Quiz 4

Name: _____ Student ID: _____

Do Not Open Until Instructed!

Before the Quiz starts:

- Read all of the instructions on this page
- Write your name and student ID on this page
- Locate your page of notes, if you have one
- Prepare your writing materials
- Put all other materials away

After the Quiz starts:

- Write your student ID (**not** your name) on all subsequent pages
- Announcements / corrections will appear on the projector
- Turn in your quiz and note page to Drew when finished.
- After the quiz time expires, answers may be presented but no new material will be given.

The quiz consists of 5 questions. You will have *35 minutes* to complete all questions. Work quickly and move on if you are stuck. If you'd like to pass the time before the quiz starts or before it ends, feel free to draw a picture of yourself in the box below:

Student ID: _____

Question 1 (2 Points)

Consider the following program snippet:

```
 1.  int glob = 1;
 2.  void goofy(int arg1, int arg2){
 3.   int a = 5;
 4.     glob = 6;
 5.     arg1 = arg2 + 7;
 6.  }
 7.  int main(){
 8.     int a = 2;
 9.     int b = 3;
10.     goofy(a, glob);
11.     print a + " " + b + " " + glob + "\n";
12.  }
```

What does the program print under each of the following parameter passing schemes?

Pass by value:

Pass by reference:

Pass by name:

You may choose to make reasonable assumptions about unspecified program semantics. If any of these assumptions are non-obvious, describe them below.

Student ID: _____

Question 2 (2 Points)

Consider the following program snippet:

```
 1.  int var1 = 10;
 2.  int var2 = 20;
 3.  void first(){
 4.     int var1 = 30;
 5.     void second(){
 6.        print var1;
 7.        print " ";
 8.        print var2;
 9.        print "\n";
10.     }
11.     second();
12.  }
13.  int main(){
14.     int var1 = 40;
15.     int var2 = 50;
16.     first();
17.  }
```

What does the program print under each of the following scoping schemes?

Static scoping:

Dynamic scoping:

You may choose to make reasonable assumptions about unspecified program semantics. If any of these assumptions are non-obvious, describe them below.

Student ID: _____

Question 3 (2 Points)

A common optimization trick is specialize compilation for "leaf functions", i.e. those functions that have no callees. What data is typically kept in an activation record that would not need to be tracked in a leaf function? Explain why leaf functions do not need to keep track of that data.

Student ID: _____

Question 4 (2 Points)

Recall that in MIPS, the compiler might take advantage of "delay slots" when scheduling instructions. Assume the following block of code:

```
1. lw $t0 0($sp)
2. nop
3. addi $t0 $t0 1
4. addi $t1 $t0 1
5. addi $t2 $t2 1
```

This code assumes a single-instruction delay slot in the load instruction, and hence puts a nop (no operation) at that instruction. Is it possible to take advantage of the load slot by rearranging the instructions above and eliminating the nop? If so, write out the sequence(s) of rearranged instructions. If you have any assumptions about MIPS semantics, describe them as well.

Student ID: _____

Question 5 (2 Points)

Assume an architecture in which the stack grows upwards (towards high memory). Implement the push and pop instructions in MIPS. If you have any assumptions about AR conventions, write them out as well (e.g. the stack pointer points to a free memory address).

MIPS Reference

In the below ("regX") refers to a register, imm refers to an constant (immediate) value.

`jal <Label>`
Jump to address stored at <Label>, set $ra to address of the next instruction.

`jr <reg>`
jump to the address held in <reg>

`sub <reg1> <reg2> <reg3>`
reg1 = reg2 - reg3

`add <reg1> <reg2> <reg3>`
reg1 = reg2 + reg3

`subi <reg1> <reg2> <imm>`
reg1 = reg2 - imm

`addi <reg1> <reg2> <imm>`
reg1 = reg2 + imm

`lw <reg1> <imm>(<reg2>)`
Set <reg1> to the value at memory address <reg2> + <imm>

`sw <reg1> <imm>(<reg2>)`
Set the value at memory address <imm> + <reg2> to the value in <reg1>