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

	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

1	$Z1 = \overline{X1} \cdot \overline{X2} \cdot \overline{X3} \cdot \overline{X4}$
2	
3	$Z1 = \overline{X1} \cdot \overline{X2} \cdot X3 \cdot \overline{X4}$
4	$Z1 = \overline{X1} \cdot \overline{X2} \cdot X3 \cdot X4$
5	
6	$Z1 = \overline{X1} \cdot X2 \cdot \overline{X3} \cdot X4$
7	$Z1 = \overline{X1} \cdot X2 \cdot X3 \cdot \overline{X4}$
8	$Z1 = \overline{X1} \cdot X2 \cdot X3 \cdot X4$
9	
10	$Z1 = X1 \cdot \overline{X2} \cdot \overline{X3} \cdot X4$
11	$Z1 = X1 \cdot \overline{X2} \cdot X3 \cdot \overline{X4}$
12	$Z1 = X1 \cdot \overline{X2} \cdot X3 \cdot X4$
13	
14	
15	$Z1 = X1 \cdot X2 \cdot X3 \cdot \overline{X4}$
16	

Output for Z2

	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

1	$Z2 = \overline{X1} \cdot \overline{X2} \cdot \overline{X3} \cdot \overline{X4}$
2	$Z2 = \overline{X1} \cdot \overline{X2} \cdot \overline{X3} \cdot X4$
3	
4	
5	$Z2 = \overline{X1} \cdot X2 \cdot \overline{X3} \cdot \overline{X4}$
6	
7	
8	
9	$Z2 = X1 \cdot \overline{X2} \cdot \overline{X3} \cdot \overline{X4}$
10	$Z2 = X1 \cdot \overline{X2} \cdot \overline{X3} \cdot X4$
11	
12	
13	
14	$Z2 = X1 \cdot X2 \cdot \overline{X3} \cdot X4$
15	$Z2 = X1 \cdot X2 \cdot X3 \cdot \overline{X4}$
16	$Z2 = X1 \cdot X2 \cdot X3 \cdot X4$

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 '$\overline{X1}$', and an input of Boolean **'1'** will result in '$X1$'.

For example:

	X1	X2	X3	X4	Z1
6	0	1	0	1	1
	$\overline{X1}$	$X2$	$\overline{X3}$	$X4$	$Z1$

The complete equation will then be $Z1 = \overline{X1} \cdot X2 \cdot \overline{X3} \cdot X4$.

Final Equation

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

$$\boldsymbol{Z1} = (\overline{X1} \cdot \overline{X2} \cdot \overline{X3} \cdot \overline{X4}) + (\overline{X1} \cdot \overline{X2} \cdot X3 \cdot \overline{X4}) + (\overline{X1} \cdot \overline{X2} \cdot X3 \cdot X4) + (\overline{X1} \cdot X2 \cdot \overline{X3} \cdot X4) + (\overline{X1} \cdot X2 \cdot X3 \cdot \overline{X4}) + (\overline{X1} \cdot X2 \cdot X3 \cdot X4) + (X1 \cdot \overline{X2} \cdot \overline{X3} \cdot X4) + (X1 \cdot \overline{X2} \cdot X3 \cdot \overline{X4}) + (X1 \cdot \overline{X2} \cdot X3 \cdot X4) + (X1 \cdot X2 \cdot X3 \cdot \overline{X4})$$

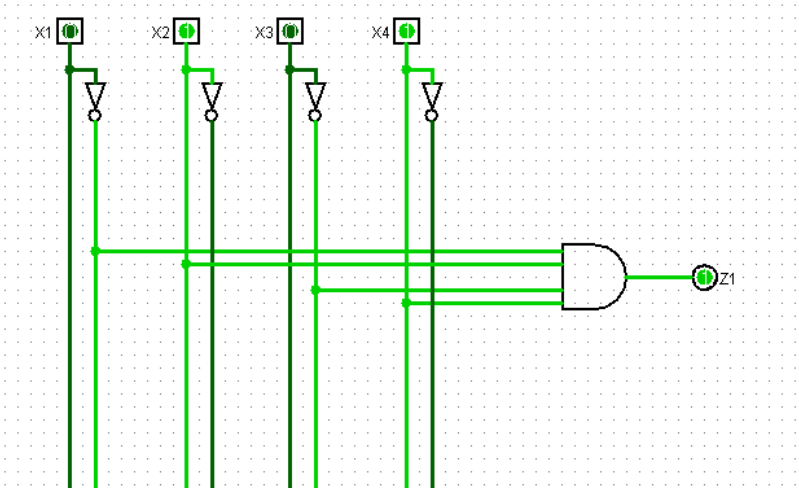
As for Z2, the same concept is applied to obtain the equation below:

$$\boldsymbol{Z2} = (\overline{X1} \cdot \overline{X2} \cdot \overline{X3} \cdot \overline{X4}) + (\overline{X1} \cdot \overline{X2} \cdot \overline{X3} \cdot X4) + (\overline{X1} \cdot X2 \cdot \overline{X3} \cdot \overline{X4}) + (X1 \cdot \overline{X2} \cdot \overline{X3} \cdot \overline{X4}) + (X1 \cdot \overline{X2} \cdot \overline{X3} \cdot X4) + (X1 \cdot X2 \cdot \overline{X3} \cdot X4) + (X1 \cdot X2 \cdot X3 \cdot \overline{X4}) + (X1 \cdot X2 \cdot X3 \cdot X4)$$

Task 1.2

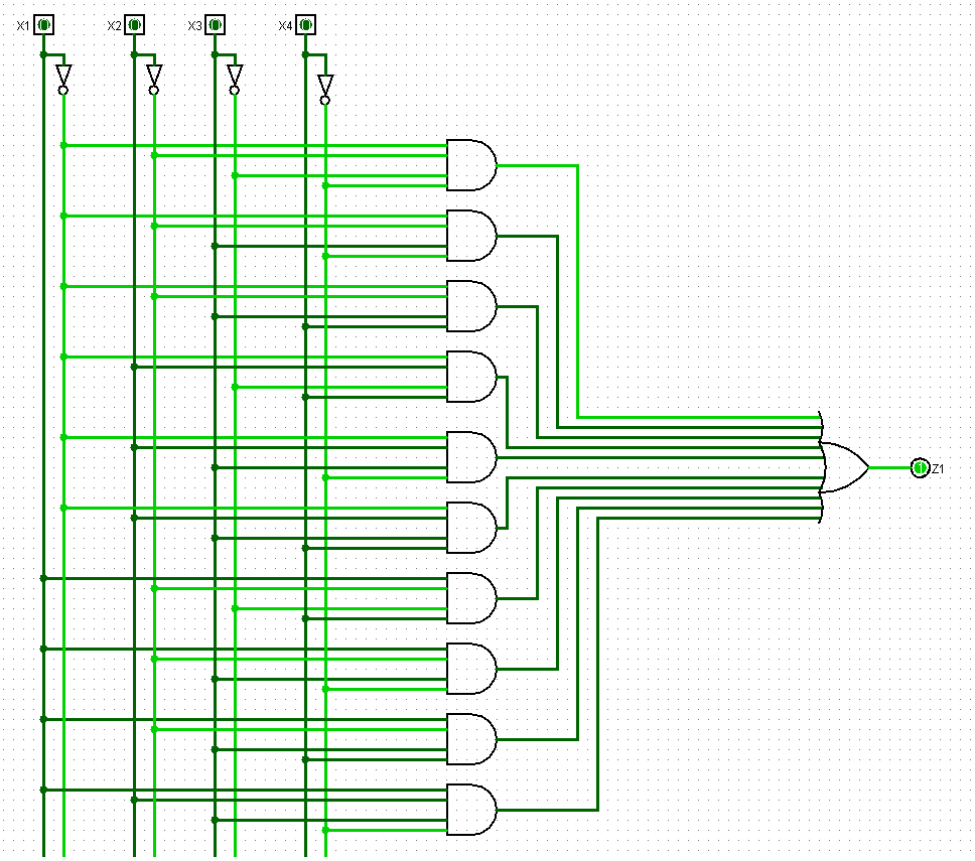
Logisim circuits are created based on the equations. A '·' sign in the equation signifies an AND gate, and a '+' sign signifies an OR gate. A NOT gate is used to express any input that is $\bar{X}$.

For example, $Z1 = \overline{X1} \cdot X2 \cdot \overline{X3} \cdot X4$ will result in Circuit1:

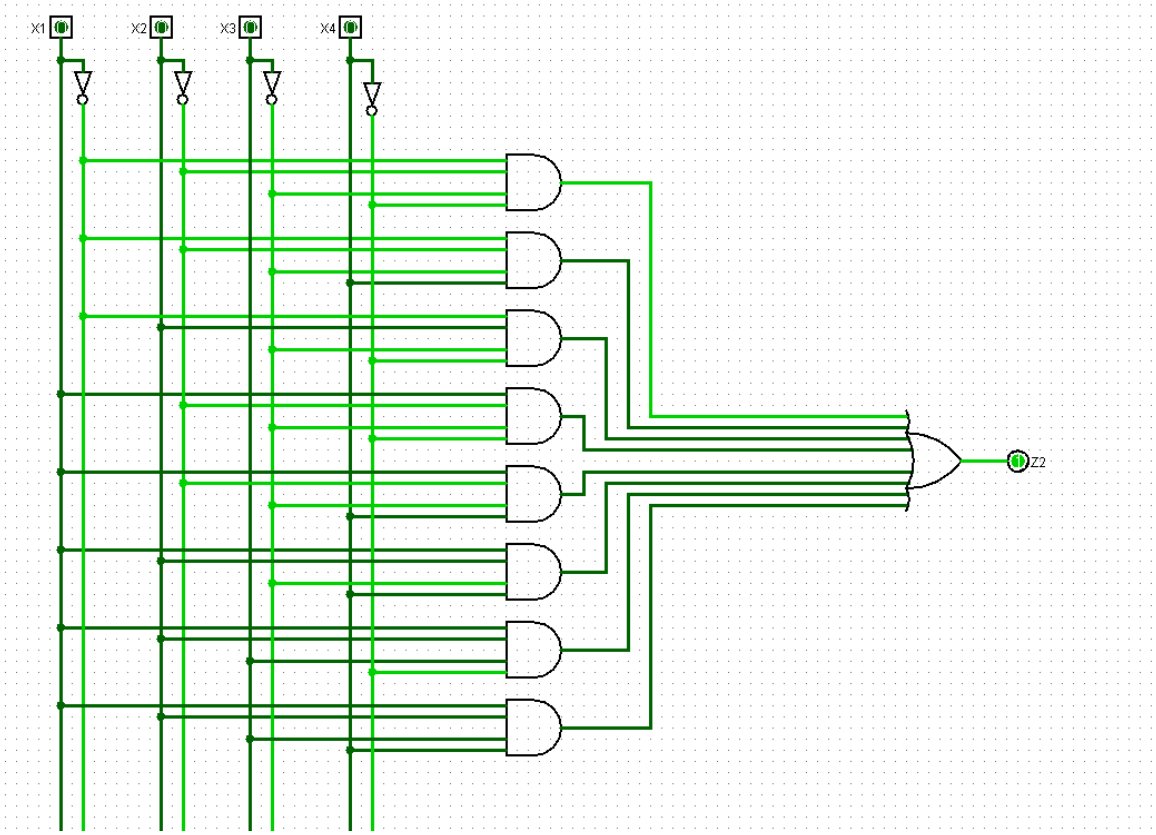


Circuit 1

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:

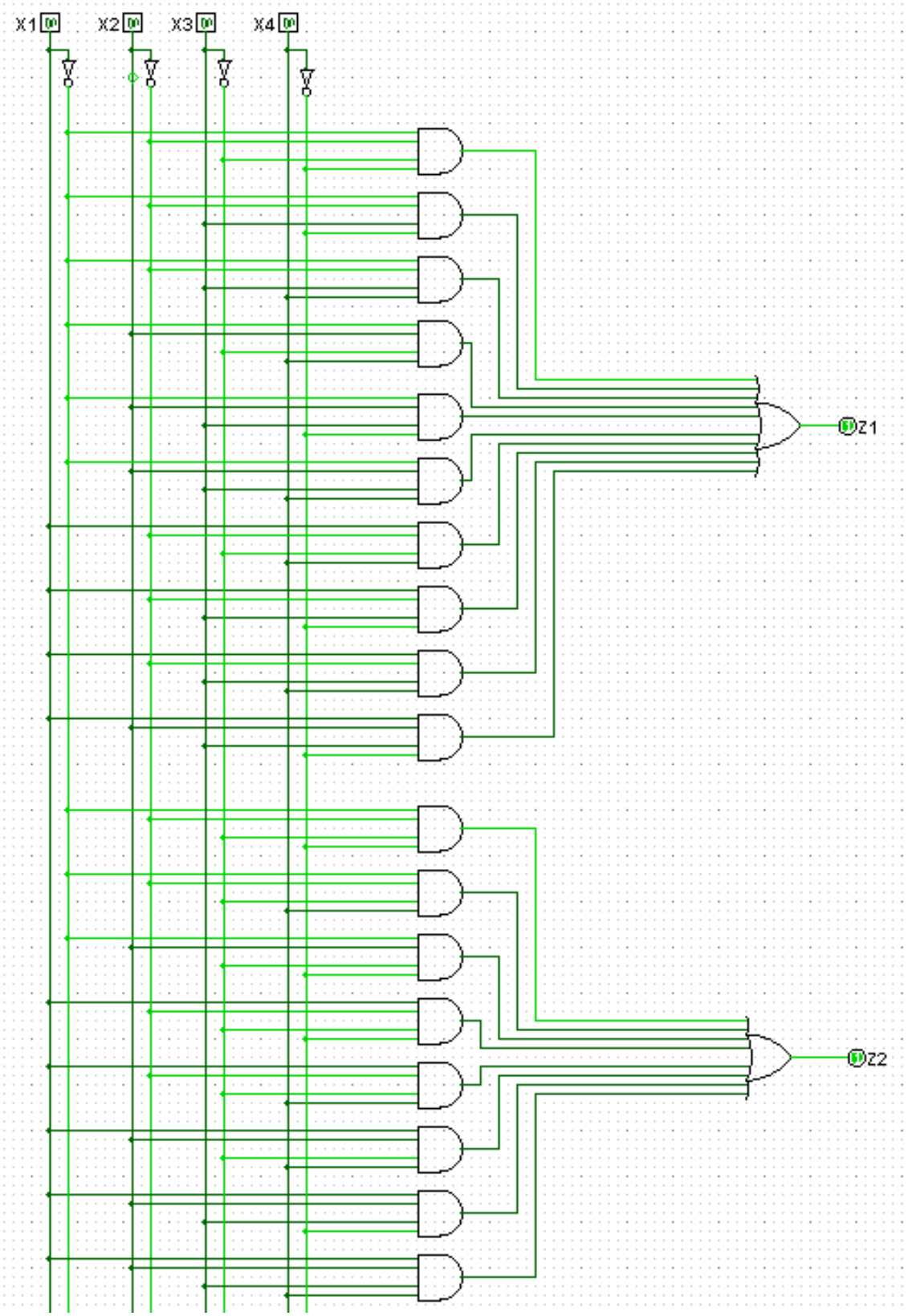


Circuit 2



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):



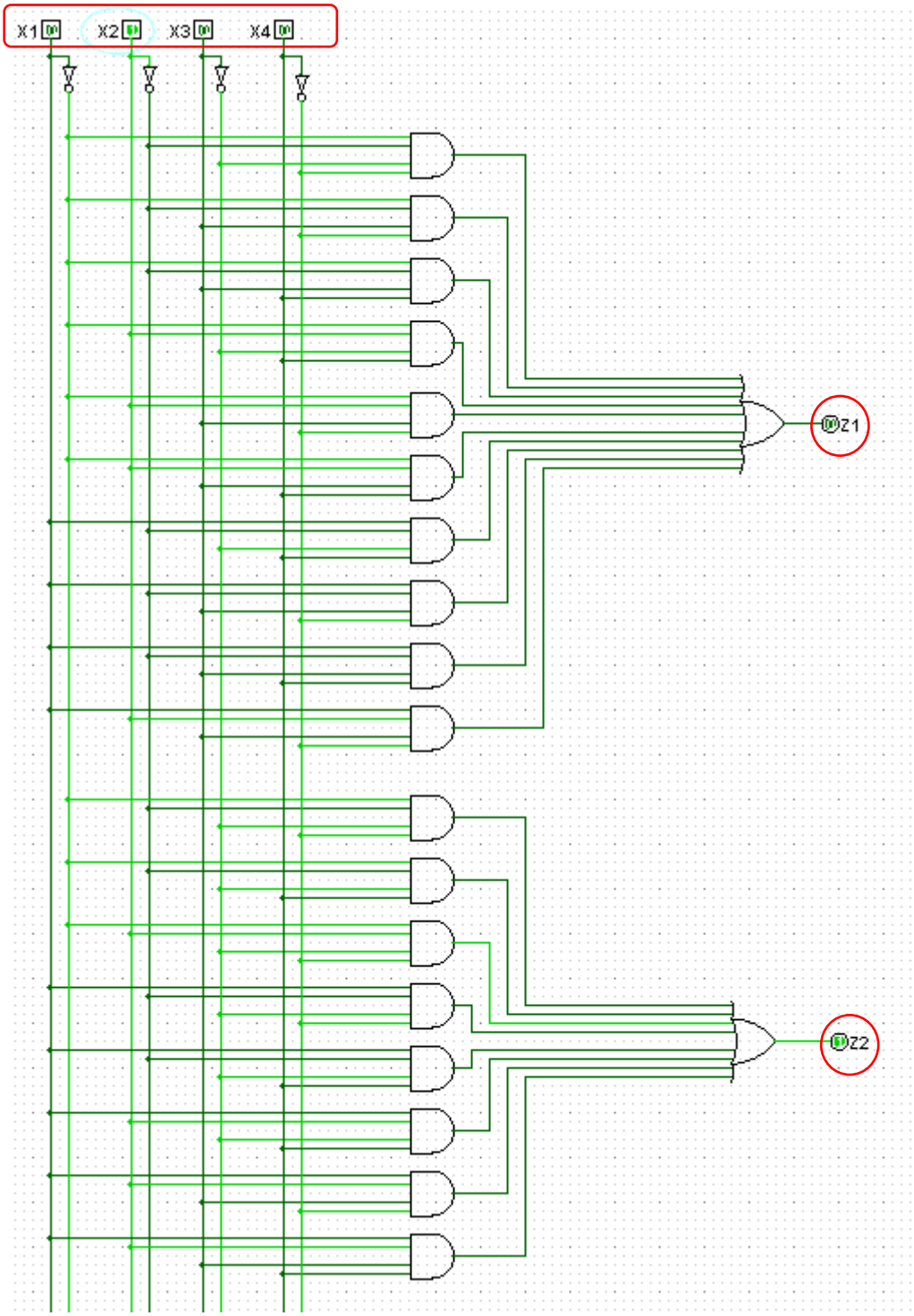
Circuit 4

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.

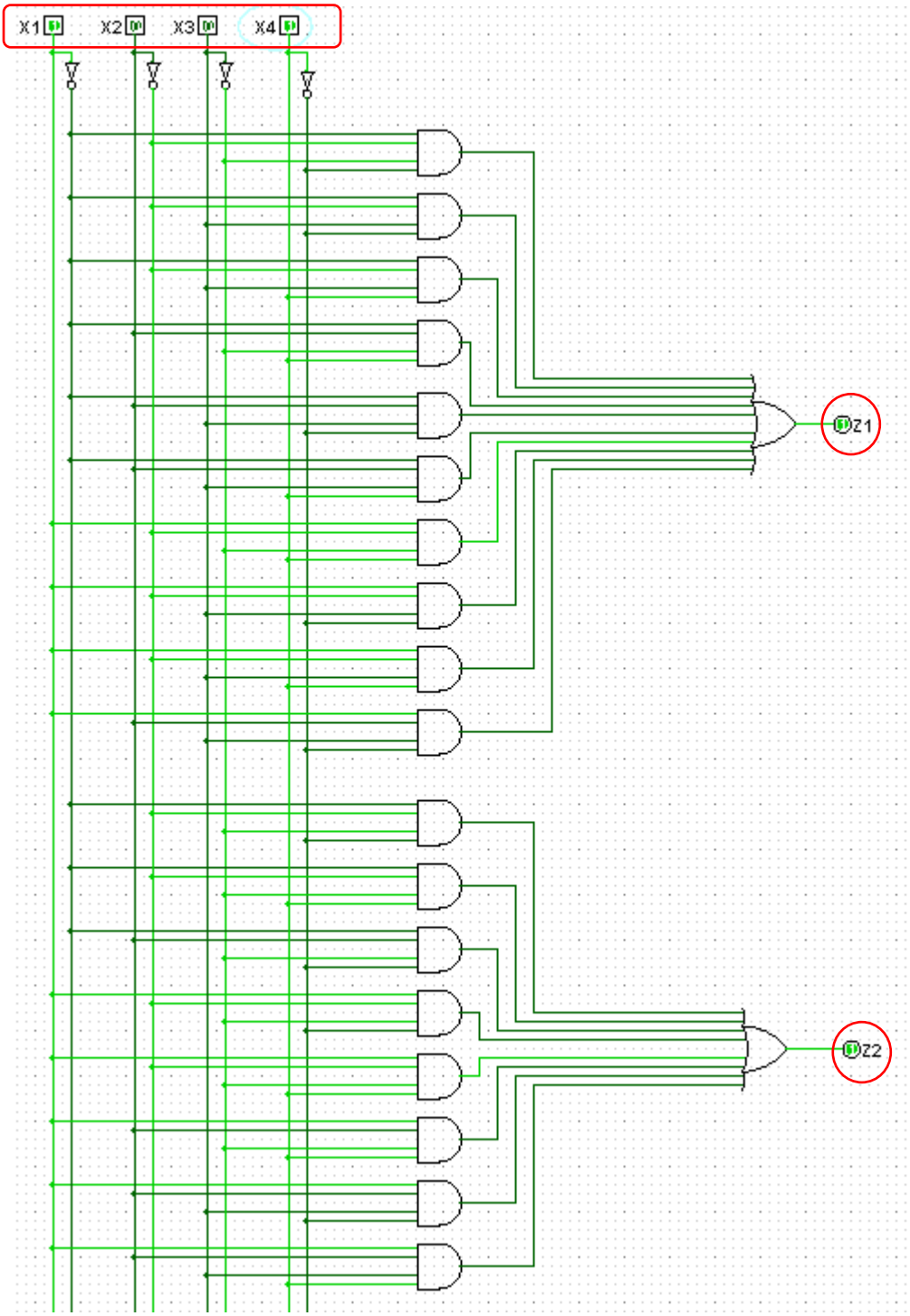
Test case 1:

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



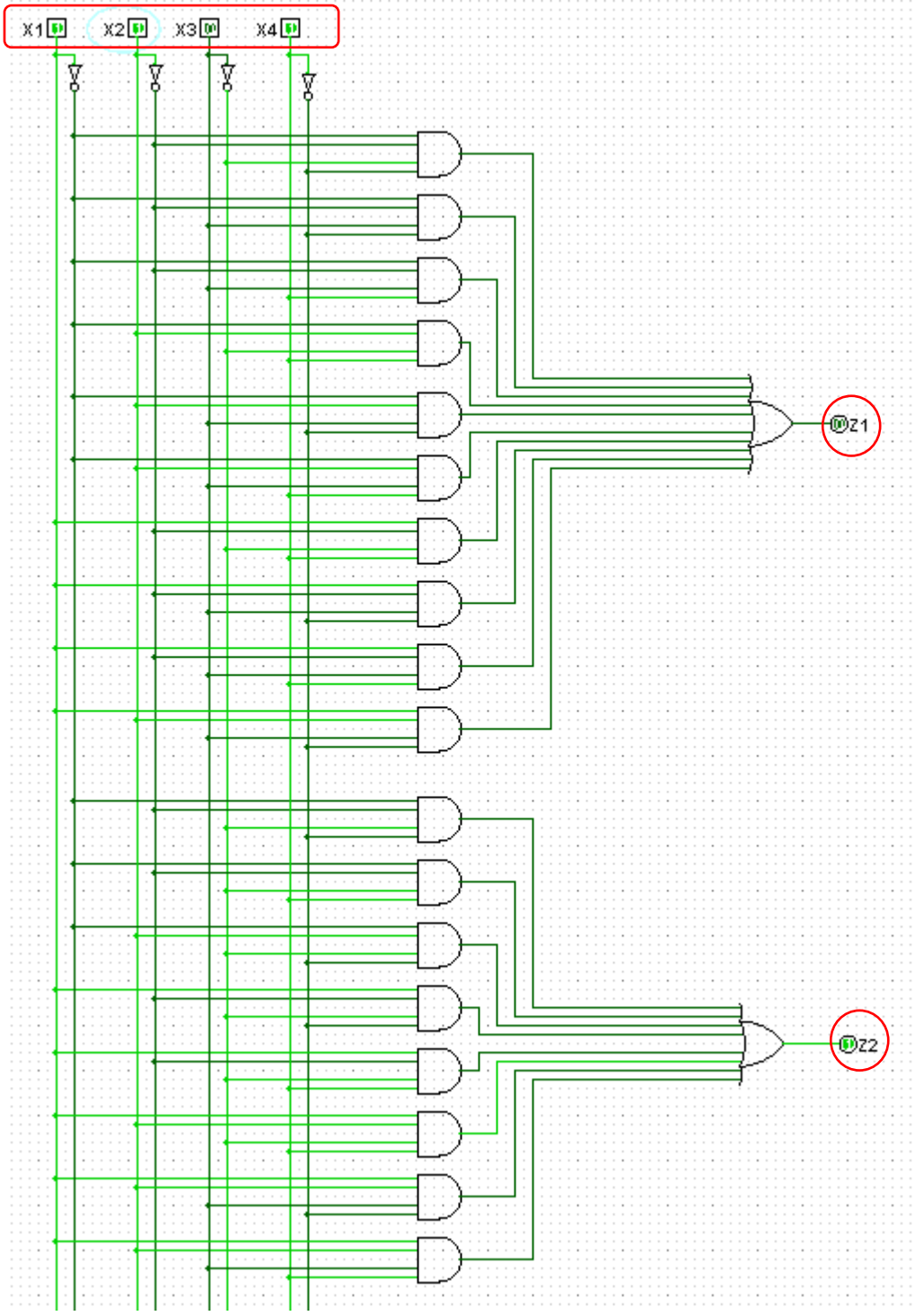
Test case 2:

	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

$$Z1 = X1 \cdot \overline{X2} \cdot X4$$

Z1	X3X4: 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	Blue	$Z1 = \overline{X1} \cdot \overline{X2} \cdot \overline{X4}$
2	Yellow	$Z1 = X3 \cdot \overline{X4}$
3	Purple	$Z1 = \overline{X1} \cdot X2 \cdot X4$
4	Red	$Z1 = X1 \cdot \overline{X2} \cdot X4$
5	Green	$Z1 = \overline{X1} \cdot X3$

Table 4.1

Z2	X3X4: 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

No.	Colour	Equation
1	Yellow	$Z2 = \overline{X1} \cdot \overline{X3} \cdot \overline{X4}$
2	Purple	$Z2 = \overline{X1} \cdot \overline{X2} \cdot \overline{X3}$
3	Red	$Z2 = X1 \cdot \overline{X2} \cdot \overline{X3}$
4	Green	$Z2 = X1 \cdot \overline{X3} \cdot X4$
5	Blue	$Z2 = X1 \cdot X2 \cdot X3$

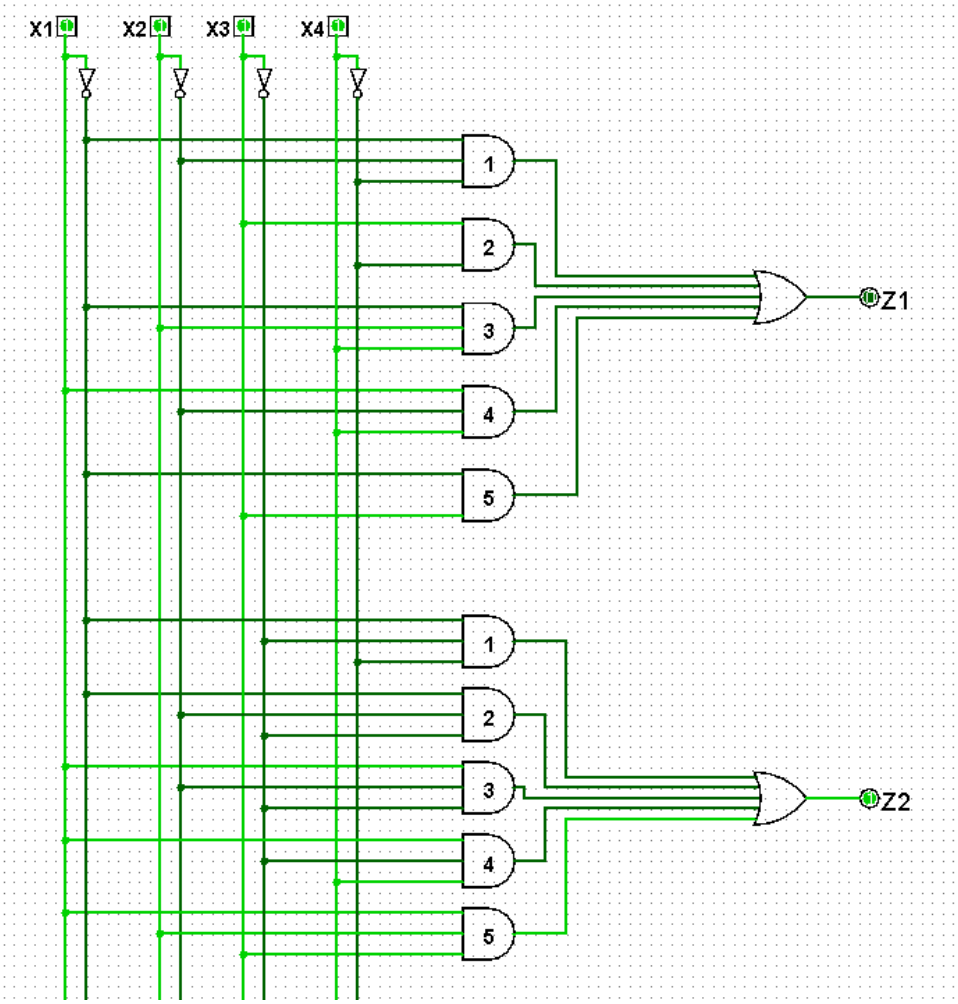
Table 5.1

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

$$Z1 = (\overline{X1} \cdot \overline{X2} \cdot \overline{X4}) + (X3 \cdot \overline{X4}) + (\overline{X1} \cdot X2 \cdot X4) + (X1 \cdot \overline{X2} \cdot X4) + (\overline{X1} \cdot X3)$$

$$Z2 = (\overline{X1} \cdot \overline{X3} \cdot \overline{X4}) + (\overline{X1} \cdot \overline{X2} \cdot \overline{X3}) + (X1 \cdot \overline{X2} \cdot \overline{X3}) + (X1 \cdot \overline{X3} \cdot X4) + (X1 \cdot X2 \cdot X3)$$

A simplified circuit is created in Logisim using AND gates for $' \cdot '$, OR gates for '+' and NOT gate for $\bar{X}$ (refer to Circuit 5).

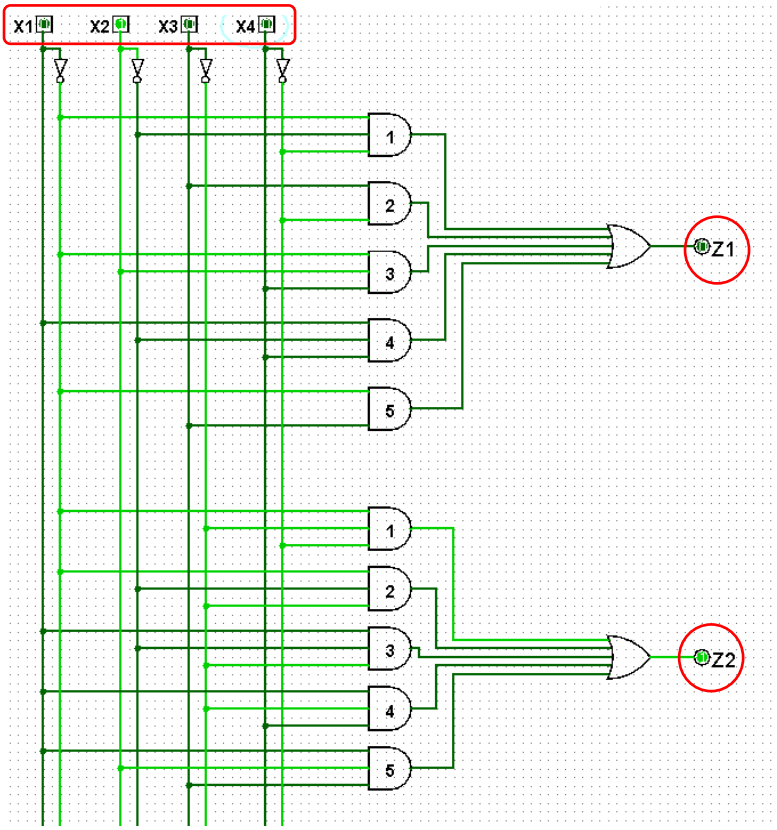


Circuit 5

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.

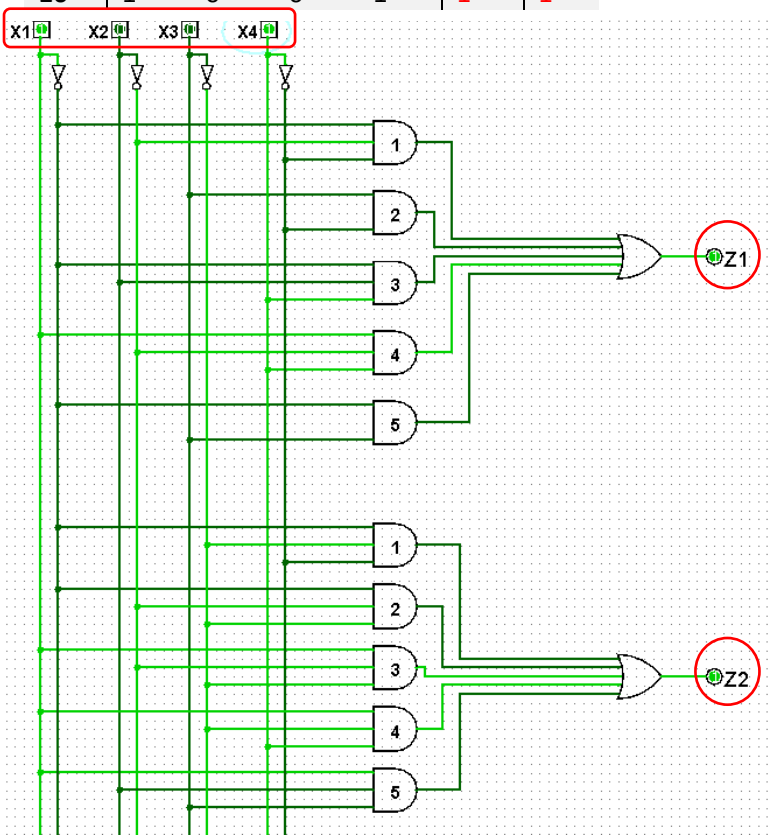
Test case 1:

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



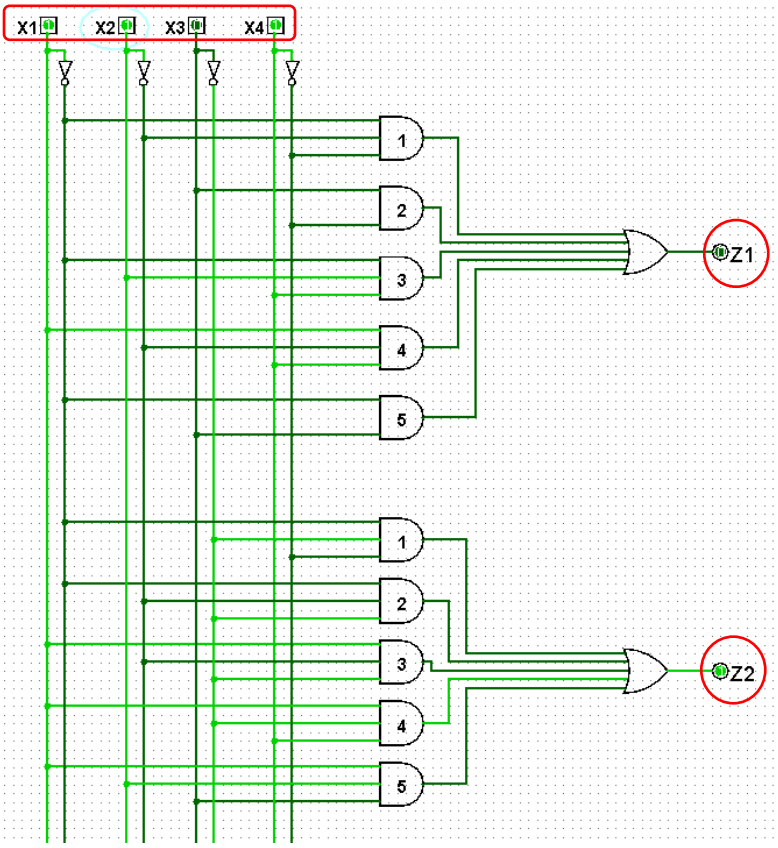
Test case 2:

	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

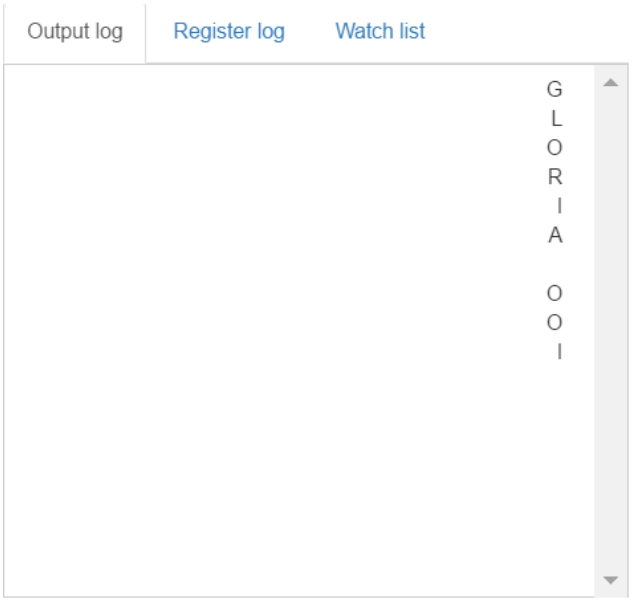
This task is to create a MARIE file that stores my name in memory. Below is a snapshot of the memory after the code is assembled with my name 'GLORIA(space)OOI'.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	0047	004C	004F	0052	0049	0041	0020	004F	004F	0049	0000	0000	0000	0000	0000	0000
010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
020	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

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.

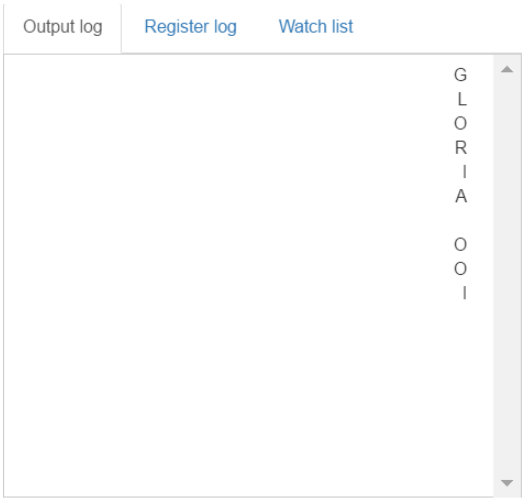
Below is the output:



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:



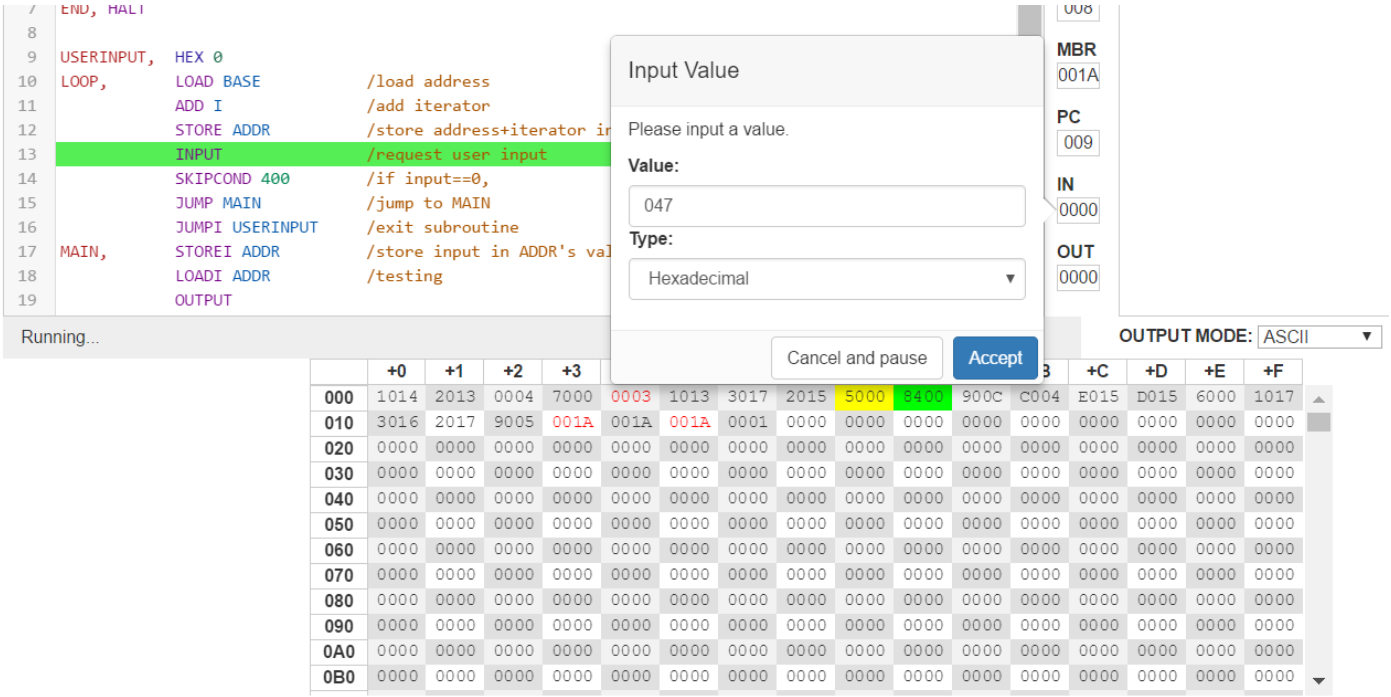
Task 2.1.4 – User Input

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:

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1014	2013	0004	7000	0000	1013	3017	2015	5000	8400	900C	C004	E015	D015	6000	1017
010	3016	2017	9005	0000	001A	0000	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000
020	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Test case 1:

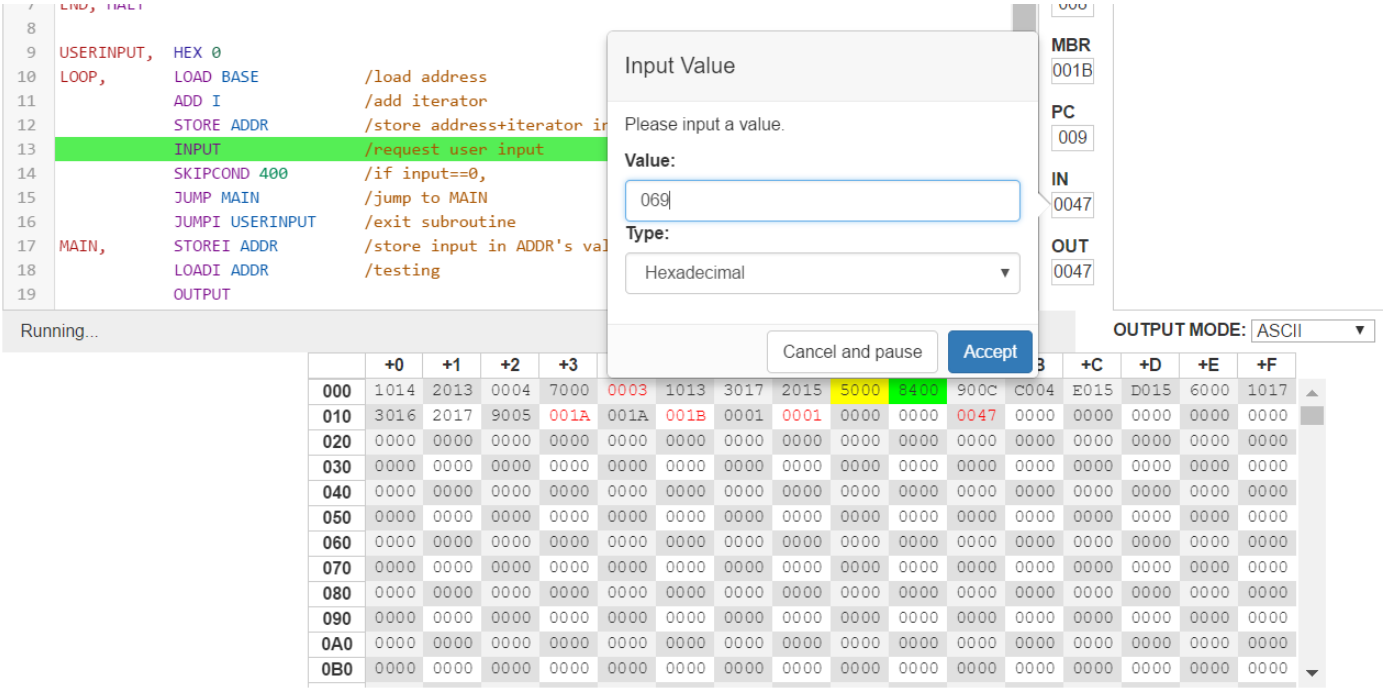


	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1014	2013	0004	7000	0003	1013	3017	2015	5000	8400	900C	C004	E015	D015	6000	1017
010	3016	2017	9005	001A	001A	001A	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000
020	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Memory after input 1:

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1014	2013	0004	7000	0003	1013	3017	2015	5000	8400	900C	C004	E015	D015	6000	1017
010	3016	2017	9005	001A	001A	001B	0001	0001	0000	0000	0047	0000	0000	0000	0000	0000
020	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Test case 2:



	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1014	2013	0004	7000	0003	1013	3017	2015	5000	8400	900C	C004	E015	D015	6000	1017
010	3016	2017	9005	001A	001A	001B	0001	0001	0000	0000	0047	0000	0000	0000	0000	0000
020	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Memory after input 2:

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1014	2013	0004	7000	0003	1013	3017	2015	5000	8400	900C	C004	E015	D015	6000	1017
010	3016	2017	9005	001A	001A	001C	0001	0002	0000	0000	0047	0069	0000	0000	0000	0000
020	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

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.

Assembly code: Autosaved file

```
39  SIXFOUR,                DEC 64
40  THREETWO,               DEC 32
41  NINEONE,                DEC 91
42  ADDR,                   HEX 0
43  ONE,                    HEX 1
44  NUM,                    DEC 43
45  PrintString,            HEX 0   /argument: address of string
46  I,                      HEX 0   /iterator
47  GLORIA_OOI,             HEX 047 /G
48                          HEX 04C /L
49                          HEX 04F /O
50                          HEX 052 /R
51                          HEX 049 /I
52                          HEX 041 /A
53                          HEX 020 /(space)
54                          HEX 04F /O
55                          HEX 04F /O
56                          HEX 049 /I
57                          HEX 000 /end
```

Machine halted normally.

Output log | Register log | Watch list

```
g
l
o
r
i
a

o
o
i
```

OUTPUT MODE: ASCII

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’.

Assembly code: Modified file

```
40  THREETWO,               DEC 32
41  NINEONE,                DEC 91
42  ADDR,                   HEX 0
43  ONE,                    HEX 1
44  NUM,                    DEC 43
45  PrintString,            HEX 0   /argument: address of string
46  I,                      HEX 0   /iterator
47  STRING,             HEX 048 /H
48                          HEX 065 /e
49                          HEX 04C /L
50                          HEX 06C /l
51                          HEX 04F /O
52                          HEX 020 /(space)
53                          HEX 057 /W
54                          HEX 06F /o
55                          HEX 052 /R
56                          HEX 06C /l
57                          HEX 044 /D
58                          HEX 000 /end
```

MARIE.js

Output log | Register log | Watch list

```
h
e
l
l
o

w
o
r
l
d
```

OUTPUT MODE: ASCII ▼

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’.

Assembly code: Autosaved file

```
46  NINESIX,                DEC 96
47  ONETWOTWO,              DEC 122
48  ONETWOTHREE,            DEC 123
49  ADDR,                   HEX 0
50  ONE,                    HEX 1
51  NUM,                    DEC 50
52  PrintString,            HEX 0   /argument: address of string
53  I,                      HEX 0   /iterator
54  GLORIA_OOI,             HEX 067 /g
55                          HEX 06C /l
56                          HEX 06F /o
57                          HEX 072 /r
58                          HEX 069 /i
59                          HEX 061 /a
60                          HEX 020 /(space)
61                          HEX 06F /o
62                          HEX 06F /o
63                          HEX 069 /i
64                          HEX 000 /end
```

Output log | Register log | Watch list

```
t
y
b
e
v
n

b
b
v
```

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.

Assembly code: Modified file

```
40          ADD ONE                     /I+1
41          STORE I                     /I = I=1
42          JUMP LOOP
43  CURRENTVALUE,           HEX 0
44  ONETHREE,               DEC 13
45  SIXFOUR,                DEC 64
46  NINESIX,                DEC 96
47  ONETWOTWO,              DEC 122
48  ONETWOTHREE,            DEC 123
49  ADDR,                   HEX 0
50  ONE,                    HEX 1
51  NUM,                    DEC 50
52  PrintString,            HEX 0   /argument: address of string
53  I,                      HEX 0   /iterator
54  STRING,                 HEX 061 /a
55                          HEX 070 /p
56                          HEX 02A /*
57                          HEX 047 /G
58                          HEX 000 /end
```

Machine halted normally.

Output log | Register log | Watch list

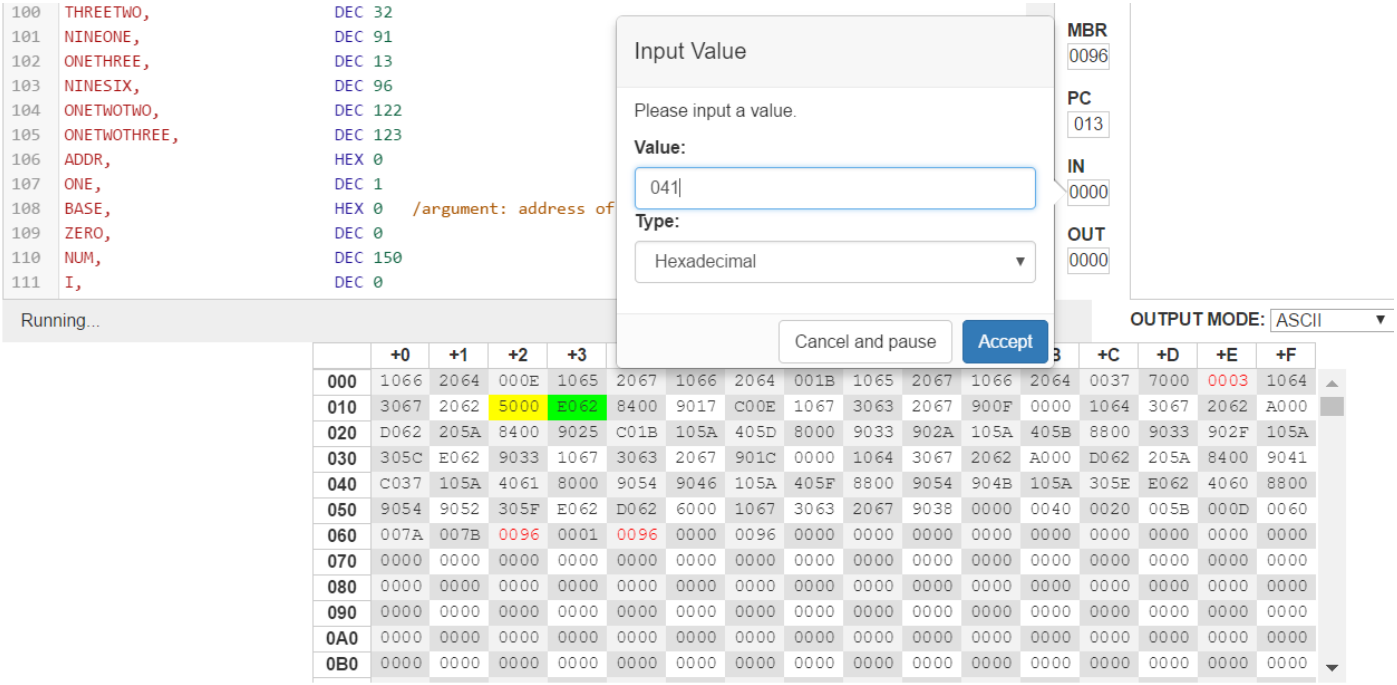
```
n
c
*
G
```

OUTPUT MODE: ASCII ▼

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.



	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1066	2064	000E	1065	2067	1066	2064	001B	1065	2067	1066	2064	0037	7000	0003	1064
010	3067	2062	5000	E062	8400	9017	C00E	1067	3063	2067	900F	0000	1064	3067	2062	A000
020	D062	205A	8400	9025	C01B	105A	405D	8000	9033	902A	105A	405B	8800	9033	902F	105A
030	305C	E062	9033	1067	3063	2067	901C	0000	1064	3067	2062	A000	D062	205A	8400	9041
040	C037	105A	4061	8000	9054	9046	105A	405F	8800	9054	904B	105A	305E	E062	4060	8800
050	9054	9052	305F	E062	D062	6000	1067	3063	2067	9038	0000	0040	0020	005B	000D	0060
060	007A	007B	0096	0001	0096	0000	0096	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

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

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1066	2064	000E	1065	2067	1066	2064	001B	1065	2067	1066	2064	0037	7000	0003	1064
010	3067	2062	5000	E062	8400	9017	C00E	1067	3063	2067	900F	0000	1064	3067	2062	A000
020	D062	205A	8400	9025	C01B	105A	405D	8000	9033	902A	105A	405B	8800	9033	902F	105A
030	305C	E062	9033	1067	3063	2067	901C	0000	1064	3067	2062	A000	D062	205A	8400	9041
040	C037	105A	4061	8000	9054	9046	105A	405F	8800	9054	904B	105A	305E	E062	4060	8800
050	9054	9052	305F	E062	D062	6000	1067	3063	2067	9038	0000	0040	0020	005B	000D	0060
060	007A	007B	0099	0001	0096	0000	0096	0003	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0041	005C	0070	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

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

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	1066	2064	000E	1065	2067	1066	2064	001B	1065	2067	1066	2064	0037	7000	0003	1064
010	3067	2062	5000	E062	8400	9017	C00E	1067	3063	2067	900F	0008	1064	3067	2062	A000
020	D062	205A	8400	9025	C01B	105A	405D	8000	9033	902A	105A	405B	8800	9033	902F	105A
030	305C	E062	9033	1067	3063	2067	901C	000D	1064	3067	2062	A000	D062	205A	8400	9041
040	C037	105A	4061	8000	9054	9046	105A	405F	8800	9054	904B	105A	305E	E062	4060	8800
050	9054	9052	305F	E062	D062	6000	1067	3063	2067	9038	0000	0040	0020	005B	000D	0060
060	007A	007B	0099	0001	0096	0000	0096	0003	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	006E	005C	0063	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0B0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

This is the output: