Problem Set 3
Computer Science 240
Fall 2014
Due: Friday, September 26

Relevant Reading. Patterson and Hennessy §2.7 – §2.8

Problem 1. Construct a control flow graph (like the one shown in Figure 2.9 of the text) for the following section of C or Java code:

```c
for (int i = 0; i < x; i = i + 1) {
    y = y + i;
}
```

Translate this code to MIPS. Assume that i, x, and y correspond to registers $s1, s2, and $s3, respectively.

Problem 2. The fragment of MIPS code below was compiled from a Java or C program that has since been lost. Assume $s3$ represents local variable i, $s4$ represents local variable j, and $s5$ holds the address of array A, where i, j, and the elements of A are 32-bit 2 complement integers. Assume also that a standard array representation is used with the 0 entry starting at the address of the array.

```
Loop:
  addi $s4, $s4, 1
  addi $s3, $s3, 1
  sll $t1, $s3, 2
  add $t1, $t1, $s5
  lw $t0, 0($t1)
  slti $t1, $t0, 10
  beq $t1, $zero, Loop
  slti $t1, $t0, 0
  bne $t1, $zero, Loop
```

The programmer recalls that the Java or C code originally had the form:

```java
do {
    something
} while (something else);
```

Reconstruct Java or C code that could have been compiled to the MIPS code above.

Problem 3. Add comments to the following MIPS code and describe in one sentence what it computes. Assume that $a0$ and $a1$ are used for the input and both initially contain the integers $a$ and $b$, respectively. Assume that $v0$ is used for the output. What does the program compute?
Problem 3

add $t0, $zero, $zero
loop:  beq $a1, $zero, finish
        add $t0, $t0, $a0
        sub $a1, $a1, 1
        j  loop
finish: addi $t0, $t0, 100
        add $v0, $t0, $zero

Problem 4. The following program tries to copy words from memory starting at the address in register $a0 to memory starting at the address in register $a1, counting the number of words copied in register $v0. The program stops copying when it finds a word equal to 0. This terminating word should be copied but not counted. The program does not need to preserve the contents of registers $v1, $a0, or $a1.

addi $v0, $zero, 0  # Init to avoid counting 0 word
loop: lw, $v1, 0($a0)  # Read next word from source
        sw   $v1, 0($a1)  # Write to destination
        addi $a0, $a0, 4  # Advance to next source
        addi $a1, $a1, 4  # Advance to next destination
        beq  $v1, $zero, loop # Loop if word copied != zero

There are multiple bugs in this MIPS program; fix them and write a bug-free version.

Problem 5. Write a MIPS procedure to compute the $n$th Fibonacci number $F(n)$ where

\[
F(n) = \begin{cases} 
0, & \text{if } n = 0; \\
1, & \text{if } n = 1; \\
F(n-1) + F(n-2), & \text{otherwise.}
\end{cases}
\]

Base your algorithm on the straightforward but inefficient procedure given below:

```c
int fib(int n){
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fib(n-1) + fib(n-2);
}
```

You have already seen one version of a program Fibonacci.asm in laboratory. However take care, that version is not recursive and the goal of this assignment is to write your first recursive procedure. To help get you started, a skeleton for this program is given below, also available in a separate file, FibRec.asm, on the website. It prompts the user for an integer, then calls the recursive procedure fib. Your job is to write this procedure. The comments given at the head of the procedure give both the algorithm to be implemented and the register usage.
# Skeleton for Problem 5.
# Description: Computes Fibonacci function using a recursive process.
# Function: \( F(n) = \)
# 0, if \( n = 0 \);
# 1, if \( n = 1 \);
# \( F(n-1) + F(n-2) \), otherwise.
# Input: \( n \), which must be a nonnegative integer.
# Output: \( F(n) \).
# Preconditions: none
#
# Algorithm for main program:
# print prompt
# call fib(read) and print result.
# Register usage
# in main program:
#   $a0 = n (passed directly to fib)
#   $s1 = f(n)
#
.data
.align 2
# Data for prompts and output description
prmpt1: .asciiz "\n\nThis program computes the Fibonacci function."
prmpt2: .asciiz "\nEnter value for n: "
descr: .asciiz "fib(n) = "
.text
.align 2
.globl main
main:
# Print the prompts
   li $v0, 4 
   la $a0, prmpt1
   syscall
   li $v0, 4 
   la $a0, prmpt2
   syscall
# Read n and call fib with result
   li $v0, 5 
   syscall
   move $a0, $v0 
   jal fib
   move $s1, $v0 
# Print result
   li $v0, 4 
   syscall
   li $v0, 1 
   syscall
   move $a0, $s1 
# Call system exit
   li $v0, 10 
system
#
# Algorithm for Fib(n):
# if \( n == 0 \) return 0
# else if \( n == 1 \) return 1
# else return \( fib(n-1) + fib(n-2) \).
#
# Register usage in fib:
# $a0 = n (argument)
# $s1 = storage for fib(n-1)
#
# Stack usage:
# 1. push n, return address, $s1, before calling fib(n-1)
# 2. Restore n from stack (but don't pop it)
#    after call to fib(n-1) but before calling fib(n-2)
# 3. pop and restore $s1, return address, and n
#    before returning to calling routine
#
# fib:
#
# Okay, you're on your own.

Be sure to use good programming style, including comments describing your overall approach and steps along the way. At the beginning of your program, also include a comment stating whether or not your program works. Turn in a printed copy of your program and upload an electronic copy to Google Drive and share it with Ben (bwood5).