Name:_____

Problem #1: Short Answers (25 points)

a) (5 points) Fill in the blank to force the assembler to use two instructions to implement this instruction.

```
addi $t0,$t1, _____
```

b) I/O definitions (10 points)

(3 points) Explain the concept of polling in 2 sentences or less

(3 points) Explain the concept of interrupts in 2 sentences or less

(4 points) Would a desktop computer use interrupts or polling to handle the mouse input? Explain your answer

Name:_____

c) C Debugging (5 points)

Consider the following definition of `append`.

```
/*
    Return the result of appending the characters in s2 to s1.
    Assumption: enough space has been allocated for s1 to store
    the extra characters.
*/
char* append (char s1[ ], char s2[ ]) {
    int s1len = strlen (s1);
    int s2len = strlen (s2);
    int k;
    for (k=0; k<s2len; k++) {
        s1[k+s1len] = s2[k];
    }
    return s1;
}
```

The code has a bug. Identify it, explain why it is a bug and modify the code to fix it.

Name:_____

d) Executing a program (10 points)
Following is a list of tasks that must be done to run a program written in C. Sort the tasks into the order in which they are executed and for each step, indicate which tool performs the step (where the tool is the compiler, assembler, linker, or loader).

a. Store resulting code in a .s file
b. Combine text and data segments from multiple input files.
c. Store resulting code in an executable file
d. Load text and data segments in to memory.
e. Compute offsets for all branch instructions.
f. Compute absolute address for all relocation table entries, rewrite instruction(s) to hold the new address.
g. Replace pseudoinstructions with one or more real instructions.
h. Translate C code to equivalent assembly language code.
i. Store resulting code in a .o

Write your answers here:

1. Task: _____Tool:_____

2. Task: _____Tool:_____

3. Task: _____Tool:_____

4. Task: _____Tool:_____

5. Task: _____Tool:_____

6. Task: _____Tool:_____

7. Task: _____Tool:_____

8. Task: _____Tool:_____

9. Task: _____Tool:_____

Name:_____

Problem #2: Number Representation (20 points)

Take the number -61 (base ten) and convert it to
a) sign-magnitude
b) one's compliment
c) two's compliment
d) IEEE 754 floating-point standard (the one discussed in class)

Each part is worth 5 points. All answers should be in 32 bit form. Reminder: A 32 bit floating point number has 1 sign bit, 8 bits for the exponent and 23 bits for the significand.

a)

b)

c)

d)

Name:_____

Problem #3: C Programming (25 points)

Write a C function that takes as a parameter a character string of unknown length, which will contain exactly one word. Your function should return a **new** string, which translates the given string into Pig Latin (igPay atinLay). You are not allowed to modify the original, parameter string.

A quick reminder on the rules of Pig Latin:

- Words that start with a vowel simply have "way" appended to the end of the word e.g. "enter" is translated to "enterway"
- Words that start with a consonant have all consonant letters up to the first vowel removed to the end of the word (as opposed to just the first consonant letter) and "ay" is appended. For example, "the" is translated to "ethay"

The function prototype is char * pig_latin(char *in)

You have one library function called bool is_vowel(char c), which returns 'true' if c is a vowel and 'false' otherwise. Assume that this function is smart enough to determine when 'y' is used as a vowel and when 'y' is used as a consonant. You are **NOT** allowed to use any other string manipulation functions.

You do not have to worry about any sort of error checking; assume that the passed in string is always correct (exactly one word, no invalid characters, whitespace, etc). Also, there is no need to worry about capitalization.

Name:_____

This page was intentionally left blank

Name:_____

Problem #4: Assembly Language (25 points)

The factorial operation N! = 1 * 2 * 3 * … * N-1 * N can be implemented using a simple recursive function *factorial*. Translate recursive factorial function into MIPS assembly language (MAL i.e. you are free to use any pseudoinstructions). **Your function must follow all MIPS function calling conventions (argument passing, return variables, saved registers, etc.).**

```
int factorial(int N)
{
        if(N > 1)
                return N * factorial(N-1);
        else
                return 1;
```

Name:_____

Problem #5: Decompilation (25 points)

Here is a mystery function. It expects one non-negative integer argument and returns one integer result.

```
hmmm:
      li $t0, 1
      li $t1, 1
loop:
      blt $a0, 2, exit
      add $t2, $t0, $t1
      move $t0, $t1
      move $t1, $t2
      addi $a0, $a0, -1
      j loop
exit:
      move $v0, $t1
      jr $ra
```

a) (20 points) Translate the *hmmm* function into C. Do not use "goto." We will not deduct points for syntax errors *unless* they are significant enough to alter the meaning of your code.

b) (3 points) What will this function return if it is called with an argument ($a0) of 4?

c) (2 points) Describe, in English, what this function computes.