University of
South Australia

COMP 3026 Computer Science

Assignment 1 – Binary Calculator

UniSA STEM
The University of South Australia
2025

Overview

In this assignment, you are asked to write a console-based program (either in Python or Java) by yourself that calculates addition or subtraction of binary numbers with arbitrary number of digits. The program must first ask the user to input the type of operation (+ or -) then ask for two binary numbers with arbitrary number of bits as strings. The program must then convert each input string into an array (or list in Python) of numbers with each element representing a binary digit (0 or 1), then calculate the sum or difference, depending on the operation chosen earlier, and give an output of the results in a format specified in this document. The program should repeat the above until the user enters 'q' when asked to input an operation.

Concepts

In your program, binary numbers must be defined as an array (or list in Python) of numbers with each element of the array representing a binary digit (a.k.a. bit). For example, for a binary number `10101101` can be stored in an array `a[]` as the following (starting from the least significant bit):

`a[0]`	`a[1]`	`a[2]`	`a[3]`	`a[4]`	`a[5]`	`a[6]`	`a[7]`
1	0	1	1	0	1	0	1

which is equivalent to:

`a[7]`	`a[6]`	`a[5]`	`a[4]`	`a[3]`	`a[2]`	`a[1]`	`a[0]`
1	0	1	0	1	1	0	1

Your program should support an arbitrary number of digits for each binary number. (If it is necessary to have an upper limit, the minimum requirement would be to support up to 80 bits.)

In this assignment, we only deal with positive binary numbers, hence no need to consider how to represent negative binary numbers.

Addition

Addition of two binary numbers could be performed by calculating the sum of each digit, starting from the lowest digit (a.k.a. least significant digit; LSD) to the highest digit (a.k.a. most significant digit; MSD). Note that when the result of addition in a certain digit is larger than 1, then it should be carried to the higher digit. For example, addition of two binary numbers 110110 and 11011 can be calculated as the following:

	MSD						LSD
carried over	1	1	1	1	1	0	0
		1	1	0	1	1	0
+		0	1	1	0	1	1
	1	0	1	0	0	0	1

Each digit can be calculated based on the truth table below, where `x` and `y` are the value at the corresponding digit, and `c` is a value carried over from the lower digit. The result of addition is given as `Z` which is the value at the corresponding digit, while `C` is the value to be carried up to the higher digit (i.e. this will be used as `c` when calculating the next digit higher in order). In the example above, when calculating the third digit from the right, `x=1`, `y=0`, `c=1`, and as a result, `Z=0`, and `C=1` which is the value used for `c` when calculating the next digit (i.e. fourth digit from the right).

Input			Output	
c	x	y	Z	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Based on this truth table, you can formulate logical expressions that can calculate `Z` and `C` values based on `x`, `y`, and `c`, as the following:

```
Z = (x XOR y) XOR c
C = (x AND y) OR (c AND (x XOR y))
```

Note that `XOR` is an exclusive OR operation.

Subtraction

Subtraction of two binary numbers can also be performed in a similar fashion, bit by bit starting from the LSD, while using the concept of borrowing from the higher digit (or lending to the lower digit).

	MSD					LSD
lent	1	1	0	1	1	0
	1	1	0	1	1	0
−	0	1	1	0	1	1
	0	1	1	0	1	1

Based on this information, you should formulate a truth table that describes subtraction in each bit. As it was in the truth table for addition, `x` and `y` are the binary values at the corresponding digit of the two numbers being subtracted, and `Z` is the corresponding digit of the result of subtraction, while `b` describes if it has lent to the lower digit and `B` describes if it had to borrow from the higher digit (which will be then used as `b` when calculating the next digit higher in order). In the example above, when calculating the third digit from the right, `x=1`, `y=0`, `b=1`, and as a result, `Z=0`, and `B=0` which is the value used for `b` when calculating the next digit (i.e. fourth digit from the right).

You must complete the truth table shown below based on the concept of binary number subtraction as described above, and also formulate the logical expressions for calculating the values of `Z` and `B` based on `x`, `y`, and `b`.

Researching about the concepts of 'full adder' and 'full subtractor' in digital electronic circuits may be helpful.

Input			Output	
b	x	y	Z	B
0	0	0	?	?
0	0	1	?	?
0	1	0	?	?
0	1	1	?	?
1	0	0	?	?
1	0	1	?	?
1	1	0	?	?
1	1	1	?	?

When needing to subtract a larger number from a smaller number, we can calculate the difference by subtracting the smaller number from the larger number instead, then adding the negative sign. For example, 3 – 8 = -5, which is a negative value with a magnitude of 8 – 3 = 5. To apply this to your program, you should first implement another binary operation that compares two binary numbers so that when the first input binary number is less than the second number, you can subtract the first number from the second number instead in order to calculate the magnitude of difference and print a negative sign in the front.

Comparing two multi-digit binary numbers can be achieved by comparing each digit starting from the LSD. The truth table below represents single-bit comparison, where `x` and `y` represent the current corresponding bits of the two binary numbers being compared, and the result `L` is set to 1 (i.e. True) if `x` is less than `y` while also considering another input `l` which represents the comparison result from lower digit (i.e. the value `L` which resulted from comparing the lower digit). You should complete the truth table based on the concept of binary number comparison as described above, and also formulate the logical expression for calculating the values of `L` based on `x`, `y`, and `l`.

Input			Output
l	x	y	L
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Programming Specification

Based on the concepts described above, write a program (in Python or Java) that calculates addition or subtraction of binary numbers with arbitrary number of digits. Your program must adhere to the specifications described below.

- Your program must print out your own personal details as following:

```
Author: Steve Jobs
Student ID: 12345678
Email ID: jobst007
This is my own work as defined by the University's
Academic Integrity Policy.
```

- Your source code must include sufficient amount of comments explaining your program.

- Your program must first define the NOR (a.k.a. NOT OR) bit operation function defined as the following truth table, considering the value 1 as True and 0 as False. This function must be implemented using `if` and `return` statements, no mathematical or bit operations are allowed.

Function name	Description
`NOR(x, y)`	Takes two input arguments `x` and `y` as binary digits (i.e., either `0` or `1`) and returns a binary digit (either `0` or `1`) based on the following truth table.

x	y	x NOR y
0	0	1
0	1	0
1	0	0
1	1	0

- Given NOR is a functionally complete operation, implement the following functions for single-bit addition/subtraction/comparison using a combination of NOR operations. Each function below must be implemented by only calling the `NOR(x, y)` function defined above and returning its result. If needed you may define your own intermediate functions and call them, yet those functions must be also implemented by only calling the `NOR(x, y)` function.

Function name	Description
`ADD_BIT(x, y, c)`	Performs a single-bit addition operation by taking two input arguments `x` and `y` as binary digits (i.e., either `0` or `1`), and returns a binary digit which is a result of adding the two input arguments `x` and `y` (i.e. `x` + `y`) while considering `c` which is carried over from the result of addition in the lower digit (i.e. output of `CARRY_BIT()` function at the lower digit).
`CARRY_BIT(x, y, c)`	Returns a value to be carried over to the higher digit when calculating single-bit addition of `x` and `y` while considering `c` which is carried over from the lower digit (i.e. output of `CARRY_BIT()` function at the lower digit).
`SUB_BIT(x, y, b)`	Performs single-bit subtraction operation by taking two input arguments `x` and `y` as binary digits (i.e., either `0` or `1`), and returns a binary digit which is a result of subtracting `y` from `x` (i.e. `x` – `y`) considering `b` which indicates it has lent to the

	lower digit subtraction (i.e. output of `BORROW_BIT()` function at the lower digit).
`BORROW_BIT(x, y, b)`	Returns whether if it had to borrow from the higher digit when calculating the single-bit subtraction of `x` and `y` considering `b` which indicates it has lent to the lower digit (i.e. output of `BORROW_BIT()` function at the lower digit).
`LESS_THAN(x, y, l)`	Performs single-bit comparison by taking two input arguments `x` and `y` as binary digits (i.e., either `0` or `1`), and returns a binary digit which indicates whether if `x` is less than `y` considering `l` which was the result of comparing the lower digit (i.e. output of `LESS_THAN()` function at the lower digit).

- Using the functions above, print the truth tables of the `NOR()`, `ADD_BIT()`, `CARRY_BIT()`, `SUB_BIT()`, `BORROW_BIT()` and `LESS_THAN()` functions as shown in the Sample Output provided in the next section.

- The program must calculate addition and subtraction of arbitrary length binary numbers using the functions defined above.

- The program must first ask the user to choose the type of operation to perform with showing a prompt as the following:

```
    Choose operation [+, -, q]:
```

- If the user chooses 'q', the program terminates. If the user enters an invalid operation, it should print an error message and ask again. Otherwise, the program must ask for two binary numbers X and Y and perform the relevant operation as described below.

- The program must ask the user to enter binary numbers as string input to perform selected operation. The program must allow for an arbitrary number of digits for each binary number. (If it is necessary to have an upper limit, the minimum requirement is to support up to 80 bits.) If the input string is not a binary number (i.e. it includes characters other than '0' or '1'), the program must ask the user to enter a binary number again. Below is an example of how binary number input works (user inputs are shown in italic font):

```
    X: 391012
    Not a binary number!
    X: 100100111101
    Y: a234
    Not a binary number!
    Y: 1010010100111
```

- The program must convert each input string into an array of numbers with each element representing a binary digit (0 or 1) as described in the "Concepts" section above. Then using the bit operation functions defined above, calculate the sum (or difference, depending on the chosen operation) and print an output of the results strictly in a format shown in the Sample Output provided in the next section. Note the numbers are right aligned with no leading zeros.

- The program should repeat the above from asking another operation to perform, unless the user enters 'q' in which case it should terminate.

Sample Output

User inputs are shown in *italic* font, while certain parts of the output are marked in red indicating that your output must have the correct relevant values in these places.

```
Author: Steve Jobs
Student ID: 12345678
Email ID: jobst007
This is my own work as defined by the University's Academic
Integrity Policy.

NOR
x y Z
-----
0 0 1
0 1 0
1 0 0
1 1 0

ADDITION
c x y Z C
---------
0 0 0 0 0
0 0 1 1 0
0 1 0 1 0
0 1 1 0 1
1 0 0 1 0
1 0 1 0 1
1 1 0 0 1
1 1 1 1 1

SUBTRACTION
b x y Z B
---------
0 0 0 ? ?
0 0 1 ? ?
0 1 0 ? ?
0 1 1 ? ?
1 0 0 ? ?
1 0 1 ? ?
1 1 0 ? ?
1 1 1 ? ?

LESS_THAN
l x y L
-------
0 0 0 ?
0 0 1 ?
0 1 0 ?
0 1 1 ?
1 0 0 ?
1 0 1 ?
1 1 0 ?
1 1 1 ?
```

```
Choose operation [+, -, q]: a
Invalid operation.

Choose operation [+, -, q]: +
X: 1101110110100102
Not a binary number!
X: 1101110110100101
Y: 101001011011

  1101110110100101
+     101001011011
------------------
  1110100000000000

Choose operation [+, -, q]: +
X: 11010010
Y: abcdef
Not a binary number!
Y: 10010001

  11010010
+ 10010001
----------
 101100011

Choose operation [+, -, q]: -
X: 110111011101001
Y: 11110110

  110111011101001
-        11110110
-----------------
  110110111110011

Choose operation [+, -, q]: -
X: 101001
Y: 0110100

  101001
- 110100
--------
   -1011

Choose operation [+, -, q]: +
X: 10001010010100110011
Y: 1011101011000011100011101010011001001011000101100011000110001010

                                              10001010010100110011
+ 1011101011000011100011101010011001001011000101100011000110001010
------------------------------------------------------------------
  1011101011000011100011101010011001001011000111101101011010111101

Choose operation [+, -, q]: -
X: 100000010001000111000000000100000101000000010010110001110000010
Y: 01100100011000110010110000100010001111000011100011000111110011
```

```
  100000010001000111000000000100000101000000010010110001110000010
-   1100100011000110010110000100010001111000011100011000111110011
-----------------------------------------------------------------
   10011101110000000101001111111110011000111110110011000110001111

Choose operation [+, -, q]: q
```

Report Specification

Your report must include your name, student ID, and email ID at the beginning of the report.

1. Present the truth tables for the single-bit subtraction and single-bit less-than comparison operations, and explain how did you derive them. Format the truth tables as shown in the "Concepts" section. Alternatively, the truth tables could be copy-pasted from the output of your program but must use a fixed-width font to make the characters aligned properly.

2. Present a NOR operation based logical expressions for the single-bit addition, single-bit subtraction and single-bit less-than comparison, and explain how you derived them.

3. Describe and explain how you perform multi-digit binary number addition, subtraction, and comparison, using the single-bit operation functions.

4. Present output of your program that answers the following questions:
   - 1101011000 + 1011010100 = ?
   - 1101110000 – 110101010 = ?
   - 111000110 – 1101110000 = ?
   - 11111000001111100000111110000011111000001111100000111110000011111 00000 + 111110000011111000001111100000111110000011111000001111100000 11111 = ?
   - 1111100000111110000011111000001111100000111110000011111000001111100000 – 11111000001111100000111110000011111000001111100000111110000011111 = ?

Submission

You must submit both the source code of your program and a report following the above specification through the learnonline course webpage by the due date as indicated on the submission link.

WARNING: Late submissions will be penalised by scaling the marks by 70%, unless a document proving acceptable reasons for late submission is provided to the course coordinator (e.g. medical certificate, or a letter from your work supervisor).

Academic Integrity

Students must be aware of the Academic Integrity Policy available at:
https://i.unisa.edu.au/policies-and-procedures/university-policies/academic/ab-69/
Deliberate breach of academic integrity such as plagiarism is subject to penalties. **Generative AI tools (e.g., ChatGPT or Copilot) are not allowed in this assessment**. Code similarities due to using Generative AI tools will be considered as a breach of academic integrity.

Marking Criteria

Criteria	Max	Mark	Comment
Program			
Correct implementation of the NOR function	1		
Correct implementation of the single-bit addition/subtraction/comparison functions, and printing correct truth tables of each operation.	5		
Correct behaviour of handling input (operation selection and binary number inputs)	1		
Correct results of operation in correct format (checked against the Sample Output provided)	6		
Subtotal	13		
Report			
Explanation of the truth tables for the single-bit subtraction and comparison	2		
Explanation of the NOR based logical expressions for the single-bit addition/subtraction/comparison	2		
Explanation of multi-digit binary number addition, subtraction and comparison	2		
Output with correct answers to specified questions	1		
Subtotal	7		
Penalties			
Program not printing correct student details	-2		
Insufficient amount of comments in code	-2		
Report missing student details	-2		
Grand Total	20		

* Late submissions will be penalised by scaling the marks by 70%.