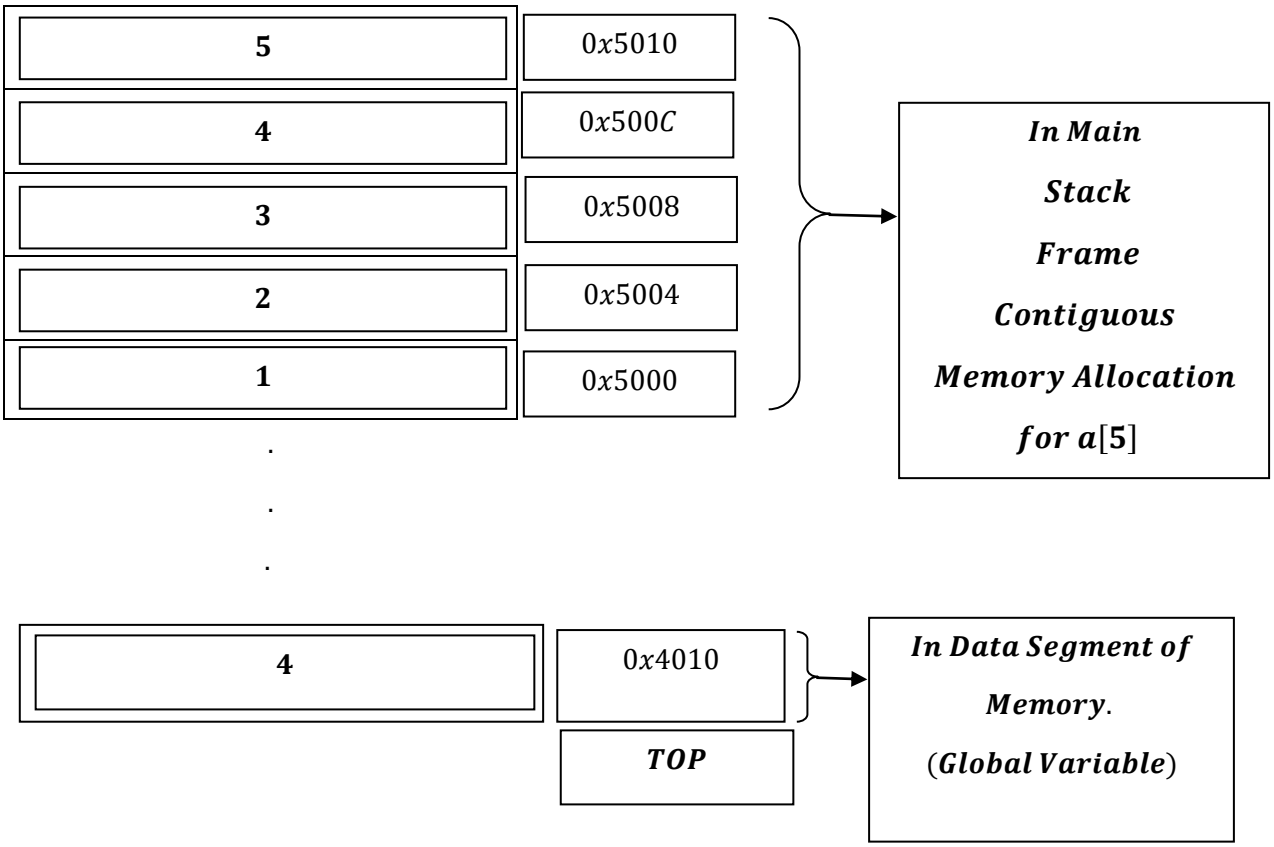
Stack Mechanism Discussion with Time Complexity

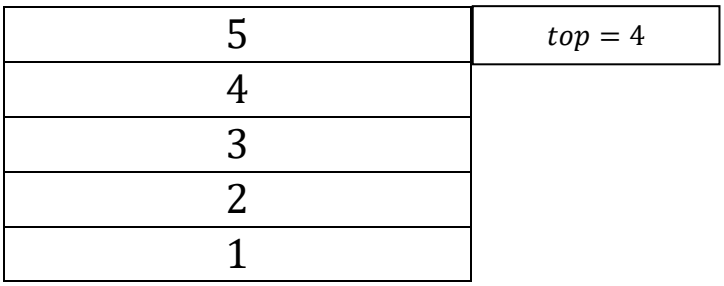
5. *Pop Operation*

```
void pop(int stack[])
{

    if (top == -1)
    {

        return;
    }
    cout << "Item popped" << stack[top] << endl;
    top--;
}
….
        case 2:
            if (top == -1)
            {

                cout << "Stack Overflow" << endl;
            }
            else
            {
                pop(stack);
            }
            break;
```

Full Stack



This is Physical Demonstration



Full Stack

This is Logical Demonstration

$\boldsymbol{Stack[Top = 4]}$.

$\boldsymbol{\Longrightarrow Stack + 4.}$ [$\boldsymbol{Stack + 4, represents\ contiguous\ memory\ allocation}$]

$\boldsymbol{\Longrightarrow Base\ Address + 4[index] \times 4bytes.}$

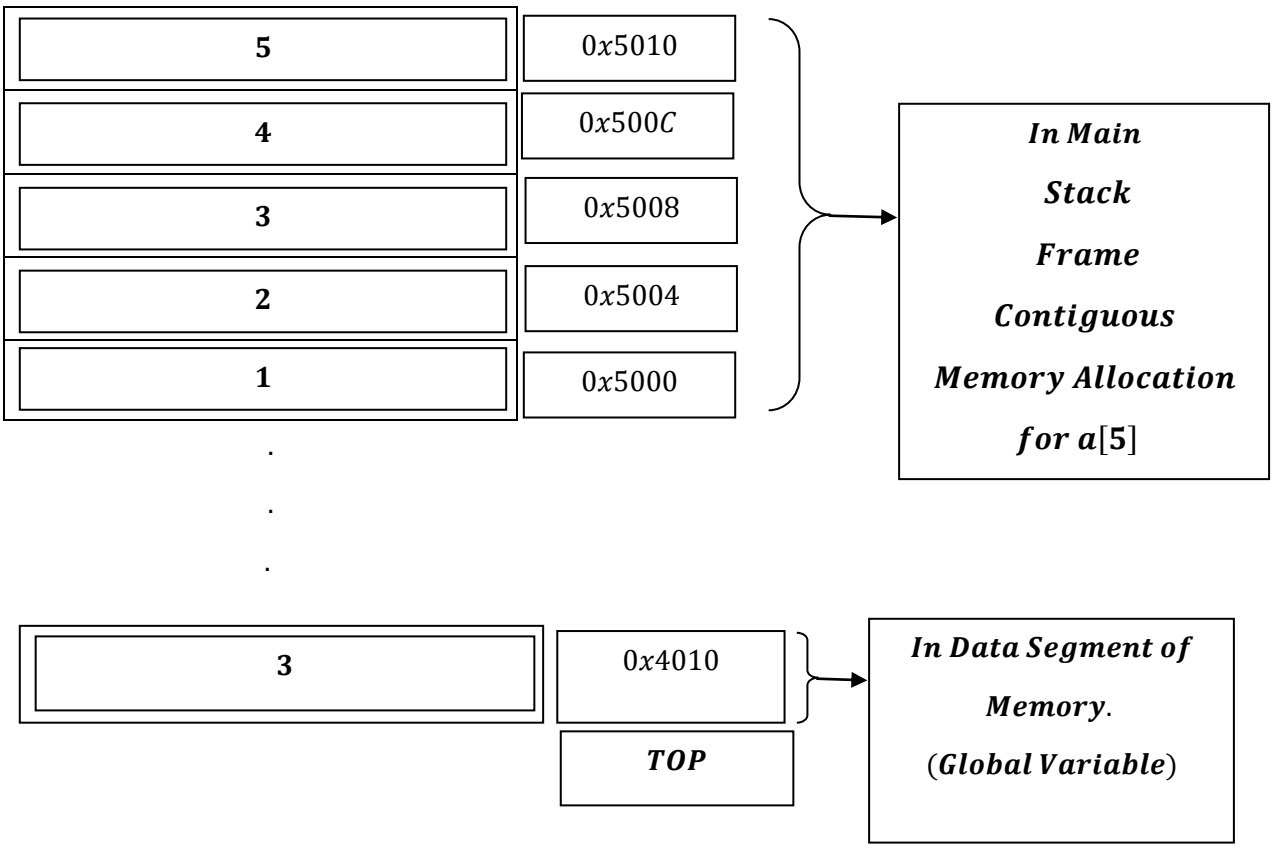
$\boldsymbol{\Longrightarrow 0x5000 + 16.}$

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

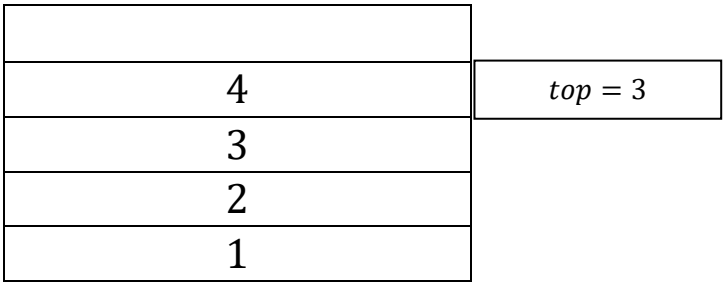
$\boldsymbol{\Longrightarrow 0x5010}$

***Output*: *Item popped ``the value stored at Address*: $\boldsymbol{0x5010}$: $\boldsymbol{5}$``.**

$\boldsymbol{Top - -\Longrightarrow Top = Top - 1 \Longrightarrow Top = 4 - 1 = 3.}$



This is Physical Demonstration



Pop(5)

This is Logical Demonstration

$\boldsymbol{Stack[Top = 3].}$

$\boldsymbol{\Longrightarrow Stack + 3.}$ [$\boldsymbol{Stack + 3, represents\ contiguous\ memory\ allocation}$]

$\boldsymbol{\Longrightarrow Base\ Address + 3[index] \times 4bytes.}$

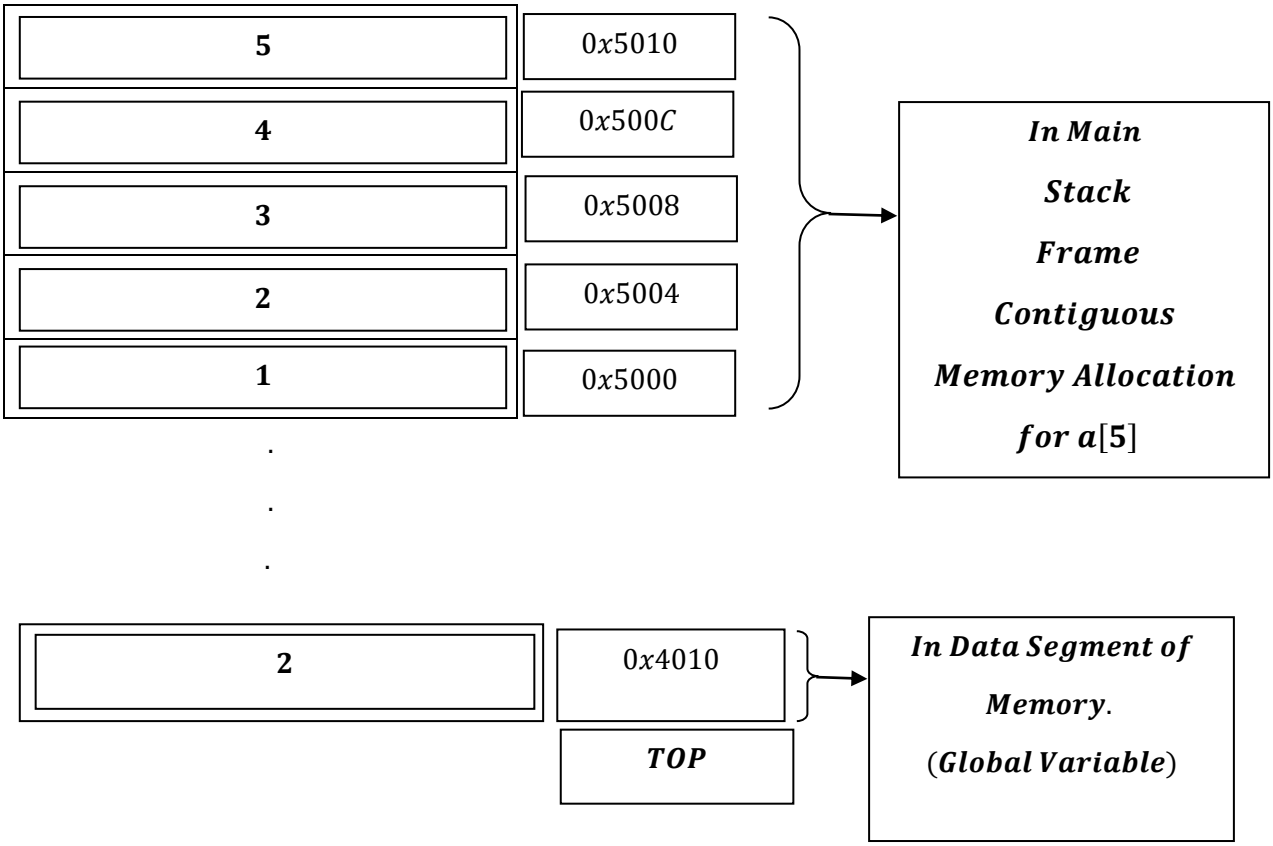
$\boldsymbol{\Longrightarrow 0x5000 + 12.}$

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

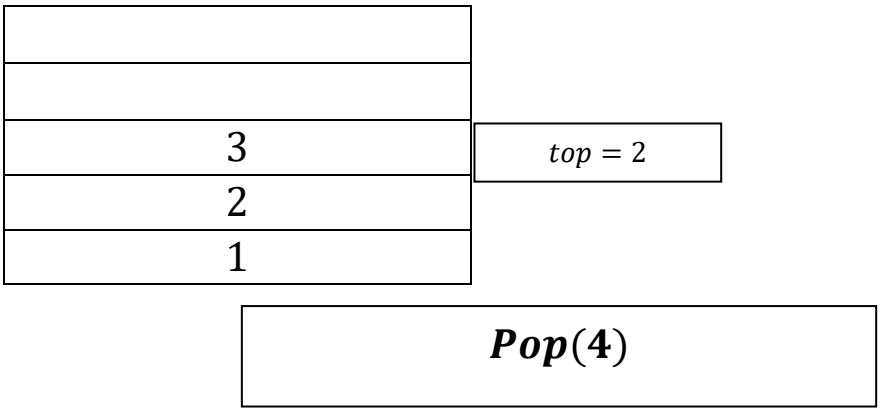
$\boldsymbol{\Longrightarrow 0x500C.}$

***Output*: *Item popped* ``*the value stored at Address*: $\boldsymbol{0x500C}$: $\boldsymbol{4}$``.**

$\boldsymbol{Top - - \Longrightarrow Top = Top - 1 \Longrightarrow Top = 3 - 1 = 2.}$



This is Physical Demonstration



This is Logical Demonstration

$\boldsymbol{Stack[Top = 2]}$.

$\boldsymbol{\Longrightarrow Stack + 2}$. [$\boldsymbol{Stack + 2, represents\ contiguous\ memory\ allocation}$]

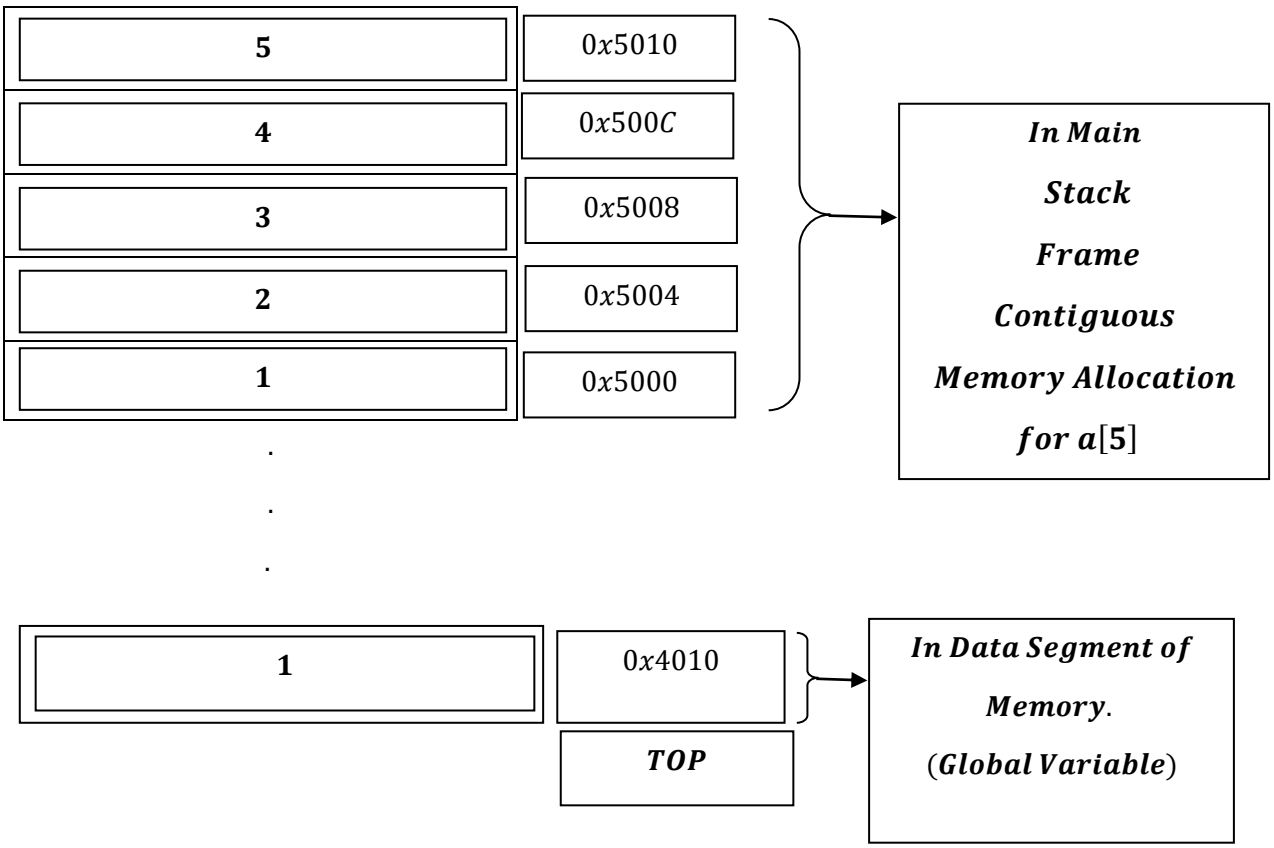
$\boldsymbol{\Longrightarrow Base\ Address + 2[index] \times 4bytes}$.

$\boldsymbol{\Longrightarrow 0x5000 + 8}$.

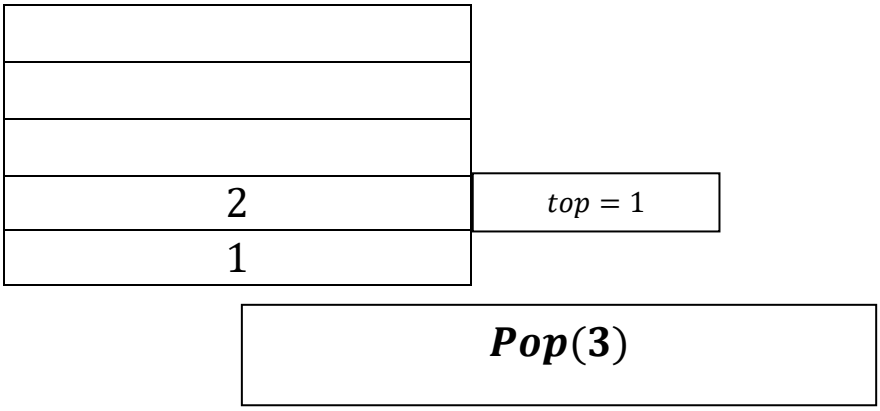
$\boldsymbol{\Longrightarrow 0x5008}$.

***Output*: *Item popped ``the value stored at Address*: $\boldsymbol{0x5008}$: $\boldsymbol{3}$``.**

$\boldsymbol{Top - -\Longrightarrow Top = Top - 1 \Longrightarrow Top = 2 - 1 = 1}$.



This is Physical Demonstration



This is Logical Demonstration

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

$\boldsymbol{\Longrightarrow Stack + 1.}$ [$\boldsymbol{Stack + 1, represents\ contiguous\ memory\ allocation}$]

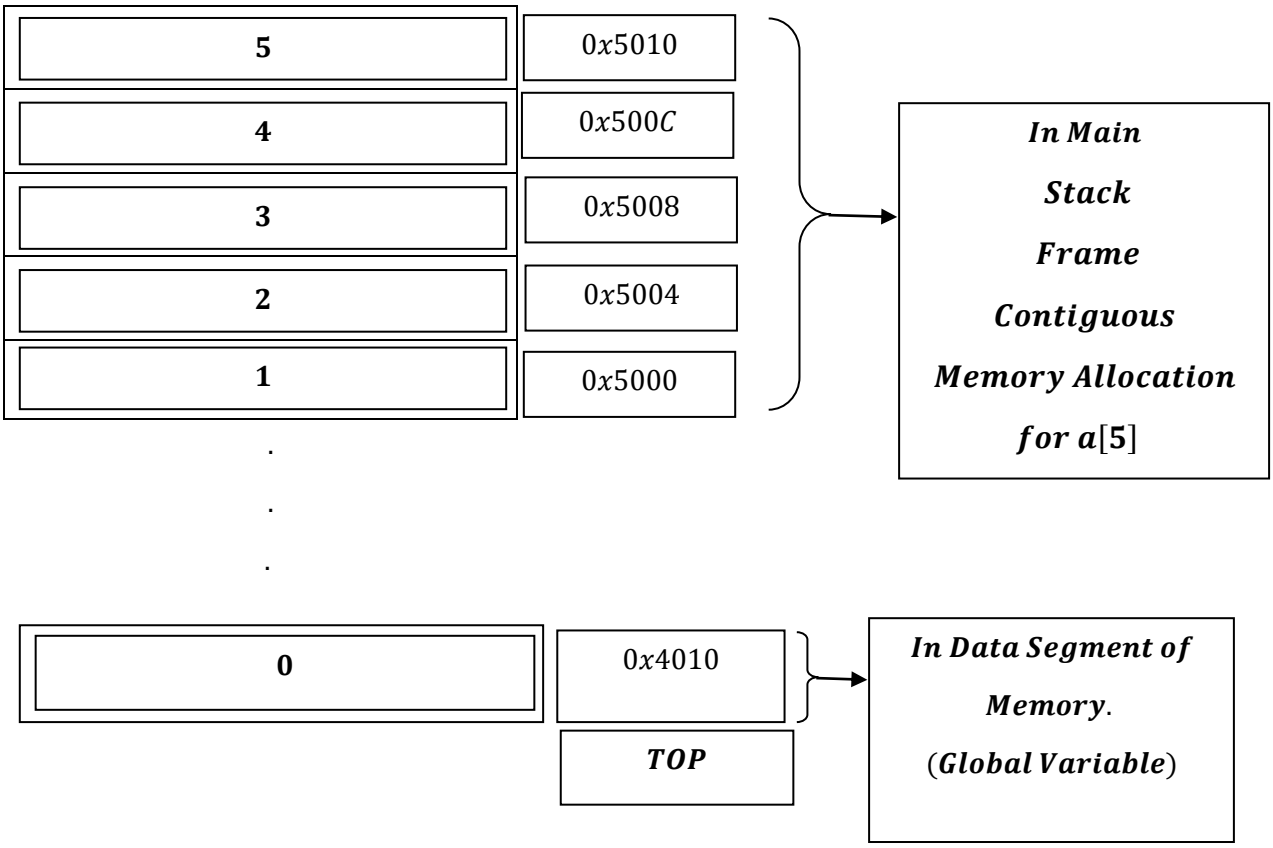
$\boldsymbol{\Longrightarrow Base\ Address + 1[index] \times 4bytes.}$

$\boldsymbol{\Longrightarrow 0x5000 + 4.}$

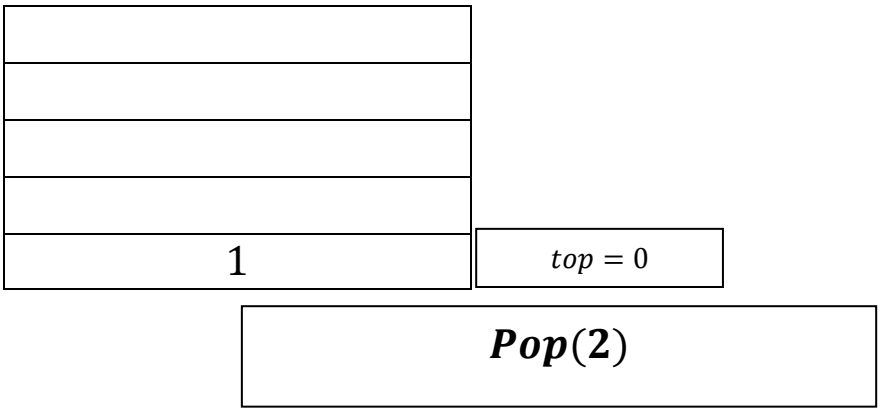
$\boldsymbol{\Longrightarrow 0x5004.}$

$\boldsymbol{Output: Item\ popped\ ``the\ value\ stored\ at\ Address: 0x5004: 2``.}$

$\boldsymbol{Top - - \Longrightarrow Top = Top - 1 \Longrightarrow Top = 1 - 1 = 0.}$



This is Physical Demonstration



This is Logical Demonstration

$\boldsymbol{Stack[Top = 0]}$.

$\boldsymbol{\Longrightarrow Stack + 0}$. [$\boldsymbol{Stack + 0, represents\ contiguous\ memory\ allocation}$]

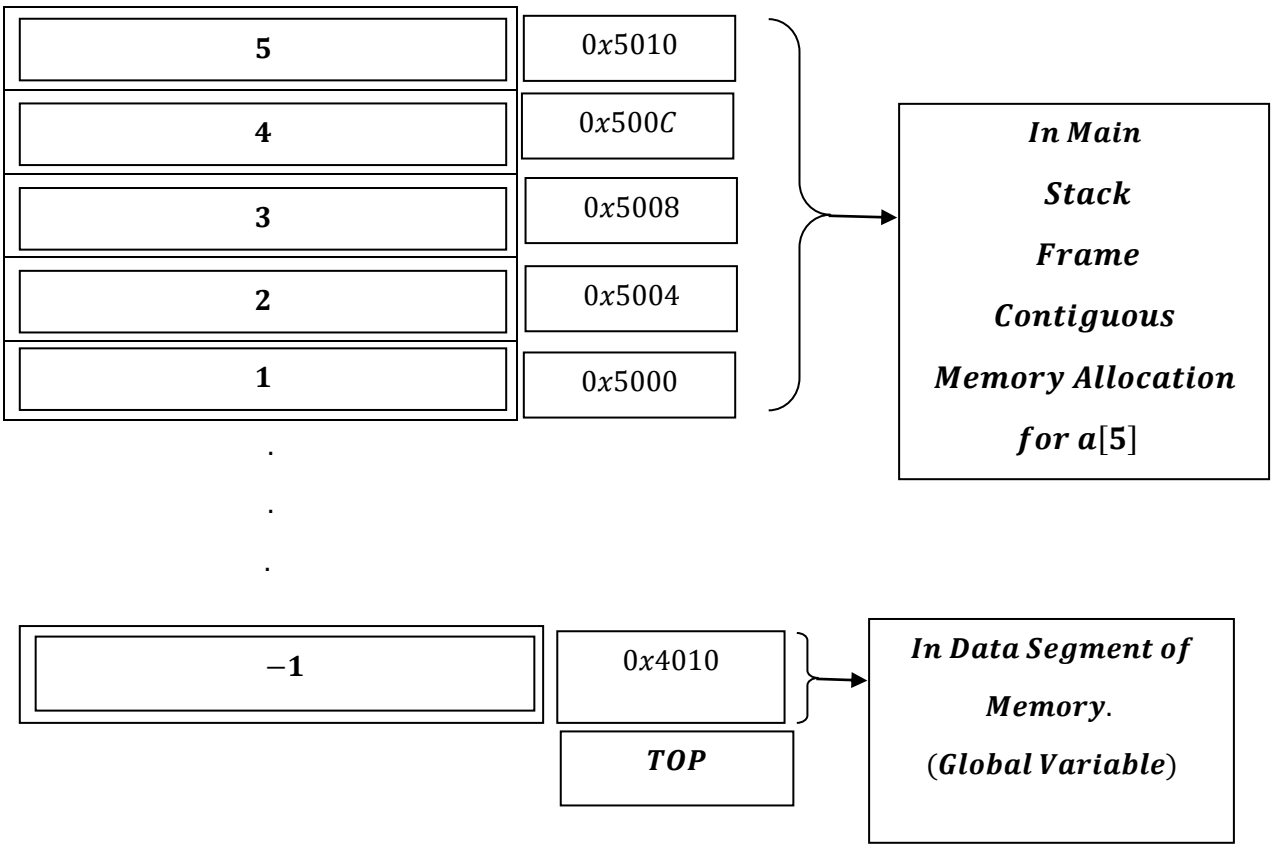
$\boldsymbol{\Longrightarrow Base\ Address + 0[index] \times 4bytes}$.

$\boldsymbol{\Longrightarrow 0x5000 + 0}$.

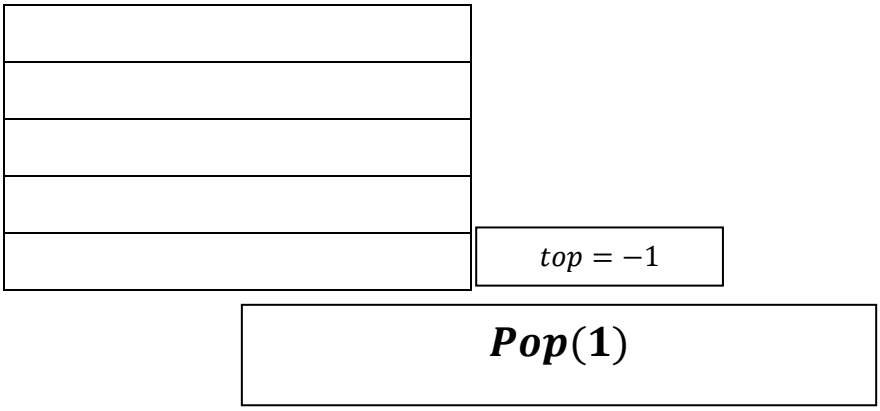
$\boldsymbol{\Longrightarrow 0x5000}$.

***Output*: *Item popped ``the value stored at Address*: $\boldsymbol{0x5000}$: $\boldsymbol{1}$``.**

$\boldsymbol{Top - -\Longrightarrow Top = Top - 1 \Longrightarrow Top = 0 - 1 = -1}$.



This is Physical Demonstration



This is Logical Demonstration

```
if (top == -1)
 {

     return;
 }
```

When top becomes -1 , it returns void and exit from the function.

Note , in actual physical memory only top's value changes but values entered in the array stored in main stack frame remain intact until whole main function stack frame doesn't get destroyed.

$\boldsymbol{Time\ Complexity}$

```
void pop(int stack[])
{

    if (top == -1)
    {

        return;
    }
    cout << "Item popped" << stack[top] << endl;
    top--;
}
```

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

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

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

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

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

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

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

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

If false then:

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