<push 함수>

```
void push(int value, const char* info) {
    if (SP < STACK_SIZE-1) {
        call_stack[++SP] = value;
        snprintf(stack_info[SP], sizeof stack_info[SP], "%s", info);
    }
}
```

푸쉬가 들어올 때, 스택 포인터가 스택 최대 사이즈보다 작다면 이를 증가시켜 새로운 항목을 추가할 공간을 만든다.

snprintf로 stack_info 버퍼에 info 문자열을 넣는다.

<pop 함수>

```
int pop() {
    if (SP >= 0) {
        int v = call_stack[SP];
        call_stack[SP] = 0;
        stack_info[SP][0] = '\0';
        SP--;
        return v;
    }
    return -1;
}
```

언더플로우를 막기 위해 스택 포인트가 음수가 아닌지 확인한다.

Pop 명령은 가장 최근의 값을 꺼내므로, 스택에서 가장 최근의 값인 call_stack[SP]를 꺼낸다.

스택 포인터를 1 감소시켜 이전 값을 가리키게 하고, 꺼내기로 했던 값을 그대로 return해준다.

<prologue 함수>

```
void prologue(const char* func_name) {
    push_return_address();
    push(FP, func_name);
    FP = SP;
}
```

```
void push_return_address() {
    push(-1, "Return Address");
}
```

현재 FP(프레임 포인터)의 값을 스택에 저장하고, 새로운 FP 값을 SP 값으로 지정한다.

Push_return_address는 Return Address를 표시한다.

<epilogue 함수>

```
void epilogue() {
    FP = pop();
    pop();
}
```

스택 최상위에 저장된, 즉 pop되는 값은 프롤로그에서 밀어넣은 old FP다. 이를 pop할 때 다시 복원시켜야 한다. 두 번째 pop()은 리턴 주소 표시를 pop한다.

<흐름>

main에서 func1이 호출된다.

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

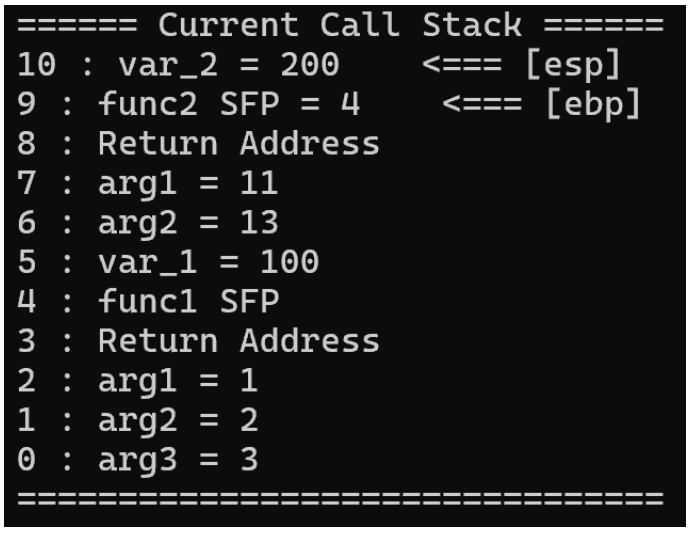
Prologue가 실행되며 새로운 스택 프레임이 생성되고, arg3=3, arg2=2, arg1=1이 차례로 푸시된다. 리턴 주소가 표시되며 FP가 갱신되고, var_1=1이 푸시된 상태의 사진이다.

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

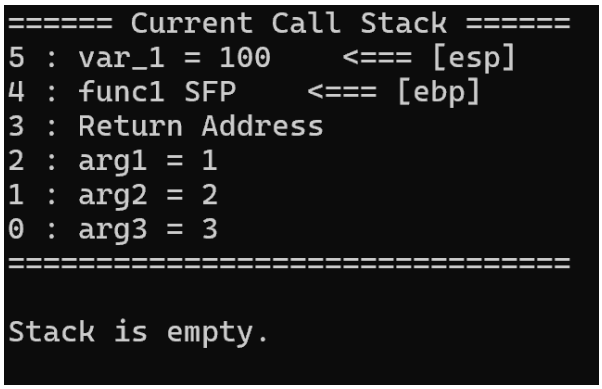
func1 함수 내에서 func2가 호출된다. func2 함수가 호출될 때도 Prologue가 실행되며 새로운 스택 프레임이 생성되고, 주어진 arg2=13, arg1=11이 푸시된다. 여기서 SFP가 이전에 찍어놓은 지점인 4를 찍는 func2 SFP=4 문구가 들어가며, 이곳이 베이스다. 이후 var_2=200 푸시한 것이 들어오며, 이곳이 esp가 된다.

```
====== Current Call Stack ======
15 : var_4 = 400    <=== [esp]
14 : var_3 = 300
13 : func3 SFP = 9    <=== [ebp]
12 : Return Address
11 : arg1 = 77
10 : var_2 = 200
9 : func2 SFP = 4
8 : Return Address
7 : arg1 = 11
6 : arg2 = 13
5 : var_1 = 100
4 : func1 SFP
3 : Return Address
2 : arg1 = 1
1 : arg2 = 2
0 : arg3 = 3
================================
```

func2 함수 내에서 func3이 호출되며, 방금과 거의 같다. Prologue가 실행되며 새로운 스택 프레임이 생성되고, 주어진 arg1=77이 푸시된다. 여기서 SFP가 이전에 찍어놓은 지점인 9를 찍는 func3 SFP=9 문구가 들어가며, 이곳이 베이스다. 이후 var_3=300, var_4=400 푸시한 것이 차례로 들어오며, 이곳이 esp가 된다.



func3이 종료된 이후, epilogue가 실행되며 SP와 FP가 복원되며 func3에서 들어왔던 값들이 pop 된다. ebp는 원래의 ebp인 13번에서 걸어놓았던 9번으로 복원되었고, esp도 현재 가장 위인 10번이 되었다.



func2가 종료된 이후, epilogue가 실행되며 SP와 FP가 복원되며 func2에서 들어왔던 값들이 pop 된다. ebp는 원래의 ebp인 9번에서 걸어놓았던 4번으로 복원되었고, esp도 현재 가장 위인 5번이 되었다.

func1에서 남은 모든 값들을 pop하자, Stack에 남아있는 값이 더 없게 되어 "Stack is empty."라는 문구가 출력된다.