

Bölüm 3: Sayı Sistemleri

Mikroişlemciler

Romen rakamları

Onluk, ikili, onaltılık

Taban dönüşümleri

İşaretili sayılar

1'e ve ikiye tümleyen

İkili toplama/çıkarma

IEEE 754 kayan nokta

Flip-flop

BCD

Little-Endian / Big-Endian

Gray kodu

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

Öğrenme Çıktıları

Konumsal sistemler

Onluk, ikili ve onaltılık sayıların değerini basamak ağırlıklarıyla hesaplamak.

Dönüşüm

Sayıları tabanlar arasında bölme ve gruplama yöntemleriyle dönüştürmek.

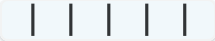
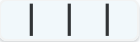
İşaretili sayılar

ikiye tümleyen gösterimini kullanarak negatif sayıları yazmak, toplamak ve çıkarmak.

Kodlamalar

BCD, IEEE 754, Gray kodu ve bayt sıralamasının (endian) bellekte nasıl görüldüğünü açıklamak.

Sayıları Temsil Etme Yöntemleri

- Sayısal bir değeri temsil etmenin birçok yolu vardır.
- İnsanlar uzun zaman önce saymak için **sopaları** kullandılar; sonra sopaların resimlerini çizmeyi ve kâğıda geçirmeyi öğrendiler.
- Bu yüzden 5 sayısı önce  şeklinde temsil edildi.
- Roma İmparatorluğu'nda  hâlâ üç sopayı temsil ederken, **V** beş, **X** on sopayı temsil etmeye başladı.
- Bugün kullandığımız **konumsal** sistemde bir rakamın değeri, bulunduğu basamağa da bağlıdır.



5 sopa



III = 3



V = 5



X = 10



konumsal: $7 \cdot 100 + 5 \cdot 10 + 4$

Romen Rakamları

Simge	Değer
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

- Antik Roma İmparatorluğu'nda kullanılan bir sayı sistemidir.
- **Toplama** ve **çıkarma** üzerine kuruludur: büyük simgeden sonra gelen küçük simge eklenir (VI = 6), önce gelen çıkarılır (IV = 4).
- Çıkarma yalnızca şu çiftlerde yapılır: IV, IX, XL, XC, CD, CM.
- **Sıfır** ve **basamak** kavramları yoktur; bu yüzden toplama, çarpma gibi işlemler çok zahmetlidir.

Romen Rakamlarını Okumak

MCMLXXXVI

M

CM

LXXX

VI

Sayı soldan sağa, değer gruplarına ayrılır.

Adım 1 / 6

MMVII

M

M

VII

1000 + 1000 + 7 = 2007

Dikkat: MMVI = 2006, **MMVII** = 1000 + 1000 + 7 = 2007.

Romen Rakamlarını Okumak

MCMLXXXVI

M

1000

CM

LXXX

VI

M = 1000.

Adım 2 / 6

MMVII

M

1000

M

1000

VII

7

= 2007

Dikkat: MMVI = 2006, **MMVII** = 1000 + 1000 + 7 = 2007.

Romen Rakamlarını Okumak

MCMLXXXVI

M	CM	LXXX	VI
1000	+	900	

CM: C, M'den önce geldiği için çıkarılır $\rightarrow 1000 - 100 = 900$.

Adım 3 / 6

MMVII

M	M	VII	
1000	+	1000	+
		7	= 2007

Dikkat: MMVI = 2006, **MMVII** = 1000 + 1000 + 7 = 2007.

Romen Rakamlarını Okumak

MCMLXXXVI



1000 + 900 + 80

$$\text{LXXX} = 50 + 10 + 10 + 10 = 80.$$

Adım 4 / 6

MMVII



1000 + 1000 + 7 = 2007

Dikkat: MMVI = 2006, **MMVII** = 1000 + 1000 + 7 = 2007.

Romen Rakamlarını Okumak

MCMLXXXVI

M	CM	LXXX	VI			
1000	+	900	+	80	+	6

$$\text{VI} = 5 + 1 = 6.$$

Adım 5 / 6

MMVII

M	M	VII			
1000	+	1000	+	7	= 2007

Dikkat: MMVI = 2006, **MMVII** = 1000 + 1000 + 7 = 2007.

Romen Rakamlarını Okumak

MCMLXXXVI

M	+	CM	+	LXXX	+	VI	=	1986
1000		900		80		6		

Toplam $1000 + 900 + 80 + 6 = 1986$.

Adım 6 / 6

MMVII

M	+	M	+	VII	=	2007
1000		1000		7		

Dikkat: MMVI = 2006, **MMVII** = $1000 + 1000 + 7 = 2007$.

Onluk Sistem

- Bugün çoğu insan saymak için **onluk** (10 tabanlı) temsili kullanır.
- Onluk sistemde 10 rakam vardır: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Bu rakamlar herhangi bir değeri temsil edebilir, örneğin **754**.
- Bir sayının değeri, her rakamın **tabanın, rakamın konumundaki üssüyle** çarpımlarının toplamıdır.

7	5	4
10^2	10^1	10^0
700	50	4

$700 + 50 + 4 = 754$

Genel formül: $d_{n-1}...d_1d_0 = \sum d_i \times b^i$, burada b taban, i basamağın konumudur (sağdan, 0'dan başlar).

Konumun Önemi

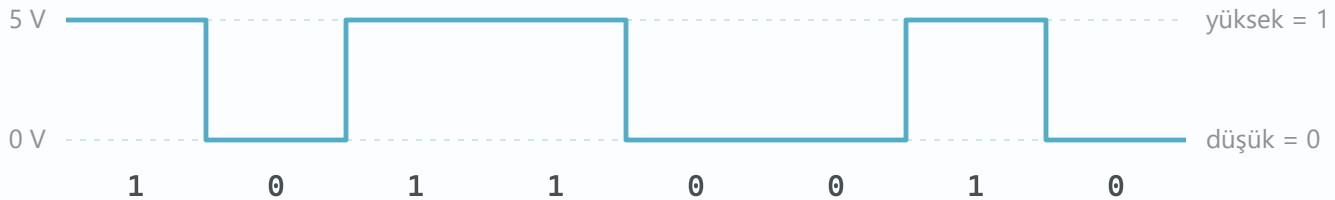
7	5	4
10^2	10^1	10^0
700	50	4
754		

5	4	7
10^2	10^1	10^0
500	40	7
547		

- Her rakamın **konumu önemlidir**: 7 sona konursa 547 olur, bambaşka bir değer.
- Herhangi bir sayının sıfıncı kuvveti 1'dir: $x^0 = 1$. Bu yüzden en sağdaki basamak her tabanda "birler" basamağıdır.
- 0^0 matematikte belirsiz kabul edilir; sayı sistemleri ve polinomlarda **uzlaşım olarak 1** alınır.
- Romen sisteminden farkı: aynı rakam (7) konumuna göre 700, 70 ya da 7 değerini alır; boş basamak için **0** gerekir.

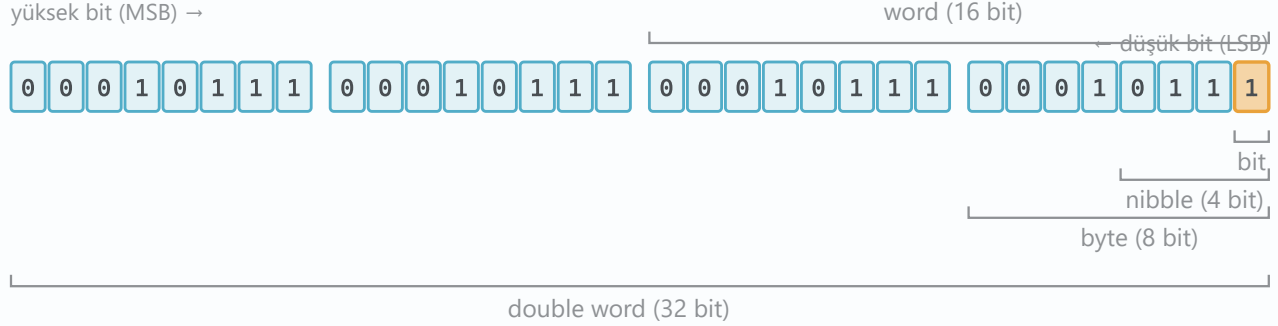
İkili Sistem

- Bilgisayarlar insanlar kadar akıllı değildir; elektronik bir makineyi **iki durumlu** yapmak kolaydır: açık/kapalı, yüksek/düşük gerilim, 1/0.
- Bu yüzden bilgisayarlar **ikili** (binary) sistemi kullanır.
- İkili sistemde 2 rakam vardır: **0** ve **1**; taban 2'dir.
- Basamak ağırlıkları 2'nin kuvvetleridir: 1, 2, 4, 8, 16, 32, 64, 128 ...



KONUMSAL SİSTEMLER

Bit, Nibble, Byte, Word



Birim	Bit	İşaretsiz aralık	8086'da
bit	1	0 ... 1	bayrak
nibble	4	0 ... 15 (0h ... Fh)	bir onaltılık rakam, bir BCD basamağı
byte	8	0 ... 255	AL, BL ..., DB
word	16	0 ... 65535	AX, BX ..., DW
double word	32	0 ... 4 294 967 295	DX:AX, DD

İkili sayıdaki her basamağa BIT (binary digit) denir. n bit ile 2^n farklı değer gösterilebilir.

İkili Sayıları Okumak

- Assembly'de ikili sayılar genellikle **b** harfiyle sonlandırılır: `101b` = 5.
- Örnek: `10100101b` , onluk **165**'e eşittir.

1	0	1	0	0	1	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	0	32	0	0	4	0	1
$128 + 32 + 4 + 1 = 165$							

Kısa yol: yalnızca **1 olan** basamakların ağırlıklarını topla. 8 bitlik ağırlıklar: $128 \cdot 64 \cdot 32 \cdot 16 \cdot 8 \cdot 4 \cdot 2 \cdot 1$.

İkiliden Onluğa: 1111011b



$(1111011)_2$: 7 basamak, ağırlıklar $2^0 \dots 2^6$.

Adım 1 / 8

$$(1111011)_2 = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 = 1 + 2 + 0 + 8 + 16 + 32 + 64 =$$

123

İkiliden Onluğa: 1111011b

1	1	1	1	0	1	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	0	2	1

toplam = 1

Basamak 0: $1 \times 2^0 = 1 \rightarrow$ ara toplam **1**.

Adım 2 / 8

$$(1111011)_2 = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 = 1 + 2 + 0 + 8 + 16 + 32 + 64 =$$

123

İkiliden Onluğa: 1111011b

1	1	1	1	0	1	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	0	2	1

toplam = 3

Basamak 1: $1 \times 2^1 = 2 \rightarrow$ ara toplam **3**.

Adım 3 / 8

$$(1111011)_2 = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 = 1 + 2 + 0 + 8 + 16 + 32 + 64 =$$

123

İkiliden Onluğa: 1111011b

1	1	1	1	0	1	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	0	2	1

toplam = 3

Basamak 2: $0 \times 2^2 = 0 \rightarrow$ ara toplam **3**.

Adım 4 / 8

$$(1111011)_2 = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 = 1 + 2 + 0 + 8 + 16 + 32 + 64 =$$

123

İkiliden Onluğa: 1111011b

1	1	1	1	0	1	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	0	2	1

toplam = 11

Basamak 3: $1 \times 2^3 = 8 \rightarrow$ ara toplam **11**.

Adım 5 / 8

$$(1111011)_2 = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 = 1 + 2 + 0 + 8 + 16 + 32 + 64 =$$

123

İkiliden Onluğa: 1111011b

1	1	1	1	0	1	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	0	2	1

toplam = 27

Basamak 4: $1 \times 2^4 = 16 \rightarrow$ ara toplam **27**.

Adım 6 / 8

$$(1111011)_2 = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 = 1 + 2 + 0 + 8 + 16 + 32 + 64 =$$

123

İkiliden Onluğa: 1111011b

1	1	1	1	0	1	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	0	2	1

toplam = 59

Basamak 5: $1 \times 2^5 = 32 \rightarrow$ ara toplam **59**.

Adım 7 / 8

$$(1111011)_2 = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 = 1 + 2 + 0 + 8 + 16 + 32 + 64 =$$

123

İkiliden Onluğa: 1111011b

1	1	1	1	0	1	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	0	2	1

toplam = 123

Basamak 6: $1 \times 2^6 = 64 \rightarrow$ ara toplam **123**.

Adım 8 / 8

$$(1111011)_2 = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 = 1 + 2 + 0 + 8 + 16 + 32 + 64 = \mathbf{123}$$

Onaltılık Sistem

10'luk	2'lik	16'lık
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

10'luk	2'lik	16'lık
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

- Onaltılık (hexadecimal) sistem 16 rakam kullanır: 0–9 ve **A, B, C, D, E, F** (10–15).
- Bir onaltılık rakam tam olarak **4 bite** (bir nibble) karşılık gelir.
- Bu yüzden onaltılık sayılar ikiliye göre **kompakt** ve okunması kolaydır: 1010 0101b = A5h.

Onaltılık Sayıları Yazmak

- Onaltılık sayılar genellikle **h** ile sonlandırılır: `5Fh` = $5 \times 16 + 15 = 95$.
- Harfle (A...F) başlayan sayıların başına **0** eklenir: `0E120h`. Aksi hâlde assembler `E120h` 'yi bir değişken/etiket adı sanır.
- Örnek: `1234h`, onluk **4660**'a eşittir.

1	2	3	4
16^3	16^2	16^1	16^0
4096	512	48	4
$4096 + 512 + 48 + 4 = 4660$			

Yazım	Anlamı
<code>MOV AL, 0FFh</code>	AL = 255 ✓
<code>MOV AL, FFh</code>	hata: FFh tanımsız bir ad
<code>MOV AX, 10</code>	onluk 10 (= 0Ah) — sonek yoksa onluktur

Onaltılıktan Onluğa: A12Fh

A	1	2	F
16^3	16^2	16^1	16^0

$(A12F)_{16}$: A = 10, F = 15. Ağırlıklar $16^0 = 1$, $16^1 = 16$, $16^2 = 256$, $16^3 = 4096$.

Adım 1 / 5

$$(A12F)_{16} = 15 + 32 + 256 + 40960 = \mathbf{41263}$$

Onaltılıktan Onluğa: A12Fh

A	1	2	F
16^3	16^2	16^1	16^0
40960	256	32	15

toplam = 15

$$F \times 16^0 = 15 \times 1 = \mathbf{15}.$$

Adım 2 / 5

$$(A12F)_{16} = 15 + 32 + 256 + 40960 = \mathbf{41263}$$

Onaltılıktan Onluğa: A12Fh

A	1	2	F
16^3	16^2	16^1	16^0
40960	256	32	15

toplam = 47

$$2 \times 16^1 = 2 \times 16 = \mathbf{32} \rightarrow 47.$$

Adım 3 / 5

$$(A12F)_{16} = 15 + 32 + 256 + 40960 = \mathbf{41263}$$

Onaltılıktan Onluğa: A12Fh

A	1	2	F
16^3	16^2	16^1	16^0
40960	256	32	15

toplam = 303

$$1 \times 16^2 = 1 \times 256 = \mathbf{256} \rightarrow 303.$$

Adım 4 / 5

$$(A12F)_{16} = 15 + 32 + 256 + 40960 = \mathbf{41263}$$

Onaltılıktan Onluğa: A12Fh

A	1	2	F
16^3	16^2	16^1	16^0
40960	256	32	15

toplam = 41263

$$A \times 16^3 = 10 \times 4096 = \mathbf{40960} \rightarrow \mathbf{41263}.$$

Adım 5 / 5

$$(A12F)_{16} = 15 + 32 + 256 + 40960 = \mathbf{41263}$$

İkili ↔ Onaltılık: Dörtlü Gruplama

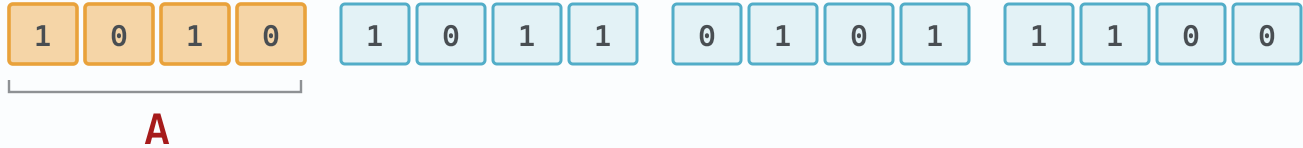


İkili sayı sağdan başlayarak **4'lü gruplara** ayrılır (gerekirse sola 0 eklenir).

Adım 1 / 5

- Ters yön de aynı: her onaltılık rakam 4 bite açılır: **3Eh** → 0011 1110b.
- Örnek: **1101011b** → 0110 1011 → **6Bh** (soldaki gruba 0 eklendi).
- Bu yüzden programcılar ikili veriyi hep onaltılık yazar: bir bayt = tam 2 onaltılık rakam.

İkili ↔ Onaltılık: Dörtlü Gruplama

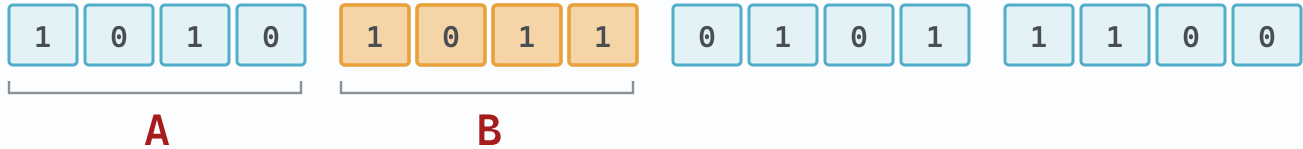


$$1010 = 8 + 2 = 10 \rightarrow \mathbf{A}.$$

Adım 2 / 5

- Ters yön de aynı: her onaltılık rakam 4 bite açılır: **3Eh** → 0011 1110b.
- Örnek: **1101011b** → 0110 1011 → **6Bh** (soldaki gruba 0 eklendi).
- Bu yüzden programcılar ikili veriyi hep onaltılık yazar: bir bayt = tam 2 onaltılık rakam.

İkili ↔ Onaltılık: Dörtlü Gruplama

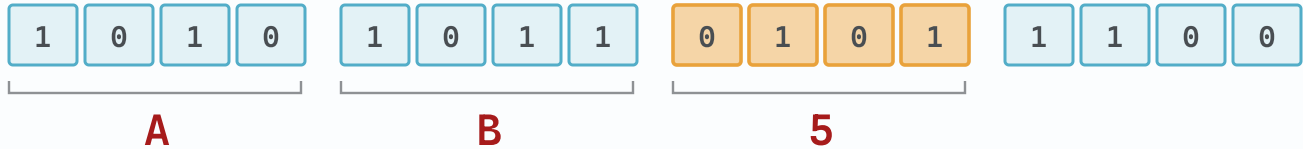


$$1011 = 8 + 2 + 1 = 11 \rightarrow \mathbf{B}.$$

Adım 3 / 5

- Ters yön de aynı: her onaltılık rakam 4 bite açılır: **3Eh** → 0011 1110b.
- Örnek: **1101011b** → 0110 1011 → **6Bh** (soldaki gruba 0 eklendi).
- Bu yüzden programcılar ikili veriyi hep onaltılık yazar: bir bayt = tam 2 onaltılık rakam.

İkili ↔ Onaltılık: Dörtlü Gruplama

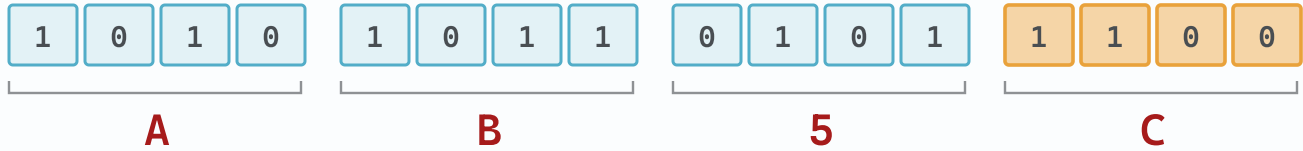


$$0101 = 4 + 1 = 5 \rightarrow \mathbf{5}.$$

Adım 4 / 5

- Ters yön de aynı: her onaltılık rakam 4 bite açılır: **3Eh** → 0011 1110b.
- Örnek: **1101011b** → 0110 1011 → **6Bh** (soldaki gruba 0 eklendi).
- Bu yüzden programcılar ikili veriyi hep onaltılık yazar: bir bayt = tam 2 onaltılık rakam.

İkili ↔ Onaltılık: Dörtlü Gruplama



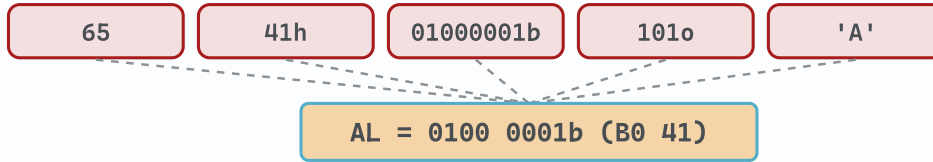
$1100 = 8 + 4 = 12 \rightarrow \text{C}$. Sonuç: **0AB5Ch**.

Adım 5 / 5

- Ters yön de aynı: her onaltılık rakam 4 bite açılır: **3Eh** → 0011 1110b.
- Örnek: **1101011b** → 0110 1011 → **6Bh** (soldaki gruba 0 eklendi).
- Bu yüzden programcılar ikili veriyi hep onaltılık yazar: bir bayt = tam 2 onaltılık rakam.

Assembly'de Sayı Gösterimleri

Yazım	Taban	Değer
MOV AL, 65	onluk (sonek yok ya da <code>d</code>)	65
MOV AL, 41h	onaltılık	65
MOV AL, 01000001b	ikili	65
MOV AL, 101o	sekizli (octal, <code>o</code> ya da <code>q</code>)	65
MOV AL, 'A'	ASCII karakter	65
MOV AL, -1	onluk, işaretli	FFh



Hepsi aynı makine kodunu üretir: B0 41 . Taban yalnızca **insanlar** için bir yazım biçimidir; bellekte her şey ikilidir.

Onluktan Diğer Tabanlara

- Onluk değer istenen tabana **bölünür**.
- Her seferinde **bölüm saklanır, kalan alınır** (kalan bir basamaktır, $0 \dots \text{taban} - 1$).
- Bölüm sıfır olana kadar bölmeye devam edilir.
- Kalanlar **sondan başa** (aşağıdan yukarı) yazılınca sonuç elde edilir: ilk kalan en düşük basamaktır.

Örnek: 39'u onaltılığa çevirelim.

Onluktan Diğer Tabanlara

- Onluk değer istenen tabana **bölünür**.
- Her seferinde **bölüm saklanır, kalan alınır** (kalan bir basamaktır, 0 ... taban-1).
- Bölüm sıfır olana kadar bölmeye devam edilir.
- Kalanlar **sondan başa** (aşağıdan yukarı) yazılınca sonuç elde edilir: ilk kalan en düşük basamaktır.

$$39 \div 16 = 2$$

kalan

7

$39 \div 16 = 2$, kalan **7** (birler basamağı).

Adım 2 / 4

Onluktan Diğer Tabanlara

- Onluk değer istenen tabana **bölünür**.
- Her seferinde **bölüm saklanır, kalan alınır** (kalan bir basamaktır, 0 ... taban-1).
- Bölüm sıfır olana kadar bölmeye devam edilir.
- Kalanlar **sondan başa** (aşağıdan yukarı) yazılınca sonuç elde edilir: ilk kalan en düşük basamaktır.

$$39 \div 16 = 2$$

kalan

7

$$2 \div 16 = 0$$

kalan

2

$2 \div 16 = 0$, kalan **2**. Bölüm 0 → dur.

Onluktan Diğer Tabanlara

- Onluk değer istenen tabana **bölünür**.
- Her seferinde **bölüm saklanır, kalan alınır** (kalan bir basamaktır, 0 ... taban-1).
- Bölüm sıfır olana kadar bölmeye devam edilir.
- Kalanlar **sondan başa** (aşağıdan yukarı) yazılınca sonuç elde edilir: ilk kalan en düşük basamaktır.

$$39 \div 16 = 2$$

$$2 \div 16 = 0$$

kalan

7

kalan

2

aşağıdan yukarı oku

27h

Kalanlar aşağıdan yukarı: **27h**. Kontrol: $2 \times 16 + 7 = 39 \checkmark$

43868 → Onaltılık

43868 sayısı onaltılık sistemde nasıl ifade edilir?

Adım 1 / 6

43868 → Onaltılık

$43868 \div 16 = 2741$ kalan **12=C**

$43868 \div 16 = 2741$, kalan 12 → **C**.

43868 → Onaltılık

43868 ÷ 16 = 2741

kalan 12=C

2741 ÷ 16 = 171

kalan 5

2741 ÷ 16 = 171, kalan 5.

TABAN DÖNÜŞÜMLERİ · ADIM ADIM

43868 → Onaltılık

$$43868 \div 16 = 2741$$

kalan

12=C

$$2741 \div 16 = 171$$

kalan

5

$$171 \div 16 = 10$$

kalan

11=B

$$171 \div 16 = 10, \text{ kalan } 11 \rightarrow \mathbf{B}.$$

Adım 4 / 6

TABAN DÖNÜŞÜMLERİ · ADIM ADIM

43868 → Onaltılık

$$43868 \div 16 = 2741$$

kalan

12=C

$$2741 \div 16 = 171$$

kalan

5

$$171 \div 16 = 10$$

kalan

11=B

$$10 \div 16 = 0$$

kalan

10=A

$10 \div 16 = 0$, kalan 10 → **A**.

Adım 5 / 6

43868 → Onaltılık

$$43868 \div 16 = 2741$$

kalan

12=C

$$2741 \div 16 = 171$$

kalan

5

$$171 \div 16 = 10$$

kalan

11=B

$$10 \div 16 = 0$$

kalan

10=A

aşağıdan yukarı oku

0AB5Ch

Sonuç **0AB5Ch** (A ile başladığı için başa 0). Kontrol: $10 \cdot 4096 + 11 \cdot 256 + 5 \cdot 16 + 12 = 43868$

✓

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

AX **0BADh**

DX **0000h**

SI **0000h**

kalan dizisi

00

00

00

00

+0 +1 +2 +3

BX = **0000h**

AX = 2989 = **0BADh**. Sayının onaltılık basamakları tekrar tekrar 16'ya bölünerek bulunacak.

Adım 1 / 13

DEĞİŞKENLER

AX = **0BADh**

DX = **0000h**

BX = **0000h**

SI = **0**

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

AX 0BADh

DX 0000h

SI 0000h

kalan dizisi

00 00 00 00 +0 +1 +2 +3

BX = 0010h (16)

BX = 16: hedef taban. BX = 2 ya da 8 olsaydı aynı kod o tabanın basamaklarını üretirdi.

Adım 2 / 13

DEĞİŞKENLER

AX = 0BADh

DX = 0000h

BX = 0010h

SI = 0

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

AX **0BADh**

DX **0000h**

SI **0000h**

kalan dizisi

00 **00** **00** **00** +0 +1 +2 +3

BX = 0010h (16)

SI = 0: sıradaki kalanın **kalan** dizisindeki yeri.

Adım 3 / 13

DEĞİŞKENLER

AX = **0BADh**

DX = **0000h**

BX = **0010h**

SI = **0**

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

AX **0BADh**

DX **0000h**

SI **0000h**

kalan dizisi

00 **00** **00** **00** +0 +1 +2 +3

BX = 0010h (16)

DX = 0. 16 bitlik **DIV BX** 32 bitlik **DX:AX**'i böler; DX sıfırlanmasaydı bölünen yanlış olurdu.

Adım 4 / 13

DEĞİŞKENLER

AX = **0BADh**

DX = **0000h**

BX = **0010h**

SI = **0**

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 ► DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

2989 ÷ 16 = 186 kalan 13 **D**

AX 00BAh

DX 000Dh

SI 0000h

kalan dizisi

00 00 00 00 +0 +1 +2 +3

BX = 0010h (16)

2989 ÷ 16 = **186**, kalan **13**: AX = 00BAh (bölüm), DX = 000Dh (kalan). İlk kalan en düşük basamaktır: **D**.

Adım 5 / 13

DEĞİŞKENLER

AX = 00BAh

DX = 000Dh

BX = 0010h

SI = 0

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

$2989 \div 16 = 186$ kalan 13 **D**

AX 00BAh

DX 000Dh

SI 0000h

kalan dizisi

0D 00 00 00 +0 +1 +2 +3

BX = 0010h (16)

kalan[0] ← DL = 0Dh.

Adım 6 / 13

DEĞİŞKENLER

AX = 00BAh

DX = 000Dh

BX = 0010h

SI = 0

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

$$2989 \div 16 = 186 \quad \text{kalan } 13 \quad \mathbf{D}$$

AX 00BAh

DX 000Dh

SI 0001h

kalan dizisi

0D 00 00 00 +0 +1 +2 +3

BX = 0010h (16)

SI = 1; bölüm 186 $\neq$ 0 $\rightarrow$ ZF = 0, **JNE** bol'a döner.

Adım 7 / 13

DEĞİŞKENLER

AX = 00BAh

DX = 000Dh

BX = 0010h

SI = 1

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 ► DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

$2989 \div 16 = 186$ kalan 13 **D**

$186 \div 16 = 11$ kalan 10 **A**

AX = 000Bh

DX = 000Ah

SI = 0001h

kalan dizisi

0D **00** **00** **00** +0 +1 +2 +3

BX = 0010h (16)

Tur 2: DX = 0 yapıldı, $186 \div 16 = 11$, kalan **10** → **A**.

Adım 8 / 13

DEĞİŞKENLER

AX = 000Bh

DX = 000Ah

BX = 0010h

SI = 1

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

$2989 \div 16 = 186$ kalan 13 **D**

$186 \div 16 = 11$ kalan 10 **A**

AX **000Bh**

DX **000Ah**

SI **0001h**

kalan dizisi

0D **0A** **00** **00** +0 +1 +2 +3

BX = **0010h** (16)

kalan[1] ← **0Ah**. Bölüm 11 ≠ 0: döngü sürer.

Adım 9 / 13

DEĞİŞKENLER

AX = **000Bh**

DX = **000Ah**

BX = **0010h**

SI = **1**

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 ► DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

$2989 \div 16 = 186$ kalan 13 **D**

$186 \div 16 = 11$ kalan 10 **A**

$11 \div 16 = 0$ kalan 11 **B**

AX = 0000h

DX = 000Bh

SI = 0002h

kalan dizisi

0D 0A 00 00 +0 +1 +2 +3

BX = 0010h (16)

Tur 3: $11 \div 16 = 0$, kalan 11 → B.

Adım 10 / 13

DEĞİŞKENLER

AX = 0000h

DX = 000Bh

BX = 0010h

SI = 2

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

$2989 \div 16 = 186$ kalan 13 **D**

$186 \div 16 = 11$ kalan 10 **A**

$11 \div 16 = 0$ kalan 11 **B**

AX 0000h

DX 000Bh

SI 0002h

kalan dizisi

0D 0A 0B 00 +0 +1 +2 +3

BX = 0010h (16)

kalan[2] ← 0Bh.

Adım 11 / 13

DEĞİŞKENLER

AX = 0000h

DX = 000Bh

BX = 0010h

SI = 2

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

$2989 \div 16 = 186$ kalan 13 **D**

$186 \div 16 = 11$ kalan 10 **A**

$11 \div 16 = 0$ kalan 11 **B**

AX 0000h

DX 000Bh

SI 0003h

kalan dizisi

0D 0A 0B 00 +0 +1 +2 +3

BX = 0010h (16)

SI = 3. Bölüm 0 → ZF = 1, **JNE** atlamaz: döngü biter, 3 basamak bulundu.

Adım 12 / 13

DEĞİŞKENLER

AX = 0000h

DX = 000Bh

BX = 0010h

SI = 3

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

Kod İzleme: DIV ile Onaltılık Basamaklar

```
1 ORG 100h
2 MOV AX, 2989 ; sayı
3 MOV BX, 16 ; taban
4 MOV SI, 0 ; sayaç
5 bol: MOV DX, 0 ; DX:AX
6 DIV BX ; bölüm, kalan
7 MOV kalan[SI], DL
8 INC SI
9 CMP AX, 0 ; bölüm 0 mı?
10 JNE bol
11 RET
12 kalan DB 4 DUP(0)
```

DIV BX ile bölme (taban 16)

$2989 \div 16 = 186$ kalan 13 **D**

$186 \div 16 = 11$ kalan 10 **A**

$11 \div 16 = 0$ kalan 11 **B**

sondan başa oku

AX 0000h

DX 000Bh

SI 0003h

kalan dizisi

0D 0A 0B 00 +0 +1 +2 +3

BX = 0010h (16)

Bellekte kalanlar **0D 0A 0B**; sondan başa okununca **0BADh**.

Kontrol: $11 \cdot 256 + 10 \cdot 16 + 13 = 2989 \checkmark$

Adım 13 / 13

DEĞİŞKENLER

AX = 0000h

DX = 000Bh

BX = 0010h

SI = 3

Aynı döngü her tabanda çalışır: kalan her zaman 0 ... taban-1 arasındır ve bir basamaktır.

45 → İkili (2'ye Bölme)

Aynı yöntem ikili için de geçerlidir: 45'i 2'ye böl.

45 → İkili (2'ye Bölme)

$45 \div 2 = 22$

kalan

1

$45 \div 2 = 22$, kalan 1.

45 → İkili (2'ye Bölme)

$$45 \div 2 = 22$$

kalan 1

$$22 \div 2 = 11$$

kalan 0

$$22 \div 2 = 11, \text{ kalan } 0.$$

Adım 3 / 8

45 → İkili (2'ye Bölme)

$$45 \div 2 = 22$$

kalan

1

$$22 \div 2 = 11$$

kalan

0

$$11 \div 2 = 5$$

kalan

1

$$11 \div 2 = 5, \text{ kalan } 1.$$

Adım 4 / 8

45 → İkili (2'ye Bölme)

$$45 \div 2 = 22$$

kalan

1

$$22 \div 2 = 11$$

kalan

0

$$11 \div 2 = 5$$

kalan

1

$$5 \div 2 = 2$$

kalan

1

$5 \div 2 = 2$, kalan 1.

Adım 5 / 8

45 → İkili (2'ye Bölme)

$$45 \div 2 = 22$$

kalan

1

$$22 \div 2 = 11$$

kalan

0

$$11 \div 2 = 5$$

kalan

1

$$5 \div 2 = 2$$

kalan

1

$$2 \div 2 = 1$$

kalan

0

$2 \div 2 = 1$, kalan 0.

Adım 6 / 8

45 → İkili (2'ye Bölme)

$$45 \div 2 = 22$$

kalan **1**

$$22 \div 2 = 11$$

kalan **0**

$$11 \div 2 = 5$$

kalan **1**

$$5 \div 2 = 2$$

kalan **1**

$$2 \div 2 = 1$$

kalan **0**

$$1 \div 2 = 0$$

kalan **1**

$1 \div 2 = 0$, kalan 1.

45 → İkili (2'ye Bölme)

$$45 \div 2 = 22$$

kalan **1**

$$22 \div 2 = 11$$

kalan **0**

$$11 \div 2 = 5$$

kalan **1**

$$5 \div 2 = 2$$

kalan **1**

$$2 \div 2 = 1$$

kalan **0**

$$1 \div 2 = 0$$

kalan **1**

aşağıdan yukarı oku

101101_b

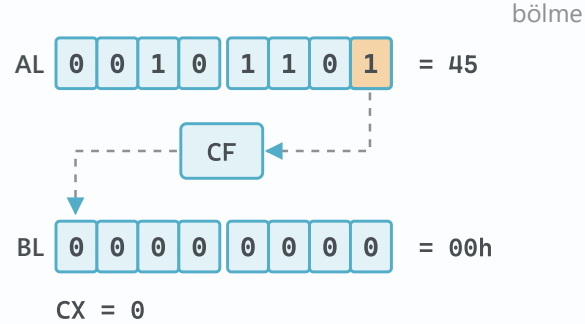
Sonuç **101101_b**. Kontrol: $32 + 8 + 4 + 1 = 45$ ✓ Alternatif: büyük sayıyı önce onaltılığa (2Dh) çevirip 4'lü açmak daha hızlıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1  ORG 100h
2  ► MOV AL, 45      ; bölünen
3  MOV CX, 8         ; 8 bit
4  bit: SHR AL, 1    ; kalan → CF
5  RCR BL, 1         ; CF → BL
6  LOOP bit
7  RET

```



AL = 45 = **0010 1101b**. Bir bit sağa kaydırmak sayıyı 2'ye böler; dışarı düşen bit bölmenin **kalanıdır**.

Adım 1 / 13

DEĞİŞKENLER

AL = 2Dh

BL = 00h

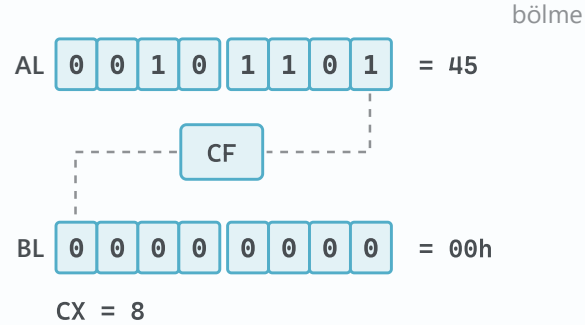
CF = 0

CX = 0

2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```
1 ORG 100h
2 MOV AL, 45 ; bölünen
3 MOV CX, 8 ; 8 bit
4 bit: SHR AL, 1 ; kalan → CF
5 RCR BL, 1 ; CF → BL
6 LOOP bit
7 RET
```



CX = 8: 8 bitin her biri için bir tur.

Adım 2 / 13

DEĞİŞKENLER

AL = 2Dh

BL = 00h

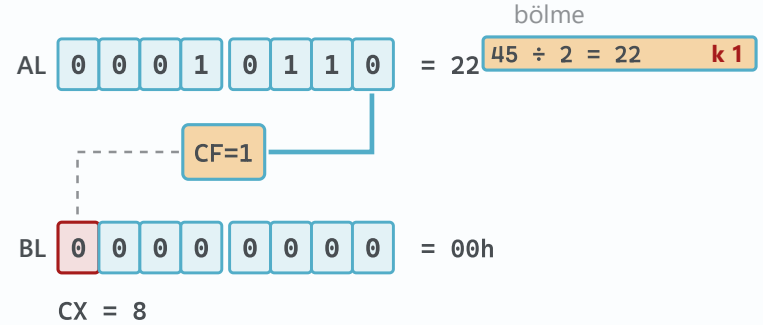
CF = 0

CX = 8

2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```
1 ORG 100h
2 MOV AL, 45 ; bölünen
3 MOV CX, 8 ; 8 bit
4 bit: SHR AL, 1 ; kalan → CF
5 RCR BL, 1 ; CF → BL
6 LOOP bit
7 RET
```



SHR AL, 1 : $45 \div 2 = 22$ (AL = 16h); düşen bit, yani kalan **1** → CF = 1.

Adım 3 / 13

DEĞİŞKENLER

AL = 16h

BL = 00h

CF = 1

CX = 8

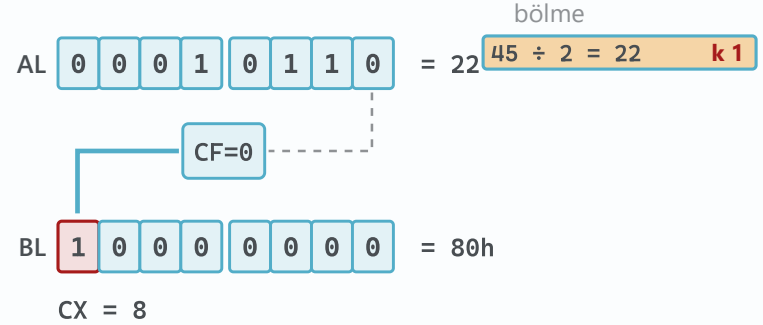
2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1 ORG 100h
2 MOV AL, 45      ; bölünen
3 MOV CX, 8        ; 8 bit
4 bit: SHR AL, 1   ; kalan → CF
5 ► RCR BL, 1      ; CF → BL
6 LOOP bit
7 RET

```



RCR BL, 1 : CF, BL'ye soldan (bit 7) girer; kalan saklandı.

Adım 4 / 13

DEĞİŞKENLER

AL = 16h

BL = 80h

CF = 0

CX = 8

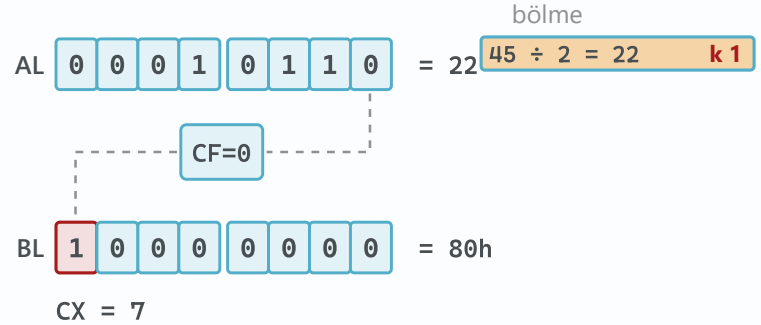
2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1 ORG 100h
2 MOV AL, 45      ; bölünen
3 MOV CX, 8        ; 8 bit
4 bit: SHR AL, 1    ; kalan → CF
5 RCR BL, 1        ; CF → BL
▶ LOOP bit
7 RET

```



LOOP : CX = 7 ≠ 0 → **bit** etiketine döner.

Adım 5 / 13

DEĞİŞKENLER

AL = 16h

BL = 80h

CF = 0

CX = 7

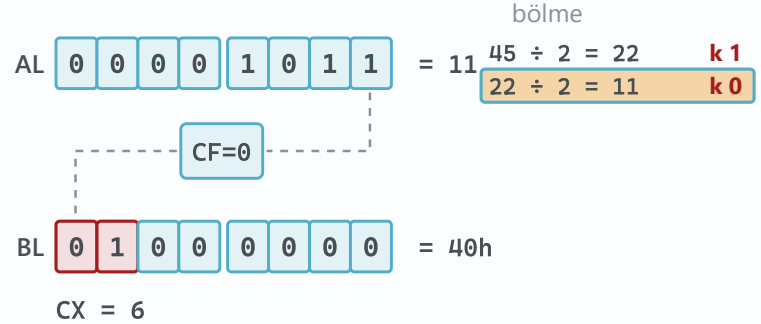
2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1 ORG 100h
2 MOV AL, 45      ; bölünen
3 MOV CX, 8        ; 8 bit
4 bit: SHR AL, 1   ; kalan → CF
5 RCR BL, 1        ; CF → BL
▶ LOOP bit
7 RET

```



Tur 2 (özet): $22 \div 2 = 11$, kalan **0**. Önceki kalanlar BL'de bir sağa kaydı; CX = 6.

Adım 6 / 13

DEĞİŞKENLER

AL = 0Bh

BL = 40h

CF = 0

CX = 6

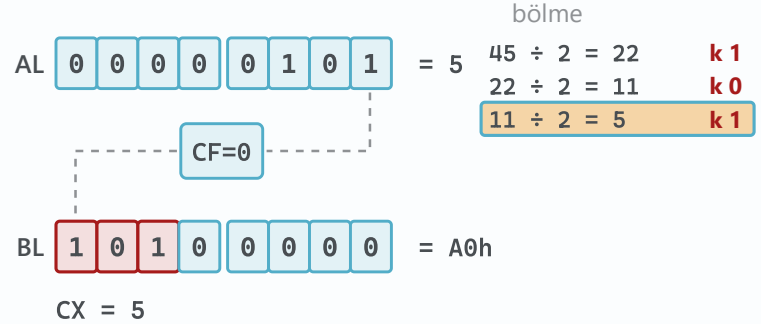
2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1 ORG 100h
2 MOV AL, 45      ; bölünen
3 MOV CX, 8        ; 8 bit
4 bit: SHR AL, 1    ; kalan → CF
5 RCR BL, 1        ; CF → BL
▶ LOOP bit
7 RET

```



Tur 3 (özet): $11 \div 2 = 5$, kalan **1**. Önceki kalanlar BL'de bir sağa kaydı; CX = 5.

Adım 7 / 13

DEĞİŞKENLER

AL = 05h

BL = A0h

CF = 0

CX = 5

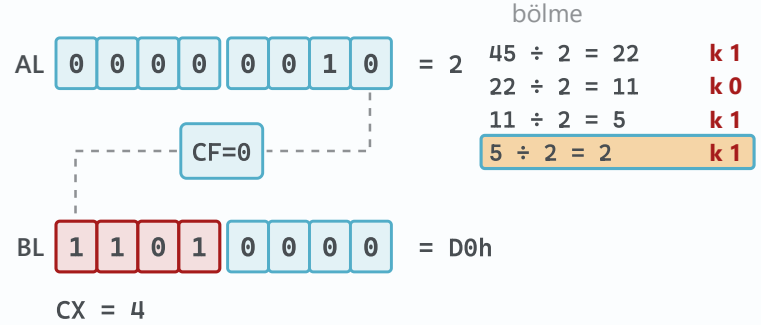
2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1 ORG 100h
2 MOV AL, 45      ; bölünen
3 MOV CX, 8        ; 8 bit
4 bit: SHR AL, 1   ; kalan → CF
5 RCR BL, 1        ; CF → BL
6 LOOP bit
7 RET

```



Tur 4 (özet): $5 \div 2 = 2$, kalan **1**. Önceki kalanlar BL'de bir sağa kaydı; CX = 4.

Adım 8 / 13

DEĞİŞKENLER

AL = 02h

BL = D0h

CF = 0

CX = 4

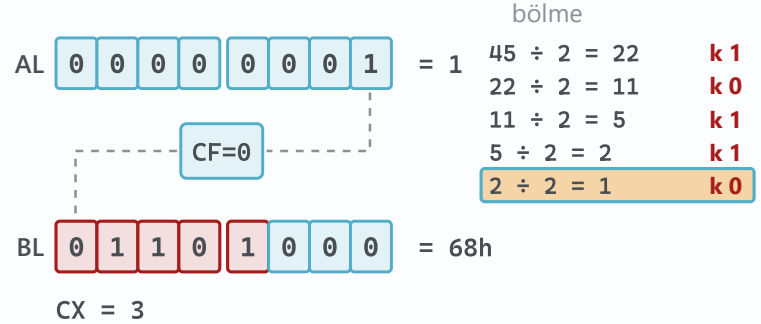
2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1 ORG 100h
2 MOV AL, 45      ; bölünen
3 MOV CX, 8        ; 8 bit
4 bit: SHR AL, 1    ; kalan → CF
5 RCR BL, 1        ; CF → BL
▶ LOOP bit
7 RET

```



Tur 5 (özet): $2 \div 2 = 1$, kalan **0**. Önceki kalanlar BL'de bir sağa kaydı; CX = 3.

Adım 9 / 13

DEĞİŞKENLER

AL = 01h

BL = 68h

CF = 0

CX = 3

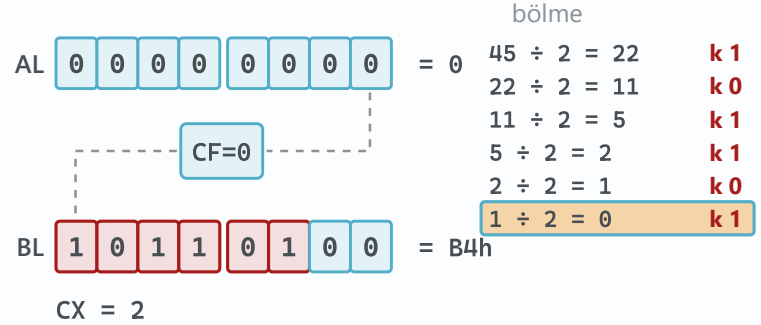
2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1 ORG 100h
2 MOV AL, 45      ; bölünen
3 MOV CX, 8        ; 8 bit
4 bit: SHR AL, 1   ; kalan → CF
5 RCR BL, 1        ; CF → BL
▶ LOOP bit
7 RET

```



Tur 6 (özet): $1 \div 2 = 0$, kalan **1**. Önceki kalanlar BL'de bir sağa kaydı; CX = 2.

Adım 10 / 13

DEĞİŞKENLER

AL = 00h

BL = B4h

CF = 0

CX = 2

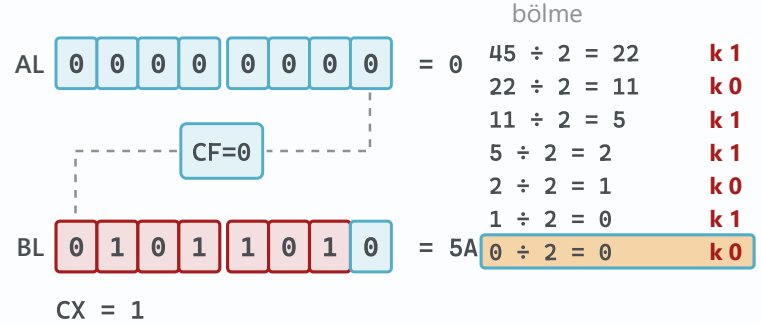
2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1 ORG 100h
2 MOV AL, 45      ; bölünen
3 MOV CX, 8        ; 8 bit
4 bit: SHR AL, 1    ; kalan → CF
5 RCR BL, 1        ; CF → BL
▶ LOOP bit
7 RET

```



Tur 7 (özet): $0 \div 2 = 0$, kalan **0**. Önceki kalanlar BL'de bir sağa kaydı; CX = 1. Baştaki sıfırlar.

Adım 11 / 13

DEĞİŞKENLER

AL = 00h

BL = 5Ah

CF = 0

CX = 1

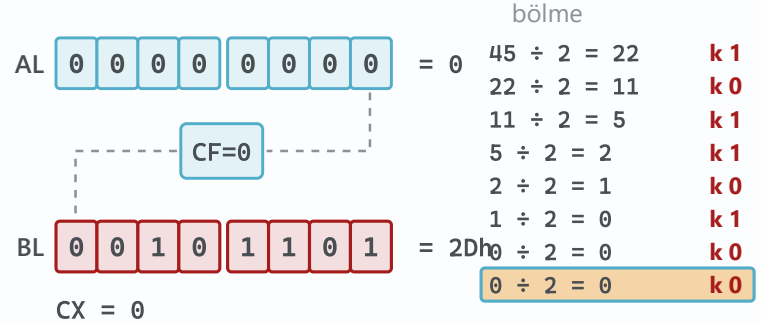
2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1  ORG 100h
2  MOV AL, 45      ; bölünen
3  MOV CX, 8        ; 8 bit
4  bit: SHR AL, 1   ; kalan → CF
5  RCR BL, 1        ; CF → BL
6  LOOP bit
7  RET

```



Tur 8 (özet): $0 \div 2 = 0$, kalan **0**. Önceki kalanlar BL'de bir sağa kaydı; CX = 0. Baştaki sıfırlar.

Adım 12 / 13

DEĞİŞKENLER

AL = 00h

BL = 2Dh

CF = 0

CX = 0

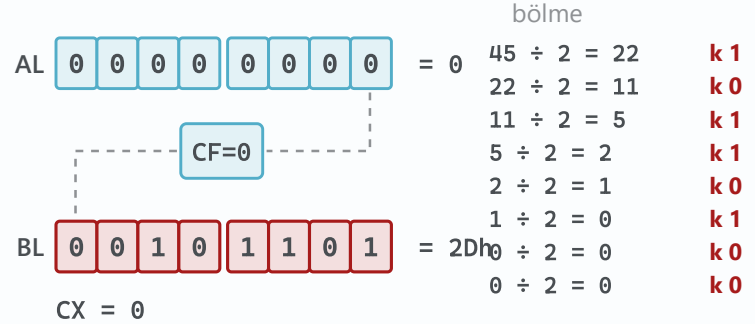
2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kod İzleme: Kalanlar CF'den Gelir

```

1 ORG 100h
2 MOV AL, 45      ; bölünen
3 MOV CX, 8        ; 8 bit
4 bit: SHR AL, 1   ; kalan → CF
5 RCR BL, 1        ; CF → BL
6 LOOP bit
▶ RET

```



İlk kalan BL'nin bit 0'ına, son kalan bit 7'sine ulaştı: BL = **0010 1101b** = 2Dh = 45. Kalanlar sondan başa okununca ikili gösterim çıkar.

Adım 13 / 13

DEĞİŞKENLER

AL = 00h

BL = 2Dh

CF = 0

CX = 0

2'nin kuvvetine bölmek için kaydırma yeter: **SHR** bölümü, CF kalanı verir. Bu yüzden ikili basamaklar 2'ye bölme kalanlarıdır.

Kesirli Kısım: 2 ile Çarpma

- Kesirli kısım **2 ile çarpılır**; çarpımın tam kısmı (0 ya da 1) bir **bit** olur.
- Kalan kesirle devam edilir; kesir **0** olunca ya da istenen bit sayısına ulaşınca durulur.
- Tamsayı ve kesir ayrı çevrilip birleştirilir: $25,625 = 11001b + 0,101b = \mathbf{11001,101b}$.

Örnek: $0,625 \rightarrow$ ikili. Kesirli kısım tekrar tekrar 2 ile çarpılır.

Kesirli Kısım: 2 ile Çarpma

- Kesirli kısım **2 ile çarpılır**; çarpımın tam kısmı (0 ya da 1) bir **bit** olur.
- Kalan kesirle devam edilir; kesir **0** olunca ya da istenen bit sayısına ulaşınca durulur.
- Tamsayı ve kesir ayrı çevrilip birleştirilir: $25,625 = 11001b + 0,101b = \mathbf{11001,101b}$.

$$0,625 \times 2 = 1,25 \quad \text{bit } \boxed{1} \quad \text{kalan kesir } 0,25$$

$0,625 \times 2 = 1,25$: tam kısım **1** → ilk kesir biti (2^{-1}). Kesir 0,25 ile devam.

Kesirli Kısım: 2 ile Çarpma

- Kesirli kısım **2 ile çarpılır**; çarpımın tam kısmı (0 ya da 1) bir **bit** olur.
- Kalan kesirle devam edilir; kesir **0** olunca ya da istenen bit sayısına ulaşınca durulur.
- Tamsayı ve kesir ayrı çevrilip birleştirilir: $25,625 = 11001b + 0,101b = \mathbf{11001,101b}$.

$$0,625 \times 2 = 1,25 \quad \text{bit } \boxed{1} \quad \text{kalan kesir } 0,25$$

$$0,25 \times 2 = 0,5 \quad \text{bit } \boxed{0} \quad \text{kalan kesir } 0,5$$

$$0,25 \times 2 = 0,5: \text{ tam kısım } \mathbf{0} \rightarrow 2^{-2} \text{ biti.}$$

Kesirli Kısım: 2 ile Çarpma

- Kesirli kısım **2 ile çarpılır**; çarpımın tam kısmı (0 ya da 1) bir **bit** olur.
- Kalan kesirle devam edilir; kesir **0** olunca ya da istenen bit sayısına ulaşınca durulur.
- Tamsayı ve kesir ayrı çevrilip birleştirilir: $25,625 = 11001b + 0,101b = \mathbf{11001,101b}$.

$0,625 \times 2 = 1,25$	bit	1	kalan kesir 0,25
$0,25 \times 2 = 0,5$	bit	0	kalan kesir 0,5
$0,5 \times 2 = 1,0$	bit	1	kalan kesir 0,0

$0,5 \times 2 = 1,0$: tam kısım **1**, kalan kesir 0 → dur.

Kesirli Kısım: 2 ile Çarpma

- Kesirli kısım **2 ile çarpılır**; çarpımın tam kısmı (0 ya da 1) bir **bit** olur.
- Kalan kesirle devam edilir; kesir **0** olunca ya da istenen bit sayısına ulaşınca durulur.
- Tamsayı ve kesir ayrı çevrilip birleştirilir: $25,625 = 11001b + 0,101b = \mathbf{11001,101b}$.

$0,625 \times 2 = 1,25$	bit	1	kalan kesir 0,25
$0,25 \times 2 = 0,5$	bit	0	kalan kesir 0,5
$0,5 \times 2 = 1,0$	bit	1	kalan kesir 0,0

yukarıdan aşağı oku

0,101b

Sonuç **0,101b**. Kontrol: $1/2 + 1/8 = 0,5 + 0,125 = 0,625 \checkmark$ Bitler **yukarıdan aşağı** okunur (bölme yönteminin tersi).

0,1 → İkili: Tekrar Eden Kesir

0,1 → ikili: aynı yöntem.

Adım 1 / 8

Payda 2'nin kuvveti olmayan kesirler ($0,1 = 1/10$) ikilide tam yazılamaz; bit sayısı sınırlı olduğu için yuvarlanır. IEEE 754'te de 0,1 yaklaşık saklanır: birçok dilde $0,1 + 0,2 \neq 0,3$ çıkmasının nedeni budur.

0,1 → İkili: Tekrar Eden Kesir

$$0,1 \times 2 = 0,2 \quad \text{bit } \boxed{0} \quad \text{kalan kesir } 0,2$$

$$0,1 \times 2 = 0,2 \rightarrow \mathbf{0}.$$

Adım 2 / 8

Payda 2'nin kuvveti olmayan kesirler ($0,1 = 1/10$) ikilide tam yazılamaz; bit sayısı sınırlı olduğu için yuvarlanır. IEEE 754'te de 0,1 yaklaşık saklanır: birçok dilde $0,1 + 0,2 \neq 0,3$ çıkmasının nedeni budur.

0,1 → İkili: Tekrar Eden Kesir

$$0,1 \times 2 = 0,2$$

bit

0

kalan kesir 0,2

$$0,2 \times 2 = 0,4$$

bit

0

kalan kesir 0,4

$$0,2 \times 2 = 0,4 \rightarrow \mathbf{0}.$$

Adım 3 / 8

Payda 2'nin kuvveti olmayan kesirler ($0,1 = 1/10$) ikilide tam yazılamaz; bit sayısı sınırlı olduğu için yuvarlanır. IEEE 754'te de 0,1 yaklaşık saklanır: birçok dilde $0,1 + 0,2 \neq 0,3$ çıkmasının nedeni budur.

0,1 → İkili: Tekrar Eden Kesir

$0,1 \times 2 = 0,2$	bit	0	kalan kesir 0,2
$0,2 \times 2 = 0,4$	bit	0	kalan kesir 0,4
$0,4 \times 2 = 0,8$	bit	0	kalan kesir 0,8

$$0,4 \times 2 = 0,8 \rightarrow \mathbf{0}.$$

Adım 4 / 8

Payda 2'nin kuvveti olmayan kesirler ($0,1 = 1/10$) ikilide tam yazılamaz; bit sayısı sınırlı olduğu için yuvarlanır. IEEE 754'te de 0,1 yaklaşık saklanır: birçok dilde $0,1 + 0,2 \neq 0,3$ çıkmasının nedeni budur.

0,1 → İkili: Tekrar Eden Kesir

$0,1 \times 2 = 0,2$	bit	0	kalan kesir 0,2
$0,2 \times 2 = 0,4$	bit	0	kalan kesir 0,4
$0,4 \times 2 = 0,8$	bit	0	kalan kesir 0,8
$0,8 \times 2 = 1,6$	bit	1	kalan kesir 0,6

$0,8 \times 2 = 1,6 \rightarrow 1$, kesir 0,6.

Adım 5 / 8

Payda 2'nin kuvveti olmayan kesirler ($0,1 = 1/10$) ikilide tam yazılamaz; bit sayısı sınırlı olduğu için yuvarlanır. IEEE 754'te de 0,1 yaklaşık saklanır: birçok dilde $0,1 + 0,2 \neq 0,3$ çıkmasının nedeni budur.

0,1 → İkili: Tekrar Eden Kesir

$0,1 \times 2 = 0,2$	bit	0	kalan kesir 0,2
$0,2 \times 2 = 0,4$	bit	0	kalan kesir 0,4
$0,4 \times 2 = 0,8$	bit	0	kalan kesir 0,8
$0,8 \times 2 = 1,6$	bit	1	kalan kesir 0,6
$0,6 \times 2 = 1,2$	bit	1	kalan kesir 0,2

$0,6 \times 2 = 1,2 \rightarrow 1$, kesir **0,2**: bu kesir daha önce (2. satırda) görüldü!

Adım 6 / 8

Payda 2'nin kuvveti olmayan kesirler ($0,1 = 1/10$) ikilide tam yazılamaz; bit sayısı sınırlı olduğu için yuvarlanır. IEEE 754'te de 0,1 yaklaşık saklanır: birçok dilde $0,1 + 0,2 \neq 0,3$ çıkmasının nedeni budur.

0,1 → İkili: Tekrar Eden Kesir

$0,1 \times 2 = 0,2$	bit	0	kalan kesir 0,2
$0,2 \times 2 = 0,4$	bit	0	kalan kesir 0,4
$0,4 \times 2 = 0,8$	bit	0	kalan kesir 0,8
$0,8 \times 2 = 1,6$	bit	1	kalan kesir 0,6
$0,6 \times 2 = 1,2$	bit	1	kalan kesir 0,2
$0,2 \times 2 = 0,4$	bit	0	kalan kesir 0,4

0,2 yeniden çarpılıyor: 2.–5. satırlar sonsuza kadar tekrar eder.

Adım 7 / 8

Payda 2'nin kuvveti olmayan kesirler ($0,1 = 1/10$) ikilide tam yazılamaz; bit sayısı sınırlı olduğu için yuvarlanır. IEEE 754'te de 0,1 yaklaşık saklanır: birçok dilde $0,1 + 0,2 \neq 0,3$ çıkmasının nedeni budur.

0,1 → İkili: Tekrar Eden Kesir

$0,1 \times 2 = 0,2$	bit	0	kalan kesir 0,2
$0,2 \times 2 = 0,4$	bit	0	kalan kesir 0,4
$0,4 \times 2 = 0,8$	bit	0	kalan kesir 0,8
$0,8 \times 2 = 1,6$	bit	1	kalan kesir 0,6
$0,6 \times 2 = 1,2$	bit	1	kalan kesir 0,2
$0,2 \times 2 = 0,4$	bit	0	kalan kesir 0,4

tekrar eden
0011

0,0 0011 0011 ...b

Sonuç **0,1 = 0,0 0011 0011 0011...b**: ikili tabanda **sonlu değildir**. 8 kesir biti ile $0,00011001b = 0,09765625$ saklanır.

Adım 8 / 8

Payda 2'nin kuvveti olmayan kesirler ($0,1 = 1/10$) ikilide tam yazılamaz; bit sayısı sınırlı olduğu için yuvarlanır. IEEE 754'te de 0,1 yaklaşık saklanır: birçok dilde $0,1 + 0,2 \neq 0,3$ çıkmasının nedeni budur.

Taban Dönüştürücü

sayı

43868

giriş tabanı

10 (onluk) ▼

bölme adımları

16'ya böl ▼

ondalık
43868

onaltılı
0AB5Ch

sekizli
125534o

ikili
1010 1011 0101
1100b

0000043868 ÷ 16 = 0000002741

kalan 12 = C

0000002741 ÷ 16 = 0000000171

kalan 5

0000000171 ÷ 16 = 0000000010

kalan 11 = B

0000000010 ÷ 16 = 0000000000

kalan 10 = A

Kalanlar aşağıdan yukarı okunur → **AB5C** (16 tabanında)

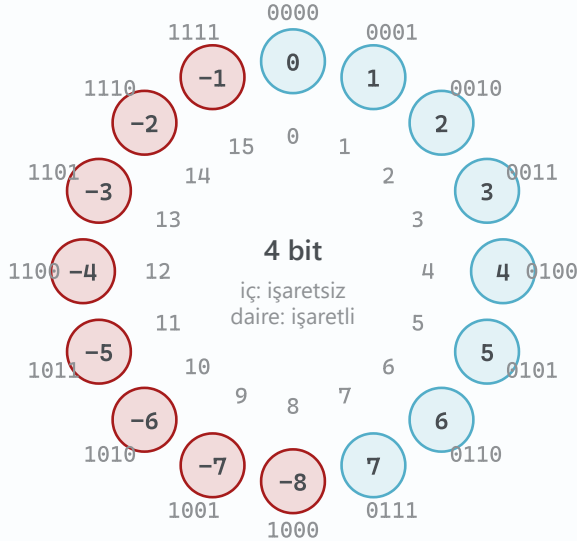
İşaretili Sayılar

- işaretsiz: 0 128 255
- 00h ... 7Fh 80h ... FFh (MSB = 1)
- işaretili: 0 +127 -128 -1
- FBh = 251 = 256 - 5 → -5**

8 bitte en büyük kombinasyon **255**'tir (0 ... 255 = 256 değer). 16 bitte de negatif aralık 32768 ... **65535**'tir.

İŞARETLİ SAYILAR

Sayı Çemberi



- 4 bitlik tüm desenler bir çember üzerinde: saat yönünde ilerlemek **+1** demektir.
- 0111 (7) ile 1000 arasında işaretli yorum **+7'den -8'e** atlar → işaretli taşma (OF).
- 1111 ile 0000 arasında işaretsiz yorum **15'ten 0'a** atlar → elde (CF).
- Negatif sayıların hepsinde en yüksek bit (MSB) **1**'dir: MSB işaret biti olarak okunur.
- Toplama ve çıkarma devresi her iki yorum için **aynıdır**; işlemci yalnızca bayrakları farklı kurar.

İŞARETLİ SAYILAR

4 Bitlik Gösterimler

İkili	İşaretsiz	İşaret-büyüklik	birler tümleyeni	ikiye tümleyen
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7

İkili	İşaretsiz	İşaret-büyüklik	birler tümleyeni	ikiye tümleyen
1000	8	-0	-7	-8
1001	9	-1	-6	-7
1010	10	-2	-5	-6
1011	11	-3	-4	-5
1100	12	-4	-3	-4
1101	13	-5	-2	-3
1110	14	-6	-1	-2
1111	15	-7	-0	-1

Tablodaki ortadaki sütun **işaret-büyüklik** gösterimidir (1000 = -0, 1111 = -7). 8086 ve tüm modern işlemciler **ikiye tümleyen** kullanır: tek bir sıfır vardır ve aralık -8 ... +7'dir.

Birler Tümleyeni Yöntemi

- **1's complement:** bir sayının negatifini almak için sayının **bütün bitleri tersine çevrilir** ($1 \rightarrow 0, 0 \rightarrow 1$); bitwise NOT.
- 8 bitlik örnek: $00101101 \rightarrow 11010010$.
- Sorun: **iki sıfır** vardır ($00000000 = +0, 11111111 = -0$) ve toplama sonrası elde geri eklenmelidir (end-around carry).
- 8086'da **NOT** komutu birler tümleyenini üretir.

Başlangıç.

Adım 1 / 3

Birler Tümleyeni Yöntemi

- **1's complement:** bir sayının negatifini almak için sayının **bütün bitleri tersine çevrilir** ($1 \rightarrow 0, 0 \rightarrow 1$); bitwise NOT.
- 8 bitlik örnek: $00101101 \rightarrow 11010010$.
- Sorun: **iki sıfır** vardır ($00000000 = +0, 11111111 = -0$) ve toplama sonrası elde geri eklenmelidir (end-around carry).
- 8086'da **NOT** komutu birler tümleyenini üretir.

+45 =

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

 2Dh

+45 = 0010 1101b.

Birler Tümleyeni Yöntemi

- **1's complement:** bir sayının negatifini almak için sayının **bütün bitleri tersine çevrilir** ($1 \rightarrow 0, 0 \rightarrow 1$); bitwise NOT.
- 8 bitlik örnek: $00101101 \rightarrow 11010010$.
- Sorun: **iki sıfır** vardır ($00000000 = +0, 11111111 = -0$) ve toplama sonrası elde geri eklenmelidir (end-around carry).
- 8086'da **NOT** komutu birler tümleyenini üretir.

+45 =	0	0	1	0	1	1	0	1	2Dh
NOT →	1	1	0	1	0	0	1	0	D2h (-45, birler t.)

Tüm bitler ters çevrilir: $1101\ 0010b = D2h$. birler tümleyeninde bu -45 'tir; ikiye tümleyende -46 !

İkiye Tümleyen Yöntemi

- Negatif sayıları temsil etmek için kullanılan standart yöntemdir: **önce tüm bitler ters çevrilir, sonra 1 eklenir.**
- 8086'da `NEG` komutu bunu tek adımda yapar ($0 - x$).
- Kısa yol: sağdan ilk **1**'e kadar (o dahil) bitleri aynen yaz, kalanları ters çevir.

Başlangıç.

Adım 1 / 4

Kontrol: $D3h + 2Dh = 100h \rightarrow 8$ bitte $00h$. Bir sayının ikiye tümleyeninin ikiye tümleyeni kendisidir.

İkiye Tümleyen Yöntemi

- Negatif sayıları temsil etmek için kullanılan standart yöntemdir: **önce tüm bitler ters çevrilir, sonra 1 eklenir.**
- 8086'da `NEG` komutu bunu tek adımda yapar ($0 - x$).
- Kısa yol: sağdan ilk **1**'e kadar (o dahil) bitleri aynen yaz, kalanları ters çevir.

+45 =

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

 2Dh

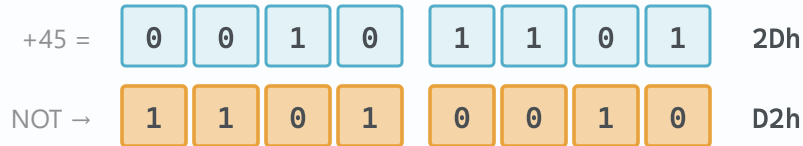
+45 = 0010 1101b.

Adım 2 / 4

Kontrol: $D3h + 2Dh = 100h \rightarrow 8$ bitte 00h. Bir sayının ikiye tümleyeninin ikiye tümleyeni kendisidir.

İkiye Tümleyen Yöntemi

- Negatif sayıları temsil etmek için kullanılan standart yöntemdir: **önce tüm bitler ters çevrilir, sonra 1 eklenir.**
- 8086'da `NEG` komutu bunu tek adımda yapar ($0 - x$).
- Kısa yol: sağdan ilk **1**'e kadar (o dahil) bitleri aynen yaz, kalanları ters çevir.



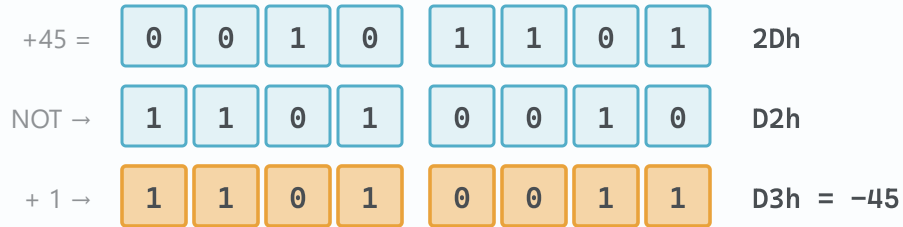
Tüm bitler ters çevrilir (bitwise NOT): 1101 0010b.

Adım 3 / 4

Kontrol: $D3h + 2Dh = 100h \rightarrow 8$ bitte 00h. Bir sayının ikiye tümleyeninin ikiye tümleyeni kendisidir.

İkiye Tümleyen Yöntemi

- Negatif sayıları temsil etmek için kullanılan standart yöntemdir: **önce tüm bitler ters çevrilir, sonra 1 eklenir.**
- 8086'da `NEG` komutu bunu tek adımda yapar ($0 - x$).
- Kısa yol: sağdan ilk **1**'e kadar (o dahil) bitleri aynen yaz, kalanları ters çevir.



1 eklenir: $1101\ 0011b = \mathbf{D3h} = 211 = 256 - 45 \checkmark$ Bu, -45'in 8 bitlik gösterimidir.

Adım 4 / 4

Kontrol: $D3h + 2Dh = 100h \rightarrow 8\ bitte\ 00h$. Bir sayının ikiye tümleyeninin ikiye tümleyeni kendisidir.

Kod İzleme: NOT + 1 ile NEG

```
1 ORG 100h
2 MOV AL, 45      ; +45
3 MOV BL, AL
4 NOT BL          ; 1'e tümleyen
5 INC BL          ; + 1
6 NEG AL          ; tek komutla
7 CMP AL, BL      ; aynı mı?
8 ADD AL, 45      ; -45 + 45
9 MOV AX, 45
10 NEG AX         ; 16 bitte -45
11 RET
```

AL

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

2Dh

işaretsiz 45 · işaretli +45

BL

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

00h

işaretsiz 0 · işaretli +0

CF	ZF	SF	OF
0	0	0	0

AL = +45 = **0010 1101b** (2Dh).

Adım 1 / 9

DEĞİŞKENLER

AX = **002Dh**

BL = **00h**

CF = **0**

ZF = **0**

SF = **0**

$-x = \text{NOT } x + 1 = 2^n - x$. NEG bunu tek komutla yapar; sonuç her genişlikte (8, 16 bit) aynı kuralla bulunur.

Kod İzleme: NOT + 1 ile NEG

```

1  ORG 100h
2  MOV AL, 45      ; +45
3  MOV BL, AL
4  NOT BL          ; 1'e tümleyen
5  INC BL          ; + 1
6  NEG AL          ; tek komutla
7  CMP AL, BL      ; aynı mı?
8  ADD AL, 45      ; -45 + 45
9  MOV AX, 45
10 NEG AX          ; 16 bitte -45
11 RET

```

AL

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

2Dh

işaretsiz 45 · işaretli +45

BL

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

2Dh

işaretsiz 45 · işaretli +45

CF	ZF	SF	OF
0	0	0	0

BL = AL: aynı değerin kopyası; tümleyen BL üzerinde iki adımda alınacak.

Adım 2 / 9

DEĞİŞKENLER

AX = 002Dh

BL = 2Dh

CF = 0

ZF = 0

SF = 0

$-x = \text{NOT } x + 1 = 2^n - x$. NEG bunu tek komutla yapar; sonuç her genişlikte (8, 16 bit) aynı kuralla bulunur.

Kod İzleme: NOT + 1 ile NEG

```

1 ORG 100h
2 MOV AL, 45      ; +45
3 MOV BL, AL
4 ▶ NOT BL      ; 1'e tümleyen
5 INC BL        ; + 1
6 NEG AL        ; tek komutla
7 CMP AL, BL    ; aynı mı?
8 ADD AL, 45    ; -45 + 45
9 MOV AX, 45
10 NEG AX       ; 16 bitte -45
11 RET

```

AL

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

2Dh

işaretsiz 45 · işaretli +45

BL

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

D2h

işaretsiz 210 · işaretli -46

CF	ZF	SF	OF
0	0	0	0

NOT: 0 ↔ 1

NOT BL : bütün bitler ters → **1101 0010b** = D2h (birler tümleyeni). NOT bayrakları değiştirmez.

Adım 3 / 9

DEĞİŞKENLER

AX = 002Dh

BL = D2h

CF = 0

ZF = 0

SF = 0

$-x = \text{NOT } x + 1 = 2^n - x$. NEG bunu tek komutta yapar; sonuç her genişlikte (8, 16 bit) aynı kuralla bulunur.

Kod İzleme: NOT + 1 ile NEG

```

1 ORG 100h
2 MOV AL, 45      ; +45
3 MOV BL, AL
4 NOT BL          ; 1'e tümleyen
5 INC BL          ; + 1
6 NEG AL          ; tek komutla
7 CMP AL, BL      ; aynı mı?
8 ADD AL, 45      ; -45 + 45
9 MOV AX, 45
10 NEG AX         ; 16 bitte -45
11 RET

```

AL

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

2Dh

işaretsiz 45 · işaretli +45

BL

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

D3h

işaretsiz 211 · işaretli -45

CF	ZF	SF	OF
0	0	1	0

NOT + 1

INC BL : + 1 → **1101 0011b** = D3h. İşaretsiz 211 = 256 – 45; işaretli yorum **-45**.

Adım 4 / 9

DEĞİŞKENLER

AX = **002Dh**

BL = **D3h**

CF = **0**

ZF = **0**

SF = **1**

$-x = \text{NOT } x + 1 = 2^n - x$. NEG bunu tek komutta yapar; sonuç her genişlikte (8, 16 bit) aynı kuralla bulunur.

Kod İzleme: NOT + 1 ile NEG

```

1 ORG 100h
2 MOV AL, 45      ; +45
3 MOV BL, AL
4 NOT BL          ; 1'e tümleyen
5 INC BL          ; + 1
6 ► NEG AL        ; tek komutla
7 CMP AL, BL      ; aynı mı?
8 ADD AL, 45      ; -45 + 45
9 MOV AX, 45
10 NEG AX         ; 16 bitte -45
11 RET

```

AL

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

D3h

işaretsiz 211 · işaretli -45

BL

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

D3h

işaretsiz 211 · işaretli -45

CF	ZF	SF	OF
1	0	1	0

NEG = 0 - x

NEG AL (0 - AL) aynı sonucu tek komutta verir: AL = **D3h**.
İşlenen ≠ 0 → CF = 1.

Adım 5 / 9

DEĞİŞKENLER

AX = **00D3h**

BL = **D3h**

CF = **1**

ZF = **0**

SF = **1**

$-x = \text{NOT } x + 1 = 2^n - x$. NEG bunu tek komutta yapar; sonuç her genişlikte (8, 16 bit) aynı kuralla bulunur.

Kod İzleme: NOT + 1 ile NEG

```

1  ORG 100h
2  MOV AL, 45      ; +45
3  MOV BL, AL
4  NOT BL          ; 1'e tümleyen
5  INC BL          ; + 1
6  NEG AL          ; tek komutla
7  CMP AL, BL      ; aynı mı?
8  ADD AL, 45      ; -45 + 45
9  MOV AX, 45
10 NEG AX          ; 16 bitte -45
11 RET

```

AL

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

D3h

işaretsiz 211 · işaretli -45

BL

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

D3h

işaretsiz 211 · işaretli -45

CF 0	ZF 1	SF 0	OF 0
---------	---------	---------	---------

fark 0 → eşit

CMP AL, BL : D3h – D3h = 0 → **ZF = 1**. İki yol aynı bit desenini üretti.

Adım 6 / 9

DEĞİŞKENLER

AX = 00D3h

BL = D3h

CF = 0

ZF = 1

SF = 0

$-x = \text{NOT } x + 1 = 2^n - x$. NEG bunu tek komutta yapar; sonuç her genişlikte (8, 16 bit) aynı kuralla bulunur.

Kod İzleme: NOT + 1 ile NEG

```

1  ORG 100h
2  MOV AL, 45      ; +45
3  MOV BL, AL
4  NOT BL          ; 1'e tümleyen
5  INC BL          ; + 1
6  NEG AL          ; tek komutla
7  CMP AL, BL      ; aynı mı?
8  ADD AL, 45      ; -45 + 45
9  MOV AX, 45
10 NEG AX          ; 16 bitte -45
11 RET

```

AL

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

00h

işaretsiz 0 · işaretli +0

BL

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

D3h

işaretsiz 211 · işaretli -45

CF 1	ZF 1	SF 0	OF 0
----------------	----------------	----------------	----------------

100h: elde düştü

ADD AL, 45 : D3h + 2Dh = 100h. 9. bit düşer (CF = 1), AL = **00h**, ZF = 1: -45 + 45 = 0 ✓

Adım 7 / 9

DEĞİŞKENLER

AX = **0000h**

BL = **D3h**

CF = **1**

ZF = **1**

SF = **0**

$-x = \text{NOT } x + 1 = 2^n - x$. NEG bunu tek komutla yapar; sonuç her genişlikte (8, 16 bit) aynı kuralla bulunur.

Kod İzleme: NOT + 1 ile NEG

```

1  ORG 100h
2  MOV AL, 45      ; +45
3  MOV BL, AL
4  NOT BL          ; 1'e tümleyen
5  INC BL          ; + 1
6  NEG AL          ; tek komutla
7  CMP AL, BL      ; aynı mı?
8  ADD AL, 45      ; -45 + 45
9  MOV AX, 45
10 NEG AX          ; 16 bitte -45
11 RET

```

AX

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

= 002Dh · işaretsiz 45 · işaretli +45

BL

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

D3h

işaretsiz 211 · işaretli -45

CF	1
----	---

ZF	1
----	---

SF	0
----	---

OF	0
----	---

AX = 45 = **002Dh**: aynı sayı 16 bitte.

Adım 8 / 9

DEĞİŞKENLER

AX = **002Dh**

BL = **D3h**

CF = **1**

ZF = **1**

SF = **0**

$-x = \text{NOT } x + 1 = 2^n - x$. NEG bunu tek komutla yapar; sonuç her genişlikte (8, 16 bit) aynı kuralla bulunur.

Kod İzleme: NOT + 1 ile NEG

```

1  ORG 100h
2  MOV AL, 45      ; +45
3  MOV BL, AL
4  NOT BL          ; 1'e tümleyen
5  INC BL          ; + 1
6  NEG AL          ; tek komutla
7  CMP AL, BL      ; aynı mı?
8  ADD AL, 45      ; -45 + 45
9  MOV AX, 45
10 NEG AX          ; 16 bitte -45
11 RET

```

AX

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

= FFD3h · işaretsiz 65491 · işaretli -45

BL

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

D3h

işaretsiz 211 · işaretli -45

CF	ZF	SF	OF
1	0	1	0

16 bit NEG

NEG AX : **FFD3h** = 65536 – 45. Düşük bayt yine D3h; üst bayt FFh, MSB = 1 → negatif (SF = 1).

Adım 9 / 9

DEĞİŞKENLER

AX = **FFD3h**

BL = **D3h**

CF = **1**

ZF = **0**

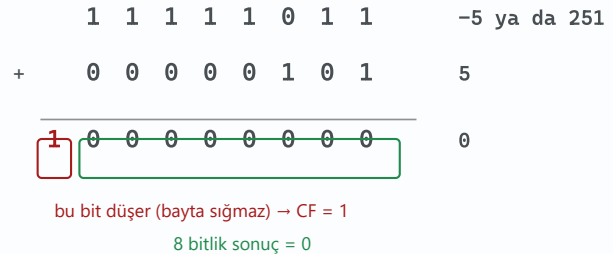
SF = **1**

$-x = \text{NOT } x + 1 = 2^n - x$. NEG bunu tek komutla yapar; sonuç her genişlikte (8, 16 bit) aynı kuralla bulunur.

İŞARETLİ SAYILAR

$$-5 + 5 = 0$$

- Yöntem matematikteki $-5 + 5 = 0$ kuralıyla uyumlu olmalıdır.
- 5 ve 251 baytları toplanırsa sonuç 255'i aşar (256).
- Dokuzuncu bit bayta sığmaz ve **düşer**; kalan 8 bit **sıfırdır**.
- Yani aynı toplama devresi hem işaretsiz hem işaretli sayılar için doğru sonuç verir.



İŞARETLİ SAYILAR

İşaretsiz Aralıklar

Genişlik	Kombinasyon	İşaretsiz	İşaretsiz (ikiye tümleyen)	Negatifler
8 bit	256	0 ... 255	-128 ... +127	80h ... FFh
16 bit	65536	0 ... 65535	-32768 ... +32767	8000h ... FFFFh
32 bit	4 294 967 296	0 ... 4 294 967 295	-2 147 483 648 ... +2 147 483 647	80000000h ...

- Kombinasyonların üst yarısı kullanıldığında **yüksek bit her zaman 1**'dir; bu bit sayının işaretini belirlemek için kullanılır (SF bayrağı).
- 16 bit ile 65536 kombinasyon: ilk 32768 (0 ... 32767) pozitif, diğer 32768 (32768 ... 65535) negatif sayılardır.
- Negatif aralık pozitiften bir fazladır: -128'in pozitif karşılığı 8 bite **sığmaz** (NEG 80h = 80h, OF = 1).
- **İşaret genişletme**: 8 bitlik işaretli sayı 16 bite taşınırken MSB sola kopyalanır: FBh (-5) → FFFBh. 8086: **CBW** (AL → AX), **CWD** (AX → DX:AX).

NEG ve CBW

► **MOV** AL, 5

2 **NEG** AL ; AL = 0 - 5

3 **ADD** AL, 5 ; -5 + 5

4 **MOV** AL, 0FBh ; -5

5 **CBW** ; AL → AX işaretli

6 **MOV** AL, 80h ; -128

7 **NEG** AL ; -(-128) = ?

AL = 05h.

Adım 1 / 7

DEĞİŞKENLER

AL = **05h**

AX = **0005h**

CF = **0**

ZF = **0**

SF = **0**

OF = **0**

NEG ve CBW

```
1 MOV AL, 5
▶ NEG AL ; AL = 0 - 5
3 ADD AL, 5 ; -5 + 5
4 MOV AL, 0FBh ; -5
5 CBW ; AL → AX işaretli
6 MOV AL, 80h ; -128
7 NEG AL ; -(-128) = ?
```

NEG: $AL = 0 - 5 = \mathbf{FBh}$ ($251 = 256 - 5$). Sonuç $\neq 0$ olduğundan $CF = 1$; $SF = 1$.

Adım 2 / 7

DEĞİŞKENLER

AL = **FBh**

AX = **00FBh**

CF = **1**

ZF = **0**

SF = **1**

OF = **0**

NEG ve CBW

```
1 MOV AL, 5
2 NEG AL      ; AL = 0 - 5
3
4 ► ADD AL, 5  ; -5 + 5
5
6 MOV AL, 0FBh ; -5
7 CBW          ; AL → AX işaretli
8
9 MOV AL, 80h  ; -128
10 NEG AL      ; -(-128) = ?
```

$FBh + 05h = 100h \rightarrow AL = \mathbf{00h}$: $CF = 1$ (elde düştü), $ZF = 1$, $OF = 0$ (işaretili sonuç doğru).

Adım 3 / 7

DEĞİŞKENLER

AL = **00h**

AX = **0000h**

CF = **1**

ZF = **1**

SF = **0**

OF = **0**

NEG ve CBW

```
1 MOV AL, 5
2 NEG AL      ; AL = 0 - 5
3 ADD AL, 5    ; -5 + 5
4 ► MOV AL, 0FBh ; -5
5 CBW          ; AL → AX işaretli
6 MOV AL, 80h  ; -128
7 NEG AL       ; -(-128) = ?
```

AL = FBh. AH hâlâ 00h → AX = 00FBh = +251 (işaretsiz).

Adım 4 / 7

DEĞİŞKENLER

AL = FBh

AX = 00FBh

CF = 1

ZF = 1

SF = 0

OF = 0

NEG ve CBW

```
1 MOV AL, 5
2 NEG AL      ; AL = 0 - 5
3 ADD AL, 5    ; -5 + 5
4 MOV AL, 0FBh ; -5
5 ► CBW       ; AL → AX işaretli
6 MOV AL, 80h  ; -128
7 NEG AL      ; -(-128) = ?
```

CBW: AL'nin MSB'si (1) AH'ye kopyalanır → AX = **FFFBh**
= -5 (16 bit). Bayraklar değişmez.

Adım 5 / 7

DEĞİŞKENLER

AL = FBh

AX = FFBh

CF = 1

ZF = 1

SF = 0

OF = 0

NEG ve CBW

```
1 MOV AL, 5
2 NEG AL      ; AL = 0 - 5
3 ADD AL, 5    ; -5 + 5
4 MOV AL, 0FBh ; -5
5 CBW          ; AL → AX işaretli
▶ MOV AL, 80h  ; -128
7 NEG AL      ; -(-128) = ?
```

AL = 80h = -128 (AH yine FFh).

Adım 6 / 7

DEĞİŞKENLER

AL = 80h

AX = FF80h

CF = 1

ZF = 1

SF = 0

OF = 0

NEG ve CBW

```
1 MOV AL, 5
2 NEG AL      ; AL = 0 - 5
3 ADD AL, 5    ; -5 + 5
4 MOV AL, 0FBh ; -5
5 CBW          ; AL → AX işaretli
6 MOV AL, 80h  ; -128
▶ NEG AL       ; -(-128) = ?
```

$-(-128) = +128$, 8 bite sığmaz: NEG sonucu yine **80h**,
OF = 1.

Adım 7 / 7

DEĞİŞKENLER

AL = **80h**

AX = **FF80h**

CF = **1**

ZF = **0**

SF = **1**

OF = **1**

Sabit Noktalı İşaretili Kesirler

8 bitin son 3'ü kesir olsun: **Q5.3** biçimi (işaret dahil 5 tamsayı + 3 kesir biti). Örnek: **-6,75**.

-16	8	4	2	1		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$
					,			

Basamak ağırlıkları: en soldaki bit **-16** (işaret biti), virgülden sonra $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$. Aralık $-16 \dots +15,875$, adım $1/8 = 0,125$.

Adım 1 / 5

İşlemci için CAh sıradan bir bayttır; virgülün yeri programcının kabulüdür. Toplama aynen çalışır: $-6,75 + 2,5 \rightarrow \text{CAh} + 14\text{h} = \text{DEh} = 11011,110\text{b} = -16 + 11,75 = \textbf{-4,25} \checkmark$

Sabit Noktalı İşaretsiz Kesirler

8 bitin son 3'ü kesir olsun: **Q5.3** biçimi (işaret dahil 5 tamsayı + 3 kesir biti). Örnek: **-6,75**.



$$+6,75 = 4 + 2 + \frac{1}{2} + \frac{1}{4} \rightarrow \mathbf{00110,110}.$$

Adım 2 / 5

İşlemci için CAh sıradan bir bayttır; virgülün yeri programcının kabulüdür. Toplama aynen çalışır: $-6,75 + 2,5 \rightarrow \text{CAh} + 14\text{h} = \mathbf{DEh} = 11011,110\text{b} = -16 + 11,75 = \mathbf{-4,25} \checkmark$

Sabit Noktalı İşaretsiz Kesirler

8 bitin son 3'ü kesir olsun: **Q5.3** biçimi (işaret dahil 5 tamsayı + 3 kesir biti). Örnek: **-6,75**.

	-16	8	4	2	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	
+6,75	0	0	1	1	0	1	1	0	36h
-6,75	1	1	0	0	1	0	1	0	CAh

ikiye tümleyen (tersle + 1; virgöl yok sayılır): 00110110 → 11001001 + 1 = **11001,010**.

Adım 3 / 5

İşlemci için CAh sıradan bir bayttır; virgölün yeri programcının kabulüdür. Toplama aynen çalışır: $-6,75 + 2,5 \rightarrow \text{CAh} + 14\text{h} = \text{DEh} = 11011,110\text{b} = -16 + 11,75 = \textbf{-4,25} \checkmark$

Sabit Noktalı İşaretili Kesirler

8 bitin son 3'ü kesir olsun: **Q5.3** biçimi (işaret dahil 5 tamsayı + 3 kesir biti). Örnek: **-6,75**.

	-16	8	4	2	1		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	
+6,75	0	0	1	1	0	,	1	1	0	36h
-6,75	1	1	0	0	1	,	0	1	0	CAh

$$-16 + 8 + 1 + \frac{1}{4} = -6,75 \checkmark$$

Kontrol: işaret biti -16 değerlidir: $-16 + 8 + 1 + 0,25 = -6,75 \checkmark$

Adım 4 / 5

İşlemci için CAh sıradan bir bayttır; virgülün yeri programcının kabulüdür. Toplama aynen çalışır: $-6,75 + 2,5 \rightarrow \text{CAh} + 14\text{h} = \text{DEh} = 11011,110\text{b} = -16 + 11,75 = -4,25 \checkmark$

Sabit Noktalı İşaretili Kesirler

8 bitin son 3'ü kesir olsun: **Q5.3** biçimi (işaret dahil 5 tamsayı + 3 kesir biti). Örnek: **-6,75**.

	-16	8	4	2	1		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	
+6,75	0	0	1	1	0	,	1	1	0	36h
-6,75	1	1	0	0	1	,	0	1	0	CAh

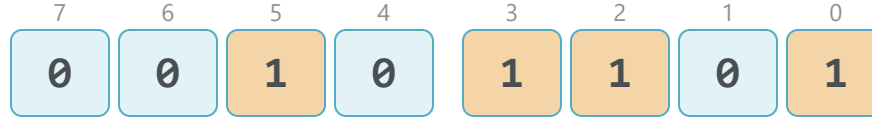
$$-6,75 \times 2^3 = -54 \rightarrow 256 - 54 = 202 = \text{CAh}$$

Kısa yol: sayıyı $2^3 = 8$ ile çarp, tamsayı gibi ikiye tümleyen al: $-54 \rightarrow \text{CAh}$, sonra virgülü 3 bit sola koy.

Adım 5 / 5

İşlemci için CAh sıradan bir bayttır; virgülün yeri programcının kabulüdür. Toplama aynen çalışır: $-6,75 + 2,5 \rightarrow \text{CAh} + 14\text{h} = \text{DEh} = 11011,110\text{b} = -16 + 11,75 = -4,25 \checkmark$

Bit Gezgini



NOT (1'e t.)

NEG (2'ye t.)

+1

SHL

sıfırla

onaltılı
2Dh

işaretsiz
45

2'ye tümleyen
45

işaret-büyükük
45

1'e tümleyen
45

paketli BCD
geçersiz

Gray kodu
00111011

PF (1 sayısı)
1 (4)

Bitlere tıkla ya da bir işlem seç.

Bitlere tıkla: aynı 8 bitin işaretsiz, işaretli, BCD ve Gray kodu yorumları birlikte görünür.

İkili Tabanda Toplama

- İki sayının her basamağı için dört durum vardır: $0+0$, $0+1$, $1+0$ ve $1+1$.
- En düşük basamaktan başlanarak iki sayının aynı basamakları toplanır.
- Basamak toplamı 0 ya da 1 ise doğrudan yazılır.
- Toplam 2 (10b) ise **elde (carry)** oluşur: basamağa 0 yazılır, 1 bir sonraki basamağa eklenir.
- Önceki eldeyle toplam 3 (11b) olabilir: 1 yazılır, 1 elde.

a	b	elde _{giriş}	toplam	elde _{çıkış}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
1	1	1	1	1

$0 + 0$ elde 0 $= 0$ yaz 0	$0 + 1$ elde 0 $= 1$ yaz 1	$1 + 0$ elde 0 $= 1$ yaz 1	$1 + 1$ elde 1 $= 10$ yaz 0
-------------------------------	-------------------------------	-------------------------------	---------------------------------------

Bu tablo bir **tam toplayıcı** (full adder) devresinin doğruluk tablosudur; 8 tanesi zincirlenince 8 bitlik toplayıcı olur.

Toplama: 110001101 + 100001011

elde

397	1	1	0	0	0	1	1	0	1
+ 267	1	0	0	0	0	1	0	1	1

110001101b (397) + 100001011b (267). Sağdan sola, sütun sütun toplanır.

Toplama: 110001101 + 100001011

elde

397

+ 267

1	1	0	0	0	1	1	0	1
1	0	0	0	0	1	0	1	1

0

Basamak 0: 1 + 1 = 2 → 0 yaz, 1 elde.

Toplama: 110001101 + 100001011

elde

397

+ 267

1

1

0

0

0

1

1

0

1

1

0

0

0

0

1

0

1

1

0

0

Basamak 1: 0 + 1 + 1 (elde) = 2 → 0 yaz, 1 elde.

Toplama: 110001101 + 100001011

elde							1	1	
397	1	1	0	0	0	1	1	0	1
+ 267	1	0	0	0	0	1	0	1	1
							0	0	0

Basamak 2: 1 + 0 + 1 (elde) = 2 → 0 yaz, 1 elde.

Toplama: 110001101 + 100001011

elde						1	1	1	
397	1	1	0	0	0	1	1	0	1
+ 267	1	0	0	0	0	1	0	1	1
						1	0	0	0

Basamak 3: $1 + 1 + 1$ (elde) = 3 → **1** yaz, 1 elde.

Toplama: 110001101 + 100001011

elde					1	1	1	1	
397	1	1	0	0	0	1	1	0	1
+ 267	1	0	0	0	0	1	0	1	1
					1	1	0	0	0

Basamak 4: 0 + 0 + 1 (elde) = 1 → 1 yaz.

Toplama: 110001101 + 100001011

elde					1	1	1	1	
397	1	1	0	0	0	1	1	0	1
+ 267	1	0	0	0	0	1	0	1	1
				0	1	1	0	0	0

Basamak 5: 0 + 0 = 0 → 0 yaz.

Toplama: 110001101 + 100001011

elde					1	1	1	1	
397	1	1	0	0	0	1	1	0	1
+ 267	1	0	0	0	0	1	0	1	1
			0	0	1	1	0	0	0

Basamak 6: $0 + 0 = 0 \rightarrow 0$ yaz.

Toplama: 110001101 + 100001011

elde

397

+ 267

110001101

100001011

10001000

Basamak 7: 1 + 0 = 1 → 1 yaz.

Toplama: 110001101 + 100001011

elde					1	1	1	1	
397	1	1	0	0	0	1	1	0	1
+ 267	1	0	0	0	0	1	0	1	1
	0	1	0	0	1	1	0	0	0

Basamak 8: 1 + 1 = 2 → 0 yaz, 1 elde.

Toplama: 110001101 + 100001011

elde	1				1	1	1	1	
397		1	1	0	0	0	1	1	0
+ 267		1	0	0	0	0	1	0	1
= 664	1	0	1	0	0	1	1	0	0

Basamak 9: $0 + 0 + 1$ (elde) = 1 → **1** yaz. Sonuç **1010011000b** = 664 ✓ (10. bit, 9 bitlik toplama sığmadı → elde).

İkili Tabanda Çıkarma

- Çıkarma, **tümleyenle toplamaya** dönüştürülür: $A - B = A + (-B)$.
- Çıkarılacak sayının bütün bitleri ters çevrilir.
- Tümleyen alınmış sayıya 1 eklenir (ikiye tümleyen).
- Elde edilen sayı ile ilk sayı toplanır; bit genişliğini aşan elde **atılır**.
- Böylece işlemcide ayrı bir çıkarıcı devreye gerek kalmaz.

İlk sayı 1011 (11), ikinci sayı 0110 (6).

Adım 1 / 7

8086'da SUB komutu bu atılan eldeyi tersine çevirerek CF'ye yazar: elde 1 ise ödünç yok (CF = 0).

İkili Tabanda Çıkarma

- Çıkarma, **tümleyenle toplamaya** dönüştürülür: $A - B = A + (-B)$.
- Çıkarılacak sayının bütün bitleri ters çevrilir.
- Tümleyen alınmış sayıya 1 eklenir (ikiye tümleyen).
- Elde edilen sayı ile ilk sayı toplanır; bit genişliğini aşan elde **atılır**.
- Böylece işlemcide ayrı bir çıkarıcı devreye gerek kalmaz.

8086'da SUB komutu bu atılan eldeyi tersine çevirerek CF'ye yazar: elde 1 ise ödünç yok (CF = 0).

A = 1 0 1 1 11

A = 1011.

Adım 2 / 7

İkili Tabanda Çıkarma

- Çıkarma, **tümleyenle toplamaya** dönüştürülür: $A - B = A + (-B)$.
- Çıkarılacak sayının bütün bitleri ters çevrilir.
- Tümleyen alınmış sayıya 1 eklenir (ikiye tümleyen).
- Elde edilen sayı ile ilk sayı toplanır; bit genişliğini aşan elde **atılır**.
- Böylece işlemcide ayrı bir çıkarıcı devreye gerek kalmaz.

8086'da SUB komutu bu atılan eldeyi tersine çevirerek CF'ye yazar: elde 1 ise ödünç yok (CF = 0).

A =	1	0	1	1	11
B =	0	1	1	0	6

$$B = 0110.$$

Adım 3 / 7

İkili Tabanda Çıkarma

- Çıkarma, **tümleyenle toplamaya** dönüştürülür: $A - B = A + (-B)$.
- Çıkarılacak sayının bütün bitleri ters çevrilir.
- Tümleyen alınmış sayıya 1 eklenir (ikiye tümleyen).
- Elde edilen sayı ile ilk sayı toplanır; bit genişliğini aşan elde **atılır**.
- Böylece işlemcide ayrı bir çıkarıcı devreye gerek kalmaz.

8086'da SUB komutu bu atılan eldeyi tersine çevirerek CF'ye yazar: elde 1 ise ödünç yok (CF = 0).

A =	1	0	1	1	11
B =	0	1	1	0	6
NOT B =	1	0	0	1	birler tümleyeni

B'nin bitleri ters çevrilir: 1001.

Adım 4 / 7

İkili Tabanda Çıkarma

- Çıkarma, **tümleyenle toplamaya** dönüştürülür: $A - B = A + (-B)$.
- Çıkarılacak sayının bütün bitleri ters çevrilir.
- Tümleyen alınmış sayıya 1 eklenir (ikiye tümleyen).
- Elde edilen sayı ile ilk sayı toplanır; bit genişliğini aşan elde **atılır**.
- Böylece işlemcide ayrı bir çıkarıcı devreye gerek kalmaz.

8086'da SUB komutu bu atılan eldeyi tersine çevirerek CF'ye yazar: elde 1 ise ödünç yok (CF = 0).

A =	1	0	1	1	11
B =	0	1	1	0	6
NOT B =	1	0	0	1	birler tümleyeni
+1 →	1	0	1	0	-B (ikiye t.)

1 eklenir: **1010** = -6 (4 bit).

Adım 5 / 7

İkili Tabanda Çıkarma

- Çıkarma, **tümleyenle toplamaya** dönüştürülür: $A - B = A + (-B)$.
- Çıkarılacak sayının bütün bitleri ters çevrilir.
- Tümleyen alınmış sayıya 1 eklenir (ikiye tümleyen).
- Elde edilen sayı ile ilk sayı toplanır; bit genişliğini aşan elde **atılır**.
- Böylece işlemcide ayrı bir çıkarıcı devreye gerek kalmaz.

8086'da SUB komutu bu atılan eldeyi tersine çevirerek CF'ye yazar: elde 1 ise ödünç yok (CF = 0).

A =	1	0	1	1	11	
B =	0	1	1	0	6	
NOT B =	1	0	0	1	birler tümleyeni	
+1 →	1	0	1	0	-B (ikiye t.)	
A + (-B) =	±	0	1	0	1	5 bit: soldaki elde atılır

$$1011 + 1010 = \mathbf{10101} \text{ (5 bit).}$$

Adım 6 / 7

İkili Tabanda Çıkarma

- Çıkarma, **tümleyenle toplamaya** dönüştürülür: $A - B = A + (-B)$.
- Çıkarılacak sayının bütün bitleri ters çevrilir.
- Tümleyen alınmış sayıya 1 eklenir (ikiye tümleyen).
- Elde edilen sayı ile ilk sayı toplanır; bit genişliğini aşan elde **atılır**.
- Böylece işlemcide ayrı bir çıkarıcı devreye gerek kalmaz.

8086'da SUB komutu bu atılan eldeyi tersine çevirerek CF'ye yazar: elde 1 ise ödünç yok (CF = 0).

A =	1	0	1	1	11	
B =	0	1	1	0	6	
NOT B =	1	0	0	1	birler tümleyeni	
+1 →	1	0	1	0	-B (ikiye t.)	
A + (-B) =	±	0	1	0	1	5 bit: soldaki elde atılır
sonuç =	0	1	0	1	5 ✓	

4 bite sığmayan soldaki 1 atılır: **0101** = 5 = 11 – 6 ✓

Adım 7 / 7

Onaltılık Toplama: 9A7Ch + 5B96h

elde

9

A

7

C

9A7Ch = 39548

+

5

B

9

6

5B96h = 23446

Onaltılık toplama: sağdan sola basamak basamak. Basamak toplamı 15'i aşarsa 16 çıkarılır, sola **1 elde** verilir.

Adım 1 / 6

Basamak değerleri onluk düşünülür (A = 10 ... F = 15). Toplam 16 veya fazlaysa basamağa toplam – 16 yazılır.

Onaltılık Toplama: 9A7Ch + 5B96h

elde

1

$$\begin{array}{rcccc} & 9 & A & 7 & C \\ + & 5 & B & 9 & 6 \\ \hline & & & & 2 \end{array}$$

$$C + 6 = 12 + 6 = 18$$

$$18 - 16 = 2, \text{ elde } 1$$

$$C + 6 = 18 \geq 16 \rightarrow \text{yaz } 2, \text{ elde } 1.$$

Adım 2 / 6

Basamak değerleri onluk düşünülür ($A = 10 \dots F = 15$). Toplam 16 veya fazlaysa basamağa toplam – 16 yazılır.

Onaltılık Toplama: 9A7Ch + 5B96h

elde		1	1		
	9	A	7	C	
+	5	B	9	6	
			1	2	

$$7 + 9 + 1 = 17$$

$$17 - 16 = 1, \text{ elde } 1$$

$$7 + 9 + 1 = 17 \rightarrow \text{yaz } 1, \text{ elde } 1.$$

Adım 3 / 6

Basamak değerleri onluk düşünülür (A = 10 ... F = 15). Toplam 16 veya fazlaysa basamağa toplam - 16 yazılır.

Onaltılık Toplama: 9A7Ch + 5B96h

elde	1	1	1		
	9	A	7	C	
+	5	B	9	6	
		6	1	2	

$A + B + 1 = 10 + 11 + 1 = 22$
 $22 - 16 = 6$, elde 1

$A + B + 1 = 22 \rightarrow$ yaz **6**, elde 1.

Adım 4 / 6

Basamak değerleri onluk düşünülür ($A = 10 \dots F = 15$). Toplam 16 veya fazlaysa basamağa toplam – 16 yazılır.

Onaltılık Toplama: 9A7Ch + 5B96h

elde	1	1	1		
	9	A	7	C	
+	5	B	9	6	
	F	6	1	2	

$$9 + 5 + 1 = 15 = F$$

16'dan küçük, elde yok

$$9 + 5 + 1 = 15 \rightarrow \text{yaz } F, \text{ elde yok.}$$

Adım 5 / 6

Basamak değerleri onluk düşünülür (A = 10 ... F = 15). Toplam 16 veya fazlaysa basamağa toplam – 16 yazılır.

Onaltılık Toplama: 9A7Ch + 5B96h

elde	1	1	1		
	9	A	7	C	
+	5	B	9	6	
	F	6	1	2	= F612h

Kontrol: 39548 + 23446
= 62994 ✓

Sonuç **F612h**. 8086'da `ADD AX, BX` aynı sonucu verir; son basamaktan elde çıkmadığı için CF = 0.

Adım 6 / 6

Basamak değerleri onluk düşünülür (A = 10 ... F = 15). Toplam 16 veya fazlaysa basamağa toplam – 16 yazılır.

Onaltılık Çıkarma: 9A7Ch – 5B96h

ödünç

–

9

A

7

C

5

B

9

6

Üstteki basamak küçükse
soldan 16 ödünç alınır

Onaltılık çıkarma: üstteki basamak yetmezse soldaki basamaktan **1 ödünç** alınır; o basamağa 16 eklenir.

Adım 1 / 6

“–1” işaretleri ödünç veren basamakları gösterir. Aynı işlem tümleyenle toplamaya da çevrilebilir.

Onaltılık Çıkarma: 9A7Ch – 5B96h

ödünç

	9	A	7	C	$C - 6 = 12 - 6 = 6$
-	5	B	9	6	ödünç yok
				6	

$$C - 6 = 6.$$

Adım 2 / 6

"-1" işaretleri ödünç veren basamakları gösterir. Aynı işlem tümleyenle toplamaya da çevrilebilir.

Onaltılık Çıkarma: 9A7Ch – 5B96h

ödünç

-1

9

A

7

C

7 – 9 olmaz → 16 ödünç

-

5

B

9

6

7 + 16 – 9 = 14 = E

E

6

$7 < 9 \rightarrow$ soldan ödünç: $7 + 16 - 9 = 14 \rightarrow \mathbf{E}$; soldaki A, 9 olur.

Adım 3 / 6

“-1” işaretleri ödünç veren basamakları gösterir. Aynı işlem tümleyenle toplamaya da çevrilebilir.

Onaltılık Çıkarma: 9A7Ch – 5B96h

ödünç	-1	-1			
	9	A	7	C	$A - 1 = 9, 9 - B \text{ olmaz}$
-	5	B	9	6	$9 + 16 - 11 = 14 = E$
		E	E	6	

$9 < B \rightarrow$ yine ödünç: $9 + 16 - 11 = 14 \rightarrow E$; soldaki 9, 8 olur.

Adım 4 / 6

"-1" işaretleri ödünç veren basamakları gösterir. Aynı işlem tümleyenle toplamaya da çevrilebilir.

Onaltılık Çıkarma: 9A7Ch – 5B96h

ödünç

-1

-1

9

A

7

C

9 – 1 = 8

-

5

B

9

6

8 – 5 = 3

3

E

E

6

8 – 5 = 3.

Adım 5 / 6

“-1” işaretleri ödünç veren basamakları gösterir. Aynı işlem tümleyenle toplamaya da çevrilebilir.

Onaltılık Çıkarma: 9A7Ch – 5B96h

ödünç	-1	-1			
	9	A	7	C	
-	5	B	9	6	Kontrol: 39548 – 23446
					= 16102 ✓
	3	E	E	6	= 3EE6h

Sonuç **3EE6h**. En soldan ödünç gerekmediği için 8086'da `SUB AX, BX` sonrası CF = 0.

Adım 6 / 6

“–1” işaretleri ödünç veren basamakları gösterir. Aynı işlem tümleyenle toplamaya da çevrilebilir.

15'e ve 16'ya Tümleyen

15'e tümleyen **FFFFh – 5B96h**

= A469h

her basamak F'den çıkarılır

5B96h'nin **15'e tümleyeni**: her basamak F'den çıkarılır (F–5, F–B, F–9, F–6) → A469h.
İkilide bu NOT işlemidir.

Adım 1 / 5

İkilideki 1'e / ikiye tümleyen onaltılık karşılığıdır. 8086'da SUB AX, BX (AX = 5B96h, BX = 9A7Ch) doğrudan **C11Ah** verir ve ödünç olduğu için **CF = 1**; C11Ah işaretli yorumda –16102'dir.

15'e ve 16'ya Tümleyen

15'e tümleyen $\text{FFFFh} - 5\text{B96h}$

$= \text{A469h}$

her basamak F'den çıkarılır

16'ya tümleyen $\text{A469h} + 1$

$= \text{A46Ah}$

$= 10000\text{h} - 5\text{B96h}$

16'ya tümleyen = 15'e tümleyen + 1 = A46Ah. Bu, 4 basamakta -5B96h 'nin gösterimidir (NEG).

Adım 2 / 5

İkilideki 1'e / ikiye tümleyeninin onaltılık karşılığıdır. 8086'da `SUB AX, BX` ($\text{AX} = 5\text{B96h}$, $\text{BX} = 9\text{A7Ch}$) doğrudan **C11Ah** verir ve ödünç olduğu için **CF = 1**; C11Ah işaretli yorumda -16102 'dir.

15'e ve 16'ya Tümleyen

15'e tümleyen $\text{FFFFh} - 5\text{B96h} = \text{A469h}$ her basamak F'den çıkarılır

16'ya tümleyen $\text{A469h} + 1 = \text{A46Ah}$ = 10000h – 5B96h

$9\text{A7Ch} - 5\text{B96h}$ $9\text{A7Ch} + \text{A46Ah} = 1\ 3\text{EE6h}$ taşan 1 atılır → 3EE6h ✓

$9\text{A7Ch} - 5\text{B96h} = 9\text{A7Ch} + \text{A46Ah} = 13\text{EE6h}$. 4 basamağı aşan 1 atılır: **3EE6h**, doğrudan çıkarmayla aynı sonuç.

Adım 3 / 5

İkilideki 1'e / ikiye tümleyeninin onaltılık karşılığıdır. 8086'da `SUB AX, BX` ($\text{AX} = 5\text{B96h}$, $\text{BX} = 9\text{A7Ch}$) doğrudan **C11Ah** verir ve ödünç olduğu için **CF = 1**; C11Ah işaretli yorumda -16102'dir.

15'e ve 16'ya Tümleyen

15'e tümleyen $\text{FFFFh} - 5\text{B96h} = \text{A469h}$ her basamak F'den çıkarılır

16'ya tümleyen $\text{A469h} + 1 = \text{A46Ah}$ = 10000h – 5B96h

$9\text{A7C} - 5\text{B96}$ $9\text{A7Ch} + \text{A46Ah} = 1\ 3\text{EE6h}$ taşan 1 atılır → 3EE6h ✓

$5\text{B96} - 9\text{A7C}$ $5\text{B96h} + 6584\text{h} = \text{C11Ah}$ taşan yok → sonuç negatif

Ters yön: $5\text{B96h} - 9\text{A7Ch} = 5\text{B96h} + 6584\text{h}$ (9A7Ch'nin 16'ya tümleyeni) = C11Ah. Taşan basamak **yok** → sonuç negatiftir.

Adım 4 / 5

İkilideki 1'e / ikiye tümleyeninin onaltılık karşılığıdır. 8086'da `SUB AX, BX` (AX = 5B96h, BX = 9A7Ch) doğrudan **C11Ah** verir ve ödünç olduğu için **CF = 1**; C11Ah işaretli yorumda -16102'dir.

15'e ve 16'ya Tümleyen

15'e tümleyen $\text{FFFFh} - 5\text{B96h} = \text{A469h}$ her basamak F'den çıkarılır

16'ya tümleyen $\text{A469h} + 1 = \text{A46Ah}$ $= 10000\text{h} - 5\text{B96h}$

$9\text{A7C} - 5\text{B96}$ $9\text{A7Ch} + \text{A46Ah} = 1\ 3\text{EE6h}$ taşan 1 atılır $\rightarrow 3\text{EE6h} \checkmark$

$5\text{B96} - 9\text{A7C}$ $5\text{B96h} + 6584\text{h} = \text{C11Ah}$ taşan yok $\rightarrow$ sonuç negatif

$-(16'\text{ya t. C11Ah}) = -3\text{EE6h}$ $10000\text{h} - \text{C11Ah} = 3\text{EE6h}$

Büüklüğü bulmak için tekrar 16'ya tümleyen alınır: $\text{C11Ah} \rightarrow 3\text{EE6h}$, yani sonuç **-3EE6h** (-16102).

Adım 5 / 5

İkilideki 1'e / ikiye tümleyeninin onaltılık karşılığıdır. 8086'da `SUB AX, BX` ($\text{AX} = 5\text{B96h}$, $\text{BX} = 9\text{A7Ch}$) doğrudan **C11Ah** verir ve ödünç olduğu için **$\text{CF} = 1$** ; C11Ah işaretli yorumda -16102 'dir.

İşaretili Taşma Kuralı

İşlem	İşaretler	Örnek (8 bit)	İşaretili yorum	OF
A + B	+ ile +	50h + 40h = 90h	80 + 64 → -112 X	1
A + B	- ile -	9Ch + CEh = 6Ah	-100 + (-50) → +106 X	1
A + B	farklı	64h + CEh = 32h	100 + (-50) = 50 ✓	0 (asla)
A - B	+ eksi -	64h - CEh = 96h	100 - (-50) → -106 X	1
A - B	- eksi +	9Ch - 32h = 6Ah	-100 - 50 → +106 X	1
A - B	aynı	9Ch - CEh = CEh	-100 - (-50) = -50 ✓	0 (asla)

- Taşma yalnızca **aynı işaretili iki sayı toplanırken** ya da **farklı işaretili iki sayı çıkarılırken** olabilir; belirtisi, sonucun işaretinin beklenenin tersi çıkmasıdır.
- Donanım kuralı: işaret bitine **giren elde** ile işaret bitinden **çıkan elde** farklıysa OF = 1. CF işaretili yorumda anlamsızdır (3. satırda CF = 1 ama sonuç doğru).
- Genişlikleri farklı işaretili sayılar toplanmadan önce kısa olan **işaretle genişletilir**: FBh (-5) + 0100h → FFFBh + 0100h = 00FBh = 251 ✓; sıfırla genişletilse 01FBh = 507 X.

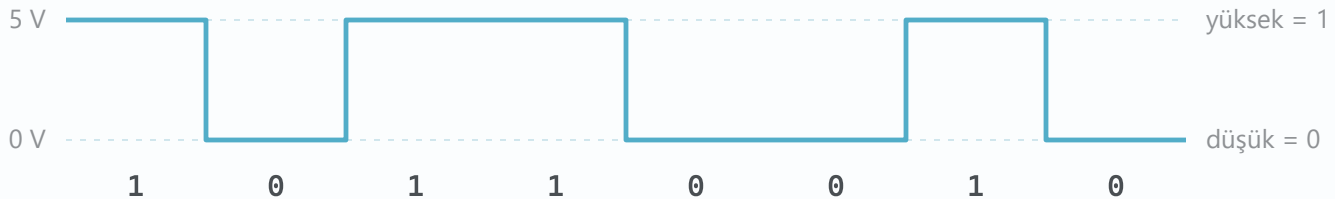
Desteklenen İşlemler

Op.	Anlam	Örnek → değer
~	not (tüm bitleri tersler)	~0Fh → F0h (8 bit)
*	çarpma	3*4 → 12
/	bölme (tam sayı)	17/5 → 3
%	mod (kalan)	17%5 → 2
+	toplama	40h+2 → 42h
-	çıkarma (ve tekli -)	-5 → FBh
<<	sola kaydırma	1<<4 → 16
>>	sağa kaydırma	80h>>3 → 10h
&	bit düzeyinde VE	0F0h&3Ch → 30h
^	bit düzeyinde XOR	0Fh^05h → 0Ah
	bit düzeyinde VEYA	0F0h 0Fh → FFh

- emu8086 assembler'ının **sabit ifadelerde** desteklediği işleçlerdir.
- İfade **derleme zamanında** hesaplanır: `MOV AX, 3*4+1` → `MOV AX, 13`.
- İşlemci çalışırken bunları yapmaz; çalışma zamanında karşılıkları `NOT`, `MUL`, `DIV`, `ADD`, `SUB`, `SHL`, `SHR`, `AND`, `XOR`, `OR` komutlarıdır.
- Yalnızca sabitlerle çalışır: `MOV AX, BX*2` geçersizdir.

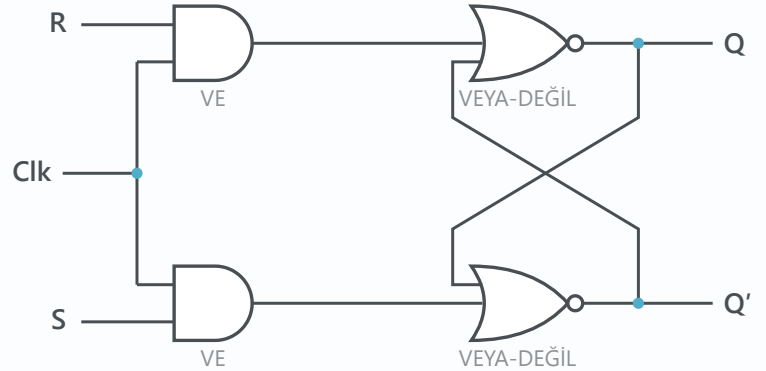
Sayısal Sistem ve İkili Sinyaller

- Sayısal sistemdeki yazılımlar bir **rakam topluluğunu** temsil eder; bu rakamlar veri ve komut öbeklerini oluşturur.
- Yazılımı oluşturan veriler ve komutlar, gerçek elektrik devrelerinde **gerilim ve akım** kullanılarak yaratılır.
- Sayısal bilgisayarlarda **yüksek** ve **düşük** olmak üzere iki çeşit sinyal kullanılır: 1 (yüksek), 0 (düşük).
- Gerilim tam 5 V ya da 0 V olmak zorunda değildir; eşik aralıkları vardır (TTL: $\geq 2,0\text{ V} \rightarrow 1$, $\leq 0,8\text{ V} \rightarrow 0$). Bu tolerans sayısal devreleri gürültüye karşı **dayanıklı** yapar.
- İkili sinyaller bilgisayarların temel iletişim dilidir; "bit" ikili sayının her bir basamağıdır.



Flip-Flop Devreleri

- Sayısal elektronikte kullanılan **temel bellek elemanlarıdır**: 1 bit saklar.
- Bir bilgiyi bir sonraki **saat darbesine** kadar saklarlar.
- Girdi ve saat sinyali olarak bir çıktı (Q ve tersi Q') üretir.
- Saat sinyali belirli bir anı temsil eder; girdi bu saat darbesi sırasında flip-flop'a **yüklenir**.
- Yazmaçlar (AX, BX ...) yan yana konmuş 16 flip-flop'tan oluşur.

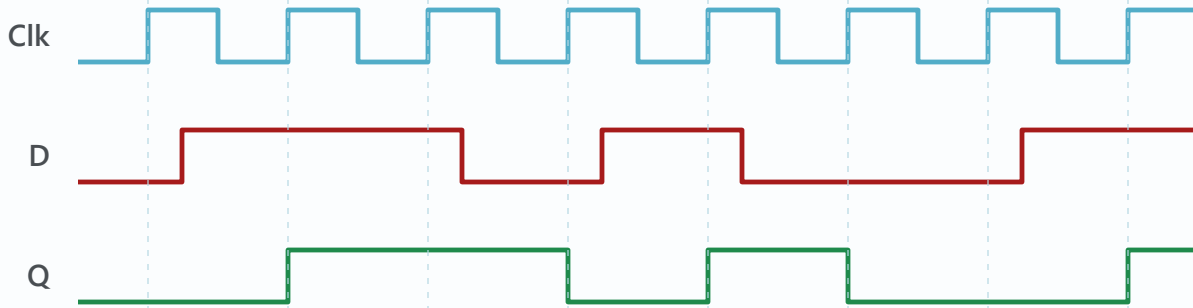


Flip-Flop Türleri

Tür	Girişler	Davranış (saat darbesinde)	Karakteristik
SR (Set-Reset)	S, R	$S = 1 \rightarrow Q = 1$ (set); $R = 1 \rightarrow Q = 0$ (reset); $S = R = 0 \rightarrow$ değişmez	$S = R = 1$ yasak
D (Data)	D	Tek veri girişini saat sinyaliyle çıkışa yükler: $Q \leftarrow D$	yazmaç/bellek hücresi
JK	J, K	$J = 1 \rightarrow$ set, $K = 1 \rightarrow$ reset, $J = K = 0 \rightarrow$ değişmez, $J = K = 1 \rightarrow$ ters çevir	SR'nin yasak durumu yok
T (Toggle)	T	$T = 1$ iken her darbeye çıkışını değiştirir; $T = 0 \rightarrow$ değişmez	sayaç, frekans bölücü

JK, set ve reset özelliklerine ek olarak $J = K = 1$ iken **tersleme (toggle)** yapar; onu esnek yapan bu üçüncü özelliktir.

D Flip-Flop Zamanlaması

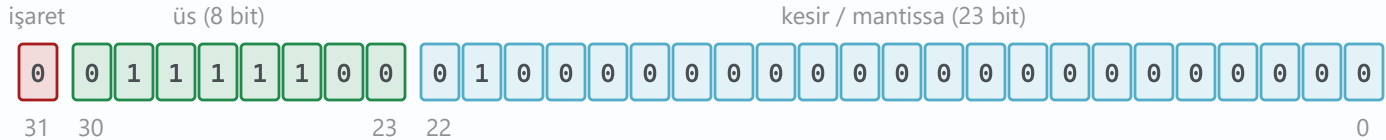


Q, yalnızca saatin yükselen kenarında (kesikli çizgiler) D'nin o anki değerini alır

- D girişi saat darbeleri arasında istediği kadar değişebilir; çıkışı **etkilemez**.
- Yükselen kenarda D örneklenir ve bir sonraki kenara kadar Q'da **saklanır**: 1 bitlik bellek.
- 8086'nın tüm iç işlemleri bu şekilde saat turlarına bölünür (örneğin 5 MHz $\rightarrow$ 200 ns'lik tur).

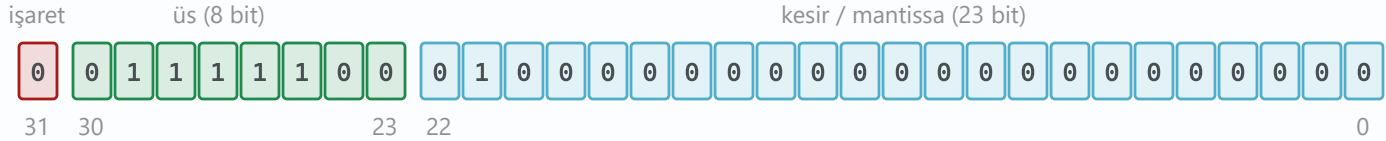
Kayan Nokta: IEEE 754

- Kesirli sayılar için **IEEE 754** standardı kullanılır; tek duyarlıklı (single) sayı **32 bittir**.
- 32 bit üç gruba ayrılır: **işaret** (sign, 1 bit), **üs** (exponent, 8 bit) ve **mantissa** (fraction, 23 bit).
- İlk bit işaret bitidir: 0 pozitif, 1 negatif.
- Üs **127 kaydırmalı** (biased) saklanır: gerçek üs = $e - 127$.
- Mantissa, normalize sayının **kesirli kısmıdır**; baştaki 1 yazılmaz (gizli bit): değer = $(-1)^s \times 1, f \times 2^{e-127}$.



8086 kayan nokta komutu içermez; bu işlemler 8087 matematik yardımcı işlemcisi ya da yazılımla yapılır. 64 bitlik double: 1 + 11 + 52 bit, kaydırma 1023.

IEEE 754 Çözümleme

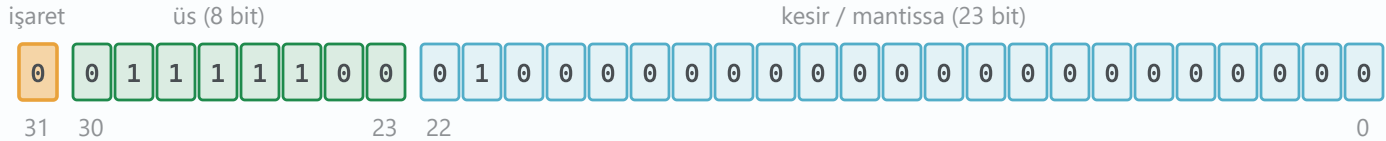


Örnek: 0 01111100 010000000000000000000000 = 3E200000h.

Adım 1 / 5

Sonuç $(+1) \times (2^{-3}) \times 1,25$; onluk değeri **0,15625**'tir.

IEEE 754 Çözümleme

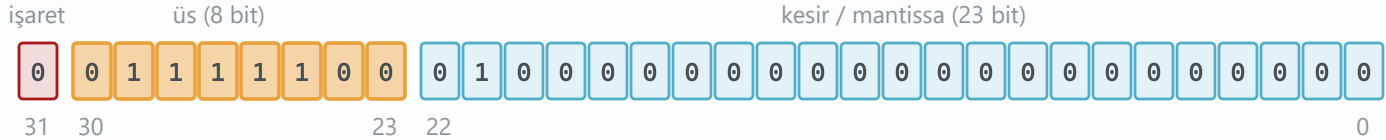


İşaret biti 0 → sayı **pozitif**: (+1).

Adım 2 / 5

Sonuç $(+1) \times (2^{-3}) \times 1,25$; onluk değeri **0,15625**'tir.

IEEE 754 Çözümleme

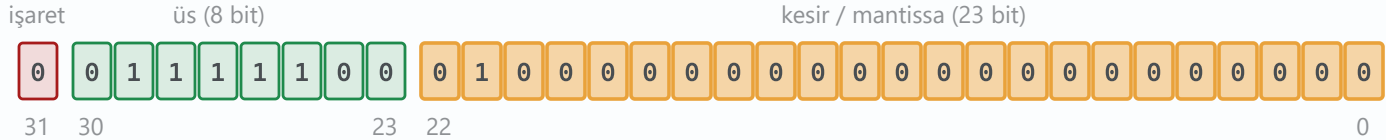


Üs 01111100b = 124 → gerçek üs $124 - 127 = -3 \rightarrow 2^{-3}$.

Adım 3 / 5

Sonuç $(+1) \times (2^{-3}) \times 1,25$; onluk değeri **0,15625**'tir.

IEEE 754 Çözümleme

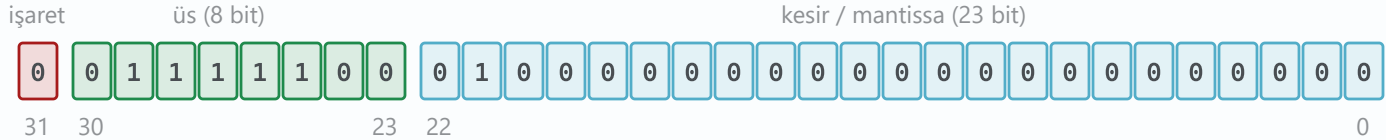


Mantissa $0100...0 = 0 \times 2^{-1} + 1 \times 2^{-2} = 0,25 \rightarrow$ gizli bitle **1,25**.

Adım 4 / 5

Sonuç $(+1) \times (2^{-3}) \times 1,25$; onluk değeri **0,15625**'tir.

IEEE 754 Çözümleme

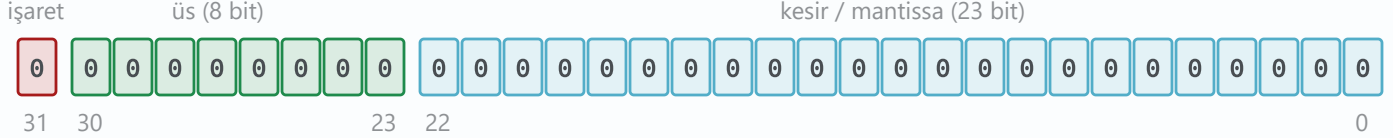


$$\text{Sonuç: } (+1) \times 2^{-3} \times 1,25 = 1,25 / 8 = \mathbf{0,15625}.$$

Adım 5 / 5

Sonuç $(+1) \times (2^{-3}) \times 1,25$; onluk değeri **0,15625**'tir.

IEEE 754 Kodlama: -6,5

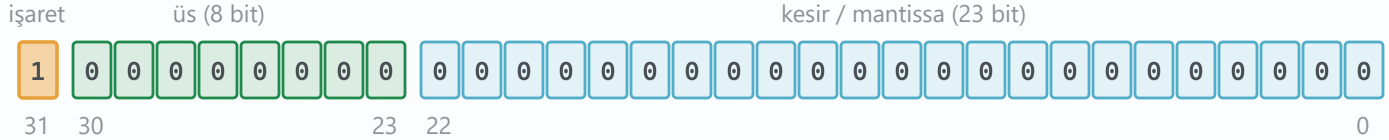


-6,5'i 32 bitlik IEEE 754'e çevirelim.

Adım 1 / 6

Desen	Anlam
$e = 0, f = 0$	± 0
$e = 255, f = 0$	$\pm \infty$ (ör. 1/0)
$e = 255, f \neq 0$	NaN (sayı deęil, ör. 0/0)
$e = 0, f \neq 0$	normalize edilmemiř (çok küçük) sayı

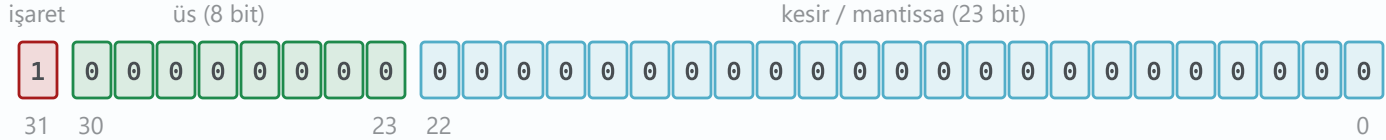
IEEE 754 Kodlama: -6,5

Negatif → işaret biti **1**.

Adım 2 / 6

Desen	Anlam
$e = 0, f = 0$	± 0
$e = 255, f = 0$	$\pm \infty$ (ör. 1/0)
$e = 255, f \neq 0$	NaN (sayı değil, ör. 0/0)
$e = 0, f \neq 0$	normalize edilmemiş (çok küçük) sayı

IEEE 754 Kodlama: -6,5

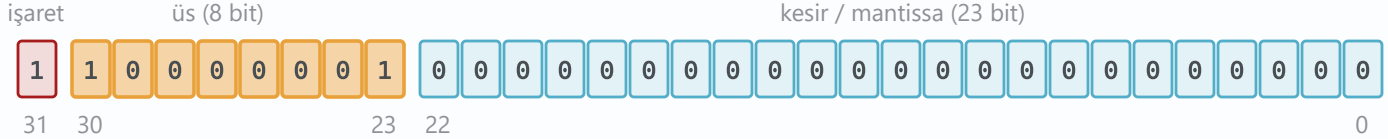


$6,5 = 110,1b (4 + 2 + 0,5)$. Normalize: **1,101b** $\times 2^2$.

Adım 3 / 6

Desen	Anlam
$e = 0, f = 0$	± 0
$e = 255, f = 0$	$\pm \infty$ (ör. 1/0)
$e = 255, f \neq 0$	NaN (sayı değil, ör. 0/0)
$e = 0, f \neq 0$	normalize edilmemiş (çok küçük) sayı

IEEE 754 Kodlama: -6,5

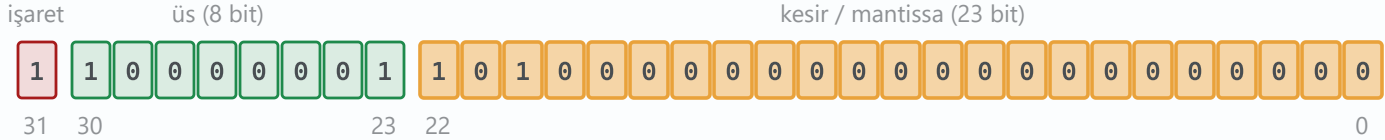


$$\text{Üs: } 2 + 127 = 129 = \mathbf{10000001b.}$$

Adım 4 / 6

Desen	Anlam
$e = 0, f = 0$	± 0
$e = 255, f = 0$	$\pm \infty$ (ör. 1/0)
$e = 255, f \neq 0$	NaN (sayı değil, ör. 0/0)
$e = 0, f \neq 0$	normalize edilmemiş (çok küçük) sayı

IEEE 754 Kodlama: –6,5

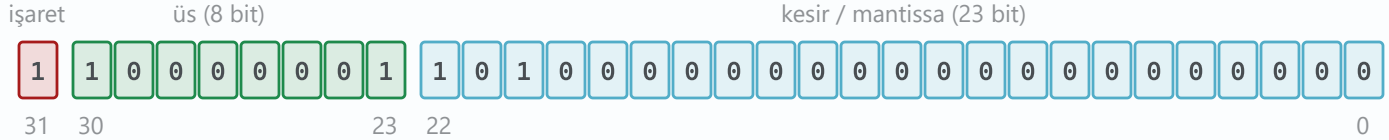


Mantissa: gizli 1 atılır, kalan **101** sağa 0'larla tamamlanır.

Adım 5 / 6

Desen	Anlam
$e = 0, f = 0$	± 0
$e = 255, f = 0$	$\pm \infty$ (ör. 1/0)
$e = 255, f \neq 0$	NaN (sayı değil, ör. 0/0)
$e = 0, f \neq 0$	normalize edilmemiş (çok küçük) sayı

IEEE 754 Kodlama: -6,5



Sonuç: 1100 0000 1101 0000 0...0b = **C0D00000h**.

Adım 6 / 6

Desen	Anlam
$e = 0, f = 0$	± 0
$e = 255, f = 0$	$\pm \infty$ (ör. 1/0)
$e = 255, f \neq 0$	NaN (sayı değil, ör. 0/0)
$e = 0, f \neq 0$	normalize edilmemiş (çok küçük) sayı

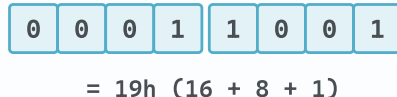
BCD: İkili Kodlu Onluk

- BCD (Binary Coded Decimal), **her onluk basamağı dört bit** ile temsil eden bir sistemdir.
- Her rakam 0'dan 9'a kadar 4 bitle ifade edilir; 1010 ... 1111 desenleri **geçersizdir**.
- Örnek: 25 → **0010 0101** (BCD) — ikili karşılığı ise **0001 1001** (19h).
- **Paketli** BCD: bir baytta iki basamak (25h).
Paketsiz: her baytta bir basamak (02h 05h).
- Kullanım: saat yongaları (RTC), 7 parçalı göstergeler, hesap makineleri, finansal hesaplar.

BCD 25



ikili 25



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

BCD Toplama ve Düzeltme

BCD 38 ile BCD 45'i toplayalım.

Adım 1 / 6

Düşük nibble'dan elde çıktığında (ör. $29 + 18 = 41h$, $AF = 1$) da 6 eklenir $\rightarrow 47$. Yüksek basamak 9'u aşarsa ona da 60h eklenir ve $CF = 1$ olur.

BCD Toplama ve Düzeltme

38 0011 1000 38h

38 → 0011 1000.

Adım 2 / 6

Düşük nibble'dan elde çıktığında (ör. $29 + 18 = 41h$, $AF = 1$) da 6 eklenir → 47. Yüksek basamak 9'u aşarsa ona da 60h eklenir ve $CF = 1$ olur.

BCD Toplama ve Düzeltme

38	0011 1000	38h
+ 45	0100 0101	45h

45 → 0100 0101.

Adım 3 / 6

Düşük nibble'dan elde çıktığında (ör. $29 + 18 = 41h$, $AF = 1$) da 6 eklenir → 47. Yüksek basamak 9'u aşarsa ona da 60h eklenir ve $CF = 1$ olur.

BCD Toplama ve Düzeltme

38	0011 1000	38h	
+ 45	0100 0101	45h	
= ?	0111 1101	7Dh	1101 = 13 > 9 → geçersiz basamak

İkili toplanır: 0111 1101 = 7Dh. Düşük basamak 1101 (13) BCD'de **geçersiz**.

Adım 4 / 6

Düşük nibble'dan elde çıktığında (ör. $29 + 18 = 41h$, $AF = 1$) da 6 eklenir → 47. Yüksek basamak 9'u aşarsa ona da 60h eklenir ve $CF = 1$ olur.

BCD Toplama ve Düzeltme

38	0011 1000	38h	
+ 45	0100 0101	45h	
= ?	0111 1101	7Dh	1101 = 13 > 9 → geçersiz basamak
+ 6	0000 0110	düzeltilme	

Geçersiz basamağa (ya da AF = 1 ise) **6** eklenir: $16 - 10 = 6$ kullanılmayan desen atlanır.

Adım 5 / 6

Düşük nibble'dan elde çıktığında (ör. $29 + 18 = 41h$, AF = 1) da 6 eklenir → 47. Yüksek basamak 9'u aşarsa ona da 60h eklenir ve CF = 1 olur.

BCD Toplama ve Düzeltme

38	0011 1000	38h	
+ 45	0100 0101	45h	
= ?	0111 1101	7Dh	1101 = 13 > 9 → geçersiz basamak
+ 6	0000 0110	düzeltilme	
= 83	1000 0011	83h	BCD 83 ✓

Sonuç 1000 0011 = **83** ✓. 8086'da bu düzeltmeyi `ADD AL, BL` ardından `DAA` yapar.

Adım 6 / 6

Düşük nibble'dan elde çıktığında (ör. 29 + 18 = 41h, AF = 1) da 6 eklenir → 47. Yüksek basamak 9'u aşarsa ona da 60h eklenir ve CF = 1 olur.

Kod İzleme: ADD ve DAA

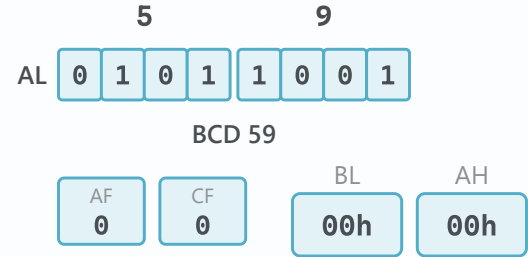
```

1  ORG 100h
▶ MOV AL, 59h      ; BCD 59
3  ADD AL, 27h      ; ikili toplama
4  DAA              ; BCD düzeltmesi
5  MOV BL, AL        ; 86'yı sakla
6  MOV AL, 75h      ; BCD 75
7  ADD AL, 48h      ; 75 + 48
8  DAA              ; CF = yüzler
9  MOV AH, 0
10 ADC AH, 0        ; AH ← CF
11 RET

```

işlem

—



AL = 59h: paketli BCD'de iki basamak, **5** ve **9**.

Adım 1 / 9

DEĞİŞKENLER

AL = 59h

AH = 00h

BL = 00h

AF = 0

CF = 0

DAA kuralı: düşük basamak > 9 ya da AF = 1 ise +06h; AL > 99h ya da CF = 1 ise +60h ve CF = 1. Kırmızı basamak BCD'de geçersizdir.

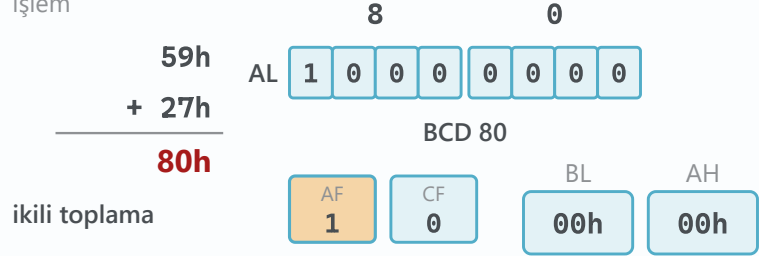
Kod İzleme: ADD ve DAA

```

1  ORG 100h
2  MOV AL, 59h      ; BCD 59
3  ► ADD AL, 27h     ; ikili toplama
4  DAA              ; BCD düzeltmesi
5  MOV BL, AL        ; 86'yı sakla
6  MOV AL, 75h       ; BCD 75
7  ADD AL, 48h        ; 75 + 48
8  DAA              ; CF = yüzler
9  MOV AH, 0
10 ADC AH, 0          ; AH ← CF
11 RET

```

işlem



ADD ikili toplar: 59h + 27h = **80h**. 9 + 7 = 16, düşük basamaktan taşı → **AF = 1**. BCD olarak 80 yanlış (59 + 27 = 86).

Adım 2 / 9

DEĞİŞKENLER

AL = 80h

AH = 00h

BL = 00h

AF = 1

CF = 0

DAA kuralı: düşük basamak > 9 ya da AF = 1 ise +06h; AL > 99h ya da CF = 1 ise +60h ve CF = 1. Kırmızı basamak BCD'de geçersizdir.

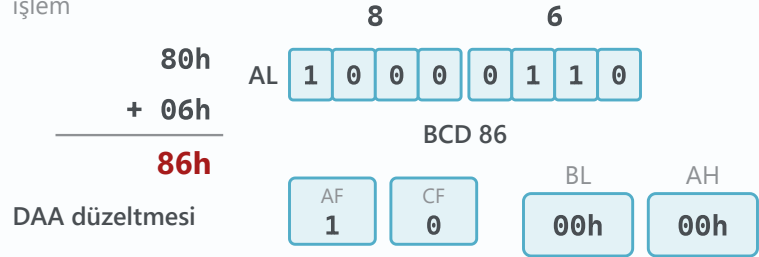
Kod İzleme: ADD ve DAA

```

1  ORG 100h
2  MOV AL, 59h      ; BCD 59
3  ADD AL, 27h      ; ikili toplama
4  DAA              ; BCD düzeltmesi
5  MOV BL, AL        ; 86'yı sakla
6  MOV AL, 75h      ; BCD 75
7  ADD AL, 48h      ; 75 + 48
8  DAA              ; CF = yüzler
9  MOV AH, 0
10 ADC AH, 0         ; AH ← CF
11 RET

```

işlem



DAA : AF = 1 olduğu için düşük basamağa 6 eklenir: 80h + 06h = **86h** ✓ CF = 0.

Adım 3 / 9

DEĞİŞKENLER

AL = 86h

AH = 00h

BL = 00h

AF = 1

CF = 0

DAA kuralı: düşük basamak > 9 ya da AF = 1 ise +06h; AL > 99h ya da CF = 1 ise +60h ve CF = 1. Kırmızı basamak BCD'de geçersizdir.

Kod İzleme: ADD ve DAA

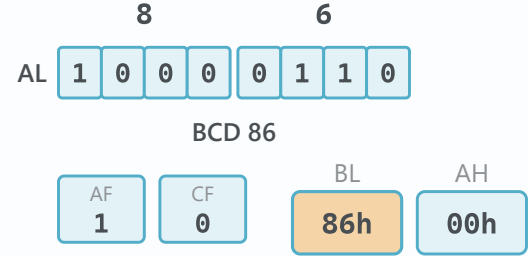
```

1  ORG 100h
2  MOV AL, 59h      ; BCD 59
3  ADD AL, 27h      ; ikili toplama
4  DAA              ; BCD düzeltmesi
5  ► MOV BL, AL      ; 86'yı sakla
6  MOV AL, 75h      ; BCD 75
7  ADD AL, 48h      ; 75 + 48
8  DAA              ; CF = yüzler
9  MOV AH, 0
10 ADC AH, 0         ; AH ← CF
11 RET

```

işlem

—



BL = 86h: ilk sonuç saklandı.

Adım 4 / 9

DEĞİŞKENLER

AL = 86h

AH = 00h

BL = 86h

AF = 1

CF = 0

DAA kuralı: düşük basamak > 9 ya da AF = 1 ise +06h; AL > 99h ya da CF = 1 ise +60h ve CF = 1. Kırmızı basamak BCD'de geçersizdir.

Kod İzleme: ADD ve DAA

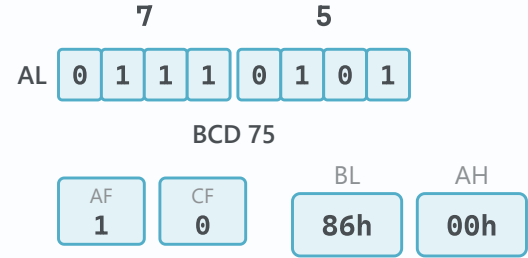
```

1  ORG 100h
2  MOV AL, 59h      ; BCD 59
3  ADD AL, 27h      ; ikili toplama
4  DAA              ; BCD düzeltmesi
5  MOV BL, AL       ; 86'yı sakla
6  MOV AL, 75h      ; BCD 75
7  ADD AL, 48h      ; 75 + 48
8  DAA              ; CF = yüzler
9  MOV AH, 0
10 ADC AH, 0        ; AH ← CF
11 RET

```

işlem

—



AL = 75h (BCD 75).

Adım 5 / 9

DEĞİŞKENLER

AL = 75h

AH = 00h

BL = 86h

AF = 1

CF = 0

DAA kuralı: düşük basamak > 9 ya da AF = 1 ise +06h; AL > 99h ya da CF = 1 ise +60h ve CF = 1. Kırmızı basamak BCD'de geçersizdir.

Kod İzleme: ADD ve DAA

```

1  ORG 100h
2  MOV AL, 59h      ; BCD 59
3  ADD AL, 27h      ; ikili toplama
4  DAA              ; BCD düzeltmesi
5  MOV BL, AL       ; 86'yı sakla
6  MOV AL, 75h      ; BCD 75
7  ► ADD AL, 48h    ; 75 + 48
8  DAA              ; CF = yüzler
9  MOV AH, 0
10 ADC AH, 0        ; AH ← CF
11 RET

```

işlem

$$\begin{array}{r}
 75h \\
 + 48h \\
 \hline
 \text{BDh}
 \end{array}$$

ikili toplama



75h + 48h = **BDh**: iki basamak da geçersiz (D = 13, B = 11). AF = 0, CF = 0.

Adım 6 / 9

DEĞİŞKENLER

AL = BDh

AH = 00h

BL = 86h

AF = 0

CF = 0

DAA kuralı: düşük basamak > 9 ya da AF = 1 ise +06h; AL > 99h ya da CF = 1 ise +60h ve CF = 1. Kırmızı basamak BCD'de geçersizdir.

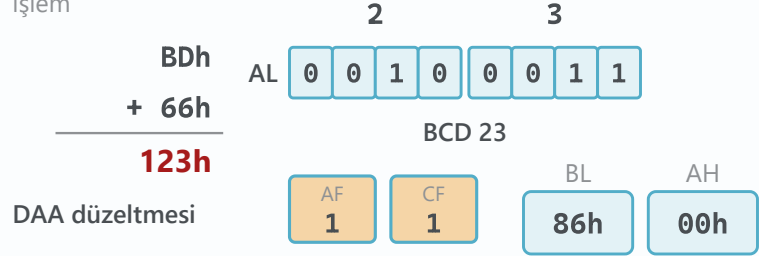
Kod İzleme: ADD ve DAA

```

1  ORG 100h
2  MOV AL, 59h      ; BCD 59
3  ADD AL, 27h      ; ikili toplama
4  DAA              ; BCD düzeltmesi
5  MOV BL, AL       ; 86'yı sakla
6  MOV AL, 75h      ; BCD 75
7  ADD AL, 48h      ; 75 + 48
8  DAA              ; CF = yüzler
9  MOV AH, 0
10 ADC AH, 0        ; AH ← CF
11 RET

```

işlem



DAA : düşük basamak > 9 → +06h, yüksek basamak > 9 → +60h. BDh + 66h = 123h: AL = **23h**, **CF = 1** (yüzler basamağı).

Adım 7 / 9

DEĞİŞKENLER

AL = 23h

AH = 00h

BL = 86h

AF = 1

CF = 1

DAA kuralı: düşük basamak > 9 ya da AF = 1 ise +06h; AL > 99h ya da CF = 1 ise +60h ve CF = 1. Kırmızı basamak BCD'de geçersizdir.

Kod İzleme: ADD ve DAA

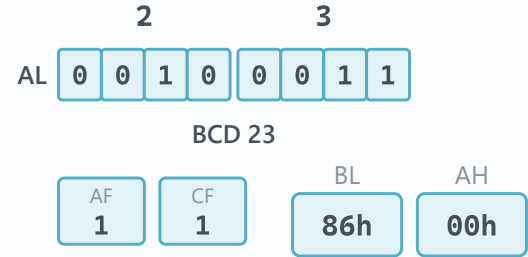
```

1  ORG 100h
2  MOV AL, 59h      ; BCD 59
3  ADD AL, 27h      ; ikili toplama
4  DAA              ; BCD düzeltmesi
5  MOV BL, AL        ; 86'yı sakla
6  MOV AL, 75h      ; BCD 75
7  ADD AL, 48h      ; 75 + 48
8  DAA              ; CF = yüzler
9  MOV AH, 0
10 ADC AH, 0        ; AH ← CF
11 RET

```

işlem

—



AH = 0. MOV bayrakları değiştirmez; CF = 1 korunur.

Adım 8 / 9

DEĞİŞKENLER

AL = 23h

AH = 00h

BL = 86h

AF = 1

CF = 1

DAA kuralı: düşük basamak > 9 ya da AF = 1 ise +06h; AL > 99h ya da CF = 1 ise +60h ve CF = 1. Kırmızı basamak BCD'de geçersizdir.

Kod İzleme: ADD ve DAA

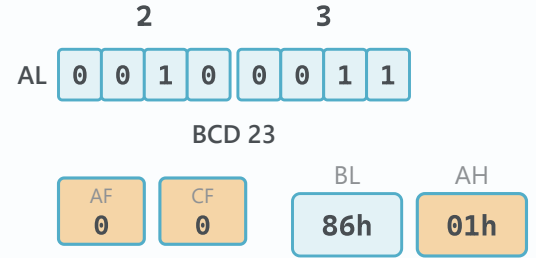
```

1  ORG 100h
2  MOV AL, 59h      ; BCD 59
3  ADD AL, 27h      ; ikili toplama
4  DAA              ; BCD düzeltmesi
5  MOV BL, AL        ; 86'yı sakla
6  MOV AL, 75h      ; BCD 75
7  ADD AL, 48h      ; 75 + 48
8  DAA              ; CF = yüzler
9  MOV AH, 0
10 ADC AH, 0         ; AH ← CF
11 RET

```

işlem

—



AX = 0123h → 123

ADC AH, 0 : AH = 0 + 0 + CF = 1 → AX = **0123h**: BCD 123 = 75 + 48 ✓

Adım 9 / 9

DEĞİŞKENLER

AL = 23h

AH = 01h

BL = 86h

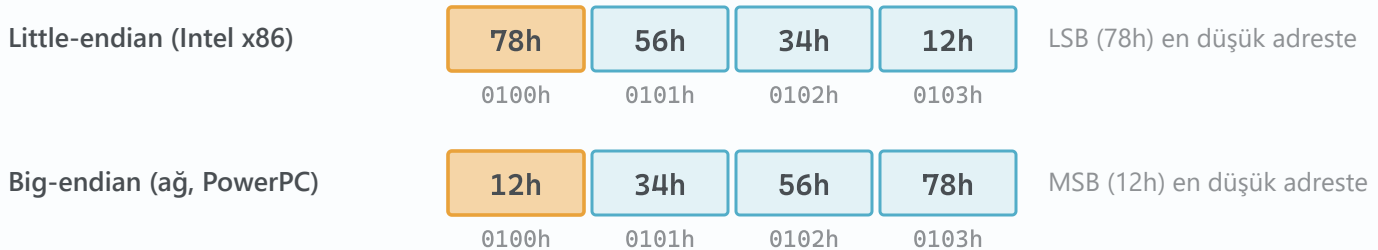
AF = 0

CF = 0

DAA kuralı: düşük basamak > 9 ya da AF = 1 ise +06h; AL > 99h ya da CF = 1 ise +60h ve CF = 1. Kırmızı basamak BCD'de geçersizdir.

Little-Endian ve Big-Endian

32 bitlik değer 12345678h, adres 0100h'den itibaren:



- **Little-endian:** en küçük anlamlı bayt (LSB) en düşük bellek adresindedir. Intel x86 (8086 dahil) little-endian'dır.
- **Big-endian:** en büyük anlamlı bayt (MSB) en düşük adrestedir. PowerPC (çoğunlukla), Motorola 68000 ve **ağ iletişimi** ("network byte order") big-endian kullanır.

01020304h için little-endian bellek düzeni 04 03 02 01 , big-endian düzeni 01 02 03 04 'tür.

8086'da Little-Endian

► **MOV** AX, 1234h

- 2 **MOV** [0200h], AX ; word yaz
- 3 **MOV** BL, [0200h] ; düşük adres
- 4 **MOV** BH, [0201h] ; yüksek adres
- 5 **MOV** CX, [0200h] ; word oku

0200h

0201h



AX = 1234h: AH = 12h, AL = 34h.

Adım 1 / 5

DEĞİŞKENLER

AX = 1234h

BX = 0000h

CX = 0000h

8086'da Little-Endian

1 **MOV** AX, 1234h

► **MOV** [0200h], AX ; word yaz

3 **MOV** BL, [0200h] ; düşük adres

4 **MOV** BH, [0201h] ; yüksek adres

5 **MOV** CX, [0200h] ; word oku

0200h

34h

0201h

12h

Sözcük 0200h'ye yazılır: **[0200h] = 34h** (düşük bayt),
[0201h] = 12h.

Adım 2 / 5

DEĞİŞKENLER

AX = 1234h

BX = 0000h

CX = 0000h

8086'da Little-Endian

```
1 MOV AX, 1234h
2 MOV [0200h], AX ; word yaz
3 ► MOV BL, [0200h] ; düşük adres
4 MOV BH, [0201h] ; yüksek adres
5 MOV CX, [0200h] ; word oku
```

0200h

34h

0201h

12h

Düşük adresteki bayt okunur: BL = **34h**.

Adım 3 / 5

DEĞİŞKENLER

AX = 1234h

BX = 0034h

CX = 0000h

8086'da Little-Endian

```
1 MOV AX, 1234h
2 MOV [0200h], AX ; word yaz
3 MOV BL, [0200h] ; düşük adres
▶ MOV BH, [0201h] ; yüksek adres
5 MOV CX, [0200h] ; word oku
```

0200h

34h

0201h

12h

Yüksek adresteki bayt: BH = **12h** → BX = 1234h.

Adım 4 / 5

DEĞİŞKENLER

AX = 1234h

BX = 1234h

CX = 0000h

8086'da Little-Endian

```
1 MOV AX, 1234h
2 MOV [0200h], AX ; word yaz
3 MOV BL, [0200h] ; düşük adres
4 MOV BH, [0201h] ; yüksek adres
▶ MOV CX, [0200h] ; word oku
```

0200h

34h

0201h

12h

Sözcük olarak okununca işlemci baytları yeniden birleştirir: CX = **1234h**. Sıralama yalnızca bayt bayt bakınca görünür.

Adım 5 / 5

DEĞİŞKENLER

AX = 1234h

BX = 1234h

CX = 1234h

Gray Kodlaması

Onluk	İkili	değişen bit	Gray
0	000	–	000
1	001	1	001
2	010	2	011
3	011	1	010
4	100	3	110
5	101	1	111
6	110	2	101
7	111	1	100

- Ardışık sayıların ikili temsiliinde **yalnızca bir bit** değişir.
- İkilide 3'ten 4'e (011 → 100) geçerken **üç bit** birden değişir; bitler aynı anda değişmezse ara değerler (ör. 111) okunabilir.
- Gray kodunda bu geçiş 010 → 110: tek bit, hatalı ara değer yok.
- Döner kodlayıcılar (encoder), analog-sayısal dönüştürücüler ve Karnaugh haritalarında kullanılır; **elektiriksel gürültüye** karşı dayanıklıdır.
- 3 bitlik dizi: 000, 001, 011, 010, 110, 111, 101, 100.

İkiliden Gray Koduna

Dönüşüm formülü: **$G = B \text{ XOR } (B \gg 1)$** .

Adım 1 / 4

- Kural: Gray'in en yüksek biti ikiliyle aynıdır; her sonraki bit, ikilideki kendisi ile solundaki bitin XOR'udur.
- 8086 ile tek komutla: `MOV BL, AL` / `SHR BL, 1` / `XOR AL, BL` .
- Geri dönüş: $B_{\text{en yüksek}} = G_{\text{en yüksek}}$, sonra $B_i = B_{i+1} \text{ XOR } G_i$.

İkiliden Gray Koduna

B

0

1

1

0

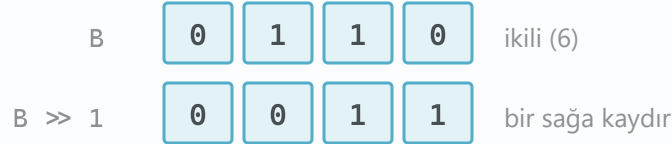
ikili (6)

$B = 0110$ (6).

Adım 2 / 4

- Kural: Gray'in en yüksek biti ikiliyle aynıdır; her sonraki bit, ikilideki kendisi ile solundaki bitin XOR'udur.
- 8086 ile tek komutla: `MOV BL, AL` / `SHR BL, 1` / `XOR AL, BL` .
- Geri dönüş: $B_{\text{en yüksek}} = G_{\text{en yüksek}}$, sonra $B_i = B_{i+1} \text{ XOR } G_i$.

İkiliden Gray Koduna



B bir bit sağa kaydırılır: 0011.

Adım 3 / 4

- Kural: Gray'in en yüksek biti ikiliyle aynıdır; her sonraki bit, ikilideki kendisi ile solundaki bitin XOR'udur.
- 8086 ile tek komutla: `MOV BL, AL` / `SHR BL, 1` / `XOR AL, BL` .
- Geri dönüş: $B_{\text{en yüksek}} = G_{\text{en yüksek}}$, sonra $B_i = B_{i+1} \text{ XOR } G_i$.

İkiliden Gray Koduna



Bit bit XOR: $0110 \oplus 0011 = \mathbf{0101} \rightarrow 6$ 'nın Gray kodu.

Adım 4 / 4

- Kural: Gray'in en yüksek biti ikiliyle aynıdır; her sonraki bit, ikilideki kendisi ile solundaki bitin XOR'udur.
- 8086 ile tek komutla: `MOV BL, AL` / `SHR BL, 1` / `XOR AL, BL` .
- Geri dönüş: $B_{\text{en yüksek}} = G_{\text{en yüksek}}$, sonra $B_i = B_{i+1} \text{ XOR } G_i$.

Kod İzleme: SHR ve XOR ile Gray Kodu

1 **ORG** 100h

► **MOV** AL, 1011b ; 11

3 **MOV** BL, AL

4 **SHR** BL, 1 ; B >> 1

5 **XOR** BL, AL ; G

6 **MOV** AL, 1100b ; 12

7 **MOV** BH, AL

8 **SHR** BH, 1

9 **XOR** BH, AL ; G

10 **RET**

B (AL)

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

 = 11

B >> 1 (BL)

--	--	--	--

G (BL)

--	--	--	--

AL = **1011b** = 11. Gray kodu formülü: $G = B \text{ xor } (B \gg 1)$.

Adım 1 / 8

DEĞİŞKENLER

AL = 0Bh

BL = 00h

BH = 00h

Kod İzleme: SHR ve XOR ile Gray Kodu

```
1 ORG 100h
2 MOV AL, 1011b ; 11
3 MOV BL, AL
4 SHR BL, 1 ; B >> 1
5 XOR BL, AL ; G
6 MOV AL, 1100b ; 12
7 MOV BH, AL
8 SHR BH, 1
9 XOR BH, AL ; G
10 RET
```

B (AL)

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

 = 11

B >> 1 (BL)

--	--	--	--

G (BL)

--	--	--	--

BL = AL: kaydırma kopya üzerinde yapılacak, B (AL) korunur.

Adım 2 / 8

DEĞİŞKENLER

AL = 0Bh

BL = 0Bh

BH = 00h

Kod İzleme: SHR ve XOR ile Gray Kodu

```
1 ORG 100h
2 MOV AL, 1011b ; 11
3 MOV BL, AL
4 ► SHR BL, 1 ; B >> 1
5 XOR BL, AL ; G
6 MOV AL, 1100b ; 12
7 MOV BH, AL
8 SHR BH, 1
9 XOR BH, AL ; G
10 RET
```

B (AL)

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

 = 11

B >> 1 (BL)

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

 = 5

G (BL)

--	--	--	--

SHR BL, 1 : **0101b**. En sağdaki bit düşer (CF = 1), sola 0 girer.

Adım 3 / 8

DEĞİŞKENLER

AL = 0Bh

BL = 05h

BH = 00h

Kod İzleme: SHR ve XOR ile Gray Kodu

```
1 ORG 100h
2 MOV AL, 1011b ; 11
3 MOV BL, AL
4 SHR BL, 1 ; B >> 1
5 XOR BL, AL ; G
6 MOV AL, 1100b ; 12
7 MOV BH, AL
8 SHR BH, 1
9 XOR BH, AL ; G
10 RET
```

B (AL)

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

 = 11

B >> 1 (BL)

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

 = 5

G (BL)

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

 = 14

xor

Gray(11) = 1110

XOR BL, AL : 1011 xor 0101 = **1110b**: 11'in Gray kodu. En yüksek bit ikiliyle aynı kaldı.

Adım 4 / 8

DEĞİŞKENLER

AL = 0Bh

BL = 0Eh

BH = 00h

Kod İzleme: SHR ve XOR ile Gray Kodu

```
1 ORG 100h
2 MOV AL, 1011b ; 11
3 MOV BL, AL
4 SHR BL, 1 ; B >> 1
5 XOR BL, AL ; G
6 MOV AL, 1100b ; 12
7 MOV BH, AL
8 SHR BH, 1
9 XOR BH, AL ; G
10 RET
```

B (AL)

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

 = 12

B >> 1 (BH)

--	--	--	--

G (BH)

--	--	--	--

Gray(11) = 1110

AL = 1100b = 12, bir sonraki sayı.

Adım 5 / 8

DEĞİŞKENLER

AL = 0Ch

BL = 0Eh

BH = 00h

Kod İzleme: SHR ve XOR ile Gray Kodu

```
1 ORG 100h
2 MOV AL, 1011b ; 11
3 MOV BL, AL
4 SHR BL, 1 ; B >> 1
5 XOR BL, AL ; G
6 MOV AL, 1100b ; 12
7 MOV BH, AL
8 SHR BH, 1
9 XOR BH, AL ; G
10 RET
```

B (AL)

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

 = 12

B >> 1 (BH)

--	--	--	--

G (BH)

--	--	--	--

Gray(11) = 1110

BH = AL.

Adım 6 / 8

DEĞİŞKENLER

AL = 0Ch

BL = 0Eh

BH = 0Ch

Kod İzleme: SHR ve XOR ile Gray Kodu

```
1 ORG 100h
2 MOV AL, 1011b ; 11
3 MOV BL, AL
4 SHR BL, 1 ; B >> 1
5 XOR BL, AL ; G
6 MOV AL, 1100b ; 12
7 MOV BH, AL
8 SHR BH, 1
9 XOR BH, AL ; G
10 RET
```

B (AL)

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

 = 12

B >> 1 (BH)

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

 = 6

G (BH)

--	--	--	--

Gray(11) = 1110

SHR BH, 1 : **0110b**.

Adım 7 / 8

DEĞİŞKENLER

AL = 0Ch

BL = 0Eh

BH = 06h

Kod İzleme: SHR ve XOR ile Gray Kodu

```
1 ORG 100h
2 MOV AL, 1011b ; 11
3 MOV BL, AL
4 SHR BL, 1 ; B >> 1
5 XOR BL, AL ; G
6 MOV AL, 1100b ; 12
7 MOV BH, AL
8 SHR BH, 1
9 XOR BH, AL ; G
10 RET
```

B (AL)

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

 = 12

B >> 1 (BH)

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

 = 6

G (BH)

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

 = 10

xor

Gray(11) = 1110 Gray(12) = 1010 **yalnız bit 2 farklı**

XOR → **1010b**. 11 → 12 geçişinde ikilide 3 bit değişir (1011 → 1100); Gray kodunda yalnız **1 bit** (1110 → 1010).

Adım 8 / 8

DEĞİŞKENLER

AL = 0Ch

BL = 0Eh

BH = 0Ah

ÖZET

Özet

Konumsal sistemler

Değer = $\sum \text{rakam} \times \text{taban}^{\text{konum}}$. İkili (b), onaltılık (h), onluk; 1 onaltılık rakam = 4 bit.

İşaretli sayılar

ikiye tümleyen: tersle + 1 (NEG). MSB işaret bitidir; 8 bit -128...127, 16 bit -32768...32767.

Dönüşüm

Onluktan: tabana böl, kalanları aşağıdan yukarı oku. İkili ↔ onaltılık: 4'lü gruplar.

Kodlamalar

IEEE 754: işaret/üs(127)/mantissa. BCD: basamak başına 4 bit, +6 düzeltme. x86 little-endian. Gray: tek bit değişir.

ÖZET

Alıştırmalar

1. MCMXCIV ve MMXXVI sayılarını onluğa çevirin.
2. 11001010b sayısını onluk ve onaltılığa çevirin.
3. 2024 sayısını onaltılık ve ikili yazın.
4. -100 'ü 8 bitlik ikiye tümleyen olarak yazın. 16 bitte nasıl görünür?
5. $0F0h + 20h$ işleminin 8 bitlik sonucu, CF ve işaretli yorumu nedir?
6. BCD $47 + 36$ toplamını düzeltmeyle yapın.
7. 0 10000010 011000000000000000000000 IEEE 754 sayısının değeri nedir?
8. 5'in 3 bitlik Gray kodu nedir? 0ABCDh, 0200h adresine sözcük olarak yazılırsa [0200h] nedir?

► Cevaplar

ÖZET

Alıştırmalar (2)

9. 0,375 ve 0,2 sayılarını ikiliye çevirin (0,2 için 8 kesir biti).
10. $-3,625$ sayısını 8 bitlik Q5.3 biçiminde (ikili ve onaltılık) yazın.
11. $7C5Eh + 3BA9h$ toplamını; $3BA9h - 7C5Eh$ farkını 16'ya tümleyenle bulun.
12. 8 bitte $70h + 30h$, $90h - 20h$ ve $90h - F0h$ işlemlerinin sonucu ve OF değeri nedir?

► Cevaplar

Kaynaklar

- M. Morris Mano, Michael D. Ciletti, *Digital Design*, 5. baskı, Pearson, 2013.
- Barry B. Brey, *The Intel Microprocessors*, 8. baskı, Pearson, 2009 — Bölüm 1: sayı sistemleri.
- IEEE Std 754-2019, *IEEE Standard for Floating-Point Arithmetic*.
- emu8086 belgeleri: *Numbering systems tutorial*.
- Mehmet Bodur, *Mikroişlemcilere Giriş: Assembler ile Yazılım ve Arayüz*, EMO, 2014.
- Şadi Çağatay Öztürk, Oğuz Ergin, *Intel 8086 ile Mikroişlemci Programlamaya Giriş*, 2009.

Son

Bölüm 3: Sayı Sistemleri

Sonraki bölüm: Bölüm 3: Assembly · [Terim sözlüğü](#)