어셈블리 언어 기초

10111 유동주

목차

1. 어셈블리 언어란

- 어셈블리 언어는 기계어와 1:1 대응을 하는 언어입니다

- 어셈블리 언어는 그 코드가 어떤 일을 할지를 추상적이 아닌, 직접적으로 보여주고

논리상의 오류나, 수행 속도, 수행 과정에 대해 명확히 해준다는 점에서 직관적인 언어입니다

2. 어셈블리를 위한 기본 지식

1) CPU

메모리에 있는 내용을 읽고, 쓰고 데이터를 메모리와 레지터로 보낸다

프로그램의 명령을 해석하고 실행한다.

하나의 프로세서는 12~14개의 레지스터를 가지고 있으며, CPU의 연산, 논리 장치는 숫자와 기호에 관한 연산자를 인식한다

2) RAM

반도체를 조립된 셀들의 집합. 프로세스가 프로그램을 실행시키고 작동하기위해서 필요한 정보들을 저장하는데 쓰인다

각각의 셀들은 숫자값을 포함하고 주소가 정해질 수 있는 형식이며 프로그램에서 흔히 메모리라고 하는 것은 메인메모리, 즉 , 램이라고 할 수 있다

(2) 80x86 프로세서

1) CPU 레지스터 종류 : 범용 레지스터, 상태 레지스터, 플래그 레지스터

- 레지스터 : CPU내부의 기억장소로 PC가 정보를 처리하기 위해서는 정보가 특정한 셀에 저장되어 있어야 한다

< 레지스터의 구조 >

1. 데이터 레지스터
   - 데이터 레지스터는 각종 데이터 처리를 대상으로 하는 32비트 레지스터 및 16비트 레지스터 일부를 프로그래머가 명령 중에서 자유롭게 지정을 할 수 있는 범용 레지스터이다
   : EAX, EBX, ECX, EDX

2. 포인터 레지스터
   : ESP, EBP

3. 인덱스 레지스터 (Index register)
   : ESI, EDI

4. 세그먼트 레지스터(segment register)
   : CS, DS, SS, ES

범용 레지스터

작은 데이터의 임시 저장 공간으로, 연산 처리 및 데이터의 주소를 지정하는 역할을 합니다.

General-Purpose Registers

31 16	15 8	7 0	16-bit	32-bit
	AH	AL	AX	EAX
	BH	BL	BX	EBX
	CH	CL	CX	ECX
	DH	DL	DX	EDX
	BP			EBP
	SI			ESI
	DI			EDI
	SP			ESP

세그먼트 레지스터

ES : 보조 세그먼트 제스터이다. 두 곳 이상의 데이터 저장영역을 가리켜야 할 때 DS와 함께 사용된다.
하지만 32bit 프로그램에서는 DS 와 ES가 같은 영역을 가리키고 있기 때문에 굳이 신경 쓰지 않아도 된다

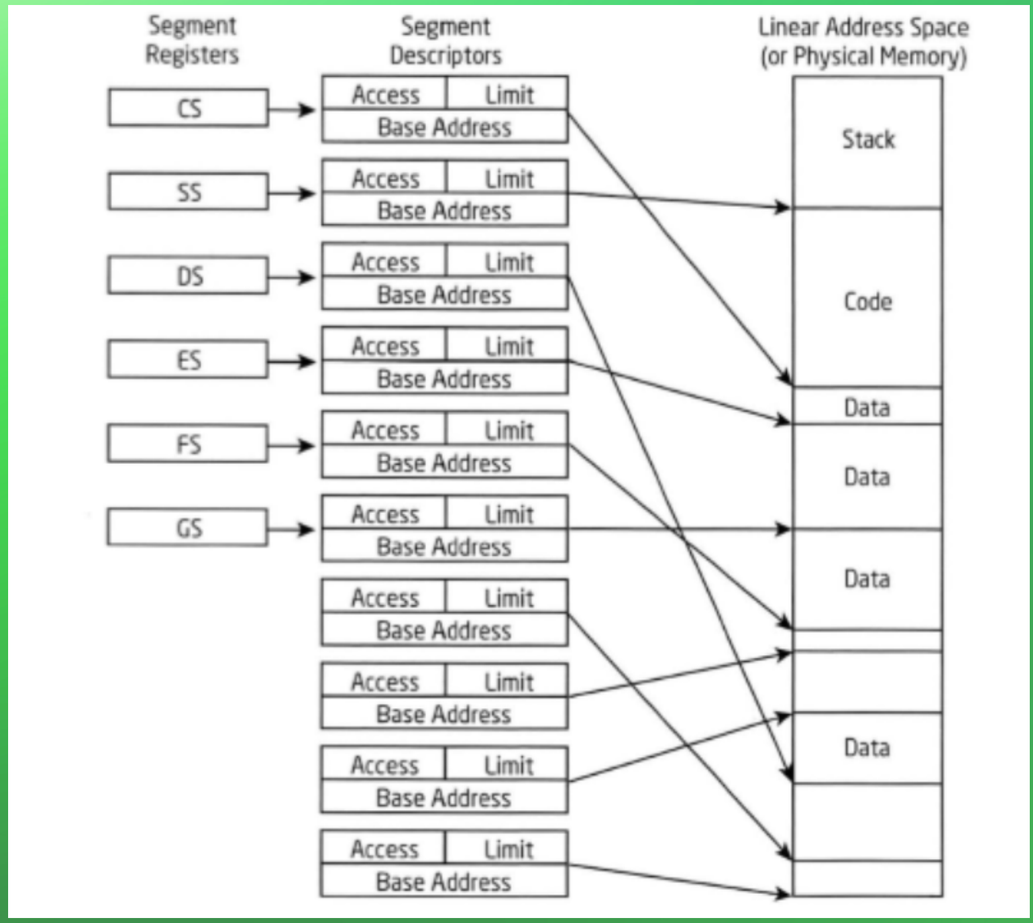
CS : 코드 세그먼트를 가리키는 레지스터. 프로그래머 코드의 시작조소를 가지고 있습니다

SS : 스택 세그먼트를 가리키는 레지스터. 스택의 시작 주소를 담고 있다.

DS : 데이터 세그먼트를 가리키는 레지스터.

FS

GS: 보조 세그먼트 레지스터 FS, GS는 286 이후에 추가된 것으로 운영 체제를 작성하는게 아니라면 없듯이 여겨도 됩니다



3.어셈블리 명령어의 구성

- 인텔 CPU 안에는 이 명령어들이 회로로 구현되어 있어서 어셈블리 코드를 실행할 수 있고 CPU는 2진수로 모든 것을 처리하는데 어셈블리 명령어들도 2진수로 되어 있다

4. 주소 지정 방식의 이해

1. 즉시 지정방식(immediate addressing)
   - mov $0x1, %eax : eax에 (16진수)1을 값을 넣는(할당) 방식입니다.

2. 레지스터 지정방식(register addreessing)
   - 속도는 빠르지만 레지스터의 크기(32비트)로 인해 크기가 제한된다.

3. 직접 주소 지정방식(directly addressing)
   - 가장 일반적인 주소지정방식이며, 메모리의 주소를 직접 지정해서 바로 찾아오는 방식이다.

4. 레지스터 간접 주소 지정 방식
   : ebx의 값을 주소로 하여(간접적으로) eax레지스터에 할당
   '()'가 들어간다면 간접 지정이라고 볼 수 있다. '()'의 의미는 괄호 안에 들어간 값의 주소입니다

5. 베이스 상대 주소 지정 방식
   - mov 0x4(%esi), %eax, %eax
   : esi레지스터에서 4(byte)를 더한 주소의 값을 eax레지스터에 할당합니다

명령어	예제	설명	분류
push	push %eax	eax의 값을 스택에 저장.	스택 조작
pop	pop %eax	스택 가장 상위에 있는 값을 꺼내서 eax에 저장	스택 조작
mov	mov %eax, %ebx	메모리나 레지스터의 값을 옮길때 사용	데이터 이동
lea	leal(%esi), %ecx	%esi의 주소값을 %ecx에 옮긴다.	주소 이동
inc	inc %eax	%eax의 값을 1 증가시킨다.	데이터 조작
dec	dec %eax	%eax의 값을 1 감소시킨다.	데이터 조작
add	add %eax, %ebx	레지스터나 메모리의 값을 덧셈할 때 쓰인다.	논리, 연산
sub	sub $0x8, %esp	레지스터나 메모리의 값을 뺄셈할 때 쓰인다.	논리, 연산
call	call proc	프로시져를 호출한다.	프로시져
ret	ret	호출했던 바로 다음 지점으로 이동.	프로시져
cmp	cmp %eax, %ebx	레지스터와 레지스터값을 비교	비교
jmp	jmp proc	특정한 곳으로 분기	분기
int	int $0x80	OS에 할당된 인터럽트 영역을 system call	인터럽트
nop	nop	아무 동작도 하지 않는다.(No Operation)	

Q&A

감사합니다.