CS232 Midterm Exam 2
November 12, 2003

Name: Darth Vader

Section: Imperial Star Ship

- This exam has 7 pages including the pipelined datapath diagram on the last page, which you are free to tear off.
- You have 50 minutes, so budget your time carefully!
- 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	50	
2	20	
3	30	
Total	100	

Question 1: Multicycle CPU implementation (50 points)

Consider extending the MIPS architecture with the instruction below, which adds **three** registers together and stores the result in a register.

add3 rd, rs, rt, ru   # rd = rs + rt + ru

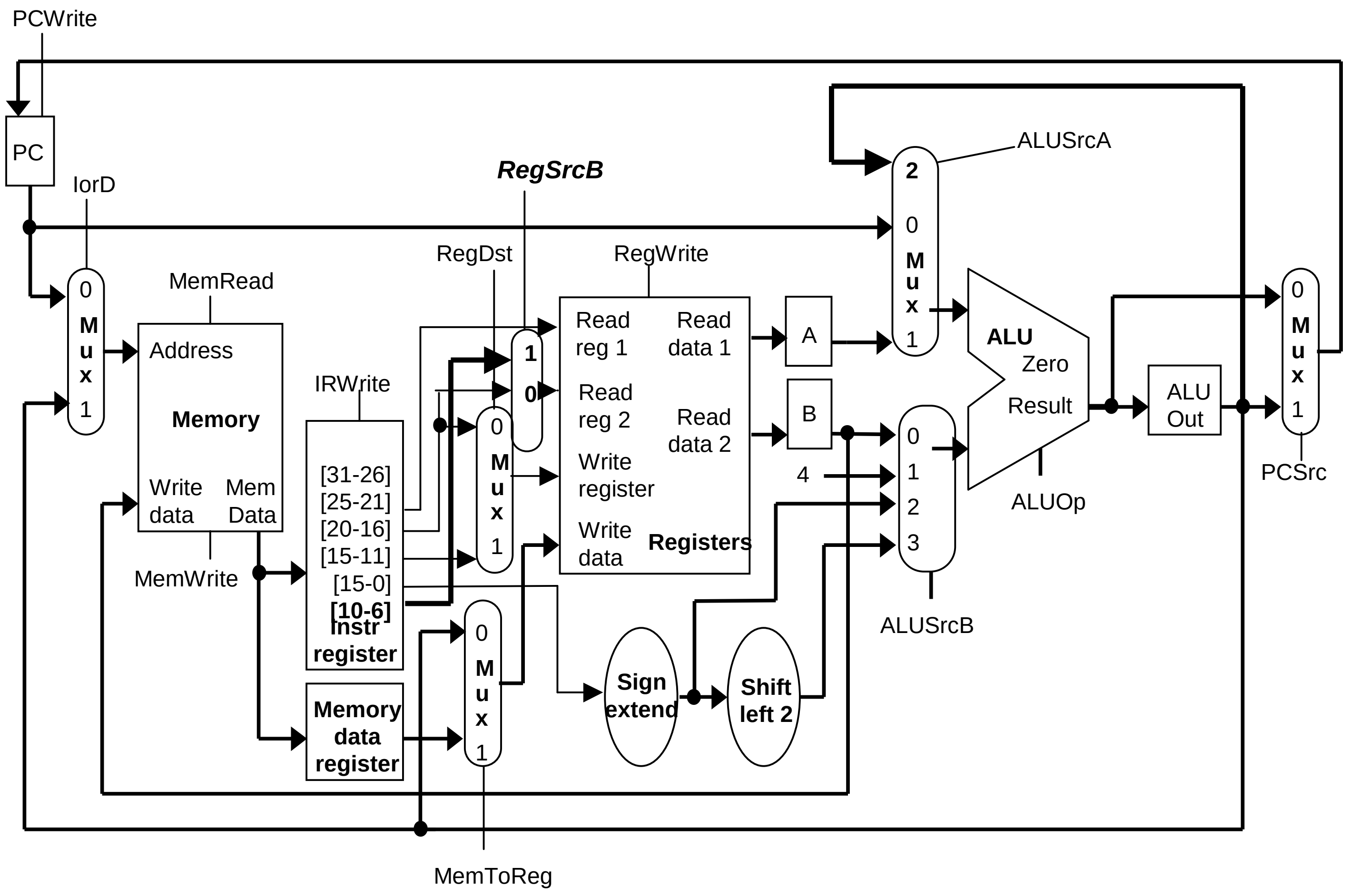
This will use the same format as R-type instructions---shown here for reference---where the *shamt* field is used to hold *ru*.

Field	op	rs	rt	rd	shamt/ru	func
Bits	31-26	25-21	20-16	15-11	10-6	5-0

An example of the usage of the *add3* instruction is shown in part (c) of question 1.

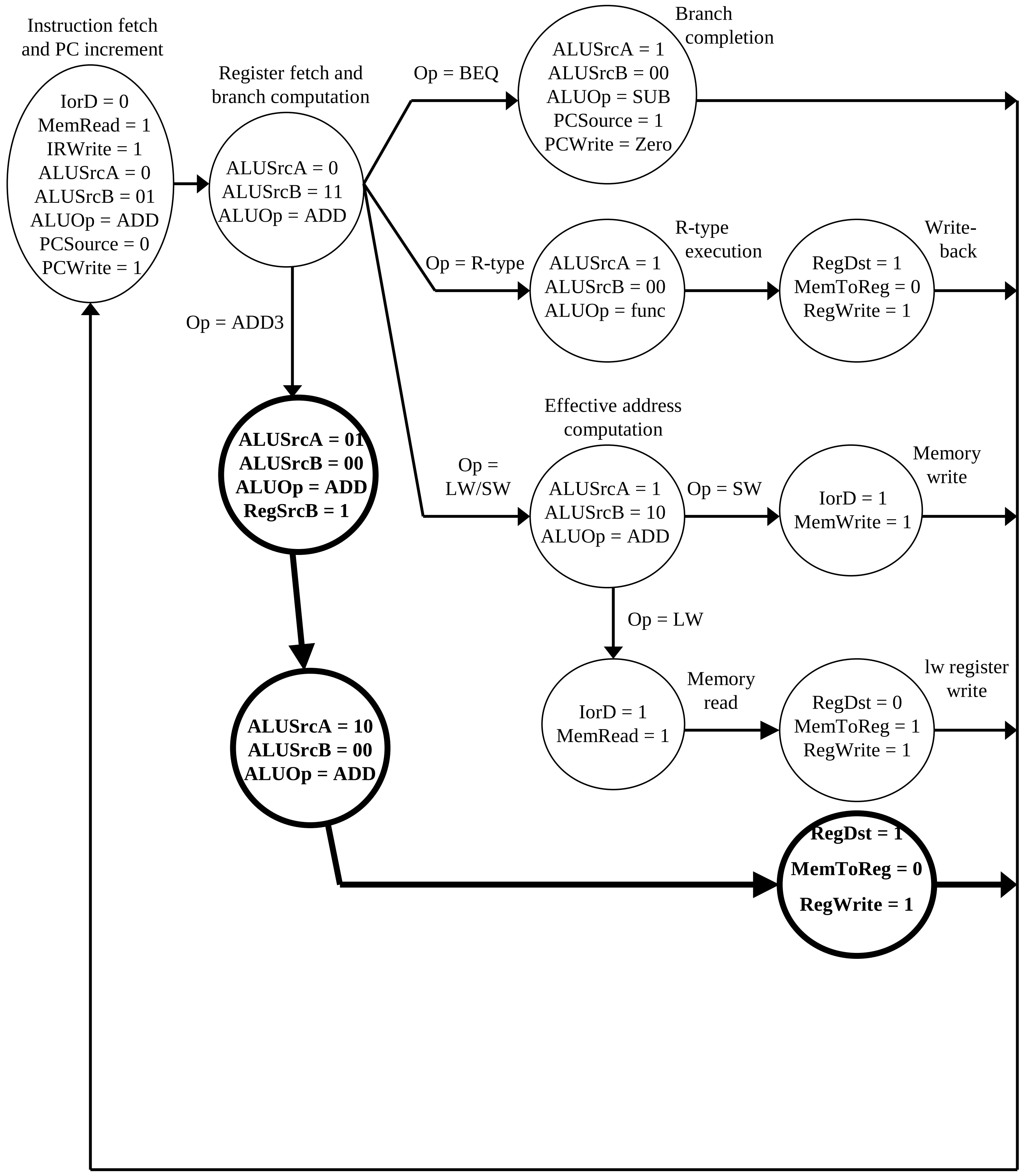
Part (a)
The multicycle datapath from lecture appears below. Show what changes are needed to support *add3*. You should only add wires and muxes to the datapath; do not modify the main functional units themselves (the memory, register file and ALU). Try to keep your diagram neat! (15 points)

Note: While we're primarily concerned about correctness, full points will only be rewarded to solutions that use a minimal number of cycles and do not lengthen the clock cycle. Assume that the ALU, Memory and Register file all take 2ns, and everything else is instantaneous.



Question 1 continued

Part (b)
Complete this finite state machine diagram for the *add3* 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! (20 points)



Question 1, continued:
Part (c)

The *add3* instruction can be used in place of two dependent add instruction, reducing the number of instructions that need to be executed. Below are two functionally equivalent programs; the second of which uses the *add3* instruction:

Program 1

```
5    lw     $t0, 0($a0)
5    lw     $t1, 4($a0)
5    lw     $t2, 8($a0)
4    add    $t3, $t0, $t1
4    add    $t3, $t2, $t3
4    sw     $t3, 0($a1)
```

Program 2

```
5    lw     $t0, 0($a0)
5    lw     $t1, 4($a0)
5    lw     $t2, 8($a0)
5    add3   $t3, $t0, $t1, $t2
4    sw     $t3, 0($a1)
```

Assuming the datapath and control that you implemented in parts (a) and (b), how much faster (or slower) is program 2 than program 1? You may leave your answer as a fraction. (10 points)

Program 2 is faster by 3/24 = 12.5%

Part (d)

Implementing the *add3* instruction in the pipelined datapath we discussed in class (shown on page 7) would be much more difficult. Give two specific reasons why. (5 points)

There are two main structural hazards

- The ALU is used twice (would need a second adder)
- There are 3 register reads (would need a 3rd read port)

Question 2: Pipelined Datapath (20 points)

The pipeline datapath diagram on page 7 (feel free to tear it off) has a number of wires/buses annotated. For each annotation (5 points per annotation):

- describe the purpose of the wire/bus and
- give an instruction or (instruction sequence) that requires it.

1

Purpose:This bus sends PC+4 back to MUX that updates PC

Instr: Everyone in IF stage e.g.,

add $a1,$t1, $s1

2

Purpose:This bus sends extended immediate value to ALU

Instr: lw and sw and addi e.g.,

lw $t1, 8($a0)

3

Purpose:Data Bus to send Data to Memory

Instr: Stores e.g.,

sw $a2, 0($t0)

4

Purpose:Forward Data from WB stage to EX stage

Instr: Any R-Type, Stores and Loads e.g.,

lw **$t1**, 0($a0)

lw $t3, 4($a0)

add $s1, **$t1**, $t2

Question 3: Pipelining (30 points)

Part (a)

Pipelining primarily improves: instruction latency. [✓ instruction throughput.] (circle one, 5 points)

For parts b, c, and d, refer to the pipelined datapath on page 7.

Part (b)
What is the penalty for taken branches (in number of cycles flushed)? How do you know? (10 points)

One cycle because branches are determined in ID stage (see the little = box)

Also, only the IF register has a flush signal.

Part (c)
Describe (in English) the situations that require a load stall. (5 points)

Load immediately followed by an instruction uses the result of the load

Part (d)
Complete the control equation for load stalls. *Hint: use the forwarding equation as a guide.* (10 points)

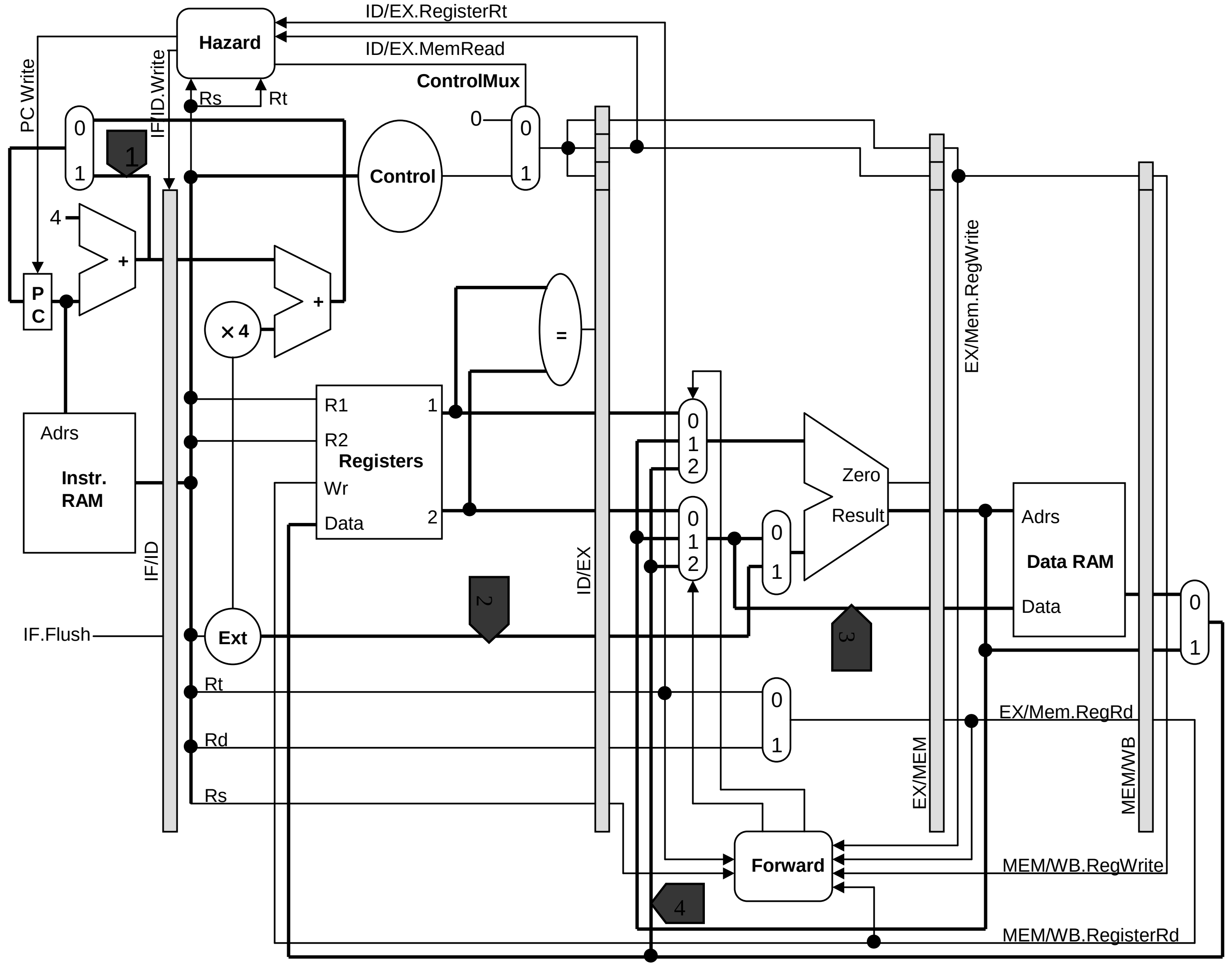
```
**if ( ID/EX.MemRead == 1 AND
((ID/EX.rt == IF/ID.rs) OR (ID/EX.rt == IF/ID.rt)))**

then PC_Write = 0, IF/ID.Write = 0, and ControlMux = 0
```

Note: by default, PC_Write, IF/ID.Write, and ControlMux are 1.

Pipelined Datapath

Here is the final datapath that we discussed in class.

- Forwarding is performed from the EX/MEM and MEM/WB latches to the ALU inputs. The control equation for the Rs register input to the ALU is shown at the bottom of the page. The Rt input is similar.
- Branches are assumed to be not taken.
- A hazard unit inserts stalls for lw instructions



```
if ((EX/MEM.RegWrite == 1) and
      (EX/MEM.RegisterRd == ID/EX.RegisterRs))
            then ForwardA = 1
else if ((MEM/WB.RegWrite == 1) and
            (MEM/WB.RegisterRd == ID/EX.RegisterRs))
            then ForwardA = 2
```