You need to submit a zipped folder with the following naming convention: KCLRollNumber_CWresti.zip that contains:

- the compiled project, as well as all source files (.asm files, .vhd files and testbenches)
- A short report explaining the results, showing screenshots of waveforms that verify the correct execution of the AluOp conditions listed in the table for all corner cases, as well as analysis of the exceptions in your SPIM program: KCLRollNumber_CWresit.pdf

1. Use QtSpim to implement the code below for calculating a recursive sequence

$$a_n = a_{n-1} + 2b_{n-1}$$
$$b_n = a_{n-1} - b_{n-1}$$

with $a_0 = b_0 = 0$, $a_1 = 2, b_1 = 1$.

Your code should prompt the user to enter the value of n and display the results in the console. Note that you should use recursion with nested procedure linking to implement the code.

[40 marks]

Also determine the smallest positive value of n for which your code results in an error. Explain the cause of error.

[10 marks]

2. In this project, you will write VHDL code for an arithmetic logic unit (ALU) and verify that it works correctly using a testbench.

Your design need to support only the following subset of operations, assuming two 32-bit inputs A and B:

add, sub, and, or

AluOp	Mnemonic	Result	Description
0000	add	A + B	Addition
0010	sub	A - B	Subtraction
0100	and	A and B	Bitwise AND
0101	or	A or B	Bitwise OR
Others	-	Don't Care	-

The ALU generates a 32-bit output that we call 'res' and an additional 1-bit flag 'zero' that will be set to 'logic-1' if all the bits of 'res' are 0. ALU also has two more outputs, 'cout' and 'ofl', which are 1-bit signals to denote Carry Out and Overflow that may occur as a result of the operation.

For example, if 'AluOp' is 0101, ALU should evaluate Result as A or B. Many values of 'AluOp' does not correspond to any operation. It is not important what the circuit does when 'AluOp' has these values since the 'Result' will simply be ignored in such cases. You can use this to your advantage to simplify the circuit.

a. First, you need to draw a block diagram of the ALU. You may use arbitrary size adders, multiplexers, logic gates, zero/sign extend, comparators and shifters.

[10 marks]

QUESTION 2 CONTINUES ON NEXT PAGE

SEE NEXT PAGE

b. Once we have a good block diagram it is straightforward to implement the circuit in VHDL. Replace each block with a VHDL description and use the signal names in the block diagram.

[30 marks]

c. Test your VHDL code using the test bench provided and verify the functionality is implemented correctly.

[10 marks]