EE 109 Homework 5

Name: ___________________________________________  Score: ________

For this HW, enter your answers on Blackboard.
Be sure to follow any formatting instructions.

See Blackboard for Due Date.

Assembler Directives
1.) (10 pts.) Examine the following C program variable declarations and translate them to the appropriate directives.

Submission Instructions: Enter the number of your desired selection.

a. short int x = 6;
   1. x:  .word 6
   2. x:  .half 6
   3. x:  .space 6
   4. x:  .align 6

b. unsigned char msg[8] = {1,4,9,7,3,6,8,2};
   1. msg:  .unsigned 1,4,9,7,3,6,8,2
   2. msg:  .byte 0x14973682
   3. msg:  .byte 1,4,9,7,3,6,8,2
   4. msg:  .half 1,4,9,7,3,6,8,2

c. int data[100];
   1. data:  .word 100
   2. data:  .space 100
   3. data:  .space 200
   4. data:  .space 0x190

d. char str[] = “hello\n”
   1. str:  .ascii “hello\n”
   2. str:  .asciiz “hello\n”
   3. str:  .byte “hello\n”
   4. str:  .asciiz “hello”
HLL to Assembly Translation

2.) (30 pts.) Translate the following C code statements to an equivalent assembly language implementation.

Submission Instructions: For constants/numbers enter your values in decimal. For registers, use the descriptive mnemonic (i.e. $t0, $s1). For labels, just enter it as shown. For opcodes, enter the full opcode (e.g. b__ => enter ‘bgt’).

```c
short data[20];
...
for(int i=0;i < 20; i++){
    data[i] = data[i] + 5;
}

// let i be stored in $t0
```

```assembly
.data .space   ________
.text
...  
    la $s0,data
    add $t0,____,______
    li $t4,___________
L0:   slt $t1,____,$t4  
L1:   b___ $t1,$zero,______
L2:   sll $t2,$t0,____
L3:   add $s1,____,$t2
L4:   lh $t3,________
L5:   addi $t3,____,5
L6:   sh $t3,_______
L7:   addi $t0,____,___
L8:   b      ________
L9:   ...
```

```c
int *x, *y;
...
if(*x >= 10 && *y < 5)
    code A
else if (*x < 10 || *y > 5)
    code B
else
    code C
...
```

```assembly
x .space  4
y .space  4
.
...
    la $s0,x
    la $s1,y
    addi $t0,_____10
    addi $t1,_____5
    lw $s2,0($s0)
    lw $s3,0($s1)
    lw $t2,______
    lw $t3,_______
    blt ___,$t0,____
    __ $t3,_______
L1:   Code A instructions
L2:   b ____________
L3:   b___ $t2,$t0,____
L4:   b__ $t3,$t1,____
L5:   Code B instructions
L6:   b ____________
L7:   Code C instructions
L8:   ...
Subroutines and Assembly
3.) (20 pts.) In addition to passing arguments in registers or by using the stack, another possible method (that no one would ever use but is interesting “academically” speaking) is to provide space for the arguments in the code itself (though you should note that this code will NOT assemble using MARS because it does not handle data directives in the text section). However, please study the code below to understand how this method works and answer the given questions. **SHOW ALL VALUES in HEX. Assume that the .data section starts at 0x10010000.**

a.) **Before execution of “jal AVG”,** what are the contents of $t0, $t1, and $t2.
b.) **Before execution of “sra $v0,$v0,1”,** what are the contents of $v0, $t3, & $ra?
c.) **After program execution,** what are the contents of $ra and the word at address RES.

Submission Instructions: Enter the hex values into the appropriate blanks.

```
.data
VALS: .half 0xbead, 0xface
RES: .space 4
.text
la $t0, VALS
lh $t1, ($t0) a.) $t0 = 0x_____
sll $t2, $t2, 16
or $t2, $t1, $t2
jal AVG
lhu $t2, 2($t0) $t1 = 0x______
sl1 $t2, $t2, 16 $t2 = 0x______
or $t2, $t1, $t2
0x400014
 jal AVG
0x400018
.word -2
0x400020
.word -6
la $t4, RES
sw $v0, ($t4)
li $v0, 10
syscall AVG:
lw $v0, ($ra) c.) $ra = 0x________
lw $t3, 4($ra) M[RES] = 0x______
add $v0, $v0, $t3
sra $v0, $v0, 1
addi $ra, $ra, 8
jr $ra
```

a.) $t0 = 0x______
b.) $v0 = 0x______
$ra = 0x______
c.) $ra = 0x______
M[RES] = 0x______