# 1주차 보고서

작성: 스마트보안학과 2023350018 서민주

## 1. Push 함수 구현

```
void push(char* info, int val)
{
   if (SP+1 >= STACK_SIZE)
   {
      printf("Push Error: Stack is Full.\n");
      return ;
   }

   SP++;
   strcpy(stack_info[SP], info);
   call_stack[SP]=val;
}
```

- 현재 스택이 가득 찼는지를 확인하고, 스택 크기를 초과하는 경우 스택이 다 찼다는 메시지를 출력하고 함수를 종료하도록 하였다.
- 스택이 남아있는 경우 스택 포인터를 1 증가시키고, 현재 스택포인터 위치에 stack_info에는 변수, 함수 등의 설명을, call_stack에는 그 값을 저장하도록 했다.

## 2. Pop 함수 구현

```
void pop()
{
   if (SP==-1)
   {
      printf("Pop Error: Stack is Empty.\n");
      return ;
   }

   SP=FP;
}
```

처음에는 단순이 SP의 위치를 FP 위치로 변경했다. 그러나, 이렇게 되면 위치만 변경될 뿐 스택의 데이터는 그대로 남아있다는 사실을 알게 되었다. 따라서, 함수를 다음과 같이 수정하게 되었다.

```
void pop()
{
    if (SP == -1)
    {
        printf("Pop Error: Stack is Empty.\n");
        return;
    }

    while (SP > FP)
    {
        call_stack[SP] = 0;
        stack_info[SP][0] = '\0';
        SP--;
    }

    return;
}
```

SP가 FP와 같은 위치가 될 때까지 call_stack과 stack_info를 0으로 초기화하였다.

## 3. 함수 프롤로그, 에필로그 구현

```
void func1(int arg1, int arg2, int arg3)
{
    int var_1 = 100;

    // func1의 스택 프레임 형성 (함수 프롤로그 + push)
    push("arg3", arg3);
    push("arg2", arg2);
    push("arg1", arg1);
    push("Return Address", -1);
    push("func1 SFP", -1);
    FP=SP;
    push("var_1", 100);
```

```
    print_stack();

    func2(11, 13);

    // func2의 스택 프레임 제거 (함수 에필로그 + pop)
    pop();
    FP = call_stack[FP];
    SP--;
    SP--;

    print_stack();
}


void func2(int arg1, int arg2)
{
    int var_2 = 200;

    // func2의 스택 프레임 형성 (함수 프롤로그 + push)
    push("arg2", arg2);
    push("arg1", arg1);
    push("Return Address", -1);
    push("func2 SFP", FP);
    FP=SP;
    push("var_2", 200);

    print_stack();

    func3(77);

    // func3의 스택 프레임 제거 (함수 에필로그 + pop)
    pop();
    FP = call_stack[FP];
    SP--;
    SP--;

    print_stack();
```

```
}


void func3(int arg1)
{
    int var_3 = 300;
    int var_4 = 400;

    // func3의 스택 프레임 형성 (함수 프롤로그 + push)
    push("arg1", arg1);
    push("Return Address", -1);
    push("func3 SFP", FP);
    FP=SP;
    push("var_3", 300);
    push("var_4", 400);

    print_stack();
}


//main 함수에 관련된 stack frame은 구현하지 않아도 됩니다.
int main()
{
    func1(1, 2, 3);

    // func1의 스택 프레임 제거 (함수 에필로그 + pop)
    pop();
    FP = call_stack[FP];
    SP--;
    SP--;

    print_stack();

    // 매개변수 제거
    pop();
    print_stack();
```

```
    return 0;
}
```

- 함수 프롤로그 과정에 따라 `func1` 의 매개변수 → 반환 주소값 → SFP → 지역 변수 순으로 `push` 를 진행하였다. 또한, SFP를 `push` 한 이후에는 FP의 값 역시 조정해주었다.
- 에필로그의 경우 pop을 이용해 스택을 비워주면서 SP를 FP 위치로 갱신하였고, FP의 값을 업데이트 해주었다. 이후 SP의 위치를 두 칸 내렸다. pop 함수를 이용하면 스택에는 SFP, 반환 주소값이 남아있는 상황이기 때문이다. 주소값을 복원하는 것을 별도로 구현하지 않았기 때문에 SP 값을 줄이는 것으로 마무리하였다.

## 4. 결과

```
myang@MYANG:/mnt/d/KUSS/Cykor/2025/week1$ ./callstack_서민주
====== Current Call Stack ======
5 : var_1 = 100    <=== [esp]
4 : func1 SFP    <=== [ebp]
3 : Return Address
2 : arg1 = 1
1 : arg2 = 2
0 : arg3 = 3
================================

====== 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
================================

====== 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
================================
```

```
====== Current Call Stack ======
11 : arg1 = 77    <=== [esp]
10 : var_2 = 200
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
================================

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

====== Current Call Stack ======
2 : arg1 = 1    <=== [esp]
1 : arg2 = 2
0 : arg3 = 3
================================

Stack is empty.
```