Cykor Assignment 1: Simulating the Function Callstack in C++

Jinho Kim

Department of Cyber Defence, Korea University
ID: 2025350219
jhk0317 at korea dot ac dot kr

May 12, 2025

Contents

This report explains an simulation of a call stack implemented in C++.

I Global Variables and Constants

```cpp
#include <iostream>
#include <cstring>

#define STACK_SIZE      50
#define ERROR_SENTINEL -999

int             call_stack[STACK_SIZE];
std::string     stack_info[STACK_SIZE];

int SP = -1;
int FP = -1;
```

- `call_stack[STACK_SIZE]`: An integer array representing the contents of the simulated stack.
- `stack_info[STACK_SIZE]`: Holds metadata describing the contents of `call_stack`.
- `SP` (Stack Pointer): Index of the current top of the stack, initialised to `-1`.
- `FP` (Frame Pointer): Index of the current frame base, also initialised to `-1`.
- `ERROR_SENTINEL`: A constant used to denote erroneous return values from `pop()`.

II Function `push()`

```cpp
void push(int value, std::string info) {
    if (SP >= STACK_SIZE - 1) {
        std::cerr << "Stack is full" << std::endl;
        return;
    }

    ++SP;
    call_stack[SP] = value;
    stack_info[SP] = info;

    return;
}
```

Inserts a value and its description into the stack:

- Prevents stack overflow.
- Increments `SP`, stores the value and its info.

III Function `pop()`

```
int pop() {
    if (SP == -1) {
        std::cerr << "Stack is already empty" << std::endl;
        return ERROR_SENTINEL;
    }

    int return_value = call_stack[SP];

    call_stack[SP] = 0;
    stack_info[SP] = "";
    --SP;

    return return_value;
}
```

Removes the top element from the stack:

- Prevents stack underflow.
- Returns the removed value and clears the associated info.

IV Function `prologue()`

```
void prologue(std::string function_name) {
    push(-1, "Return Address");
    push(FP, function_name + " SFP");
    FP = SP;
}
```

Simulates function entry:

- Pushes a dummy return address (`-1`).
- Pushes current `FP` as the saved frame pointer.
- Updates `FP` to the new base at the current `SP`.

V Function `epilogue()`

```
void epilogue() {
    if (SP < 1) {
        std::cerr << "Error" << std::endl;
        return;
    }

    while (SP > FP) {
        pop();
    }

    FP = call_stack[FP];
    pop();
    pop();
}
```

Simulates function return:

- Pops all local and argument values above the frame base.
- Restores `FP` using the stored frame pointer.
- Pops the saved frame pointer and dummy return address to fully clean the frame.
- Added validation to avoid errors when `SP < 1`.

VI Function: `main()`

The `main()` function:

- Calls `func1()`.
- After return, it prints the final state of the stack.
- Does not implement a frame for itself, following the assignment instructions.

VII Default Functions (`func1()`, `func2()`, `func3()`, `print_stack()`)

Default functions are not explained.

VIII Output

```
====== Current Call Stack ======
5: var_1 = 100    <=== [esp]
4: arg3 = 3
3: arg2 = 2
2: arg1 = 1
1: func1 SFP    <=== [ebp]
0: Return Address
================================


====== Current Call Stack ======
10: var_2 = 200    <=== [esp]
9: arg2 = 13
8: arg1 = 11
7: func2 SFP = 1    <=== [ebp]
6: Return Address
5: var_1 = 100
4: arg3 = 3
3: arg2 = 2
2: arg1 = 1
1: func1 SFP
0: Return Address
================================


====== Current Call Stack ======
15: var_4 = 400    <=== [esp]
14: var_3 = 300
13: arg1 = 77
12: func3 SFP = 7    <=== [ebp]
11: Return Address
```

```
10: var_2 = 200
9: arg2 = 13
8: arg1 = 11
7: func2 SFP = 1
6: Return Address
5: var_1 = 100
4: arg3 = 3
3: arg2 = 2
2: arg1 = 1
1: func1 SFP
0: Return Address
================================


====== Current Call Stack ======
10: var_2 = 200    <=== [esp]
9: arg2 = 13
8: arg1 = 11
7: func2 SFP = 1    <=== [ebp]
6: Return Address
5: var_1 = 100
4: arg3 = 3
3: arg2 = 2
2: arg1 = 1
1: func1 SFP
0: Return Address
================================


====== Current Call Stack ======
5: var_1 = 100    <=== [esp]
4: arg3 = 3
3: arg2 = 2
2: arg1 = 1
1: func1 SFP    <=== [ebp]
0: Return Address
================================


Stack is empty.
```