T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE3038 Project I

Project Report

Group Members

170517033 Yasin Alper Bingül

150117031 Ahmet Faruk Yılmaz

150116069 Hakan Adaklı

1. Base Converter

The base converter function takes the binary representation of the number to be converted as input and the type input that specifies whether the number should be displayed as hexadecimal or 2's complement.

The length of the input given in the program is calculated first. Based on the input length, the last bit of the given input is reached and calculations are made from right to left. In this step, 2's complement or hexadecimal function is called according to the user's type input.

a. Type 1 (2’s Complement)

In 2’s complement notation, we convert binary numbers to decimal forms. The biggest difference from the normal binary to decimal conversions of 2’s complement is that the given number can be negative. This is achieved by having the leftmost bit set to 1. In this way, the range of numbers that can be displayed becomes wider.

While calculating, we start from the right and move toward the left. If the bit value is 1, we take the basis of the index and add it to the current value. If the bit value is 0, we move to the next bit. If we have arrived at the leftmost bit, we must check whether the leftmost bit is 1 or 0. Depending on this situation, the number can be negative or positive. If the leftmost bit is 0, we continue the operations normally. If it is 1, we find the decimal value of the leftmost bit and subtract the sum of the other bits from this value and multiply the result by -1 to get the negative value.

b. Type 2 (Hexadecimal)

In hexadecimal representation, the binary number is divided into 4-bit parts. The maximum value that a 4-bit piece can take in decimal form is 15 (1111). Hexadecimal representation can take 0-9, A-F values. The maximum value that can have the F value is 15 in decimal form.

When calculating hexadecimal, firstly the input length is divided by 4 and 0 is added to the left as the remaining value. After this process, the entire input is split into 4-bit chunks, similar to the 2's complement calculation. However, there are no negative values in hexadecimal representation. If the leftmost bit becomes 1, it is added to the current value. When the addition of each 4 bits is complete, the mapping function is called according to the current decimal value. For example, if the calculated number is 10, the value A is added to the value to be printed at the end. This operation is done from left to right. When all bits have been calculated, the result is printed on the screen.

Sample runs:

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:1

Input:1100101
Type:1
Output:-27
```

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:1

Input:1100101
Type:2
Output:65
```

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:1

Input:1101
Type:1
Output:-3
```

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:1

Input:11111101
Type:1
Output:-3
```

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:1

Input:00111010
Type:2
Output:3A
```

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:1

Input:0101
Type:1
Output:5
```

2. Add Rational Number

The Rational Number option takes two rational numbers and prints sum of them as a rational number too. The user enters numerator and denominator of both numbers. Then function sums them and simply converts into canonical form which done by using Euclid's algorithm.

According to Rational Number Function:

i. Firstly, takes two rational numbers

$$Rational\ Number1 = \frac{First\ Nominator}{First\ Denominator}$$

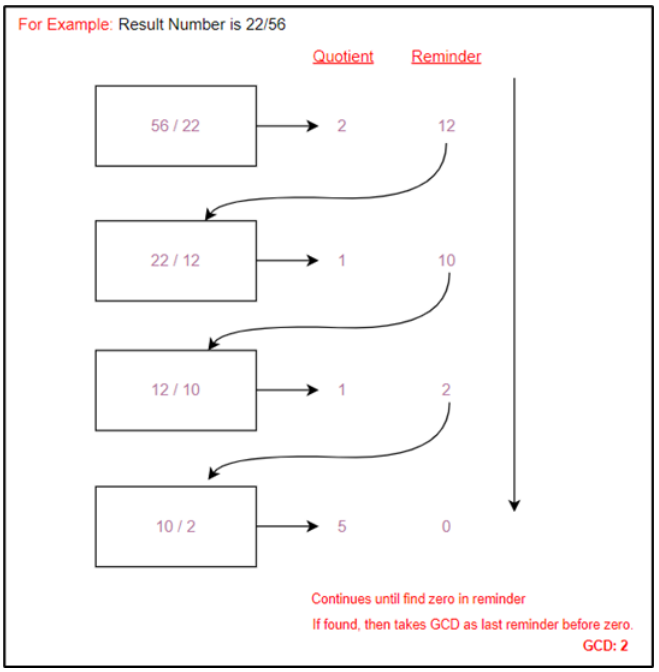
$$Rational\ Number2 = \frac{Second\ Nominator}{Second\ Denominator}$$

ii. Sums them with the formula below.

$$Result\ Number = \frac{First\ Nominator * Second\ Denominator + Second\ Nominator * First\ Denominator}{First\ Denominator * Second\ Denominator}$$

iii. Simplifies this result number into canonical form with using Euclid's Algorithm.

a. Firstly, divides Nominator and Denominator of Result Number. And find the Greatest Common Divisor of two numbers by **Euclid's Algorithm.**



a. Then divides both numerator and denominator numbers with greatest common divisor which has found previous step. Thus, we can simplify the solution of result number.

$$\frac{\frac{22}{2}}{\frac{56}{2}} = \frac{11}{28}$$

Sample runs:

i.

```
Please select an option:2
Enter the first numerator:3
Enter the first denominator:8
Enter the second numerator:1
Enter the second denominator:7
3/8+1/7=29/56
```

ii.

```
Please select an option:2
Enter the first numerator:4
Enter the first denominator:9
Enter the second numerator:3
Enter the second denominator:11
4/9+3/11=71/99
```

iii.

```
Please select an option:2
Enter the first numerator:9
Enter the first denominator:4
Enter the second numerator:5
Enter the second denominator:6
9/4+5/6=37/12
```

3. Text Parser

Text parser separates words from spaces and punctuation marks. It prints the words separately on the lines.

First it takes the inputs and prints them. Check the punctuation marks in the second input for all characters in the first input. If the characters are the same, it switches to reading the next character in the first input. If the characters are different, it prints this character in the first input. Also, if the first input has a whitespace, or if the two inputs have a similar character, the word is finished so it can go to the bottom line. **Our main goal is to print a word on each line.**

The codes compared the ascii values, and used the mips function 'beq'. When the ascii value of the character in the first input is equal to any value in the second input, it is passed to the bottom line. In the second input, we set the character limit to eight.

Sample runs:

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:3
Input text:!!!The first CSE-3038 project is due on 20-April-2022 (Wednesday).!!!
Parser characters:,.!?_-()
Output:
The
first
CSE
3038
project
is
due
on
20
April
2022
Wednesday
```

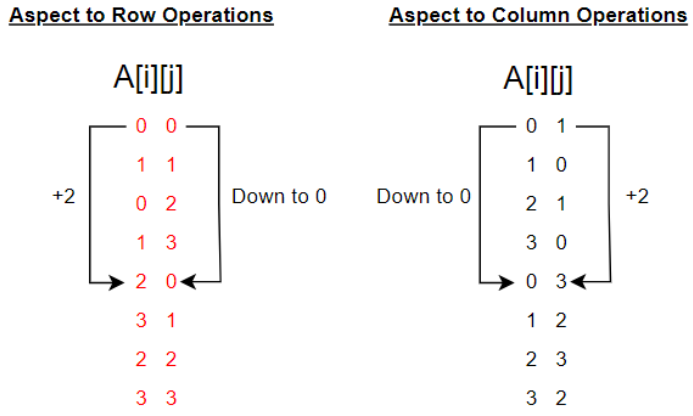
```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:3
Input text:(Midterms start on 11-April-2022 for this semester.)
Parser characters:-()
Output:
Midterms
start
on
11
April
2022
for
this
semester.
```

4. Mystery Matrix Operation

The Mystery Matrix Operation simply takes the input array and prints multiplied the specified array indexes.

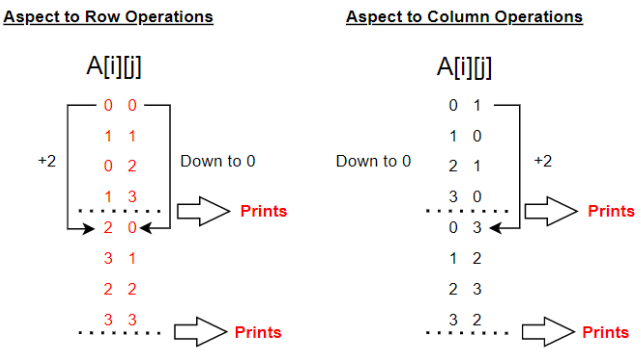
The funcrion firstly takes the input string from the user. Then stores the values in this string into an array. Calculates the size of the array and find out the square root of size of the array. (The square root of size of the array will be used in reaching the value of specified array index.) After finding the valuable informations about array, program computes specified array index values according to given index pattern. You can see down below the founded pattern for given route.

00	01	02	03
10	11	12	13
20	21	22	23
30	31	32	33



For computing row operations, each iteration increases the row base number by 2 and updates the column number down to zero. For column operations, each iteration increases the column case number by 2 and updates the row number down to zero. As mentioned before, this process continues until the row and column numbers are finished.

After computing all array index values by the given pattern, program multiplies the index values per iteration and prints them on console screen. At the end of program, whole array index multiplications will be printed on console screen.



Sample runs:

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:4
Input:
3 8 12 1 2 3 4 3 2 4 5 4 5 6 7 11 4 2 9 3 23 14 5 58 5 3 4 5 7 8 4 2 4 9 1 2
3456    341040  5040
20736   10120   1200
```

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:4
Input:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
144     22176
1300    5040
```

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:4
Input:
2 5 8 7
14
40
```

5. Main Menu

The main menu function provides the user with the ability to manage different operations from a single screen. The user enters any of the functions that appear on the screen as an

input to the program and can access that function. The program runs continuously as long as the user does not call the exit function.

The program first prints the functions on the screen and asks the user to enter a value from 1 to 5. The first 4 values are the available functions in the program. If the user enters the value 5, the program is terminated. According to the user's input, the desired function is reached with the jump operation and the function works. When the function is finished, the registers are cleared and the menu screen is returned.

Sample runs:

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:1

Input:1111
Type:1
Output:-1

Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:2
Enter the first numerator:3
Enter the first denominator:5
Enter the second numerator:7
Enter the second denominator:10
3/5+7/10=13/10
```

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:3
Please enter the input text : CSE-3038!2022
Please enter the parser characters : -!
Output:
CSE
3038
2022

Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:
```

```
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:4
Input:
3 4 5 6
18
20

Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option:5
Program ends, bye.
-- program is finished running --
```