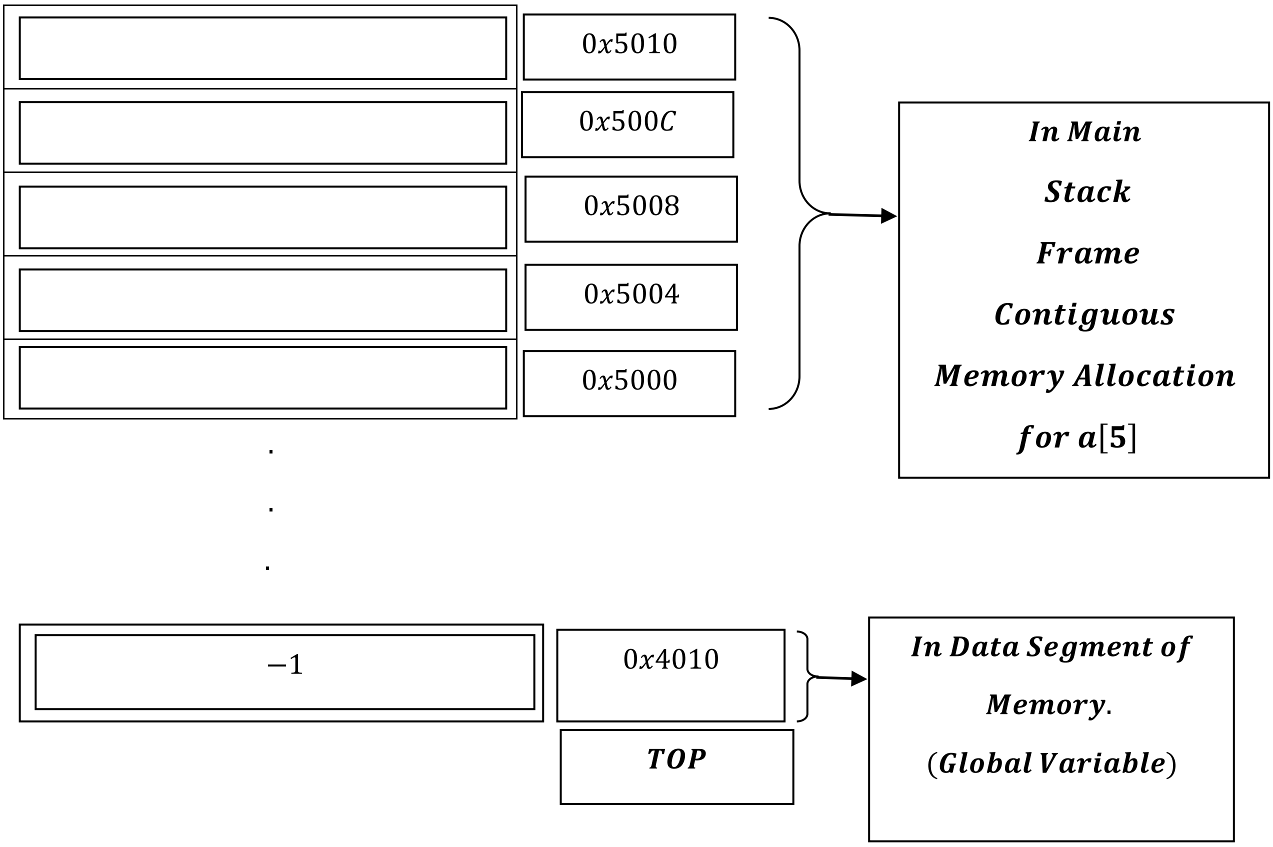
Stack Mechanism Discussion with Time Complexity

4. *Push Operation*

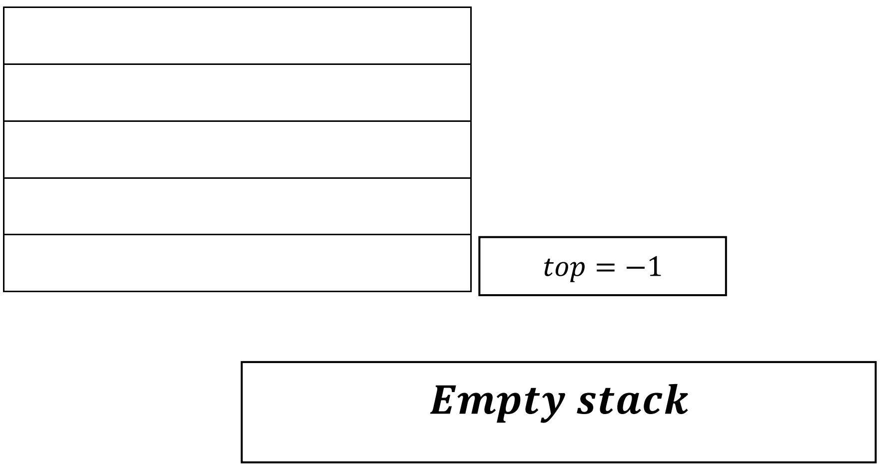
```
void push(int stack[], int item, int size)
{

    if (top == size - 1)
    {

        cout << "Stack Overflow" << endl;
        return;
    }

    top++;
    stack[top] = item;
}
….
       case 1:
            cout << "Enter the item to be pushed" << endl;
            cin >> item;
            push(stack, item, size);
            break;
```

$Say\ size = 5.$

Empty Stack



This is Physical Demonstration



This is Logical Demonstration

$As\ Stack\ is\ now\ empty\ , hence\ top = -1\ \neq size - 1\ , therefore:$

$Top = Top + 1 = \ -1 + 1 = 0.$

$Stack[Top = 0] = item.$

$\Longrightarrow Stack + 0 = item.\ [\ Stack + 0\ represents\ contiguous\ memory\ allocation]$

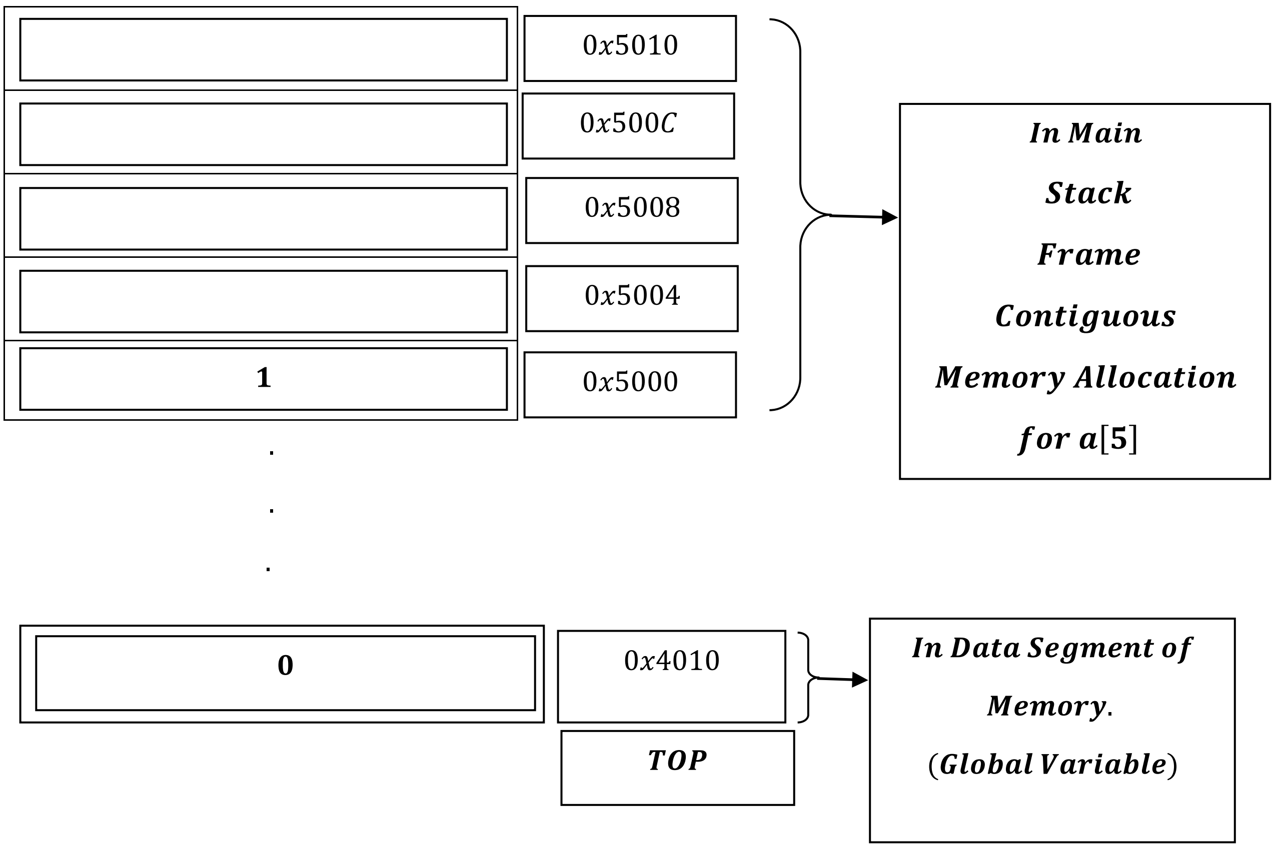
$\Longrightarrow Base\ Address + 0[index] \times 4bytes = item.$

$\Longrightarrow 0x5000 + 0 = item.$

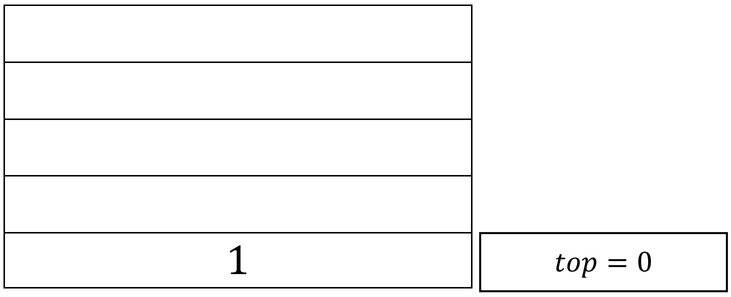
$\Longrightarrow 0x5000 = item.$

$Let, item = 1.$

$Push(1)$



$This\ is\ Physical\ Demonstration$



$\boldsymbol{Push(1)}$

This is Logical Demonstration

$\boldsymbol{Now, top = 0 \neq size - 1, therefore:}$

$\boldsymbol{Top = Top + 1 = 0 + 1 = 1.}$

$\boldsymbol{Stack[Top = 1] = item.}$

$\boldsymbol{\Rightarrow Stack + 1 = item.}$ [$\boldsymbol{Stack + 1 \text{ represents contiguous memory allocation}}$]

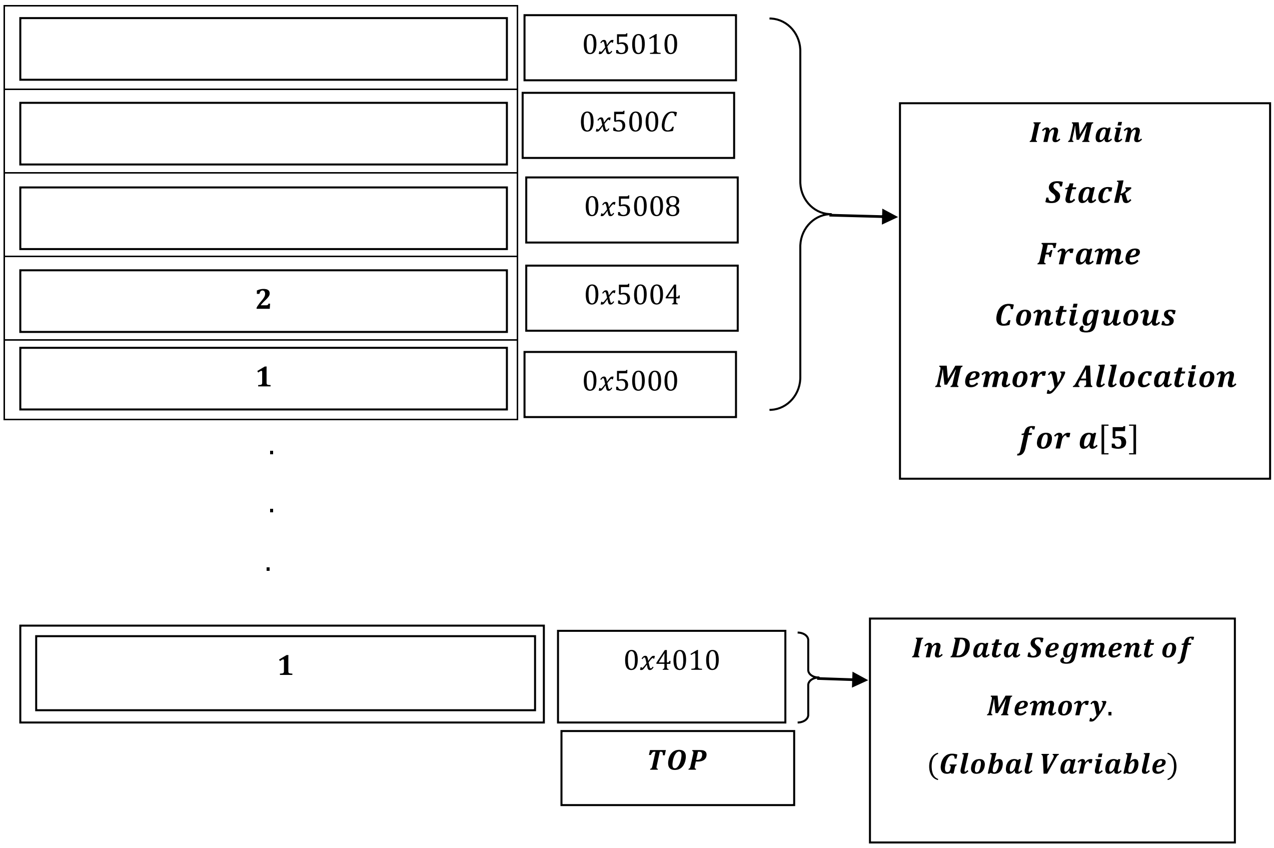
$\boldsymbol{\Rightarrow Base\ Address + 1[index] \times 4bytes = item.}$

$\boldsymbol{\Rightarrow 0x5000 + 4 = item.}$

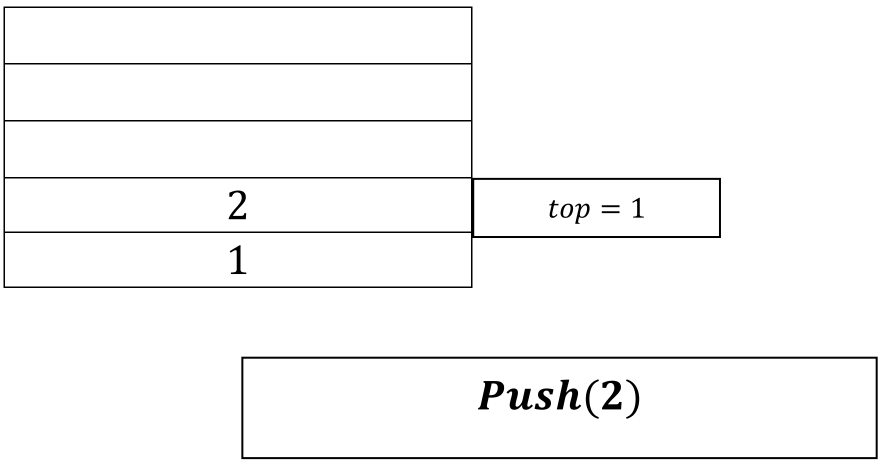
$\boldsymbol{\Rightarrow 0x5004 = item.}$

$\boldsymbol{Let, item = 2.}$

$\boldsymbol{Push(2)}$



This is Physical Demonstration



This is Logical Demonstration

$Now, top = 1 \neq size - 1, therefore:$

$Top = Top + 1 = 1 + 1 = 2.$

$Stack[Top = 2] = item.$

$\Longrightarrow Stack + 2 = item.$ [$Stack + 2, represents\ contiguous\ memory\ allocation$]

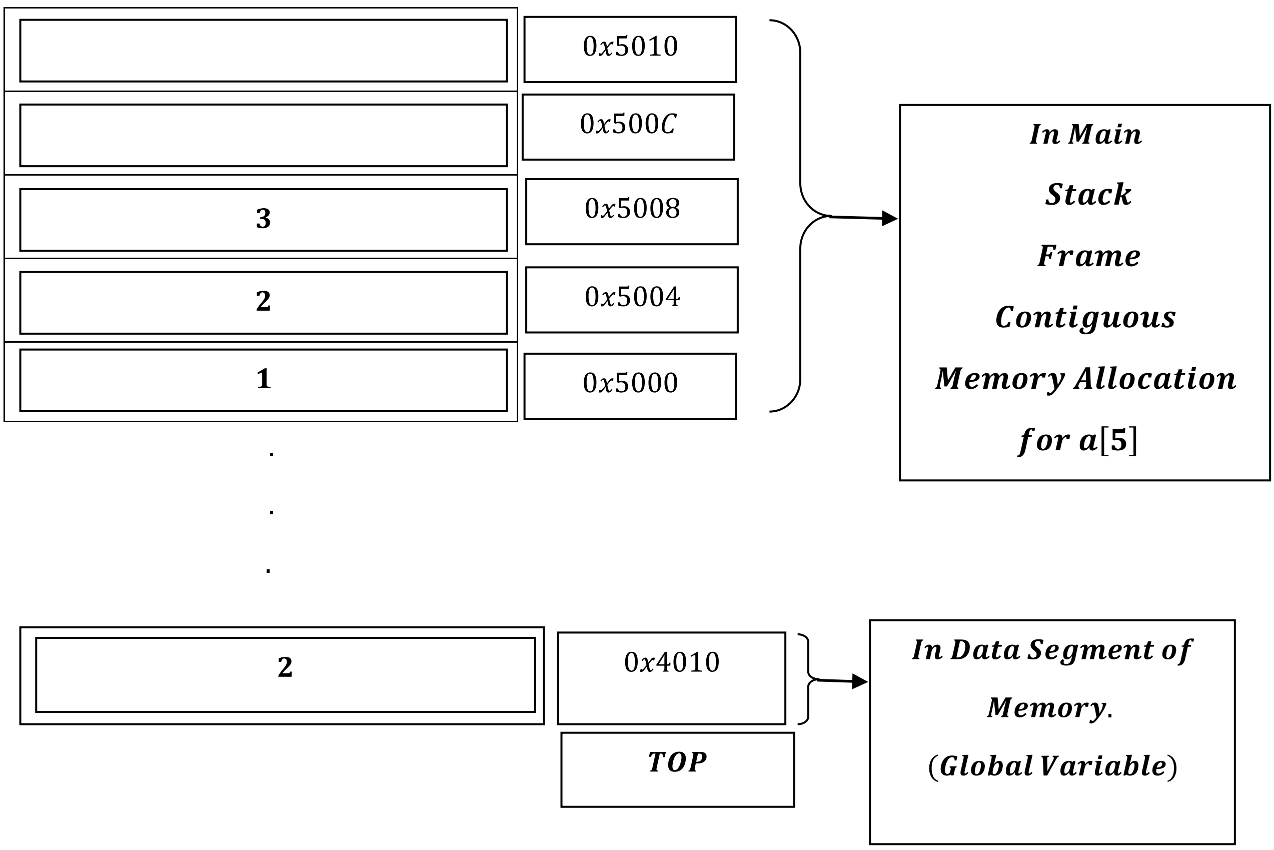
$\Longrightarrow Base\ Address + 2[index] \times 4bytes = item.$

$\Longrightarrow 0x5000 + 8 = item.$

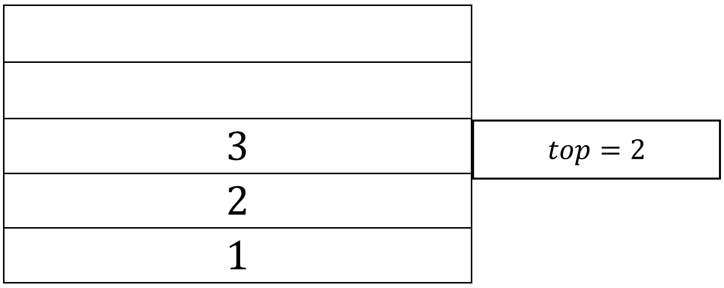
$\Longrightarrow 0x5008 = item.$

$Let, item = 3.$

$Push(3)$



$This\ is\ Physical\ Demonstration$



$$\boldsymbol{Push(3)}$$

This is Logical Demonstration

$Now, top = 2 \neq size - 1, therefore:$

$Top = Top + 1 = 2 + 1 = 3.$

$Stack[Top = 3] = item.$

$\Longrightarrow Stack + 3 = item.$ [$Stack + 3, represents\ contiguous\ memory\ allocation$]

$\Longrightarrow Base\ Address + 3[index] \times 4bytes = item.$

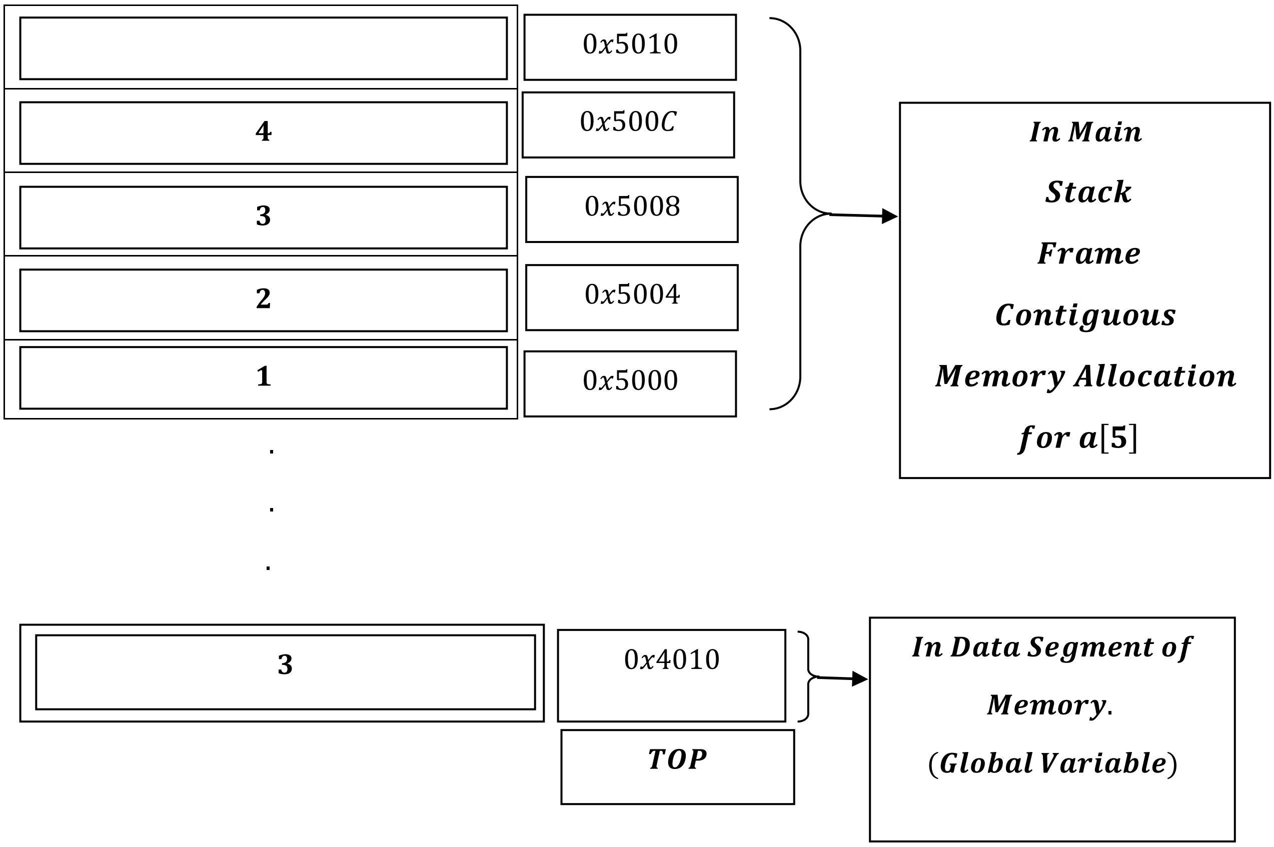
$\Longrightarrow 0x5000 + 12 = item.$

$\Longrightarrow 0x5000 + C = item. [12_{10} \approx C_{16}]$

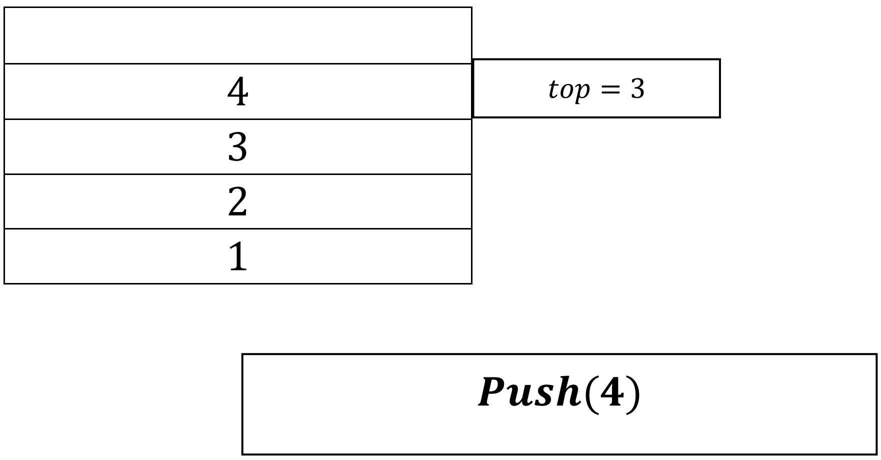
$\Longrightarrow 0x500C = item.$

$Let, item = 4.$

$Push(4)$



This is Physical Demonstration



This is Logical Demonstration

$Now, top = 3 \neq size - 1, therefore:$

$Top = Top + 1 = 3 + 1 = 4.$

$Stack[Top = 4] = item.$

$\Longrightarrow Stack + 4 = item.$ $[Stack + 4, represents\ contiguous\ memory\ allocation]$

$\Longrightarrow Base\ Address + 4[index] \times 4bytes = item.$

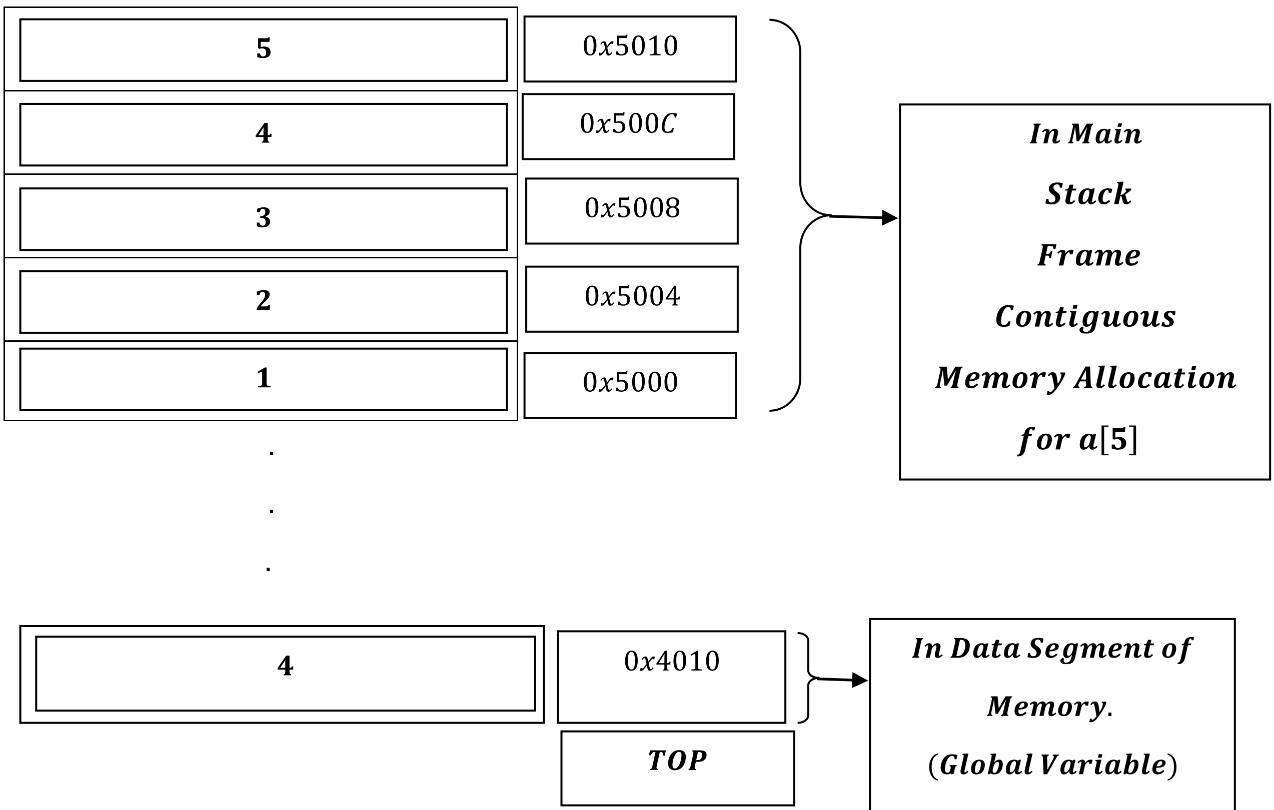
$\Longrightarrow 0x5000 + 16 = item.$

$\Longrightarrow 0x5000 + 10 = item. [16_{10} \approx 10_{16}]$

$\Longrightarrow 0x5010 = item.$

$Let, item = 5.$

$Push(5)$



$This\ is\ Physical\ Demonstration$

5	$top = 4$
4	
3	
2	
1	

$$\boldsymbol{Push(5)}$$

$\boldsymbol{This\ is\ Logical\ Demonstration}$

$\boldsymbol{Now, top = 4 = size - 1\ , is\ true\ , hence}$:

$\boldsymbol{Output}$: ``$\boldsymbol{Stack\ is\ Full}$``.

Time Complexity

```
void push(int stack[], int item, int size)
{

    if (top == size - 1)
    {

        cout << "Stack Overflow" << endl;
        return;
    }

    top++;
    stack[top] = item;
}
```

$\rightarrow$ ***Function overhead or stack frame creation when push() is called takes constant time `c` takes*** $\boldsymbol{O(1)}$.

$\rightarrow$ $\boldsymbol{if\ (top = size - 1) True\ [Takes\ constant\ `c`\ time : O(1)] then}$:

$$\rightarrow \boldsymbol{Output: ``Stack\ Overflow``} \begin{bmatrix} \boldsymbol{Takes\ constant\ `c`\ time:} \\ \boldsymbol{O(1)} \end{bmatrix}$$

$$\rightarrow \boldsymbol{return\ void\ and\ exit.} \begin{bmatrix} \boldsymbol{Takes\ constant\ `c`\ time:} \\ \boldsymbol{O(1)} \end{bmatrix}$$

$\rightarrow$ $\boldsymbol{if\ (top = size - 1) False\ then}$:

$$\rightarrow \boldsymbol{Top = Top + 1} \begin{bmatrix} \boldsymbol{Takes\ constant\ `c`\ time:} \\ \boldsymbol{O(1)} \end{bmatrix}$$

$$\rightarrow \boldsymbol{stack[top] = item;} \begin{bmatrix} \boldsymbol{Takes\ constant\ `c`\ time:} \\ \boldsymbol{O(1)} \end{bmatrix}$$

$\boldsymbol{If\ true\ then}$:

$$\boldsymbol{Time\ Complexity} = \boldsymbol{O}(\mathbf{1}) + \left(\boldsymbol{O}(\mathbf{1}) + \left(\boldsymbol{O}(\mathbf{1}) + \boldsymbol{O}(\mathbf{1})\right)\right) = \boldsymbol{O}(\mathbf{1}).$$

$\boldsymbol{If\ false\ then}$:

$$\boldsymbol{Time\ Complexity} = \boldsymbol{O}(\mathbf{1}) + \left(\boldsymbol{O}(\mathbf{1}) + \left(\boldsymbol{O}(\mathbf{1}) + \boldsymbol{O}(\mathbf{1})\right)\right) = \boldsymbol{O}(\mathbf{1}).$$
