

| Team EmulParty |

EmulParty Advance

: beyond x86 for beginners

경량 ISA 기반 오픈소스 보안 실습 아키텍처

이재원(PM) · 조용진(개발 총괄)

문석주 멘토 · 장형범 PL
서희영 · 강다운 · 최민준 · 박정은 · 한상기

GIF/영상은
하단 링크를 참고해주세요

발표 자료 링크 : <https://docs.google.com/presentation/d/1rDqCGlxBH0cDRgvbriqvA1aKglxgi5tfms-ejChVf1E/edit?usp=sharing>



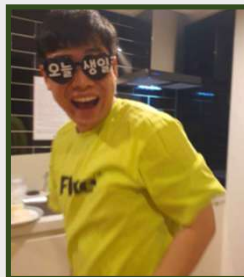
화이트햇 스쿨
WhiteHat School



에뮬레이터Emulator × 아모르파티AmorFati

에뮬르파티

화이트햇 스쿨 3기 팀 프로젝트
「미니 에뮬레이터 만들기」



문석주 Mentor



장형범 PL



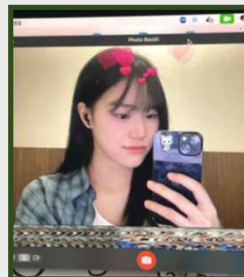
이재원 PM



조용진 Dev.



서희영 Dev.



강다운 Dev.



최민준 Dev.



박정은 Dev.



한상기 Dev.

발 표

발 표

01 프로젝트 개요

02 프로젝트 수행 과정

- ① Chip-8 Emulator
- ② 32-bit extension
- ③ r/w Syscall
- ④ Stack Frame
- ⑤ [실험] Stack BOF

03 취약점 시연

Stack BOF 실험 시연

04 오픈소스 프로젝트 발전

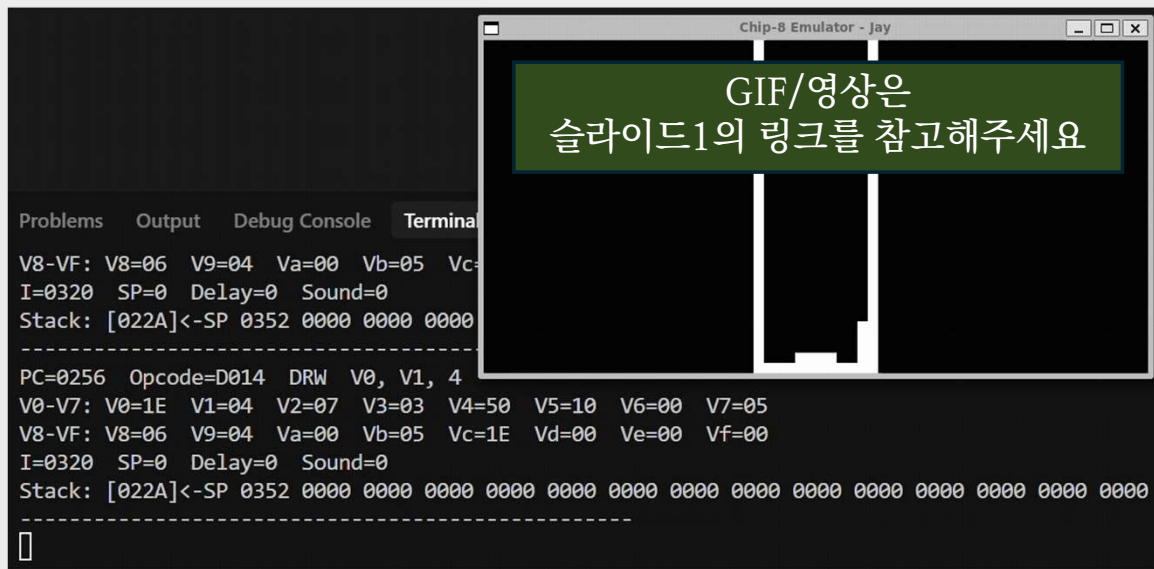
05 결론



1. 프로젝트의 목표와 진행 방향



2.1. Chip-8 Emulator



▲ Chip-8 Emulator 디버거 실행 장면 (tetris.ch8 ROM)

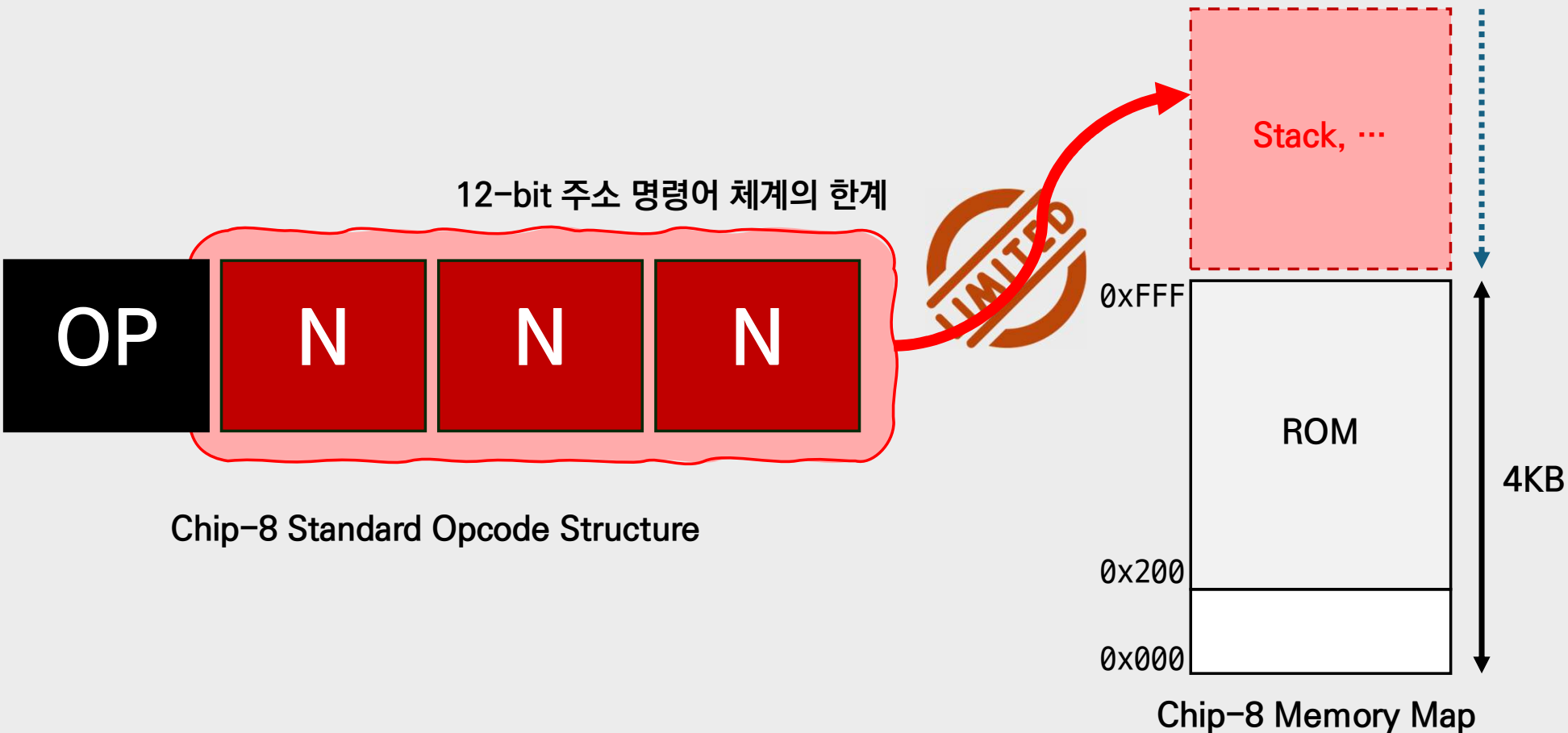
```
PC=0200 Opcode=A2B4 LD I, 02B4
V0-V7: V0=00 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00
V8-VF: V8=00 V9=00 Va=00 Vb=00 Vc=00 Vd=00 Ve=00 Vf=00
I=0000 SP=0 Delay=0 Sound=0
Stack: [0000]<-SP 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

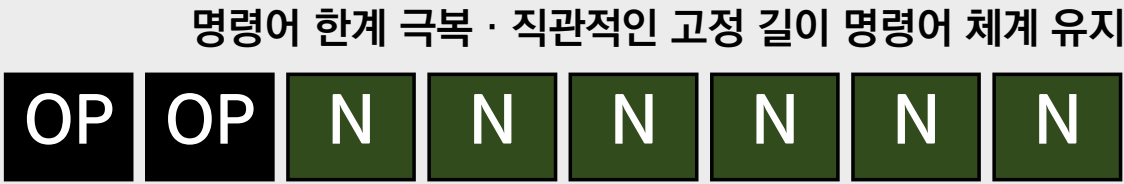
A2B4	LD I, addr	ANNN - 인덱스 레지스터 I에 0xNNN을 로드
------	------------	------------------------------

인덱스 레지스터 I에 0x2B4를 로딩한다.

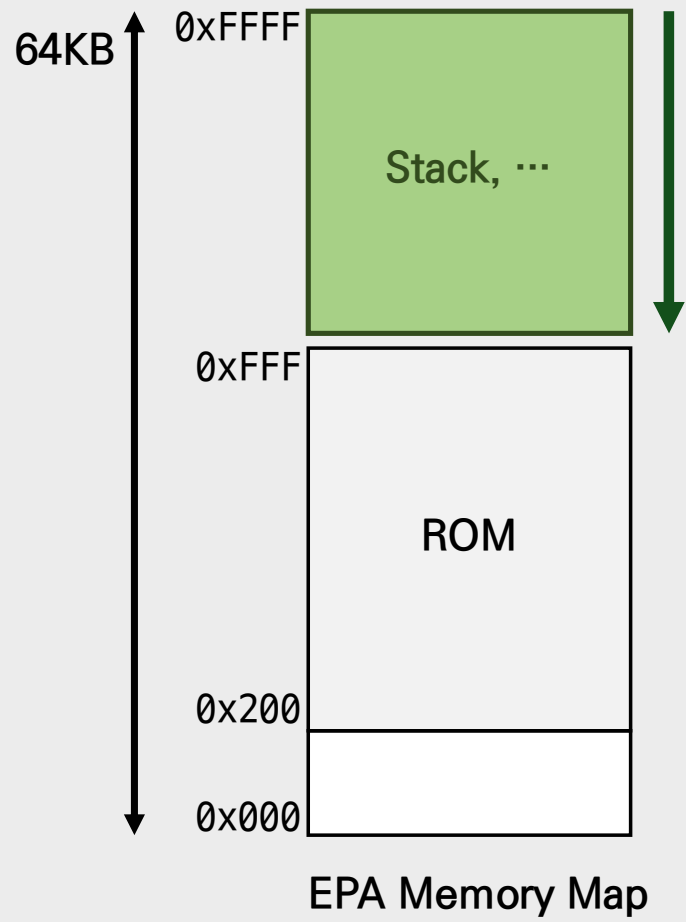
PC-0202 Opcode=22E5 CALL 0350									
V0-V7: V0=00 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0									
Stack: [0000]<SP									
PC-0304 Opcode=2286 CALL 0350									
V0-V7: V0=19 V1=19 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0 Sound=0									
Stack: [0000]<SP									
23E6	C								
서브루틴 0x3E5으로									
2286 CALL									
서브루틴 0x205으로									
PC-0105 Opcode=V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0204 [0000]									
6A00	LD								
상수 0을 레지스터									
6705 LD Vx									
상수 0을 레지스터									
PC-0101 Opcode=V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0204 [0000]									
6019	LD								
상수 0을 레지스터									
6800 LD Vx									
상수 0을 레지스터									
PC-0104 Opcode=V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
6019	LD								
상수 0을 레지스터									
6800 LD Vx									
상수 0을 레지스터									
PC-020A Opcode=00E8 RET									
V0-V7: V0=19 V1=19 V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=00 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0 Sound=0									
Stack: 0206 [0000]<SP									
00E8	RET								
서브루틴에서 복귀									
PC-020E Opcode=7001 AND Vx, 01									
V0-V7: V0=19 V1=19 V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=0011 DRW Vx, V1, 1									
V0-V7: V0=1A V1=1F V2=07 V3=00 V4=00 V5=10 V6=00 V7=05									
V8-VF: V8=06 V9=04 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=0 Delay=0 Sound=0									
Stack: [0200]<SP									
001F	LD Vx								
상수 1을 레지스터									
PC-020C Opcode=611F V0-V7: V0=19 V1=00 V2=00 V3=00 V4=00 V5=00 V6=00 V7=00									
V8-VF: V8=00 V9=00 V10=00 V11=00 V12=00 V13=00 V14=00 V15=00									
I-0204 SP=1 Delay=0									
Stack: 0206 [0000]<SP									
7001	AND Vx, byte								
레지스터 V0에 상수 01을 대입									
PC-0208 Opcode=001									

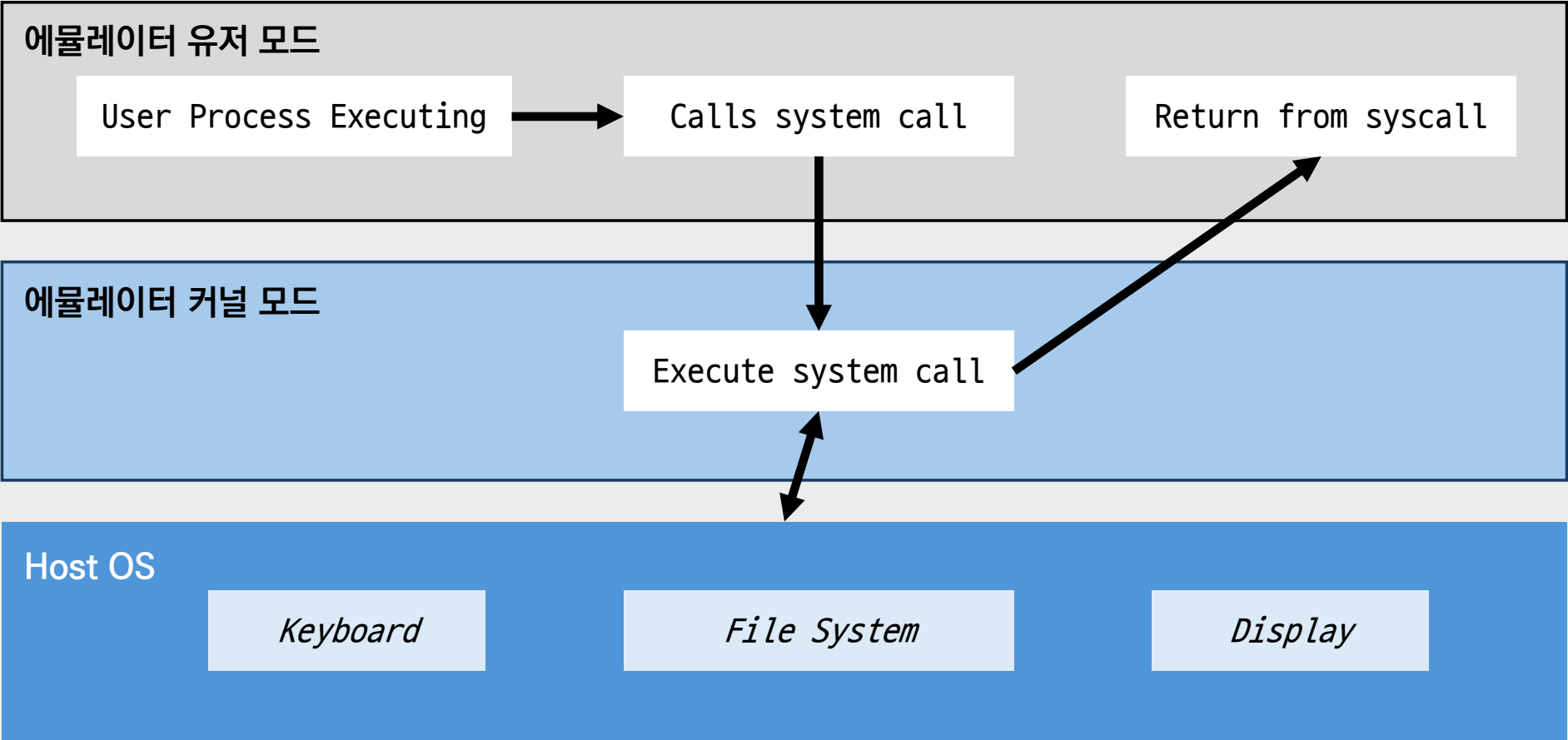
디버거 & ROM 실행
→ 동작 분석 가능





EPA Extended Opcode Structure

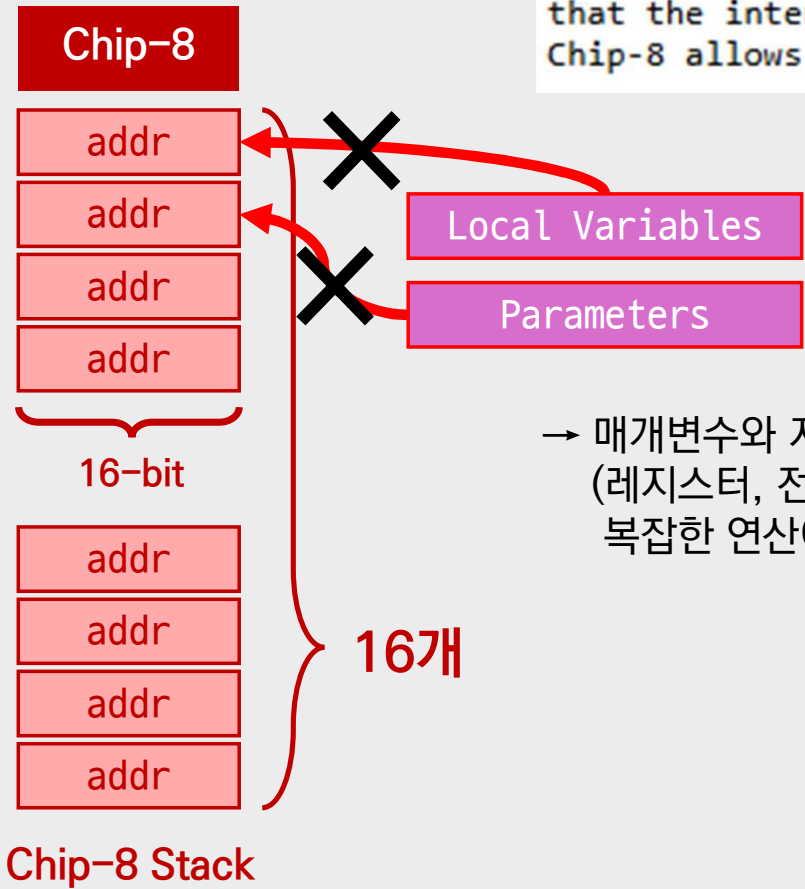




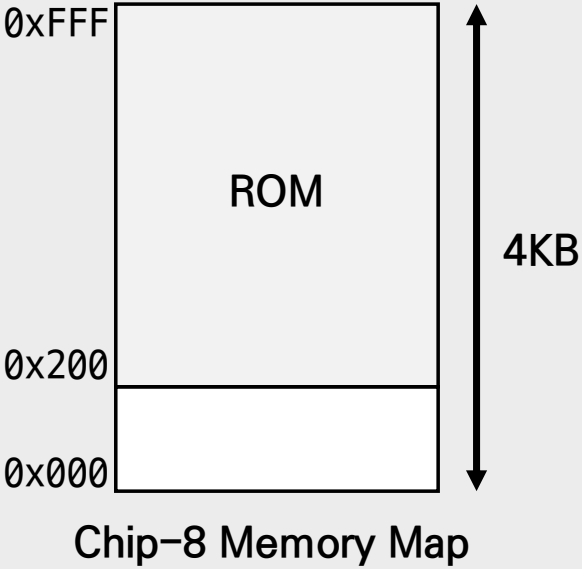


2.4. Stack Frame

The stack is an array of 16 16-bit values, **used to store the address** that the interpreter should return to when finished with a subroutine. Chip-8 allows for up to 16 levels of nested subroutines.



→ 매개변수와 지역변수 저장 불가
(레지스터, 전역 메모리 과부하로
복잡한 연산이나 재귀 호출 어려움)



Stack Frame의 본질

EBP를 기준으로 “각 함수 호출마다 독립된 작업 공간“ 확보

x86-64의 호출 규약을 참고(caller cleanup)

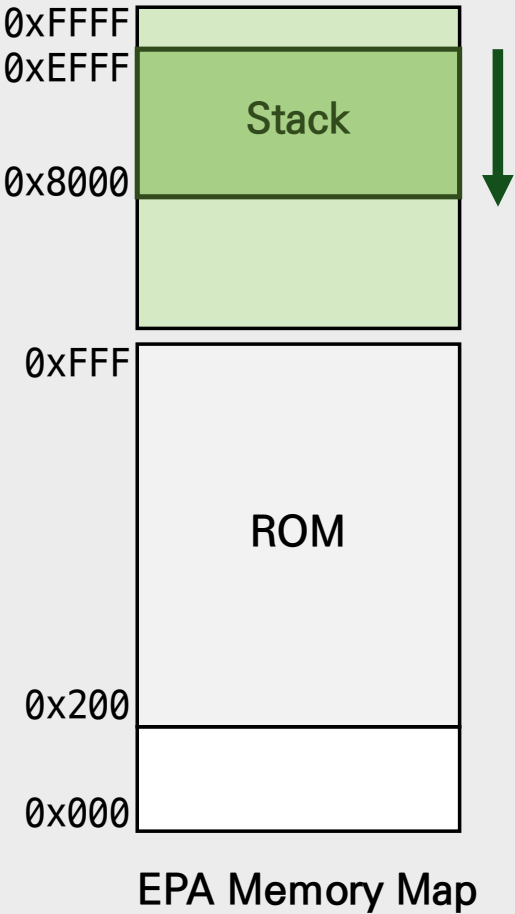
프레임 프롤로그

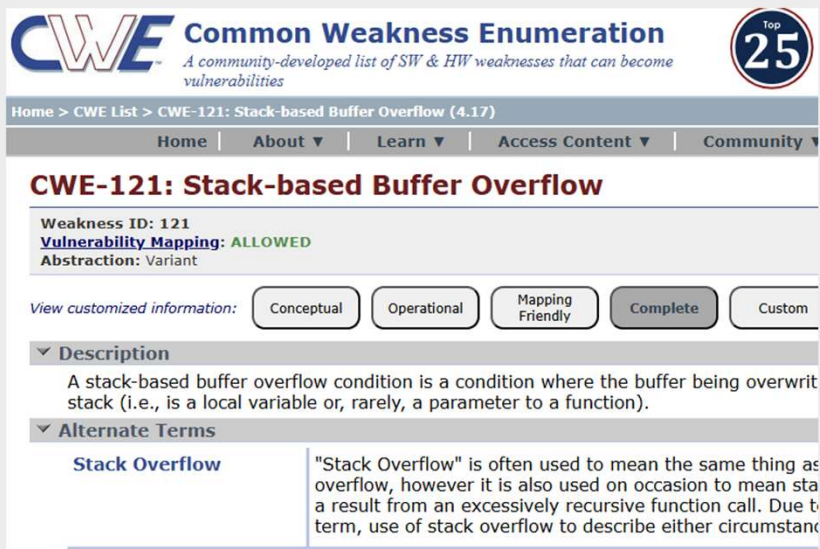
```
push ebp
mov  ebp, esp
sub  esp, <locals>
```

프레임 에필로그

```
mov  esp, ebp
pop  ebp
ret  <n>
```

- ▶ 중첩된 호출이 일어나도 함수마다 독립된 스택 구역 사용
- ▶ 복귀 시 ebp 체인을 따라가며 콜 스택을 추적할 수 있음





✓ 주어진 환경에서 실험 가능한가?

Read Syscall

Stack Frame

Write Syscall

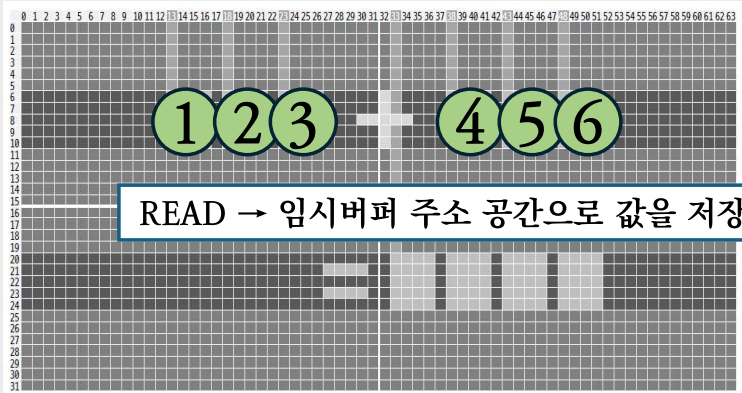
✓ 지금까지 구현한 디버거 사양으로 원리를 보여줄 수 있는가?

레지스터

스택 프레임

스택에 할당된 버퍼(지역변수나 함수 인수)가 고정 크기를 초과하는 입력으로 다른 메모리를 덮어쓰면서, 그 뒤에 저장된 중요한 데이터를 변조할 수 있는 상태.

목표 : READ & 스택 프레임 동작을 활용해서 덧셈 계산기 만들기 (*실험 고려)



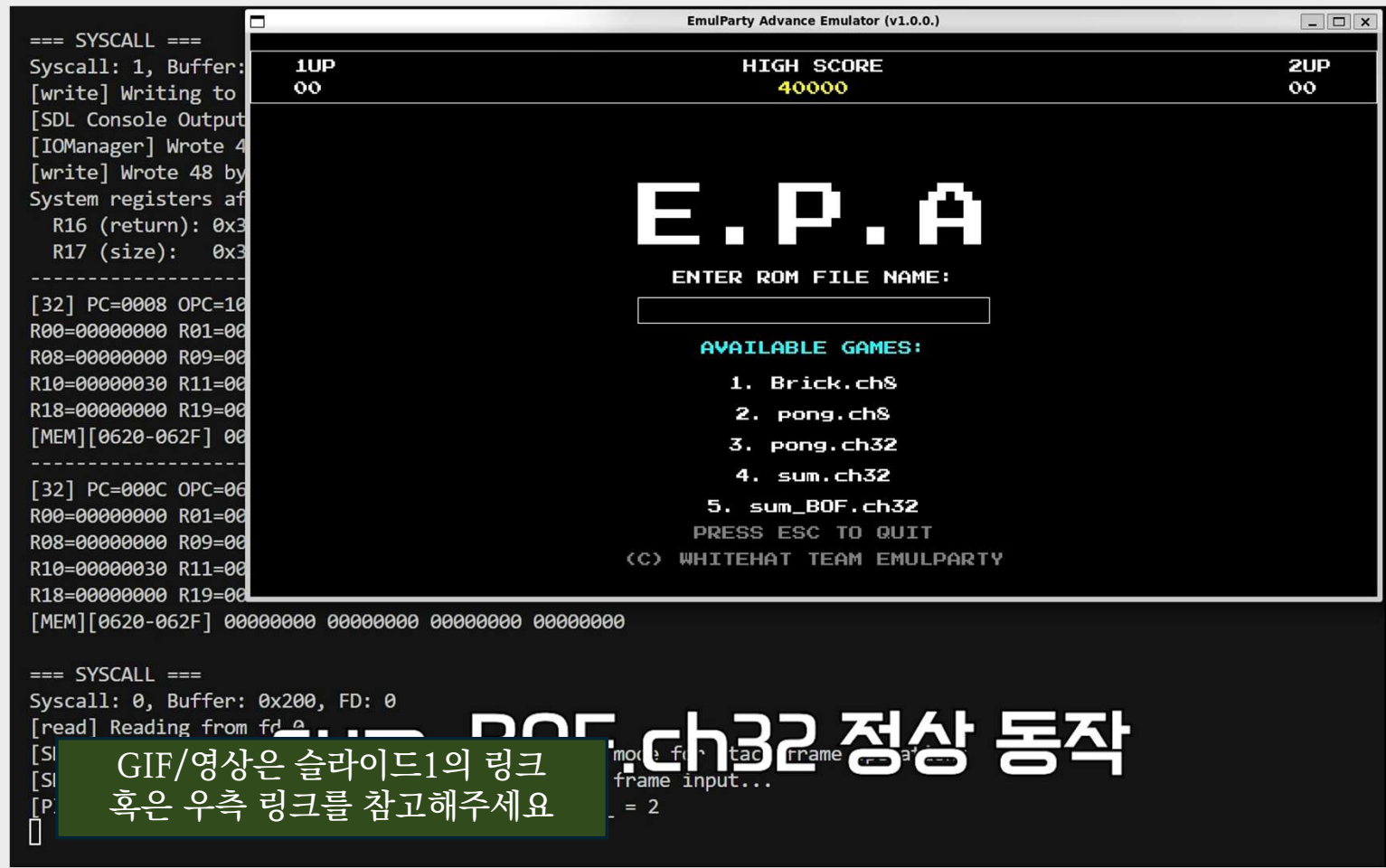
Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0600	01	02	03													
0610																
0620	04	05	06													
0630																

E(Enter) 키로 sum 동작을 스택 프레임에서 수행

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
EFE0																
EFF0	a	r	g	2	a	r	g	1	이	전	B	P	리	턴	주	소
	04	05	06	-	01	02	03	-	00	00	F0	00	00	00	?	?

[READ] 키패드 입력 → 임시 버퍼 주소 공간에 값 저장 → 스택 프레임으로 입력 값 복사
→ CALL Func(덧셈 함수) → 스택 프레임 내의 인자 접근 및 계산 → 결과 리턴
(* x86 함수 호출 규약상 함수 인자는 호출 전에 스택에 PUSH, CALL 시에 스택 프레임 생성)

3.1. 정상 동작 (시연 영상)



[HOW TO PLAY sum_BOF.ch32]

- ▼ 피연산자를 나란히 입력
- ▼ E('F'에 맵핑, Enter)키로 계산

<https://youtu.be/tdlNDdrqbrM>

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
00000000:	06	00	00	1E	06	01	00	06	0A	00	02	14	00	00	01	05
00000010:	01	00	02	1C	20	20	F8	20	20	00	00	00	06	00	00	18
00000020:	06	01	00	14	0A	00	02	2C	0D	00	01	05	00	F0	00	F0
00000030:	00	00	00	00	0F	07	00	0A	06	00	00	00	06	01	00	06
00000040:	0F	07	02	09	00	00	01	05	06	0E	06	24	20	07	0E	00
00000050:	0F	08	00	0A	06	00	00	12	0F	08	02	09	00	00	01	05
00000060:	06	0E	06	25	20	08	0E	00	0F	09	00	0A	06	00	00	17
00000070:	0F	09	02	09	00	00	01	05	06	0E	06	26	20	09	0E	00
00000080:	0F	0A	00	0A	06	00	00	26	0F	0A	02	09	00	00	01	05
00000090:	06	0E	06	20	20	0A	06	00	00	00	00	0A	06	00	00	2B
000000A0:	00	00	00	00	00	00	00	00	00	00	00	01	20	08	00	00
000000B0:	00	00	00	06	00	00	00	00	00	00	00	00	00	00	00	00
000000C0:	07	0E	00	01	20	0C	0E	00	0F	00	00	0A	04	00	00	0E
000000D0:	01	00	02	F0	06	00	00	35	0F	00	02	09	00	00	01	05
000000E0:	07	0E	00	01	20	00	0E	00	00	00	01	05	01	00	02	C8
000000F0:	21	07	06	24	21	08	06	25	21	09	06	26	01	00	06	2C	!..\$!..%!..&...;
00000100:	06	0F	00	00	08	09	0C	04	06	00	00	30	06	01	00	14
00000110:	0F	09	02	09	00	00	01	05	08	08	0F	04	06	0F	00	00
00000120:	08	08	08	04	06	00	00	28	0F	08	02	09	00	00	01	05
00000130:	08	07	0F	04	06	0F	00	00	08	07	0A	04	06	00	00	26
00000140:	0F	07	02	09	00	00	01	05	06	00	00	21	0F	0F	02	09
00000150:	00	00	01	05	01	00	03	54	00	00	00	00	00	00	00	00
00000160:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000170:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

디버거 작동 시 하단 Stack Frame 영역 메모리에 주목

3.2. Local Variable Corruption, Return Address Hijacking (시연 영상)

```
=== SYSCALL ===
Syscall: 1, Buffer:
[write] Writing to
[SDL Console Output
[IOManager] Wrote 4
[write] Wrote 48 by
System registers af
  R16 (return): 0x3
  R17 (size):  0x3
-----
[32] PC=0008 OPC=10
R00=00000000 R01=00
R08=00000000 R09=00
R10=00000030 R11=00
R18=00000000 R19=00
[MEM][0620-062F] 00
-----
[32] PC=000C OPC=06
R00=00000000 R01=00
R08=00000000 R09=00
R10=00000030 R11=00
R18=00000000 R19=00
[MEM][0620-062F] 00000000 00000000 00000000 00000000
-----
=== SYSCALL ===
Syscall: 0, Buffer: 0x200, FD: 0
[read] Reading 0x200 from fd 0
[SI] For stack frame corruption
[SI] frame input...
[P] = 2
```

1UP
00

HIGH SCORE
40000

2UP
00

E.P.A

ENTER ROM FILE NAME:

AVAILABLE GAMES:

1. Brick.ch8
2. pong.ch8
3. pong.ch32
4. sum.ch32
5. sum_BOF.ch32

PRESS ESC TO QUIT

(C) WHITEHAT TEAM EMULPARTY

GIF/영상은 슬라이드1의 링크
혹은 우측 링크를 참고해주세요

[HOW TO PLAY sum_BOF.ch32]

- ▼ 피연산자를 나란히 입력
- ▼ 뒤에 악의적인 값을 입력
- ▼ E('F'에 맵핑, Enter)키로 계산

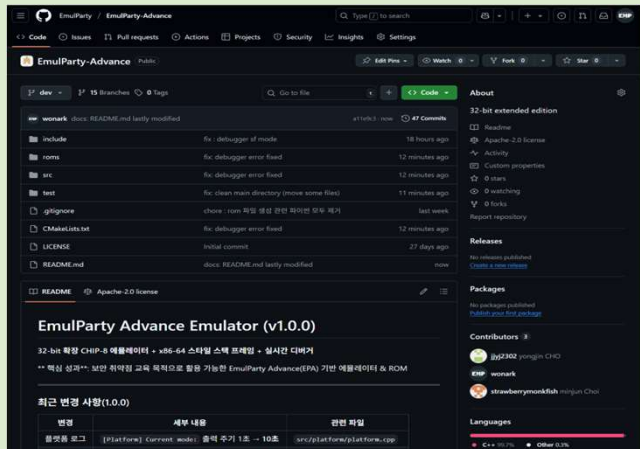
https://youtu.be/zTfB_F_8Ik4

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
00000000:	06	00	00	1E	06	01	00	06	0A	00	02	14	00	00	01	05
00000010:	01	00	02	1C	20	20	F8	20	20	00	00	00	06	00	00	18
00000020:	06	01	00	14	0A	00	02	2C	0D	00	01	05	00	F0	00	F0
00000030:	00	00	00	00	0F	07	00	0A	06	00	00	00	06	01	00	06
00000040:	0F	07	02	09	00	00	01	05	06	0E	06	24	20	07	0E	00
00000050:	0F	08	00	0A	06	00	00	12	0F	08	02	09	00	00	01	05
00000060:	06	0E	06	25	20	08	0E	00	0F	09	00	0A	06	00	00	17
00000070:	0F	09	02	09	00	00	01	05	06	0E	06	26	20	09	0E	00
00000080:	0F	0A	00	0A	06	00	00	26	0F	0A	02	09	00	00	01	05
00000090:	06	0E	06	20	20	0A	06	00	00	00	0A	06	00	00	00	2B
000000A0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000B0:	00	00	00	00	06	00	00	00	00	02	00	00	00	00	00	00
000000C0:	07	0E	00	01	20	0C	0E	00	0F	00	00	0A	04	00	00	0E
000000D0:	01	00	02	F0	06	00	00	35	0F	00	02	09	00	00	01	05
000000E0:	07	0E	00	01	20	00	0E	00	0D	00	01	05	01	00	02	C8
000000F0:	21	07	06	24	21	08	06	25	21	09	06	26	01	00	06	2C	!..\$!..%!..&...;
00000100:	06	0F	00	00	08	09	0C	04	06	00	00	30	06	01	00	14
00000110:	0F	09	02	09	00	00	01	05	08	08	0F	04	06	0F	00	00
00000120:	08	08	08	04	06	00	00	28	0F	08	02	09	00	00	01	05
00000130:	08	07	0F	04	06	0F	00	00	08	07	0A	04	06	00	00	26
00000140:	0F	07	02	09	00	00	01	05	06	00	00	21	0F	0F	02	09
00000150:	00	00	01	05	01	00	03	54	00	00	00	00	00	00	00	00
00000160:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000170:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

디버거 작동 시 하단 Stack Frame 영역 메모리에 주목



GitHub



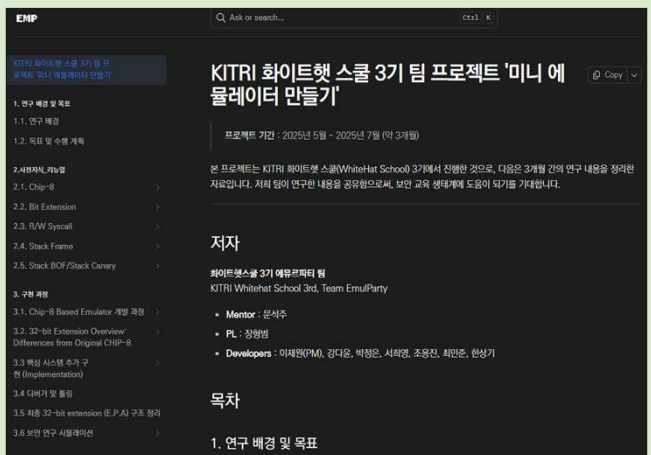
GitHub 메인 페이지

#README.md #LICENSE

EmulParty-Advance

소스코드 공개

GitBook



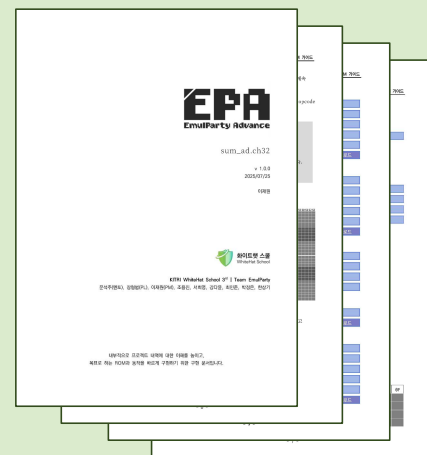
GitBook 메인 페이지



#연구배경및목표
#사전지식 #구현과정
#결론

프로젝트 내역 공개

Docs



ROM 가이드 문서

#ROM설계법 #도표설명포함
#Custom명령어활용법

입문자 학습용 문서 공개





| Team EmulParty |

EmulParty Advance

: beyond x86 for beginners

경량 ISA 기반 오픈소스 보안 실습 아키텍처

Team EmulParty

문석주 멘토 · 장형범 PL

이재원 · 조용진 · 서희영 · 강다윤 · 최민준 · 박정은 · 한상기

QnA

질의응답



화이트햇 스쿨
WhiteHat School