Computer Systems and -architecture

MIPS: Stacks and subroutines

1 Ba INF 2016-2017

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 15, 23u55**

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 12
push 66
pop
push 48
pop
push 97
pop
pop
```

2. One of your friends comes to you with the code found in the file *mips2.asm*. The purpose of the program is to print out the result of 2+3+1 and, on a new line, the result of 2+3. The values 2 and 3 are stored in s-registers. The code uses a function call to *sum_add_one* in order to calculate the first sum. The second sum is performed by using *add* without a function call. The only problem is that your friend does not get the results he wants and cannot find the problem, due to the lack of comments added to his own code. Describe what went wrong in this code (add comments to the file) and fix the error by using a stackframe. And add some comments to the code in order to make it easier to read.

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 subroutine that prints an array. The subroutine has two parameters: the address of the first element of the array and the number of elements in the array. Call the subroutine with the array on the heap. Use a stack frame (or activation record) in your implementation!

   (c) Add a subroutine that sorts an array. Call the subroutine with the array on the heap. Implement the algorithm from the C++ function below. Use a stack frame (or activation record) in your implementation!

```
void sort(int array[], int arrayLength)
{
   int nrOfSwaps;
   do {
      nrOfSwaps = 0;
      for(int i = 0; i < arrayLength - 1; i++) {
         if( a[i] > a[i + 1] ) {
            int temp = a[i];
            a[i] = a[i + 1];
            a[i + 1] = temp;
            nrOfSwaps++;
         }
      }
   }
   while(nrOfSwaps > 0);
}
```