EEM212 - SAYISAL DEVRE TASARIMI DERS NOTLARI

DERS NOTU 1: DİJİTAL SİSTEMLERE GİRİŞ

Dr. İsmail Öztürk *

<ismail.ozturk@amasya.edu.tr>

İçindekiler

*Amasya Üniversitesi Teknoloji Fakültesi EEM Bölümü
Daha fazla bilgi için: https://iozturk.com

1 Dijital Sistemler

Günümüzde hemen hemen hepimiz, dijital sistemlere şu veya bu şekilde bağlı olarak yaşıyoruz. Haberleşme araçlarımızın çoğu, bilgisayarlar, tıbbi cihazlar, akıllı televizyonlar, kara, hava ve deniz ulaşım araçları, beyaz eşyalar gibi günlük hayatta kullandığımız pek çok aygıt çeşitli dijital sistemler içermektedir. İnternetin; kullandığımız akıllı telefonun; netflix gibi yeni nesil yayın servislerinin; twitter, whatsapp ve instagram gibi sosyal medya uygulamalarının dijital sistemlere bağlı olduğunu düşündüğümüzde, dijital sistemlerin hayatımızdaki yerinin önemi daha iyi anlaşılacaktır.

Kamera şu kadar piksel dediğimizde aslında ızgara şeklinde dizilmiş fotosensörlerin sayısından bahsediyoruz.

Buna rağmen pek çoğumuz günlük hayatta dijital sistemlerin ne kadar yer tuttuğunun farkına varmayız. Örneğin, telefonunuzla bir fotoğraf çekip bunu instagrama saniyeler içerisinde yüklediğinizde, bu işlemin arka planında ne kadar çok dijital sistemin kullanıldığını düşünmeyiz bile. Bu işlemde, öncelikle ışık kamera üzerindeki lens yardımıyla sensör üzerine toplanır. Dijital kamera sensörü, temel olarak bir ızgara şeklinde yan yana dizilmiş fotosensörlerden oluşur. Fotosensörler, üzerlerine düşen ışık şiddetiyle orantılı olarak çıkışa bir elektrik akımı verir.

Tanım 1.1: Transdüser

Transdüserler enerjiyi bir formdan alıp diğerine dönüştüren aygıtlardır. Yukarıdaki örnekte ışık enerjisi elektrik enerjisine dönüştürülmektedir.

Bu aşamada çıkışta elde edilen elektriksel akımın **analog sinyal**ler olduğuna dikkat etmek gerekir. Analog sinyaller Tanım 1.2 gibi tanımlanmaktadır.

Tanım 1.2: Sinyal, Analog Sinyal ve Sürekli Sinyal

Fiziksel bir değişkene ait veri dizilerine sinyal denir. Örneğin osiloskop ekranında bir gerilimin zamana göre değişimini gözlemlediğinizde gördüğünüz "elektriksel gerilime ait veri dizileri" yani kısaca bir elektriksel sinyaldir.

Analog ismi ise ing. *analogous*'un kısaltması olup "eşdeğer" anlamına gelmektedir. Bu bakımdan, başka bir sinyalin eşdeğeri olan sürekli sinyallere analog sinyal denir.

Bir sinyalin sürekli olması ise o sinyalin alabileceği değerlerin sürekli olmasıdır. Mesela, -1V ile $+1$V arasında değişen "analog" bir sinyal herhangi bir anda -1V ile $+1$V arasındaki herhangi bir değeri alabilecektir: $+0.736$V, -0.0000123V, $-0.37475733...$V gibi.

Bu tanımlamalara göre, yukarıdaki örnekte ışık şiddetinin zamana göre değişimine ışık sinyali dersek, hem ışık sinyali hem de transdüserin çıkışındaki elektrik akımı sinyali sürekli olacaktır. Ayrıca, elektrik akımı sinyali ışık sinyaliyle orantılı olarak üretildiği için ışık sinyalinin eşdeğeri (analoğu) olmaktadır. Bu nedenle, transdüser çıkışındaki elektriksel akım analog bir sinyaldir diyebiliriz.

Fotosensörün çıkışındaki analog elektrik akımı sinyalinin dijital bir sistem olan cep telefonunuz tarafından doğrudan kullanılması mümkün değildir. Çünkü dijital sistemler (adı üstünde) ancak **dijital sinyal**leri değerlendirebilir.

Tanım 1.3: Dijital Sinyal

Dijital sinyaller herhangi bir anda sadece sınırlı sayıda ayrık değer alabilen sinyallerdir. Burada ayrıktan kasıt sinyalin alabileceği değerler arasında boşluklar olmasıdır. Dijital sistemlerde bu boşluklar dijital sinyalin çözünürlüğünün ya da bir veriyi ne kadar doğrulukla temsil edebileceğinin bir ölçüsüdür.
Örneğin 0.25V aralıklı değerlere sahip -1V ile $+1$V arasında değişen "dijital" bir sinyalin herhangi bir anda alabileceği değerler $\{-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1\}$ 'dir.

Peki dijital sistemler neden bu şekilde ayrık değerlerle çalışmak zorunda? Bunun nedeni dijital sistemlerde verilerin **bit** adı verilen ve sadece 0 ile 1 değerlerini alabilen temel yapı taşlarının sonlu sayıdaki kombinasyonuyla ifade edilebilmesidir. Elimizde sonlu sayıda bitler olduğu için dijital bir sistemde saklayabileceğimiz bir sinyal herhangi bir anda sadece sınırlı sayıda değer alabilecektir. Başka bir deyişle, dijital sistemlerde saklayabileceğimiz sinyaller sadece dijital sinyallerdir ve bu nedenle dijital sistemlere "dijital" adını veriyoruz.

Peki analog bir sinyal olan transdüser çıkışını dijital bir sistem olan telefonunuza nasıl aktaracağız? İşte bu gibi durumlarda **analog / dijital dönüştürücü**ler (kısaca A/D dönüştürücü veya ADC [1]) dediğimiz elektronik elemanlardan yararlanıyoruz. Adından da anlaşılacağı üzere bu devreler girişteki analog bir sinyali çıkışta dijital bir sinyale dönüştürür. Çıkış dijital olduğu için çıkışın alabileceği değerler sınırlıdır.

Örnek 1.1:

Bir A/D dönüştürücünün çıkışı toplam 8 bite sahipse bu dönüştürücü çıkışta kaç farklı değeri ifade edebilir?

Bit sayısı arttıkça çıkışın analog sinyali daha doğru ifade edebileceğini fark ettiniz mi?

Daha önce söylediğimiz üzere dijital sistemlerin temel yapı taşı olan bitler sadece 0 ve 1 verisini taşıyabilir. Buna göre tek bir bitin tüm olası kombinasyonları $\{0, 1\}$ olmak üzere 2 tanedir. Eğer 2 tane biti yan yana koyarsak bu sefer $\{00, 01, 10, 11\}$ olmak üzere $2 \times 2 = 2^2 = 4$ farklı kombinasyonu ifade edebiliriz.

[1] ing. Analog Digital Converter

Bunu 8-bitlik çıkış için genelleştirirsek çıkışta toplam $2\times2\times2\times...\times2 = 2^8 = 256$ farklı değeri ifade edebileceğimiz sonucuna ulaşırız.

Instagram örneğimize geri dönecek olursak, bu aşamada artık A/D dönüştürücü kullanarak ışık şiddeti bilgisini taşıyan analog sinyali bir dijital sinyale dönüştürerek dijital sistem içerisinde kullanabiliriz [2]. A/D dönüştürücünün çıkışından elde edilen bit dizileri öncelikle telefon üzerindeki işletim sisteminin anlayabileceği bir görüntü formatına dönüştürülmelidir. Bunun için bitlerin pozisyon ve renk bilgisinin işlenmesi, A/D dönüşümden kaynaklanan hataların giderilmesi, milyonlarca bit verisinin hafızada daha az yer kaplaması için sıkıştırma algoritmalarıyla sıkıştırılması [3] gibi pek çok işlem kameranın kendine ait dijital devreleri tarafından halledilir. Dijital kameraların bu işlemleri yapmak için kendilerine ait mikroişlemcileri [4] vardır.

Tüm bu işlemler yapılıp görüntü formatı hazırlandıktan sonra veri telefona aktarılmalıdır. Dijital sistemlerde bit verilerinin saklandığı dijital devrelere **hafıza birimi** adı verilir. Telefonun ilgili hafıza birimine kameradan aktarım yapılması için de dijital devrelere ihtiyaç vardır. Dijital sistemlerde farklı donanımlar arasında iletişim yapmak için kullanılan bu tür devrelere **veri yolu** (ing. bus) adı verilir. Veri yolu ile telefonun hafıza birimine aktarılan görüntü artık telefonun işletim sistemi (Android, iOS, vb.) üzerindeki herhangi bir yazılım (ya da uygulama) tarafından kullanıma hazırdır.

Ek Bilgi: Kamera donanımından görüntünün telefona aktarılması işletim sistemi kerneli üzerindeki kamera sürücüsü ile sağlanır.

Genel olarak, işletim sistemi mikroişlemci üzerine komutlar göndererek mikroişlemcinin çalışmasını kontrol eden bir yazılımdır. Mikroişlemcinin çalışması ve haberleşmesi için kullanılan donanımların tümü, yine çeşitli dijital devrelerin birleşimi olan ve **anakart** adı verilen baskı devre kartında [5] toplanmıştır. İşletim sistemi, belli uzunlukta bit dizilerinden oluşan komutlar göndererek mikroişlemciyi kontrol eder. Diğer programlar ve instagram gibi uygulamalar da mikroişlemciye gönderilecek komutlar bütününden başka bir şey değildir.

Şimdi çekmiş olduğunuz fotoğrafı instagram uygulaması üzerinden paylaştığınızı düşünelim. Bu durumda instagram uygulaması (işletim sistemi üzerinden) mikroişlemciye gönderdiği komutlarla fotoğrafın olduğu hafıza adresine erişim talep edecektir. Daha sonra hafızadan okuduğu bit verilerinin veri yolu üzerinden (bir başka dijital devreler bütünü olan) kablosuz internet (wi-fi) donanımına çeşitli bilgilerle beraber iletilmesini sağlayacaktır. Bu çeşitli bilgiler, gönderilen bit verilerinin internet üzerindeki hangi adrese gönderileceği, verilerin hangi kullanıcıya ait olduğu gibi detaylar barındıracaktır. Bu bilgileri alan kablosuz internet donanımı, aldığı bilgileri bağlı olduğu modeme gönderecek ve modem de bağlı olduğu internet servis sağlayıcısının fiber optik hatları üzerinden telefondan gönderilen bit verilerinin dünyanın başka bir

[2] A/D dönüştücünün bit sayısı ne kadar fazla olursa elde edilen görüntü o kadar net olacaktır. Dijital kameralarda bu veri "renk derinliği" olarak adlandırılır.
[3] Fotoğraf dosyalarının sonundaki "jpg", "tiff", "png" gibi uzantılar fotoğrafın sıkıştırıldığı algoritmayı ifade eder.
[4] Dijital sinyal işlemcisi (ing. Digital Signal Processor)
[5] ing. PCB (Printed Circuit Board)

yerinde bulunan instagram sunucularına gitmesini sağlayacaktır. İnstagram sunucularına ulaşan fotoğraf verisi artık internet üzerinden dünyanın herhangi bir yerindeki takipçileriniz tarafından görülebilecektir.

Yukarıda ana hatlarıyla vermiş olduğumuz örnek, günlük hayatta dijital sistemleri kullanarak yaptığımız çok basit görünen işlerin altında ne kadar farklı elektronik dijital devrenin kullanıldığını göstermektedir. Cep telefonunuzun altında dijital kamera, kablosuz internet gibi pek çok alt donanım (**dijital modül**) bulunmakta ve bu modüllerin kendileri de başka başka işlemleri yapmak için tasarlanmış pek çok dijital devre ve mikroişlemciler içermektedir. Bu bakımdan, günlük hayatta kullandığımız pek çok dijital cihazın sadece tek bir dijital elektronik devre olarak tasarlanmadığı; aksine farklı farklı dijital modüllerin bir araya getirilmesiyle oluşturuldukları görülmektedir. İşte dijital bir sistem bu modüllerin bir bütünüdür.

Tanım 1.4: Dijital Sistem

Bir dijital sistem, pek çok dijital modül ve dijital devrenin belli amaçları yerine getirmek için bir araya getirildiği genel yapıya verilen isimdir.

Bu bölümde dijital sistemlerde sayıların ve karakterlerin nasıl temsil edildiği, bunların nasıl saklandığı ve aritmetik işlemlerin nasıl yapıldığı gibi temel bilgiler verilecektir.

2 Dijital Sistemlerde Sayılar

Önceki kısımda ifade ettiğimiz üzere bir dijital sistem sadece 0 ve 1 değerlerini anlayabilir ve sadece bu değerleri üzerinden işlem yapabilir. Peki bu durumda, günlük hayatta kullandığımız $2020, -12$, 3.14 gibi sadece 0 ve 1 rakamlarından oluşmayan sayıları dijital sistemlerde nasıl kullanabiliyoruz? Cevap aslında çok basit. Sadece 0 ve 1 rakamlarını kullanarak da tüm sayıları ifade edebiliriz.

$\{0, 1, 2, ..., 8, 9\}$ **on** tane rakam olduğu için **on**dalık gösterim.

Normalde sayıları ifade ederken sürekli $\{0, 1, 2, \ldots, 8, 9\}$ rakamlarını kullanarak sayıları ifade ediyoruz. Bu tür sayı gösterimine **ondalık gösterim** adı verilmektedir. $r_i \in \{0, 1, 2, \ldots, 8, 9\}$ olmak üzere (yani r_i ondalık bir rakam olmak üzere) bir ondalık sayı genel olarak aşağıdaki gibi ifade edilmektedir:

$$\ldots r_3 r_2 r_1 r_0 . r_{-1} r_{-2} r_{-3} \ldots$$

Burada noktadan sonraki kısım küsuratlı kısmı temsil etmektedir. Örneğin, 359.24 sayısı için $r_2 = 3$, $r_1 = 5$, $r_0 = 9$, $r_{-1} = 2$ ve $r_{-2} = 4$'dür. Bu gösterimde rakamların bulunduğu pozisyon (r_i'nin indisi) ile sayının kendisi arasındaki ilişki aşağıdaki gibidir:

$$\begin{aligned} 359.24 &= 3 \times 10^2 + 5 \times 10^1 + 9 \times 10^0 + 2 \times 10^{-1} + 4 \times 10^{-2} \\ &= r_2 \times 10^2 + r_1 \times 10^1 + r_0 \times 10^0 + r_{-1} \times 10^{-1} + r_{-2} \times 10^{-2} \\ &= \sum_{i=-2}^{2} r_i \times 10^i \end{aligned}$$

Görülebileceği üzere sayıların değeri rakam değerleri ile 10'un katlarının çarpımıyla hesaplanıyor. Aslında burada yaptığımız ondalık gösterim kullandığımız için sayı değerini hesaplarken 10 değerini **taban** olarak almak. Bu nedenle ondalık gösterim aynı zamanda **onluk tabanda gösterim** olarak da ifade edilir.

Sayıları ifade ederken 10 tane parmağımız olduğu için 10'luk tabanı kullanmak tarihteki pek çok medeniyet için en doğal seçim olmuş ve neticede bugün biz de günlük hayatta sayıları 10 tabanında ifade ediyoruz. Fakat, bu böyle olmak zorunda değil ve her zaman her medeniyet için de böyle olmamış. Tarihte Sümerler ve Babiller gibi pek çok medeniyet sayıları 12 ve 60 tabanında kullanmış. Günümüzde bu kullanımın mirası olarak (öğleden önce ve sonra olmak üzere) bir günü 12'ye ve dakikaları 60'a bölüyoruz.

Ek Bilgi

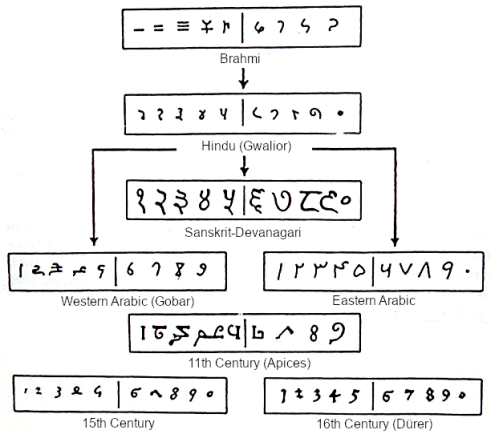
Tarihte neden 12 ve 60 tabanında sayıların kullanıldığını merak ediyorsanız sağ elinizi alın ve sağ el baş parmağınızla diğer parmaklarınızdaki boğumları tek tek sayın. Sonuç olarak (eğer fazladan boğuma sahip değilseniz) 12 bulacaksınız. Bu çift el kullanarak 10'a kadar saymaktansa tek el ile 12'ye kadar saymanızı sağlayacaktır. Şimdi bunu yaparken her 12 sayımda sol elinizin 5 parmağından birini kapatırsanız iki el ile 60'a kadar sayabilirsiniz. Günümüzde parmak saymak önemli olmasa da Sümerler ve Babiller için zamanında bu yöntem 10 parmak kullanan medeniyetlere göre oldukça önemli bir teknik avantaj olmuş olmalı.

Sümerler'in kullandığı 60 tane rakamı internetten araştırıp bulabilirsiniz.

Sayıları herhangi bir tabanda göstermek için tek yapmamız gereken o tabana karşılık gelen sayı kadar rakam bulmak. Mesela, günümüzde onluk tabanda kullandığımız $\{0, 1, 2, \ldots, 8, 9\}$ rakamlarının kökeni Hintliler ve Araplar'dan gelmekte olup tarihsel gelişim evreleri Şekil 1 'de görüldüğü gibidir. Biz tek tek rakam bulmak yerine yine her rakamı r_i ile ifade edersek, genel olarak T tabanındaki bir sayıyı Eşitlik 1'deki gibi ifade edebiliriz.

$$(r_N \ldots r_2 r_1 r_0 . r_{-1} r_{-2} \ldots r_{-M})_T = \sum_{i=-M}^{N} r_i \times T^i \tag{1}$$

Küsurat M basamak iken tamsayı kısmının $N+1$ basamak olduğuna dikkat edin.



Şekil 1: Onluk tabanda kullandığımız rakamların kökeni.

Kaynak: Tobus, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=78836110

Eşitlik 1'de sayının etrafındaki $()_T$ şeklindeki parantez o sayının kullanıldığı tabanı belirtmek için kullanılır. Fakat 10 tabanını sıklıkla kullandığımız için 359.24 gibi sayıları $(359.24)_{10}$ şeklinde ifade etmemiz şart değildir. Yine aynı eşitlikte, N ve M sayıları tam ve küsuratlı kısımlardaki **basamak** sayısını ifade etmektedir.

Örnek 2.1:

Beşlik tabanda kullanacağımız rakamlar $\{0, 1, 2, 3, 4\}$ *olsun. Bu durumda* $(3201.3)_5$ *sayısının ondalık karşılığını bulunuz.*

Sayı Eşitlik 1'e göre $(r_3 r_2 r_1 r_0 . r_{-1})_5$ şeklinde bir sayı olup taban $T = 5$'dir. Buna göre sayının onluk sistemdeki karşılığı:

$$\begin{aligned}(3201.3)_5 &= \sum_{i=-1}^{3} r_i \times 5^i \\ &= 3 \times 5^{-1} + 1 \times 5^0 + 0 \times 5^1 + 2 \times 5^2 + 3 \times 5^3 \\ &= 426.6\end{aligned}$$

Buraya kadar anlattıklarımızla nereye varmak istediğimizi şimdiye kadar anlamışsınızdır. Sadece 0 ve 1 'leri (yani bitleri) kullanarak dijital sistemlerde sayıları nasıl ifade ederiz sorusuna cevap bulmaya çalışıyorduk. Cevabımız çok basit: Eğer sayıları sadece $\{0, 1\}$ rakamlarını kullanarak **ikili taban**da (yani $T = 2$) ifade edersek, tüm sayıları bitlerle ifade etmiş oluruz ve dolayısıyla dijital sistemler üzerinde sayıları kullanabiliriz [6].

[6] Dikkat ederseniz dijital sistemler üzerinde "tüm" sayıları kullanabiliriz demedim. Teorik olarak tüm sayılar bitlerle ifade edilebilir ama az sonra göreceğimiz üzere pratikte bu mümkün değil.

Tanım 2.1: İkili Sayılar

İkili tabanda ifade edilen sayılara kısaca **ikili sayılar** (ing. *binary numbers*) denilir.

Şimdi dijital sistemlerde ikili sayılar kullanılarak farklı sayı türlerinin nasıl temsil edildiğini inceleyelim.

2.1 İşaretsiz Tam Sayılar

Tam sayıları ifade etmek için küsurata ihtiyacımız yoktur. Dolayısıyla, Eşitlik 1'de rakamlar arasındaki noktayı kullanmamıza gerek kalmayacaktır. Ayrıca, ikili tabanda rakamlar bit olduğu için rakamları ifade ermede r_i yerine bitin kısaltması olan b_i kullanacağız. Buna göre, $b_i \in \{0, 1\}$ olmak üzere dijital sistemlerde işaretsiz ikili tam sayılar aşağıdaki gibi ifade edilir:

$$(b_{N-1} \dots b_2 b_1 b_0)_2 \tag{2}$$

Bu notlarda kısaca MSB ve LSB kullanılacaktır.

Burada dikkat etmemiz gereken birkaç husus bulunmaktadır. Öncelikle dijital sistemlerde b_0 ve b_{N-1} (en baştaki ve en sondaki bitler) çok sık kullanıldığı için bunlara özel isimler verilmiştir. b_0 biti en düşük basamağa ($2^0 = 1$) karşılık geldiği için bu bit değerine "**en düşük değerlikli bit**" (ing. *least significant bit*, kısaca **LSB**) denilir. b_{N-1} biti de en yüksek basamağa (2^{N-1}) karşılık geldiği için bu bite de "**en yüksek değerlikli bit**" (ing. *most significant bit*, kısaca **MSB**) denilir. Bu gösterimde MSB'nin indisini $N-1$ seçmemizin nedeni ikili sayının N basamaklı olmasını sağlamaktır. Yoksa MSB'yi b_N ile de temsil edebilirdik; tabi bu durumda ikili sayı $N+1$ basamaklı olurdu.

Şimdi ise bu bit grubunun hangi değerler arasındaki sayıları temsil edebildiğini belirleyelim. Dijital sistemlerde sadece pozitif tam sayıları temsil eden değerlere **işaretsiz tam sayılar** denilir [7]. Dolayısıyla, Eşitlik 2 ile ifade edilen sayıların sadece pozitif sayıları temsil etmesi yeterlidir. Bu da işaretsiz tam sayıların ondalık karşılıkları için Eşitlik 1'deki genel formülümüzü aşağıdaki gibi doğrudan kullanabileceğimiz anlamına gelir:

$$\begin{aligned}(b_{N-1} \dots b_2 b_1 b_0)_2 &= \sum_{i=0}^{N-1} b_i \times 2^i \\ &= b_0 \times 2^0 + b_1 \times 2^1 + \dots + b_{N-1} \times 2^{N-1}\end{aligned}$$

Buna göre, Eşitlik 2'nin alabileceği en düşük değer, tüm bitlerin 0 olduğu değer; yani 0'dır. Eşitliğin alabileceği en büyük değer ise tabi ki tüm bitlerin 1 olduğu

[7] Buna neden işaretsiz denildiği Bölüm 2.2'de açıklanacaktır.

$(11\ldots111)_2$ değeridir. Bu durumda ikili sayının alabileceği maksimum değer yukarıdaki eşitliği kullanarak

$$(11\ldots111)_2 = 1 + 2^1 + 2^2 + \ldots + 2^{N-1} = 2^N - 1$$

şeklinde bulunur. Elde ettiğimiz sonucu aşağıdaki gibi özetleyebiliriz:

İkili tabanda N basamaklı işaretsiz bir tam sayının alabileceği minimum ve maksimum değerler sırasıyla 0 ve $2^N - 1$'dir.

Ek Bilgi

Bazılarınız yukarıda maksimum değerin neden $2^N - 1$ olduğunu anlamamış olabilir. Merak edenler için burada açıklayalım.

Yukarıdaki $1 + 2^1 + 2^2 + \ldots + 2^{N-1}$ toplamına S diyelim. Bu durumda $2S = 2^1 + 2^2 + \ldots + 2^{N-1} + 2^N$ olacaktır. $2S$'den S'yi çıkarırsak eşitliklerin sağ tarafları birbirini aşağıdaki gibi götürür ve S'nin değerini yine aşağıdaki gibi buluruz:

$$\begin{aligned} S = 2S - S = \; & \cancel{2^1} + \cancel{2^2} + \ldots + \cancel{2^{N-1}} + 2^N \\ & - 1 - \cancel{2^1} - \cancel{2^2} - \ldots - \cancel{2^{N-1}} = 2^N - 1 \end{aligned}$$

Örnek 2.2:

a) *$(b_7\ldots b_1b_0)_2$ sayısının alabileceği maksimum ve minimum değerler nelerdir?* ***b)*** *$(11001010)_2$ işaretsiz tam sayısının ondalık karşılığını bulunuz.* ***c)*** *$(11111111)_2$ işaretsiz tam sayısının ondalık karşılığını bulunuz.*

—

a) İşaretsiz ikili sayı 8 basamaklı olduğu için alabileceği minimum ve maksimum değerler sırasıyla 0 ve $2^8 - 1 = 255$ 'dir.

b) Birinci, üçüncü, altıncı ve yedinci bitler 1 olduğuna göre ondalık karşılığı

$$(11001010)_2 = 2^1 + 2^3 + 2^6 + 2^7 = 202$$

c) Tüm bitlerin ağırlıklarını tek tek hesaplayıp toplama yapmaya gerek yok çünkü bu 8 bitlik bir sayının alabileceği maksimum değer. Bu değeri yukarıda hesaplamıştık: 255.

Tanım 2.2: Bayt

Dijital sistemlerde adreslenebilen en küçük bit bloğuna **bayt** adı verilir. Bir bayt 8 bite karşılık gelmektedir.
Örnek 2.2'den görülebileceği üzere bir baytlık veri işaretsiz tam sayı olarak kullanıldığında 0 ve 255 arasındaki tam sayıları temsil edebilir. Eğer sayıları temsil etmek dışında kullanılırsa Örnek 1.1'den hatırlayacağınız üzere 256 farklı veriyi temsil edebilir. (Dikkat ederseniz sayı olarak kullanıldığında bir bayt 0 ve 255 arasındaki 256 farklı tam sayıyı temsil ediyor.)

Ek Bilgi

İşaretsiz tam sayıları bilgisayarda kullanmak isterseniz C programlama dilinde sayı türü olarak "*unsigned int*" seçmeniz gerekir. Bu diğer programlama dillerinde de benzer bir şekilde kullanılır. İşaretsiz tam sayının kaç bayt olacağı programlama diline ve bilgisayar mimarisine göre (32, 64-bit vb.) değişir.

2.2 İşaretli Tam Sayılar

Normalde hangi tabanda olursa olsun sayıların pozitif veya negatif olup olmadığını başlarına + veya − işaretleri koyarak belirliyoruz. Fakat dijital sistemlerde elimizde 0 ve 1 bitleri dışında kullanabileceğimiz + veya − sembolleri olmadığına göre yapabileceğimiz tek şey bu semboller yerine yine 0 ve 1 bitlerini işaret olarak kullanmak. Dijital sistemlerde bir sayının pozitif mi negatif mi olduğunu belirlemek için MSB işaret biti olarak kullanılır. Eğer MSB 0 ise bu + işareti; eğer MSB 1 ise bu − işareti anlamına gelecek şekilde işlemler yapılır. İşte tam da bu nedenle, dijital sistemlerde pozitif veya negatif değerler alan tam sayılara **işaretli tam sayılar** adı verilir.

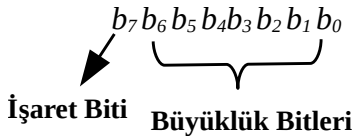
İşaretli tam sayıları ifade etmek için

- İşaretli büyüklük gösterimi (ing. *signed magnitude representation*)
- Birlere tümleyen gösterim (ing. *ones' complement representation*)
- İkiye tümleyen gösterim (ing. *two's complement representation*)

olmak üzere üç farklı yöntem kullanılır.

2.2.1 İşaretli Büyüklük Gösterimi

Bu gösterim dijital sistemlerde negatif sayıları da ifade edebilmek için kullanılabilecek en doğrudan yöntemdir. Bu gösterim tarzında MSB işaret biti olarak kullanılırken MSB'nin sağında kalan bitler o sayının büyüklüğünü (ya da başka bir deyişle mutlak değerini) verir. Bir baytlık veri için bu gösterim aşağıdaki gibi olacaktır:



Burada b_7 sıfır iken $b_6 \dots b_0$ arasındaki bitlerin sayı değeri pozitif; b_7 bir iken ise bu bitlerin ifade ettiği sayı değeri negatif kabul edilir. Bu gösterime ait örnekler Örnek 2.3'deki gibidir.

Örnek 2.3: İşaretli Büyüklük Gösterimi

$(00110010)_2 = +(2^1 + 2^4 + 2^5) = +50$
$(10110010)_2 = -(2^1 + 2^4 + 2^5) = -50$
$(01010101)_2 = +(2^0 + 2^2 + 2^4 + 2^6) = +85$
$(11010101)_2 = -(2^0 + 2^2 + 2^4 + 2^6) = -85$

Genel olarak baktığımızda, bu gösterimde N bitlik bir ikili tam sayımız varsa bir biti işaret için harcadığımız için geriye sayıyı ifade etmek için kullanabileceğimiz $N-1$ bit kalır. Bölüm 2.1'de görmüş olduğumuz üzere bu $N-1$ bitlik sayı 0 ile $2^{N-1}-1$ arasındaki değerleri ifade edebilir. İşaret bitimizi de işin içerisine katarsak aşağıdaki sonuca ulaşırız:

İşaretli büyüklük gösteriminde N bitlik bir tam sayının alabileceği minimum ve maksimum değerler sırasıyla $-(2^{N-1}-1)$ ve $+(2^{N-1}-1)$'dir.

Örnek 2.4:

İşaretli büyüklük gösteriminde bir baytlık veri, tam sayı olarak hangi aralıktaki sayıları ifade edebilir?

$N = 8$ için sağdaki 7 bit 0 ile $2^7 - 1 = 127$ arasındaki sayıları temsil edebildiği için -127 ile $+127$ arasındaki tam sayıları ifade edebilir.

İşaretli büyüklük gösteriminde sıfır sayısını nasıl ifade edebiliriz dediğimizde aklımıza ilk olarak "tabi ki bütün bitleri sıfır yaparak" cevabı gelir. Bu doğrudur. Mesela 4-bitlik işaretli büyüklük gösteriminde $(0000)_2$ değeri $+0$'a karşılık gelmektedir. Fakat, aynı büyüklük için işaret bitini bir yaparsak bu sefer $(1000)_2 = -0$ değerini elde ederiz. Günlük hayatta $+0$ ve -0 aynı şey olduğu için sadece 0 kullansak da, işaretli büyüklük gösteriminde sayıları ifade etmek için kullandığımız yöntem bize aynı sayının hem pozitif hem de negatifini saklamamızı zorunlu kılıyor. Bu nedenle, işaretli büyüklük gösteriminde sıfır sayısı $+0$ ve -0 olmak üzere iki ayrı formatta temsil edilir.

2.2.2 Birlere Tümleyen Gösterim

İşaretli büyüklük gösterimi negatif sayıları temsil etmek için akla ilk gelecek yöntem olsa da kullanım açısından en pratik yöntem değildir. Çünkü bu yöntemde toplama ve çıkarma için ayrı ayrı dijital devreler tasarlanmalıdır. Fakat dijital sistemlerde ne kadar fazla devre kullanılırsa maliyet ve güç tüketimi de o kadar aratacaktır. Bu ise hem üreticiler hem de kullanıcılar açısından istenmeyen bir durumdur. Bu nedenle, dijital devre tasarlayan mühendisler hem toplama hem çıkarma işleminin sadece toplama işlemiyle gerçekleştirilebileceği **tümleyen matematiği** kullanan bir yöntem geliştirdiler. Bu sayede, toplama ve çıkarma için ayrı devreler kullanmak yerine sadece toplayıcı devrelerin kullanılması mümkün olmuş oldu.

Tümleyen matematiği sadece ikili sayılarla değil herhangi bir tabanda kullanılabilir. Daha fazla aşina olduğumuz için öncelikle ondalık sayıları kullanalım. Örnek olarak elimizde sadece üç basamaklı tam sayılar olsun. Üç basamak ile ifade edebileceğimiz sayılar 000 ile 999 arasındaki sayılardır. Sadece toplama işlemi ile hem toplama hem de çıkarma yapabilmek için bu sayıların yarısını pozitif sayıları; diğer yarısını da negatif sayıları temsil etmek için kullanalım. Yani, 000 ile 499 arasındaki sayılar pozitif sayıları temsil ederken 500 ile 999 arasındaki sayılar negatif sayıları temsil etsin. Burada her iki yarıdaki sayıların birbirinin 999'a göre **tümleyeni** olduğuna dikkat edin. Mesela, ilk yarıdaki 350 sayısını 999'dan çıkardığımızda $999 - 350 = 649$ sayısını elde ederiz. 999 sayısından bu 649 sayısını da çıkardığımızda ise yine ilk başlaki $999 - 649 = 350$ sayımıza ulaşırız. Bu durumu, 350 ve 649 sayıları birbirinin 999'a göre tümleyenidirler şeklinde adlandırırız.

Yani tümleyen sayılardan herhangi birini 999'dan çıkardığımızda diğer sayıyı elde ederiz.

Dikkat ederseniz 000 ile 499 arasındaki sayıların 500 ile 999 arasında 999'a göre tek bir tümleyeni vardır. Bu ilişkiyi kullanarak 0 ile 499 arasındaki bir sayının 500 ile 999 arasındaki tümleyenini o sayının negatifi olarak kabul edebiliriz. Mesela, az önceki örneğimizde 350 ve 649 sayılarının birbirinin 999'a göre tümleyeni olduğunu göstermiştik. Buna göre, 350 sayısı +350'yi temsil ederken 649 sayısı tümleyeninin negatifini, yani −350'yi temsil edecektir.

Yukarıda söylemiş olduğumuz üzere bu tip bir gösterim sadece toplama işlemi ile hem toplama hem de çıkarma yapabilmemizi sağlayacaktır. Bunun nasıl mümkün olacağını kavrayabilmek için A sayısından B sayısını çıkarmak istediğimizi varsayalım. $A - B$ şeklinde doğrudan çıkarma işlemi yapmak yerine aynı işlemi $A + (-B)$ ile toplama şeklinde de yapabiliriz. Tabi burada $-B$ değeri B sayısının tümleyeni olacaktır. Fakat, bu toplama işlemini yaparken normal toplamadan farklı özel bir kuralı takip etmemiz gerekir. Bu kural şu şekildedir:

1. Eğer toplam sonucunda en solda bir elde değeri yoksa çıkan sonucu aynen kullanırız.
2. Eğer toplam sonucunda en solda bir elde varsa çıkan sonucu elde ile toplayıp kullanırız.

Her iki duruma da örnek aşağıda verilmektedir:

Örnek 2.5:

$215 - 460$ *işlemini* 999*'a tümleyen gösterimi kullanarak hesaplayınız.*

-460 değerini bulmak için 460'ın tümleyeni alınmalı. Bu ise $999 - 460 = 539$'dur. Buna göre çıkarma işlemini 215 ile 539'u toplayarak da bulabiliriz:

$$\begin{array}{rr} & 215 \\ + & 539 \\ \hline & 754 \end{array}$$

En solda bir elde olmadığı için işlem sonucu budur. Hatırlayacağınız üzere üç basamaklı tümleyen gösteriminde 499'dan büyük sayılar negatif sayıları temsil etmekteydi. Bu nedenle bu sayı aslında $999 - 754 = 245$ olduğu için -245 değerine karşılık gelmektedir.

Örnek 2.6:

$300 - 150$ *işlemini* 999*'a tümleyen gösterimi kullanarak hesaplayınız.*

-150 değerini bulmak için 150'nin tümleyeni alınmalı. Bu ise $999 - 150 = 849$'dur. Buna göre çıkarma işlemini 300 ile 849'u toplayarak da bulabiliriz:

$$\begin{array}{rr} & 300 \\ + & 849 \\ \hline 1 & 149 \\ + & 1 \\ \hline & 150 \end{array}$$

Görmüş olduğunuz üzere toplam sonucunda en solda bir elde olduğu için (kırmızı ile gösterilen) çıkan sonucu bu elde ile toplayıp asıl sonucumuz olan 150'ye sadece toplama işlemi kullanarak ulaşmış olduk.

Dikkat ederseniz üç basamak için kullandığımız 999'a tümleyen gösterimi pozitif olarak $+0$ ile $+499$ arasındaki değerleri; negatif olarak da -0 ile -499 arasındaki değerleri temsil edebilir. Evet, bir kez daha 0 sayısının iki farklı gösterimiyle karşı karşıyayız: 000 sayısı $+0$ iken bunun tümleyeni olan 999 sayısı -0'dır.

Üç basamak için 999'a göre almış olduğumuz tümleyeni iki basamak için 99'a, dört basamak için 9999'a, beş basamak için 99999'a göre (bu böyle uzar gider) alarak da aynı işlemleri yapabiliriz. Genel olarak N basamaklı bir tam sayı için tümleyen değeri N tane dokuzun $999\ldots999$ şeklinde yan yana gelmesiyle ifade edildiğinden, genel olarak bu tümleyen gösterimine **dokuzlara tümleyen gösterim** adı verilir.

***

Şimdi ikili gösterime geri dönelim. Bu aşamada "birlere tümleyen gösterim" adının nereden geldiğini tahmin edebiliyor olmanız lazım. Örnek olarak 4-bitlik ikili sayıları

ele alalım. Bu dört bitlik sayılar $(0000)_2$ ile başlayıp $(1111)_2$ e kadar olan $2^4 = 16$ farklı sayıyı temsil edebilir. Bu 16 sayıyı sekizer sekizer iki yarıya ayırırsak $(0000)_2$ ile $(0111)_2$ arasındaki sayılar bir grup; $(1000)_2$ ile $(1111)_2$ arasındaki sayılar diğer bir grubu oluşturur. Hatırlayacağınız üzere tümleyen matematiğinde bu gruplardan birini negatif sayıları temsil etmek için kullanıyoruz. Dikkat ederseniz bir grubun MSB'si hep 0 iken diğer grubun MSB'si sürekli 1. İşaretli büyüklük gösteriminde MSB 0 ise işaret +; MSB 1 ise işaret − kabul ediyorduk. Birlere tümleyen gösteriminde de ikinci grubu negatif sayıları temsil etmek için kullanırsak, MSB benzer şekilde sayının pozitif mi negatif mi olduğunu anlamak için kullanılabilir!

Onluk tabanda **dokuzlara** tümleyen gösterimi ikili tabanda **birlere** tümleyen gösterimine dönüşür.

Bu durumda tümleyen değerimiz 4 basamaklı en büyük sayı değerimiz olan $(1111)_2$ olacaktır. Bir sayının $(1111)_2$'e ya da kısaca birlere göre tümleyenini bulmak için yapmamız gereken o sayıyı bu tümleyen değerinden çıkarmak. Örnek olarak $(1010)_2$ sayısının birlere göre tümleyenini hesaplayalım:

$$\begin{array}{rr} & 1111 \\ - & 1010 \\ \hline & 0101 \end{array}$$

Buna göre, $(1010)_2$ sayısının birlere tümleyeni $(0101)_2$'dir. Dikkat ederseniz sonuç 0 bitlerinin 1; 1 bitlerinin ise 0 yapılmış hali. Bu tesadüf değil. Hangi sayıyı alırsanız alın birlere göre tümleyeni bu şekilde kolayca bulunabilecektir. Örneğin,

$$\begin{array}{ccc} (1111)_2 & \leftrightarrow & (0000)_2 \\ (0001)_2 & \leftrightarrow & (1110)_2 \\ (0100)_2 & \leftrightarrow & (1011)_2 \\ (1001)_2 & \leftrightarrow & (0110)_2 \end{array}$$

Bu bit değerlerinin tersini alma işlemine dijital sistemlerde "mantıksal DEĞİL" işlemi adı verilir. Bunu sonraki notlarda göreceğiz. Buna göre, ikili bir sayının birlere göre tümleyenini bulmak için tek yapmamız gereken o sayının değilini almak. Yani tümleyen bulmak için herhangi bir çıkarma işlemine ihtiyacımız yok! Zaten asıl amacımız da buydu: Çıkarma işlemi olmaksızın aritmetik işlemler yapabilmek. Şimdi bunu yine dokuzlara tümleyen gösteriminde bahsettiğimiz toplama kuralını kullanarak gerçekleştirebiliriz (yani en solda elde varsa eldeyi bulduğun sonuca ekle; en solda elde kalmıyorsa sonucu aynen kullan):

Örnek 2.7:

Ondalık $6 - 7$ *işlemini* 4*-bitlik birlere tümleyen gösterimi ile gerçekleştiriniz.*

$6 = 2^2 + 2^1$ olduğu için 6'nın 4-bitlik ikili gösterimi $(0110)_2$ olacaktır. Yine aynı şekilde $7 = 2^2 + 2^1 + 2^0 = (0111)_2$. Bizim yapmak istediğimiz işlem $6 + (-7)$ olduğu için hesaplamada kullanmamız gereken değer 7'nin birlere göre tümleyeni olan $(1000)_2$ değeri. Buna göre,

$$\begin{array}{rr} & 0110 \\ + & 1000 \\ \hline & 1110 \end{array}$$

MSB değeri 1 olduğu için bu sayının değeri negatiftir. Değerini belirlemek için birlere tümleyenini alırsak sonucun $-(0001)_2 = -1$ olduğunu buluruz.

Örnek 2.8:

Ondalık $5 - 1$ *işlemini* 4*-bitlik birlere tümleyen gösterimi ile gerçekleştiriniz.*

$5 = 2^2 + 2^0 = (0101)_2$ ve $1 = 2^0 = (0001)_2$. 1 sayısının birlere göre tümleyeni ise $(1110)_2$. Buna göre $5 + (-1)$ işlemi şu şekildedir:

$$\begin{array}{rr} & 0101 \\ + & 1110 \\ \hline 1 & 0011 \\ + & \quad 1 \\ \hline & 0100 \end{array}$$

En solda elde kaldığı için toplama kuralımız gereği bunu sonuca ekleriz. En son bulduğumuz sonucun MSB değeri 0 olduğu için sonucumuz pozitiftir ve değeri de $(0100)_2 = 4$'tür.

Birlere tümleyen gösterimi kullanarak sadece toplama devreleri ve daha sonra göreceğimiz mantıksal DEĞİL kapıları ile hem toplama hem de çıkarma işlemi yapmamız mümkündür. Yukarıda verdiğimiz 4-bitlik örnekte temsil edebileceğimiz sayılar $(0000)_2 = +0$ ile $(0111)_2 = +7$ arasındaki pozitif sayılar ve bunların $(1000)_2 = -7$ ile $(1111)_2 = -0$ arasındaki negatif karşılıklarıdır. Yani 4-bitlik birlere tümleyen gösterim ile -7 ile $+7$ arasındaki değerleri ifade edebilmemiz mümkündür. Bunu aşağıdaki gibi genelleştirebiliriz:

Birlere tümleyen gösterimde N bitlik bir tam sayının alabileceği minimum ve maksimum değerler sırasıyla $-(2^{N-1} - 1)$ ve $+(2^{N-1} - 1)$'dir.

Bu sonuç işaretli büyüklük gösterimi için bulmuş olduğumuz sınır değerleri ile aynıdır. Dolayısıyla, bir baytlık veri birlere tümleyen gösterimde de Örnek 2.4'de hesaplanan değerleri alabilecektir.

Ek Bilgi

Birlere tümleyen gösterim Türkçe literatürde hatalı bir şekilde "bire tümleyen", "birin tümleyeni", "bir tümleyeni" gibi çevrilmektedir. Bunun neden (çoğul olarak) birlere tümleyen olması gerektiğini yukarıda açıkladık. Fakat, başka kaynaklarda aynı yöntem farklı isimlerle karşınıza çıkabilir.

2.2.3 İkiye Tümleyen Gösterim

Birlere tümleyen gösterim bize donanım avantajı sağlamış olsa da $+0$ ve -0 olmak üzere iki farklı sıfır değerinin olması dijital sistemler üzerinde gerçekleştirilecek pek çok işlem için uygun değildir. Eğer birlere tümleyen gösterim kullanılırsa dijital sistem üzerinde bir değerin sıfıra eşit olup olmadığını belirlemek için değerin hem $+0$ hem de -0'a eşit olup olmadığının kontrolü gerekir. Bu ise hem donanımı hem de kullanıcının göz önünde bulundurması gereken şartları karmaşıklaştıracaktır. İşte bu iki farklı sıfır değeri oluşumunun önüne geçmek için tümleyen değeri temsil edilebilecek maksimum değer yerine bu değerin bir fazlası tümleyen değeri olarak seçilir.

Mesela üç basamaklı ondalık sayı örneğimizdeki tümleyen değerimiz olan 999 değeri bir arttırılırsa yeni tümleyen değerimiz 1000 olacaktır. Tabi elimizde sadece üç basamak olduğu için bu değeri saklamamız mümkün değildir. Peki o zaman bir sayının 1000'e göre tümleyenini sadece üç basamak kullanarak nasıl alırız? Bunun için A bir sayı olmak üzere $1000 - A$ işleminin $999 - A + 1$ işlemine eşit olduğunu fark etmeniz gerekiyor. Yani önce dokuzlara göre tümleyen alırız $(999 - A)$, daha sonra elde ettiğimiz sonuca 1 ekleriz ve böylece sadece üç basamak kullanarak A sayısının 1000'e göre tümleyenini hesaplamış oluruz.

Peki bu durumda nasıl tek bir sıfır değeri elde ediyoruz? Önce 000 sayısının tümleyenine bakalım. Bu değer $999 - 000 + 1 = 1000$. Fakat elimizde sadece üç basamak olduğu için en soldaki 1 saklanmaz ve yine 000 değerini elde ederiz. Yani 000 sayısının 1000'e göre tümleyeni yine kendisidir. 999'a göre tümleyen gösteriminde -0 değeri döndüren sayı olan 999'un 1000'e göre tümleyeni ise 001 olup yeni gösterimde -1 değerini temsil eder. Buradan görülebileceği üzere 1000'e göre tümleyen gösterimde tek bir sıfır değeri vardır.

Fakat bu gösterimde temsil edebilecek değerler biraz farklılaşır. Yine 000 ve 499 arasındaki sayılar pozitif değerleri; 500 ile 999 arasındaki değerler ise negatif değerleri temsil eder. Fakat 1000'e tümleyen gösterimde 500 sayısının tümleyeni yine 500'dür. Bütünlük olması açısından 500 sayısının değeri -500 olarak kabul edilir. Bu nedenle 1000'e tümleyen gösterimde temsil edilebilecek değerler -500 ile $+499$ arasındadır.

Hatırlarsanız 999'a tümleyen gösterimde temsil -449 ile $+499$ arasıydı.

1000'e göre tümleyen gösterimin en iyi yanı toplama kuralının 999'a göre tümleyen gösterimden çok daha basit olmasıdır: Toplama yapıldıktan sonra en solda bir elde varsa bile bu elde değeri atılır ve işleme katılmaz. Aşağıdaki örnek bu durumu göstermektedir:

Örnek 2.9:

$360 - 80$ *işlemini* 1000*'e tümleyen gösterimi ile gerçekleştiriniz.*

80'nin tümleyeni $999 - 80 + 1 = 920$. Buna göre,

$$\begin{array}{rr} & 360 \\ + & 920 \\ \hline \not{1} & 280 \end{array}$$

en soldaki elde değerini atarız ve sonuç olarak 280 elde ederiz.

Genel olarak iki basamaklı ondalık sayılar için tümleyen değeri 10^2; üç basamak için 10^3; dört basamak için 10^4 vb. şeklinde hep 10'un katları olduğu için, ondalık tabanda bu tür tümleyen gösterim formatına **ona tümleyen gösterim** adı verilir.

***

Benzer şekilde ikili tabanda bu tür gösterime genel olarak **ikiye tümleyen gösterim** denir. Çünkü tümleyen değerleri artık 2'nin katlarıdır. Örnek olarak yine 4-bitlik sayıları kullanalım . Bu gösterimde sayıların tümleyeni alınırken yukarıda görmüş olduğumuz üzere önce sayının birlere göre tümleyeni (yani değili) alınır ve elde edilen sonuca 1 eklenir. Bazı örnekler aşağıdaki gibidir:

4-bit için ikiye tümleyen değeri $(10000)_2 = 2^4$

$$\begin{aligned}(1111)_2 &\leftrightarrow (0001)_2\\ (0101)_2 &\leftrightarrow (1011)_2\\ (0100)_2 &\leftrightarrow (1100)_2\\ (1001)_2 &\leftrightarrow (0111)_2\end{aligned}$$

Yukarıda bahsettiğimiz üzere $(0000)_2 = 0$ ve $(1000)_2 = -8$'in tümleyenleri yine kendileridir. Dolayısıyla tek bir sıfır vardır ve 4-bitlik ikiye tümleyen gösterimin temsil edebileceği sayılar $(0000)_2 = 0$ ve $(0111)_2 = +7$ arasındaki pozitif tam sayılar ile $(1000)_2 = -8$ ve $(1111)_2 = -1$ arasındaki negatif sayılardır. Bunu genelleştirirsek aşağıdaki sonucu elde ederiz:

İkiye tümleyen gösterimde N bitlik bir tam sayının alabileceği minimum ve maksimum değerler sırasıyla -2^{N-1} ve $+(2^{N-1}-1)$'dir.

Örnek 2.10:

İkiye tümleyen gösterimde bir baytlık veri, tam sayı olarak hangi aralıktaki sayıları ifade edebilir?

$N = 8$ olduğu için $-2^7 = -128$ ile $2^7 - 1 = 127$ arasındaki sayıları ifade edebilir.

Örnek 2.11:

Ondalık $5 - 1$ *işlemini* 4*-bitlik birlere tümleyen gösterimi ile gerçekleştiriniz.*

$5 = (0101)_2$ ve $1 = (0001)_2$. 1 sayısının ikiye göre tümleyeni ise $(1111)_2$. Buna göre $5 + (-1)$ işlemi şu şekildedir:

$$\begin{array}{rr} & 0101\\ + & 1111\\ \hline \not{1} & 0100\end{array}$$

En soldaki elde atılır. En son bulduğumuz sonucun MSB değeri 0 olduğu için sonucumuz pozitiftir ve değeri de $(0100)_2 = 4$'tür.

Sadece tek bir sıfırın temsil edilmesini sağladığı için modern dijital sistemlerin hemen hemen tümünde ikiye tümleyen gösterim kullanılır.

2.3 Kesirli Sayılar

Kesirli sayıları temsil etmek için dijital sistemlerde genel olarak

- Sabit noktalı gösterim
- Kayan noktalı gösterim

olmak üzere iki tür yöntem sıklıkla kullanılır.

2.3.1 Sabit Noktalı Gösterim

Bu gösterim türünde Eşitlik 1 doğrudan kullanılır. Buna göre sonlu uzunluktaki bitler aşağıdaki gibi dizilir ve değerlendirilir:

$$(b_{N-1} \dots b_1 b_0 . b_{-1} b_{-2} \dots b_{-M})_2 = \sum_{i=-M}^{N} b_i \times 2^i$$

Tabi ki bu değerler donanım üzerinde saklanırken ortada bir nokta yoktur. Sadece biz ortada sabit bir nokta varmışçasına noktanın sağındaki bitleri küsuratı temsil etmek için kullanırken noktanın solundaki bitleri tam sayı değerlerini saklamak için kullanırız. Sabit noktadan kastımız tam sayı ve küsuratı temsil etmek için kullanılan bit sayılarının hesaplamalar boyunca sabit kaldığını belirtmektir.

Tıpkı tam sayılarda olduğu gibi sabit noktalı sayılar da işaretli ve işaretsiz olarak kullanılabilir. Eğer işaretli olarak kullanılırsa MSB değeri sayının pozitif mi negatif mi olduğunu belirler. Sabit noktalı sayılarda işaret türü olarak büyük çoğunlukla ikiye tümleyen gösterim kullanılır. Tümleyen alma işlemleri tıpkı tam sayılara uygulanıyormuşçasına uygulanır.

Sabit noktalı sayıları kullanmak için öncelikle tam sayı ve küsurat bitlerinin uzunluklarının belirlenmesi gerekir. Bu QN.M notasyonuyla gösterilir. Burada N tam sayı bitlerinin sayısı iken M küsurat bitlerinin sayısıdır. Eğer sabit noktalı sayı işaretli bir sayı ise yine QN.M notasyonu kullanılır, fakat bu durumda N tam sayı bitinin MSB değerinin işaret biti olduğu kabul edilir.

Q4.2 formatında bazı işaretsiz sabit noktalı sayıların bit gösterimleri ve bunların ondalık karşılıkları aşağıdaki gibidir:

$$(1000.01)_2 = 2^{-2} + 2^3 = 8.25$$
$$(0010.10)_2 = 2^{-1} + 2^1 = 2.5$$
$$(0101.11)_2 = 2^{-2} + 2^{-1} + 2^0 + 2^2 = 5.75$$

Q4.2 formatında bazı işaretli (ikiye tümleyen) sabit noktalı sayıların bit gösterimleri ve bunların ondalık karşılıkları aşağıdaki gibidir:

$$(1000.01)_2 = -(0111.11) = -(2^{-2} + 2^{-1} + 2^0 + 2^1 + 2^2) = -7.75$$
$$(0010.10)_2 = 2^{-1} + 2^1 = 2.5$$
$$(0101.11)_2 = 2^{-2} + 2^{-1} + 2^0 + 2^2 = 5.75$$

Görüldüğü üzere artı işaretli oldukları için son iki örnek işaretsiz gösterimle aynıdır. İlk örnekte ise işaret eksi olduğu için asıl değeri hesaplarken önce sayının birlere göre tümleyenini alıp bu değere 1 ekledik. (1 eklemesini LSB'ye yapıyorsunuz noktanın solundaki bite değil. Çünkü donanım noktanın ne olduğundan habersiz bir şekilde tüm 6-bitlik bloğu tek bir sayı gibi değerlendirir.)

Sabit noktalı gösterimin donanımda gerçekleştirimi kolaydır ve basit bir gösterim tarzı olduğu için aritmetik işlemleri hızlı yapılır. Fakat küsurat kısmı sabit olduğu için büyük ve küçük değerlerin beraber kullanıldığı hesaplamalarda yuvarlamalardan kaynaklı hatalar çok büyük olur.

2.3.2 Kayan Noktalı Gösterim

Kayan noktalı gösterimde sayılar $(1000.01)_2$ şeklinde temsil edilmek yerine normalize edilerek $(1.00001)_2 \times 2^3$ şeklinde saklanır. Yani sayı bir küsurat ve bunun 2 tabanında bir üs değeri ile çarpımı şeklinde ifade edilir. Bu sayıları saklamak için bir işaret bitine, bir grup küsurat bitine ve bir grup da üs verisi bitine ihtiyacımız vardır. IEEE 754 standardına göre 32-bitlik kayan noktalı bir sayı aşağıdaki formatta saklanır:

$$b_{31}|b_{30}b_{29}\ldots b_{23}|b_{22}b_{21}\ldots b_1b_0$$

Burada b_{31} (MSB değeri) sayının işaretini; b_{30} ve b_{23} arasındaki bitler üs değerini; son olarak da b_{22}'den b_0'a kadar olan bitler küsuratı saklamak için kullanılır. Sayıların değerine göre üs değeri sürekli değiştiği için sayının küsuratı ifade eden nokta değerinin yeri sürekli değiştiği için bu sayılar kayan noktalı olarak adlandırılırlar. Çok büyük ve çok küçük değerlerin bir arada olduğu hesaplamalarda yuvarlama hatalarının en aza inmesini sağlarlar. Fakat, donanım gerçekleştirimleri daha zordur ve pek çok dijital sistemde kayan noktalı aritmetik için özel geliştirilmiş donanımların kullanılması gerekmektedir.

Ek Bilgi

IEEE 754 standardına göre 64-bitlik bir kayan noktalı gösterim formatı daha bulunmaktadır. Doğal olarak bu format daha yüksek hassasiyete ve doğruluğa sahiptir. C programlama dilinde 32-bit ve 64-bit kayan noktalı sayılar sırasıyla "*float*" ve "*double*" türleriyle ifade edilir.

2.4 Sayıların Farklı Tabanlarda Gösterimi

Dijital sistemlerde sayılar sadece bitlerle saklansa bile günlük hayatta bu değerleri sadece bitlerle ifade etmek çok uzun bit dizilerinin yazılmasına neden olur. Bu yazımı kısaltmak adına mühendisler ve bilgisayar programcıları sıklıkla bit dizilerini sekizli (oktal) ve on altılı (hex) tabanlarda ifade ederler. Sekizli tabandaki sayıları ifade etmek için (daha önce görmüş olduğumuz üzere) 8 tane rakama ihtiyacımız vardır. Bunun için $\{0, 1, \ldots, 7\}$ rakamları kullanılır. Tek bir oktal rakam üç bitlik veriye karşılık geldiği için bu tabanı kullanarak bit dizilerini 1/3 oranında kısaltmak mümkündür. Mesela, aşağıdaki örnekteki bit dizisi üçlü gruplara ayrılarak her üçlü grup yan tarafa bir oktal değer şeklinde yazılmıştır:

$$(001110101000)_2 = (001\ 110\ 101\ 000)_2 = (1650)_8$$

Benzer şekilde bir bit dizisini on altılı (hex) tabanda ifade etmek için 16 rakama ihtiyacımız var. Onluk tabandaki rakamlarımız 10 tane olduğu için ekstradan 6 karakter daha kullanmamız gerekir ki bu karakterler yerine şu şekilde harfler kullanılır: $\{0, 1, \ldots, 9, a, b, c, d, e, f\}$ [8]. Tek bir hex rakamı dört bitlik veriye karşılık geldiği için bu tabanı kullanarak bit dizilerini 1/4 oranında kısaltmak mümkündür. Mesela, aşağıdaki örnekteki bit dizisi dörtlü gruplara ayrılarak her dörtlü grup yan tarafa bir hex değeri şeklinde yazılmıştır:

$$(001110101000)_2 = (0011\ 1010\ 1000)_2 = (3a8)_{16}$$

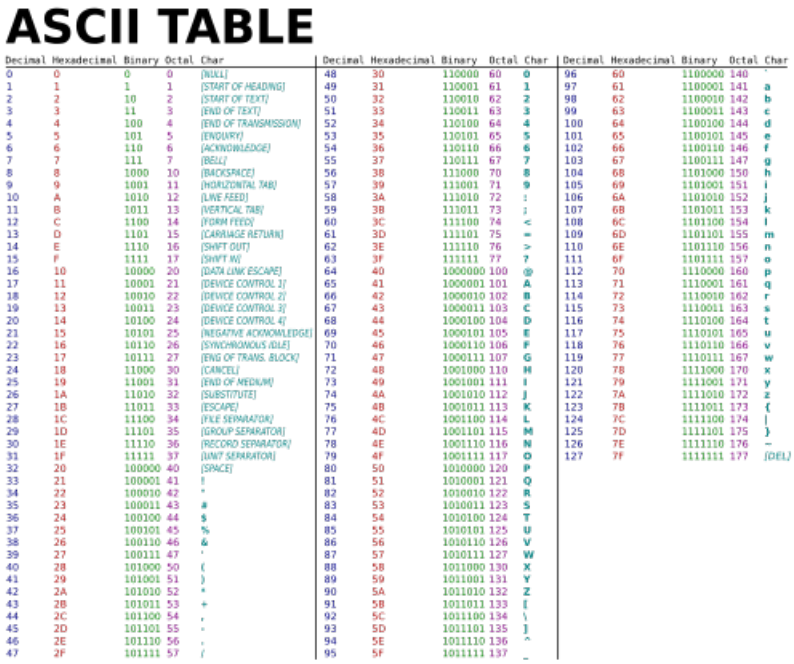
3 Dijital Sistemlerde Karakter Kodlama

Dijital sistemlerde ekranda gördüğünüz, dosyaya yazdığınız veya klavye ile sisteme gönderdiğiniz harfler, rakamlar ve diğer şekiller dijital sistemlerde yine bitlerle saklanır. Bu harf, rakam ve sembollerin her birine **karakter** adı verilir. Hangi bit kombinasyonunun hangi karaktere karşılık geleceğinin belirlenmesine ise **karakter kodlama** denilir. Dijital sistemlerde çok çeşitli karakter kodlama formatları bulunmaktadır. Burada en temel olanlarından kısaca bahsedeceğiz.

3.1 ASCII Karakter Kodlama

ASCII (American Standard Code for Information Interchange) kodlama dijital sistemlerde kullanılmak için tasarlanmış ilk kodlama formatlarından biridir. Bu kodlamada her bir karakter 7-bit ile temsil edilir. Dolayısıyla, bu format sadece $2^7 = 128$ farklı karakteri temsil edebilir. Modern dijital sistemlerde ASCII karakterleri saklamak için bir bayt kullanılır. ASCII karakter kodları Şekil 2'de görülmektedir.

[8] $A, B, \ldots$ şeklinde büyük harfler de kullanılabilir.



Şekil 2: ASCII Karakter Tablosu

Kaynak: Public Domain, https://commons.wikimedia.org/wiki/File:ASCII-Table.svg

Örnek olarak, bu tabloya göre 'A' harfini temsil etmek için bayt değeri $(41)_{16}$ ve 'z' karakterini kodlamak için bayt değeri $(7a)_{16}$ olmalıdır. Görebileceğiniz üzere bu karakter kodlama formatı ne Türkçe ne de İngilizce'den başka dillerdeki hiçbir karakteri içermemektedir. Bu nedenle, bu karakter kodlama günümüzde nadiren kullanılmaktadır.

3.2 Unicode Karakter Kodlama

Unicode kodlama dünyadaki bütün yazı sistemlerinin standart olarak kodlanmasını hedefleyen bir projedir. UTF-8 [9] ve UTF-16 gibi versiyonları vardır. UTF-8 versiyonu bir karakteri en az bir bayt en fazla dört bayt ile temsil eden oldukça kapsamlı bir karakter kodlama formatıdır. ASCII ile uyumludur; yani ASCII bayt değerleri UTF-8'de de aynı karakterlere karşılık gelmektedir. Bunun dışında tüm Türkçe karakterleri ve diğer pek çok dildeki karakterleri de bünyesinde barındırmaktadır. Çok kapsamlı olduğu için tüm UTF-8 karakter tablosunu burada paylaşmamız mümkün değildir. Fakat bu tablolara internetten rahatlıkla ulaşabilirsiniz.

Örnek olarak Türkçe 'ü' harfi UTF-8'de iki bayt olarak temsil edilmektedir. Bu bayt değerleri hex olarak $(c3bc)_{16}$ şeklindedir. Eğer UTF-8 destekli bir bilgisayarınız varsa, bu karakterleri bir programlama dili ile terminale gönderdiğinizde ekrana 'ü' harfinin yazdırıldığını görürsünüz.

[9] UTF=Unicode Transformation Format

3.3 Diğer Karakter Kodlama Formatları

Dijital donanım üreticileri donanımlarında standart karakter kodlamaları kullanmak yerine kendi karakter kodlama formatlarını ortaya koyup kullanabilir. Siz bile kendi karakter kodlama formatınızı oluşturup bu karakterleri dijital ekranlarda yazdırabilirsiniz. Bu nedenle, kullanımda olan daha pek çok karakter kodlama formatı bulunmaktadır. Mesela, Türkçe Windows işletim sistemleri "cp1254" adı verilen bir karakter kodlama formatı kullanır. Eğer Türkçe Windows kullanıyorsanız bilgisayarınızdan alıp Unicode kullanan başka bir bilgisayara kopyaladığınız dosyaları açtığınızda bazı karakterlerin hatalı bir şekilde değiştiğini görürsünüz. Bu gibi durumlarda dönüştürücü programlar kullanıp bir kodlamayı diğerine dönüştürmek gerekir.