제05강

# 분기명령 및 반복처리

분기명령

반복처리

블록전송, 블록합산, 순차검색, 버블정렬

실습과제

ref.) Chapter 5

# 반복 처리

## * 반복 처리

: 동일 연산이 반복될 때 실행의 흐름을 변경하여
이를 효과적으로 처리

## * 분기 명령과 레이블

: C언어에서 goto문 활용과 유사

–분기명령 : 프로그램에서 실행의 흐름을 바꾸는 명령

: 조건검사여부에 따라, 무조건분기 vs. 조건분기

: 분기주소표현에 따라, 절대주소분기 vs. 상대주소분기

–레이블: 분기할 위치를 표시

# 분기 명령

## * 무조건 분기명령

: 조건 검사 없이 지정된 위치로 실행의 흐름을
변경하는 명령

: 절대 분기 명령과 상대 분기 명령 지원

```
LJMP <16bit Address>   ;64KB 범위
AJMP <11bit Address>
    ;현재의 PC 값을 기준으로 2KB 범위 내 위치로 분기
    ;PC의 최상위 5비트와 분기할 주소의 하위 11비트 결합
SJMP <8bit offset>
    ;현 PC 값 기준 -128~+127범위 내
```

# 분기 명령(계속)

* JMP Label

  : 상기 분기 명령들 대용

  : 어셈블러가 위의 명령 중의 하나로 적절히 대체

* JMP @A+DPTR

  : 간접 레지스터 주소지정방식의 무조건 분기 명령

  : A+DPTR의 위치로 분기

```
MOV DPTR,#0000H
MOV A,#00H
JMP @A+DPTR
    ; JMP MONITOR와 동일 기능
```

# 분기 명령(계속)

## * 조건 분기 명령

: 조건을 검사하여 지정된 위치로 분기하는 명령

: 8051에서 조건 분기명령은 모두 상대 분기명령임

```
JC <8bit offset>          ; if C=1, jump
JNC <8bit offset>         ; if C=0, jump
JZ <8bit offset>              ; if A=0, jump
JNZ <8bit offset>             ; if A≠0, jump
```

## * 비교 조건 분기명령( CJNE 명령 )

: 두 값을 비교하여 값이 다르면 지정된 위치로 분기

```
CJNE <dst>,<src>,<8bit offset>
        ; if (dst-src)≠0, jump
```

# 분기 명령(계속)

## * 비트검사 분기명령

: 비트 주소지정된 한 비트를 검사하여 조건을
만족할 때 지정된 위치로 분기

```
JB  <bit address>,<8bit offset>
        ;조건 검사 비트가 1이면, 분기
JNB <bit address>,<8bit offset>
        ;조건 검사 비트가 0이면, 분기
JBC <bit address>,<8bit offset>
        ;조건검사 비트가 1이면 그 비트를 0으로 설정하고
         지정된 위치로 분기
```

# 분기 명령(계속)

예) 조건에 따라 여러 위치로 분기하는 루틴 사례

```
        JB 00H,BR1
        JNB 20H.1,BR2
        JBC ACC.0,BR3
        ....
BR1:    ....
        ....
```

: 첫 명령은 비트 메모리의 첫 주소로 20H.0 비트를 검사하여 1이면 BR1으로 분기

: 두 번째 명령은 20H.1의 비트를 검사하여 0이면 BR2로 분기

: 세 번째 명령은 ACC의 최하위비트를 검사하여 1이면 해당 비트를 0으로 설정한 후 BR3으로 분기

# 반복 처리

## * 기본 형식

: 반복 처리할 루틴의 첫 명령에 레이블 부여

: 그 루틴의 마지막에 분기 명령을 사용

```
Label:      .....
            .....
            .....
            JMP Label
```

: 상기 반복은 무한 반복임

# 반복 처리(계속)

* DJNZ 명령

: Rn, 혹은 direct에 반복될 횟수 지정

```
DJNZ Rn,<8bit offset>        ; if Rn-1≠0, jmp
DJNZ direct,<8bit offset>    ; if direct-1≠0, jmp
```

예) R0에 반복할 횟수를 지정할 경우

```
        MOV R0,#10H
LOOP:   ....
        ....
        ....
        DJNZ R0,LOOP
```

# 반복 처리(계속)

* 무한루프를 회피한 반복처리

( 앞의 DJNZ 사용 않고... )

```
        MOV A,#10H
LOOP:   .....
        .....
        .....
        DEC A
        JNZ LOOP
```

# 참고사항

## * MOD51 파일(ASM51 어셈블러 사용시)

: 각 레지스터의 기호와 실제 어드레스간 사상

: 소스 화일의 선두에 "$MOD51"로 삽입하여 참조 가능

```
      :
      :
IP        DATA     0B8H     ;INTERRUPT PRIORITY
PSW       DATA     0D0H     ;PROGRAM STATUS WORD
PSW1      DATA     0D1H     ;PROGRAM STATUS WORD 1
ACC       DATA     0E0H     ;ACCUMULATOR
B         DATA     0F0H     ;MULTIPLICATION REGISTER
IT0       BIT      088H     ;TCON.0 - EXT. INTERRUPT 0 TYPE
IE0       BIT      089H     ;TCON.1 - EXT. INTERRUPT 0 EDGE FLAG
IT1       BIT      08AH     ;TCON.2 - EXT. INTERRUPT 1 TYPE
IE1       BIT      08BH     ;TCON.3 - EXT. INTERRUPT 1 EDGE FLAG
      :
      :
```

# 블록전송

## * 블록전송

### : 외부 램 상에서 한 블록의 데이터를 다른 곳으로 블록 전송

```
                        1     ;===================================
                        2     ;   LOOP_01.ASM
                        3     ;    Block Transfer
                        4     ;===================================
                        5     $mod51
8000                    6             ORG 8000H
  0000                  7     MONITOR EQU 0000H
                        8             ;
8000                    9     BLOCK_TRANSFER:
8000 7B10              10             MOV R3,#10H      ;반복횟수
8002 7C00              11             MOV R4,#00H      ;offset
8004 7D00              12             MOV R5,#00H      ;읽은 데이터 퇴피
8006                   13     LOOP:
8006 908100            14             MOV DPTR,#SRC    ;주)
8009 E582              15             MOV A,DPL
800B 2C                16             ADD A,R4
```

```
800C F582          17              MOV DPL,A
800E E0            18              MOVX A,@DPTR
800F FD            19              MOV R5,A
                   20
8010 908200        21              MOV DPTR,#DST   ;
8013 E582          22              MOV A,DPL
8015 2C            23              ADD A,R4
8016 F582          24              MOV DPL,A
8018 ED            25              MOV A,R5
8019 F0            26              MOVX @DPTR,A
                   27
801A 0C            28              INC R4
801B DBE9          29              DJNZ R3,LOOP
801D 020000        30              JMP MONITOR
                   31              ;
8100               32              ORG 8100H
8100 23455667      33      SRC:    DB 23H,45H,56H,67H
8104 78891277      34              DB 78H,89H,12H,77H
8108 23455667      35              DB 23H,45H,56H,67H
810C 78891277      36              DB 78H,89H,12H,77H
8200               37              ORG 8200H
8200 00000000      38      DST:    DB 00H,00H,00H,00H
```

```
8204 00000000          39              DB 00H,00H,00H,00H
8208 00000000          40              DB 00H,00H,00H,00H
820C 00000000          41              DB 00H,00H,00H,00H
                       42              END
```

* PC(program counter) 관련...

* 13번 라인의 MOV DPTR,#SRC 명령의 머신코드

  => 8051에서의 바이트 오더링 방식은

  빅 엔디안(Big Endian) 방식을 채택

```
8006 908100          13              MOV DPTR,#SRC          ;(주)
....
```

# 블록합산

## * 블록 내 데이터들의 합 구하기

```
                        1     ;================================
                        2     ;   LOOP_02.ASM
                        3     ;       Addition of block data
                        4     ;================================
8000                    5             ORG 8000H       ;
  0000                  6     ZERO    EQU 00H
  000A                  7     COUNT   EQU 10
                        8             ;
8000 7B0A               9             MOV R3,#COUNT
8002 908100            10             MOV DPTR,#SRC
8005 7900              11             MOV R1,#ZERO    ;결과의 상위바이트
8007 7A00              12             MOV R2,#ZERO    ;결과의 하위바이트
                       13
8009 E0                14     L2:     MOVX A,@DPTR
800A 2A                15             ADD A,R2
800B FA                16             MOV R2,A
800C 400E              17             JC L1
```

```
                        18
800E A3                 19      L3:     INC DPTR
800F DBF8               20              DJNZ R3,L2
                        21
8011 908120             22              MOV DPTR,#RES   ;
8014 E9                 23              MOV A,R1
8015 F0                 24              MOVX @DPTR,A    ;상위바이트 기록
8016 A3                 25              INC DPTR
8017 EA                 26              MOV A,R2
8018 F0                 27              MOVX @DPTR,A    ;하위바위트 기록
                        28
8019 020000             29              JMP 0000H       ; go Monitor
                        30
801C 09                 31      L1:     INC R1          ;상위바이트 증가
801D 80EF               32              SJMP L3
                        33
8100                    34              ORG 8100H
8100 01020304           35      SRC:    DB 1,2,3,4,5,6,7,8,9,10
8104 05060708
8108 090A
                        36
                        37
```

```
                    38
8120                39              ORG 8120H
8120 0000           40      RES:    DB 0,0
                    41              END
```

## * 절대분기 및 상대분기명령

```
8019 020000         29              JMP 0000H       ; PC <- 0000H

800C 400E           17              JC L1           ; PC <- PC + 14

801D 80EF           32              SJMP L3         ; PC <- PC + (-17)
   ; -11H(-17)
        = 0001_0001(17)
             -> 1110_1110(1' s cpl)
                  -> 1110_1111(2' s cpl)
                       = EFH
```

# 순차검색

## * 순차 검색

: 데이터 블록 내에 원하는 데이터가 있는지 검색

: 검색 성공시 해당 데이터의 위치를 기록하고,
그렇지 않은 경우는 FFH로 표시

```
                          1     ;================================
                          2     ;   LOOP_03.ASM
                          3     ;    Sequential Search
                          4     ;================================
                          5     $mod51
8000                      6             ORG 8000H
  0000                    7     MONITOR EQU 0000H
                          8             ;
8000                      9     SEARCH_IN_BLOCK:
8000 7B10                10             MOV R3,#10H
8002 908100              11             MOV DPTR,#SRC
```

```
8005                    12      LOOP:
8005 E0                 13              MOVX A,@DPTR
8006 B41102             14              CJNE A,#11H,NOT_EQUAL   ; 11H 검색
8009 8005               15              JMP FOUND_IT
800B                    16      NOT_EQUAL:
800B A3                 17              INC DPTR
800C DBF7               18              DJNZ R3,LOOP
800E 800E               19              JMP NOT_FOUND_IT
8010                    20      FOUND_IT:
8010 AE83               21              MOV R6,DPH
8012 AF82               22              MOV R7,DPL
8014 908200             23              MOV DPTR,#RES
8017 EE                 24              MOV A,R6
8018 F0                 25              MOVX @DPTR,A
8019 A3                 26              INC DPTR
801A EF                 27              MOV A,R7
801B F0                 28              MOVX @DPTR,A
801C 8006               29              JMP TO_EXIT
801E                    30      NOT_FOUND_IT:
801E 908200             31              MOV DPTR,#RES
8021 74FF               32              MOV A,#0FFH
8023 F0                 33              MOVX @DPTR,A
```

```
8024                    34      TO_EXIT:
8024 020000             35              JMP MONITOR
                        36              ;
8100                    37              ORG 8100H
8100 23455667           38      SRC:    DB 23H,45H,56H,67H
8104 11121315           39              DB 11H,12H,13H,15H
8108 21222325           40              DB 21H,22H,23H,25H
810C 31323335           41              DB 31H,32H,33H,35H
8200                    42              ORG 8200H
8200 0000               43      RES:    DB 00H,00H
                        44              END
```

# 블록정렬(버블 정렬)

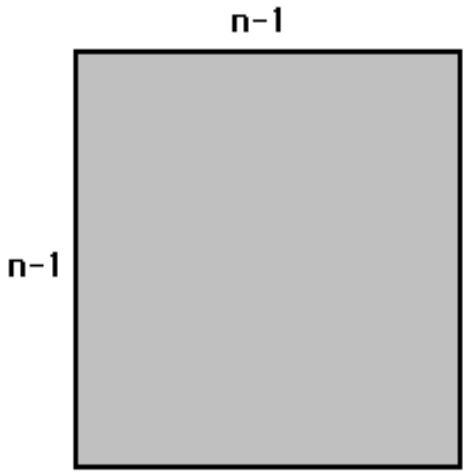
## * 버블 정렬의 개념(Ascending...)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 데이터 | 44 | 55 | 66 | 11 | 22 | 88 | 99 | 33 | 77 |
| 단계1 | 44 | 55 | 66 | | | | | | |
| | | | *11* | *66* | | | | | |
| | | | | *22* | *66* | 88 | 99 | | |
| | | | | | | | *33* | *99* | |
| | | | | | | | | *77* | *99* |
| | 44 | 55 | 11 | 22 | 66 | 88 | 33 | 77 | 99 |
| 단계2 | 44 | 11 | 55 | 22 | 66 | 33 | 77 | 88 | 99 |
| 단계3 | 11 | 44 | 22 | 55 | 33 | 66 | 77 | 88 | 99 |
| 단계4 | 11 | 22 | 44 | 33 | 55 | 66 | 77 | 88 | 99 |
| 단계5 | 11 | 22 | 33 | 44 | 55 | 66 | 77 | 88 | 99 |
| 단계6 | 11 | 22 | 33 | 44 | 55 | 66 | 77 | 88 | 99 |
| 단계7 | 11 | 22 | 33 | 44 | 55 | 66 | 77 | 88 | 99 |
| 단계8 | 11 | 22 | 33 | 44 | 55 | 66 | 77 | 88 | 99 |

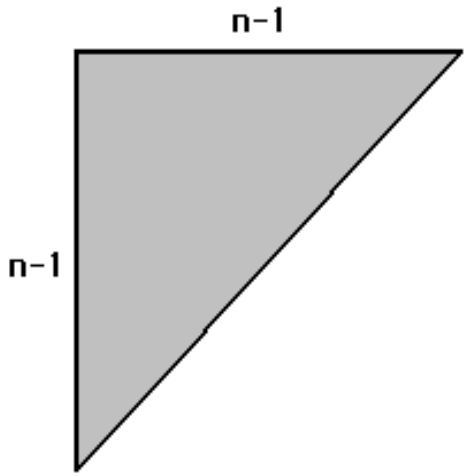
# 블록정렬(버블 정렬)

* 최적화

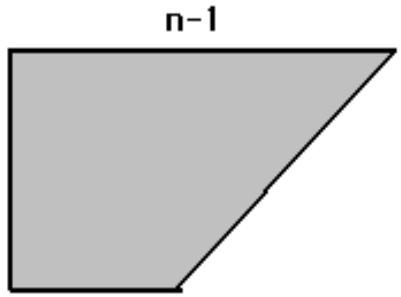
: 블록 데이터가 n개라 가정




최적화비고려




기정렬회피




정렬종료후
단계회피

# 블록정렬(버블 정렬)

## * 버블 정렬(오름차순) 프로그램

```
                        1     ;===================================
                        2     ;   LOOP_04.ASM
                        3     ;    Bubble Sort ... Ascending
                        4     ;===================================
                        5     $mod51
8000                    6             ORG 8000H
  0000                  7     MONITOR EQU 0000H
                        8             ;
8000                    9     SORT_IN_BLOCK:
8000 7F0F              10             MOV R7,#0FH          ;#Step
8002                   11     STEP:
8002 7B0F              12             MOV R3,#0FH          ;#Compare
8004 908100            13             MOV DPTR,#SRC
8007                   14     LOOP:
8007 E0                15             MOVX A,@DPTR
8008 FC                16             MOV R4,A             ;i
8009 A3                17             INC DPTR
```

```
800A E0          18          MOVX A,@DPTR        ;i+1
800B B50400      19          CJNE A,04H,NEXT
800E 4007        20  NEXT:   JC EXCHANGE
8010 DBF5        21  AAA:    DJNZ R3,LOOP
8012 DFEE        22          DJNZ R7,STEP
8014 020000      23          JMP MONITOR
                 24          ;
8017             25  EXCHANGE:
8017 1582        26          DEC DPL
8019 F0          27          MOVX @DPTR,A
801A EC          28          MOV A,R4
801B A3          29          INC DPTR
801C F0          30          MOVX @DPTR,A
801D 80F1        31          JMP AAA
                 32          ;
8100             33          ORG 8100H
8100 45552382    34  SRC:    DB 45H,55H,23H,82H
8104 12668882    35          DB 12H,66H,88H,82H
8108 39211177    36          DB 39H,21H,11H,77H
810C 43895634    37          DB 43H,89H,56H,34H
                 38          END
```

# 실습과제

## [실습1] 다중바이트 연산

: 다중 바이트 가산 혹은 감산

: 데이터는 외부메모리에 블록형태로 위치

: 반복처리 활용

## [실습2] 블록 설정 및 전송

: 외부 데이터 메모리 영역 -> 내부 램 영역 ->
또 다른 외부 데이터 메모리 영역으로 블록 전송

# 실습과제(계속)

## [실습3] 순차 검색

: 본문 예제 실습

## [실습4] 최대값/최소값 구하기

: 블록 데이터에서 최대값, 혹은 최소값

## [실습5] 선택정렬

: 최대값/최소값 구하기 활용

# 실습과제(계속)

## [실습6] 버블 정렬

: 예제의 버블정렬 프로그램을 내림차순 정렬되도록 수정

: 최적화 문제 고려

(각 경우의 비교연산 회수를 계수하여 연산량을 비교)