제12강

C51 프로그래밍 I

추가 데이터형

메모리 형

일반포인터 및 메모리형 지정 포인터

실습과제

Ref.) Chapter 12, 13, C51_01.zip 다운로드

C51

* 데이터 형(표준C+추가) : 'reg51.h' 헤더파일 참조

■ bit 형 : 비트주소지정 영역, 선언과 초기화 분리할 것!!

```
bit first, third;       // 변수 선언
first = 0x00;           // first에 비트주소 0x00(20H.0) 초기화
third = 0x24^1;         // third에 비트주소 24H.1로 초기화
```

■ sfr 형 : SFR

```
sfr P1 = 0x90;          // 90H의 SFR를 P1으로 선언
sfr TMOD = 0x89;        // 89H의 SFR를 TMOD로 선언
```

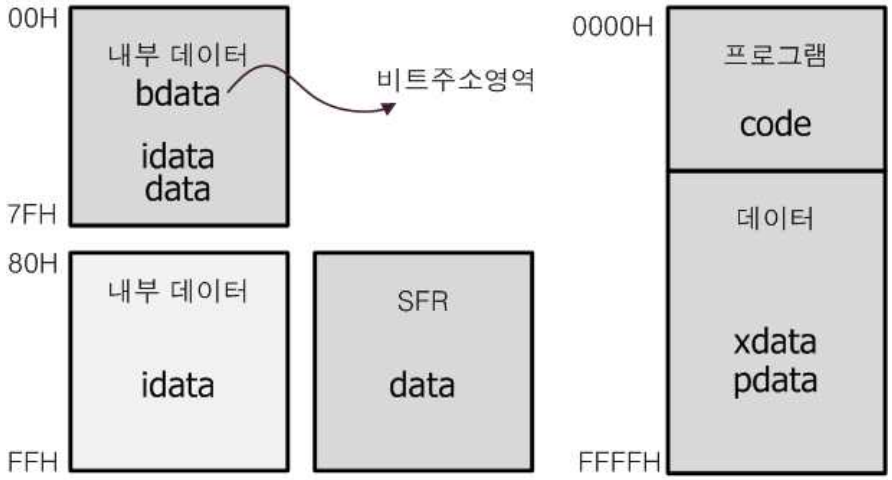
■ sbit 형 : SFR중 비트주소가능 영역, bit형 대용가능

```
sbit EA = 0xAF;         // IE.7을 EA로 선언
sbit CY = 0xD0^7        // PSW.7(D0H.7)을 CY로 선언
                        // PSW^7, 0xD0^7, 0xD7은 동일함
```

C51(계속)

* 메모리 영역(형)

메모리영역	설 명	주소범위
data	직접주소지정 가능한 내부 데이터 메모리	0x00 ~ 0xFF
bdata	비트주소지정 가능한 내부 데이터 메모리(바이트주소 20H~2FH)	0x00 ~ 0x7F
idata	간접주소지정 가능한 내부 데이터 메모리	0x00 ~ 0xFF
pdata	MOVX @Ri 명령으로 접근가능한 외부 데이터 메모리	0xxx00 ~ 0xxxFF(256B)
xdata	MOVX @DPTR 명령으로 접근가능한 외부 데이터 메모리	0x0000 ~ 0xFFFF
code	MOVC @A+DPTR 명령으로 접근가능한 프로그램 메모리	0x0000 ~ 0xFFFF



C51(계속)

* 메모리 영역 선언 및 사용 예

1) 변수

```
char sw;
    // 메모리영역 명시적 선언 없을 때, 내부램 영역에 할당
    // char data sw; 와 동일
char xdata tmp;
    // 메모리영역 명시적 선언, 외부메모리 공간에 할당
```

2) 포인터

```
char xdata * pLED = 0x8000;
    // char xdata * data pLED = 0x8000; 와 동일
char xdata * pSW = 0x8010;
    // char xdata * data pSW = 0x8010; 와 동일

*pLED = 0xFF;          // 외부데이터메모리 8000H에 FFH 쓰기
sw = *pSW;             // 외부데이터메모리 8010H의 값을 읽기
```

C51(계속)

* 일반 포인터(generic pointer)

: 항상 3바이트를 이용하여 저장, 메모리형은 실행시 결정

메모리 형	idata	xdata (default)	pdata	data, bdata	code
상수값	0x01	0x02	0x03	0x04	0x05

예) xdata 메모리의 0x8100 주소에 대한 포인터 구조

Mem Type	higher Byte	lower Byte
0x02	0x81	0x00

선언 예)

```
char * strptr;                  // generic ptr
int * numptr;                   // generic ptr
char * xdata strptr;            // generic ptr stored in xdata
int * data numptr;              // generic ptr stored in data
```

C51(계속)

* 메모리 지정 포인터(memory specific pointer)

: 포인터구조에서 메모리 형 불필요

=> 포인터 구조는 1 혹은 2바이트로 간결

: 메모리 영역이 컴파일시 결정

=> 고속 접근가능하나 유연성 결여

: 메모리형 지정 포인터 선언형식

데이터형 [데이터메모리형] * [포인터메모리형] 포인터명

// 가리키는 데이터의 메모리영역, 포인터자체가 위치할 메모리영역

선언 예)

```
char data * str;              // ptr to string in data
int xdata * numtab;           // ptr to int(s) in xdata
char data * xdata str;        // ptr in xdata to data char
int xdata * data numtab;      // ptr in data to xdata int
```

C51 컴파일

* 다운로드(C51_01.zip) 및 풀기

  : .\BASIC의 basic.Uv2 프로젝트 열기

* 스타트업 코드 추가

  : "STARTUP.A51" 파일 참조

  : KUT51 실습보드에 맞게 수정(125번 라인)

```
        :
        PUBLIC    ?C_STARTUP

        CSEG      AT   8000H          // for KUT51 board
?C_STARTUP:LJMP       STARTUP1
        :
```

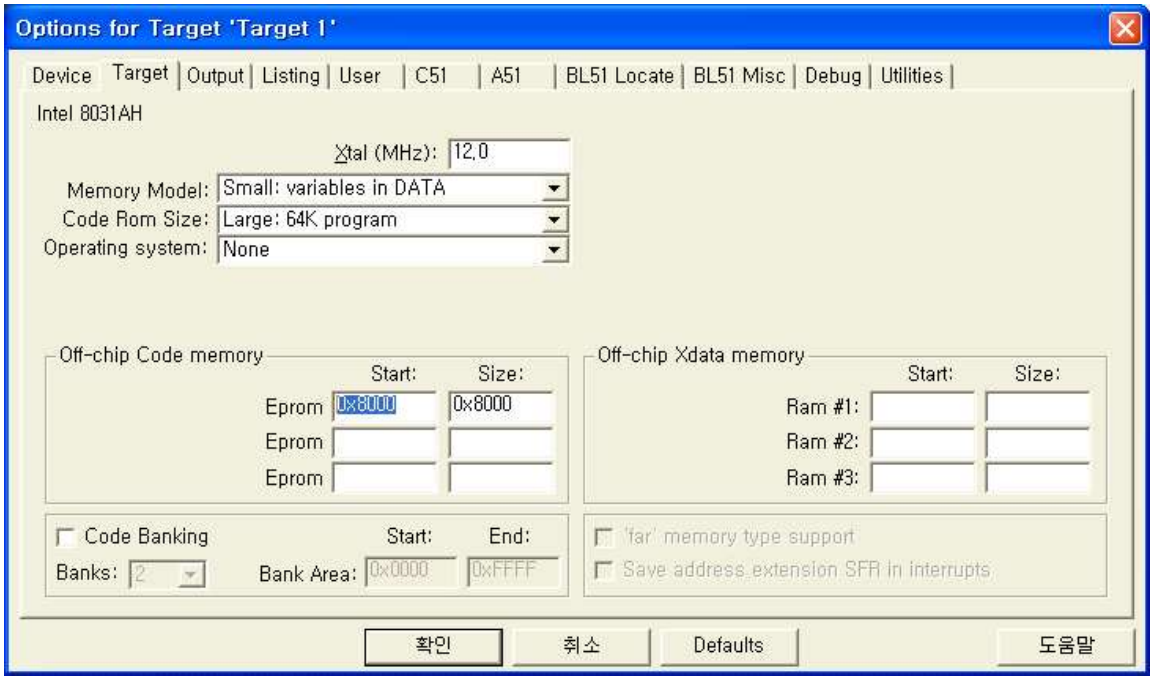
C51 컴파일(계속)

* "Target" 탭에서

: 메모리 모델 3종 (small, compact, large)

: Memory Model 항목 : small

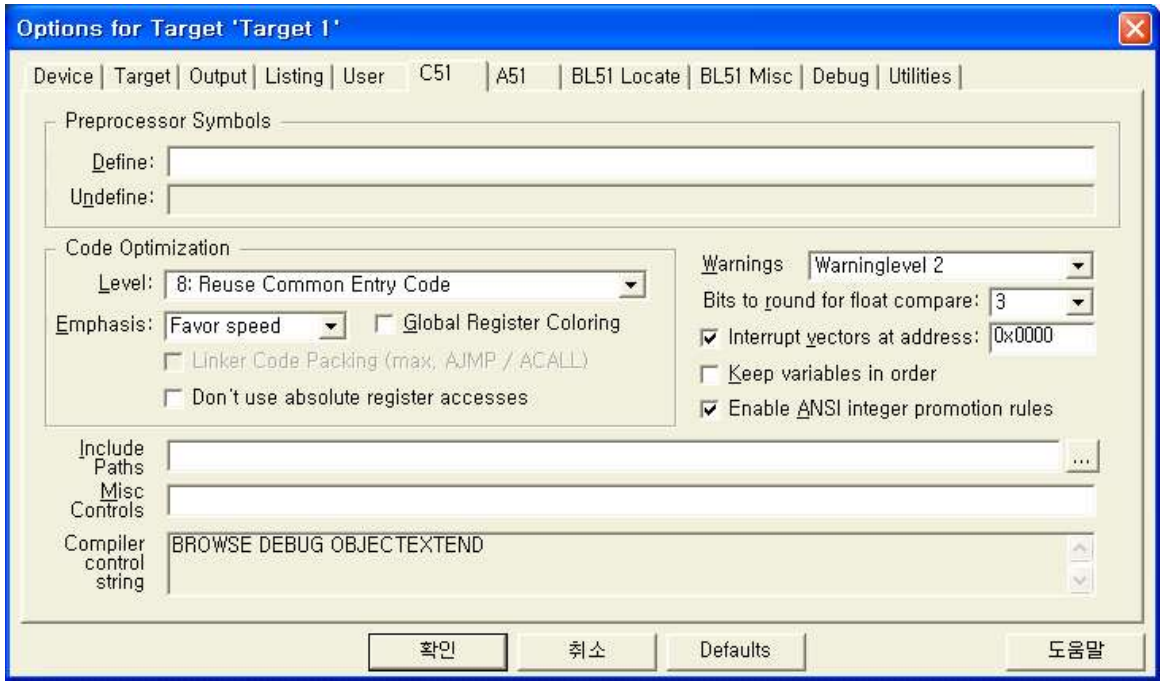
: 실습관련 예제는 small 모델에서 진행함



C51 컴파일(계속)

* "C51"탭에서

: 인터럽트 벡터의 시작주소 설정부분인 'Interrupt vectors at address'에 0x0000으로 설정되어 있고, 체크되었는지를 확인(특히, 인터럽트 취급 경우)



C51 컴파일(계속)

* C51 소스 작성시

: 헤더파일 포함토록...

```
#include <reg51.h>
```

```
/*-----------------------------
REG51.H
-----------------------------*/
/*  BYTE Register  */
sfr P0    = 0x80;
sfr P1    = 0x90;
sfr P2    = 0xA0;
sfr P3    = 0xB0;
sfr PSW   = 0xD0;
/*  BIT Register  */
/*  PSW    */
sbit CY    = 0xD7;
sbit AC    = 0xD6;
sbit F0    = 0xD5;
sbit RS1   = 0xD4;
sbit RS0   = 0xD3;
     :
```

예제실습

* C프로그램 실행후, 모니터프로그램의 실행을 위해 아래의 어셈블리 소스 활용

```
;=====================================
; monitor.ASM
;=====================================
gotoMonitor     SEGMENT CODE
    PUBLIC    _gotoMonitor          ; public
    RSEG      gotoMonitor           ; relocatable

_gotoMonitor:
    JMP 0000H             ; 실행의 흐름을 모니터 프로그램으로
    RET

    END
```

예제실습(계속)

[실습*] 메모리 영역 관련 예제

```
#include <reg51.h>

extern void gotoMonitor(void);

int main(void) {
    char b, c;                          // 08H부터 자동할당
    char xdata *pa = 0x8100;            // 외부메모리
    char xdata *pb = 0x8110;
    char xdata *pc = 0x8120;

    char data *pp = 0x08;               // 내부램 08H

    *pa = 0x99;                         // 8100H <= 0x99
    b = *pa;                            // 08H주소 <= 0x99
    *pb = b;                            // 8110H <= 08주소내용

    *pc = &b;                           // 8120H <= b의 주소
    *(pc+1) = &c;                       // 8121H <= c의 주소
    *(pc+2) = *pp;                      // 8122H <= 08번지내 정보

    gotoMonitor();                      // 모니터프로그램으로
}
```

예제실습(계속)

[실습1] sfr & sbit 형

: sbit 형 변수 선언 및 초기화 병행

```
//================================
// C51_01.C
//      data type : sfr and sbit
//      P1_L : LED module
//================================
#include <reg51.h>

// sfr 형 및 sbit 형 선언
sfr PORT = 0x90;                        // 절대 주소 설정으로 선언
sbit p0 = PORT^0;

// 특정 비트 위치의 접근을 위한 sbit 선언
unsigned char bdata bPat;               // bdata 메모리 형으로 선언
sbit bPat0 = bPat^0;                    // sbit 형 선언 및 초기화 병행
sbit bPat1 = bPat^1;
sbit bPat2 = bPat^2;
sbit bPat3 = bPat^3;

void delay(unsigned int t) {
   while(t--);
```

```
}

void main(void){
   unsigned char pat;

   bPat0 = 0;                          // 비트별 값 설정, 비트 단위 접근
   bPat1 = 1;
   bPat2 = bPat0;
   bPat3 = bPat1;

   pat = bPat;                         // 바이트 단위 접근
   while(1) {
        PORT = pat;                    // PORT 변수는 P1을 의미
        delay(50000);
        pat = ~pat;
   }

}
```

예제실습(계속)

[실습2] sfr & bit 형

: [실습1]의 수정본 (bit형 선언 및 대입)

: bit 형의 선언 및 초기화 분리

```
//================================
// C51_02.C
//      data type : sfr and bit
//      P1_L : LED module
//================================
#include <reg51.h>

// sfr 형 및 sbit 형 선언
sfr PORT = 0x90;                        // 절대 주소 설정으로 선언
sbit p0 = PORT^0;

// 특정 비트 위치의 접근을 위한 sbit 선언
unsigned char bdata bPat;               // bdata 메모리 형으로 선언
bit bPat0;                              // bit 형 선언
bit bPat1;
bit bPat2;
bit bPat3;
```

```
void delay(unsigned int t) {
   while(t--);
}

void main(void){
   unsigned char pat;

   bPat0 = bPat^0;                    // bit형 변수의 대입
   bPat1 = bPat^1;
   bPat2 = bPat^2;
   bPat3 = bPat^3;

   bPat0 = 0;                         // 비트별 값 설정, 비트 단위 접근
   bPat1 = 1;
   bPat2 = bPat0;
   bPat3 = bPat1;

   pat = bPat;                        // 바이트 단위 접근
   while(1) {
        PORT = pat;                   // PORT 변수는 P1을 의미
        delay(50000);
        pat = ~pat;
   }
}
```

예제실습(계속)

[실습3] 메모리형 지정 변수

```
//================================
// C51_03.C
//      memory type specific variable
//================================
#include <reg51.h>

extern void gotoMonitor(void);

void main(void){
    unsigned char xdata * resPtr = 0x8200;      // 결과 저장 시작 주소

    unsigned char data d1 = 0x11;      // data영역내 임의주소에 값 0x11
    unsigned char data d2 = 0x22;
    unsigned char data d3 = 0x33;
    unsigned char data d4 = 0x44;
    unsigned char idata d5 = 0x55;     // idata 영역내 임의주소에 값 0x55
    unsigned char bdata d6 = 0x66;     // bdata 영역내 임의주소에 값 0x66
    unsigned char xdata d7 = 0x77;     // xdata 영역내 임의주소에 값 0x77
    unsigned char code d8;             // code 영역내 선언만

    *(resPtr++) = d1;                  // 외부 메모리에 저장
    *(resPtr++) = d2;
```

```
    *(resPtr++) = d3;
    *(resPtr++) = d4;
    *(resPtr++) = d5;
    *(resPtr++) = d6;
    *(resPtr++) = d7;
    *(resPtr++) = d8;

    *(resPtr++) = d1 + d2;              // 연산
    *(resPtr++) = d5 + d6;

    *(resPtr++) = &d7;                  // 변수가 저장된 위치

    gotoMonitor();                      // 모니터프로그램으로
}
```

예제실습(계속)

[실습4] 메모리형 지정 포인터

```
//================================
// C51_04.C
//      memory type specific pointer
//================================
#include <reg51.h>

extern void gotoMonitor(void);

void main(void){
    unsigned char xdata * resPtr = 0x8200; // 결과 시작 위치

    unsigned char data * ptr1 = 0x40;      // ptr to 40H in data
    unsigned char data * ptr2 = 0x41;
    unsigned char data * ptr3 = 0x42;
    unsigned char data * ptr4 = 0x43;
    unsigned char idata * ptr5 = 0x50;     // ptr to 50H in idata
    unsigned char bdata * ptr6 = 0x20;     // ptr to 20H in bdata
    unsigned char xdata * ptr7 = 0x8240;   // ptr to 8240H in xdata
    unsigned char code * ptr8 = 0x0003;    // ptr to 0003H in code

    *ptr1 = 0x11;                          // (40H) <- 0x11
    *ptr2 = 0x22;
```

```
    *ptr3 = 0x33;
    *ptr4 = 0x44;
    *ptr5 = 0x55;                       // (50H) <- 0x55
    *ptr6 = 0x66;                       // (20H)
    *ptr7 = 0x77;                       // (8240H)
    *ptr8;                              // can not write

    *(resPtr++) = *ptr1;                // (8200H) <- (40h), 0x11
    *(resPtr++) = *ptr2;
    *(resPtr++) = *ptr3;
    *(resPtr++) = *ptr4;
    *(resPtr++) = *ptr5;
    *(resPtr++) = *ptr6;
    *(resPtr++) = *ptr7;
    *(resPtr++) = *ptr8;

    *(resPtr++) = *ptr1 + *ptr2;        // 연산
    *(resPtr++) = *ptr5 + *ptr6;

    gotoMonitor();                      // 모니터프로그램으로

}
```

예제실습(계속)

[실습5] 데이터 전송

```
//============================
// C51_05.C
//      data transfer
//============================
#include <reg51.h>

extern void gotoMonitor(void);

void main(void) {
   unsigned char xdata *src = 0x8100;
   unsigned char data *tmp = 0x50;
   unsigned char xdata *dst = 0x8120;

   // a data transfer
   *src = 0xF0;                    // 데이터 F0H
   *tmp = *src;
   *dst = *tmp;

    gotoMonitor();                 // 모니터프로그램으로
}
```

C51(계속)

* 외부메모리 및 입출력 디바이스 접근(4가지)

1) 메모리영역지정 포인터에 의한 접근

```
#define  LED (*(unsigned char xdata *)0x8000)
#define  KBD (*(unsigned char xdata *)0x8100)

char sw;

sw = KBD;                    // Read
LED = 0xab;                  // Write
      ⋮
```

C51(계속)

2) 일반 포인터 표현에 의한 접근

: 메모리형을 포함한 포인터 구조

```
#define CODE_Mem ((char *) 0x050000)
                                    // Code Mem 0000H
#define XDATA_Mem ((char *) 0x028000)
                                    // Ext Data Mem 8000H
#define XDATA_PTR (*(char *) 0x028100)
                                    // Ext Data Mem 8100H

unsigned char tmp;

XDATA_PTR = 50;                     // 8100H 번지에 50을 저장
XDATA_Mem = tmp;
tmp = CODE_Mem;
```

C51(계속)

3) 매크로에 의한 접근

: 'absacc.h' 파일에 정의된 매크로 kBYTE, kWORD 이용

: 메모리형에 따라,

k는 C(code), D(data), P(pdata), X(xdata)로 대체

```
#include <absacc.h>
#define KBD 0x8100

unsigned char val8;
unsigned int val16;

val8 = XBYTE[KBD];              // Read 8100H, xdata영역
XBYTE[0x8200] = 0xFF;           // Write 8200H, xdata영역

XWORD[0x8000] = 0xFFFF;         // Write 2 byte, 8000H, 8001H
val16 = XWORD[0x8010];          // Read 2 byte, 8010H, 8011H
```

C51(계속)

4) _at_ 키워드에 의한 접근

: 메모리의 고정된 주소를 지정하기 위해 필요한 키워드

```
unsigned char xdata LCD _at_ 0x8000;
unsigned char xdata KBD _at_ 0x8100;
unsigned char tmp;

tmp = KBD;                    // Read
LCD = 0xF0;                   // Write
      ⋮
      ⋮
```

예제실습(계속)

[실습6] 외부메모리/입출력장치 접근

1) 메모리지정 포인터에 의해

```
//=============================
// C51_06_1.C
//      data transfer ... using memory specific pointer
//=============================
#include <reg51.h>

extern void gotoMonitor(void);

// using memory specific pointer
#define SRC  (* (unsigned char xdata *)0x8101)
#define DST  (* (unsigned char xdata *)0x8141)

void main(void) {
   unsigned char data *tmp = 0x50;

   // a data transfer
   SRC = 0x99;              // 데이터 99H
   *tmp = SRC;
   DST = *tmp;
```

```
    gotoMonitor();                  // 모니터프로그램으로
}
```

2) 일반 포인터에 의해

```
//=============================
// C51_06_2.C
//      data transfer ... using generic pointer
//=============================
#include <reg51.h>

extern void gotoMonitor(void);

// using generic pointer
#define SRC       ((unsigned char *)0x028102)   // xdata 8102H
#define DST_PTR   (*(unsigned char *)0x028142)  // ptr xdata 8142H

void main(void) {
   unsigned char data *tmp = 0x50;

   // a data transfer
   *SRC = 0x99;              // 데이터 99H
   *tmp = *SRC;
   DST_PTR = *tmp;

   gotoMonitor();            // 모니터프로그램으로
}
```

3) 매크로에 의해

```
//=============================
// C51_06_3.C
//      data transfer ... using Macro
//=============================
#include <reg51.h>
#include <absacc.h>                 // include macro header file

extern void gotoMonitor(void);

#define      SRC  0x8103            // xdata memory
#define      TMP  0x50              // data memory 50H
#define      DST  0x8143            // xdata memory

void main(void) {
   // a data transfer
   XBYTE[SRC] = 0x77;               // xdata Macro XBYTE[]
   DBYTE[TMP] = XBYTE[SRC];         // data Macro DBYTE[]
   XBYTE[DST] = DBYTE[TMP];

   gotoMonitor();              // 모니터프로그램으로
}
```

4) _at_ 키워드에 의해

```
//=============================
// C51_06_4.C
//      data transfer ... using _at_ keyword
//=============================
#include <reg51.h>

extern void gotoMonitor(void);

// using _at_ keyword
unsigned char xdata    SRC  _at_ 0x8104;        // memory
unsigned char xdata    DST  _at_ 0x8144;        // memory

void main(void) {
   unsigned char data *tmp = 0x50;

   // a data transfer
   SRC = 0x99;                                 // 데이터 99H
   *tmp = SRC;
   DST = *tmp;

   gotoMonitor();            // 모니터프로그램으로
}
```

예제실습(계속)

[실습7] 블록데이터 설정 및 전송

: 내부 데이터메모리에서 외부 데이터메모리로 전송

```
//=============================
// C51_07.C
//      block transfer
//=============================
#include <reg51.h>

extern void gotoMonitor(void);

void main(void) {
   unsigned char blk[16];          // 내부 데이터 메모리에 위치
   unsigned char xdata *dst = 0x8200;
   unsigned char i;

   // arr transfer
   for(i=0; i<16; i++)             // data set
        blk[i] = i*2;
   for(i=0; i<16; i++)             // data transfer
        *(dst+i) = blk[i];
```

```
    gotoMonitor();              // 모니터프로그램으로
}
```

예제실습(계속)

[실습8] 비트단위 전송

: 비트단위 역순 재조합

```
//=============================
// C51_08.C
//      bit transfer
//=============================
#include <reg51.h>

extern void gotoMonitor(void);

unsigned char bdata src;
sbit src0 = src^0;
sbit src1 = src^1;
sbit src2 = src^2;
sbit src3 = src^3;
sbit src4 = src^4;
sbit src5 = src^5;
sbit src6 = src^6;
sbit src7 = src^7;

unsigned char bdata dst;
sbit dst0 = dst^0;
```

```
sbit dst1 = dst^1;
sbit dst2 = dst^2;
sbit dst3 = dst^3;
sbit dst4 = dst^4;
sbit dst5 = dst^5;
sbit dst6 = dst^6;
sbit dst7 = dst^7;

void main(void) {
   unsigned char xdata *res = 0x8100;

   src = 0x69;
   *(res++) = src;
   dst0 = src7;                    // bit transfer by reverse
   dst1 = src6;
   dst2 = src5;
   dst3 = src4;
   dst4 = src3;
   dst5 = src2;
   dst6 = src1;
   dst7 = src0;
   *(res++) = dst;

   *(res++) = &src;                // get address of src
   *(res++) = &dst;

   gotoMonitor();                  // 모니터프로그램으로
```

```
}
```

실습과제

[과제1] 블록 설정 및 전송

: 외부메모리 -> 내부메모리 -> 외부메모리

: 외부메모리 접근방식(4가지)으로 각각

[과제2] 순차검색

: 블록내에서 원하는 데이터의 탐색

[과제3] 버블정렬

: 블록내 데이터의 오름차순 및 내림차순 정렬