NAME: _____ STUDENT #: _____

UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

EECE 476: Computer Architecture Quiz 1
October 5, 2012

(Problems continue on the other side of the page … flip this page over).

1. **[5 marks]** Which CPU, Y or Z, is faster? Fully justify your answer.

   ExecutionTime = *InstructionsPerProgram* x *CyclesPerInstruction* x *SecondsPerCycle*

- Program A has 40% arithmetic, 10% load, 20% branch, 30% store instructions.
- Program B has 50% arithmetic, 30% load, 10% branch, 10% store instructions.
- CPU Y is a 1.0GHz CPU where every instruction takes 1 clock cycle.
- CPU Z is 2.6GHz CPU where arithmetic and branch instructions take 2 cycles, stores take 3 cycles, and loads take 4 cycles.

A: Exec(Y)/Exec(Z)=1.0/1.0*2.6/(0.6*2+0.3*3+0.1*4)=2.6/(1.2+0.9+0.4)=2.6/2.5 ➔ Z is faster
B: Exec(Y)/Exec(Z)=1.0/1.0*2.6/(0.6*2+0.1*3+0.3*4)=2.6/(1.2+0.3+1.2)=2.6/2.7 ➔ Y is faster
Overall: arithmetic average says both are same speed. Geomean: can't compute without calculator.

2. In the following code sequence, assume each variable is initially stored in memory:

```
A = B * C
A = A + D
```

(a) [3 marks] Invent your own assembly mnemonics and write the best equivalent assembly language for the above code for a **stack** architecture. Your assembly code must store the final value of A in memory **and** be as short as possible.

There are many possible solutions. Here are two:

```
PUSH B          or      LI #B
PUSH C                  LOAD
MUL                     LI #C
PUSH D                  LOAD
ADD                     MUL
STORE D                 LI #D
                        LOAD
                        ADD
                        LI #d
                        STORE
```

(b) [2 marks] Repeat part (a), but for a **register-register** architecture.

```
LD R2, B(R4)
LD R3, C(R4)
MUL R2, R2, R3
LD R3, D(R4)
ADD R2, R2, R3
SW R2, A(R4)
```

NAME: _____ STUDENT #: _____

3. **[10 marks]** A sequence of load and store memory addresses is given in the table below. The sequence starts at the left (L1) and ends at the right (S17). Each of the addresses is given as a word address (shift left by 2 if you want the byte address). Loads are indicated with L and stores are indicated with S. The cache is direct-mapped with 4 lines, 4 words per line, using a **write-back** policy. On a store-miss, this cache must bring the data into the cache (**write-allocate**). Show all of your work for full credit.

   a. **[3 marks]** For each memory reference, **indicate** a cache MISS (**M**) or hit (**H**).

   b. **[3 marks]** For each memory reference, **indicate activity** between the cache and main memory: the cache reads data from memory (**R**), writes data to memory (**W**), write-followed-by-read (**WR**), or leaves memory idle (use a dash: **–**).

	Memory Address Sequence (addresses also given in binary)									
	L1 00001	S2 00010	L3 00011	L5 00101	S13 01101	L13 01101	S17 10001	L6 00110	L0 00000	S17 10001
b) Indicate **M**,**H**	M	H	H	M	M	H	M	H	M	M
c) Indicate **R**,**W**,**WR**,**–**	R	—	—	R	R	—	WR	—	WR	R
Offset #	01	10	11	01	01	01	01	10	00	01
Index	00	00	00	01	11	11	00	01	00	00
TAG	0	0	0	0	0	0	1	0	0	1

***Note**: in the above table, the Offset #, Index, and TAG rows will not be marked.*

   c. **[3 marks]** Give the **final cache contents**. In addition to the latest TAG and STATE, be sure to **write the last access (eg, L1)** in the "Offset" columns of the table below. The solution for the last two rows is provided. Be sure to indicate final **valid** and **dirty** bits.

	TAG	V, D State		Word Offset 11	Word Offset 10	Word Offset 01	Word Offset 00
		V	D				
Index 00	1	1	1			S17	
Index 01	0	1	0		L6	L5	
Index 10		0	0				
Index 11	0	1	1			S13	

   d. **[1 mark]** What is the total cache capacity? (show your work)

Total Capacity = 4 words/line * 4 lines * 4 bytes/word = 64 bytes.