Computation Structures: *Introsort* in β-assembly

November 16, 2022

General information

- **Deadline: December 16, 2022, 23:59** (There will be a penalty for late submission).
- Project must be done **individually**.
- Questions will no longer be answered 24 hours before the deadline.
- Contact: `gaulthier.gain@uliege.be`, Office 1.8 (B37).

1 Introduction

The goal of this project is to get more acquainted with the β-assembly language by getting your hands dirty and writing code. In this project, you will implement an efficient sorting algorithm called *Introsort* in β-assembly.

1.1 Introsort

Introsort is an efficient sorting algorithm published by David R. Musser in 1997[1]. This variant of Quicksort improves the latter worst-time complexity from $\Theta(n^2)$ down to $\Theta(n \log n)$ using several tricks. The most important one consists in switching from *Quicksort* to *Heapsort* when the number of recursive calls exceeds $\log_2 n^2$ where n is the size of the array. It also uses a median-of-3 pivot selection strategy. It is not necessary to fully understand the algorithm to make the project, although it is advised to get at least a basic understanding. In order to do so, you can refer to the original article or other resources online.

2 Project & Implementation

The implementation of *Introsort* is split into several functions and files that you will have to implement. You are provided with `introsort.c`, a full C implementation of the algorithm that you can use as basis for your own. You are also provided with a file `util.asm` containing definitions of useful macros (*i.e.* `ADDR`, ...) and procedures (*i.e.* `log2`) that you can use in your implementation. Regarding procedure calls, you **must** use the "*last-argument-pushed-first*" (LAPF) convention throughout the whole project.

[1] Musser, D. R. (1997). Introspective sorting and selection algorithms. Software: Practice and Experience, 27(8), 983-993.

First, you have to implement the *Heapsort* algorithm in a file called heapsort.asm. Especially, you are required to implement a procedure labeled `heapsort`. Similarly as in the C implementation, the procedure should have two arguments: `array`, a pointer to the array to sort, and `size`, its size[2]. Any function needed for implementing `heapsort` but missing from util.asm should also be in this file (*e.g.* `heapify`).

Then, you have to implement the *Introsort* algorithm in a file called introsort.asm. You are required to implement two functions: `sort` and `introsort`. They both have `array` and `size` as first two parameters. `introsort` has an additional parameter `maxd` which indicates the maximum number of recursive calls before switching to *Heapsort*. Again, every other procedure needed for implementing those two and missing from util.asm would have to be implemented in introsort.asm.

Finally, you are provided with a main.asm file that you can use to test your project. It creates an array of a given size (that you can change) with random values and then calls `sort` to sort it. As you will see, all assembly file inclusions (*i.e.* using the `.include` statement) are performed in the main file and you should not include any file in introsort.asm or heapsort.asm.

You are advised to comment extensively your assembly code. In particular, document all procedures and macros you define (their arguments, what they do and their return value if any). It is good practice to document what your code is doing on a per-block basis (usually, a line of C translates into several instructions) and to state which registers are assigned to which variables.

3 Additional guidelines

In order to learn β-assembly effectively, **this assignment will be done individually**. Plagiarism is of course not allowed and will be severely punished. Asking questions on the Forum (on eCampus), and helping your fellow students on the Forum is allowed (to a reasonable extent).

You will include your completed introsort.asm and heapsort.asm files in a ZIP named sXXXXXX.zip where sXXXXXX is your student ID. Naming your files differently or submitting other files **will result in a penalty**.

Submit your archive to the **Montefiore Submission Platform**[3] (course INFO0012-2). If you encounter any problem with the platform, contact the assistant (gaulthier.gain@uliege.be) as soon as possible.

[2] Following LAPF, the size is pushed on the stack first when the procedure is invoked.

[3] http://submit.montefiore.ulg.ac.be/