

Orijinal slaytların
çevirisidir.

William Stallings Computer Organization and Architecture 10th Edition

- Bilgisayar sistemi performansı konusu
- Bilgisayar kaynaklarının dengeli kullanımı ihtiyacını ele alma
- Mevcut ve öngörülen talebi karşılamak için performans artışı sağlamayı amaçlayan modern bilgisayar organizasyonu tasarımları
- Karşılaştırmalı bilgisayar sistemi performansını değerlendirmek için bir araç sağlamak üzere geliştirilmiş araçlar ve modeller

+ Chapter 2

Performance Issues



Designing for Performance

- Bilgisayar sistemlerinin maliyeti önemli ölçüde düşmeye devam ederken, bu sistemlerin performansı ve kapasitesi de eşit ölçüde artmaya devam ediyor
- Bugünün dizüstü bilgisayarları, 10 veya 15 yıl öncesine ait bir IBM mainframe bilgi işlem gücüne sahip
- İşlemciler o kadar ucuz ki artık attığımız mikroişlemcilerimiz var
- Günümüzün mikroişlemci tabanlı sistemlerinin büyük gücünü gerektiren masaüstü uygulamaları şunları içerir:
 - Image processing
 - Three-dimensional rendering
 - Speech recognition
 - Videoconferencing
 - Multimedia authoring
 - Voice and video annotation of files
 - Simulation modeling
- İşletmeler, transaction ve veritabanı işlemlerini yönetmek ve geçmiş yılların devasa ana bilgisayar merkezlerinin yerini alan büyük istemci / sunucu ağlarını desteklemek için giderek daha güçlü sunuculara güveniyor
- Bulut hizmeti sağlayıcıları, geniş bir müşteri yelpazesi için yüksek hacimli, yüksek transaction hızı olan uygulamaları karşılamak için devasa yüksek performanslı sunucu bankaları kullanır



Ancak mikroişlemcinin ham hızı (*raw speed*), bilgisayar komutları şeklinde yapılacak sabit bir iş akışı ile beslenmedikçe potansiyeline ulaşamayacaktır. Düzgün akışın (*smooth flow*) önüne geçen her şey, işlemcinin gücünü zayıflatır.

Microprocessor Speed

Günümüz işlemcilerinde yerleşik teknikler şunları içerir:

Intel x86 işlemcilerine veya IBM ana bilgisayarlarına böylesine akıllara durgunluk veren derecede güç veren şey, işlemci yongası üreticilerinin aralıksız hız arayışıdır. Bu makinelerin evrimi, daha önce bahsedilen Moore yasasını tasdik etmeye devam ediyor.

Pipelining	• İşlemci veri ve komutların süreçlerini, tıpkı bir boru-hattındaki çalışma gibi aynı anda işler.
Branch prediction	• İşlemci, bellekten getirilen komut kodunda ileriye bakar ve hangi dallanmaların veya komut gruplarının daha sonra işleneceğini tahmin eder.
Superscalar execution	• Bu, her işlemci saat döngüsünde (clock cycle) birden fazla komut icrası yeteneğidir. (Aslında, birden fazla paralel boru hattı kullanılmaktadır.)
Data flow analysis	• İşlemci, optimize edilmiş bir komut çizelgesi oluşturmak için hangi komutların birbirlerinin sonuçlarına veya verilerine bağlı olduğunu analiz eder
Speculative execution	• Dallanma tahminini ve veri akışı analizini kullanarak, bazı işlemciler, program icrasındaki gerçek görünümünden önce komutları spekülasyon olarak yürütür, sonuçları geçici konumlarda tutar ve icra birimlerini olabildiğince meşgul tutar.

İşlemci gücü son derece yüksek bir hızla ilerlerken, bilgisayarın diğer kritik bileşenleri buna ayak uyduramadı.

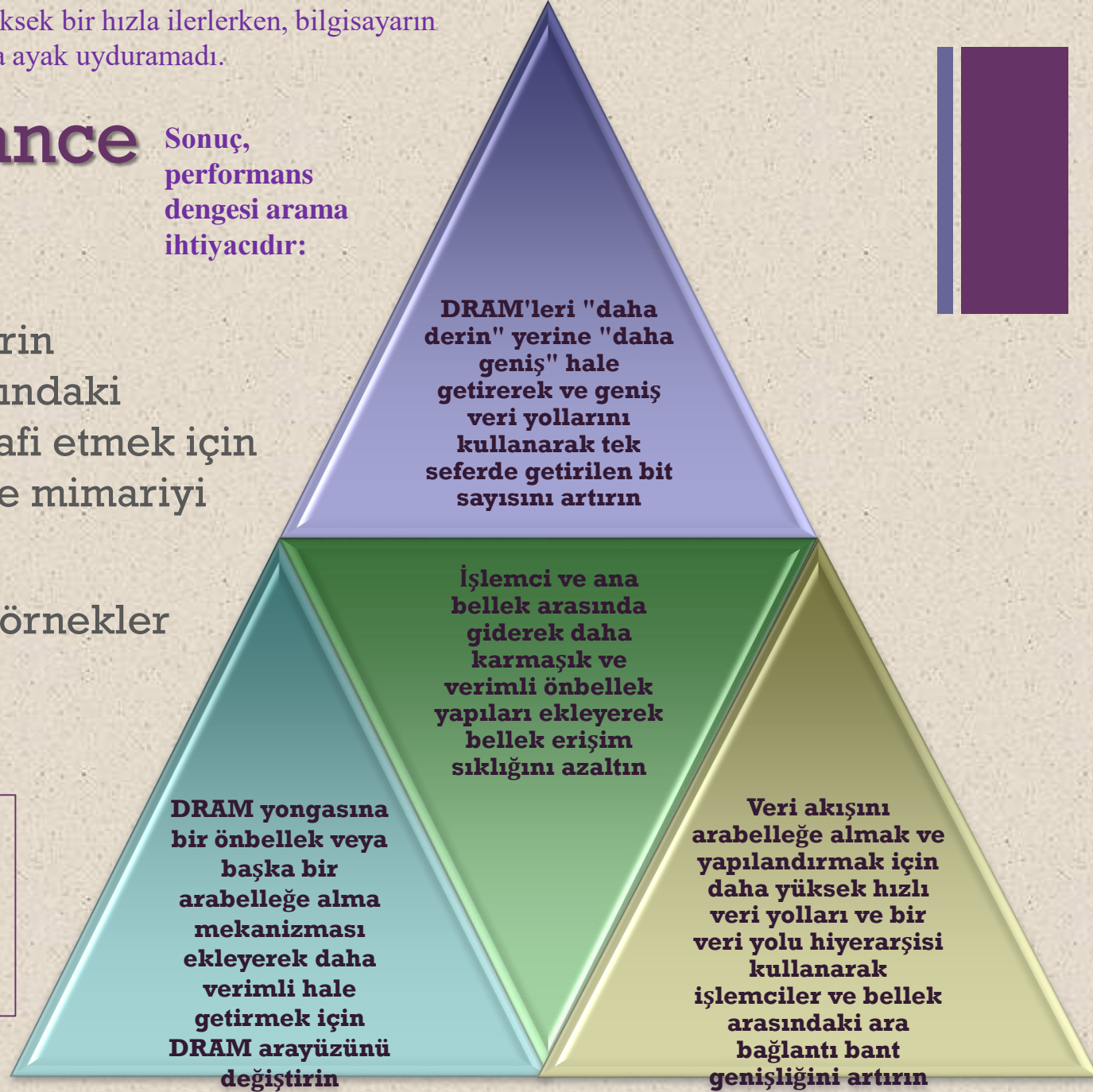


Performance Balance

Sonuç,
performans
dengesi arama
ihtiyacıdır:

- Çeşitli bileşenlerin yetenekleri arasındaki uyumsuzluğu telafi etmek için organizasyonu ve mimariyi ayarlama
- Modern mimari örnekler şunları içerir:

Bu tür uyumsuzlukların yarattığı sorun, özellikle işlemci ve ana bellek arasındaki arayüzde kritiktir.

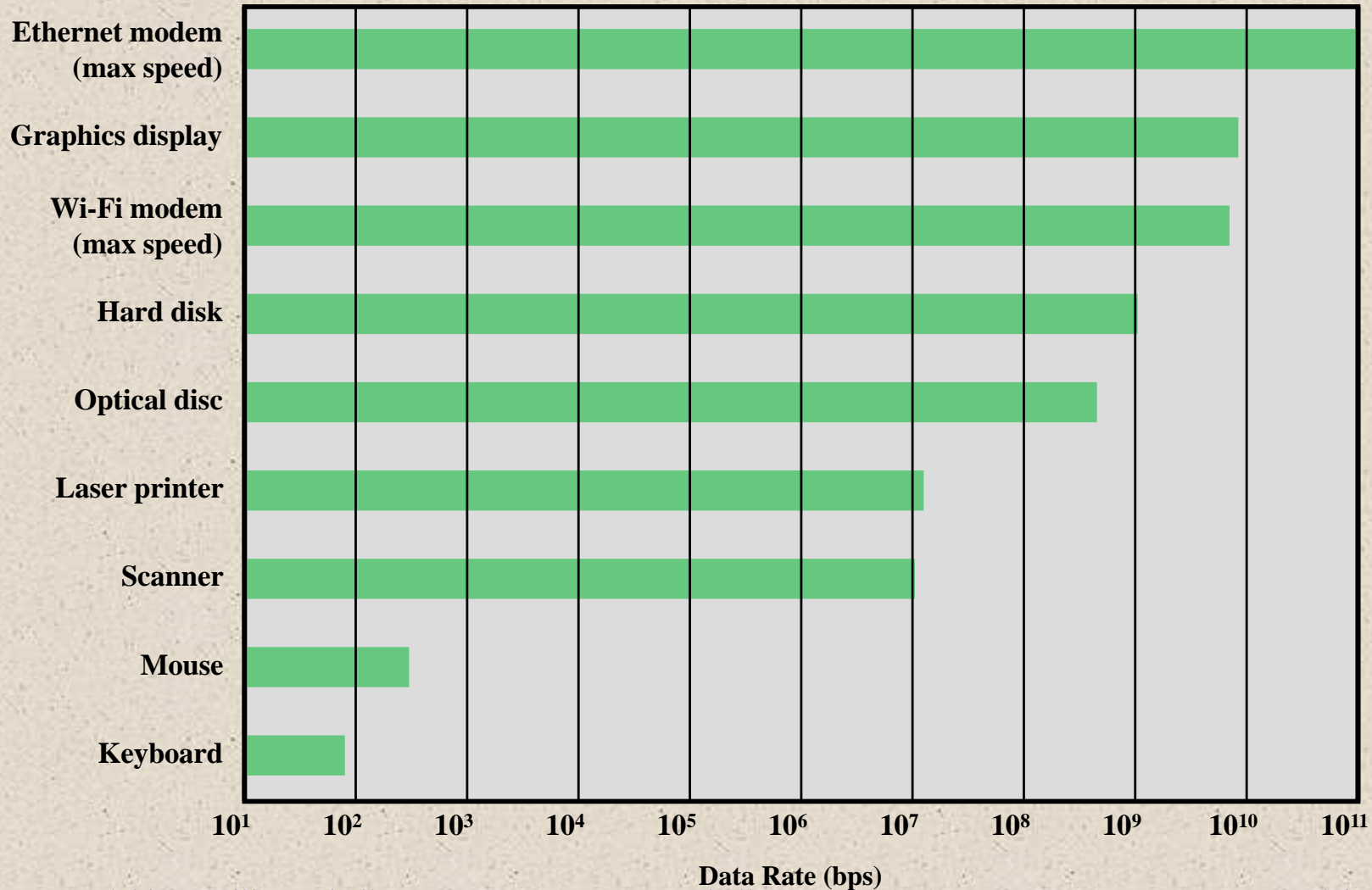


Tasarımın odaklandığı diğer bir alan, **I / O cihazlarının kullanımıyla ilgilidir**. Bilgisayarlar daha hızlı ve daha yetenekli hale geldikçe, yoğun I / O taleplerine sahip çevre birimlerinin kullanımını destekleyen daha karmaşık uygulamalar geliştirilir.



Bu cihazlar, muazzam veri işleme talepleri yaratır.

Mevcut işlemciler nesli bu cihazlar tarafından pompalanan verileri idare edebilirken, bu verilerin **işlemci ve çevre birimi arasında taşınmasını sağlama sorunu** devam etmektedir.



Buradaki stratejiler, önbelleğe alma (**caching**) ve arabelleğe alma (**buffering**) tekniklerinin yanı sıra daha yüksek hızlı ara bağlantı bus'larının ve daha ayrıntılı bus yapılarının kullanımını içerir.

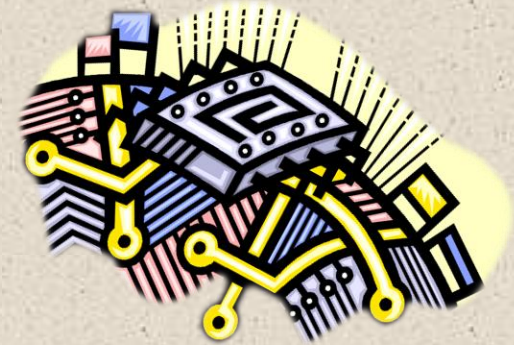
Figure 2.1 Typical I/O Device Data Rates

Şekil 2.1, kişisel bilgisayarlarda ve iş istasyonlarında kullanılan tipik çevresel aygıtların bazı örneklerini verir.

+ Improvements in Chip Organization and Architecture

Tasarımcılar işlemci performansını ana bellek ve diğer bilgisayar bileşenleriyle dengeleme zorluğuyla boğuşurken, işlemci hızını artırma ihtiyacı devam ediyor. Daha yüksek işlemci hızına ulaşmak için üç yaklaşım vardır:

- İşlemcinin donanım hızını artırma
 - Temelde küçülen lojik kapı boyutu nedeniyle
 - Daha sıkı paketlenmiş daha fazla kapı, saat hızını (*clock rate*) artırıyor
 - Sinyaller için yayılma süresi (*propagation time*) azaltıldı
- Önbelleklerin (*caches*) boyutunu ve hızını artırma
 - İşlemci yongasının bir kısmını ayırmak
 - Önbellek erişim süreleri önemli ölçüde düşer
- İşlemci organizasyonunu ve mimarisinde değişiklik
 - Efektif komut yürütme hızını artırma
 - Paralellik (*Parallelism*)



+ Problems with Clock Speed and Logic Density

Geleneksel olarak, performans kazanımlarındaki baskın faktör, saat hızındaki (*clock speed*) ve mantık yoğunluğundaki (*logic density*) artışlardır. Bununla birlikte, saat hızı ve mantık yoğunluğu arttıkça, bir dizi engel daha önemli hale gelir.

■ Power

- Güç yoğunluğu (*power density (Watts/cm²)*), lojik yoğunluğu ve saat hızı ile artar
- Yayılan ısı

Yüksek yoğunluklu, yüksek hızlı yongalarda üretilen ısıyı dağıtmanın zorluğu ciddi bir tasarım sorunu haline geliyor.

■ RC delay

- Elektronların akış hızı, onları bağlayan metal tellerin direnci ve kapasitansı ile sınırlıdır
- RC çarpımı arttıkça gecikme artar
- Çip üzerindeki bileşenlerin boyutu küçüldükçe, tel ara bağlantıları inceler ve direnci artırır
- Ayrıca, teller birbirine daha yakın hale gelmesi kapasitansı artırır

■ Memory latency

- Bellek hızları işlemci hızlarının gerisinde kalır

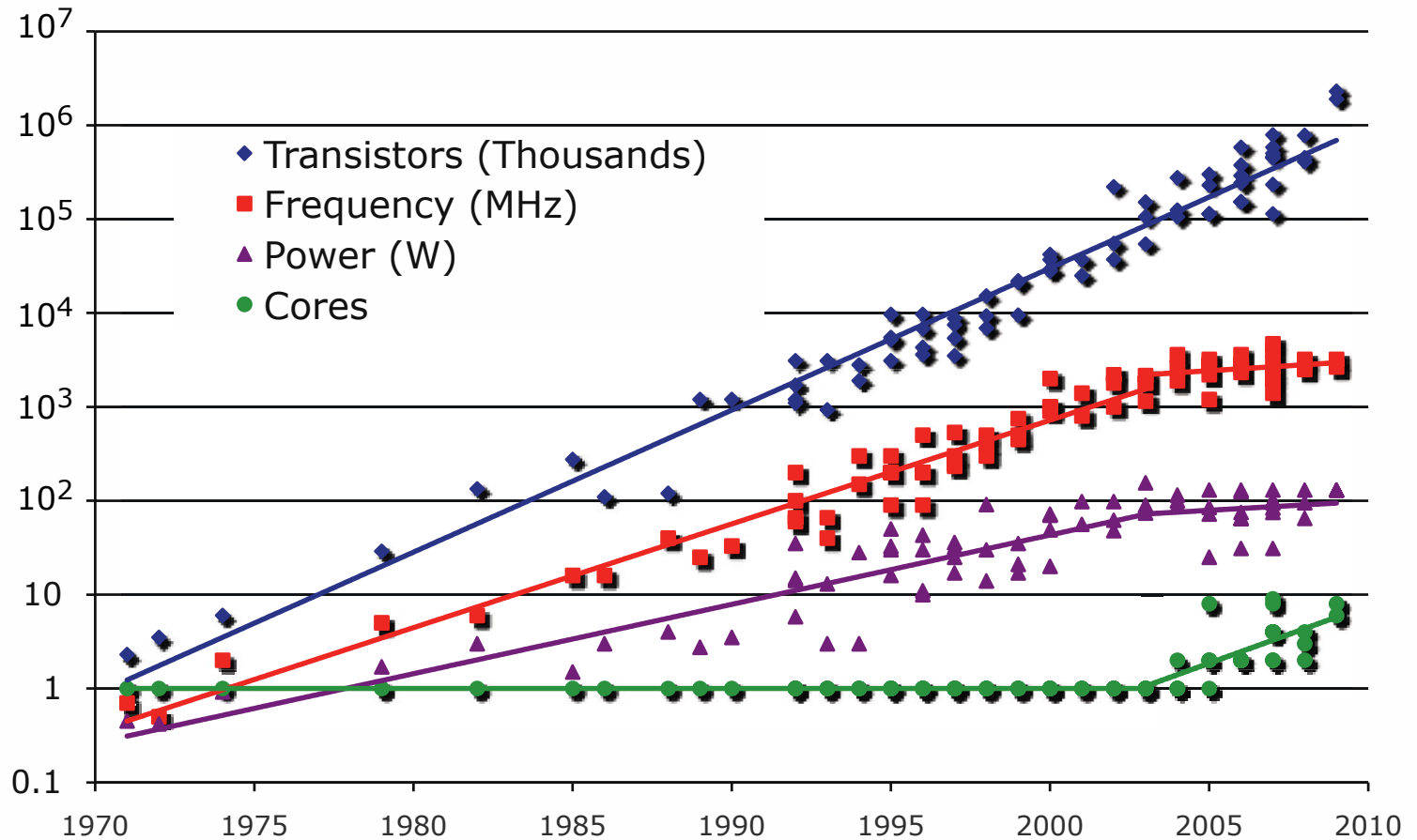


Figure 2.2 Processor Trends

1980'lerin sonlarından başlayarak ve yaklaşık 15 yıl boyunca devam eden, performansı basitçe saat hızını artırarak elde edilebileceklerin ötesinde artırmak için iki ana strateji kullanıldı.

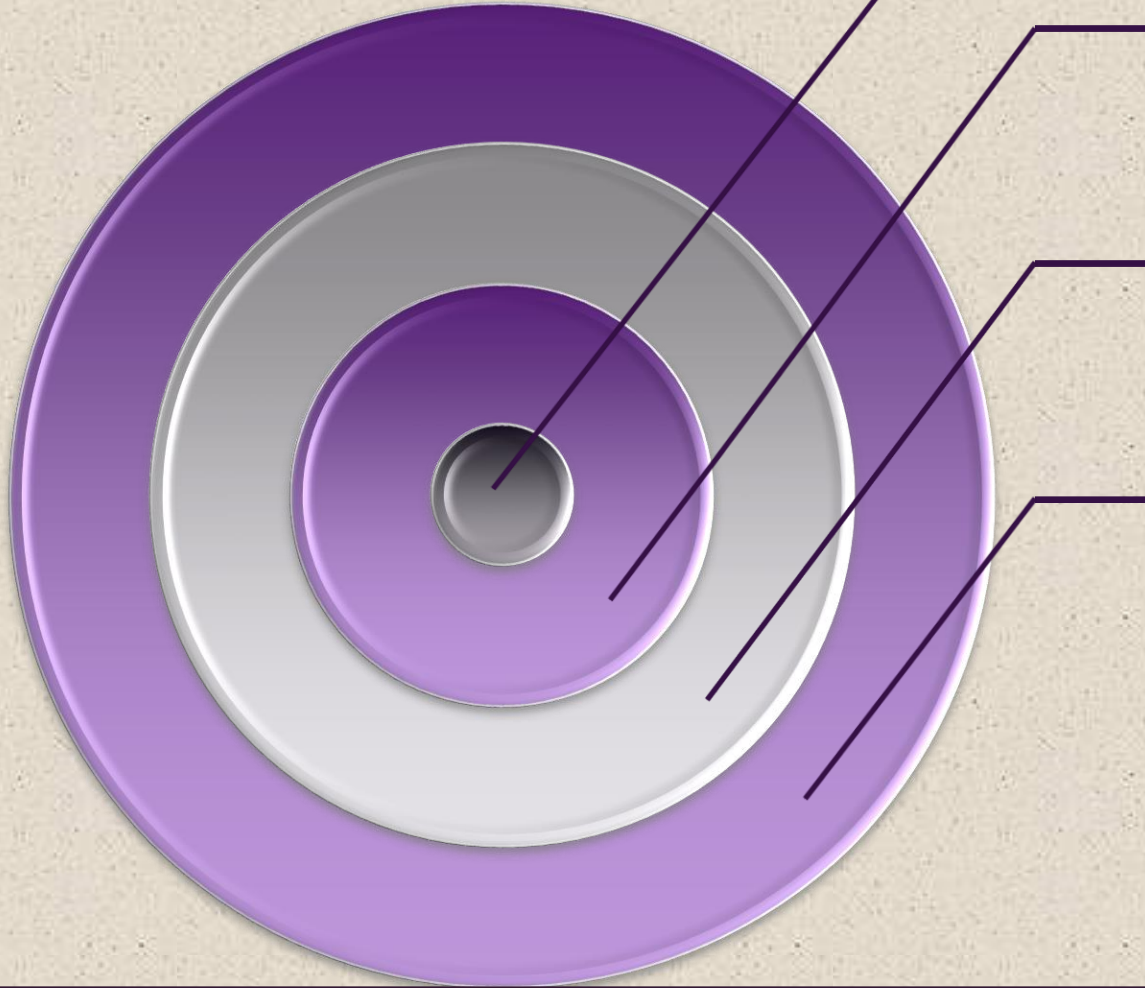
- Birincisi, önbellek kapasitesinde (*cache capacity*) bir artış oldu.
 - Artık işlemci ve ana bellek arasında tipik olarak iki veya üç seviye önbellek vardır.
 - Yonga yoğunluğu arttıkça, daha fazla önbellek yongaya dahil edilerek daha hızlı önbellek erişimi sağlandı.
 - Örneğin, orijinal Pentium yongası, yonga üzerindeki alanın yaklaşık %10'unu önbelleklere ayırdı.
 - Modern çipler, çip alanının yarısından fazlasını önbelleklere ayırır ve tipik olarak yaklaşık diğer yarısının dörtte üçü boru hattıyla ilgili kontrol ve tamponlama içindir.

- İkinci olarak, bir işlemci içindeki komut yürütme mantığı, işlemci içindeki komutların paralel olarak yürütülmesini sağlamak için giderek daha karmaşık hale gelmiştir. (***complex execution logic***)
 - Dikkate değer iki tasarım yaklaşımı, boru-hattı ve süperskalar (***pipelining and superscalar***) olmuştur.
 - Boru hattı, bir üretim tesisindeki bir montaj hattı gibi çalışır ve boru hattı boyunca aynı anda farklı komutların icrasının farklı aşamalarını mümkün kılar.
 - Esasen bir süperskalar yaklaşım, tek bir işlemci içinde birden çok işlem boruhattına izin verir, böylece birbirine bağımlı olmayan komutlar paralel olarak yürütülebilir.

- ❖ 90'ların ortasından sonuna kadar, bu yaklaşımların her ikisi de azalan bir getiri noktasına ulaşıyordu.
- ❖ Modern işlemcilerin iç organizasyonu son derece karmaşıktır ve komut akışından büyük ölçüde paralelliği sığdırabilir/sonuna kadar kullanabilir.
- ❖ Bu yöndeki diğer önemli artışların nispeten mütevazı olması muhtemel görünüyor.
- ❖ İşlemci yongasında her seviye önemli kapasite sağlayan üç seviyeli önbellek ile, önbellekten sağlanan faydaların da bir sınıra ulaştığı görülüyor.

Belirtilen tüm zorluklar göz önünde bulundurularak, tasarımcılar performansı iyileştirmek için temelde yeni bir yaklaşıma döndüler: **birden fazla işlemciyi büyük bir paylaşılan önbellek ile aynı yongaya yerleştirmek.**

Multicore



Aynı çip üzerinde birden çok işlemcinin kullanılması, saat hızını artırmadan performansı artırma potansiyeli sağlar

Strateji, çip üzerinde bir tane daha karmaşık işlemci yerine daha basit iki işlemci kullanmaktır

İki işlemciyle, daha büyük önbellekler makul olur

Önbellekler büyüdükçe, bir yonga üzerinde iki ve daha sonra üç düzeyde önbellek oluşturmak performans açısından anlamlı hale geldi



Many Integrated Core (MIC) Graphics Processing Unit (GPU)

MIC


- Bu kadar çok sayıda çekirdekten yararlanmak için yazılım geliştirmedeki zorlukların yanı sıra performansta sıçrama
- Çok çekirdekli ve MIC stratejisi, tek bir çip üzerinde genel amaçlı işlemcilerin homojen bir koleksiyonunu içerir

GPU

- Grafik verileri üzerinde paralel işlemler gerçekleştirmek için tasarlanmış çekirdek
- Geleneksel olarak bir eklenti grafik kartında bulunur, 2D ve 3D grafikleri kodlamak ve işlemek için ve ayrıca video işlemek için kullanılır
- Tekrarlayan hesaplamalar gerektiren çeşitli uygulamalar için vektör işlemcileri (*vector processors*) olarak kullanılır

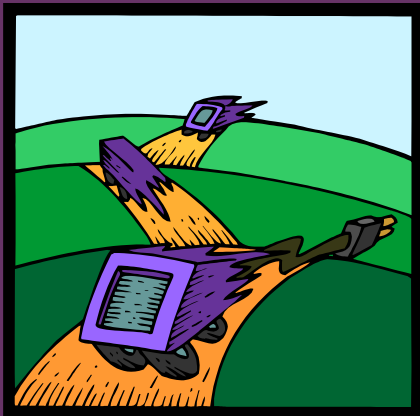
Çip üreticileri, yonga başına 50'den fazla çekirdekle yonga başına çekirdek sayısında büyük bir sıçrama yapma sürecindedir. Performanstaki sıçrama ve bu kadar çok sayıda çekirdeği kullanmak için yazılım geliştirmedeki zorluklar, yeni bir terimin ortaya çıkmasına neden oldu: **many integrated core (MIC)**.

Aynı zamanda, çip üreticileri başka bir tasarım seçeneğinin peşinde koşuyor: birden çok genel amaçlı işlemciye sahip bir yonga artı grafik işleme birimleri (GPU'lar) ve video işleme için özel çekirdekler ve diğer görevler. Geniş anlamda, **GPU**, grafik verileri üzerinde paralel işlemler gerçekleştirmek için tasarlanmış bir çekirdektir.

- 
- ❖ GPU'lar birden çok veri kümesi üzerinde paralel işlemler gerçekleştirdiğinden, tekrarlayan hesaplamalar gerektiren çeşitli uygulamalar için vektör işlemcileri (*vector processors*) olarak giderek daha fazla kullanılmaktadır.
 - ❖ Bu, GPU ile CPU arasındaki çizgiyi bulanıklaştırır. Böyle bir işlemci tarafından geniş bir uygulama yelpazesi desteklendiğinde, GPU'lar üzerinde genel amaçlı hesaplama **general-purpose computing on GPUs (GPGPU)** terimi kullanılır.

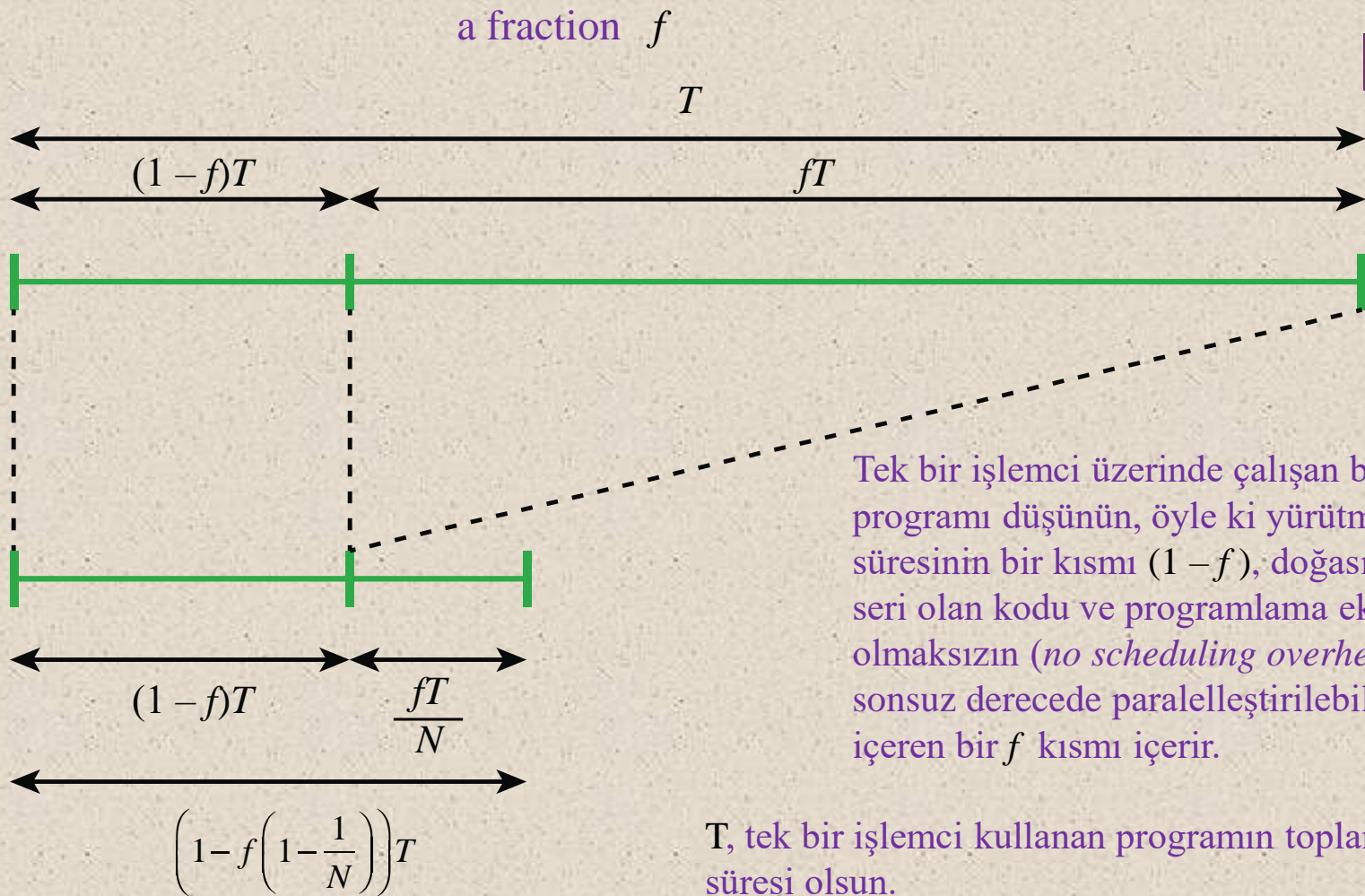


Amdahl's Law



- Gene Amdahl
- Tek bir işlemciye kıyasla birden çok işlemci kullanan bir programın potansiyel hızlanma sürecini ele alır
- Çok çekirdekli makinelerin geliştirilmesinde endüstrinin karşılaştığı sorunları tasvir eder
 - Paralel işlemenin gücünden yararlanmak için yazılım oldukça paralel bir yürütme ortamına uyarlanmalıdır.
- Bir bilgisayar sistemindeki teknik iyileştirmeyi değerlendirmek ve tasarlamak için genelleştirilebilir

- Bilgisayar sistemi tasarımcıları, teknolojideki ilerlemeler veya tasarımdaki değişikliklerle sistem performansını iyileştirmenin yollarını ararlar.
- Örnekler arasında paralel işlemcilerin kullanımı, bir bellek-önbellek hiyerarşisinin kullanılması ve teknolojik gelişmeler nedeniyle bellek erişim süresinde ve I/O aktarım hızında hızlanma yer alır.
- Tüm bu durumlarda, teknolojinin veya tasarımın bir yönündeki hızlanmanın performansta buna birebir karşılık gelen bir iyileşme ile sonuçlanmadığına dikkat etmek önemlidir.
- Bu sınırlama, Amdahl yasası tarafından kısa ve öz bir şekilde ifade edilmiştir.



T , tek bir işlemci kullanan programın toplam yürütme süresi olsun.
O zaman, programın paralel kısmını tamamen kullanan N işlemcili bir paralel işlemci kullanarak hızlandırma aşağıdaki gibidir:

Figure 2.3 Illustration of Amdahl's Law

$$= \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$$

Speedup =

Time to execute program on a single processor
Time to execute program on N parallel processors

Speedup =

Time to execute program on a single processor

Time to execute program on N parallel processors

$$= \frac{T(1 - f) + Tf}{T(1 - f) + \frac{Tf}{N}} = \frac{1}{(1 - f) + \frac{f}{N}}$$

- İki önemli sonuç çıkarılabilir:
 1. f küçük olduğunda, paralel işlemcilerin kullanımının çok az etkisi vardır.
 2. N sonsuza yaklaştıkça, hızlanma $1 / (1 - f)$ ile sınırlıdır, böylece daha fazla işlemci kullanmanın getirileri azalır.

Speedup =

Time to execute program on a single processor

Time to execute program on N parallel processors

$$= \frac{T(1 - f) + Tf}{T(1 - f) + \frac{Tf}{N}} = \frac{1}{(1 - f) + \frac{f}{N}}$$

- ❖ Örneğin, bir görevin kayan nokta işlemlerinden (*floating-point operations*) kapsamlı bir şekilde yararlandığını ve zamanın % 40'ının kayan nokta işlemleri tarafından tüketildiğini varsayalım. Yeni bir donanım tasarımıyla paralel işleme sayesinde, kayan nokta modülü K katsayısı kadar hızlandırılsın. Daha sonra genel hızlandırma:

$$\text{Speedup} = \frac{1}{0.6 + \frac{0.4}{K}}$$

Böylece, K 'dan bağımsız olarak maksimum hız artışı 1.67'dir.

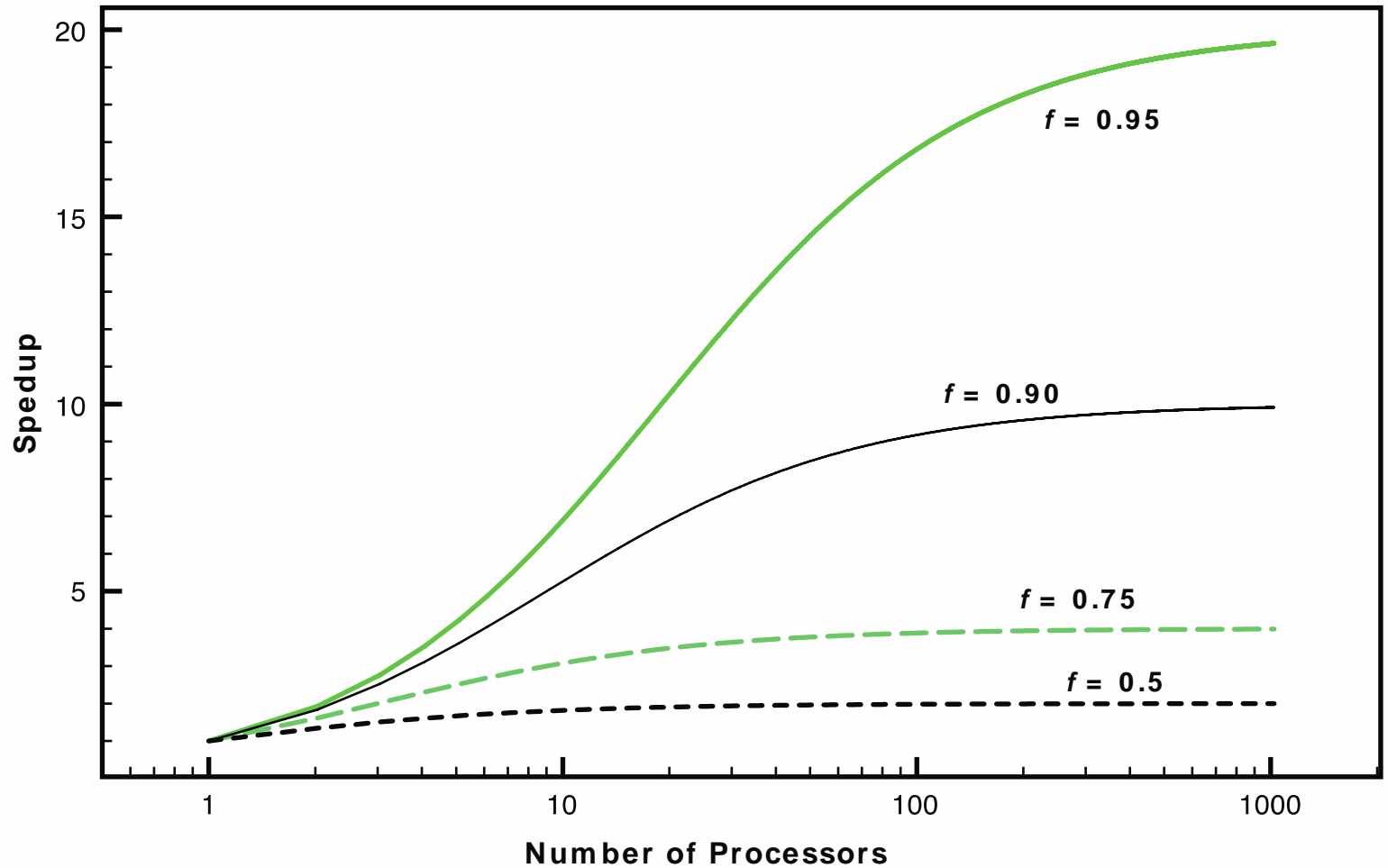


Figure 2.4 Amdahl's Law for Multiprocessors



Little's Law

is a theorem by John Little which states that the long-term average number L of customers in a stationary system is equal to the long-term average effective arrival rate λ multiplied by the average time W that a customer spends in the system

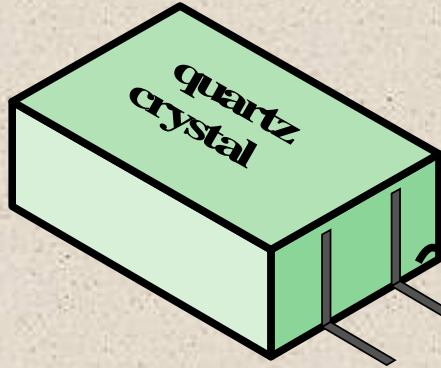
$$L = \lambda W$$

- Geniş uygulamalarla temel ve basit ilişki
- İstatistiksel olarak kararlı durumda (*steady state*) olan ve kaçağın olmadığı (*no leakage*) hemen hemen her sisteme uygulanabilir.
- Queuing system
 - Sunucu boşdaysa, bir öge hemen hizmet alır, aksi takdirde gelen bir öge bir kuyruğa katılır
 - Tek bir sunucu veya birden çok sunucu için tek bir kuyruk olabilir veya birden çok sunucunun her biri için bir tane olmak üzere birden çok kuyruk olabilir.
- Bir kuyruk sistemindeki ortalama öge sayısı, ögelerin ortalama geliş hızının (*average rate at which items arrive*), bir ögenin sistemde geçirdiği süre (*time that an item spends in the system*) ile çarpımına eşittir.
 - İlişki çok az varsayım gerektirir
 - Sadeliği ve genelliği nedeniyle son derece kullanışlıdır

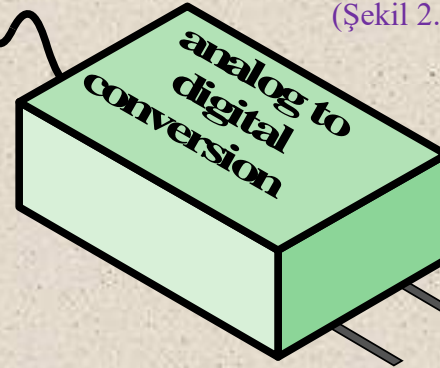
Servis süresi dağılımının ne olduğunu bilmemize gerek yok, varış zamanlarının dağılımı veya ögelerin servis sırası veya önceliği...

Sadeliği ve genelliği nedeniyle, Little Yasası son derece kullanışlıdır ve çok çekirdekli bilgisayarlarla ilgili performans sorunlarına olan ilgi nedeniyle biraz canlanma yaşadı.

İşlemci tarafından bir komutun getirilmesi (*fetch*), komutun kodunun çözülmesi (*decode*), aritmetik bir işlemin gerçekleştirilmesi gibi işlemler, bir sistem saati (*system clock*) tarafından yönetilir. Tipik olarak, tüm işlemler saat darbesiyle başlar. Bu nedenle, en temel düzeyde, bir işlemcinin hızı, saat tarafından üretilen, saniye başına döngü olarak ölçülen darbe frekansı veya Hertz (Hz) tarafından belirlenir.



Tipik olarak saat sinyalleri, güç uygulanırken sabit bir sinüs dalgası oluşturan bir **kuvars kristali** tarafından üretilir. Bu dalga, işlemci devresine sabit bir akışla sağlanan bir dijital voltaj darbe akışına dönüştürülür (Şekil 2.5).



Saat hızı rastgele değildir, işlemcinin fiziksel düzenine uygun olmalıdır. İşlemci içindeki eylemler, sinyallerin bir işlemci ögesinden diğerine gönderilmesini gerektirir.

From Computer Desktop Encyclopedia
1998, The Computer Language Co.

Örneğin, 1 GHz'lik bir işlemci saniyede 1 milyar darbe alır. Darbe hızı, saat hızı (**clock rate, or clock speed**) olarak bilinir. Saatin bir artışı veya darbesi, bir saat döngüsü (clock cycle) veya bir saat tıklaması (clock tick) olarak adlandırılır. Darbeler arasındaki süre döngü süresidir (**cycle time**).

Figure 2.5 System Clock

Bir komutun yürütülmesi, komutun bellekten alınması, komutun çeşitli bölümlerinin kodunun çözülmesi, verilerin yüklenmesi ve depolanması ve aritmetik ve mantıksal işlemlerin gerçekleştirilmesi gibi bir dizi ayrık adımı içerir. Bu nedenle, çoğu işlemci içindeki **çoğu komutun tamamlanması için birden fazla saat döngüsü gerekir**.

Tablo 2.1, bir boyutun beş performans faktörünü ve diğer boyutun dört sistem özelliğini gösterdiği bir matristir. Hücredeki X, bir performans faktörünü etkileyen bir sistem özelliğini gösterir.

(İcra edilen programdaki)
Komut sayısı

p , komutu
çözmek ve
yürütmek
için gereken
işlemci
döngüsü
sayısıdır

m , gereken
bellek
referanslarını
 n sayısıdır

k , bellek
döngü süresi
ile işlemci
döngü süresi
arasındaki
orandır

Bir işlemci,
sabit frekansta
 f veya eşdeğer
olarak sabit
döngü süresine
sahip bir saat
tarafından
çalıştırılır.

$$\tau = 1/f$$

	I_c	p	m	k	t
Instruction set architecture	X	X			
Compiler technology	X	X	X		
Processor implementation		X			X
Cache and memory hierarchy				X	X

compiler technology (how effective the compiler is in producing an efficient machine language program from a high-level language program),

Table 2.1 Performance Factors and System Attributes

INSTRUCTION EXECUTION RATE (KOMUT YÜRÜTME/İCRA HIZI)

- Bir işlemci, sabit frekansı f veya eşdeğer olarak sabit bir döngü süresi τ ye sahip bir saat tarafından çalıştırılır, burada $\tau = 1/f$.
- Bir program için komut sayısını, I_C , bitene kadar veya belirli bir zaman aralığı için çalıştırılana kadar o program için yürütülen makine komutlarının sayısı olarak tanımlanır.
 - Bunun, programın amaç kodundaki (*object code*) komut sayısı değil, komut yürütme sayısı olduğuna dikkat edin
- Önemli bir parametre, bir program için komut başına ortalama döngüdür (*average cycles per instruction-CPI*). Tüm komutlar aynı sayıda saat döngüsü gerektiriyor olsaydı, *CPI* bir işlemci için sabit bir değer olacaktı.
- Fakat, herhangi bir işlemcide, gereken saat döngüsü sayısı, yükleme (load), depolama (store), dallanma (branch) vb. gibi farklı komut türleri için değişir.

INSTRUCTION EXECUTION RATE (KOMUT YÜRÜTME/İCRA HIZI)

- CPI_i , komut tipi i için gerekli döngü sayısı olsun. ve I_i , belirli bir program için çalıştırılan i tipi komutların sayısıdır. O zaman aşağıdaki gibi genel bir CPI hesaplayabiliriz:

$$CPI = \frac{\sum_{i=1}^n (CPI_i \times I_i)}{I_c}$$

Verilen bir programı yürütmek için gereken işlemci süresi T şu şekilde ifade edilebilir:

$$T = I_c \times CPI \times \tau$$

Bu formülasyonu, bir komutun yürütülmesi sırasında işin bir kısmının işlemci tarafından yapıldığını ve zamanın bir kısımının kelime transferi için belleğe erişim ile geçtiğini kabul ederek geliştirebiliriz. Bu ikinci durumda, transfer süresi, işlemci döngü süresinden daha büyük olabilecek bellek döngü süresine bağlıdır.

$$T = I_c \times [p + (m \times k)] \times \tau$$

INSTRUCTION EXECUTION RATE (KOMUT YÜRÜTME/İCRA HIZI)

- MIPS hızı/oranı (**MIPS rate**)
 - Bir işlemci için genel bir performans ölçüsü olarak adlandırılır
 - Saniyede milyonlarca komut (MIPS) olarak ifade edilen, komutların yürütüldüğü hızdır.
- MIPS hızını, saat hızı (*clock rate*) ve *CPI* cinsinden şu şekilde ifade edebiliriz:

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

INSTRUCTION EXECUTION RATE (KOMUT YÜRÜTME/İCRA HIZI)

- Örneğin, 400 MHz'lik bir işlemcide 2 milyon komutun yürütülmesiyle sonuçlanan bir programın yürütülmesini düşünün. Program, dört ana komut türünden (*Instruction type*) oluşur. Her komut türü için komut karışımı ve *CPI*, bir program izleme deneyinin sonucuna göre aşağıda verilmiştir:

Instruction Type	CPI	Instruction Mix
Arithmetic and logic	1	60%
Load/store with cache hit	2	18%
Branch	4	12%
Memory reference with cache miss	8	10%

Program, yukarıdaki izleme sonuçlarıyla tek işlemcide yürütüldüğünde ortalama *CPI*,

$$CPI = 0.6 + (2 \times 0.18) + (4 \times 0.12) + (8 \times 0.1) = 2.24$$

Karşılık gelen MIPS hızı

$$(400 \times 10^6) / (2.24 \times 10^6) \approx 178$$


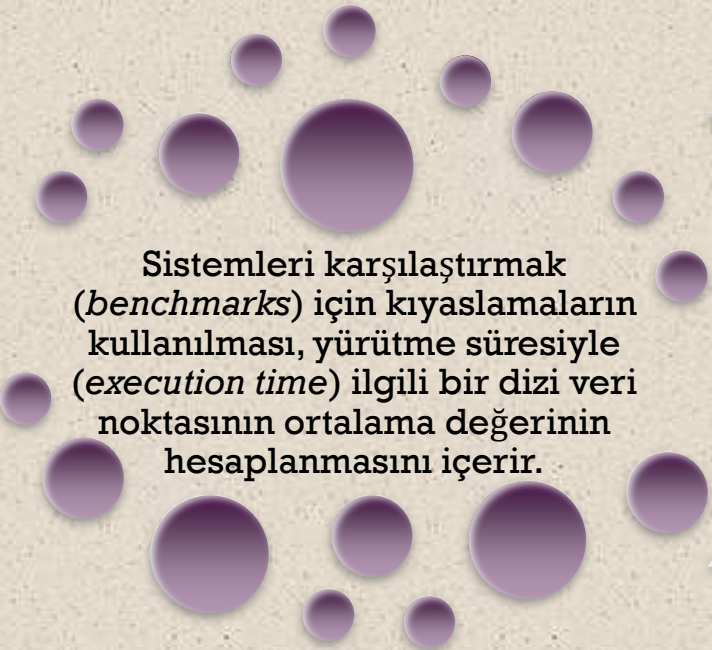
$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$$

INSTRUCTION EXECUTION RATE (KOMUT YÜRÜTME/İCRA HIZI)



- Diğer bir yaygın performans ölçüsü, yalnızca kayan nokta komutlarıyla ilgilenir.
 - Bunlar birçok bilimsel ve oyun uygulamasında yaygındır. Kayan nokta performansı, saniyede milyonlarca kayan nokta işlemi (millions of floating-point operations per second-**MFLOPS**) olarak ifade edilir.

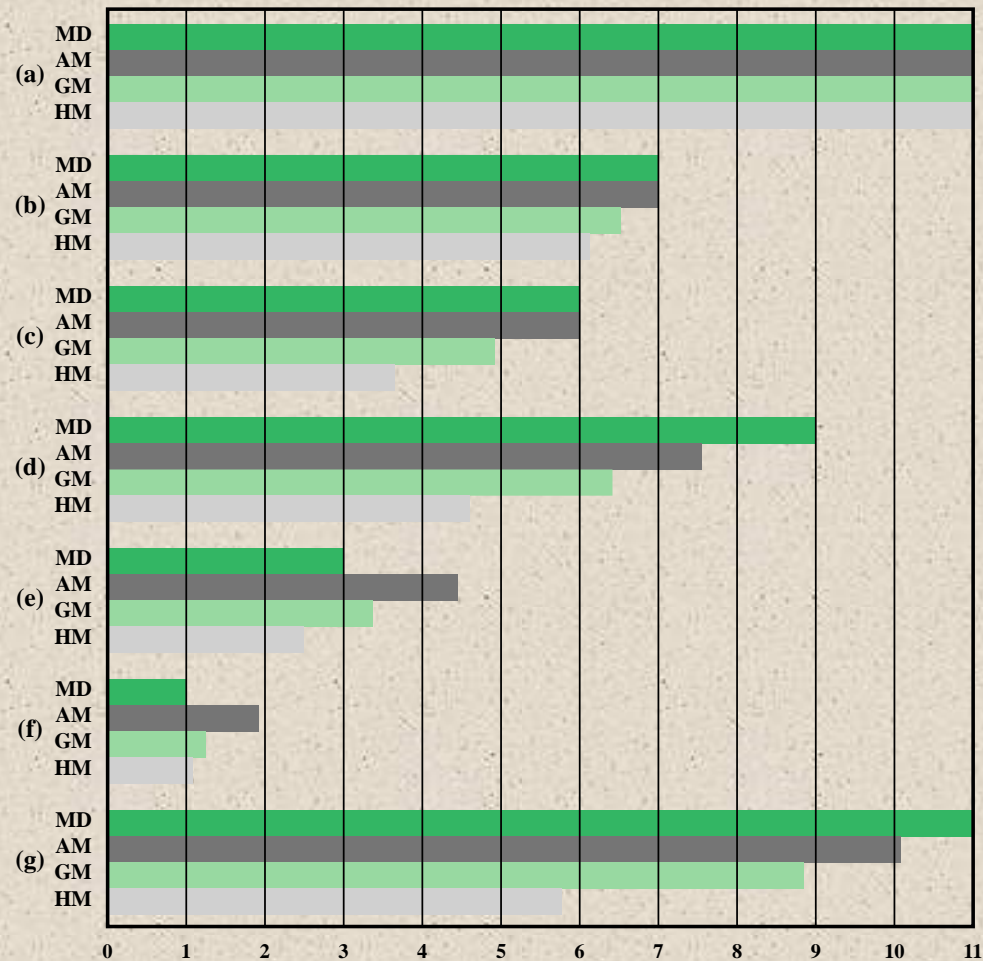
Calculating the Mean



Sistemleri karşılaştırmak (*benchmarks*) için kıyaslamaların kullanılması, yürütme süresiyle (*execution time*) ilgili bir dizi veri noktasının ortalama değerinin hesaplanmasını içerir.

The three common formulas used for calculating a mean are:

- Arithmetic
- Geometric
- Harmonic



- (a) Constant (11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11)
- (b) Clustered around a central value (3, 5, 6, 6, 7, 7, 8, 8, 9, 1 1)
- (c) Uniform distribution (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1 1)
- (d) Large-number bias (1, 4, 4, 7, 7, 9, 9, 10, 10, 1 1, 11)
- (e) Small-number bias (1, 1, 2, 2, 3, 3, 5, 5, 8, 8, 1 1)
- (f) Upper outlier (11, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
- (g) Lower outlier (1, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11)

MD = median
 AM = arithmetic mean
 GM = geometric mean
 HM = harmonic mean

**Figure 2.6 Comparison of Means on Various Data Sets
 (each set has a maximum data point value of 11)**

- Arithmetic Mean (AM)-Aritmetik Ortalama, tüm ölçümlerin toplamı anlamlı ve ilgi çekici bir değer ise uygun bir ölçüdür
- Aritmetik Ortalama, birkaç sistemin yürütme süresi performansını karşılaştırmak için iyi bir adaydır

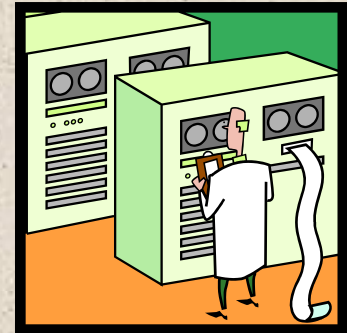
Arithmetic

Mean

Örneğin, bir sistem kullanmakla ilgilendiğimizi varsayalım. Büyük ölçekli simülasyon çalışmaları için ve birkaç alternatif ürünü değerlendirmek isteyelim. Her sistemde simülasyonu her çalıştırma için farklı girdi değerleriyle birden çok kez çalıştırabilir ve ardından tüm çalıştırmalarda ortalama yürütme süresini alabiliriz. Farklı girdilerle birden fazla çalışmanın kullanılması, sonuçların belirli bir girdi kümesinin bazı olağandışı özelliklerinden büyük ölçüde etkilenmemesini sağlamalıdır. Tüm çalışmaların Aritmetik Ortalaması, sistemin simülasyonlardaki performansının iyi bir ölçüsüdür ve sistem karşılaştırması için kullanılacak iyi bir ölçüdür.

+

- Program yürütme süresi gibi zamana dayalı bir değişken için kullanılan Aritmetik Ortalama, toplam süre ile doğru orantılı olması gibi önemli özelliğe sahiptir.
 - Toplam süre iki katına çıkarsa, ortalama değer iki katına çıkar



	Computer A time (secs)	Computer B time (secs)	Computer C time (secs)	Computer A rate (MFLOPS)	Computer B rate (MFLOPS)	Computer C rate (MFLOPS)
Program 1 (10^8 FP ops)	2.0	1.0	0.75	50	100	133.33
Program 2 (10^8 FP ops)	0.75	2.0	4.0	133.33	50	25
Total execution time	2.75	3.0	4.75			
Arithmetic mean of times	1.38	1.5	2.38			
Inverse of total execution time (1/sec)	0.36	0.33	0.21			
Arithmetic mean of rates				91.67	75.00	79.17
Harmonic mean of rates				72.72	66.67	42.11

Tablonun sağ yarısı, MFLOPS cinsinden ifade edilen hızlar (rates) açısından bir karşılaştırma sağlar. Hız hesaplaması basittir. Örneğin, Program 1, 100 milyon kayan nokta işlemi yürütür. Computer A'nın bu programı yürütmesi 2 saniye sürer, yani $100/2 = 50$ MFLOPS hızındadır.

Table 2.2

A Comparison of Arithmetic and Harmonic Means for Rates

HM değerlerine baktığımızda bilgisayarların hız sıralamasını doğru yansıttığını görüyoruz. Bu, oranları hesaplarken HM'nin tercih edildiğini doğrular.

Basit olması için, her programın icrasının 10^8 kayan nokta işlemlerinin yürütülmesiyle sonuçlandığını varsayıyoruz.

Table 2.3 A Comparison of Arithmetic and Geometric Means for Normalized Results



(a) Results normalized to Computer A

	Computer A time	Computer B time	Computer C time
Program 1	2.0 (1.0)	1.0 (0.5)	0.75 (0.38)
Program 2	0.75 (1.0)	2.0 (2.67)	4.0 (5.33)
Total execution time	2.75	3.0	4.75
Arithmetic mean of normalized times	1.00	1.58	2.85
Geometric mean of normalized times	1.00	1.15	1.41

Tablo 2.3a'da, tüm sonuçlar Bilgisayar A'ya normalize edilmiştir ve ortalamalar normalleştirilmiş değerler üzerinden hesaplanmıştır. Toplam yürütme süresine bağlı olarak, A, B'den daha hızlıdır, B de C'den. Normalleştirilmiş zamanların hem AM'leri hem de GM'leri bunu yansıtır.

(b) Results normalized to Computer B

	Computer A time	Computer B time	Computer C time
Program 1	2.0 (2.0)	1.0 (1.0)	0.75 (0.75)
Program 2	0.75 (0.38)	2.0 (1.0)	4.0 (2.0)
Total execution time	2.75	3.0	4.75
Arithmetic mean of normalized times	1.19	1.00	1.38
Geometric mean of normalized times	0.87	1.00	1.22

Tablo 2.3b'de sistemler artık B'ye normalize edilmiştir. Yine GM'ler üç bilgisayarın göreceli hızlarını (*relative speeds*) doğru bir şekilde yansıtır, ancak şimdi AM'ler farklı bir sıralama üretmektedir.

Table 2.4 Another Comparison of Arithmetic and Geometric Means for Normalized Results

(a) Results normalized to Computer A

	Computer A time	Computer B time	Computer C time
Program 1	2.0 (1.0)	1.0 (0.5)	0.20 (0.1)
Program 2	0.4 (1.0)	2.0 (5.0)	4.0 (10)
Total execution time	2.4	3.00	4.2
Arithmetic mean of normalized times	1.00	2.75	5.05
Geometric mean of normalized times	1.00	1.58	1.00

(b) Results normalized to Computer B

	Computer A time	Computer B time	Computer C time
Program 1	2.0 (2.0)	1.0 (1.0)	0.20 (0.2)
Program 2	0.4 (0.2)	2.0 (1.0)	4.0 (2)
Total execution time	2.4	3.00	4.2
Arithmetic mean of normalized times	1.10	1.00	1.10
Geometric mean of normalized times	0.63	1.00	0.63

Ne yazık ki, tutarlılık her zaman doğru sonuçlar vermez.

Tablo 2.4'te bazı yürütme süreleri değiştirilmiştir. Bir kez daha, AM iki normalleştirme için çelişkili sonuçlar bildirir.

GM tutarlı sonuçlar bildirir, ancak sonuç, B'nin eşit olan A ve C'den daha hızlı olmasıdır.



SPEC: System Performance Evaluation Corporation

- SPEC, birkaç nedenden ötürü GM'yi kullanmayı seçmiştir:
 1. Belirtildiği gibi, GM hangi sistemin referans olarak kullanıldığına bakılmaksızın tutarlı sonuçlar verir. «benchmarking» öncelikle bir karşılaştırma analizi olduğu için bu önemli bir özelliktir.
 2. SPEC analistleri tarafından yapılan sonraki analizlerde onaylandığı üzere, GM, aykırı değerler (outliers) itibarıyla HM veya AM'den daha az önyargılıdır.
 3. Normalize sayıların genel olarak çarpık dağılımı nedeniyle performans oranlarının dağılımlarının lognormal dağılımlarla normal olanlara göre daha iyi modellendiğini göstermektedir.



The need for benchmarking

- MIPS ve MFLOPS gibi ölçülerin, işlemcilerin performansını değerlendirmek için yetersiz olduğu kanıtlanmıştır. Komut setlerindeki farklılıklar nedeniyle, komut yürütme hızı (*instruction execution rate*), farklı mimarilerin performansını karşılaştırmak için geçerli bir araç değildir.
- Dikkate alınması gereken bir diğer husus, belirli bir işlemcinin belirli bir programdaki performansının, bu işlemcinin çok farklı bir uygulama türünde nasıl performans göstereceğini belirlemede yararlı olmayabileceğidir.
 - Buna göre, 1980'lerin sonlarından ve 1990'ların başlarından başlayarak, endüstri ve akademik ilgi, bir dizi kıyaslama programı (*benchmark programs*) kullanarak sistemlerin performansını ölçmeye kaydı.
- Aynı program seti farklı makinelerde çalıştırılabilir ve yürütme süreleri karşılaştırılabilir. Karşılaştırmalar, hangi sistemi satın alacaklarına karar vermeye çalışan müşterilere rehberlik sağlar ve kıyaslama hedeflerini karşılamak için sistemleri nasıl tasarlayacaklarını belirlemede satıcılar ve tasarımcılar için yararlı olabilir.



+ Benchmark Principles

- Bir «*benchmark*» programının arzu edilen özellikleri:
 1. Yüksek seviyeli bir dilde yazılmıştır ve bu da farklı makinelerde çalıştırılabilir (*portable*) olmasını sağlar
 2. Sistem programlama, sayısal programlama veya ticari programlama gibi belirli bir programlama alanı veya paradigmasının temsilcisidir.
 3. Kolayca ölçülebilir
 4. Geniş dağıtıma (*distribution*) sahiptir





System Performance Evaluation Corporation (SPEC)



Endüstride, akademik ve araştırma topluluklarında genel kabul görmüş bilgisayar performansı ölçümlerine duyulan ortak ihtiyaç, standartlaştırılmış kıyaslama takımlarının geliştirilmesine yol açmıştır.

■ Benchmark suite

- Yüksek seviyel dilde tanımlanmış bir program koleksiyonu
- Birlikte, belirli bir uygulama veya sistem programlama alanında bir bilgisayarın temsili bir testini sağlamaya çalışır

■ SPEC

- Bir endüstri konsorsiyumu
- Bilgisayar sistemlerini değerlendirmeyi amaçlayan en iyi bilinen benchmark takımları koleksiyonunu tanımlar ve devamlılığını sağlar
- Performans ölçümleri, karşılaştırma ve araştırma amacıyla yaygın olarak kullanılmaktadır.



SPEC CPU2006



- En iyi bilinen SPEC benchmark süite paketi
- İşlemci yoğun uygulamalar için endüstri standardı paket
- Zamanının çoğunu I / O yerine hesaplama yaparak geçiren uygulamaların performansını ölçmek için uygundur
- C, C ++ ve Fortran ile yazılmış 17 kayan nokta programı ile C ve C ++ ile yazılmış 12 tamsayı programından oluşur
- Suite, 3 milyondan fazla satır kod içerir
- SPEC'den beşinci nesil işlemci yoğun paketleri



Table 2.5

SPEC CPU2006 Integer Benchmarks

Benchmark	Reference time (hours)	Instr count (billion)	Language	Application Area	Brief Description
400.perlbench	2.71	2,378	C	Programming Language	PERL programming language interpreter, applied to a set of three programs.
401.bzip2	2.68	2,472	C	Compression	General-purpose data compression with most work done in memory, rather than doing I/O.
403.gcc	2.24	1,064	C	C Compiler	Based on gcc Version 3.2, generates code for Opteron.
429.mcf	2.53	327	C	Combinatorial Optimization	Vehicle scheduling algorithm.
445.gobmk	2.91	1,603	C	Artificial Intelligence	Plays the game of Go, a simply described but deeply complex game.
456.hmmer	2.59	3,363	C	Search Gene Sequence	Protein sequence analysis using profile hidden Markov models.
458.sjeng	3.36	2,383	C	Artificial Intelligence	A highly ranked chess program that also plays several chess variants.
462.libquantum	5.76	3,555	C	Physics / Quantum Computing	Simulates a quantum computer, running Shor's polynomial-time factorization algorithm.
464.h264ref	6.15	3,731	C	Video Compression	H.264/AVC (Advanced Video Coding) Video compression.
471.omnetpp	1.74	687	C++	Discrete Event Simulation	Uses the OMNet++ discrete event simulator to model a large Ethernet campus network.
473.astar	1.95	1,200	C++	Path-finding Algorithms	Pathfinding library for 2D maps.
483.xalancbmk	1.92	1,184	C++	XML Processing	A modified version of Xalan-C++, which transforms XML documents to other document types.

Benchmark	Reference time (hours)	Instr count (billion)	Language	Application Area	Brief Description
410.bwaves	3.78	1,176	Fortran	Fluid Dynamics	Computes 3D transonic transient laminar viscous flow.
416.gamess	5.44	5,189	Fortran	Quantum Chemistry	Quantum chemical computations.
433.milc	2.55	937	C	Physics / Quantum Chromodynamics	Simulates behavior of quarks and gluons
434.zeusmp	2.53	1,566	Fortran	Physics / CFD	Computational fluid dynamics simulation of astrophysical phenomena.
435.gromacs	1.98	1,958	C, Fortran	Biochemistry / Molecular Dynamics	Simulate Newtonian equations of motion for hundreds to millions of particles.
436.cactusADM	3.32	1,376	C, Fortran	Physics / General Relativity	Solves the Einstein evolution equations.
437.leslie3d	2.61	1,273	Fortran	Fluid Dynamics	Model fuel injection flows.
444.namd	2.23	2,483	C++	Biology / Molecular Dynamics	Simulates large biomolecular systems.
447.dealII	3.18	2,323	C++	Finite Element Analysis	Program library targeted at adaptive finite elements and error estimation.
450.soplex	2.32	703	C++	Linear Programming, Optimization	Test cases include railroad planning and military airlift models.
453.povray	1.48	940	C++	Image Ray-tracing	3D Image rendering.
454.calculix	2.29	3,041	C, Fortran	Structural Mechanics	Finite element code for linear and nonlinear 3D structural applications.
459.GemsFDTD	2.95	1,320	Fortran	Computational Electromagnetics	Solves the Maxwell equations in 3D.
465.tonto	2.73	2,392	Fortran	Quantum Chemistry	Quantum chemistry package, adapted for crystallographic tasks.
470.lbm	3.82	1,500	C	Fluid Dynamics	Simulates incompressible fluids in 3D.
481.wrf	3.10	1,684	C, Fortran	Weather	Weather forecasting model
482.sphinx3	5.41	2,472	C	Speech recognition	Speech recognition software.

Table 2.6

SPEC CPU2006 Floating-Point Benchmarks

(Table can be found on page 70 in the textbook.)

+ Terms Used in SPEC Documentation

- Benchmark
 - Derleyici yürüten herhangi bir bilgisayarda derlenebilen ve çalıştırılabilen yüksek seviyeli bir dilde yazılmış bir program
- Test edilen sistem (*System under test*)
 - Değerlendirilecek sistem
- Referans makine
 - Bu, SPEC tarafından tüm kıyaslamalar (*all benchmarks*) için bir temel performans oluşturmak için kullanılan bir sistemdir.
 - Her kıyaslama, bu kıyaslama için bir referans zaman oluşturmak üzere bu makinede çalıştırılır ve ölçülür.
- Baz metrik (*Base metric*)
 - Bunlar, bildirilen tüm sonuçlar için gereklidir ve derleme için katı yönergelere sahiptir
- Peak metric
 - Bu, kullanıcıların derleyici çıktısını optimize ederek sistem performansını optimize etmeyi denemesini sağlar.
- Speed metric
 - Bu, derlenmiş bir benchmark yapmak için geçen sürenin bir ölçüsüdür.
 - Bir bilgisayarın münferit görevleri tamamlama yeteneğini karşılaştırmak için kullanılır
- Rate metric
 - Bu, bir bilgisayarın belirli bir süre içinde kaç görevi yerine getirebileceğinin bir ölçüsüdür.
 - Buna işlem hacmi (*throughput*), kapasite veya hız ölçüsü (*rate measure*) denir
 - Birden çok işlemciden yararlanmak için test edilen sistemin eşzamanlı görevleri yürütmesine izin verir

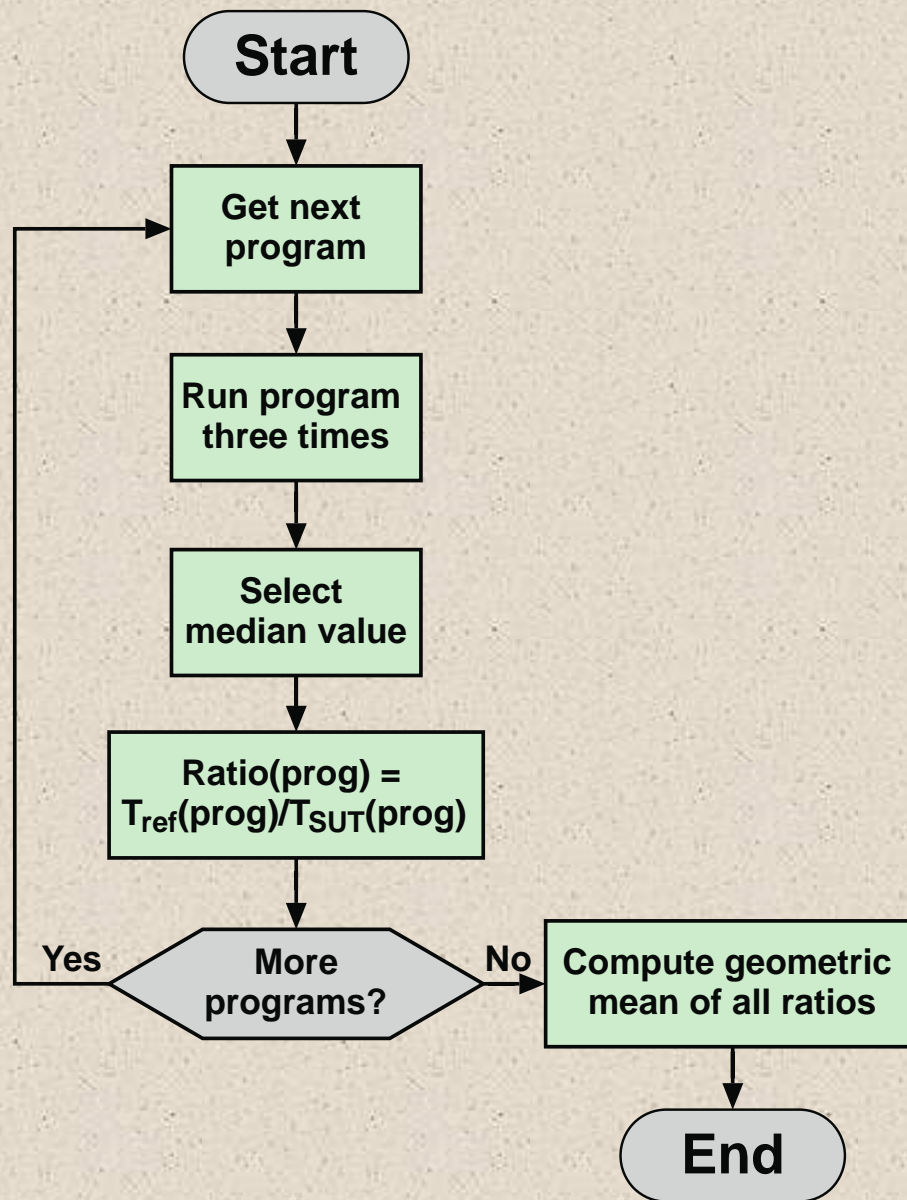


Figure 2.7 SPEC Evaluation Flowchart

Table 2.7 Some SPEC CINT2006 Results

(a) Sun Blade 1000

Benchmark	Execution time	Execution time	Execution time	Reference time	Ratio
400.perlbench	3077	3076	3080	9770	3.18
401.bzip2	3260	3263	3260	9650	2.96
403.gcc	2711	2701	2702	8050	2.98
429.mcf	2356	2331	2301	9120	3.91
445.gobmk	3319	3310	3308	10490	3.17
456.hmmer	2586	2587	2601	9330	3.61
458.sjeng	3452	3449	3449	12100	3.51
462.libquantum	10318	10319	10273	20720	2.01
464.h264ref	5246	5290	5259	22130	4.21
471.omnetpp	2565	2572	2582	6250	2.43
473.astar	2522	2554	2565	7020	2.75
483.xalancbmk	2014	2018	2018	6900	3.42

Sun Blade 1000 için sonuçlar Tablo 2.7a'da gösterilmektedir. SPEC CPU2006 tamsayı benchmarklarından biri 464.h264ref'dir. Bu, en son teknoloji video sıkıştırma standardı olan H.264/AVC'nin (Advanced Video Coding) referans uygulamasıdır. Sun Blade 1000, bu programı ortalama (median) **5259** saniyelik bir sürede çalıştırır. Referans uygulaması 22.130 saniye gerektirir. Oran şu şekilde hesaplanır: $22.130 / 5.259 = 4.21$.

Hız metriği, oranların çarpımının on ikinci kökü alınarak hesaplanır: $(3.18 * 2.96 * 2.98 * 3.91 * 3.17 * 3.61 * 3.51 * 2.01 * 4.21 * 2.43 * 2.75 * 3.42)^{1/12} = 3.12$



(b) Sun Blade X6250

Benchmark	Execution time	Execution time	Execution time	Reference time	Ratio	Rate
400.perlbench	497	497	497	9770	19.66	78.63
401.bzip2	613	614	613	9650	15.74	62.97
403.gcc	529	529	529	8050	15.22	60.87
429.mcf	472	472	473	9120	19.32	77.29
445.gobmk	637	637	637	10490	16.47	65.87
456.hmmer	446	446	446	9330	20.92	83.68
458.sjeng	631	632	630	12100	19.18	76.70
462.libquantum	614	614	614	20720	33.75	134.98
464.h264ref	830	830	830	22130	26.66	106.65
471.omnetpp	619	620	619	6250	10.10	40.39
473.astar	580	580	580	7020	12.10	48.41
483.xalancbmk	422	422	422	6900	16.35	65.40

+ Summary

Chapter 2

Performance Issues

- Designing for performance
 - Microprocessor speed
 - Performance balance
 - Improvements in chip organization and architecture
- Multicore
- MICs
- GPGPUs
- Amdahl's Law
- Little's Law
- Basic measures of computer performance
 - Clock speed
 - Instruction execution rate
- Calculating the mean
 - Arithmetic mean
 - Harmonic mean
 - Geometric mean
- Benchmark principles
- SPEC benchmarks