Sabit Noktalı Sayı Gösterimi Giriş (Introduction to Fixed Point Number Representation)

CS61c Spring 2006

Yazar: Hayden So

Son Güncelleme: 2/28/06

Giriş: Gerçek Dünyada Gerçek Sayılar

Gerçek hayatta, gerçek sayılarla -- kesirli kısımları olan sayılarla -- uğraşırız. Modern bilgisayarların çoğu floating point sayıları için doğal (hardware) desteğe sahiptir. Ancak floating point kullanımı, kesirli sayıları temsil etmenin tek yolu değildir. Bu makale, gerçek sayıların fixed point gösterimini açıklar. Fixed point veri türünün kullanımı, dijital sinyal işleme (DSP) ve oyun uygulamalarında yaygındır; burada performans bazen hassasiyetten daha önemlidir. Daha sonra göreceğimiz gibi, fixed point aritmetiği floating point aritmetiğinden çok daha hızlıdır.

Her Şey Bir Integer ile Başlar

Binary bir sayının:

$$110101_2$$

şu değeri temsil ettiğini hatırlayın:

$$1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$= 32 + 16 + 4 + 1$$
$$= 53_{10}$$

Şimdi, 53 sayısını 2'ye böldüğümüzde sonucun 26.5 olması gerektiğini biliyoruz. Ancak sadece integer gösterimlerimiz olsaydı bunu nasıl temsil ederdik?

Binary Point

26.5 gibi kesirli sayıları temsil etmenin anahtarı binary point kavramıdır. Binary point, decimal sistemdeki decimal point gibidir. Bir sayının integer ve kesirli kısmı arasında ayırıcı görevi görür.

Decimal sistemde, decimal point bir sayıdaki katsayının $10^0 = 1$ ile çarpılması gereken pozisyonu belirtir. Örneğin, 26.5 sayısında, 6 katsayısının $10^0 = 1$ ağırlığı vardır. Peki decimal point'in sağındaki 5'e ne olur? Deneyimlerimizden biliyoruz ki 10^{-1} ağırlığını taşır. "26.5" sayısının "yirmi altı buçuk" değerini temsil ettiğini biliyoruz çünkü:

$$2 \times 10^1 + 6 \times 10^0 + 5 \times 10^{-1} = 26.5$$

Decimal point'in aynı kavramı binary gösterimize de uygulanabilir ve "binary point" oluşturur. Decimal sistemde olduğu gibi, binary point $2^0 = 1$ teriminin katsayısını temsil eder. Binary point'in solundaki tüm rakamlar (veya bitler) 2^0, 2^1, 2^2, vb. ağırlıkları taşır. Binary point'in sağındaki rakamlar (veya bitler) 2^{-1}, 2^{-2}, 2^{-3}, vb. ağırlıkları taşır. Örneğin, şu sayı:

```
11010.1₂
```

şu değeri temsil eder:

```
2⁵ 2⁴ 2³ 2² 2¹ 2⁰ 2⁻¹ 2⁻² 2⁻³
... 1  1  0  1  0  1  0   ...
```

$= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1}$
$= 16 + 8 + 2 + 0.5$
$= 26.5$

Shifting Anahtar

Dikkatli bir okuyucu şimdi 53 ve 26.5'in bit deseninin tamamen aynı olduğunu fark etmelidir. Tek fark, binary point'in pozisyonudur. 53_{10} durumunda, "hiç" binary point yoktur. Alternatif olarak, binary point'in en sağda, 0 pozisyonunda bulunduğunu söyleyebiliriz. (Decimal'de düşünün, 53 ve 53.0 aynı sayıyı temsil eder.)

```
2⁵ 2⁴ 2³ 2² 2¹ 2⁰ Binary 2⁻¹ 2⁻² 2⁻³
          Point
1  1  0  1  0  1  .   0   0   0
```

26.5_{10} durumunda, binary point 53_{10}'dan bir pozisyon sola yerleştirilmiştir:

```
2⁵ 2⁴ 2³ 2² 2¹ 2⁰ Binary 2⁻¹ 2⁻² 2⁻³
          Point
0  1  1  0  1  0  .   1   0   0
```

Şimdi, derste bir integer'ı 1 bit pozisyonu sağa shift etmenin sayıyı 2'ye bölmeye eşdeğer olduğunu tartıştığımızı hatırlayın. Integer durumunda, kesirli bir kısmımız olmadığından, binary point'in sağındaki rakamları temsil edemeyiz, bu da bu shifting işlemini bir integer bölme yapar. Ancak bu sadece binary sayıların integer gösterimlerinin bir sınırlamasıdır.

Genel olarak, matematiksel açıdan, sabit bir binary point pozisyonu verildiğinde, bir sayının bit desenini 1 bit sağa shift etmek her zaman sayıyı 2'ye böler. Benzer şekilde, bir sayıyı 1 bit sola shift etmek sayıyı 2 ile çarpar.

Fixed Point Sayı Gösterimi

Yukarıdaki shifting işlemi, fixed point sayı gösterimini anlamanın anahtarıdır. Bilgisayarlarda (veya genel olarak herhangi bir hardware'de) gerçek bir sayıyı temsil etmek için, binary point'i bir sayının bir pozisyonunda örtük olarak sabitleyerek bir fixed point sayı türü tanımlayabiliriz. Daha sonra sayıları temsil ederken bu örtük kurala uyacağız.

Kavramsal olarak bir fixed point türü tanımlamak için ihtiyacımız olan tek şey iki parametredir:

- sayı gösteriminin genişliği, ve
- sayı içindeki binary point pozisyonu

Bu makalenin geri kalanında **fixed<w,b>** notasyonunu kullanacağız; burada **w** bir bütün olarak kullanılan bit sayısını (bir sayının Genişliği), **b** ise en az anlamlı bitten sayarak binary point pozisyonunu belirtir (0'dan saymaya başlayarak).

Örneğin, **fixed<8,3>** 8-bit'lik bir fixed point sayıyı belirtir; bunun 3 en sağdaki biti kesirlidir. Bu nedenle bit deseni:

```
0 0 0 1 0 1 1 0
```

şu gerçek sayıyı temsil eder:

$$00010.110_2$$

$= 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-2}$
$= 2 + 0.5 + 0.25$
$= 2.75$

Bilgisayarda bir bit deseninin her şeyi temsil edebileceğini unutmayın. Bu nedenle aynı bit deseni, **fixed<8,5>** türü gibi başka bir türe "cast" edilirse, şu sayıyı temsil edecektir:

$$000.10110_2$$

$= 1 \times 2^{-1} + 1 \times 2^{-3} + 1 \times 2^{-4}$
$= 0.5 + 0.125 + 0.0625$
$= 0.6875$

Bu bit desenini integer olarak ele alırsak, şu sayıyı temsil eder:

$$10110_2$$

$= 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1$
$= 16 + 4 + 2$
$= 22$

Negatif Sayılar

Şimdiye kadar pozitif sayılar hakkında konuştuk, ama negatif sayıları da temsil etmek istiyoruz, değil mi? Fixed point negatif sayıları nasıl temsil ederiz o halde?

Bilgisayarda negatif sayıları temsil etmek için 2's complement kullanırız. 2's complement sayılarının özelliklerinden biri, pozitif veya negatif sayıların aritmetik işlemlerinin aynı olmasıdır. Bu, toplama, çıkarma ve şaşırtıcı olmayacak şekilde shifting'i içerir. Pozitif sayılarda yapabildiğimiz gibi, negatif 2's complement sayıları da basit bir 1 bit sağ shift ile sign extension yaparak 2'ye bölebiliriz.

Bu makalenin başında fixed point sayıların basitçe bir integer'ın shifted versiyonu olduğunu (binary point'i sıfır olmayan bir pozisyona ayarlayarak) tartıştığımızı hatırlayın. Shift işleminin pozitif sayılar kadar negatif 2's complement sayıları için de geçerli olduğu gözlemiyle birleştirince, fixed point'te negatif sayıları nasıl temsil edebileceğimizi kolayca görebiliriz: 2's complement kullanın.

Örnek olarak, aşağıda 4-bit 2's complement ile temsil edilebilen tüm sayılar bulunmaktadır:

Bit Deseni	**Temsil Edilen Sayı (n)**	**n / 2**
1 1 1 1	-1	-0.5
1 1 1 0	-2	-1
1 1 0 1	-3	-1.5
1 1 0 0	-4	-2
1 0 1 1	-5	-2.5
1 0 1 0	-6	-3
1 0 0 1	-7	-3.5
1 0 0 0	-8	-4
0 1 1 1	7	3.5
0 1 1 0	6	3
0 1 0 1	5	2.5
0 1 0 0	4	2
0 0 1 1	3	1.5
0 0 1 0	2	1
0 0 0 1	1	0.5
0 0 0 0	0	0

Bu tabloya bakarak, binary point'in 1 pozisyonunda olduğunu varsayarsak, -2.5 sayısını "1011" bit deseniyle temsil edebileceğimizi kolayca anlayabiliriz.

Fixed Point Sayı Gösteriminin Artıları ve Eksileri

Şimdiye kadar fixed point sayıların gerçekten de integer gösteriminin yakın akrabası olduğunu bulmalısınız. İkisi yalnızca binary point pozisyonunda farklılık gösterir. Aslında, integer gösterimini binary point'in 0 pozisyonunda olduğu fixed point sayılarının "özel bir durumu" olarak bile düşünebilirsiniz. Bir bilgisayarın integer üzerinde gerçekleştirebileceği tüm aritmetik işlemler bu nedenle fixed point sayılara da uygulanabilir.

Bu nedenle, fixed point aritmetiğinin faydası, bilgisayarlarda integer aritmetiği kadar anlaşılır ve verimli olmalarıdır. Integer aritmetiği için inşa edilmiş tüm hardware'i fixed point gösterimi kullanarak gerçek sayı aritmetiği gerçekleştirmek için yeniden kullanabiliriz. Başka bir deyişle, fixed point aritmetiği bilgisayarlarda ücretsiz gelir.

Fixed point sayının dezavantajı, elbette floating point sayı gösterimleriyle karşılaştırıldığında aralık ve hassasiyet kaybıdır. Örneğin, **fixed<8,1>** gösteriminde, kesirli kısmımız yalnızca 0.5'lik bir quantum'a kadar hassastır. 0.75 gibi sayıları temsil edemeyiz. 0.75'i **fixed<8,2>** ile temsil edebiliriz, ama sonra integer kısmında aralık kaybederiz.

C'de Fixed Point Sayı Kullanımı

C'de fixed point sayı için doğal bir "tür" yoktur. Ancak fixed point gösteriminin doğası gereği, basitçe bir türe ihtiyacımız yoktur. Fixed point sayılardaki tüm aritmetiklerin integer ile aynı olduğunu hatırlayın; fixed point aritmetiği gerçekleştirmek için C'deki **int** integer türünü yeniden kullanabiliriz. Binary point pozisyonu yalnızca onu ekranda yazdırdığımızda veya farklı "türlerle" aritmetik gerçekleştirdiğimizde önemlidir (örneğin **int**'i **fixed<32,6>**'ya eklerken).

Sonuç

Fixed point, bilgisayarda kesirli sayıları temsil etmek için basit ama çok güçlü bir yoldur. Bir bilgisayarın tüm integer aritmetik devrelerini yeniden kullanarak, fixed point aritmetiği floating point aritmetiğinden kat kat daha hızlıdır. Bu nedenle birçok oyun ve DSP uygulamasında kullanılmaktadır. Öte yandan, floating point sayı gösteriminin sunduğu aralık ve hassasiyetten yoksundur. Bir programcı veya devre tasarımcısı olarak sizin bu değiş tokuşu yapmanız gerekir.