

This project is done as part of Computer Organization course. The code was implemented using emu8086 emulator to run the code use the emulator.

Team work by:

- Sondos Ahmad Aabed 1190652
- Rama Hani Abuadas 1192100

Contents

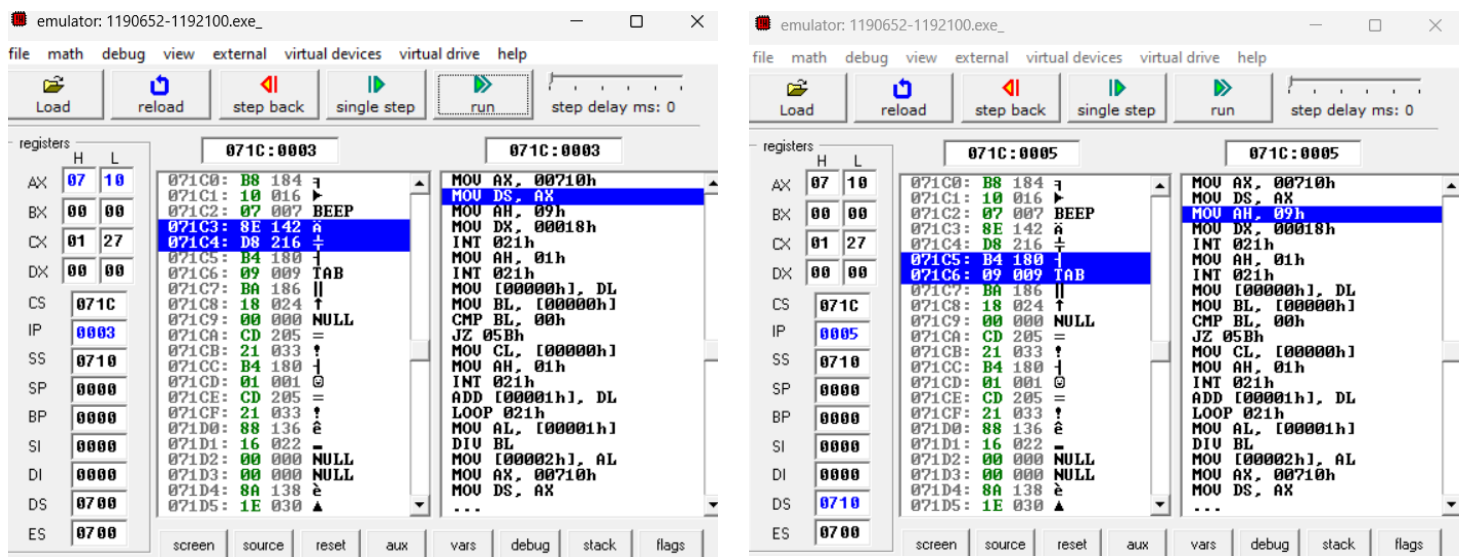
Section one: introduction	2
User Greetings	2
N input and validation.....	3
Size input and validation.....	4
Section two: read the numbers	5
String input.....	5
Input validation	5
Convert to BCD	6
Section three: Calculations	6
Calculate the average	6
Print the output.....	7
Conclusion	8

Section one: introduction

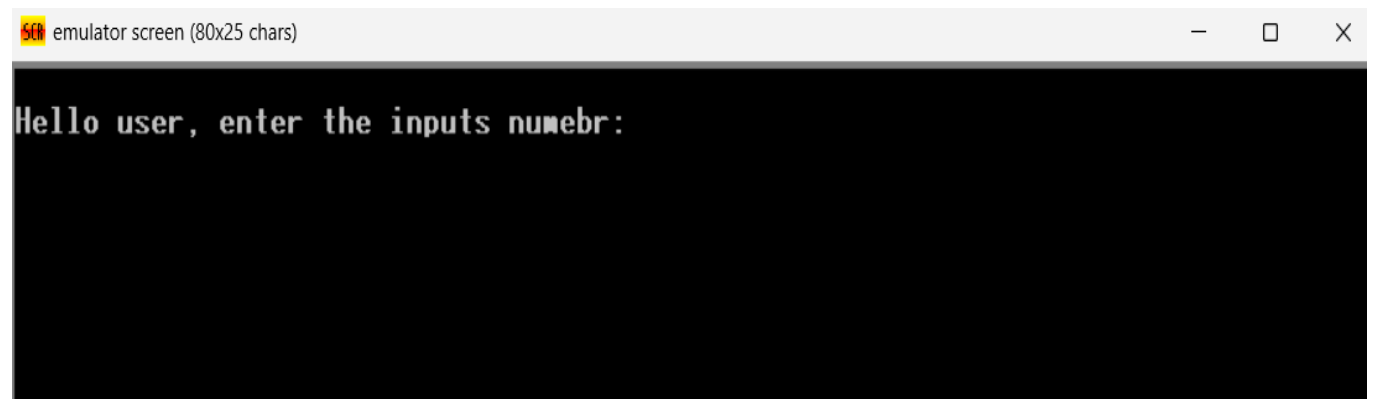
In this project, we write an Intel 8086 Assembly program that reads N numbers as Strings, convert them into variable sized Integer numbers, and then we print the summation and average of the numbers. The program allow the user to decide the size of the input number itself (variable sized inputs).

User Greetings

In this section, our program prints the greeting message to the user and asks them to enter the value of the count that they want to input using our program:

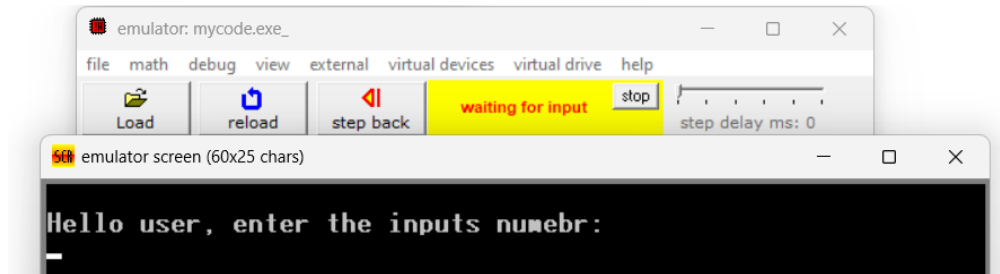


After moving 09h to ah register, and calling the INT 21h and loading the effective address of the data, the screen will show the greeting message as follow:

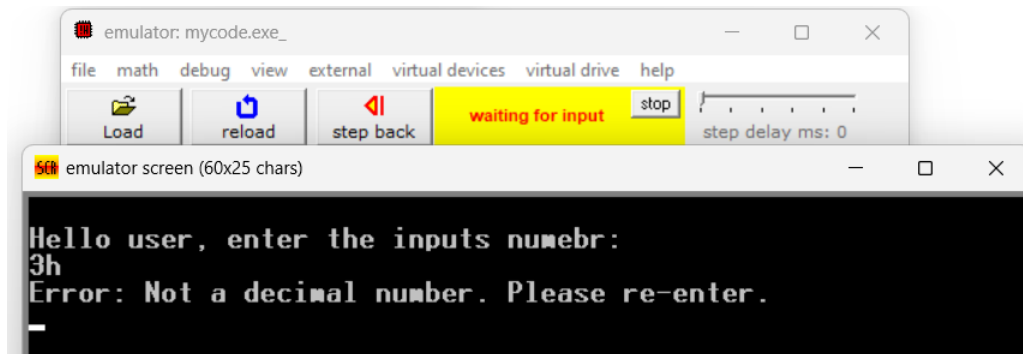


N input and validation

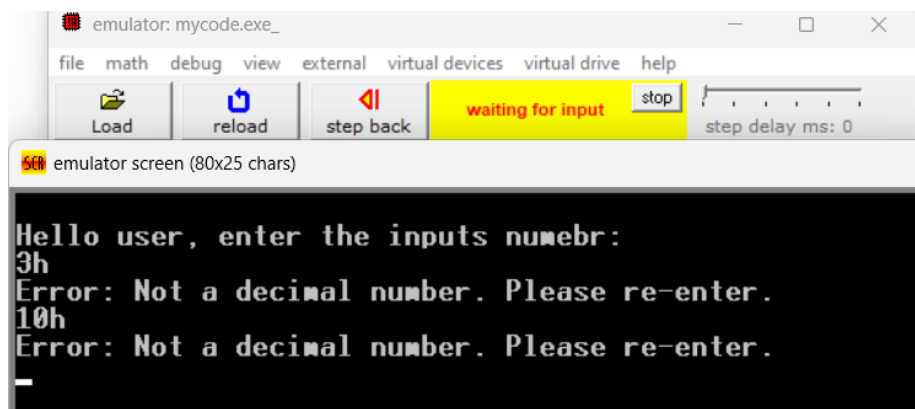
Using the 0ah function, and calling the interrupt will show the cursor waiting for an input from the user that will be more than one character, it will be read and converted into decimal if valid:



The user choses to enter 3h numbers for example, which is invalid input it has to be decimal. Now that the user has inputted the count of the numbers, our program will validate the input of the user to make sure that N is a non-zero positive integer if it's not it will print the error message and let them try again until it works:

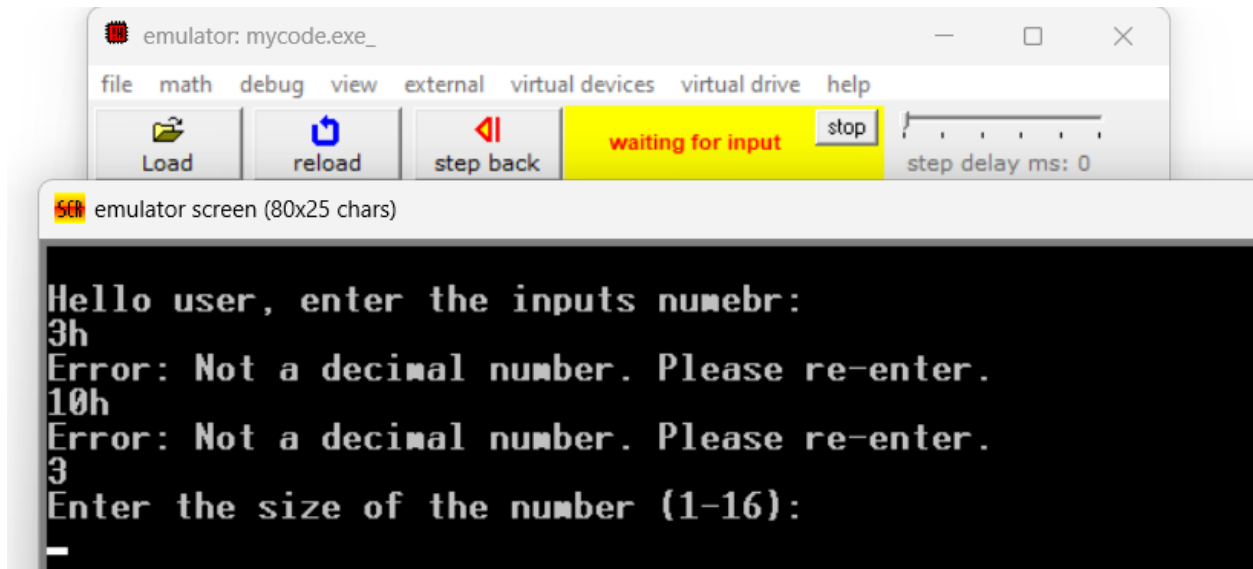


Then the user decides to enter 10h which is invalid too, the program will still ask them to input a valid number until they do: our program also checks if it's zero and ask them to re input

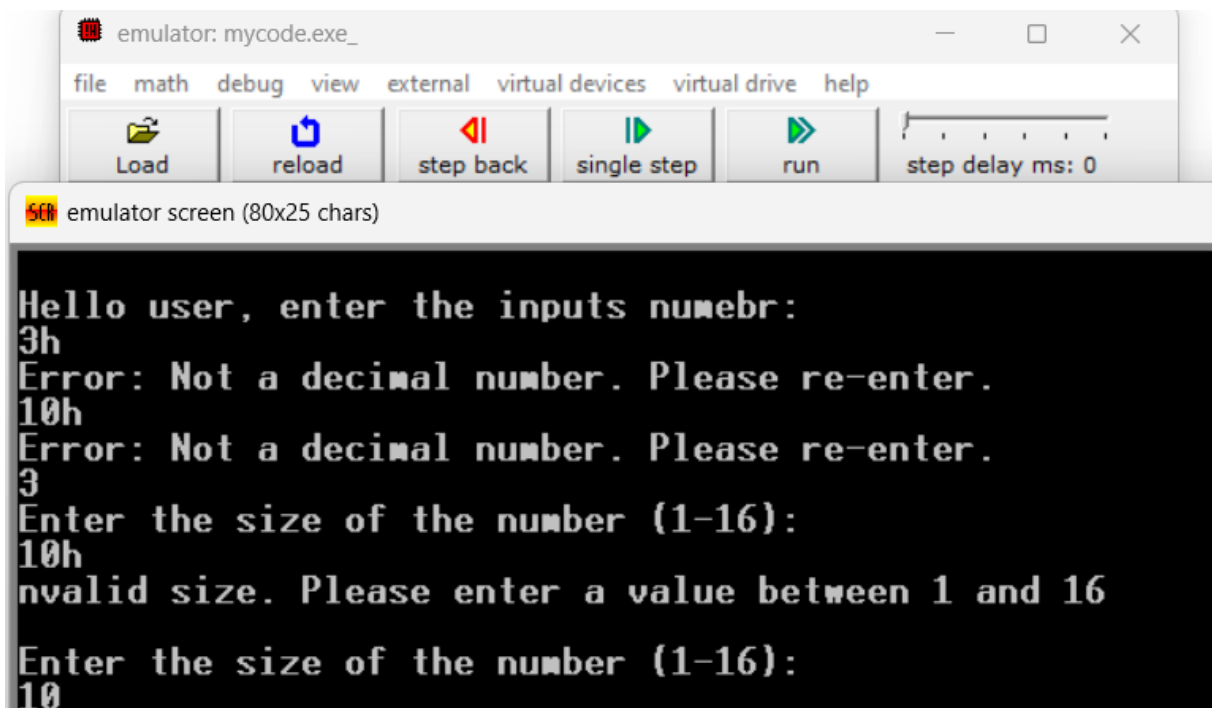


Size input and validation

Repeating the first step, we will print another message to the user that enters the size of the numbers in bytes are going to input as shown in the above screenshot. This will only happen if and only if the user inputs a valid number for inputs size.



Now that the user has inputted the size, our program will validate the input of the user to make sure that the size is a valid between 1-16 bytes. If it's not it will keep asking for the right one until it's valid.

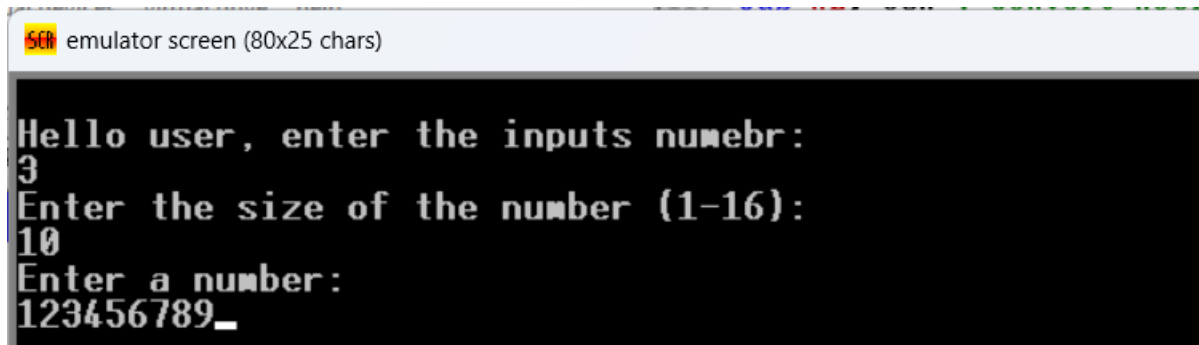


Section two: read the numbers

In this section the user inputs the numbers that they want, we take them as string and then perform input validation so that the numbers are converted into Binary Coded Decimal (BCD) numbers and be used in the next section. *Note: In this case, we rerun the program for convenience reasons so that the screenshot of the program doesn't take space like the above test.*

String input

Now that the user inputted the count of the numbers and the size and they were both validated, it is the time to enter those numbers, which will be read as strings, we ask the user to input their numbers n times.

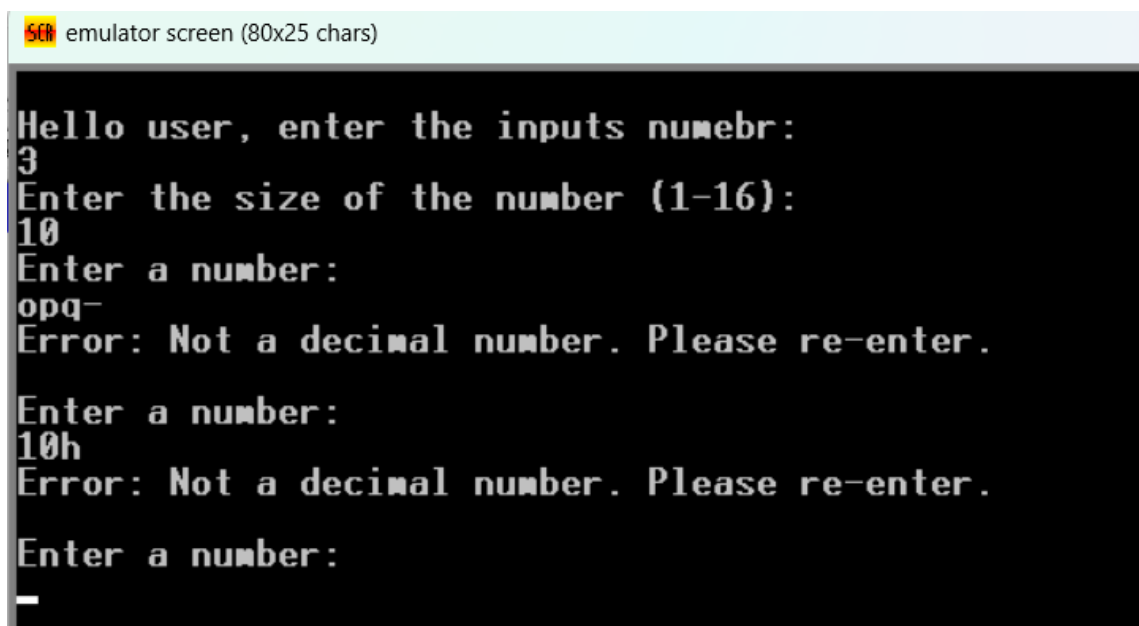


```
emulator screen (80x25 chars)

Hello user, enter the inputs numebr:
3
Enter the size of the number (1-16):
10
Enter a number:
123456789_
```

Input validation

After the input is done, our program will validate that the number is both a decimal and in the specified sized that was entered as follows:



```
emulator screen (80x25 chars)

Hello user, enter the inputs numebr:
3
Enter the size of the number (1-16):
10
Enter a number:
opq-
Error: Not a decimal number. Please re-enter.

Enter a number:
10h
Error: Not a decimal number. Please re-enter.

Enter a number:
_
```

Convert to BCD

Although we have read during our research for this project that BCD is a valid format for representing decimal numbers, but it is not commonly used in modern computer systems it was not recommended to convert but we did it as required.

Our program converts a decimal input string to a binary-coded decimal (BCD) value by processing each digit of the string one at a time. It does this by first converting the ASCII character representation of the digit to its corresponding decimal value, then using the AAM (ASCII-adjusted multiplication we learned this instruction during our research to solve the project) instruction to convert the decimal value to its BCD representation.

Then we combine the high and low of the resulting BCD value into a single byte using the OR instruction. Finally, the BCD value is added to the sum.

Section three: Calculations

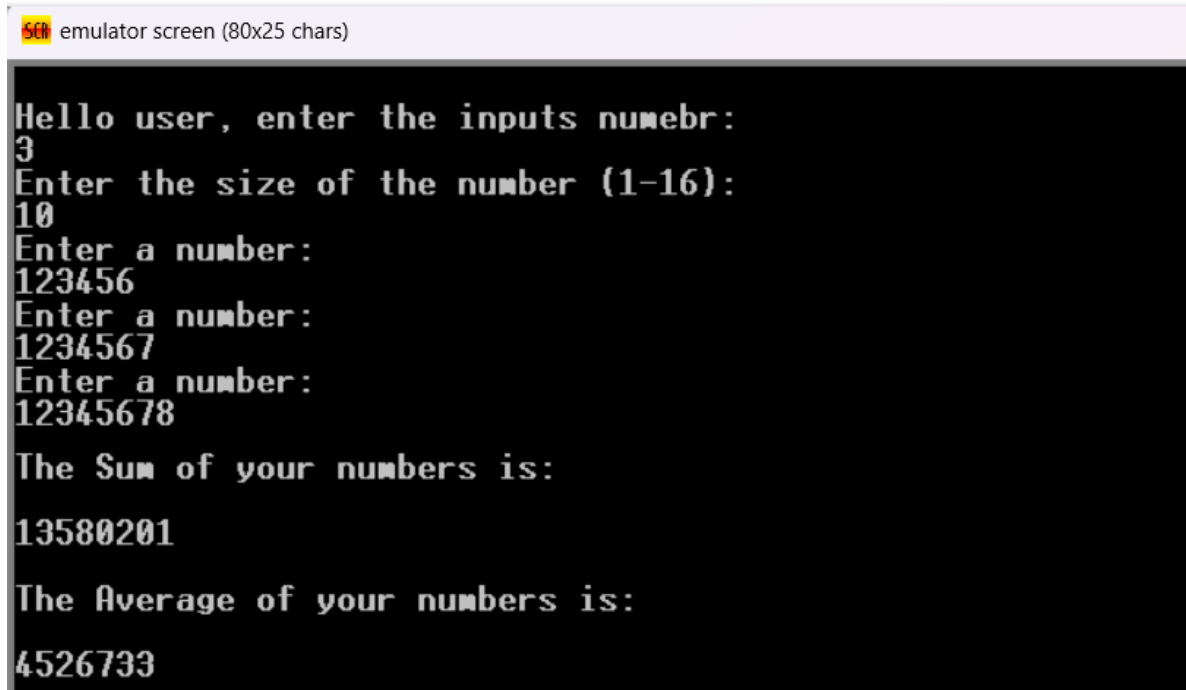
In this section the calculations are performed. The summation of the inputted numbers and the average of them. Then these will be print to the screen as the final outputs.

Calculate the average

In this section the calculations are performed, our program calculates the average by using the div instruction in which it divides the sum which our program calculated in the previous section by the number of inputs N and stores the value in avg variable to be printed.

Print the output

Once the inputs are done and validated to be correct our program prints the output which is both the sum and the average of the numbers entered previously as follows:



The screenshot shows a terminal window titled "emulator screen (80x25 chars)". The text displayed is as follows:

```
Hello user, enter the inputs numebr:
3
Enter the size of the number (1-16):
10
Enter a number:
123456
Enter a number:
1234567
Enter a number:
12345678

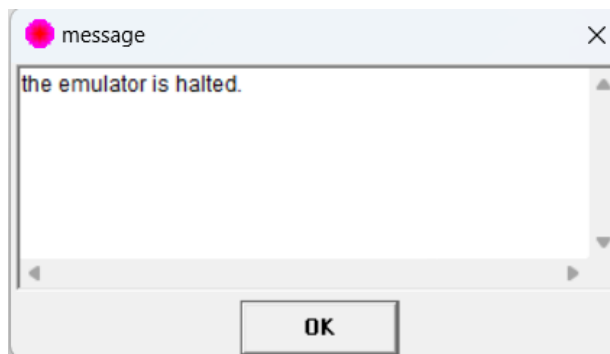
The Sum of your numbers is:

13580201

The Average of your numbers is:

4526733
```

This is how the program ends and performs as required.



Conclusion

In conclusion, and to sum up the work we have done this is an algorithm that describes our approach in the code:

1. *Prompt the user to input value of N and validate the input.*
2. *Prompt the user to input size of the numbers in bytes and validate the input.*
3. *Loop the following steps N times:*
 - a. *Read the input string from the user.*
 - b. *Validate the input to make sure it is a decimal number with the correct size*
 - c. *Convert the input string to a BCD format with two digits per byte*
 - d. *Add the number to the sum.*
4. *Divide the sum by N to get the average.*
5. *Print the sum and average of the numbers.*

Overall we found this approach to be efficient and practical to work with variable sized inputs. We also organized our main sections in our program in a way that each section has a clear purpose and flow.

Our program also defined several prompts and error messages for improving the user experience and handling possible error that may occur.