CS&A: Lab Sessions

Exercises: Stacks and Subroutines

Ruben Van den Bossche

1BA INF - 2010-2011

1 Time Schedule

Exercises are made individually. Fill in all solutions to the exercises in the file `oefeningen.html`. **Include all .asm files that contain your MIPS programs.**

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, 13 2010, 23u55**

2 Exercises

Write a MIPS program for the MARS simulator for each of the following exercises. As always, document your solution well (use #).

1. Write a MIPS program that pushes and pops integers on the stack. Execute this sequence of push and pop operations:

```
push 7
push 15
pop
push 31
pop
push 63
pop
pop
```

2. Write a MIPS program that reads a number of names from the **command line**, and prints "Hello *name*!" for each of these names. Enable command line arguments in MARS in the "Settings" menu, and provide a variable number of names as command line arguments in the MARS Program Argument box (above the text segment).

3. (a) Write a MIPS program that reads an integer n (using a syscall), after which it reads n integers (using syscalls), and stores them in an array. Because you don't know the size of the array in advance, you will have to allocate space for it on the heap (*Hint: use syscall 9 for `sbrk`*).

   (b) Add a (leaf) procedure that prints an array. The procedure has two parameters: the address of the first element of the array and the number of elements in the array. Call the procedure with the array on the heap.

   (c) Add a (leaf) procedure that sorts an array. Call the procedure with the array on the heap. Implement the algorithm from the Oberon-procedure below.

```
PROCEDURE Sort*(a : ARRAY OF INTEGER);
    VAR
        nrOfSwaps, temp : INTEGER;
        i : LONGINT;
BEGIN
    REPEAT
        nrOfSwaps := 0;
        FOR i := 0 TO LEN(a) - 2 DO
            IF( a[i] > a[i + 1] ) THEN
                temp := a[i];
                a[i] := a[i+1];
                a[i+1] := temp;
                INC(nrOfSwaps);
            END;
        END;
    UNTIL nrOfSwaps = 0;
END Sort;
```

4. Write a MIPS program that prints out the English word for an entered number. If the user enters “2”, the output is “two”. The program gives output for each number from 0 to 4.

   (a) Use if-then-else operations to implement this.

   (b) Use a case table with indirect addressing to implement this. *Hint: use `la` and `jr`.*

   Make a comparison of these two solutions in terms of (a) number of instructions used (size of the program) and (b) number of instructions executed (performance of the program). Compare the worst case and best case scenarios (with respect to the input value) of the two implementations. Give your conclusion.