SUNUM İÇERİĞİ

- **Bit ve Byte kavramları.**
- **İkilik (Binary), Onluk (Decimal) ve Onaltılık (Hexadecimal) sistemler.**
- **Negatif sayıların bilgisayarda tutulması (Two's Complement)**

Onur İpek
24360859090

BÖLÜM 1:BİT VE BYTE KAVRAMLARI

- 1.1 Bitler ve depolamaları:

Günümüzde bilgisayarlarda bilgiler 0 ve 1'lerden oluşur.Bu rakamlara bit denir.Bitler bir çok şeyi temsil edebilir.Örneğin bazen bir rakamı temsil ederken bazen bir harfi bazen ise bir Noktalama işaretini temsil eder.

Mantıksal işlemler (Boolean Operations):

Tekil Bitler bilgisayarlarda 0 biti(yanlış) değerini 1 biti ise(true) Değerini temsil eder.Doğru/Yanlış değerlerini işleyen işlemlere Mantıksal (Boolean) işlemleri denir. Temel üç mantıksal işlem; AND(VE),OR(VEYA) ve XOR'dur(Özel Veya).Bu işlemler aynı Matematikteki aritmetik işlemler gibi 2 değerin bir işleme sokulup Yeni üçüncü bir değer oluşturmasıdır. Ancak aritmetik işlemlerden Farkı sayısal değerler yerine doğru yanlış değerlerini birleştirir

AND (VE) işlemi, iki ifadenin “ve” bağlacı ile birleştirilmesiyle Oluşturulan bir işlemdir.Bu işlemler genel olarak şu şekildedir:

F AND Q

Burada F bir ifadeyi Q da bir ifadeyi temsil eder. Örneğin:

Onur 20 yaşındadır ve 1.75 boyundadır.

And işlemine giren girdiler bu bileşik ifadenin doğrulunu ifade Eder. “F AND Q” formundaki bir ifade yalnızca her iki bileşeni Doğru (1) olduğunda ve 1 AND 1’in 1 olması gerektiğinden Diğer tüm işlemlerin sonucu ‘0’ çıkar.

Benzer şekilde OR(Veya) işlemide şu formdaki bileşik ifadelere Dayanır:

P OR Q

Burada P bir ifadeyi Q ise başka bir ifadeyi temsil eder. Bu Tür ifadelerde en az biri doğru olduğunda doğrudur. İkiside Yanlışssa yanlıştır.

AND İŞLEMİ	0 AND 0 = “0”	0 AND 1 = “0”	1 AND 0 = “0”	1 AND 1 = “1”
OR İŞLEMİ	0 OR 0 = “0”	0 OR 1= “1”	1 OR 0 = “1”	1 OR 1 = “1”
XOR İŞLEMİ	0 XOR 0 = “0”	0 XOR 1 = “1”	1 XOR 0 = “1”	1 XOR 1 = “0”

İngiliz dilinde XOR işleminin anlamını tam olarak karşılayan tek bir bağlaç Yoktur. Girdilerden biri 1(Doğru) diğeri 0(Yanlış) olduğunda 1(Doğru) çıktısını Üretir. Kısacası, XOR işlemi girdileri birbirinden farklı olduğunda 1(Doğru) Değerini üretir, ikiside aynı ise 0(Yanlış) değerini üretir.

NOT (Değil) işlemi ise bir başka mantıksal işlemdir. Bir girdisi Olması ile diğer işlemlerden ayrılır. Çıktısı o girdinin zıttıdır. NOT (Değil) işleminde girdi doğru(1) ise çıktı yanlış’ dır (0). Bunun tam tersi içinde geçerlidir. Girdi yanlış(0) ise çıktı Doğru’dur(1).

KAPILAR VE FLİP-FLOPLAR

Mantıksal bir işlemin girdi değerleri verildiğinde o değerlerin Çıktısını üreten bi aygıta kapı(Gate) denir. Günümüz bilgisayarlarında kapılar genellikle, 0 ve 1 rakamlarının voltaj seviyeleriyle temsil edildiği küçük elektronik devreler olarak uygulanır. kapıları Şekil 1.2'de gösterildiği gibi sembolik formlarıyla temsil etmek yeterlidir.

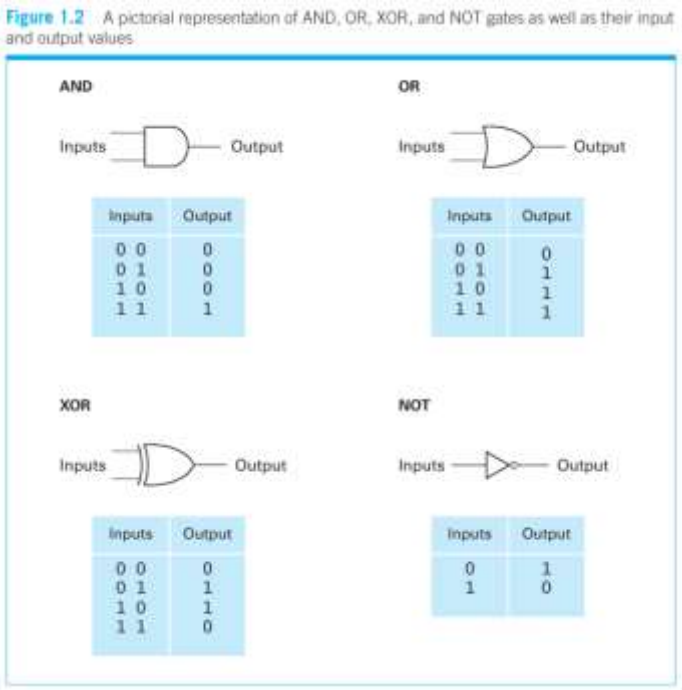
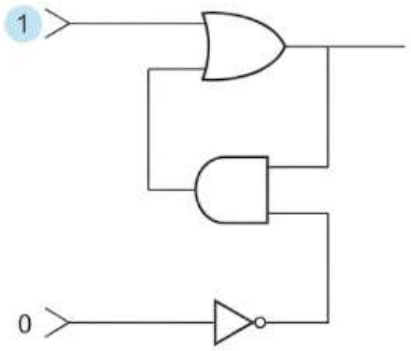


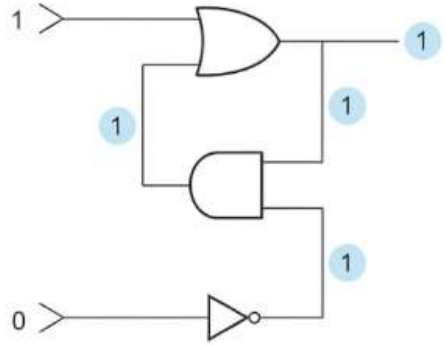
Figure 1.2 A pictorial representation of AND, OR, XOR, and NOT gates as well as their input and output values

KAPILAR VE FLİP-FLOPLAR ÖRNEĞİ;

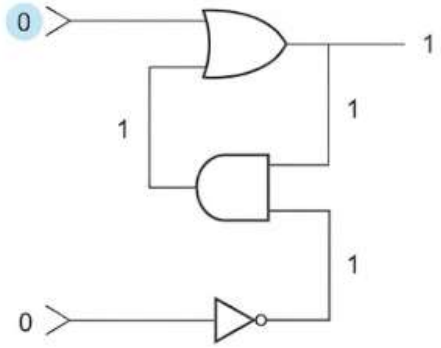
a. Öncelikle, üst girişe 1 yerleştirilir.



b. Bu, VEYA kapısının çıkışının 1 olmasına ve sırasıyla VE kapısının çıkışının 1 olmasına neden olur.



c. Son olarak, VE kapısından gelen 1, üst giriş 0'a döndükten sonra VEYA kapısının değişmesini engeller.



- **Onaltılık (Hexadecimal) Gösterim**

Bir bilgisayarın dahili faaliyetlerini incelerken, bazen oldukça uzun olabilen ve bit dizisi (string of bits) olarak adlandıracağımız bit desenleriyle uğraşmamız gerekir. Uzun bir bit dizisine genellikle **akış (stream)** denir. Ne yazık ki, akışları insan zihninin kavraması zordur. Sadece "101101010011" desenini kâğıda dökmek bile zahmetli ve hataya müsaittir. Bu nedenle, bu tür bit desenlerinin temsilini basitleştirmek için genellikle **onaltılık gösterim (hexadecimal notation)** adı verilen bir kısayol notasyonu kullanırız.

- Bir makine içindeki bit desenleri genellikle dördün katları olan uzunluklara sahip olma eğilimindedir. Özellikle onaltılık gösterim, dört bitlik bir deseni temsil etmek için tek bir sembol kullanır. Örneğin, on iki bitlik bir dizi, üç onaltılık sembolle temsil edilebilir.
- Şekil 1.6, onaltılık kodlama sistemini sunmaktadır. Sol sütun, dört uzunluğundaki tüm olası bit desenlerini gösterir; sağ sütun ise bu bit desenini temsil etmek için onaltılık gösterimde kullanılan sembolü gösterir. Bu sistemi kullanarak, 10110101 bit deseni B5 olarak temsil edilir. Bu, bit deseninin dört uzunluğundaki alt dizilere bölünmesi ve ardından her alt dizinin onaltılık karşılığıyla temsil edilmesiyle elde edilir—1011, B ile ve 0101, 5 ile temsil edilir. Bu şekilde, 1010010011001000 şeklindeki 16 bitlik desen, çok daha anlaşılır olan A4C8 formuna indirgenebilir.

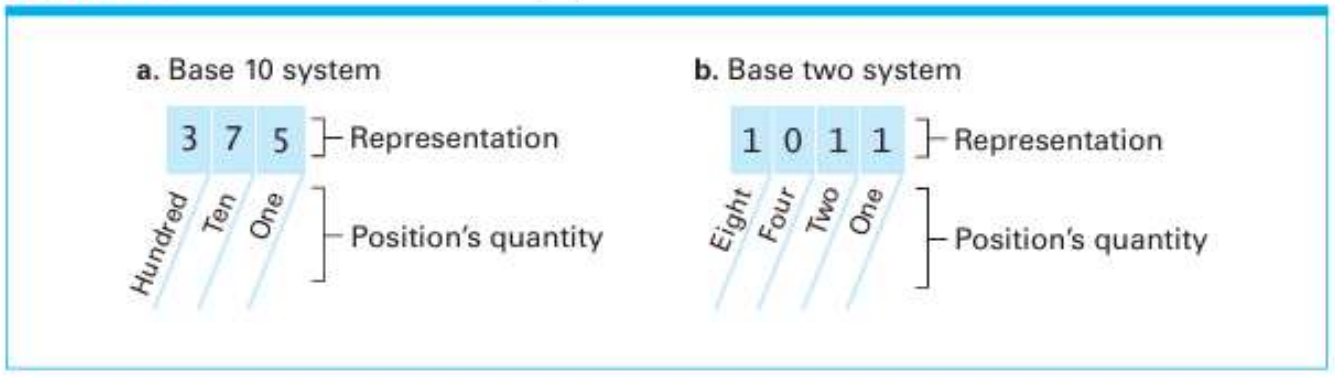
Figure 1.6 The hexadecimal encoding system

Bit pattern	Hexadecimal representation
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

İkili Gösterim (Binary Notation)

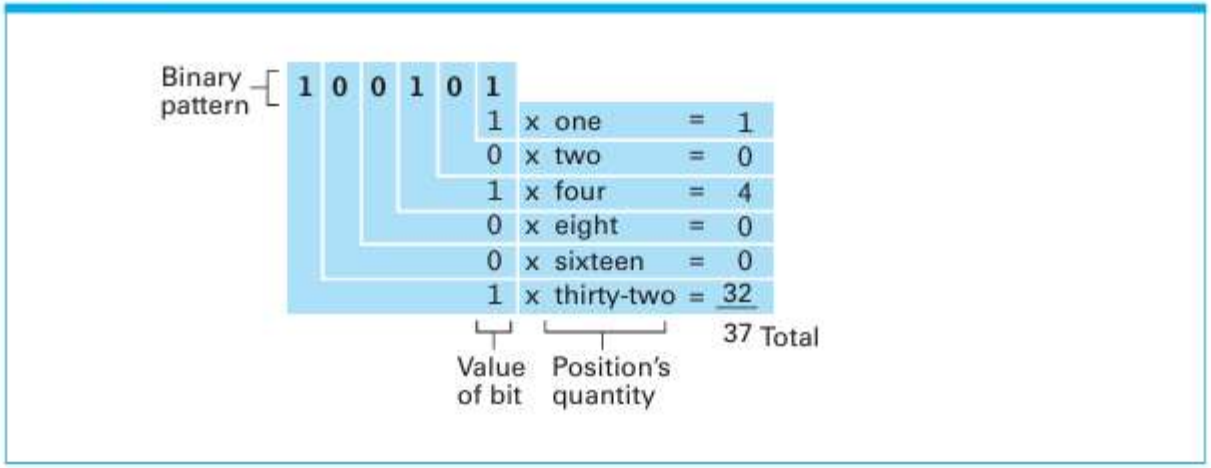
- İkilik gösterimde her bir rakamın konumu bir miktarla ilişkilidir; ancak her konuma karşılık gelen miktar, sağındaki konumun miktarının iki katıdır. Daha kesin bir ifadeyle, bir ikili gösterimdeki en sağdaki rakam birler (2^0) miktarıyla, soldaki bir sonraki konum ikiler (2^1) miktarıyla, bir sonraki dörder (2^2) miktarıyla, sonraki sekizer (2^3) miktarıyla ilişkilidir ve bu böyle devam eder. Örneğin, 1011 ikili gösteriminde, en sağdaki 1 rakamı birler miktarıyla, yanındaki 1 rakamı ikiler miktarıyla, 0 rakamı dörder miktarıyla ve en soldaki 1 rakamı ise sekizer miktarıyla ilişkilidir (Şekil 1.13b).

Figure 1.13 The base 10 and binary systems



Bir ikili gösterimin temsil ettiği değeri bulmak için taban 10'daki ile aynı yöntemi izleriz: her rakamın değerini konumuyla ilişkili miktarla çarpar ve sonuçları toplarız. Örneğin, Şekil 1.14'te gösterildiği gibi 100101'i temsil ettiği değer 37'dir. İkili gösterimin sadece 0 ve 1 rakamlarını kullandığına dikkat edin; bu çarpma-ve-toplama işlemi, sadece 1'lerin kapladığı konumlara karşılık gelen miktarların toplanmasına indirgenir. Dolayısıyla, 1011 ikili deseni on bir değerini temsil eder; çünkü 1'ler birler, ikiler ve sekizler konumlarında bulunur.

Figure 1.14 Decoding the binary representation 100101



- Bölüm 1.4'te, küçük tam sayıları kodlamamıza olanak tanıyan ikili notasyonda (binary notation) saymayı öğrendik. Büyük değerlerin ikili gösterimlerini bulmak için Şekil 1.15'teki algoritma ile açıklanan yaklaşımı tercih edebilirsiniz. Bu algoritmayı on üç değerine uygulayalım (Şekil 1.16). Önce on üçü ikiye böleriz; altı bölümünü ve bir kalanını elde ederiz. Bölüm sıfır olmadığı için Adım 2 bize bölümü (altı) ikiye bölmemizi söyler; böylece üç olan yeni bir bölüm ve sıfır kalanını elde ederiz. En son bölüm hâlâ sıfır değildir, bu yüzden onu ikiye böleriz; bir bölümünü ve bir kalanını elde ederiz. Bir kez daha, en son bölümü (bir) ikiye böleriz, bu sefer sıfır bölümünü ve bir kalanını elde ederiz. Artık sıfır bölümünü elde ettiğimiz için Adım 3'e geçeriz; burada orijinal değerin (on üç) ikili gösteriminin, kalanlar listesinden elde edilen 1101 olduğunu öğreniriz.

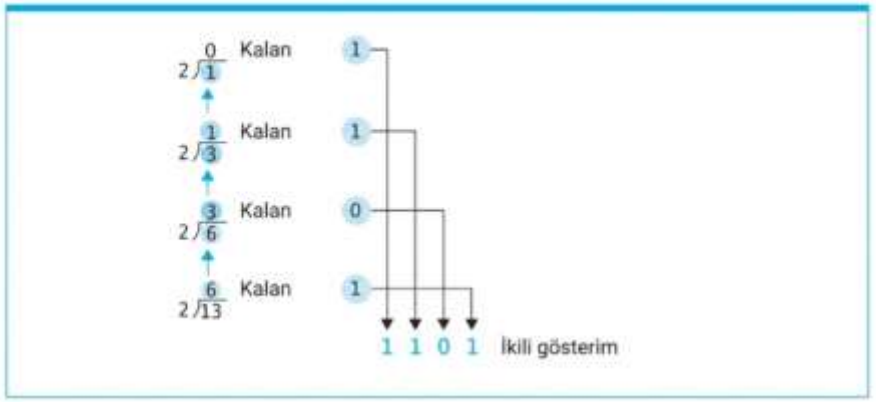
Şekil 1.15 Pozitif bir tamsayının ikili gösterimini bulmak için bir algoritma

Adım 1. Değeri ikiye bölün ve kalanı kaydedin.

Adım 2. Elde edilen bölüm sıfır olmadığı sürece, en yeni bölümü ikiye bölmeye ve kalanı kaydetmeye devam edin.

Adım 3. Artık sıfır bölümü elde edildiğine göre, orijinal değerin ikili gösterimi, kalanların kaydedildikleri sırayla sağdan sola listelenmesinden oluşur.

Şekil 1.16 On üçün ikili gösterimini elde etmek için Şekil 1.15'teki algoritmanın uygulanması



- İkili notasyonda (binary) temsil edilen iki tam sayıyı toplamak için, tüm toplamların ilkokulda öğrendiğiniz geleneksel 10 tabanı kuralları yerine **Şekil 1.17**'de gösterilen toplama kuralları kullanılarak hesaplanması dışında aynı prosedürü izleriz. Örneğin, şu problemi çözmek için:

```
  111010
+  11011
```

Figure 1.17 The binary addition facts

```
  0     1     0     1
+ 0   + 0   + 1   + 1
  0     1     1    10
```

- En sağdaki 0 ve 1'i toplayarak başlarız; 1 elde ederiz ve bunu sütunun altına yazarız. Şimdi bir sonraki sütundaki 1 ve 1'i toplarız ve 10 (ikili sistemde iki) elde ederiz. Bu 10'un 0'ını sütunun altına yazarız ve 1'i bir sonraki sütunun tepesine taşırız. Bu noktada çözümümüz şöyle görünür:

```
    1
  111010
+  11011
    01
```

- Bir sonraki sütundaki 1, 0 ve 0'ı toplarız, 1 elde ederiz ve bu 1'i sütunun altına yazarız. Bir sonraki sütundaki 1 ve 1'in toplamı 10 eder; 0'ı sütunun altına yazarız ve 1'i bir sonraki sütuna taşırız. Artık çözümümüz şöyle görünür:

```
   1
  111010
+  11011
   0101
```

Bir sonraki sütundaki 1, 1 ve 1'in toplamı 11'dir (üç değerinin ikili gösterimi); düşük basamaklı 1'i sütunun altına yazarız ve diğer 1'i bir sonraki sütunun tepesine elde olarak taşırız. Bu 1'i o sütunda zaten bulunan 1 ile toplayarak 10 elde ederiz. Yine, düşük düşük basamaklı 0'ı kaydederiz ve 1'i bir sonraki sütuna elde olarak taşırız. Şimdi elimizde şunlar var:

```
  1
   111010
+  11011
--------
   010101
```

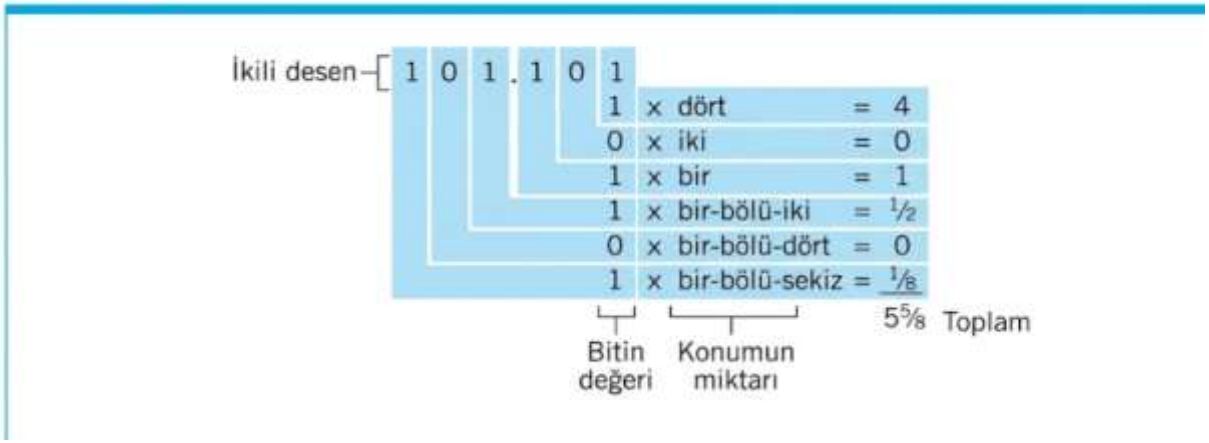
Bir sonraki sütundaki tek giriş, önceki sütundan taşıdığımız 1'dir, bu yüzden onu cevaba kaydederiz. Nihai çözümümüz şöyledir:

```
  111010
+ 11011
--------
 1010101
```

İkili Sistemde Kesirler

- İkili (binary) notasyonu kesirli değerleri de kapsayacak şekilde genişletmek için, ondalık notasyondaki ondalık noktanın (virgülün) oynadığı rolün aynısını üstlenen bir **taban noktası (radix point)** kullanırız. Yani, noktanın solundaki basamaklar değerin tam sayı kısmını (bütün kısmını) temsil eder ve daha önce tartışılan ikili sistemdeki gibi yorumlanır. Sağındaki basamaklar ise değerin kesirli kısmını temsil eder ve konumlarına kesirli nicelikler atanması dışında diğer bitlere benzer bir şekilde yorumlanır.
- Şöyle ki; taban noktasının sağındaki ilk konuma 1/2 (2^{-1}) değeri, bir sonraki konuma
- 1/4(2^{-2}) değeri, bir sonrakine 1/8 (2^{-4}) değeri atanır ve bu böyle devam eder. Bunun, daha önce belirtilen kuralın sadece bir devamı olduğuna dikkat edin: Her konuma, sağındakinin iki katı büyüklüğünde bir nicelik (değer) atanır.
- Bit konumlarına atanan bu değerlerle, bir taban noktası içeren ikili gösterimin kodunu çözmek (decode), taban noktası olmayanla aynı prosedürü gerektirir. Daha kesin bir ifadeyle, her bir bit değerini, o bitin gösterimdeki konumuna atanan nicelikle çarparız. Örneklemek gerekirse, Şekil 1.18'de gösterildiği gibi, **101.101** ikili gösterimi **5 5/8** (beş tam sekizde beş) olarak çözümlenir.

Şekil 1.18 İkili gösterim 101.101'in kodunu çözme



Tam Sayıları Saklama

- Matematikçiler uzun zamandır sayısal notasyon sistemleriyle ilgilenmektedir ve fikirlerinin çoğu dijital devre tasarımıyla oldukça uyumlu çıkmıştır. Bu bölümde, bilgi işlem ekipmanlarında tam sayı değerlerini temsil etmek için kullanılan iki notasyon sistemini, yani "ikinin tümleyeni" (two's complement) ve "fazlalık" (excess) notasyonunu ele alacağız. Bu sistemler ikili sisteme (binary system) dayanır ancak onları bilgisayar tasarımıyla daha uyumlu hale getiren ek özelliklere sahiptir. Ancak bu avantajlarla birlikte dezavantajlar da gelir. Amacımız bu özellikleri ve bilgisayar kullanımını nasıl etkilediklerini anlamaktır.

- **İkinin Tümleyeni Notasyonu (Two's Complement Notation)**
- Günümüz bilgisayarlarında tam sayıları temsil etmek için en popüler sistem **ikinin tümleyeni** notasyonudur. Bu sistem, sistemdeki değerlerin her birini temsil etmek için sabit sayıda bit kullanır. Günümüz ekipmanlarında, her değerin 32 bitlik bir desenle temsil edildiği bir ikinin tümleyeni sistemi kullanmak yaygındır. Böylesine büyük bir sistem, geniş bir sayı aralığının temsil edilmesine izin verir ancak gösterim amaçları için elverişsizdir. Bu nedenle, ikinin tümleyeni sistemlerinin özelliklerini incelemek için daha küçük sistemlere odaklanacağız.

- Şekil 1.19, iki tam ikinin tümleyeni sistemini göstermektedir; biri üç uzunluğunda bit desenlerine, diğeri dört uzunluğunda bit desenlerine dayanır. Böyle bir sistem, uygun uzunlukta bir 0 dizisiyle başlayıp, tek bir 0 ve ardından gelen 1'lerden oluşan desene ulaşılana kadar ikili sistemde sayılarak oluşturulur. Bu desenler 0, 1, 2, 3, ... değerlerini temsil eder. Negatif değerleri temsil eden desenler ise, uygun uzunlukta bir 1 dizisiyle başlayıp, tek bir 1 ve ardından gelen 0'lardan oluşan desene ulaşılana kadar ikili sistemde geriye doğru sayılarak elde edilir. Bu desenler -1, -2, -3, ... değerlerini temsil eder. (Eğer ikili sistemde geriye doğru saymak zorsa, tablonun en altından tek bir 1 ve ardından gelen 0'lardan oluşan desenle başlayın ve tamamı 1'lerden oluşan desene kadar yukarı doğru sayın.)

Şekil 1.19 İkinin tümleyeni gösterim sistemleri

a. Üç uzunluklu desenler kullanarak

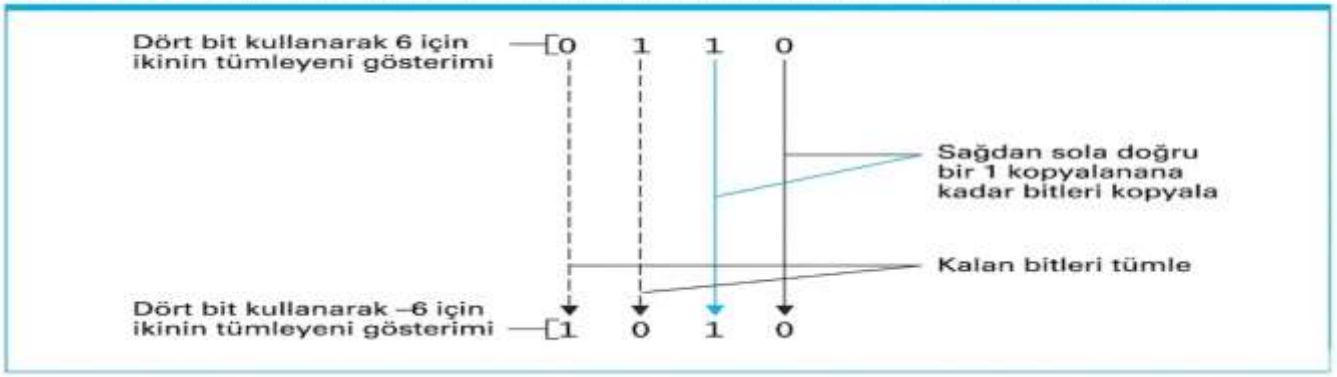
Bit deseni	Temsil edilen değer
011	3
010	2
001	1
000	0
111	-1
110	-2
101	-3
100	-4

b. Dört uzunluklu desenler kullanarak

Bit deseni	Temsil edilen değer
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	-1
1110	-2
1101	-3
1100	-4
1011	-5
1010	-6
1001	-7
1000	-8

- Bir ikinin tümleyeni sisteminde, bir bit deseninin en solundaki bitin, temsil edilen değerin işaretini belirttiğine dikkat edin. Bu nedenle, en soldaki bit genellikle **işaret biti (sign bit)** olarak adlandırılır. Bir ikinin tümleyeni sisteminde, negatif değerler işaret biti 1 olan desenlerle temsil edilir; negatif olmayan değerler ise işaret biti 0 olan desenlerle temsil edilir.
- Bir ikinin tümleyeni sisteminde, aynı büyüklükteki pozitif ve negatif değerleri temsil eden desenler arasında uygun bir ilişki vardır. Sağdan sola okunduğunda, ilk 1 dahil olmak üzere o noktaya kadar (ilk 1'i görene kadar) aynıdırlar. O noktadan sonra, desenler birbirinin tümleyenidir. (Bir desenin **tümleyeni (complement)**, tüm 0'ların 1'e ve tüm 1'lerin 0'a değiştirilmesiyle elde edilen desendir; 0110 ve 1001 birbirinin tümleyenidir. Örneğin, Şekil 1.19'daki 4 bitlik sistemde 2 ve -2'yi temsil eden desenlerin her ikisi de **10** ile biter, ancak 2'yi temsil eden desen **00** ile başlarken, -2'yi temsil eden desen **11** ile başlar. Bu gözlem, aynı büyüklükteki pozitif ve negatif değerleri temsil eden bit desenleri arasında ileri geri dönüşüm yapmak için bir algoritmaya yol açar. Tek yapmamız gereken, orijinal deseni sağdan sola doğru, bir 1 kopyalanana kadar (o 1 dahil) aynen kopyalamaktır; ardından geri kalan bitleri son bit desenine aktarırken tümleyenlerini (terslerini) alırız

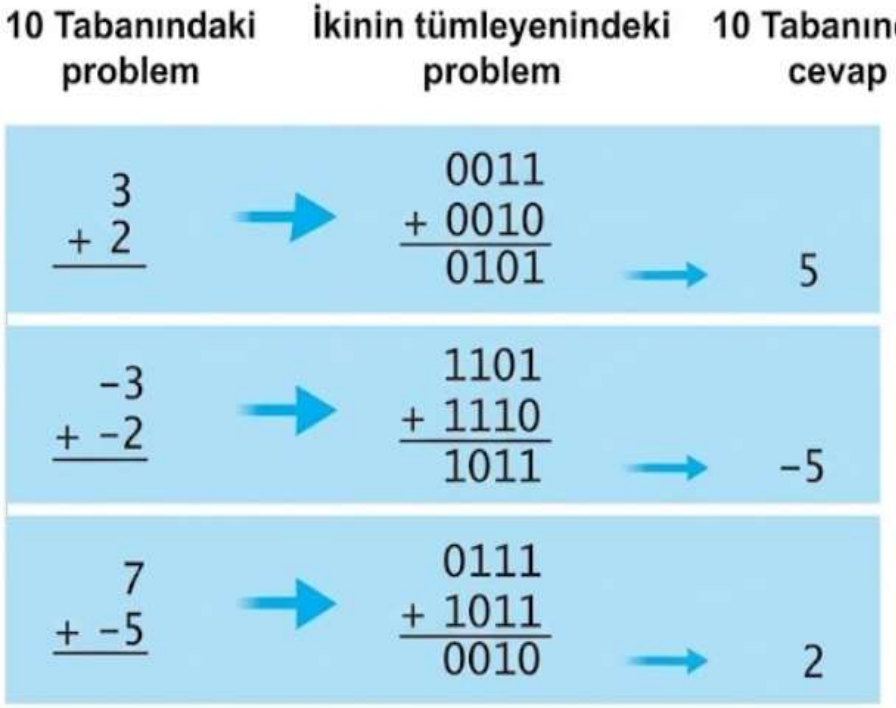
Şekil 1.20 4 bit kullanarak ikinin tümleyeni gösteriminde —6 değerini kodlama



- İkinin tümleyeni (two's complement) sistemlerinin bu temel özelliklerini anlamak, aynı zamanda ikinin tümleyeni gösterimlerinin kodunu çözmek (gerçek değerini bulmak) için de bir algoritmaya yol açar. Eğer desen kodu çözülecek desenin işaret biti 0 ise, değeri sanki desen bir ikili (binary) gösterimmiş gibi okumamız yeterlidir. Örneğin, **0110**, 6 değerini temsil eder; çünkü **110**, 6'nın ikilisidir.
- Eğer kodu çözülecek desenin işaret biti 1 ise, temsil edilen değerin negatif olduğunu biliriz ve geriye kalan tek şey değerin büyüklüğünü bulmaktır. Bunu, Şekil 1.20'deki “kopyala ve tümleyenini al” (copy and complement) prosedürünü uygulayarak ve ardından elde edilen deseni, sanki basit bir ikili gösterimmiş gibi çözerek yaparız.
- Örneğin, **1010** deseninin kodunu çözmek için; öncelikle işaret biti 1 olduğu için temsil edilen değerin negatif olduğunu teşhis ederiz. Bu nedenle, **0110** desenini elde etmek için “kopyala ve tümleyenini al” prosedürünü uygularız; bunun 6 için ikili gösterim olduğunu fark ederiz ve orijinal desenin **-6**'yı temsil ettiği sonucuna varırız.

- **İkinin Tümleyeni Notasyonunda Toplama**
- İkinin tümleyeni notasyonunda temsil edilen değerleri toplamak için, tüm bit desenlerinin (cevap dahil) aynı uzunlukta olması haricinde, ikili (binary) toplama için kullandığımız algoritmanın aynısını uygularız. Bu, bir ikinin tümleyeni sisteminde toplama yaparken, son elde (carry) işlemiyle cevabın solunda oluşturulan herhangi bir fazladan bitin kesilip atılması (kırpılması) gerektiği anlamına gelir. Böylece, 0101 ve 0010'u "toplamak" 0111'i üretir; 0111 ve 1011'i "toplamak" ise 10010 (0111 + 1011 = 10010) sonucunu verir, bu da **0010** olarak kırpılır.
- Bu anlayışla, Şekil 1.21'deki üç toplama problemini ele alalım. Her durumda, problemi ikinin tümleyeni notasyonuna çevirdik (dört uzunluğundaki bit desenlerini kullanarak), daha önce açıklanan toplama işlemini gerçekleştirdik ve sonucu tekrar alışılagelmiş 10 tabanındaki notasyonumuza çözdük (çevirdik).
- Şekil 1.21'deki üçüncü problemin pozitif bir sayı ile negatif bir sayının toplanmasını içerdiğine dikkat edin; bu durum, ikinin tümleyeni notasyonunun büyük bir faydasını ortaya koyar: İşaretli sayıların herhangi bir kombinasyonunun toplanması, aynı algoritma ve dolayısıyla aynı devre kullanılarak gerçekleştirilebilir. Bu, insanların aritmetik hesaplamaları geleneksel olarak yapma şekline tam bir tezat oluşturur. İlkokul çocuklarına önce toplamayı ve daha sonra çıkarmayı öğretirken, ikinin tümleyeni notasyonunu kullanan bir makinenin sadece nasıl toplayacağını bilmesi yeterlidir.

Şekil 1.21 İkinin tümleyeni gösterimine dönüştürülmüş toplama problemleri



- **Taşma (Overflow) Sorunu**
- Önceki örneklerde kaçındığımız bir sorun, herhangi bir ikinin tümleyeni (two's complement) sisteminde temsil edilebilecek değerlerin boyutunda bir sınır olmasıdır. 4 bitlik desenlere sahip ikinin tümleyeni kullanıldığında, temsil edilebilecek en büyük pozitif tam sayı 7, en küçük negatif tam sayı ise -8'dir. Özellikle 9 değeri temsil edilemez; bu da $5 + 4$ problemine doğru cevabı almayı umamayacağımız anlamına gelir. Aslında sonuç -7 olarak görünür. Bu olguya **taşma (overflow)** denir. Yani taşma, bir hesaplamanın temsil edilebilecek değerler aralığının dışında kalan bir değer üretmesi durumunda ortaya çıkan sorundur. İkinin tümleyeni notasyonu kullanıldığında, bu durum iki pozitif değer toplanırken veya iki negatif değer toplanırken meydana gelebilir. Her iki durumda da, durum cevabın işaret biti kontrol edilerek tespit edilebilir. İki pozitif değerin toplanması negatif bir değer için desen (işaret biti 1) oluşturuyorsa veya iki negatif değerin toplamı pozitif görünüyorsa (işaret biti 0), bir taşma olduğu belirtilir.

- Elbette, çoğu bilgisayar örneklerimizde kullandığımızdan daha uzun bit desenlerine sahip ikinin tümleyeni sistemleri kullandığından, daha büyük değerler taşmaya neden olmadan işlenebilir. Günümüzde, değerleri ikinin tümleyeni notasyonunda saklamak için 32 bitlik desenler kullanmak yaygındır; bu da taşma meydana gelmeden önce 2,147,483,647 kadar büyük pozitif değerlerin birikmesine olanak tanır. Eğer daha da büyük değerlere ihtiyaç duyulursa, daha uzun bit desenleri kullanılabilir veya belki de ölçü birimleri değiştirilebilir. Örneğin, inç yerine mil cinsinden bir çözüm bulmak, daha küçük sayıların kullanılmasına neden olur ve yine de gerekli doğruluğu sağlayabilir.
- Buradaki asıl nokta, bilgisayarların hata yapabileceğidir. Bu nedenle, makineyi kullanan kişi söz konusu tehlikelerin farkında olmalıdır. Bir sorun, bilgisayar programcılarının ve kullanıcıların rehavete kapılması ve küçük değerlerin birikerek büyük sayılar üretebileceği gerçeğini görmezden gelmesidir. Örneğin, geçmişte değerleri ikinin tümleyeni notasyonunda temsil etmek için 16 bitlik desenler kullanmak yaygındı; bu da 2^{15} = 32,768 veya daha büyük değerlere ulaşıldığında taşma olacağı anlamına geliyordu. 19 Eylül 1989'da, bir hastane bilgisayar sistemi yıllarca süren güvenilir hizmetin ardından arızalandı. Yakından yapılan inceleme, bu tarihin 1 Ocak 1900'den sonraki 32,768. gün olduğunu ve makinenin tarihleri o başlangıç tarihine göre hesaplayacak şekilde programlandığını ortaya çıkardı. Böylece, taşma nedeniyle 19 Eylül 1989 tarihi negatif bir değer üretti; bu, bilgisayarın programının işlemeye göre tasarlanmadığı bir durumdu.

Fazlalık Notasyonu (Excess Notation)

- Tam sayı değerlerini temsil etmenin bir başka yöntemi de fazlalık notasyonu (excess notation)'dur. İkinin tümleyeni notasyonunda olduğu gibi, bir fazlalık notasyonu sistemindeki değerlerin her biri aynı uzunlukta bir bit deseni ile temsil edilir. Bir fazlalık sistemi oluşturmak için, önce kullanılacak desen uzunluğunu seçeriz, ardından o uzunluktaki tüm farklı bit desenlerini ikili (binary) sistemde sayıyormuşuz gibi sırayla yazarız. Daha sonra, en anlamlı biti (en solundaki bit) 1 olan ilk desenin listenin yaklaşık yarısında göründüğünü gözlemleriz. Sıfırı temsil etmek için bu deseni seçeriz; bunu izleyen desenler 1, 2, 3... sayılarını temsil etmek için kullanılır; ondan önceki desenler ise -1, -2, -3... sayılarını temsil etmek için kullanılır. Dört uzunluğundaki desenler kullanıldığında ortaya çıkan kod Şekil 1.22'de gösterilmiştir. Orada, 5 değerinin **1101** deseniyle ve -5 değerinin **0011** ile temsil edildiğini görürüz. (Bir fazlalık sistemi ile bir ikinin tümleyeni sistemi arasındaki bir farkın, işaret bitlerinin tersine çevrilmiş olması olduğuna dikkat edin.)

Şekil 1.22 Bir fazlalık sekiz dönüşüm tablosu

Bit deseni	Temsil edilen değer
1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	−1
0110	−2
0101	−3
0100	−4
0011	−5
0010	−6
0001	−7
0000	−8

- Şekil 1.22'de temsil edilen sistem, **fazlalık sekiz notasyonu (excess eight notation)** olarak bilinir. Nedenini anlamak için, önce koddaki desenlerin her birini geleneksel ikili sistemi kullanarak yorumlayın ve ardından bu sonuçları fazlalık notasyonunda temsil edilen değerlerle karşılaştırın. Her durumda, ikili yorumlamanın (değerin), fazlalık notasyonu yorumlamasından 8 değeri kadar fazla olduğunu göreceksiniz. Örneğin, ikili notasyonda **1100** deseni 12 değerini temsil eder, ancak fazlalık sistemimizde 4'ü temsil eder; ikili notasyonda **0000**, 0'ı temsil eder, ancak fazlalık sisteminde negatif 8'i temsil eder. Benzer bir şekilde, beş uzunluğundaki desenlere dayanan bir fazlalık sistemi, **fazlalık 16 notasyonu** olarak adlandırılır; çünkü örneğin **10000** deseni, olağan değeri olan 16'yı temsil etmek yerine sıfırı temsil etmek için kullanılır. Aynı şekilde, üç bitlik fazlalık sisteminin **fazlalık dört notasyonu** olarak bilineceğini doğrulamak isteyebilirsiniz (Şekil 1.23).

Bit deseni	Temsil edilen değer
111	3
110	2
101	1
100	0
011	–1
010	–2
001	–3
000	–4

Şekil 1.23 Üç uzunluklu bit desenleri kullanan bir fazlalık gösterim sistemi

- BİLGİSAYAR MÜHENDİSLİĞİNE GİRİŞ PROJESİ

ONUR İPEK

24360859090