Cykor 1주차
메모리 stack 구조 구현

2025350002 스마트보안학부 권기완

1. 과제 목적

이번 과제의 목표는 함수가 실행될 때 메모리에 함수 스택 프레임이 어떻게 쌓이고(push), 종료될 때 어떻게 제거(pop)되는지를 직접 구현을 통해 이해하는 것이었다. 이를 통해 함수 프롤로그와 에필로그 순서를 재현하는 것을 목표로 하였다.

2. 개념 정리(영상 기반)

Push - Stack에서 데이터를 쌓음
Pop - 가장 위에 쌓인 데이터를 빼냄
LIFO - 스택에서 마지막으로 push한 데이터가 pop 실행시 가장 먼저 빠지는 형태
stack의 장점 - 프로세스의 함수 호출 상황에서 데이터 관리를 쉽게 할 수 있게 함
Call stack - 컴퓨터 프로세스의 함수에 대한 정보를 저장하는 자료구조
Stack Frame - 함수가 호출될 때 스택에 생성되는 하나의 독립적인 블록으로 함수의 정보를 저장함 (함수의 인자, 함수가 끝나고 반환할 주소, 지역변수 공간, 이전 스택프레임의 EBP 값)
함수를 호출 할 때마다 호출된 함수의 Stack Frame이 쌓이게 됨
Stack/Frame Pointer - 함수가 호출될 때 스택에 생성되는 하나의 독립적인 블록
EBP - 현재 함수의 스택프레임 기준점의 주소(포인터)를 저장하는 레지스터(메모리에 저장된 함수의 스택프레임을 가리킴)
ESP - 실시간 변하는 스택의 최상단 주소(포인터)를 저장하는 레지스터
FP - 현재 실행 중인 함수의 프레임 기준점이 되는 포인터, 잘 변하지 않음(EBP에 저장되어 있음)
SP - 스택의 최상단을 가리키는 포인터, 실시간으로 변함(ESP에 저장되어 있음)
Saved Frame Pointer - 새로운 함수가 호출될 때 이전 함수의 프레임 포인터를 저장하는 포인터(스택에 저장됨)
EBP에는 현재 함수 정보만 저장돼 있고, 새 함수를 호출할 때는 현재 레지스터 값을 메모리(스택)에 백업(push)해서 저장한다.
함수 프롤로그
1) 함수가 실행될 때 스택 프레임을 만드는 과정
2) 매개 변수 저장
3) 반환 주소값, 이전 함수의 FP값(SFP)를 저장, 새로 호출한 함수의 FP값을 SP(SFP) 값으로 갱신

4) 지역변수들의 전체 크기에 맞게 SP를 미리 새로 설정
5) 지역변수를 저장
함수 에필로그 - 함수가 종료될 때 호출 이전 상태로 스택을 복원하는 과정
1) 지역변수를 제거
2) SP값을 FP값으로 갱신
3) FP에 저장된 주소인 SFP를 통해 이전 FP값 복원함
4) SP를 내려 SFP를 제거함
5) 반환주소값을 통해 호출한 코드로 돌아가면서 반환주소값을 제거함
6) 남은 매개변수는 Caller가 제거함(C, C++기준)

3. 코드 구현

이 코드는 func1, func2, func3 순으로 push하고 func3, func2, func1 순으로 pop이 되는 구조이다.

push단계에서는 함수 프롤로그를 구현하기 위해 매개 변수 push, 반환 주소값 push, 이전 함수의 FP값(SFP)를 push, 새로 호출한 함수의 FP값을 SP(SFP)값으로 갱신, 지역변수들의 전체 크기에 맞게 SP를 미리 새로 설정, 지역변수를 push하는 순으로 진행되도록 구현하였다. 한 번 push할 때마다 SP값을 하나씩 올려 계속 위로 쌓이도록 구현하였다.

pop단계에서는 지역변수를 제거, SP값을 FP값으로 갱신, FP에 저장된 주소인 SFP를 통해 이전 FP값 복원, SP를 내려 SFP를 제거, 반환주소값을 통해 호출한 코드로 돌아가면서 반환주소값을 제거, 매개변수를 제거 순으로 진행되도록 구현하였다. 지역변수 제거는 잘 드러나진 않지만 반복문 안의 제거 과정에서 지워지게 구현하였다. 다만 매개 변수 제거는 C기준 caller함수가 제거해야 하나 구현이 어려워 pop_func 함수에서 제거하도록 구현하였다.

4. 과제 중 고민한 점

문자열을 call_stack 배열에 넣는 것이 어려워서 검색을 해 보니 strcpy라는 함수를 이용하여 편리하게 문자열을 배열에 넣을 수 있다는 것을 알게 되어 이용하였다.
새로운 func함수를 시작할 때 SP값과 FP값을 다시 초기화하고 시작해야 하는지 헷갈렸지만 stack이 계속해서 쌓이는 구조라는 것을 기억하고 초기화하면 안 된다는 것을 알게 되었다.
func3의 push과정에서 지역변수가 2개라서 SP값을 미리 2칸 올려서 저장하였다.
pop_func3에서 처음에 SP를 내릴지 말지 고민했으나 SP가 이미 값이 저장되어있는 가장 높은 stack지점을 가리킨다는 것을 알고 값을 내리지 않았다.
pop과정에서 SFP를 어떻게 다시 불러와야 할지가 어려웠으나 SFP값은 현재 FP가 가리키는 배열의 지점 안에 저장되어 있다는 것을 알고 이용하였다.
원래 pop을 일일이 지우는 구조로 가려고 했으나 지우는 것이 너무 반복되어서 for

문을 이용하여 제거하였다.
arg 매개변수가 오른쪽부터 먼저 저장된다는 것을 간과했다가 잘못한 것을 깨닫고 다시 설계하였다.

5. 느낀점

메모리의 stack 영역에서 어떤 방식으로 스택프레임이 쌓이고 지워지게 되는지 알 수 있었다. 스택과정에서 SFP라는 것이 매우 중요한 것 같았다. SFP를 통해 FP와 SP값을 조정하는 것이 이루어진다는 것을 알게 되었다. stack에 지역변수가 쌓인다는 것만 알고 있었는데 어떤 방식으로 쌓이는 지 직접 구현해 보는 좋은 경험이 되었다.

예시 출력