COMP1036 Coursework

This coursework is worth **50%** of the final mark.

Deadline: 13th December 2019 16:00

Part I (25 marks)

Previous exercises have used combinatorial logic and sequential logic to construct many of the various logic circuits that form the basis of a computer. In this exercise, we will put some of these circuits together to build a custom 16-bit ALU chip, with the capacity to save and re-use the output.

Part I requires you to build a custom ALU, this consists of constructing an ALU that allows the following features to be evaluated manually and with a test file:

Subtract y from x [2 marks]

Multiply x by y (eg. When x = 12, y = 4 then out = 48) **[note, both x and y can only be between -100 and +100]** [4 marks]

Divide x by 2 (eg. When x = 8 then out = 4) **[note, only for positive values of x and fractional results should be rounded down]** [3 marks]

Calculate an exponential expression (eg. When x =4 and y = 3 then out =64) **[note, x or y cannot be negative, maximum values of x= 9 and y = 5]**
[6 marks]

Signal, using a control pin, if a number is greater than 32767 (overflow) [2 marks]

Decide if x is bigger than y, output = 1 if true, -1 if false [4 marks]

The remaining 4 marks will be at the markers discretion and be for:

- instructions for use
- comments
- efficiency
- structure

The format of the ALU will as shown in figure 1

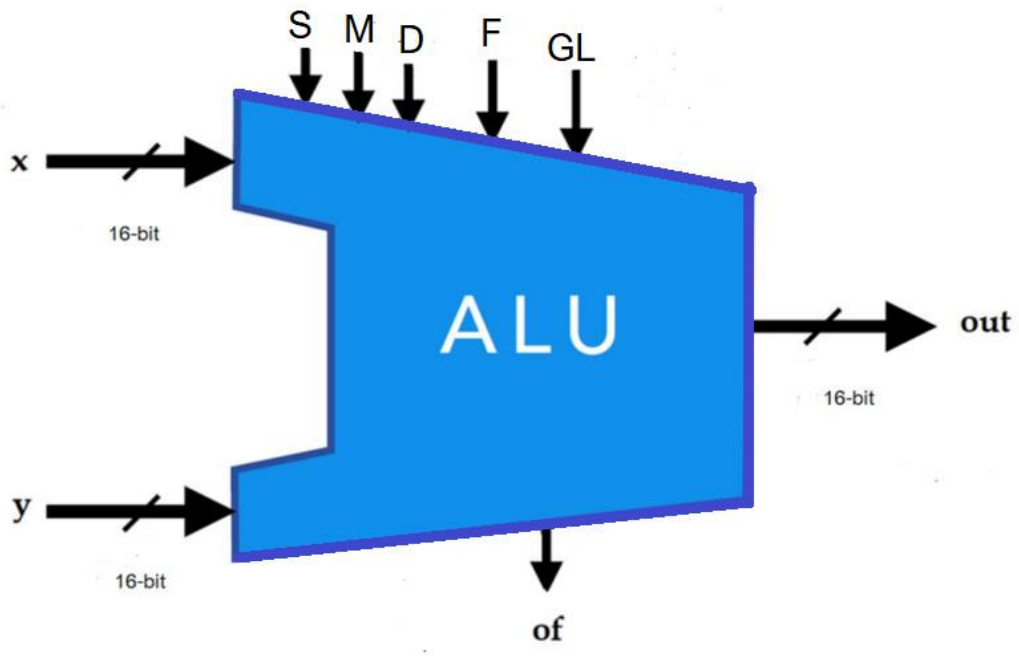

Figure 1. ALU schematic

The project template is supplied in the file CWALU.HDL

How to submit

You should zip all your files into one zip file to “Coursework - Part 1 Assignment”. You should name your file as: YOURSTUDENTID_YOURNAME_1.zip. Submit your zip file onto Moodle submission page. Please note that every next submission overwrites all the files in the previous one. If you submit several times, make sure that your last submission includes all the necessary files. Include all required chips, instructions for use, and any instruction text file.

For late submission, the standard late submission policy applies, i.e. 5% mark deduction for every 24 hours.

Deadline: 20th December 2019 16:00

Part II (25 marks)

Part II consists of the following tasks:

1. To implement a division function of integers z=x/y, where x, y, z are **integers**, and z is the **round-off** value of x/y. You should store x, y, z in RAM[0], RAM[1] and RAM[2], respectively. Name your assembly file as **Task1.asm**. (**12 marks in total.** 10 marks for 10 tests, 2 marks for documentation.)

   Hint: If x = 10, y = 3, you should get z = 3; if x = 10, y = 2, you should get z = 5; if x = 10, y = 10, you should get z = 1; if x = 4, y = 10, you should get z = 0; if x = 5, y = 10, you should get z = 1; if x = 6, y = 10, you should get z = 1; if x = -4, y = 10, you should get z = 0; if x = -5, y = 10, you should get z = 0; if x = -6, y = 10, you should get z = -1.

   You may determine the sign of z first, based on the signs of x and y. Then, you calculate the round-off value of |x|/|y|, where |x| is the absolute value of x, and |y| is the absolute value of y. Then based on the sign of z, you may have z = |x|/|y| or z = -|x|/|y|. Be careful of the difference when handling the round-off of positive number and the round of negative number.

   To calculate the **round-down** value of z, you may use the idea of repetitive addition, e.g. for positive x and y, y+y+y+..+y≤x, and determine the number of y in this inequality, which is z. (Note that this z value will be the **round-down** value, which is different from **round-off** value.) You should set RAM[3]=-1 if y = 0, otherwise RAM[3]=1.

2. To implement a power function of integers z=x^y, where x, y, z are **non-negative integers**. You should store x, y, z in RAM[0], RAM[1] and RAM[2], respectively. Name your assembly file as **Task2.asm**. (**7 marks in total.** 5 marks for 5 tests, 2 marks for documentation.)

Hint: If x = 2, y = 3, you should get z = 8; if x = 1, y = 2, you should get z = 1; if x = 0, y = 2, you should get z = 0. You may use the idea of repetitive multiplication, e.g. z = x*x*...*x, and y is the number of multiplications. You **do not** need to handle overflow for this task.

3. To implement a log function of integers z=log x, where x is a **positive integer**, and z = log x returns the logarithm of x, where the base is 2. z is **rounded down** to the nearest integer. You should store x and z in RAM[0] and RAM[2], respectively. Name your assembly file as **Task3.asm**. (**6 marks in total.** 4 marks for 4 tests, 2 marks for documentation.)

Hint: If x = 2, you should get z = log 2 = 1; if x = 5, you should get z = 2; if x = 15, you should get z = 3. You may use the idea of repetitive multiplication, e.g. for 2*2*...*2≤x, determine the maximum number of multiplications where the inequality holds, which gives you the value of z.

If you use code you found in a textbook or on the web, you must acknowledge it. I will run the plagiarism detector tools to check for similarities between submissions and web-based material.

You are reminded of the School's Policy on Plagiarism.

Notes

1. The sample test scripts and compare files have been uploaded into moodle page to "Coursework - Part 2 Assignment". You may use these sample scripts to evaluate the basic functionality of your program. Please note that the final test scripts for coursework marking may be different. Make sure that your program is bug-free.
2. Be reminded that marks are allocated to comments (documentation). You may briefly explain the functionality of the program at the beginning of your assembly file. Your program should be intensively commented, e.g. I expected at least 1 comment for 2-6 lines of code.

How to submit

You should zip all your files into one zip file. You should name your file as: YOURSTUDENTID_YOURNAME_2.zip. Remember to include your student ID and your name at the beginning of each asm file. Submit your zip file onto Moodle page. Please note that every next submission overwrites all the files in the previous one. If you submit several times, make sure that your last submission includes all the necessary files.

For late submission, the standard late submission policy applies, i.e. 5% mark deduction for every 24 hours.