FIT1047 Tutorial 5

Topics

- Revision of weeks 1 to 4

Instructions

- The tasks are supposed to be done in groups. In some tasks, it might be useful to have different roles for people in the group.
- Please complete **Task 0** before the lab.

Task 0: Homework

0.a Make yourself familiar with Karnaugh maps. Karnaugh maps have been briefly introduced in the lecture and are part of the first assignment. Also, task 2 of this lab will look at Karnaugh maps. There are quite a number of introductory videos on Karnaugh maps on YouTube. One example is here: https://www.youtube.com/watch?v=b4JiIknHm1Y

0.b The concept of **subroutines** is quite important. Not only in MARIE, but also in all types of programming. This task is from last week. If you were not able to complete it last week please try it at home now (don't look at the sample solution just yet, try it by yourself first):

- Implement a simple subroutine that computes 2 × X, i.e., it takes the value in a memory location X, doubles it, and returns to where the original program left off.
- Implement a small program that uses the subroutine and ckeck the result in MARIE.
- Use `JnS` and `JumpI`.

Task 1: Data Representation

1.a Given the following two binary numbers: 11111100 and 01110000.

  (i) Which of the two numbers is the larger unsigned binary number?

  (ii) Which of these two numbers is larger when it is being interpreted on a computer using signed two's complement representation?

  (iii) Which of these two numbers is smaller when it is being interpreted on a computer using signed-magnitude representation?

1.b Add the following unsigned binary numbers:

$$\begin{array}{r} 01000100 \\ \underline{10111011} \end{array}$$

1.c Subtract the following signed binary numbers using two's complement arithmetic (Note: These numbers use 2's complement notation, meaning the first number is negative and the second is positive. Before you start to do any calculations, think about how subtraction works with two's complement arithmetic).

$$\begin{array}{r} 11000100 \\ \underline{-00111011} \end{array}$$

$$\begin{array}{r} 01011011 \\ \underline{-00011111} \end{array}$$

$$\begin{array}{r} 10101100 \\ \underline{-00100100} \end{array}$$

1.d Assume a 24-bit word in a computer. In these 24 bits, you should represent the string B95. If your computer uses 8-bit ASCII with **even** parity (i.e. from left to right 7 bit for the character and 1 parity bit), how would the computer represent the string B95? (Hint: Start with searching for the 7-bit ASCII table in the Internet to find the coding for the 3 characters).

Task 2: Boolean Algebra

Given the function $F(X,Y,Z) = Y(\overline{X}Z + \overline{Z}) + \overline{Y}(XZ + \overline{Z})$

2.a Complete the truth table for F.

X	Y	Z	$F(X,Y,Z)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

2.b Use logisim to draw the logic diagram using the original Boolean expression.

2.c Simplify the expression using a Karnaugh map.

2.d Use logisim to draw the logic diagram for the simplified expression in part 2.c.

Task3: CPUs and MARIE

3.a Explain why, in MARIE the MAR is only 12 bits wide and the AC and MBR are 16 bits wide (Hint: Consider the difference between memory cells and addresses).

3.b Write the following code segment in MARIE assembly language:

```
X = 1
while X < 10 do
  X = X + 1
endwhile
```

(Hint: The `while` - `endwhile` loop means that the line `X = X + 1` is repeated as long as X is smaller than 10. You should use skipcond to check if X equals ten. If not, you execute the addition, jump back to before the skipcond, etc.)

3.c The following table shows MARIE's datapath control signals:

	Memory	MAR	PC	MBR	AC	IN	OUT	IR
P_2,P_1,P_0 (Read)	000	001	010	011	100	101	110	111
P_5,P_4,P_3 (Write)	000	001	010	011	100	101	110	111

In addition to these signals there are special signals such as IncrPC to increment the PC and M_R and M_W for read from memory and write into memory.

Provide RTL and control signals for the 6 steps of the `JumpI` instruction:

1. Read the program counter and write into the memory address register (Fetch, Time t_0).

2. Read the instruction at this address and write into instruction register (Fetch, Time t_1).
3. Increment the program counter (Decode, Time t_2).
4. Read operand from instruction register and write into memory address register (Execute, Time t_3).
5. Read memory at this address and write into memory buffer register (Execute, Time t_4).
6. Read memory buffer register and write into program counter (Execute, Time t_5).