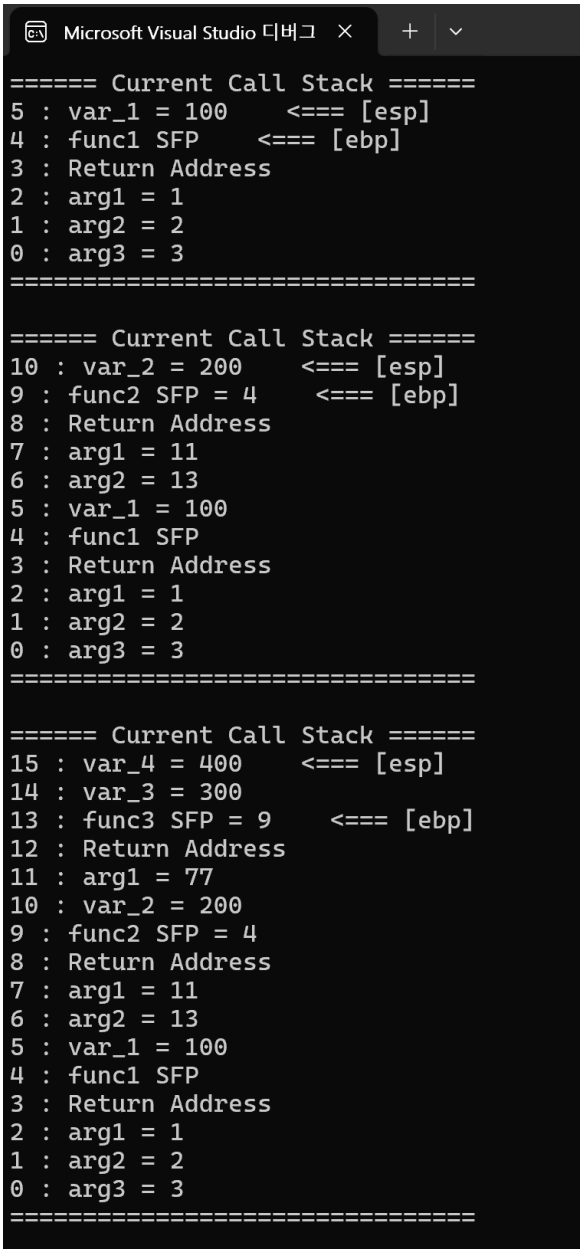
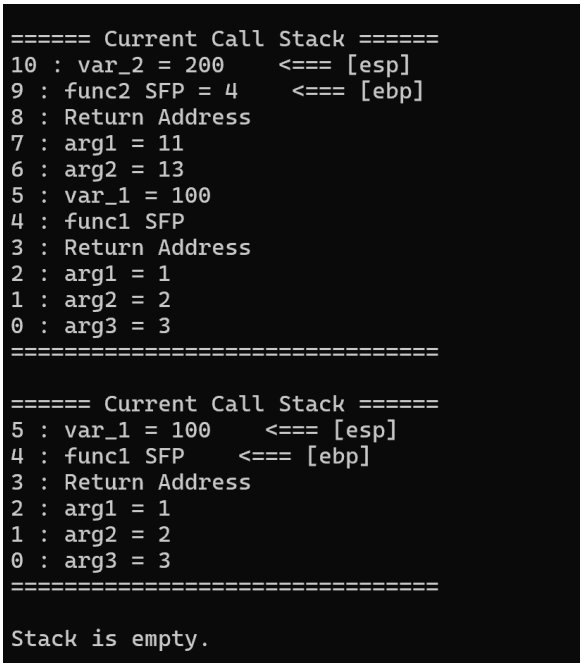
Cykor1-callstack_2024350022_남유찬






```
// func1의 스택 프레임 형성 (함수 프롤로그 + push)
SP++;                   // SP = 0
strcpy_s(stack_info[SP], sizeof(stack_info[SP]), "arg3");
call_stack[SP] = arg3;
SP++;                   // SP = 1
strcpy_s(stack_info[SP], sizeof(stack_info[SP]), "arg2");
call_stack[SP] = arg2;
SP++;                   // SP = 2
strcpy_s(stack_info[SP], sizeof(stack_info[SP]), "arg1");
call_stack[SP] = arg1;
```

우선 이 부분은 일반적인 push 과정으로 구현하였습니다. SP를 한 칸 위로 늘리고,

stack_info에는 각 매개변수의 이름을, call_stack에는 각 매개변수의 값을 넣었습니다.

여기에서 단순히 stack_info[SP] = “arg3”와 같이 구현하려 했으나, "식이 수정할 수 있는 lvalue여야 합니다.”라는 에러 메시지가 발생하여 포인터를 직접 대입하지 않고, strcpy_s (_s는 Visual Studio에서 가능)를 사용하여 대입하였습니다.

```
SP++;                   // SP = 3
strcpy_s(stack_info[SP], sizeof(stack_info[SP]), "Return Address");
call_stack[SP] = -1;
SP++;
```

```
FP = SP;                  // SP = FP = 4
strcpy_s(stack_info[SP], sizeof(stack_info[SP]), "func1 SFP");
call_stack[SP] = -1;

SP++;                     // SP = 5, FP = 4
strcpy_s(stack_info[SP], sizeof(stack_info[SP]), "var_1");
call_stack[SP] = var_1;
```

위에 이어 이 부분은 push로 비슷하지만, 조건에 주어진 대로 Return Address의 call_stack에 -1을 대입하였습니다.

SFP 부분을 push하는 경우에는 SP를 한칸 늘리고 FP를 그 값과 같게 만들어 ebs가 해당 부분을 가리키게 만들었습니다. 여기에서도 call_stack에 -1을 대입하였습니다.

지역변수는 기본적인 push 과정을 따랐습니다.

이러한 과정은 “func2의 스택 프레임 형성”, “func3의 스택 프레임 형성”에서도 동일합니다.

이후 “func3의 스택 프레임 제거” 부분은 다음과 같습니다.

```
// func3의 스택 프레임 제거 (함수 에필로그 + pop)
for (SP; SP > FP; SP--) {
   call_stack[SP] = 0;
}                         // SP = 13, FP = 13

FP = call_stack[SP];      // SP = 13, FP = 9

call_stack[SP] = 0;
SP--;                     // SP = 12, FP = 9

call_stack[SP] = 0;
SP--;                     // SP = 11, FP = 9

call_stack[SP] = 0;
SP--;                     // SP = 10, FP = 9
```

먼저, 기존에는 단순히 SP = FP와 같이 구현하여 SP만 내리려 했으나, call_stack을 0으로 초기화시켜야 하는 듯 하여 for문을 통해 전부 초기화하는 방식으로 구현하였습니다.

이후 FP(ebp)를 이전 함수의 SFP를 가리키도록 구현했습니다.

이후 부분은 SFP와 Return Address 부분은 pop하는 방식으로 하였습니다.

원래는 여기에서 SP = 11일 때까지만 구현하려 하였으나, caller에서 매개변수를 정리한다는 것(calling convention), 그리고 이전 출력과의 대칭성(func3 스택 프레임 쌓기 전과, 제거 후가 같도록)을 고려하여 매개변수까지 정리하였습니다.

이후의 “func2의 스택 프레임 제거”, “func1의 스택 프레임 제거” 또한 같은 과정으로 구현했습니다.

“func1의 스택 프레임 제거”에서 마지막에 FP, SP를 -1로 초기화해 줌으로써, 최종적으로 "Stack is empty.”라는 출력이 발생하도록 하였습니다.