CS232 Midterm Exam 2
March 19, 2004

Name: Mr. T . Rick Question

Section: _____

- This exam has 8 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	30	
2	50	
3	20	
Total	100	

Question 1: Single-cycle CPU implementation (30 points)

In sections, you have seen how to implement **jal** instruction. This instruction is useful only if we can also return from function calls. To perform the return, implement **jr rs** instruction (jump to register rs) in the single-cycle datapath. The format of **jr** instruction is shown below.

Field	op = 0	rs	0	func = 8
Bits	31-26	25-21	20-6	5-0

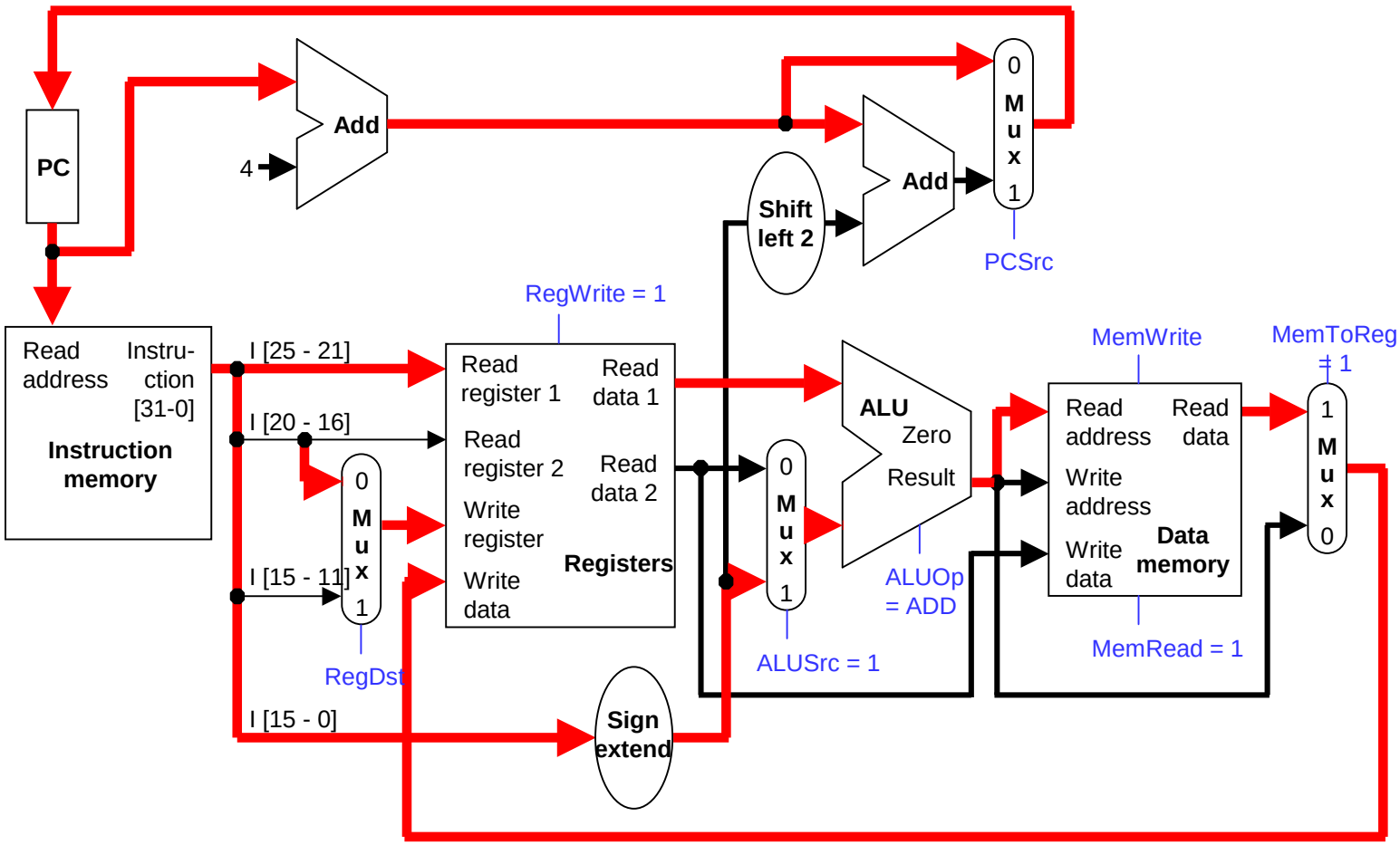
Before implementing the **jr** instruction, do parts (a) and (b) as a small warm-up exercise.

Part (a)

The single-cycle datapath from lecture appears below. Clearly mark all wires that are active during the execution of **lw instruction**. (10 points)

Part (b)

On the diagram below, write (next to the signal's name) values of **all** non-0 control signals required for the **lw** instruction. (5 points)



Question 1 continued

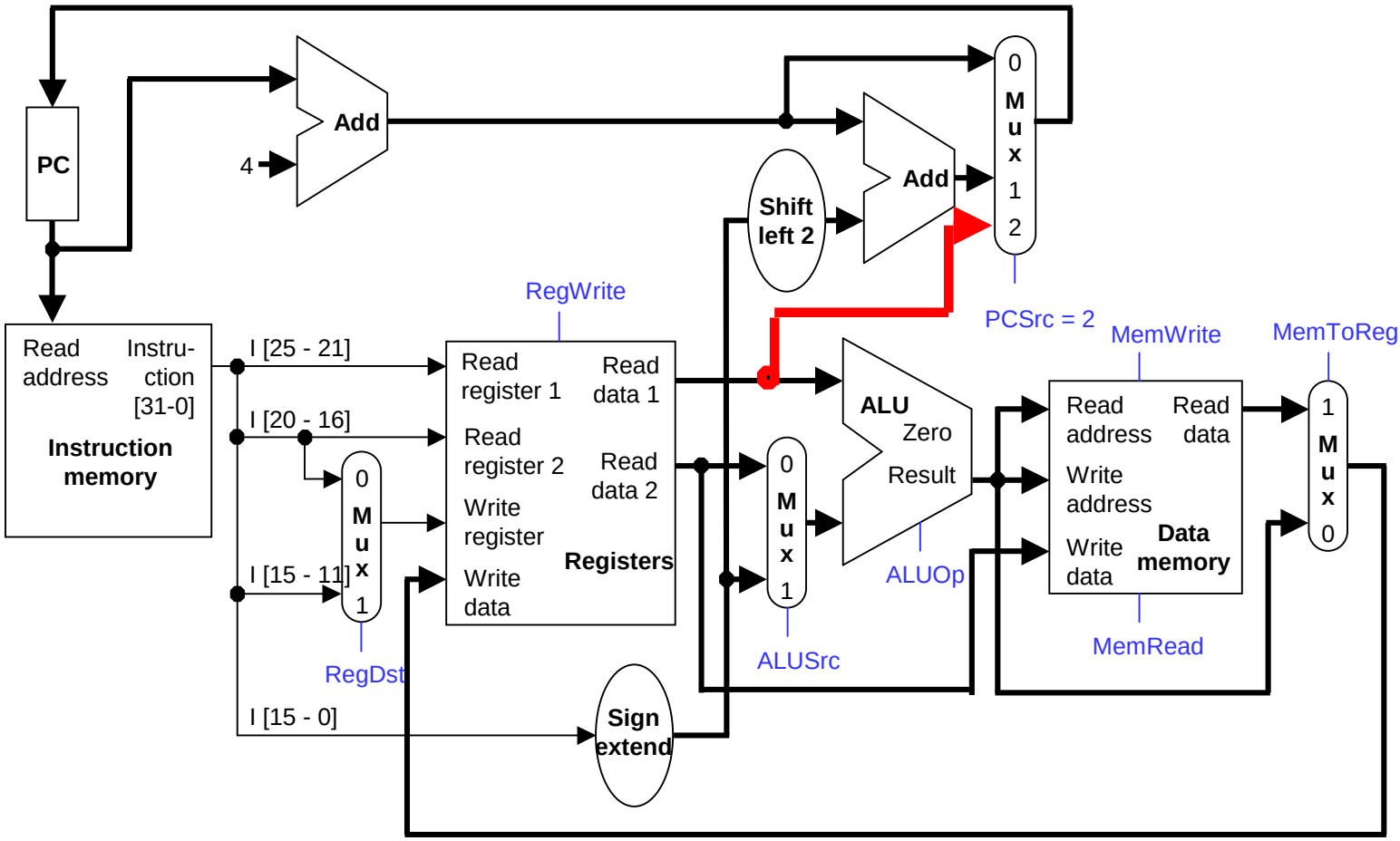
Part (c)

The single-cycle datapath from lecture (same as on the previous page) appears below. Show what changes are needed to support **jr** instruction. 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! (10 points)

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

Part (d)

On the diagram below, write (next to the signal's name) values of **all** non-0 control signals required for the **jr** instruction. (5 points)



Question 2: Multi-cycle CPU implementation and its performance (50 points)

Assume that the ALU can perform the max2 operation (*i.e.*, return the greater of 2 inputs):

```
alu_result = (A_input > B_input) ? A_input : B_input;
```

ALUOp for this instruction is MAX2.

Given this improved ALU, implement the **max4** instruction that writes into register rd the largest value of 4 registers:

```
max4 rs, rt, rd, rm              # rd = max(rs, rt, rd, rm)
```

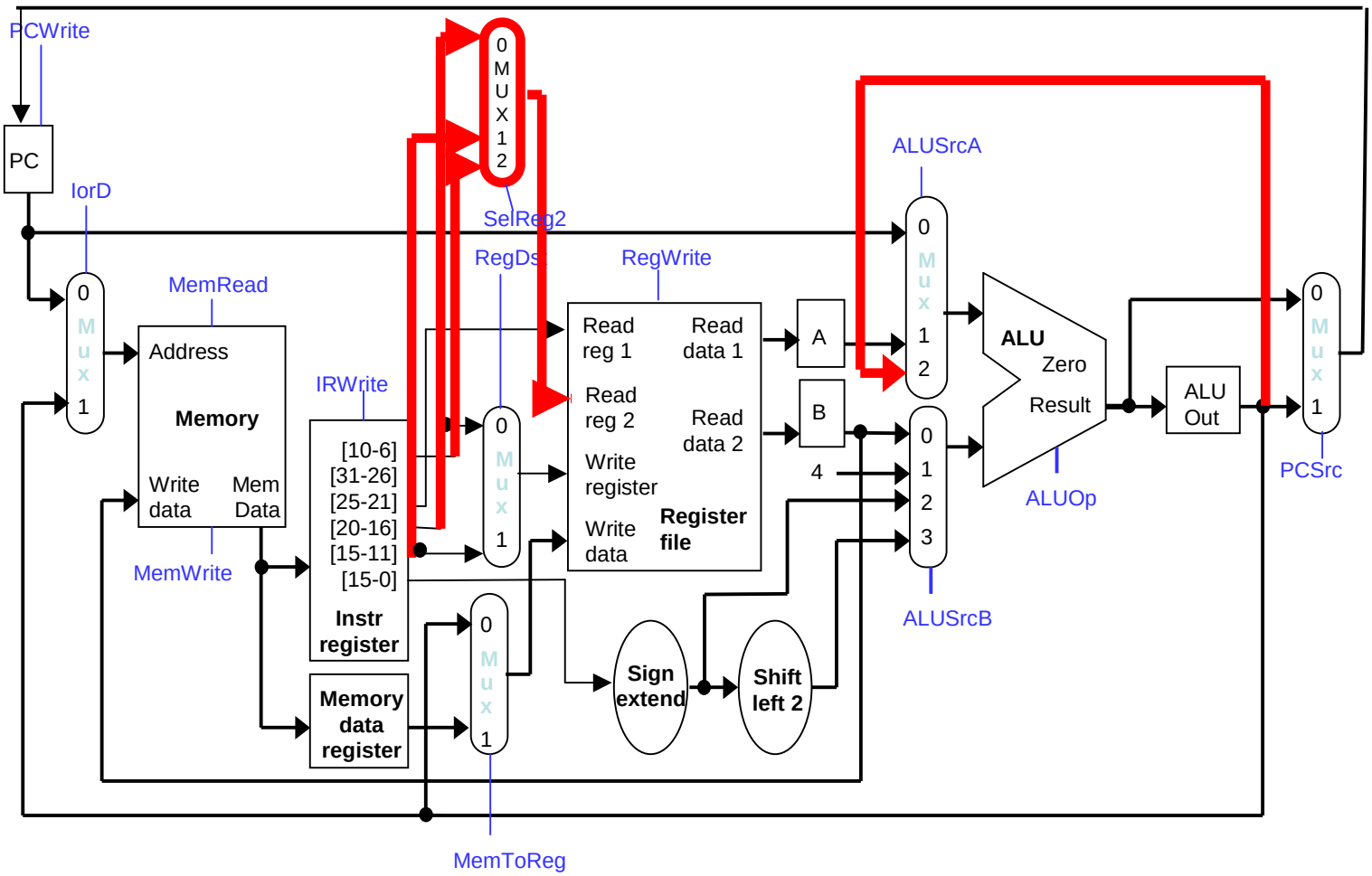
Note that register **rd** is both an *input* and an *output*. Instruction **max4** has the following format:

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

Part (a)

The multicycle datapath from lecture appears below. Show what changes are needed to support **max4**. 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! (10 points)

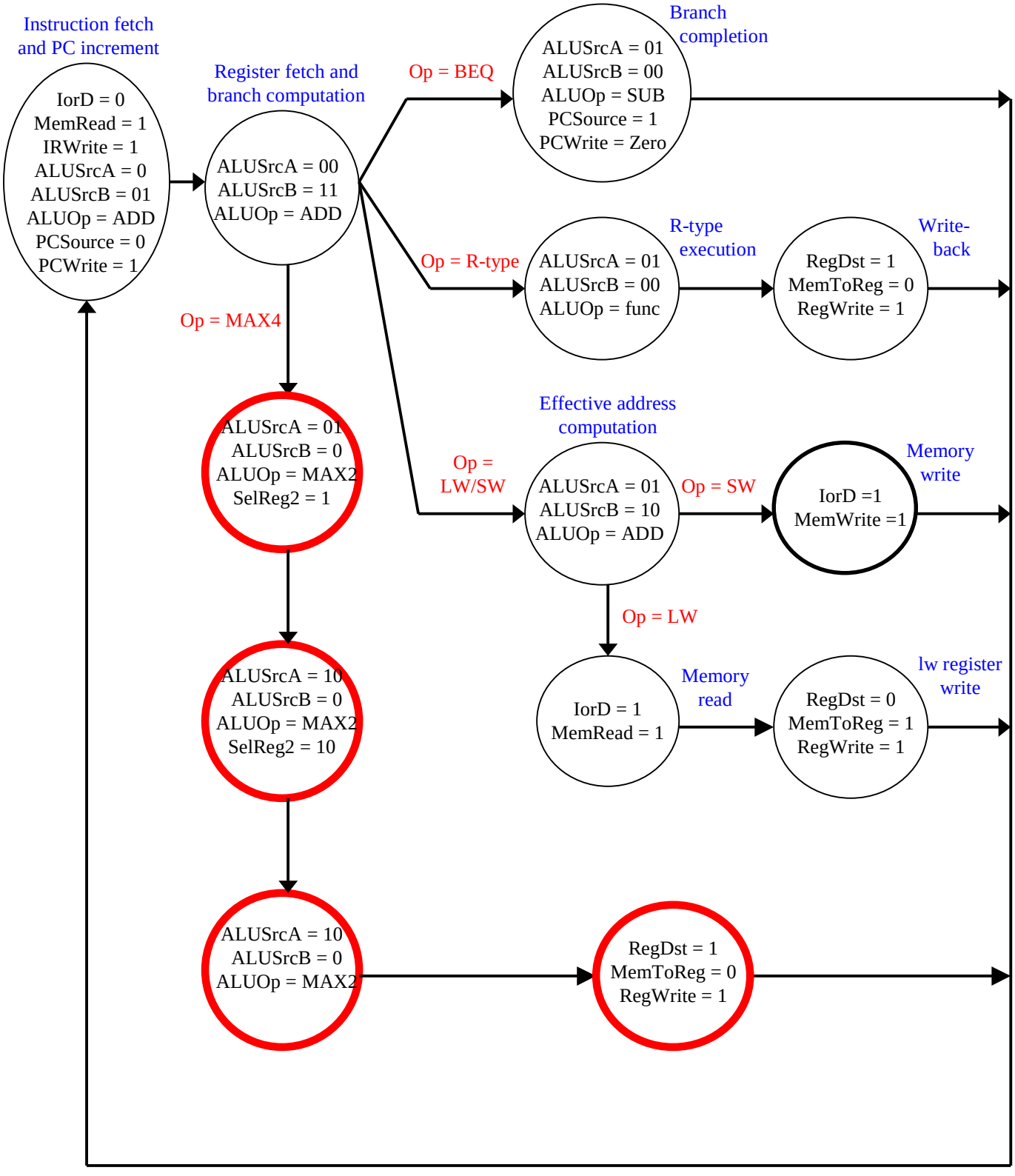
*Note: While we're primarily concerned about correctness, full points will only be rewarded to solutions that use a **minimal number of cycles** and do not lengthen the clock cycle. Assume that the ALU, Memory and Register file all take 2ns, and everything else is instantaneous.*



Question 2 continued

Part (b)

Complete this finite state machine diagram for the **max4** 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! (15 points)



Question 2 continued

Part (c)
The FSM diagram on the previous page is incomplete. The control signals for the fourth cycle of **sw instruction** are missing. Fill in the missing control signals in the FSM diagram. (5 points)

See Solution on the previous page

Part (d)
The **max4** instruction can be used in place of three branch instructions, reducing the number of instructions that need to be executed. Below are two functionally equivalent programs; the second of which uses the **max4** instruction:

Program 1

	lw	v0, 0(a0)	***5***
	add	a1, a0, 12	***4***
label:	add	a0, a0, 4	***4***
	lw	t1, 0(a0)	***5***
	slt	t2, t1, v0	***4***
	bne	t2, $zero, skip	***3***
	move	v0, t1	***4***
skip:	bne	a0, a1, label	***3***
	jr	$ra	***3***

Program 2

lw	v0, 0(a0)	***5***
lw	t0, 4(a0)	***5***
lw	t1, 8(a0)	***5***
lw	t2, 12(a0)	***5***
max4	v0, t0, t1, t2	***6***
jr	$ra	***3***

Assume the datapath and control that you implemented in parts (a) and (b); *assume* **slt** *and* **move** *can be considered as R-type instructions and* **jr** *and* **bne** *take the same amount of time as* **beq**. Also assume that the first element in the input array is the largest (so that the first branch of program 1 is always taken). How much faster (or slower) is program 2 than program 1? You may leave your answer as a fraction. (15 points)

Program 1

5 + 4 + 3 (4 + 5 + 4 + 3 + 3) + 3 = 69 cycles

Program 2

5 + 5 + 5 + 5 + 6 + 3 = 29 cycles

Program 2 is faster by 69/29

Part (e)
What is average CPI of **program 1**? You may leave your answer as an expression. (5 points)

69/18 cycles/ instruction

Question 3: More performance (20 points)

Part (a)
Assume the following delays for the main functional units:

Functional Unit	Time delay
Memory	5 ns
ALU	4 ns
Register File	3 ns

Given the following instructions: **lw**, **sw**, **add**, **beq,** calculate:
- minimum time to perform each instruction
- time required on a single-cycle datapath (from q. 1)
- time required on multi-cycle datapath (from q. 2).

Write your answers in the table below. State any assumptions. (15 points)

	Time to perform the instruction:		
Instruction	**minimum time (ns)**	**in single-cycle (ns)**	**in multi-cycle (ns)**
lw	20	20	25
sw	17	20	20
add	15	20	20
beq	12	20	15

The Single Cycle implementation is limited by the length Of the longest instruction i.e LW which takes 20ns

Part (b)
How much faster than a 1GHz single-cycle MIPS processor would a 3.0Ghz Pentium4 x86 processor be if it achieved a CPI of 0.5 on a workload? (5 points)

If the two ISAs had been the same the answer would be a speedup of
Ex Time = CPI * IC * CYCLE TIME
(1 * 1) / (1/3 * 0.5) / = 6 over the MIPS

However since we can't say anything about Instruction Count. We can't find the answer with the information given.

Performance

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

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

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:

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