Tutorial Sheet 10 – Sigma16 Introduction

Setup
Pen and paper + Computer exercises

Instruction Sets, and Sigma16 (Pen-paper exercises)

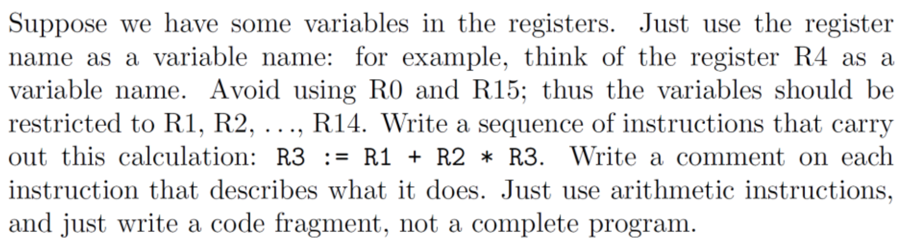
This lab exercise is an introduction to computer architecture, using Sigma16. This material is essential for the rest of the unit. The aims are:

- to solidify your understanding of the basic instructions: *add/sub/mul/div* for doing arithmetic, *load/store* for copying variables between memory and registers, and lea for putting a constant into a register.
- to learn how to write a complete program that terminates properly and that defines its variables;
- to see how to assemble a program and run it on the emulator.

First solve the paper problems with paper and pencil (or here in the word version of this document). It is important to understand what you're doing before starting to type code into a computer. Later, as we get into more advanced programming, we will take the same approach: start by working on paper and really thinking about what you are doing, and only then typing it into the computer.

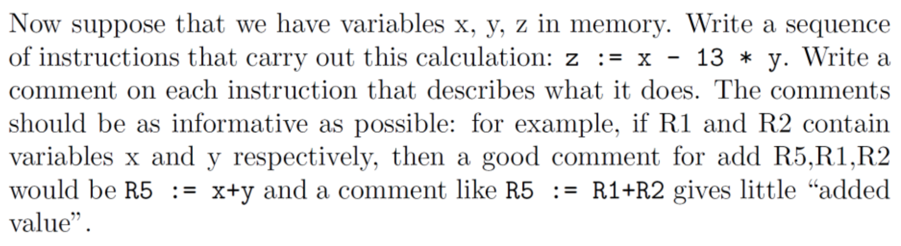
The exercise is not assessed; there is nothing to hand in, and you are encouraged to work with your groups.

3.



Suppose we have some variables in the registers. Just use the register name as a variable name: for example, think of the register R4 as a variable name. Avoid using R0 and R15; thus the variables should be restricted to R1, R2, . . ., R14. Write a sequence of instructions that carry out this calculation: `R3 := R1 + R2 * R3`. Write a comment on each instruction that describes what it does. Just use arithmetic instructions, and just write a code fragment, not a complete program.

4.



Now suppose that we have variables x, y, z in memory. Write a sequence of instructions that carry out this calculation: `z := x - 13 * y`. Write a comment on each instruction that describes what it does. The comments should be as informative as possible: for example, if R1 and R2 contain variables x and y respectively, then a good comment for add R5,R1,R2 would be `R5 := x+y` and a comment like `R5 := R1+R2` gives little "added value".

5.

Hand-execute the following program fragment. After each instruction, show what register has changed, and its new value. Add a comment to each instruction that describes what it does.

```
lea    R2,23[R0]
lea    R3,4[R0]
lea    R5,30[R0]
add    R3,R5,R3
add    R4,R2,R0
sub    R6,R4,R5
```

6.

Explain the difference between the `load` and `store` instructions. Write the instructions needed to perform

```
R4 := x
total := R5
```

7.

Explain the difference between the `load` and `lea` instructions. Write the instructions needed to perform

```
R1 := 27
R2 := x
```

8.

Sometimes in programming you need to swap the values of two variables. For example, suppose that x = 29 and y = 67. After swapping the variables, the values are x = 67 and y = 29.

(a) Explain why you can't just write x := y; y := x.

(b) Write assignment statements that swap the two variables in a high level language.

(c) Write assembly language instructions that swap the two variables. Hint: you'll need to use registers and some load and store instructions; there is no way to do this without passing the data through registers.

(d) Discuss the two versions of swap that you wrote (the high level language version and the assembly language version).

(e) Write a complete assembly language program that defines two variables, x (with initial value 3) and y (with initial value 19). The program should swap the variables and then halt. The first line of the program should be a comment that gives the name of the program: `; Program Swap`.

Instruction Sets, and Sigma16 (Computer-based exercises)

We will be using Sigma16 for the rest of the course. This lab gives a brief introduction. Follow the instructions on Moodle to find the Sigma16 application and launch it (links at the bottom of this tutorial too). The User Guide contains a sequence of tutorials; in this exercise you'll work through a few of them.

9.

Read the first tutorial, *Hello, world!*, and follow its instructions. This gives you an overview of the main components of the app.

10.

Study the second tutorial, *Registers, constants, and arithmetic*, again following its instructions. This tutorial covers the arithmetic instructions and the lea instruction, and shows how to do arithmetic using the registers. The programs here use lea to put constants into the registers, and they use the arithmetic instructions to perform calculations. For now, all variables are kept in the register file.

11.

Finally, work through the third tutorial, *Keeping variables in memory*. This introduces the memory and shows how to use the load and store instructions to copy values between the register file and the memory.

As you work through the tutorials, be sure that you understand what each of the instructions does. Hand execute the program: write down the effect of each instruction. It's a good idea to work through the programs very slowly and carefully (they are short!). First think about what the next instruction will do and predict what you think will happen; then click Step to execute the instruction and check your prediction. Now go back to the Editor pane, clear it, and type in Program Swap (which you wrote above). Assemble it, boot it, and step through it in the Processor pane.

Links to Sigma16 resources

- Sigma 16 home page
- Sigma 16 online emulator
- Sigma 16 user guide