Homework #2

Due Jan. 11

5 points each

1. We wish to compare the performance of two different machines: M1 and M2. The following measurements have been made on these machines:

Program	Time on M1	Time on M2
1	3 seconds	1.5 seconds
2	4 seconds	10 seconds

a) Which computer is faster for each program, and how many times as fast is it?
Program_1 -> **M2 is faster**. M2 is 3/1.5 = **2 times faster than M1**
Program_2 -> **M1 is faster.** M1 is 10/4 = **2.5 times faster than M2**

b) If the following additional measurements were made:

Program	Instructions executed on M1	Instructions executed on M2
1	$5x\ 10^9$	$6x\ 10^9$

and the clock rates of machine M1 and M2 are 2.5GHz and 2GHz respectively, find the clock cycles per instruction (CPI) for program 1 on each computer.

M1:
IC = $5 * 10^9$
CR = 2.5 GHz
CPU time = 3 seconds
*IC * CPI / CR*
CPI = $3 * 2.5 * 10^9 / 5 * 10^9$
3 * 2.5 / 5
7.5 / 5
CPI = 1.5

M2:
IC = $6 * 10^9$
CR = 2 GHz
CPU time = 1.5 seconds
*CPU * CR / IC*
CPI = $1.5 * 2 * 10^9 / 6 * 10^9$
1.5 * 2 / 6
1.5 / 3
CPI = 0.5

c) Assuming that CPI for program 2 on each computer is the same as the CPI for program 1 found in b), find the instruction count for program 2 running on each computer using the execution times from the first table.

	M1	M2
CPI	1.5	0.5
CPU time	4	10
CR	$2.5 * 10^9$	$2 * 10^9$

IC = CPU time * CR / CPI

M1:
IC = $4 * 2.5 * 10^9 / 1.5$ = **$6.7 * 10^9$**
M2:
IC = $10 * 2 * 10^9 / 0.5$ = **$40 * 10^9$**

2. Consider three different processors P1, P2, and P3 executing the same instruction set. **P1 has a 3 GHz clock rate and a CPI of 1.5**. **P2 has a 2.5 GHz clock rate and a CPI of 1.0**. **P3 has a 4.0 GHz clock rate and has a CPI of 2.2.**

(a) Which processor has the highest performance expressed in instructions per second?

P1 = $3 * 10^9 / 1.5$
= $2 * 10^9$

P2 = $2.5 * 10^9 / 1.0$
= $2.5 * 10^9$

P3 = $4 * 10^9 / 2.2$
= $1.8 * 10^9$

P2 is the largest

(b) If the processor **P1** execute a program in **10 seconds**, find the number of cycles and the number of instructions.

Number of cycles:

$3 * 10^9 * 10$

= $\mathbf{3.0 * 10^{10}}$

Number of instructions:

$3 * 10^9 * 10 / 1.5$

= $\mathbf{2.0 * 10^{10}}$

(c) We are trying to reduce the execution time by **20% for P1**, but this leads to an increase of **30% in the CPI.** What clock rate should we have to get this time reduction for **P1**?

CR = IC * CPI / CPU time
CPU time = IC * CPI / CR

IC * CPI * 130/100 / CPU time * 80/100
IC *CPI * 1.30 / CPU time * 0.80

IC * CPI / CPU time
$3 * 10^9 * 1.30 / 0.80$
= $3 * 1.625 * 10^9$
= $\mathbf{4.875 * 10^9}$ **Hz**
= **4.875 GHz**

3. Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (classes A, B, C, and D). **P1** with a **clock rate 3 GHz** and **CPIs 1, 2, 3, and 3**, and **P2** with a **clock rate 2 GHz** and **CPIs 2, 2, 2, and 2**.

Given a program with a dynamic instruction count of 10^3 instructions divided into classes as follows: **50% class A, 20% class B, 20% class C, and 10% class D**,

	CPI_A 50%	CPI_B 20%	CBI_C 20%	CBI_D 10%	CR	IC
P1	1	2	3	3	$3 * 10^9$	10^3
P2	2	2	2	2	$2 * 10^9$	10^3

(a) What is the global CPI for each implementation?

global CPI = (CPU time * CR) / IC

P1
$0.6 * 10^{-6} * 3 * 10^9 / 10^3$
= 1.8

P2
$1 * 10^{-6} * 2 * 10^9 / 10^3$
= 2

(b) Which is faster: P1 or P2?

CPU time = IC * CPI / CR

P1	P2
10^3*(1*0.50+2*0.20+3*0.20 + 3*0.10) / 3*10^9	10^3*(2*0.50+2*0.20+2*0.20+2*0.10) / 2*10^9
10^3*(0.5 + 0.4 + 0.6 + 0.3) / 3 * 10^9	10^3*(1 + 0.4 + 0.4 + 0.2) / 2 * 10^9
= 10^3*(1.8) / 3 * 10^9	10^3*(2) / 2 *10^9
= 0.6 * 10^{-6}	= 1 * 10^{-6}

P1 IS FASTER

4. It is often needed to exchange the contents of two registers. It takes three instructions to accomplish this in LEGv8. It might be useful to add a new instruction to the LEGv8 ISA that takes two registers and exchanges their contents. Suppose that the CPI is the same across all instruction types, and moreover is unaffected by the new instruction, but that the clock rate decreases by 15%. Suppose that the addition of this new instruction also reduces the total instruction count by 9%. Is it beneficial to implement this enhancement? If yes, what is the overall speedup factor obtained?

Current	IC * 1.0	CR * 1.0	CPI
New	IC * 0.91	CR * 0.85	CPI

0.91 / 0.85

= 1.07058

= 1.1

1.1 > 1 -> The new design will not be beneficial to implement this enhancement

5. Assume that computer M1 has CPU clock rate 2GHz, and supports four classes of instructions: ALU, load, store, branch/jump. Load has CPI = 5, store has CPI =4, ALU instruction has CPI = 4, and branch/jump has CPI = 3. The following program P is used in testing the computer performance:

```
F: SUBI SP, SP, #16 // (*) make room on the stack
   STUR X19, [SP, #0] // preserve X19
   STUR X30, [SP, #8] // preserve the return location
   ADD X19, X2, X3 // calculate c + d
   // Clean up in preparation for tail call
   LDUR X19, [SP, #0]
   LDUR X30, [SP, #8]
   LSL X30, X30, #2
   ORR X30, X30, X19
   ADDI SP, SP, #16 // (*)
   B F // call g(g(a,b),c+d) with tail-call optimization
```

a. What is the average CPI for program P?

\# of load instructions: 2

\# of store instructions: 2

\# of ALU instructions: 3

\# of branch/jump instructions: 1

Total instructions: 8

Load has CPI = 5, store has CPI = 4, ALU has CPI = 4, branch/jump has CPI = 3

AVG CPI = 2 * 5/8 + 2 *4/8 + 3 * 4/8 + 1 *3/8

= 4.13

b. Another computer M2 with clock rate of 3GHz (2000 MHz) has the **same performance** as computer M1 when it runs the program P. What is the average CPI of computer M2?

M1: CPU time = IC * CPI / CR	M2: CPI M2 * .0030 * 10^{-6}
IC * 4.12* (1/3 * 10^9)	1.37 * 10^{-9} = CPI M2 * .0030 * 10^{-6}
= 1.37 * 10^9	= 456.67* 10^{-3} = 0.45667

6. A pitfall is expecting to improve the overall performance of a computer by improving only one aspect of the computer. Consider a computer running a program that requires 250s, with 70s spent executing FP instructions, 85s executed load/store instructions, and 40s spent executing branches instructions.

FP Instructions	L/S Instructions	Branch Instructions	Misc Instruction	Total Exec Time
70s	85s	40s	250-195 = 55s	250s

a. By how much is the total time reduced if the time for FP operations is reduced by 20%?

70 – (20% * 70) = 56s

newExecTime = 85 + 40 + 55 + 56
= 236s

execTimeDecreased = ((250 – 236) / 250) * 100
= 28/5
= 5.6

b. Can the total time can be reduced by 20% by reducing only the time for branch instructions? Why?
= 250 – 250 * (20/100)
= 250 – 250 * 0.20
= 250 – 50
= 200
Cannot reduce 20% of total time for only branch instructions

7. Given the 32-bit binary number:
1101 0011 0110 0000 0001 1100 1010 1111

$2^0+2^1+2^2+2^3+2^5+2^7+2^{10}+2^{12}+2^{21}+2^{22}+2^{24}+2^{25}+2^{28}+2^{30}+2^{31}$

What does it represent respectively, assuming that it is
a. a signed (i.e., already in 2's complement) integer? (convert it to decimal)
- -748675921

b. an unsigned integer? (convert it to decimal)
- 3546291375

c. a single precision floating point number? (write in scientific notation)

1	10100110	110 0000000001 11001010 1111

$1\cancel{6}^{5}6$
-127
39

$110000000001110010101111 \cdot 10^{-39}$

d. a LEGv8 instruction? R type:

1101 0011 0110 0000 0001 1100 1010 1111

$\underbrace{1101\,0011\,011}_{\text{opcode}}\ \underbrace{0\,0000}_{Rs}\ \underbrace{000111}_{\text{shamt}}\ \underbrace{00101}_{Rn}\ \underbrace{01111}_{Rd}$

opcode: LSl — Rs: 0 — shamt: 7 — Rn: x5 — Rd: x15

LSL x15, x5, #7

8. With x = 0000 0000 0000 0000 0000 0000 0101 1011 and y = 0000 0000 0000 0000 0000 0000 0000 1101 representing two signed integers, perform, showing all work:

a. x + y

```
   1 1 1 1
  01011011
+ 00001101
----------
  01101000
```

→ 00000000000000000000000001101000

b. x – y

```
  11110010
+        1
----------
  11110011
```

```
  1  1   1 1
  01011011
+ 11110011
----------
 1]01001110
```

→ 00000000000000000000000001001110

c. x * y

```
      1011011
 •       1101
-------------
      1011011
    101101100
   01011011000
-------------
  10010011111
```

→ 00000000000000000000010010011111

9. Show the IEEE 754 binary representation for the floating point number 20.125 and -5.75 in single and double precision, respectively.

Convert 20.125 and -5.75

20.125

$= 10100.001 \rightarrow 1.0100001 \cdot 2^4$

127 + 4 = 131

SP

0	10000011	01000010000000000000000

1023 + 4 = 1027

DP

0	10000000011	0100001 00 000

52

-5.75

$= -101.11 \rightarrow 1.0111 \cdot 2^2$

$127+2 = 129$

SP	1	1000 0001	0111 0000 0000 0000 0000 0000

$1023+2 = 1025$

DP	1	10000000001	$\underbrace{01111\ 0000\ldots 0}_{S_2}$

10. Perform (20.125) + (- 5.75) and (20.125) x (- 5.75) in binary normalized scientific format obtained from Question #9. Remember to normalize the results.

1.0100001

$101.11 \rightarrow 1.0111 \cdot 2^4$

$$\begin{array}{r} 010.00 \\ + \qquad 1 \\ \hline 010.01 \end{array}$$

$$\begin{array}{r} 001.0100001 \\ +\ 010.0100000 \\ \hline 011.1000001 \cdot 2^4 \end{array}$$

$= 1.11000001 \cdot 2^2$

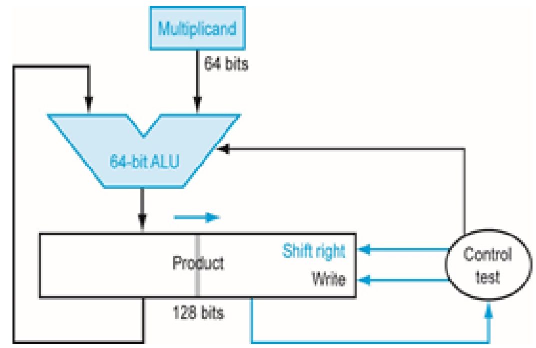
$$\begin{array}{r} 1.0100001 \cdot 2^4 \\ -101.11 \qquad \cdot 2^4 \\ \hline 1\,1\,0\,1\,0\,0\,0\,0\,1 \\ 1\,1\,0\,1\,0\,0\,0\,0\,1\,0 \\ 1\,0\,1\,0\,0\,0\,0\,1\,0\,0 \\ 1 \qquad 0\,0\,0 \\ 1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0 \\ \hline -111.00111011111 \cdot 2^8 \end{array}$$

$7+2$ = 9 decimal spaces

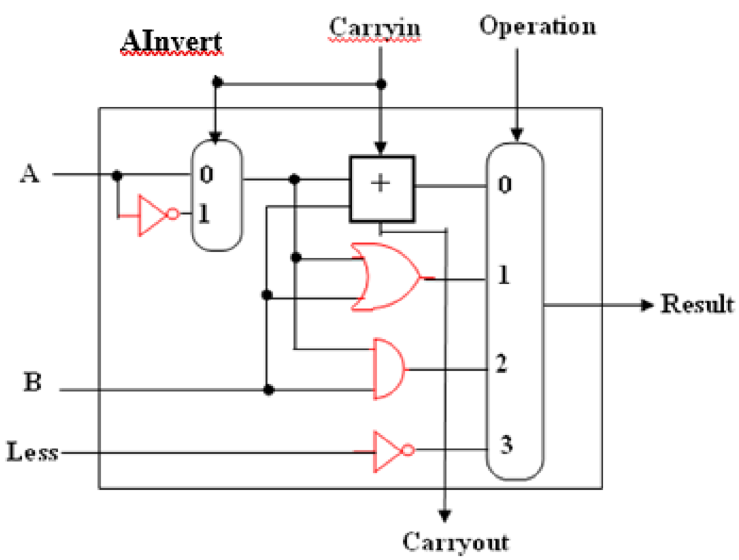
$-1.1100111011111 \cdot 2^6$

11. Multiply 1010 by 0101 using the optimized multiplier. For each step show the contents of the product register and the operation using the following table:

	Operation	Multiplicand	Product
0	Initial value (load multiplier to the lower half of the product register)	1010	0000 0101
1	Add: Prod = Prod + Mcand	1010	1010 0101
	Shift: shift	1010	0101 0010
2	Add: don't add	1010	0101 0010
	Shift: shift	1010	0010 1001
3	Add: Prod=Prod+Mcand	1010	1100 1001
	Shift: Shift	1010	0110 0100
4	Add: don't add	1010	0110 0100
	Shift: Shift	1010	0011 0010



12. Given the following 1-bit ALU, and 3-bit control codes (Ainvert, Operation), write the operations performed corresponding to the 8 control codes in the following table



Ainvert, Operation		Operation performed Result =
0	0 0	A+B
0	0 1	A|B
0	1 0	A & B
0	1 1	Less'
1	0 0	A' + B
1	0 1	A' | B
1	1 0	A' & B
1	1 1	Less'