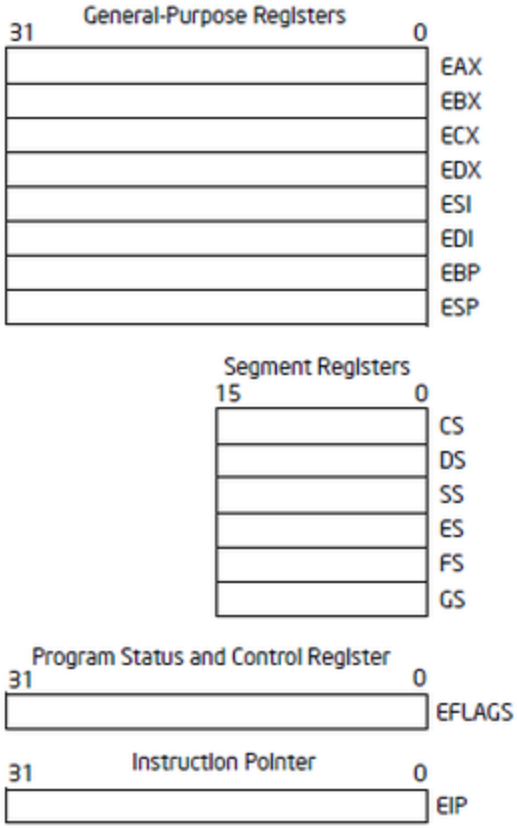
04 IA-32 Register

> *IA : Intel Architecture*

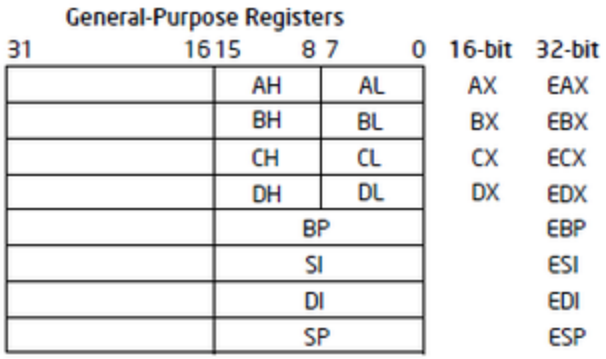
1. CPU Register

1) Register : CPU 내부에서 존재하는 다목적 저장 공간

2. Basic program execution registers



1) General Purpose Register (32bit - 8개)



- 범용 레지스터
  - 보통 상수 / 주소 등을 저장할때 주로 사용
- 피연산자와 포인터를 저장하여 사용하는 레지스터

(1) Registers 1 : 산술 연산 관련

> *Win32 API 함수들은 내부에서 ECX와 EDX를 사용합니다.*
> *따라서 이런 API가 호출되면 ECX와 EDX의 값은 변경됩니다.*
> *따라서 중요한 데이터가 있다면 API 호출 전 Backup이 필요합니다.*

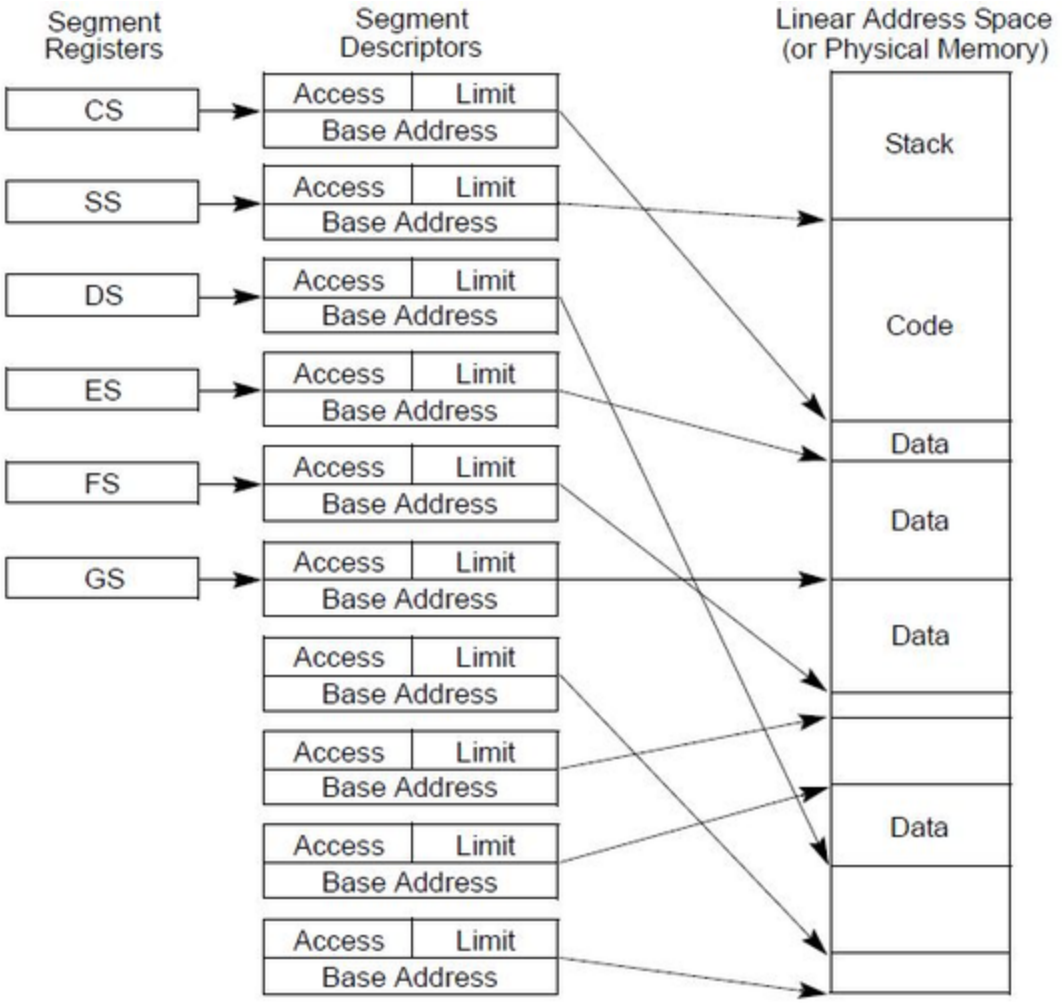
- 산술 연산명령에서 상수/변수 값의 저장 용도
- 어셈블리 명령(MUL, DIV, LODS 등) 들은 특정 레지스터를 직접 조작
  - EAX : Accumulator for operands and results data
    - 일반적으로 함수 리턴 값에 사용됨
      - 모든 Win32 API 함수들은 반환값을 EAX에 저장한 후 반환
  - EBX : Pointer to data in the DS segment
  - ECX : Counter for string and loop operaions
    - 반복 명령어(loop)에서 반복 카운트(loop count)로 사용됨
      (Loop 돌 때마다 ECX - 1)
  - EDX : I/O pointer

(2) Registers 2 : 메모리 주소 저장 관련

- 메모리 주소를 저장하는 포인터로 사용합니다.
  - EBP : Pointer to data on the stack (in the SS segment)
    - Stack frame 기법
      - 함수가 호출되었을 때 그 순간의 ESP를 저장하고 있다가,
        함수가 리턴하기 직전에 다시 ESP에 값을 반환시켜 스택이 깨지지 않도록 합니다.
  - ESI : Source pointer string operations
  - EDI : Destination pointer for string operations
    - ESI와 EDI는 특정 명령어(LODS, STOS, REP, MOVS 등)와 함께 메모리 복사에 사용

- ESP : Stack pointer (in the SS segment)
  - 스택 메모리 주소를 가리킨다.
  - 어떤 명령어들(PUSH, POP, CALL, RET)은 ESP를 직접 조작하기도 한다.

2) Segment Registers (16bit - 6개)



(1) 개념

IA-32 보호 모드에서 세그먼트란 메모리를 조각내어 각 조각마다 시작 주소, 범위, 접근 권한 등을 부여해서 메모를 보호하는 기법을 말한다.

세그먼트는 페이징(Paging) 기법과 함께 가상 메모리를 실제 물리 메모리로 변경할 때 사용됩니다.

세그먼트 메모리는 Segment Descriptor Table(SDT)에 기술되는데 세그먼트 레지스터는 바로 이 SDT의 Index를 가지고 있다.

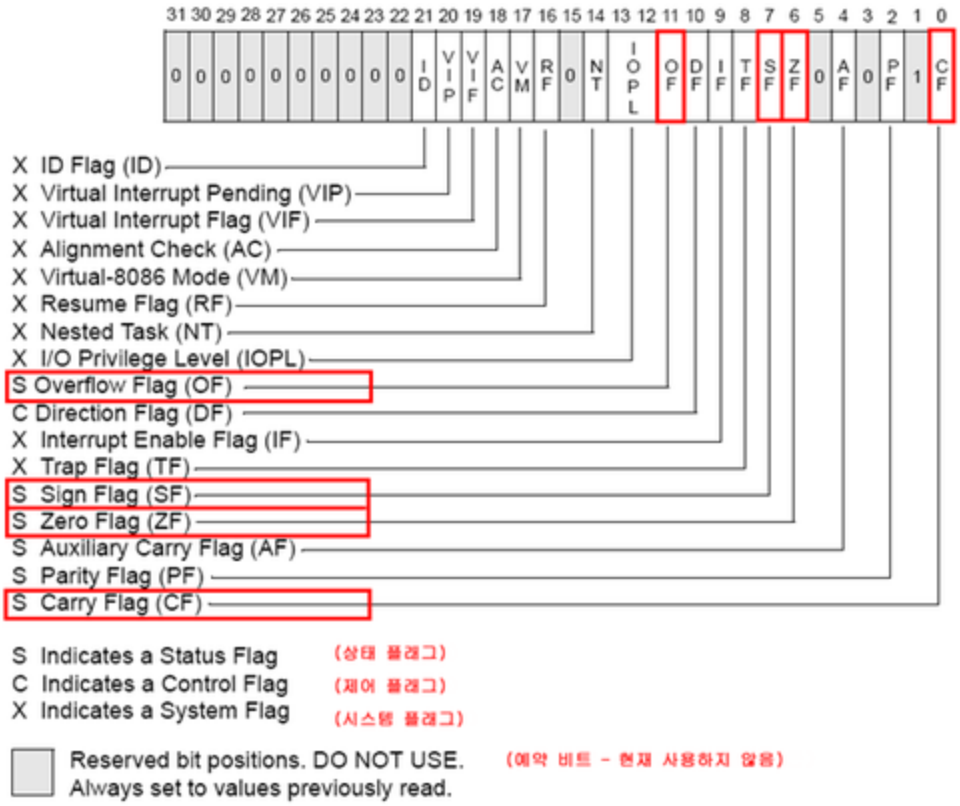
즉, 세그먼트 레지스터가 가리키는 세그먼트 디스크립터(Segment Descriptor)와 가상 메모리가 조합되어 선형주소(Linear Address)가 되며, 페이징 기법에 의해서 선형 주소가 최

종적으로 물리주소(Physical Address)로 변환됩니다.

- CS : Code Segment
- SS : Stack Segment
- DS : Data Segment
- ES : Extra(Data) Segment
- FS : Data segment
- GS : Data Segment

3) Program Status and Control Register (32bit - 1개)

> *EFLAGS : Flag Register*



(1) 개념

- 플래그 레지스터의 이름은 EFLAGS이며 32bit 크기
  (EFLAGS 레지스터 역시 16bit의 FLAGS 레지스터의 32bit 확장 형태이다.)
- 각 bit는 1 또는 0의 값을 가지는데, 이는 On/Off or True/False를 의미한다.

(2) Flag

- Zero Flag(ZF)
  - 연산 명령 후 결과 값이 0이 되면 ZF가 1로 세팅
- Overflow Flag(OF)

    - 부호 있는 수(signed integer)의 오버플로가 발생했을 때 1로 세팅
    - MSB가 변경되었을 때 1로 세팅
- Carry Flag(CF)
    - 부호 없는 수(unsigned integer)의 오버플로가 발생했을 때 1로 세팅
- Trap Flag(TF)
    - Trap과 비슷한 녀석으로 Exception이 있다. Exception은 예외가 발생했을 경우 예외처리 핸들러를 통해 예외처리를 실행하지만 Trap은 약간 다르다. 덫이라는 의미를 그대로 이해하면 쉬워지는데 Trap이 설정된 경우 무조건 Trap에 대한 핸들러를 실행하게 된다.

각 비트의 의미

비트	약어	이름	설명
0	CF	Carry Flag	연산시 자리올림이나 자리빌림이 있을 경우 set 된다
1			1
2	PF	Parity Flag	연산결과 1인 비트수가 짝수라면 set 된다
2			0
4	AF	Auxiliary Flag	CF 가 Nibble 에 대하여 적용된다
5			0
6	ZF	Zero Flag	연산결과가 0 이 되었을때 set 된다
7	SF	Sign Flag	연산결과 MSB 가 1일때 set 된다
8	TF	Trap Flag	하나의 명령어 처리후에 single-step interrupt 를 발생시킨다
9	IF	Interrupt Flag	Enables the external interrupt
10	DF	Direction Flag	Increment or decrement mode for
11	OF	Overflow Flag	연산결과 사인비트가 반전된다면 set 된다
12-13	IO-PL	I/O Privilege Level	Used in protected mode operation to select the privilege level for I/O device(00~11).

			If current privilege level > IOPL, I/O (IN, INS, OUT, OUTS, CLI, STI) can be executed. Otherwise, an exception occurs (286+ only)
14	NF	Nested Task	Instruction caused nested task switch (286+)
15			0
16	RF	Resume Flag	Debug faults disabled during instruction execution (386+)
17	VM	Virtual 8086	Currently executing 8086 code on virtual processor (386+)
18	AC	Alignment Check	Data aligned to four-byte boundary (486+)
19	VIF	Virutal interrupt flag	A copy of the interrupt flag bit available to the Pentium II processors
20	VIP	Virtual interrupt pending flag	Provides information about a virtual mode interrupt for the Pentium-Pentium II processors. Used in multitasking environment
21	ID	ID Flag	CPUID 명령이 지원되는지 나타내는 플래그

4) Instruction Pointer (32bit - 1개)

> *EIP : Instruction pointer*

(1) 개념

- CPU가 처리할 명령의 주소를 나타내는 레지스터
- 32bit (16bit IP 레지스터의 확장 형태)
- EIP에 저장된 메모리 주소의 명령어를 하나 처리하고 난 후에 자동으로 그 명령어 길이만큼 EIP를 증가시킵니다.

5) Tip

(1) 각 레지스터들은 16bit 하위 혼환을 위하여 몇 개의 구획으로 나뉜다.

- EAX 기준

- EAX : (0 ~ 31) 32bit
- AX : (0~15) EAX 하위 16bit
- AH : (8~15) EAX 상위 8bit
- AL : (0~7) AX의 하위 8bit