



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*



Course Information

- **Prerequisites**
 - Basic UNIX/LINUX OS and compiler knowledge
 - High-level languages and data structures
 - Programming experience with C and/or C++
 - » Extent required depends on project that you choose
 - Assembly language
- **Academic Integrity and Collaboration Policy**
 - Homework/Project
 - » Individual or groups of 2 or 3
 - » One homework/project submission per group
 - General
 - » Must do original work (no use of web material, even if sited)
 - Severe consequences!
- **Reading**

8/31/10 Textbook, technical research papers

3



EEL 5764

- Instructor:** Ann Gordon-Ross
Office: 221 Larsen Hall, ann@ece.ufl.edu
Office Hours: TBA
Answer questions between periods on Tuesdays
Different classrooms – Tuesday NEB 201
Thursday NEB 102
- Text:** *Computer Architecture: A Quantitative Approach, 4th Edition (Oct, 2006)*
- Web page:** linked from <http://www.ann.ece.ufl.edu/>
Sakai – lecture videos, project/homework turn in
- Communication:**
Discussion groups on Sakai
TA – Zubin Kumar (zubinkumar@ufl.edu)
When sending email, include [EEL5764] in the subject line.

8/31/10

2



Course Components

- **Midterms - 50%**
 - 2 midterms
 - » After chapter 3 and after chapter 6 (not cumulative) - Tuesdays
 - » Edge students – 3 day window (Monday-Wednesday)
- **Project – 40%**
 - Individual or groups of 2 or 3
 - Four graded components
- **Homework - 10%**
 - 4 anticipated
 - Work with same group as project, turn in one copy with all names
 - Spot graded, perhaps
 - Answers are likely on the web, but take this seriously! It WILL help you on the midterms.
- **Other (non-graded) components**
 - Research paper reading
 - » Grad students are now researchers, paper reading is a skill
 - Presentation for top course projects if time permits

8/31/10 Presentation for top course projects if time permits

4



Course Project - Overview

- Open ended computer architecture-based project
- Topic of your choice
- Learn first steps of research and tool usage
 - Read scholarly articles to get ideas
 - » In computer science and engineering based fields, conferences are MORE IMPORTANT than journals
 - » Survey of topic, what has been done, what needs to be done
 - » Textbook contains many references to top papers
 - Tools
 - » Use existing tool – always try this option first!
 - » Create custom tool
- Team effort
 - Individually or in groups of 2 or 3

8/31/10

5



Course Project – Components

- Four graded components:
 - Group member selection – Tuesday, Sept 7 @ 5pm (5%)
 - » Can't change once selected
 - » Group size does not affect work expectations or grading
 - Project proposal – Tuesday, Sept 28 @ 5pm (20%)
 - Status update – Tuesday, Nov 2 @ 5pm (5%)
 - » With log files
 - Final report – Tuesday, Nov 30 @ 5pm (70%)
 - » With log files
- All components turned in via Sakai
 - ***NO LATE WORK WILL BE ACCEPTED! YES, THIS MEANS A 0!***
 - Turn in early, turn in often.
- Optional component, if time permits
 - Top projects will be invited to give a 15 minute in-class presentation

8/31/10

6



Course Project – Tool Set

- Implement using SimpleScalar
 - Extensively used architectural simulator for over a decade
 - Many versions (e.g., sim-cache, sim-cheetah, sim-fast, etc.), must use sim-outoforder
 - Very little information provided on course webpage
 - Plethora of information available via Google – Learn how to teach yourself!
 - Linux based, can run over Cygwin or virtual machine
 - Steps:
 - » Read documentation
 - » Install tool suite, execute sample benchmarks
 - » Install cross compiler, compile benchmarks
- SimpleScalar not originally for multi-core
 - Multi-core version out there
 - Other simulators available
 - Can get permission to use other simulator, must ask me first

8/31/10

7



Course Project – Possible Topics

- Implement a new research idea or verify an existing research idea
 - Urged to implement new idea
 - » Could lead to publication
 - » Hardest part about research
 - Existing idea
 - » Still learn the fundamentals of research
 - » Verification is an important part of research
 - Some people knowingly report false data, highly unethical, PhDs have been revoked
 - Grade will not be based on topic idea
 - » Won't receive extra credit for new idea
 - » Won't be graded down for existing idea
- Bottom line – Choose a topic that interests you!!

8/31/10

8

Course Project – Possible Topics



- Code scheduling for ILP
- Instruction/data EncodingCache-based enhancements (e.g., trace cache, filter cache, loop cache, victim cache, stream buffers, etc.)
- Pipeline clocking
- Low power architectures
- Quantifying architectural characteristics of database workloads and comparing them to other workloads
- Achieve fault tolerance by running 2 copies of instructions in unused cycles in a superscalar (e.g., a 4-way machine may commit less than 4 instructions due to dependencies) and do instruction replication only in those cycles.

8/31/10

9

Course Project – Possible Topics



- Evaluate cache behavior of networking (or other) applications or algorithms, with modification to exploit caches and memory hierarchies.
- Etc....

8/31/10

11

Course Project – Possible Topics



- Compare Qureshi and Patt's insertion policies in ISCA 2007 to victim caches
- Use old register values to predict addresses of subsequent memory accesses.
- Attempt to quantify how much of processor performance gain in the past decade has come from faster clocks and how much from ILP.
- Implement and compare victim caches and skewed-associative caches.
- Implement and compare two recent prefetching schemes. Study prefetching methods (hardware and/or software) and their impact on performance

8/31/10

10

Course Project – Project Proposal



- 1-2 page document describing proposed work
 - See course project webpage for detailed content requirements
 - Format like a scholarly publication
 - Main points:
 - » Identify topic
 - » Why is the topic important
 - » Initial previous work survey
 - » Implementation plan (experimental setup)
 - » GANTT chart planning out work for each member
 - » Expected results
- Must get approval from me
 - May need to modify proposal
 - Turn in early to get early feedback

8/31/10

12

Course Project – Status Update

- Informal document outlining status/progress of the project
 - Not graded on content or progress, just graded on the fact that this update is turned in
 - Describe any deviations from proposed GANTT timeline
 - » Revise GANTT chart if necessary
 - » Can't change final goals, can only revise intermediate steps based on slower/faster than expected progress
 - Preliminary results
 - Log files

8/31/10

13

Course Project – Final Report

- 6 page final project report
- IEEE or ACM formatting
- See course project webpage for detailed content requirements
- Main points:
 - Abstract
 - Related work
 - Methodology
 - Experimental results
- Common pitfalls
 - Assume the readers already knows everything
 - Insufficient related work
 - Insufficient results analysis

8/31/10

14

Project – Log File Sample

Name: Bob Davidson (percentage effort - 40%)
Partner 1: Joe Smith (percentage effort - 20%)
Partner 2: Jane Doe (percentage effort - 40%)

Monday, Sept 1 - 4 hours

Brainstormed on delegation of tasks for the project. Discussed potential instruction set modifications

....

Monday, Sept 8 - 5 hours

Installed Cygwin twice. Read SimpleScalar documentation while installing

Tuesday, Sept 9 - 3 hours

Installed SimpleScalar for Pisa. Ran test benchmarks

Thursday, Sept 11 - 3 hours

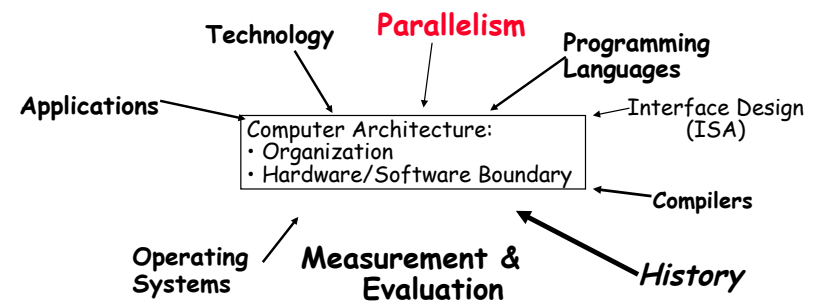
Installed cross-compiler. Started to look at SimpleScalar code structure

8/31/10

15

Course Focus

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



8/31/10

16



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**

8/31/10

17



Classes of Computers




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

8/31/10

18



Classes of Computers

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

8/31/10

19



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**

8/31/10

20



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?
 - More transistors than we can afford to run
 - » Can’t dissipate enough heat in an air cooled system
 - Clock rate not increasing anymore. Per-core clock rates lower.
- **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**

21



Crossroads: Conventional Wisdom in Comp. Arch

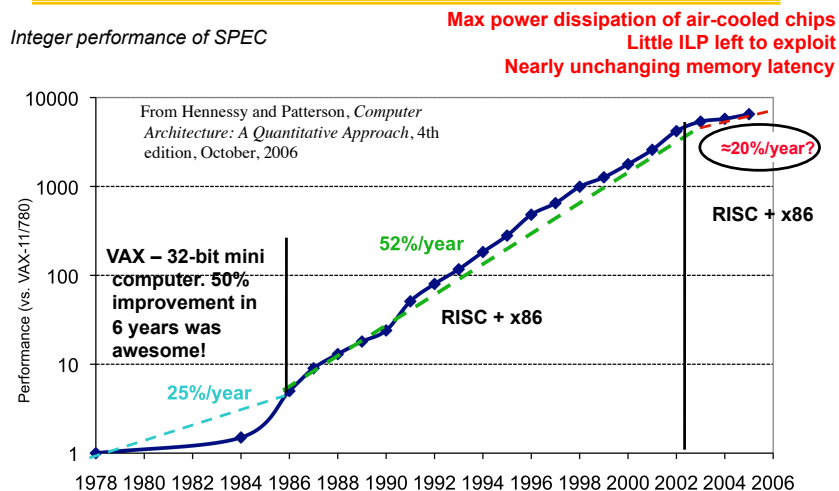
- **Old CW: Multiplies are slow, Memory access is fast**
 - **New CW: “Memory wall” Memory slow, multiplies fast (200+ clock cycles to DRAM memory, 4 clocks for multiply (even float))**
 - **Old CW: Uniprocessor performance 2X / 1.5 yrs**
 - **New CW: Power Wall + ILP Wall + Memory Wall = Brick Wall**
 - 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, again!

8/31/10

22



Crossroads: Uniprocessor Performance



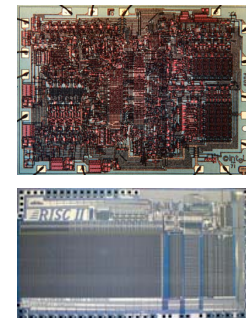
8/31/10

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**
- **2006 - 125 mm² chip, 0.065 micron CMOS**
 - Equivalent of 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?**

8/31/10

24

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
 - ⇒ Why make the effort? Didn't matter.
 - Procrastination rewarded: 2X seq. perf. / 1.5 years

8/31/10

25

Déjà vu all over again?

- Now multi-processing is not a claim, it is actually happening !
- “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 to 200 CPUs!

8/31/10

26

Problems with Change



- **We are not ready!**
 - 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 just by computer architects and compiler writers alone, but also cannot be solved *without* participation of computer architects
- **Field of dreams approach**
 - Ship them (multi-core) and they (programmers) will use them well
 - E.g. Sony's Cell
- **Computer Architecture: A Quantitative Approach** explores shift from Instruction Level Parallelism to Thread Level Parallelism / Data Level Parallelism

8/31/10

27

Outline

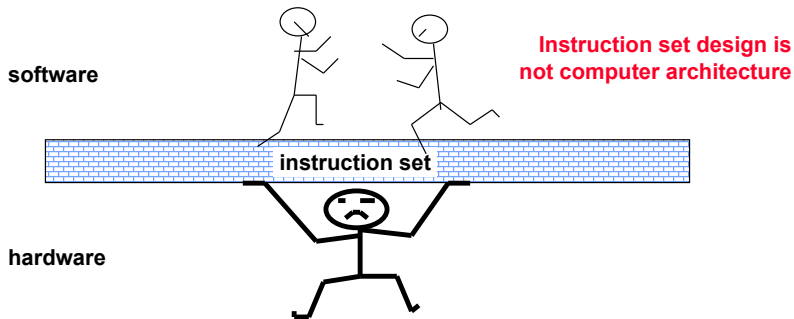


- Classes of Computers
- Computer Science at a Crossroads
- Computer Architecture v. Instruction Set Arch.
- What Computer Architecture brings to table
- Technology Trends: Culture of tracking, anticipating and exploiting advances in technology
- Careful, quantitative comparisons:
 1. Define and quantify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

8/31/10

28

Instruction Set Architecture: Critical Interface

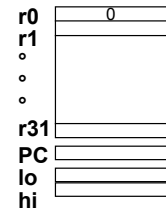


- Properties of a good instruction set architecture and good abstraction
 - Portable - lasts through many generations
 - General - used in many different ways
 - Provides **convenient** functionality to higher levels
 - Permits an **efficient** implementation at lower levels

4/3 Difficult!

29

Example: 32-bit MIPS Instruction Set



Program storage: 32-bit instructions on word boundary
 2³² bytes
 31 32-bit General Purpose Registers (GPRs) (R0=0)
 32 32-bit Floating Point (FP) regs (paired for double precision (DP))
 3 Special Purpose Registers (SPRs) - HI, LO, PC

MIPS is:

- Elegant and simple
- Widely used in embedded systems and gaming consoles
- Basis for countless instruction sets (i.e., MIPS-like)
- Basis for official ISA of China

Instruction types:

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

8/31/10

30

Instruction Set Architecture

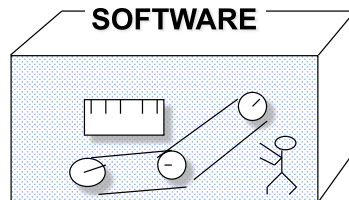
“... the attributes of a [computing] system as seen by the programmer (i.e., the conceptual structure and functional behavior), as distinct from the organization of the data flows and controls the logic design, and the physical implementation.” - Amdahl, Blaauw, Brooks - 1964

Bold claim - 1 instruction set with many different architectures

ISA Defines:

- 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
- Exception Conditions

Hardware is free to provide support any way



Moore's Law

8/31/10

31

ISA vs. Computer Architecture

Old definition of computer architecture was simply ISA design

- Other aspects of computer design (hardware) were called *implementation*
- Insinuates that implementation is uninteresting or less challenging
- But really, ISA design is not very exciting

Our view is computer architecture >> ISA

Architect's job is much more than ISA design alone; technical hurdles today are more challenging than those in ISA design

Since ISA design was not where the action was, some conclude that computer architecture (using old definition) is not where the action is

- Disagree on conclusion
- But agree that ISA design is not where the action is

8/31/10

32

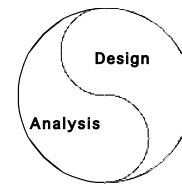
Comp. Arch. is an Integrated Approach

- What really matters is the functioning of the complete system
 - hardware, runtime system, compiler, operating system, and application
 - All parts working together to deliver a good design
 - In networking, this is called the “End-to-End argument”
- Computer architecture is not just about transistors, individual instructions, or particular implementations

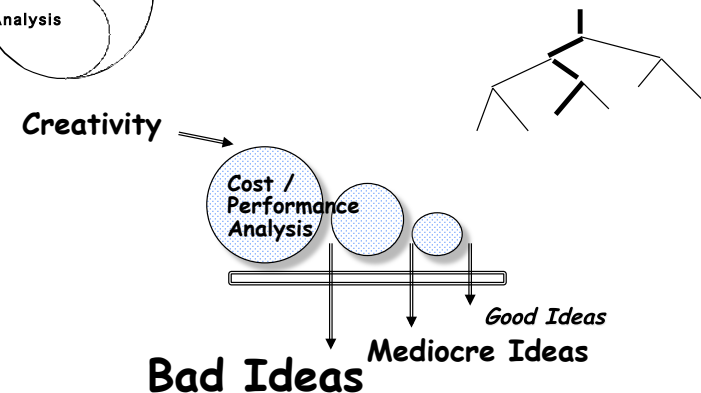
8/31/10

33

Computer Architecture is Design and Analysis



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



8/31/10

34

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

8/31/10

35

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
- Why? *Architects Know the 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

8/31/10

36

What Computer Architecture brings to Table

- **Careful, quantitative comparisons – numbers driven field**
 - Define, quantify, and summarize relative performance
 - Define and quantify relative cost
 - Define and quantify dependability
 - Define and quantify 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

8/31/10

37

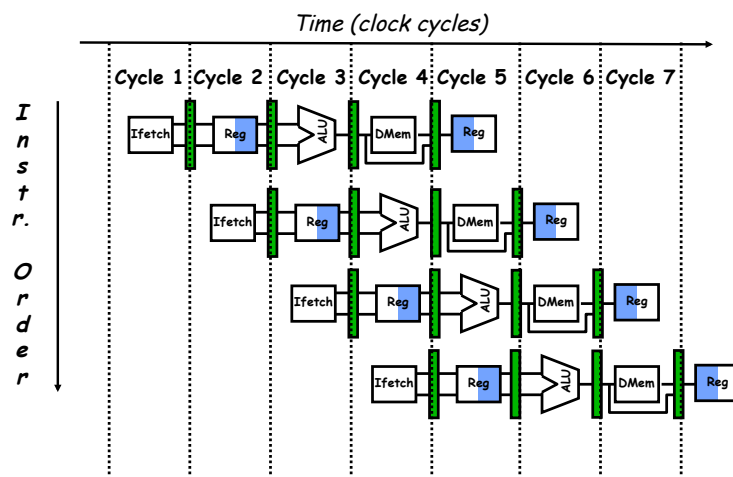
1) Taking Advantage of Parallelism

- **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)

8/31/10

38

Pipelined Instruction Execution



8/31/10

39

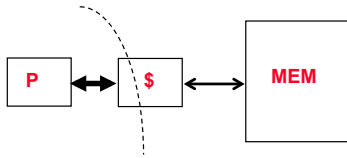
Limits to pipelining - Dependencies

- **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
 - » 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
 - **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).
 - » If you could get rid of branches, the world would be a better place =)

8/31/10

2) The Principle of Locality

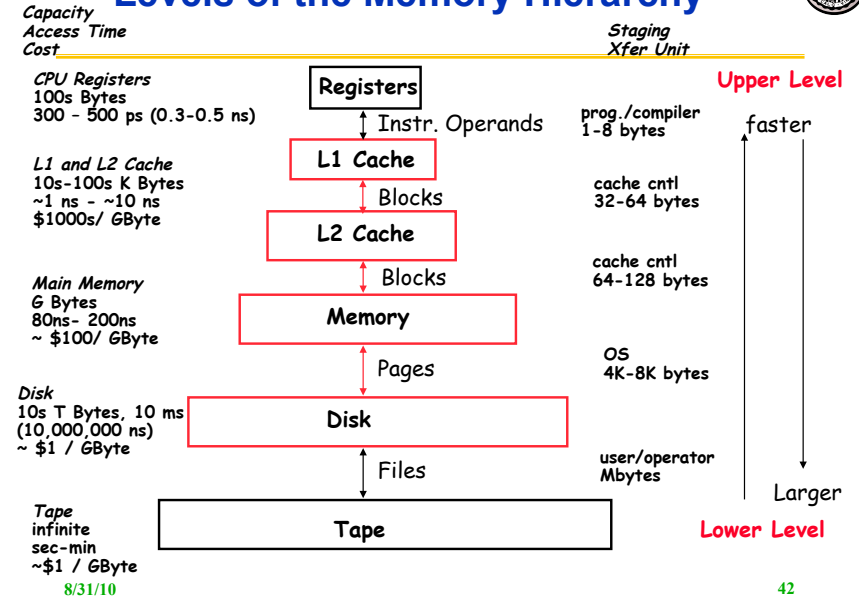
- **The Principle of Locality:**
 - Program access a relatively small portion of the address space at any instant of time.
- **Two Different Types of Locality:**
 - **Temporal Locality** (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
 - **Spatial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



8/31/10

41

Levels of the Memory Hierarchy



8/31/10

42

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**

8/31/10

43

4) Amdahl's Law

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

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

Best you could ever hope to do:

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



8/31/10

44

Amdahl's Law example

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

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

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

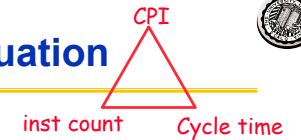
- Apparently, its human nature (blinded by numbers) to be attracted by 10X faster, vs. keeping in perspective its just 1.6X faster
- Why do people gamble??

8/31/10

45

5) Processor performance equation

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

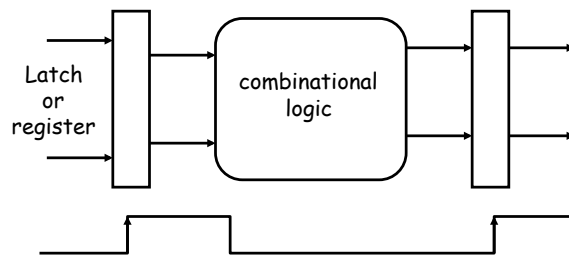


CPU time	= Seconds Program	= Instructions Program	x Cycles Instruction	x Seconds Cycle
	Inst Count	CPI	Clock Rate	
Program	X			
Compiler	X	(X)		
Inst. Set.	X	X		
Organization		X	X	
Technology			X	

8/31/10

46

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

8/31/10

47

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

8/31/10

48

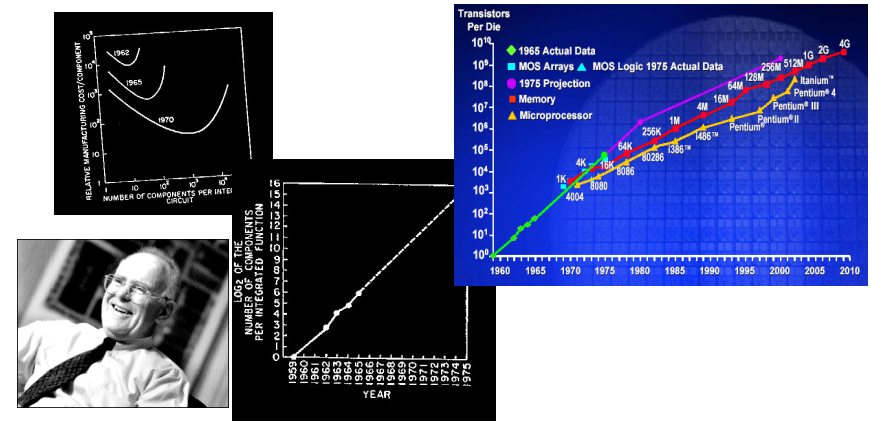
Trends in IC Technology

- The most important trend in embedded systems - **Moore's Law**
 - Predicted in 1965 by Intel co-founder Gordon Moore (Berkeley 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**

8/31/10

49

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 ($12 \leq N \leq 24$)

8/31/10

CS252-s06, Lec 02-int#0

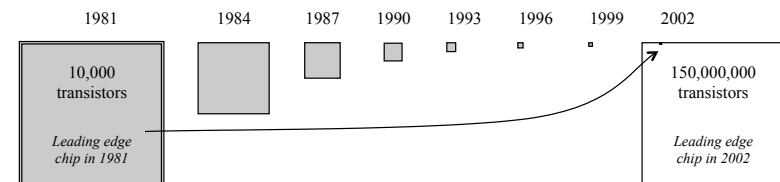
Moore's Law

- **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!

8/31/10

51

Graphical Illustration of Moore's Law



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

8/31/10

52

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

8/31/10

53

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

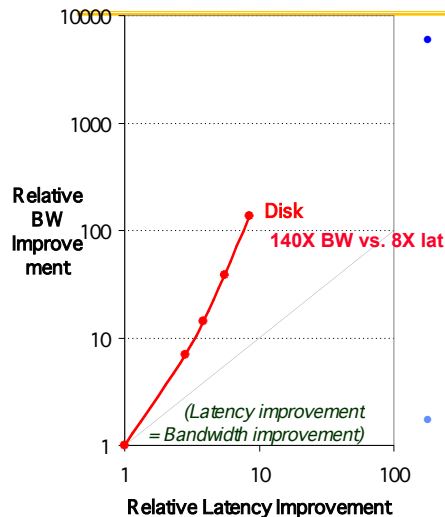


- | | |
|-----------------------------|--|
| • CDC Wren I, 1983 | • Seagate 373453, 2003 |
| • 3600 RPM | • 15000 RPM (4X) |
| • 0.03 GBytes capacity | • 73.4 GBytes (2500X) |
| • Tracks/Inch: 800 | • Tracks/Inch: 64000 (80X) |
| • Bits/Inch: 9550 | • Bits/Inch: 533,000 (60X) |
| • Three 5.25" platters | • 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 |

8/31/10

54

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)

8/31/10

55

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

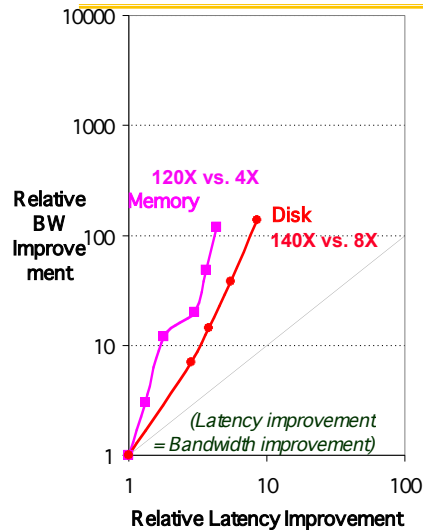


- | | |
|--|--|
| • 1980 DRAM (asynchronous) | • 2000 Double Data Rate Synchr. (clocked) DRAM |
| • 0.06 Mbits/chip | • 256.00 Mbits/chip (4000X) |
| • 64,000 xtors, 35 mm ² | • 256,000,000 xtors, 204 mm ² |
| • 16-bit data bus per module, 16 pins/chip | • 64-bit data bus per DIMM, 66 pins/chip (4X)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

8/31/10

56

Latency Lags Bandwidth (last ~20 years)



8/31/10

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)

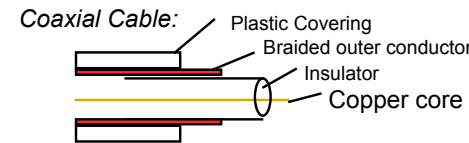
(latency = simple operation w/o contention
BW = best-case)

57

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



- | | |
|---|---|
| <ul style="list-style-type: none"> • Ethernet 802.3 • Year of Standard: 1978 • 10 Mbits/s link speed • Latency: 3000 μsec • Shared media • Coaxial cable | <ul style="list-style-type: none"> • Ethernet 802.3ae • Year of Standard: 2003 • 10,000 Mbits/s link speed (1000X) • Latency: 190 μsec (15X) • Switched media • Category 5 copper wire |
|---|---|



"Cat 5" is 4 twisted pairs in bundle
Twisted Pair:

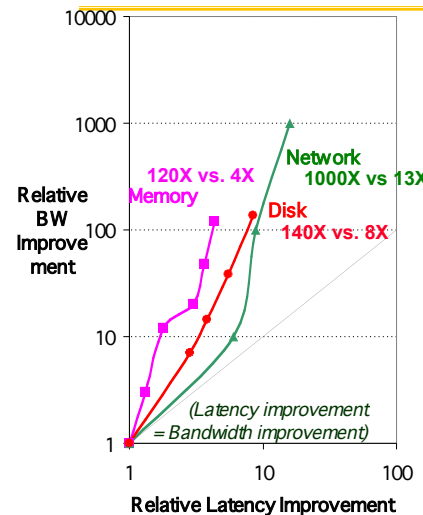


Copper, 1mm thick,

8/31/10

58

Latency Lags Bandwidth (last ~20 years)



8/31/10

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)

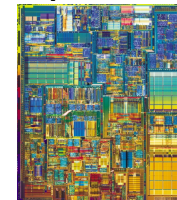
(latency = simple operation w/o contention
BW = best-case)

59

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



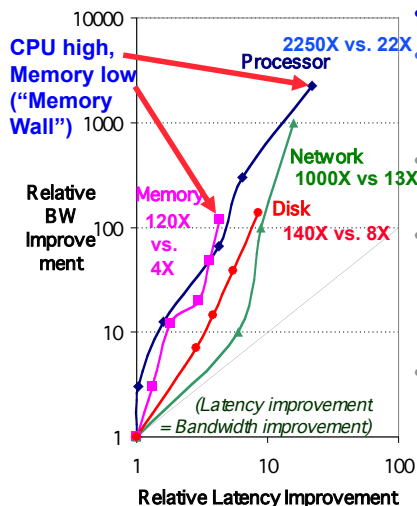
- | | |
|---|--|
| <ul style="list-style-type: none"> • 1982 Intel 80286 • 12.5 MHz • 2 MIPS (peak) • Latency (inst) 320 ns • 134,000 xtors, 47 mm² • 16-bit data bus, 68 pins • Microcode interpreter, separate FPU chip • (no caches) | <ul style="list-style-type: none"> • 2001 Intel Pentium 4 • 1500 MHz (120X) • 4500 MIPS (peak) (2250X) • Latency 15 ns (20X) • 42,000,000 xtors, 217 mm² • 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 |
|---|--|



8/31/10

60

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)

8/31/10

61

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

8/31/10

62

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 mm² (ratio sqrt ⇒ 2X)
 - » DRAM Die Size: 47 vs. 217 mm² (ratio sqrt ⇒ 2X)

8/31/10

63

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**

8/31/10

64

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 \Rightarrow 15000 RPM = 4.2X
 - » Average rotational latency: 8.3 ms \Rightarrow 2.0 ms
 - » Things being equal, also helps BW by 4.2X
- Lower DRAM latency \Rightarrow More access/second (higher bandwidth)
- Higher linear density helps disk BW (and capacity), but not disk Latency
 - » 9,550 BPI \Rightarrow 533,000 BPI \Rightarrow 60X in BW

8/31/10

65

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

- Short messages, more packet overhead, less bandwidth but faster latency
- Longer messages, less packet overhead, more bandwidth but longer latency

8/31/10

66

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

8/31/10

67

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

8/31/10

68



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

8/31/10 Die cost + packaging cost + testing cost

69



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

8/31/10

70



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

8/31/10

71



Define and quantify 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)

8/31/10

72



Example of quantifying power

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

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

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

8/31/10

73



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

8/31/10

74



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

8/31/10

75



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

8/31/10

76

Define and quantify 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)** = $1/\text{MTTF}$, 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)** = $\text{MTTF} + \text{MTTR}$
- **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)** = $\text{MTTF} / (\text{MTTF} + \text{MTTR})$

8/31/10

77

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 $\text{MTTF}_{\text{powersupply}}/2$

$$\text{MTTF}_{\text{pairps}} = \frac{\text{MTTF}_{\text{ps}}/2}{\frac{\text{MTTR}_{\text{ps}}}{\text{MTTF}_{\text{ps}}}} = \frac{\text{MTTF}_{\text{ps}}^2/2}{\text{MTTR}_{\text{ps}}} = \frac{\text{MTTF}_{\text{ps}}^2}{2 * \text{MTTR}_{\text{ps}}}$$

8/31/10

79

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):

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

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

8/31/10

78

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 $\text{MTTF}_{\text{pair}}$? $\text{MTTF}_{\text{system}}$?

$$\text{MTTF}_{\text{pair}} = \frac{200,000^2}{2 * 24} = 830,000,000$$

$$\begin{aligned} \text{FailureRate} &= \frac{10}{1,000,000} + \frac{1}{5,000,000} + \frac{1}{830,000,000} \\ &= \frac{10 + 2 + 0}{1,000,000} = \frac{12}{1,000,000} = 12,000 \text{ FIT} \end{aligned}$$

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

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

8/31/10

80



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

8/31/10

81



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

$$\text{performance}(x) = \frac{1}{\text{execution_time}(x)}$$

" X is n times faster than Y" means

$$n = \frac{\text{Performance}(X)}{\text{Performance}(Y)} = \frac{\text{Execution_time}(Y)}{\text{Execution_time}(X)}$$

8/31/10

82



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

8/31/10 TPC-App application server and web services benchmark

83



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

8/31/10

84



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)

8/31/10

85



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 ⇒ Buy 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]

8/31/10

86