FIT1047 Tutorial 3

Topics

- Programming MARIE assembly code
- Circuits for adding and subtracting

Instructions

The tasks are supposed to be done in groups of two or three students. You will need two different *simulators* for this assignment (both are linked from Moodle):

- The *MARIE* simulator lets you write and execute programs for the *MARIE* architecture.

- The *Logisim* tool allows you to draw digital circuits and simulate their behaviour.

Task 1: MARIE basics

Try out the online MARIE simulator with a simple program like the one from the lecture:

```
Load  004    / Load value from address 004 into AC
Add   004    / Add value from address 004 to value in AC
Store 004    / Store AC into address 004
Halt         / Stop execution
DEC 42       / This is address 004, initialise with value 42
```

- Enter the program above and step through it.

- Rewrite the program to use a label instead of hard-coding the address 004.

- Save the code to a file. *Hint:* don't use the web browser's *Save file* dialog, but use the File -> Download option from the MARIE.js menu. Upload the file again. You will need this feature for the assignment!

Task 2: Multiplication

MARIE only has arithmetic instructions for adding and subtracting numbers, but not for multiplication. Your task is to implement a multiplication algorithm. Assume that we want to compute $A \times B$, with A and B stored in some memory location. The result should be stored in memory location C. You can use the following pseudo-code as a starting point.

```
01   C := 0
02   if A = 0 then jump to line 06
03   C := C + B
04   A := A - 1
05   jump to line 02
06   halt
```

- Discuss the pseudo-code for multiplication. It assumes that variables (or storage locations) A and B hold the two integers that should be multiplied, and the result should be stored in C. Which MARIE instructions can you use for each construct in the code (such as the := operator, the jump to line instructions, the if ... then)?

- Implement the algorithm using the MARIE simulator. Test your code with several different inputs.

- What happens if A is negative before the algorithm starts? What happens if B is negative? **Bonus task:** Fix the algorithm so that it also works in these cases. First write your solution in pseudo-code, then implement it using MARIE.

Task 3: Adder/Subtractor

You have seen half-adders, full-adders and ripple-carry adders in this week's lecture. You will now use this knowledge to build a 4-bit adder/subtractor in Logisim.

- Your first task is to draw and simulate a simple *half-adder* and *full-adder*. Discuss the truth tables for these two circuits and how they correspond to the circuit diagrams you saw in the lecture. Then implement both in Logisim and try them out.

- Now combine four full adders into a 4-bit ripple carry adder, using copy-and-paste. It should have eight inputs (two 4-bit numbers A and B) and five outputs (the 4-bit result and the final carry bit). The carry-in for the first full adder has to be fixed to 0 (use the *Ground* symbol in the *Wiring* folder).

  Use the simulator to enter different numbers and make sure the adder produces the correct result.

- Now let's assume that we are using the *2's complement* representation for negative numbers. Simulate a few computations, such as $5 + (-7)$ and $(-3) + (-4)$.

- Finally, modify the adder into a *subtractor* that computes $A - B$.

  Hint: We will implement subtraction as $A + (-B)$. Computing $-B$ in 2's complement means negating every bit and adding 1. So for the B input, we can simply negate all bits. Now the only thing that's required is to add another 1. Is there something in the existing ripple-carry adder that you can reuse to do that?