CS232 Midterm Exam 2
March 30, 2005

Name: ______________________________

Section: ______________________________

- This exam has 6 pages (including a cheat sheet at the end).
- Read instructions carefully!
- You have 50 minutes, so budget your time !
- No written references or calculators are allowed.
- To make sure you receive credit, please write clearly and show your work.
- We will not answer questions regarding course material.

Question	Maximum	Your Score
1	40	
2	40	
3	20	
Total	100	

Question 1: Single-cycle CPU implementation (40 points)

On the last page of the exam is a single-cycle datapath for a machine **very different** than the one we saw in lecture. It supports the following (complex) instructions:

```
lw_add   rd, (rs), rt          # rd = Memory[R[rs]] + R[rt];
addi_st  (rs), rt, imm         # Memory[R[rs]] = R[rt] + imm;
sll_add  rd, rs, rt, imm       # rd = (R[rs] << imm) + R[rt];
```

All instructions use the same format (shown below), but not all instructions use all of the fields.

Field	op	rs	rt	rd	imm
Bits	31-26	25-21	20-16	15-11	10-0

Part (a)
For each of the above instructions, specify how the control signals should be set for correct operation. Use **X** for **don't care**. ALUOp can be **ADD**, **SUB**, **SLL**, **PASS_A**, or **PASS_B** (*e.g.,* PASS_A means pass through the top operand without change). Full points will only be awarded for the fastest implementation. (20 points)

inst	ALUsrc1	ALUsrc2	ALUsrc3	ALUop1	ALUop2	MemRead	MemWrite	RegWrite
lw_add								
addi_st								
sll_add								

Part (b)
Given the functional unit latencies as shown to the right, compute the minimum time to perform each type of instruction. Explain. (15 points)

Func. Unit	**Latency**
Memory	3 ns
ALU	4 ns
Register File	2 ns

inst	Minimum time	Explain
lw_add		
addi_st		
sll_add		

Part (c)
What is the CPI and cycle time for this processor? (5 points)

Question 2, Multi-cycle implementation (40 points)

The (imaginary) *jump memory* (jmem) instruction is like a *jump-and-link* (jal) instruction, except both the target is loaded from memory and the return address is saved to memory. The i-type format is used, as shown below. You can assume that R[rt] and (R[rs] + offset) are distinct (non-overlapping) addresses.

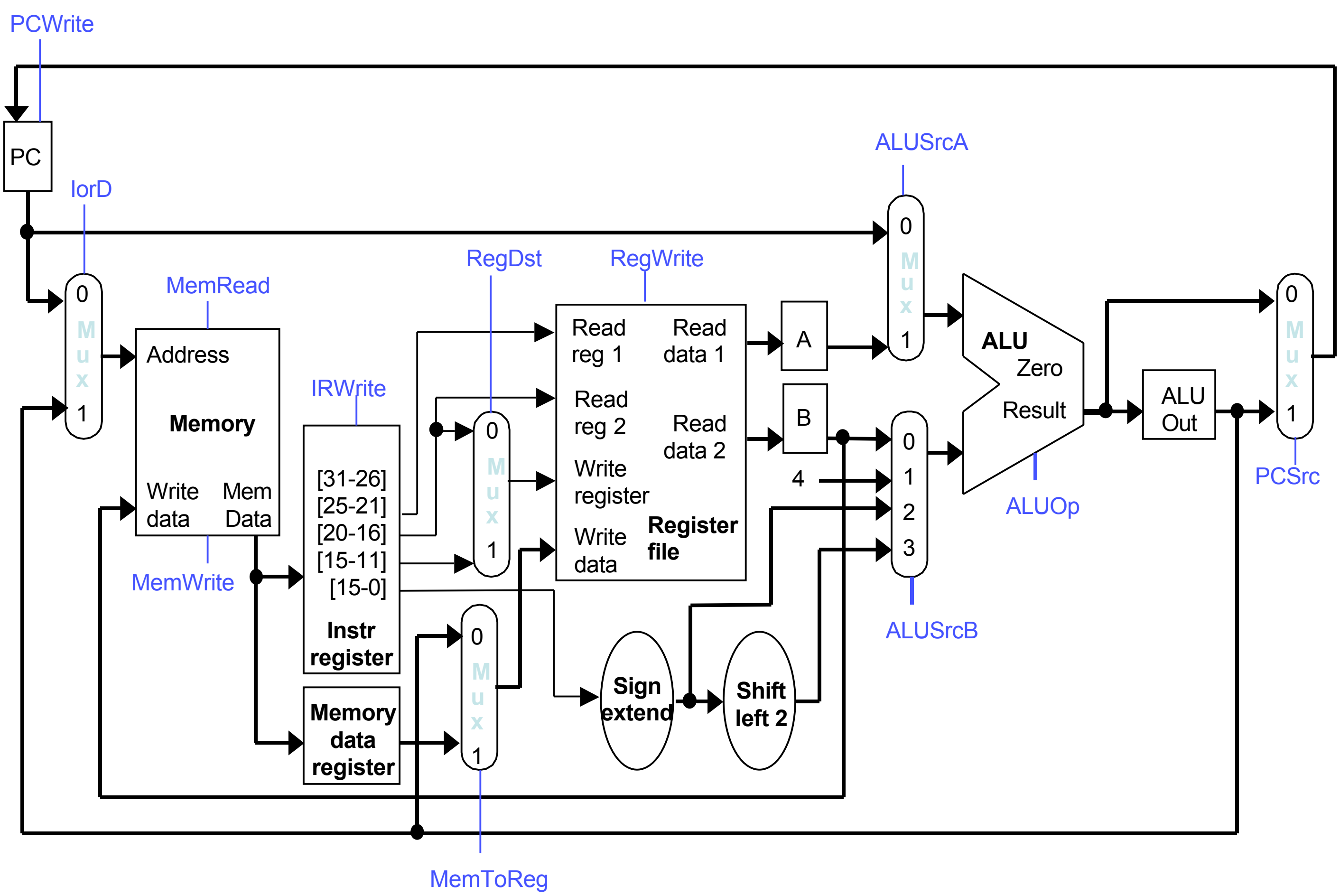
```
jmem (rt), offset(rs)          # Memory[R[rs]+offset] = PC+4;
                               # PC = Memory[R[rt]]
```

Field	op	rs	rt	imm
Bits	31-26	25-21	20-16	15-0

Part (a)

The multicycle datapath from lecture appears below. Show what changes are needed to support **jmem**. You should only add wires and muxes to the datapath; do not modify the main functional units themselves (the memory, register file, and ALU). Try to keep your diagram neat! (15 points)

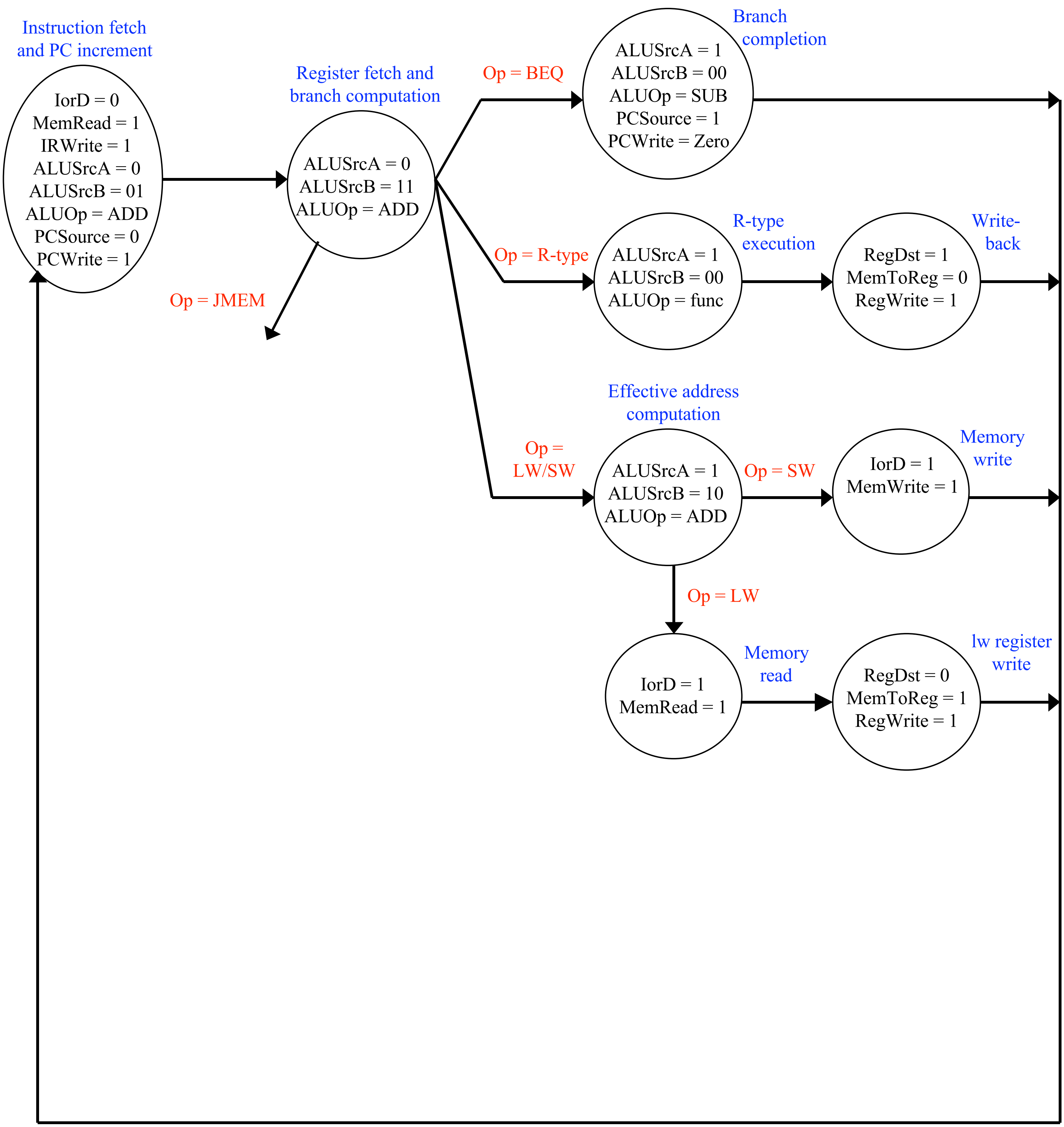
*Note: While we're primarily concerned about correctness, five (5) of the points will only be rewarded to solutions that use a **minimal number of cycles** and do not lengthen the clock cycle. Assume that everything besides the ALU, Memory and Register File is instantaneous.*



Question 2, continued

Part (b)

Complete this finite state machine diagram for the **jmem** instruction. Control values not shown in each stage are assumed to be 0. Remember to account for any control signals that you added or modified in the previous part of the question! (25 points)



Question 3: Conceptual Questions (20 points)

Write a short answer to the following questions. For full credit, answers should not be longer than **two sentences**.

Part (a)

Can the following factors of performance be affected by the implementation (*e.g.*, single-cycle, multi-cycle, etc.)? Explain. (10 points)

Number of Instructions:

Cycles per Instruction (CPI):

Clock Period:

Part (b)

What is optimistic (or eager) execution? How does it relate to the machine implementations we've seen? (5 points)

Part (c)

What differentiates one computer from another? List 5 distinct, important ways (other than ISA) . Single word answers are fine, if they are clear. (5 points)

1.

2.

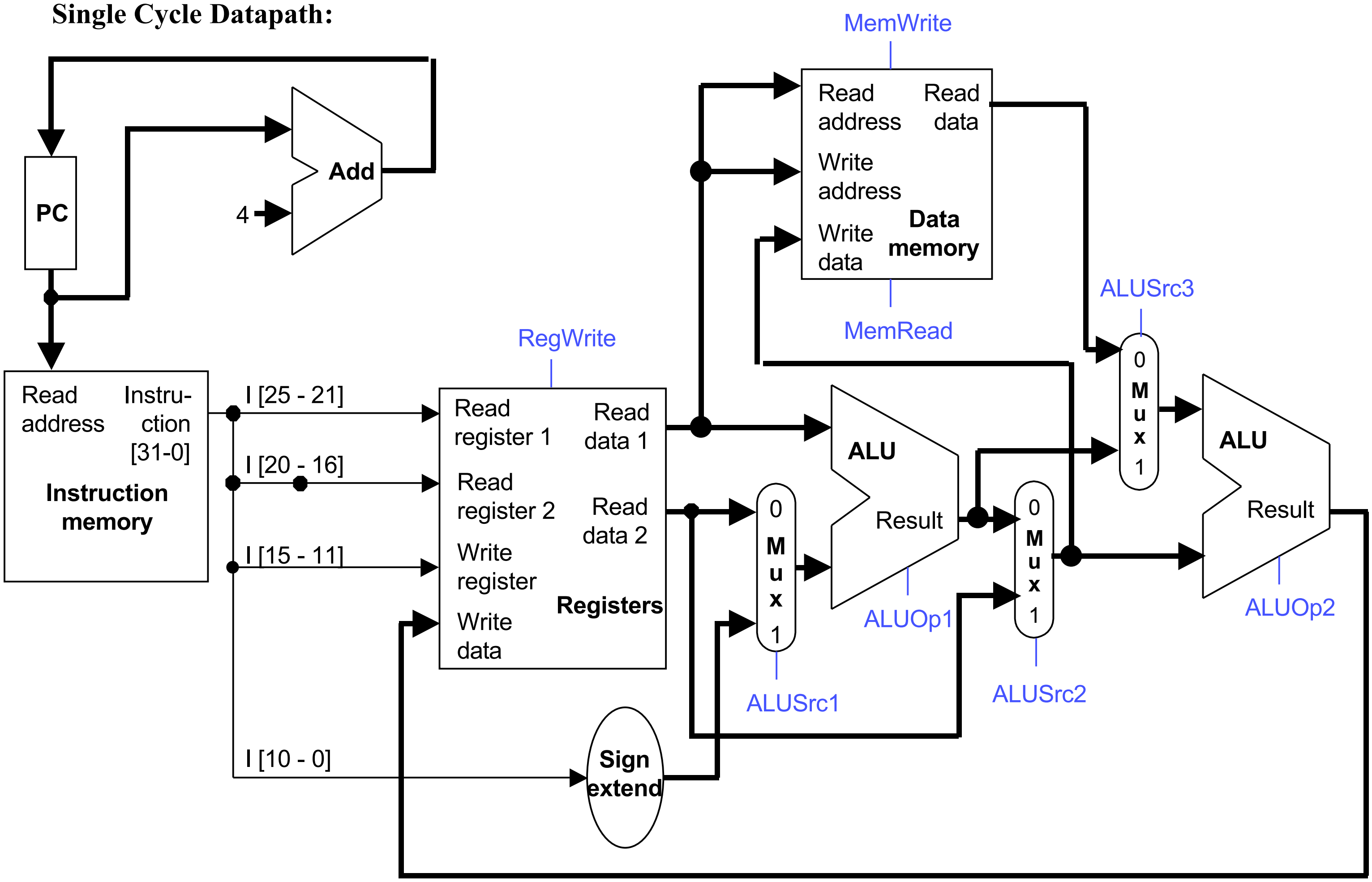
3.

4.

5.

Do not write in shaded region

Single Cycle Datapath:



Performance

1. Formula for computing the CPU time of a program P running on a machine X:

$$CPU\ time_{X,P} = Number\ of\ instructions\ executed_P\ x\ CPI_{X,P}\ x\ Clock\ cycle\ time_X$$

2. CPI is the average number of clock cycles per instruction:

$$CPI = Number\ of\ cycles\ needed\ /\ Number\ of\ instructions\ executed$$

3. Speedup is a metric for relative performance of 2 executions:

$$Speedup = Performance\ after\ improvement\ /\ Performance\ before\ improvement$$
$$= Execution\ time\ before\ improvement\ /\ Execution\ time\ after\ improvement$$