제04강

기본 연산 명령

기본 연산 명령

전송, 산술, 논리

실습과제

ref.) Chapter 4

명령어 관련(2장)

* #data : 8비트의 즉치 데이터
  ; #을 앞세우며, 진법을 위해 B, O, D, H 접미
  예) #01010111B ; 2진수
  #127O ; 8진수
  #99 혹은 #99D ; 10진수
  #99H ; 16진수
  #0F2H ; 16진수(A~F로시작)
* #data16은 16비트의 즉치
* direct는 직접주소를 의미
* Rn은 범용 레지스터
  ; 선택된 레지스터 뱅크내의 8개(R7, R6, ..., R0)
* @Ri는 레지스터 뱅크내의 R0, R1을 간접주소지정

전송 명령

* 대상 메모리 공간에 따라(3가지)

MOV 명령

;내부 데이터 메모리간의 데이터 전송 명령

MOVX 명령

;내부 데이터 메모리와 외부 메모리간 전송명령

;외부 메모리간 전송 명령

;즉, 외부의 프로그램 메모리나 데이터 메모리가
데이터 근원지나 목적지로 사용될 때

MOVC 명령

;프로그램 메모리는 ROM이 사용되므로
이러한 메모리로부터 판독 기능만 제공

전송 명령(계속)

* 전송 명령의 사용 예

```
MOV   A,#40H          ; A <- 40H

MOV DPTR,#8000H
MOVX @DPTR,A          ; (8000H) <- A
```

: #40H은 즉치를 나타내는 것으로,
#을 생략하면 전혀 다른 의미임을 유의!

```
MOV A,40H             ; A <- (40H)
```

;내부 데이터메모리 40H주소의 내용을 A로 전송

전송 명령(계속)

* 주변장치와의 입출력에 전송명령 사용

: MCS-51은 Memory mapped I/O방식 사용

: 전용의 입출력 명령이 불필요

예) 입출력 포트 P1(90H)에 출력 및 입력시

```
MOV P1,#0FFH        ; 출력
MOV A,P1            ; 입력
```

예) 출력디바이스에 출력시

```
MOV DPTR,#0F800H    ; 디바이스주소 F800H라 가정
MOVX @DPTR,A        ; 출력
```

전송 명령(계속)

* 내부데이터 메모리간의 전송명령(MOV 명령)

```
MOV A, #data
MOV A, direct
MOV A, @Ri
MOV A, Rn

MOV Rn, A
MOV Rn, #data
MOV Rn, direct

MOV @Ri, A
MOV @Ri, #data
MOV @Ri, direct
```

MOV direct, A
MOV direct, #data
MOV direct, Rn
MOV direct, @Ri
MOV direct, direct
MOV DPTR, #data16

-간접주소 지정가능한 범용레지스터(@Ri)는
R0와 R1뿐임

MOV @R0,A ; (R0) <- ACC

전송 명령(계속)

* 외부 데이터 메모리와의 전송 명령

```
MOVX A,@DPTR          ; 판독

MOV P2,#80H           ; 외부메모리 주소의 상위바이트
MOVX A,@R0            ; A <- P2_(R0)
혹은, MOVX A,@R1      ; A <- P2_(R1)

MOVX @DPTR,A          ; 기록

MOV P2,#80H           ; 외부메모리 주소의 상위바이트
MOVX @R0,A            ; P2_(R0) <- A
혹은, MOVX @R1,A      ; P2_(R1) <- A
```

전송 명령(계속)

1) DPTR을 이용한 간접주소지정 방식을 사용

```
MOV DPTR,#8020H
MOVX A,@DPTR
```

2) R0와 R1을 이용하는 경우

: 상위 바이트의 값은 P2에 미리 설정하고

: R0와 R1은 하위 바이트에 해당하는 값을 설정

```
MOV P2,#80H      ;주소의 상위 바이트
MOV R1,#20H      ;주소의 하위 바이트
MOVX @R1,A       ;8020H주소에 A값을 기록
```

전송 명령(계속)

* 프로그램 메모리와의 전송명령

: 지시되는 위치 내용의 판독만 가능

: DPTR 혹은 PC의 위치를 기준으로
ACC(offset)에 의한 간접 주소지정방식 사용

: 일반적으로 LUT(Look Up Table) 참조시 사용

: 데이터메모리(RAM)영역을 읽기만 하는 경우에 사용가능

```
MOVC  A,@A+DPTR
MOVC  A,@A+PC
    ; DPTR, PC 값은 접근시 기준주소가 됨
```

PSW(Program status word)

* 산술 연산의 결과를 반영하여 상태정보 유지

7	6	5	4	3	2	1	0
C	AC	F0	RS1	RS0	OV	–	P

- C(carry) 플래그 비트
- AC(auxiliary carry) 플래그 비트
- F0 플래그 비트 : 범용 목적
- RS1과 RS0 비트 : 사용할 레지스터 뱅크를 설정
- OV(overflow) 플래그 비트
- P(parity) 플래그 비트 : Even 패리티 비트 생성

PSW(Program status word)(계속)

* 예) 레지스터 뱅크를 RB2로 설정

```
    ....                          ;리셋상태시 RB0선택
MOV R0,#69H                       ;(00H) <- 69H
MOV PSW,#00010000B                ;RB2 설정
MOV R0,#69H                       ;(10H) <- 69H
```

* 수정 가능한 비트는 C 플래그 비트

```
CLR C                 ; C <- 0
SETB C                ; C <- 1
CPL C                 ; C <- NOT C
```

증감 명령

* 지정된 데이터 값을 1씩 증가 혹은 감소

INC A	DEC A
INC @Ri	DEC @Ri
INC Rn	DEC Rn
INC direct	DEC direct
INC DPTR	~~DEC DPTR~~(지원않음)

유의!!) DEC DPTR 명령은 정의되지 않음

: DPTR을 바이트 별로 취급

: DPH는 그대로 두고, DEC DPL 사용

산술연산 명령

* 덧셈 명령

: 연산에 캐리 포함여부에 따라 2가지 형태

ADD A,#data	ADDC A,#data
ADD A,@Ri	ADDC A,@Ri
ADD A,Rn	ADDC A,Rn
ADD A,direct	ADDC A,direct

* 뺄셈 명령

: 빌림을 포함하여 연산하는 SUBB 명령만 지원

: 위와 동일한 유형으로 사용(SUBB 명령)

SUBB A,#data , ...

산술연산 명령(계속)

* 곱셈 및 나눗셈을 위한 명령 지원

: ACC와 B 레지스터를 고정된 목적으로 사용

MUL AB

; A*B 결과의 상위바이트는 B,

; 하위바이트는 A에

DIV AB

; A/B 결과의 나머지는 B에,

; 몫은 A에

산술연산 명령(계속)

* 10진 보정 명령(DA A 명령)

: 그 결과가 ACC에 저장되는 산술 연산 명령이
수행된 직후에 수행되도록 하여야 함

: BCD 연산에 이용

```
예)     MOV A,#98H
        ADD A,#37H          ; A <- CFH
        DA A                ; C <- 1, A <- 35
```

논리연산 명령

: 피연산자들간의 비트단위 논리 연산

* ANL 명령 : AND 논리 연산을 수행하는 명령

```
ANL A,#data
ANL A,@Ri
ANL A,direct
ANL direct,#data
ANL direct,A
```

* 동 유형의 ORL(OR연산), XRL(XOR연산) 명령

* 비트반전 및 클리어 관련 명령(ACC를 대상)

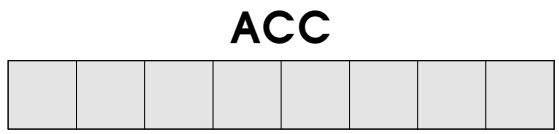
```
CPL A          ; A <- A의 비트반전
CLR A          ; A <- 0
```

회전 명령

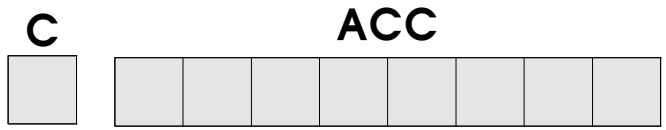
* 회전명령

: ACC만을 대상으로 연산

```
RL A                ;C 비트의 고려없이 8비트를
RR A
```




```
RLC A               ;C 비트까지 포함하여 9비트를
RRC A
```



교환 명령

* XCH 명령

  : 바이트 단위로 두 피연산자간 교환

```
XCH A,@Ri
XCH A,Rn
XCH A,direct
```

* XCHD 명령

  : 피연산자의 하위 니블만을 서로 교환

```
XCHD A,@Ri
```

* SWAP 명령

  : ACC를 대상으로 상위 니블과 하위니블을 교환

```
SWAP A
```

비트단위 처리명령

* 비트 단위의 주소지정 방식

: 내부 램의 주소와 비트 인덱스를 이용

```
20H.7        ; 20H주소의 비트인덱스 7 (≡07H)
```

: SFR 명과 비트 인덱스를 이용

```
P1.0         ; P1 SFR의 비트인덱스 0
```

* 비트 연산 명령

```
CLR <bit address>        ;0으로 설정
SETB <bit address>       ;1로 설정
CPL <bit address>        ;비트반전
```

비트단위 처리명령(계속)

* 비트 단위 논리연산

: 비트단위 논리연산을 위한 피연산자는 C 플래그

ANL C,<bit address>

ANL C,/<bit address>

ORL C,<bit address>

ORL C,/<bit address>

참고) /<bit address>

: 해당 비트주소의 내용을 비트 반전하라는 의미

비트단위 처리명령(계속)

* 비트 단위 전송 명령

: 캐리 비트와 비트 주소로 구성

```
MOV C,<bit address>
MOV <bit address>,C
```

* 비트 검사 분기 명령

: 특정 비트를 검사하여 그 결과에 따라 분기

```
JB <bit address>,<8bit offset>        ; 레이블
JNB <bit address>,<8bit offset>
JBC <bit address>,<8bit offset>
```

실습과제

[실습1] 두 정수의 덧셈

```
;=====================================
;   OPR_01.ASM
;       Addition of two data
;=====================================
$MOD51
            ORG 8000H
MONITOR     EQU 0000H
            ;
START:
            MOV R0,#20H     ; OP1
            MOV R1,#30H     ; OP2

            MOV A,R0
            ADD A,R1        ; 덧셈

            MOV DPTR,#RES
            MOVX @DPTR,A

            JMP MONITOR     ; 모니터 프로그램으로

            ORG 80F0H
```

```
RES:        DB  00H             ; 결과
            END
```

실습과제(계속)

[실습2] 산술 연산 명령

```
;=====================================
;    OPR_02.ASM
;         산술연산
;=====================================
$MOD51
              ORG 8000H
MONITOR       EQU 0000H
    ;
START:
              MOV R0,#89H      ; OP1
              MOV R1,#11H      ; OP2

              ; 더하기
              MOV A,R0
              ADD A,R1         ; ADD

              MOV DPTR,#RES    ; 결과저장 위치설정
              MOVX @DPTR,A     ; 결과저장

              ; 빼기
              CLR C
              MOV A,R0
```

```
SUBB A,R1           ; SUBB

INC DPTR
MOVX @DPTR,A

; 곱하기
MOV A,R0
MOV B,R1
MUL AB              ; MUL

INC DPTR            ; low byte
MOVX @DPTR,A

INC DPTR
MOV A,B             ; high byte
MOVX @DPTR,A

; 나누기
MOV A,R0
MOV B,R1
DIV AB              ; DIV

INC DPTR            ; 몫
MOVX @DPTR,A

INC DPTR
MOV A,B             ; 나머지
MOVX @DPTR,A
```

```
        JMP  MONITOR      ; 모니터 프로그램으로

        ORG  8100H
RES:    DB  00H,00H,00H,00H    ; 결과
        DB  00H,00H,00H,00H
        DB  00H,00H,00H,00H
        DB  00H,00H,00H,00H

        END
```

실습과제(계속)

[실습3] 논리 연산 명령

[실습4] 회전 및 교환 명령

[실습5] 다중 바이트 연산

: 매우 큰 수간의 덧셈

예) 745C7898H + A9AD3432H = ?

실습과제(계속)

[실습6] 다중 바이트 BCD 연산

예) 12345678 + 56781234 = ?