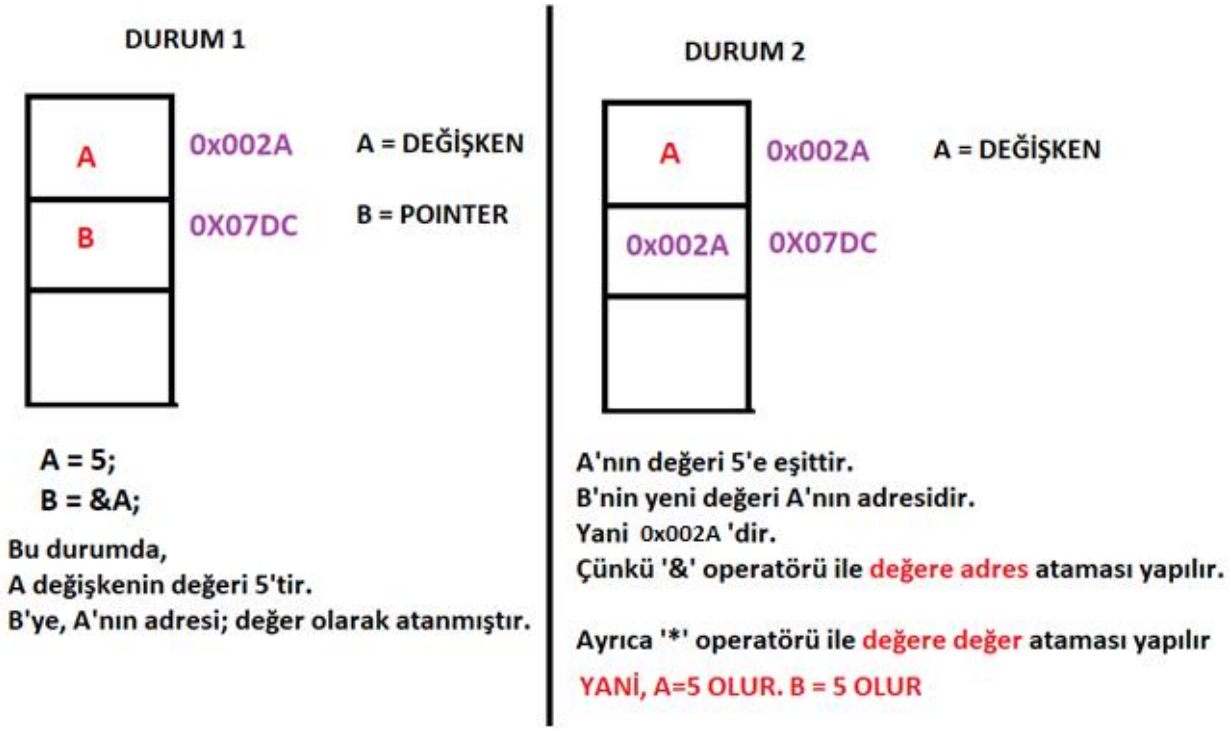
# POINTERS (İŞARETÇİLER)

## By Mustafa Onur Parlak

- **Pointer** kullanımına göre ikiye ayrılmaktadır.
  - **Bellekte** yer alan bir **adrese (&) (Degisken = *Adres)**
  - **Bellekte** yer alan bir **değişkendeki veriye (*) (Adres = &Degisken)** erişim olarak tanımlayabiliriz.
- **Belleklerde kalıcılık durumuna** göre **ikiye** ayrılır.
  - **VM: Volatile Memory, Geçici Hafızadır.** RAM kısmında saklanır. Elektrikselliğe bağlı olarak silinebilir.
  - **NVM: Non-Volatile Memory, Kalıcı Hafızadır.** Flash ve Disklerde saklanır. Elektrikselliğe bağlı olarak silinemez.




```
/* 1- unsigned 8-bitlik bir POINTER tanımlanması isteniyor
*  2- POINTER'a atama yapılacaktır
*  3- Atama yaparken tam sayı ve pointer doğruluk şartı aranıyor
*  4- Adres ataması yapılsın.
*
*  Sonuç olarak;
*  POINTER için bir değer yazılırsa, ilgili atanan adrese gider
*/
uint16_t *pPointer = (uint16_t *)(0x10101010)
```

```
#include <stdio.h>
#include <stdint.h>

int16_t main()
{
   int16_t A = 5;
   int16_t *pB;
   pB = &A;

   printf("A'ın adresi: %x\n", &A);
   printf("B'nin degeri: %x\n", pB);
   printf("A'ın degeri: %d\n\n", A);

/*
  *pB = 123123123;
   printf(“A’nin pB degeri arttırılarak degeri: \n\n”, A);
   */
printf("B'pointer ile gosterdigi deger: %d\n", *pB);
   printf("B'adres ile gosterdigi adresi: %x\n\n", &pB);

   ++pB;
   printf("B'yeni gosterdigi deger: %d\n", *pB);
   printf("B'nin yeni adresi: %x\n", &pB);

   return 0;
}
```

- **POINTER, bellekte saklanan ilgili adrese erişim sağlar, yani adresteki veriyi açığa çıkarır**

## (BU KODLARI YAZIP ASLA BAŞLATMAYINIZ)

```c
#define Durum (0x007_JamesBOND)

int32_t main()
{
  int8_t *Durum = (int8_t*)Durum;

  printf("%c\n", *Durum);

  return 0;
}
```

- **POINTER, bellekte saklanan ilgili veriye erişim sağlar, yani veriye bağlı adresi açığa çıkarır**

## (BU KODLARI YAZIP ASLA BAŞLATMAYINIZ)

```c
#define Durum (0x007_JamesBOND)
#define Durum_Baslat ('A')
#define Durum_Beklet ('B')
#define Durum_Reset  ('C')

int32_t main()
{
  int8_t *Durum = (int8_t*)Durum ;

  // İlgili adrese 'A' yaz. Cihazı başlat
  *Durum = Durum_Baslat;

  // İlgili adrese 'B' yaz. Cihazı duraklat
  *Durum = Durum_Beklet ;

  // İlgili adrese 'C' yaz. Cihazı Resetle
  *Durum = Durum_Reset ;

  return 0;
}
```

- **Gerçekçi durumlarda,** konfigürasyonlar gömülü sistem programlamada sistemin register'larında **struct** olarak saklanır.
    - **Yani, struct – pointer** arasındaki ilişkide struct'ı sabit adrese gösterme yöntemlerine odaklanmamız gerekmektedir.

```
// İlgili etiketleri doğrudan register ile işaret ettirme
#define GPIO_PORTA  (*((volatile uint16_t *)0xXXXXXXXX))
#define GPIO_PORTB  (*((volatile uint16_t *)0xXXXXXXXX))
#define GPIO_PORTC  (*((volatile uint16_t *)0xXXXXXXXX))
```

- **Pointer – Fonksiyon**
    - **Fonksiyon Tanımlama – Çağırma – Değer Geçme – Değer Döndürme**
    -

- **Void Pointer (void*)**

```
// 2 veriyi toplayan Fonksiyonu, Pointer ile oluşturma

#include <stdint.h>

#include <stdio.h>

uint32_t Degiskenler (void* pParametre_1, void* pParametre_2)

{

	uint32_t u32DonusDegeri;

	if (0 != pParametre_1 && 0 != pParametre_2) // NULL == 0 == Güvenli Bellek Alanı

	{

		u32DonusDegeri = ((*(uint32_t*)pParametre_1) + (*(uint32_t*)pParametre_2));

	}

	return u32DonusDegeri;

}

int32_t main()

{

	int32_t i32Numara_1 = 23;

	int32_t i32Numara_2 = 37;


	int8_t i8Numara_1   = 2;

	int8_t i8Numara_2          = 3;x

	printf("int32_t Tipindeki Numaralar Toplami :
%d\n\n",(int32_t)Degiskenler(&i32Numara_1,&i32Numara_2));

	printf("int8_t Tipindeki Numaralar Toplami : %d\n\n",(int8_t)Degiskenler(&i8Numara_1,&i8Numara_2));

	return 0;

}
```