# Informatics 2C — Computer Systems
# Practical 1
# Deadline: 28th October 2009, 4:00 PM

Stratis Viglas

## 1 Description

This practical is based on material in the Computer Systems thread of the course. Its aim is to increase your familiarity with programming a MIPS processor. The practical asks you to write and submit three assembly programs for the MIPS processor, as implemented by the SPIM simulator. For details on SPIM, see the first lab script available at:

`http://www.inf.ed.ac.uk/teaching/courses/inf2c/labs/inf2c_cs_lab1.html`

This is the first of two practicals for the Computer Systems part of Inf2C. It is worth 25% of the coursework mark of the whole Inf2C. Finally you should bear in mind the guidelines on plagiarism, which can be found via a link on the Informatics 2 course guide.

### 1.1 Calculator Implementation

For the first part of the practical you are required to write an assembly program that will implement a simple calculator. The calculator will prompt the user for two operands (integers) and an operator and then output the result of the computation to the terminal. The calculator should support addition, subtraction, multiplication and division.

*Hint:* You will have to compare the operation character you read with the ASCII codes of the operation required. The ASCII equivalents are:

- Addition (+): `2B` (Hex)
- Subtraction (-): `2D` (Hex)
- Multiplication (*): `2A` (Hex)
- Division (/): `2F` (Hex)
- New Line Feed: `0A` (Hex)

Note that if you choose to use the read char system call to read the operator, you should not type any other characters, not even press the enter/return key! If you want to press Enter after the operator, you may have to use a second read char to read the new-line character (`0x0A`).

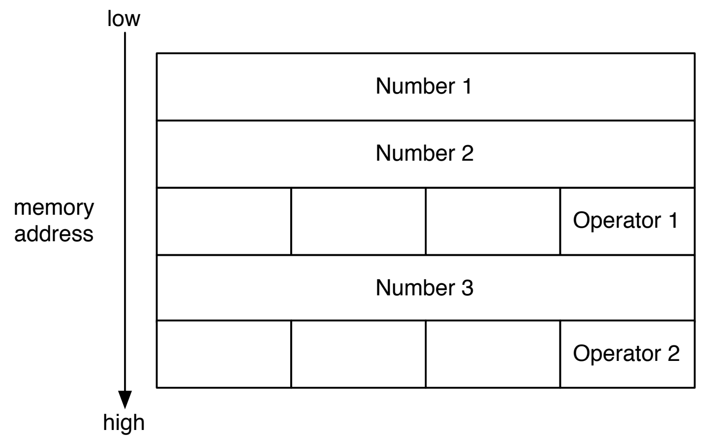

Figure 1: Postfix expression in memory

## 1.2 Calculating Expressions

For the second section of the practical, you will write a program that calculates the expression $3x^2 + 5x + 6$, taking $x$ as input, using the previously constructed calculator. Note that you will have to convert the first part of the practical to a function (method) and call it appropriately, using the MIPS convention for passing parameters and returning values, otherwise it will be considered incorrect and marks will be deducted.

## 1.3 Postfix Expressions

For the final part of the practical, you have to write a program that processes an arithmetic expression stored in memory and calculates the result using the function/subroutine used for the other parts of the practical. Don't forget to print the result too!

The expression will be stored in postfix (reverse polish) notation. In postfix notation, operators follow their operands. `(2+5)` would be expressed in postfix notation as `2 5 +`. As another example, `(8+4)/(5-2)` would become `8 4 + 5 2 - /`. Postfix notation is particularly useful since it innately expresses the order of operations and no brackets are required.

The memory format of the arithmetic expression is shown in Figure 1. It must be evaluated from top to bottom. Note that the memory addresses are increasing in that direction. Also note that although each operator is one byte (char) in size, the next number is *word aligned* in memory. In order for your program to finish, you must add a special value in the memory that will signal the end of the sequence. There also needs to be some way of telling an operator apart from an operand. We can do this by encoding operators as `0xAA00 00` (*ASCII code*) and the end of the sequence as `0xAA00 0000`. You can safely assume that operands in the range `0xAAFF FFFF` to `0xAA00 0000` will not be used.

The stack based algorithm for evaluating postfix expressions can be found at:

`http://en.wikipedia.org/wiki/Reverse_Polish_notation#The_postfix_algorithm`

## 2 Submission

Submit your completed program with the command:

```
submit inf2 inf2c cs1 part1.s part2.s part3.s
```

where the filenames above correspond to the assembly programs for the three parts of the assignment. Normally you will not be allowed to submit coursework late. If you have any special circumstances contact the Inf2 course organiser (Jacques Fleuriot) or the course lecturer (Stratis Viglas) as soon as you can.

## 3 Assessment

We will use SPIM to run your programs, so try to ensure that they load and run on the simulator. Your programs will be primarily judged according to correctness, completeness, performance, and code size. In assembly programming commenting a program and keeping it tidy is very important. Make sure that you comment the code throughout and format it neatly. A proportion of the marks will be allocated to these.

## 4 Questions

If you have questions about this assignment, ask for help from your demonstrators and the newsgroup `eduni.inf.course.inf2c`. If your query is confidential (*e.g.*, it would reveal much of the answer) contact the Inf2 Computer Systems lecturer directly, Stratis Viglas (`sviglas@inf.ed.ac.uk`).