Assignment 2 (graded) – individual - solutions

Problem

Assume the following C code:

```
int sum(int n) {
    if (n != 0)
        // sum() function calls itself
        return n + sum(n-1);
    else
        return n;
}
```

a. (2 marks) Convert the C code into MIPS assembly assuming:

1. argument *n* is in $a0
2. result is in $v0

Note: *sum* is a **non-leaf recursive** procedure. Hence, registers have to be saved in the stack.

```
sum:
        addi $sp, $sp,-8
        sw $ra, 4($sp)
        sw $a0, 0($sp)
        bne $a0, $zero, L1
        add $v0, $a0, $zero
        addi $sp,$sp,8
        jr $ra
L1:
        addi $a0,$a0,-1
        jal sum
        lw $a0, 4($sp)
        lw $ra, 0($sp)
        addi $sp, $sp, 8
        add $v0, $v0, $a0
        jr $ra
```

b. (1 mark) Assuming n=5, how many write accesses in the stack will be needed to execute the procedure?

2 items are pushed in the stack per call. Since the procedure is called 6 times (from 5 to 0 included), we have 12 write accesses in total.

c. (2 marks) We now write another procedure, named *sum2*, which realizes the same functionality like *sum*. However, sum2 is a **leaf** procedure, i.e. no recursive calls can be made.
Write the MIPS assembly code for sum2, assuming:

- argument n is in $a0
- result is in $v0.

```
sum2:
        beq $a0, $zero, Exit
        add $v0, $v0, $a0
        addi $a0, $a0, -1
        j sum2
Exit:   jr $ra
```

d. (1 mark) Compare and discuss procedures *sum* and *sum2*.

The number of static and dynamic instructions is smaller for sum2. For this application, using a loop statement is thus a better option.