제 4 장

어셈블리어 2

학습내용

* 프로그래밍 언어
* 어셈블리어 개요
* 어셈블리 프로그래밍

화면에 A문자를 출력하는 프로그램

```
1  MAIN SEGMENT
2        ASSUME CS : MAIN
3 ;
4        MOV DL, 'A'
5        MOV AH, 2
6        INT 21H
7 ;
8        MOV AH, 4CH
9        INT 21H
10;
11       MAIN ENDS
12       END
```

① MOV <Operand 1>, <Operand 2>
- 연산 항에는 레지스터, 메모리 주소, 상수 등 올 수 있음

② 6번 문장의 INT 21H 명령
- 21H번의 인터럽트를 호출하는 명령
  강제적으로 CPU가 하던 일을 중단시키고 MS-Dos에 의한 입출력 등의 시스템 함수의 호출에 사용
  AH레지스테에 들어 있는 값에 따라 다른 기능 수행

AH가 2 일때 :DL레지스터에 저장되어있는 ASCII코드에 해당하는 문자가 화면상에 표시

③ 8, 9번 문장
- 4CH를 레지스터 AH에 저장하고
  INT 21H명령으로
  인터럽트 번호 21H를 호출하여
  4CH에 해당되는 기능 수행

AH가 4CH 일때 :실행중의 프로그램을 끝내고 DOS상으로 되돌아가라는 명령

※ DOS 함수 호출
  2 : 콘솔로 한 문자를 출력
  4C : 프로그램 종료

응용1-1. 화면에 문자 “AB” 를 출력하는 프로그램 (ab_1.asm)

응용1-1. 화면에 문자 “AB” 를 출력하는 프로그램 (ab_1.asm)

```
1 MAIN SEGMENT ; 세그먼트를 알리는 의사명령
2 ASSUME CS:MAIN ;
3 ;
4 MOV DL,41H ; 아스키코드 41H ' A ' 자 이다.
5 MOV AH,2
6 INT 21H
7 MOV DL,'B'
8 MOV BL,2
9 MOV AH,BL
10 INT 21H
11 ; ; 단순히 줄을 띄우기 위해 삽입된 설명문
12 MOV AH,4CH ; 프로그램의 끝냄
13 INT 21H
14 MAIN ENDS
15 END
```

MOV 명령과 문자 출력

데이터의 입출력과 전송명령에 대해서 해설 합니다.

*.레지스터에 수치를 대입한다.

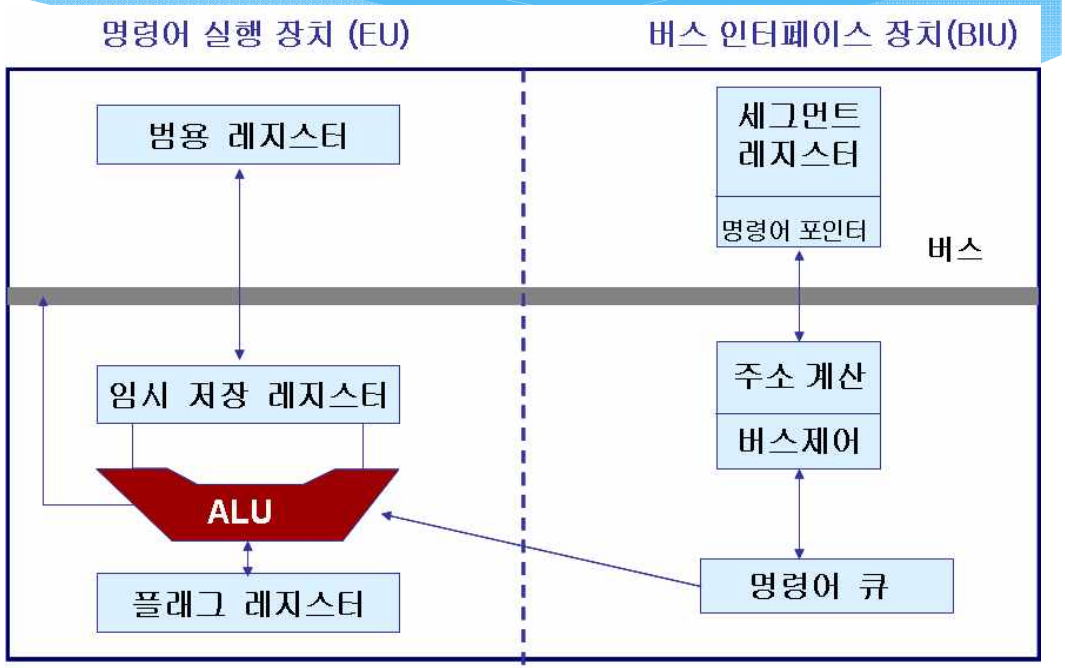
*.레지스터와 레지스터 사이에서 데이터를 전송한다.

*.레지스터와 메모리 사이에서 데이터를 전송한다.

응용1-2. 화면에 문자 “AB” 를 출력하는 프로그램 (ab_2.asm)=> 데이터 정의 지시어 사용

분기와 표지 기호

1. 순차적 실행과 분기



1) 순차적 실행

(1) 개념

- 프로그램은 명령어가 저장된 순서대로 실행

(2) 절차 (p44 그림2-5)

① BIU는 IP에 있는 내용을 주소버스로 출력

② 주소가 가리키는 기억장소에서 명령어를 인출하여 명령어 큐에 저장

③ EU에서 명령어를 명령어 큐로부터 읽어 와서 해독

④ 명령어 실행

⑤ IP 증가 , ④와 ⑤는 동시에 수행될 수 있음.

2. 분기

(1) 개념

- 프로그램 실행 순서를 바꾸는 것
- 레이블(Label)을 붙여 사용
- 분기에는 무조건 분기와 조건 분기

부호가 없는 경우	A : Above	크다
	B : Below	작다
부호가 있는 경우	G : Greater	크다
	L : Less	작다
	E : Equal	같다
	N : Not	~가 아니다
	O : Overflow	오버플로
	P : Parity	패리티
	S : Sign	음수
	Z : Zero	혹은 같음

(2) 조건 분기 명령어 형식

Jcc <분기되어 갈 주소>

- cc에 올 수 있는 문자들

(3) 조건 분기 명령

: 일반적으로 비교 명령이나 연산 명령의 결과에 의해 플래그 레지스터가 세팅된다. 그러면 조건 분기 명령은 세팅된 플래그 레지스터의 값에 따라서 분기되어진다.

```
CMP AX,100
JA Loop1 ; if AX > 100 then Loop1으로 분기
JE Loop2 ; if AX=100 then Loop2로 분기
```

(4) 조건 분기 명령의 종류

플래그	1:분기	0:분기	1일 때의 의미
SF	JS	JNS	보수표현으로 음수
ZF	JZ/JE	JNZ/JNE	결과가 0
PF	JP/JPE	JNP/JPO	1의 수가 짝수 개
CF	JB/JNAE	JNB/JAE	올림수 발생
OF	JO	JNO	결과의 범위초과
CX	-	JCXZ	CX 레지스터가 0일 때만 분기

for(cx=1; cx<=100; cx++)
AX += CX

(5) 1~100 합을 구하는 프로그램(예제4-2:sum_1.asm)

```
1   MAIN    SEGMENT
2           ASSUME    CS:MAIN, DS:MAIN
3           MOV       AX, CS
4           MOV       DS, AX
5           MOV       CX, 1
6           MOV       AX, 0
7   LOOP1 : ADD       AX, CX
8           INC       CX
9           CMP       CX, 100
10          JBE       LOOP1
11          MOV       SUM, AX
12          MOV       AH, 4CH
13          INT       21H
14  SUM     DW        ?
15  MAIN    ENDS
16          END
```

AX를 이용하여 CS,DS의 주소 통일

CX=1, AX=0

AX += CX
CX++
CX <=100이면 LOOP1으로 분기

(6) 1~100 합 프로그램 실행전과 실행후의 기억장소 상태

세그먼트 베이스 주소

오프셋 주소

0A5C :	0000	MOV	AX, CS
0A5C :	0002	MOV	DS, AX
0A5C :	0004	MOV	CX, 0001
0A5C :	0007	MOV	AX, 0000
0A5C :	000A	ADD	AX, CX
0A5C :	000C	INC	CX
0A5C :	000D	CMP	CX, +64
0A5C :	0010	JBE	000A
0A5C :	0012	MOV	[0019],AX
0A5C :	0015	MOV	AH, 4C
0A5C :	0017	INT	21
0A5C :	0019	00	00
0A5C :	001B		

실행 전

0A5C :	0000	MOV	AX, CS
0A5C :	0002	MOV	DS, AX
0A5C :	0004	MOV	CX, 0001
0A5C :	0007	MOV	AX, 0000
0A5C :	000A	ADD	AX, CX
0A5C :	000C	INC	CX
0A5C :	000D	CMP	CX, +64
0A5C :	0010	JBE	000A
0A5C :	0012	MOV	[0019],AX
0A5C :	0015	MOV	AH, 4C
0A5C :	0017	INT	21
0A5C :	0019	BA	13
0A5C :	001B		

실행 후

[응용 2(예제 4.2) :sum_2.asm] 1~100 합을 구하는 프로그램

```
CODE   SEGMENT
*        ASSUME  CS:CODE, DS:CODE
*    MOV  AX,CODE  ; DS설정
*    MOV  DS,AX
*    ;
*    MOV  CX,100    ; CX에 100을 지정
*    MOV  AX,0      ; AX 에 0을 지정
 NEXT: ADD AX,CX
*    (                )
*    (                ); CX의 내용이 0인가 비교
*    (                );0이면 밑으로 아니면 NEXT 라는 라벨로 이동
*    MOV  TOTAL,AX
*    MOV  AH,4CH  ;끝내고 MS-DOS로 돌아간다.
*    INT 21H
*    ;
*    TOTAL  DW ?
*    CODE  ENDS
*    END
```

라벨(label)의 사용:

라벨이라는 것은 명령 등이 있는 번지에 붙여진 이름입니다.

라벨이름의 직후에 콜론" : " 을 넣는다

[응용 3:sum_3.asm] 4040H+0102H 를 더하고 그 결과 4142H 의 41H ,42H에 해당하는 문자를 출력

⇒ 두 값을 더하는 것은 BX레지스터를 이용하여 사용한다

[응용4:sum_4.asm] 변수 BB의 200 에서 100 을 빼 그 결과를 변수 BB에 저장하는 프로그램

=> 변수 BB를 정의하고 AL레지스터를 이용

[응용5:sum_5.asm] 4개의 숫자 50H,60H,80H,F0H를 합하여 ANS에 저장한다.

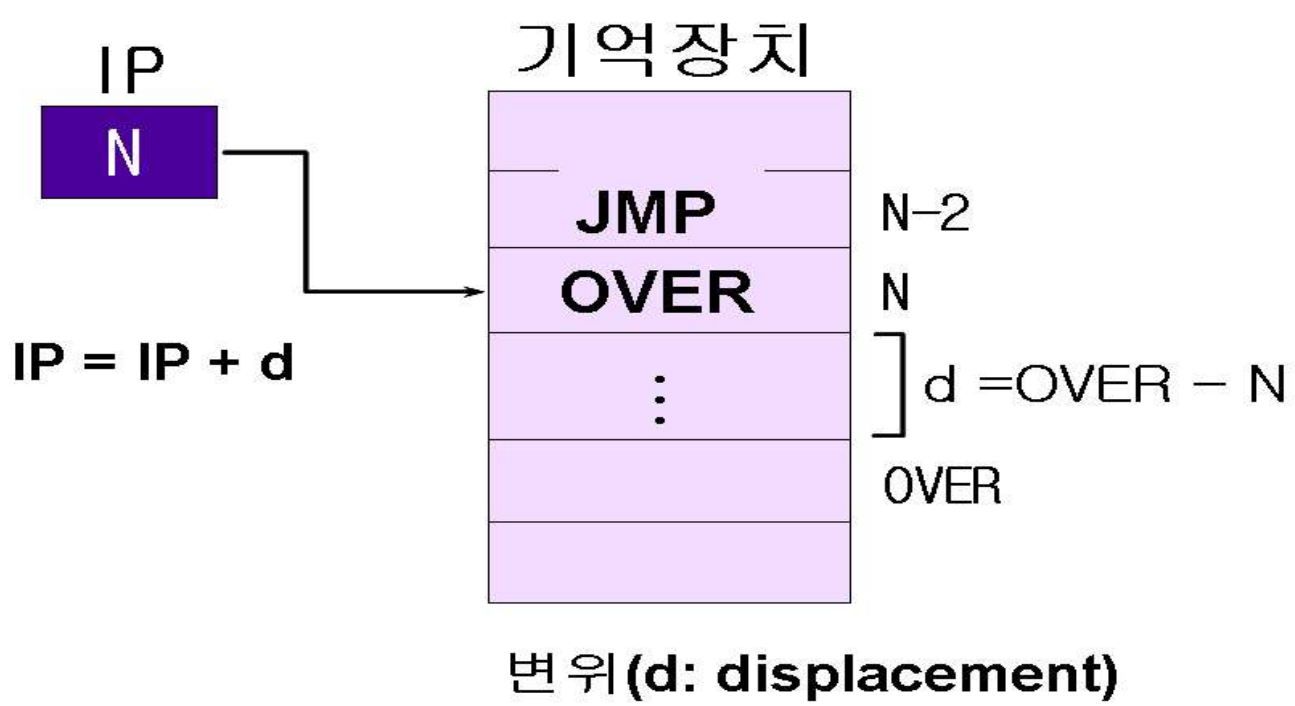
=> 변수들을 정의하고 AX와 DX레지스터를 이용하여 합하고 최종 결과 값을 변수 ANS에 저장한다.

3. 무조건분기

(1) 개념

- 무조건 분기를 사용하기 위해 **JMP, CALL** 명령어 사용
- 보통 고급언어의 **GOTO**문장과 같음
- **JMP**는 순차적 실행 절차와 같고 단 명령어 실행과 **IP**값 지정 절차가 다름

(2) 과정



- **JMP OVER : OVER**는 분기되어 갈 곳의 주소
- 분기되어 갈 주소는 **IP**에 저장되며, 이것은 **OVER**의 주소와 현재 명령어 주소의 차이로 결정
- 이 차이를 변위(d) 즉 변위(d)=OVER-IP

(3) 절차

① **BIU**는 **IP**에 있는 내용을 주소버스로 출력

② 주소가 가리키는 기억장소에서 명령어를 인출하여 명령어 큐에 저장

③ **EU**에서 명령어를 명령어 큐로부터 읽어 와서 해독

④ 분기되어 갈 주소(**JMP** 명령어의 연산항에 있는 기호)와 현재 **IP**의 값의 차이(변위 **:d**)를 계산함

⑤ **IP = IP + d**

[예제 4.3:keyb_1.asm] 키보드로 부터 입력 받은 문자 중 소문자를 대문자로 변환

```
 1   MAIN  SEGMENT
 2         ASSUME  CS:MAIN
 3   L1: MOV     AH, 1
 4         INT       21H
 5         CMP     AL, 1AH    ; ^ z
 6         JE       FIN
 7         CMP      AL,'a'
 8         JB       L2
 9         CMP      AL,'z'
10         JA       L2
11         SUB      AL, 'a'-'A'
12    L2:  MOV      DL, AL
13         MOV      AH, 2
14         INT      21H
15         JMP      L1
16   FIN:  MOV      AH, 4CH
17         INT      21H
18   MAIN  ENDS
19         END
```

키입력 방법

키보드로 부터 한 문자를 입력하려면 ,MS-DOS 의 펑션 호출(function)의 1 번을 사용.

AH 레지스터에 1 을 설정하고 펑션 호출을 수행하면, 키보드로부터 입력이 있을때 까지 기다리고 있다가 ,입력된 문자의 아스키코드를 AL register 로 돌려 보내줍니다.

① SUB 명령어

- SUB <operand 1>, <operand 2>
- SUB AL, 20h : AL에 저장되어 있는 값에서 20h를 뺀 후 이 값을 다시 AL에저장

[예제 4.3:keyb_1.asm] 키보드로 부터 입력 받은 문자 중 소문자를 대문자로 변환

❖ 키보드로 부터 입력 받은 문자 중 소문자를 대문자로 변환해주는 프로그램

❖ 기본 알고리즘

  ❖ 키보드로 부터 입력 받은 문자가 'a' 보다 작거나 'z' 보다 크면 그대로 출력하고 아니면 20H(32)값을 빼주고 그 값에 해당하는 문자를 화면에 출력

❖ 'a'의 ASCII 코드 값은 61H이고 'A'의 ASCII 코드 값은 41H

❖ 키입력 방법

❖ 키보드로 부터 한 문자를 입력하려면 ,MS-DOS 의 펑션 호출(function)의 1 번을 사용.

❖ AH 레지스터에 1 을 설정하고 펑션 호출을 수행하면, 키보드로부터 입력이 있을 때 까지 기다리고 있다가 ,입력된 문자의 아스키코드를 AL register 로 돌려 보내줍니다.

[응용 6: keyb_2.asm] 키보드로 부터 입력 받은 문자 중 대문자를 소문자로 변환