CSE 3038

COMPUTER ORGANIZATION

PROGRAMMING PROJECT 1

-REPORT-

Stakeholders:

1-İsmail ÖKSÜZ - 150119516

2-Emre SAĞIROĞLU - 150119766

3-Barış HAZAR - 150118019

4-Ulaş Deniz IŞIK – 150118887

Menu

Basically, in the menu section if user enters a valid number which is in the list, code jumps to the corresponding question. The menu text repeats itself until the user enters a valid number.

- Sample of invalid number input.

```
Welcome to our MIPS project!
Main Menu:
1. Base Converter
2. Add Rational Number
3. Text Parser
4. Mystery Matrix Operation
5. Exit
Please select an option: 7
Invalid input! Try again.
```

- If the input number between 1-4, program runs the questions according to the input number.

```
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:
```

- The user can exit the program by entering 5 which is exit option.

```
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 --
```

Question 1

In the first question, the program asks the user to enter a binary number. Then it asks for the type. Firstly, the program calculates both types of answers. Then, the type selected by the user is printed on the screen.

- This is a type of solution which is two's complement binary digits converted to decimal numbers

Example Runs for type 1

```
Please select an option: 1
Input: 1111 1111 1111 1111
Type: 1
Output: -1
```

```
Input: 01111
Type: 1
Output: 15
```

Example Runs for type 2

- These examples, which are the second type, convert the entered numbers to the hexadecimal format and print the values to the screen as follows.

```
Input: 1111 1110 1101 1100 1011 1010 1001 1000
Type: 2
Output: FEDCBA98
```

```
Please select an option: 1
Input: 11 1111 0001 0101
Type: 2
Output: 3F15
```

Question 2

Firstly, the user enters 4 numbers as 2 numerators and 2 denominators. Then the program adds the two fractions. The entered numbers are printed on the screen as integer pairs. The number obtained after summing is printed on the screen in its simplified form. The simplification process is done with the Euclid Algorithm.

Example Run 1:

```
Please select an option: 2
Enter the first numerator: 1
Enter the first denominator: 5
Enter the second numerator: 7
Enter the second denominator: 10
1/5 + 7/10 = 9/10
```

- (1/5) + (7/10) = (2/10) + (7/10) = 9/10

Example Run 2:

```
Enter the first numerator: 7
Enter the first denominator: 24
Enter the second numerator: 3
Enter the second denominator: 8
7/24 + 3/8 = 2/3
```

- (7/24) + (3/8) = (7/24) + (9/24) = 16/24 = 2/3

Example Run 3:

```
Enter the first numerator: 11
Enter the first denominator: 45
Enter the second numerator: 1
Enter the second denominator: 5
11/45 + 1/5 = 4/9
```

- (11/45) + (1/5) = (11/45) + (9/45) = 20/45 = 4/9

Question 3

First, we store the address of our 2 input strings in different registers. Then we loop the first input byte by byte and for every byte, we jump to the “checkSpecials” function which loops compares that byte with every byte of second string which it holds the special characters. If they are equal, that means the current byte we pass to the function “checkSpecials” is a special character which we have to remove. We then store whitespace into that address and proceed the loop. This was the first step.

In the second step we loop our firs string and if we encounter any non-whitespace character, we jump to the function “extract” and start to store every byte until we encounter a whitespace. When we encounter a whitespace, we do not concatenate it into the “output” string, instead we store “\n” (new line) to the end of the string and then we jump back to the loop. When we encounter “0” which means end of the string, we go out of the loop. In the end, we will have a string that there is no whitespace in it and every word has “\n” character at the end of them.

To sum up, we first replace the special characters with whitespace and then extract all whitespace characters and add “\n” (new line) at the end of every extracted word.

Example Runs:

```
Please select an option: 3

Input Text: #"Marmara Universitesi - Muhendislik Fakultesi" | Aydinevler-Maltepe_Istanbul/\

Parser characters: #"-|_/\

Output:
Marmara
Universitesi
Muhendislik
Fakultesi
Aydinevler
Maltepe
Istanbul
```

```
Input Text: //Riza Baba, "Kisin yola cikmadan once mutlaka kar lastigi taktiginizdan emin olun." dedi.//

Parser characters: / , " .

Output:
Riza
Baba
Kisin
yola
cikmadan
once
mutlaka
kar
lastigi
taktiginizdan
emin
olun
dedi
```

Question 4

In the fourth question, which is the last question of the program, we defined a mysterious operation that can calculate the given operation from the project file. First of all, the inputs should be entered with a space between them. Then the entered inputs were multiplied with their cross neighbors and positioned where they should be in the matrix. Finally, a new 2x2 matrix was created from a 4x4 matrix.

Example Run 1:

```
Please select an option: 4
Input: 11 27 8 45
Output:
495
216
```

Example Run 2:

```
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
Output:
3456 341040 5040
20736 10120 1200
```

Clear Registers Function:

At the beginning of the program, all the registers are reset each time the program restarts.

```
915
916  clearregisters:
917      li $t0, 0
918      li $t1, 0
919      li $t2, 0
920      li $t3, 0
921      li $t4, 0
922      li $t5, 0
923      li $t6, 0
924      li $t7, 0
925      li $t8, 0
926      li $t9, 0
927
928      li $s0, 0
929      li $s1, 0
930      li $s2, 0
931      li $s3, 0
932      li $s4, 0
933      li $s5, 0
934      li $s6, 0
935      li $s7, 0
936
937      jr $ra
938
```