CHAPTER 4: DSP YAZILIMI

Ad Soyad: Reşat Karakaya **Numara:** 348284 **Öğretim:** 1.Öğretim

Sayı Kesinliği

Sayı gösterimi ile ilgili hatalar, ADC sırasındaki niceleme hatalarına çok benzer. İstenilen değerleri sürekli bir dizide saklamak için; ancak, yalnızca sınırlı sayıda nicelenmiş düzeyi temsil edebilir. Bazı programlama dilleri, 32 bit, sabit nokta ve ikinin gümleyicisi olarak saklanan uzun tamsayı olarak adlandırılan bir değişkene izin verir. Sabit nokta değişkenlerinde, bitişik sayılar arasındaki boşluklar her zaman tam olarak birdir. Kayan noktalı gösterimde, bitişik sayılar arasındaki boşluklar temsil edilen sayı aralığında değişir. Rastgele bir kayan nokta sayısı seçersek, bu sayının yanındaki boşluk, sayının kendisinden yaklaşık on milyon kat daha küçüktür. Aritmetik işlemlerin her birindeki yuvarlama hatası, X değerinin, başlangıç değerinden kademeli olarak uzaklaşmasına neden olur. Bu sapma, hataların nasıl bir araya geldiğine bağlı olarak iki şekilde olabilir. Yuvarlama hataları rastgele pozitif ve negatif ise, değişkenin değeri rastgele artacak ve azalacaktır. Hatalar ağırlıklı olarak aynı işarete sahipse, değişkenin değeri çok daha hızlı ve tekdüze bir şekilde uzaklaşacaktır. Ek hata, kabaca tek bir işlemin yuvarlama hatasına eşittir ve toplam işlem sayısı ile çarpılır. Karşılaştırıldığında, rastgele hata yalnızca işlem sayısının kareköküyle orantılı olarak artar. Belirli bir algoritmanın bu iki davranıştan hangisinin yaşayacağını kontrol etmek veya tahmin etmek neredeyse imkansızdır. Tek duyarlıklı kayan nokta, +16,8 milyon (tam olarak ± 2 24) arasındaki her tam sayı için tam bir ikili gösterime sahiptir. Bu değerin üzerinde, seviyeler arasındaki boşluklar birden büyüktür ve bazı tam sayı değerlerinin gözden kaçmasına neden olur. Bu, kayan noktalı tam sayıların (± 16,8 milyon arasında) yuvarlama hatası olmadan eklenmesine, çıkarılmasına ve çarpılmasına izin verir.

TABLE 4-1
Program for demonstrating floating point error accumulation. This program initially sets the value of X to 1.000000, and then runs through a loop that should ideally do *nothing*. During each loop, two random numbers, A and B, are added to X, and then subtracted back out. The accumulated error from these additions and subtraction causes X to wander from its initial value. As Fig. 4-4 shows, the error may be random or additive.

```
100 X = 1                 'initialize X
110 '
120 FOR I% = 0 TO 2000
130   A = RND             'load random numbers
140   B = RND             'into A and B
150   '
160   X = X + A           'add A and B to X
170   X = X + B
180   X = X - A           'undo the additions
190   X = X - B
200   '
210   PRINT X             'ideally, X should be 1
220 NEXT I%
230 END
```

Tablo 4-1

Tablo 4-1'deki program, yuvarlama hatasının (matematik hesaplamalarda niceleme hatası) DSP'de nasıl sorunlara neden olduğunu göstermektedir. Program döngüsü içinde, kayan noktalı değişken X'e rastgele iki sayı eklenir ve ardından tekrar çıkarılır. İdeal olarak, bu hiçbir şey yapmamalıdır.

Yürütme Hızı: Programlama Dili

DSP programlama gevşek bir şekilde üç karmaşıklıktan oluşur: Montaj, Derlenmiş ve Uygulamaya Özel. İkili sistemde çalışmak zor olduğundan bu birler ve sıfırlara gerçekleştirdikleri işleve göre adlar atanır. Bu programlama seviyesine assembly adı verilir. Çalıştırılabilir kod doğrudan mikroişlemci üzerinde çalıştırılabilir. Bir sonraki karmaşıklık düzeyi, belirli donanıma herhangi bir referans olmaksızın soyut değişkenleri işleyebilir. Bunlara derlenmiş veya yüksek seviyeli diller denir. C, BASIC, FORTRAN, PASCAL, APL, COBOL, LISP. Gibi bir düzine kadar yaygın kullanımdadır. Derleyici üst düzey kaynak kodunu doğrudan makine koduna dönüştürmek için kullanılır. Bu, derleyicinin başvurulan soyut değişkenlerin her birine donanım belleği konumları atamasını gerektirir. Derleyici karmaşık matematiksel ifadeleri daha temel aritmetiğe böler. Mikroişlemciler yalnızca toplama, çıkarma, çarpma ve bölme işlemlerini bilir. Daha karmaşık olan her şey, bu temel işlemlerin bir dizisi olarak yapılmalıdır. Üst düzey diller, programcıyı donanımdan ayırır. Bu, programlamayı çok daha kolay hale getirir ve kaynak kodunun farklı mikroişlemciler arasında taşınmasına izin verir. En önemlisi, derlenmiş bir dil kullanan programcının bilgisayarın iç işleyişi hakkında hiçbir şey bilmesine gerek yoktur. Mevcut projenize eklemek için yeni geliştirilmiş bir DSP mikroişlemci satın aldığınızı varsayalım.

Bu cihazlar genellikle DSP için birçok yerleşik özelliğe sahiptir: analog girişler, analog çıkışlar, dijital G / Ç, kenar yumuşatma ve yeniden yapılandırma filtreleri. Kişisel bilgisayarların hızı her yıl yaklaşık %40 arttığından, montajda bir rutin yazmak, donanım teknolojisinde yaklaşık iki yıllık bir sıçramaya eşdeğerdir. DSP uygulamasını C dışında bir dille programlamanın 3 nedeni vardır: Birincisi, DSP o kadar hızlı büyüdü ki, bazı kuruluşlar ve bireyler FORTRAN ve PASCAL gibi diğer dillerde sıkışıp kaldılar. İkinci olarak, bazı uygulamalar yalnızca montaj programlamasıyla elde edilebilen en yüksek verimliliği gerektirir. Üçüncüsü, C, özellikle yarı zamanlı programcılar için öğrenmesi kolay bir dil değildir. Bilgisayar bilimcileri ve programcıları C'yi (veya daha gelişmiş C ++) kullanır. Güç, esneklik, modülerlik; C her şeye sahiptir. Genel bir kural olarak, montajda yazılan bir alt yordamın, karşılaştırılabilir üst düzey programdan 1.5 ila 3,0 kat daha hızlı olmasını bekleyin. Tam değeri bilmenin tek yolu kodu yazmak ve hız testleri yapmaktır. Kişisel bilgisayarların hızı her yıl yaklaşık %40 arttığından, montajda bir rutin yazmak, donanım teknolojisinde yaklaşık iki yıllık bir sıçramaya eşdeğerdir.

Yürütme Hızı: Donanım

Orijinal IBM PC, 4.77 MHz saat hızı ve 8 bit veri yoluna sahip 8088 mikroişlemciye dayanan 1981'de tanıtıldı. Bunu her 3-4 yılda bir tanıtılan yeni nesil kişisel bilgisayarlar izledi: 8088 → 80286 → 80386 → 80486 → 80586 (Pentium). Bu yeni sistemlerin her biri, bilgi işlem hızını önceki teknolojiye göre yaklaşık beş kat artırdı. 1996'da saat hızı 200 MHz'e ve veri yolu 32 bite yükseldi. Diğer iyileştirmelerle, bu, yalnızca 15 yılda bilgi işlem gücünde yaklaşık binlik bir artışla sonuçlandı. Bilgisayarlar birçok alt sistemden oluştuğundan, belirli bir görevi yürütmek için gereken süre iki ana faktöre bağlı olacaktır:

1) Bireysel alt sistemlerin hızı

2) Bu bloklar arasında veri aktarımı için geçen süre. Merkezi İşlem Birimi (CPU) sistemin kalbidir her biri 32 bit tutabilen bir düzine kadar kayıttan oluşur.

CPU'ya ayrıca, bitlerin etrafında hareket ettirilmesi ve sabit nokta aritmetiği gibi temel işlemler için gerekli olan dijital elektronikler dahildir. Karmaşık matematik, verilerin matematik işlemcisi olan aritmetik mantık birimi veya ALU olarak da adlandırılan özel bir donanım devresine aktarılmasıyla ele alınır. Matematik yardımcı işlemci, CPU ile aynı çipte bulunabilir veya ayrı bir elektronik cihaz olabilir. Çoğu kişisel bilgisayar yazılımı, bir matematik işlemcisi ile veya onsuz kullanılabilir. Bu, derleyicinin her iki durumu da işlemek için makine kodu üretmesini sağlayarak gerçekleştirilir ve bunların tümü son çalıştırılabilir programda depolanır. Verilerin alt sistemler arasında aktarılabileceği hız, sağlanan paralel veri hatlarının sayısına ve her bir hat boyunca iletilebilen dijital sinyallerin maksimum hızına bağlıdır. Çipler arasında veri aktarımına kıyasla dijital veriler genellikle tek bir yonga içinde çok daha yüksek bir hızda aktarılabilir. Bir ardışık düzen mimarisi, belirli bir görev için gerekli donanımı birbirini takip eden birkaç aşamaya böler. Artan hızı elde etmek için kullanılan dahili mimari şunları içerir:

1) Çipte bulunan çok sayıda çok hızlı önbellek,

2) Program ve veriler için ayrı veri yolları, bu ikisine aynı anda erişilmesine izin verir (Harvard Mimarisi olarak adlandırılır)

3) Doğrudan mikroişlemcide bulunan matematik hesaplamaları için hızlı donanım ve

4) Bir boru hattı tasarımı. Bir boru hattı mimarisi, belirli bir görev için gerekli donanımı birbirini takip eden birkaç aşamaya böler.

İki sayının eklenmesi üç boru hattı aşamasında yapılabilir. İşlem hattının ilk aşaması, eklenecek sayıları bellekten almaktan başka bir şey yapmaz. İkinci aşamanın tek görevi iki sayıyı birbirine eklemektir. Üçüncü aşama, sonucu bellekte saklar. Her aşama görevini tek bir saat döngüsünde tamamlayabilirse, tüm prosedürün yürütülmesi üç saat döngüsü alacaktır.

Yürütme Hızı: Programlama İpuçları

Bilgisayar donanımı ve programlama dilleri, yürütme hızını en üst düzeye çıkarmak için önemli olsa da bunlar günlük olarak değişen bir şey değildir. Buna karşılık, nasıl programlandığı herhangi bir zamanda değiştirilebilir ve programın ne kadar süre çalıştırılacağını büyük ölçüde etkiler.3 önemli öneri verilebilir. İlk olarak, mümkün olduğunda kayan noktalı değişkenler yerine tamsayılar kullanılır. Kişisel bilgisayarlarda kullanılanlar gibi geleneksel mikroişlemciler, tam sayıları kayan noktalı sayılardan 10 ila 20 kat daha hızlı işler. Matematik işlemcisi olmayan sistemlerde, fark 200'e 1 olabilir. Bunun bir istisnası, genellikle değerleri kayan noktaya dönüştürerek gerçekleştirilen tamsayı bölmesidir. Bu, işlemi diğer tamsayı hesaplamalarıyla karşılaştırıldığında çok yavaş hale getirir. İkinci olarak, karmaşık işlemleri kullanmaktan kaçının. Üçüncüsü, bilgisayar sisteminde neyin hızlı neyin yavaş olduğunu öğrenin. Bu deneyim ve testlerle birlikte gelir ve her zaman sürprizler olacaktır. Grafik komutlarına ve G / Ç'ye özellikle dikkat edin. Bu

gereksinimleri karşılamanın genellikle birkaç yolu vardır ve hızlar çok farklı olabilir. İşlevler bir dizi toplama, çıkarma ve çarpma olarak hesaplanır. Diğer bir seçenek, bu yavaş fonksiyonları önceden hesaplamak ve değerleri bir başvuru tablosunda (LUT) saklamaktır. Örneğin, bir direnç boyunca voltajı sürekli olarak izlemek için kullanılan 8 bitlik bir veri toplama sistemi hayal edin. İlgilenilen parametre dirençte dağılan güç ise, ölçülen voltaj şu hesaplamada kullanılabilir: P = V 2 / R. Daha hızlı bir alternatif olarak, olası 256 voltaj ölçümünün her birine karşılık gelen güç önceden hesaplanabilir ve bir LUT'ta saklanabilir. Sistem çalışırken, 0 ile 255 arasında bir dijital sayı olan ölçülen voltaj, ilgili gücü bulmak için LUT'ta bir indeks haline gelir. Arama tabloları, doğrudan hesaplamadan yüzlerce kat daha hızlı olabilir.

EQUATION 4-3
Maclaurin power series expansion for three transcendental functions. This is how computers calculate functions of this type, and why they execute so slowly.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \cdots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} + \cdots$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots$$

Maclaurin güç serisinde ilişkiler sonsuz uzunlukta olsa da terimler hızla göz ardı edilebilecek kadar küçülür. Örneğin: sin (1) = 1- 0,166666 + 0,008333- 0,000198 + 0,00002-…

Bu işlevlerin hesaplanmasının, tek bir toplama veya çarpmaya göre yaklaşık on kat daha uzun olması gerekir. Bu hesaplamaları atlamak için birkaç numara kullanılabilir, Örneğin: x3 = x · x · x; x çok küçük olduğunda sin (x) ≈x; sin (-x) = -sin (x) olarak kullanılır. Burada değerlerden birinin biliniyor olması ve diğerinin bulunması gerekiyor. Çoğu dil yalnızca birkaç işlev sağlar ve diğerlerinin ilişkiler aracılığıyla türetilmesi beklenir.