1. **Title:** Design a stack machine, its instruction set must be stack oriented (no register!)
2. **Objective:** Main goal of this project is to build a stack machine, which won’t use any type of register/accumulator. The system will be totally based on stack. Stack is a data structure, which basically use the LIFO (Last In First Out) process. As this system is not allowed to use any register, so the value will be directly inserted into the stack and by the rule of stack machine, it will complete its operations like, sum, subtraction, multiplication etc.
3. **Theory:** Push(), Pop(), top() are the main operations to add values and remove values. Push() is used for adding any value and pop() is used for deleting any value from the top. Top() function is used for fetching the first element of the stack. This stack machine will calculate the values based on the sign between the given numbers.It will work first for the first coming values and so on. Finally, only a single value will be stored in the stack and that will be the result. For example: **7 + 5 = ?**

Steps:

1. Push 🡪 7
2. Got the sign (+)
3. Push 🡪 5
4. Fetch the top 🡪 5 (Stored in a variable)
5. Pop 🡪 5
6. Fetch the top 🡪 7 (Stored in a variable)
7. Add 🡪 5 + 7 = 12
8. Push 🡪 12 (Which is the required result)
9. **Design:**

Algorithm:

**stackMachine(String[], arraySize n)**

**Step-1:** Start.

**Step-2:** Starts the iteration from the first index of the String to the last index n.

**Step-3:** If the element of the String is an operand, then push it into the Stack.

**Step-4:** If the current element is an operator, then go for the next operand and push it into the Stack.

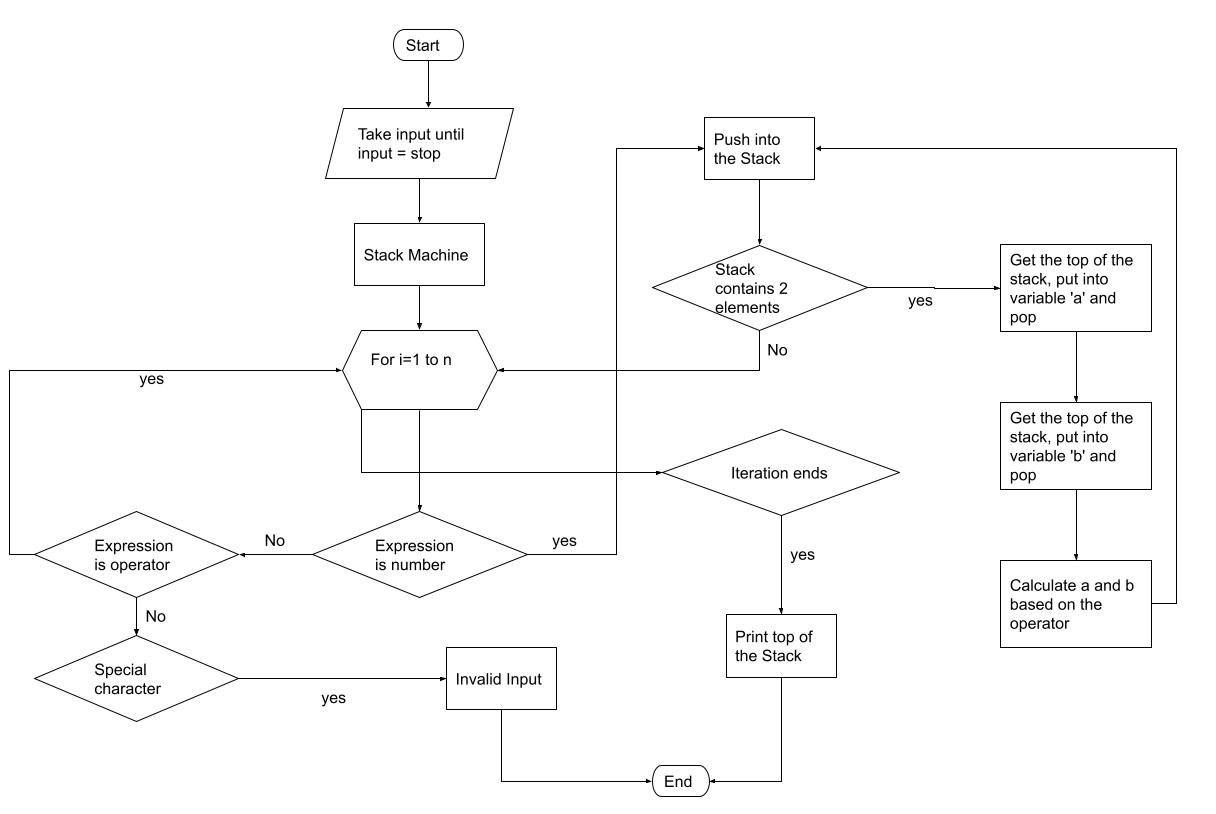
**Step-5:**  If there are 2 elements pushed successfully in the Stack, then get the top element of the Stack and store it into a variable (A) and pop it. Again get the top of the stack and put it into a variable (B) and pop it.

**Step-6:**  Now calculate both the variable A and B based the sign of the operator.

**Step-7:**  After the calculation, push it back to the Stack. Continue this process, until the iteration ends.

**Step-8:**  End.

Flow Chart:



1. **Implementation:**

Functional Modules of Code:

We have implemented a function named stackMachine. This is used for the main calculation of the Stack Machine. We didn’t use the built in stack in the code. We have created the Stack functions like- top(), pop(), push() etc by using data structure.

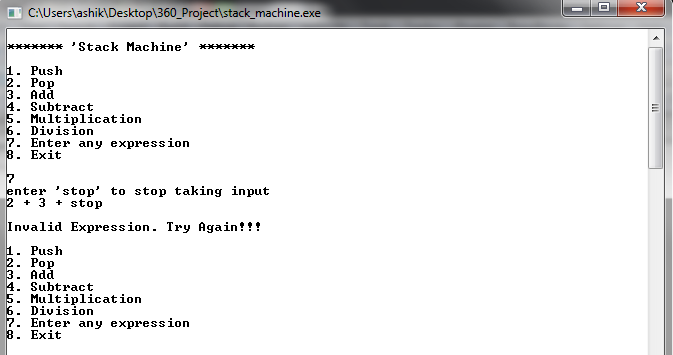
Built in Documentation: As our program can take multi digit numbers as input, that’s why we have taken the input as String. To handle the string we have use a built in function, which is stod(). It converts the String to double. So that, we can calculate the values.

System Requirements:

* Compiler: CodeBlocks, or any other compiler which can execute C++.
* RAM: 4 GB
* Operating System: Windows 7/8/10

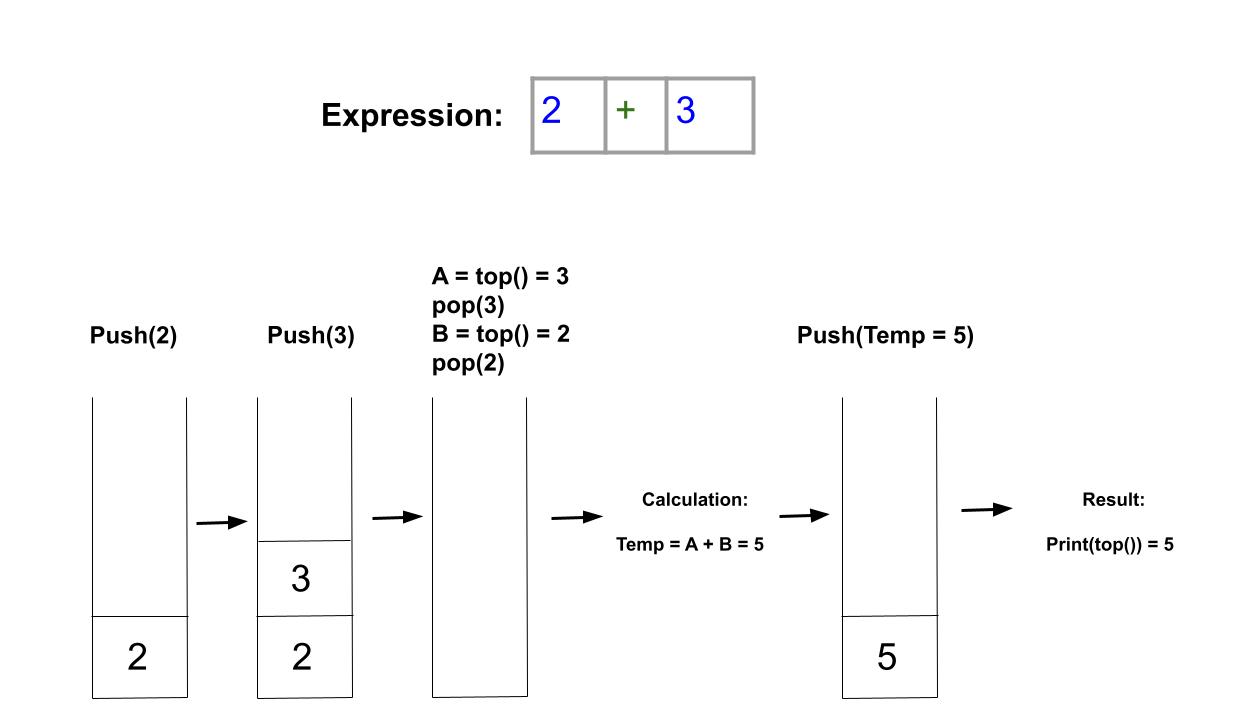
1. **Debugging-Test-Run:** We have noticed that in many of the online sources, they make the programme only for the single digit numbers. But We have done something better from this. Our program can deal with the multi digit numbers like- 12, 100 and could be any real number. Our Program has been Debugged very nicely and it can detect wrong expressions/invalid expression.

Invalid Expression:

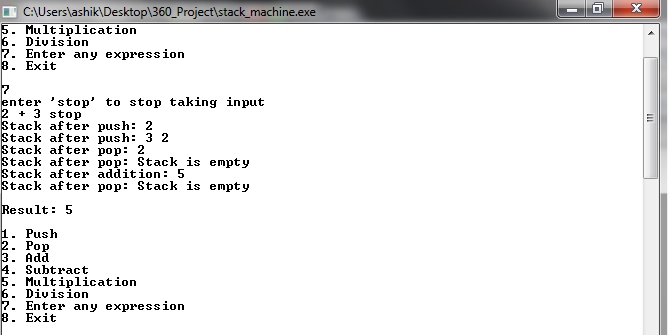


1. **Results Analysis:**

Example-1:



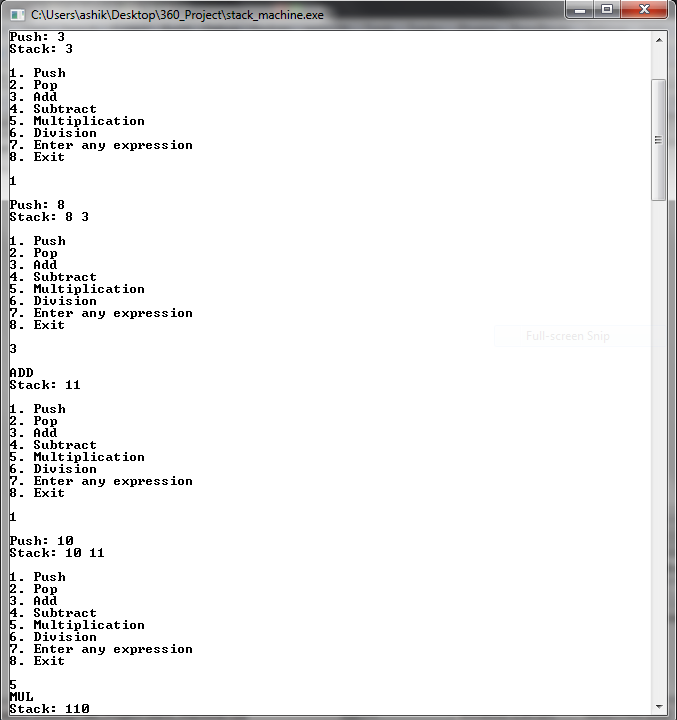
Outputs:



Example-2:

|  |  |
| --- | --- |
| **Instruction** | **Stack** |
| PUSH 3 | 3 |
| PUSH 8 | 8, 3 |
| ADD | 11 |
| PUSH 10 | 10, 11 |
| MUL | 110 |

Output:



Time Complexity:

Time Complexity of Stack machine is O(n). Because we used single loop all over the code, and there is no nested loop present in the stackMachine function.

The array size is n, so the iteration will be active from 1 to n. We have implemented the push(), pop(), top() etc functions by own self.

There is no loop present in those mentioned functions except print(). So the time complexity for these functions will be O(1), which is constant.

1. **Conclusion and Future Improvements:** As per the title of the project, we didn’t use any register to design the stack machine. This was little bit challenging. But we have learned a lot from here that, how the stack machine works and what is the working process of it.

Future Improvements:

* This program can’t understand that; which operator should be work first. As it works with the system of Stack, it works sequentially. In future we will try to solve it.
* This program can’t handle with special character like comma, percentage sign, dollar sign and others. So, we will try to debug it more efficiently, which will be able to detect those.
* It can’t deal with alphabets. In the future we will make this program comfortable with alphabets also.

1. **Bibliography:**

[1] Tiwari, Ayushman. (2019). What is the time complexity of the push and pop operation of an array-based stack?. *Quora*. Available at: <https://www.quora.com/What-is-the-time-complexity-of-the-push-and-pop-operation-of-an-array-based-stack> [Accessed Day: 25 May 2021].

[2] Year Published: 2020. Stack machine in Computer Organisation. *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/stack-machine-in-computer-organisation/> [Accessed Day: 27 May 2021].

[3] Year Published: 2021. Stack machine. *Wikipedia*. Available at: <https://en.wikipedia.org/wiki/Stack_machine> [Accessed Day: 28 May 2021].

# [4] Year Publilshed: 2020. Compiler Design Module 51 : Stack Machine. *YouTube.* Available at:[*https://www.youtube.com/watch?v=eYk0EennUrA*](https://www.youtube.com/watch?v=eYk0EennUrA) *[Accessed Day: 29 May 2021 ]*