Universidad de Puerto Rico
Recinto Universitario de Mayagüez

ICOM 4036 – Programming Languages
Otoño 2004

Ejercicios de práctica
Examen Parcial II

1. Consider the following MIPS implementation of athe div recursive function discussed in class:

```
d:     sub  $sp, $sp, 28     # Alloc space for 28 byte stack frame
       sw   $a0, 24($sp)     # Save argument registers
       sw   $a1, 20($sp)     # a in $a0
       sw   $ra, 16($sp)     # Save other registers as needed
       sw   $s1, 12($sp)     # Save callee saved registers ($sx)
       sw   $s2, 8($sp)
       sw   $s3, 4($sp)      # No need to save $s4, since not used
       li   $s3, 0
       sw   $s3, 0($sp)      # int res = 0;
                             # Allocate registers for locals
       lw   $s1, 24($sp)     #   a in $s1
       lw   $s2, 20($sp)     #   b in $s2
       lw   $s3, 0($sp)      #   res in $s3

if:    bgt  $s2, $s1, else   # if (a >= b) {
       sub  $a0, $s1, $s2    #
       move $a1, $s2
       jal  d                #
       addi $s3, $v0, 1      #   res = div(a-b, b) + 1;
       j    endif            # }
else:  li   $s3, 0           # else { res = 0; }
endif:
       sw   $s1, 32($sp)     #   a in $s1
       sw   $s2, 28($sp)     #   b in $s2
       sw   $s3, 0($sp)      #   res in $s3
       move $v0, $s3         # return res

       lw   $a0, 24($sp)     # Restore saved registers
       lw   $a1, 20($sp)     # a in $a0
       lw   $ra, 16($sp)     # Save other registers as needed
       lw   $s1, 12($sp)     # Save callee saved registers ($sx)
       lw   $s2, 8($sp)
       lw   $s3, 4($sp)      # No need to save $s4, since not used
       addu $sp, $sp, 28     # pop stack frame
enddiv:    jr   $ra          # return;
```

Answer the following questions assuming that a *main* function calls *div*(12, 6):

a. How many times does div call itself recursively?

b. What are the arguments to div on each of these calls?

c. Show the contents of registers $s1, $s2, $S3, $a0, $a1, $v0, and $sp right after the jal instruction returns from each recursive call.

d. Draw the contents of the call stack right before div returns from its last recursive call. Make sure that you indicate the contents of each word-sized cell in the stack. You do not need to provide numeric values for the return addresses, just indicate the function that the return address belongs to. Also assume that the value of the stack pointer just before the first call div(12,6) was some constant X.

2. **Write a recursive version in MIPS assembly language for the following C++ functions:**

   a. int fact(int n) { return ((n<=0)?1:(n * fact(n – 1))); }

   b. int gcd(int a, int b) { return ((a%b==0)? b: gcd(a%b,b)); }

3. **Write FORTRAN subroutines or functions to compute the following:**

   a. Reverse the elements of an array

   b. The sum of two n x n matrices

   c. The product of two n x n matrices

   d. Sort and array using bubble sort

   e. The solution to a system of n independent linear equations in n unknowns (HINT: use some ideas from PA#2)