Fundamentals of Computer Design

Original slides created by: David Patterson, UCB

Edited by: Muhamed Mudawar, KAUST

Outline:

Introducing Computer Architecture

Classes of Computers

Conventional Wisdom in Computer Architecture

Quantitative Principles of Computer Design

Trends in Technology

Power in Integrated Circuits

Trends in Cost

Dependability

Performance

Fallacies and Pitfalls

Computer Architecture Is …

- The attributes of a [computing] system as seen by the programmer, i.e., the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls, the logic design, and the physical implementation. (Amdahl, Blaaw, and Brooks, 1964)

3

Computer Architecture's Changing Definition

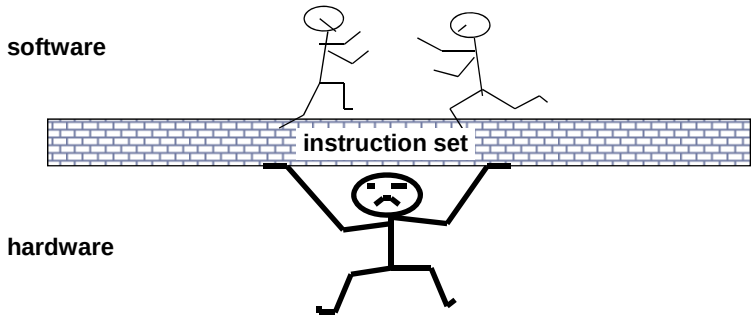
- 1950s to 1960s: Computer Architecture Course
  - Computer Arithmetic
- 1970s to mid 1980s: Computer Architecture Course
  - Instruction Set Design, especially ISA appropriate for compilers
- 1990s to 2000s: Computer Architecture Course
  - Design of CPU, memory system, I/O system, Multiprocessors

4

ISA vs. Computer Architecture

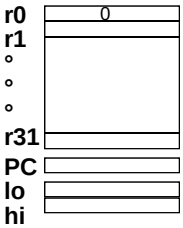
- Old definition of computer architecture = instruction set design
  - Other aspects of computer design called implementation
  - Insinuates implementation is uninteresting or less challenging
- Our view is computer architecture >> ISA
- Architect's job much more than instruction set design; technical hurdles today *more* challenging than those in instruction set design
- Since instruction set design not where action is, some conclude computer architecture (using old definition) is not where action is
  - We disagree on conclusion

Instruction Set Architecture is a Critical Interface



- Properties of a good abstraction
  - Lasts through many generations (portability)
  - Used in many different ways (generality)
  - Provides convenient functionality to higher levels
  - Permits an efficient implementation at lower levels

Example: MIPS architecture



Programmable storage

- 2^32 x bytes
- 31 x 32-bit GPRs (R0=0)
- 32 x 32-bit FP regs (paired DP)
- HI, LO, PC

Data types

Formats

Addressing Modes

Arithmetic logical

Add, AddU, Sub, SubU, And, Or, Xor, Nor, SLT, SLTU,
AddI, AddIU, SLTI, SLTIU, AndI, OrI, XorI, *LUI*
SLL, SRL, SRA, SLLV, SRLV, SRAV

Memory Access

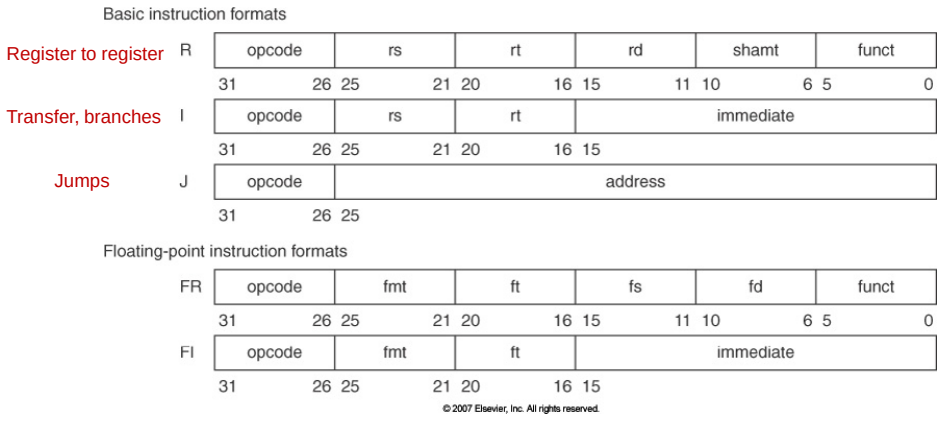
LB, LBU, LH, LHU, LW, LWL,LWR
SB, SH, SW, SWL, SWR

Control

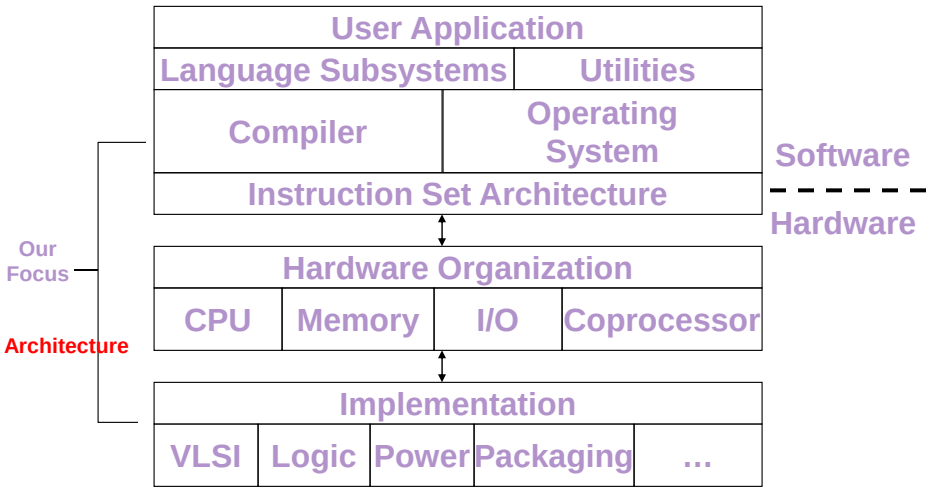
32-bit instructions on word boundary

J, JAL, JR, JALR
BEq, BNE, BLEZ,BGTZ,BLTZ,BGEZ,BLTZAL,BGEZAL

MIPS architecture instruction set format



Aspects of Computer Design



9

Next:

- Introducing Computer Architecture
- Classes of Computers
- Conventional Wisdom in Computer Architecture
- Quantitative Principles of Computer Design
- Trends in Technology
- Power in Integrated Circuits
- Trends in Cost
- Dependability
- Performance
- Fallacies and Pitfalls

Three main classes of computers

Desktop: personal computer

Server: web servers, file servers, database servers

Embedded: handheld devices (phones, cameras), dedicated parallel computers

Feature	Desktop	Server	Embedded
Price of system	$500 - $5000	$5000 - $5,000,000	$10 - $100,000
Price of multiprocessor module	$50 - $500	$200 - $10,000	$.01 - $100
Critical system design issues	Price-performance, Graphics performance	Throughput, Availability, Scalability	Price, Power consumption, Application-specific performance

Next:

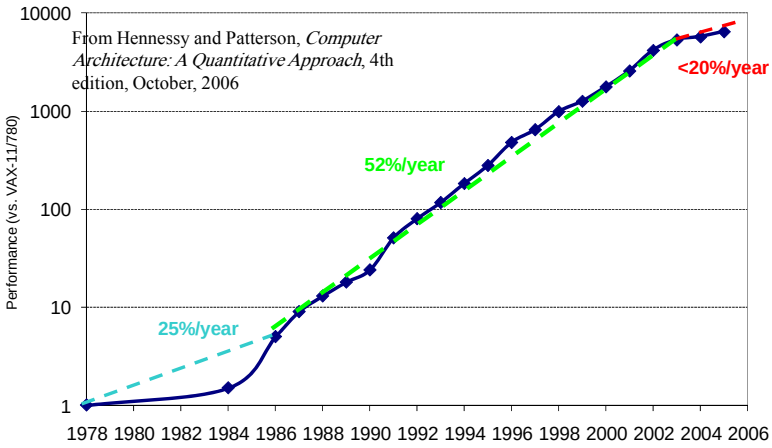
- Introducing Computer Architecture
- Classes of Computers
- **Conventional Wisdom in Computer Architecture**
- Quantitative Principles of Computer Design
- Trends in Technology
- Power in Integrated Circuits
- Trends in Cost
- Dependability
- Performance
- Fallacies and Pitfalls

Conventional Wisdom in Computer Architecture

- Old Conventional Wisdom: Power is free, Transistors are expensive
- New Conventional Wisdom: "Power wall" Power is expensive, Transistors are free (Can put more on chip than can afford to turn on)
- Old CW: Sufficiently increasing Instruction Level Parallelism via compilers, innovation (Out-of-order, speculation, …)
- New CW: "ILP wall" law of diminishing returns on more HW for ILP
- Old CW: Multiplies are slow, Memory access is fast
- New CW: "Memory wall" Memory is slow, multiplies are fast (200 clock cycles to DRAM memory, 4 clocks for multiply)
- Old CW: Uniprocessor performance 2X / 1.5 yrs
- New CW: Power Wall + ILP Wall + Memory Wall = Brick Wall
  - Uniprocessor performance now 2X / (? 5) yrs

  ⇒ Change in chip design: multiple "cores" (2X processors per chip / ~ 2 years)
    - More simpler processors are more power efficient

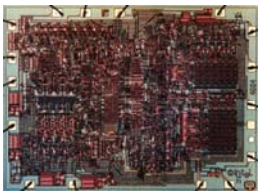
Crossroads: Uniprocessor Performance



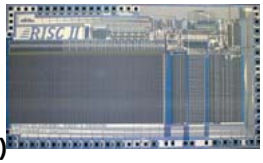
- **VAX : 25%/year 1978 to 1986**
- **RISC + x86: 52%/year 1986 to 2002**
- **RISC + x86: less than 20%/year 2002 to present**

Sea Change in Chip Design

- Intel 4004 (1971): 4-bit processor, 2312 transistors, 0.4 MHz, 10 micron PMOS, 11 mm² chip



- **RISC II (1983): 32-bit, 5 stage pipeline, 40,760 transistors, 3 MHz, 3 micron NMOS, 60 mm² chip**



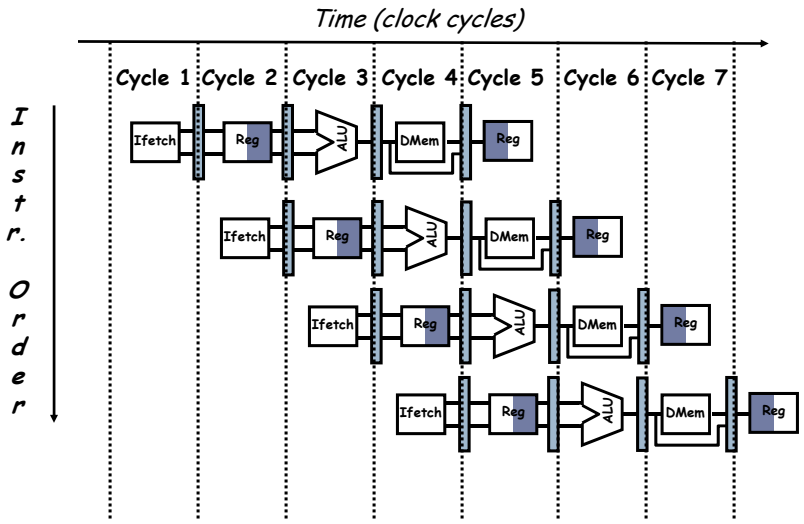
- **125 mm² chip, 0.065 micron CMOS = 2312 RISC II+FPU+Icache+Dcache**
  - **RISC II shrinks to ~ 0.02 mm² at 65 nm**
  - **Caches via DRAM or 1 transistor T-RAM (www.t-ram.com)**
  - **Proximity Communication via capacitive coupling at > 1 TB/s ? (Ivan Sutherland @ Sun / Berkeley)**

- **Processor is the new transistor?**

Taking Advantage of Parallelism

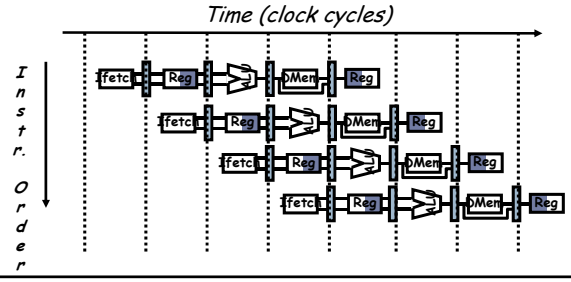
- Increasing throughput of server computer via multiple processors or multiple disks
- Detailed HW design
  - Carry lookahead adders uses parallelism to speed up computing sums from linear to logarithmic in number of bits per operand
  - Multiple memory banks searched in parallel in set-associative caches
- Pipelining: overlap instruction execution to reduce the total time to complete an instruction sequence.
  - Not every instruction depends on immediate predecessor ⇒ executing instructions completely/partially in parallel is possible
  - Classic 5-stage pipeline:
    1) Instruction Fetch (Ifetch),
    2) Register Read (Reg),
    3) Execute (ALU),
    4) Data Memory Access (Dmem),
    5) Register Write (Reg)

Pipelined Instruction Execution



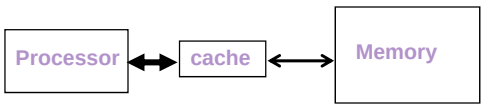
Limits to pipelining

- Hazards prevent next instruction from executing during its designated clock cycle
  - Structural hazards: attempt to use the same hardware to do two different things at once
  - Data hazards: Instruction depends on result of prior instruction still in the pipeline
  - Control hazards: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps).

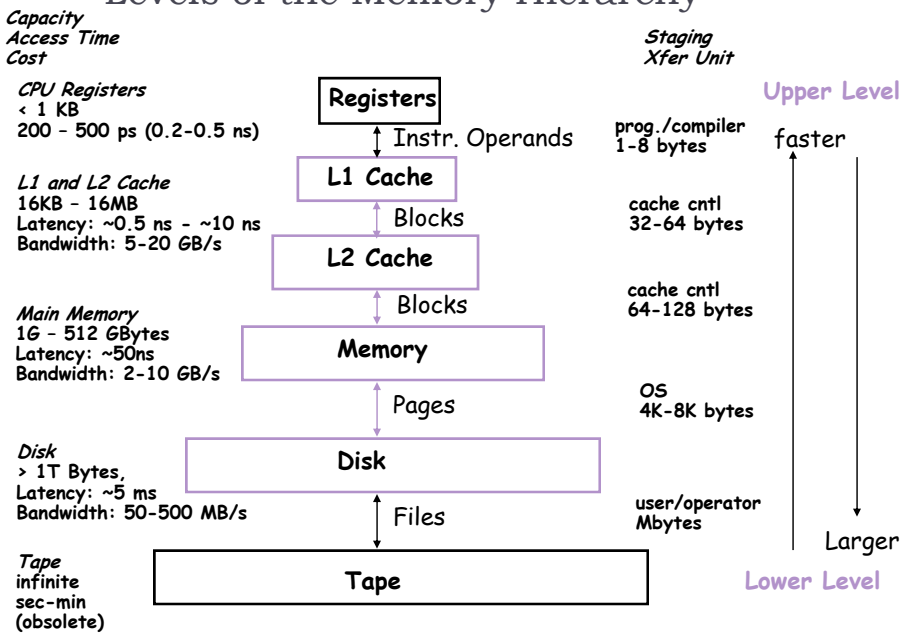


The Principle of Locality

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straight-line code, array access)
- Last 30 years, Cache memory relied on locality for performance



Levels of the Memory Hierarchy



What Computer Architecture brings to Table

- Quantitative Principles of Design
  1. Take Advantage of Parallelism
  2. Principle of Locality
  3. Focus on the Common Case
  4. Amdahl's Law
  5. The Processor Performance Equation
- Defining, quantifying, and comparing
  - Performance
  - Cost
  - Dependability
  - Power
- Anticipating and exploiting advances in technology
- Well-defined interfaces that are carefully implemented and thoroughly checked

Next:

- Introducing Computer Architecture
- Classes of Computers
- Conventional Wisdom in Computer Architecture
- **Quantitative Principles of Computer Design**
- Trends in Technology
- Power in Integrated Circuits
- Trends in Cost
- Dependability
- Performance
- Fallacies and Pitfalls

Focus on the Common Case

- Common sense guides computer design
  - Since its engineering, common sense is valuable
- In making a design trade-off, favor the frequent case over the infrequent case
  - E.g., Instruction fetch and decode unit used more frequently than multiplier, so optimize it first
  - E.g., If database server has 50 disks / processor, storage dependability dominates system dependability, so optimize it first
- Frequent case is often simpler and can be done faster than the infrequent case
  - E.g., overflow is rare when adding 2 numbers, so improve performance by optimizing more common case of no overflow
  - May slow down overflow, but overall performance improved by optimizing for the normal case
- What is frequent case and how much performance improved by making common case faster => Amdahl's Law

Amdahl's Law

$$\text{ExTime}_{new} = \text{ExTime}_{old} \times \left[\left(1 - \text{Fraction}_{enhanced}\right) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}} \right]$$

$$\text{Speedup}_{overall} = \frac{\text{ExTime}_{old}}{\text{ExTime}_{new}} = \frac{1}{\left(1 - \text{Fraction}_{enhanced}\right) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}}}$$

If $\text{Speedup}_{enhanced}$ → infinity

$$\text{Speedup}_{maximum} = \frac{1}{\left(1 - \text{Fraction}_{enhanced}\right)}$$

Amdahl's Law example: Web Server

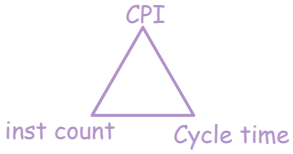
- Original CPU: 40% computation and 60% waiting for I/O
- New CPU: 10X faster on computation than original CPU

$$\text{Speedup}_{overall} = \frac{1}{\left(1 - \text{Fraction}_{enhanced}\right) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}}}$$

$$= \frac{1}{\left(1 - 0.4\right) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

- **Apparently, its human nature to be attracted by 10X faster, vs. keeping in perspective its just 1.56X faster**

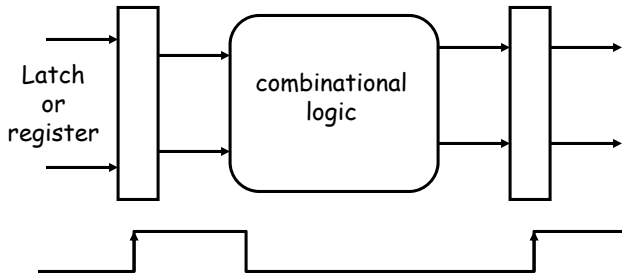
Processor performance equation



$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
Inst. Set.	X	X	
Organization	X		X
Technology			X

What's a Clock Cycle?



- Old days: 10 levels of gates
- Today: determined by numerous time-of-flight issues + gate delays
  - clock propagation, wire lengths, drivers

Next:

Introducing Computer Architecture

Classes of Computers

Conventional Wisdom in Computer Architecture

Quantitative Principles of Computer Design

Trends in Technology

Power in Integrated Circuits

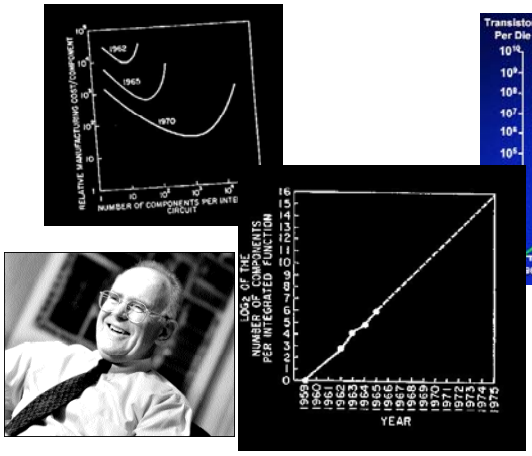
Trends in Cost

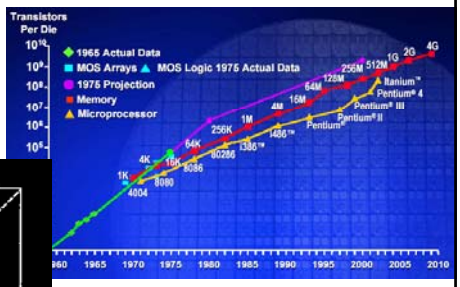
Dependability

Performance

Fallacies and Pitfalls

Moore's Law: 2X transistors / "year"





- "Cramming More Components onto Integrated Circuits"
  - Gordon Moore, Electronics, 1965
- # on transistors / cost-effective integrated circuit double every N months (12 ≤ N ≤ 24)

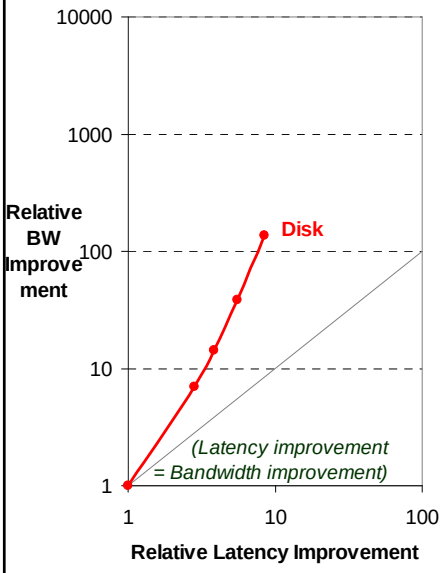
Tracking Technology Performance Trends

- Drill down into 4 technologies:
  - Disks,
  - Memory,
  - Network,
  - Processors
- Compare ~1980 Archaic vs. ~2000 Modern
  - Performance Milestones in each technology
- Compare for Bandwidth vs. Latency improvements in performance over time
- Bandwidth: number of events per unit time
  - E.g., M bits / second over network, M bytes / second from disk
- Latency: elapsed time for a single event
  - E.g., one-way network delay in microseconds, average disk access time in milliseconds

Disks: Archaic (Nostalgic) v. Modern (Newfangled)

CDC Wren I, 1983	Seagate 373453, 2003	
3600 RPM	15000 RPM	(4X)
0.03 GBytes capacity	73.4 GBytes	(2500X)
Tracks/Inch: 800	Tracks/Inch: 64000	(80X)
Bits/Inch: 9550	Bits/Inch: 533,000	(60X)
Three 5.25" platters	Four 2.5" platters (in 3.5" form factor)	
Bandwidth: 0.6 MBytes/sec	Bandwidth: 86 MBytes/sec	(140X)
Latency: 48.3 ms	Latency: 5.7 ms	(8X)
Cache: none	Cache: 8 MBytes	

Latency Lags Bandwidth (for last ~20 years)



- Performance Milestones
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

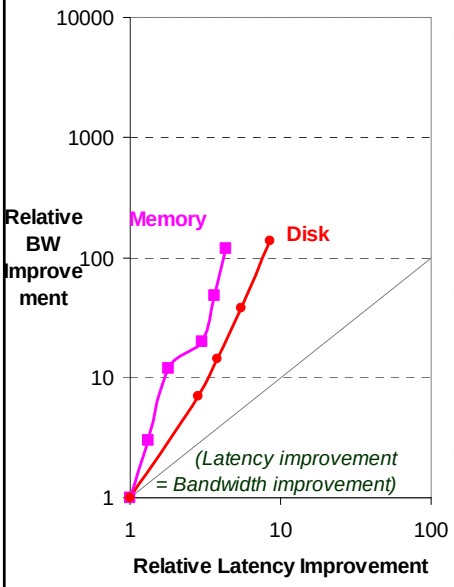
(latency = simple operation w/o contention
BW = best-case)

Memory: Archaic (Nostalgic) v. Modern (Newfangled)

- 1980 DRAM (asynchronous)
- 0.06 Mbits/chip
- 64,000 xtors, 35 mm^2
- 16-bit data bus per module,
- 16 pins/chip
- 13 Mbytes/sec
- Latency: 225 ns
- (no block transfer)

- 2000 Double Data Rate Synchr. (clocked) DRAM
- 256.00 Mbits/chip (4000X)
- 256,000,000 xtors, 204 mm^2
- 64-bit data bus per DIMM (4X)
- 66 pins/chip
- 1600 Mbytes/sec (120X)
- Latency: 52 ns (4X)
- Block transfers (page mode)

Latency Lags Bandwidth (last ~20 years)



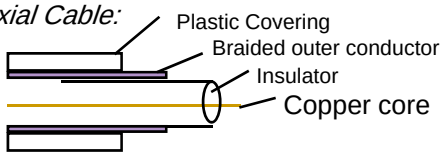
- Performance Milestones
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x,120x)
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

(latency = simple operation w/o contention
BW = best-case)

LANs: Archaic (Nostalgic)v. Modern (Newfangled)

- Ethernet 802.3
- Year of Standard: 1978
- 10 Mbits/s link speed
- Latency: 3000 μsec
- Shared media
- Coaxial cable

Coaxial Cable:



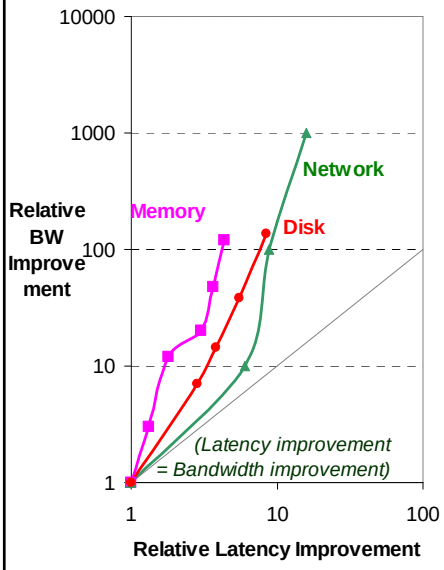
- **Ethernet 802.3ae**
- **Year of Standard: 2003**
- **10,000 Mbits/s (1000X) link speed**
- **Latency: 190 μsec (15X)**
- **Switched media**
- **Category 5 copper wire**

"Cat 5" is 4 twisted pairs in bundle

Twisted Pair:

Copper, 1mm thick,
twisted to avoid antenna effect

Latency Lags Bandwidth (last ~20 years)

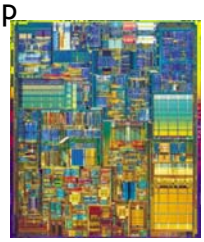


- Performance Milestones
- Ethernet: 10Mb, 100Mb, 1000Mb, 10000 Mb/s (16x,1000x)
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x,120x)
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

(latency = simple operation w/o contention
BW = best-case)

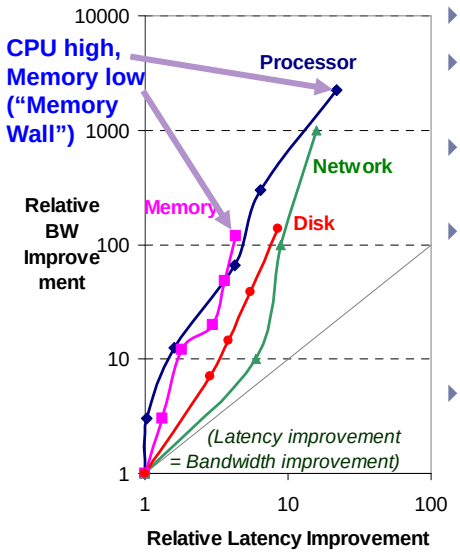
CPUs: Archaic (Nostalgic) v. Modern (Newfangled)

- 1982 Intel 80286
- 12.5 MHz
- 2 MIPS (peak)
- Latency 320 ns
- 134,000 xtors, 47 mm^2
- 16-bit data bus, 68 pins
- Microcode interpreter, separate FPU chip
- (no caches)



- 2001 Intel Pentium 4
- 1500 MHz (120X)
- 4500 MIPS (peak) (2250X)
- Latency 15 ns (20X)
- 42,000,000 xtors, 217 mm^2
- 64-bit data bus, 423 pins
- 3-way superscalar, Dynamic translate to RISC, Superpipelined (22 stage), Out-of-Order execution
- On-chip 8KB Data caches, 96KB Instr.Trace cache, 256KB L2 cache

Latency Lags Bandwidth (last ~20 years)



- Performance Milestones
- Processor: '286, '386, '486, Pentium, Pentium Pro, Pentium 4 (21x,2250x)
- Ethernet: 10Mb, 100Mb, 1000Mb, 10000 Mb/s (16x,1000x)
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x,120x)
- Disk : 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

Rule of Thumb for Latency Lagging BW

- In the time that bandwidth doubles, latency improves by no more than a factor of 1.2 to 1.4
  (and capacity improves faster than bandwidth)
- Stated alternatively:
  Bandwidth improves by more than the square of the improvement in Latency

6 Reasons Latency Lags Bandwidth

1. Moore's Law helps BW more than latency

- Faster transistors, more transistors, more pins help Bandwidth
  - CPU Transistors: 0.130 vs. 42 M xtors (300X)
  - DRAM Transistors: 0.064 vs. 256 M xtors (4000X)
  - CPU Pins: 68 vs. 423 pins (6X)
  - DRAM Pins: 16 vs. 66 pins (4X)
- Smaller, faster transistors but communicate over (relatively) longer lines: limits latency
  - Feature size: 3 vs. 0.18 micron (17X)
  - CPU Die Size: 35 vs. 204 mm^2 (6X)
  - DRAM Die Size: 47 vs. 217 mm^2 (5X)

6 Reasons Latency Lags Bandwidth (cont'd)

2. Distance increases latency

- Size of DRAM block ⇒ long word lines and bit lines ⇒ most of DRAM access time
- Speed of light and computers on network
- 1. & 2. explains linear latency vs. square BW

3. Bandwidth easier to sell ("bigger=better")

- E.g., 10 Gbit/s Ethernet ("10 Gig") vs. 10 μsec latency Ethernet
- 4400 MB/s DIMM ("PC4400") vs. 50 ns latency
- Even if just marketing, customers now trained
- Since bandwidth sells, more resources thrown at bandwidth, which further tips the balance

6 Reasons Latency Lags Bandwidth (cont'd)

4. Latency helps BW, but not vice versa
   - Spinning disk faster improves both rotational latency and bandwidth
     - 3600 RPM ⇒ 15000 RPM = 4.2X
     - Average rotational latency: 8.3 ms ⇒ 2.0 ms
     - Things being equal, also helps BW by 4.2X
   - Lower DRAM latency ⇒ More DRAM access/second (higher bandwidth)
   - Higher linear density helps disk BW (and capacity), but not disk Latency
     - 9,550 BPI ⇒ 533,000 BPI ⇒ 60X in BW, but not in latency

6 Reasons Latency Lags Bandwidth (cont'd)

5. Bandwidth hurts latency
   - Adding chips to widen a memory module increases Bandwidth but higher fan-out on address lines may increase Latency
6. Operating System overhead hurts Latency more than Bandwidth
   - Scheduling queues increase latency (more delays)
   - Queues help Bandwidth, but hurt Latency

Next:

- **Introducing Computer Architecture**
- **Classes of Computers**
- **Conventional Wisdom in Computer Architecture**
- **Quantitative Principles of Computer Design**
- **Trends in Technology**
- **Power in Integrated Circuits**
- **Trends in Cost**
- **Dependability**
- **Performance**
- **Fallacies and Pitfalls**

Dynamic Power

- For CMOS chips, traditional dominant energy consumption has been in switching transistors, called *dynamic power:*

$$\text{Power}_{dynamic} = 0.5 \times \text{CapacitiveLoad} \times \text{Voltage}^2 \times \text{FrequencySwitched}$$

- For mobile devices, energy is a better metric

$$\text{Energy}_{dynamic} = \text{CapacitiveLoad} \times \text{Voltage}^2$$

- For a fixed task, slowing clock rate (frequency switched) reduces power, but not energy
- Capacitive load a function of number of transistors connected to output and technology, which determines capacitance of wires and transistors
- Dropping voltage helps both, so went from 5V to 1V
- To save energy & dynamic power, most CPUs now turn off clock of inactive modules

Example of Dynamic Power

- Some microprocessors today are designed to have adjustable voltage, so that a 15% reduction in voltage results in a 15% reduction in frequency. What is impact on dynamic power?

$$\begin{aligned} Power_{dynamic} &= 1/2 \times CapacitiveLoad \times Voltage^2 \times Frequency \\ &= 1/2 \times CapacitiveLoad \times (.85 \times Voltage_{old})^2 \times (0.85 \times Frequency_{old}) \\ &= (.85)^3 \times OldPower_{dynamic} \\ &\approx 0.61 \times OldPower_{dynamic} \end{aligned}$$

- Power is reduced to about 61% of original power

Static Power

- Because leakage current flows even when a transistor is off, now *static power* important too

$$Power_{static} = Current_{static} \times Voltage$$

- Leakage current increases in processors with smaller transistor sizes
- Increasing the number of transistors increases power even if they are turned off
- In 2006, goal for leakage is 25% of total power consumption; high performance designs at 40%
- Very low power systems even gate voltage to inactive modules to control loss due to leakage

Next:

- Introducing Computer Architecture
- Classes of Computers
- Conventional Wisdom in Computer Architecture
- Quantitative Principles of Computer Design
- Trends in Technology
- Power in Integrated Circuits
- **Trends in Cost**
- Dependability
- Performance
- Fallacies and Pitfalls

Cost of Integrated Circuits depends of several factors

Time:

- The price drops with time, learning curve increases

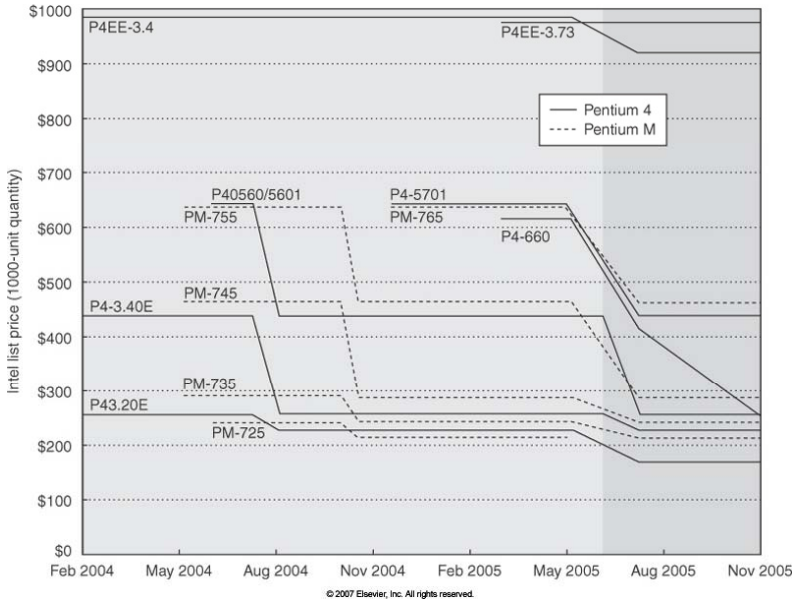
Volume:

- The price drops with volume increase

Commodities:

- Many manufacturers produce the same product
- Competition brings prices down

The price of Intel Pentium 4 and Pentium M



Cost of Integrated Circuit

$$\text{Cost of IC} = \frac{\text{Cost of Die} + \text{Cost of Testing die} + \text{Packaging and Final test}}{\text{Final Test Yield}}$$

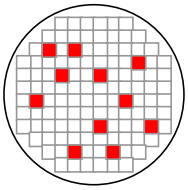
$$\text{Cost of Die} = \frac{\text{Cost of Wafer}}{\text{Dies per wafer} \times \text{Die Yield}}$$

$$\text{Die Yield} = \left(1 + \frac{\text{Defects per unit area} \times \text{Die area}}{\alpha}\right)^{-\alpha}$$

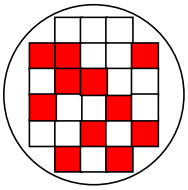
Formula for Die Yield is empirical, developed from manufacturing data

α is a measure of manufacturing complexity, which corresponds to the number of critical masking levels. A good estimate of $\alpha = 4$.

Effect of Die Size on Yield



120 dies, 109 good



26 dies, 15 good

Good Die

Defective Die

Dramatic decrease in yield with larger dies

Die Yield = (Number of Good Dies) / (Total Number of Dies)

$$\text{Die Yield} = \frac{1}{(1 + (\text{Defect per area} \times \text{Die area} / \alpha))^{\alpha}}$$

Die Cost = (Wafer Cost) / (Dies per Wafer × Die Yield)

Dies per Wafer

$$\text{Dies per Wafer} = \frac{\pi \times (\text{Wafer Diameter} / 2)^2}{\text{Die Area}} - \frac{\pi \times \text{Wafer Diameter}}{\sqrt{2 \times \text{Die Area}}}$$

First Term is ratio of Wafer Area to Die Area

Second Term is approximately the number of dies along the edge

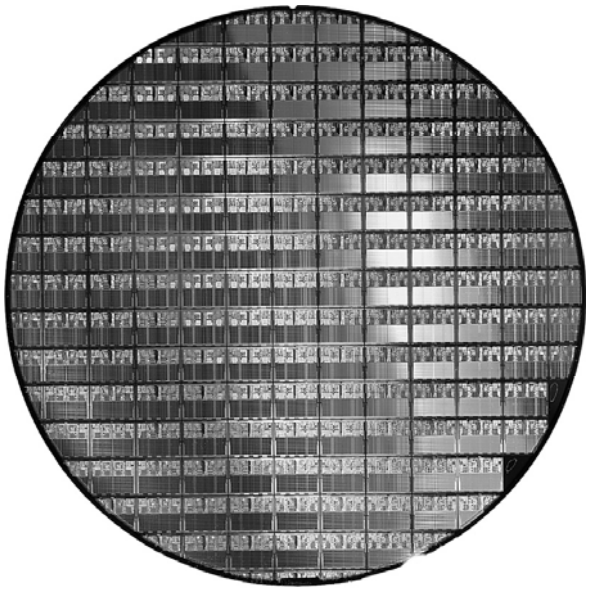
Second Term divides the Wafer Circumference by the Diagonal of a square die

Example:

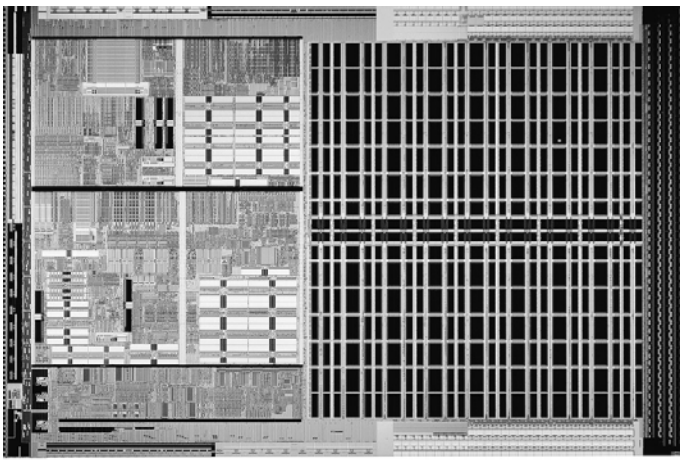
Wafer Diameter = 300mm (30 cm), Die Area = 2.25 cm^2

$$\text{Dies per Wafer} = \frac{\pi \times (30 / 2)^2}{2.25} - \frac{\pi \times 30}{\sqrt{2 \times 2.25}} = 270$$

A 300mm silicon wafer contains 117 AMD Opteron microprocessor chips in a 90nm process



AMD Opteron Microprocessor Die



Example on Die Yield (Assume $\alpha = 4$)

Die area = 1.5 cm × 1.5 cm = 2.25 cm^2

Defect density = 0.4 per cm^2

$$\text{Die Yield} = \left(1 + \frac{0.4 \times 2.25}{4}\right)^{-4} = 0.44$$

Die area = 1 cm × 1 cm = 1 cm^2

Defect density = 0.4 per cm^2

$$\text{Die Yield} = \left(1 + \frac{0.4 \times 1.0}{4}\right)^{-4} = 0.68$$

Smaller die area gives more die yield

Next:

Introducing Computer Architecture
Classes of Computers
Conventional Wisdom in Computer Architecture
Quantitative Principles of Computer Design
Trends in Technology
Power in Integrated Circuits
Trends in Cost
Dependability
Performance
Fallacies and Pitfalls

Define and quantify dependability

- How decide when a system is operating properly?
- Internet service providers now offer Service Level Agreements (SLA) to guarantee that their networking or power service would be dependable
- Systems alternate between 2 states of service with respect to an SLA:

1. Service accomplishment, where the service is delivered as specified in SLA
2. Service interruption, where the delivered service is different from the SLA

- Failure = transition from state 1 to state 2
- Restoration = transition from state 2 to state 1

Define and quantify dependability-cont'd

- *Module reliability* = measure of continuous service accomplishment (or time to failure).
  2 metrics

1. *Mean Time To Failure* (*MTTF*) measures Reliability
2. *Failures In Time* (*FIT*) = 10^9/MTTF, the rate of failures
   - Traditionally reported as failures per billion hours of operation

- *Mean Time To Repair* (*MTTR*) measures Service Interruption
  - *Mean Time Between Failures* (*MTBF*) = MTTF+MTTR
- *Module availability* measures service as alternate between the 2 states of accomplishment and interruption (number between 0 and 1, e.g. 0.9)
- *Module availability = MTTF / (MTTF + MTTR)*

Example calculating reliability

- Calculate FIT and MTTF for 10 disks (1,000,000 hour MTTF per disk), a disk controller (500,000 hour MTTF), and a power supply (200,000 MTTF)
- Assume Failures are independent and lifetimes are exponentially distributed (age of module does not affect probability of failure). Then overall failure rate is the sum of failure rates of the modules.

$$
\begin{aligned}
FailureRate &= 10 \times (1/1{,}000{,}000) + 1/500{,}000 + 1/200{,}000 \\
&= (10 + 2 + 5)/1{,}000{,}000 \\
&= 17/1{,}000{,}000 \\
&= 17{,}000\,FIT \\
MTTF &= 1{,}000{,}000{,}000/17{,}000 \\
&\approx 59{,}000\,hours \text{ (under 7 years)}
\end{aligned}
$$

Next:

Introducing Computer Architecture
Classes of Computers
Conventional Wisdom in Computer Architecture
Quantitative Principles of Computer Design
Trends in Technology
Power in Integrated Circuits
Trends in Cost
Dependability
Performance
Fallacies and Pitfalls

Response Time and Throughput

- Response Time
  - Time between start and completion of a task, as observed by end user
  - Response Time = CPU Time + Waiting Time (I/O, OS scheduling, etc.)
- Throughput
  - Number of tasks the machine can run in a given period of time
- Decreasing execution time improves throughput
  - Example: using a faster version of a processor
  - Less time to run a task ⇒ more tasks can be executed
- Increasing throughput can also improve response time
  - Example: increasing number of processors in a multiprocessor
  - More tasks can be executed in parallel
  - Execution time of individual sequential tasks is not changed
  - But less waiting time in scheduling queue reduces response time

Defining Performance

- For some program running on machine *X*

$$\text{Performance}_X = \frac{1}{\text{Execution time}_X}$$

- *X* is *n* times faster than *Y*

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$

Performance: What to measure

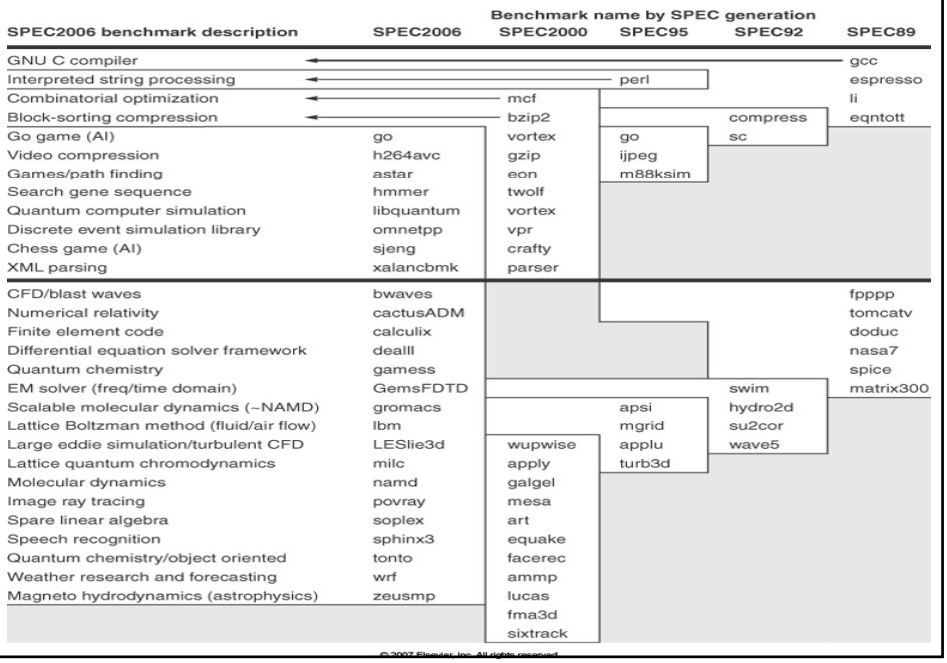
- Usually rely on benchmarks vs. real workloads
- To increase predictability, collections of benchmark applications, called *benchmark suites*, are popular
- SPECCPU: popular desktop benchmark suite
  - CPU only, split between integer and floating point programs
  - SPECint2000 has 12 integer, SPECfp2000 has 14 floating-point programs
  - SPECCPU2006 announced 2006 (12 integer + 17 FP programs)
  - SPECSFS (NFS file server) and SPECWeb (WebServer) added as server benchmarks
- Transaction Processing Council measures server performance and cost-performance for databases
  - TPC-C Complex query for Online Transaction Processing
  - TPC-H models ad hoc decision support
  - TPC-W a transactional web benchmark
  - TPC-App application server and web services benchmark

The SPEC CPU2000 Benchmarks

12 Integer benchmarks (C and C++)		14 FP benchmarks (Fortran 77, 90, and C)	
Name	Description	Name	Description
gzip	Compression	wupwise	Quantum chromodynamics
vpr	FPGA placement and routing	swim	Shallow water model
gcc	GNU C compiler	mgrid	Multigrid solver in 3D potential field
mcf	Combinatorial optimization	applu	Partial differential equation
crafty	Chess program	mesa	Three-dimensional graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Neural networks image recognition
perlbmk	Perl application	equake	Seismic wave propagation simulation
gap	Group theory, interpreter	facerec	Image recognition of faces
vortex	Object-oriented database	ammp	Computational chemistry
bzip2	Compression	lucas	Primality testing
twolf	Place and route simulator	fma3d	Crash simulation using finite elements
		sixtrack	High-energy nuclear physics
		apsi	Meteorology: pollutant distribution

- **Wall clock time is used as metric**
- **Benchmarks measure CPU time, because of little I/O**

SPEC 2006 Benchmarks



How to Summarize Suite Performance (1/5)

- Arithmetic average of execution time of all programs?
  - But they vary by 4X in speed, so some would be more important than others in arithmetic average
- Could add a weight per program, but how to pick weights?
  - Different companies want different weights for their products
- SPECRatio: Normalize execution times to reference computer, yielding a ratio proportional to performance

$$\text{SPECRatio} = \frac{\text{time on reference computer}}{\text{time on computer being rated}}$$

SPECfp2000 Execution Times & SPECRatios

Benchmark	Ultra 5 Time (sec)	Opteron Time (sec)	SpecRatio Opteron	Itanium2 Time (sec)	SpecRatio Itanium2	Opteron/ Itanium2 Times	Itanium2/ Opteron SpecRatios
wupwise	1600	51.5	31.06	56.1	28.53	0.92	0.92
swim	3100	125.0	24.73	70.7	43.85	1.77	1.77
mgrid	1800	98.0	18.37	65.8	27.36	1.49	1.49
applu	2100	94.0	22.34	50.9	41.25	1.85	1.85
mesa	1400	64.6	21.69	108.0	12.99	0.60	0.60
galgel	2900	86.4	33.57	40.0	72.47	2.16	2.16
art	2600	92.4	28.13	21.0	123.67	4.40	4.40
equake	1300	72.6	17.92	36.3	35.78	2.00	2.00
facerec	1900	73.6	25.80	86.9	21.86	0.85	0.85
ammp	2200	136.0	16.14	132.0	16.63	1.03	1.03
lucas	2000	88.8	22.52	107.0	18.76	0.83	0.83
fma3d	2100	120.0	17.48	131.0	16.09	0.92	0.92
sixtrack	1100	123.0	8.95	68.8	15.99	1.79	1.79
apsi	2600	150.0	17.36	231.0	11.27	0.65	0.65
Geometric Mean			20.86		27.12	1.30	1.30

Geometric mean of ratios = 1.30 = Ratio of Geometric means = 27.12 / 20.86

How Summarize Suite Performance (2/5)

- If program SPECRatio on Computer A is 1.25 times bigger than Computer B, then

$$1.25 = \frac{SPECRatio_A}{SPECRatio_B} = \frac{\frac{ExecutionTime_{reference}}{ExecutionTime_A}}{\frac{ExecutionTime_{reference}}{ExecutionTime_B}}$$

$$= \frac{ExecutionTime_B}{ExecutionTime_A} = \frac{Performance_A}{Performance_B}$$

- **Note that when comparing 2 computers as a ratio, execution times on the reference computer drop out, so choice of reference computer is irrelevant**

How Summarize Suite Performance (3/5)

- Since ratios, proper mean is geometric mean (SPECRatio is unitless, so arithmetic mean is meaningless)

$$GeometricMean = \sqrt[n]{\prod_{i=1}^{n} SPECRatio_i}$$

1. **Geometric mean of the ratios is the same as the ratio of the geometric means**
2. **Ratio of geometric means = Geometric mean of performance ratios ⇒ choice of reference computer is irrelevant!**

- **These two points make geometric mean of ratios attractive to summarize performance**

Ratio of Geometric Means = Geometric mean of the performance Ratios

$$\frac{\text{Geometric mean A}}{\text{Geometric mean B}} = \frac{\sqrt[n]{\prod_{i=1}^{n} \text{SPECRatio A}_i}}{\sqrt[n]{\prod_{i=1}^{n} \text{SPECRatio B}_i}} = \sqrt[n]{\prod_{i=1}^{n} \frac{\text{SPECRatio A}_i}{\text{SPECRatio B}_i}}$$

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{\frac{\text{Execution time ref}_i}{\text{Execution time A}_i}}{\frac{\text{Execution time ref}_i}{\text{Execution time B}_i}}} = \sqrt[n]{\prod_{i=1}^{n} \frac{\text{Execution time B}_i}{\text{Execution time A}_i}} = \sqrt[n]{\prod_{i=1}^{n} \frac{\text{Performance A}_i}{\text{Performance B}_i}}$$

How Summarize Suite Performance (4/5)

- Does a single mean well summarize performance of programs in benchmark suite?
- Can decide if mean a good predictor by characterizing variability of distribution using standard deviation
- Like geometric mean, geometric standard deviation is multiplicative rather than arithmetic
- Can simply take the logarithm of SPECRatios, compute the standard mean and standard deviation, and then take the exponent to convert back:

$$GeometricMean = \exp\left(\frac{1}{n} \times \sum_{i=1}^{n} \ln\left(SPECRatio_i\right)\right)$$

$$GeometricStDev = \exp\left(StDev\left(\ln\left(SPECRatio_i\right)\right)\right)$$

How Summarize Suite Performance (5/5)

- Standard deviation is more informative if know distribution has a standard form
  - *bell-shaped normal distribution*, whose data are symmetric around mean
  - *lognormal distribution*, where logarithms of data--not data itself--are normally distributed (symmetric) on a logarithmic scale
- For a lognormal distribution, we expect that

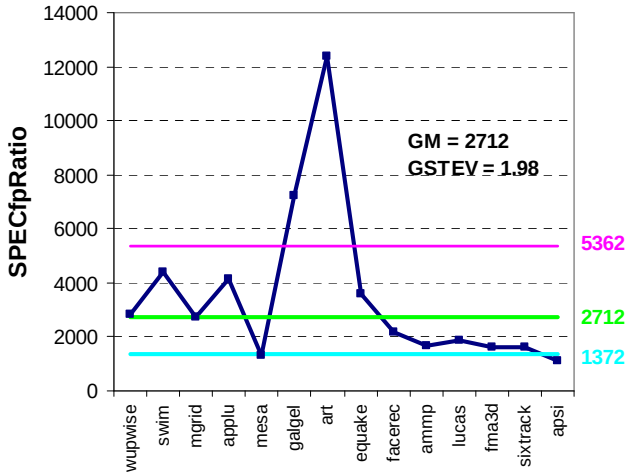
68% of samples fall in range $[mean / gstdev, mean \times gstdev]$

95% of samples fall in range $[mean / gstdev^2, mean \times gstdev^2]$

- Note: Excel provides functions EXP(), LN(), and STDEV() that make calculating geometric mean and multiplicative standard deviation easy

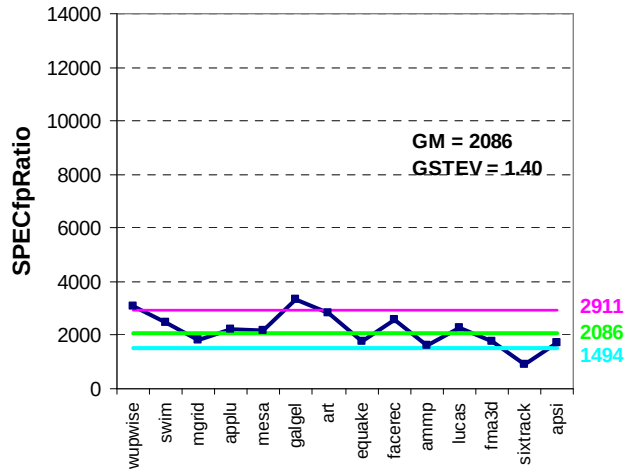
Example Standard Deviation (1/2)

- GM and multiplicative StDev of SPECfp2000 for Itanium 2



Example Standard Deviation (2/2)

- GM and multiplicative StDev of SPECfp2000 for AMD Athlon



Comments on Itanium 2 and Athlon

- Standard deviation of 1.98 for Itanium 2 is much higher--vs. 1.40--so results will differ more widely from the mean, and therefore are likely less predictable
- Falling within one standard deviation:
  - 10 of 14 benchmarks (71%) for Itanium 2
  - 11 of 14 benchmarks (78%) for Athlon
- Thus, the results are quite compatible with a lognormal distribution (expect 68%)

Next:

Introducing Computer Architecture
Classes of Computers
Conventional Wisdom in Computer Architecture
Quantitative Principles of Computer Design
Trends in Technology
Power in Integrated Circuits
Trends in Cost
Dependability
Performance
Fallacies and Pitfalls

Fallacies and Pitfalls

- Fallacies - commonly held misconceptions
  - When discussing a fallacy, we try to give a counterexample.
- Pitfalls - easily made mistakes.
  - Often generalizations of principles true in limited context
  - Show Fallacies and Pitfalls to help you avoid these errors
- Fallacy: Benchmarks remain valid indefinitely
  - Once a benchmark becomes popular, tremendous pressure to improve performance by targeted optimizations or by aggressive interpretation of the rules for running the benchmark: "benchmarksmanship."
  - 70 benchmarks from the 5 SPEC releases. 70% were dropped from the next release since no longer useful
- Pitfall: A single point of failure
  - Rule of thumb for fault tolerant systems: make sure that every component was redundant so that no single component failure could bring down the whole system (e.g, power supply)

- Fallacy - Rated MTTF of disks is 1,200,000 hours or ≈ 140 years, so disks practically never fail
- But disk lifetime is 5 years ⇒ replace a disk every 5 years; on average, 28 replacements wouldn't fail
- A better unit: percentage that fail (1.2M MTTF = 833 FIT)
- Fail over lifetime: if had 1000 disks for 5 years
  = 1000*(5*365*24)*833 /10^9 = 36,485,000 / 10^6 = 37
  = 3.7% (37/1000) fail over 5 yr lifetime (1.2M hr MTTF)
- But this is under pristine conditions
  - little vibration, narrow temperature range ⇒ no power failures
- Real world: 3% to 6% of SCSI drives fail per year
  - 3400 - 6800 FIT or 150,000 - 300,000 hour MTTF [Gray & van Ingen 05]
- 3% to 7% of ATA drives fail per year
  - 3400 - 8000 FIT or 125,000 - 300,000 hour MTTF [Gray & van Ingen 05]