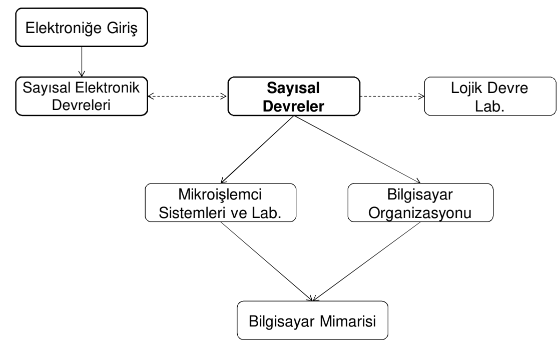
SAYISAL DEVRELER

Doç.Dr. Feza BUZLUCA
İstanbul Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü

http://akademi.itu.edu.tr/buzluca
http://www.buzluca.info

İTÜ Bilgisayar Mühendisliği Bölümündeki donanım derslerinin bağlantıları

Sayısal devreler bölümdeki diğer donanım dersinin temelini oluşturmaktadır.

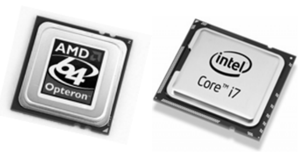


Sayısal devrelerin kullanım yerleri:

Günümüzde çevremizde gördüğümüz nerdeyse tüm elektronik cihazlar sayısal devreler içerirler.

Örnekler:

- Merkezi işlem birimi (*CPU*): Eş zamanlı ardışıl devredir.
  Bu tür sayısal devreleri dersin ikinci bölümünde inceleyeceğiz.
- Bilgisayar bellekleri: Belleklerin yapı taşlarını oluşturan "flip-flop" ve tutucu (*latch*) adlı sayısal devre elemanlarını bu ders kapsamında göreceğiz.
- Ev elektroniği: TV, çamaşır makinesi kontrol birimi, ses ve görüntü cihazları
- Otomobiller: ABS, motor ateşleme denetimi
- Cep telefonları

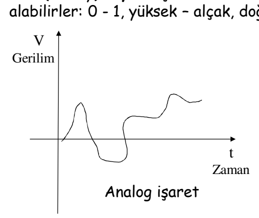


Analog - Sayısal (Dijital) İşaretler:

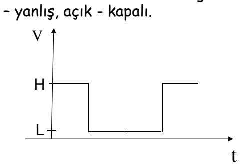
Gerçek dünyada karşılaştığımız bir çok fiziksel büyüklüğün (akım, gerilim, sıcaklık, ışık şiddeti vb.) değeri sürekli bir aralık içinde değişmektedir.

Sınırlar arasındaki her türlü olası değeri alabilen bu tür işaretlere **analog** işaretler denir.

İkili (*binary*) **sayısal** işaretler ise belli bir anda sadece olası iki değerden birini alabilirler: 0 - 1, yüksek – alçak, doğru – yanlış, açık - kapalı.



Analog işaret



Bir bitlik ikili sayısal işaret

Bir analog işareti temsil edebilmek için bir bitten daha fazla ikili sayısal işarete gerek duyulur.

Sayısal Sistemlerin Avantajları:

Eskiden analog sistemlerin kullanıldığı bir çok alanda günümüzde daha avantajlı olduğundan sayısal sistemler kullanılmaktadır.

Örnekler: Fotoğrafçılık, video, ses kayıtları, otomobil motorları, telefon sistemleri vb.

Sayısal Sistemlerin Avantajları:

• Bir sayısal sisteme aynı giriş kümesi defalarca uygulandığında hep aynı çıkış kümesi elde edilir.
Burada aynı giriş kümesinin uygulanması demek her defasında aynı değer dizisinin aynı sırada uygulanması demektir. Analog sistemler ise çevre koşullarından daha çok etkilenirler ve çıkışları değişkenlik gösterebilir.

• Sayısal tasarım (lojik tasarım) dayandığı matematiksel temeller açısından daha kolaydır. Ayrıca sayısal sistemleri test etmek ve hatalardan arındırmak da analog sistemlere göre daha kolaydır.

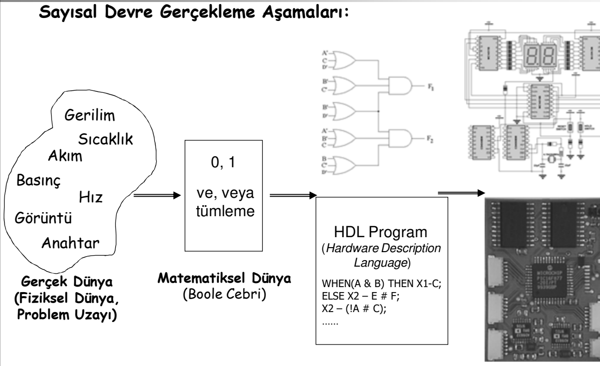
• Esneklik ve programlanabilirlik. Günümüzde sayısal sistemleri programlanabilir bilgisayarlar şeklinde gerçeklemek mümkündür. Bu sayede aynı tasarım yeni gereksinimlere göre yeniden programlanarak tekrar kullanılabilmektedir.

• Sayısal verileri bilgisayar ortamında saklamak ve işlemek mümkündür.

• Sayısal sistemler daha hızlı çalışmaktadır.

• Sayısal sistemler küçülmekte ve ucuzlamaktadır. Sayısal sistemler gelişmeye devam ediyor (ancak gelişme yavaşlamaktadır). Bkz. Bilgisayar Mimarisi ders notları.

Sayısal Devre Gerçekleme Aşamaları:

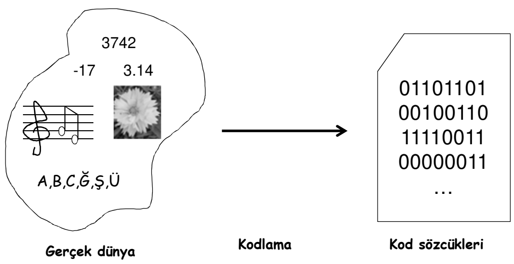


Modelleme, soyutlama → Tasarım → Gerçekleme
(Kodlama)

İkili Sayısal Kodlama (*Binary Digital Coding*):

Sayısal sistemler ikili sayısal işaretler üzerinde işlemler yaptıklarından sadece iki farklı değeri (*binary*) işleyebilirler; AÇIK-KAPALI, ALÇAK-YÜKSEK, **0-1**.

Bu nedenle sayısal devreler yardımıyla üzerinde işlem yapılacak olan fiziksel büyüklüklere (gerilim, sıcaklık vs.) ve her türlü veriye (harf, sayı, renk, ses) ikili sayılar karşı düşürülür.



Sayısal Kodlama (devamı):

n basamaklı (n bitlik "*Binary digit*") bir ikili sayı kullanarak 2^n tane farklı "şey" ifade edebiliriz.

n bit $\rightarrow$ 2^n farklı "şey"

Örneğin 8 basamaklı (8 bitlik) bir ikili sayı kullanarak 2^8 tane (256) farklı "şey" ifade edebiliriz.

Bunlar 256 farklı renk, 256 sembol, 0 ile 255 arası tamsayılar, 1 ile 256 arası tamsayılar, -128 ile +127 arası tamsayılar olabilir.

00000000, 00000001, 00000010, … , 11111101, 11111110, 11111111.

Farklı türde veriler için farklı **kodlama sistemleri** (yöntemleri) kullanılır.

Bir ikili değerin (Örneğin 10001101) ne anlama geldiğine o değeri kullanacak olan sistem (donanım ya da yazılım sistemi olabilir) ya da kişi belirler.

Bu değer bir sayı da olabilir, bir renk de.

Özellikle sayıların kodlanması büyük önem taşır.

Bu nedenle derste sayıların kodlanmasına ilişkin temel bilgiler verilecektir.

BCD (*Binary Coded Decimal*) İkili kodlanmış onlu sayılar:

0-9 arasındaki rakamlara 4 bitlik bir ikili kod karşı düşürülür.

Doğal BCD:

Sayı:	Kod:	Sayı:	Kod:
0:	0000	5:	0101
1:	0001	6:	0110
2:	0010	7:	0111
3:	0011	8:	1000
4:	0100	9:	1001

Örnek:
Sayı: 805
Kod:1000 0000 0101

Artıklı (*redundant*) kodlamadır. Çünkü 4 bit ile 16 farklı kodlama yapılabilmekte, ancak bunlardan sadece 10 tanesi kullanılmaktadır.

BCD sayılar üzerinde işlem yapmak zor olduğundan günümüz bilgisayar-larında sayıları göstermek için bu kodlama kullanılmamaktadır.

Ağırlıklı Kodlama:

Bitlerin konumlarına birer ağırlık verilir.

Doğal ikili kodlama: Sayıların ağırlıklı kodlama ile 2 tabanında gösterilmesidir.

Örneğin: $11010 = 1{\cdot}2^4 + 1{\cdot}2^3 + 0{\cdot}2^2 + 1{\cdot}2^1 + 0{\cdot}2^0 = 26$

Soldaki ilk basamağa yüksek anlamlı bit (*Most Significant Bit – MSB*)

Sondaki basmağa düşük anlamlı bit (*Least Significant Bit – LSB*) denir.

Bilgisayarlarda sayıları göstermek için doğal ikili kodlama kullanılır.

Hamming uzaklığı: n uzunluğundaki iki kod sözcüğünün arasındaki Hamming uzaklığı, o sözcüklerdeki aynı sırada olup değerleri farklı olan bileşenlerin sayısıdır.

Örneğin: 011 ile 101 arasındaki uzaklık 2 dir.

Richard Wesley Hamming (1915-1998) Matematikçi, ABD

Bitişik kodlar (*Adjacent Codes*)**:** Bir birini izleyen sayılara karşı gelen kodlar arasındaki uzaklık 1 ise o kodlama bitişiktir.

Ayrıca son sayı ile ilk sayı arasındaki uzaklık da 1 olursa kod çevrimlidir.

Örnek: Çevrimli bir BCD kodu (doğal BCD'den farklı)

Sayı:	Kod:	Sayı:	Kod:
0:	0000	5:	1110
1:	0001	6:	1010
2:	0011	7:	1000
3:	0010	8:	1100
4:	0110	9:	0100

Gray Kodu: 2^n elemanlı bir küme için 2 tabanında **artıksız** ve **çevrimli** (*cyclic*) bir kodlama yapılırsa **gray kodu** elde edilir.

Örnek: 2 bitlik bir Gray kodu:

Sayı:	Kod:
0:	00
1:	01
2:	11
3:	10

Gray Kodunun patenti 1953'te Bell Laboratuvarında çalışan fizikçi *Frank Gray* tarafından alınmıştır.

Sayıların Bilgisayarda (Sayısal Sistemlerde) Gösterilimi

Bu derste tamsayıların gösterilimine ilişkin bilgiler verilecektir.

Kayan noktalı (*floating point*) sayıların gösterilimi Bilgisayar Mimarisi dersinde ele alınmaktadır (Bkz. http://www.buzluca.info/dersler.html).

Sayılar kodlanmadan önce **işaretsiz** ya da **işaretli** sayılarla çalışılacağı belirlenmelidir. Çünkü işaretsiz ve işaretli sayıların kodlanmasında farklı yöntemler kullanılmaktadır.

İşaretsiz (*Unsigned*) Tamsayıların Kodlanması:

Bilgisayarlarda işaretsiz tamsayıların ifade edilmesinde "doğal ağırlıklı ikili kodlama" kullanılır.

Örnek: $215_{10} = (1101\ 0111)_2 = 1{\cdot}2^7 + 1{\cdot}2^6 + 0{\cdot}2^5 + 1{\cdot}2^4 + 0{\cdot}2^3 + 1{\cdot}2^2 + 1{\cdot}2^1 + 1{\cdot}2^0$

215/2 = 107 kalan 1 (düşük anlamlı bit "*Least Significant Bit – LSB*") son basmak
107/2 = 53 kalan 1
53/2 = 26 kalan 1
26/2 = 13 kalan 0
13/2 = 6 kalan 1
6/2 = 3 kalan 0
3/2 = 1 kalan 1
1/2 = 0 kalan 1 (yüksek anlamlı bit "*Most Significant Bit – MSB*") ilk basamak

8 bit ile ifade edilebilecek **en büyük işaretsiz** tamsayı: $1111\ 1111_2 = 255_{10}$
8 bit ile ifade edilebilecek **en küçük işaretsiz** tamsayı: $0000\ 0000_2 = 0_{10}$

İşaretli (*Signed*) Tamsayıların Kodlanması:

Pozitif ve negatif sayıları ayırt etmek için ikili sayının ilk basamaktaki en yüksek anlamlı bitine bakılır.

- "**0**" ile başlayan sayıların **pozitif**,
- "**1**" ile başlayan sayıların **negatif** olduğu kabul edilir.

Pozitif tamsayılar:

Pozitif sayıların kodlanmasında (işaretsiz sayılarda olduğu gibi) "doğal ağırlıklı ikili kodlama" kullanılır.

Dikkat edilmesi gereken nokta sayının 0 ile başlamasıdır.

Buna göre 8 bit ile temsil edilebilecek pozitif işaretli sayılar:

0000 0000 ile **0**111 1111 arasında (yani 0 ile +127 arasında) değişecektir.

Pozitif Tamsayı Örnekleri:

8 bit $+5_{10}$: **0**000 0101
8 bit $+100_{10}$: **0**110 0100
4 bit $+5_{10}$: **0**101
4 bit $+7_{10}$: **0**111

Negatif Tamsayılar:

Negatif sayıların kodlanmasında **2'ye tümleme** (*2's complement*) yöntemi kullanılmaktadır.

Bu yöntemde pozitif bir sayının 2'ye tümleyeni hesaplandığında o sayının negatif gösterilimi elde edilmiş olur.

Bir sayının **2'ye tümleyenini elde etmek** için

- Önce sayı 1'e tümlenir, yani 0'lar 1, 1'ler 0 yapılır,
- 1'e tümlenmiş sayıya 1 eklenir.

2'ye tümleyen $(A) = \bar{A} + 1$ $\bar{A}$, 1'e tümleyeni göstermektedir.

2'ye tümleme yöntemi aritmetik işlemlerde kolaylık sağladığı için tercih edilmektedir.

Bu kodlama yönteminin kullanılması aritmetik işlemler için daha basit (ucuz) devreler tasarlanmasını sağlar.

Negatif Sayılara Örnekler:

8 bitlik $+5_{10}$: **0**000 0101
1'e tümleme : 1111 1010
1 ekleme : + 1
Sonuç -5_{10} : **1**111 1011

4 bitlik $+7_{10}$: **0**111
1'e tümleme : 1000
1 ekleme : + 1
Sonuç -7_{10} : **1**001

2'ye tümleme yöntemi (devamı) :

2'ye tümleme işlemi bir sayının **işaretini değiştirir**.

Bir negatif sayıya 2'ye tümleme işlemi uygulandığında o sayının pozitif değeri elde edilmiş olur.

2'ye tümleme işlemi:

pozitif $\rightarrow$ negatif
negatif $\rightarrow$ pozitif

Örnek: Negatif bir sayının pozitif yapılması:

8 bitlik -5_{10} : **1**111 1011
1'e tümleme : 0000 0100
1 ekleme : + 1
Sonuç $+5_{10}$: **0**000 0101

İşaretli sayıların bir çember grafik üzerinde gösterilmesi:

Aşağıda 4 bitlik sayılar gösterilmiştir.

4 bit ile ifade edilebilecek mutlak değeri en büyük negatif tamsayı: 1000 = - 8
Mutlak değeri en küçük 4 bitlik negatif tamsayı: 1111 = -1

8 bit mutlak değeri en büyük negatif tamsayı: 1000 0000 = -128
Mutlak değeri en küçük 4 bitlik negatif tamsayı: 1111 1111 = -1

İkili Sayıların Uzatılması (*Sign Extension*)

Sayısal sistemlerde ikili sayılar için belli uzunlukta yerler (bellek gözleri) ayrılır.

Bazı durumlarda daha az bit ile ifade edilebilen bir sayıyı daha büyük bir yere yazmak ya da daha uzun bir sayı ile işleme sokmak gerekebilir.

Bu durumda kısa olan sayı uzatılır.

Örneğin 4 bitten 8 bite veya 8 bitten 16 bite uzatma.

Uzatma işleminde sayının işaretsiz ya da işaretli olmasına göre farklı yollar izlenir.

İşaretsiz Sayılar: Sayının başına (yüksek anlamlı kısmına) gerektiği kadar sıfır '0' eklenir.

Örnek: 4 bitlik 3_{10}: 0011 8 bitlik 3_{10}: 0000 0011
Örnek: 4 bitlik 9_{10}: 1001 8 bitlik 9_{10}: 0000 1001

İşaretli Sayılar: Sayının başına (yüksek anlamlı kısmına) **sayının işareti** gerektiği kadar eklenir. Buna **işaret uzatma** (*sign extension*) denir.

Örnek: 4 bitlik $+3_{10}$ = 0011 8 bitlik $+3_{10}$ = 0000 0011
Örnek: 4 bitlik -7_{10} = 1001 8 bitlik -7_{10} = 1111 1001
Örnek = 4 bitlik -1_{10} = 1111 8 bitlik -1_{10} = 1111 1111

16 Tabanının (*Base-16*) Kullanılması

Sayısal devrelerin yapıları 2'li sayıların kullanılmasını zorunlu hale getirmiştir.

Ancak 2'li sayıların yazılması ve okunması uzunlukları nedeniyle zor olmaktadır.

Bu nedenle kağıt üstündeki gösterilimlerde kolaylık sağladığı için 16 tabanında (*hexadecimal*) sayılar kullanılmaktadır.

2'li – 16'lı Dönüşüm:

- 2'li sayı 4 bitlik gruplar halinde yazılır,
- Her dörtlü için 16'lık karşılığı yazılır.

Örnek:

01011101_2 = 0101 1101 (İkili - *Binary*)
= 5 D (Onaltılı - *Hexadecimal*)

10 tabanına dönüşüm :

$5D_{16}$ = (5 x 16) + 13 = 93

Sonuç: $01011101_2 = 5D_{16} = 93_{10}$

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

16 tabanındaki sayıları göstermek için genellikle **$** ve **h** simgeleri kullanılır.
Örnek: $5D veya 5Dh.

Sayısal Sistemlerde Toplama ve Çıkarma İşlemleri

Bilgisayarlarda tamsayı aritmetik işlemleri Aritmetik/Lojik Birim (ALB) tarafından yapılır (*Arithmetic Logic Unit – ALU*).

Tamsayı toplama ve çıkarma işlemleri işaretsiz ve işaretli sayılar üzerinde aynı şekilde yapılır (2'ye tümleyen yönteminin bir yararı).

Ancak çıkan sonucun yorumlanması işaretsiz ve işaretli sayılarda farklı olmaktadır.

Farklı uzunluklarda sayılar ile işlem yaparken kısa sayının uzatılması gerekir. Uzatma işlemi işaretsiz ve işaretli sayılarda farklı olur. (Bkz. 1.17)

Toplama:

İşaretsiz Tamsayılar:

n bitlik iki işaretsiz sayının toplanması sonucu n+1 bitlik bir sayı oluşabilir. (n+1). bit **elde** (*carry*) adını alır.

a b	Elde	Sonuç
0 0	0	0
0 1	0	1
1 0	0	1
1 1	1	0

Örnekler: 8 bitlik işaretsiz sayıların toplanması

```
  01110101 : 117
+ 01100011 :  99   <- Sadece sağlama için
  11011000 : 216
```

Elde oluşmadı. (Elde=0)

```
  11111111 : 255
+ 00000001 :   1   <- Sadece sağlama için
 100000000 : 256
```

Elde oluştu. (Elde=1)

Toplama:

İşaretli Tamsayılar:

- İşlem işaretsiz sayılarda olduğu gibi yapılır. Ancak sonucun yorumlanması farklıdır.
- Toplanan sayıların işaretleri farklı olsa da ek bir işlem yapmaya gerek yoktur (2'ye tümleme yönteminin bir yararı).
- n bitlik işaretli iki sayının toplanması sonucu (n+1). bit oluşursa bu bit göz ardı edilir.

Örnek: 8 bitlik işaretli sayıların toplanması

```
  11111111:  −1
+ 00000001:  +1
 100000000:   0
```

Göz ardı edilir. İşaret (+)

```
  11111111:  −1
+ 11111111:  −1
 111111110:  −2
```

Göz ardı edilir. İşaret (-)

Dikkat:

- Eğer n bitlik sayılarla çalışılıyorsa işaret her zaman en yüksek anlamlı bit (sağdan sola doğru sayıldığında) n. bittir (n+1. değil).
- (n+1). elde (*carry*) bitidir.

Taşma (*Overflow*) (işaretli tamsayılar):

İşaretli sayılarda toplama sonucu oluşan değer n bit ile gösterilemeyebilir.

Örneğin 8 bit ile gösterilebilecek sayılar -128, +127 arasındadır. Oluşan sonuç bu aralığın dışına çıkıyorsa **taşma** oluşur.

Toplama sonucunda taşma oluştuğu toplanan sayıların ve sonucun işaretinden anlaşılır.

Toplamada iki durumda taşma oluşabilir:

poz + poz $\rightarrow$ neg ve neg + neg $\rightarrow$ poz

Örnek:

```
  01111111:+127
+ 00000010:  +2
  10000001: Gösterilemiyor
```

İki pozitif sayı toplandı.
Sonuç negatif çıktı.
Taşma vardır.
Not: n+1. bit oluşmadı.
Bu bit göz ardı edilir.

```
  10000000:−128
+ 11111111:  −1
 101111111: Gösterilemiyor
```

İki negatif sayı toplandı.
Sonuç pozitif çıktı.
Taşma vardır.
Not: n+1. bit oluştu.
Bu bit göz ardı edilir.

Çıkarma:

- Bilgisayarlar, çıkarma işlemi için 2'ye tümele yönteminden yararlanırlar.
- Çıkartılacak olan sayının işareti değiştirilip (2'ye tümlenip) birinci sayı ile toplanır.

$$\begin{aligned} \boldsymbol{A}-\boldsymbol{B} &= \boldsymbol{A}+(-\boldsymbol{B}) \\ &= \boldsymbol{A}+\boldsymbol{2'ye\ t\ddot{u}mleme}\ (\boldsymbol{B}) \\ &= \boldsymbol{A}+\overline{\boldsymbol{B}}+\boldsymbol{1} \end{aligned}$$

2'ye tümleme kullanıldığından tek bir toplama devresi ile hem toplama hem çıkarma yapmak mümkün olur.

Toplama ve çıkarma devreleri 5. Bölümde ele alınacaktır.

Toplama işleminde olduğu gibi, çıkarma işlemleri de işaretsiz ve işaretli sayılar üzerinde aynı şekilde yapılır (2'ye tümleme nedeniyle).
Ancak çıkan sonucun yorumlanması işaretsiz ve işaretli sayılarda farklı olmaktadır.

Çıkarma (devamı):

İşaretsiz Tamsayılar:

n bitlik iki işaretsiz sayı arasında 2'ye tümleyen yöntemine göre çıkarma yapıldığında n+1. bit oluşursa sonuç geçerlidir, borç (*borrow*) yoktur.

Eğer n+1. bit oluşmazsa (sıfırsa) birinci sayı ikinciden küçüktür, **borç** vardır.

Örnekler: 8 bitlik işaretsiz sayılar ile çıkarma

```
  00000101: 5    2'ye tümleme      00000101: 5
− 00000001: 1   ------------->   + 11111111:−1
                                  100000100: 4
```

Elde oluştu : **Borç Yok**

```
  00000001: 1    2'ye tümleme      00000001: 1
− 00000101: 5   ------------->   + 11111011:−5
                                  011111100: Borç(İlk sayı küçüktür)
```

Elde oluşmadı : **Borç Var**

Çıkarma (devamı):

İşaretli Tamsayılar:

İşaretli tamsayılar arasındaki çıkarma da, işaretsizlerde olduğu gibi 2'ye tümleyen yöntemine göre yapılır.

Elde biti göz ardı edilir.

Toplamada olduğu gibi işaretli sayıların çıkarılmasında da taşma olabilir.

Çıkarmada iki durumda taşma oluşabilir:

poz - neg $\rightarrow$ neg ve neg - poz $\rightarrow$ poz

Örnek: 8 bitlik işaretli sayılar ile çıkarma

```
  00000101: 5    2'ye tümleme      00000101:   5
− 00001100:12   ------------->   + 11110100:−12
                                   11111001: 2'ye tüm.:00000111:−7
```

İşaret **1**, sonuç **negatif**

```
  11111101:  −3    2'ye tümleme      11111101:   −3
− 01111111: 127   ------------->   + 10000001:−127
                                    101111110: Gösterilemiyor
```

Neg – poz = poz. **Taşma** vardır.

İşaret biti **0**, Sonuç **pozitif**.

Tamsayıların karşılaştırılması:

Tamsayıların karşılaştırılması için çıkarma işlemi kullanılır

S = A - B, çıkarma işleminden sonra ilgili bayraklar (durum bitleri: elde, taşma) kontrol edilir.

İşaretsiz Tamsayılar:

Borç	Sonuç (S)	Karşılaştırma
X (önemli değil)	=0	A=B
YOK (elde = 1)	≠0	A>B
VAR (elde = 0)	≠0	A<B

İşaretli Tamsayılar :

Taşma	Sonuç (S)	Karşılaştırma
X (önemli değil)	=0	A=B
YOK	Pozitif, ≠0	A>B
YOK	Negatif	A<B
VAR	Pozitif	A<B
VAR	Negatif	A>B

Elde (*Carry*), Borç (*Borrow*), Taşma (*Overflow*) Kavramlarının Özeti

Elde: İşaretsiz sayıların toplanmasında oluşabilir. Sonucun n bite sığmadığını, (n+1). bitin gerekli olduğunu gösterir.

Borç: İşaretsiz sayıların çıkartılmasında oluşabilir. Birinci sayının ikinciden küçük olduğunu, sonucun negatif çıktığını gösterir.

2'ye tümleyen yöntemine göre yapılan çıkarmada n+1. bit oluşursa borç yoktur.

Taşma: Sadece işaretli sayılar üzerinde yapılan toplama ve çıkarma işlemlerinde oluşur. Sonucun, ayrılan bit sayısı ile ifade edilemediğini gösterir.

Taşma olduğu, işleme giren sayıların ve sonucun işareti incelenerek anlaşılır.

Aşağıdaki durumlar oluştuğunda taşma var demektir:

poz + poz $\rightarrow$ neg   poz – neg $\rightarrow$ neg

neg + neg $\rightarrow$ poz   neg – poz $\rightarrow$ poz