your name on this page if you use it)

SCRATCH PAGE #1

Name: _____ **(Write your name on this page if you use it)**

SCRATCH PAGE #2