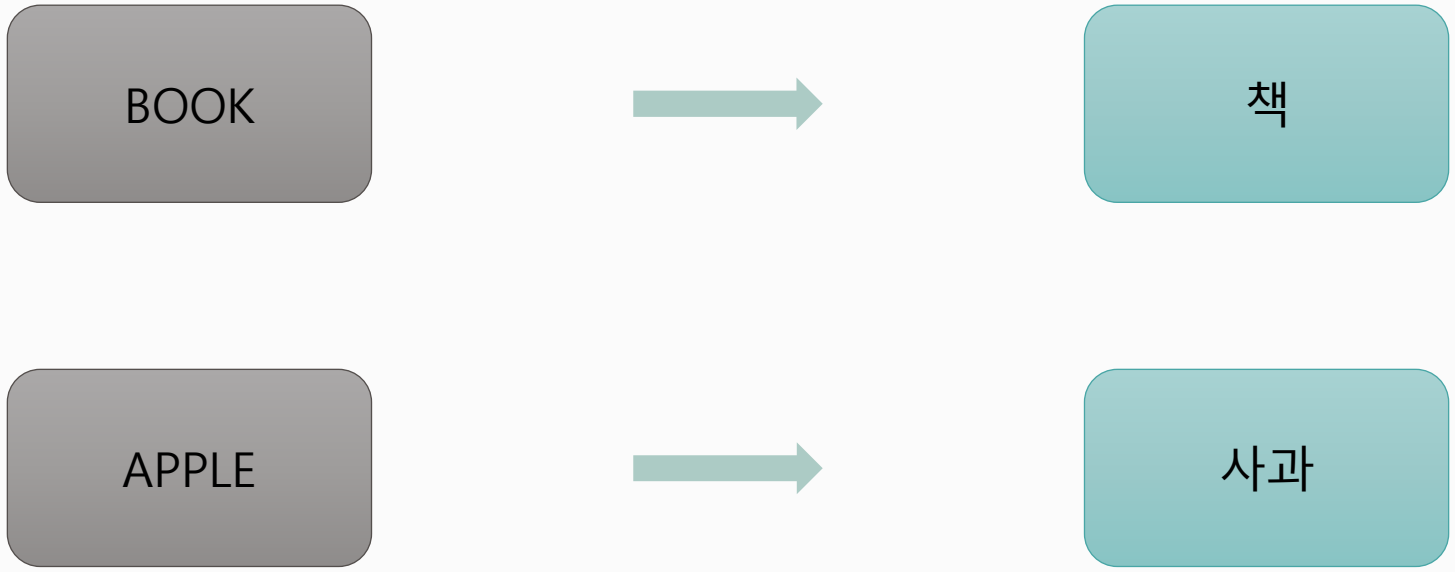
어셈블리어 기초

신재형

1. 어셈블리어란?

기계어 어셈블리어

1. 어셈블리어란?



1. 어셈블리어란?

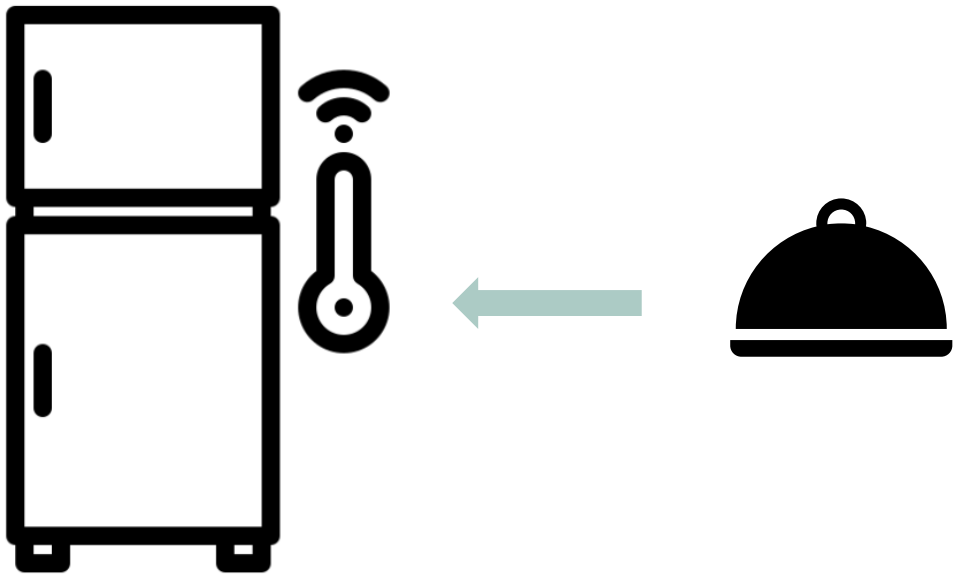
어셈블리어

C, C++, Java, Python

Low Level Language

High Level Language

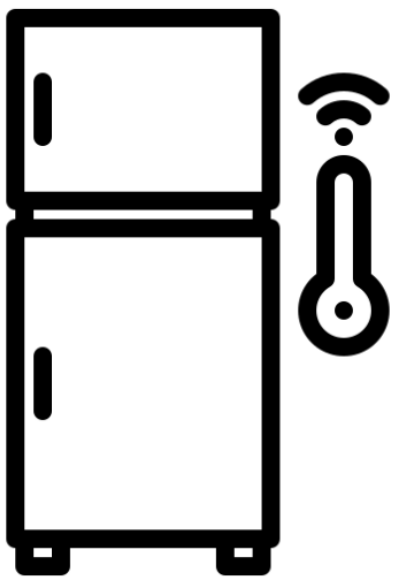
1. 어셈블리어란?



1. 냉장고 문을 연다.
2. 냉장고에 음식을 넣는다.
3. 냉장고 문을 닫는다.

Low Level Language

1. 어셈블리어란?



1. 냉장고 문을 연다.
2. 냉장고에 음식을 넣는다.
3. 냉장고 문을 닫는다.

냉장고에 음식 넣어

High Level Language

Complier

2. 어셈블리어의 문법

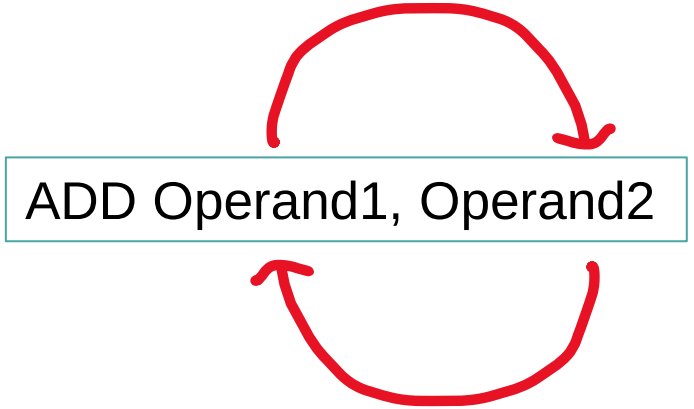
공통점

Opcode Operand1, Operand2

AT&T = Intel

2. 어셈블리어의 문법

차이점: 산술연산



AT&T ≠ Intel

2. 어셈블리어의 문법

차이점: 숫자표기

$1 $2 $3 $4 $5
$6 $7 $8 $9 $0

1 2 3 4 5
6 7 8 9 0

AT&T ≠ Intel

2. 어셈블리어의 문법

메모리주소

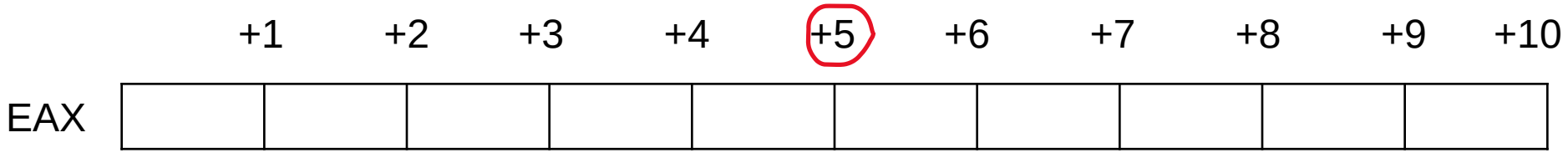
(EAX)

[EAX]

AT&T ≠ Intel

2. 어셈블리어의 문법

오프셋



5(EAX)

[EAX+5]

AT&T ≠ Intel

3. 레지스터

범용 레지스터

- EAX
  산술연산에 사용되는 레지스터
- EBX
  주소를 저장하는 레지스터
- ECX
  카운트하는 레지스터
- EDX
  EAX와 같이 쓰이는 레지스터

3. 레지스터

범용 레지스터

ESI
데이터를 복사나 비교할 때 소스의 출발지의 주소가 저장되는 레지스터

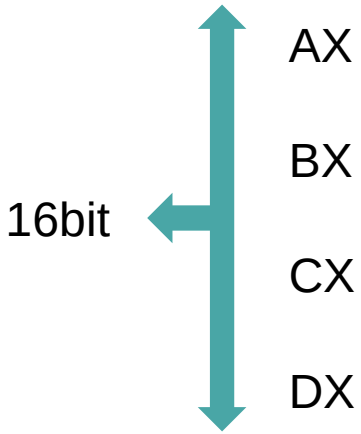
EDI
데이터를 복사나 비교할 때 소스의 목적지의 주소가 저장되는 레지스터

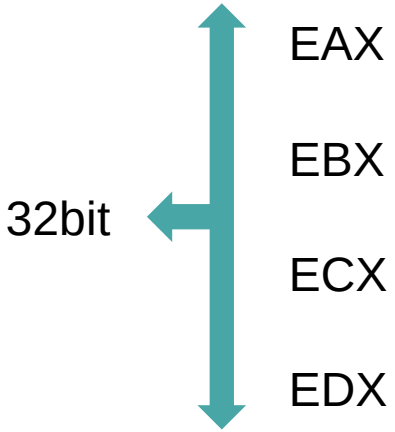
ESP
가장 최근에 생성된 스택 공간의 주소를 저장하는 레지스터

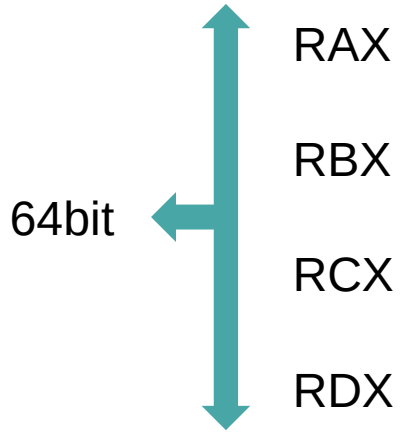
EBP
스택 프레임의 시작 지점 주소가 저장되는 레지스터

3. 레지스터

범용 레지스터



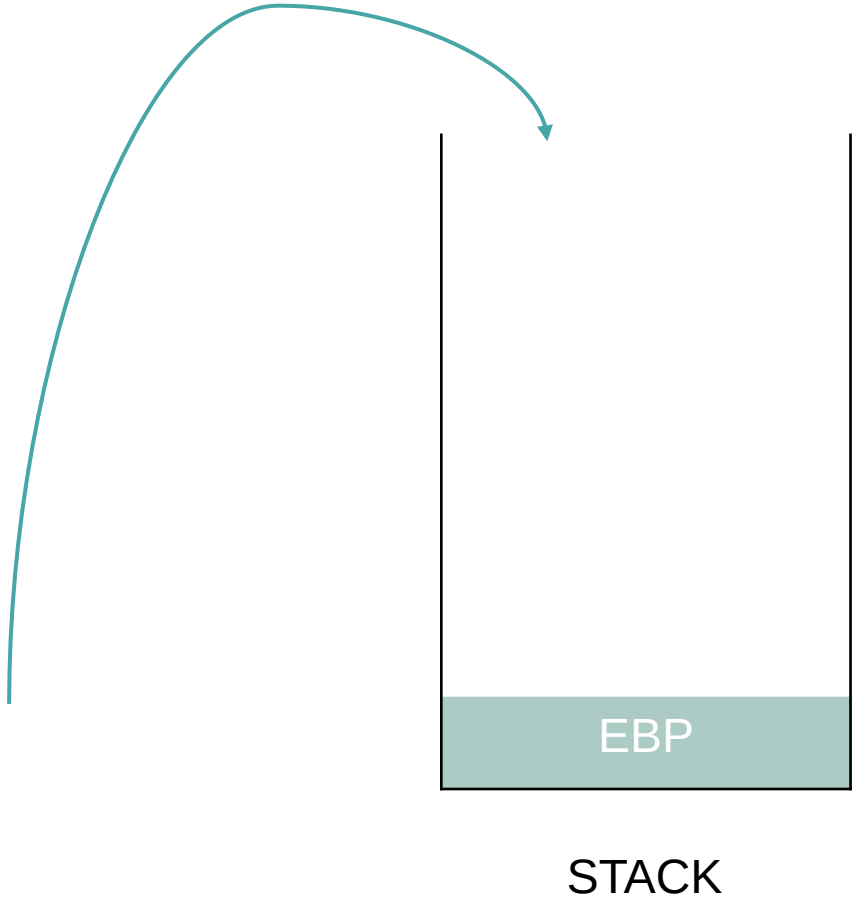




4. Opcode

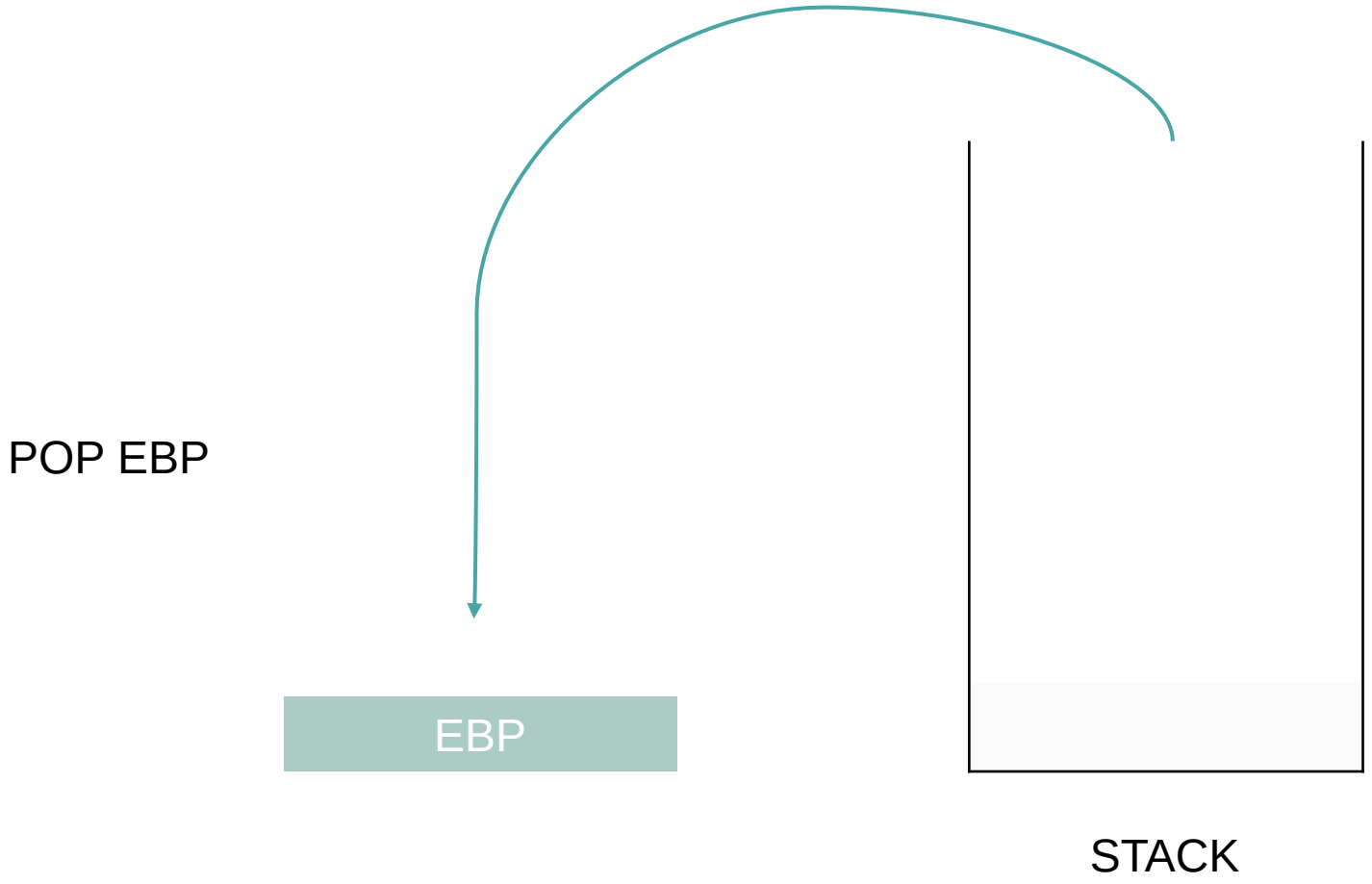
PUSH, POP

PUSH EBP



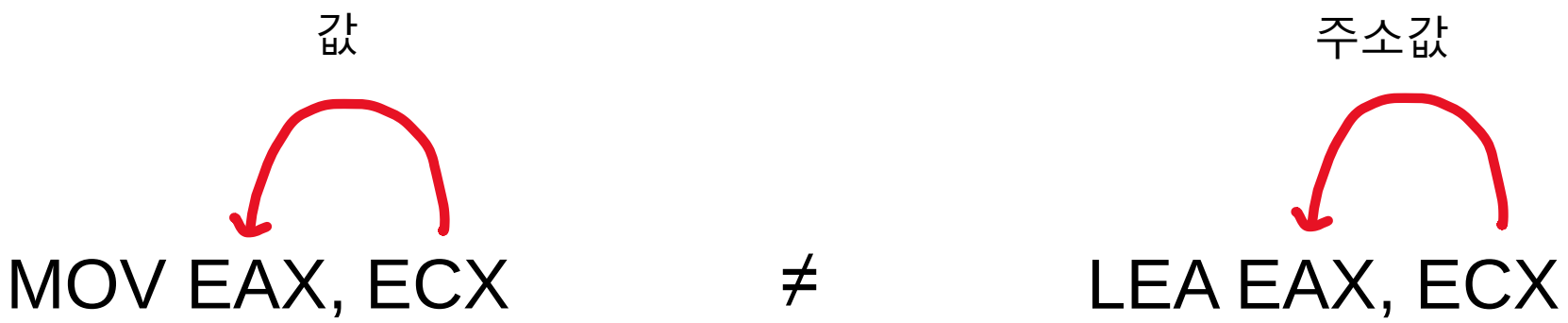
4. Opcode

PUSH, POP



4. Opcode

MOV, LEA



4. Opcode

ADD, SUB

4. 실습

```c
#include <stdio.h>

int main() {
    int a = 5;
    int b = 10;
    int c = a + b;

    printf("a+b=%d\n", c);



    return 0;
}
```

4. 실습

```
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], 5
        mov     DWORD PTR [rbp-8], 10
        mov     edx, DWORD PTR [rbp-4]
        mov     eax, DWORD PTR [rbp-8]
        add     eax, edx
        mov     DWORD PTR [rbp-12], eax
```

```
#include <stdio.h>

int main() {
    int a = 5;
    int b = 10;
    int c = a + b;

    printf("a+b=%d\n", c);



    return 0;
}
```

4. 실습

```
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], 5
        mov     DWORD PTR [rbp-8], 10
        mov     edx, DWORD PTR [rbp-4]
        mov     eax, DWORD PTR [rbp-8]
        add     eax, edx
        mov     DWORD PTR [rbp-12], eax
```

```
#include <stdio.h>

int main() {
    int a = 5;
    int b = 10;
    int c = a + b;

    printf("a+b=%d\n", c);



    return 0;
}
```

4. 실습

```
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], 5
        mov     DWORD PTR [rbp-8], 10
        mov     edx, DWORD PTR [rbp-4]
        mov     eax, DWORD PTR [rbp-8]
        add     eax, edx
        mov     DWORD PTR [rbp-12], eax
```

```
#include <stdio.h>

int main() {
    int a = 5;
    int b = 10;
    int c = a + b;

    printf("a+b=%d\n", c);


    return 0;
}
```

4. 실습

```
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], 5
        mov     DWORD PTR [rbp-8], 10
        mov     edx, DWORD PTR [rbp-4]
        mov     eax, DWORD PTR [rbp-8]
        add     eax, edx
        mov     DWORD PTR [rbp-12], eax
```

```
#include <stdio.h>

int main() {
    int a = 5;
    int b = 10;
    int c = a + b;

    printf("a+b=%d\n", c);



    return 0;
}
```

4. 실습

```c
#include <stdio.h>

int main() {
    int a = 5;
    int b = 10;
    int c = a + b;

    printf("a+b=%d\n", c);


    return 0;
}
```

```
        mov     eax, DWORD PTR [rbp-12]
        mov     esi, eax
        mov     edi, OFFSET FLAT:.LC0
        mov     eax, 0
        call    printf
        mov     eax, 0
        leave
        ret
```

```
.LC0:
        .string "a+b=%d\n"
```

```
a+b=15
```

4. 실습

```c
#include <stdio.h>

int main() {
    int a = 5;
    int b = 10;
    int c = a + b;

    printf("a+b=%d\n", c);


    return 0;
}
```

```
    mov     eax, DWORD PTR [rbp-12]
    mov     esi, eax
    mov     edi, OFFSET FLAT:.LC0
    mov     eax, 0
    call    printf
    mov     eax, 0
    leave
    ret
```

4. 실습

```c
#include <stdio.h>

int main() {
    int a = 5;
    int b = 10;
    int c = a + b;

    printf("a+b=%d\n", c);


    return 0;
}
```

```
    mov     eax, DWORD PTR [rbp-12]
    mov     esi, eax
    mov     edi, OFFSET FLAT:.LC0
    mov     eax, 0
    call    printf
    mov     eax, 0
    leave
    r
```

Mov esr `bp

pop eip

감사합니다