CSCI 3301

Ivan Zelenkov

Abdullah Yasin Nur

May 1, 2022

**Programming Assignment Report**

Program for factorial of a number (*Facto.s*)

Factorial of a non-negative integer is the multiplication of all integers smaller than or equal to n. It can be calculated using the following recursive formula.

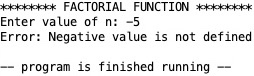
The program can calculate the factorial of numbers from 0 to 12 inclusive. For example, factorial of is which is . You can verify this by compiling and running my program and comparing the output with the following data.

Text

Description automatically generated with medium confidence

In addition, the code must be able to handle edge cases. An edge case is an issue that occurs at an extreme (maximum or minimum) operating parameter. These are the cases that are handled:

1. Negative factorial is not defined. Return an error message.



You can see lines 47 and 48 which contain the following code:

(set on less than immediate) will set to a register if value in register is less than . If we consider the case of user input , then the register is going to have value . Therefore the next instruction (branch if equal) will be true because and will branch on where it will output an error message *"Error: Negative value is not defined".*

1. After factorial, the output is larger than bits. Therefore, return out of boundary error message.

Text

Description automatically generated

You can see lines and which contain the following code:

First, there was created a register that contains the value Then, register which contains a value from the input will be compared with . (branch if greater than) is going to check if $a0 value is greater than $t1 value. If we consider the case of user input , then the register is going to have this value and it is greater than bound . Therefore, a branch will be taken and the error message *"Error: Result is larger than 32 bits"* will be shown.

Text

Description automatically generated

You can see line which contains the following code:

Register will contain value and be returned to the user as output if the register equals 1 at line 52.

**Text

Description automatically generated**

The code above shows that there are many checks of value in register done. From lines to was checked if the input value is negative, from lines to if the input value is greater than , and you can see that from line to value is also checked if it is less than . If we consider the case of user input , then all the checks above the line will be false, therefore the range of values that are left to check for lines to is from to , and we can be sure that the register at line has a value of , of course, if the input value was . Otherwise, the register has a value from to , which will branch to else block of code.

**` Conclusion:** computing large factorials is a more interesting problem. There are practical issues that arise on the border between ordinary and bignum arithmetic calculations. This program works efficiently and covers all important edge cases. The improvement of this program is that instead of throwing an error, it immediately checks the input value from the user. For example, if the user input value of 50, then the program would calculate until 12 and then threw an error. This process is useless. Instead, it immediately validates the entered value and responds very quickly to the user without a long wait. Therefore, the responsiveness of this program is very high.

Program for sorting a set of given integers (*Sort.s*)