WEEK 1 과제

2025350021 스마트보안학부 박지우

목차

```
void push(char *name, int a)  //push 함수 구현
{
    SP += 1;
    for (int i = 0;i < 20;i++) { //원래 i<20 이 아니라 i<sizeof(name) 이였는데 수정함.
        stack_info[SP][i] = name[i];
    }
    call_stack[SP]=a;


}
```

PUSH 함수

- Push 함수는 변수의 이름을 배열(배열의 이름을 포인터로) 받고, 그 변수의 값을 받는 함수임.
- Push 함수를 통해 해당 SP 값에 해당하는 칸에 stack_info를 저장함. 또한, call_stack에 값을 저장함.
- 원래 for문 안에 배열의 크기만큼 반복을 통해 배열 크기 낭비를 하지 않으려 했으나.. Sizeof(배열의 이름) 을 하면 항상 포인터의 크기인 8 바이트가 나와서 실패 그냥 넉넉하게 과제에서 주어진 최댓값인 20만큼 for 을 실행하도록 바꿈

ClearFP 함수

- SP값과 FP값을 같게 초기화. 왜 이렇게 했는지는 func1 함수에서 설명하겠습니다.

```
void clearFP(void)
{
    FP = SP;
}
```

Pop 함수

- SP의 값을 -1씩 함으로써, pop 기능을 구현했음. FP값은 왜 -2씩 하는지는 뒤에서 func1에서 설명하겠습니다.

```
void pop(void)
{
    SP -= 1;
    FP -= 2;
}
```

Func 함수

Pop 함수와 clearFP 함수에서 설명 건너뛴 FP 값의 조정의 이유를 여기서 설명하자면,

함수의 프롤로그 과정에서 함수 안에서의 지역변수가 지정되기 전에 SP=FP 를 한 후에 지역변수가 들어올 때에는 FP값을 움직이지 않음. 따라서, 함수를 구분할 수 있는 ebp 가 설정되게 됨.

또한, 함수의 프롤로그에서는 POP되면서 작아지는 SP의 크기보다 1 더 작아지도록 설정함.

따라서, 각각의 에필로그에서 맨 마지막 팝의 직전에 clearFP를 설정

```
void func1(int arg1, int arg2, int arg3)
{
    push("arg3", arg3); //전달인자 순서대로 push
    push("arg2", arg2);
    push("arg1", arg1);
    push("Return Address", -1); //반환주소 push
    push("func1 SFP", -1);
    int var_1 = 100;
    clearFP();
    push("var_1", var_1);
    // func1의 스택 프레임·형성 (함수 프롤로그 + push)
    print_stack();
    func2(11, 13);
    // func2의 스택 프레임 제거 (함수 에필로그 + pop)
    pop();
    pop();
    pop();
    pop();
    clearFP();
    pop();

    print_stack();
}
```

Main 함수

- 프롤로그의 시작과 에필로그의 끝

```
nt main()


    func1(1, 2, 3);
    // func1의 스택 프레임 제거 (함수
    pop();
    pop();
    pop();
    pop();
    pop();
    pop();
    print_stack();
    return 0;
```

작동 원리 (프롤로그)

- 함수가 호출되고 파라미터가 스택에 쌓임. (push 를 통해)
- 이때, 파라미터의 값이랑 이름이 같이 쌓임. 이름은 배열로 받았음.
- 그 후에 FP설정.
- 이후에 지역변수가 쌓임.
- 그리고 현재 스택을 보여줌.

작동 원리(에필로그)

- Func3 이 끝난 후에 func2 func1 그리고 메인까지 가면서 pop 이 됨.

- Pop 이 되면서 push 되기 전으로 스택이 원상복귀 됨.