Lab session Recursion

Group A: November 27, 2009
Group B: November 24, 2009

Work in the given groups of two. Submit your solutions to the respective assignment on Blackboard. The file name is:
`s06_s0XXXXX_s0XXXXX.tar.gz`
One of the group members commits your solution. Keep an eye on the deadline (see Blackboard)!

1 Project

Consider the following C program implementing a factorial:

```
int fact(int n) {
    if (n < 1)
        return (1);
    else
        return (n * fact(n - 1));
}
```

Inspect the factorial assembler code (you can download on the website) - an example of recursion. Note the use of the stack pointer $sp, frame pointer $fp, return address register $ra, argument register $a0 and return register $v0. Execute it and discover how recursion works. Look at the stack during execution.

In essence, `fact(4)` is executed as the following recurrence relation:

```
fact(4)    = 4 * fact(3)
           = 4 * (3 * fact(2))
           = 4 * (3 * (2 * fact(1)))
           = 4 * (3 * (2 * (1 * fact(0))))
           = 4 * (3 * (2 * (1 * 1)))
           = 4 * (3 * (2 * 1))
           = 4 * (3 * 2)
           = 4 * 6
```

```
        = 24
```

Your project consists of implementing the following C code:

```
int f1(int a) {
    if (a < 3)
        return (a);
    else
        return (f2(a-1, a-2) + f2(a-2, a-1));
}
int f2(int x, int y) {
    return 2*f1(x) + y;
}
```

f1 is a indirect recursive function. The function behaves as follows:

i	1	2	3	4	5	6	7	8	9	10
f1(i)	1	2	9	27	79	221	611	1677	4591	12553

1. Compose the recurrence relation for $i = 4$, and write this in your report of this project. Keep in mind what part will be first resolved/calculated (this is stricter than simply applying mathematical properties)! That way, you know how your stack will behave.

2. Implement this code in MIPS assembly language. You're obligated to use the stack pointer, the frame pointer, the return address register, the argument registers and the return registers. Implement the two functions as two seperate subroutines (so do not substitute expressions).