EAST WEST UNIVERSITY

“Computer Architecture”

Course Code: **CSE360**

Semester: Spring 2024

Section: 01

Group: 11

Mini Project on

Stack machine ISA: Design a stack machine, its instruction set must be stack oriented. (No Register)

Submitted to

Dr. Md. Nawab Yousuf Ali

Professor

Department of Computer Science & Engineering

East West University

Submitted by:

Student ID	Student Name
2021-2-60-020 (Roll-21)	Kazi Minhajul Goni Sami
2021-2-60-015 (Roll-19)	Syed Musayedul Hussain
2021-2-60-136 (Roll-38)	Md. Sajjad Hossain

Date of Submission: 28 -05-2024

Table of Contents:

Introduction:

In computer science, computer engineering and programming language implementations, a stack machine is a computer processor or a virtual machine in which the primary interaction is moving short-lived temporary values to and from a push down stack. In the case of a hardware processor, a hardware stack is used. The use of a stack significantly reduces the required number of processor registers. Stack machines extend push-down automaton with additional load/store operations or multiple stacks and hence are Turing-complete.

Problem Definition:

Design a stack machine, its instruction set must be stack oriented (no register!)

Objectives:

Main goal of this project is to build a stack machine, which will not use any type of register/accumulator. The system will be totally based on stack. Stack is a data structure, which uses the LIFO (Last in First Out) process. As this system isn't allowed to use any register, 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.

Methodology:

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: **9 + 8 =?** Steps:

1. Push 9
2. Got the sign (+)

3. Push 8
4. Fetch the top 8 (Stored in a variable)
5. Pop 8
6. Fetch the top 9 (Stored in a variable)
7. Add 8+9 = 17
8. Push 17 (Which is the required result)

Algorithm:

Stack Machine (String [ ], array Size n)

Step-1: Start.

Step 2: Take user input until input = stop.

Step 3: Loop for iteration from the first index of the string to the last index n.

Step 4: if the element of the string is operand push it into stack.

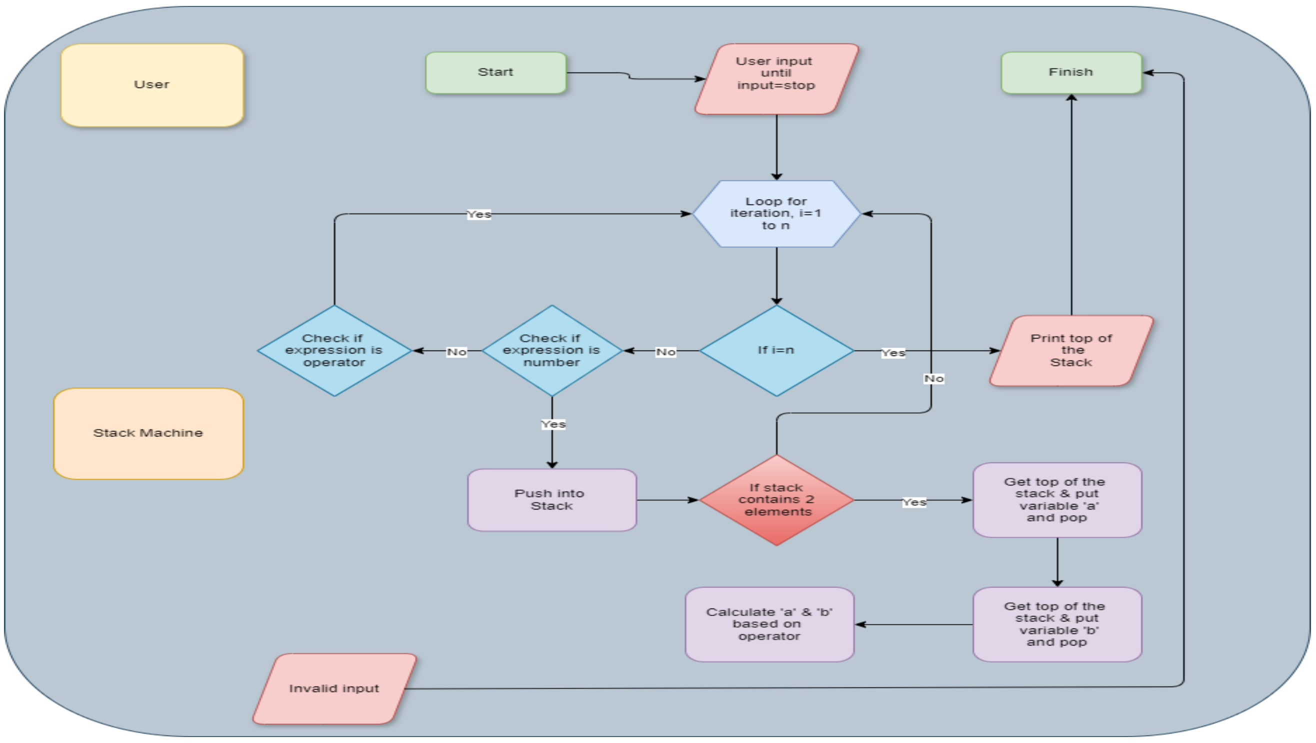
Step 5: If the element is operator, then go for the next element to check the operand and then push it into the stack.

Step 6: If there are 2 elements successfully pushed on the stack, then calculate them with the operator.

Step 7: Then pop the top of the element, which is result, and print top of the stack.

Step 8: Finish.

Flow Chart:


User
Start
User input until input=stop
Finish
Loop for iteration, i=1 to n
Yes
Check if expression is operator
No
Check if expression is number
No
If i=n
Yes
Print top of the Stack
No
Yes
Stack Machine
Push into Stack
If stack contains 2 elements
Yes
Get top of the stack & put variable 'a' and pop
Calculate 'a' & 'b' based on operator
Get top of the stack & put variable 'b' and pop
Invalid input

Code (C++ File):

```
#include<bits/stdc++.h>
#include<math.h>
#include<stdlib.h>
#include<iostream>
#include<stack>
using namespace std;
class Node
{
public:
    double* key;
    Node* next;
    Node()
    {
        next=NULL;
    }
    Node(double value)
    {
        key=new double(value);
        next=NULL;
    }
};
class Stack
{
public:
    Node* head;
    Stack()
    {
        head=NULL;
    }
    void push(double key);
    void pop();
    double* top();
    void print();
    bool empty();
    int size();
};
void Stack::push(double key)
{
    Node* node = new Node(key);
    node->next = head;
    head = node;
}
void Stack::pop()
{
    if(head==NULL)
    {
        cout << "Stack is empty" << endl;
        return;
    }
    Node* temp = head;
    head = head->next;

    delete temp;
}
double* Stack::top()
{
    return head->key;
}
bool Stack::empty()
{
    if(head==NULL)
        return true;
    else
        return false;
}
void Stack::print()
{
    Node* p=head;
    if(p==NULL)
    {
        cout << "Stack is empty" << endl;
        return;
    }
    while(p!= NULL)
    {
        cout <<*(p->key)<< " ";
        p = p->next;
    }
    cout << endl;
}
int Stack::size()
{
    Node* p=head;
    int c=0;
    while(p!= NULL)
    {
        //cout <<*(p->key)<< " ";
        p = p->next;
        c++;
    }
    return c;
```

```
}
bool isOperator(string c)
{
    if(c=="+" || c=="-" || c=="/" || c=="*" || c=="^" || c== "(" || c== ")" || c== "{" || c== "}" || c==
"[" || c== "]")
    {
        return true;
    }
    else
    {
        return false;
    }
}
bool isValid(string exp[], int n)
{
    int a=2,b=0,c=0,d=0;
    for(int i=0; i<=n-1; i++)
    {
        if(!isOperator(exp[i]) && (a%2)==0)
        {
            a = a - 1;
        }
        else
        {
            a = a + 1;
        }
    }
    if(a==1)
    {
        return true;
    }
    else
    {
        //printf("\nInvalid Expression. Please Try Again!!!");
        return false;
    }
}
void stackMachine(string exp[],int n)
{
    Stack* temp = new Stack;
    for(int i=0; i<=n-1; i++)
    {
        if(!isOperator(exp[i]))
        {
            double d=stod(exp[i]);
            temp->push(d);
            cout<<"\nStack after push: ";
            temp->print();
        }
        else
        {
            i++;
            double c = stod(exp[i]);
            temp->push(c);
            cout<<"\nStack after push: ";
            temp->print();
            if(temp->empty())
            {
                cout<<"The Given Expression is not correct"<<endl;
                return;
            }
            double x= *(temp->top());
```

```
temp->pop();
cout<<"\nStack after pop: ";
temp->print();
if(temp->empty())
{
   cout<<"The Given Expression is not correct"<<endl;
   return;
}
double y= *(temp->top());
temp->pop();
cout<<"\nStack after pop: ";
temp->print();
if(exp[i-1]=="+")
{
   temp->push(y+x);
   cout<<"\nStack after addition: ";
   temp->print();
}
else if(exp[i-1]=="-")
{
   temp->push(y-x);
   cout<<"\nStack after subtraction: ";
   temp->print();
}
else if(exp[i-1]=="*")
{
   temp->push(y*x);
   cout<<"\nStack after multiplication: ";
   temp->print();
}
else if(exp[i-1]=="/")
```

```
          {
             temp->push(y/x);
             cout<<"\nStack after division: ";
             temp->print();
          }
          else if(exp[i-1]=="^")
          {
             temp->push(pow(y,x));
             cout<<"\nStack after ^: ";
             temp->print();
          }
       }
    }
    double result=*(temp->top());
    temp->pop();
    cout<<"\nStack after pop: ";
    temp->print();
    if(!temp->empty())
    {
       cout<<"The Given Expression is not correct"<<endl;
       return;
    }
    cout<<"\nResult: "<<result<<endl;
}
void menu()
{
    cout<<"\n1. Push\n";
    cout<<"2. Pop\n";
    cout<<"3. Add\n";
    cout<<"4. Subtract\n";
    cout<<"5. Multiplication\n";
    cout<<"6. Division\n";
    cout<<"7. Enter any expression\n";
    cout<<"8. Exit\n\n";
}
int main()
{
    cout<<"\n******* 'Stack Machine' *******\n";
    Stack* stack1 = new Stack;
    menu();
    int ch;
    double num;
    for(int i=0; ; i++)
    {
        cin>>ch;
        if(ch==1)
        {
            cout<<"\nPush: ";
            cin>>num;
            stack1->push(num);
            cout<<"Stack: ";
            stack1->print();
            menu();
        }
        else if(ch==2)
        {
            cout<<"\nPop Done\n";
            stack1->pop();
            cout<<"Stack: ";
            stack1->print();
            menu();
        }
```

```
else if(ch==3)
{
    if(stack1->size() > 1)
    {
        cout<<"\nADD\n";
        double top = *(stack1->top());
        stack1->pop();
        double add = top + *(stack1->top());
        stack1->pop();
        stack1->push(add);
        cout<<"Stack: ";
        stack1->print();
    }
    else
    {
        cout<<"\nAt Least Two Elements Required To Perform ADD!!!\n";
    }
    menu();
}
else if(ch==4)
{
    if(stack1->size()>1)
    {
        cout<<"\nSUB\n";
        double top = *(stack1->top());
        stack1->pop();
        double sub = *(stack1->top()) - top;
        stack1->pop();
        stack1->push(sub);
        cout<<"Stack: ";
        stack1->print();
        }
        else
        {
            cout<<"\nAt Least Two Elements Required To Perform SUB!!!\n";
        }
        menu();
    }
    else if(ch==5)
    {
        if(stack1->size()>1)
        {
            cout<<"\nMUL\n";
            double top = *(stack1->top());
            stack1->pop();
            double mul = top **(stack1->top());
            stack1->pop();
            stack1->push(mul);
            cout<<"Stack: ";
            stack1->print();
        }
        else
        {
            cout<<"\nAt Least Two Elements Required To Perform MUL!!!\n";
        }
        menu();
    }
    else if(ch==6)
    {
        if(stack1->size() > 1)
        {
            cout<<"\nDIV\n";
            double top = *(stack1->top());
            stack1->pop();
            double divi = *(stack1->top())/top;
            stack1->pop();
            stack1->push(divi);
            cout<<"Stack: ";
            stack1->print();
        }
        else
        {
            cout<<"\nAt Least Two Elements Required To Perform DIV!!!\n";
        }
        menu();
    }
    else if(ch==7)
    {
        stack<string>s1_stack;
        string st;
        double val;
        cout<<"\nEnter 'stop' to stop taking input\n"<<endl;
        while (true)
        {
            cin>>st;
            if(st=="stop")
            {
                break;
            }
            s1_stack.push(st);
        }
        int len=s1_stack.size();
        string expression[len];
```

```
            for(int i=len-1; i>=0; i--)
            {
                expression[i]= s1_stack.top();
                s1_stack.pop();
            }
            if(!isValid(expression, len))
            {
                printf("\nInvalid Expression. Try Again!!!\n");
                menu();
                continue;
            }
            else
            {
                stackMachine(expression,len);
                menu();
            }
        }
        else if(ch==8)
        {
            break;
        }
        else
        {
            cout<<"Invalid Input";
        }
    }
    return 0;
}
```

Implementation:

Functional Modules of Code:

We have implemented a function named Stack Machine. This is used for the main calculation of the Stack Machine. We did not 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 is 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 Requirement:

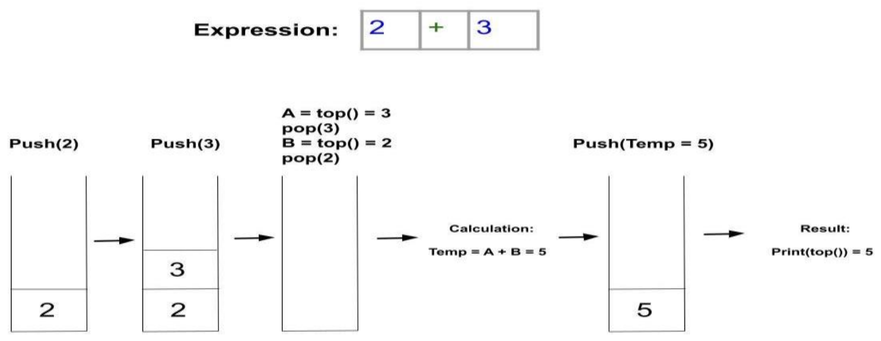
- Compiler: Code Blocks, or any other compiler, which can execute C++.
- RAM: 4 GB
- Operating System: Windows 7/8/10/11.

Debugging Test:

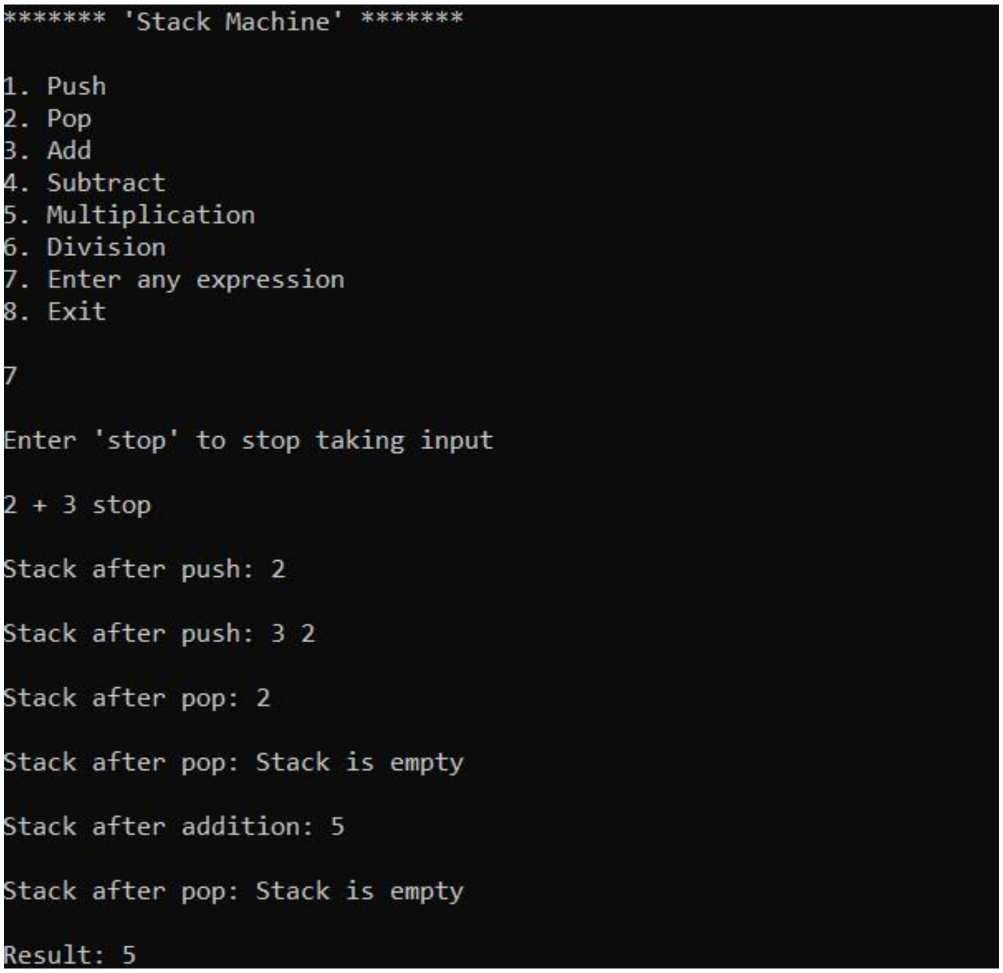
We have noticed that in many of the online sources, they make the program only for the single digit numbers. However, we have done something better from this. Our program can deal with the multi digit numbers like- 12, 100 and could be any real number. Our Program has been debugged very nicely and it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

Result Analysis:

Example 1**:**



Output:

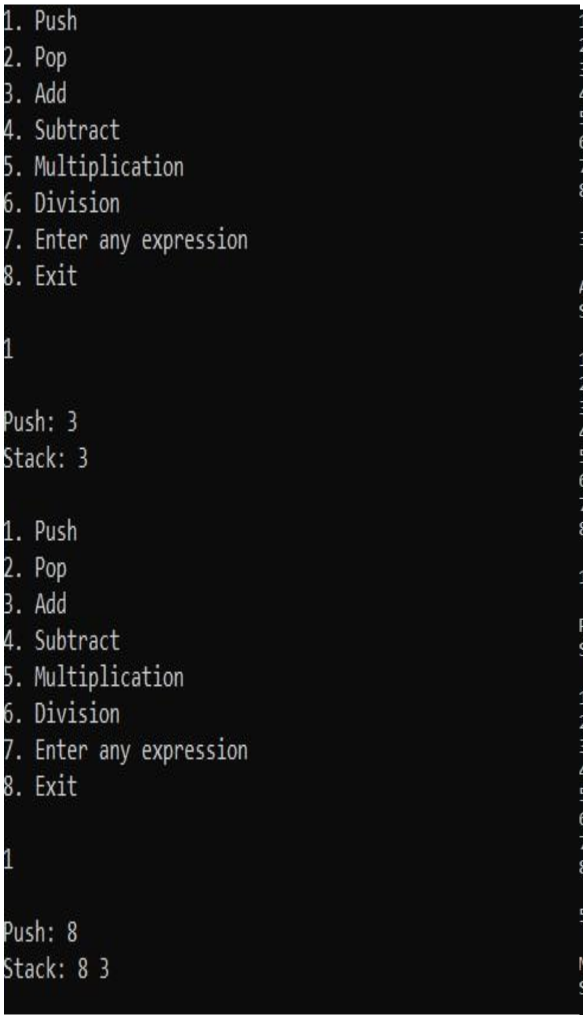


```
******* 'Stack Machine' *******

1. Push
2. Pop
3. Add
4. Subtract
5. Multiplication
6. Division
7. Enter any expression
8. Exit

7

Enter 'stop' to stop taking input

2 + 3 stop

Stack after push: 2

Stack after push: 3 2

Stack after pop: 2

Stack after pop: Stack is empty

Stack after addition: 5

Stack after pop: Stack is empty

Result: 5
```

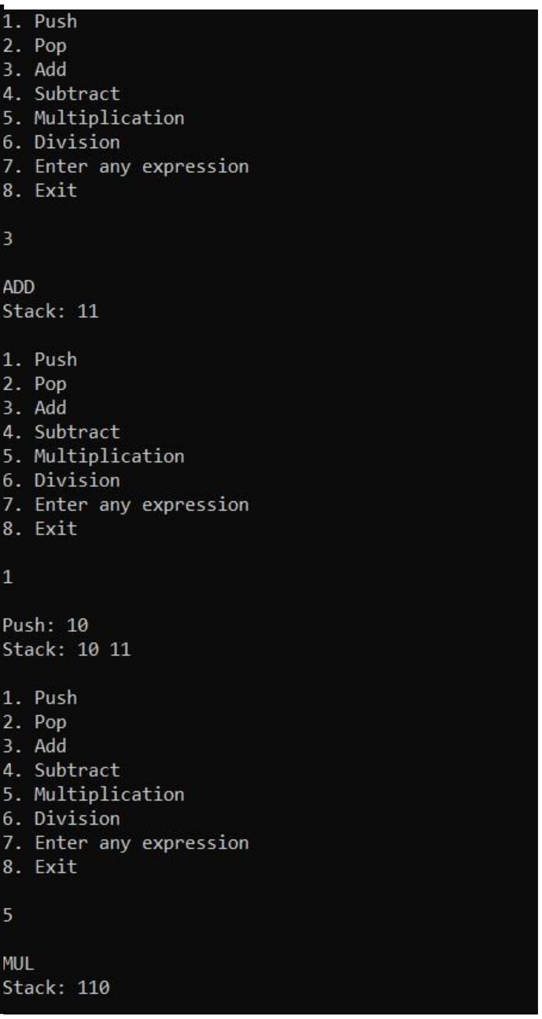
Example 2:

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

Output:



```
1. Push
2. Pop
3. Add
4. Subtract
5. Multiplication
6. Division
7. Enter any expression
8. Exit

1

Push: 3
Stack: 3

1. Push
2. Pop
3. Add
4. Subtract
5. Multiplication
6. Division
7. Enter any expression
8. Exit

1

Push: 8
Stack: 8 3
```



```
1. Push
2. Pop
3. Add
4. Subtract
5. Multiplication
6. Division
7. Enter any expression
8. Exit

3

ADD
Stack: 11

1. Push
2. Pop
3. Add
4. Subtract
5. Multiplication
6. Division
7. Enter any expression
8. Exit

1

Push: 10
Stack: 10 11

1. Push
2. Pop
3. Add
4. Subtract
5. Multiplication
6. Division
7. Enter any expression
8. Exit

5

MUL
Stack: 110
```

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 Stack Machine 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 (). Therefore, the time complexity for these functions will be O (1), which is constant.

Conclusion:

As per the title of the project, we did not use any register to design the stack machine. This was little bit challenging. However, we have learned a lot from here that, how the stack machine works and what is the working process of it.

Future Improvement:

- ❖ This program cannot 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 cannot handle with special character like comma, percentage sign, dollar sign and others. Therefore, we will try to debug it more efficiently, which will be able to detect those.
- ❖ It cannot deal with alphabets. In the future, we will make this program comfortable with alphabets also.

Reference:

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-timecomplexity-of-the-push-and-pop-operation-of-an-array-based-stack
2. Year Published : 2020. Stack machine in Computer Organization. *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/stack-machine-in-computer-organisation/
3. Year Published : 2021. Stack Machine. *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Stack_machine
4. Year Published: 2020. Compiler Design Module 51 : Stack Machine. *YouTube.* Available at: *https://www.youtube.com/watch?v=eYk0EennUrA*
5. From Wikipedia - Stack machine - Wikipedia.