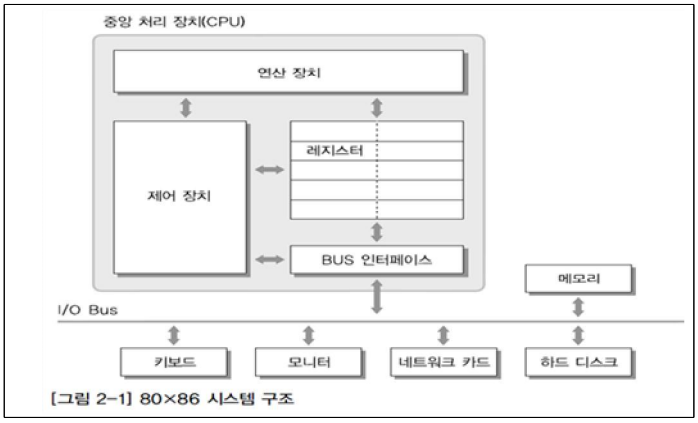
2주차 1차시 80x86 시스템 CPU, 레지스터, 메모리

【학습목표】

1. 80x86 시스템에 대해 알고, 80x86 시스템 CPU의 구조와 메모리에 대해 설명할 수 있다.

학습내용1 : 80x86 시스템 CPU의 구조

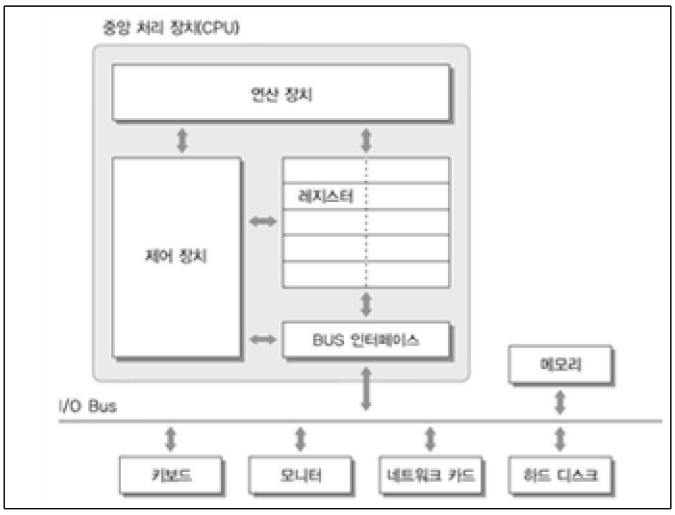
1. 80x86 시스템 CPU 구조



[그림 2-1] 80×86 시스템 구조

2. 연산 장치(ALU,Arithmetic and Logic Unit)

*CPU(중앙 처리 장치)의 핵심 부분 중 하나
*산술과 논리 연산을 수행하는 연산 회로 집합



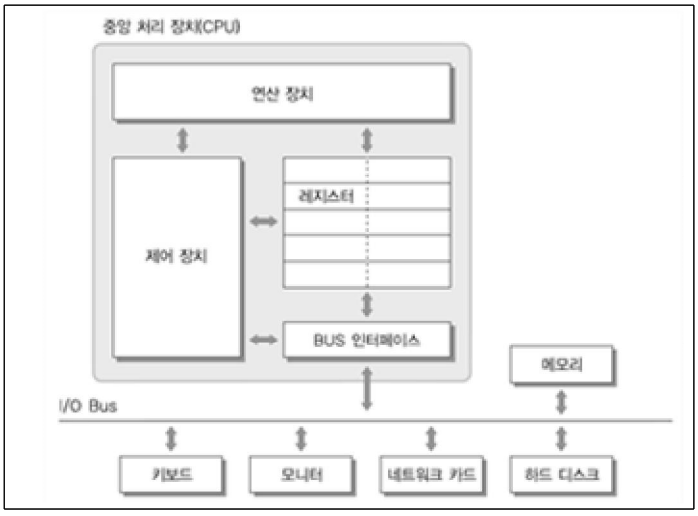
3. 연산 장치의 구성 요소

구성요소		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

구성요소		기능
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

4. 제어 장치(Control Unit)

* 입력, 출력, 기억, 연산 장치를 제어하고 감시
* 주기억 장치에 저장된 명령을 차례로 해독하여 연산 장치로 보내 처리되도록 지시



5. 제어 장치의 구성 요소

구성요소		기능
내부 장치	명령 해독기 (Instruction Decoder)	명령 레지스터에 있는 명령을 해독하여 부호기로 전송
	부호기(Decoder)	명령 해독기가 전송한 명령을 신호로 만들어 각 장치 전송
	주소 해독기 (Address Decoder)	명령 레지스터에 있는 주소를 해독하여 메모리의 실제주소로 변환한 후, 이를 데이터 레지스터에 저장

구성요소		기능
관련 레지스터	프로그램 카운터 (Program Counter)	다음에 실행할 명령의 주소 저장
	명령(Instruction) 레지스터	현재 실행 중인 명령 저장
	메모리 주소 (Memory Address) 레지스터	주기억 장치의 번지 저장
	메모리 버퍼 (Memory Buffer) 레지스터	메모리 주소 레지스터에 저장된 주소의 실제 내용 저장

학습내용2 : 레지스터의 종류

1. 레지스터(Register)

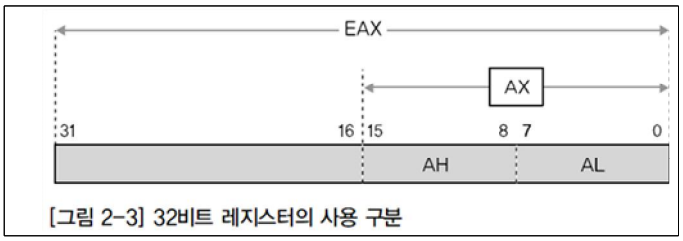
* 처리 중인 데이터나 처리 결과를 임시 보관하는 CPU 내의 기억 장치
* 레지스터의 종류

범용 레지스터(General Resistor)
세그먼트 레지스터(Segment Resistor)
포인터 레지스터(Pointer Resistor)
인덱스 레지스터(Index Resistor)
플래그 레지스터(Flag Resistor)

2. 범용 레지스터(General Resistor)

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

* 앞의 E는'확장된(Extended)'을 의미



[그림 2-3] 32비트 레지스터의 사용 구분

* EAX(Extended Accumulator Resistor)

누산기(Accumulator)
32비트
산술 연산과 입출력에 사용

* EBX(Extended Base Resistor)
베이스 레지스터(Base Resistor)
32비트
DS 세그먼트에 대한 포인터를 저장
ESI나 EDI와 결합하여 메모리의 주소 지정을확장하기 위한 인덱스에 사용되는 유일한 범용 레지스터

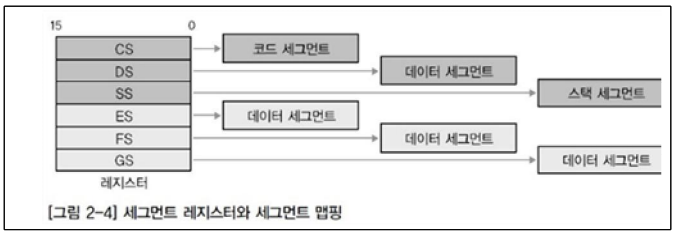
* ECX(Extended Count Resistor)
카운트 레지스터(Count Resistor)
32비트
루프가 반복되는 횟수를 제어하는 값

* EDX(Extended Data Resistor)
데이터 레지스터(Data Resistor)
32비트
입출력 연산에 사용
큰 수의 곱셈과 나눗셈 연산을 위해 EXA와 함께 사용

3. 세그먼트 레지스터(Segment Resistor)

* 프로그램에 정의된 메모리상의 특정 영역
* 코드, 데이터, 스택 등을 포함
* 기본 세그먼트 레지스터
CS, DS, SS

* 여분 세그먼트 레지스터(Extra Segment Resistor) ES, FS, GS



[그림 2-4] 세그먼트 레지스터와 세그먼트 맵핑

* CS(Code Segment Resistor)
실행될 기계 명령을 포함
코드 세그먼트의 시작 주소를 나타냄

* DS(Data Segment Resistor)
프로그램에 정의된 데이터, 상수, 작업 영역을 포함
데이터 세그먼트의 시작주소를 나타냄
데이터의 오프셋을 DS 레지스터에 저장된 주소 값에더해 데이터 세그먼트내에 위치해 있는데이터의 주소를 참조

* SS(Stack Segment Resistor)
실행 과정에서 필요한 데이터나 연산 결과 등을임시로 저장하거나 삭제할 때 사용
스택 세그먼트의 시작 주소를 나타냄

* ES
추가로 사용된 데이터 세그먼트의 주소를 나타냄
16비트

* FS, GS
ES와 목적은 이와 비슷하지만 거의 사용되지 않음
16비트

4. 포인터 레지스터(Pointer Resistor)

* 프로그램 실행 과정에서 사용되는 주요 메모리 주소 값 저장
* EBP(Extended Base Pointer Resistor)
스택 세그먼트에서 호출되어 사용되는 함수의시작 주소 값 저장
ESP 레지스터와 함께 스택 프레임 형성
실제 메모리 주소 참고할 경우 SS 레지스터와 함께 사용
32비트

* ESP(Extended Stack Pointer Resistor)
현재 스택의 가장 하위 주소 저장
실제 메모리 주소 참고할 경우 SS 레지스터와 함께 사용

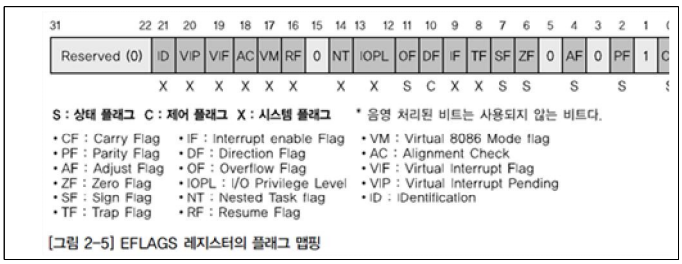
* EIP(Extended Instruction Pointer)
현재 실행 중인 코드 세그먼트에 속한 현재 명령을 나타냄
메모리상의 주소 참조를 위해 CS와 함께 사용

5. 인덱스 레지스터(Index Resistor)

*데이터를 복사할 때 출발지와 목적지 주소를 나타냄
*ESI(Extended Source Index Resistor)
출발지 주소에 대한 값 저장
32비트
*EDI(Extended Destination Index Resistor)
목적지 주소에 대한 값 저장
32비트

6. 플래그 레지스터(Flag Resistor)

* 32비트 크기로 연산 결과 및 시스템 상태와 관련된 여러 가지 플래그 값 저장



[그림 2-5] EFLAGS 레지스터의 플래그 맵핑

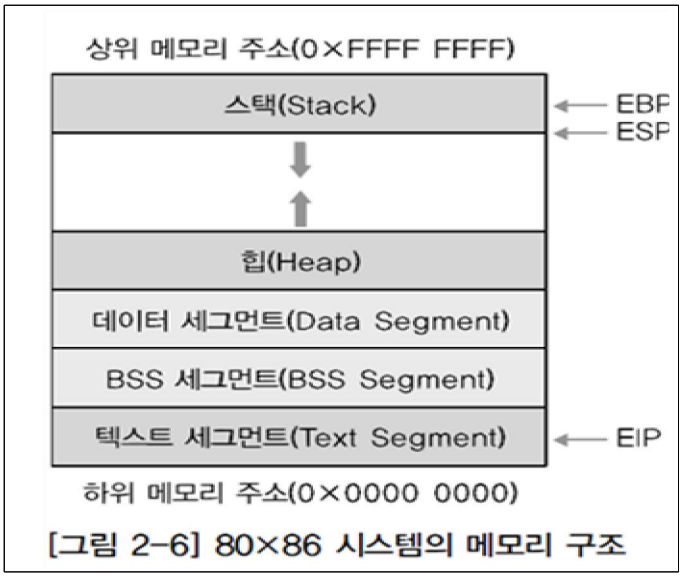
* 상태 플래그(State Flag)
산술 명령(ADD, SUB, MUL, DIV) 결과를 반영
CF(Carry Flag, 비트 0)> 산술 연산 결과로 자리올림, 자리내림 발생할 때 세트
ZF(Zero Flag, 비트 6)> 산술 연산 결과 0이면 세트, 이외에는 클리어
OF(Overflow Flag, 비트 11)> 부호가 있는 수의 오버플로우가 발생하면 세트> MSB(Most Significant Bit)가 변경되었을 때 세트
PF(Parity Flag, 비트 2) : 산술 연산의 결과가 짝수면 세트
AF(Adjust Flag, 비트 4) : 8비트 피연산자를 사용한 산술연산에서 비트 3에서 비트 4로 자리올림하면 세트
SF(Sign Flag, 비트 7) : 산술 및 논리 연산의 결과로 음수가 생기면 세트

* 제어 플래그(Control Flag)
DF(Direction Flag, 비트 10)
스트링 명령(MOVS, CMPS, SCAS, LODS, STOS)을 제어
DF가 1이면 스트링 명령 자동 감소
DF가 0이면 스트링 명령 자동 증가
STD 명령은 DF 플래그를 세트
CLD 명령은 DF 플래그를 클리어

* 시스템 플래그(System Flag)
TF(Trap Flag, 비트 8)> 디버깅 시 Single Step Mode 모드를 활성화하면 세트
IF(Interrupt enable Flag, 비트 9)> 프로세서의 인터럽트 처리 여부를 제어
IOPL(I/O Privilege Level, 비트 12/13)> 현재 실행되는 프로그램이나 태스크의 입출력 특권 레벨 지시
NT(Nested Task flag, 비트 14)> 인터럽트되거나 호출된 태스크를 제어
RF(Resume Flag, 비트 16)> 프로세서의 디버그 예외 반응을 제어
VM(Virtual 8086 Mode flag, 비트 17)> V86 모드를 활성화하면 세트
AC(Alignment Check, 비트 18)> 메모리 참조 시 정렬 기능을 활성화하면 세트
VIF(Virtual Interrupt Flag, 비트 19), VIP(Virtual Interrupt Pending, 비트 20)> 가상 모드 확장과 관련해 사용
ID(IDentification, 비트 21)> CPUID 명령의 지원 유무를 결정

학습내용3 : 80x86 시스템의 메모리

1. 메모리의 기본 구조



[그림 2-6] 80×86 시스템의 메모리 구조

2. 스택(Stack)

* 후입선출(LIFO : Last-In, First Out) 방식에 의해 정보를 관리하는 데이터 구조
* 가장 나중에 삽입된 정보가 가장 먼저 읽힘
* Top 이라고 불리는 스택의 끝부분에서 데이터의 삽입과 삭제가 발생

3. 힙(Heap)

* 프로그램의 실행 중 필요한 기억 장소를 할당하기 위해 운영체제에 예약되어 있는 기억 장소영역
* 힙에 대한 기억 장소는 포인터를 통해 동적으로 할당되거나 반환

4. 데이터 세그먼트(Data Segment)

* 초기화된 외부 변수나 static 변수 등이 저장되는 영역
* 보통 텍스트 세그먼트(Text segment)와 데이터 세그먼트 영역을 합쳐 프로그램이라 지칭

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

5. BSS 세그먼트(BSS Segment)

* 초기화 되지 않은 데이터 세그먼트
(Uninitalized data segment)
* 프로그램이 실행될 때 0이나 NULL 포인터로 초기화
* 외부 변수나 static 변수 중 초기화 되지 않은 변수들이 정의될 때 저장

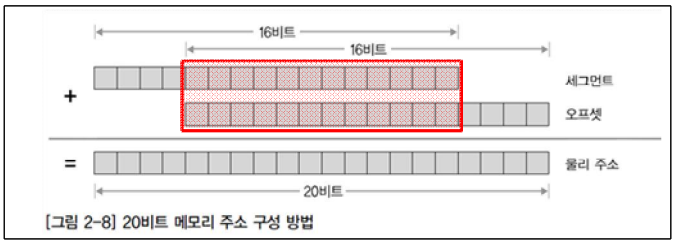
```
 static int a ;
```

6. 텍스트 세그먼트(Text Segment)

* CPU에 의해 실행되는 머신 코드가 있는 영역

7. 메모리 접근 모드와 동작

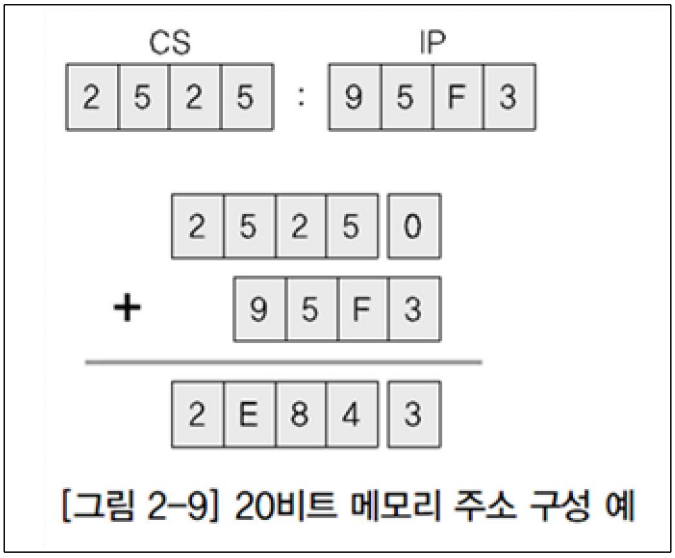
* 실제 모드
8086 CPU에서 사용되던 동작 모드
20비트 주소 버스 사용 위해 16비트 레지스터 사용
1MB(220 = 1,048,567)의 메모리 사용 가능
20비트 주소를 나타내기 위해 세그먼트 레지스터를도입



[그림 2-8] 20비트 메모리 주소 구성 방법

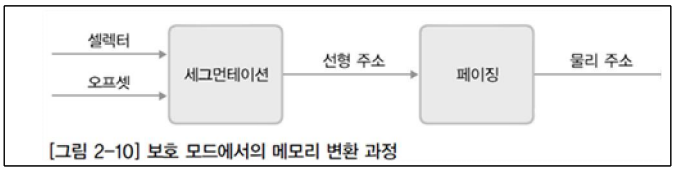
* 세그먼트 주소인 CS 레지스터가 0x2525h,
오프셋인 IP가 0x95F3h면 0x2525h 뒤에
한 자리의 0x0h를 붙여 95F3h를 더한 2E843h가 실제 가리키는 물리 주소가 됨

* 2525h:95F3h, 또는 [CS]:96F3h로 표현



[그림 2-9] 20비트 메모리 주소 구성 예

* 보호 모드
80286부터 도입하여 32비트 CPU 80386에 완성
32비트 주소 버스를 통해 4GB의 메모리를 사용 가능
메모리 보호 기능과 페이징(Paging) 등을 통해 가상 메모리를 효율적으로 구현



[그림 2-10] 보호 모드에서의 메모리 변환 과정

【학습정리】

1. 80x86 시스템 CPU는 연산장치, 제어 장치, 레지스터의 구성 요소로 이루어져 있다.

2. 레지스터는 CPU 내부에서 데이터를 일시적으로 저장하는 장소로서 범용 레지스터, 세그먼트 레지스터, 포인터 레지스터, 인덱스 레지스터, 플래그 레지스터로 분류된다.