EEL 5764: Graduate Computer Architecture

Introduction
Ch 1 - Fundamentals of Computer Design

Ann Gordon-Ross
Electrical and Computer Engineering
University of Florida

http://www.ann.ece.ufl.edu/

These slides are provided by:
David Patterson
Electrical Engineering and Computer Sciences, University of California, Berkeley
Modifications/additions have been made from the originals

EEL 5764

Instructor: Ann Gordon-Ross

Office: 221 Larsen Hall, ann@ece.ufl.edu

Office Hours: MW - 11:45 to 1 (or by appointment only on MW)

Text: *Computer Architecture: A Quantitative Approach, 4th Edition* (Oct, 2006)

Web page: linked from http://www.ann.ece.ufl.edu/

Communication: When sending email, include [EEL5764] in the subject line.

Course Information

- **Prerequisites**
  - Basic UNIX/LINUX OS and compiler knowledge
  - High-level languages and data structures
  - Programming experience with C and/or C++
  - Assembly language
- **Academic Integrity and Collaboration Policy**
  - Homework
  - Project
  - General
- **Reading**
  - Textbook
  - Technical research papers

Course Components

- **Midterms - 60%**
  - 3 midterms
    » One after chapter 3
    » One after chapter 6
    » One on the class presentations/advanced topics
- **Project – 30-40%**
  - Groups of 2-3 students only. No singles. No groups greater than 3
- **Class presentation – EC (5-10%) or 10% of grade**
  - Grad students are now researchers, paper reading is a skill
  - 15 minute presentation on current research topics
  - Presentations as time permits
- **Homework - 0%**
  - I will assign homeworks and it is your responsibility to complete them before the due date (solutions will be provided)
  - Take this seriously! It WILL help you on the midterms

Project - ISS (Part 1)

- **ISS for your own custom assembly language**
  - Reads in program in intermediate format
  - Pipelined (5 stage) and cycle accurate
  - Must deal with data and control hazards
  - Must implement any potential pipeline forwarding and resource sharing (register file) to minimize stall cycles
  - Outputs any computed values in registers or memory to verify functionality
- **Assembler**
  - Input = assembly code
  - Output = intermediate format (opcodes and addresses)
- **Testing**
  - You will need to write applications
  - Matrix multiple, GCD, etc

Project - ISS + Optimization (Part 2)

- **Implement an architectural optimization of your choice**
  - Must be approved by me
  - Technique
    » New idea
    » Take existing idea and improve and/or modify
  - Do research to see what else has been done
    » Choose an area, survey papers
    » Related work section of your final paper
  - Quantify your optimization
    » Choose a metric to show change
      - I.E. CPI, area, power/energy, etc
    » Not graded on how much better your technique is
- **Research paper**
  - Preparation for being a grad student

Project - Grading

- **Part 1**
  - Due Oct 26. Email code, documentation and supporting files
  - Make an appointment to demo what you turned in within the next 3-4 weeks
    » 30 minutes
    » Pass provided test cases and surprise test vectors (same program, different inputs)
    » Provide useful custom benchmarks and pass your test vectors
    » Organization of demo
    » Organization of code including good standard programming principles an sufficient comments /documentation.
  - Log file
    » One for each group member
    » VERY DETAILED

Project – Log File Sample

Name: Bob Davidson
Parter 1: Joe Smith
Parter 2: Jane Doe

Monday, Sept 1 - 4 hours
    Brainstormed on delegation of tasks for the project. Discussed potential instruction set

....

Monday, Sept 8 - 5 hours
    Began writing assembler. Program can parse the input file.
Tuesday, Sept 9 - 3 hours
    Found a bug in the input file parsing. Spent the day debugging it with my partners.
Thursday, Sept 11 - 3 hours
    Team coded with Jane to complete the output generation for the assembler....
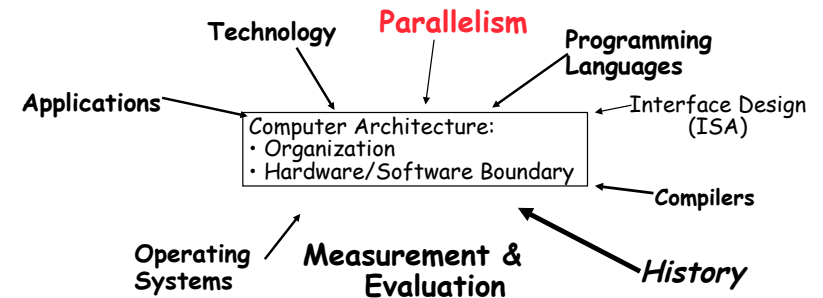
Project - Grading

- **Part 2**
  - **Due Sunday, Dec 7**
  - **What to turn in:**
    - » **All code for assembler and simulator**
    - » **Documentation on how to run your simulator**
    - » **Research paper**
    - » **Log file for each partner**

Course Focus

Understanding the design techniques, machine structures, technology factors, evaluation methods that will determine the form of computers in 21st Century

Outline

- **Classes of Computers**
- **Computer Science at a Crossroads**
- **Computer Architecture v. Instruction Set Arch.**
- **What Computer Architecture brings to table**
- **Technology Trends: Culture of tracking, anticipating and exploiting advances in technology**
- **Careful, quantitative comparisons:**
  1. **Define and quantify cost**
  2. **Define and quantify power**
  3. **Define and quantify dependability**
  4. **Define, quantify , and summarize relative performance**
- **Fallacies and Pitfalls**

Classes of Computers

- **Three main classes of computers**
  - **Desktop Computing**
  - **Servers**
  - **Embedded Computing**
- **Goals and challenges for each class differ**

Classes of Computers

	Price of system	Price of micro-processor module	Critical system design issues
	$500 -$5,000	$50-$500	•Price-performance •Graphics performance
	$5,000 -$5,000,000	$200 -$10,000	•Throughput •Availability/Dependability •Scalability
	$10 -$100,000	$0.01 -$100	•Price •Power consumption •Application-specific performance

Outline

- **Classes of Computers**
- **Computer Science at a Crossroads**
- **Computer Architecture v. Instruction Set Arch.**
- **What Computer Architecture brings to table**
- **Technology Trends: Culture of tracking, anticipating and exploiting advances in technology**
- **Careful, quantitative comparisons:**
  1. **Define and quantify cost**
  2. **Define and quantify power**
  3. **Define and quantify dependability**
  4. **Define, quantify , and summarize relative performance**
- **Fallacies and Pitfalls**

Crossroads: Conventional Wisdom in Comp. Arch

- **Old Conventional Wisdom: Power is free, Transistors expensive**
  - Power increased unabated, but not enough transistors to do everything
  - Clock rate continued to increase
- **New Conventional Wisdom: "Power wall" Power expensive, Xtors free**
  - Surprise!?!? Was it really?
  - Can put more on chip than can afford to turn on
    » More transistors than we can afford to run
  - Can't dissipate enough heat in an air cooled system
- **Old Conventional Wisdom: Sufficiently increasing Instruction Level Parallelism via compilers**
  - SW programmers sat back and code got faster. Compilers, architects did the work
    » I.e. Superscalar, speculative execution, VLIW, out of order execution, etc.
- **New CW: "ILP wall" law of diminishing returns on more HW for ILP**
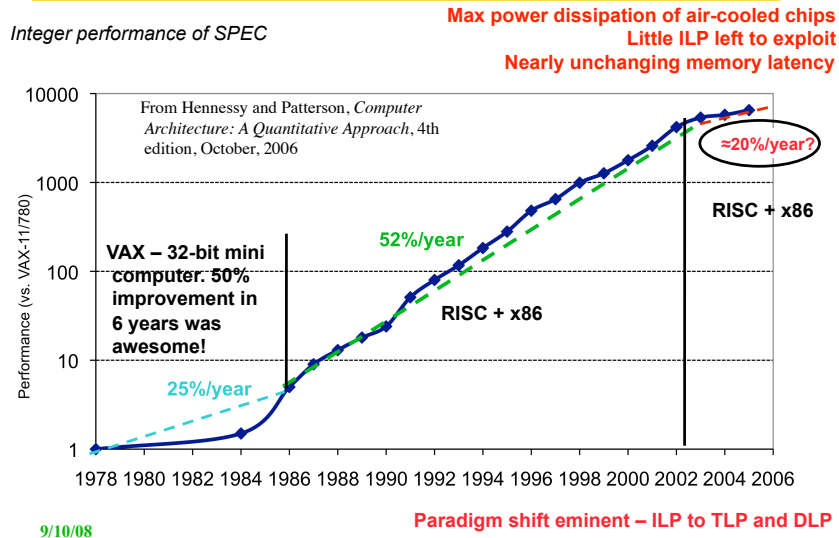
Crossroads: Conventional Wisdom in Comp. Arch

- **Old CW: Multiplies are slow, Memory access is fast**
- **New CW: "Memory wall" Memory slow, multiplies fast (200+ clock cycles to DRAM memory, 4 clocks for multiply (even float))**
- **Old CW: Uniprocessor performance 2X / 1.5 yrs**
- **New CW: Power Wall + ILP Wall + Memory Wall = Brick Wall**
  - Uniprocessor performance now 2X / 5(?) yrs
  - Can't sell based on Mhz anymore
    » Took companies by surprise – Intel cancelled products, wasn't planning for future
- ⇒ **Sea change in chip design: multiple "cores" (2X processors per chip / ~ 2 years)**
    » Can't just crank up clock rate anymore
    » More simpler processors are more power efficient
    » 2 processors with a slower clock and lower voltage is faster
    » Intel now interested in multicore
   » Parallel programming is now a hot topic

Crossroads: Uniprocessor Performance

Integer performance of SPEC

Max power dissipation of air-cooled chips
Little ILP left to exploit
Nearly unchanging memory latency

From Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 4th edition, October, 2006

≈20%/year?

RISC + x86

52%/year

VAX – 32-bit mini computer. 50% improvement in 6 years was awesome!

RISC + x86

25%/year

Performance (vs. VAX-11/780)

10000 — 1000 — 100 — 10 — 1

1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006

Paradigm shift eminent – ILP to TLP and DLP

Sea Change in Chip Design – Raising the Abstraction Level

- **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**
  - **Smaller Caches via DRAM or 1 transistor SRAM (www.t-ram.com) ?**
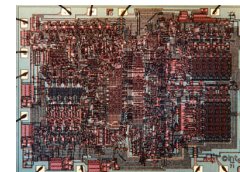
• **Processor is the new transistor?**
• **Can we have the same # of processors as transistors in the first machine?**

Déjà vu all over again?

- **Multiprocessors imminent in 1970s, '80s, '90s, …**
  - **Some progress but not much**
- **Already at the limit? In 1989?**
  - **"… today's processors … are nearing an impasse as technologies approach the speed of light.."**
    - *David Mitchell (startup comp president), The Transputer: The Time Is Now (1989)*
  - **Uniprocessors aren't going to get any faster, need to be multiprocessor now**
- **Transputer was premature**
  ⇒ **Custom multiprocessor strove to lead uniprocessors, but no one wanted to write parallel code**
  ⇒ **Didn't matter. Procrastination rewarded: 2X seq. perf. / 1.5 years**

Déjà vu all over again?

- **"We are dedicating all of our future product development to multicore designs. … This is a sea change in computing"**

  Paul Otellini, President, Intel (2004)

- **So why is this actuality now and not just a claim as before?**
  - **Difference is all microprocessor companies switch to multiprocessors (AMD, Intel, IBM, Sun; Apple)**
  ⇒ **Procrastination penalized: 2X sequential perf. / 5 yrs**
  ⇒ **Biggest programming challenge: 1 to 2 CPUs**

Problems with Sea Change

- **Algorithms, Programming Languages, Compilers, Operating Systems, Architectures, Libraries, … not ready to supply Thread Level Parallelism or Data Level Parallelism for 1000 CPUs / chip**
  - Past efforts were half-hearted, didn't really need to try
- **Architectures not ready for 1000 CPUs / chip**
  - Need a new style of computers
  - Unlike Instruction Level Parallelism, cannot be solved by just by computer architects and compiler writers alone, but also cannot be solved *without* participation of computer architects
- **Field of dreams approach**
  - Ship multi-core, programmers will figure out what to do
- **Computer Architecture: A Quantitative Approach explores shift from Instruction Level Parallelism to Thread Level Parallelism / Data Level Parallelism**

Outline

- Classes of Computers
- Computer Science at a Crossroads
- **Computer Architecture v. Instruction Set Arch.**
- **What Computer Architecture brings to table**
- **Technology Trends: Culture of tracking, anticipating and exploiting advances in technology**
- **Careful, quantitative comparisons:**
  1. Define and quantify cost
  2. Define and quantify power
  3. Define and quantify dependability
  4. Define, quantify , and summarize relative performance
- **Fallacies and Pitfalls**

Instruction Set Architecture: Critical Interface

Instruction set design is not computer architecture

software

instruction set

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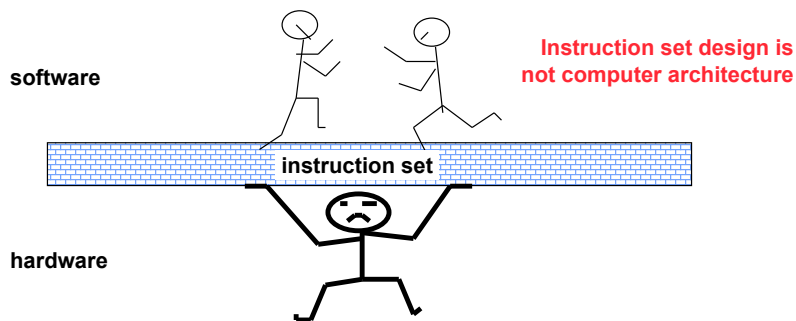
- **Properties of a good IS and good abstraction**
  - Lasts through many generations (portability)
  - Used in many different ways (generality)
  - Provides convenient functionality to higher levels
  - Permits an efficient implementation at lower levels
- **Difficult!**

Example: MIPS

r0   0
r1
°
°
°
r31
PC
lo
hi

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 ?

Format ?

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

Instruction Set Architecture

"... 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, Blaauw, and Brooks, 1964

1 instruction set and many different architectures

-- **Organization of Programmable Storage**

-- **Data Types & Data Structures: Encodings & Representations**

-- **Instruction Formats**

-- **Instruction (or Operation Code) Set**

-- **Modes of Addressing and Accessing Data Items and Instructions**

-- **Exceptional Conditions**

SOFTWARE
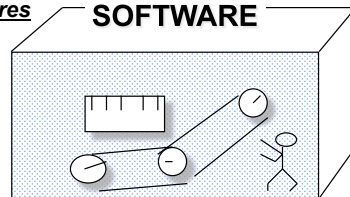
ISA vs. Computer Architecture

- **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**
    - Disagree on conclusion
   - Agree that ISA not where action is (ISA in CA:AQA 4/e appendix)

Comp. Arch. is an Integrated Approach

- **What really matters is the functioning of the complete system**
    - hardware, runtime system, compiler, operating system, and application
    - In networking, this is called the "End to End argument"
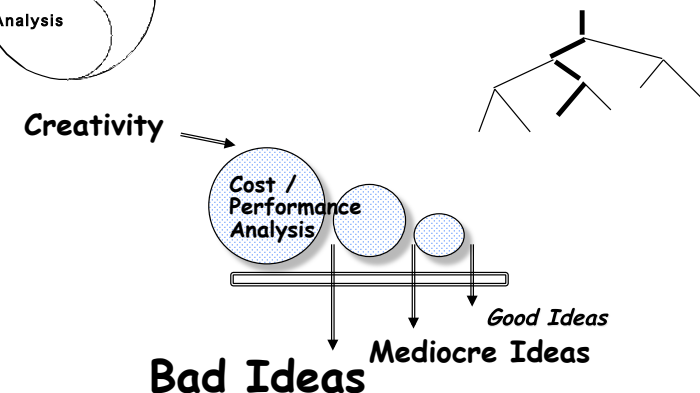- **Computer architecture is not just about transistors, individual instructions, or particular implementations**

Computer Architecture is Design and Analysis

Design

Analysis
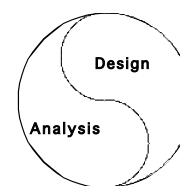
Architecture is an iterative process:
· Searching the space of possible designs
· At all levels of computer systems

Creativity

Cost / Performance Analysis

Bad Ideas

Mediocre Ideas

Good Ideas

Outline

- Classes of Computers Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- **What Computer Architecture brings to table**
- **Technology Trends: Culture of tracking, anticipating and exploiting advances in technology**
- **Careful, quantitative comparisons:**
  1. Define and quantify cost
  2. Define and quantify power
  3. Define and quantify dependability
  4. Define, quantify , and summarize relative performance
- **Fallacies and Pitfalls**

What Computer Architecture brings to Table

- **Other fields often borrow ideas from architecture**
  - **Lots of good ideas are born here**
  - **Google hires architects**
    - » **Data centers = computers, architects bring an understanding and unique skill set**
- **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

What Computer Architecture brings to Table

- **Careful, quantitative comparisons – numbers driven field**
  - **Define, quantity, and summarize relative performance**
  - **Define and quantity relative cost**
  - **Define and quantity dependability**
  - **Define and quantity power**
- **Culture of anticipating and exploiting advances in technology**
  - **Always on the forefront, the cutting edge**
- **Culture of well-defined interfaces that are carefully implemented and thoroughly checked**
  - **HW designers have a tough job and different mindset, must be right the first time, there is no recompilation and downloading an update.**
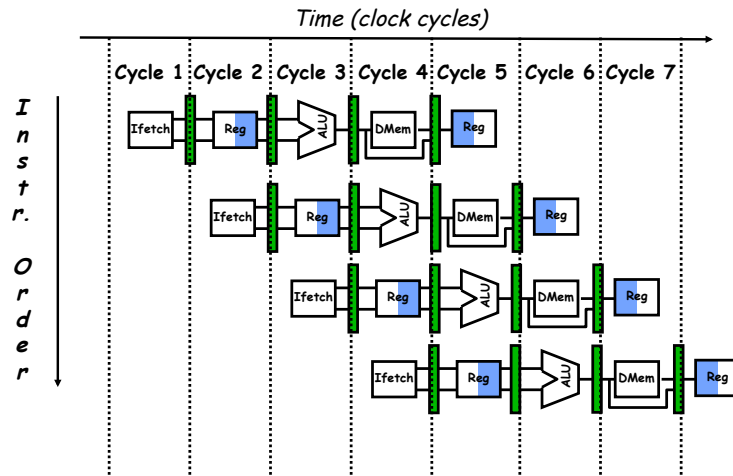 - **RAID**

1) Taking Advantage of Parallelism

- **Increasing throughput of server computer via multiple processors or multiple disks**
  - **Database servers have upwards of 50 disks/CPU**
- **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 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

Time (clock cycles)

| Cycle 1 | Cycle 2 | Cycle 3 | Cycle 4 | Cycle 5 | Cycle 6 | Cycle 7 |

I n s t r.   O r d e r

Ifetch | Reg | ALU | DMem | Reg

Ifetch | Reg | ALU | DMem | Reg

Ifetch | Reg | ALU | DMem | Reg

Ifetch | Reg | ALU | DMem | Reg

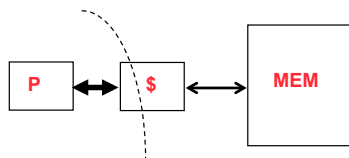## Limits to pipelining - Dependencies

- **Hazards** prevent next instruction from executing during its designated clock cycle
  - <u>Structural hazards</u>: attempt to use the same hardware to do two different things at once
    - » **Early processors had a single cache, now separate level 1 instruction and data caches**
    - » **More likely in X86 CISC – adders used in multiple pipeline stages**
  - <u>Data hazards</u>: Instruction depends on result of prior instruction still in the pipeline
  - <u>Control hazards</u>: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps).
    - » **If you could get rid of branches, the world would be a better place =)**
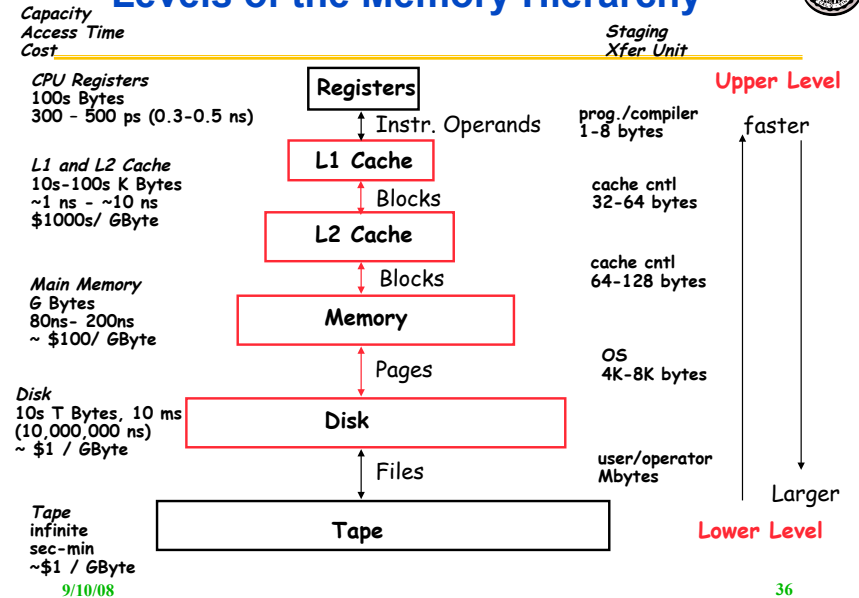
2) The Principle of Locality

- **The Principle of Locality:**
  - **Program access a relatively small portion of the address space at any instant of time.**
- **Two Different Types of Locality:**
  - <u>Temporal Locality</u> (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - <u>Spatial Locality</u> (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, HW relied on locality for memory perf.**

P ↔ **$** ↔ **MEM**

Levels of the Memory Hierarchy

Capacity
Access Time
Cost

| | | *Staging* |
| | | *Xfer Unit* |

CPU Registers
100s Bytes
300 - 500 ps (0.3-0.5 ns)

Registers

↕ Instr. Operands   prog./compiler 1-8 bytes   **Upper Level**   faster

L1 and L2 Cache
10s-100s K Bytes
~1 ns - ~10 ns
$1000s/ GByte

L1 Cache

↕ Blocks   cache cntl 32-64 bytes

L2 Cache

↕ Blocks   cache cntl 64-128 bytes

Main Memory
G Bytes
80ns- 200ns
~ $100/ GByte

Memory

↕ Pages   OS 4K-8K bytes

Disk
10s T Bytes, 10 ms
(10,000,000 ns)
~ $1 / GByte

Disk

↕ Files   user/operator Mbytes   Larger

Tape
infinite
sec-min
~$1 / GByte

Tape

Lower Level

3) 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 1st
  - E.g., If database server has 50 disks / processor, storage dependability dominates system dependability, so optimize it 1st
- **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 for overflow, but overall performance improved by optimizing for the normal case
- **What is frequent case and how much performance improved by making case faster => Amdahl's Law**

4) Amdahl's Law

$$ExTime_{new} = ExTime_{old} \times \left[(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}} \right]$$

$$Speedup_{overall} = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

Best you could ever hope to do:

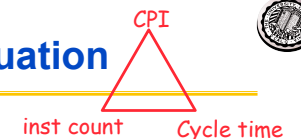$$Speedup_{maximum} = \frac{1}{(1 - Fraction_{enhanced})}$$

Amdahl's Law example

- **New CPU 10X faster**
- **I/O bound server, so 60% time waiting for I/O**

$$Speedup_{overall} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

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

- **Apparently, its human nature (blinded by numbers) to be attracted by 10X faster, vs. keeping in perspective its just 1.6X faster**
- **Why do people gamble??**

5) Processor performance equation

CPI

inst count   Cycle time

Need to consider all of these!! Not just clock rate!

CPU time	= Seconds	= Instructions x	Cycles x	Seconds
	Program	Program	Instruction	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?
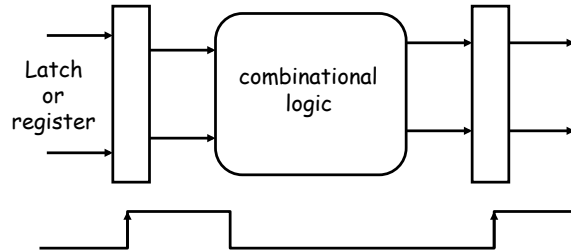
- **Old days:**
  - Gate delays = clock cycle time
  - I.e. 10 levels of gates
- **Today: determined by numerous time-of-flight issues + gate delays**
  - clock propagation, wire lengths, drivers

Outline

- Classes of Computers Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- What Computer Architecture brings to table
- **Technology Trends: Culture of tracking, anticipating and exploiting advances in technology**
- **Careful, quantitative comparisons:**
  1. Define and quantify cost
  2. Define and quantify power
  3. Define and quantify dependability
  4. Define, quantify , and summarize relative performance
- **Fallacies and Pitfalls**

Trends in IC Technology

- **The most important trend in embedded systems - *Moore's Law***
  - Predicted in 1965 by Intel co-founder Gordon Moore (Berkley graduate)
    - » Most predictive articles are just entertainment
  - *IC transistor capacity has doubled roughly every 18 months for the past several decades*
    - » *Driven by both technology and business*
    - » *Large investment in equipment to drive technology*

Moore's Law: 2X transistors / "year"
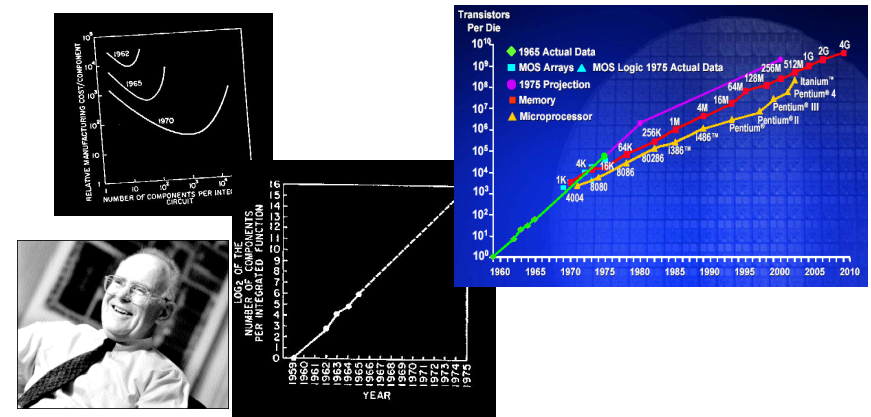
- **"Cramming More Components onto Integrated Circuits"**
  - Gordon Moore, Electronics, 1965
- **# on transistors / cost-effective integrated circuit double every N months (12 ≤ N ≤ 24)**
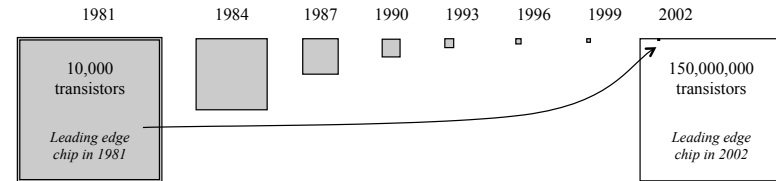
Moore's Law

- **Wow**
  - **This growth rate is hard to imagine, most people underestimate**
  - **How many ancestors do you have from 20 generations ago**
    - » **I.e. roughly how many people alive in the 1500's did it take to make you**
    - » 2^{20} = more than *1 million people*
  - **This underestimation is the key to pyramid schemes!**

Graphical Illustration of Moore's Law

| 1981 | 1984 | 1987 | 1990 | 1993 | 1996 | 1999 | 2002 |

10,000 transistors

Leading edge chip in 1981

150,000,000 transistors

Leading edge chip in 2002

- **Something the doubles frequently grows more quickly than most people realize**
  - **A 2002 chip can hold about 15,000 1981 chips inside itself**

How Do We Track Technology Performance Trends?

- **Drill down into 4 technologies:**
  - **Disks,**
  - **Memory,**
  - **Network,**
  - **Processors**
- **Compare ~1980 Archaic (Nostalgic) vs. ~2000 Modern (Newfangled)**
  - **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**

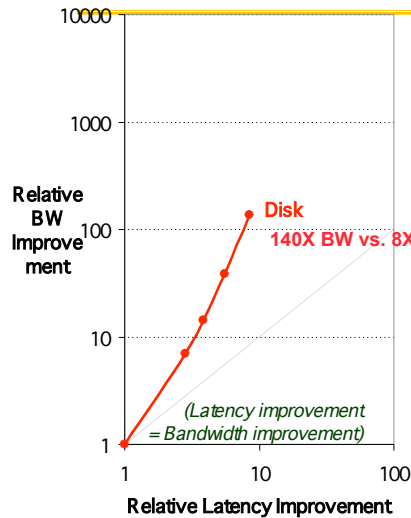## Disks: Archaic(Nostalgic) v. Modern(Newfangled)

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**

(Latency improvement = Bandwidth improvement)

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

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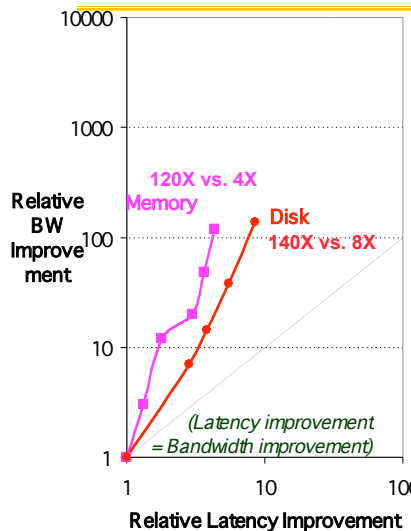

Memory: Archaic (Nostalgic) v. Modern (Newfangled)

• 1980 DRAM (asynchronous)	• 2000 Double Data Rate Synchr. (clocked) DRAM
• 0.06 Mbits/chip	• 256.00 Mbits/chip (4000X)
• 64,000 xtors, 35 mm^2	• 256,000,000 xtors, 204 mm^2
• 16-bit data bus per module, 16 pins/chip	• 64-bit data bus per DIMM, 66 pins/chip (4X)
• 13 Mbytes/sec	• 1600 Mbytes/sec (120X)
• Latency: 225 ns	• Latency: 52 ns (4X)
• (no block transfer)	• 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)

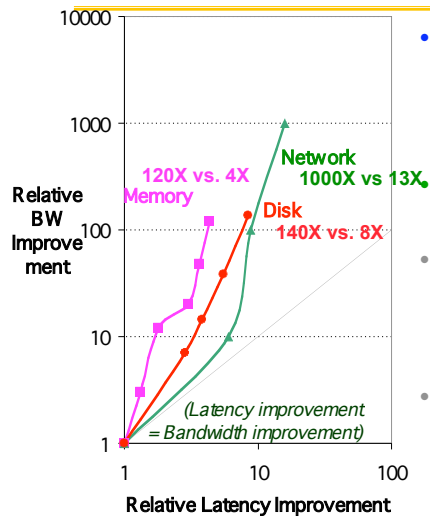|	
• Ethernet 802.3	• Ethernet 802.3ae
• Year of Standard: 1978	• Year of Standard: 2003
• 10 Mbits/s link speed	• 10,000 Mbits/s link speed (1000X)
• Latency: 3000 μsec	• Latency: 190 μsec (15X)
• Shared media	• Switched media
• Coaxial cable	• Category 5 copper wire

Coaxial Cable:

Plastic Covering
Braided outer conductor
Insulator
Copper core
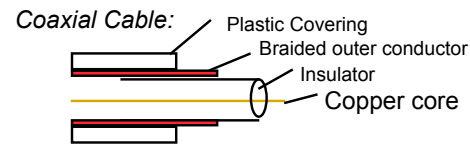
"Cat 5" is 4 twisted pairs in bundle
Twisted Pair:

Copper, 1mm thick,

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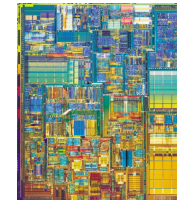
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

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

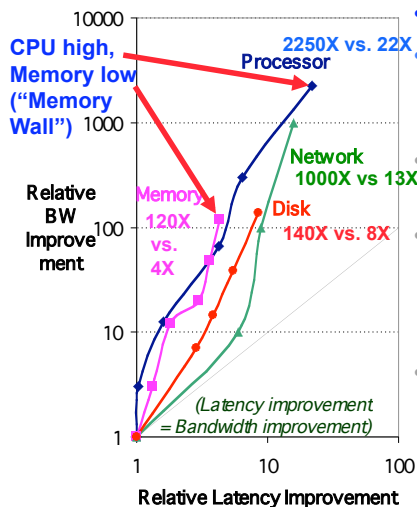## CPUs: Archaic (Nostalgic) v. Modern (Newfangled)

• **1982 Intel 80286**	• **2001 Intel Pentium 4**
• **12.5 MHz**	• **1500 MHz** (120X)
• **2 MIPS (peak)**	• **4500 MIPS (peak)** (2250X)
• **Latency (inst) 320 ns**	• **Latency 15 ns** (20X)
• **134,000 xtors, 47 mm²**	• **42,000,000 xtors, 217 mm²**
• **16-bit data bus, 68 pins**	• **64-bit data bus, 423 pins**
• **Microcode interpreter, separate FPU chip**	• **3-way superscalar, Dynamic translate to RISC, Superpipelined (22 stage), Out-of-Order execution**
• **(no caches)**	• **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, smaller transistors more pins help Bandwidth**
  - » **MPU Transistors:** **0.130 vs. 42 M xtors** **(300X)**
  - » **DRAM Transistors:** **0.064 vs. 256 M xtors** **(4000X)**
  - » **MPU Pins:** **68 vs. 423 pins** **(6X)**
  - » **DRAM Pins:** **16 vs. 66 pins** **(4X)**
- **Smaller, faster transistors but communicate over (relatively) longer lines (dies are bigger): limits latency**
  - » **Feature size:** **1.5 to 3 vs. 0.18 micron** **(8X,17X)**
  - » **MPU Die Size:** **35 vs. 204 mm^2** **(ratio sqrt ⇒ 2X)**
  - » **DRAM Die Size:** **47 vs. 217 mm^2** **(ratio sqrt ⇒ 2X)**

6 Reasons Latency Lags Bandwidth (cont'd)

2. Distance limits latency

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

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

- **Numbers game**
- **E.g., 10 Gbits/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 bandwidth and rotational latency**
  - » **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 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**

6 Reasons Latency Lags Bandwidth (cont'd)

5. Bandwidth hurts latency

- **Queues help Bandwidth, hurt Latency (Queuing Theory)**
- **Wider helps BW, hurts latency**
  - **Adding chips to widen a memory module increases Bandwidth but higher fan-out on address lines may increase Latency**

6. Operating System (SW) overhead hurts Latency more than Bandwidth

- **Longer messages amortize overhead (overhead bigger part of short messages)**

Summary of Technology Trends

- **For disk, LAN, memory, and microprocessor, bandwidth improves by square of latency improvement**
  - **In the time that bandwidth doubles, latency improves by no more than 1.2X to 1.4X**

- **Lag probably even larger in real systems, as bandwidth gains multiplied by replicated components**
  - Multiple processors in a cluster or even in a chip
  - Multiple disks in a disk array
  - Multiple memory modules in a large memory
  - Simultaneous communication in switched LAN

- **HW and SW developers should innovate assuming Latency Lags Bandwidth**
  - If everything improves at the same rate, then nothing really changes
  - When rates vary, require real innovation

Outline

- Classes of Computers Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- What Computer Architecture brings to table
- Technology Trends: Culture of tracking, anticipating and exploiting advances in technology
- **Careful, quantitative comparisons:**
  1. Define and quantify cost
  2. Define and quantify power
  3. Define and quantify dependability
  4. Define, quantify , and summarize relative performance
- **Fallacies and Pitfalls**

Define and quantify cost (1/2)

- 3 factors lower costs:
1. **Learning curve** - manufacturing costs decrease over time measured by change in yield
   - Yield = % manufactured devices that survives the testing procedure
   - Yield increases over time = more efficient process over time
2. **Volume** - double volume cuts cost 10%
   - Decrease time to get down the learning curve
   - Increases purchasing and manufacturing efficiency
   - Amortizes development (NRE) costs over more devices
3. **Commodities** = Competition
   - Produces sold by multiple vendors in large values are essentially identical
   - E.g.; Keyboards, monitors, DRAMs, disks, PCs
     - IBM invented many, then sold off
- Most of computer cost in integrated circuit
  - Cost of producing chips
  - Die cost + packaging cost + testing cost

Define and quantify cost (2/2)

- **Profit Margin** = **Price product sells - cost to manufacture**

- **Margins pay for research and development (R&D), marketing, sales, manufacturing equipment, maintenance, building rental, cost of financing, pretax profits, and taxes**

- **Most companies spend 4% (commodity PC business) to 12% (high-end server business) of income on R&D, which includes all engineering**

Outline

- Classes of Computers Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- What Computer Architecture brings to table
- Technology Trends: Culture of tracking, anticipating and exploiting advances in technology
- **Careful, quantitative comparisons:**
  1. Define and quantify cost
  2. Define and quantify power
  3. Define and quantify dependability
  4. Define, quantify , and summarize relative performance
- **Fallacies and Pitfalls**

Define and quantity power (1 / 2)

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

$$Power_{dynamic} = 1/2 \times CapacitiveLoad \times Voltage^2 \times FrequencySwitched$$

- **For mobile devices, energy better metric**

$$Energy_{dynamic} = CapacitiveLoad \times Voltage^2$$

- **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**
- **For a fixed task, slowing clock rate (frequency switched) reduces power, but not energy**
- **To save energy & dynamic power, most CPUs now turn off clock of inactive modules (e.g. Fl. Pt. Unit)**

Example of quantifying power

- **Suppose 15% reduction in voltage results in a 15% reduction in frequency. What is impact on dynamic power?**

$$Power_{dynamic} = 1/2 \times CapacitiveLoad \times Voltage^2 \times FrequencySwitched$$
$$= 1/2 \times .85 \times CapacitiveLoad \times (.85 \times Voltage)^2 \times FrequencySwitched$$
$$= (.85)^3 \times OldPower_{dynamic}$$
$$\approx 0.6 \times OldPower_{dynamic}$$

- **40% reduction in power**
- **Hence 2 simpler (lower capacitance), slower cores (lower frequency) could replace 1 complex core for same power per chip**

Define and quantity power (2 / 2)

- **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**

Outline

- Classes of Computers Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- What Computer Architecture brings to table
- Technology Trends: Culture of tracking, anticipating and exploiting advances in technology
- **Careful, quantitative comparisons:**
    1. Define and quantify cost
    2. Define and quantify power
    3. **Define and quantify dependability**
    4. **Define, quantify , and summarize relative performance**
- **Fallacies and Pitfalls**

Define and quantity dependability (1/3)

- **How to decide when a system is operating properly?**
- **Infrastructure providers now offer Service Level Agreements (SLA) to guarantee that their networking or power service would be dependable**
    - If call short, money for compensation
- **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**
- **Repair = transition from state 2 to state 1**

Define and quantity dependability (2/3)

- *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*) **= 1/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 (% time available) = MTTF / (MTTF + MTTR)*

Example calculating reliability

- **If modules have** *exponentially distributed lifetimes* **(age of module does not affect probability of failure), overall failure rate is the sum of failure rates of the modules**
- **Calculate FIT (per billion hours) and MTTF for 10 disks (1M hour MTTF per disk), 1 disk controller (0.5M hour MTTF), and 1 power supply (0.2M hour MTTF):**

$$FailureRate = 10 \times (1/1,000,000) + 1/500,000 + 1/200,000$$
$$= (10 + 2 + 5)/1,000,000$$
$$= 17/1,000,000$$
$$= 17,000 \, FIT$$
$$MTTF = 1,000,000,000/17,000$$
$$\approx 59,000 \, hours$$

But doesn't consider infant mortality or likelihood of being thrown away before dead

Focus on common case

- Power supply MTTF limits system MTTF
- What if added redundant power supply, so system still works if one fails?
- MTTF of pair is now mean time until one power supply fails divided by chance of other will fail before 1st is replaced
- Since 2 power supplies and independent failures, mean time to one power supply fails is MTTF$_{powersupply}$/2

$$MTTF_{pairps} = \frac{MTTF_{ps}/2}{\dfrac{MTTR_{ps}}{MTTF_{ps}}} = \frac{MTTF^2_{ps}/2}{MTTR_{ps}} = \frac{MTTF_{ps}^2}{2*MTTR_{ps}}$$

Example recalculating reliability

- Calculate FIT and MTTF for 10 disks (1M hour MTTF per disk), 1 disk controller (0.5M hour MTTF), 2 power supplies (0.2 M hour MTTF), and MTTR for replacing a failed power supply is 1 day. How much better is MTTF$_{pair}$? MTTF$_{system}$?

$$MTTF_{pair} = \frac{200,000^2}{2*24} = 830,000,000$$

$$FailureRate = \frac{10}{1,000,000} + \frac{1}{5,000,000} + \frac{1}{830,000,000}$$

$$= \frac{10+2+0}{1,000,000} = \frac{12}{1,000,000} = 12,000\,FIT$$

$$MTTF = \frac{1,000,000,000}{12,000} = 83,000\,hours$$

- MTTF$_{pair}$ 4200x; MTTF $_{system}$ is 1.4x; Amdahl's Law!

Outline

- Classes of Computers Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- What Computer Architecture brings to table
- Technology Trends: Culture of tracking, anticipating and exploiting advances in technology
- Careful, quantitative comparisons:
  1. Define and quantify cost
  2. Define and quantify power
  3. Define and quantify dependability
  4. Define, quantify , and summarize relative performance
- Fallacies and Pitfalls

Definition: Performance

- What does it mean to be 10% faster or slower?
- Performance is in units of things per sec
  - bigger is better
- If we are primarily concerned with response time

$$performance(x) = \frac{1}{execution\_time(x)}$$

" X is n times faster than Y" means

$$n = \frac{Performance(X)}{Performance(Y)} = \frac{Execution\_time(Y)}{Execution\_time(X)}$$

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 pgms
  - SPECCPU2006
  - 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

Outline

- Classes of Computers Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- What Computer Architecture brings to table
- Technology Trends: Culture of tracking, anticipating and exploiting advances in technology
- Careful, quantitative comparisons:
  1. Define and quantify cost
  2. Define and quantify power
  3. Define and quantify dependability
  4. Define, quantify , and summarize relative performance
- **Fallacies and Pitfalls**

Fallacies and Pitfalls (1/2)

- **Fallacies - commonly held misconceptions**
  - When discussing a fallacy, will 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."
    » Tune to benchmark
  - 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)**

Fallacies and Pitfalls (2/2)

- **Fallacy - Rated MTTF of disks is 1,200,000 hours or ≈ 140 years, so disks practically never fail**
- **And disk lifetime is 5 years ⇒ By 28 disks, replace every 5 years; on average, 28 replacements won't fail**
- **A better unit: % 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]