

Giriş

BM307 Mikroişlemcili Sistemler
(1. Konu)

Ders Hakkında

- 3 Saat Teorik + 1 Saat Uygulama
- Mikroişlemci ve Mikrodenetleyici temel kavramları, Sayısal elektronik devre elemanları, MikroC programlama dili ve derleyicisi, Hata ayıklama, PIC mikrodenetleyicilerin özellikleri, Zamanlayıcı, Osilatör, Analog-Dijital Dönüştürücü, Bellek birimleri, Kesme ve çeşitli PIC16F887 MCU deneysel uygulamaları.
- Ders ile ilgili konularda görüşmek üzere ders saatleri dışında öğretim elemanına doğrudan dahili numara ya da e-posta adresi üzerinde ulaşabilir veya ofis saatleri içerisinde odasını ziyaret edebilirsiniz.

Ders Hakkında

- Öğretim Elemanı
Arş. Gör. Halil İbrahim ÇAKIR
- Dahili Telefon
(0228) 214 19 51
- E-Posta
halilibrahim.cakir@bilecik.edu.tr
- Ofis Yeri
Oda No: C348
- Ofis Saatleri
Çarşamba 15:00-17:00
- Derslik
L021 Gömülü Sistemler
Laboratuvarı (-1.Kat)

Uygulama Materyalleri

- UNI-DS6 Deney Seti (LCD, Grafik LCD, Dokunmatik Panel, ...)
- Çeşitli deneysel elektronik devre elemanları (Breadboard, Direnç, ...)
- "PIC16F887 MCU" + "mikroBoard for PIC 40-pin" MCU Programlayıcısı
- "MikroProg Suit for PIC" MCU Programlayıcı Yazılımı
- "MikroC Pro for PIC" Geliştirme Ortamı ve Derleyici Yazılımı
- "Proteus Design Suite" Elektronik Devre Simülatör Yazılımı

Faydalanılan Kaynaklar

Temel kaynak kitap:

PIC Microcontrollers - Programming in C, Milan Verle, 2009
(<https://www.mikroe.com/ebooks/pic-microcontrollers-programming-in-c>)

Yardımcı kaynak kitaplar:

- PIC Programlama, Devrim Çamoğlu, 2016.
- A'dan Z'ye C ile PIC Programlama, Ali Ekber Özdemir, 2017.
- MikroC İle PIC Programlama, Nursel Ak, 2016.

Mikroişlemci ve Mikrodenetleyici

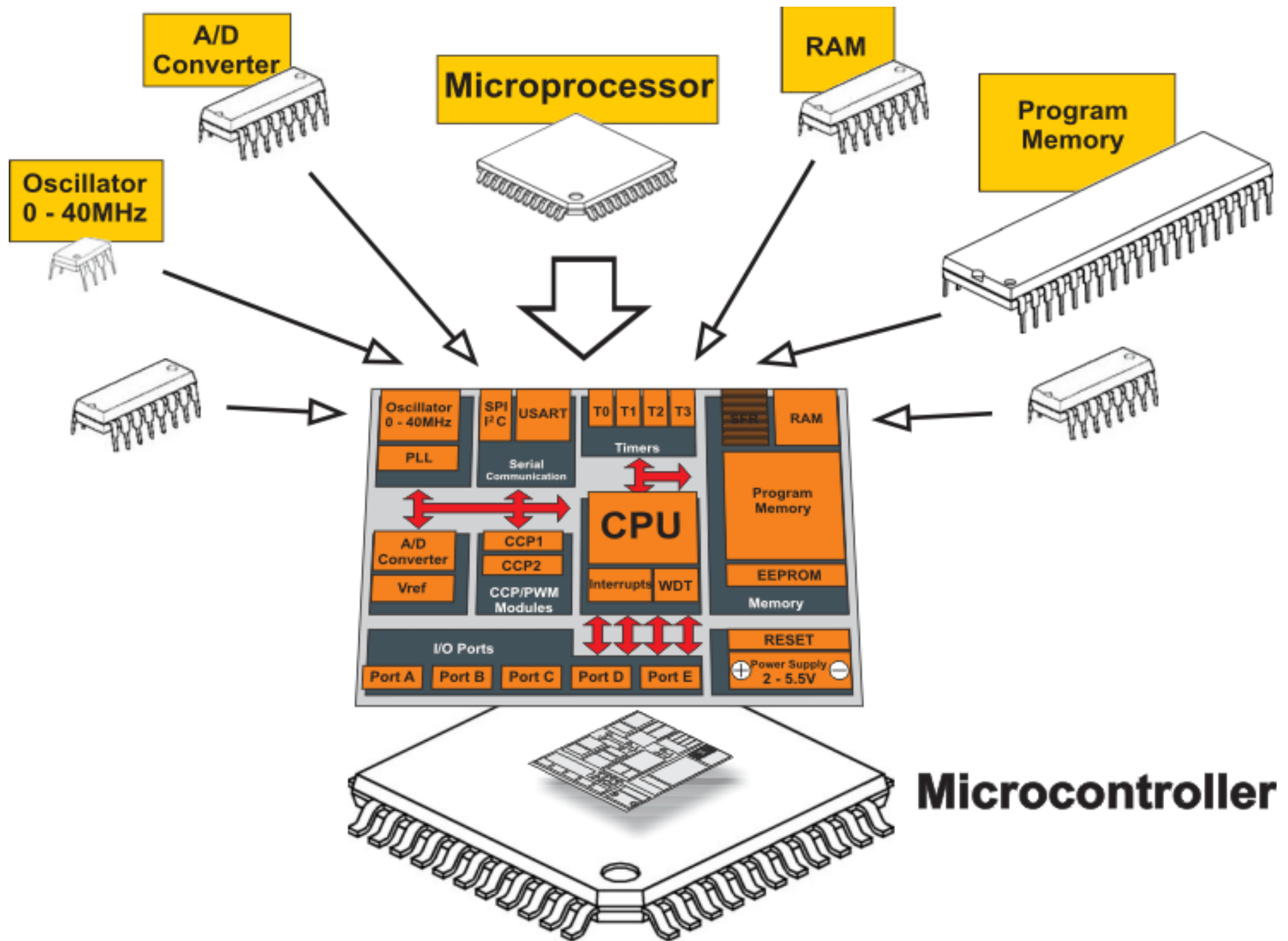
Mikroişlemcinin temel işlevi, işlenen ve kullanılan verileri birimler arasında iletmek, iletilen verileri işlemek, verileri bir durumdan diğerine çevirmek ve verileri saklamaktır.

Mikrodenetleyici (MCU-**Micro Controller Unit** veya μC), elektronik ve elektromekanik sistemleri denetlemek (yönetmek) amacı ile bir **merkezi işlemci ünitesi** (CPU) çevresinde yapılandırılmış, **bellek**, programlanabilir giriş ve çıkışlar (**Inputs/Outputs**), analog/sayısal **dönüştürücü**, sinyal **üretici**, sayıcı, iletişim arabirimi, kristal salınım üretici gibi çevre birimlerinin **tümleşik** bir biçimde yani tek bir yonga şeklinde üretildiği bir **mikro bilgisayardır**.

Mikrodenetleyici Nedir? (Basit Tanım)

- Mikrodenetleyici: Bir denizaltı, vinç veya asansör gibi araçların kontrolünün sağlanabilmesi için gerekli olan elektronik parçaları içeren tek parça halindeki çiplere denir.
- Mikrodenetleyicinin yapmasını istediğiniz şey tamamen sizin isteğinize kalmış. Tabi ki bizi anlayabilmesi için onu kodlamamız gerekiyor.
- İhtiyacımız olan tek şey bilgisayarımızla yazdığımız kodu bu çipin içine transfer etmek.

Mikrodenetleyici Nedir?



Neden mikrodenetleyici kullanırız? (Tarih)

1960'lı yıllarda Sovyetler Birliği ile bir yarışa dönüşen Amerikan uzay çalışmaları, bilgisayarın yoğun bir şekilde kullanımını gerektirmekteydi. NASA'nın beklentileri **uzay araçlarının kısıtlı elektrik stoğunu verimli kullanan**, yani daha az enerji harcayan bilgisayar elektroniği çalışmalarını teşvik etti. Böylece bilgisayar bilimi ile uğraşan Amerikalı bilim adamları **daha küçük, daha az enerji harcayan** Merkezi işlemciler (CPU) üzerinde yoğunlaştı. Bilinen ilk MCU, **1968** yılında yapılmıştır.

Film Önerisi: *Ekim Düşü (October Sky), 1999*

Neden mikrodeneetleyici kullanırız? (Varsayım)

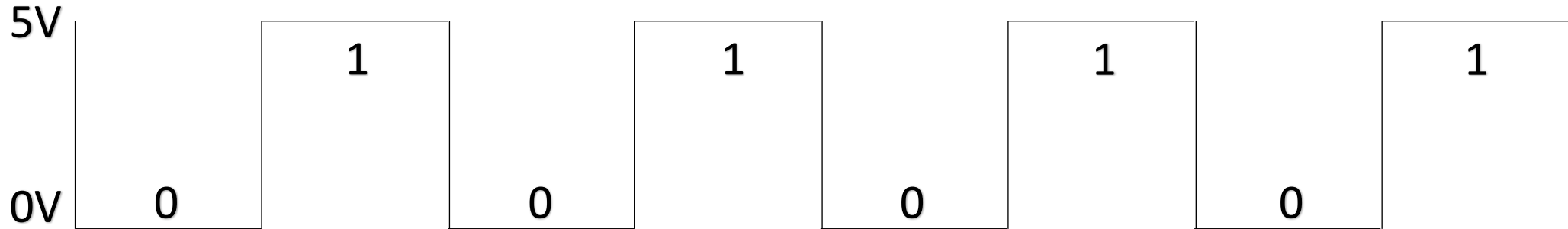
MCU'nun olmadığı bir dünya düşünelim: Örneğin bir ya da birden fazla kişi aynı anda bir asansörü çağırınca ne olur? Asansör önce hangi kata gitmelidir? Ya da elektrikler kesildiğinde asansör nasıl bir davranış sergilemeli? Bu sorulara elbette cevap bulabiliriz fakat bu işlerin yapılması için elektronik bir devreye ihtiyacımız olacak. Hem de her farklı senaryo için farklı bir elektronik devre tasarlamamız gerekli! Haftalarca belki de aylarca çalıştıktan sonra ortaya bir baskı devre kartı çıkacaktır. Eğer işinizde gerçekten iyiyseniz bu kart çalışır, aksi halde kartı tekrardan iyileştirmek ve hatalarını gidermek için günlerce uykusuz kalabilirsiniz...

Neden mikrodenetleyici kullanırız? (Gereklilik)

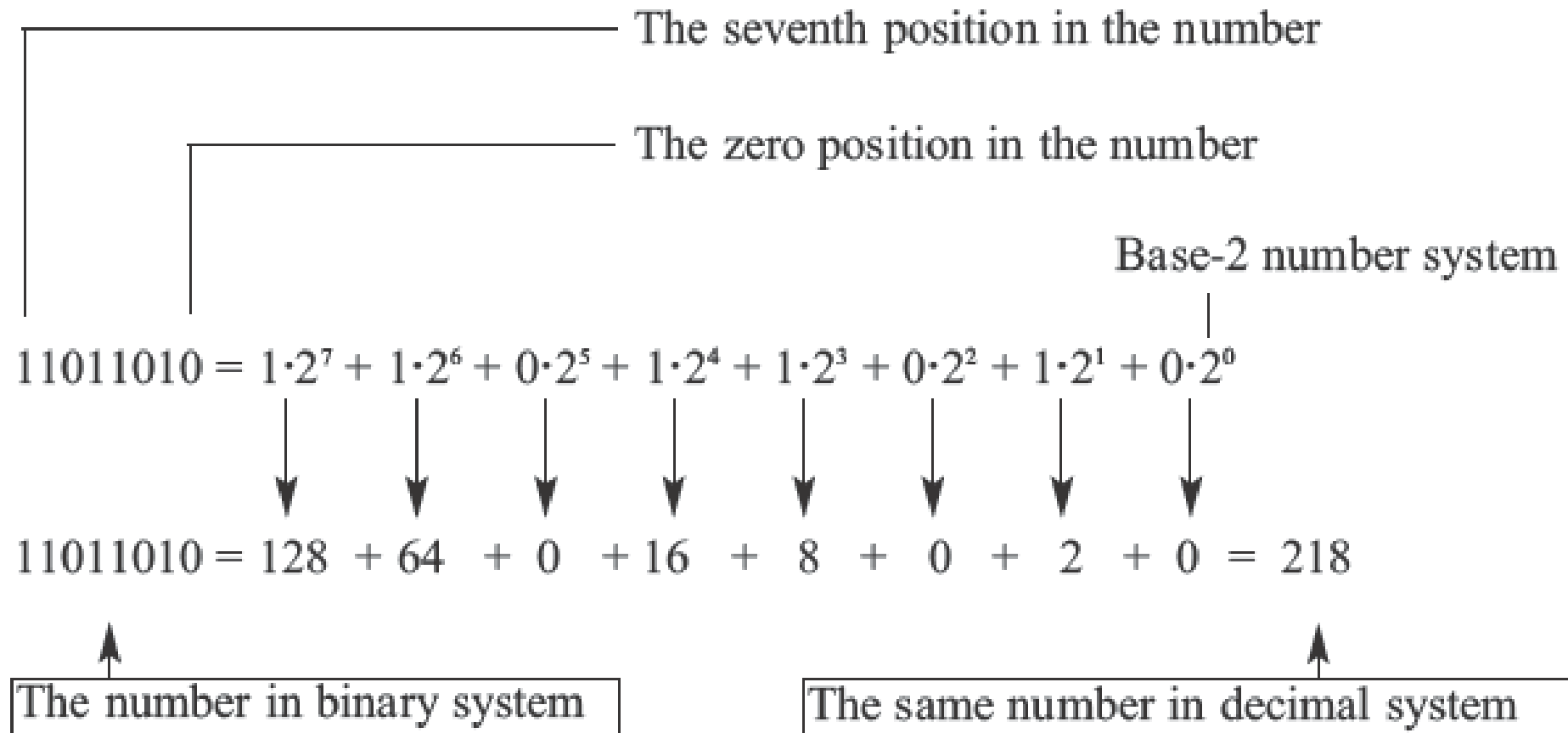
Sonunda asansörün çalıştığını varsayalım. Tüm testler başarıyla tamamlandı. Parayı cebinize koydunuz. Peki sonra başka apartmanlara da aynı kartı taktığınızda sistem düzgün bir şekilde çalışabilecek mi? Yoksa farklı senaryolar için bütün kartı yeniden mi tasarlamamız gerekecek? Belki de evrensel bir cihaz geliştirmelisiniz! Bazı asansörlerde kameralar olacak, bazıları tek kapılı iken bazıları iki kapılıdır. Pek mümkün gözüküyor değil mi? Peki hem ucuz, hem güçlü hem de küçük bir çip kullanırsak ve sistemi kontrol etme işini ona bırakırsak nasıl olur? Envai çeşit projede kullanabileceğimiz, içerisinde kendi programımızı çalıştırabileceğimiz, hatta önce simülatörde test edip sonra baskı devrede kullanabileceğimiz bir çip neden olmasın!

İkili Sayı Sisteminin Bilgisayarda Kullanımı

- İki voltaj seviyesi kullanılarak temsil edilebilecek olan ikili sayı sistemi aritmetiğinin ve mantığının devre tasarımında kullanılabileceği.
- 1937 yılında MIT’de Claude Elwood Shannon tarafından yüksek lisans tezi olarak önerilmiştir.
- **Sinyal Yok:** Alçak (0V), **Sinyal Var:** Yüksek (3,3V veya 5V) olarak iki voltaj seviyesi.

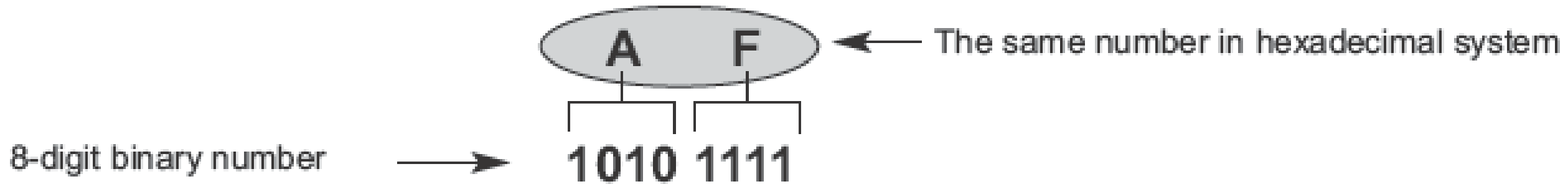


İkili (Binary) Sayı Sistemi – Var (1), Yok (0)



Onaltılı (Hexadecimal) Sayı Sistemi

İkili sayılarla çalışmak insanlara daha zor geldiği için onaltılık sistem geliştirilmiştir:



4 adet ikili sayıya karşılık 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F kullanılır.

İkili Sayıların Onluk Sayılara Dönüştürülmesi

Binary number

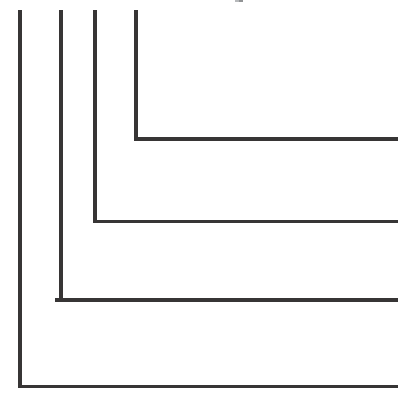
The same number in decimal system

$$110 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 6$$

$2^n - 1$ formülünü kullanarak ikili bir sayının ondalık sayı olarak en fazla gösterebileceği değeri bulabiliriz: Örneğin $n=3$ için 7. Yani $111 = 7$ gibi.

Onaltılık - Onluk Dönüşümü

A37E (hexadecimal number)



	$14 \cdot 16^0 = 14 \cdot 1$	$= 14$
	$7 \cdot 16^1 = 7 \cdot 16$	$= 112$
	$3 \cdot 16^2 = 3 \cdot 256$	$= 768$
	$10 \cdot 16^3 = 10 \cdot 4096$	$= \underline{40960}$

41854 (the same number in decimal system)

Onaltılık-İkili Dönüşümü

$$\text{E4} = \frac{11100100}{\text{E} \quad 4}$$

Sizce neden 255'e kadar?

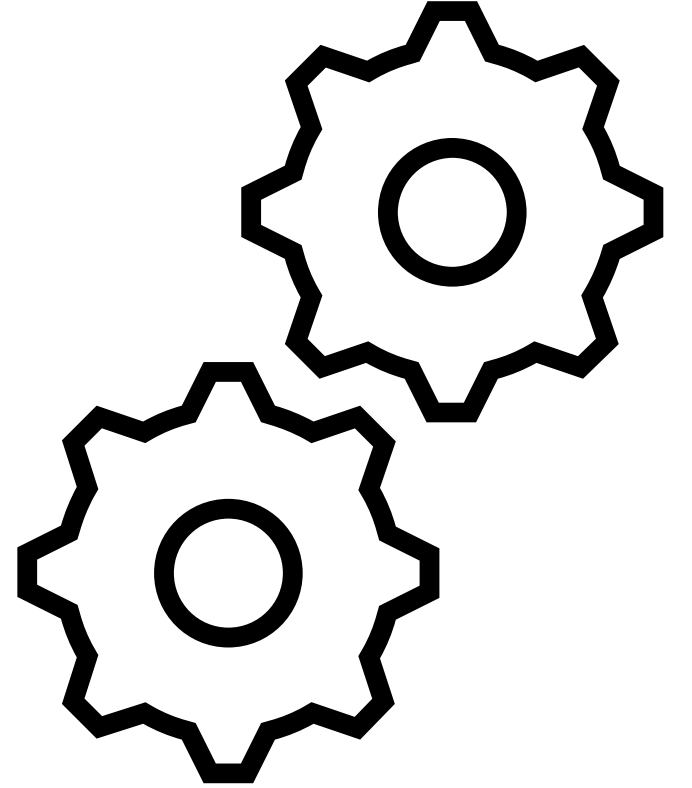
DEC.	BINARY								HEX.
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	2
3	0	0	0	0	0	0	1	1	3
4	0	0	0	0	0	1	0	0	4
5	0	0	0	0	0	1	0	1	5
6	0	0	0	0	0	1	1	0	6
7	0	0	0	0	0	1	1	1	7
8	0	0	0	0	1	0	0	0	8
9	0	0	0	0	1	0	0	1	9
10	0	0	0	0	1	0	1	0	A
11	0	0	0	0	1	0	1	1	B
12	0	0	0	0	1	1	0	0	C
13	0	0	0	0	1	1	0	1	D
14	0	0	0	0	1	1	1	0	E
15	0	0	0	0	1	1	1	1	F
16	0	0	0	1	0	0	0	0	10
17	0	0	0	1	0	0	0	1	11

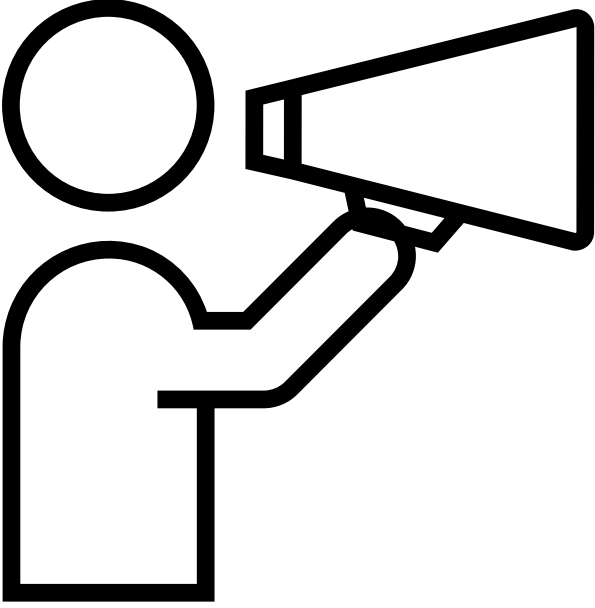
☐ ☐ ☐ ☐ ☐

[illegible]

Uygulama #01

Sayı Dönüşümleri



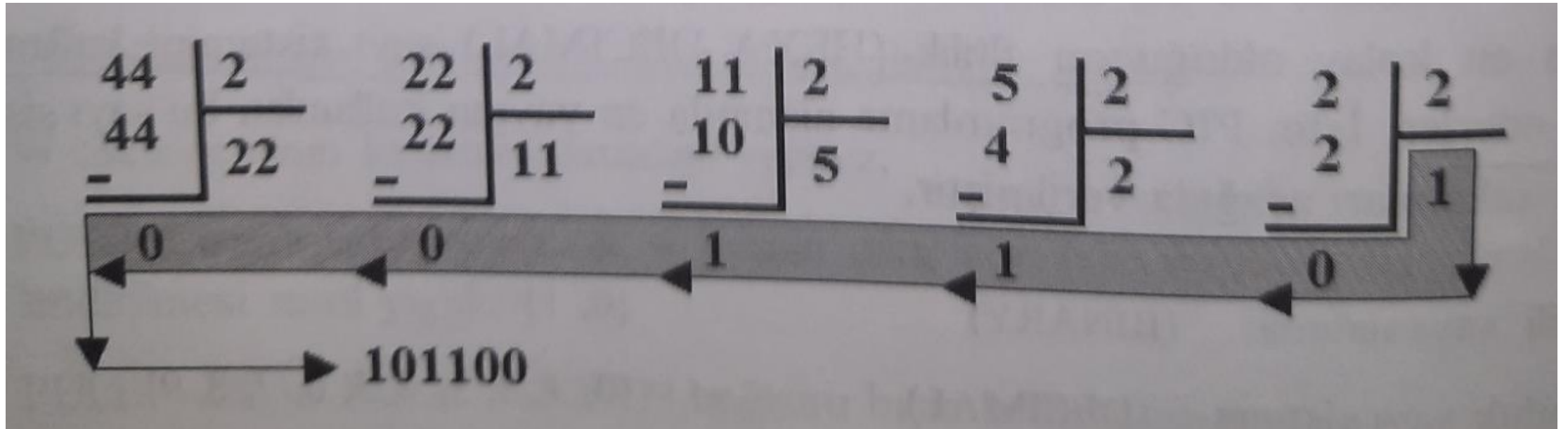


Gruplar ve Ders Günleri

I. Grup: Her Pazartesi Saat 13:00-15:00 arası

II. Grup: Her Cuma Saat 10:00-12:00 arası

Onluk'tan İkili'ye Dönüşüm (Pratik Yol)

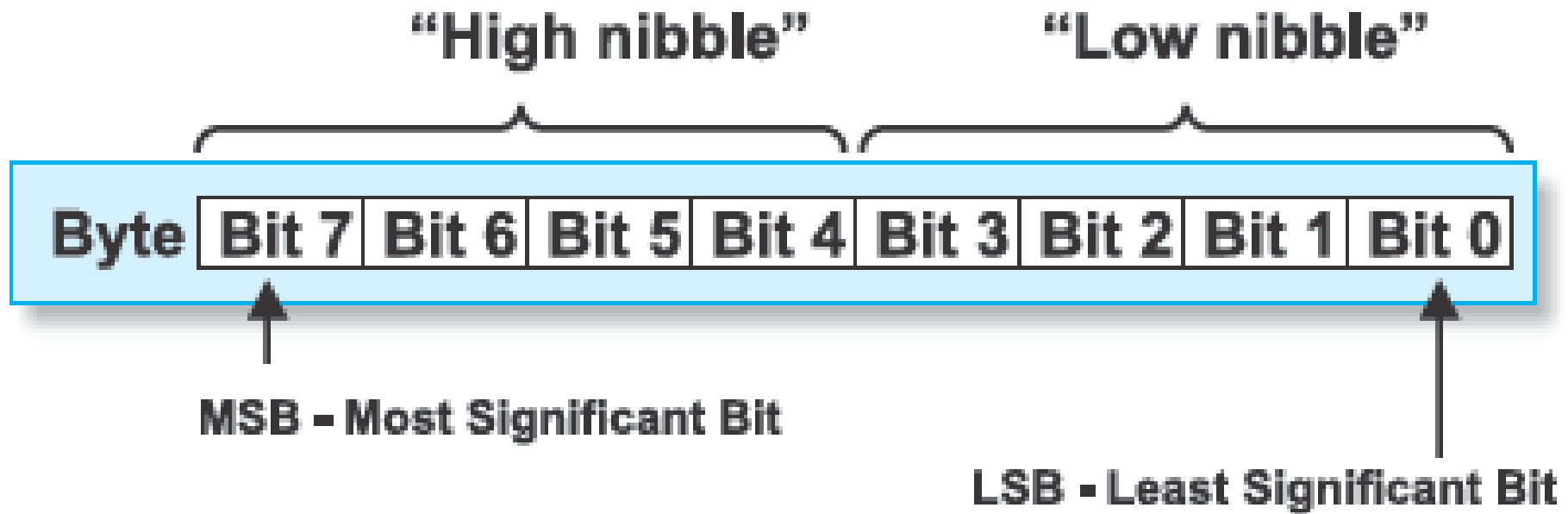


Sayıların Gösterimi ve Bit Kavramı

- **Onaltılık Sayılar:** \$, 0x veya h kullanılarak gösterilir. Örneğin: \$10AF, 0x10AF or 10AFh gibi.
- **İkili Sayılar:** % veya b kullanılarak gösterilir.
- **Onluk Sayılar:** Herhangi bir ön ek ya da son ek almazlar.
- **Bit:** İkili sayıların her hanesi bit olarak isimlendirilir.
- Bit'ler bulunduğu yere göre değerleri değişir (**significance**).
- En sağdaki bit için 0 biti (zero bit-sıfıncı bit) şeklinde isimlendirme yapılabilir. Ayrıca bu en değersiz bit olarak yani Least-Significant-Bit (**LSB**) olarak adlandırılır. En soldaki bit ise Most-Significant-Bit (**MSB**) olarak adlandırılır.

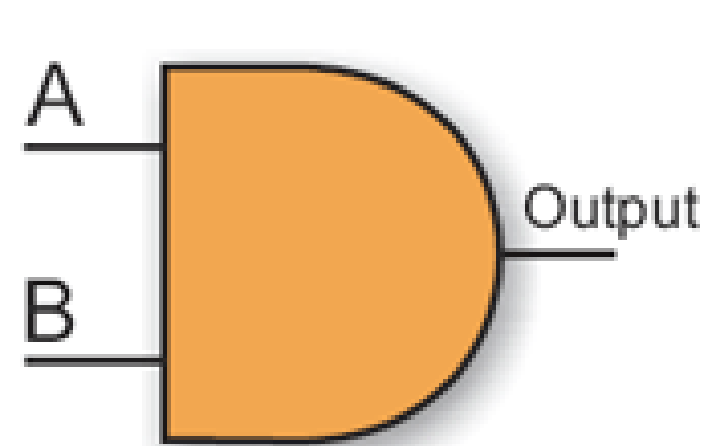
Byte

- 8'lik bit gruplarına denir.

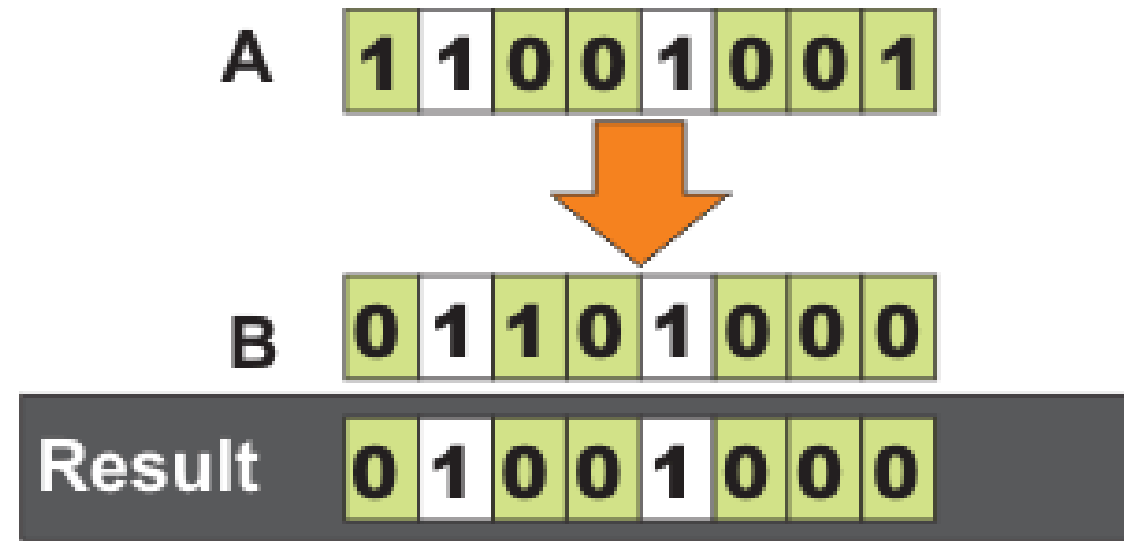


- Nibble: Yarım byte

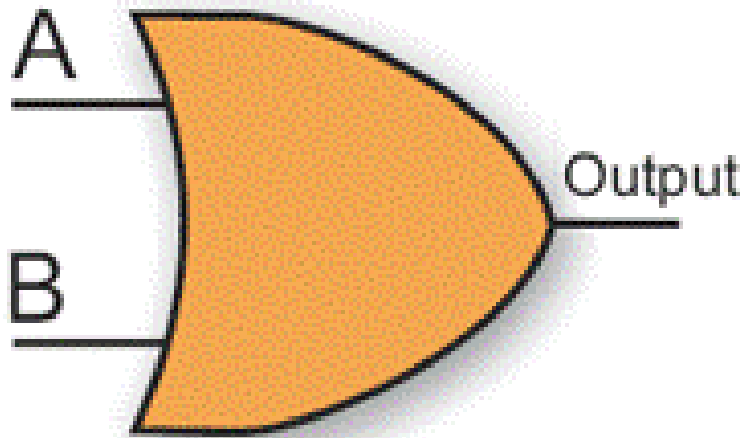
VE Kapısı (AND Gate)



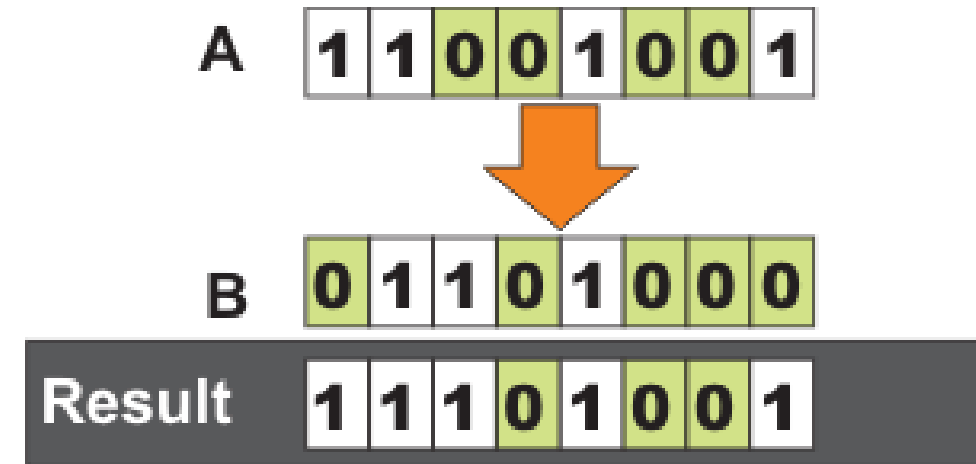
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



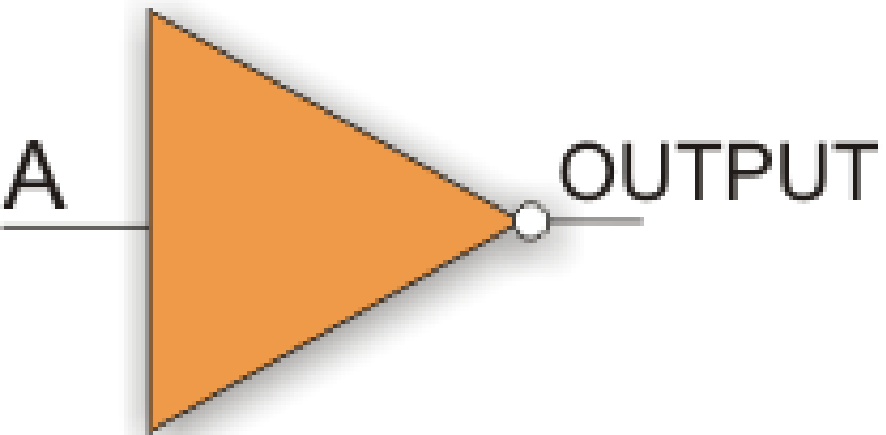
VEYA Kapısı (OR Gate)



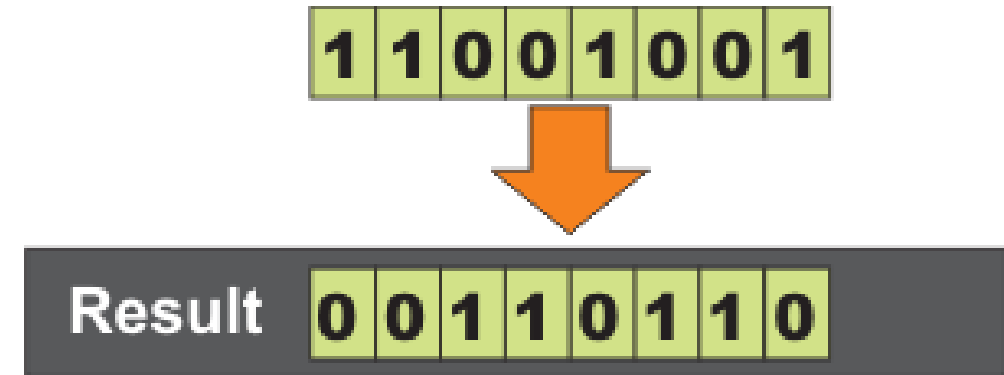
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



DEĞİL Kapısı (NOT GATE) - Inverter

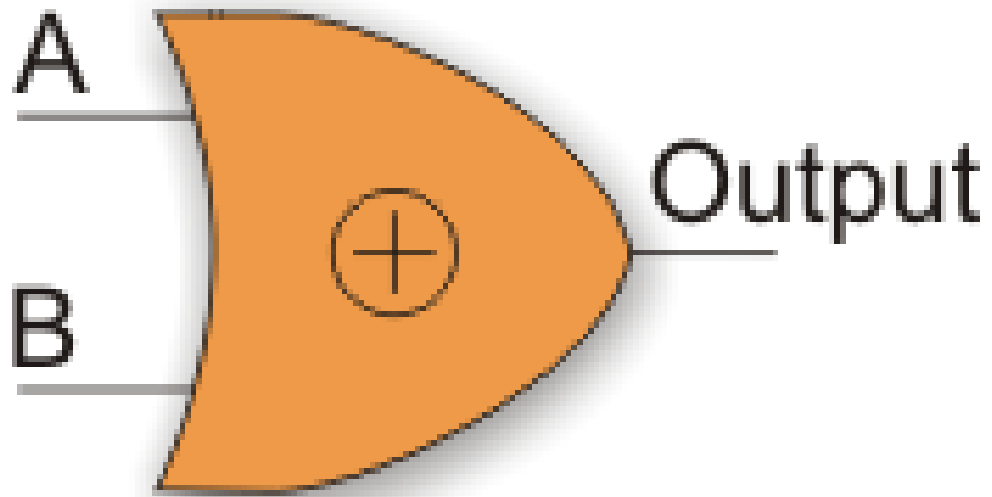


A	Output
0	1
1	0



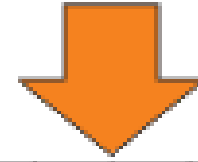
XOR Kapısı (EXCLUSIVE OR GATE)

- Girdiler farklıysa çıktı 1; aksi halde 0 olur.



A

1	1	0	0	1	0	0	1
---	---	---	---	---	---	---	---



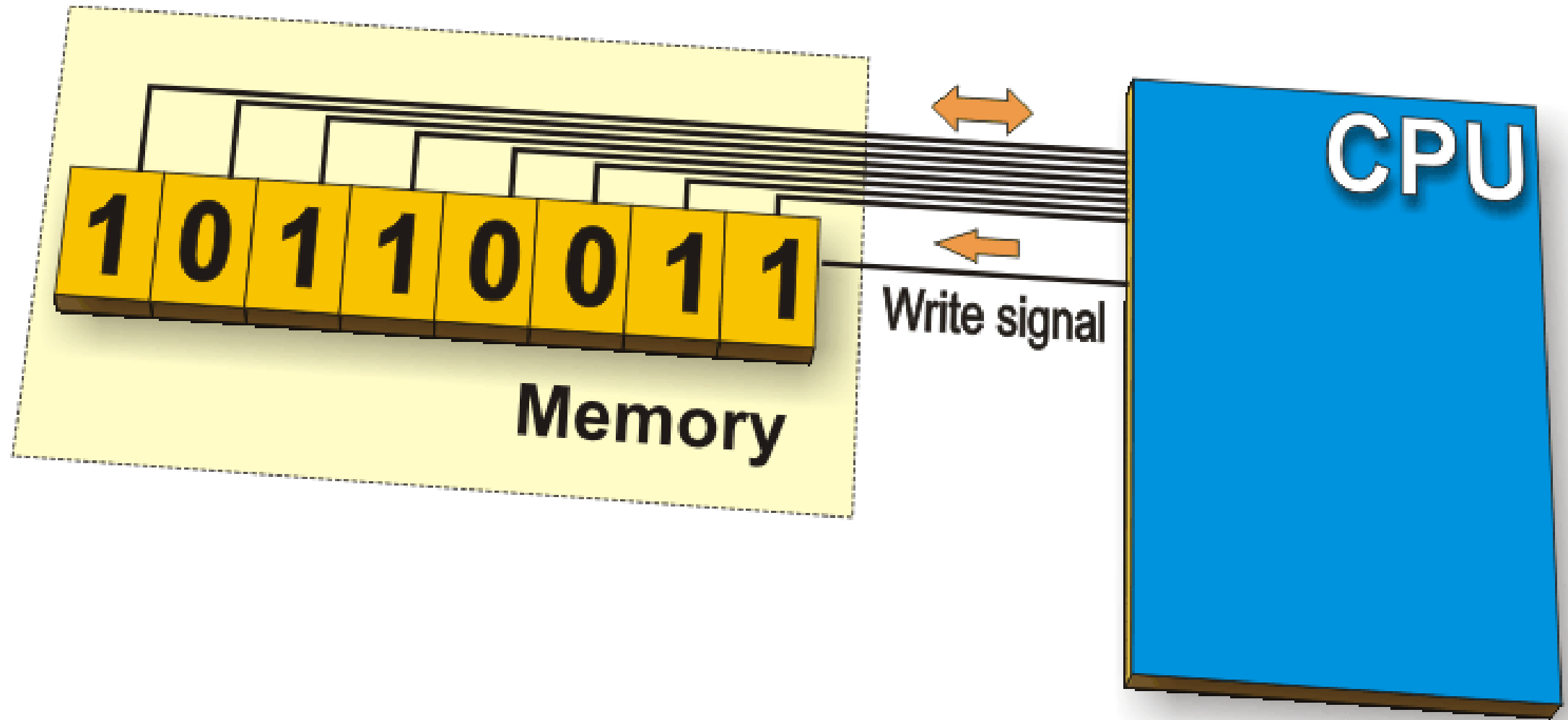
B

0	1	1	0	1	0	0	0
---	---	---	---	---	---	---	---

Result

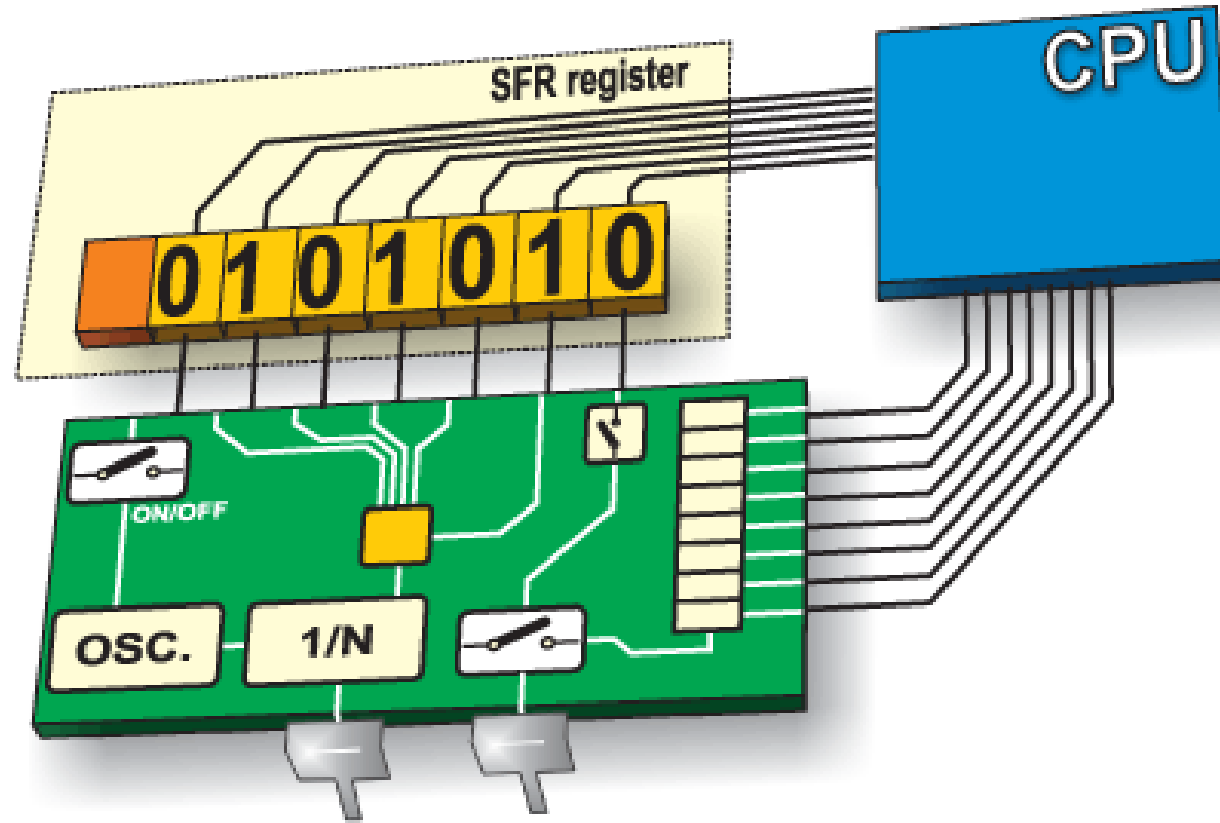
1	0	1	0	0	0	0	1
---	---	---	---	---	---	---	---

Register (Memory Cell)



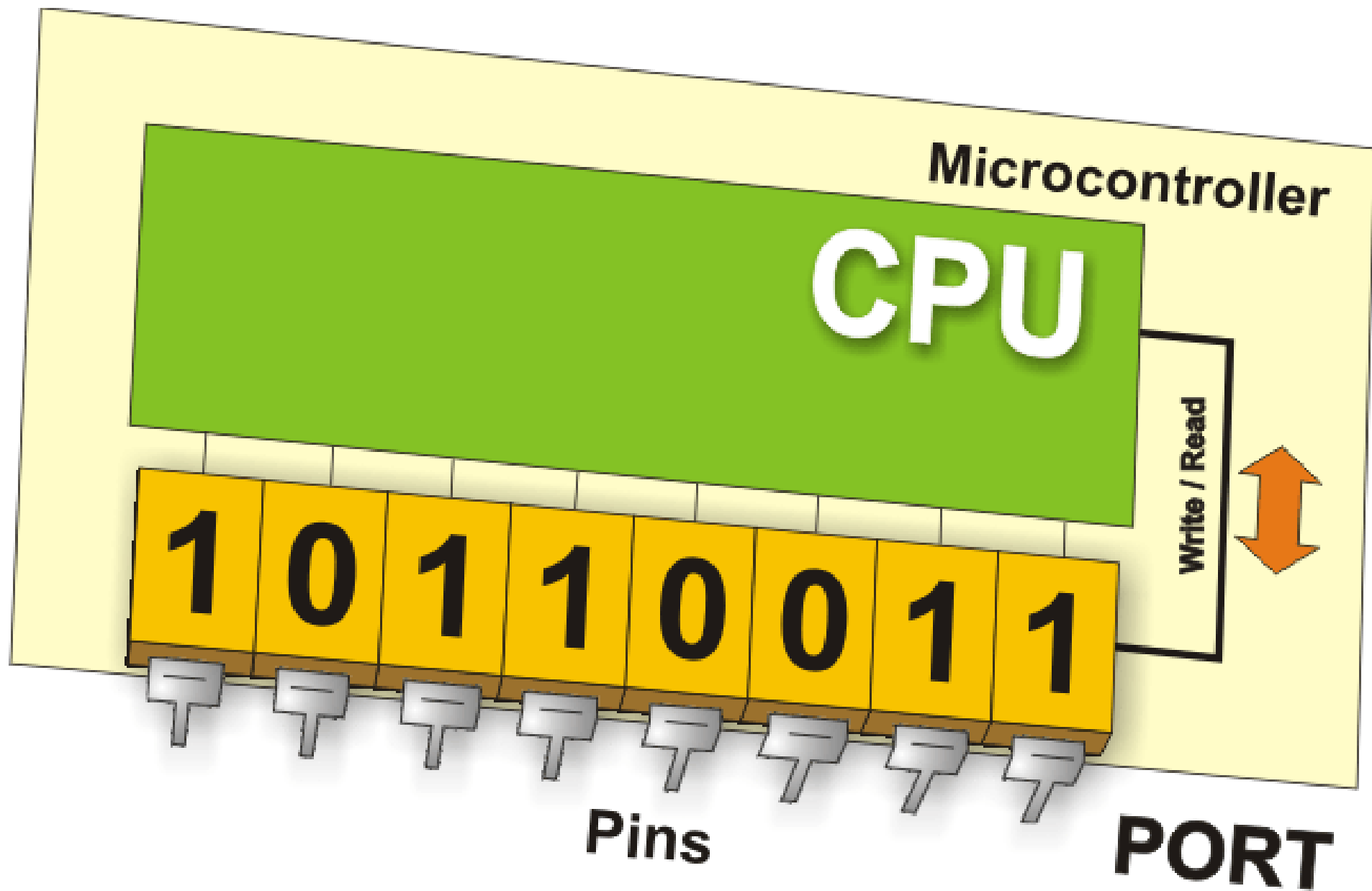
SFR (Special Function Registers)

- Özel bir amaca hizmet eden register'lara denir.
- Doğrudan mikrodenetleyici içindeki devre elemanlarına bağlıdır.



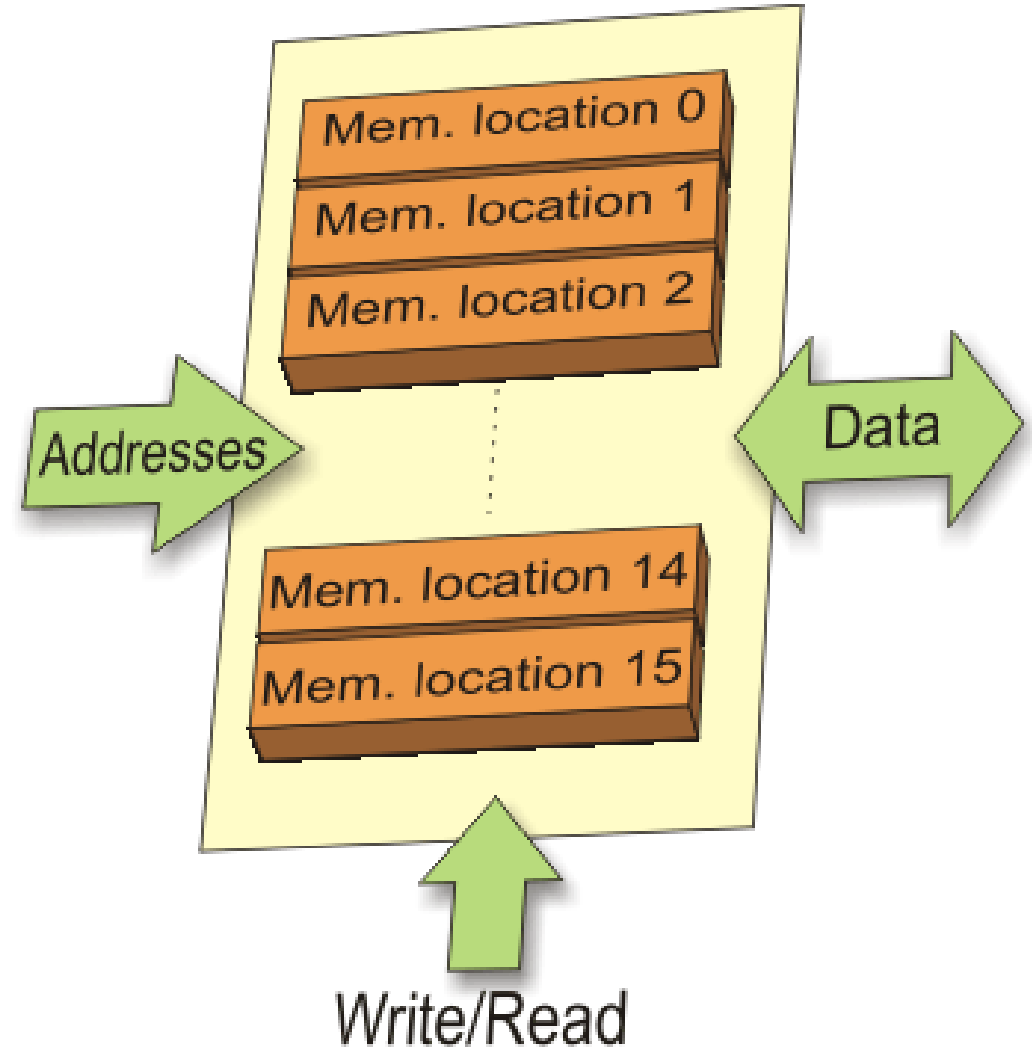
GİRDİ/ÇIKTI (INPUT/OUTPUT) PORT'LARI

- Mikrodenetleyicinin **dışarıya açılan kapıları** diyebileceğimiz register'lara port denir.
- **Örneğin:** 5 adet LED'in MCU tarafından yakılması/söndürülmesi için portlara bu LED'ler bağlanır. Ayrıca 3 adet sensör bağlayıp bunlardan gelen veriler de MCU içine aktarılarak program içinde kullanılabilir.
- Portların **girdi mi yoksa çıktı mı** olarak kullanılacağı önceden belirlenmelidir. Örnekteki gibi bir durumda 5 adet LED çıktı, 3 adet sensör ise girdi olarak ayarlanmalıdır. Portların bu şekilde yönlendirilmesi işi **yazılımsal olarak program çalışırken** yapılır.
- Port'ların her bir biti saklayan kısmının fiziksel uzantısına **pin** diyoruz.



HAFIZA BİRİMİ (MEMORY UNIT)

- Veri saklama amacıyla kullanılan birimdir.
- Dosyaların saklandığı çekmeceli bir dolaba benzetilebilir.
- Çekmecelerin üstünde belirtilen etiketlere bakılarak hangi dosyaların hangi çekmecedeki olduğu bulunabilir (Adresleme).
- Her hafıza lokasyonu belli bir adrese sahiptir.



BAZI HAFIZA BİRİMLERİ

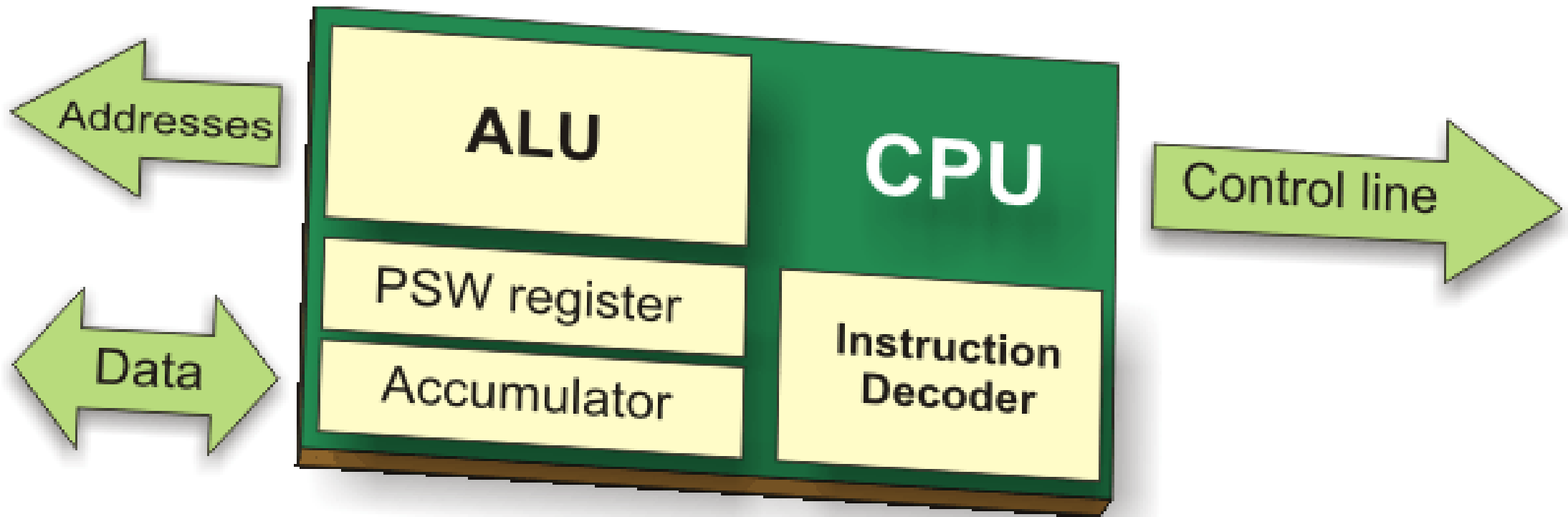
- **READ ONLY MEMORY (ROM):** MCU'nun içinde çalışan *programı kalıcı olarak* barındıran hafızadır. Dolayısıyla yazdığımız programın uzunluğu da bu hafızanın kapasitesine bağlı olarak sınırlıdır.
- **RANDOM ACCESS MEMORY (RAM):** Güç kesintisinde içeriği tamamen *silinen* birimdir. MCU çalışırken ihtiyaç duyulan *geçici* verileri saklar.
- **ELECTRICALLY ERASABLE PROGRAMMABLE ROM (EEPROM):** Hem ROM hem de RAM'in özelliklerine sahiptir. Veriler hem kalıcı saklanır, hem de çalışma anında işlem yapmak için değiştirilebilir. Örneğin şifreli bir kapının şifresini her girdiğinizde kapı açılır. Elektrik kesilse de bu şifre hafızadan silinmez. Ayrıca bu şifre daha sonra değiştirilebilir.

KESME (INTERRUPT)

- Örneğin bir uzaktan kumandanın butonuna basıldığında önce hangi butona basıldığı, eğer bu buton sesi açma düğmesi ise televizyonun sesinin azaltılması işinin yapılması gibi bir olay örgüsü düşünelim.
- Sürekli olarak butonlara basılıp basılmadığı kontrol edilmelidir. Tek işi bu değilse eğer MCU bunu yaparken başka işlere de odaklanması gerekebilir.
- Dışarıdan gelen bu tip değişimlerin algılanması işi kesme dediğimiz özel bir yapıyla sağlanır. MCU bu değişimleri beklemek yerine başka işlere odaklanabilir.

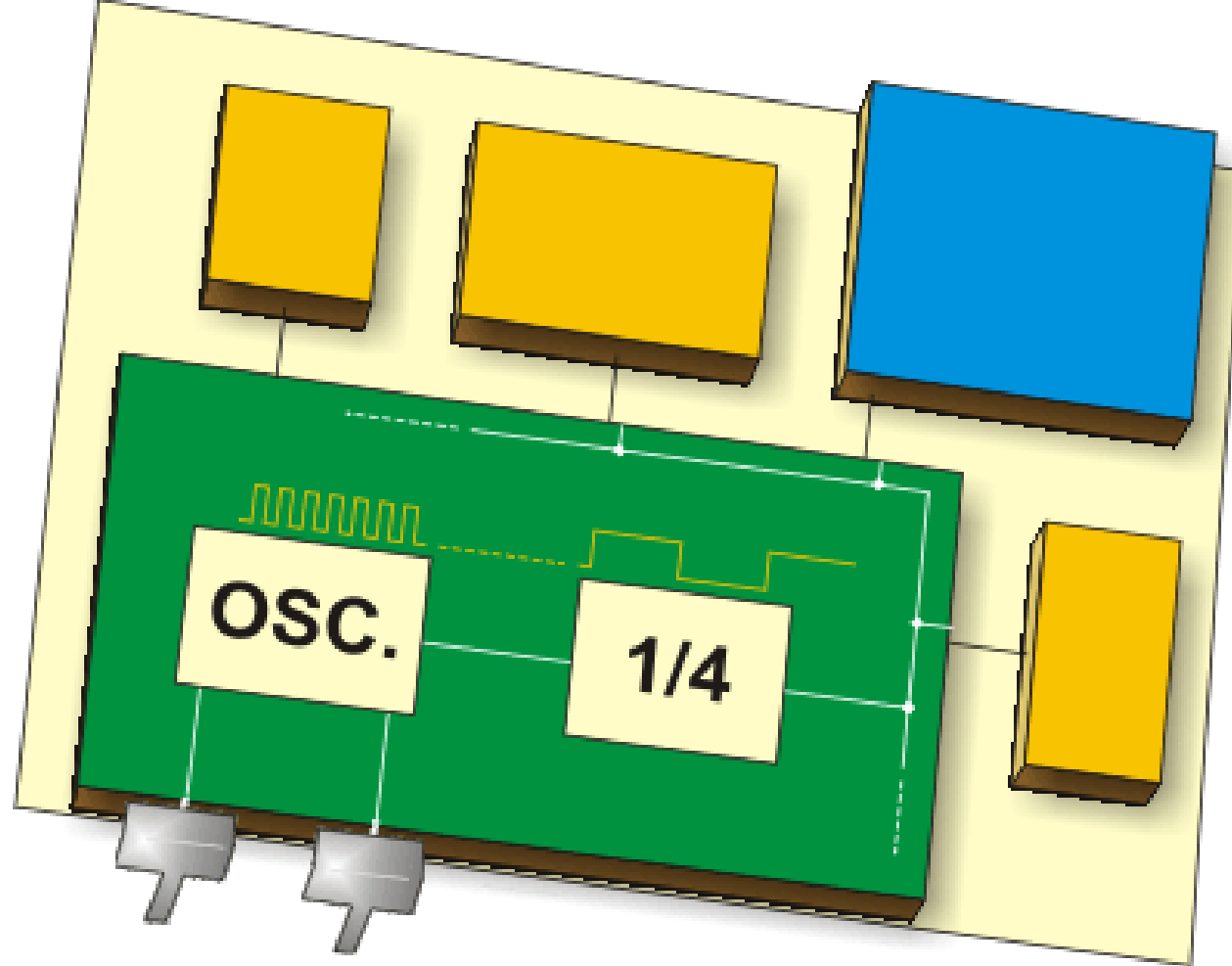
CENTRAL PROCESSOR UNIT (CPU)

- MCU içindeki tüm süreçleri gözlemleyen ve yöneten birimdir.



OSİLATÖR (OSCILLATOR)

- Sinyal üreten bu birim düzenli senkron operasyonların yapılabilmesini sağlar.
- Örneğin MCU içindeki programın baştan sona çalışması bir osilatör koordinasyonunda gerçekleşir.
- Osilatör her darbe ürettiğinde bir komut icra edilir. Bazı komutların icrası birden fazla darbe gerektirebilir.

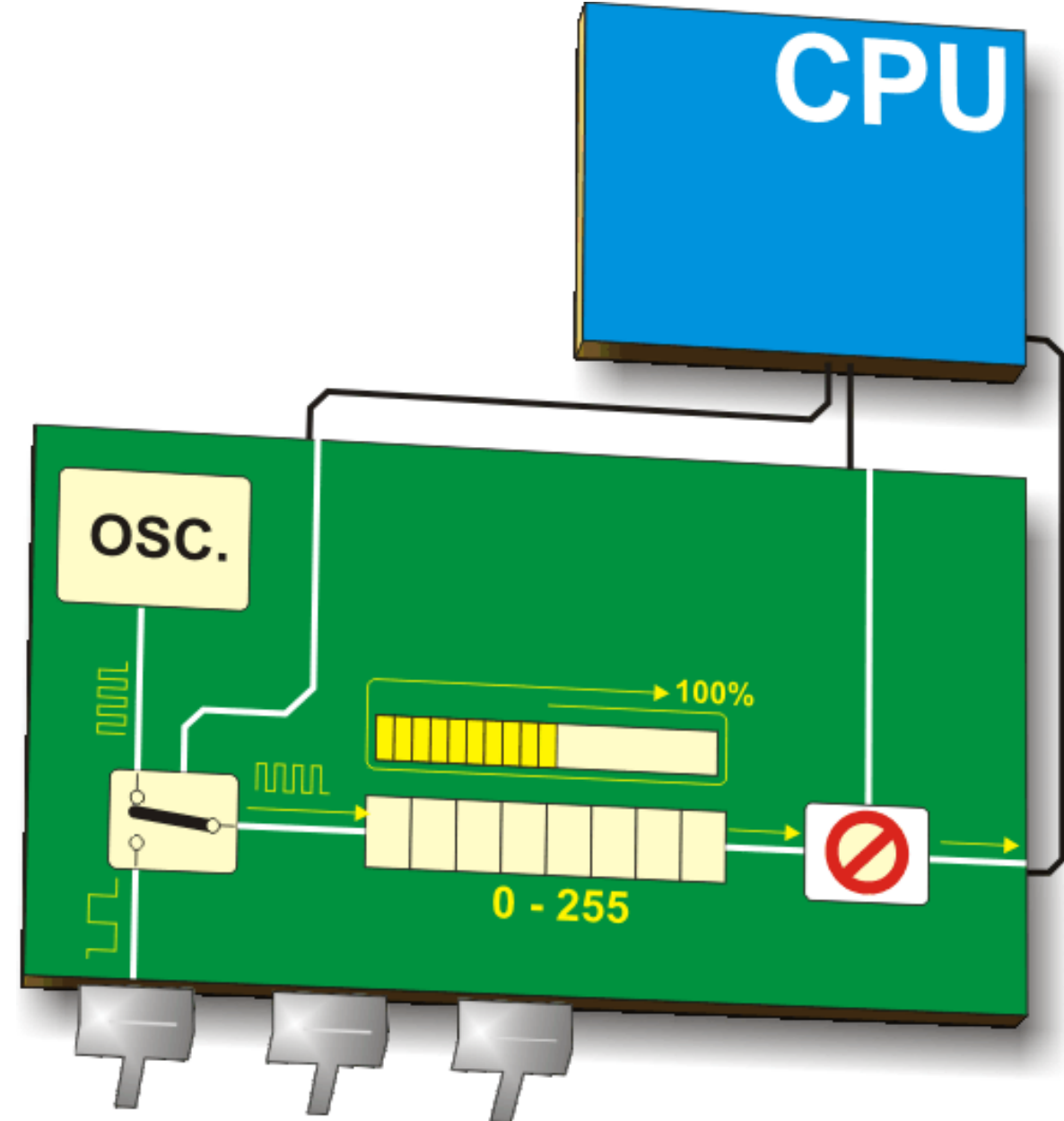


TIMERS/COUNTERS

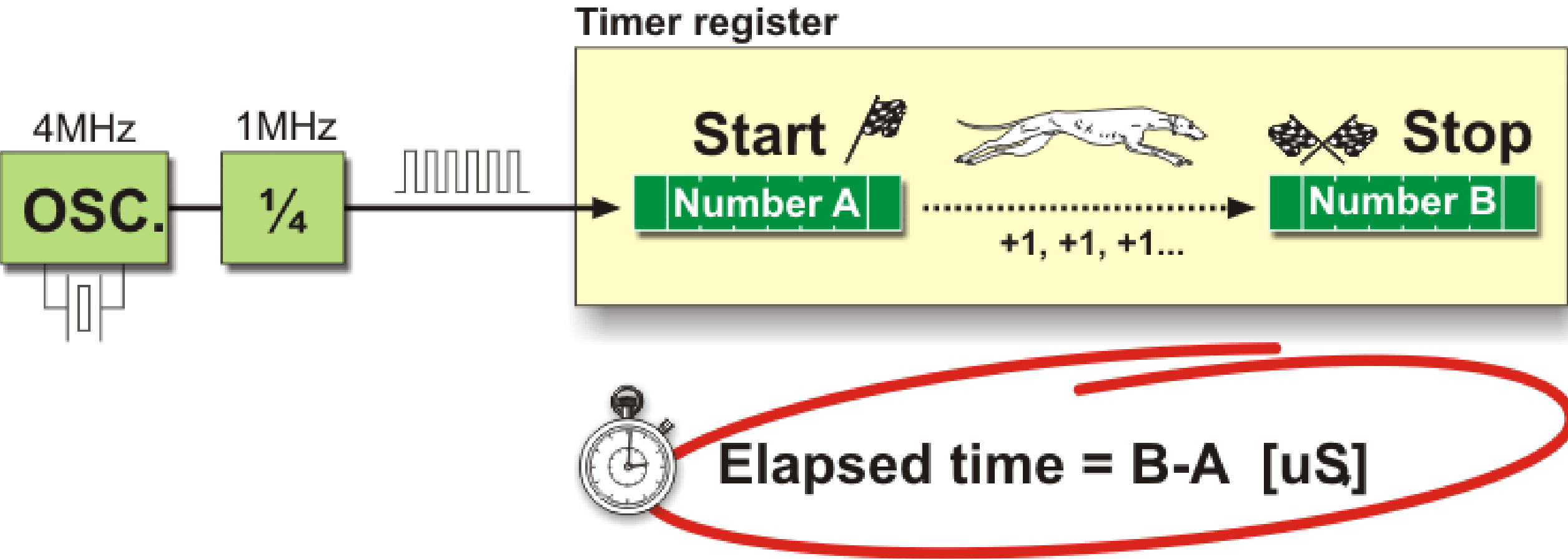
Osilatör'ün ürettiği darbeleri sayar ve iki olay arasında geçen zamanı bu şekilde belirler. Çünkü osilatör frekansına bağlı olarak her darbenin üretilmesi için geçen süre sabittir. Örneğin 20MHz bir osilatörde bir darbe için geçen süre 50nS (nano saniye) kadardır:

$$F = \frac{1}{T} = 20 \times 10^6 \text{ Hz dolayısıyla}$$

$$T = 50 \times 10^{-9} \text{ s} = 50 \text{ nS bulunur.}$$

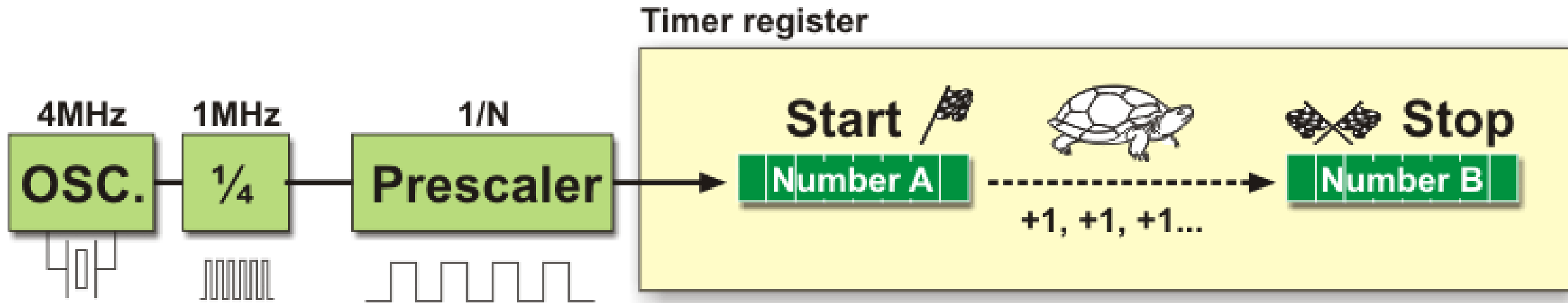


TIMER NASIL ÇALIŞIR?



$$F = 1\text{MHz} = 10^6\text{Hz} \rightarrow T = 10^{-6}\text{s} = 1\mu\text{S}$$

PRESCALER

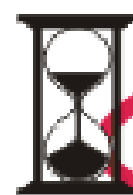
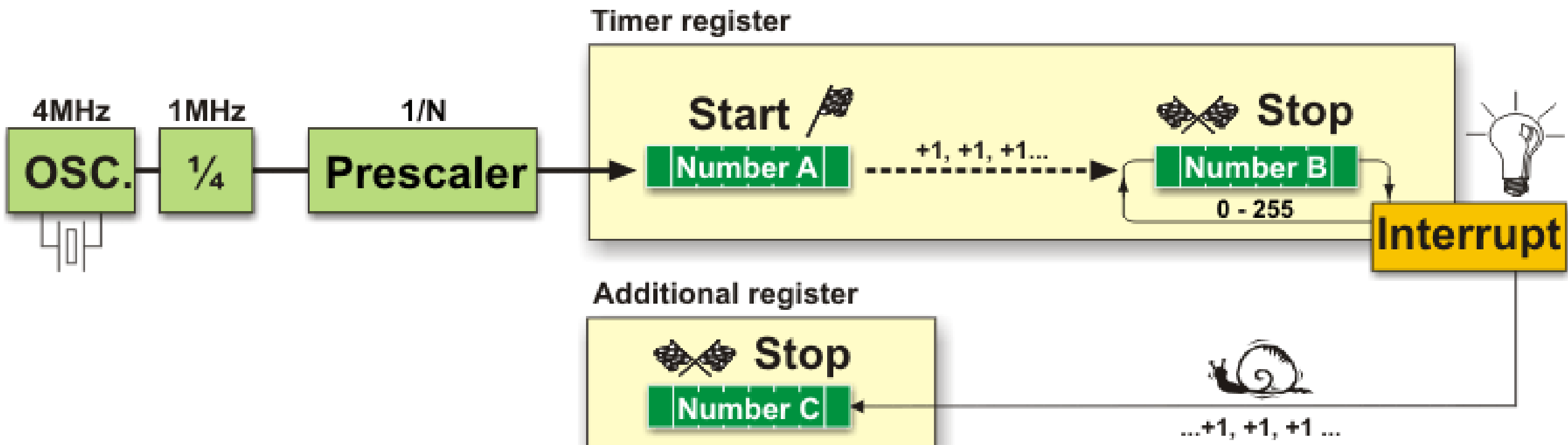


Elapsed time = $N \times (B - A)$ [uS]

$$F = \frac{1\text{MHz}}{N} = \frac{10^6}{N} \text{Hz} \rightarrow T = N \times 10^{-6} \text{s} = N \text{ uS}$$

KESME'NİN TIMER'LA KULLANIMI

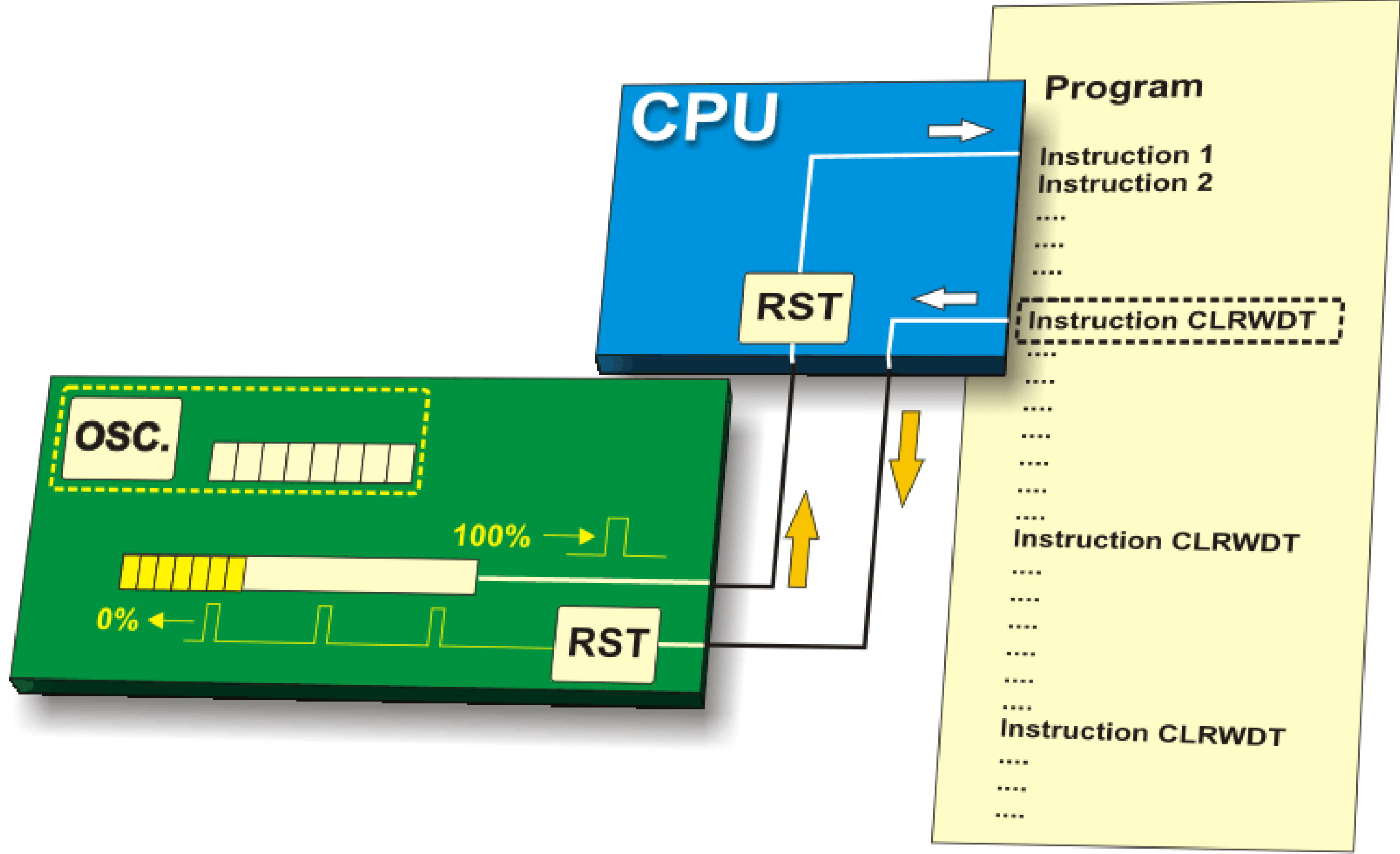
- 8 bitlik bir timer en fazla 255'e kadar değer saklayabilir. 16 bitlik bir timer ise 65.535'e kadar.
- Eğer timer saklayabileceği maksimum sayıya ulaşırsa tekrar başa yani 0'a dönmek zorundadır. Bu duruma taşma (overflow) diyoruz. Her taşma olduğunda kesme rutini (routine) devreye girer.
- Eğer timer saat darbelerini dahili osilatör yerine MCU'nun portlarına ait bir pin'den alıyorsa bu durumda sayıcıya (counter) dönüşür.
- Ancak bu durumda gelen her bir darbenin süresi aynı olmayabilir. Dolayısıyla sağlıklı bir süre hesabı yapmak mümkün olmaz. Sadece sayma amaçlı kullanılabilir.



$$\text{Elapsed time} = N \times (256C + B - A) \text{ [}\mu\text{S]}$$

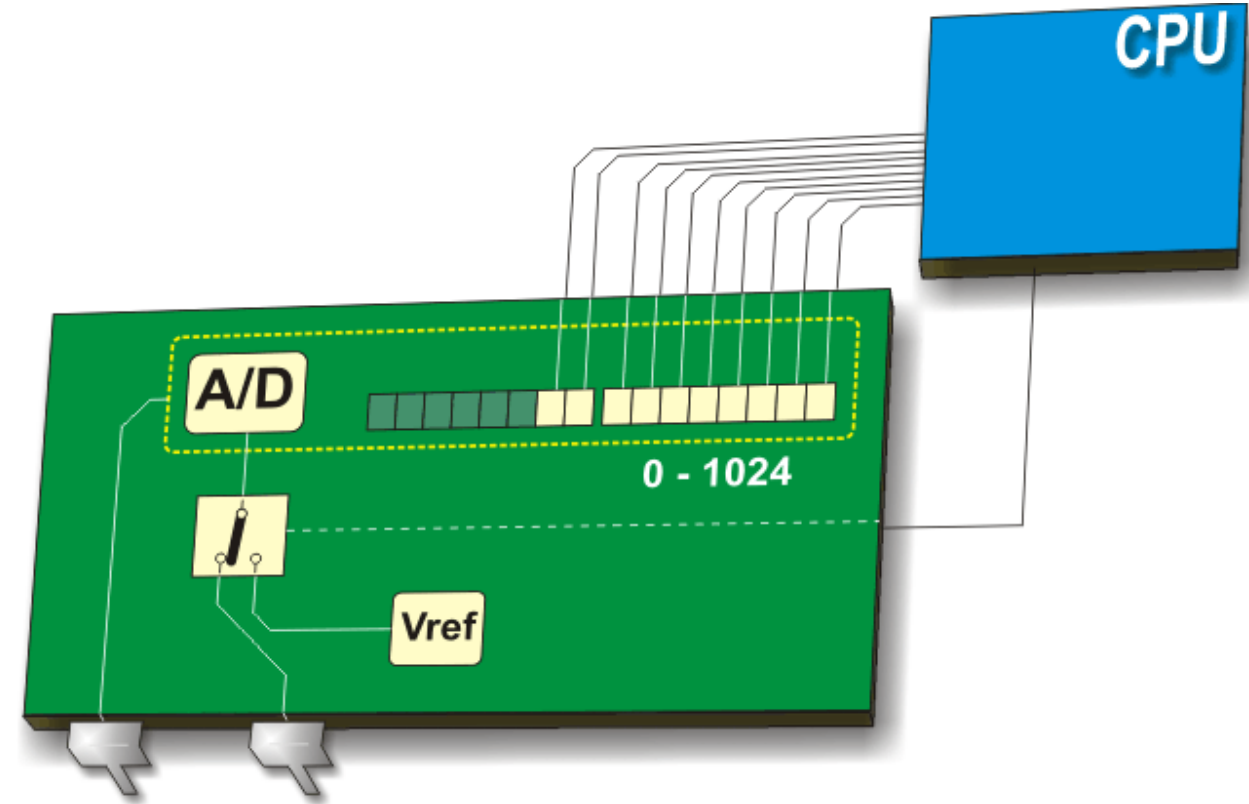
WATCHDOG TIMER (WDT)

- Kendine ait bir osilatörü vardır ve MCU çalışmasıyla birlikte saymaya başlar. Maksimum değere ulaştığında MCU resetlenir. Yani içindeki program tekrar başa döner.
- WDT'yi devre dışı bırakmak mümkündür. Ya da sürekli olarak WDT'nin içeriğini silerek program çalışmasının aksaması önlenabilir.
- Amacı elektronik cihazların başına sıkça gelen elektriksel gürültüler nedeniyle yaşanan donma (stuck) durumlarına çözüm olmaktır. Örneğin cihazınız çalışırken içindeki program birdenbire belli bir noktada kalakalır. Ardından kapatıp açınca düzelir. İşte böyle durumlarda MCU kendini WDT sayesinde resetleyebilir.



A/D Çevirici (ADC: Analogue to Digital Converter)

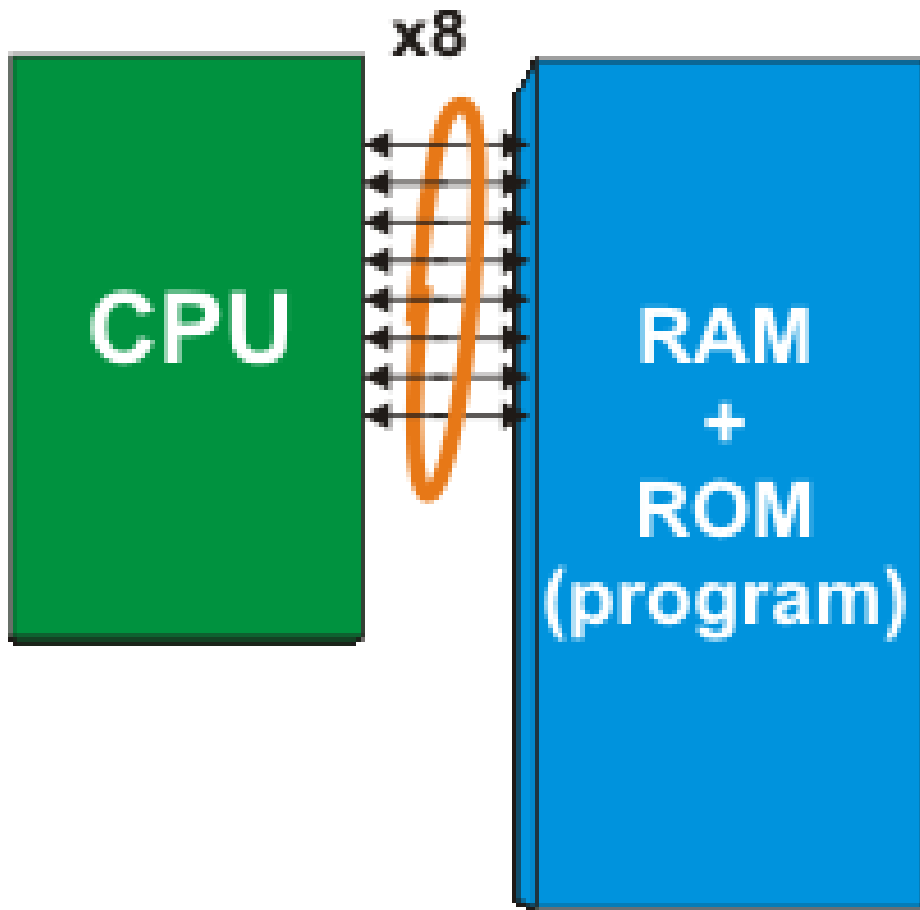
- MCU dışından gelen elektriksel sinyaller genellikle onun anlayabileceği (0 veya 1 gibi) türden değildir.
- ADC ise sürekli (continuous) sinyalleri kesikli (discrete) dijital sayılara dönüştüren elektronik bir devredir. Yani analog bir değeri CPU'nun işleyebileceği ikili bir sayıya çevirir.



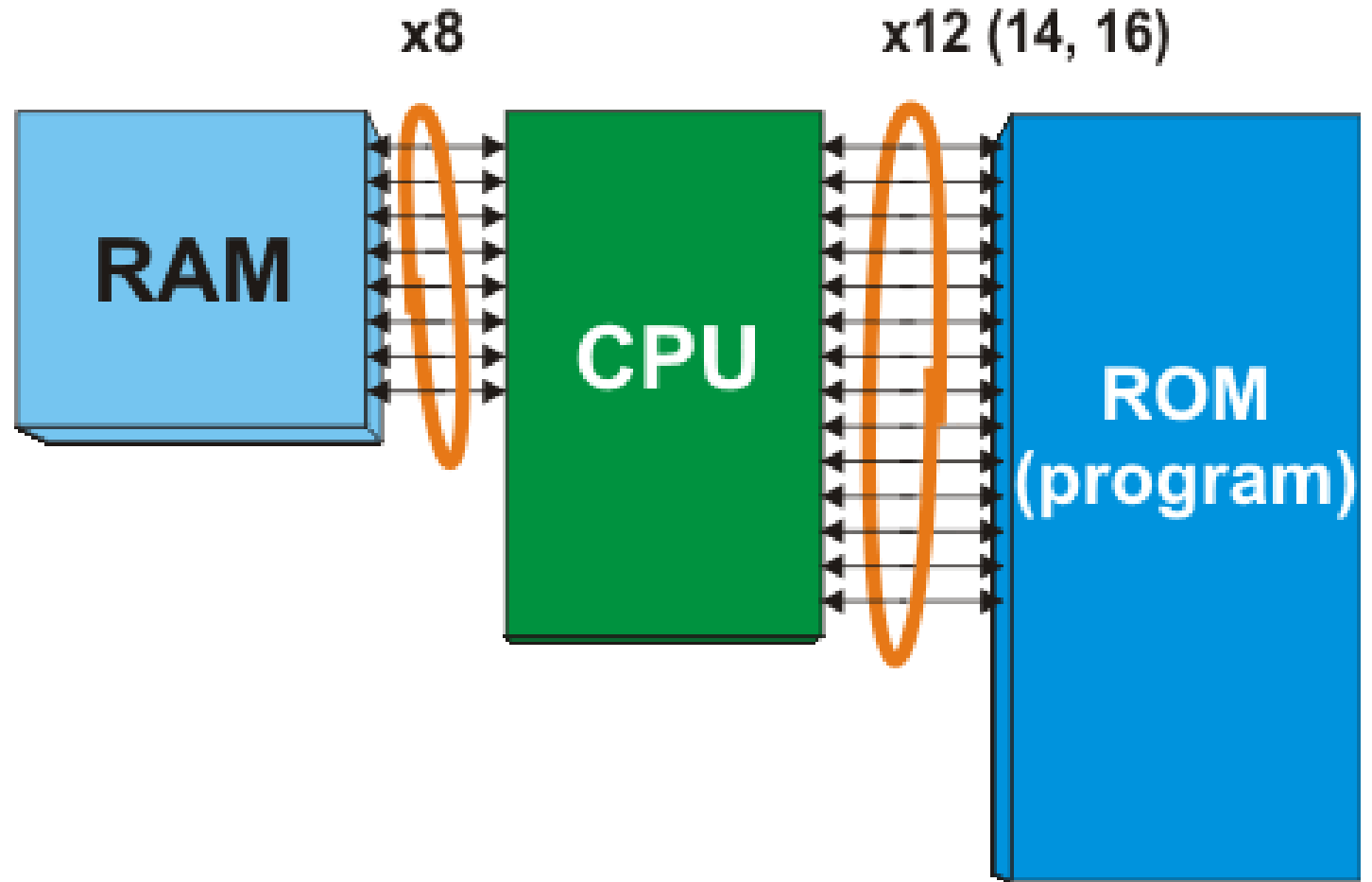
İşlemci Mimarileri

CPU ile bellek arasındaki veri alışverişinin türüne göre iki farklı mimari kullanılır.

- 1. Von-Neumann Mimarisi:** Tüm veri alışverişi sadece 8 bitlik tek bir veri yolu (bus) üzerinden yapılır. Bu yüzden fazlaca trafik oluşturur. CPU aynı yol üzerinden ya program verisi (data) ya da program komutlarını (instruction) okur (read) veya yazar (write).
- 2. Harvard Mimarisi:** İki farklı veri yolu kullanılır. Birincisi RAM ile CPU arasında ikincisi ise ROM ile CPU arasındadır. Dolayısıyla aynı anda farklı türden veriler aktarılabilir.




Von-Neumann



Harward

İşlemci Komut Kümesi (Instruction Set)

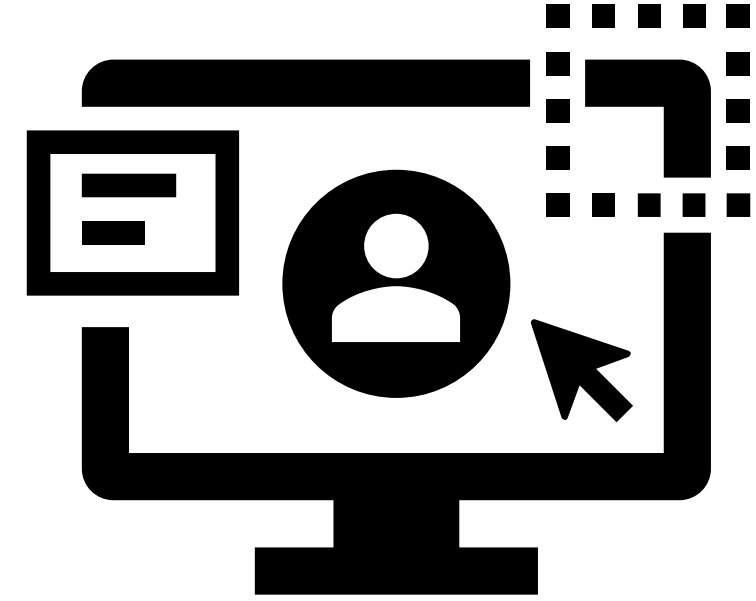
- MCU'nun anlayabileceği komutların tümüne denir.
- Bu komutların dili assembly'dir.
- Komut sayısına göre iki farklı komut kümesi bulunur:
 1. **RISC (REDUCED INSTRUCTION SET COMPUTER):** Öğrenmesi kolay. Bazen bir iş birden fazla komut kullanılarak yapılmalıdır.
 2. **CISC (COMPLEX INSTRUCTION SET COMPUTER):** Öğrenmesi zor.



```
...      ...  
...      ...  
movlw    0x3F  
movwf    TEMP1  
btfsc    MAX3, 7  
goto     check  
btfsc    MAX3, 6  
goto     opening  
btfsc    MAX3, 5  
goto     closure  
...      ...  
...      ...
```

RISC vs CISC

- RISC ile yapılan işlemler CISC'e göre 2 veya 4 katı performanslıdır.
- RISC'te aynı anda çok sayıda komut işlenebilir.
- CISC'te transistör sayısı nedeniyle alan gereksinim daha fazladır ve çalışırken daha fazla ısı ortaya çıkarır (soğutma gereksinimi).
- RISC'in komutları CISC kadar güçlü değildir, aynı işi daha fazla komut kullanarak yapmak gerekebilir.
- Genellikle mikrodenetleyicilerde ve bazı süper bilgisayarlarda RISC mimarisi tercih edilirken, kişisel bilgisayarlarda ise CISC tercih edilmektedir.



Gerekli Yazılımların Kurulumu

- Proteus Design Suite 8.10 (veya 8.6) Professional
- MikroC Pro for PIC Version 7.6.0 (veya v7.1.0) - Registered
- MikroProg Suit for PIC v2.80

Kaynaklar

<https://www.mikroe.com/ebooks/pic-microcontrollers-programming-in-c> (PIC Microcontrollers - Programming in C, Milan Verle, 2009)

PIC Programlama, Devrim Çamoğlu, 2016.

MikroC İle PIC Programlama, Nursel Ak, 2016.

A'dan Z'ye C ile PIC Programlama, Ali Ekber Özdemir, 2017.