

Stack Operation Vs Loop Operation in Memory.

```
int sum = 0;  
for(int i = 0; i < n; i + +){  
sum = sum + i;  
}
```

<i>i = 0</i>	0x0000
<i>Sum = 0</i>	0x0003
Data Register [ADD]	

- *i and sum are local variables.*
- *They are created once inside the function's stack frame[say inside main function's stack frame].*
- *Their memory location remains the same throughout the loop.*
- *Only their values change during each iteration.*
- *The statement `sum = sum + i` executes *n* times.*

- *Read value of i*
- *Read value of sum*
- *Compute Result*
 - *ADD $i, 1$ [Sent to Instruction Register]*
 - *ADD SUM, i [Sent to Instruction Register]*
- [Note: This is simpler version]*
- *We get $i = 1$ and Sum = 1 , therefore :*
 - *Write $i = 1$ to the reserved memory for i .*
 - *Write Sum = 1 to the reserved memory for Sum.*

$i = 1$	0x0000
Sum = 1	0x0003

Data Register [ADD]

It replaces the values but no push and pop , hence during for loop memory is used again and again `n` times.

Hence the process will continue up to n times at same location. We can state that the statement or operation $sum = sum + i$ runs up to $i = n$ i.e., n times .

Generating Time Complexity : $O(n)$.

Where as in Push and Pop operation:

Stack Push Operation [Time Complexity]

<i>[Single Element Pushed]</i>	<i>$\rightarrow O(1)$</i>
<i>[Single Element Pushed]</i>	<i>$\rightarrow O(1)$</i>
<i>[Single Element Pushed]</i>	<i>$\rightarrow O(1)$</i>
<i>[Single Element Pushed]</i>	<i>$\rightarrow O(1)$</i>
<i>[Single Element Pushed]</i>	<i>$\rightarrow O(1)$</i>
<i>[Single Element Pushed]</i>	<i>$\rightarrow O(1)$</i>
<i>.....</i>	
<i>[Single Element Pushed]</i>	<i>$\rightarrow O(1)$</i>

i. e., Push will be inserted at each Top of the stack.

i. e., At each location the Push operation will be performed at $O(1)$ complexity.

Also, if we see there is lower bound (1) and upper bound(n i.e. size of the stack = n):

$$1 \leq f(n) \leq n$$

Here if we do average if it runs up to n times:

$$\frac{1 + 1 + 1 + 1 \dots + n \text{ times}}{n} = \frac{n}{n} = 1 = O(1).$$

Stack Pop Operation [Time Complexity]

[Single Element Popped]	$\rightarrow O(1)$
[Single Element Popped]	$\rightarrow O(1)$
[Single Element Popped]	$\rightarrow O(1)$
[Single Element Popped]	$\rightarrow O(1)$
[Single Element Popped]	$\rightarrow O(1)$
[Single Element Popped]	$\rightarrow O(1)$
.....	
[Single Element Popped]	$\rightarrow O(1)$

i.e., Pop of element will be done at each Top (First Element) of the stack.

i. e., At each location the Pop operation will be performed at $O(1)$ complexity.

Also, if we see there is lower bound (1) and upper bound(n i. e. size of the stack = n):

$$1 \leq f(n) \leq n$$

Here if we do average if it runs up to n times:

$$\frac{1 + 1 + 1 + 1 \dots + n \text{ times}}{n} = \frac{n}{n} = 1 = O(1).$$

-----*****-----