

# G22.2243-001 High Performance Computer Architecture

Instructor: Vijay Karamcheti  
vijayk@cs.nyu.edu

Lecture: Tuesdays, 5:00pm-7:00pm  
101 CIWW

Office Hours: Wednesdays, 4:00pm-5:00pm  
715 Broadway, Room 704  
phone: 8-3496

Mailing List: g22\_2243\_001\_fa04@cs.nyu.edu

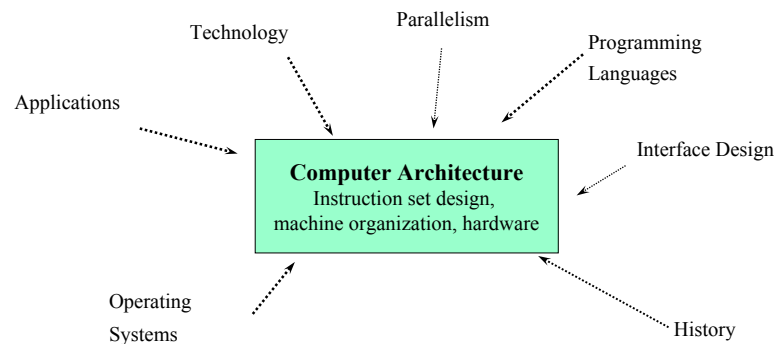
## Outline

- Introduction
- Administrative stuff
  - course organization
  - workload and grading
- SimpleScalar toolset
- Fundamentals of computer design
  - Target markets
  - Technology trends
  - Cost vs. price
  - Measuring and reporting performance
  - Quantitative principles of computer design
  - Price performance

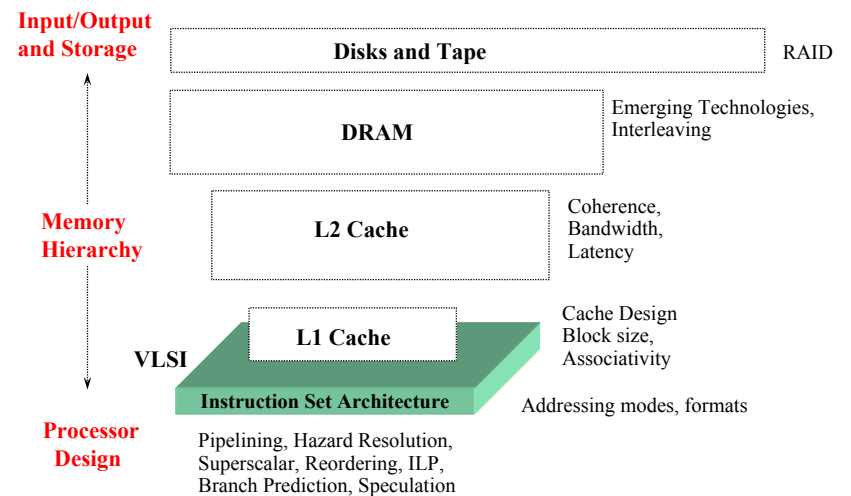
[ Hennessy/Patterson CA:AQA (3<sup>rd</sup> Edition): Chapter 1]

## Course Objective

- The course objective is to gain the knowledge required to design and analyze high-performance computer systems.

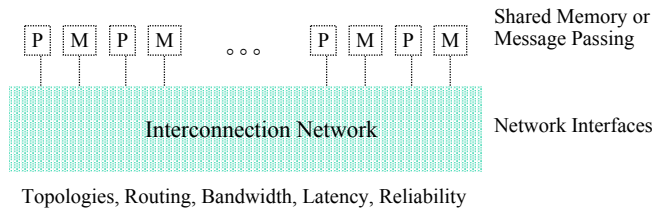


## Computer Architecture Topics



## Computer Architecture Topics

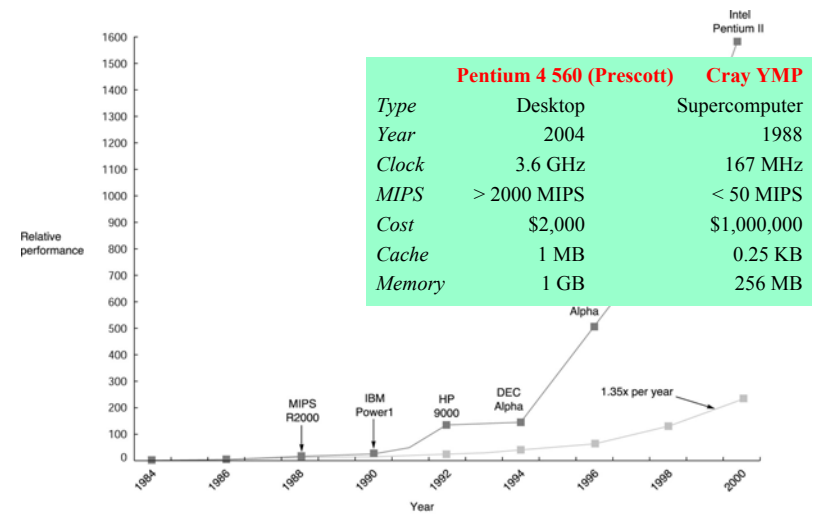
### Networks, Interconnections, and Multiprocessors



9/6/2004

5

## Growth in Microprocessor Performance



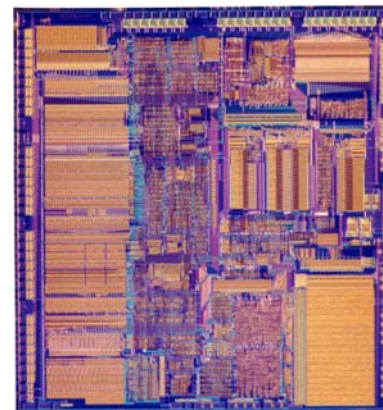
9/6/2004

6

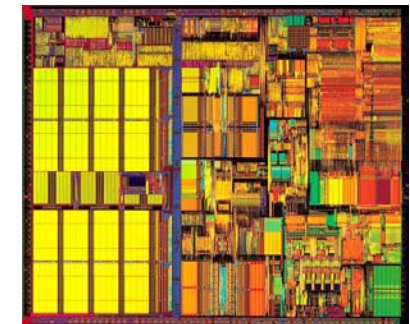
## Reasons for this Performance Improvement

- Technology
  - More transistors per chip
  - Faster logic
- Machine Organization/Implementation
  - Deeper pipelines
  - More instructions executed in parallel
- Instruction Set Architecture
  - Reduced Instruction Set Computers (RISC)
  - Multimedia extensions
  - Explicit parallelism
- Compiler technology
  - Finding more parallelism in code
  - Greater levels of optimization

## From the Intel 386 to the Pentium III



Intel 386, introduced 1985  
275,000 transistors, 1 micron  
16 MHz clock speed

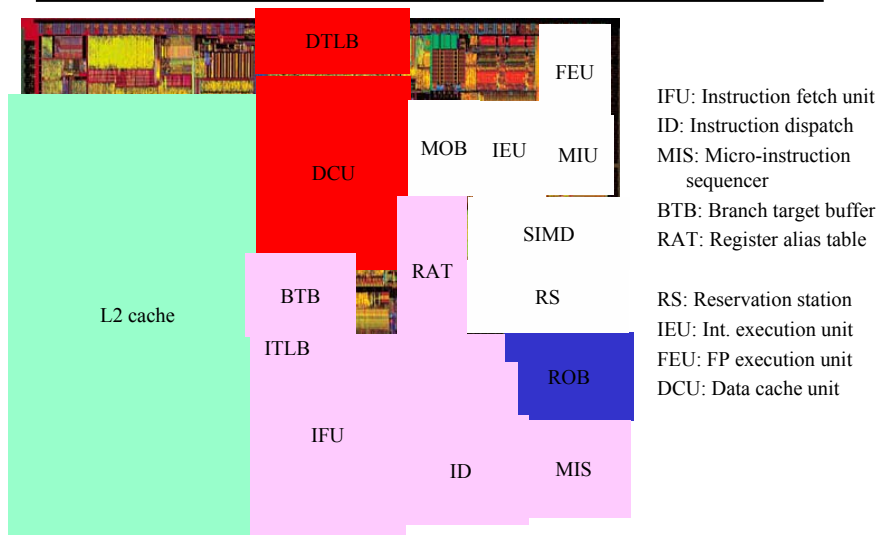


Intel Pentium III, introduced 1999  
9.5M transistors, 0.25 micron  
600 MHz clock speed

9/6/2004

8

## Intel Pentium III Microarchitecture



9/6/2004

9

## What is Ahead?

- Today's desktop microprocessors (e.g., Pentium 4)
  - 125 million transistors, 90 nanometer technology, 3.6 GHz clock speed
  - Internally: “hyper-pipelining”, multithreading, 128-bit SIMD instructions
- The future
  - Greater instruction level parallelism
  - Bigger caches, and more levels of cache
  - Multiple processors per chip (“multicore”)
    - Complete systems on a chip
  - High performance interconnects
  - New application characteristics
    - Multimedia, new interface technologies (video, speech ...)
  - Breakdown into desktop, enterprise, and embedded target markets
    - Different performance criteria
- This course provides the background for you to design, analyze, and effectively use such systems

## Topic Coverage

- Textbook
  - Hennessy and Patterson, Computer Architecture: A Quantitative Approach, 3rd Edition, 2003
- Fundamentals of Computer Design (Chapter 1) 1 lecture
- Instruction Set Architecture (Chapter 2) 0.5 lectures
- Pipelining Basics (Appendix A) 2 lectures
- Instruction Level Parallelism (Chapters 3 and 4) 5 lectures
- Memory Hierarchy (Chapter 5) 2 lectures
- Multiprocessors (Chapter 6) 1.5 lectures
- Stream Processing Architectures 2 lectures
- Plus, relevant computer architecture research papers.

9/6/2004

11

## Course Workload

- Lectures (sometimes may be given by a guest lecturer)
  - Reading assignments from text
- Five programming assignments (65% of course grade)
 

Build a simulator for a **multiple-issue, out-of-order execution** microprocessor in stages (Pentium III class)

  - Assignment 1: Pipelining
  - Assignment 2: Branch prediction
  - Assignment 3: Reservation stations (Multiple functional units)
  - Assignment 4: Multiple issue, out-of-order execution
  - Assignment 5: Cache hierarchies

Use this simulator to analyze impact of architectural techniques
- Final exam (35% of course grade)
  - Sample questions will be provided in class (end of chapter exercises)
    - Turn-in is optional

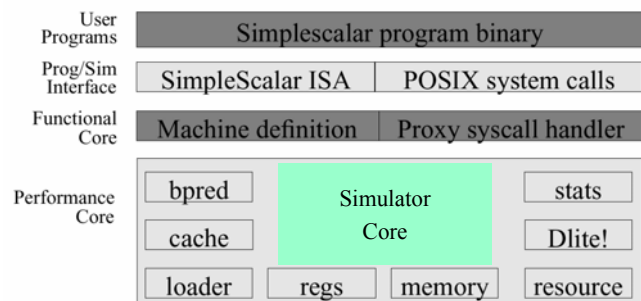
9/6/2004

12

## SimpleScalar Toolset

---

- Comprehensive collection of tools for evaluating new architectural techniques
  - Possible to define new instruction-set architectures (support for Alpha, PISA)
  - Modules for writing own execution-driven simulators
    - bpred, caches, statistics collection, program loading, functional unit construction, ...
  - More than 50% of papers at recent architecture conferences use SimpleScalar



9/6/2004

13

## SimpleScalar Toolset (cont'd)

---

- For the course assignments we will be using a small subset of the tools
  - You will be using an instructional ISA called PISA
    - Closely resembles MIPS 64 (described in the textbook)
  - PISA executables produced using GNU cross-compiler tools
- Goal of the assignments: To understand the issues involved in **implementing** and to assess the **potential benefits** from architectural techniques used in modern-day microprocessors
  - E.g., Branch prediction in modern-day microprocessors
    - At what stage during instruction execution is branch prediction used?
      - It takes some time to figure out that an instruction is a branch
    - How should the branch predictor be updated with information about seen branches?
    - What impact does prediction have on performance?

9/6/2004

14

## Background Survey

---

- Programming in C/C++
  - Required for using the SimpleScalar toolset
- Use of Unix systems
  - SimpleScalar installs available for SPARC/Solaris and x86/Linux
- Prior coursework
  - Logic design:
    - Bits, gates, combinational and sequential logic
    - Adders, multipliers
  - Computer organization and assembly-level programming
    - ALUs, MIPS-like data path (without pipelining)
    - Register versus memory operations
    - Buses, I/O

9/6/2004

15

## Fundamentals of Computer Design

## Context for Designing New Architectures

What is the target market?

- Desktop computing
  - General-purpose applications
  - Performance improvements must be traded off against cost
  - Performance metric of interest is usually **response time**
- Enterprise servers
  - Cost is important but not as much of a concern
  - Performance is paramount, but metric of interest is usually **throughput**
- Embedded computers
  - Cost is paramount
  - Power concerns
  - Specialized applications

9/6/2004

17

## Designing New Architectures: Other Considerations

- Application Area
  - Special Purpose / General Purpose
  - Scientific (FP intensive) / Commercial (Integer Intensive)
- Level of Software Compatibility Required
  - Object Code / Binary Compatible
  - Assembly Language
  - Programming Language
- Operating System Requirements
  - Size of Address Space
  - Memory Management / Protection
  - Context Switch Capability
  - Interrupts and Traps
- Standards:
  - IEEE 754 Floating Point
  - I/O Bus
  - Networks
- Technology Improvements
  - Increased Processor Performance
  - Larger Memory and I/O devices
  - Software/Compiler Innovations

9/6/2004

18

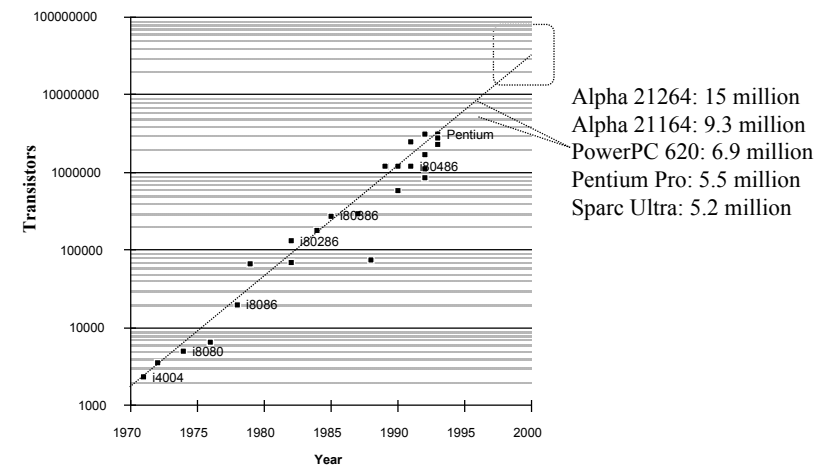
## Technology Trends

- Helps an architect plan for the evolution of an architecture

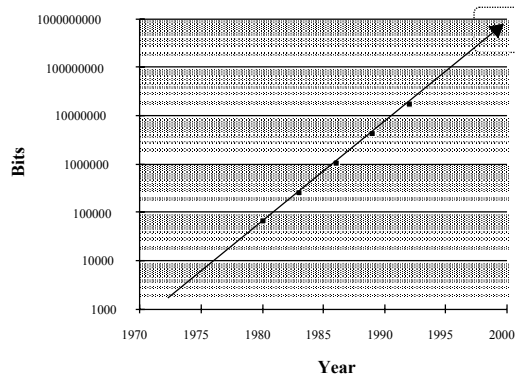
Four implementation technologies of interest

- **Integrated circuit logic**
  - Transistor density: increases by ~35% per year
  - Die size: increases by ~10-20% per year
  - Transistor count/die: increases by ~55% per year
- **Semiconductor DRAM**
  - Capacity increases by ~40-60% per year
  - Cycle time has not decreased as much: ~33% over 10 years
  - Bandwidth has increased: about ~66% more over 10 years
    - also, changes to the interface have helped further improve bandwidth
- **Magnetic disk technology**
  - Recently, capacity improving by ~100% every year
- **Network technology**
  - More improvements in bandwidth, less in latency

## Technology Trends: Microprocessor Capacity



## Memory Capacity (Single Chip DRAM)



year	size	cyc time
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1996	64 Mb	125 ns
2000	256 Mb	100 ns

## Transistor Performance, Wires, and Power

- Driving factor in integrated circuit technology is **feature size**
  - Decreased from 10 microns (1971) to 0.09 microns (2004)
- Transistor count/density improves **quadratically** with feature size
  - But, performance only increases **linearly**
    - Need to reduce operating voltage to ensure correct operation at small sizes
- Unfortunately, wires do **not** improve in performance with decreasing feature size
  - Signal delay proportional to product of resistance and capacitance
  - Both increase as feature size decreases
  - Modern-day processors spend large fraction of time cycle just propagating signals between different portions of the chip
    - 2 stages of the Pentium 4 pipeline dedicated for this
- Power requirements grow dramatically with decreasing feature size
  - Pentium 4 consumes ~100 watts
  - Likely to become future bottleneck

9/6/2004

22

## Cost vs. Price

- Price: How much the finished good sells for
- Cost: Amount spent to produce it (including overhead)
- Price affected by
  - time (maturity of process)
  - volume (amortization of non-recurring costs)
  - commoditization (competition)
- Cost affected by
  - Yield
  - Direct costs (labor, purchasing, scrap, warranty)
  - Gross margin (R&D, marketing, sales, maintenance ...)
- Read Section 1.4 for additional details

9/6/2004

23

## Measurement and Evaluation

- Architecture is an iterative process:
  - Search the possible design space
  - Make selections
  - Evaluate the selections made
- Good measurement tools are required to accurately evaluate the selection
- The book argues for what has now become widely accepted: A **quantitative approach** for evaluating selections
  - most accurate measure of performance is the **execution time** of representative **real programs** (benchmarks)

## Performance Factors

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

- **Instruction Count (IC):** Number of instructions/program
- **Cycles per instruction (CPI)**
  - Sometimes the reciprocal is used: **Instructions per cycle (IPC)**
- The number of seconds per cycle is the **clock period**
  - **clock rate** is the multiplicative inverse of the clock period

9/6/2004

25

## Aspects of CPU Performance

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

	Instr. Count	CPI	Clock Rate
Program	X		
Compiler	X		
Instr. Set	X	X	
Organization		X	X
Technology			X

## Comparing CPU Time

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

- A 500 MHz Pentium III processor takes 2 ms to run a program with 200,000 instructions.
- A 300 MHz UltraSparc processor takes 1.8 ms to run the same program with 230,000 instructions.
- Which processor is faster and by how much?
  - The UltraSparc is  $2/1.8 = 1.11$  times as fast, or 11% faster.
- What is the CPI for each processor for this program?
  - $\text{CPI} = \frac{\text{Cycles}}{\text{Instruction Count}} = \frac{\text{CPU time} \times \text{Clock Rate}}{\text{Instruction Count}}$
  - $\text{CPI}_{\text{Pentium}} = 2 \times 10^{-3} \times 500 \times 10^6 / 200 \times 10^3 = 5.00$
  - $\text{CPI}_{\text{SPARC}} = 1.8 \times 10^{-3} \times 300 \times 10^6 / 230 \times 10^3 = 2.35$

9/6/2004

27

## Cycles Per Instruction

“Average Cycles per Instruction”

$$\text{CPU time} = \text{Cycle Time} \times \text{Number of cycles}$$

$$\text{CPU time} = \text{CycleTime} \times \sum_{i=1}^n \text{CPI}_i \times \text{IC}_i$$

“Instruction Frequency”

$$\text{CPI} = \sum_{i=1}^n \text{CPI}_i \times F_i \quad \text{where } F_i = \frac{\text{IC}_i}{\text{Instruction Count}}$$

- Implication: Invest resources where time is spent!

## Example: Calculating CPI

### Base Machine (Reg / Reg)

Op	Freq	Cycles	$F_i * CPI_i$	(% Time)
ALU	50%	1	.5	(33%)
Load	20%	2	.4	(27%)
Store	10%	2	.2	(13%)
Branch	20%	2	.4	(27%)
			<u>1.5</u>	

Typical Mix

## Example (cont'd)

- Add **register / memory ALU** operations:
  - One source operand in memory
  - One source operand in register
  - Cycle count of **2**
- Branch cycle count to increase to **3**
- What fraction of the loads must be eliminated for this to pay off, assuming the clock rate is not affected?

Op	Freq	Cycles	CPI	Freq	Cycles	CPI
ALU	.50	1	.5	.5 - X	1	.5 - X
Load	.20	2	.4	.2 - X	2	.4 - 2X
Store	.10	2	.2	.1	2	.2
Branch	.20	2	.3	.2	<b>3</b>	.6
Reg/Mem				X	<b>2</b>	2X
			1.00	1 - X		$(1.7 - X)/(1 - X)$

## Programs to Evaluate Processor Performance

- (Toy) Benchmarks
  - 10-100 line program
  - e.g.: sieve, puzzle, quicksort
- Synthetic Benchmarks
  - Attempt to match average frequencies of real workloads
  - e.g., Whetstone, dhrystone
- Kernels
  - Time critical excerpts
- **Real Benchmarks**
  - Ideal: exactly the programs one would like to run on the architecture
  - More generally: a collection of programs that possess the characteristics of the real workload

## SPEC: System Performance Evaluation Cooperative

- First Round SPEC CPU89
  - 10 programs yielding a single number
- Second Round SPEC CPU92
  - SPEC CINT92 (6 integer programs) and SPEC CFP92 (14 floating point programs)
  - Compiler flags can be set differently for different programs
- Third Round SPEC CPU95
  - new set of programs: SPEC CINT95 (8 integer programs) and SPEC CFP95 (10 floating point)
  - Single flag setting for all programs
- Fourth Round SPEC CPU2000
  - new set of programs: SPEC CINT2000 (12 integer programs) and SPEC CFP2000 (14 floating point)
  - Single flag setting for all programs
  - Programs in C, C++, Fortran 77, and Fortran 90
  - Report both **baseline** and **best** performance

## SPEC 2000

- 12 integer programs:
  - 2 Compression
  - 2 Circuit Placement and Routing
  - C Programming Language Compiler
  - Combinatorial Optimization
  - Chess, Word Processing
  - Computer Visualization
  - PERL Programming Language
  - Group Theory Interpreter
  - Object-oriented Database.
- Written in C (11) and C++ (1)
- 14 floating point programs:
  - Quantum Physics
  - Shallow Water Modeling
  - Multi-grid Solver
  - 3D Potential Field
  - Parabolic / Elliptic PDEs
  - 3-D Graphics Library
  - Computational Fluid Dynamics
  - Image Recognition
  - Seismic Wave Simulation
  - Image Processing
  - Computational Chemistry
  - Number Theory / Primality Testing
  - Finite-element Crash Simulation
  - High Energy Nuclear Physics
  - Pollutant Distribution
- Written in Fortran (10) and C (4)

9/6/2004

33

## Other SPEC Benchmarks

- JVM98:
  - Measures performance of Java Virtual Machines
- SFS97:
  - Measures performance of network file server (NFS) protocols
- Web99:
  - Measures performance of World Wide Web applications
- HPC96:
  - Measures performance of large, industrial applications
- APC, MEDIA, OPC
  - Measure performance of graphics applications
- For more information about the SPEC benchmarks see:  
<http://www.spec.org>.

9/6/2004

34

## Implication of the Quantitative Approach

- **Make the common case fast**
  - One of the principles behind RISC: **R**educed **I**nstruction **S**et **C**omputers
    - Identify most frequently-used instructions
      - Implement them in hardware
    - Emulate other instructions (slowly) in software
  - Pretty much every technique used in current-day microprocessors
- Is there a way of quantifying the gains one is likely to see by improving some portion of the design?
  - What is the best one can hope to do?
- General principle for the above: **Amdahl's Law** (sometimes also called Amdahl's curse)

9/6/2004

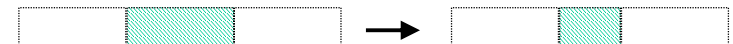
35

## Amdahl's Law

- Speedup due to enhancement E:

$$\text{Speedup (E)} = \frac{\text{Execution time without E}}{\text{Execution time with E}} = \frac{\text{Performance with E}}{\text{Performance without E}}$$

- Suppose that enhancement E accelerates a fraction  $f$  of the task by a factor  $s$ , and the remainder of the task is unaffected



- New execution time and the overall speedup?

$$\text{Exec. time}_{\text{new}} = \text{Exec. time}_{\text{old}} \times [ (1-f) + f/s ]$$

$$\text{Speedup (E)} = \frac{\text{Exec. Time}_{\text{old}}}{\text{Exec. Time}_{\text{new}}} = \frac{1}{[1-f+f/s]} \leq \frac{1}{(1-f)}$$

9/6/2004

35

## Example of Amdahl's Law

---

- Floating point instructions improved to run **2x**; but only **10%** of the time was spent on these instructions
- How much improvement in performance should one expect?

$$\text{Exec. time}_{\text{new}} = \text{Exec. time}_{\text{old}} \times [ (1-f) + f/s ]$$

$$\text{Speedup (E)} = \frac{\text{Exec. Time}_{\text{old}}}{\text{Exec. Time}_{\text{new}}} = \frac{1}{[1-f+f/s]} \leq \frac{1}{(1-f)}$$

$$\text{Exec. time}_{\text{new}} = \text{Exec. time}_{\text{old}} \times [ (1 - 0.1) + 0.1/2 ] = \text{Exec. time}_{\text{old}} \times 0.95$$

$$\text{Speedup (E)} = \frac{\text{Exec. Time}_{\text{old}}}{\text{Exec. Time}_{\text{new}}} = \frac{1}{0.95} = 1.053$$

- The new machine is **5.3%** faster for **this mix of instructions**

## Summary

---

- A fundamental rule in computer architecture is to make the common case fast
- The most accurate measure of performance is the execution time of representative real programs (benchmarks)
- Execution time is dependent on the number of instructions per program, the number of cycles per instruction, and the clock rate
- When designing computer systems, both cost and performance need to be taken into account
  
- See Section 1.7 of the textbook for several examples of performance and price-performance in desktop, server, and embedded computers