İLERI MIKRODENETLEYICILER

Ege Üniversitesi Ege MYO
Mekatronik Programı

BÖLÜM 1

Embedded C, C51

Temel Veri Tipleri

Veri tipi (Data Type)	Bit	Bayt	Değer
bit	1		0, 1
char	8	1	-128 , +127
unsigned char	8	1	0, 255
enum	16	2	-32768 , +32767
short	16	2	-32768, +32767
unsigned short	16	2	0, 65535
int	16	2	-32768, +32767
unsigned int	16	2	0, 65535
long	32	4	-2147483648, 2147483647
unsigned long	32	4	0, 4294967295
float	32	4	±1.175494E-38, ±3.402823E+38
sbit	1		0, 1
sfr	8	1	0, 255
sfr16	16	2	0, 65535

char

- C dilinde en yaygın kulanılan veri tipidir.
  - Char rakam ve harfleri belirtmek amacıyla kullanılır, embeded C'de 8 bitlik sayılar için de kullanılmıştır. Karakterleri ASCII formatında tanımlar.
    - 'A' Büyükharf 'A' karakteridir.
    - '\n' yeni satır.
    - '\t' tab.
    - '\0' "null" karakteri.
    - '\012' octal 12 sayısı, onlu 10'a eşittir.

int, long

- C dilinin en temel veri tipidir.
  - İşaretli ve işaretsiz 16 bit tam sayıları tanımlar
  - 16 bitin yeterli olmadığı durumlarda tam sayılar, 32 bit olan long olarak tanımlanır

```
int veri;
veri=124;
unsigned int veri1;
veri1=124;
long uzun_veri;
unsigned long uzun_veri1;
```

float, double

- Kayan noktalı veri türünü ifade etmek için kullanılır.
  - C dilinde yer alan BCD float, double, long double veri türleri de embedded C'de veri türleri de desteklenir.
  - Float veri türleri bellekte 4 ile 7 bayt aralında yer kaplar, kulllanırken dikkat!

```
float kesirli_sayi;
kesirli_sayi=1.273;
```

bit, sbit

- Standart C'de bulunmayan bu veri türü embedded C'de eklenmiştir.
- bit veri türü 8051'in iç veri belleğinde yer alan bit adreslenebilir bölgeyi kullanmayı amaçlar.
- sbit veri türü ise SFR bölgesindeki bit adresli yazaçların bitlerini kullanmayı amaçlar.

```
bit led;
led=0;
sbit c=PSW^7;
sbit EA=0xAF;
```

sfr, sfr16

- sfr SFR bellek bölgesine erişmek için kullanılan 8 bitlik veri türüdür.
- sfr16 16 bitlik SFR yazaçlarına erişmek için kullanılan 16 bitlik veri türüdür.

```
sfr P0=0x80;
sfr P1=0x90;
sfr16 T2=0xCC;
```

C51'in Bellek Kullanımı

- Program (Code) belleği
  - Sadece okunabilir tür bellektir.
  - Mikrodenetleyicinin yapısına göre iç veya dış olabilir.
  - Maksimum 64 Kbayt olabilir, sadece bu bellekten kod yürütülebilir.
  - Program belleğine code tanımlaması ile erişilebilir.

```
code int carpan1=65;
char code 7_segment [16]={0x3f,0x06,……};
```

- İç veri belleği
  - Mikrodenetleyici içerisinde yer alan 256 baytlık RAM bellektir
  - Alt 128 bayt, üst 128 bayt ve bit adreslenebilir olmak üzere 3 bölgeye ayrılmıştır.
  - Alt 128 bayt data, üst 128 bayt idata, bit adreslenebilir bölge bdata olarak adlandırılır.

```
char data yol;
int idata hiz;
bit bdata sifir_bayrak;
char bdata led_dizi [8];
```

C51'in Bellek Kullanımı (Devamı)

- Dış Veri Belleği (external data)
  - 8051'in MOVX komutu ile eriştiği dış RAM bellektir.
  - MOVX A,@Ri komutu ile erişilen sayfalı yapıya sahip dış veri bellleği pdata olarak tanımlanır.
  - MOVX A,@DPTR komutu ile erişilen veri bellleği xdata olarak tanımlanır.

```
float xdata sayi1;
sayi1=13.25;
int pdata sayi2;
sayi2=1123;
```

C51'in Bellek Modelleri

- C51'in belleği nasıl kullanması gerektiğini ona söylemek gerekir.
- SMALL, COMPACT ve LARGE modellerinden birini seçebiliriz, seçim yapılma ise SMALL default olarak seçilir.
- SMALL modelde değişkenler iç veri belleğinin alt 128 baytlık kısmında saklanır. Bu bölgeye erişim hızlıdır ve üretilen kod az olur. Sakıncası tanımlanan değişken sayısı fazla olduğunda bellek dolar ve programlar beklendiği gibi çalışmaz. Bellek dolduğunda compiler uyarmaz!
- COMPACT modelde bütün değişkenler dış veri belleğinin bir sayfasında saklanır. (Pdata olarak tanımlanan bölgede). Veri aktarımı MOVX A,@Ri komutu ile olduğu için boyutu en fazla 256 olabilir. Bu modelde program yavaş çalışır.
- LARGE modelinde tüm değişkenler 64Kbaytlık dış veri belleğinde saklanır. Bu modelde dış veri belleğine MOVX A@DPTR komutuyla erişilir. Programınızda çok fazla değişken tanımlamanız gerekiyorsa bu modeli seçmeniz gerekir.

C51'de Kesme fonksiyonun tanımlanması

- 8051'de bulunan donanımsal kelmeleri tanımlamak amacıyla kesme fonksiyonu yazılmalıdır.
- C51 her kesme kaynağına bir numara verir maksimum 32 adet (0-31) kesme kaynağını denetleyebilir.
- Hangi numara hangi kesmeye ait olduğunu standart 8051'de tabloda verilmiştir. Türev ürünlerde üretici veri yapraklarına bakılmalıdır.

Kesme Kaynağı	KVA	C51'deki NO
Dış kesme 0	0003H	0
Z/S 0	000BH	1
Dış kesme 1	0013H	2
Z/S 1	001BH	3
Seri Port	0023H	4
Z/S 2	002BH	5

Diğer Kesme Kaynakları

Interrupt Number	Address
0	0003h
1	000Bh
2	0013h
3	001Bh
4	0023h
5	002Bh
6	0033h
7	003Bh
8	0043h
9	004Bh
10	0053h
11	005Bh
12	0063h
13	006Bh
14	0073h
15	007Bh

Interrupt Number	Address
16	0083h
17	008Bh
18	0093h
19	009Bh
20	00A3h
21	00ABh
22	00B3h
23	00BBh
24	00C3h
25	00CBh
26	00D3h
27	00DBh
28	00E3h
29	00EBh
30	00F3h
31	00FBh

C51 Program Yapısı

```
 /*****************************************/
/* uygulama adı:                         */
/* Dosya Adı:                            */
/* tarih:                                */
/* kısaca açıklama                       */
/*****************************************/
#include <89s52.h>
#include <stdio.h>

#define
sfr
sbit
int

Void kesme 0 () interrupt 0
{

}
```

Kütüphane tanımı

Global Tanımlamalar

Kesme fonksiyonu

```
Void topla ()
{

}

main ()
{

}
end
```

Fonksiyon (altprogram)
Sayısı sınırsız

Ana program

C'de Denetim Yapıları

- if Deyimi
  - if deyimi tek başına ve else ile birlikte kullanılır.
  - Test edilen koşul doğru ise belirtilen işlemi yapar değilse yapmaz.

Örnek:

```
if (x>z)                    //x z'den büyük mü?
      z=0;                  //evet z=0 yap
      x=0;                  // hayır x=0 yap
Veya
if (x>z)                    //x z'den büyük mü?
{
      z=0;                   //evet z=0 yap
}
      x=0;                   // hayır x=0 yap
```

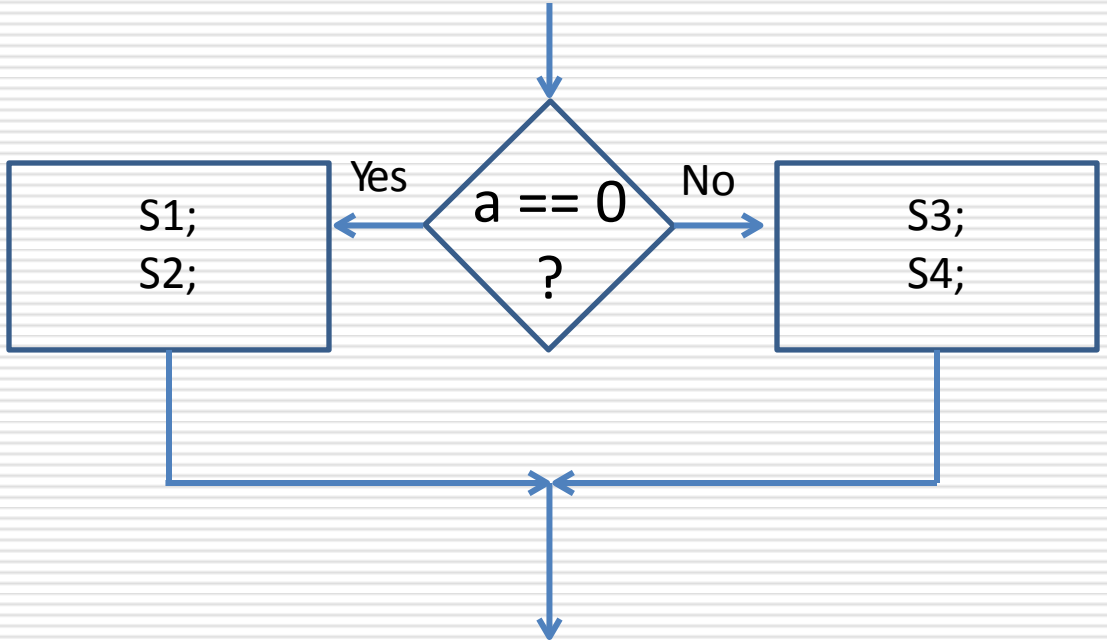
else ile birlikte kullanılabilir;

```
if (x>z)                    //x z'den büyük mü?
      z=0;                  //evet z=0 yap
else
      x=0;                  // hayır x=0 yap
```

IF-THEN-ELSE Yapısı

- Test sonucu doğru ise farklı yanlış ise farklı işlemleri yapar.

```
if (a == 0)
   {
     statement s1;
     statement s2;
   }
else
  {
    statement s3;
    statement s4:
  }
```



Çoklu ELSE-IF Denetim yapısı

- Belirlenen sıraile birden fazla denetim yapar.

```
if (n == 1)
        işlem 1;    //eğer n == 1 ise yap
  else if (n == 2)
        işlem 2;   //eğer n == 2 ise yap
  else if (n == 3)
        işlem 3;  // eğer n == 3 ise yap
else
    statement4;  //yukarıdakilerden hiç biri değilse yap
```

SWITCH denetimi

- ELSE-IF yapısının yaptığı benzer denetimi daha düzenli bir şekilde yapar.

```
switch ( n)          //n test edilen değişken
{
    case 0:
        statement1;  //n == 0 ise yap
    case 1:
        statement2;  // n == 1 ise yap
    case 2:
        statement3;  // n == 2 ise yap
    default:
        statement4;  //diğerlerinden hiç biri değilse ise yap
}
```

C'de Denetim Yapıları

- switch Deyimi
  - switch deyimi birden fazla test edilmesi gereken durum olduğunda ve her durumda farklı iş yapılacak sa kullanılır.
  - Test edilen koşul doğru ise belirtilen işlemi yapar değilse yapmaz.

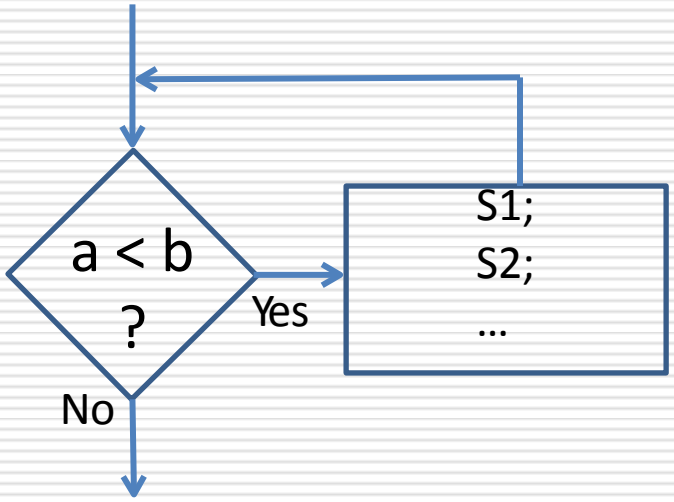
Örnek:

```
M3 = get_key();
switch(M3)
{
 case 0x05:     key_value=2;          break;
 case 0x09:     key_value=3; break;
 case 0x0d:     key_value=4; break;
 case 0x06:     key_value=5; break;
 case 0x0a:     key_value=6; break;
 case 0x0e:     key_value=7; break;
 case 0x07:     key_value=8; break;
 case 0x0b:     key_value=9; break;
 }
```

WHILE Döngüsü

- Koşul doğru olduğu sürece döngü devam eder

```
while (a < b)
  {
   İşlem 1 s1;
   İşlem 2 s2;
   ....
  }
```



a >= b, olduğunda döngüden çıkar bir sonraki komutu yürütür.

C'de Döngü yapıları

- While döngüsü, test edilen koşul doğru olduğu sürece çalışır.

```
While (test ifadesi)
{
    işlem
}
```

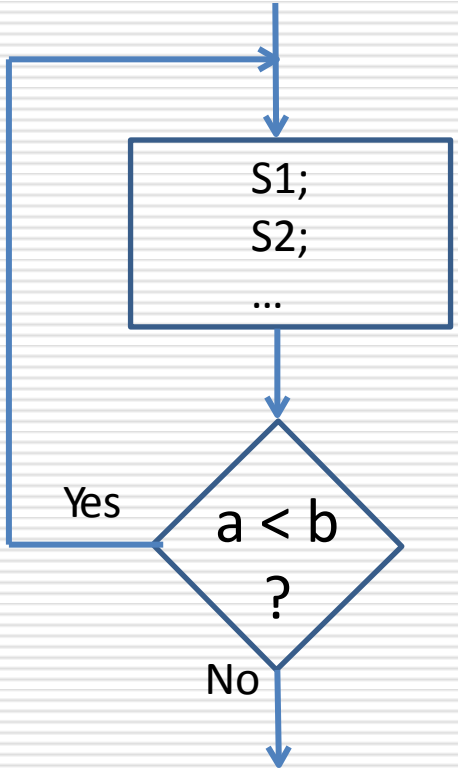
Örnek:

```
While (1)
{   x=0;
}
while (key_value < 1) //basılan tus 1'den küçükse
{
    bul_key();
}
```

DO-WHILE Döngü yapısı

Do While döngüsü, test edilen koşul doğru olduğu sürece çalışır.

```
do
   {
     İşlem 1 s1;
     İşlem 2 s2;
     ....
   }
while (a < b);
```



Önce işlem yapılır sonra test işemi yapılır.

C'de Döngü yapıları

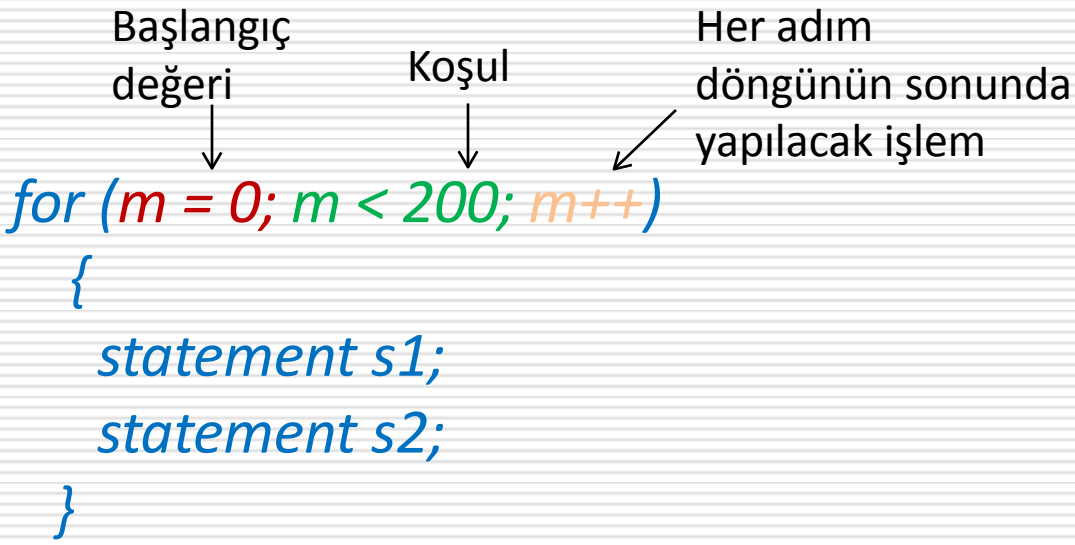
```
do
{
    işlem
} while (test ifadesi)
```

Örnek:

```
X=0;
do
{  X++;
} while (X==4)
```

FOR Döngü Yapısı

- Koşul gerçekleşene kadar adım adım sonuca yaklaşan döngüdür.
  - yapılacak iterasyonun adedi verilir.



C'de Döngü yapıları

- for döngüsü, test edilen koşul doğru olduğu sürece çalışır.

```
for (başlangıç değeri; test ifadesi; sayma)
{
    işlem
}
```

Örnek:

```
void msec(int Delay)      /* 1 milisaniyelik zaman geciktirme
    */
{   int k, z;
   for(k=0; k<Delay; k++)
   {
    for (z=0; z<500; z++);
    }
}
```

C'de Döngü yapıları

```
//--- read ADC result 8 bit --------
AdcResult=0;
for(i=0;i<8;i++) {
   ADC_CLK=1;
   k++;k++;                    // delay about 2 uS
   ADC_CLK=0;
   k++;k++;                    // delay about 2 uS
   AdcResult<<=1;
    AdcResult=AdcResult | (ADC_DO & 0x01);
}
ADC_CS=1;
return(AdcResult);
```

Mantıksal ifadeler

- Sonucu Doğru veya Yanlış olan ifadelerdir. Sonuç sıfır ise yanlış aksi halde doğru kabul edilir.
- iki değer arasındaki ilişkiyi test etmek için kullanılır.

İfade	Anlamı
>	büyük
>=	büyük - eşit
==	eşit
<	küçük
<=	küçük - eşit
!=	eşit değil
!	DEĞİL (NOT)
&&	VE (AND)
||	VEYA (OR)

Bit-Wise (bayt) İşleyen Mantık İfadeler

~	Bit bit işleyen DEĞİL
&	Bit bit işleyen AND
|	Bit bit işleyen OR
^	Bit bit işleyen özel veya (XOR)
>>	Sağa ötele, tabana bölme
<<	Sola ötele, tabanla çarpma

Aritmetik İfadeler

+	Toplama
-	Çıkarma
/	Bölme
*	Çarpma
%	Kalan (mod)
++	bir arttırma
--	Bir eksiltme

Örnek 1: Buton -LED

```
/* bu program butona her basıldığında LED'in durumunu tersler */
#include <REG52.H>
#include <stdio.h>
sbit buton = P1^1;                    //Butonu'i P1.1'a bağla
sbit led = P1^0;                      //LED'i P1.0'a bağla
void main(void)
{
led=0;
    while(1)
    {   while(!buton);
        while(buton);
        led=!led;
    }
}
```

Örnek 2: LED flash yapar

```
/* bu program 1 saniye aralıklarla LED'i flash yapar */
#include <reg52.h>                      // SFR tanımlamaları
#include <stdio.h>
#define SYSCLK    12000000              // SYSCLK frequency in Hz
sbit led = P2^0;                        //LED'i P2.0'a bağla
void msec(unsigned int Delay)           // 1 msaniye geçiktirir.
{    int i, j;
   for(j=0; j<Delay; j++)
   {
     for (i=0; i<500; i++);
    }
}
void main (void)                        // ana program
{     while (1)
      {  led=!led;
        msec (1000);
      }
}
```