CallStack_2025350214_홍서연 보고서

코드	설명
callstack_2025350214_HongSeoyeon(2).c 1 #include <stdio.h> 2 #include <string.h> 3 #define STACK_SIZE 50 4 5 int call_stack[STACK_SIZE]; 6 char stack_info[STACK_SIZE][20]; 7 8 int SP = -1; 9 int FP = -1; 10 11 void func1(int arg1, int arg2, int arg3); 12 void func2(int arg1, int arg2); 13 void func3(int arg1); 14	제공된 call_stack, stack_info 배열과 지정된 전역변수와 fun1~fun3 이 정의되어 있음. Stack_info 에 문자열을 저장해야 하기도 하고, (strcpy 사용 예정) 추후 문자열끼리 합쳐야 할 일이 생기기에, (strcat 사용 예정) #include<string.h>를 추가로 선언함.
15 16 //------------------push, pop func---------------- 17 void push(int memory_data, char* about_info) { 18 SP += 1; 19 call_stack[SP] = memory_data; 20 strcpy(stack_info[SP], about_info); 21 } 22	**<push 함수 선언>** **Push(call_stack 에 저장할 데이터값, stack_info 에 저장할 문자열)** Call_stack 에 저장할 데이터 값과 (memory_data) Stack_info 에 저장할 문자열 값(about_info)를 받음. 18: 우선 이전 SP 를 하나 올려준 후 19~20: 그 SP 를 인덱스 삼아 두 배열에 저장.
23 int pop() { 24 int return_val = call_stack[SP]; 25 call_stack[SP] = 0; 26 strcpy(stack_info[SP], "0"); 27 SP -= 1; 28 return return_val; 29 } 30 //--- 31 32	**<pop 함수 선언>** 매개변수를 받지는 않고, int 형 반환값을 가지도록 함. (추후 에필로그 시 SFP 의 값을 따라 FP 를 변경할 때 SFP 를 pop 하며 받기 위함.) 24: 해당 SP 에 대한 call_stack 에 저장된 데이터 값을 나중에 리턴할 변수에 저장함. 25~26: 두 배열의 데이터 값 지우기 27: SP 를 하나 내려주고, 28: 미리 저장해 놓은 return_val 반환함. =>SFP 의 단계에서 pop 할 시 이를 받아 FP 로 설정할 예정임.

```
33
34  //----------prolog, epillog func -------------------
35  void prolog(char* func_name, int local_num){
36      //1. return adress
37      push(-1, "Return Adress");
38
39      //2. SFP
40      char buf[20];
41      sprintf(buf, "%s SFP", func_name);
42      push(FP, buf);
43      FP = SP;
44
45      //3. local num +
46      SP += local_num;
47  }
```

<프롤로그 함수 선언>
Prolog(실행시킬 함수의 이름, 지역변수의 개수)
(매개변수는 콜러함수 쪽에서 push 할 것이기에 반환 주소 값부터 push 함.)

37: 반환 주소 값을 push

40~42: SFP 를 push
stack_info 에 "함수명+SFP" 를 저장하는 과정.
미리 넉넉하게 20 의 공간으로 buf 배열을 선언 후, sprintf 함수를 활용해 buf 배열에 "함수명 SFP"를 저장.

43: call_stack 에는 현재 FP 의 위치를 넣고, stack_info 에는 buf 에 저장한 문자열을 넣음.

46: 지역변수를 저장할 만큼의 메모리 공간을 만들기 위해 SP 를 local_num 만큼 더 올려줌.

```
48
49  void epillog(){
50      while(SP!=FP){
51          pop();
52      }
53      FP = pop();
54      pop();
55  }
56  //---------------------------------
57
```

<에필로그 함수 선언>

50~52: 저장된 지역변수를 pop 하며 메모리에 저장된 값들을 비움. SP 가 FP 랑 같아지는 순간 멈춤. 이때 SP 는 SFP 를 가리키고 있을 것.

53: 이후 pop 을 해 SFP 값을 반환받아 FP 갱신.

54: 이후 pop 은 return adress 값을 반환할 것, 어차피 쓸 일 없으니까 SFP 처럼 따로 받아서 저장하지 않고 pop 만 함.

```
58  //----------remove parameter-----------------
59  void remove_parameter(int parameter_num){
60      for (int i=0;i<parameter_num;i++){
61          pop();
62      }
63  }
64  //-----------------------------
65
66
```

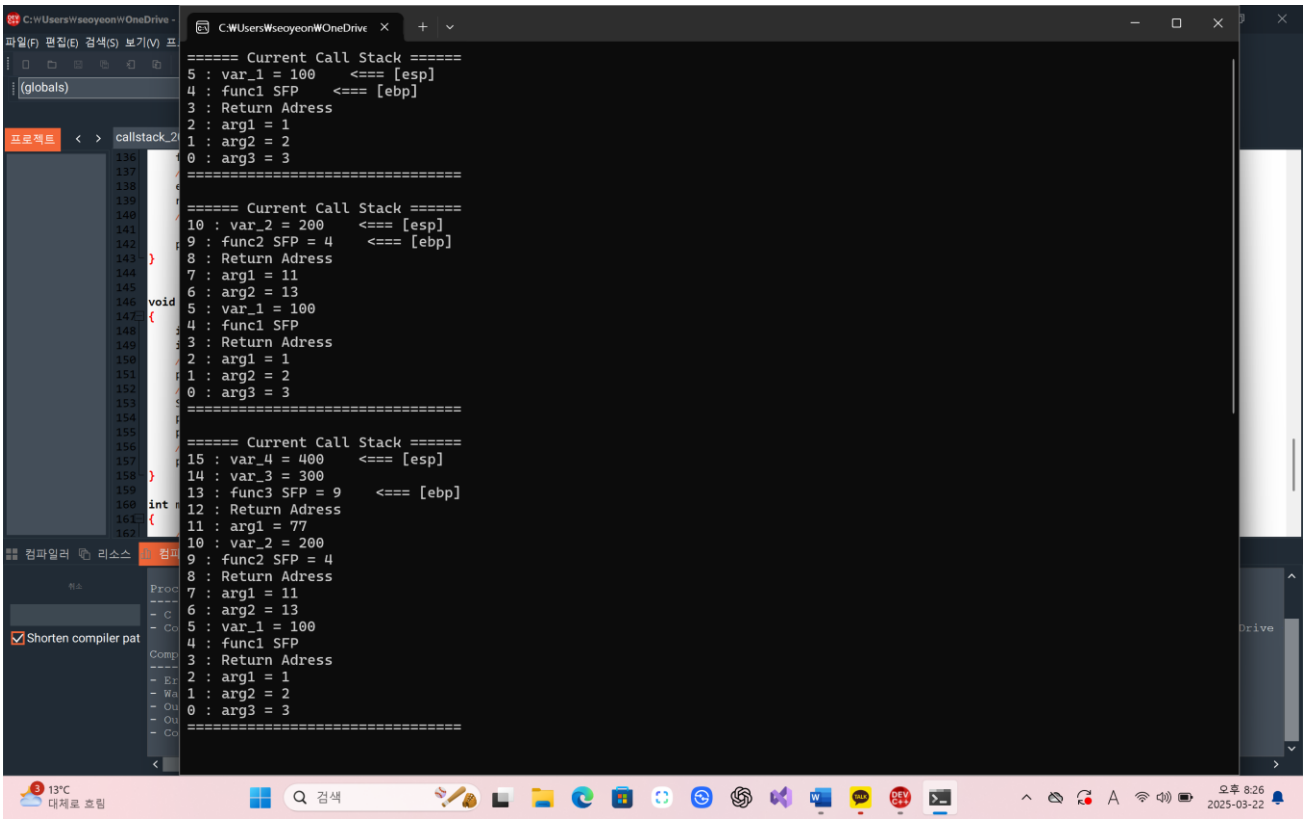
<매개변수 제거 함수>
Remove_parameter(매개변수 개수)

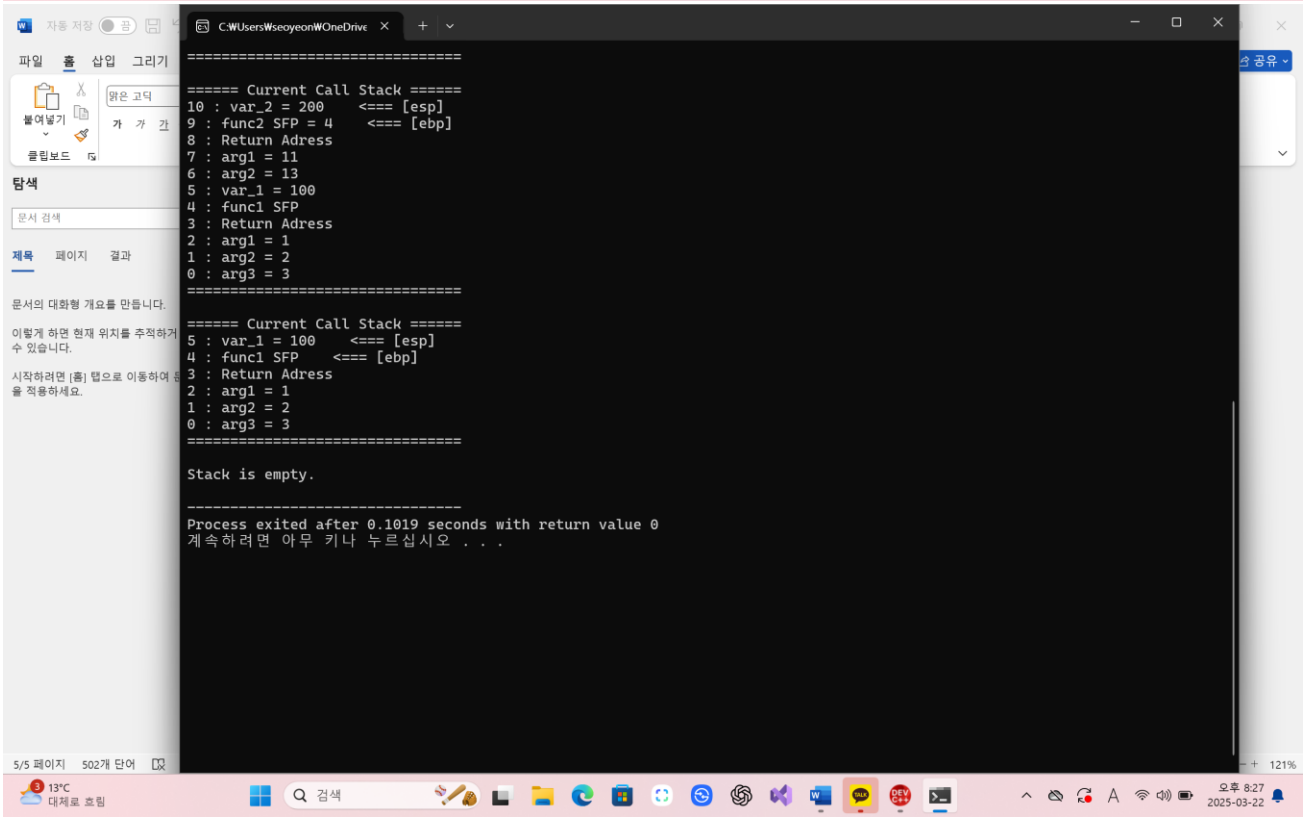
Calling convention(cdcel)에 의해 c 는 대부분 콜러가 매개변수를 정리함.
에필로그를 끝낸 후 콜러 함수에서 매개변수를 지우는 것을 구현하기 위해 에필로그 함수 안에 이 기능을 추가하지 않고, 따로 remove_parameter 라는 이름으로 함수를 만들어줌.

제공된 print_stack 함수	67 번 줄부터 92 번 줄까지는 모두 제공해주신 print_stack 함수입니다. 안의 내용은 따로 건들지 않았음으로 생략합니다
이제부터 본격적인 main 함수, func1,2,3 함수를 호출하고 정리하는 과정입니다. 보고서의 가독성을 위해 main 함수 설명을 먼저 적었습니다.	
159 160 int main() 161 { 162 //save next's func parameter 163 push(3, "arg3"); 164 push(2, "arg2"); 165 push(1, "arg1"); 166 167 func1(1, 2, 3); 168 169 //epillog start---------- 170 epillog(); 171 //remove Parameter 172 remove_parameter(3); 173 //epillog end---------- 174 print_stack(); 175 return 0; 176 }	163~165: 호출할 func1 함수의 매개변수를 main 에서 미리 push 함. (매개변수 push 는 프롤로그가 아니기에 의도적으로 main 함수에서 따로 push 해줌.) 167: fun1 호출 ==========func1 이 끝난 후======= 170: 에필로그 함수 선언. 172: 매개변수들 pop. 함수 호출을 종료할 때 매개변수는 caller 가 정리하기에 일부러 메인함수에서 remove_parameter()를 선언함.
95 //func , 96 void func1(int arg1, int arg2, int arg3) 97 { 98 int var_1 = 100; 99 100 //prolog start-------------- 101 prolog("func1", 1); 102 //local value - 103 SP -= 1; 104 push(100, "var_1"); 105 //prolog end-------------- 106 print_stack(); 107 108 push(13, "arg2"); 109 push(11, "arg1"); 110 func2(11, 13); 111 112 //epillog start-------- 113 epillog(); 114 remove_parameter(2); 115 //epillog end-------- 116 117 print_stack(); 118 }	101: prolog 를 호출해서 return address 쌓고, SFP 쌓고 현재 SP 를 저장할 지역변수 개수만큼 올림. 103~104: 올라간 SP 가 만든 메모리에 지역변수 값 저장함. 108~109: func2 호출 전 이 함수의 매개변수를 미리 쌓아주기 위해 push 함. 110: fun2 호출 ==========func2 가 끝난 후======= 113: 에필로그 함수 선언. 지역변수 지우고 (SP 를 FP 위치로 내리고) SFP 를 FP 가 받아 이동 후 반환주소까지 지움. 114: 매개변수들 pop. 함수 호출을 종료할 때 매개변수는 caller 가 정리한다는 것을 반영하려고 일부러

	func1 에서 remove_parameter()를 선언함.
120 121 void func2(int arg1, int arg2) 122 { 123 int var_2 = 200; 124 //prolog start--------- 125 prolog("func2", 1); 126 //save local value 127 SP -= 1; 128 push(200, "var_2"); 129 //prolog end----------- 130 131 print_stack(); 132 133 //save next func's parameter 134 push(77, "arg1"); 135 136 func3(77); 137 //epillog start---------- 138 epillog(); 139 remove_parameter(1); 140 //epillog end---------- 141 142 print_stack(); 143 } 144	123: func2 의 지역변수 정보 125: prolog 를 호출해서 return address 쌓고, SFP 쌓고 현재 SP 를 저장할 지역변수 크기만큼 올림. 127~128: 올라간 SP 가 만든 메모리에 지역변수 값 저장함. 134: func2 호출 전 이 함수의 매개변수를 미리 쌓아주기 위해 push 함. 136: fun2 호출 ==========func3 이 끝난 후======= 138: 에필로그 함수 선언. 139: 매개변수들 pop.
145 146 void func3(int arg1) 147 { 148 int var_3 = 300; 149 int var_4 = 400; 150 //prolog start----- 151 prolog("func3", 2); 152 //local value push 153 SP -= 2; 154 push(300, "var_3"); 155 push(400, "var_4"); 156 //prolog end------- 157 print_stack(); 158 } 159	148~149: func3 에서 사용할 지역변수. 151: 프롤로그 함수 호출. Prolog(함수이름, 지역변수 개수) 153~155: SP 가 지역변수 개수만큼 올라감. 이때 만들어지는 빈 메모리에 지역변수를 채워넣는 과정을 구현함. -----함수종료----
원래 main 함수가 있어야 할 자리	

코드 실행 결과


C:₩Users₩seoyeon₩OneDrive
====== Current Call Stack ======
5 : var_1 = 100 <=== [esp]
4 : func1 SFP <=== [ebp]
3 : Return Adress
2 : arg1 = 1
1 : arg2 = 2
0 : arg3 = 3
==========
====== Current Call Stack ======
10 : var_2 = 200 <=== [esp]
9 : func2 SFP = 4 <=== [ebp]
8 : Return Adress
7 : arg1 = 11
6 : arg2 = 13
5 : var_1 = 100
4 : func1 SFP
3 : Return Adress
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 Adress
11 : arg1 = 77
10 : var_2 = 200
9 : func2 SFP = 4
8 : Return Adress
7 : arg1 = 11
6 : arg2 = 13
5 : var_1 = 100
4 : func1 SFP
3 : Return Adress
2 : arg1 = 1
1 : arg2 = 2
0 : arg3 = 3
==========


C:₩Users₩seoyeon₩OneDrive
==========
====== Current Call Stack ======
10 : var_2 = 200 <=== [esp]
9 : func2 SFP = 4 <=== [ebp]
8 : Return Adress
7 : arg1 = 11
6 : arg2 = 13
5 : var_1 = 100
4 : func1 SFP
3 : Return Adress
2 : arg1 = 1
1 : arg2 = 2
0 : arg3 = 3
==========
====== Current Call Stack ======
5 : var_1 = 100 <=== [esp]
4 : func1 SFP <=== [ebp]
3 : Return Adress
2 : arg1 = 1
1 : arg2 = 2
0 : arg3 = 3
==========
Stack is empty.

Process exited after 0.1019 seconds with return value 0
계속하려면 아무 키나 누르십시오 . . .