

# CSE3038 Computer Organization - Spring 2021

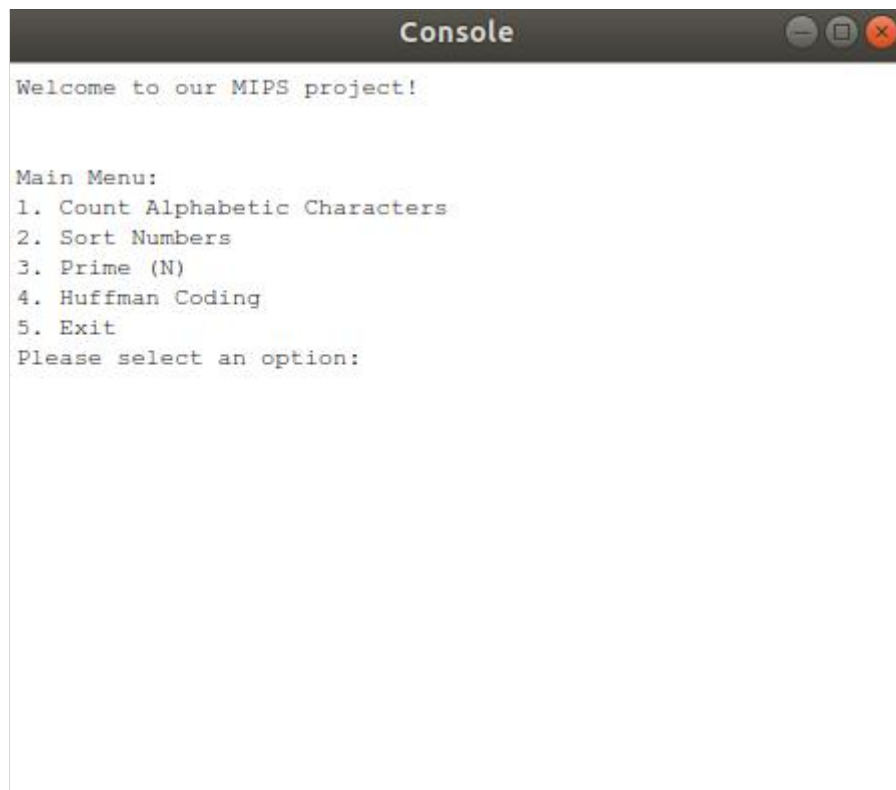
## Assignment 1 Report

**Sameeh Kunbargi**  
150119693

**Alper Dokay**  
150119746

**Ali Reza Ibrahimzada**  
150119870

The main menu feature of our project is placed in an infinite loop, and the only termination condition is when the user enters 5 (Exit option in the menu). We use a case-switch format in order to efficiently handle user inputs. That is, initially we have a variable initialized to 1, and at each step we compare the variable's value with the user input. If the user input equals the variable's value, then we execute the procedure from the corresponding main menu item. In case the variable's value does not match the user input, we simply increment the variable and try to compare again.



```
Console
Welcome to our MIPS project!

Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option:
```

Figure 1: Main Menu

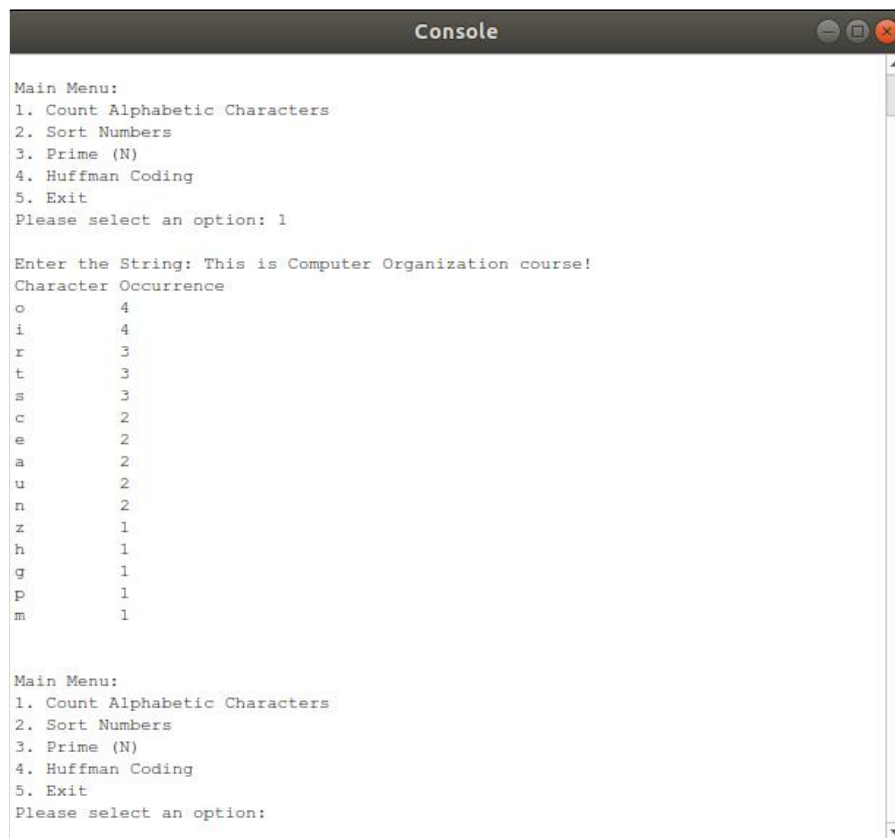
### Problem 1 - Occurrence Counter

The first problem is related to counting the occurrences of English alphabet characters (case insensitive) from a string, and then printing them in a sorted manner. When the user enters 1 in the main menu, the program asks for a string and uses that string as an argument to `q1` procedure. At the beginning of procedure, first we create some space in the stack in order to preserve the arguments, as well as saving the frequencies of

characters. In particular, we use 26 words (a word for each character in English alphabet) in stack to keep track of the character frequencies.

As a preprocessing step, we initialize the stack values with the ascii codes of the characters and take them to the upper 8-bits of the word. For instance, the first word on the top of the stack holds the character and frequency information related to the character 'a'. In the preprocessing step, that word is first initialized to 0x00000061, and then the ascii code value is taken to the upper 8-bits like 0x61000000. Later, we use the lower 24-bits of the word to save the frequency of the corresponding character from the string. The purpose of doing this is to keep the character value and frequency value in the same word.

After the preprocessing step, we start a loop to go over all characters in the string and increment their frequencies in stack. When doing this, we make necessary checks to skip punctuations, convert uppercase to lowercase, etc. Before incrementing the character's frequency, we mask out the upper 8-bits, but later we combine it again with the latest frequency. When the loop ends, we are sure that we have 26 words in the stack for each character in the alphabet. We simply apply a sorting algorithm (i.e., selection sort) on these words, and then print the characters from the stack in order of higher frequency to lower one.



```
Console
Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: 1

Enter the String: This is Computer Organization course!
Character Occurrence
o      4
i      4
r      3
t      3
s      3
c      2
e      2
a      2
u      2
n      2
z      1
h      1
g      1
p      1
m      1

Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option:
```

Figure 2: Character Occurrences in a Sorted Order

## Problem 2 - Sort Numbers

The second problem is about getting more than one number in only one input and then sorting the given set of numbers in ascending order. The way of doing it is to get them as string and process them iteratively to get each number separated by a space character. Once the preparation of these numbers has been done, it is time to sort those numbers in memory so as to print them in the order requested.

The strategy that we follow is starting with getting the ascii value of each character in the string until it sees a space or a null character. If the value is a dash ("-"), it stores 1 in a register for each substring to describe that the given consecutive numbers should be negative. Once it reads a character which is actually a number, we assign it to the stack. Whenever the input character is either space or null character, it starts reading the stack from bottom to top, meaning that numbers are converted to an integer from right to left. For example, let's consider the string 108. For this string to be converted into an integer, we firstly take them to the stack 1, 0, 8, respectively. Once it sees a null character, it reads the most right value, subtracts it from 48 to get the real integer value (as 48 ASCII value corresponds to the character 0) and multiplies it by the power of 10 including the power 0 which is actually 1 iteratively. After each multiplication, it keeps them in a total value to get the real integer at the end. Once the conversion is completed, it checks the register we store negativity status, if it is 1, we update the current integer with the value by subtracting itself from 0, else we keep it as it is. Then, the real integer value which is 108 in this case is stored in the stack instead of the first character address - here it is 1. We do this iteratively until the string is finished. At the end, the stack is actually storing an array which has the initial value at the stack's top address.

Right after the stack is ready to be sorted, we serve the stack address to the sorting part to sort those values in ascending order. Selection sort is selected to be used as the sorting algorithm, and we implemented it in MIPS language. As it has the complexity of  $n^2$ , we implemented it by using loops and some procedures in MIPS. The main result of sorting occurs in stack addresses. The addresses of values are interchanged in ascending order.

Lastly, a procedure for representing the last states is implemented. By using this procedure, the numbers are printed in ascending order to show the sorted version of the given numbers in a string.

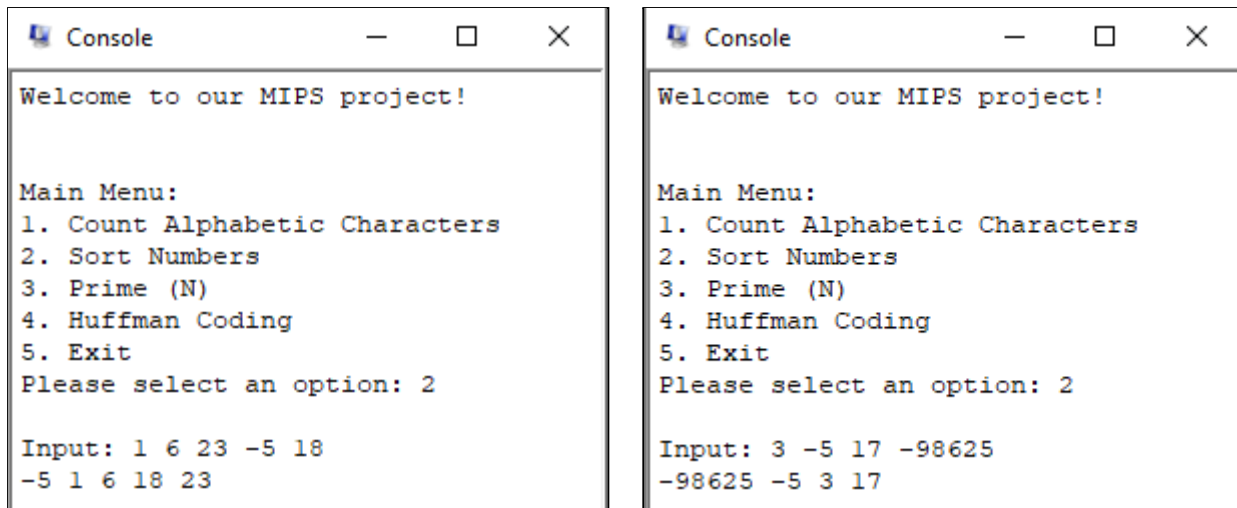


Figure 3-4: Example in the manual - execution with different inputs including -98625.

### Problem 3 - Prime (N)

The third problem is about finding the count of prime numbers between 2 and a given input N. We can use the Sieve of Eratosthenes algorithm to generate all prime numbers between 2 and N, then we count them and give the output. At the beginning of the procedure, we start with reserving space for used arguments and to save space for an array of size N which we are using in this algorithm. We initialize all the values with 1's ( True ), then we go on checking the numbers. If a number is not prime, we write 0's ( False ) for it and for all its multiples. Later in the end of the algorithm, we are left with an array of size N with True values only for Prime numbers. We then loop over that array and count how many prime numbers there are between 2-N. After the counting process has been finished successfully, we print the output on the screen.

**Note:** In this problem, we want to mention the limitations of machines/simulators. That is, as we increase the number of N in the problem, the stack size increases at runtime and on some machines with less memory, it throws an exception. But on some other machines the same implementation gives the correct output for large N. On the other hand, the same N on QTSpim and Mars simulators does not work at the same. For instance, on the same machine when N=500000, QTSpim throws an exception related to stack size, but Mars is able to give the correct output.

```
Console
Welcome to our MIPS project!

Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: 3

Enter N between 2 and 1,000,000
10

prime(10) is 4

Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: 3

Enter N between 2 and 1,000,000
100

prime(100) is 25

Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: |

Console
Welcome to our MIPS project!

Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: 3

Enter N between 2 and 1,000,000
1000

prime(1000) is 168

Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option: 3

Enter N between 2 and 1,000,000
10000

prime(10000) is 1229

Main Menu:
1. Count Alphabetic Characters
2. Sort Numbers
3. Prime (N)
4. Huffman Coding
5. Exit
Please select an option:
```

Figure 5-6: Successful run for prime(100) - prime(10000)

## Problem 4 - Huffman Coding

For this problem we only completed the first part which basically does character counting from the input string. We did not complete constructing the Huffman Code Tree.