

**Homework 4**  
**(Floating point representation, Performance, Recursion and Stacks)**  
**Maximum points: 80 points**

**Directions**

This assignment is due Friday, Feb. 11<sup>th</sup>. Submit your solutions on a separate sheet of paper.

**Learning Objectives**

In the process of completing this homework assignment, students will develop their abilities to:

- Determine the performance of a processor.
- Determine how different parameters affect the performance of a machine
- Apply Amdahl's law to determine how much speedup can be obtained by improving an architecture.
- Interpret and represent floating point numbers
- Write recursive procedures.

**Problems**

- 1) In this problem, you will analyze the performance of two implementations of the MULDER instruction set architecture.
  - a. [5 points] For the FOX implementation, the clock rate is 1 GHz, and instructions fall into two categories. Category X instructions require 4 cycles to execute, while category Y instructions require 5 cycles. If the File program executes  $1.27 \times 10^{11}$  category X instructions and  $1.01 \times 10^{11}$  category Y instructions, how long will it take to execute? Express your answer in seconds.
  - b. [5 points] The SPOOKY implementation is identical to the FOX implementation, except that the clock rate is 1.2 GHz and category Y instructions require 6 cycles. How long will the File program take to execute on this implementation?
  - c. [5 points] The AGENT implementation is identical to the FOX implementation, except that it includes a category Z instruction that requires 8 cycles to execute and does the same thing as a common combination consisting of one category X instruction and one category Y instruction. This combination accounts for 10% of the category X instructions used by the File program on the FOX implementation. How long will the File program take to execute on the AGENT implementation?
  - d. [5 points] Between the FOX and SPOOKY implementations, which is faster for the File program, and by how much?
- 2) [8 points] Suppose we have two implementations of the same instruction set architecture. Computer A has a clock cycle time of 4 ns and a CPI of 2.0 for some program, and computer B has a clock cycle time of 2 ns and a CPI of 1.2 for the same program. Which computer is faster for this program?

3) [9 points] Suppose we could improve the speed of the CPU in a computer by a factor of 5 (without affecting I/O performance) for 5 times the cost. Also assume that the CPU is used 50% of the time, and the rest of the time the CPU is waiting for I/O. If the CPU is one-third of the total cost of the computer, is increasing the CPU speed by a factor of 5 a good investment from a cost/performance viewpoint?

4) [8 points] The incomplete MIPS procedure below computes the recursive function

$$foo(a,b) = \begin{cases} b, & \text{if } a = 0 \\ foo(a-1,b-a) * b, & \text{otherwise} \end{cases}$$

Finish the procedure by adding the code for the procedure entrance, the recursive procedure call, and the procedure exit. Do not modify any of the given code, and be sure to follow the MIPS convention for register usage.

Hint: \$t0 and \$s0 hold the local values of *a* and *b*, respectively.

You may write the answer to this question in the boxes provided and attach the sheets with the rest of your solutions.

foo:# Procedure entrance

```
# if (a == 0) return b
bne $t0, $zero, Else
move $t1, $s0
j Exit
```

```
# otherwise, return foo(a-1,b-a)*b
Else: sub $t2, $t0, 1
sub $t3, $s0, $t0
# Recursive procedure call
```

```
mul $t1, $t1, $s0
```

```
Exit: move $v0, $t1
#Procedure exit
```

- 5) [10 pts] Some computer architectures do not provide instructions for multiplication. The following Java code describes one (not particularly efficient) algorithm that could be used to perform multiplication within such an instruction set:

```
public static int myMultiply( int a, int b ) {  
  
    int rv;  
  
    if ( a == 0 ) {  
        rv = 0;  
    } else if ( a > 0 ) {  
        rv = myMultiply( a - 1, b ) + b;  
    } else {  
        rv = - myMultiply( -a, b );  
    }  
  
    return rv;  
}
```

Write a MIPS program that implements this algorithm.

- 6) [10 points] Convert the following decimal numbers to single precision IEEE 754 floating point format:  
(a) -21.625  
(b) 9.25
- 7) [15 points] The following bit patterns are floating point numbers in single precision IEEE 754 format. Convert them to decimal:  
(a) 1 1001 1000 011011000000000000000000  
(b) 0 0111 1011 001100000000000000000000  
(c) 0 0000 0000 011100000000000000000000