제 4장 데이터 전송 명령

MOV 명령과 문자 출력
데이터의 입출력과 전송명령에 대해서 해설 합니다.
*.레지스터에 수치를 대입한다.
*.레지스터와 레지스터 사이에서 데이터를 전송한다.
*.레지스터와 메모리 사이에서 데이터를 전송한다.

화면에 AB 를 표시 하는 프로그램

```
MAIN    SEGMENT             ; 세그먼트를 알리는 의사명령
        ASSUME  CS:MAIN     ;
;
        MOV     DL,41H      ; 아스키코드 41H   A    자 이다.
        MOV     AH,2
        INT     21H
        MOV     DL,'B'
        MOV     BL,2
        MOV     AH,BL
        INT     21H
;                           ; 단순히 줄을 띄우기 위해 삽입된 설명문
        MOV     AH,4CH      ; 프로그램의 끝냄
        INT     21H
;
MAIN    ENDS
        END
```

자기가 작성한 소스프로그램을 그대로 보고 싶으면 리스트 화일인 .LST 를 살펴 보는 것이 좋다.의사명령은 직접 기계어로 번역되지 않지만 , 어셈블 작업의 흐름을 정할때에 중요한 역할을 해 줍니다.
여기서 MAIN 은 어떤 이름이든지 상관없다.

```
ASSUME  CS:MAIN
```

이라는 의사 명령은 어셈블러가 어셈블을 할때 코드세그먼트(CS:)를 참조 하라는 명령이 있으면 MAIN 이라는 이름의 세그먼트를 참조하라는 선언 을 하는 문장입니다.
ASSUME 문장은 4개의 세그먼트(CS,DS,SS,ES)와 세그먼트 이름을 대응시키는 선언문입니다.
명령은 반드시 CS:(코드세그먼트)에
END 문으로 어셈블러는 끝난다. END 문으로 시작번지를 지정할수가있다.

어셈블러 프로그램의 기본형

```
AMIN    SEGMENT
        ASSUME  CS:MAIN
       ..........
        본 문
       ..........
MAIN    ENDS
        END
```

설명문 (comment, 주석문)을 쓰는법

```
;
```

은 설명문 행 입니다. ; 이 다음 부터 쓰여진 문자는 설명문입니다.

8086 의 레지스터(register)

범용 레지스터 AX,BX,CX,DX 는 8 비트로 나누어 사용할수 있다.
범용 레지스터 SI,DI,BP,SP 는 16 비트로서 밖에 사용 할수 없다.

어셈블러에서는 특별히 표시하지 않는한 수치는 10진수로서 취급됩니다.
16진수를 사용할때는 숫자의 뒤에 H(Hexa decimal) 를 붙여서 표시합니다.
수치가 알파벳으로 시작할때는 0 을 붙여 변수이름과 혼동을 피한다.

```
MOV     DL,  41H        ; DL  <      41H      41H를 DL 에 대입하라.
```

행선지(DESTINATION), 출원지(SOURCE)

시스템 호출과 입출력은 INT 21H <   인터럽트 (interrupt-끼어들기) 명령은
강제로 CPU가 하던 일을 중지시키고 끼어들기

```
MOV     DL,'1'      ; 문자를 인용부호로 에워싼 것을 오른쪽에 쓴경우에는
                      그문자에 대응하는 아스키코드가 쓰여진 것으로 간주됩니다.
                      1 은 아스키코드값 31H 로 대치됩니다.
```

프로그램을 끝마치는 방법

```
MOV     AH,4CH      ; MS-DOS 의 SYSTEM CALL 인 4CH 번째의 루틴
INT     21H         ; 프로그램을 마치고 OS 레벨로 돌아간다.
```

데이터를 두는 방법(1)

프로그램중에 데이터를 두고 , 그 데이터를 레지스터에 전송하는 방법을 설명
변수이름 선언법 :

```
XXX     DB      'X'       ; DB 는 define byte
YYY     DW      3456H     ; DW 는 define word
                           > 변수에 들어 가는 초기치
                          >변수의 형
                         >변수이름
```

DB ..... 바이트의 정의
DW ..... 워드(2바이트)정의
DD ..... 더블 워드(4바이트)정의
DQ ..... 쿼트 워드(8바이트)정의
DT ..... 10 바이트 정의
*.사용법
변수이름 DB 식

세그먼트를 정의 하는 ASSUME 의사 명령

```
MAIN    SEGMENT
        ASSUME  CS:MAIN, DS:MAIN
```

MAIN : 세그먼트 개시를 선언
CS:MAIN 코드세그먼트가 MAIN 이라는 이름의 세그먼트에 연결되어있다는것
DS:MAIN 데이타세그먼트가 MAIN 이라는 이름의 세그먼트에 연결되어 있다는것

8086에서는 데이타는 데이타세그먼트 내에 있는 것으로서 해석되므로 ,데이터세그먼트

가 어디있는가 를 지정해야한다.

명령도 데이타도 둘다 MAIN 이라는 이름의 세그먼트 내에 있으므로
```
ASSUME  CS:MAIN, DS:MAIN
```

그러나 이것은 어셈블러에 대한 지시이다
데이타를 처리하는 명령으로는 번역되지않는다
세그먼트를 초기설정(initialize) 하기위해서는
```
MOV     AX,CS
MOV     DS,AX
```
이와같이하여 ,AX 레지스터를 경유하여 DS 에 CS 의값을 전송하여
2 개의 세그먼트를 일치시키지않으면 안됩니다.
 이것은 어셈블러의 정해진 문구입니다.
여기서
```
MOV     DS,CS
```
라고 할수는 없읍니다.세그먼트 레지스터(CS,DS,SS,ES)는 범용 레지스터들로 밖에는
데이터를 전송하는 것이 허용되어 있지않기 때문입니다.
세그먼트 레지스터에 직접 숫자나 변수를 대입할수는 없읍니다.
어셈블 리스트에서
```
8A 16 0020 R
```
R .... 이라는 문구는 상대번지(relative address) 지정방식임을 나타내고 있읍니다.

데이터를 두는 방법(2)

여러개의 세그먼트에 두는 법
```
MAIN    SEGMENT
        ......
MAIN    ENDS
DATA    SEGMENT
        .....
DATA    ENDS          이와 같은 두개의 세그먼트가 존재하게 설계한다.
```

```
ASSUME  CS:MAIN,DS:DATA
```
데이터를 참조하는 명령이 있는 경우에는 데이터는 DATA 라는 이름의 세그먼트에 있다
고 간주하고 명령을 작성하라는 지시입니다.

```
MOV     AX,DATA
MOV     DS,AX
```
데이터의 세그먼트의 값을 초기 설정하는 부분입니다.

프로그램을 시작할때 데이터 세그먼트의 지정법만 잘 익혀두면 본분 중에서는 데이터
가 어느 세그먼트 내에 있는가 신경쓰지 않고 ,프로그램을 작성할수있읍니다.

번지의 간접 지정법(1)

바이트형 데이터는 연속하여 정의할수있다.
DB 를 사용하여 데이터를 정의 하는 경우에 ABC ′ 와 같이 연속하여 몇 문자라도
데이터를 정의 할수가 있읍니다.
```
XXX     DB      'ABC'               모두같다.
XXX     DB      'A','B','C'
XXX     DB      41H,42H,43H
```

>DW,DD,DQ,DT 와 같은 다른 변수형의 데이터도 나열할수는 있습니다.
그러나 수치로써는 가능하지만 문자열로써 주어지는 경우는 한번에 2문자 씩의 문
자열밖에 줄수가 없다.

```
AAA     DD      'AB','CD','EF'는 허용이 되지만
AAA     DD      'ABCD', 'EFGH' 는 허용이 안된다.
```

DB 의 경우에만이 특별하고 3 문자 이상의 문자열의 나열을
'A','B','C' 로 하는 대신에 'ABC' 로 할수있다.

자,그럼 어떻게 XXX 라는 변수이름의 장소에 서 나열되어있는 데이터를 꺼내어 쓸수가 있는가

```
MOV     AL,XXX
```

라고 하면 XXX 의 장소에 나열된 문자열의 제일 처음 의 것인 'A'=41H 가 AL 레지스터에 전송됩니다.(XXX라고 하는 이름의 장소에 있는 데이터를 AL레지스터에 전송하라)

변수가 자리잡고 있는 번지를 꺼내는 오프셋(OFFSET 명령):

```
MOV     BX,offset XXX
```

변수이름이 붙여진 번지 자체의 값을 BX 레지스터에 전송하
> 는 명령입니다.

OFFSET : 변수이름이 붙여진 데이터가 있는 번지를 꺼내는 연산자.
오프셋번지 : 세그먼트를 기준으로 한 번지

```
MOV     BL,XXX ; BL 과  XXX 는 8 비트 이므로 가능
MOV     BX,XXX ; BX 는 16비트 레지스터, XXX는 8비트 레지스터이므로 에러
```

[ ] 를 사용한 간접 번지 지정방식:
어떤번지의 내용을 꺼낼때에는 [번지] 와 같이 [ ] 를 붙인다.

```
MOV     DL,[1FH] ; 1FH 번지에 있는 데이터를 DL 레지스터로 전송하라
```

BX 레지스터내에 변수 XXX 의 처음 데이터가 DL 레지스터로 전송됩니다.
그 다음은 이것을 표시 합니다.
2 번째 데이터는 선두 번지 +1 번지에 저장되어 있으므로, 여기서는 BX + 1 번지의 내용을 꺼내면 되는 것입니다.같은 방법으로 BX+2,BX+3 .....

```
MOV     DL,[BX+1]
```

[ ] 내의 값이 나타내는 번지에
저장되어 있는 내용을 꺼내는 것을 지시하고 있읍니다.
이와 같은 지정법을 간접지정방식(indirect addressing)이라고 부릅니다.

```
MOV     AX,[1234H]
```

와 같은 지정법도 간접번지 지정방식의 일종으로서의
직접번지지정방식(direct addressing mode) 이라고 부릅니다

```
MAIN    SEGMAIN                 ;  *.데이터를 세그먼트 내에두는 경우
        ASSUME  CS:CODE,DS:DATA

        MOV     AX,DATA
        MOV     DS,AX

          본  문

MAIN    ENDS

DATA    SEGMENT
XXX     DB      'ABC'
```

```
DATA    ENDS
        END
```

번지의 간접 지정법(2)

간접지정에 사용할수 있는 레지스터는 4개뿐이다
BX,BP,SI,DI
왜그런가 하는 것은 8086 설계자가 정해놓은 일이다.

[BX+DI]
[BX+SI+1]
와 같은 이들끼리의 조합지정이 가능 합니다
예) [BX+SI+4]
BX=1000H
SI=1234H 인 경우
1000H + 1234H + 4H = 2238H 이 된다.

간접지정 조합법의 규칙:

BX		SI		16비트 숫자
BP	+	DI	+	8비트 숫자
없다		없다		없다

[BP] 는 단독으로 사용할수가 없음니다.
단순히 기계적 제약에서 오는 것입니다 사용하고 싶다면 [BP+0] 으로 합니다.

```
MOV     DL,[BX+SI]
```

; BX에는 변수의 시작번지 , SI에는 3을 대입,
따라서 DL 에는 변수의 선두로 부터 4 번째 바이트의 데이터가 DL 레지스터에 전송한다 . *.SI= 0 부터 선두를 가리킨다.

예제) MOV5.ASM
메모리에서 저장된 문자열 ABCDEFG" 중 네번째, 다섯번째 문자 DE 를 출력

```
MAIN    SEGMENT
        ASSDUME CS:MAIN,DS:DATA
;
        MOV     AX,DATA
        MOV     DS,AX
        MOV     BX,OFFSET AAA
        MOV     SI,3
        MOV     DL,[BX+SI]
        MOV     AH,2
        INT     21H
        MOV     DL,[BX+SI+1]
        MOV     AH,2
        INT     21H
;
MAIN    ENDS
;
DATA    SEGMENT
AAA     DB      'ABCDEFG'
;
DATA    ENDS
        END
```

*. 간접지정에 사용 하는 레지스터에 BP 레지스터가 포함되는 경우에는 ,
스텍세그먼트(SS:)에 데이터가 있는 거으로서 해석된다.
(보통은 BP 레지스터를 사용한 간접 지정법은 안하는것이 무난할 것입니다.)

간접 번지 지정방식의 사용법

```
CODE    SEGMENT
        ASSUME  CS:CODE,DS:DATA
;
        MOV     AX,DATA          ; DS를 설정
        MOV     DS,AX
;
        MOV     BX,OFFSET AAA    ; AAA의 번지값을 BX 에 대입
        MOV     AX,'AB'          ; 문자 하나는 8비트
        MOV     [BX],AX          ; BX 가 가리키는 AAA 변수에  AB'를 넣는다
        MOV     CX,AAA
        MOV     DL,CH            ; DL 에 A 를 넣는다
        MOV     AH,2
        INT     21H              문자 A를 보인다.
        MOV     DL,CL            ; DL 에 B 를 넣는다.
        MOV     AH,2             문자 B를 보인다.
        INT     21H
        MOV     AH,4CH
        INT     21H              ; 종료하고 OS 로돌아간다.
;
CODE    ENDS
;
DATA    SEGMENT
AAA     DW                       ; AAA 라는 변수를 위해서 워드 영역을 확보하라
DATA    ENDS
        END
```

AAA DW ? 는 '? ' 는 숫자는 아무것이라도 좋다라는 의미입니다.

이밖의 사용법으로는
MOV [BX+SI+5],1234H
MOV AAA,'AB'
라는 사용법이 가능 합니다.

MOV [BX],[SI+3]
그러나 이와 같은 ,오퍼랜드 양쪽 다 간접지정으로 할수는 없읍니다.

MOV BX,[0008] > 8 번지의 내용을 꺼내온다 라는 간접 지정방식
레지스터를 사용하지않고 수치만으로 번지를 지정하는 이러한 방법을 직접번지 지정(direct addressing)이라고 말합니다.
또한, MASM 에서는 이와 같은 직접 번지 지정방식을 어셈블러 속에 쓸수가 없읍니다.
그것은 메모리 번지는 링커에 의해 동적(dynamic)으로 활당되기때문에 ,고정된 번지 지정방식을 사용 할수없다고 가정되기 때문이라고 생각됩니다.

EQU 의사 명령

상수를 정의 하는 EQU 의사 명령

```
MAIN    SEGMENT
        ASSUME  CS:MAIN
CHAR1   EQU     'A'     ;'A' = 41H 가 정의 됩니다
CHAR2   =       42H     ;  = 는 재정의가 가능
DISP    EQU     2       ;  EQU는 재정의가 불가능,    상수는 어셈블할때 치환
;
        MOV     DL,CHAR1
        MOV     AH,DISP
        INT     21H
        MOV     DL,CHAR2
        MOV     AH,DISP
        INT     21H
        MOV     AH,4CH
        INT     21H
;
MAIN    ENDS
        END
```

EQU 의사명령에 의한 상수의 정의와 DB 등에 의한 변수의 정의와의 차이는 :
EQU 에 의해서 선언된 상수 - 어셈블할때 직접숫자로 치환
DB 등에 의해서 선언된 변수 - 데이터가 저장되어 있는 번지로서 치환
예)
MOV CHAR1,DL > 불가능
'=' 에 의해서 정의된 숫자를 다시 정의 할때는 ,다시 = 를 사용해서 정의 한다.
어셈블러는 시작순서대로 어셈블을 해나가면서 가장 최근 에 정의된 숫자를 그 상수의 숫자로써 사용합니다.
따라서 다음과 같이 됩니다.

```
CHAR1   =       41H
        MOV     DL,CHAR1
CHAR1   =       56H
        MOV     DL,CHAR1
```

결과 :

```
=       0041
0000    B2      41
=       0056
0002    B2      56
```

데이터의 형과 PTR 연산자

데이터의 형(type):
바이트,워드(2바이트),더블워드(4바이트)
데이터의 형은 레지스터의 크기와 같다고 간주된다.

주의)
오퍼랜드에 레지스터를 포함하지 않은 경우 크기의 지정이 필요
예) 한쪽이 간접 번지지정 ,또 한쪽이 숫자의 경우입니다.

```
MOV     [BX],12H
```

여기에서 BX 의 값이 10H 라고 합시다.그런데 여기서는
1) 10H 번지에 바이트 데이터 12H 를 저정할것인지,
2) 10H 번지와 11 번지에 워드 데이터 0012H 를 저장하는 것인지 알수가 없음니다.

*.바이트 데이터의 경우

번지			
10H	55	12H	12
		>	
11H	66		66

*.워드데이터의 경우

번지			
10H	55	0012H	12
		>	
11H	66		00

12H 를 바이트 데이터로써 저장한 경우에는 11H 번지의 내용은 사라지지 않고 남지만
12H 를 워드 데이터로써 저장한 경우에는 11H 의 내용은 지워져버립니다.

PTR 연사자의 사용

데이터의 형을 명확하게 하기위하여
```
MOV     BYTE PTR [BX],12H  ; PTR 은 POINTER 의 약자
MOV     WORD PTR [BX],12H
```
이와 같이 전송되는 측에 BYTE 또는 WORD 를 지정합니다.
이때 PTR 이라는 오버라이트 (overwrite) 연산자와 함께 사용하도록 되어 있읍니다.
변수로 선언되어 있는 형이 DB 이든 DW 이든 BYTE , WORD 양쪽다 사용할수가 있읍
니다.

MOV 명령과 번지지정방식의 정리

ADDRESING MODE:
*.즉치 방식(immediate mode)
```
AAA     EQU     1234H
        MOV     AX,AAA  ;메모리의 바이트수가 레지스터바이트 이내일것
```
CS,DS,ES,SS 및 IP,FL 레지스터에 직접 수치를 전송할수는 없읍니다.

*.직접 방식(레지스터 번지지정 방식)
레지스터의 내용을 직접 전송하는 방식
```
MOV     DS,AX
MOV     CL,BH   ; 양쪽의 레지스터의 크기가 일치하는 8비트 혹은 16비트여야 한다.
```
세그먼트 레지스터간의 전송은 할수가 없다 . IP ,FL 레지스터는 사용할수가 없고 ,
CS 레지스터로의 값을 전송은 할수 없읍니다.

*.간접 방식(메모리 번지지정 방식)
전송하는 값이 저장되어 있는 번지를 지정- 방법2가지
1) BX,BP,SI,DI 의 4 가지 레지스터와 숫자를 조합시킨다.
```
MOV     AX,[BX+DI+4]
```
2) 데이터를 변수로서 정의하고 ,그변수이름을 사용하여 지정하는 경우
```
MOV     AX,AAA ; 숫자가 있는 번지에 붙여진 이름을사용하여 간접지정을 하는것
```

간접 방식에서의 번지지정법:

BX		SI		8비트 숫자
BP	+	DI	+	16비트 숫자
없음		없음		없 음

위에서 허용되지 않는 조합방법
1) 8비트 숫자만을 사용하는 경우(번지는 16비트가 아니면 지정할수 없다)
 그러나 MOV AX,[1234H] 와 같은 직접 번지지정 방식은 사용할수 없다.
 직접 메모리 번지를 지정하여 간접 방식을 사용하고 싶을때는 -
```
MOV     BX,1234H
MOV     AX,[BX]    ; 와 같이 하지 않으면 안됩니다.
```
2) [BP]를 사용 할수 없다.
 [BP]를 사용하고 싶을 때에는 [BP + 0]를 하면 된다

간접지정할때의 주의 사항 :
1)명령은 모두 코드 세그먼트에 있다고 본다
2)데이터는 데이터 세그먼트 내에 있다고 가정된다.
3)BP를 포함하는 간접지정에서는 데이터는 스택 세그먼트 내에 있다고 가정된다.
4)프로그램의 처음에서 데이터 세그먼트의 초기설정이 필요
```
  MOV   AX,DATA
  MOV   DS,AX    ; DATA  를 데이타 세그먼트에 맞춘다.

  MOV   AX,CS
  MOV   DS,AX    ; 데이타 세그먼트를 코드세그먼트와 맞춘다.
```

그런데 어떻게 해서라도 데이터 세그먼트 이외의 세그먼트로부터 데이터를 가져오고 싶다는 경우 세그먼트 오버라이트 프리픽스(segment overwrite prefix)를 설정,그 명령에 한해서 지정한 세그먼트의 지정한 번지로 부터 데이터를 갖고 오거나,가지고 갈 수 있읍니다.
예)
```
MOV     AX,ES:[BX]
MOV     CS:[DI+2],CX
```
세그먼트 오버라이트 프리픽스는 다음의 4가지 입니다.
CS:,DS:,SS:,ES:

MOV 명령으로 전송가능한 조합

```
MOV     AX,CS
MOV     DS,AX
```
등 과 같은 형태로 DS <---- CS 를 한 이유는 ,세그먼트 레지스터간의 전송이 허용되지 않기 때문입니다.
또한 ,세그먼트 레지스터에는 직접 숫자를 대입 하는 것이 허용되어 있지 않습니다.
IP,FL 레지스터는 MOV 명령에서 데이터 전송을 할수 없는 것으로 되어 있읍니다.
CS 레지스터에의 데이터의 전송은 허용되지 않습니다.
CS 레지스터는 읽어내기만 가능합니다.