2024350222 오동현

Cykor 과제 1

우선 가장 기본적으로 필요한 push와 pop함수를 구현했다.

*Push 함수

```
void push(int var, const char *info) {

    if (SP >= STACK_SIZE - 1) {
        printf("Stack is full.₩n");
        exit(1);
    }

    SP++;
    call_stack[SP] = var;

    snprintf(stack_info[SP], sizeof(stack_info[SP]), "%s", info);


}
```

우선 stack에 넣을 변수 var과 그에 따르는 stack_info에 넣을 info 값을 받는다. 이때 어차피 info는 수정하지 않고 읽어와서 쓰기만 할 것이니, const char *로 자료형을 지정했다. SP가 49보다 크거나 같으면, stack이 가득 차있다는 말이니까 Stack이 다 찼다고 프린트하고 함수를 종료한다. 그리고 SP를 1 증가시키고, 그 인덱스의 스택에 변수 값을 저장한다. 그리고 문자열을 복사하는 함수인 snprintf를 사용하여, info로 넘어온 정보 값을 stack_info 배열에 저장했다.

*Pop 함수

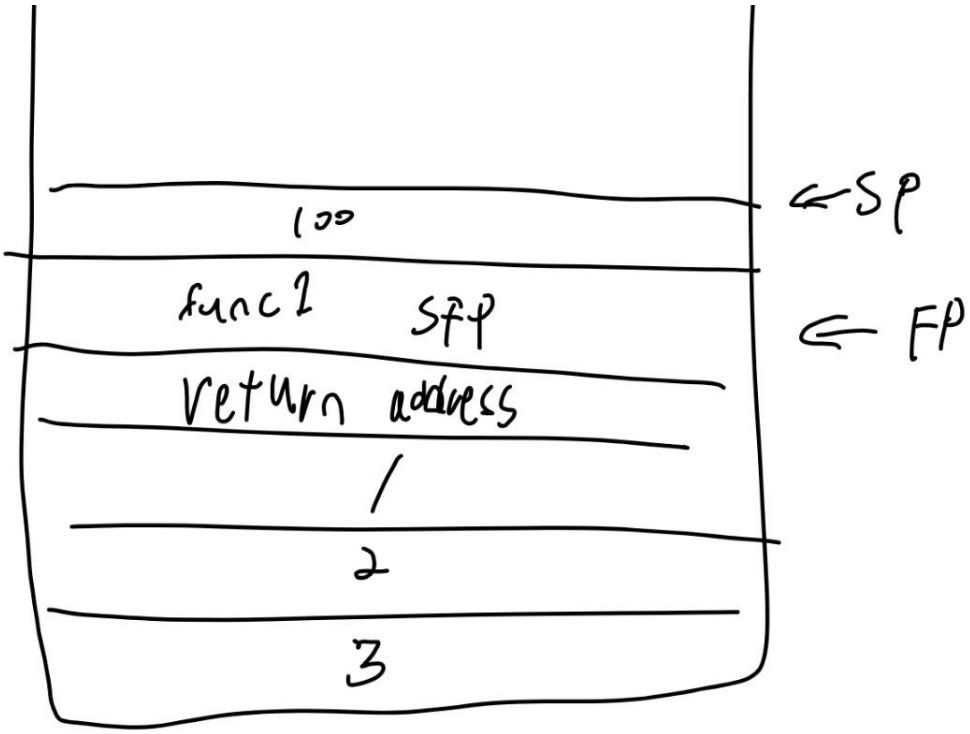
```
void pop() {

    call_stack[SP] = 0;
    SP--;

    snprintf(stack_info[SP], sizeof(stack_info[SP]), "");

}
```

Pop 함수는 현재 SP에 해당하는 스택값을 0으로 초기화한다. 그리고 SP값을 1 감소시킨다. 만약 스택 값이 0인 변수가 존재할 수 도 있기 때문에 이와 구분 가능하도록, info 값도 제거했다.

- 시작전

스택이 비어있다.

1, main함수에서 func1을 먼저 호출하는 코드이기 때문에 func1을 수정했다.



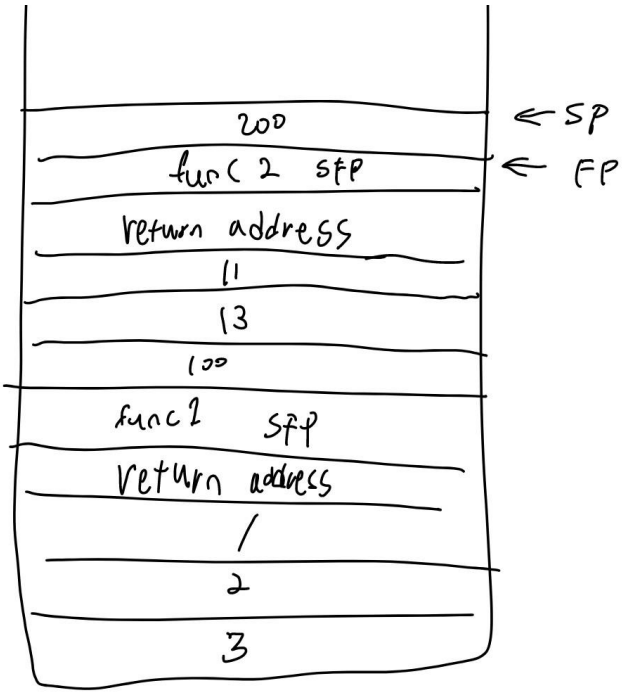
---

fun1을 호출할 때 스택 값은 위 이미지와 같이 표현된다.

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

    // func1의 스택 프레임 형성 (함수 프롤로그 + push)
    //1. 매개 변수
    push(arg3, "arg3");
    push(arg2, "arg2");
    push(arg1, "arg1");
    //2. Return Address
    push(-1, "Return Address");
    //3. SFP
    push(FP, "func1 SFP");
    //4. FP 갱신
    FP = SP;
    //5. 지역 변수
    push(var_1, "var_1");

    print_stack();
    func2(11, 13);
    // func2의 스택 프레임 제거 (함수 에필로그 + pop)
```

우선 main에서 넘어오는 매개 변수들이 스택에 쌓이고, return address, sfp, 마지막으로 지역변수가 쌓이도록 구현했다.

이 다음은 func2의 함수 push를 구현해야 한다.

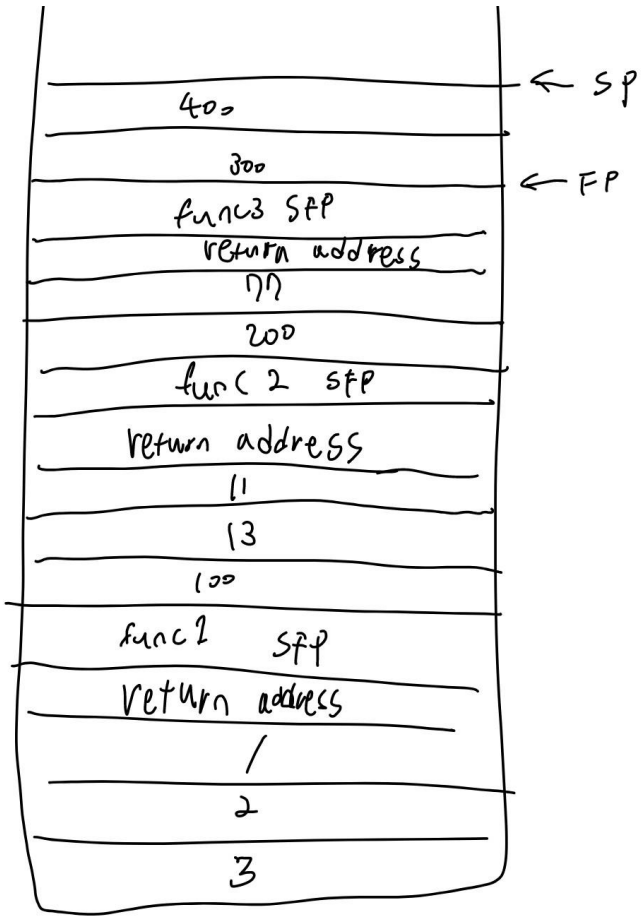


func2 함수가 실행되면 스택은 위와 같이 바뀌게 된다. 이 이미지에 맞게 구현해준다면,

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

    // func2의 스택 프레임 형성 (함수 프롤로그 + push)
    //1. 매개 변수
    push(arg2, "arg2");
    push(arg1, "arg1");
    //2. Return Address
    push(-1, "Return Address");
    //3. SFP
    push(FP, "func2 SFP");
    //4. FP 갱신
    FP = SP;
    //5. 지역 변수
    push(var_2, "var_2");


    print_stack();
    func3(77);
    // func3의 스택 프레임 제거 (함수 에필로그 + pop)
    print_stack();
}
```

func1과 같이, 매개 변수부터 지역 변수까지 순서대로 스택에 쌓이도록 구현하였다.

이제 마지막 func3의 함수 스택을 구현할 차례이다.

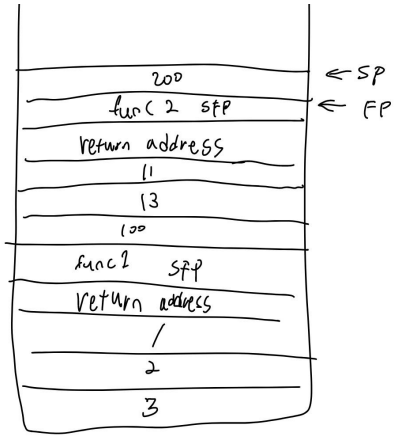


이 이미지처럼 최종적으로 스택이 쌓인 모습이다.

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

    // func3의 스택 프레임 형성 (함수 프롤로그 + push)
    //1. 매개 변수
    push(arg1, "arg1");
    //2. Return Address
    push(-1, "Return Address");
    //3. SFP
    push(FP, "func3 SFP");
    //4. FP 갱신
    FP = SP;
    //5. 지역 변수
    push(var_3, "var_3");
    push(var_4, "var_4");

    print_stack();
}
```

다른 함수와 마찬가지로 같은 방식으로 스택 프레임을 형성하였다.

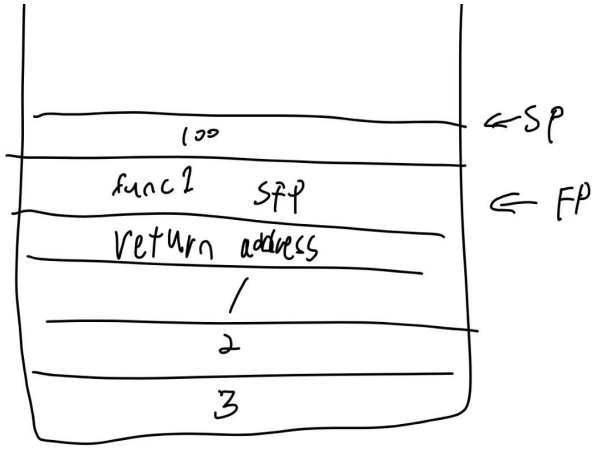
이제 반대로 pop 함수를 이용해 스택 프레임을 순서대로 제거할 차례이다. 쌓을때와 역순으로 제거해야 한다고 생각했다.



이 상태로 다시 되돌려야한다는 말이다.

```
    // func3의 스택 프레임 제거 (함수 에필로그 + pop)
    //1. 지역 변수 제거
    pop(); //var_4
    pop(); //var_3
    //2. FP 복원
    FP = call_stack[FP];
    //3. SFP 제거
    pop(); // func3 SFP;
    //4.return address 제거
    pop();
    //5. 매개 변수 제거
    pop(); // arg1

    print_stack();
}
```

역순으로 가야하므로, 제일 위에 있는 지역 변수부터 pop으로 제거해주었다. 그리고, 지역 변수가 모두 제거되면, SFP가 있으므로, FP를 SFP를 이용해 그 전으로 복원해준다. 그리고 SFP는 이제 쓸모가 없으니 역시 pop으로 제거해주고, return address 까지 제거해주었다. 마지막으로 제일 아래에 있는 매개 변수를 pop으로 제거해준다.



이제 func2 스택을 제거하여 위 같은 이미지의 스택으로 만들어야한다.

```
    // func2의 스택 프레임 제거 (함수 에필로그 + pop)
    //1. 지역 변수 제거
    pop(); //var_2
    //2. FP 복원
    FP = call_stack[FP];
    //3. SFP 제거
    pop(); // func2 SFP;
    //4.return address 제거
    pop();
    //5. 매개 변수 제거
    pop(); // arg1
    pop(); // arg2

    print_stack();
}
```

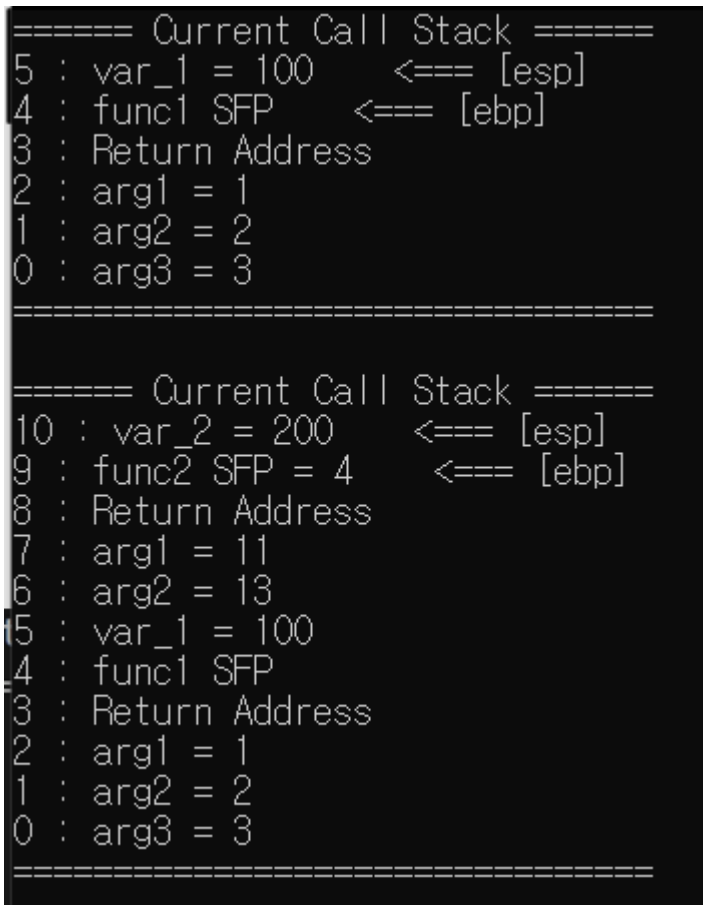
Func3 스택 프레임을 제거할 때와 똑같이 역순으로 제거, FP 복원 과정을 거쳤다.

이제 마지막으로 저 func1의 스택 프레임까지 제거해준다.

```
int main()
{
    func1(1, 2, 3);
    // func1의 스택 프레임 제거 (함수 에필로그 + pop)
    //1. 지역 변수 제거
    pop(); //var_1
    //2. FP 복원
    FP = call_stack[FP];
    //3. SFP 제거
    pop(); // func1 SFP;
    //4.return address 제거
    pop();
    //5. 매개 변수 제거
    pop(); // arg1
    pop(); // arg2
    pop(); // arg3

    print_stack();
    return 0;
}
```

똑같이 역순으로 제거해준다.

이렇게 모든 함수의 스택 프레임을 push로 쌓고, pop으로 해제해주었다. 그리고 실행했다.



위는 func1 스택프레임, 아래는 func1 스택 프레임+ func2 스택 프레임이다.

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

func1 + func2 +func3 이 잘 된걸 확인 가능하다.

```
====== Current Call Stack ======
10 :  = 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 ======
5 :  = 100    <=== [esp]
4 : func1 SFP    <=== [ebp]
3 : Return Address
2 : arg1 = 1
1 : arg2 = 2
0 : arg3 = 3
================================

Stack is empty.
```

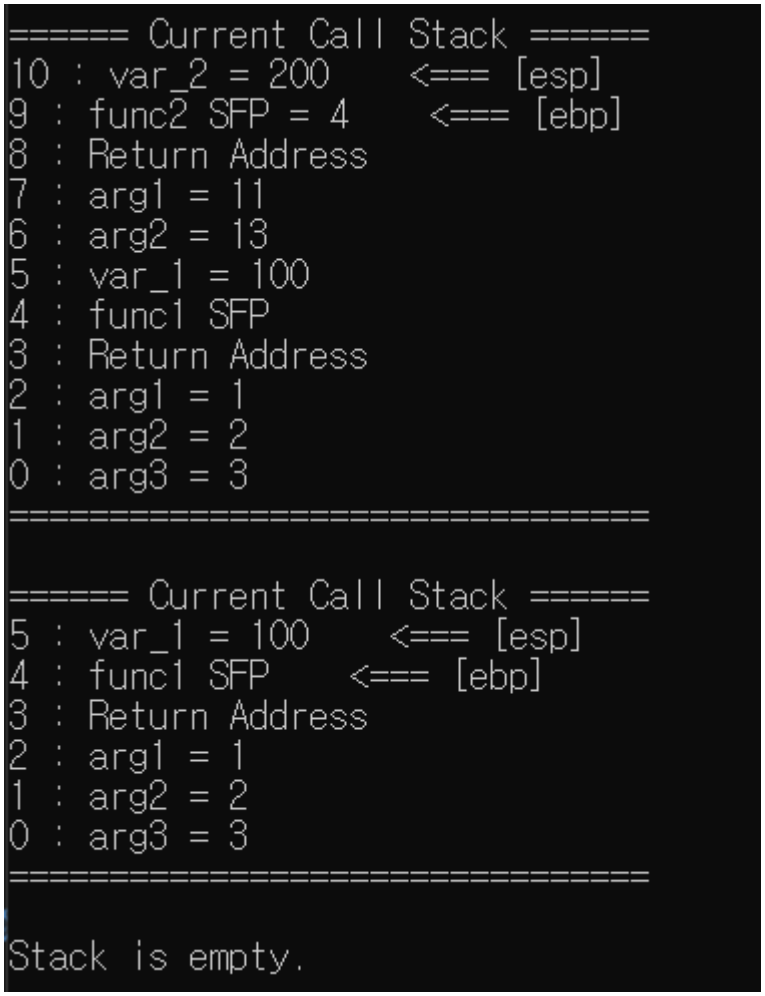
그런데 그 다음에 제거되지 않았어야할 지역변수가 제거된 오류를 발견하였다. 이를 수정하기 위하여 pop 함수를 다시 보았다.

```
void pop() {

    call_stack[SP] = 0;
    SP--;

    snprintf(stack_info[SP], sizeof(stack_info[SP]), "");

}
```

다시 살펴보니 Info 정보를 지우기 위한 snprintf 함수가 SP-- 뒤에 있어서 발생하는 문제라는 것을 발견할 수 있었다. SP를 하나 낮추고, 즉 그 다음 info를 지우고 있던것이다.

따라서 이를 아래와 같이 수정해주고 다시 실행했다.

```
void pop() {

    call_stack[SP] = 0;
    snprintf(stack_info[SP], sizeof(stack_info[SP]), "");

    SP--;

}
```



pop함수를 수정하니, 정상적으로 func3, func2, func1 스택프레임이 차례대로 제거된 사실을 확인할 수 있다.