Computer Systems and -architecture

MIPS: Recursion

1 Ba INF 2012-2013

Bart Meyers
bart.meyers@ua.ac.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 15, 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.

1. Suppose you have a MIPS program implementing a recursive algorithm to calculate the nth Fibonacci number. **Draw 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 $n = 5$. i.e. We have the following chain of calls: $F(5) \rightarrow F(3) \rightarrow F(1)$. The fibonacci numbers are recursively defined as follows:
   $F_0 = 0$
   $F_1 = 1$
   $F_i = F_{i-2} + F_{i-1}$ for $i > 1$
   You may send in a scan of your solution.

2. 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:

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

3. Take your exercises of last week and add a recursive procedure that sorts an array of integers using a quicksort algorithm. Call the procedure with the array on the heap, and left = 0, right = array size.

```
void quickSort(int arr[], int left, int right) {
      int i = left, j = right;
      int tmp;
      int pivot = arr[(left + right) / 2];

      /* partition */
      while (i <= j) {
            while (arr[i] < pivot)
                  i++;
            while (arr[j] > pivot)
                  j--;
            if (i <= j) {
                  tmp = arr[i];
                  arr[i] = arr[j];
                  arr[j] = tmp;
                  i++;
                  j--;
            }
      };

      /* recursion */
      if (left < j)
            quickSort(arr, left, j);
      if (i < right)
            quickSort(arr, i, right);
}
```