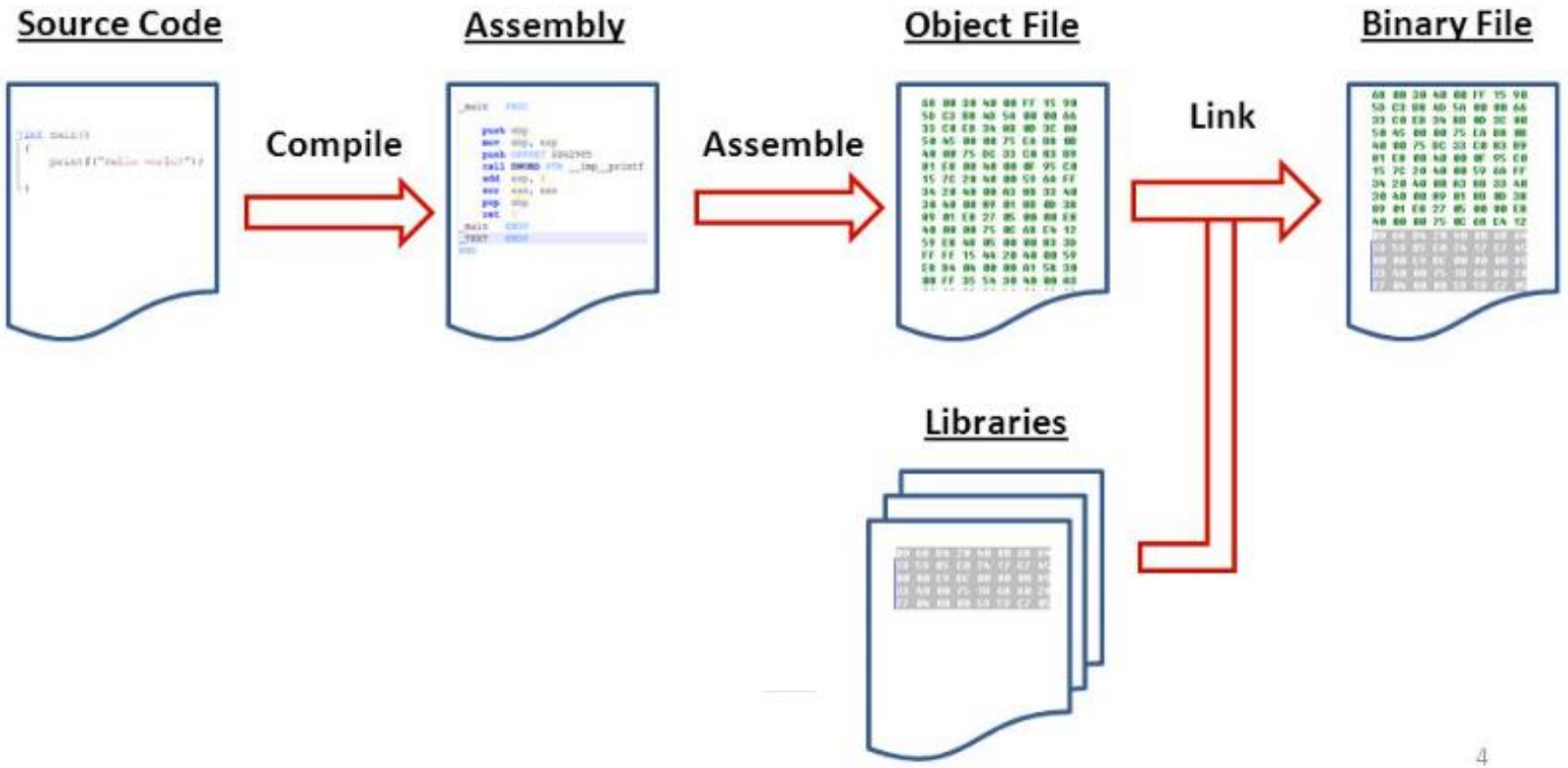
정보보호론 #8

코드 보안 - 추가자료

Prof. Byung Il Kwak

배경지식

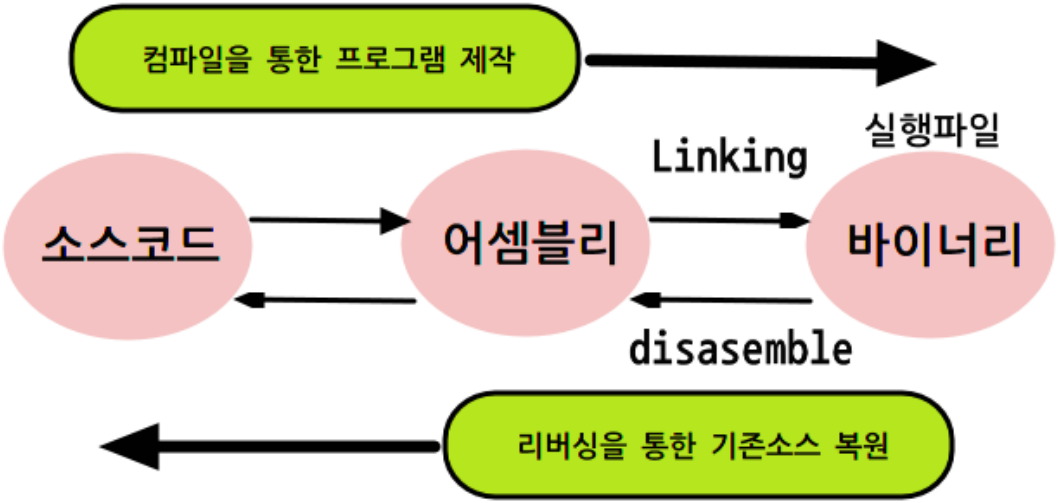


https://slideplayer.com/slide/4695781/

배경지식

https://p3ace.tistory.com/36

https://www.hex-rays.com/products/ida/



BINARY CODE

```
...0010001
1110101111
0101101001
0111010001
1100001100
0001001101
0111010101
11010100...
```

ASSEMBLY CODE

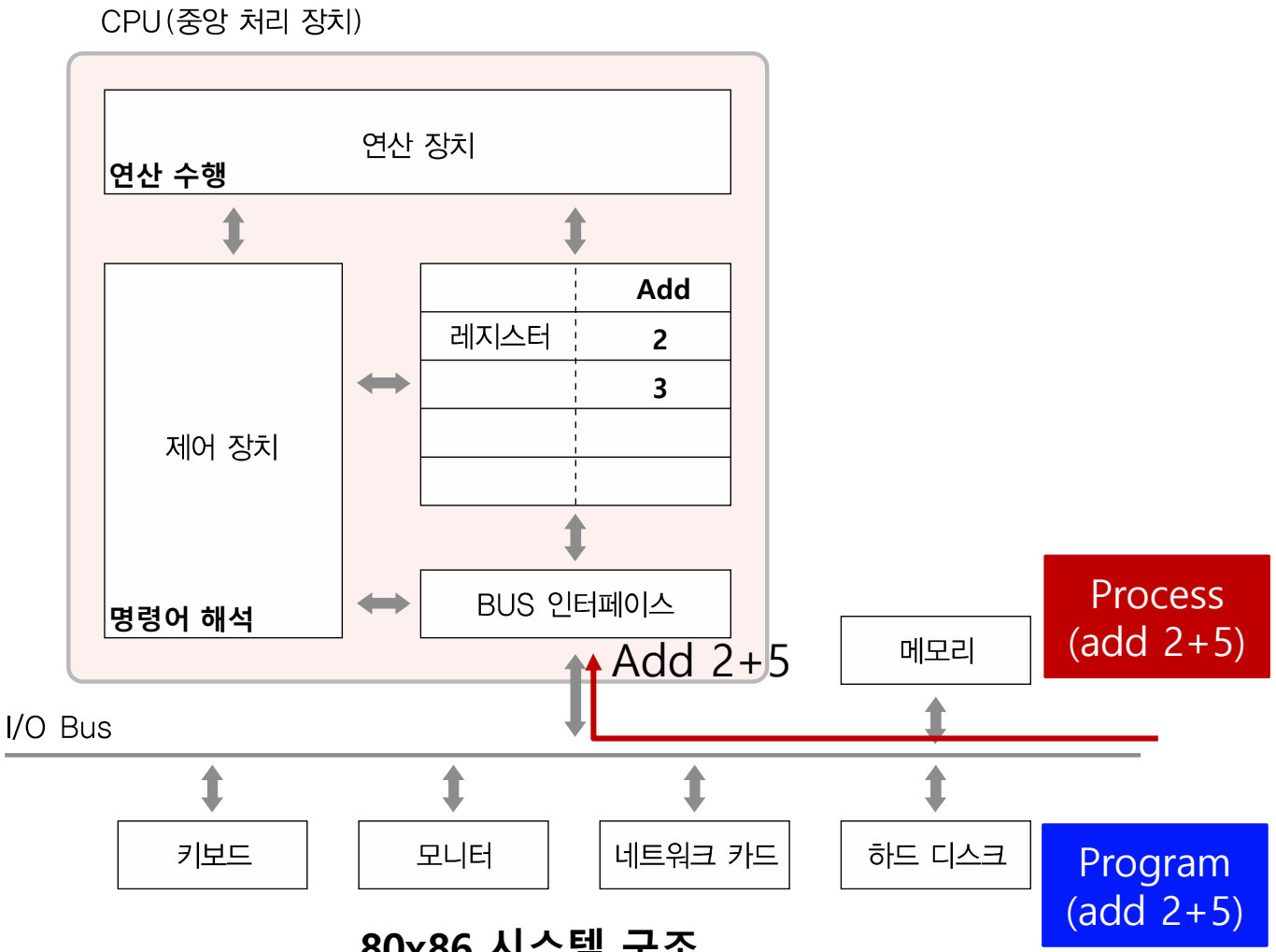
```
...
mov r0,#1
mov r1,#1
l:
add r2,r0,r1
str r2,[r3]
add r3,#4
...
```

HIGH LEVEL CODE

```
i = 1 ; j = 1 ;
while (true) {
  *val++ = i + j ;
  j = i + ( i = j ) ;
}
```

x86 시스템 CPU의 구조와 레지스터

- 80x86 시스템 CPU 구조

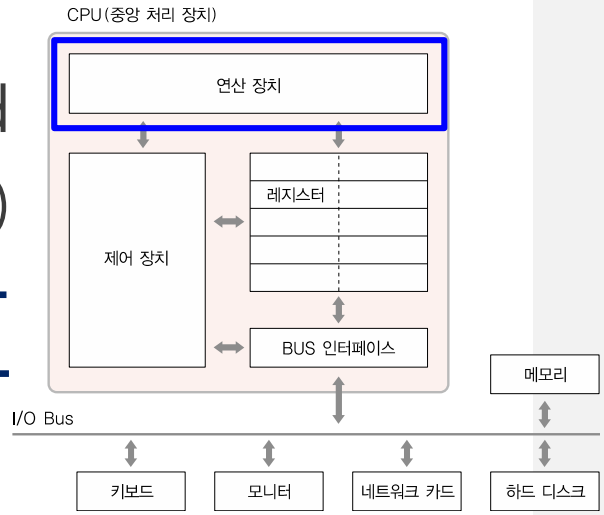


80x86 시스템 구조

x86 시스템 CPU의 구조와 레지스터

연산장치

- 연산 장치(ALU: Arithmetic and Logic Unit)는 CPU(중앙 처리 장치)의 핵심 부분 중 하나로, **산술과 논리 연산을 수행하는 연산 회로 집합으로 구성**

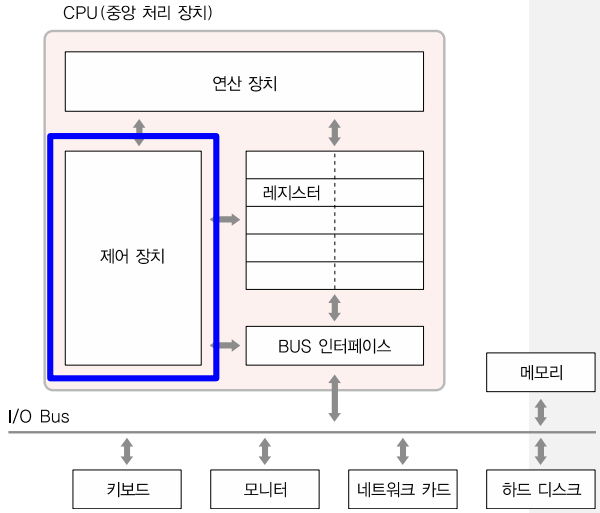


구성 요소		기능
내부 장치	가산기(Adder)	덧셈 연산 수행
	보수기(Complementer)	뺄셈 연산 수행. 1의 보수나 2의 보수 방식 이용
	시프터(Shifter)	비트를 오른쪽이나 왼쪽으로 이동하여 나눗셈과 곱셈 연산 수행
관련 레지스터	누산기(Accumulator)	연산의 중간 결과 저장
	데이터 레지스터 (Data Register)	연산에 사용할 데이터 저장
	상태 레지스터(Status Register)	연산 실행 결과로 나타나는 양수와 음수, 자리올림, 오버 플로우의 상태 기억

x86 시스템 CPU의 구조와 레지스터

제어장치

- 제어 장치(Control Unit)는 **입력, 출력, 기억, 연산 장치를 제어하고 감시, 주기억 장치에 저장된 명령을 차례로 해독하여 연산 장치로 보내 처리되도록 지시**

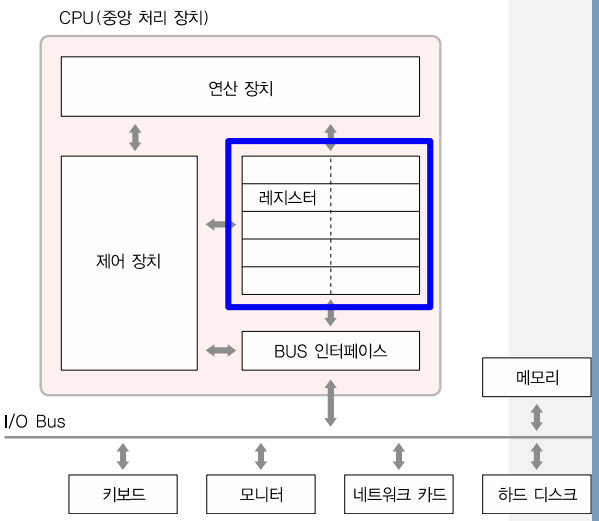


구성 요소		기능
내부 장치	명령 해독기	명령 레지스터에 있는 명령을 해독하여 부호기로 전송
	부호기	명령 해독기가 전송한 명령을 신호로 만들어 각 장치 전송
	주소 해독기	명령 레지스터에 있는 주소를 해독하여 메모리의 실제주소로 변환한 후, 이를 데이터 레지스터에 저장
관련 레지스터	프로그램 카운터	다음에 실행할 명령의 주소 저장
	명령 레지스터	현재 실행 중인 명령 저장
	메모리 주소 레지스터	주기억 장치의 번지 저장
	메모리 버퍼 레지스터	메모리 주소 레지스터에 저장된 주소의 실제 내용 저장

x86 시스템 CPU의 구조와 레지스터

❑ 레지스터의 종류와 용도

범주	80386 레지스터	이름	비트	용도
범용 레지스터 (General Register)	EAX	누산기 (Accumulator)	32	주로 산술 연산에 사용 (함수의 결과 값 저장)
	EBX	베이스 레지스터 (Base Register)	32	특정 주소 저장 (주소 지정을 확대하기 위한 인덱스로 사용)
	ECX	카운트 레지스터 (Count Register)	32	반복적으로 실행되는 특정 명령에 사용 (루프의 반복 횟수나 좌우 방향 시프트 비트 수 기억)
	EDX	데이터 레지스터 (Data Register)	32	일반 자료 저장(입출력 동작에 사용)
세그먼트 레지스터 (Segment Register)	CS	코드 세그먼트 레지스터 (Code Segment Register)	16	실행될 기계 명령어가 저장된 메모리 주소 지정
	DS	데이터 세그먼트 레지스터 (Data Segment Register)	16	프로그램에서 정의된 데이터, 상수, 작업 영역의 메모리 주소 지정
	SS	스택 세그먼트 레지스터 (Stack Segment Register)	16	프로그램이 임시로 저장해야 하는 데이터나 사용자의 피호출 서브 루틴(called subroutine)이 사용할 데이터와 주소 저장
	ES, FS, GS	엑스트라 세그먼트 레지스터 (Extra Segment Register)	16	문자 연산과 추가 메모리 지정을 위해 사용되는 여분의 레지스터

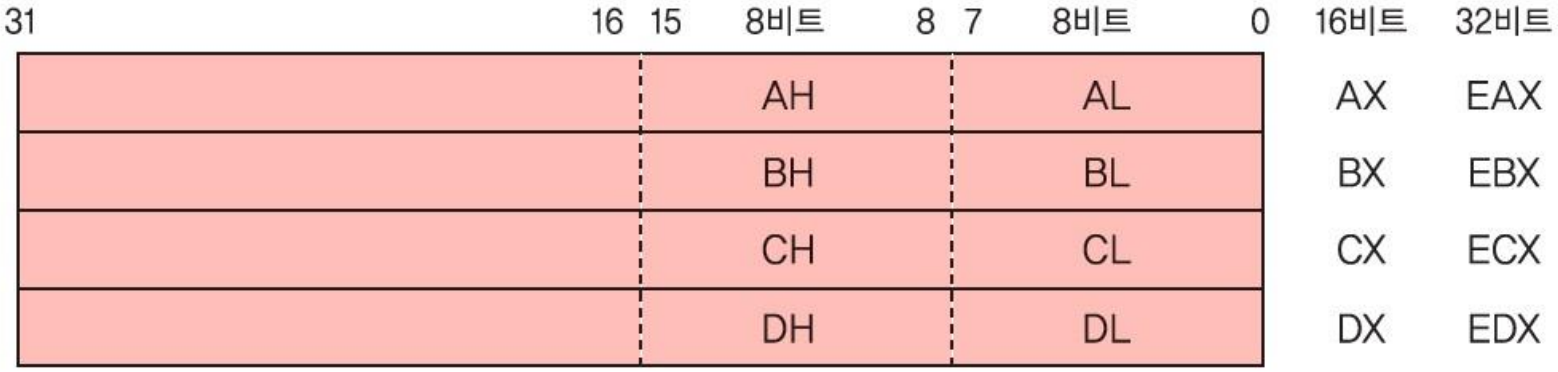


*** 레지스터**

처리 중인 **데이터나 처리 결과를 임시보관하는 CPU 내의 기억 장치**

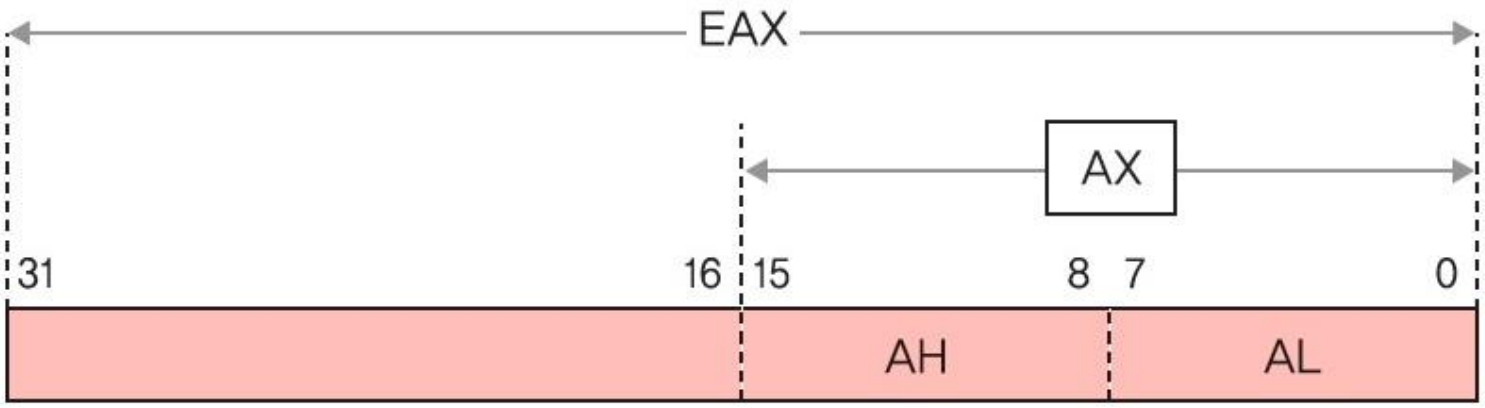
x86 시스템 CPU의 구조와 레지스터

- 범용 레지스터
  - 연산 장치가 수행한 계산 결과의 **임시 저장, 산술 및 논리 연산, 주소 색인 등의 목적으로 사용될 수 있는 레지스터 EAX, EBX, ECX, EDX등**.



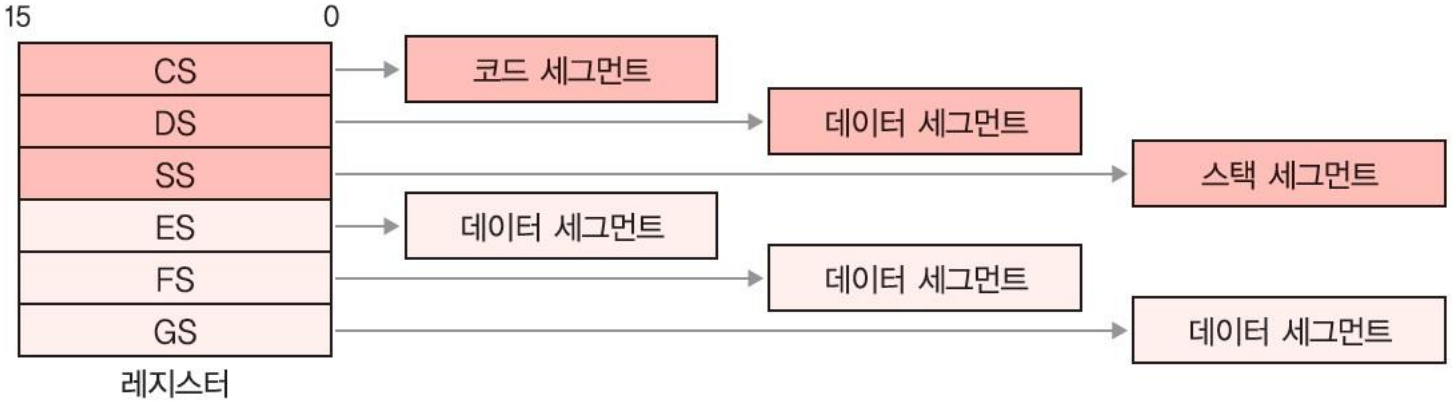
x86 시스템 CPU의 구조와 레지스터

- 범용 레지스터
  - **E**AX, **E**BX, **E**CX, **E**DX는 32비트 레지스터로 앞의 **E**는 '확장된(Extended)'을 의미. 이 레지스터의 오른쪽 16비트를 각각 AX, BX, CX, DX라 부르고, 이 부분은 다시 둘로 나누어짐
    - 예) AX는 왼쪽 8비트 상위(high) 부분을 AH, 오른쪽 8비트 하위(low) 부분을 AL이라 함



x86 시스템 CPU의 구조와 레지스터

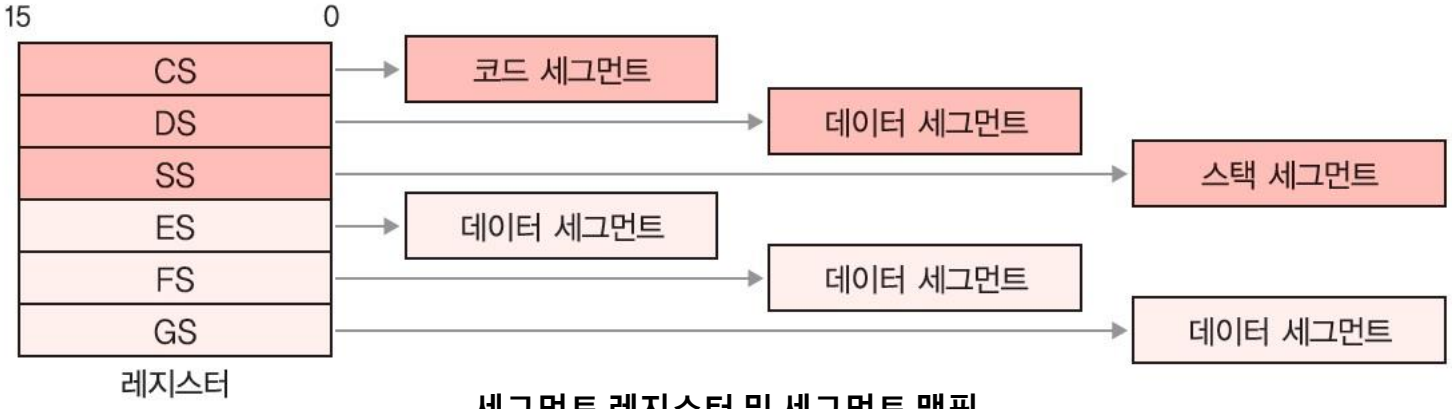
- 세그먼트 레지스터
  - 프로그램에 **정의된 메모리상의 특정 영역. 코드, 데이터, 스택 등을 포함**
  - CS, DS, SS 3개의 레지스터가 기본 레지스터이고, ES, FS, GS 레지스터는 여분의 레지스터임



세그먼트 레지스터 및 세그먼트 맵핑

x86 시스템 CPU의 구조와 레지스터

- 세그먼트 레지스터
  - ES, FS, GS 레지스터 : ES 레지스터는 추가로 사용된 데이터 세그먼트의 주소를 가리킴. FS와 GS 레지스터도 목적은 이와 비슷하지만, 두 레지스터는 거의 사용되지 않음.



세그먼트 레지스터 및 세그먼트 맵핑

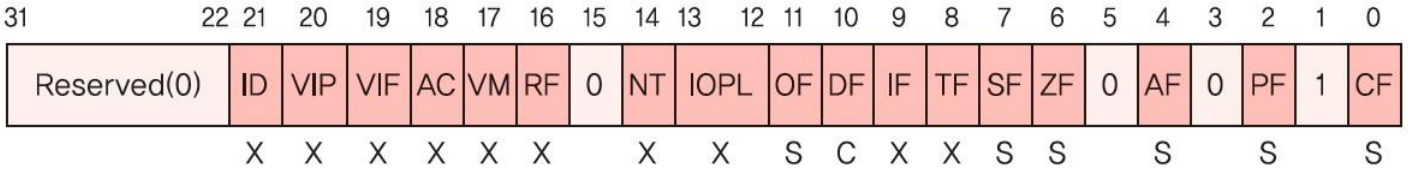
x86 시스템 CPU의 구조와 레지스터

레지스터의 종류와 용도

범주	80386 레지스터	이름	비트	용도
포인터 레지스터 (Pointer Register)	EBP	베이스 포인터 (Base Pointer)	32	SS 레지스터와 함께 사용되어 스택 내의 변수 값을 읽는 데 사용
	ESP	스택 포인터 (Stack Pointer)	32	SS 레지스터와 함께 사용되며 스택의 가장 끝 주소를 가리킴
	EIP	명령 포인터 (Instruction Pointer)	32	다음 명령어의 오프셋(Offset, 상대 위치 주소)을 저장하며 CS 레지스터와 합쳐져 다음에 수행될 명령의 주소 형성
인덱스 레지스터 (Index Register)	EDI	목적지 인덱스 (Destination Index)	32	목적지 주소에 대한 값 저장
	ESI	출발지 인덱스 (Source Index)	32	출발지 주소에 대한 값 저장
플래그 레지스터	EFLAGS	플래그 레지스터 (Flag Register)	32	연산 결과 및 시스템 상태와 관련된 여러 가지 플래그 값 저장

x86 시스템 CPU의 구조와 레지스터

- 플래그 레지스터
  - 32비트로, 컴퓨터 상태를 나타내는 비트 포함
  - 상태 플래그, 제어 플래그, 시스템 플래그로 구성



S: 상태 플래그 C: 제어 플래그 X: 시스템 플래그 * 음영 처리된 부분은 사용하지 않는 비트이다.

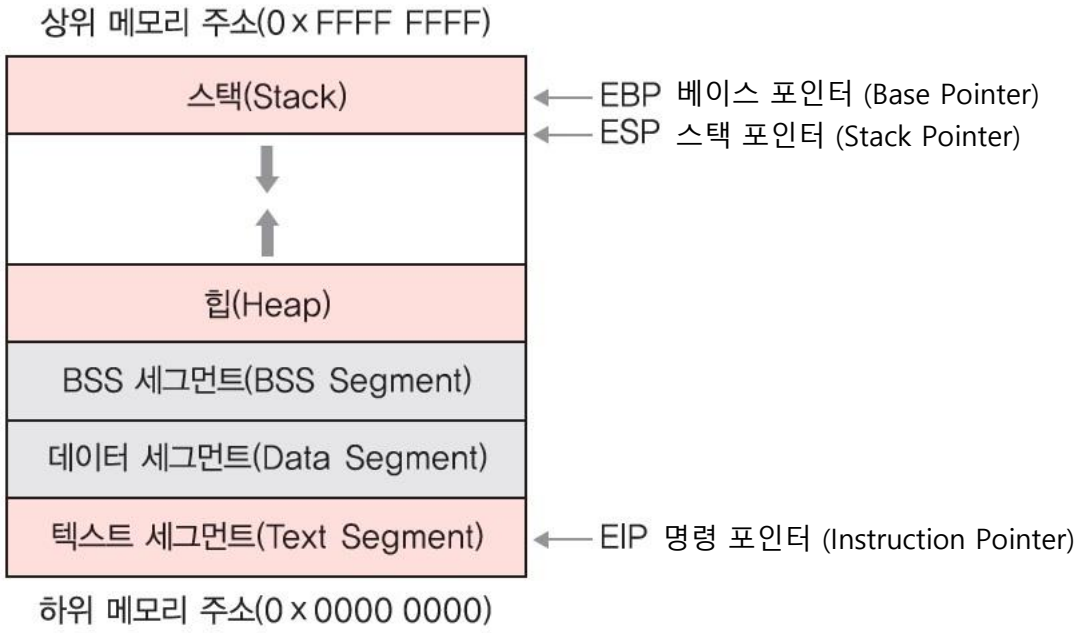
- CF: Carry Flag
- PF: Parity Flag
- AF: Adjust Flag
- ZF: Zero Flag
- SF: Sign Flag
- TF: Trap Flag
- IF: Interrupt enable Flag
- DF: Direction Flag
- OF: Overflow Flag
- IOPL: I/O Privilege Level
- NT: Nested Task flag
- RF: Resume Flag
- VM: Virtual 8086 Mode flag
- AC: Alignment Check
- VIF: Virtual Interrupt Flag
- VIP: Virtual Interrupt Pending
- ID: IDentification

외워야함
11학자 수
17:00

- 상태 플래그 : 산술 명령(ADD, SUB, MUL, DIV) 결과를 반영.
  - CF(Carry Flag, 비트 0) : 산술 연산 결과로 자리올림, 자리 내림 발생할 때 세트(1).
  - ZF(Zero Flag, 비트 6) : 산술 연산 결과 0이면 세트(1), 이외에는 클리어(0).
  - OF(Overflow Flag, 비트 11) : 부호가 있는 수의 오버플로가 발생하거나 MSB (MostSignificant Bit)가 변경되었을 때 세트
- 제어 플래그와 시스템 플래그는 Skip!

x86 시스템 CPU의 구조와 레지스터

메모리의 기본 구조



- 스택 : 후입선출(LIFO : Last-In, First Out) 방식에 의해 정보를 관리
  - 메모리상의 스택은 프로그램 함수와 관련된 인자, 로컬 변수 등을 관리하기 위해 사용됨
  - Top이라고 불리는 스택의 끝부분에서 데이터의 삽입과 삭제가 발생
  - 가장 나중에 삽입된 정보가 가장 먼저 읽힘

x86 시스템 CPU의 구조와 레지스터

- 메모리의 기본 구조
  - 힙: 연결 리스트, 트리, 그래프처럼 동적인 특성이 있는 데이터 구조에서 사용

```
char *p = new char[1000];
```

  - BSS 세그먼트 : 초기화 되지 않은 데이터 세그먼트(Uninitialized data segment)라고 불리며, 프로그램이 실행될 때 0이나 NULL 포인터로 초기화.
    - 외부 변수나 static 변수 중 초기화 되지 않은 변수들이 정의될 때 저장

```
static int a;
```

  - 데이터 세그먼트 : 초기화된 외부 변수나 static 변수 등이 저장되는 영역

```
static int a = 1;
```

  - 텍스트 세그먼트 : CPU로 실행되는 머신 코드가 있는 영역

어셈블리어의 기본 문법과 명령

- 어셈블리어의 구조
  - 어셈블리어에는 Intel 문법과 AT&T 문법이 있음
    - 윈도우에서는 Intel 문법 사용
    - 리눅스에서는 AT&T 문법 사용
  - Intel 문법에서는 목적지(destination)가 먼저 오고 원본(source)이 뒤에 오지만, AT&T에서는 반대

- Intel 문법에서 어셈블리어의 명령 형식은 다음과 같음
  - 1피연산자: 목적지, 2피연산자: 원본 (리눅스는 반대)

Label :	MOV	AX,	BX	; comment
라벨	작동 코드	제1피연산자	제2피연산자	설명

어셈블리어의 기본 문법과 명령

- 어셈블리어의 데이터 타입
  - 바이트(Byte) : 1바이트(8비트) 데이터 항목
  - 워드(Word) : 2바이트(16비트) 데이터 항목
  - 더블워드(Doubleword) : 4바이트(32bit) 데이터 항목

어셈블리어의 기본 문법과 명령

어셈블리어의 주소 지정 방식

레지스터 주소 지정

- 레지스터의 주소 값을 직접 지정 복사하는 방식
- 처리 속도 가장 빠름

```
MOV     DX,     BX
```

BX 레지스터 내용을 DX 레지스터로 복사

직접 메모리 주소 지정

- 가장 일반적인 주소 지정 방식
- 보통 피연산자 하나가 **메모리 위치**를 참조하고 다른 하나는 **레지스터**를 참조.

어셈블리어의 기본 문법과 명령

어셈블리어의 기본 명령

산술 연산 명령 (기본적인 정수 계산)

- ADD (Add)
  - 제1피연산자와 제2피연산자 값의 더한 값을 제1피연산자에 저장

형식	ADD	[제1피연산자]	[제2피연산자]
사용 예시	ADD	AL	4
	AL이 원래 3이었다면, 명령 실행 후 AL은 7이 된다.		

- SUB (Subtract)
  - 제1피연산자에서 제2피연산자를 뺀 값을 제1피연산자에 저장

형식	SUB	[제1피연산자]	[제2피연산자]
사용 예시	SUB	AL	4
	AL이 원래 7이었다면, 명령 실행 후 AL은 3이 된다.		

어셈블리어의 기본 문법과 명령

어셈블리어의 기본 명령

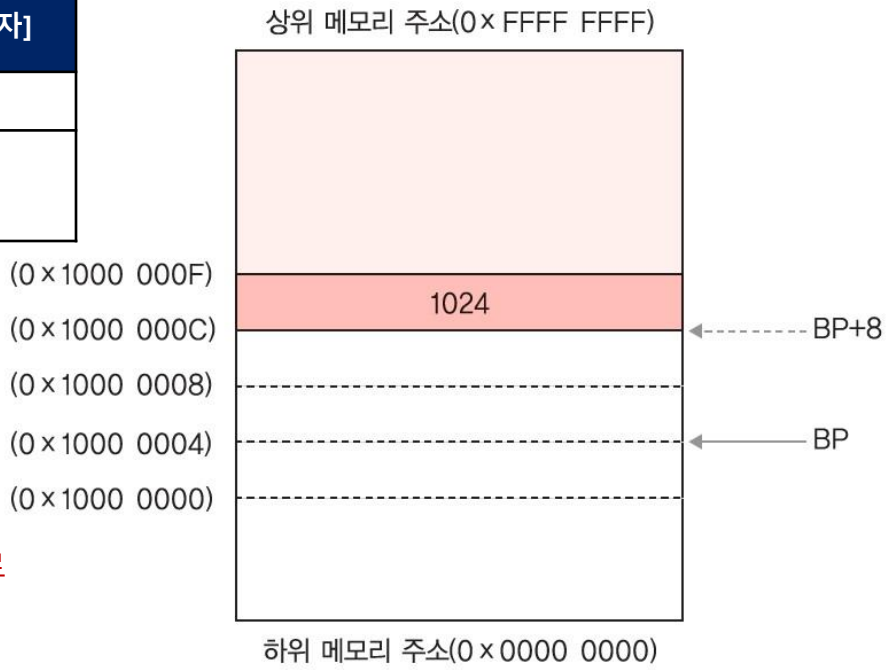
데이터 전송 명령

- 메모리, 범용 레지스터, 세그먼트 레지스터로 참조되는 주소에 존재하는 데이터 전송
- MOV (Move) : 데이터 이동할 때 사용

형식	MOV	[제1피연산자]	[제2피연산자]
사용 예	MOV	AX	[BP+8]
	BP의 주소에 8 더해진 주소에 있는 데이터 값을 AX에 대입한다.		

BP의 현재 값이 0x1000 0004라면,
BP+8은 0x1000 000C

0x1000 000C에 있는 값이 1024므로
AX에는 1024가 입력



MOV 실행 시 스택 동작

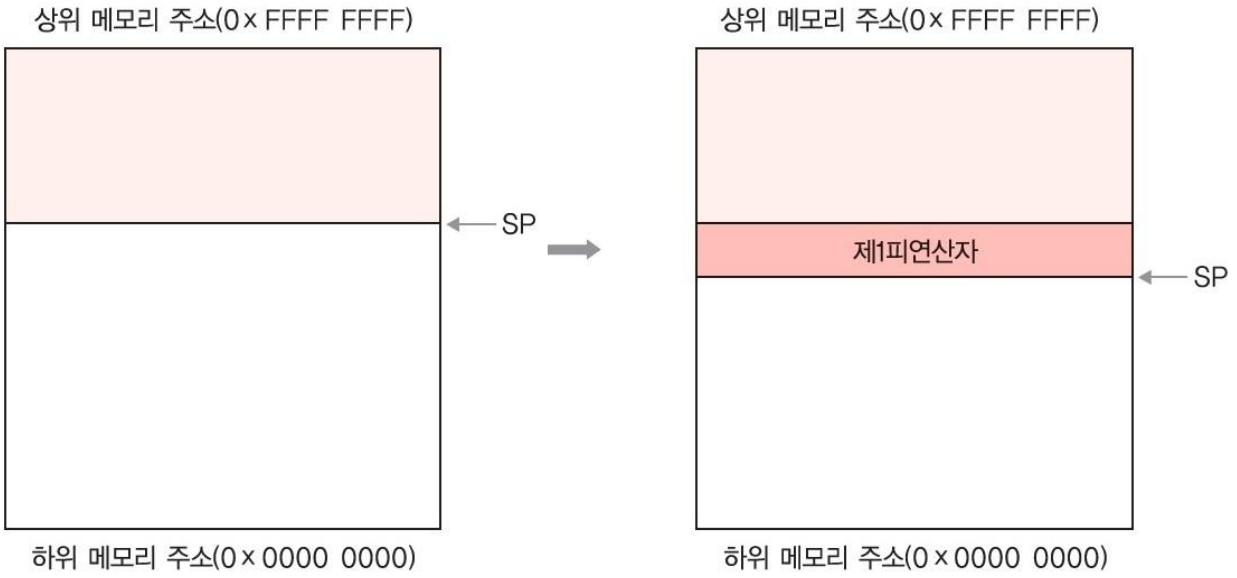
어셈블리어의 기본 문법과 명령

어셈블리어의 기본 명령

데이터 전송 명령

- PUSH (Push) : 스택에 데이터를 삽입할 때 사용.
  - 스택 포인터(Stack Pointer)는 데이터 크기만큼 감소

형식	PUSH	[제1피연산자]



PUSH 명령을 실행할 때 스택 동작

Q & A