CS 2200 Homework 1

Fall 2018

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.**
- This is an individual assignment. No collaboration is permitted.
- Due Date: **5th September 2018 – 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) A UGA CS student thinks either the caller or the callee should be solely responsible for saving and restoring all the registers in the procedure calling convention. Explain why you think they are right or wrong.

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 supports load word, load byte, and load half word instructions.

```
struct x {
    int b;      // value 0xB1E555ED
    char a;     // value 0x96
    short c[2];  // values {0xC0DE, 0xBABE}
    int d;      // value 0x5E1F1E55
};
```

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

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). The C code is provided below. Some operands and instructions are given.

```
int x = 0;
for (int i = 16; i > 0; i -= 2) {
    x += i;
}
     addi $t0, $zero, ____# loop counter
     addi $t1, ____, ____ # loop limit

loop:   beq $t0, $t1, ________
        lw ____, ____(____)
        add ____, ____, ____
        ____ ____, 0x0($s0)
        ____ $t0, $t0, ____
        ____ ____, ____, loop

bye: ...
```

4) The following is a function call in assembly using the LC-2200 instructions but with some new registers (`$r1 - $r5`) and its own calling convention. Label each register listed below as caller-saved or as callee-saved based on the code. Assume `$sp` is the stack pointer register and `$at` already contains the starting address of the function (labeled func).

```
             ...
             addi $sp, $sp, -1
             sw $r3, 0($sp)
             addi $sp, $sp, -1
             sw $r5, 0($sp)
             jalr $at, $zero
             ...

 func:       addi $sp, $sp, -1
             sw $r4, 0($sp)
             addi $sp, $sp, -1
             sw $r2, 0($sp)
             addi $sp, $sp, -1
             sw $r1, 0($sp)
```

$r1 ____________________

$r2 ____________________

$r3 ____________________

$r4 ____________________

$r5 ____________________