COMP1036 Coursework

This two-part coursework is worth **48%** of the final mark, **2%** has already been awarded in the in class quiz.

Deadline: 18 December 2020 16:00

Part I (28 marks)

CSF Coursework 1 consists of the following tasks:

Task 1. [12 marks]
Input
Write a database using the hardware simulator that takes:
A 16-Bit version of an ID number called **inID**
An associated exam mark called **inMARK**
Load each value onto 1 of 8 designated Registers, assigned by a 3-Bit **address**

Input pins		Output pins	
Name	Value	Name	Value
inID[16]	12345	outID[16]	0
inMARK[16]	97	outMARK[16]	0
load	1	sum[16]	0
probe	0	overflow	0
address[3]	2		
sl	0		
priority	0		

Output
Allow the user to request an exam mark be output based on address. For instance, after entering the above values, setting probe to 1 and address to 2 like so:

Input pins

Name	Value
inID[16]	0
inMARK[16]	0
load	0
probe	1
address[3]	2
sl	0
priority	0

Produces the following output

Output pins

Name	Value
outID[16]	12345
outMARK[16]	97
sum[16]	0
overflow	0

Task 2. [7 marks]

Input

Take the functionality of Task 1 and add the ability for the data entry to happen sequentially. So if sequential load (sl) is turned on data will be entered onto the RAM in sequential order and will overwrite existing values once the RAM is full. In effect, this is creating Sequential Access Memory (SAM)

So if we first enter:

Input pins

Name	Value
inID[16]	12345
inMARK[16]	97
load	1
probe	0
address[3]	0
sl	1
priority	0

Output pins

Name	Value
outID[16]	0
outMARK[16]	0
sum[16]	0
overflow	0

And then:

Input pins

Name	Value
inID[16]	12346
inMARK[16]	79
load	1
probe	0
address[3]	0
sl	1
priority	0

Output pins

Name	Value
outID[16]	0
outMARK[16]	0
sum[16]	0
overflow	0

Output
Allow the user to request an exam mark to be outputted using the student ID. So if we enter:

Input pins

Name	Value
inID[16]	12346
inMARK[16]	0
load	0
probe	1
address[3]	0
sl	1
priority	0

We get the output:

Output pins

Name	Value
outID[16]	0
outMARK[16]	79
sum[16]	0
overflow	0

Task 3. [5 marks]

Input

Take the functionality of Task 1 and 2 and add the ability to use a 1 bit entry called '**priority**'.

Output

When priority is equal to 1, the chip will calculate the sum of all ID values stored in RAM and output the value to sum[16], if this value is greater than the maximum value for a 16 bit signed integer then the overflow value should equal 1

So, after the values for student ID 12345 and 12346 have been entered, if we enter:

Input pins

Name	Value
inID[16]	0
inMARK[16]	0
load	0
probe	1
address[3]	0
sl	0
priority	1

The desired output would be:

Output pins

Name	Value
outID[16]	0
outMARK[16]	0
sum[16]	24691
overflow	0

Task 4 [4 marks]
Write a readme file that supports the use of this software, explaining special cases. Write a test script and compare file that demonstrates the functionality. Write comments in your code to explain its behavior. You may be asked to discuss your code in detail in the lab

How to submit

You should zip all your files into one zip file. You should name your file as: YOURSTUDENTID_YOURNAME.zip. Your zip file should include:

- one master hdl file called CW.hdl file
- all additional hdl files used by CW.hdl
- one readme.txt file
- any supplied test scripts and compare files.

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.

Do check the submission after you submit it. It happens in the past that the submitted file was corrupted. You should ensure that the submitted file is complete and executable. You will receive 0 mark if your submitted file is corrupted or not executable.

For late submission, the standard late submission policy applies, i.e. 5% mark deduction for every 24 hours. (Including weekends and public holidays)

Part II (20 marks)

CSF Coursework 2 consists of the following tasks:

1. To implement a modulo function of integers z=x mod y, where x, y, z are **signed integers**, and z is the remainder 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**. (**7 marks in total.** 4 marks for 4 tests, 1 marks for documentation, 1 mark for good programming practice and 1 mark for program efficiency.)

   Hint: Use Octave function "z=mod(x,y)" to check what should be the desired output if you are not sure. You can find the online Octave here:

   https://octave-online.net/

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

For unsigned integers x and y, it is easy to obtain z. If x and y are signed integers, the sign of z is the same as the sign of y. For example, mod(10,-3) should take one of the values, -2 or 1, as |z|<|y|. We require that the sign of z is the same as the sign of y, thus we have mod(10,-3) = -2.

You may use the idea of repetitive subtraction to implement this function, e.g. z=|x|-|y|-|y|-|y|-...-|y| until |z|<|y|.

You do not need to handle the case where y = 0.

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.** 4 marks for 4 tests, 1 mark for documentation, 1 mark for good programming practice and 1 mark for program efficiency.)

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. Implement the following function: when any key (except "ESC") is pressed, print the last two digits of your student ID on the screen. When the key is released, immediately print a blank (white) screen. When "ESC" is pressed, jump to the end of the program. Name your assembly file as **Task3.asm**. (**6 marks in total.** 3 marks for testing, 1 mark for documentation, 1 mark for good programming practice and 1 mark for program efficiency.)

Hint: If your student ID is 01234567, then you should print "67" on the screen. There is no requirement on the size, the shape or the location of the digit, as long as the digit is clearly visible and can be clearly identified. You **may** use the following digits as an example when you try to display the digit.

Plagiarism

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 will be uploaded into moodle page. 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 **must** briefly explain the functionality of the program at the beginning of your assembly file. Your program **must** be intensively commented, e.g. I expected at least 1 comment for 2-6 lines of code.
3. Marks are allocated to good programming practice, including built-in symbols, variables, labels, pointers, naming conversions and so on.
4. Marks are allocated to efficiency of the program. I will use a reasonable number of tick-tocks to test your program. If your program could not generate the expected results within the given number of tick-tocks, you will lose one mark for efficiency for that task.

How to submit

You should zip all your files into one zip file. You should name your file as: YOURSTUDENTID_YOURNAME.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.

Do check the submission after you submit it. It happens in the past that the submitted file was corrupted. You should ensure that the submitted file is complete and executable. You will receive 0 mark if your submitted file is corrupted or not executable.

For late submission, the standard late submission policy applies, i.e. 5% mark deduction for every 24 hours. (Including weekends and public holidays)