Name:____

Problem 1: (20 points) Short Answers (10 minutes)

1. **(5 points)**: What is the result in register $t0 after these lines of MIPS assembly code are executed?

```
add $t0, $zero, $zero
ori $t0, $t0, 0xC3C3C3C3
andi $t0, $t0, 0xBBBBBBBB
ori $t0, $t0, 0x2A2A2A2A
```

a) 0x83838383
b) 0x02020202
c) 0xABABABAB
d) 0x2A2A2A2A
e) none of the above

2. **(5 points):** Assuming the MIPS memory model, i.e. memory is byte addressed, the processor works on 32-bit (word) data and word accesses must be word aligned, which of the following hexadecimal memory addresses accesses are valid?

a) 0x12345678
b) 0x24A19E8A
c) 0xAAABBC23
d) 0x84F2FFED
e) 0x00004400

(5 points): Given a 32-bit memory address in hexadecimal, describe a method for determining whether it is word aligned.

3. **(5 points)** Name the 5 components of a computer

Name:____

Problem 2: (10 points) Fill-in the Blanks (10 minutes)

Given the C code below:

```
int x,y,err;

... //the code to input the value of x is not shown

if(x<10)
     err = 0;

else {
     x = 0;
     err = 1;
     }

y = x;
```

Fill in the lines for the following MIPS code so that it will execute the C code properly. Each line can only be one instruction. Labels are given in front of every line of assembly code so that you don't have to add any. Assume the variables are stored as follows:

```
#$s0 = int x;
#$s1 = int y;
#$s2 = int err;
```

```
L1: ______________________
L2: beq $t0, $0, L5
L3: add $s2, $0, $0
L4: _______________________
L5: add $s0, $0, $0
L6: addi $s2, $0, 1
L7: _______________________
```

Name:____

Problem 3: (30 points) Pseudo-instructions (15 minutes)

Pseudoinstructions are not actually part of the MIPS instruction set, but are often used to when writing MIPS assembly programs. These pseudoinstructions are then translated by the assembler into a sequence of "true" MIPS assembly instructions. For each pseudoinstruction below, produce the shortest possible sequence of true MIPS instructions. You can only use one additional register $at to store temporary values. *You must use true MIPS instructions i.e. any instruction from the green card.*

1. **(10 points)** constant multiply – Write the shortest MIPS instruction sequence for multiplication by 32 and 15 using only shifts, add and subtract instructions.
    a. `mul32` – Multiply the operand stored in register `$t2` by the constant 32 and store the result in register `$t1`. You may assume that we only care about least significant 32 bits for the resultant operand. In other words, we are creating the pseudo-instruction `mul $t1, $t2, 32`

    b. `mu15` – The same as mul32, except now the constant is 15 i.e. `mul $t1, $t2, 15`

Name:____

2. **(10 points)** branch less than equal –.Branch to “Label” bif the operand in register \$t0 ≤ the operand in register \$t1. In other words, if(\$t0 ≤ \$t1) goto Label, which performs the pseudo-instruction “`blte $t0, $t1, Label`”

3. **(10 points)** absolute value addition - Write the shortest MIPS instruction sequence to add two 2’s complement operands as their absolute values. In other words, we are like creating a pseudo-instruction “`addabs $t1, $t2, $t3`”, which performs the operation `$t1 = |$t2| + |$t3|` i.e. take the absolute value of the operand in register `$t2`, add it to the absolute value of operand in register `$t3`, and store the result in register `$t1`.

Name:____

Problem 4: (25 points) Understanding MIPS Programs (20 minutes)

```
mastershake: addi $t0,$a2,1
frylock: bge $t0,$a1,meatwad
mul $t1,$t0,4
add $t1,$t1,$a0
lw $t2,0($t1)
sub $t1,$t1,4
sw $t2,0($t1)
addi $t0,$t0,1
j frylock
meatwad:
```

a) (20 points): Translate the *mastershake* assembly code above into a high-level language like C or Java. You should include a header that lists the types of any arguments and return values. Also, your code should be as concise as possible, without explicit pointers. We will not deduct points for syntax errors unless they are significant enough to alter the meaning of your code. You are not allowed to use go to statements; go to statements are harmful (see below).

b) (5 points): Describe briefly, in English, what this function does.

c) (5 bonus points): In 1968, I wrote a famous letter to the *Communications of the ACM* entitled “Go To Statement Considered Harmful” that argued that go to statements should not be used in high level languages. Who am I? Hint: I’m also the author of a well known shortest path algorithm.

Name:____

Problem 5: (30 points) Compilation (20 minutes)

The following lines of code find the smallest integer in the array V, which has n elements.

```
int V[], int n;

...

int min, i;
min = V[0]; // min is initialized with the first element of V
for (i=1; i<n; i++)
      if (V[i] < min) // Found an element smaller than the current min
             min = V[i];
```

Assuming assignment of variables,

Variable	Register
the base address of V	$a0
n	$a1
min	$s0
i	$s1

Write the MIPS assembly code that correctly executes the code.