Name:_____

Problem 1: (25 points) Short Answers

a) **(5 points)** Executing a program

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:_____

b) **(5 points)** Explain how a machine knows whether to interpret a group of bits as an integer, a floating point number, or an instruction.

c) **(5 points)** What is abstraction? How does it relate to instruction set architectures (ISAs)?

d) **(5 points)** Name the 4 regions of a program's memory space

e) **(5 points)** Name the 5 classic components of a computer

Name:_____

Problem 2: (20 points) Number Representations

a) **Floating Point (10 points)**

a. **(5 points)** Translate the decimal number 28.5 to a 32 bit IEEE 754 single precision floating-point number. Your answer should be a ***hexadecimal number.***

b. **(5 points)** Translate `0x87654321` (a 32-bit IEEE 754 floating point number) to a binary number in normalized scientific notation form. E.g. I want the answer in the form $1.100010001 \times 2^{345}$

b) **(10 Points) Hexadecimal to MIPS Conversion**

a. **(5 points)** Translate the hexadecimal number `0x00041080` to a MIPS instruction using register names, e.g. $s0, $t0, …

b. **(5 points)** Translate the hexadecimal number `0xAC240204` to a MIPS instruction using register names, e.g. $s0, $t0, …

Name:_____

Problem 3: (30 points) Compilation

The goal of this problem is to write a MIPS function *flipimage* which flips an image horizontally. For example, a simple image is shown on the left, and its flip is shown on the right.

A picture is composed of individual dots, or pixels, each of which will be represented by a single byte. The entire two-dimensional image is then stored in memory row by row. For example, we can store a 4 x 6 picture in memory starting at address 1000 as follows:

- The first row (consisting of 6 pixels) is stored at addresses 1000-1005.
- The second row is stored at addresses 1006-1011.
- The third row is stored at 1012-1017
- The last row is stored at addresses 1018-1023.

a. **(15 points)** Write a MIPS function fliprow to flip a single row of pixels. The function has two arguments, passed in $a0 and $a1: the address of the row and the number of pixels in that row. There is no return value. Be sure to follow all MIPS calling conventions.

b. **(15 points)** Using the fliprow function, you should now be able to write flipimage. The arguments will be:

- The memory address where the image is stored
- The number of rows in the image
- The number of columns in the image

Again, there is no return value, and you should follow normal MIPS calling conventions.

Name:_____

Problem 4: (60 points) MIPS Reverse Compilation

Consider the following MIPS assembly code:

```
scooby:
      ori $t0, $0, 0
      ori $t1, $0, 0
      lui $at, 0x3FFF
      ori $t2, $at, 0xFFFF
shaggy:
      slt $at, $t0, $a1
      beq $at, $0, velma
      ori $at, $0, 4
      mul $t3, $t0, $at
      add $t3, $a0, $t3
      lw $t3, 0($t3)
      slt $at, $t1, $t3
      beq $at, $0, fred
      addu $t1, $0, $t3
fred:
      slt $at, $t3, $t2
      beq $at, $0, daphne
      addu $t2, $0, $t3
daphne:
      addi $t0, $t0, 1
      j shaggy
velma:
      sw $t1, 0($a2)
      sw $t2, 0($a3)
      jr $ra
```

a) **(30 points)** Translate the function `scooby` above into a high-level language like C or Java. Your function header should list the types of any arguments and return values. Also, your code should be as concise as possible, without any gotos or pointer arithmetic. We will not deduct points for syntax errors unless they are significant enough to alter the meaning of your code.

Name:_____

b) **(10 points)** Describe briefly, in English, what this function does.

c) **(20 points)** Convert the following instructions from the code above into 32 bit hexadecimal number. Assume that the address of the first instruction (`ori $t0, $0, 0`) is located at address `0x00400028`.

(5 points) `beq $at, $0, velma`

(5 points) `j shaggy`

(5 points) `lw $t3, 0($t3)`

(5 points) `slt $at, $t3, $t2`

Name:_____

Problem 5: (95 points) Data Structures and Recursion

The following structure is similar to the one from project #3. It is used for both a binary tree and a double linked list.

```
struct node {
      int value;
      struct node * left;
      struct node * right;
      struct node * previous;
      struct node * next;
      } * nodePtr;
```

a) **(5 points)** Assuming we compile to the MIPS processor discussed in class, what is the size of a single node structure, i.e. what value would a call the `sizeof(struct node)` return?

b) **(20 points)** Translate that following memory layout to the pictorial description of a tree and list. The variable "listHead" is 0x00123AF0. The variable treeRoot is 0xE123014C.

Address	Value
0xFFFFFFFF	. . .
	0x00000000
	0x472AF014
	0x041ABC28
	0x472AF014
0xE123014C	0x00000001
	. . .
	0x041ABC28
	0x00123AF0
	0x00000000
	0x00000000
0x80042AB0	0x00000003
	. . .
	0xE123014C
	0x041ABC28
	0x00123AF0
	0x80042AB0
0x472AF014	0x00000002
	. . .

Address	Value
	. . .
	0xE123014C
	0x00123AF0
	0x472AF014
	0x80042AB0
0x123ABC00	0x00000006
	. . .
	0x472AF014
	0x80042AB0
	0x00000000
	0x00000000
0x041ABC28	0x00000004
	. . .
	0x80042AB0
	0x00000000
	0x00000000
	0x00000000
0x00123AF0	0x00000005
	. . .

Name:_____

c) **(15 points)** Below is a recursive function called `listCount` that traverses the list and returns the number of elements in the list.

```
int listCount(struct node * l ) {
 if ( l == NULL ) return 0;
 else return 1 + listCount( l->next ); }
```

Convert `listCount` to MIPS assembly. You must exactly translate the code above, i.e. you should not try to optimize it and it must be recursive. Also, you must follow all of the MIPS procedure conventions. Failure to do either of these will result in a significant loss of points.

d) **(5 points)** What does `listCount(0x00123AF0)` return?

e) (5 points) How many times does `listCount(0x00123AF0)` get called recursively (do not include the initial call)?

f) (5 points) What is the maximum number of `listCount` on the stack at any point during the execution of `listCount(0x00123AF0)` (do not include the initial call)?

Name:_____

g) **(25 points)** Below is a recursive function called `maxDepth` that traverses the tree and returns the sum of all of the values in the tree.

```
/* Compute the "maxDepth" of a tree -- the number of nodes
along
the longest path from the root node down to the farthest leaf
node. */

int maxDepth(struct node * node) {
  if (node == NULL) {
    return(0); }
  else {
    // compute the depth of each subtree
    int lDepth = maxDepth(node->left);
    int rDepth = maxDepth(node->right);

    // use the larger one
    if (lDepth > rDepth) return(lDepth+1);
    else return(rDepth+1);
  }
}
```

Convert `maxDepth` to MIPS assembly. You must exactly translate the code above, i.e. you should not try to optimize it and it must be recursive. Also, you must follow all of the MIPS procedure conventions. Failure to do either of these will result in a significant loss of points.

Name:_____

h) **(5 points)** What does `maxDepth(0xE123014C)` return?

i) **(5 points)** How many times does `maxDepth(0xE123014C)` get called recursively (do not include the initial call)?

j) **(5 points)** What is the maximum number of `maxDepth` on the stack at any point during the execution of `maxDepth(0xE123014C)` (do not include the initial call)?

Extra Credit: (10 bonus points) History of the Computing World: Part I

1. **(2 points)** Who was the first person to add "control" to a machine?

2. **(2 points)** What were the names of Charles Babbage's two machines?

3. **(6 points)** What is a Universal Turing Machine (UTM)? Use a UTM to prove that the halting problem is not decidable using the Turing Machine model.