2주차 2차시. 어셈블리어 기본 문법

【학습목표】

1. 어셈블리어 데이터 타입과 저장방식,기본 명령에 대해 설명할 수 있다.

학습내용1 : 어셈블리어의 개요

1. 어셈블러(Assembler)

저급언어로 쓰여진 프로그램을 읽어서 그와 동일한 기능을 하는 기계언어 프로그램으로 변환하는 기능 수행

2. 저급언어를 사용하여 프로그래밍할 때 프로그래머가 알아야 할 것

* 명령문과 사용규칙
* 프로그램이 실행될 기계의 기본 구조
메모리에 자료가 저장되는 방식
프로세서가 자료를 처리하는 방식 등

3. 어셈블리어

* 기계어와 일대일 대응이 되는 컴퓨터 프로그램의 저급언어

4. 고급언어의 특징

① 개념적인 언어
② 대상 컴퓨터의 세부적인 사항을 숨기고 있음
③ 프로그램이 읽기 쉽고 수정하기 편함

5. 어셈블리 언어를 사용해야 하는 경우

① 프로세서를 직접 제어하려고 할 때
② 속도 빠른 코드를 생성하려고 할 때
③ 프로그램의 크기를 최소화하려고 할 때

6. 어셈블리어 프로그램 개발 단계

7. 진수 체계

* 2진수
2진법(2진 위치 기수법)을 사용한 진수 체계
디지털 시스템에서 사용되는 0과 1을 이용
소자 특성에 편리하고 논리의 조립이 간단하기 때문에 컴퓨터 내부 디지털 신호는 2진법을 사용

* 2진수 - 10진수와 2진수의 대응관계

10진수	2진수
$0 = 0\times 2^0$	0
$1 = 1\times 2^0$	1
$2 = 1\times 2^1 + 0\times 2^0$	10
$3 = 1\times 2^1 + 1\times 2^0$	11
$4 = 1\times 2^2 + 0\times 2^1 + 0\times 2^0$	100
$5 = 1\times 2^2 + 0\times 2^1 + 1\times 2^0$	101
$6 = 1\times 2^2 + 1\times 2^1 + 0\times 2^0$	110
$7 = 1\times 2^2 + 1\times 2^1 + 1\times 2^0$	111
$8 = 1\times 2^3 + 0\times 2^2 + 0\times 2^1 + 0\times 2^0$	1000
$9 = 1\times 2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0$	1001

* 8진수
8진법(8진 위치 기수법)을 사용한 진수 체계
0, 1, 2, 3, 4, 5, 6, 7을 사용
오래된 컴퓨터 시스템에서 데이터를 표현하기 위한 단위
8진법 한 자리는 2진법의 세 자리와 일대일 대응

* 16진수
16진법(16진 위치 기수법)을 사용한 진수 체계
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F을 사용
컴퓨터 시스템에서 데이터를 표현하기 위한 단위
16진법 한 자리는 2진법의 네 자리와 일대일 대응

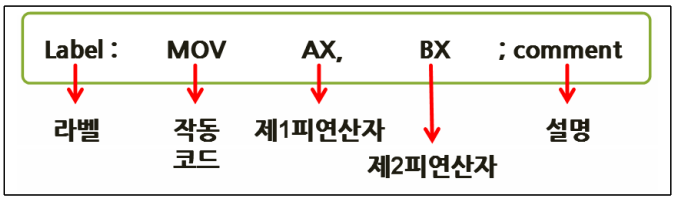
10진수	2진수	8진수	16진수	10진수	2진수	8진수	16진수
0	0000	0	0	8	1000	10	8
1	0001	1	1	9	1001	11	9
2	0010	2	2	10	1010	12	A
3	0011	3	3	11	1011	13	B
4	0100	4	4	12	1100	14	C
5	0101	5	5	13	1101	15	D
6	0110	6	6	14	1110	16	E
7	0111	7	7	15	1111	17	F

학습내용2 : 어셈블리어의 구조

1. Intel 문법

① 윈도우에서 사용

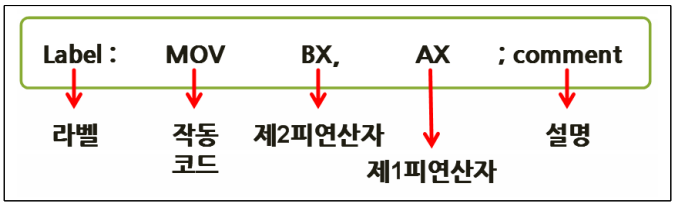
② 목적지 (Destination)가 먼저 오고 원본(Source)이 뒤에 위치



2. AT&T 문법

① 리눅스에서 사용

② 원본(Source)이 먼저 오고 목적지 (Destination)가 뒤에 위치



3. MOV

① 제2피연산자의 값을 제1피연산자로 복사하는 명령

② 어셈블리어 작동 코드 또는 명령 코드라고 불림

4. Comment

① 기계어 코드로 번역되지 않음

② 해당 명령에 대한 사용자 주석

5. Label

① 기계어 코드로 번역되지 않음
② 점프(분기) 명령 등에서 참조될 때 메모리 주소 계산에 사용
Label_1 : MOV AX, BX
.
.
.
JMP Label_1 ; Label_1로 무조건 이동하라는 명령

6. Label 명명법

① 31문자까지 가능
② Label 명명 후 콜론(:) 으로 종료
③ 알파벳, 숫자, 특수 문자(? @ _ $) 사용 가능
④ 첫 문자를 숫자로 사용할 수 없음
⑤ 레지스터의 이름은 사용하지 않음

* 적합한 예
HELLO $MARKET A12345 LONG_NAME PART_3
* 적합하지 않은 예
LONG-NAME 3_PART

7. 어셈블리어에 사용되는 연산자 특징

* 명령어가 작용하는 레지스터나 기억 장소의 위치
* 연산항을 갖지 않는 명령어
CLD > 캐리 플래그를 클리어하는 명령
* 연산항이 한 개인 명령어
DEC CX > 피연산자 CX 내용을 하나 감소시킴
* 연산항이 두 개인 명령어
MOV AX, BX > BX의 내용을 AX로 이동시킴

8. 숫자 정의 규칙

* 숫자는 10진수, 16진수와 2진수로 표현될 수 있음
1) 10진수
0~9를 사용
예) AX 레지스터에 10진수인 855를 로드
MOV AX, 855

2) 16진수
0과 9 사이의 숫자와 A~F(대문자 또는 소문자) 사용
16진수라는 것을 나타내기 위해 “H”를 숫자 뒤에 붙임
예) AX 16진수인 855H(10진수로 2133)를 로드
MOV AX, 855H
16진수가 A~F 중의 한 문자로 시작될 때 0을 앞에 첨가
예) AX 16진수인 FFH(10진수로 255)를 로드
MOV AX, 0FFH

3) 2진수
0과 1만 사용하고, 숫자 뒤에 “B”를 붙임
예) AX에 2진수인 011010010110B(10진수로 1686, 16진수로 696H) 를 로드
MOV AX, 011010010110B

학습내용3 : 어셈블리어의 데이터 타입과 리틀 엔디언 방식

1. 데이터 타입

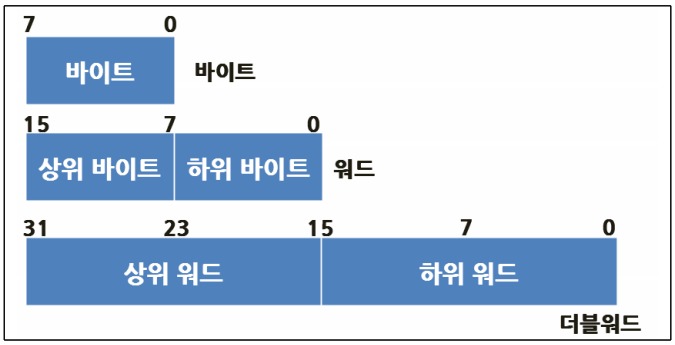
비트(Bit)
- 데이터 표현의 최소단위
- 한 개의 2진수
- 0 또는 1

니블(Nibble)
- 4비트(하프바이트, Half-byte)
- 단순한 디지털 장치의 데이터를 처리
- 예) 1101

바이트(Byte)
- 8비트
- 예) 1100 0111

워드(Word)
- 16비트(일반적인 정의)
- 더블워드(Double-word) : 32비트

* 어셈블리어의 데이터 타입



2. 리틀 엔디언 방식

2개의 번지로 나누어 저장해야 하는 16비트 데이터(워드)의 경우 하위 바이트는 하위 번지에 상위 바이트는 상위 번지에 저장

* 어셈블리어의 데이터 타입별 메모리 저장 위치

주소	1500번지	1501번지
저장된 데이터	F3	34

학습내용4 : 어셈블리어의 주소 지정 방식

1. 주소 지정 방식

* 직접 주소 지정 방식
레지스터 주소 지정
직접 메모리 주소 지정
* 간접 주소 지정 방식
레지스터 간접 주소 지정
인덱스 주소 지정
베이스 인덱스 주소 지정
변위를 갖는 베이스 인덱스 주소 지정

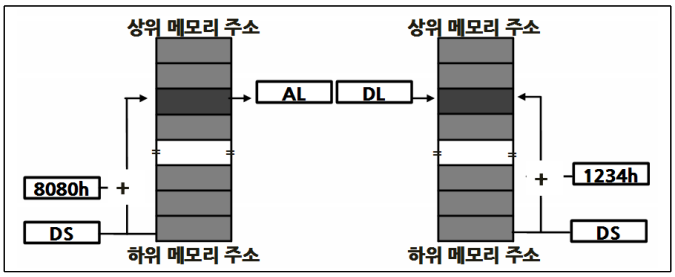
2. 레지스터 주소 지정

레지스터의 주소 값을 직접 지정 복사, 처리 속도 가장 빠름
예시 : BX 레지스터의 내용을 DX 레지스터로 복사
MOV DX, BX

3. 직접 메모리 주소 지정

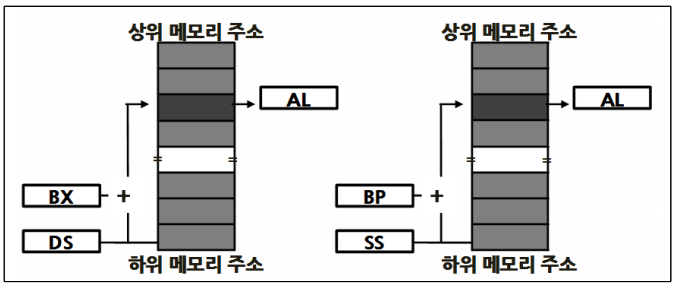
가장 일반적인 주소 지정 방식
피연산자 하나가 메모리 위치를 참조하고 다른 하나는 레지스터를 참조
세그먼트:오프셋 형식 사용
두 연산자가 메모리를 직접 참조하는 것은 MOVS와 CMPS만 가능

* 직접 메모리 주소 지정의 예
 MOV AL, DS:[8088h]
 MOV DS:[1234h], DL



4. 레지스터 간접 주소 지정

* 직접 메모리 주소 지정 방식처럼 세그먼트: 오프셋 형식 사용
* 세그먼트는 명시적으로 적지 않고 주소에 레지스터만 표시함
레지스터 간접 주소 지정의 예
 MOV AL, [BX]
 MOV AL, [BP]

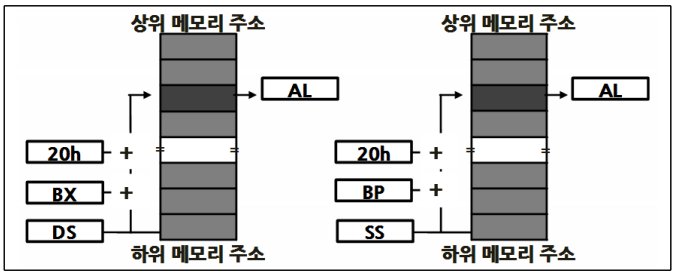


* 다음과 같이 기본이 아닌 세그먼트를 강제로 지정 할 수도 있음
 MOV AL, CS:[BX]
 MOV AL, DS:[BP]

5. 인덱스 주소 지정

레지스터 간접 지정 방식에 변위가 더해진 메모리 주소 지정 방식
예시 : 20h만큼 더해 메모리를 참조한 명령

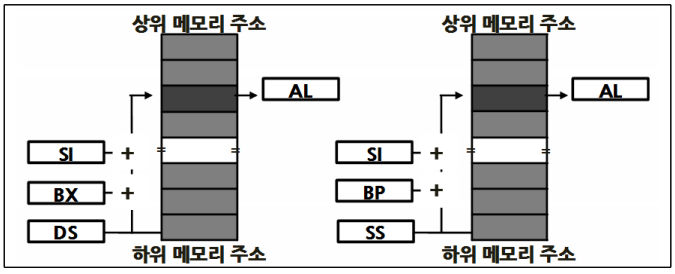
* 인덱스 주소 지정의 예
MOV AL, [BX+20h]
MOV AL, [BP+20h]



* 다음과 같이 바꿔서 표현할 수 있음
MOV AL, 20h[BX]
MOV AL, 20h[BP]

6. 베이스 인덱스 주소 지정

실제 주소 생성 위해 베이스 레지스터(BX 또는 BP)와 인덱스 레지스터(DI 또는 SI)를 결합
2차원 배열의 주소 지정에 사용
* 베이스 인덱스 주소 지정의 예
MOV AL, [BX+SI]
MOV AL, [BP+SI]

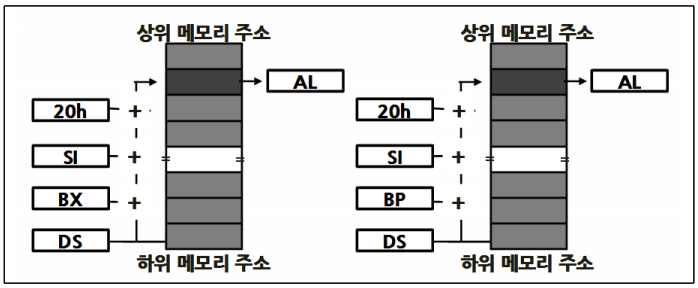


* 다음과 같이 바꿔서 표현할 수 있음
MOV AL, [BX][SI]
MOV AL, [BP][SI]

7. 변위를 갖는 베이스 인덱스 주소 지정

베이스-인덱스의 변형으로 실제 주소 생성 위해 베이스 레지스터, 인덱스 레지스터, 변위 결합

* 변위를 갖는 베이스 인덱스 주소 지정의 예
 MOV AL, [BX+SI+20h]
 MOV AL, [BP+SI+20h]



* 다음과 같이 바꿔서 표현할 수 있음
 MOV AL, [BX][SI][20h]
 MOV AL, [BP][SI][20h]

【학습정리】

1. 데이터를 리틀 엔디언 방식으로 스택에 저장하기 위해서는 상위 바이트와 하위 바이트를 나누어 각각 상위 번지와 하위 번지에 저장한다.

2. 어셈블리어의 주소 지정 방식은 레지스터 주소 지정, 직접 메모리 주소 지정, 레지스터 간접 주소 지정, 인덱스 주소 지정, 베이스 인덱스 주소 지정, 변위를 갖는 베이스 인덱스 주소 지정 방식이 있다.