**FIT 1047 - Assignment 1**

Name: Gloria Ooi Siew Phing

Student ID: 2746 7449

Tutorial: Tuesday 4-6pm

Tutor: Safi Uddin

**Task 1.1**

Given Truth Table

Table 1 shows the truth table that was provided in the Assignment1 question.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | X1 | X2 | X3 | X4 | Z1 | Z2 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 0 | 1 | 0 |
| 4 | 0 | 0 | 1 | 1 | 1 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 | 1 |
| 6 | 0 | 1 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 0 | 1 | 0 |
| 8 | 0 | 1 | 1 | 1 | 1 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 | 1 |
| 10 | 1 | 0 | 0 | 1 | 1 | 1 |
| 11 | 1 | 0 | 1 | 0 | 1 | 0 |
| 12 | 1 | 0 | 1 | 1 | 1 | 0 |
| 13 | 1 | 1 | 0 | 0 | 0 | 0 |
| 14 | 1 | 1 | 0 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 0 | 1 | 1 |
| 16 | 1 | 1 | 1 | 1 | 0 | 1 |

Table 1

Based on the truth table provided, it is split into 2 separate tables, with output Z1(Table 2) and Z2(Table 3) separately for easy understanding and explanation.

Output for Z1

|  |  |
| --- | --- |
|  |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **X1** | **X2** | **X3** | **X4** | **Z1** |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 1 | 1 |
| 7 | 0 | 1 | 1 | 0 | 1 |
| 8 | 0 | 1 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 1 | 1 |
| 11 | 1 | 0 | 1 | 0 | 1 |
| 12 | 1 | 0 | 1 | 1 | 1 |
| 13 | 1 | 1 | 0 | 0 | 0 |
| 14 | 1 | 1 | 0 | 1 | 0 |
| 15 | 1 | 1 | 1 | 0 | 1 |
| 16 | 1 | 1 | 1 | 1 | 0 |

Table 2

Output for Z2

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **X1** | **X2** | **X3** | **X4** | **Z2** |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 | 1 | 0 |
| 5 | 0 | 1 | 0 | 0 | 1 |
| 6 | 0 | 1 | 0 | 1 | 0 |
| 7 | 0 | 1 | 1 | 0 | 0 |
| 8 | 0 | 1 | 1 | 1 | 0 |
| 9 | 1 | 0 | 0 | 0 | 1 |
| 10 | 1 | 0 | 0 | 1 | 1 |
| 11 | 1 | 0 | 1 | 0 | 0 |
| 12 | 1 | 0 | 1 | 1 | 0 |
| 13 | 1 | 1 | 0 | 0 | 0 |
| 14 | 1 | 1 | 0 | 1 | 1 |
| 15 | 1 | 1 | 1 | 0 | 1 |
| 16 | 1 | 1 | 1 | 1 | 1 |

Table 3

Boolean Algebra Equation

First, only **outputs with Boolean 1** are highlighted (in yellow/green). Then, based on the inputs X1, X2, X3 and X4, the equation is created. An input of Boolean ‘**0’** for X1 will result in ‘’, and an input of Boolean **‘1’** will result in .

For example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **X1** | **X2** | **X3** | **X4** | **Z1** |
| 6 | 0 | 1 | 0 | 1 | 1 |

The complete equation will then be .

Final Equation

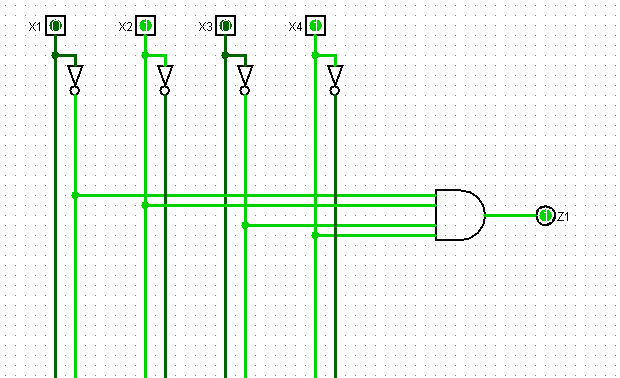
For Table 2, there are 10 equations that result in Z1. To combine all the equations, the equations are simply summed up.

As for Z2, the same concept is applied to obtain the equation below:

**Task 1.2**

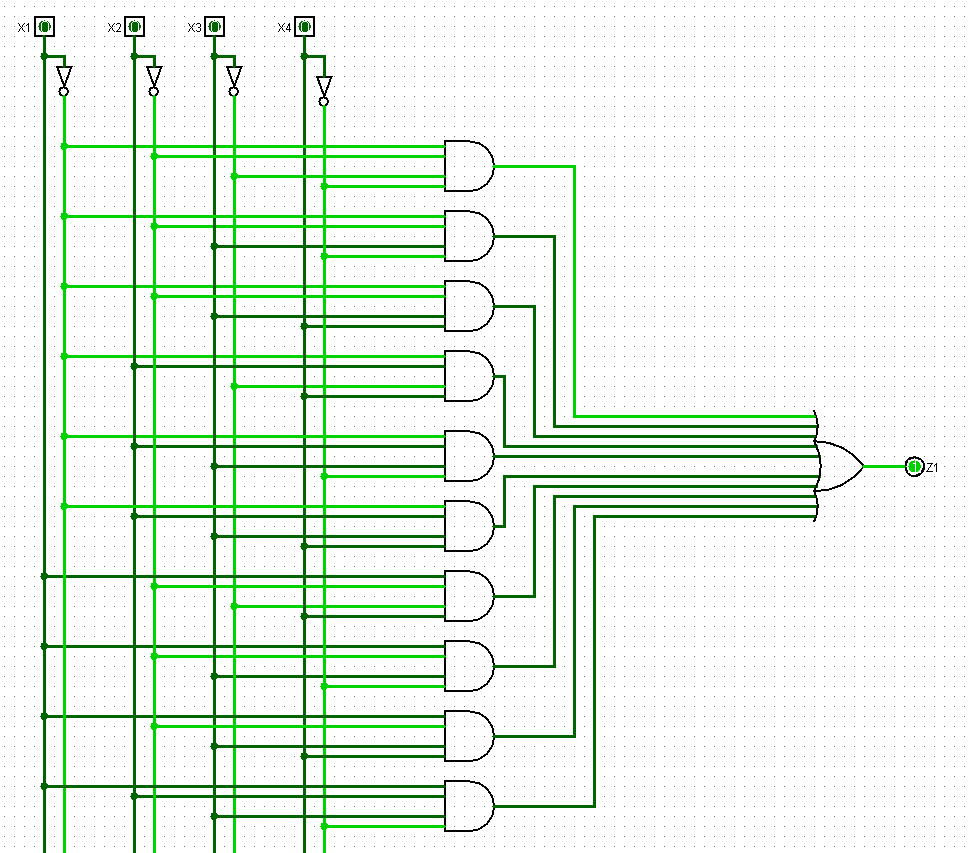
Logisim circuits are created based on the equations. A ‘∙’ sign in the equation signifies an AND gate, and a ‘+’ sign signifies an OR gate. A NOT gate is used to express any input that is .

For example, will result in Circuit1:

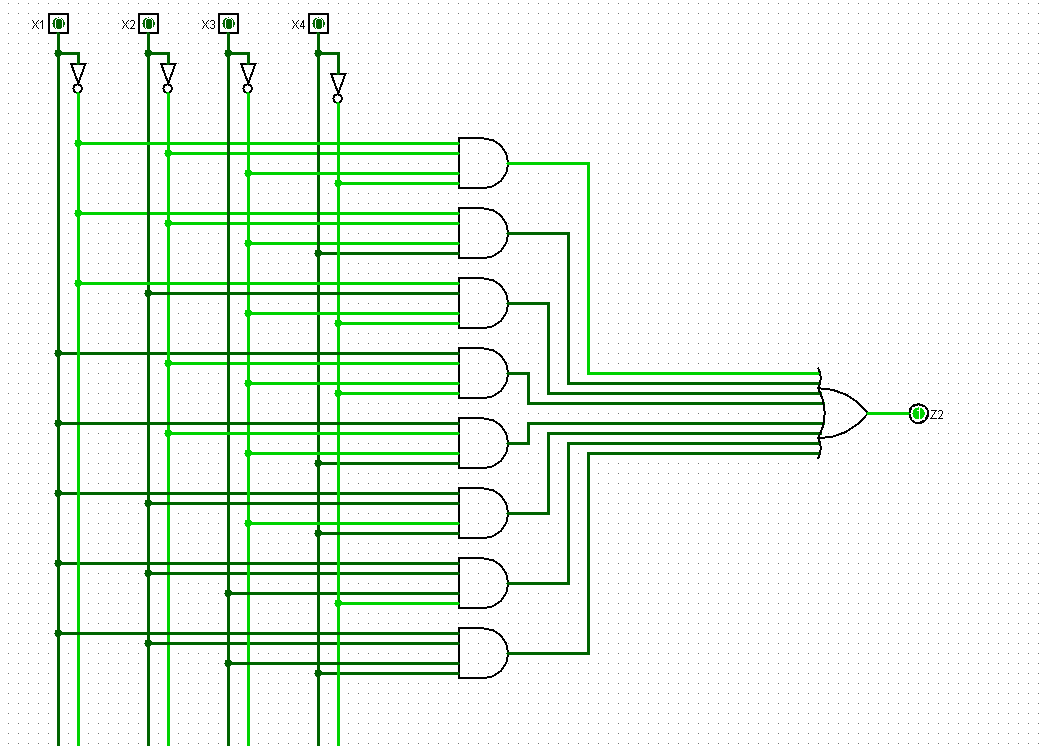


Circuit 1

For Z1, as there are ten equations, ten AND gates are used. After that, an OR gate is used to combine all the AND gates to give an accurate output of Z1. The same concept is applied for Z2. Below are the circuits for Z1(Circuit 2) and Z2(Circuit 3) respectively:

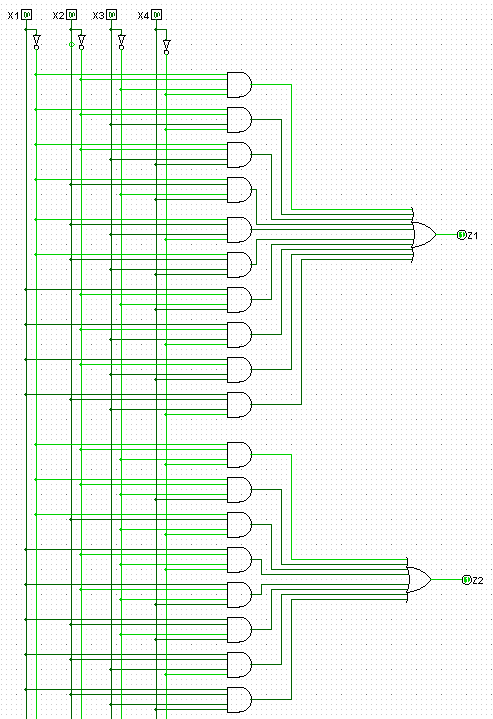


Circuit 2



Circuit 3

As Z1 AND Z2 have same input variables X1, X2, X3 and X4, the circuits can be combined to form the integrated circuit below containing outputs for both Z1 and Z2(Circuit 4):



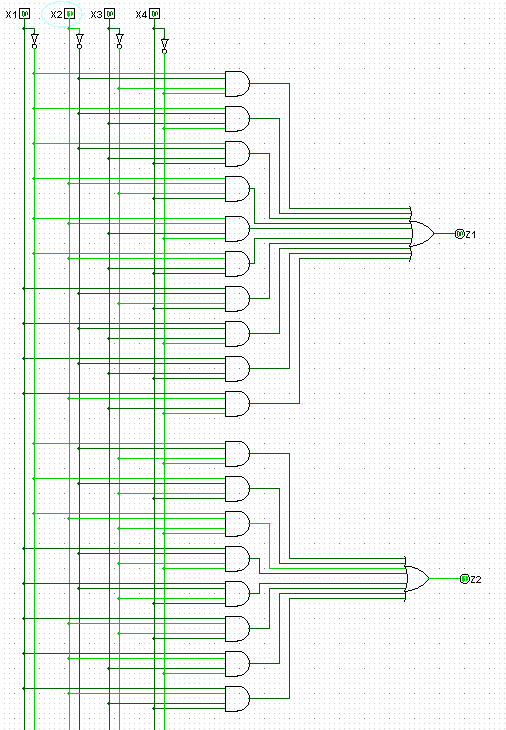
Circuit 4

Test Cases

3 test cases are documented to show that the integrated circuit gives the desired outputs. It is done by randomly selecting a given input and ensuring that it gives the respective outputs for Z1 and Z2.

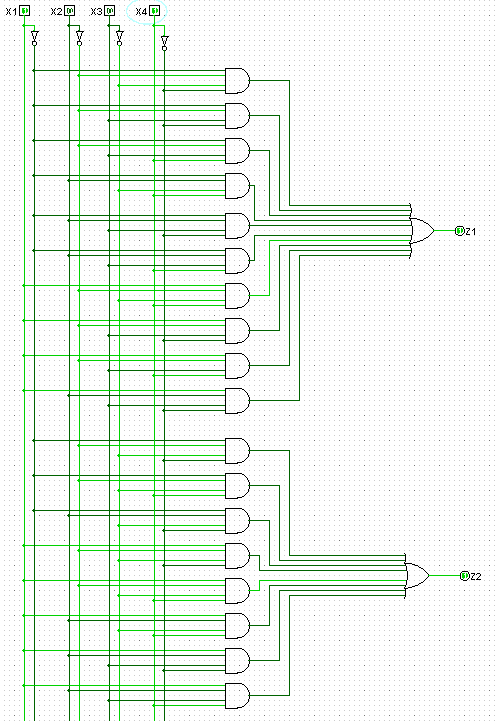
Test case 1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | X1 | X2 | X3 | X4 | Z1 | Z2 |
| 5 | 0 | 1 | 0 | 0 | 0 | 1 |



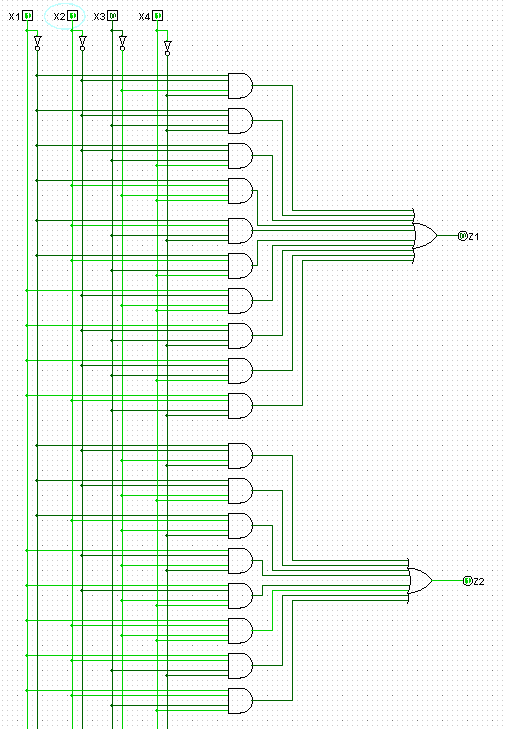
Test case 2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | X1 | X2 | X3 | X4 | Z1 | Z2 |
| 10 | 1 | 0 | 0 | 1 | 1 | 1 |



Test case 3:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | X1 | X2 | X3 | X4 | Z1 | Z2 |
| 14 | 1 | 1 | 0 | 1 | 0 | 1 |



**Task 1.3**

The final equation for Z1 and Z2 is quite large, and therefore can be simplified using Karnaugh maps(K-maps). For Z1, a table is created with the axis containing inputs for X1X2 on one side and inputs for X3X4 on the other (K-map 1). After that, the table is filled with the outputs of Z1. The same is done for Z2 with K-map 2.

Some rules for K-maps:

1. **Only** **Ones** are grouped together
2. **All Ones** must be grouped
3. Groups cannot be diagonal.
4. Groups must contain 2^n of cells, where n = 1,2,3..
5. Groups must be as large as possible
6. Groups may overlap
7. Groups can wrap around the table

After the grouping for K-maps is complete, similar inputs from the same groups are analysed to build equations. Table 4.1 and 4.2 contains the equations for K-map 1 and K-map 2 respectively.

Eg:

We want to determine the equation of the **Red** group from **K-map 1**. It can be seen that for both cells, X1X2 are constant, as well as X4. However, X3 varies in the 2 cells, so it can be ignored. Hence, the equation for the red group is

|  |  |  |
| --- | --- | --- |
| No. | Colour | Equation |
| 1 | Blue |  |
| 2 | Yellow |  |
| 3 | Purple |  |
| 4 | Red |  |
| 5 | Green |  |

Table 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | X3X4 | | | |
|  | **Z1** | 00 | 01 | 11 | 10 |
| X1X2 | 00 | 1 | 0 | 1 | 1 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 0 | 0 | 1 |
| 10 | 0 | 1 | 1 | 1 |

K-map 1

|  |  |  |
| --- | --- | --- |
| No. | Colour | Equation |
| 1 | Yellow |  |
| 2 | Purple |  |
| 3 | Red |  |
| 4 | Green |  |
| 5 | Blue |  |

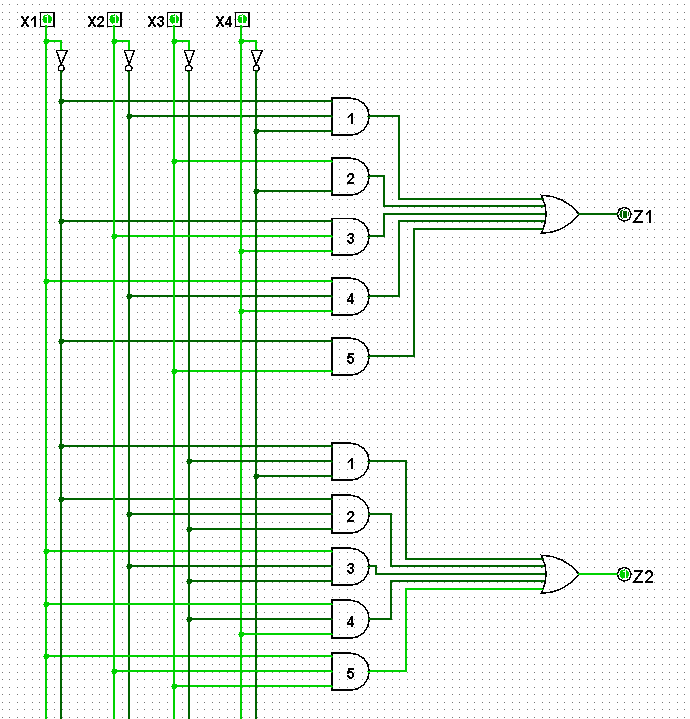
Table 5.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | X3X4 | | | |
|  | **Z2** | 00 | 01 | 11 | 10 |
| X1X2 | 00 | 1 | 1 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 1 | 1 |
| 10 | 1 | 1 | 0 | 0 |

K-map 2

Similar to previous explanation for combining equations, the final simplified equation for Z1 and Z2 is:

A simplified circuit is created in Logisim using AND gates for , OR gates for ‘+’ and NOT gate for (refer to Circuit 5).

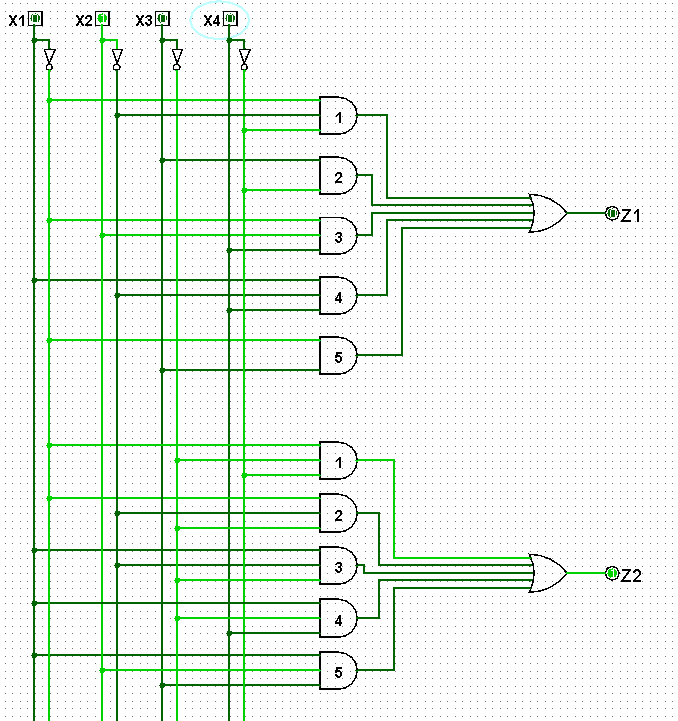


Circuit 5

The **same** test cases from Task 1.2 are again used to determine if the simplified circuits produce the correct output. The outputs for the simplified final circuit should be the same as the ones in Task 1.2.

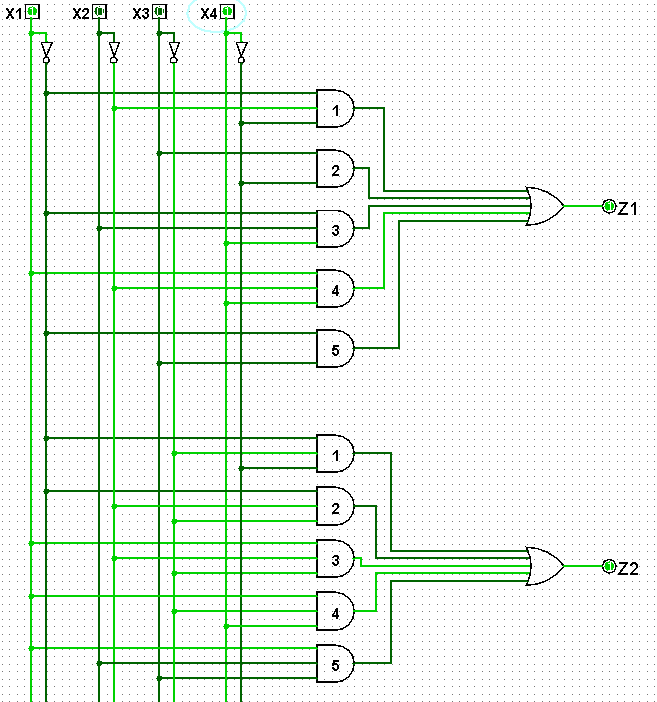
Test case 1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | X1 | X2 | X3 | X4 | Z1 | Z2 |
| 5 | 0 | 1 | 0 | 0 | 0 | 1 |



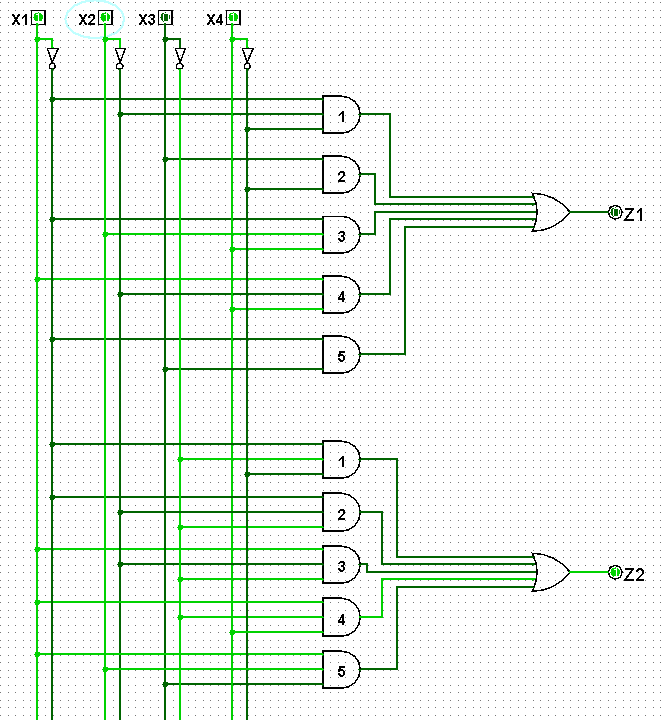
Test case 2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | X1 | X2 | X3 | X4 | Z1 | Z2 |
| 10 | 1 | 0 | 0 | 1 | 1 | 1 |



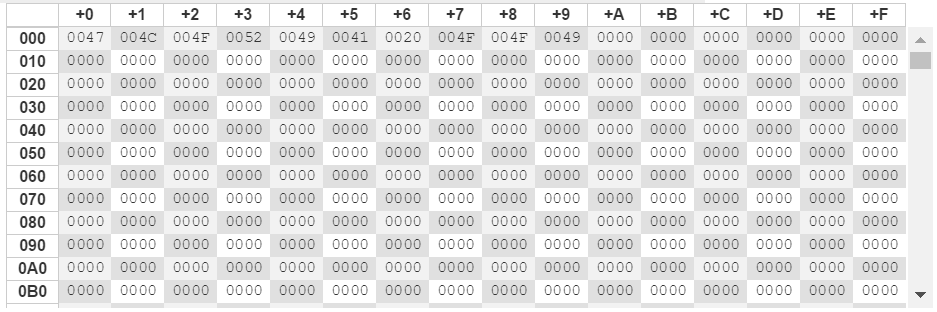
Test case 3:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | X1 | X2 | X3 | X4 | Z1 | Z2 |
| 14 | 1 | 1 | 0 | 1 | 0 | 1 |



**Task 2.1.1 – Your name as MARIE string**

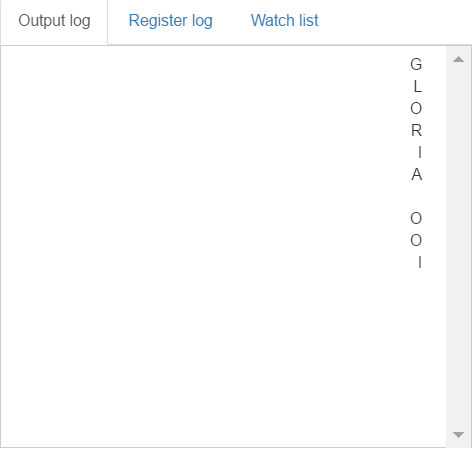
This task is to create a MARIE file that stores my name in memory. Below is a snapshot of the memory after the code is assembled with my name ‘GLORIA(space)OOI’.



**Task 2.1.2 – Printing a String**

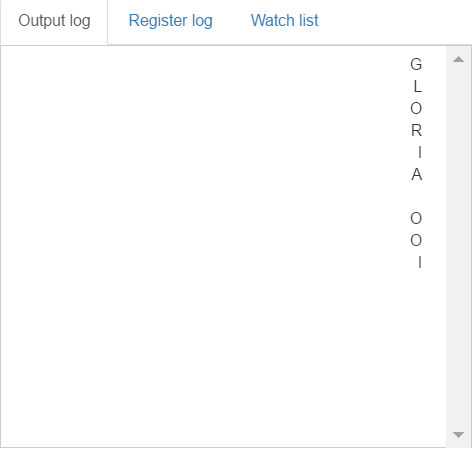
This task requests the program to print out the name stored in memory. A loop is created to iterate each string (in ASCII) until it reaches the value HEX 000. Hence, when I stored my name, the last value I stored is HEX 000. For each iteration, the programme will check if the value is 0 and end loop.

Below is the output:



**Task 2.1.3 – A Subroutine for Printing a String**

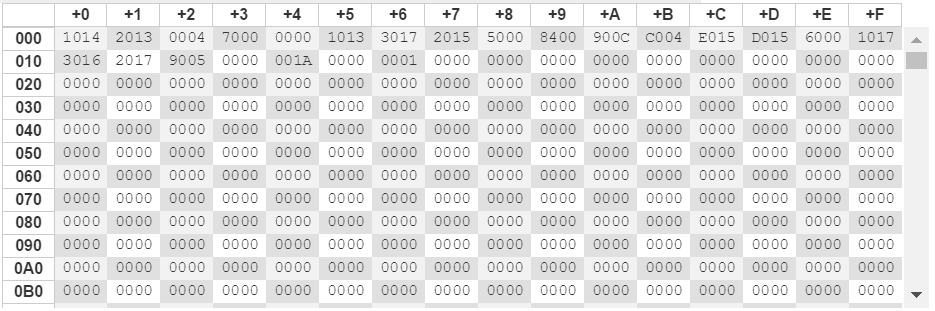
In Task 3, the address of the start of the string is 1st loaded and stored in *PrintString.* After that, the programme jumps into a subroutine called Print, iterates and prints each string found in memory until it reaches HEX 000, then exit the subroutine.

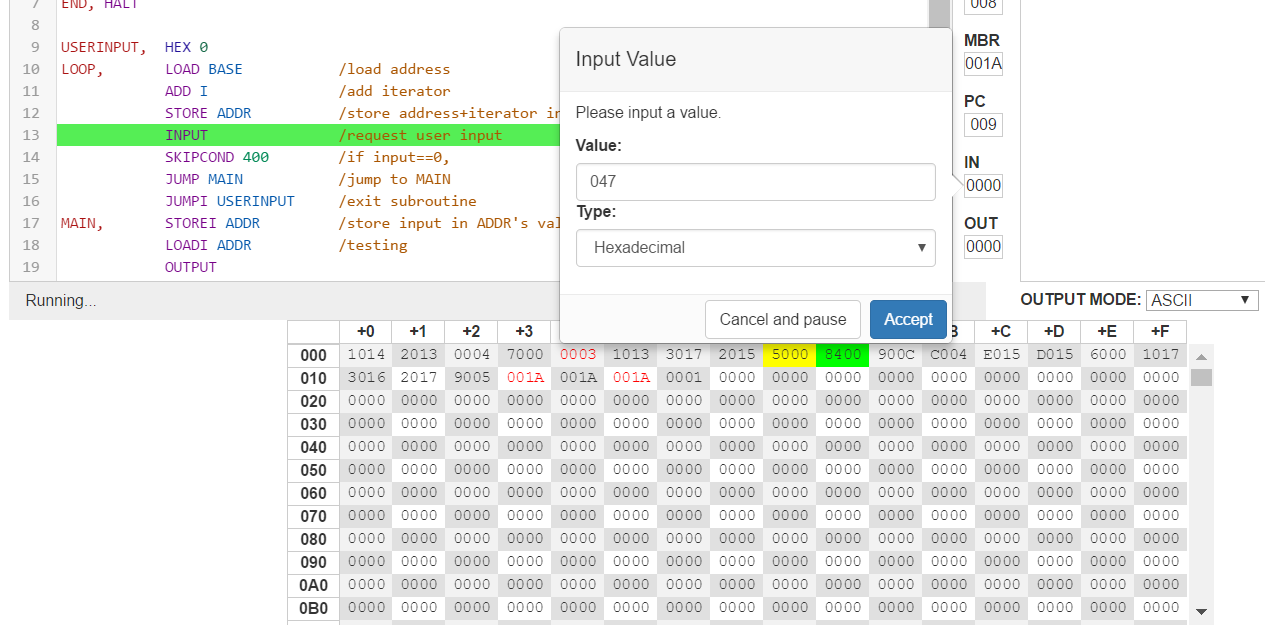
A test case is done by hard coding my name ‘GLORIA(space)OOI’. The output is as below: 

**Task 2.1.4 – User Input**

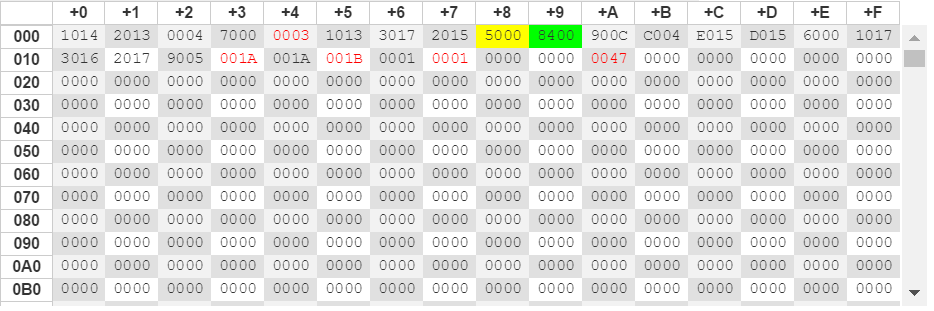
This task requires the programme to ask for user input. As can be seen from the test cases below, memory will store the value entered by user.

Memory after assembly:

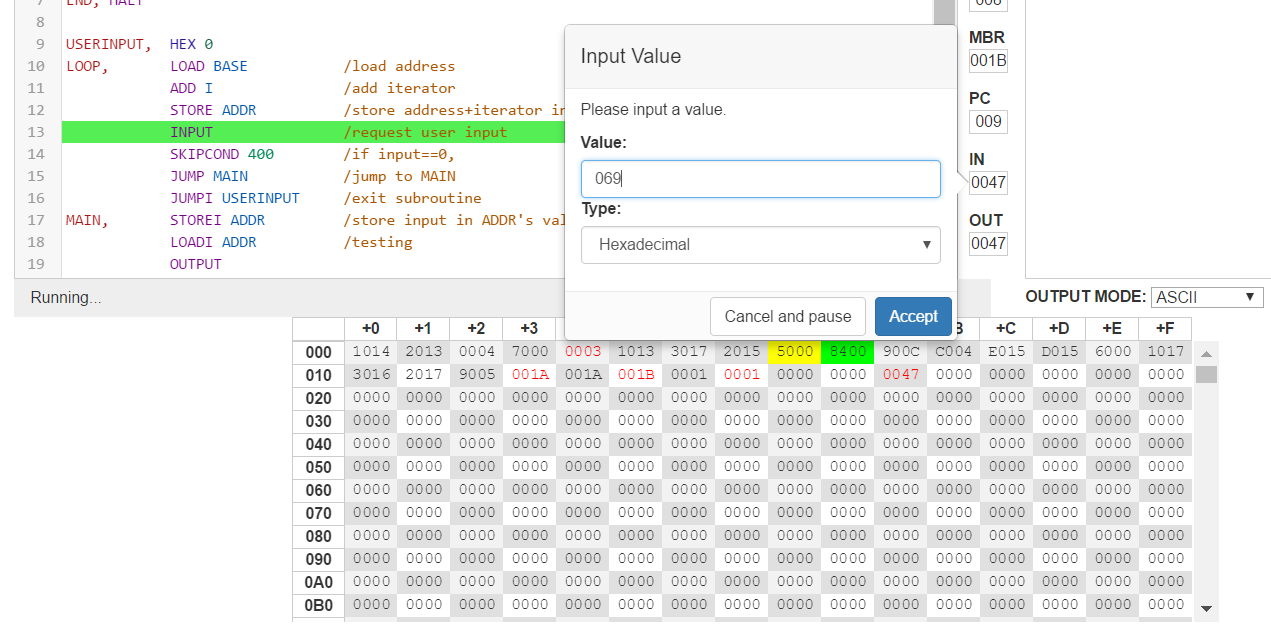


Test case 1: 

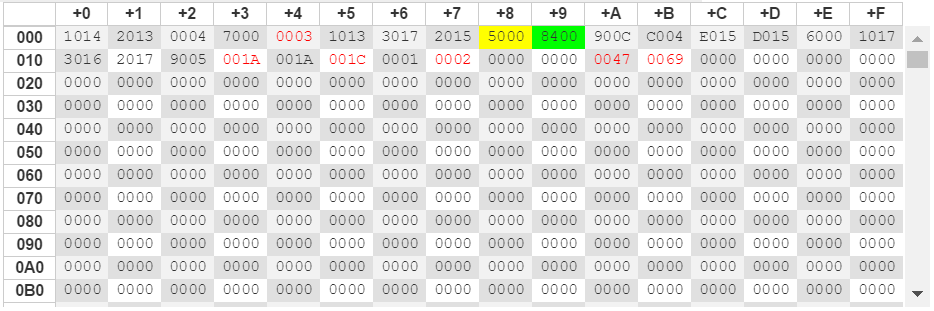
Memory after input 1:



Test case 2:



Memory after input 2:

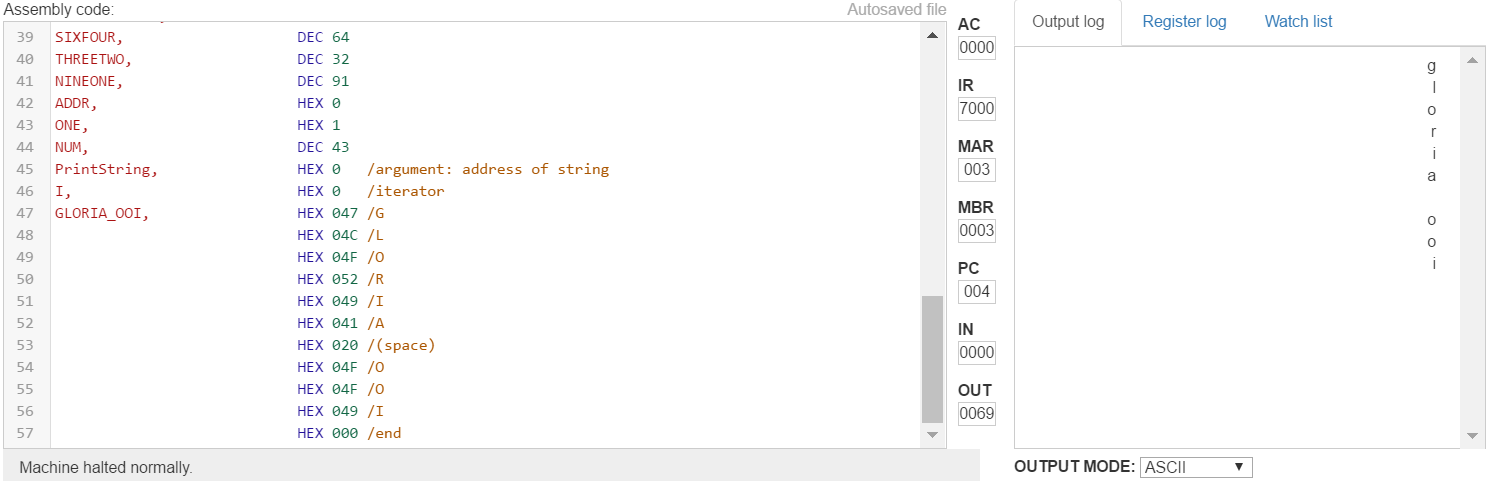


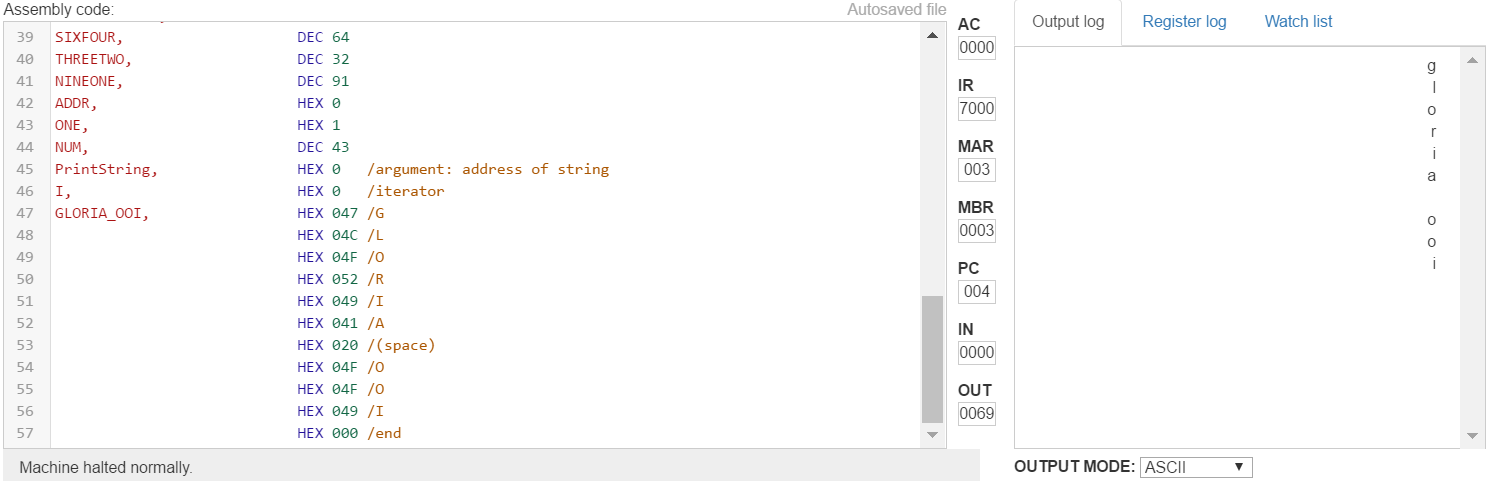
**Task 2.1.5 – Lower Case**

This task is to convert any upper-case letters into lowercase and ignore those that are not alphabets, such as symbols.

Test case 1:

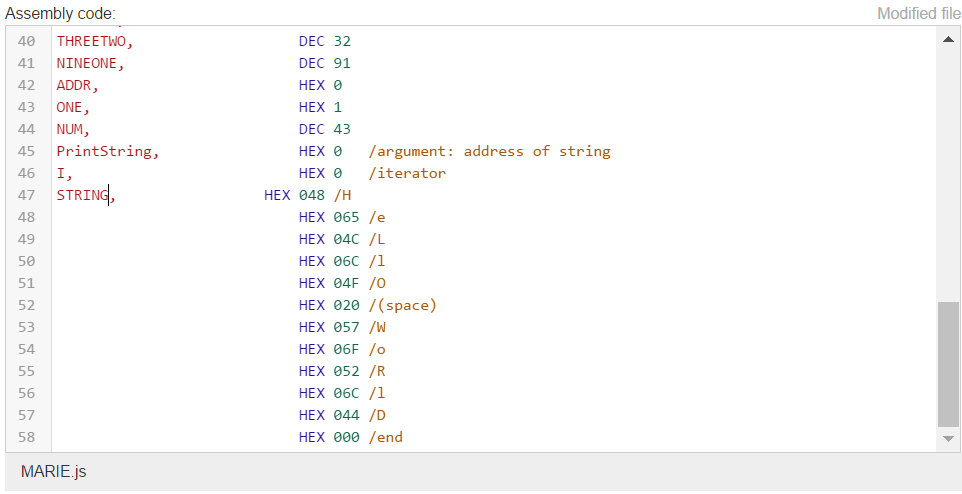
In test case one, the string stored is ‘GLORIA(space)OOI’, and the output is converted to ‘gloria(space)ooi’. Here, the space is ignored as it is not an alphabet.

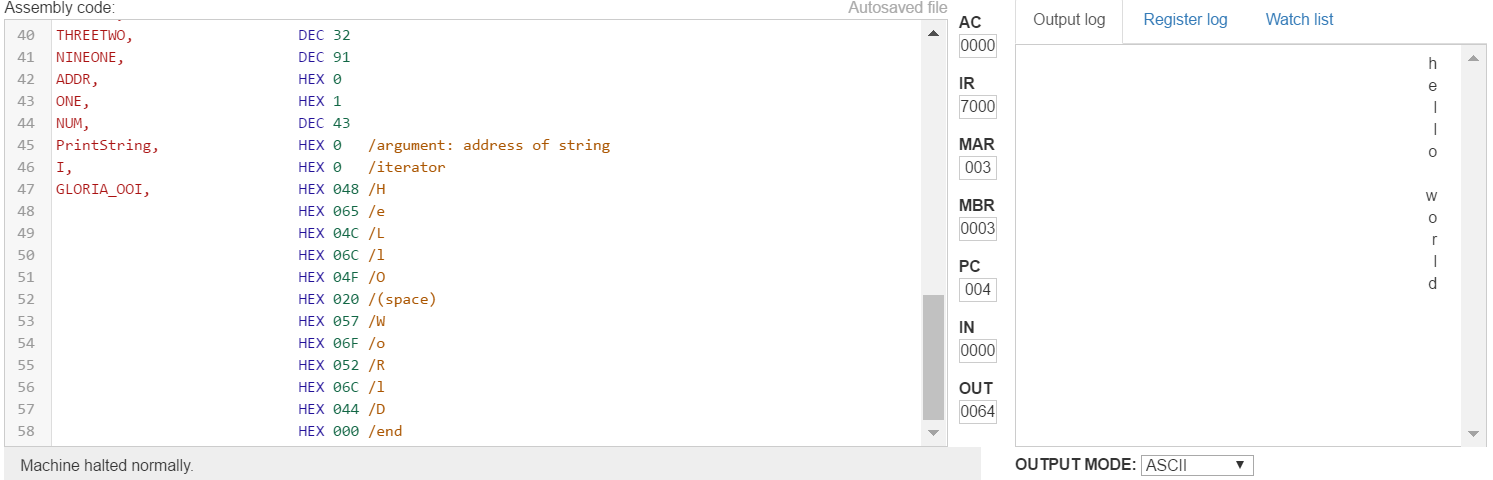




Test case 2:

In test case 2, the string stored is ‘HeLlo(space)WoRlD’. It has a mixture of upper and lower case letters. The output is ‘hello(space)world’.



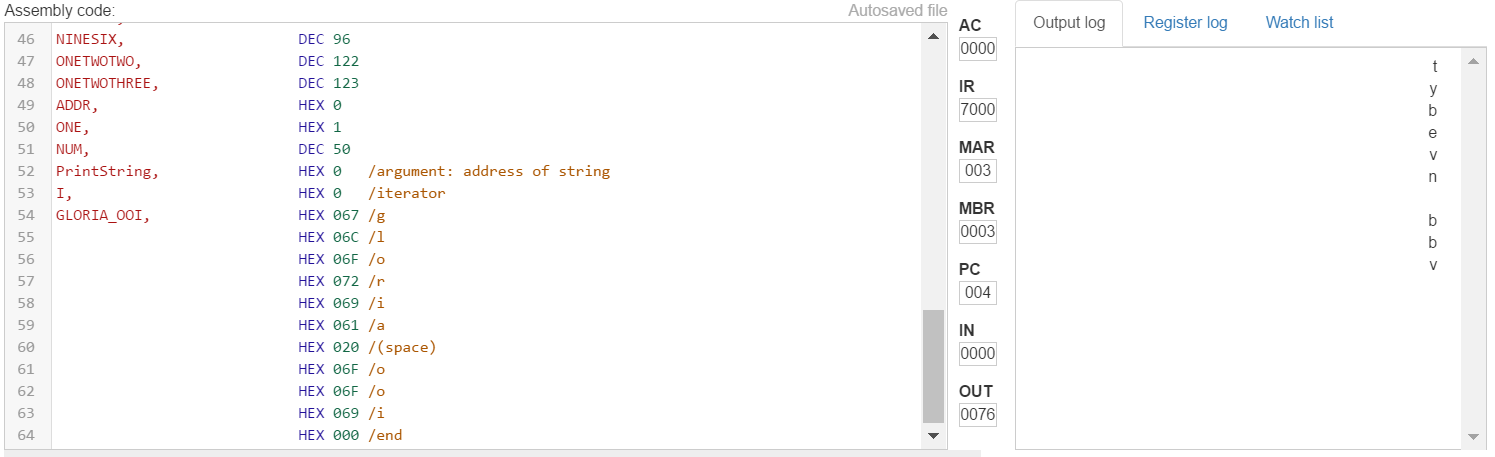


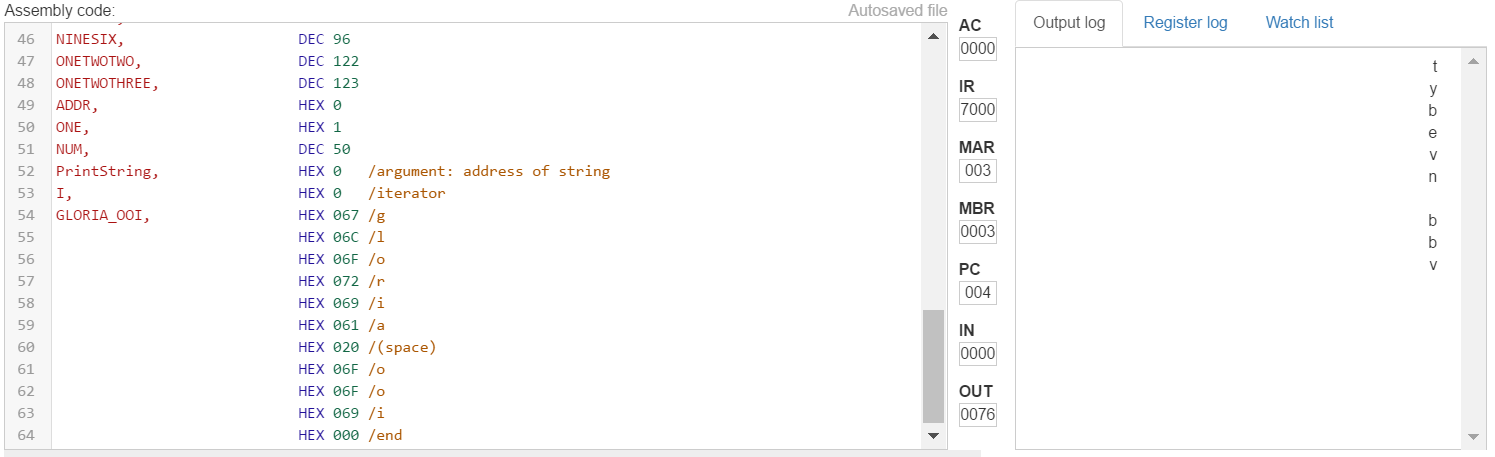
**Task 2.1.6 – ROT13**

This task require the programme to print out a string that is 13 places further in the alphabet, wrapping around from **z to a**. The programme will only make the conversion if the input is lower case alphabets, if not, the input will be left as it is.

Test case 1:

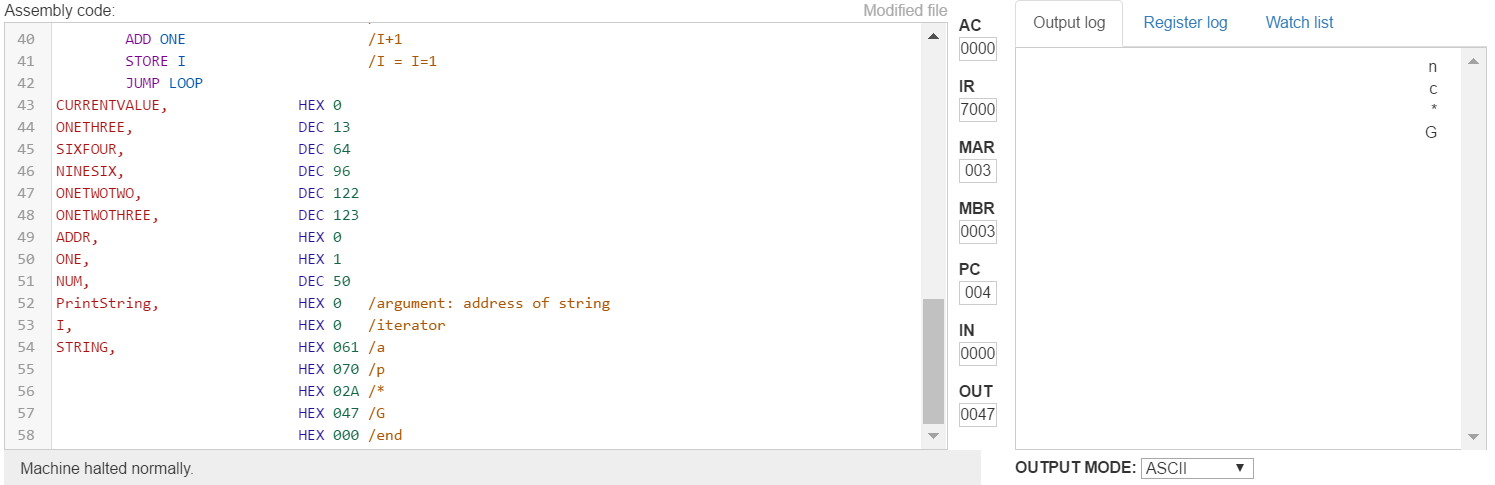
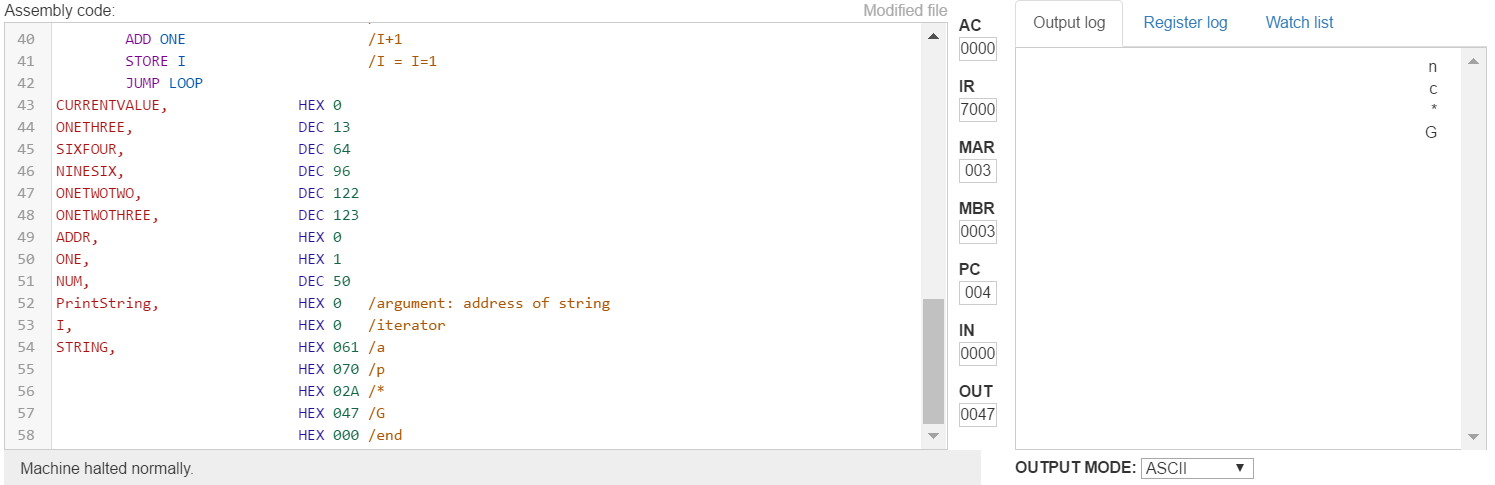
The string stored is my name ‘gloria(space)ooi’. After performing ROT13, the output is ‘tybevn(space)bbv’.





Test case 2:

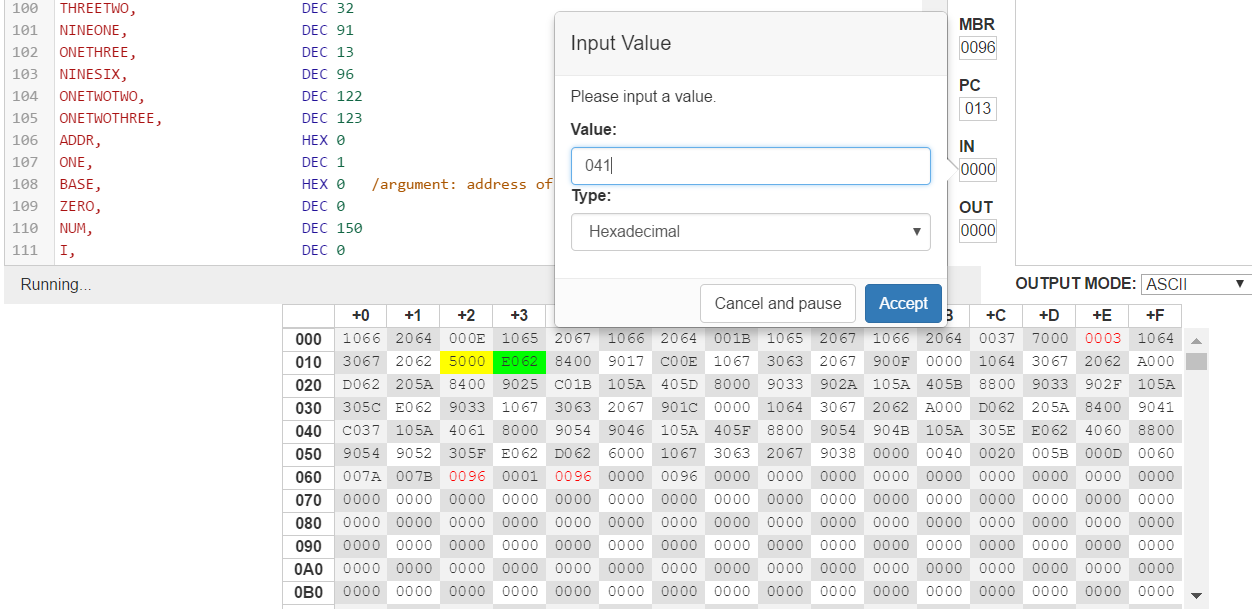
The input for test case 2 is ‘ap\*G’ and the output is ‘nc\*G’, proving that only lower case alphabets are converted.

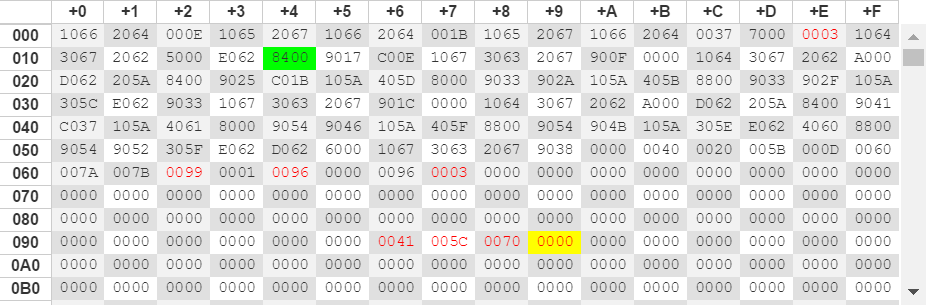
**Task 2.1.7 – Complete Programme**

For this task, a user is to input a string. If the string is upper-case, the programme will convert it into lower-case, and then perform ROT13. Strings that are not alphabets will be ignored.

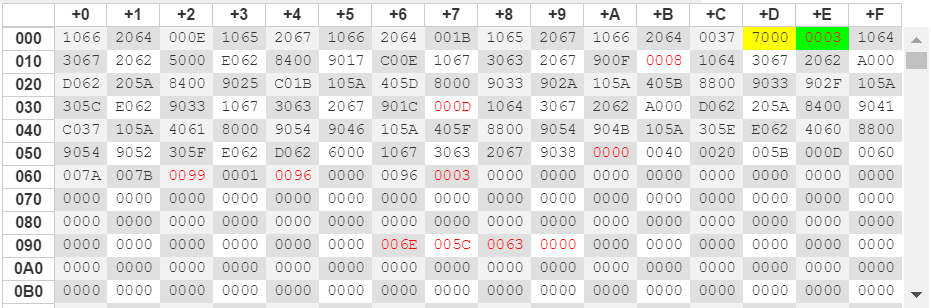
Below is a snapshot of the programme requesting for user input. Inputs of 041, 05C, 070, 000 are entered. They are ‘A’,’\’, ‘p’, ’null’ respectively.



After the inputs are entered, the memory will look like this



After converting upper-case alphabets to lower case and performing ROT13, the memory stores the new values.



This is the output:

