

Örneğin, $I_{1,1} = [0, t_1]$, $I_{1,2} = [1 - t_1, 1]$ ve $J_{1,1} = (t_1, 1 - t_1)$; $I_{2,1} = [0, t_2]$, $I_{2,2} = [t_1 - t_2, t_1]$, $I_{2,3} = [1 - t_1, 1 - t_1 + t_2]$, $I_{2,4} = [1 - t_2, 1]$ ve $J_{2,1} = (t_2, t_1 - t_2)$, $J_{2,2} = (1 - t_1 + t_2, 1 - t_2)$; ... C_n , C , V_n ve V 'nin de tanımları aynı kalır.

Uzunlukları önceki gibi hesaplarız; $(C_n) = 2^n t_n$ ve $m(V_n) = 2^{n-1} r_n$ buluruz. O zaman

$$m(C) = \lim_{n \rightarrow \infty} 2^n t_n$$

ve

$$m(V) = \lim_{n \rightarrow \infty} 2^{n-1} r_n = 1 - m(C)$$

olur; yukarıda t_n 'ler üzerine koyduğumuz şarttan dolayı bu limitler vardır. Tabii artık $m(C) = 0$ olması gerekmez. Hatta, bir $0 \leq s < 1$ alıp,

$$t_n = \frac{s}{2^n} + \frac{1-s}{3^n}$$

seçerek $m(C) = s$ olmasını sağlayabiliriz.

Özellik K13. Genel bir Cantor kümesi, basit Cantor kümesinin **K1**, **K2**, **K3**, **K7**, **K9**, **K11** ve **K12** özelliklerini paylaşır. Ayrıca uzunluğu 0 ile 1 arasındaki her hangi bir değeri alabilir.

F. Kaynakça

Bu yazıda anlattıklarımız, genellikle üniversitelerin matematik bölümlerinde 4. sınıfta veya yüksek lisansta okunan ve Lebesgue integrali kullanan Reel Analiz derslerinin konusudur. Bu konuda daha fazla bilgi edinmek isteyenler böyle bir ders kitabına başvurabilirler. Biz, İngilizce olmalarına rağmen, nispeten daha fazla bilgi veren 3 tanesini önereceğiz. Aşağıdaki kitapların ilki, genel Cantor kümeleri için, üçüncüsü ise, Lebesgue tekil fonksiyonun değişik bir tanımı için faydalıdır.

K. R. Stromberg, *An Introduction to Classical Real Analysis*, Wadsworth, Belmont, 1981.

I. P. Natanson, *Theory of Functions of a Real Variable*, Frederick Ungar, New York, 1955.

W. Rudin, *Real and Complex Analysis*, McGraw-Hill, New York, 1974.

BİLGİSAYARLAR NASIL TOPLAMA YAPAR?

George C. Bush*

Çeviren: Hüseyin Dursun

12 veya 17 gibi bir rakamla ne demek istediğimizi bir bilgisayar nasıl anlayabilir? Bunları nasıl toplayabilir veya kendi aralarındaki basit aritmetik işlemleri nasıl yapabilir? Sayıların 0, 1, 2, ..., 9 rakamlarının kombinasyonları ile gösterilmesi hepimiz tarafından bilinmektedir. Bunun dışında bir yöntem önerilmesi veya mevcut yöntemin detaylarının düşünülmesi ilginç olacaktır. 325 sayısı ile ne kastedildiği aşağıdaki şekilde açıklanabilir.

$$\begin{aligned} 325 &= 300 + 20 + 5 \\ &= 3(10^2) + 2(10^1) + 5(10^0) \end{aligned}$$

Bu işlemlerde 10 sayısının seçilmesi sayma işlemlerinde kullandığımız 10 parmağımızdan

kaynaklanıyor olabilir. Fakat bu işlem için 10 dışında bir sayı da kullanılabilir. Fakat 10'dan daha büyük bir sayı kullanılması yeni semboller keşfetmenizi zorunlu kılacaktır. 10'dan daha küçük bir taban seçildiğinde yukarıdaki belirtilen rakam kümesinden istediğimiz kısmını kullanabiliriz.

Aritmetik işlemlerde 10 tabanının kullanılması hiçbirimize ters gelmemektedir. Fakat sayıların bilgisayarlarda bu şekilde gösterilmesi mümkün olmamaktadır. Hemen hemen tüm elektronik aletler ve bilgisayarlar kapalı ve açık anlamına gelen 2 konumla hareket etmektedirler. Bu iki konum 0 ve 1 rakamlarıyla gösterilmektedir. İkilik sistem olarak bilinen

* ODTÜ Matematik Bölümü misafir öğretim üyesi

bu sayı sistemindeki bir sayı '110101' şeklinde gösterilebilir. Bu sayının değeri 10'luk sistem için yukarıda uygulanan metodla bulunabilir.

$$\begin{aligned} 110101 &= 1(2^5) + 1(2^4) + 0(2^3) + 1(2^2) \\ &\quad + 0(2^1) + 1(2^0) \\ &= 32 + 16 + 0 + 4 + 0 + 1 \\ &= 53 \end{aligned}$$

Aşağıda ikilik sistemde verilen sayıların 10'luk sistemdeki karşılıklarını bulunuz.

$$1000, 1101, 1100011, 11110000.$$

Kağıt ve kalemle yapılan hesaplamalarda sayıların büyüklüğü, diğer bir deyişle basamak sayılarının fazlalığı korkutucu bir durum değildir. Sabırlıysanız ve yeterince kağıdınız varsa toplama yaptığımız sayıların basamak adedi için "sadece 5 basamağa kadar olan sayıları toplayabiliyorum" gibi bir bahane öne sürmezsiniz. Fakat, bilgisayarlarda her tamsayı için önceden belirlenmiş hafıza alanını aşamazsınız. Örneğin, kişisel bir bilgisayarda tamsayılar için genellikle 16 bitlik (0'ların ve 1'lerin sayısı) bir alan ayrılmıştır. Sadece pozitif sayılarla ilgileniyorsak

$$0000000000000000 = 0_{10}$$

ile

$$1111111111111111 = 65536_{10}$$

arasındaki sayıları kullanabiliriz. Bundan sonraki verilen örneklerde kolaylık açısından sadece 4 bit kullanacağız.

Bilgisayar 0101 ve 0001 gibi ikilik sistemde verilmiş iki sayıyı nasıl toplar? Çok büyük bir toplama tablosunun önceden yüklenip her toplama işleminde bu tablonun kullanılması tahmin edebileceğiniz gibi etkili denilemeyecek bir yöndendir. 0'larla 1'leri elektronik anlamlarıyla bağdaştırarak kendi yaptığımız gibi sağdan başlayıp sola doğru hareket ederek toplama yaptırmak daha etkili olacaktır. Bilgisayar donanımından yapmasını istediğimiz işi bir örnekle gösterelim:

$$\begin{array}{r} M = 0101 \\ N = 0001 \\ \text{Elde} = \quad \underline{+001} \\ \text{Toplam} = 0110 \end{array}$$

Onluk sistemde $M = 5, N = 1$ ve toplam 6 olmaktadır. Bu şekilde yapılan işlemin

sağlamasını da yapmış olduk. Birkaç örnekle daha pekiştirecek olursak:

$$\begin{array}{r} 2 \rightarrow 0010 \quad 4 \rightarrow 0100 \\ +3 \rightarrow +0011 \quad +1 \rightarrow +0001 \\ 5 \rightarrow 0101 \rightarrow 5 \quad 0101 \rightarrow 5 \rightarrow \underline{\quad} \end{array}$$

$$\begin{array}{r} 6 \rightarrow 0110 \\ +1 \rightarrow +0001 \\ 0111 \rightarrow \underline{\quad} 7 \end{array}$$

Aşağıdaki işlemleri de siz yapmaya çalışınız.

$$\begin{array}{r} 0010 \quad 0100 \quad 0011 \\ + 0101 \quad + 0010 \quad + 0011 \\ \hline ? \quad ? \quad ? \end{array}$$

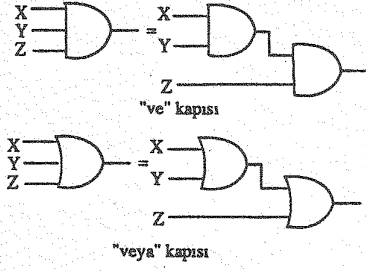


Şekil 1. Temel Bilgisayar Kapıları

0 ve 1'lerin hafızada saklanabilmeleri için pek çok yöntem vardır. Tüm bu yöntemler için Şekil 1'de verilen "kapı" olarak adlandırılan basit elektronik yapılar kullanılmaktadır. "değil" kapısının bir girdisi ve bir çıktısı olup sonuç girdinin tersidir. Diğer bir deyişle girdi 0 ise 1, 1 ise 0'a dönüştürülecektir. Bir "ve" kapısının 2 girdisi, 1 çıktısı mevcuttur. Girdilerin ikisinin 1 olması durumunda çıktı 1 diğer tüm durumlarda ise 0 olmaktadır. Son olarak, bir "veya" kapısının yine aynı şekilde 2 girdisi, 1 çıktısı olup, girdilerin ikisinin 0 olması durumunda çıktı 0, diğer durumlarda ise 1 değerini alacaktır.

Tüm bu temel yapılar çok ucuza üretilmektedir. Bu yüzden çok küçük bir bilgisayarın içerisinde bile bu kapılardan binlercesi mevcuttur.

Herhangi bir aritmetik ifadede ikiden fazla girdi olması çok sık rastlanan bir durumdur. Bu yüzden verilen temel yapıları kullanarak daha karmaşık yapılar oluşturmalıyız. Şekil 2, 3 girdili kapıları nasıl oluşturabileceğimizi göstermektedir.



Şekil 2. 3 girdili kapılar

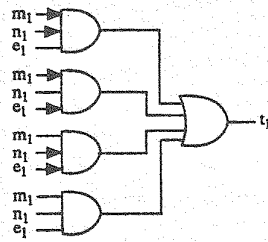
2 girdili temel kapılardan sadece 3 tanesini kullanarak 4-girdili "veya" kapısını yapmaya çalışınız.

5 girdili veya 6 girdili, bir "veya" kapısı için kaç adet 2 girdili temel kapıya ihtiyaç vardır? Daha genel anlamda, 'k' girdili bir "veya" kapısı için kaç temel kapıya ihtiyaç vardır?

Herhangi iki tamsayıyı toplarken başlangıçta verilen örneklerde olduğu gibi her sütun için "elde" kısmını da düşünmek zorundayız. Verilen örneği daha genel hale dönüştürecek olursak:

| | | | | | |
|--------|-----|-------|-------|-------|-------|
| M | $=$ | m_3 | m_2 | m_1 | m_0 |
| N | $=$ | n_3 | n_2 | n_1 | n_0 |
| Elde | $=$ | e_3 | e_2 | e_1 | e_0 |
| Toplam | $=$ | t_3 | t_2 | t_1 | t_0 |

Tahmin edebileceğiniz gibi $e_0 = 0$ olacaktır. Şimdi 1 nolu sütunu inceleyelim. Toplamdaki $t_1; m_1, n_1, l_1$ sayılarından biri veya tamamı 1 olduğunda 1, aksi takdirde 0 olacaktır.

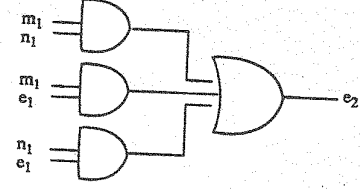


Şekil 3. Toplam basamağının hesaplanması

Şekil 3'te verilen kapı kombinasyonları belirtilen işlemi yapmaktadır.

2. sütundaki elde, $e_2; m_1, n_1, e_1$ sayılarından herhangi ikisi veya üçü 1 olduğu zaman 1, diğer durumlarda 0 olacaktır. Şekil 4'te

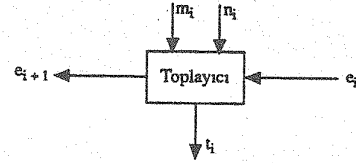
verilen kapı kombinasyonları ise bu işlemi yapmaktadır.



Şekil 4. "elde" basamağının hesaplanması

m_1, n_1 ve e_1 değerleri için mümkün olan tüm kombinasyonları listeleyiniz. Çıkan değerleri 3 ve 4. şekillere uygulayarak sonuçların doğruluğunu kontrol ediniz.

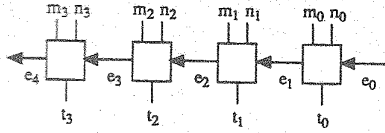
Herhangi 2 tamsayıyı toplarken bu kadar detayı düşünmemeliyiz. Şekil 3 ve 4'te verilen kapıları bir araya getirerek "toplayıcı" adını vereceğimiz bir "karakutu" yapısı oluşturabiliriz. Bu yapı Şekil 5'te verilmiştir.



Şekil 5. Bir "kara kutu" toplayıcı

"Kara kutu" kavramı günlük hayatta bir çok uygulamada kullanıldığı gibi renkle ilgili bir olgu değildir. Bu isim, içinde yapılan işlerin gizliliğini ve bizi hiçbir şekilde ilgilendirmediğini göstermek için kullanılmaktadır. Bizi ilgilendiren sadece doğru çıktı vermesidir.

2 adet 4 basamaklı pozitif tamsayıyı toplayabilmek için bu toplayıcılardan 4 tanesine ihtiyaç duyulmaktadır. Bu toplayıcılar Şekil 6'da verildiği gibi birbirlerine bağlanacaktır. 16 basamaklı sayılar arasında işlem yapabilen bir kişisel bilgisayarda bu toplayıcılardan 16 tanesine ihtiyaç duyulmaktadır. Bu toplayıcılar yaklaşık 250 adet temel kapı içerecektir.



Şekil 6. 4 basamaklı sayılar için bir toplayıcı. Eğer, e_4 eldesi sıfırdan farklıysa, sonucun 4 haneli bir alanda saklanabilmesi problem yaratacaktır. Bir tamsayı için ne kadarlık alan tanımlanırsa tanımlansın toplamın sınırı aştığı duruma ulaşabiliriz. Bu durumların kontrol edilmemesi bizi hatalı sonuçlarla karşı karşıya bırakacaktır.

Aşağıda verilen toplamları kara kutulu toplayıcılardan geçirerek hesaplayınız.

$$\begin{array}{r} 0100 \quad 0110 \quad 1111 \\ +1001 \quad +0101 \quad +0000 \\ \hline ? \quad ? \quad ? \end{array}$$

İkilik sistemde verilen sayıları 10 tabanına çevirerek sonuçlarınızı kontrol edebilirsiniz.

Bilgisayarların pozitif tamsayıları nasıl topladığını gördük. Peki, çıkarma işlemi nasıl yapılacak? Hafızada saklama ve negatif sayıların toplanması? Eğer negatif sayılarla ilgili işlemleri halledebilsek çıkarma işlemi kolaylaşacaktır.

$$M - N = M + (-N).$$

Negatif bir sayının hafızada saklanması için en bilinen yol sol baştaki basamağın işaret olarak kullanılmasıdır. Bu yöntemde 0, '+' ve 1, '-' anlamına gelmektedir. Eğer 16 bitlik bir bilgisayarı düşünecek olursak

$$\begin{array}{l} 1111111111111111 = -32767_{10} \text{ ile} \\ 0111111111111111 = +32767_{10} \end{array}$$

arasındaki sayıları hafızada saklayabiliriz.

Kıyaslama işlemleri sırasında eşitlik kontrolü yaparken +0 ve -0 değerlerinde bir problem yaşanacaktır. Bu durum dikkate alınmazsa çok büyük hatalara yol açabilir.

Pozitif ve negatif sayıların kombinasyonundan oluşan bir aritmetik işlem yaparken daha ciddi problemler yaşanabilir. M ve N sayılarının toplanması için daha karmaşık kurallara ihtiyaç duyacağız. İkisi de pozitif mi? İkisi de negatif mi? İşaretleri farklı mı? Eğer işaretler farklı ise

hangisinin mutlak değeri daha büyük? Günlük hayatta aritmetik yaparken de tüm bu kuralları uygulamaktayız. Aşağıdaki örneklerden bunu görebilirsiniz.

$$(+35) + (-45) = -10$$

$$(-35) + (+45) = +10$$

Tüm bu kuralları bilgisayarda çalışabilir hale getirmek pek de kolay görünmemektedir.

Sıkıntılı gibi görülsün de, negatif sayıların gösterimiyle ilgili bilgisayarda daha iyi çahşabilecek yöntemler mevcuttur.

İkilik sistemde kullandığımız 4 basamaklı örneklerimize tekrar dönelim. -2 sayısını gösterimi için $24 - 2 = 14$ sayısının ikilik sistemdeki karşılığı olan 1110'ı kullanacağız. 2 sayısının, ikilik sistemindeki karşılığını (0010) aldıktan sonra bütün basamaklar "tamamlanacaktır". Tamamlamanın anlamı 0'ların 1, 1'lerin ise 0'a dönüştürülmesidir. Sonuç 1101 olacaktır. Son olarak bu sayıya 1 (0001) ekleyerek 1110 sayısını elde edebiliriz. Artık, bu işlemler için kolaylıkla bir donanım tasarlayabiliriz. Tüm basamaklara 'değil' kapısı uygulayıp 0001 sayısını eklememiz yeterli olacaktır. Bu gösterim "2'lik tamamlayıcısı" olarak adlandırılmaktadır.

-1, -3, -4 ve -6 sayılarının 2'lik tamamlayıcı gösterimlerini bulunuz.

-2 ve -3 sayılarının 2'lik tamamlayıcılarını toplarsak ne olur?

$$\begin{array}{r} -2 \quad 1110 \\ +(-3) \quad + \quad 1101 \\ \hline 11011 \end{array}$$

Sadece 4 basamak saklayabildiğimiz için sol baştaki '1' sayısı kaybolacaktır. Kalan 1011'in anlamı ne olabilir? 2'lik tamamlayıcıya benzeyen bu sayı neyin 2'lik tamamlayıcısı olabilir? Tüm basamaklar tamamlanırsa 0100 elde edilir. 0001'in eklenmesiyle elde edilen 0101 değeri 10'luk sistemde 5 sayısına eşittir. Elde edilen 1011 sonucunun, -5 sayısının 2'lik tamamlayıcısı olduğu kolayca görülebilmektedir. Bu kazara elde edilmiş bir sonuç olmayıp, aşağıdaki örneklerle kontrol edebilirsiniz.

$$\begin{array}{r} -4 \quad \rightarrow \quad 1100 \\ +(-1) \quad \rightarrow \quad 1111 \\ \hline (1)1011 \rightarrow -5 \end{array} \quad \begin{array}{r} -2 \quad 1110 \\ +(-5) \quad 1011 \\ \hline (1)1001 \rightarrow \end{array}$$

Aşağıdaki işlemleri, negatif sayıların 2'lik tamam-

BUSH

layıcılarını bularak yapmaya çalışınız.

$$\begin{array}{r} -3 \quad -3 \\ + (-5) \quad + (-3) \\ \hline \end{array}$$

Yöntem çok karmaşık gibi görünsede, iyi denilebilecek bazı yönleri de mevcuttur. 2'lik sistemden 2'lik tamamlayıcıya geçiş ve 2'lik tamamlayıcıdan, 2'lik sisteme dönüş aynı hesaplamalarla yapılmaktadır. Öyleki, bu işlemler için sadece bir donanım tasarlamak yeterli olacaktır. Her iş için ayrı bir donanıma gerek yoktur.

Ayrıca, 0 için tek gösterim mevcuttur. Böylelikle +0, -0 karmaşası da ortadan kalkacaktır. +0, 0000 ile gösterilmektedir. -0 ise, 0000'nin 2'lik tamamlayıcısının alınıp (1111), bir (0001) eklenmesiyle elde edilen (1)0000 ile gösterilecektir. Sol baştaki 1 gözardı edildiğinde aynı sonucun elde edildiği görülmektedir. 4 basamak için

$$1000 = -8_{10} \text{ ile } 0111 = +7_{10}$$

arasındaki sayılar saklanabilir. 16 bitlik bir kişisel bilgisayarda ise

$$\begin{array}{l} 1000000000000000 = -32768(10) \text{ ile} \\ 0111111111111111 = +32767(10) \end{array}$$

arasındaki sayılar gösterilebilmektedir.

Asıl önemli avantaj ters işaretli sayılar arasındaki işlemlerde ortaya çıkacaktır. Örneğin:

$$\begin{array}{r} 3 \rightarrow 0011 \\ +(-2) \rightarrow + \underline{1110} \\ \hline (1)0001 \end{array}$$

Her zamanki gibi sol baştaki basamağı gözardı edersek 0001 değeri, 10'luk sistemde ise 1 değeri elde edilecektir. Bu sonucun doğru olması yine bir tesadüf değildir. Aşağıdaki örnekleri inceleyiniz.

$$\begin{array}{r} 1 \rightarrow 0001 \quad -4 \quad 1100 \\ -4 \rightarrow + \underline{1100} \quad +3 \quad 0011 \\ \hline 1101 \rightarrow -3 \quad \quad \quad 1111 \rightarrow -1 \end{array}$$

$$\begin{array}{r} 5 \rightarrow 0101 \\ 06 \rightarrow + \underline{1010} \\ \hline 1111 \rightarrow -1 \end{array}$$

Sonuç -8 ile +7 arasında olduğu müddetçe doğru cevap elde edilecektir. 16 bitlik bir kişisel bilgisayarda ise tamsayılar için yapılan işlemlerde sonuç -32768_{10} ile 32767 arasında olduğu sürece doğru sonuca ulaşılabilecektir.

Aşağıdaki işlemleri, pozitif sayıları 2'lik sisteme, negatif sayıları ise 2'lik tamamlayıcılarına çevirerek yapmaya çalışınız.

$$\begin{array}{r} -6 \quad -3 \quad -4 \\ +2 \quad +3 \quad 0 \\ \hline ? \quad ? \quad ? \end{array}$$

Eğer cevap verilen aralığa düşmüyorsa çıkan sonucu mevcut basamak sayıları ile göstermek mümkün olmayacaktır. Aşağıdaki örneği inceleyecek olursak sonucun mümkün olan aralığı geçtiğini göreceksiniz.

$$\begin{array}{r} 4 \rightarrow 0100 \\ 5 \rightarrow + \underline{0101} \\ \hline 1001 \end{array}$$

Sonucu bir sayının 2'lik tamamlayıcısı olarak kabul edersek -7 sayısını elde ederiz. Bu tip aksamalar "tamsayı taşması" olarak adlandırılmaktadır. Bazı programlama dilleri bu gibi durumlarda hiç işlem yapmadan hata mesajı verselerde, bazıları (Turbo Pascal'da dahil) hiç bir mesaj vermeden işlemi hatalı olarak sürdürmektedirler. Eğer kullandığımız programlama dili bu tip hataları veremiyorsa, yaptığımız programlarda bu olasılığı da gözönünde bulundurmalı ve uyarı mesajı verdirmeliyiz.