Computer Systems and -architecture

MIPS: Recursion

1 Ba INF 2015-2016

Bart Meyers
bart.meyers@uantwerpen.be

Stephen Pauwels
stephen.pauwels@uantwerpen.be

Time Schedule

Exercises are made individually. Put all your files in a tgz archive, as explained on the course's website, and submit your solution to the exercises on Blackboard.

- Deadline: **December 20, 23u55**

Exercises

Write a MIPS program for the MARS simulator for each of the following exercises. As always, document your solution well (use #).
Use stack frames in all your procedure calls. Make sure you are using the stack in a correct way.

1. Write a MIPS program that reads an integer n (using a syscall), and calculates the fibonacci numbers from 1 to n. Use a recursive procedure! The fibonacci numbers are defined as follows:
   $F_0 = 0$
   $F_1 = 1$
   $F_i = F_{i-2} + F_{i-1}$ for $i > 1$

2. Consider the previous excercise. **Draw and explain on a sheet of paper** what the stack looks like when reaching one of the base cases for the first time after calling this with $n = 6$. i.e. We have the following chain of calls: $F(6) \rightarrow F(4) \rightarrow F(2) \rightarrow F(0)$. You may send in a scan of your solution.

3. Write a MIPS program that reads two integers a and b, and calculates the greatest common divisor.

   - Write a (leaf) `remainder` procedure that takes two arguments a and b, and calculates the remainder of the division of a and b.
   - Write a (recursive) procedure `gcd` with two arguments a and b, which calculates the greatest common divisor using this recursive definition:

$$gcd(x, y) = \begin{cases} x & : \text{ if } y = 0 \\ gcd(y, remainder(x, y)) & : \ x \geq y \text{ and } y > 0 \end{cases} \quad (1)$$

4. Take your exercises of last week and add a recursive procedure that sorts an array of integers using a mergesort algorithm. The algorithm you have to implement is given below. Call the procedure with the array on the heap (i.e. only pass the address as an argument) and provide the length of the array as the second argument. The use of helper functions is allowed and is recommended in order to avoid code duplication. Make sure you handle your stack frames properly in order to avoid problems with the different function calls.

```
void mergeSort(int[] data, int size) {
    if (size <= 1)
        return

    int size_a = size / 2;
    int[size_a] a;
    int size_b = size - size_a;
    int[size_b] b;
    for (int i = 0; i < size; i++) {
        if (i < size_a)
            a[i] = data[i];
        else
            b[i - size_a] = data[i];
    }

    mergeSort(a)
    mergeSort(b)

    int ai = 0;
    int bi = 0;

    while(ai + bi < size) {
        if (bi >= size_b or (ai < size_a and a[ai] < b[bi])) {
            data[ai + bi] = a[ai];
            ai++;
        } else {
            data[ai + bi] = b[bi];
            bi++;
        }
    }
}
```