C Programming & Imbedded Systems

Student Name: Benjamin Samuel Edwards

Student ID: 18026826

Cover picture used, [3], found in references

Contents: -

Index: -

Fig (1.1)

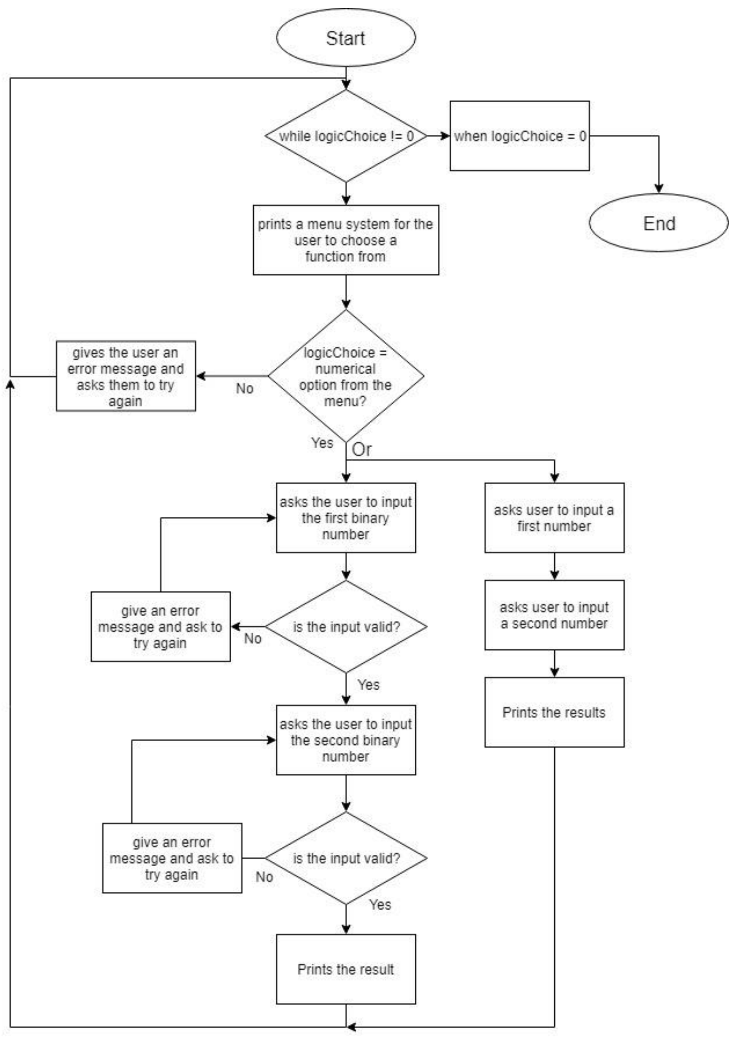

Start
while logicChoice != 0
when logicChoice = 0
End
prints a menu system for the user to choose a function from
logicChoice = numerical option from the menu?
No
gives the user an error message and asks them to try again
Yes
Or
asks the user to input the first binary number
is the input valid?
No
give an error message and ask to try again
Yes
asks the user to input the second binary number
is the input valid?
No
give an error message and ask to try again
Yes
Prints the result
asks user to input a first number
asks user to input a second number
Prints the results

Fig (1.2)

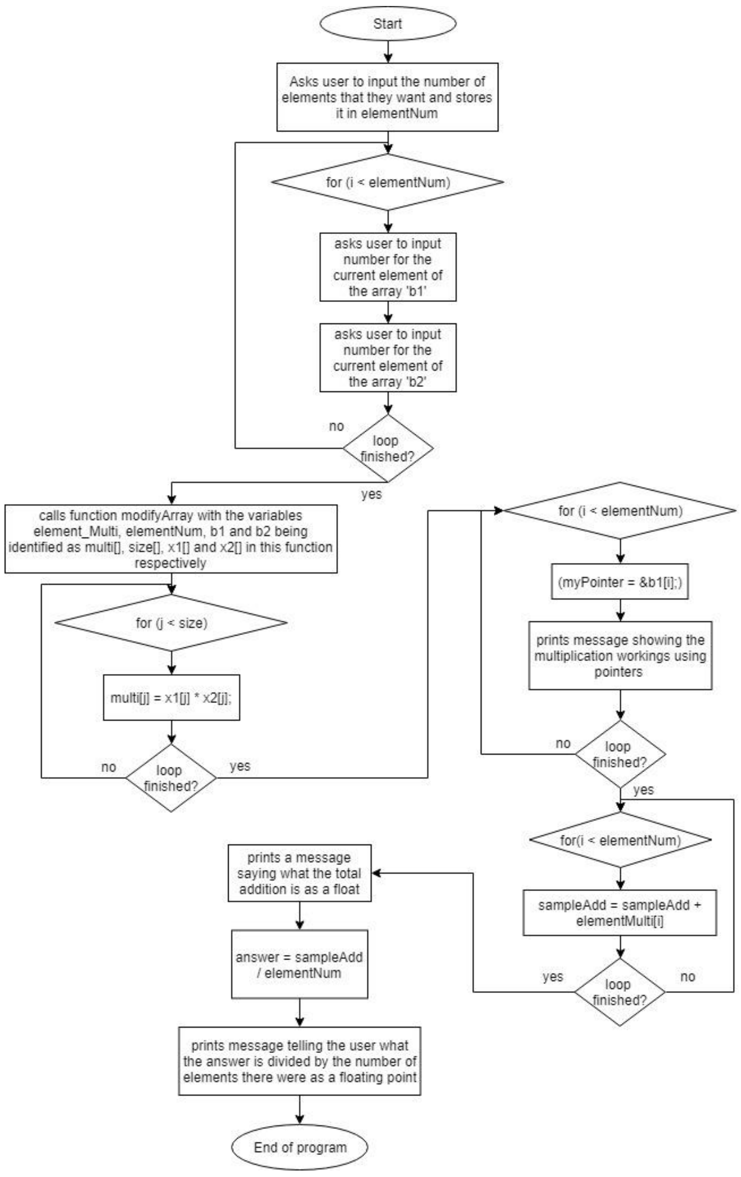

Start
Asks user to input the number of elements that they want and stores it in elementNum
for (i < elementNum)
asks user to input number for the current element of the array 'b1'
asks user to input number for the current element of the array 'b2'
loop finished?
no
yes
calls function modifyArray with the variables element_Multi, elementNum, b1 and b2 being identified as multi[], size[], x1[] and x2[] in this function respectively
for (j < size)
multi[j] = x1[j] * x2[j];
loop finished?
no
yes
for (i < elementNum)
(myPointer = &b1[i];)
prints message showing the multiplication workings using pointers
loop finished?
no
yes
for(i < elementNum)
sampleAdd = sampleAdd + elementMulti[i]
loop finished?
yes
no
prints a message saying what the total addition is as a float
answer = sampleAdd / elementNum
prints message telling the user what the answer is divided by the number of elements there were as a floating point
End of program

Fig (1.3)

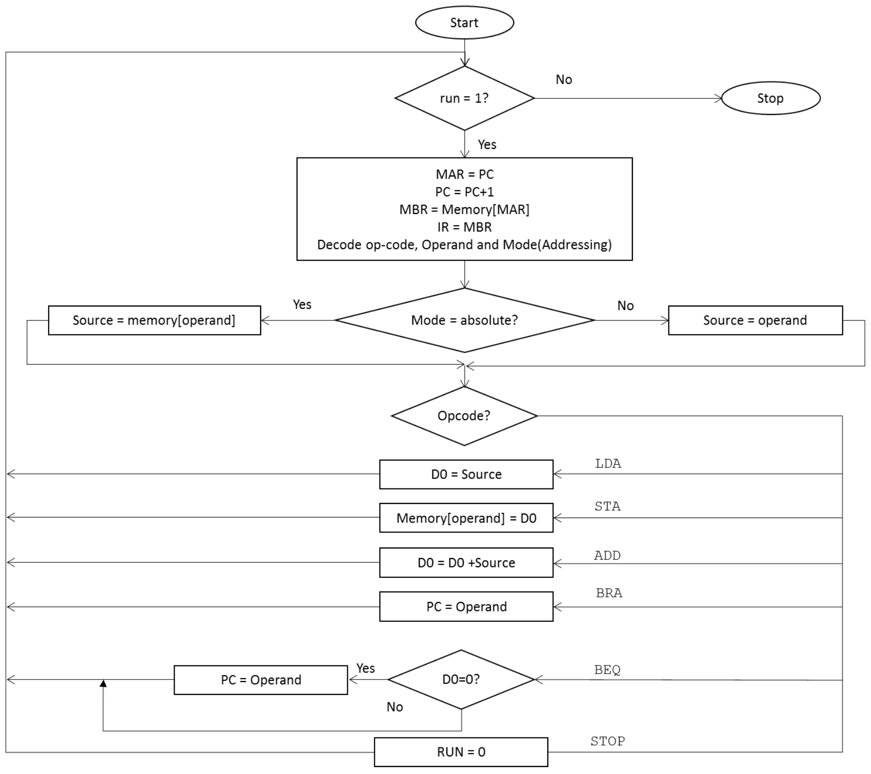

Start
run = 1?
No
Stop
Yes
MAR = PC
PC = PC+1
MBR = Memory[MAR]
IR = MBR
Decode op-code, Operand and Mode(Addressing)
Mode = absolute?
Yes
Source = memory[operand]
No
Source = operand
Opcode?
D0 = Source
LDA
Memory[operand] = D0
STA
D0 = D0 +Source
ADD
PC = Operand
BRA
D0=0?
BEQ
Yes
PC = Operand
No
RUN = 0
STOP

Fig (1.4)

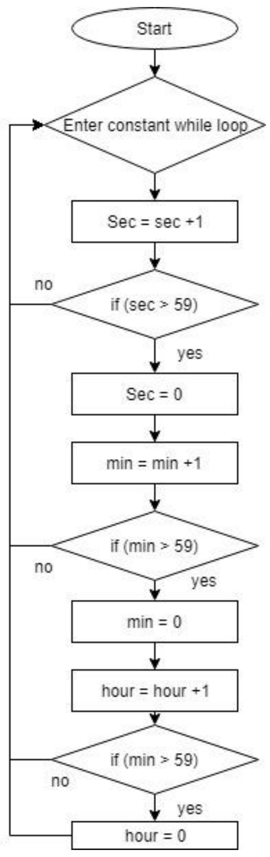
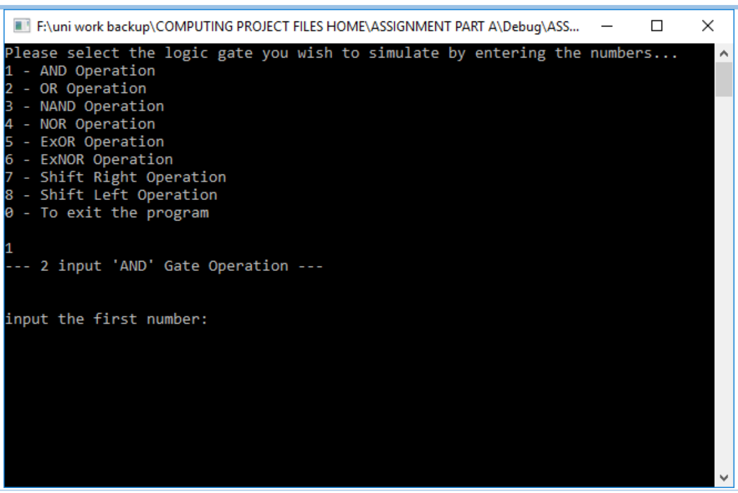
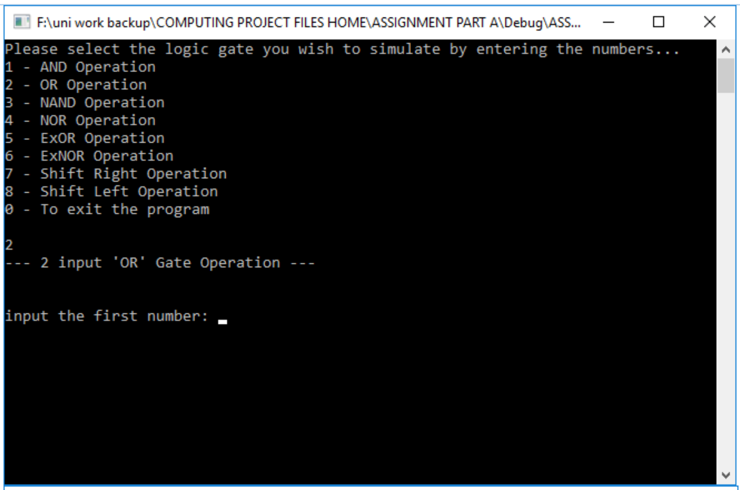


Table of Testing

Logic Functions: -

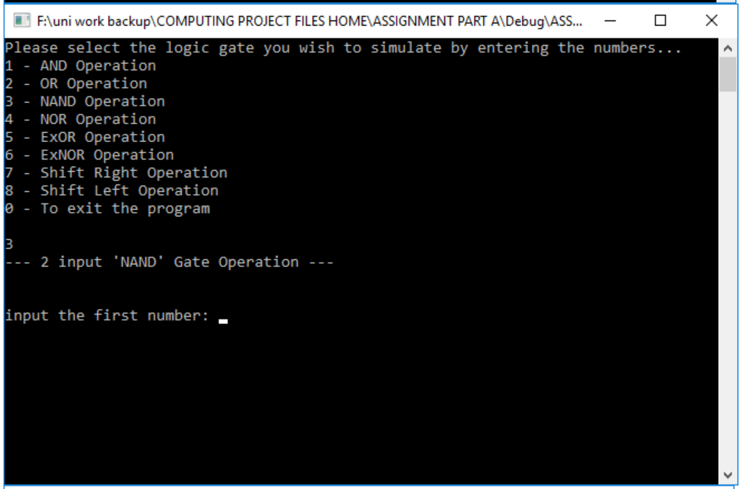
Method	Results
Selecting each Logic Gate option from the main menu.	 F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS... ```Please select the logic gate you wish to simulate by entering the numbers... 1 - AND Operation 2 - OR Operation 3 - NAND Operation 4 - NOR Operation 5 - ExOR Operation 6 - ExNOR Operation 7 - Shift Right Operation 8 - Shift Left Operation 0 - To exit the program 1 --- 2 input 'AND' Gate Operation --- input the first number:```

```
Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

1
--- 2 input 'AND' Gate Operation ---


input the first number:
```



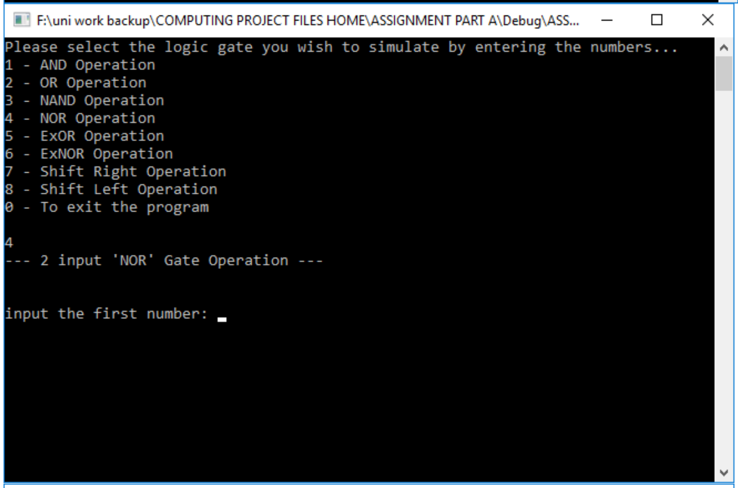
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

2
--- 2 input 'OR' Gate Operation ---


input the first number: _
```



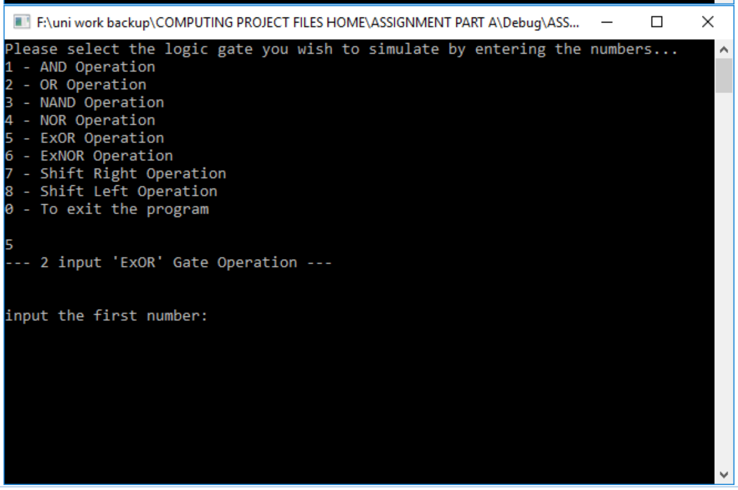
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

3
--- 2 input 'NAND' Gate Operation ---


input the first number: _
```



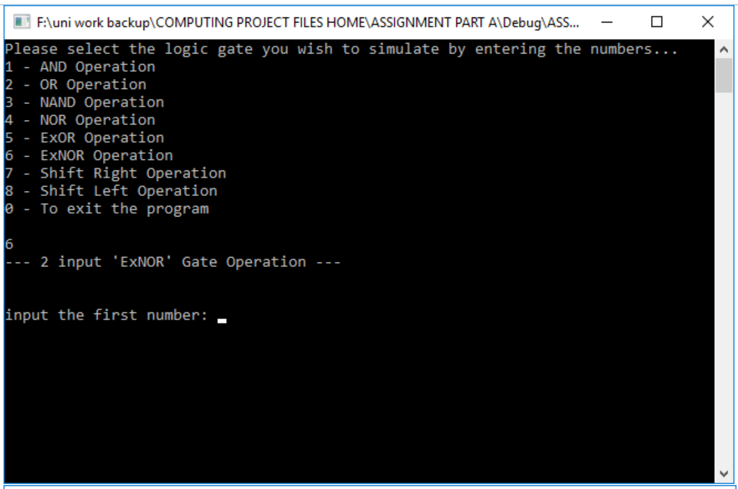
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

4
--- 2 input 'NOR' Gate Operation ---


input the first number: _
```



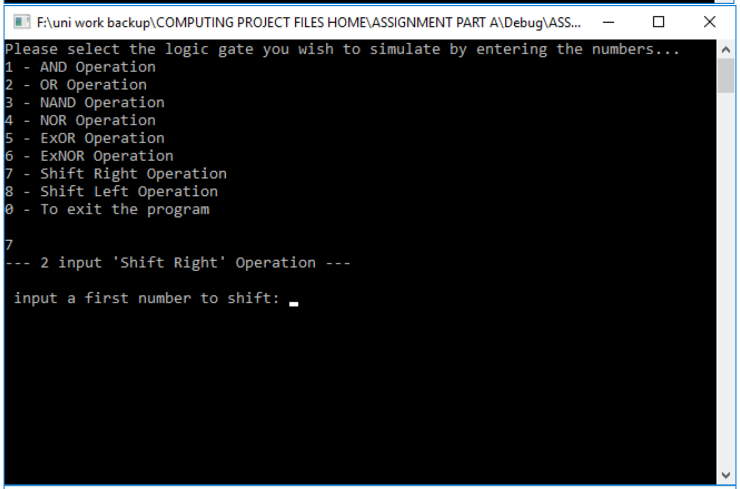
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

5
--- 2 input 'ExOR' Gate Operation ---


input the first number:
```



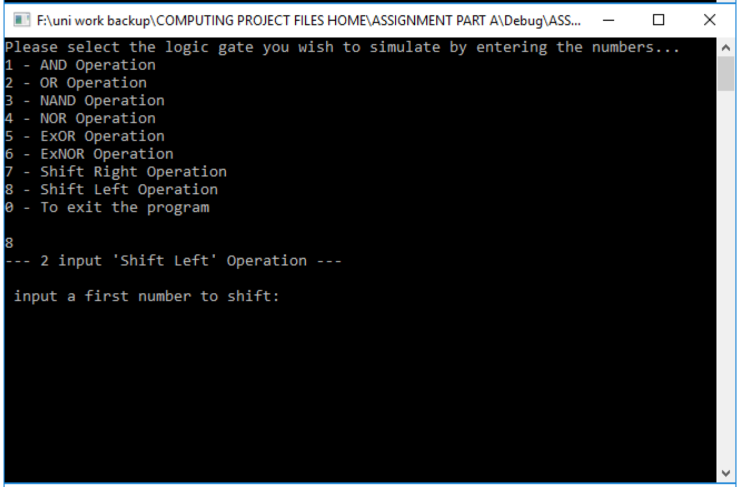
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

6
--- 2 input 'ExNOR' Gate Operation ---


input the first number: _
```



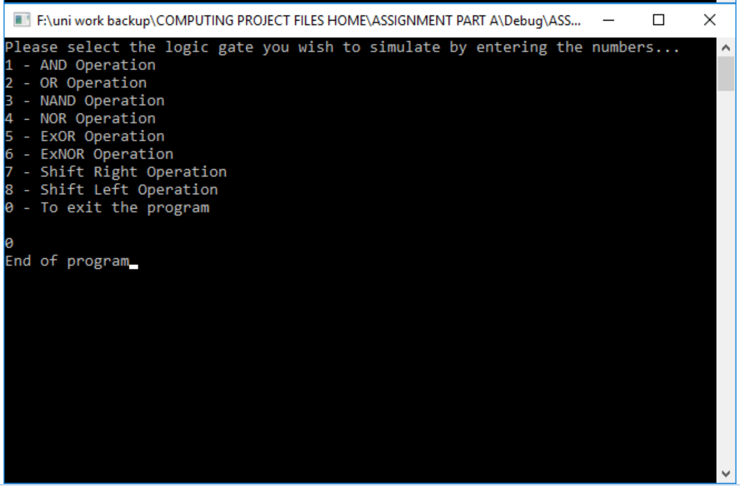
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

7
--- 2 input 'Shift Right' Operation ---

 input a first number to shift: _
```



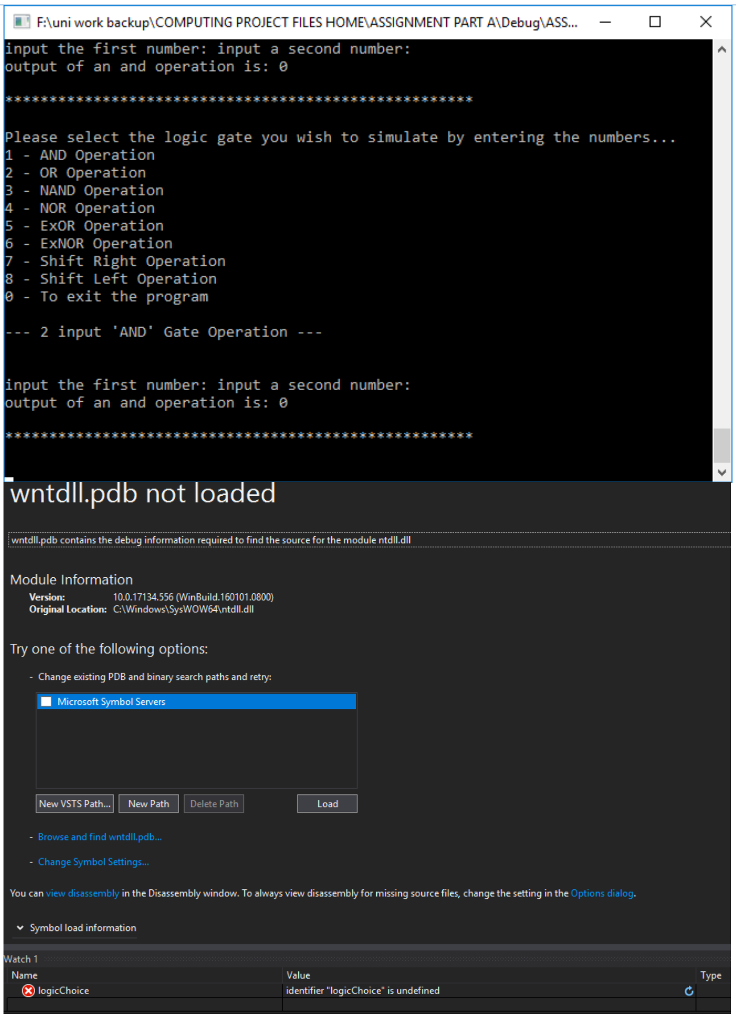
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

8
--- 2 input 'Shift Left' Operation ---

 input a first number to shift:
```



F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

0
End of program_
```

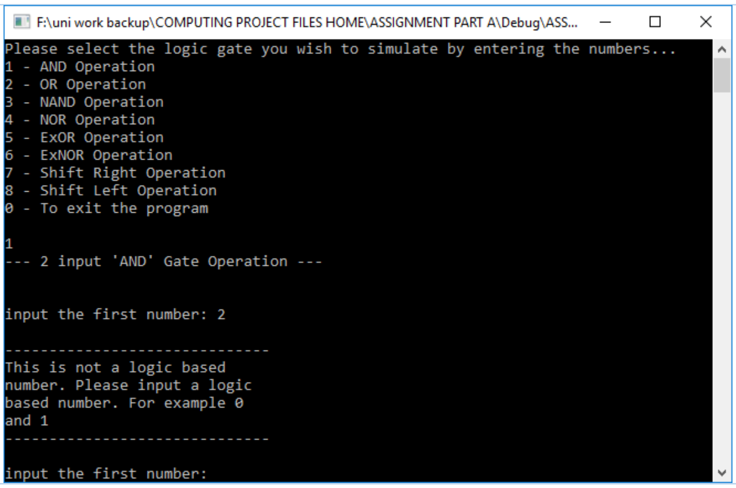


```
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...
input the first number: input a second number:
output of an and operation is: 0

*****************************************************

Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

--- 2 input 'AND' Gate Operation ---


input the first number: input a second number:
output of an and operation is: 0

*****************************************************
```

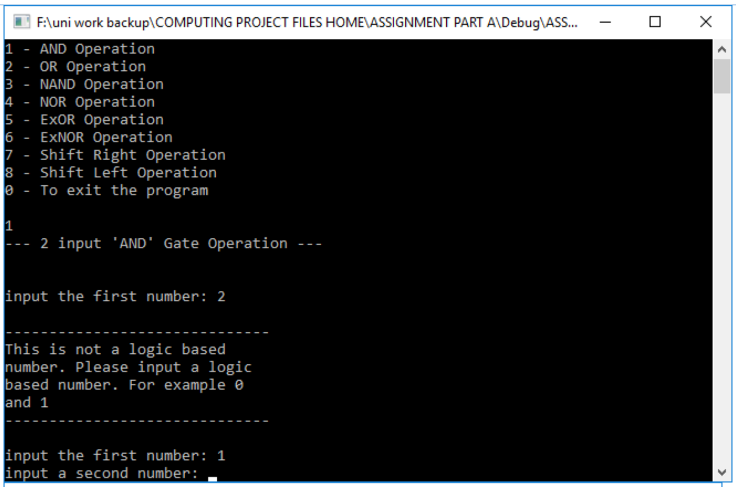
```
wntdll.pdb not loaded

wntdll.pdb contains the debug information required to find the source for the module ntdll.dll

Module Information
    Version:           10.0.17134.556 (WinBuild.160101.0800)
    Original Location: C:\Windows\SysWOW64\ntdll.dll

Try one of the following options:

  - Change existing PDB and binary search paths and retry:
      Microsoft Symbol Servers
  - Browse and find wntdll.pdb...
  - Change Symbol Settings...

You can view disassembly in the Disassembly window. To always view disassembly for missing source files, change the setting in the Options dialog.

Symbol load information

Watch 1
Name           Value                                     Type
logicChoice    identifier "logicChoice" is undefined
```

//program glitches and gives an error saying logicChoice is undefined so will not accept floating point numbers or characters

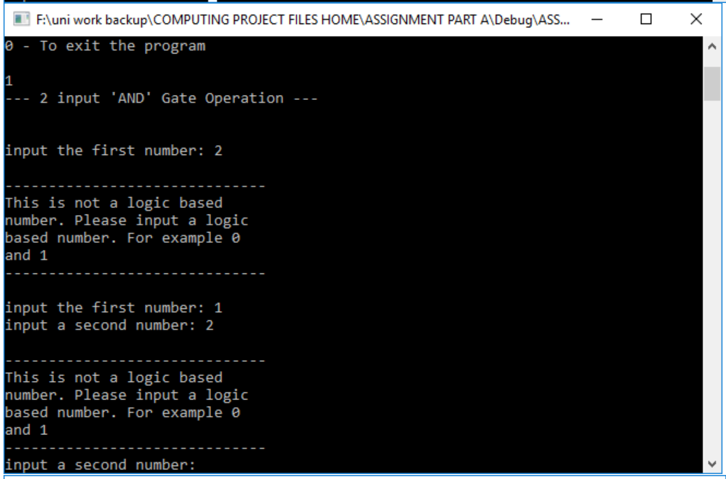
Testing the input check code for all Gates.



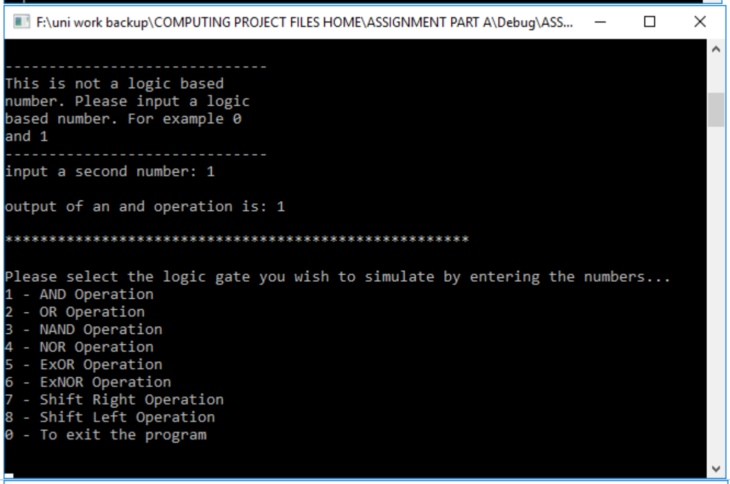
```
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...
Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

1
--- 2 input 'AND' Gate Operation ---


input the first number: 2

------------------------------
This is not a logic based
number. Please input a logic
based number. For example 0
and 1
------------------------------

input the first number:
```



```
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

1
--- 2 input 'AND' Gate Operation ---


input the first number: 2

------------------------------
This is not a logic based
number. Please input a logic
based number. For example 0
and 1
------------------------------

input the first number: 1
input a second number: _
```

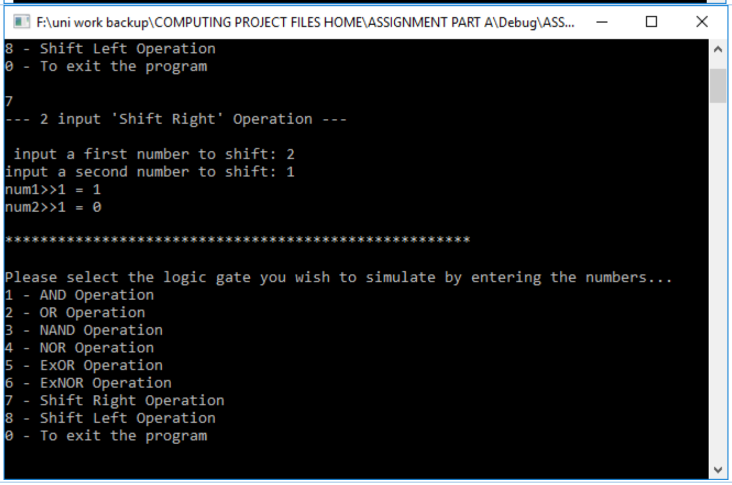


```
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...
0 - To exit the program

1
--- 2 input 'AND' Gate Operation ---


input the first number: 2

------------------------------
This is not a logic based
number. Please input a logic
based number. For example 0
and 1
------------------------------

input the first number: 1
input a second number: 2

------------------------------
This is not a logic based
number. Please input a logic
based number. For example 0
and 1
------------------------------
input a second number:
```

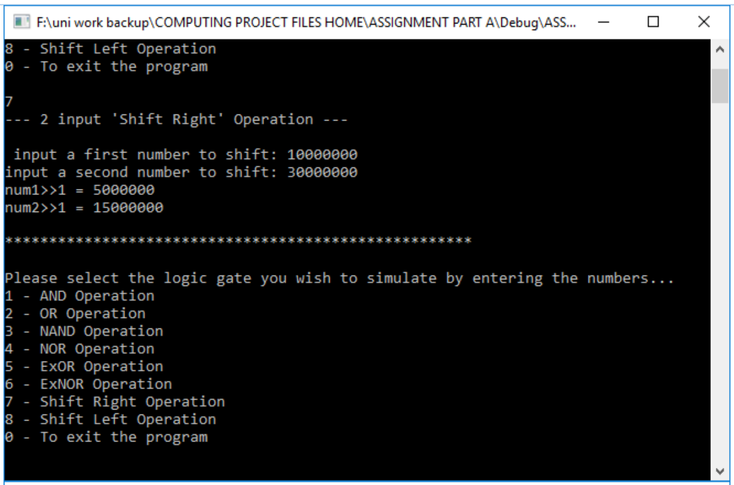


```
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

------------------------------
This is not a logic based
number. Please input a logic
based number. For example 0
and 1
------------------------------
input a second number: 1

output of an and operation is: 1

*****************************************************

Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program
```

Testing the Shift Left and Right operation using small numbers, large numbers and float numbers.

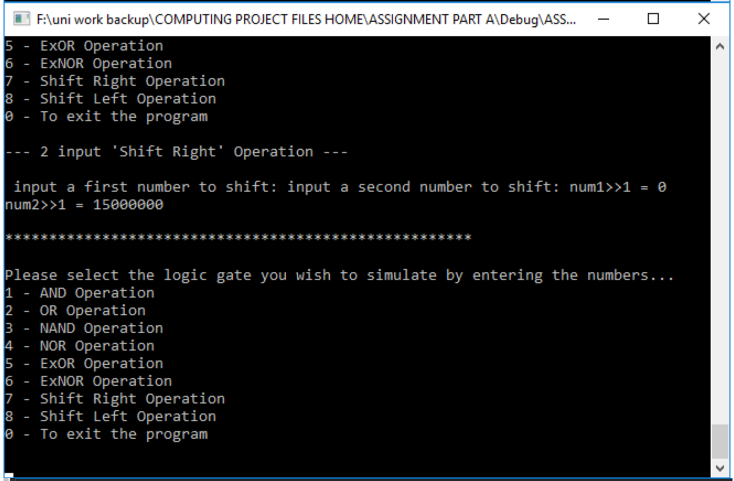


```
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...
8 - Shift Left Operation
0 - To exit the program

7
--- 2 input 'Shift Right' Operation ---

 input a first number to shift: 2
input a second number to shift: 1
num1>>1 = 1
num2>>1 = 0

*****************************************************

Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program
```



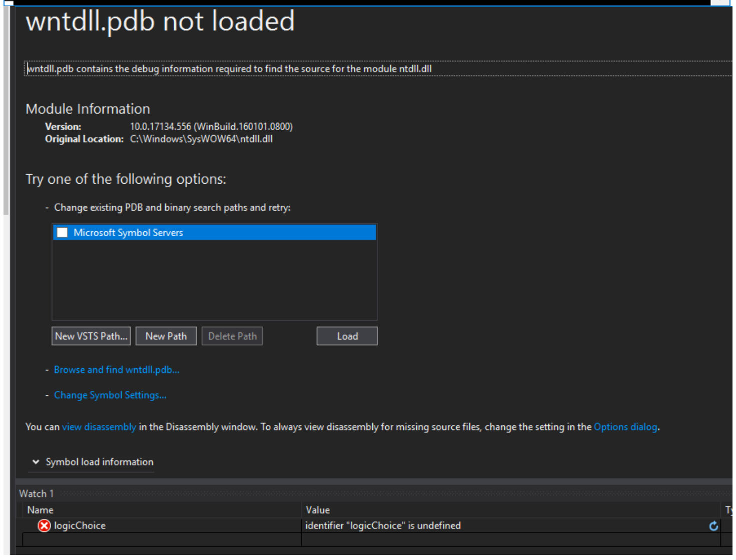
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
8 - Shift Left Operation
0 - To exit the program

7
--- 2 input 'Shift Right' Operation ---

 input a first number to shift: 10000000
input a second number to shift: 30000000
num1>>1 = 5000000
num2>>1 = 15000000

*****************************************************

Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program
```



F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

--- 2 input 'Shift Right' Operation ---

 input a first number to shift: input a second number to shift: num1>>1 = 0
num2>>1 = 15000000

*****************************************************

Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program
```



wntdll.pdb not loaded

wntdll.pdb contains the debug information required to find the source for the module ntdll.dll

Module Information
Version: 10.0.17134.556 (WinBuild.160101.0800)
Original Location: C:\Windows\SysWOW64\ntdll.dll

Try one of the following options:

- Change existing PDB and binary search paths and retry:

- Browse and find wntdll.pdb...
- Change Symbol Settings...

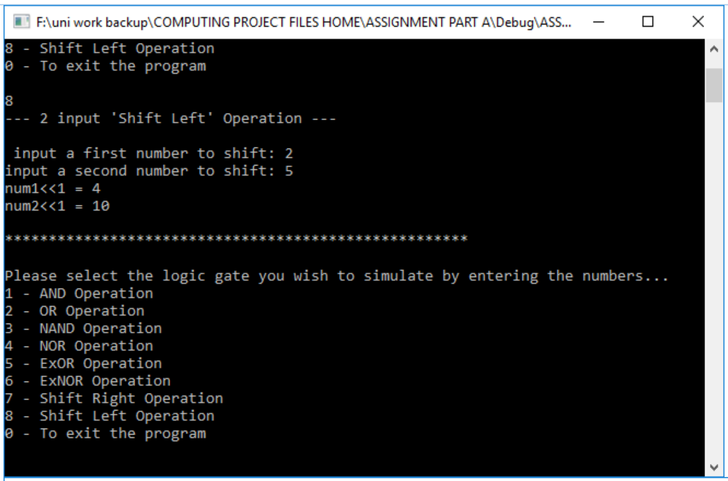
You can view disassembly in the Disassembly window. To always view disassembly for missing source files, change the setting in the Options dialog.

Symbol load information

Watch 1

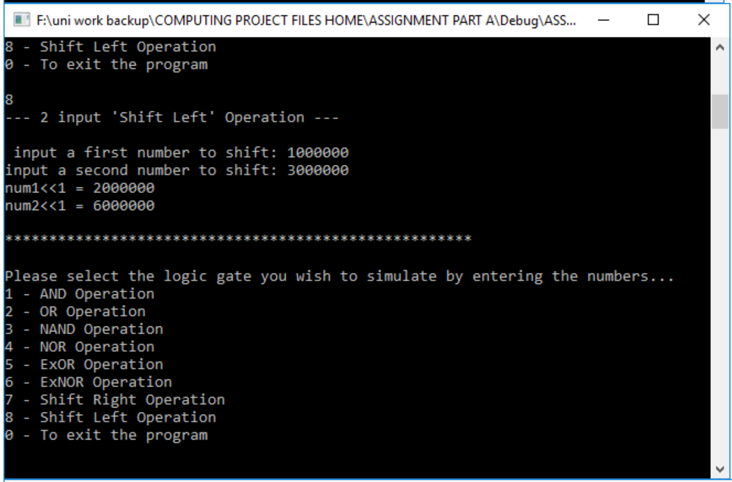
Name	Value
logicChoice	identifier "logicChoice" is undefined

//program glitches and gives an error saying logicChoice is undefined so will not accept floating point numbers



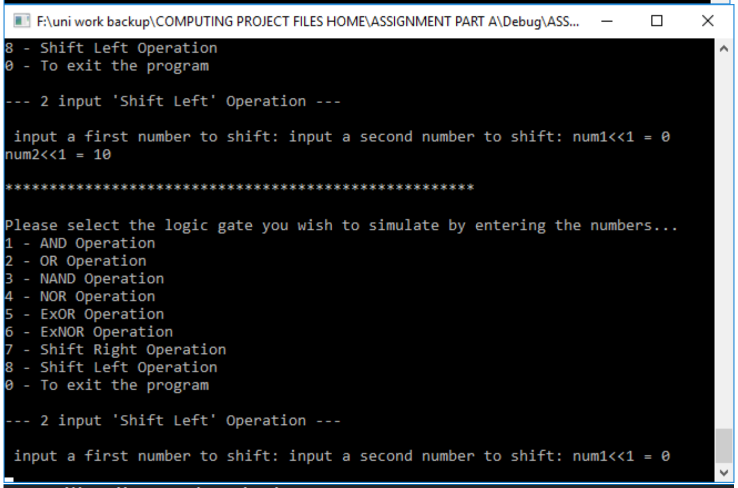
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
8 - Shift Left Operation
0 - To exit the program

8
--- 2 input 'Shift Left' Operation ---

 input a first number to shift: 2
input a second number to shift: 5
num1<<1 = 4
num2<<1 = 10

*****************************************************

Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program
```



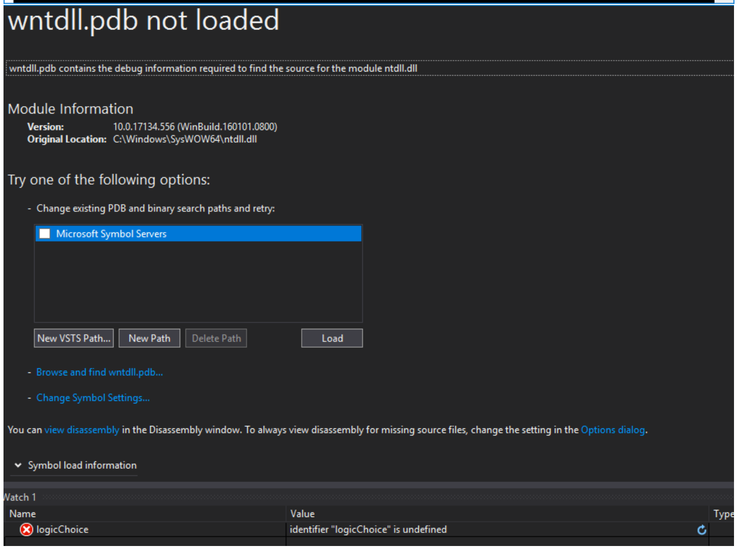
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
8 - Shift Left Operation
0 - To exit the program

8
--- 2 input 'Shift Left' Operation ---

 input a first number to shift: 1000000
input a second number to shift: 3000000
num1<<1 = 2000000
num2<<1 = 6000000

*****************************************************

Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program
```



F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART A\Debug\ASS...

```
8 - Shift Left Operation
0 - To exit the program

--- 2 input 'Shift Left' Operation ---

 input a first number to shift: input a second number to shift: num1<<1 = 0
num2<<1 = 10

*****************************************************

Please select the logic gate you wish to simulate by entering the numbers...
1 - AND Operation
2 - OR Operation
3 - NAND Operation
4 - NOR Operation
5 - ExOR Operation
6 - ExNOR Operation
7 - Shift Right Operation
8 - Shift Left Operation
0 - To exit the program

--- 2 input 'Shift Left' Operation ---

 input a first number to shift: input a second number to shift: num1<<1 = 0
```



wntdll.pdb not loaded

wntdll.pdb contains the debug information required to find the source for the module ntdll.dll

Module Information

Version: 10.0.17134.556 (WinBuild.160101.0800)
Original Location: C:\Windows\SysWOW64\ntdll.dll

Try one of the following options:

- Change existing PDB and binary search paths and retry:

- Browse and find wntdll.pdb...
- Change Symbol Settings...

You can view disassembly in the Disassembly window. To always view disassembly for missing source files, change the setting in the Options dialog.

Symbol load information

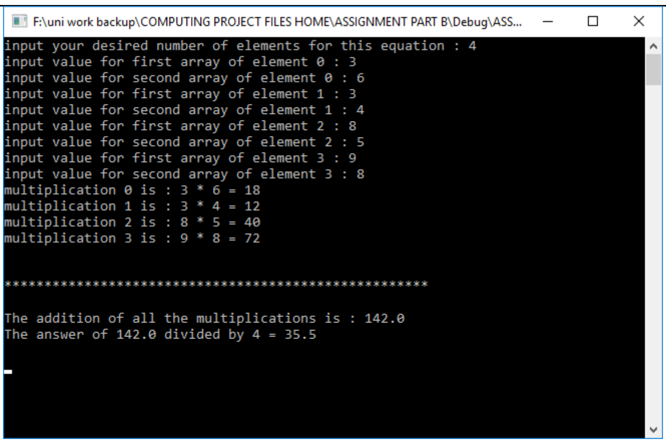
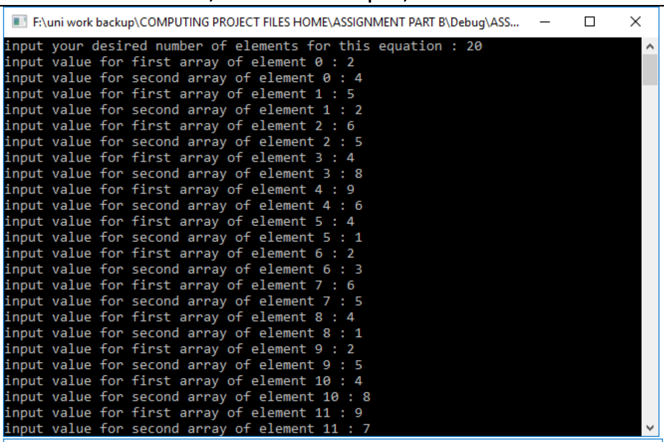
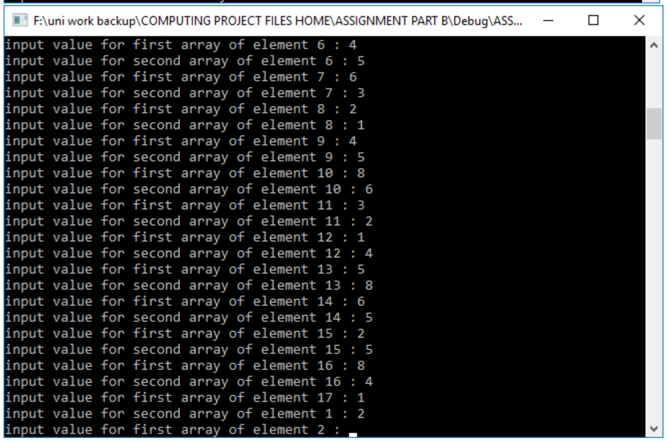
Watch 1

Name	Value	Type
logicChoice	identifier "logicChoice" is undefined	

//program glitches and gives an error saying logicChoice is

	undefined so will not accept floating point numbers

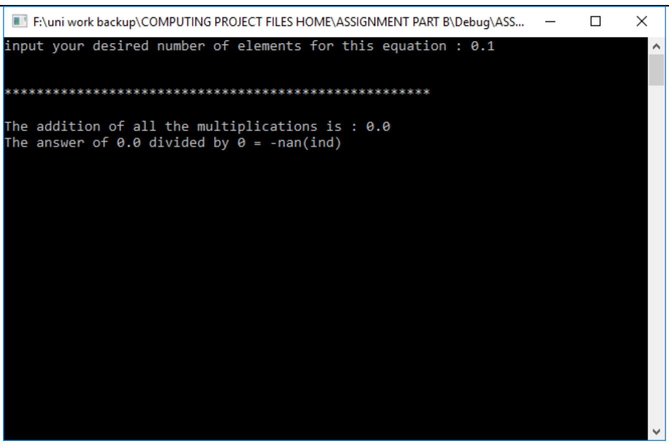
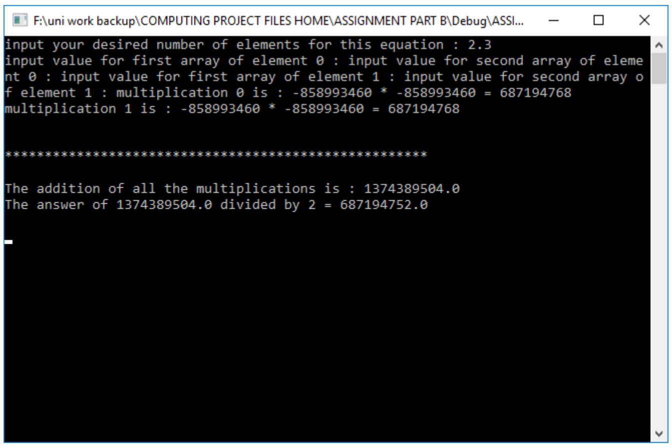
Mathematical Equations: -

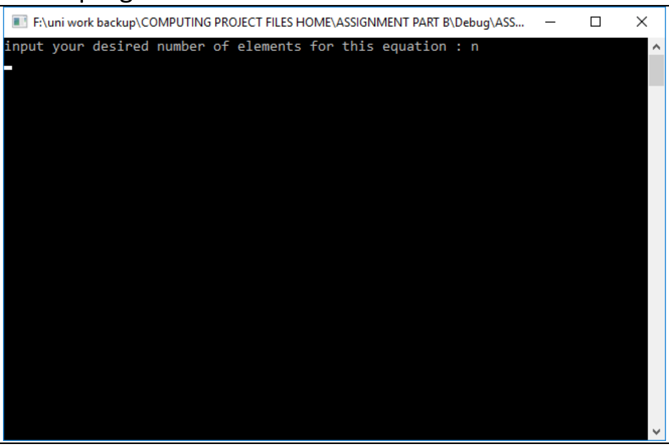
Testing the program with a small sample number	 The program works as intended when the number of elements are low, in this example, 4 elements.
Testing the program with a large sample number	  The number of elements being given values resets when the value for 17 has been given to one of the arrays when I set the number of elements to 20.

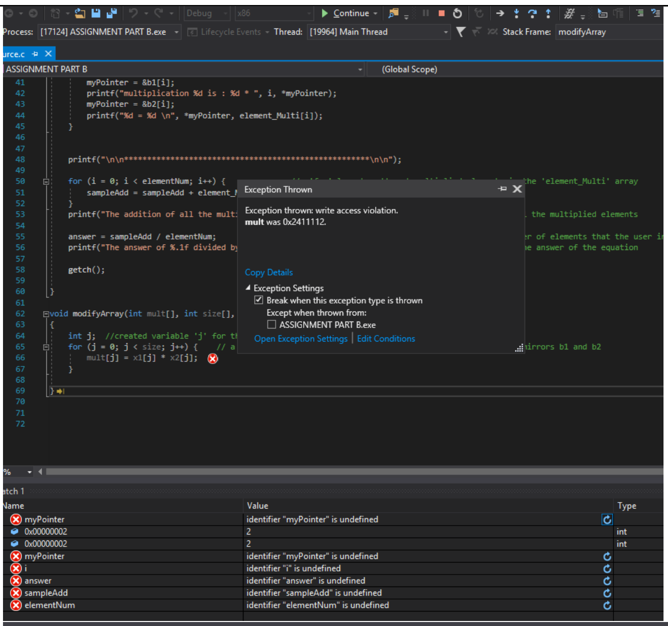
```
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART B\Debug\ASS...
input your desired number of elements for this equation : 4
input value for first array of element 0 : 3
input value for second array of element 0 : 6
input value for first array of element 1 : 3
input value for second array of element 1 : 4
input value for first array of element 2 : 8
input value for second array of element 2 : 5
input value for first array of element 3 : 9
input value for second array of element 3 : 8
multiplication 0 is : 3 * 6 = 18
multiplication 1 is : 3 * 4 = 12
multiplication 2 is : 8 * 5 = 40
multiplication 3 is : 9 * 8 = 72


****************************************************


The addition of all the multiplications is : 142.0
The answer of 142.0 divided by 4 = 35.5
```

```
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART B\Debug\ASS...
input your desired number of elements for this equation : 20
input value for first array of element 0 : 2
input value for second array of element 0 : 4
input value for first array of element 1 : 5
input value for second array of element 1 : 2
input value for first array of element 2 : 6
input value for second array of element 2 : 5
input value for first array of element 3 : 4
input value for second array of element 3 : 8
input value for first array of element 4 : 9
input value for second array of element 4 : 6
input value for first array of element 5 : 4
input value for second array of element 5 : 1
input value for first array of element 6 : 2
input value for second array of element 6 : 3
input value for first array of element 7 : 6
input value for second array of element 7 : 5
input value for first array of element 8 : 4
input value for second array of element 8 : 1
input value for first array of element 9 : 2
input value for second array of element 9 : 5
input value for first array of element 10 : 4
input value for second array of element 10 : 8
input value for first array of element 11 : 9
input value for second array of element 11 : 7
```

```
F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART B\Debug\ASS...
input value for first array of element 6 : 4
input value for second array of element 6 : 5
input value for first array of element 7 : 6
input value for second array of element 7 : 3
input value for first array of element 8 : 2
input value for second array of element 8 : 1
input value for first array of element 9 : 4
input value for second array of element 9 : 5
input value for first array of element 10 : 8
input value for second array of element 10 : 6
input value for first array of element 11 : 3
input value for second array of element 11 : 2
input value for first array of element 12 : 1
input value for second array of element 12 : 4
input value for first array of element 13 : 5
input value for second array of element 13 : 8
input value for first array of element 14 : 6
input value for second array of element 14 : 5
input value for first array of element 15 : 2
input value for second array of element 15 : 5
input value for first array of element 16 : 8
input value for second array of element 16 : 4
input value for first array of element 17 : 1
input value for second array of element 1 : 2
input value for first array of element 2 :
```

Testing the program with a floating sample number	 F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART B\Debug\ASS...
	`input your desired number of elements for this equation : 0.1`
	
	`*****************************************************`
	
	`The addition of all the multiplications is : 0.0`
	`The answer of 0.0 divided by 0 = -nan(ind)`
	
	The program fails to run the first chunk of the code and instead skips to the final lines of code displaying the addition and the final answer with the answers of 0.0 and '-nan(ind)' which I am unsure what it means. This is from using the floating number 0.1
	
	 F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART B\Debug\ASSI...
	`input your desired number of elements for this equation : 2.3`
	`input value for first array of element 0 : input value for second array of eleme`
	`nt 0 : input value for first array of element 1 : input value for second array o`
	`f element 1 : multiplication 0 is : -858993460 * -858993460 = 687194768`
	`multiplication 1 is : -858993460 * -858993460 = 687194768`
	
	`*****************************************************`
	
	`The addition of all the multiplications is : 1374389504.0`
	`The answer of 1374389504.0 divided by 2 = 687194752.0`
	This is from using the floating number 2.3 and the program skips past the scanf statements and displays the memory address, proceeding to use those numbers in the program.

Testing the program with a character as the sample number	 F:\uni work backup\COMPUTING PROJECT FILES HOME\ASSIGNMENT PART B\Debug\ASS... `input your desired number of elements for this equation : n`



Process: [17124] ASSIGNMENT PART B.exe Thread: [19964] Main Thread Stack Frame: modifyArray

ASSIGNMENT PART B (Global Scope)

```
            myPointer = &b1[i];
            printf("multiplication %d is : %d * ", i, *myPointer);
            myPointer = &b2[i];
            printf("%d = %d \n", *myPointer, element_Multi[i]);
        }


        printf("\n\n*****************************************************\n\n");

        for (i = 0; i < elementNum; i++) {                                    the 'element_Multi' array
            sampleAdd = sampleAdd + element_
        }
        printf("The addition of all the mult                                  . the multiplied elements

        answer = sampleAdd / elementNum;                                      r of elements that the user i
        printf("The answer of %.1f divided by                                 he answer of the equation

        getch();

    }

    void modifyArray(int mult[], int size[],
    {
        int j;  //created variable 'j' for t
        for (j = 0; j < size; j++) {    // a                                  mirrors b1 and b2
            mult[j] = x1[j] * x2[j];
        }

    }
```

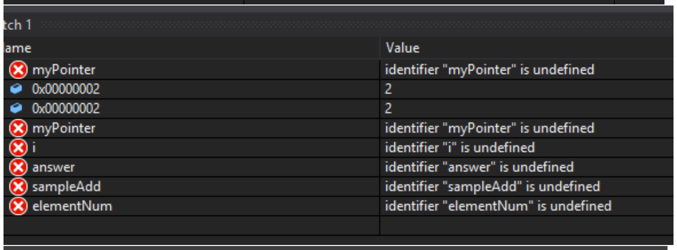
Exception Thrown

Exception thrown: write access violation.
mult was 0x2411112.

Copy Details
Exception Settings
[✓] Break when this exception type is thrown
Except when thrown from:
[ ] ASSIGNMENT PART B.exe
Open Exception Settings | Edit Conditions

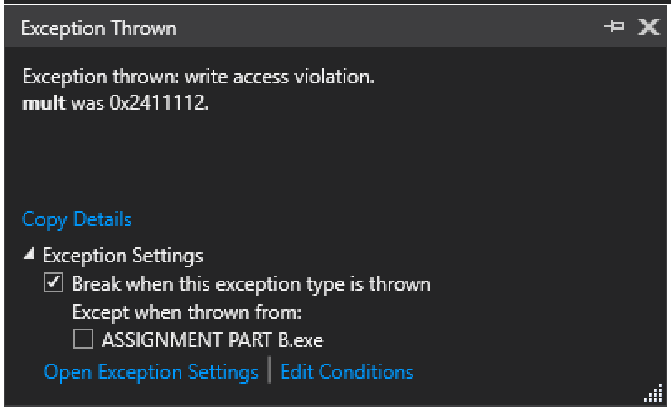
Watch 1

Name	Value	Type
myPointer	identifier "myPointer" is undefined	
0x00000002	2	int
0x00000002	2	int
myPointer	identifier "myPointer" is undefined	
i	identifier "i" is undefined	
answer	identifier "answer" is undefined	
sampleAdd	identifier "sampleAdd" is undefined	
elementNum	identifier "elementNum" is undefined	



tch 1

ame	Value
myPointer	identifier "myPointer" is undefined
0x00000002	2
0x00000002	2
myPointer	identifier "myPointer" is undefined
i	identifier "i" is undefined
answer	identifier "answer" is undefined
sampleAdd	identifier "sampleAdd" is undefined
elementNum	identifier "elementNum" is undefined



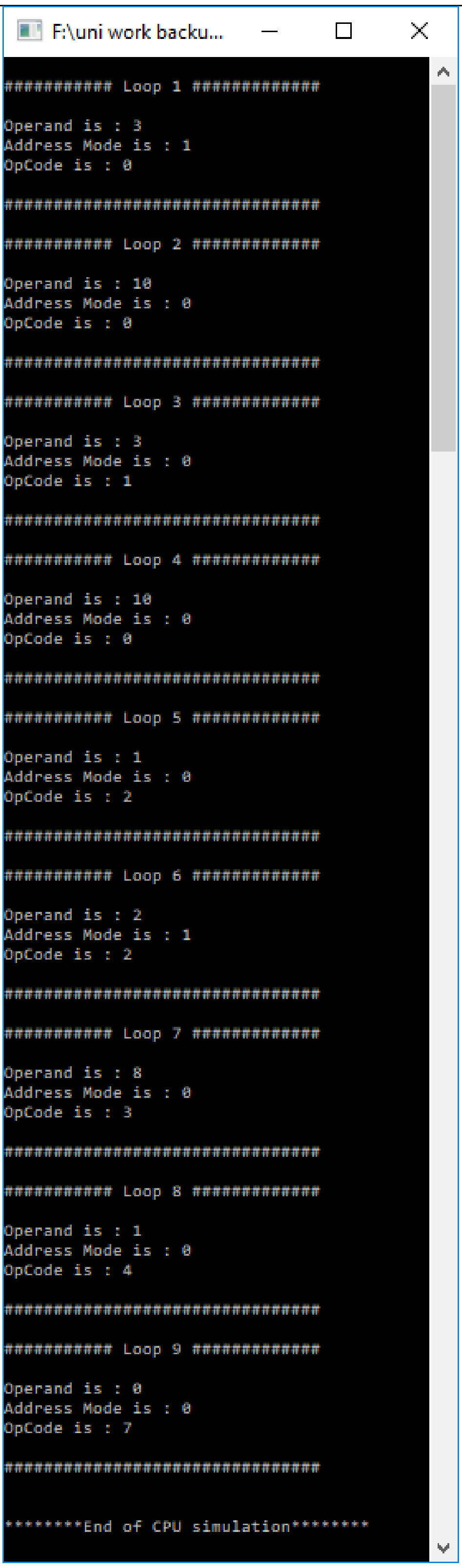
Exception Thrown

Exception thrown: write access violation.
mult was 0x2411112.

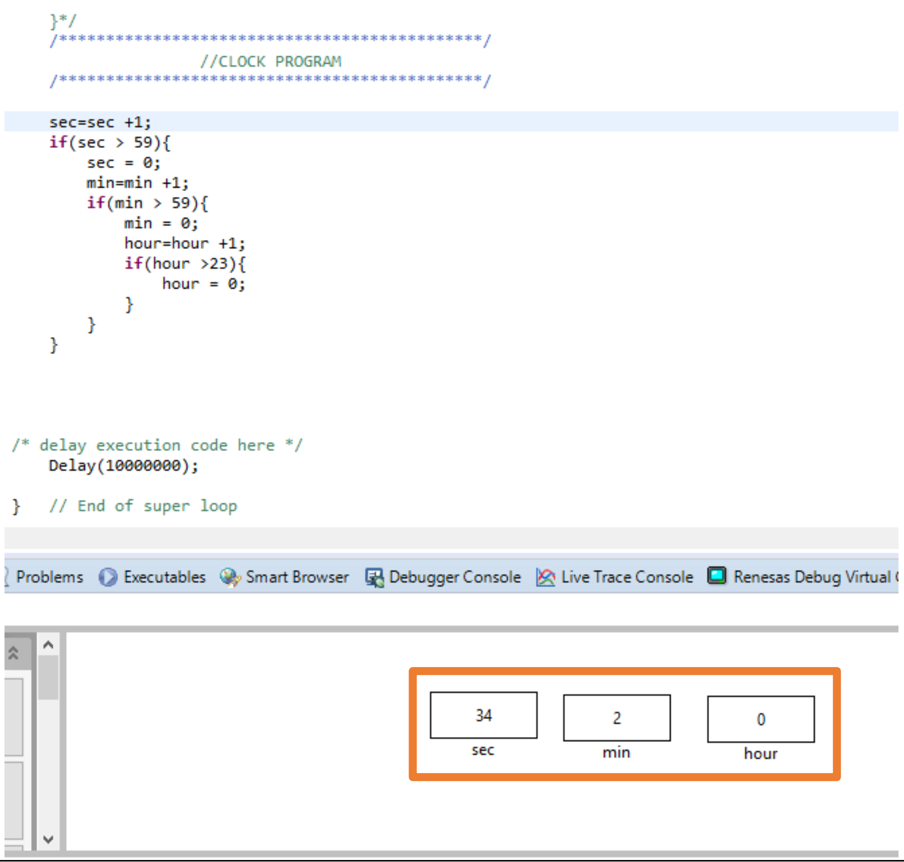
Copy Details
Exception Settings
[✓] Break when this exception type is thrown
Except when thrown from:
[ ] ASSIGNMENT PART B.exe
Open Exception Settings | Edit Conditions

Several variables are declared as undefined because of this

CPU Simulation with 8-bit Instruction: -

Allowing the program to run with a predetermined set of instructions as it normally would	 F:\uni work backu...
	`########### Loop 1 #############`
	`Operand is : 3`
	`Address Mode is : 1`
	`OpCode is : 0`
	`################################`
	`########### Loop 2 #############`
	`Operand is : 10`
	`Address Mode is : 0`
	`OpCode is : 0`
	`################################`
	`########### Loop 3 #############`
	`Operand is : 3`
	`Address Mode is : 0`
	`OpCode is : 1`
	`################################`
	`########### Loop 4 #############`
	`Operand is : 10`
	`Address Mode is : 0`
	`OpCode is : 0`
	`################################`
	`########### Loop 5 #############`
	`Operand is : 1`
	`Address Mode is : 0`
	`OpCode is : 2`
	`################################`
	`########### Loop 6 #############`
	`Operand is : 2`
	`Address Mode is : 1`
	`OpCode is : 2`
	`################################`
	`########### Loop 7 #############`
	`Operand is : 8`
	`Address Mode is : 0`
	`OpCode is : 3`
	`################################`
	`########### Loop 8 #############`
	`Operand is : 1`
	`Address Mode is : 0`
	`OpCode is : 4`
	`################################`
	`########### Loop 9 #############`
	`Operand is : 0`
	`Address Mode is : 0`
	`OpCode is : 7`
	`################################`
	`********End of CPU simulation********`
	
	The program does as it's supposed to, and runs the program showing the user the Operand, Mode and OpCode values for each instruction until the program ends.

E2 Real Time Clock

Letting the program run for a little while before recording the results.	

```
    }*/
    /**********************************************/
                    //CLOCK PROGRAM
    /**********************************************/

    sec=sec +1;
    if(sec > 59){
        sec = 0;
        min=min +1;
        if(min > 59){
            min = 0;
            hour=hour +1;
            if(hour >23){
                hour = 0;
            }
        }
    }



/* delay execution code here */
    Delay(10000000);

}   // End of super loop
```

```
34      2      0
sec     min    hour
```

Introduction: -

For this assignment, I have been tasked with writing a multitude of programs, using the programming language C. One task asks of me to simulate a multitude of logic gates using the users input information. These gates must be displayed in a C console window, with the condition that it must be user friendly and will give the user error messages if they have input an invalid value, providing a copy of the source code as evidence of my work. To finalize this section of the assignment, I must also give adequate test results that provides proof that my code performs as expected, with each possible action that the user could make being predicted and recorded.

I must then write a program that will perform a formulaic operation using an array of values that are created by the user with any number of elements that the user wishes to make. So, the conditions are that the user must be able to control the number of elements there are and the value of each of those elements, while also keeping the program as user friendly as possible. I will also provide testing evidence for this to prove that the program works as intended.

My third task will then be to create a program that will simulate the operations of a CPU, using case statements for the program to cycle through and choose an operation to execute. This is based on what information the program can decode from an instruction being processed by the program. From this project I will be able to learn how to structure switch case statements and how to use them to perform a specific instruction depending on the specified argument.

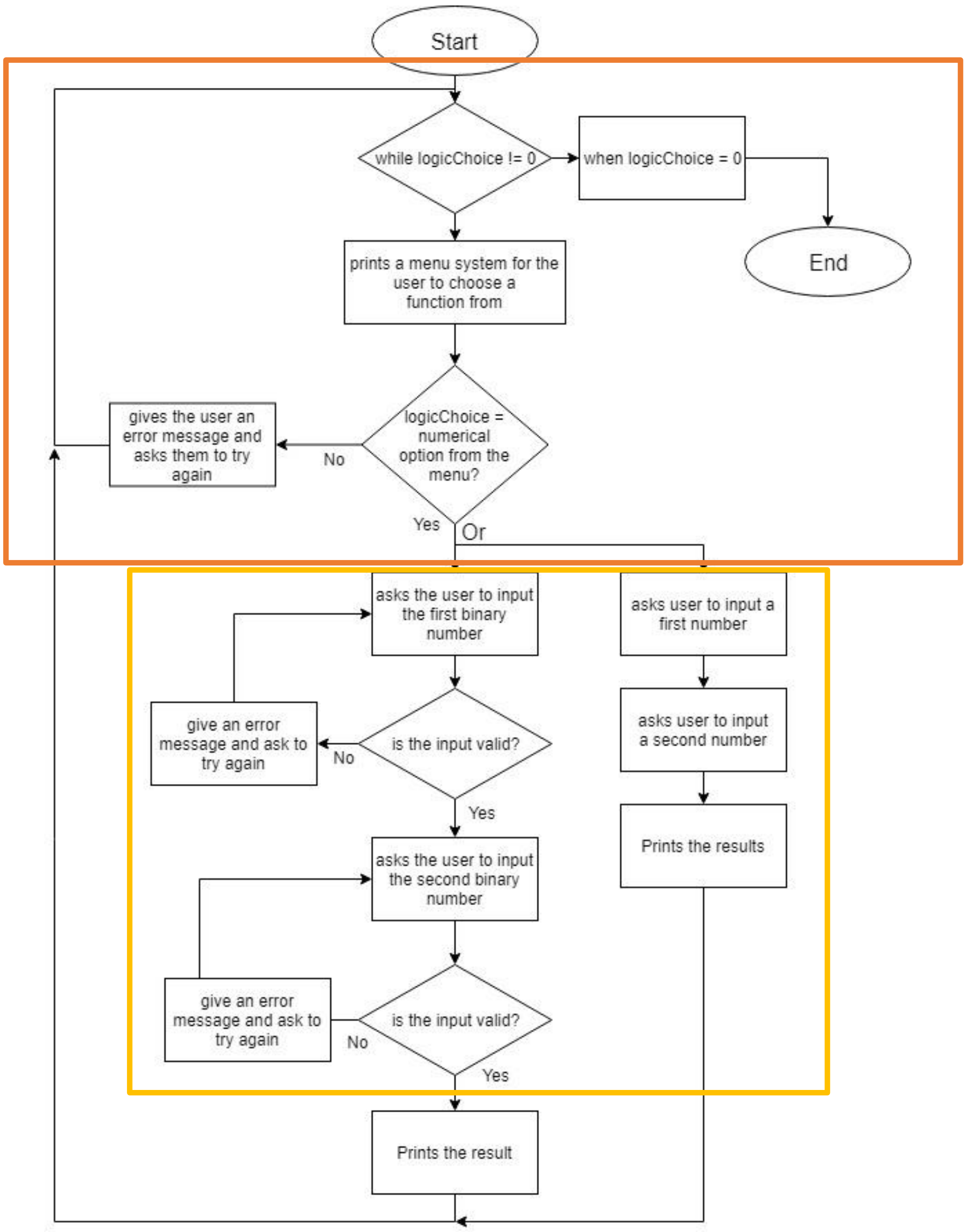
For my fourth and final project, I will be tasked with using e2 studio to create a program that will allow me to simulate a real time clock, that will count the seconds, minutes and hours at a time. The program must be able to make multiple cycles every time it counts to 24 hours, resetting back to 0 seconds, 0 minutes and 0 hours. From this I will be able to learn how to use the e2 studio software to create programs and how to simulate those programs using the interface tools given to me by the software.

After this assignment, my two goals are to be able to understand the concepts and constructs of the programming language called ‘C’, during my time planning, creating and testing these series of programs. A second goal that I pursue is to understand, and how to use the e2 studio editing and simulation software to program software that can be programmed to a microcontroller of the Renesas board provided by the university.

Logic Functions (A.)

Gates (A. I): -

Looking at the flow chart seen in fig (1.1), we can see a visual representation of the program and how the order of operation will flow throughout the runtime of this simulation program.


Start
while logicChoice != 0
when logicChoice = 0
End
prints a menu system for the user to choose a function from
logicChoice = numerical option from the menu?
No
gives the user an error message and asks them to try again
Yes
Or
asks the user to input the first binary number
is the input valid?
No
give an error message and ask to try again
Yes
asks the user to input the second binary number
is the input valid?
No
give an error message and ask to try again
Yes
Prints the result
asks user to input a first number
asks user to input a second number
Prints the results

First a while loop is used to simulate the main menu, presenting the user with a series of options that they can choose from, including an option to close the program. The program will first prompt the user to choose and option by inputting a number that corresponds to one of the options within the menu. Secondly, the program will then use the value input by the user and search through each if statement that contains the individual gate functions and then begin performing that operation.

Each gate has a similar chunk of code with very minor differences being the type of logic operation it performs. First the code for the gates starts by printing the type of gate the user selected from the menu before entering a while loop. This loop is to make sure that the user inputs a valid input when prompted.
The user inputs a value and if its more than 1 and 0 then the program will print an error message asking the user to try again. It checks this by using the variable boollogic, so when boollogic = 1 then the program will print the error message. However, the program will continue with boollogic = 0 if the user inputs a valid value. This is then repeated for the second and final input before processing the logic gate output and printing it for the user to see the results. The program will then return to the menu system ready for the user to select a new operation.

Shift Operation (A. II): -

The other type of operation the user can perform is a shift operation. This is when a value has its binary value shifted in either a left or right direction, changing the numerical value of the original number. A right shift is halving the original number for every number of shifts the number has been shifted by. So, the left shift is the opposite of this, meaning the number doubles every shift.
So simply, the program asks the user to input two number to perform a right or left shift on depending on the user's operation choice, displaying the results before returning to the menu.

To see the individual code for each logical gate and shift register, they can be found in the index from Fig (1.2) – Fig (1.9).

Notes on the Testing process (A. III): -

The first test to see how the menu system would perform when the user tries to select an option from the menu system, has shown us that the program is perfectly capable of selecting any operation that the user requested that has been displayed in the menu. These operations include six logical gate types, right and left shift operations and an option for the user to close the program when they have finished requesting simulations of logical operations. We can see the results from this test in the testing table of contents for the Logical operations program.

The second test was to see how the program would handle the user inputting an invalid number to be used in a logic gate operation. First the user had to select a logic gate from the menu and input a number that isn't either a 1 or a 0. The program will recognize this and give the user an error message stating that the input value is invalid and to re-enter a new valid value. The program will continue to loop this until a valid value has been entered for input 1 and input 2. So, the program can identify invalid whole numbers that have been input in the program, however when trying to input a float or a character, the program glitches and is stuck in a constant loop.

For the final test with the logic functions program was to test how the program would react if the shift operators had been given a small and large whole number including a floating-point number to be processed. When using the small and large numbers, the program was able shift the bits for both values left and right. However, when using a floating-point value, the program crashes due to the variable meant to store the value not being defined as a float variable.

Evaluation of the Logic functions (A. IV): -

The final product was able to detect when an invalid value had been used at the input and requesting the user to input a valid number with an error message, but it was only able to do this with a few operations. For example, the logical gates had the feature to detect when an invalid number had been used, but not when a character had been input into the program to be processed.

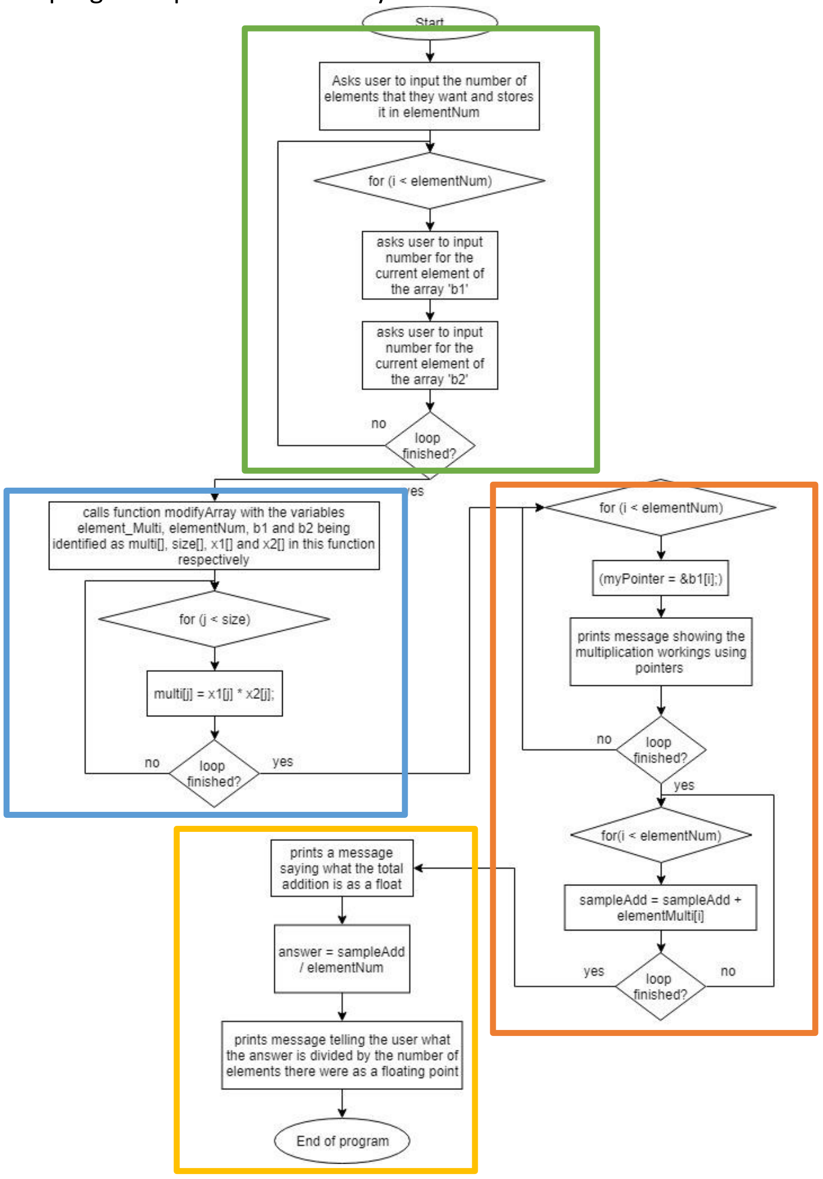
The same was true with the shift operators as it was unable to process floating numbers when being input by the user, immediately crashing the program as soon as the value had been input by the user.

This shows that the program still has issues with handling invalid values and characters, not being able to distinguish them and produce a stable error message for the user so that they could correct their mistake.

Mathematical Equations: - (B.)

Code for Mathematical Equation Explanation (B. I): -

Looking at the flowchart seen in fig (1.2), we can see a flow chart for the program explaining how the program operates in an easy to follow format.


Start
Asks user to input the number of elements that they want and stores it in elementNum
for (i < elementNum)
asks user to input number for the current element of the array 'b1'
asks user to input number for the current element of the array 'b2'
no
loop finished?
yes
calls function modifyArray with the variables element_Multi, elementNum, b1 and b2 being identified as multi[], size[], x1[] and x2[] in this function respectively
for (j < size)
multi[j] = x1[j] * x2[j];
no
loop finished?
yes
for (i < elementNum)
(myPointer = &b1[i];)
prints message showing the multiplication workings using pointers
no
loop finished?
yes
for(i < elementNum)
sampleAdd = sampleAdd + elementMulti[i]
yes
loop finished?
no
prints a message saying what the total addition is as a float
answer = sampleAdd / elementNum
prints message telling the user what the answer is divided by the number of elements there were as a floating point
End of program

The program first starts by requesting that the user inputs a number to determine the size of the arrays and how many elements they are going to have. The program will then take this value and will continue to use it in numerus 'for loops' to determine the number of times that loop will repeat. The first for loop will ask the user to input a value for the first element of the first array, and then the second array. These arrays will be used to store the information that the user wishes to use in the formula.

Secondly, after the first loop has finished, the function 'modifyArray' is called using the arguments 'element_Multi', 'elementNum', 'b1' and 'b2'. This function being called would then enter a 'for loop' to repeat a calculation for multiplying the elements of both arrays from element 0 to element n, storing the results in an array and returning it.

When the calculation loop has finished, the program would then return to where it was in the main function, continuing to the next 'for loop' that would use pointers to show the user what the calculations were, to get the result from multiplying the elements of the input arrays. Giving the results for each calculation to the user by printing them in the console. This would continue until all the calculations for each number of elements had been displayed.

Thirdly, for this next and final loop of the program, the program must add all the elements of the multiplied input values found in the array ‘element_Multi’. Adding the values until all the elements of the array have been calculated.

At the end of the loop, a message will be printed telling the user what the total addition was for the last loop, storing that value as a float, before dividing that number by the number of samples the user decided upon and input into the program and storing the answer as ‘answer’. The program would then display the workings out for this division while also finally giving the user the final answer as a floating-point value, printed on the console window. Therefor ending the program.

Notes on the Testing Process (B. II): -

From the testing we can see that, with the first method, that the program runs as expected when using a low number of elements for both arrays ‘b1’ and ‘b2’ displaying the methods of calculation for the multiplication, addition and division along with displaying the final answer.

The second test however showed some signs of bugs in the program, resetting the element number that the user is currently trying to store information in, past the 17th element. I am unsure why this happened, but I am assuming that the memory allocated to these arrays was not large enough to store these values or the for loop could have been the cause for this in some way. Possibly using a do while loop might change the outcome of this result?

The third test shows that the program will have a bug where it would skip the rest of the code past the declaration of element number, besides the final for loop and the two printf statements printing the answer of both the addition and the division by the number of elements, giving the final answer of ‘nan(ind)’. This is using the floating number of 0.1
When testing the program with a float of 2.3 the program has a bug where it skips the scanf statements and instead uses what I assume to be the memory locations of the array elements and proceeds to use those values for the calculations.

In the final test of the program, I tested the program by using a character as the input for the number of elements that was going to be used in the program. As expected the results show that the program does nothing, and visual studio gives several errors related to ‘undefined variables’ due to the data type of the variable not being a character.

Evaluation of the Mathematical Equation (B. III): -

From the results of the test, we can conclude that the program can properly operate and perform the purpose that it was designed for. That is to take an input number from the user to determine the number of elements the user wants to use in the formula, then also asking for a value for each element to be input. The program would then take these values and put them into a sum formula and give the output for the user to see.

However, the program starts to run into problems with the number of elements it can use, not being able to ask for a value past the 17th element of the arrays. This shows that the program can only operate when the number of elements is smaller than the number 17.

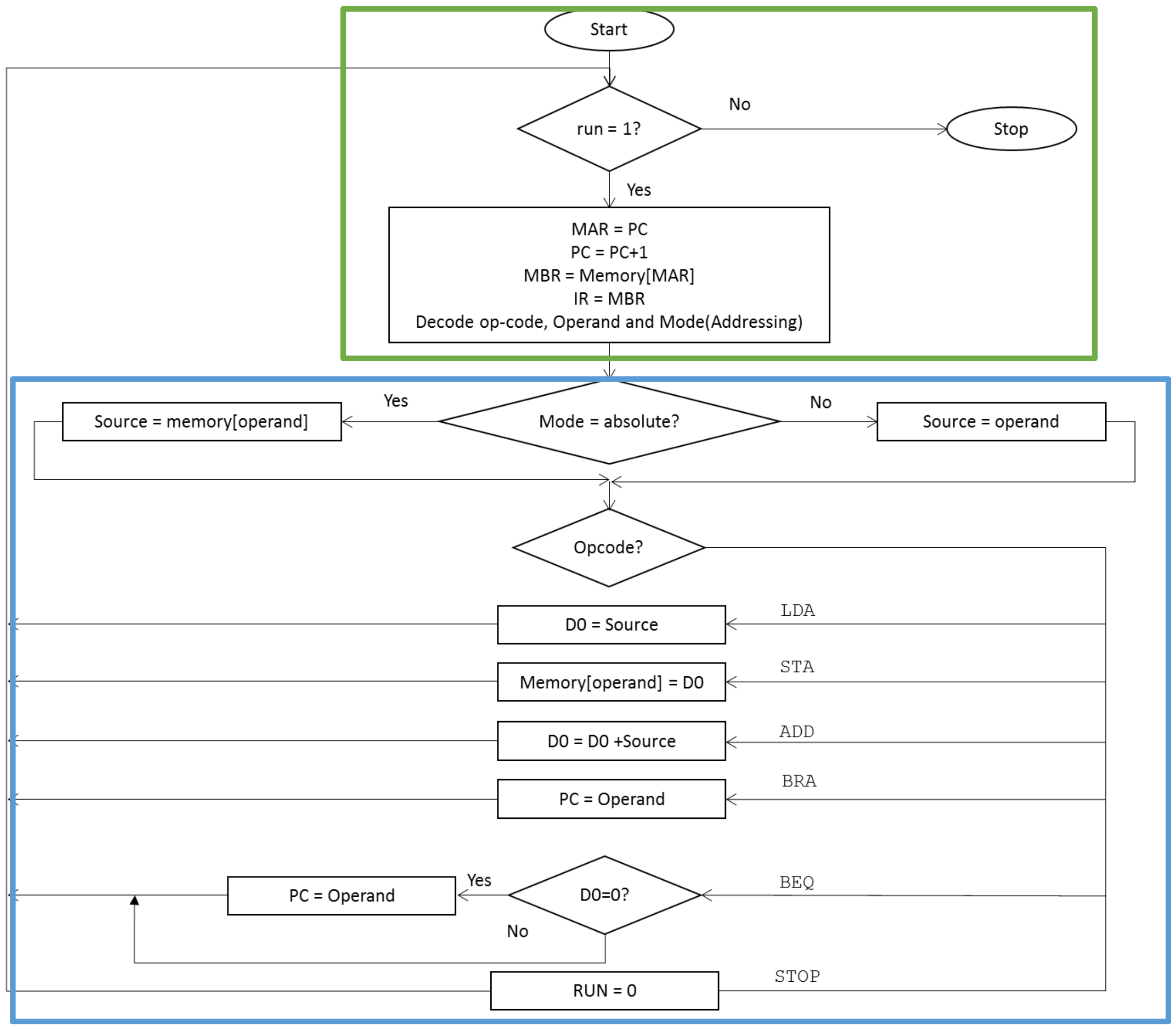
The program also has difficulties when the user inputs a character or floating-point number as the number of elements. The character causes the program to crash and gives an error message while the floating-point number rounds the number down and uses that as the number of elements, but with a twist, as it not only refuses to give the user a chance to input values for each element by skipping over them, it then uses what I would assume to be memory locations to be used as the value for each element. The program would then run as it should while using these memory location numbers, giving the final answer and workings out.

This shows that the program has issues when being confronted by input values of data types other than an integer number, running into some major bugs that break the program. Making it unusable for the user when this human error occurs, not being able to handle these values due to non-precise coding methods. However, the program is still able to function as intended if the number of elements doesn't exceed or equal to 17.

Simulation of the CPU with 8-bit instructions: - (C.)

Code for CPU Simulation Explanation (C. I): -

Looking at fig (1.3) in the index, we can see how the program runs and functions when in operation, and the steps it takes when executing these instructions that will allow the



Firstly, the program has an array that contains the multiple values that will be put through the program. Only then, the program would enter a while loop to make a constant loop until a condition was met that would end the loop, leaving the user with the printed results from the program.

While inside the loop the appropriate variables have been initialized, so ‘MAR =PC’, ‘PC = PC +1’, ‘MBR = Memory [MAR]’ and ‘IR = MBR’

When decoding the first instruction stored in the memory array, I will first mask the current instruction by AND-ing the binary number that represents the instruction by a second number that will only give the first 4 least significant bits from the instruction and store it in a variable called ‘Operand’.

The program will then need to collect the mode bit from the instruction by masking the instruction number so that the value of the fifth bit will be left, shifted 4 bits to the right and then stored in a variable called ‘addressMode’.

The final piece of information the program requires is the ‘OpCode’, which is the final three bits in the instruction number. Once again, the program will mask the instruction, so the final three most significant bits are taken, and the rest of the bits have been cleared for this purpose.

Secondly, the program will then print out the values taken from this instruction before moving on to the following ‘if statement’. This ‘if statement’ checks to see if the ‘addressMode’ is equal to 1. And if it is, then the variable source is equal to the operand value, else source is equal to memory[operand].

The program will then follow into a switch case structure using the ‘OpCode’ as the argument. This basically means that the program will look through each individual case statements listed in the program, until that case argument is equal to the value of the OpCode. [2] In which case, would then follow would be the lines of code within the case statement would be executed before using a break statement to end the process and return to the beginning of the while loop.

The loop will continue to repeat until the OpCode is equal to 7, which is the code used for initializing the run variable to equal 0. Ending the while loop operation, printing a final message stating that the it was the “End of CPU simulation”.

Explanation of the 8-bit Instruction Format (C. II): -

During this process of collecting data from an 8-bit instruction, the process of masking is used to collect this data.
The first thing to note is that you want to AND the binary 8-bit instruction with the masking number. By doing this, the program can compare each bit with its respective, related bit between both numbers. The bit value of 1 means to mask or keep that bit, while the bit value of 0 is to clear that respective bit [1].
So as an example, using the instruction of, 0x35 as the instruction number with the masking bit of, 0x0F, to find the masked value.

Instruction:	0011 0101b
Mask:	0000 1111b
Result:	0000 0101b

Looking at this table, we can see that with the masking binary of 0000 1111, will only mask or keep the right half of the instruction number in binary. This results in a result of a binary number of 5 (0000 0101) after masking. And this can be done using any number for the binary masking number.

Notes on the Testing Process (C. III): -

For the testing of the CPU 8-bit simulation, I will give the program several instructions that it will be able to cycle through and allow it to process them until it reaches an instruction that will cause the program to finish executing, giving its final message declaring that the CPU simulation has ended.

Evaluation of the Simulation (C. IV): -

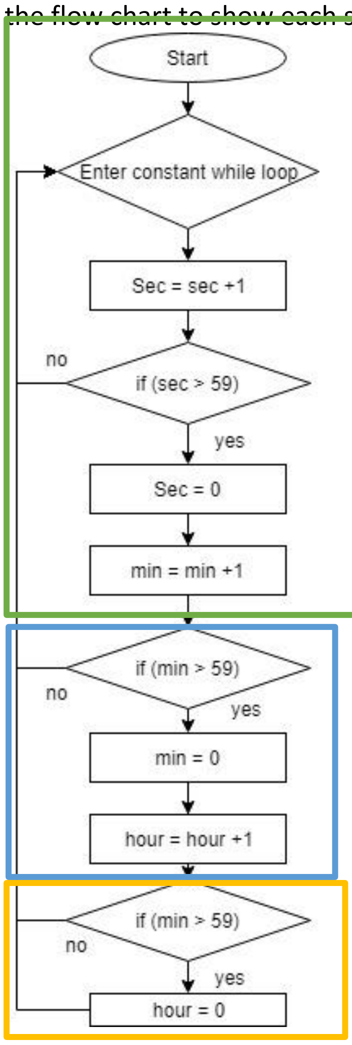
After running the program and recording the results, we can see that the program was able to run successfully, displaying all the values of the 'Operand', 'addressMode' and 'OpCode' for each consecutive instruction.

The program was able to display the value of these variable. However, it was not able to display to the user the effects of each of the case statements as they happened within the program in a clear and understandable way. Because of this it cannot be easily understood without working out each instruction and following the code of the program to understand the steps that the process took.

E2 Studio Real Time Clock: - (D.)

Explanation of the Real Time Clock Code (D. I): -

In the flow chart, which can be seen in fig (1.4), we can see how the program operates by following the flow chart to show each step of the process.



In this flow chart, it is shown that when the program first starts that it enters an infinite while loop, where the program will continue to run for as long as the simulation keeps running.

Firstly, the program will add plus 1 to the variable sec which has been initialized to 0. Then entering the first if statement to see if sec is more than 59. When this becomes true, the sec variable is then initialized back to equal 0, and the variable min is incremented by 1.

This loop will continue to happen while also leading into the next if statement to see if min is more than 59. When it is, a similar thing happens again where min is initialized to 0 and the variable hour is incremented by 1.

While also leading into the third and final if statement, the program checks to see if the variable hour is equal to more than 23. When it does, the program will then initialize hour to equal 0 and then loop back to the beginning of the while loop.

A delay function is also used to slow down the processing time of the program to allow the clock to count in real time, in seconds.

Notes on the Testing Process (D. II): -

For the testing process I will let the program run for a while to allow the program to build up enough time, that I can properly display the counting of the units. The program will also speed up this process by decreasing the delay time just enough, so the clock counts fast enough so that the cycle can be observed.

Evaluation of the Real Time Clock (D. III): -

Looking at the results for the clock test, we can see that the program was able to cycle through each if statement without any error, increasing each unit of the clock with each passing second. The sec, min and hour boxes increment by 1. There doesn't seem to be any bugs or errors in this program due to its simple nature and is very consistent with the values it presents.

Final Evaluation

After completing all these projects to specification, using both visual studio and e2 studio as my main tool, I have been able to conclude that I was able to almost complete two of my goals during my time writing this assignment and building these projects.

Firstly, I have been able to become quite competent when programming using the C language. Being able to create and code programs in a faster and efficient rate. Planning the flow of the program by creating a flow chart and using that as a tool to make sure that I can stay on track with writing the code and not getting lost half way through the writing process.

I had also been able to become quite competent with using e2 studio, programming and simulating with it any code that I had written. I had easily learned how to create a new project in the software and how to change all the settings for a software debug when first simulating the project. Problems begin to arise when trying to debug the software when the project file has been set to hardware debug when using multiple computers and not being able to change it back to a software debug. This has led me to having to create multiple project files just, so I am able to successfully simulate my code using the software.

Throughout this assignment I have been able to test and record all my projects and display it in a table format, to test that my programs are able to function as they are intended to. However, this has also meant that I have been able to discover some fatal flaws in my programs and code that will result in the program giving the user a complete incorrect answer or rendering the program completely useless from crashes and bugs. This prevents the user from being given a suitable answer by the program. And now that I know what these issues are, I can identify most of the issues that include the error from storing different types of data in variables that were not defined for holding those values.

The results show that I have a few issues with the stability of my programs that stem from the values that the user inputs into the system to be stored in a variable of the wrong type. So, in future, if I was to try and make these programs again, I would improve on the stability of the program by including an input check function that would recognize if the user had input an invalid value into the program. It would then return the user back to the input request giving an error message that the input was invalid. This function would also need to recognize the type of data being input by the user to check that it is even the correct data type in the first place.
I would also need to fix the while loop in the Formula program so that the input loop wouldn't reset when it reaches the 17th loop.

References: -

[1]: User239558. 2012. Stack Overflow. [Online]. [5 April 2019]. Available from: https://stackoverflow.com/questions/10493411/what-is-bit-masking

[2]: Tutorialspointcom. 2019. Wwwtutorialspointcom. [Online]. [5 April 2019]. Available from: https://www.tutorialspoint.com/cprogramming/c_decision_making.htm

[3]: Thoughtcocom, P.L. 2019. ThoughtCo. [Online]. [5 April 2019]. Available from: https://www.thoughtco.com/example-java-code-for-building-a-simple-gui-application-2034066

Projects Code: -

Logic Functions code

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


int num1, num2, f; //declareing the two input variables and the result variable
int logicChoice = 1;     //declaring the selecting AND/OR variable
char boollogic = 1;      //declaring a boolean variable

void main(void)
{

        while (logicChoice != 0) // a while loop so after each simulated gate the user is put back to
the selection menu again until the exit option is selected.
        {

                //display user instructions
                printf_s("Please select the logic gate you wish to simulate by entering the
numbers...\n1 - AND Operation\n2 - OR Operation\n3 - NAND Operation\n4 - NOR Operation\n5 - ExOR
Operation\n6 - ExNOR Operation\n7 - Shift Right Operation\n8 - Shift Left Operation\n0 - To exit the
program\n\n");  //prints the selection list for the user
                scanf_s("%d", &logicChoice);
        //capture the user input and store it in the logicChoice variable

                if (logicChoice == 0) {          // a message printing the end of the program
                        printf("End of program");
```

```
		}

		if (logicChoice == 1)
		{
			//AND gate opperation
			printf_s("--- 2 input 'AND' Gate Operation ---\n\n ");

			while (boollogic == 1)          // a while loop that asks the user to input
the correct value if they had first entered the wrong value.
			{
				printf_s("\ninput the first number: ");  //prints a string asking the
user to input the first number thats either 1 or 0
				scanf_s("%d", &num1);    //takes the value input by the user and is
placed in the num1 variable

				if (num1 > 1) {                                         //   an   if
statement that checks if the value input by the user is correct and changes the boollogic variable if
the entered value is appropriate.
					printf("\n------------------------------\nThis is not a logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
				}
				else {
					boollogic = 0;
				}

			}
			boollogic = 1;

			while (boollogic == 1)
			{
				printf_s("input a second number: ");     //prints a string asking the
user to input the second number thats either 1 or 0
				scanf_s("%d", &num2);                            //takes   the   value
input by the user and is placed in the num2 variable

				if (num2 > 1) {
					printf("\n------------------------------\nThis is not a logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
				}
				else {
					boollogic = 0;
				}

			}

			boollogic = 1;

			f = (num1)& (num2);
		//preform AND operation
			printf_s("\noutput      of      an      and      operation      is:      %d
\n\n*****************************************************\n\n", f);      //dislpays    AND    operation
result

			/*
			Truth Table:
			0 0 = 0
			0 1 = 0
			1 0 = 0
			1 1 = 1
			*/

		}
		else if (logicChoice == 2)
		{
			//OR gate opperation
			printf_s("--- 2 input 'OR' Gate Operation ---\n\n ");

			while (boollogic == 1)          // a while loop that asks the user to input
the correct value if they had first entered the wrong value.
			{
				printf_s("\ninput the first number: ");  //prints a string asking the
user to input the first number thats either 1 or 0
				scanf_s("%d", &num1);    //takes the value input by the user and is
placed in the num1 variable

				if (num1 > 1) {                                         //   an   if
statement that checks if the value input by the user is correct and changes the boollogic variable if
the entered value is appropriate.
```

```
                                        printf("\n------------------------------\nThis is not a logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
                                }
                                else {
                                        boollogic = 0;
                                }

                        }
                        boollogic = 1;

                        while (boollogic == 1)
                        {
                                printf_s("input a second number: ");     //prints a string asking the
user to input the second number thats either 1 or 0
                                scanf_s("%d", &num2);                           //takes   the   value
input by the user and is placed in the num2 variable

                                if (num2 > 1) {
                                        printf("\n------------------------------\nThis is not a logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
                                }
                                else {
                                        boollogic = 0;
                                }

                        }

                        boollogic = 1;

                        f = num1 | num2;
        //preform OR operation
                        printf_s("\noutput       of       an       or       operation       is:       %d
\n\n*****************************************************\n\n", f);      //dislpays    OR    operation
result

                        /*
                        Truth Table:
                        0 0 = 0
                        0 1 = 1
                        1 0 = 1
                        1 1 = 1
                        */
                }

else if (logicChoice == 3)
                {
                        //NAND gate opperation
                        printf_s("--- 2 input 'NAND' Gate Operation ---\n\n ");

                        while (boollogic == 1)          // a while loop that asks the user to input
the correct value if they had first entered the wrong value.
                        {
                                printf_s("\ninput the first number: ");  //prints a string asking the
user to input the first number thats either 1 or 0
                                scanf_s("%d", &num1);   //takes the value input by the user and is
placed in the num1 variable

                                if (num1 > 1) {                                         //   an   if
statement that checks if the value input by the user is correct and changes the boollogic variable if
the entered value is appropriate.
                                        printf("\n------------------------------\nThis is not a logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
                                }
                                else {
                                        boollogic = 0;
                                }

                        }
                        boollogic = 1;

                        while (boollogic == 1)
                        {
                                printf_s("input a second number: ");     //prints a string asking the
user to input the second number thats either 1 or 0
                                scanf_s("%d", &num2);                           //takes   the   value
input by the user and is placed in the num2 variable

                                if (num2 > 1) {
                                        printf("\n------------------------------\nThis is not a logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
                                }
                                else {
                                        boollogic = 0;
                                }

                        }

                        boollogic = 1;

                        f = (num1)& (num2);
                //preform NAND operation
                        printf_s("\noutput      of      an      NAND      operation      is:      %d
\n\n*****************************************************\n\n", !f);     //dislpays   NAND   operation
result by making the result of AND in 'f' and 'not'ing it with '!'

                        /*
                        Truth Table:
                        0 0 = 1
                        0 1 = 1
                        1 0 = 1
                        1 1 = 0
                        */
                }
                else if (logicChoice == 4)
                {
                        //NOR gate opperation
                        printf_s("--- 2 input 'NOR' Gate Operation ---\n\n ");

                        while (boollogic == 1)          // a while loop that asks the user to input
the correct value if they had first entered the wrong value.
                        {
                                printf_s("\ninput the first number: ");  //prints a string asking the
user to input the first number thats either 1 or 0
                                scanf_s("%d", &num1);    //takes the value input by the user and is
placed in the num1 variable

                                if (num1 > 1) {                                         //   an   if
statement that checks if the value input by the user is correct and changes the boollogic variable if
the entered value is appropriate.
                                        printf("\n------------------------------\nThis is not a logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
                                }
                                else {
                                        boollogic = 0;
                                }

                        }
                        boollogic = 1;

                        while (boollogic == 1)
                        {
                                printf_s("input a second number: ");     //prints a string asking the
user to input the second number thats either 1 or 0
                                scanf_s("%d", &num2);                           //takes   the   value
input by the user and is placed in the num2 variable

                                if (num2 > 1) {
                                        printf("\n------------------------------\nThis is not a logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
                                }
                                else {
                                        boollogic = 0;
                                }

                        }

                        boollogic = 1;

                        f = (num1) | (num2);
                //preform NOR operation
                        printf_s("\noutput      of      an      NAND      operation      is:      %d
\n\n*****************************************************\n\n", !f);     //dislpays   NOR   operation
result by making the result of OR in 'f' and 'not'ing it with '!'

                        /*
                        Truth Table:
```

```
                        0 0 = 1
                        0 1 = 0
                        1 0 = 0
                        1 1 = 0
                        */
                }
                else if (logicChoice == 5)
                {
                        //EXOR gate opperation
                        printf_s("--- 2 input 'ExOR' Gate Operation ---\n\n ");

                        while (boollogic == 1)          // a while loop that asks the user to input
the correct value if they had first entered the wrong value.
                        {
                                printf_s("\ninput the first number: ");  //prints a string asking the
user to input the first number thats either 1 or 0
                                scanf_s("%d", &num1);    //takes the value input by the user and is
placed in the num1 variable

                                if (num1 > 1) {                                         //   an   if
statement that checks if the value input by the user is correct and changes the boollogic variable if
the entered value is appropriate.
                                        printf("\n------------------------------\nThis is not a logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
                                }
                                else {
                                        boollogic = 0;
                                }

                        }
                        boollogic = 1;

                        while (boollogic == 1)
                        {
                                printf_s("input a second number: ");     //prints a string asking the
user to input the second number thats either 1 or 0
                                scanf_s("%d", &num2);                        //takes   the   value
input by the user and is placed in the num2 variable

                                if (num2 > 1) {
                                        printf("\n------------------------------\nThis is not a logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
                                }
                                else {
                                        boollogic = 0;
                                }

                        }

                        boollogic = 1;

                        f = num1 ^ num2;
        //preform ExOR operation
                        printf_s("\noutput       of       an       or       operation       is:       %d
\n\n*****************************************************\n\n", f);      //dislpays   ExOR   operation
result

                        /*
                        Truth Table:
                        0 0 = 0
                        0 1 = 1
                        1 0 = 1
                        1 1 = 0
                        */
                }



                else if (logicChoice == 6)
                {
                //EXNOR gate opperation
                printf_s("--- 2 input 'ExNOR' Gate Operation ---\n\n ");

                while (boollogic == 1)          //  a  while  loop  that  asks  the  user  to  input  the
correct value if they had first entered the wrong value.
                {
                        printf_s("\ninput the first number: ");  //prints a string asking the user to
input the first number thats either 1 or 0
```

```
				scanf_s("%d", &num1);   //takes the value input by the user and is placed in
the num1 variable

				if (num1 > 1) {                                         //  an  if  statement
that  checks  if  the  value  input  by  the  user  is  correct  and  changes  the  boollogic  variable  if  the
entered value is appropriate.
					printf("\n------------------------------\nThis   is   not   a   logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
				}
				else {
					boollogic = 0;
				}

			}
			boollogic = 1;

			while (boollogic == 1)
			{
				printf_s("input a second number: ");     //prints a string asking the user to
input the second number thats either 1 or 0
				scanf_s("%d", &num2);                        //takes  the  value  input  by
the user and is placed in the num2 variable

				if (num2 > 1) {
					printf("\n------------------------------\nThis   is   not   a   logic
based\nnumber. Please input a logic\nbased number. For example 0\nand 1\n-----------------------------
-\n");
				}
				else {
					boollogic = 0;
				}

			}

			boollogic = 1;

			f = (num1) ^ (num2);     //preform ExNOR operation
			f = !f;
			printf_s("\noutput        of        an        or        operation        is:        %d
\n\n*******************************************************\n\n", f);       //dislpays   ExNOR   operation
result

			/*
			Truth Table:
			0 0 = 1
			0 1 = 0
			1 0 = 0
			1 1 = 1
			*/
			}



			else if (logicChoice == 7)
			{
				//Right shift of 9 '00001001' = 4 '00000100'
				printf_s("--- 2 input 'Shift Right' Operation ---\n\n ");

				printf_s("input a first number to shift: ");     //prints a string asking the
user to input the first number
				scanf_s("%d", &num1);                                         //takes   the
value input by the user and is placed in the num1 variable
				printf_s("input a second number to shift: ");    //prints a string asking the
user to input the second number
				scanf_s("%d", &num2);                                         //takes   the
value input by the user and is placed in the num2 variable

				printf_s("num1>>1 = %d\n", num1 >> 1);
				printf_s("num2>>1                                                              =
%d\n\n*******************************************************\n\n", num2 >> 1);
			}
			else if (logicChoice == 8)
			{
				//Left shift of 9 '00001001' = 18 '00010010'
				printf_s("--- 2 input 'Shift Left' Operation ---\n\n ");

				printf_s("input a first number to shift: ");     //prints a string asking the
user to input the first number
```

```
                        scanf_s("%d", &num1);                                           //takes   the
value input by the user and is placed in the num1 variable
                        printf_s("input a second number to shift: ");    //prints a string asking the
user to input the second number
                        scanf_s("%d", &num2);                                           //takes   the
value input by the user and is placed in the num2 variable

                        printf_s("num1<<1 = %d\n", num1 << 1);
                        printf_s("num2<<1                                                             =
%d\n\n****************************************************\n\n", num2 << 1);

                }
        }

        getch();

}
```

Mathematical Equation

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void main(void);                //declaring the main function
void modifyArray(int b[], int size[], int x1[], int x2[]);              //declaring function
modifyArray



void main(void)
{
        int element_Multi[10];          //assigning arrays for b1, b2 and element_Multi
        int i;                                  //assigning variables for 'i', for the loops,
sampleAdd and answer
        float sampleAdd = 0;
        float answer;
        int b1[9];
        int b2[9];
        int elementNum;                 //elementNum is used to store the int value that the user
inputs reprisenting the number of elements the b1, b2 and element_Multi arrays have

        /*******************************************************************************************
/

        printf("input your desired number of elements for this equation : ");            // asks the
user to input a value of the array elements and stores the value in elementNum
        scanf_s("%d", &elementNum);

        /*******************************************************************************************
/

        int *myPointer;         //assigning a pointer

        for (i = 0; i < elementNum; i++) {              // a 'for' loop that takes the inputs from
the user and stores them in each element of b1 and b2
                printf("input value for first array of element %d : ", i);
                scanf_s("%d", &b1[i]);

                printf("input value for second array of element %d : ", i);
                scanf_s("%d", &b2[i]);

        }

        modifyArray(element_Multi, elementNum, b1, b2);          //calling the function 'modifyArray'

        for (i = 0; i < elementNum; i++) {              // a 'for' loop that uses pointers to display
the values of elements b1 and b2, putting them in a formula that shows the working out of the
multiplication
                myPointer = &b1[i];
                printf("multiplication %d is : %d * ", i, *myPointer);
                myPointer = &b2[i];
                printf("%d = %d \n", *myPointer, element_Multi[i]);
        }
        printf("\n\n*****************************************************\n\n");

        for (i = 0; i < elementNum; i++) {                              //a 'for' loop to add each
multiplied elements in the 'element_Multi' array
                sampleAdd = sampleAdd + element_Multi[i];
        }
        printf("The addition of all the multiplications is : %.1f \n", sampleAdd);        //prints the
sum of all the multiplied elements

        answer = sampleAdd / elementNum;                        //finding the answer by dividing sampleAdd by
the number of elements that the user input
        printf("The answer of %.1f divided by %d = %.1f \n\n", sampleAdd, elementNum, answer);
        //prints the answer of the equation

        getch();

}

void modifyArray(int mult[], int size[], int x1[], int x2[])              // the modifyArray function
{
        int j;  //created variable 'j' for the 'for' loop
        for (j = 0; j < size; j++) {     // a 'for' loop that multiplies each element from x1 and x2
which mirrors b1 and b2
                mult[j] = x1[j] * x2[j];
        }

}
```

CPU 8-bit simulation

```
#include <stdio.h>
#include <conio.h>


/*****************************************************************************************************
***************************/
char IR;
char Operand;
char AddressMode;
char OpCode;
int i = 0;
unsigned char PC, MAR, MBR, source;
unsigned char D0 = 0;

unsigned char memory[16] = { 0x13, 0x0A, 0x23, 0x33, 0x41, 0x52, 0x68, 0xEF, 0x81, 0xE0, 0x0A, 0x03,
0x08, 0x0F, 0, 0 };
unsigned char run = 1;



void main(void); //declaring function main


void main(void)
{
        /*****************************************************************************************************
***********************************/


        /*beginnin of the CPU simulation*/


        PC = 0;         //initialising the variable PC

        while (run == 1) {       // a while loop that loops the program while the varibale 'run' is
equal to '1'

                i++;

                MAR = PC;
                PC = PC + 1;
                MBR = memory[MAR];
                IR = MBR;

                /*decode and display operand*/
                Operand = IR & 0x0F;                            // mask IR with 0x0F(b00001111)
                printf("\n########### Loop %d #############\n\nOperand is : %d", i, Operand);

                /*decode and display addressing mode*/
                AddressMode = IR & 0x10;         // shift 4 bits to the right
                printf("\nAddress Mode is : %d", AddressMode >> 4);               //mask IR with
0x10(b00010000) and shift 4 bits to the right

                /*decode and display Op-code*/
                OpCode = (IR & 0xE0) >> 5;               // mask IR with 0xE0(b11100000) and shift
right by 5
                printf("\nOpCode is : %d\n\n#################################\n", OpCode);


                if (AddressMode == 1) {          //an if statment to check if the addressMode is 1 or
0
                        source = Operand;        //mode = 1
                }
                else
                {
                        source = memory[Operand];        //mode = 0
                }
                /* process OP code */
                switch (OpCode) {                //a switch statement to search for a case that
matches the OpCode value and executes it
                case 0x00:
                        D0 = source;
                        break;

                case 0x01:
                        memory[Operand] = D0;
                        break;

                case 0x02:
                        D0 = D0 + source;
                        break;

                case 0x03:
                        PC = Operand;
                        break;

                case 0x04:
                        if (D0 = 0) {    //checks to see if D0 = 0
                                PC = Operand;
                        }
                        break;
                case 0x05:
                        printf("Instruction not found");
                        break;

                case 0x06:
                        printf("Instruction not found");
                        break;

                case 0x07:
                        run = 0;// initialises run = 0 to end the loop
                        break;
                }

        }
        printf("\n\n********End of CPU simulation********");

        getchar();
        /*******************************************************************************************
******************************************/
}
```

```
/***********************************************************************/
/*                                                                     */
/*   FILE        : Main.c                                  */
/*   DATE        :Tue, Oct 31, 2006                                    */
/*   DESCRIPTION :Main Program                                         */
/*   CPU TYPE    :                                                     */
/*                                                                     */
/*   NOTE:THIS IS A TYPICAL EXAMPLE.                                   */
/*                                                                     */
/***********************************************************************/
//#include "typedefine.h"
#ifdef __cplusplus
//#include <ios>                        // Remove the comment when you use ios
//_SINT ios_base::Init::init_cnt;       // Remove the comment when you use ios
#endif


void main(void);
#ifdef __cplusplus
extern "C" {
void abort(void);
}
#endif

void Delay(unsigned long delay);
unsigned char incriment = 0;     // increment variable
unsigned char sec = 0;
unsigned char min = 0;
unsigned char hour = 0;

void main(void)
{



        while(1)// beginning of Super loop
        {
         /* Add your code here */

        /*      if(incriment <=100){
                        incriment=incriment +1;
                }
                else if (incriment > 100){
                        incriment = 0;
                }*/
                /**********************************************/
                                                //CLOCK PROGRAM
                /**********************************************/

                sec=sec +1;
                if(sec > 59){    //checking if sec is over 59 and if it is then set sec to 0 and
increment min by 1
                        sec = 0;
                        min=min +1;
                        if(min > 59){    //checking if min is over 59 and if it is then set min to 0
and increment hour by 1
                                min = 0;
                                hour=hour +1;
                                if(hour >23){    //checking if hour is over 23 and if it is then set
hour to 0
                                        hour = 0;
                                }
                        }
                }




        /* delay execution code here */
                Delay(10000000);

        }       // End of super loop

}       // end of main

#ifdef __cplusplus
void abort(void)
{

}
#endif
```

```
void Delay(unsigned long delay)
{
        unsigned long i;
        for (i=0; i<delay;i++)
        {}
}
```