Computer Engineering 110: Computer Architecture

Midterm Examination 1

Fall 2016

Name: Solution

Email: _____

Q1	12	
Q2	12	
Q3	18	
Q4	18	
Total	60	

This exam is closed book and closed notes. Personal calculators (four-function calculators only) *are* allowed. Show your work on the attached sheets (front and back) and insert your answer in the space(s) provided. **Please provide details on how you reach a result.** Ask for extra paper sheets if necessary.

You have 70 minutes to complete the exam. This exam is worth 60 points. This exam counts for 15% of your course grade.

(12 points) Q1. Short Answer.

1) If a given program runs on a processor with a higher clock frequency, does it imply that the processor always executes more instructions per second (compared to a processor with a lower clock frequency)?

No. (3 points)

Explanation (3 points)

2) If a processor executes more of a given program's instructions per second, does it imply that the processor always finishes the program faster (compared to a processor that executes fewer instructions per second)?

No. (3 points)

Explanation (3 points). Instruction count can be different.

(12 points) Q2. Performance.

Computer A uses the MIPS ISA and has a 2GHz clock frequency. Computer B uses the x86 ISA and has a 3GHz clock frequency. On average, MIPS programs execute 1.5 times as many instructions as x86 programs. For program P1, computer A has a CPI of 2 and computer B a CPI of 3. Which computer has faster execution time? What is the speedup?

Computer	ISA	Clk	CPI	# Instr
A	MIPS	2 GHz	2	1.5×
B	x86	3 GHz	3	1×

Solu 1.

A = (2*1.5)/2 = 1.5

B = (3*1)/3 = 1, so B is faster by 50% Speedup = 50% or 1.5

Solu 2.

Assume 10 instructions in total by running program P1 on Computer B, then A will need 15 instructions.

latency A = CPI_A * 20 * $cycle\_time_A$ = 2 * 15* (1/2) = 15 ns

latency$B=CPI_B$ *10*$cycle\_time_B$ =3*10*(1/3)=10ns

B is faster. Speedup = (15/10 – 1) * 100% = 50% or Speedup = 15/10 = 1.5

Grading instruction: First check the result, and then look into the details.

1. If wrong result, take 3 points off
2. If no details about how to reach the result, take 5 points off
3. I provide two possible solutions. But there may be other solutions too.
4. Give partial credits by looking at the details, even if the result is wrong.
5. Take 2 points off on every incorrect detail, until 0 point left.

(18 points) Q3. Optimization.
Assume a typical program has the following instruction type breakdown. Assume the processor that this program will be running on has the following instruction latencies.

Instruction	Instr. Frequency	Latency (Cycles)
load	30%	4
store	10%	4
add	50%	2
multiply	8%	16
divide	2%	50

If you could pick one type of instruction to make twice as fast (half the latency) in the next-generation of this processor, which instruction type would you pick? Why?

	Base	loads	stores	adds	mults	divs
Loads 30%*4	1.2	0.6	1.2	1.2	1.2	1.2
Stores 10%*4	0.4	0.4	0.2	0.4	0.4	0.4
Adds 50%*2	1.0	1.0	1.0	0.5	1.0	1.0
Mults 8%*16	1.28	1.28	1.28	1.28	0.64	1.28
Divs 2%*50	1.0	1.0	1.0	1.0	1.0	0.5
	4.88	4.28	4.68	4.38	4.24	4.38

So I would pick the multiplies to make twice as fast because it reduces the effective CPI the most (by 15% over the Base case)

Grading instruction: First check the result, and then look into the details.

1. If wrong result, take 3 points off

2. If no details about how to reach the result, take 10 points off

3. Give partial credits by looking at the details, even if the result is wrong.

4. Take 2 points off on every incorrect detail, until 0 point left.

(18 points) Q4. ISA and Performance.

Given the following instruction categories and execution latencies (assuming and non-pipelined CPU):

Instruction	Latency (Cycles)
add(addu, addiu)	4
load (lw)	10
multiply(mul)	20
branch (bne)	8

Given the following assembly code:

```
L3:  addu R7, R4, R3
     lw R7, (R7)
     addu R8, R5, R3
     lw R8, (R8)
     mul R7, R7, R8
     addu R2, R2, R7
     addiu R3, R3, #4
     bne R3, R6, L3
```

Where registers are written as R(reg number) for example R1

Answer following questions:

1) What is the CPI of the loop for one iteration (the bne instruction included)?

There are a total of 8 instructions in the loop. The percentage breakdown by type is:

- add: 50%
- load: 25%
- multiply: 12.5%
- branch: 12.5%

So the CPI is: 0.5*4 + 0.25*10 + 0.125*20 + 0.125*8 = 8

2) Which of the following optimizations would produce the biggest CPI improvement for one iteration (the bne instruction included)?

- Implement prefetching to reduce the latency of loads from 10 cycles to 7 cycles.
- Implement branch prediction to reduce the latency of branch instructions from 8 cycles to 2 cycles.

The performance improvement is the same for both optimizations:

CPI for prefetching: 0.5*4 + 0.25*7 + 0.125*20 + 0.125*8 = 7.25
CPI for branch prediction: 0.5*4 + 0.25*10 + 0.125*20 + 0.125*2 = 7.25

Grading instruction: First check the result, and then look into the details.

1. If wrong result, take 2 points off

2. If no details about how to reach the result, take 8 points off

3. Give partial credits by looking at the details, even if the result is wrong.

4. Take 2 points off on every incorrect detail, until 0 point left.