Stack Mechanism Discussion with Time Complexity

3. *Is Full*

```
int isFull(Stack st)
{
    return st.top == st.size - 1;
}
…
        case 6:

            if (isFull(stck))
            {
                cout << isFull(stck) << endl;
                cout << "Stack is Full" << endl;
            }
            else
            {
                cout << "Stack is not Full" << endl;
            }
            break;
```

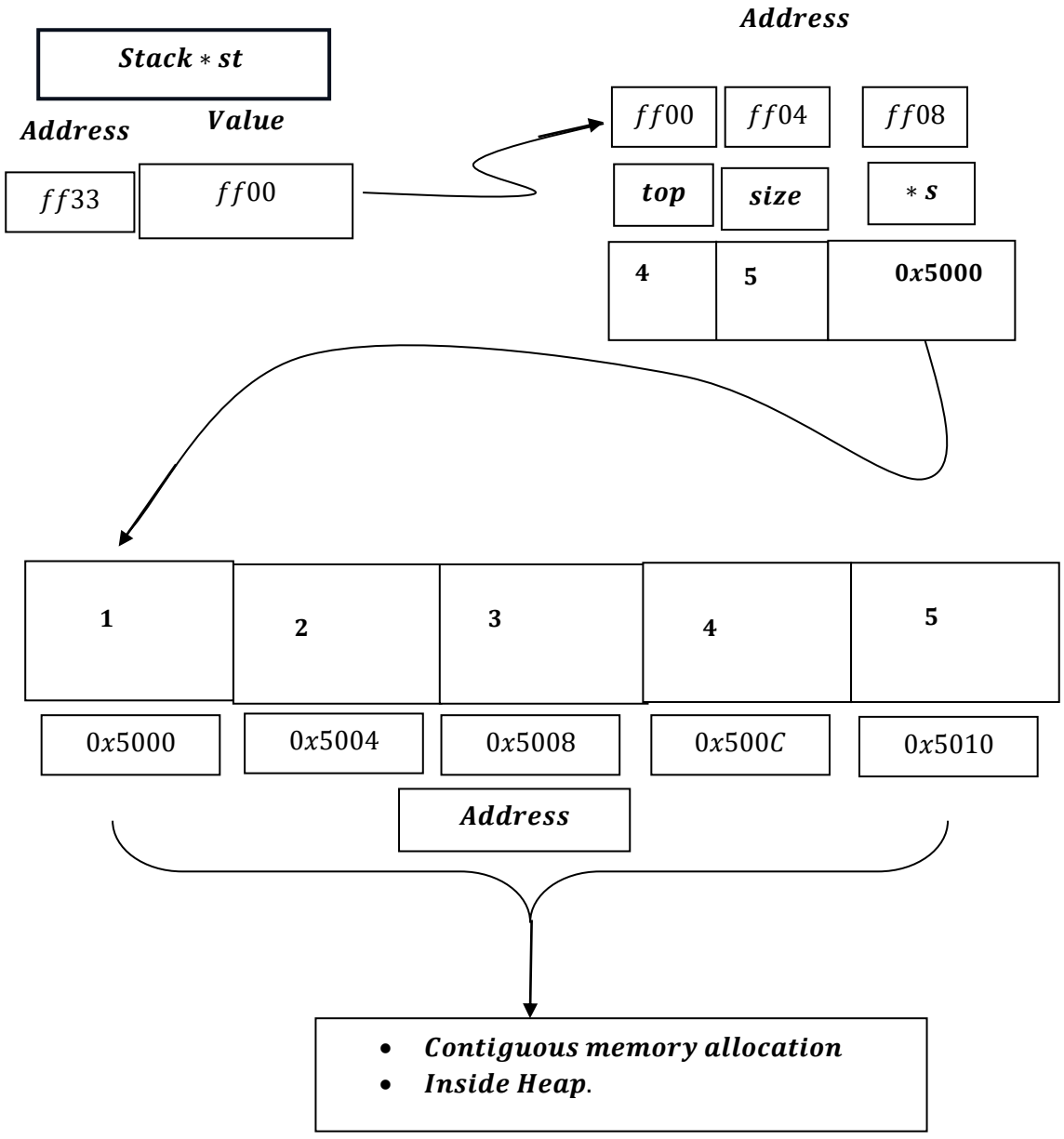
if Top of the stack is : SIZE − 1 , then stack is full and it will print ``Stack is full`` else it will print ``Stack is not full``.

5
4
3
2
1

$top = 4$

***Size of Stack is* 5**

Empty stack



Time Complexity of IsFull function

```
int isFull(Stack st)
{
    return st.top == st.size - 1;
}
```

Step-by-step Time Analysis

1. **Function call overhead**
   → Takes constant time
   → c units → **O(1)**
2. **Accessing variables**
   `st.top` and `st.size`
   → Each access takes constant time
   → c units → **O(1)**
3. **Arithmetic operation**
   `st.size - 1`
   → One subtraction
   → c units → **O(1)**
4. **Comparison operation**
   `st.top == st.size - 1`
   → One comparison
   → c units → **O(1)**
5. **Return result (true/false → 1/0)**
   → Constant time
   → c units → **O(1)**

Total Time Complexity : $O(1) + O(1) + O(1) + O(1) + O(1)$
$=$ **O(1)**

What is Function Call Overhead?

Function call overhead means the small amount of extra work the system does when a function is called.

```
int isFull(Stack st)
{
    return st.top == st.size - 1;
}
```

Function call:

```
 isFull(stck)
```

Even if our function does almost nothing, the system still has to:

1. Save the current execution state
2. Pass arguments to the function
3. Allocate space for local variables
4. Jump to the function's memory location
5. After execution, return back to the caller

All of this takes a **constant amount of time**.

Stack Data structure basically shows how call stack frame works i.e. in LIFO method , but in Stack Data Structure we manually do push() and pop() , where in call stack frame, it creates stack frame for entire function containing local variables , parameters , return address and saved registers and adds or push another stack frame if called again and pops whole stack frame automatically thus different from Stack datastructure , yet conceptually similar.
