CS 2200 Homework 1

Fall 2019

Rules:

- Please print a copy of the assignment and hand write your answers. No electronic submissions are allowed. **Please print as one double-sided page. Do NOT staple multiple sheets together. There will be a 10 point penalty if you do this improperly.**
- This is an individual assignment. No collaboration is permitted.
- Due Date: **September 4th 2019 – 6:05 PM** . Bring your BuzzCard. Show up on time.

Name:_______________________________ GT Username:____________ Section____

1) Give two reasons why it is preferable to use registers over making memory accesses.

2) What are two differences between a Complex Instruction Set (CISC) ISA and a Reduced Instruction Set (RISC) ISA? Which type of ISA is LC-2200?

3) For the struct defined below, show how a smart compiler might pack the data to **minimize wasted space** and follow alignment restrictions. Pack in such a way that you can **guarantee aligned accesses** to all the elements of the struct. Assume the compiler **cannot** intelligently reorder fields of the struct in memory. Assume a char is 1 byte, int is 4 bytes, and a short is 2 bytes. Moreover, assume the architecture is **little endian** and its addressability supports load word, load byte, and load half word instructions.

```
struct x {
    int b;      // value 0x13E7C0DE
    char a;     // value 0x96
    short c[2];  // values {0xFEFE, 0xBABE}
    int d;      // value 0xB0BACAFE
};
```

In the following memory picture each row represents a memory word comprising of 4 bytes, and each cell represents a byte. You do not necessarily need to use all rows. Write each byte in with the hexadecimal values from the comments above.

+3	+2	+1	+0	Starting address
				0x1000
				0x1004
				0x1008
				0x100C

4) Fill in the missing lines below. The LC-2200 assembly program should increment the value in the memory location **pointed to by x** (assume x is already in $s0) from 1 to 10. The C code is provided below. Some operands and instructions are given.

```
int x = 0;
for (int i = 10; i > 0; i--) {
    x += 1;
}
     addi $t0, $zero, _____ # loop counter
     addi $t1, _____, _____ # loop limit

loop:   beq $t0, $t1, _____
        lw _____ , 0x0(_____)
        addi _____ , $t2, 1
        _____ _____ , _____($s0)
        _____ $t0, $t0, _____
        _____ $zero, $_____, loop

bye: ...
```