

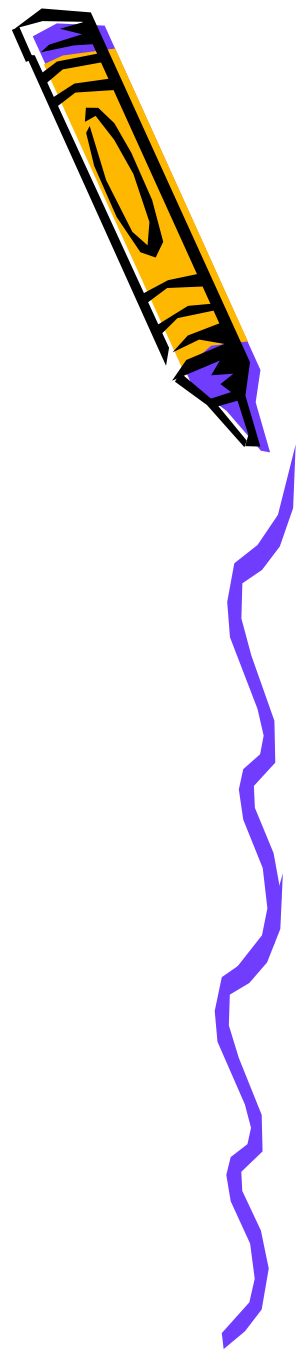


# Computer Architecture

Hossein Asadi  
Department of Computer Engineering  
Sharif University of Technology  
[asadi@sharif.edu](mailto:asadi@sharif.edu)

# Today's Topics

- Performance Evaluation
- Standard Benchmarks



# Copyright Notice



- Parts (text & figures) of this lecture adopted from:
  - Computer Organization & Design, The Hardware/Software Interface, 4<sup>th</sup> Edition, by D. Patterson and J. Hennessey, MK publishing, 2012.
  - "Intro to Computer Architecture" handouts, by Prof. Hoe, CMU, Spring 2009.
  - "Computer Architecture & Engineering" handouts, by Prof. Kubiawicz, UC Berkeley, Spring 2004.
  - "Intro to Computer Architecture" handouts, by Prof. Hoe, UWisc, Spring 2021.
  - "Computer Arch I" handouts, by Prof. Garzarán, UIUC, Spring 2009.



# Performance Metrics



- Latency
  - Time between start and finish of a single task
- Throughput
  - Number of tasks finished in a given unit of time
- Question:
  - $\text{Throughput} = 1 / \text{Latency} (?)$
  - $\text{Latency} = 1 / \text{Throughput} (?)$



# Throughput vs. Latency



- Example: Airbus A380 vs. BAC Concorde
  - $\text{Speed}_{\text{Concorde}} > \text{Speed}_{\text{Boeing}}$ 
    - $\text{Latency}_{\text{Concorde}} < \text{Latency}_{\text{Boeing}}$
  - Throughput???
  - How to define throughput?

Airplane	Passenger Capacity	Speed (mph)
Airbus A380	850	610
Concorde	132	1350



# Throughput vs. Latency (cont.)



- Throughput = Passenger Capacity \* Speed

Airplane	Passenger Capacity	Speed (mph)	Passenger Throughput
Airbus A380	850	610	518,500
Concorde	132	1350	178,200



# Throughput vs. Latency: Single-Core Example



- Consider Two Single-Core Computers
  - Less latency (response time)
    - The one that finishes tasks faster
  - More throughput
    - The one that finishes more jobs
      - Or the one that finishes tasks faster
  - In this example
    - Latency  $\downarrow \Rightarrow$  Throughput  $\uparrow$
    - Latency  $\uparrow \Rightarrow$  Throughput  $\downarrow$
  - Not always true!



# Throughput vs. Latency: Multi-Core Example



- Computer A
  - Two cores, running at 3Mhz
- Computer B
  - Four cores, running at 2.5Mhz
- $\text{Latency}_A < \text{Latency}_B$
- $\text{Throughput}_A < \text{Throughput}_B$

Computer	Frequency	# of cores
Type A	3 Mhz	2
Type B	2.5 Mhz	4





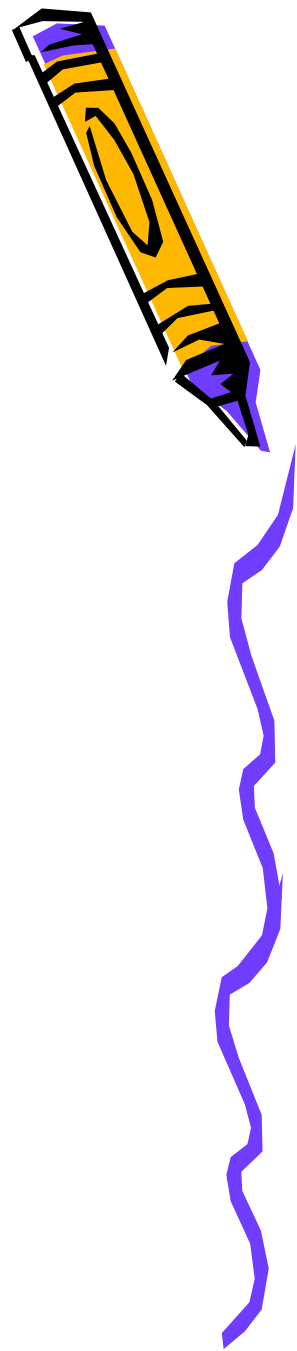
# Performance Definition



- Response Time
  - Total time to complete a task (program)
  - Also called, wall-clock time, elapsed time
- Performance = 1 / response time
  - Response time  $\downarrow \Rightarrow$  Performance  $\uparrow$
  - $\text{Performance}(x) / \text{Performance}(y) =$   
Execution time (y) / Execution time (x)



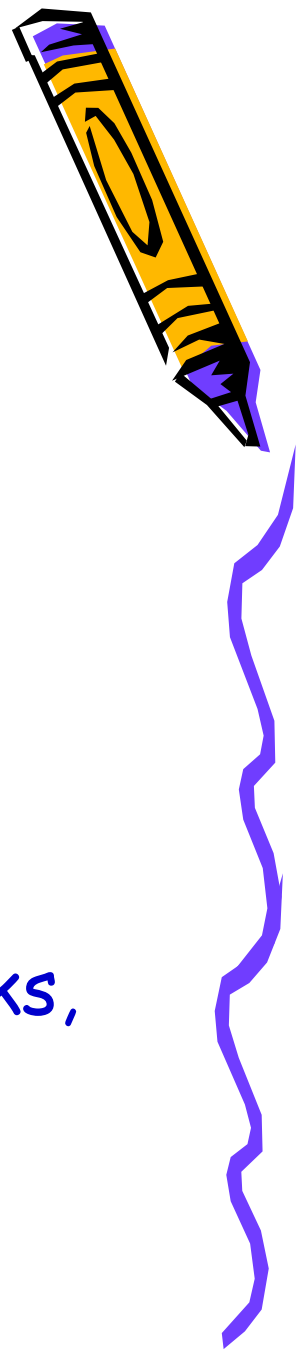
# Performance Definition (cont.)



- Response Time Consists of:
  - CPU time
    - CPU time spent on a program
  - I/O time
    - Time elapsed to wait for I/O transactions
- CPU Time
  - User CPU time
    - CPU time directly spent on a program
  - System CPU time
    - CPU time spent in OS doing tasks on behalf of a program



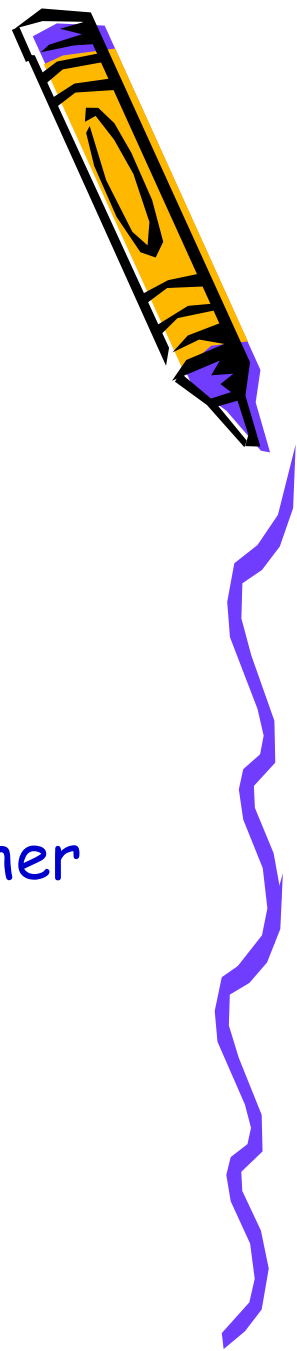
# Performance Definition (cont.)



- System Performance
  - 1 / Elapsed time
- CPU Performance
  - 1 / User CPU time
- Clock Cycles
  - Clock periods, cycles, clock ticks, ticks, clocks



# Response Time vs. Throughput



- Which One is More Important?
  - Response time or throughput
- User Perspective
  - Response time more visible
    - Unless a user runs bunch of tasks together
- System Admin Perspective
  - Throughput more visible



# CPU Time

= CPU Clock Cycles \* Clock Cycle Time

= CPU Clock Cycles / Clock Rate

- Example

- $\text{CPU}_A(\text{clock rate}) = 4 \text{ Ghz}$

- $\text{CPU}_B(\text{clock rate}) = 3 \text{ Ghz}$

- Which one runs program X faster?

- Depends on number of clock cycles spend on program X



## CPU Time (cont.)

= CPU Clock Cycles \* Clock Cycle Time

= CPU Clock Cycles / Clock Rate

- CPU Clock Cycles

  - = # of Instructions of a program \* CPI

- CPI?

  - Average Clock Cycles per Instruction



## CPU Time (cont.)

$$\begin{aligned} &= \text{Instr. Count} * \text{CPI} * \text{Clock Cycle Time} \\ &= (\text{Instr. Count} * \text{CPI}) / \text{Clock Rate} \end{aligned}$$

$$\text{Processor Performance} = \frac{\text{Time}}{\text{Program}}$$

$$\begin{array}{ccccc} \boxed{\begin{array}{c} \text{Instructions} \\ = \frac{\text{-----}}{\text{Program}} \end{array}} & \times & \boxed{\begin{array}{c} \text{Cycles} \\ \frac{\text{-----}}{\text{Instruction}} \end{array}} & \times & \boxed{\begin{array}{c} \text{Time} \\ \frac{\text{-----}}{\text{Cycle}} \end{array}} \\ \text{(code size)} & & \text{(CPI)} & & \text{(cycle time)} \end{array}$$

# CPU Time (cont.)



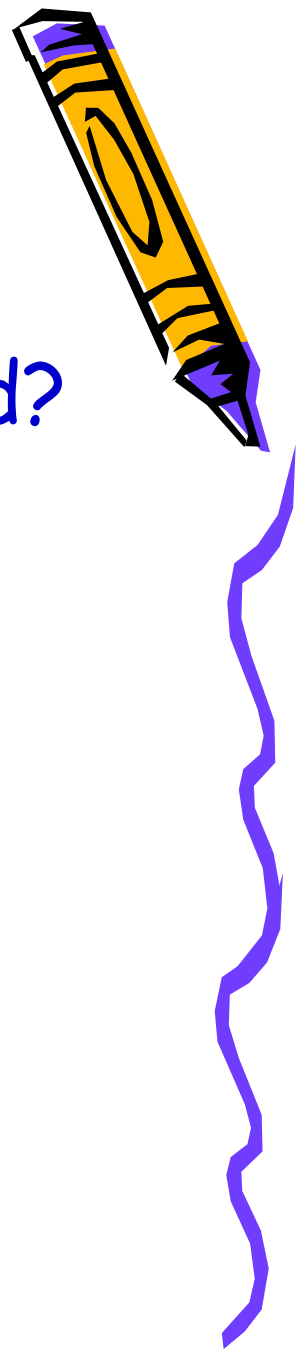
- Example
  - Instruction count (IC) = 10,000
  - CPI = 4
  - Clock cycle time = 500ps or 0.5 ns
  - CPU time =  $10,000 * 4 * 0.5 = 20 \text{ us}$
- Question:
  - IC same as code size or lines of code?



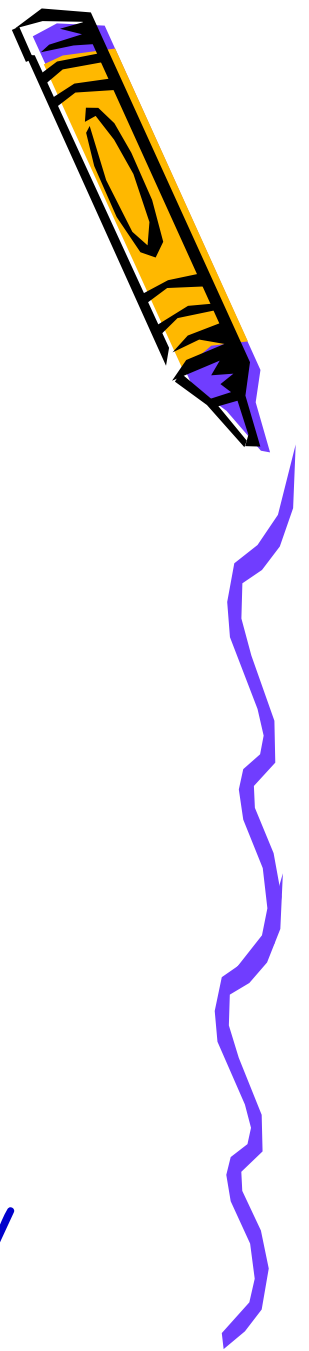


# CPU Time (cont.)

- How these Parameters Determined?
  - Instruction count
  - CPI
  - Clock cycle time



# CPU Time (cont.)



- Instruction Count (IC)
  - Determined by program
  - ISA
  - Compiler
- CPI
  - Determined by uArch
    - The way processor is implemented
  - Code CPI also depends on program
- Clock Cycle Time
  - Determined by uArch and Technology



# Easy Practice



- Assume:
  - A C program compiled on two computers
    - Computer A with a RISC ISA
    - Computer B with a CISC ISA
  - Q1: which one would have higher IC?
  - Q2: which one would have smaller CPI?
  - Q3: which one would have higher performance?
  - Q4: can we have CPI less than one?



# CPI Classes



- Question:
  - Do all instructions have same CPI?
    - No

	CPI for instruction classes		
	A	B	C
CPI	1	2	3

Code Sequence	Instruction count		
	A	B	C
1	2	1	2
2	4	1	1



# CPI Classes (cont.)



- CPU Clock Cycles

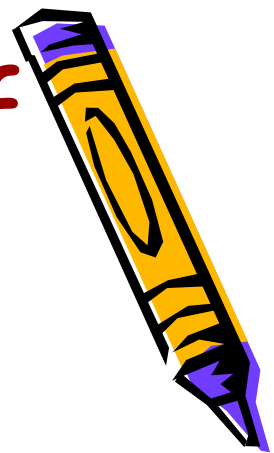
$$= CPI_1 * IC_1 + CPI_2 * IC_2 + ... + CPI_n * IC_n$$

	CPI for instruction classes		
	A	B	C
CPI	1	2	3

Code Sequence	Instruction count		
	A	B	C
1	2	1	2
2	4	1	1



# Performance Evaluation of Computers



- So Far
  - Learnt how to measure CPU Performance
- Consider Two CPUs
  - CPU1 runs faster than CPU2 on program A
  - CPU2 runs faster than CPU1 on program B
- Questions:
  - How we can decide which CPU is faster?
  - Which candidate programs should we choose to compare CPU1 and CPU2?



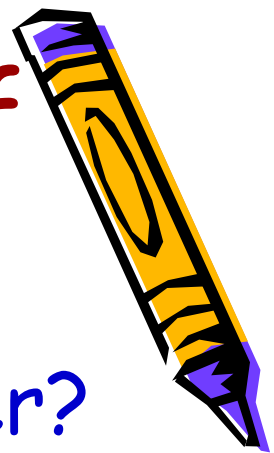
# Performance Evaluation of Computers (cont.)



- Assume
  - A user typically runs programs A, B, and C in his computer (CPU1)
  - "Workload"
    - Set of programs A, B, and C
  - If CPU2 runs this workload faster
    - $\rightarrow \text{Performance}(\text{CPU2}) > \text{Performance}(\text{CPU1})$
  - In reality, we use a set of programs called **benchmarks** to compare performance of processors



# Performance Evaluation of Computers (cont.)



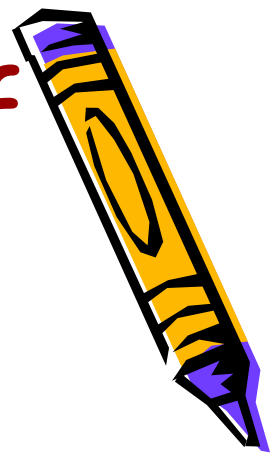
- Which CPU Runs a Workload Faster?
  - Depends on type of average we take over execution times
    - Simple arithmetic mean
    - Weighted arithmetic mean
    - Average over ratios
    - Geometric mean

CPI	Computer A	Computer B
Program 1	1	10
Program 2	1000	100
Total Time	1001	110





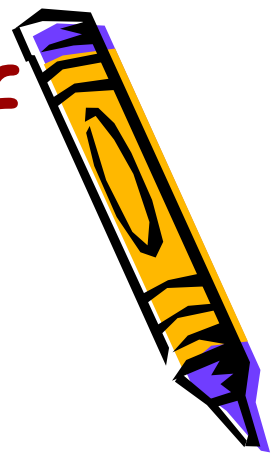
# Performance Evaluation of Computers (cont.)



- Another Question:
  - Why not just running one simple program to compare performance of CPUs?
    - Agreeing on one simple program very hard
    - Designers can optimize processors towards fast running that simple program



# Performance Evaluation of Computers (cont.)



- Which Programs to Choose?
  - Real programs such as MS-Word, Internet explorer, latex compilers
  - Synthetic benchmarks
    - Emulate frequency of different instructions in real programs
  - Standard benchmarks
    - Examine processor and memory hierarchy



# Performance Evaluation of Computers (cont.)



- Which Programs to Choose?
  - Programs from different applications
  - SPEC CPU2000 benchmarks

Integer Benchmarks (12)		FP Benchmarks (14)	
Name	Description	Name	Description
gzip	Compression	ammp	Computation chemistry
vpr	FPGA circuit P&R	swim	Shallow water model
gcc	C compiler	art	Image recognition using neural network
parser	Word processing program	galgel	Computational fluid dynamics



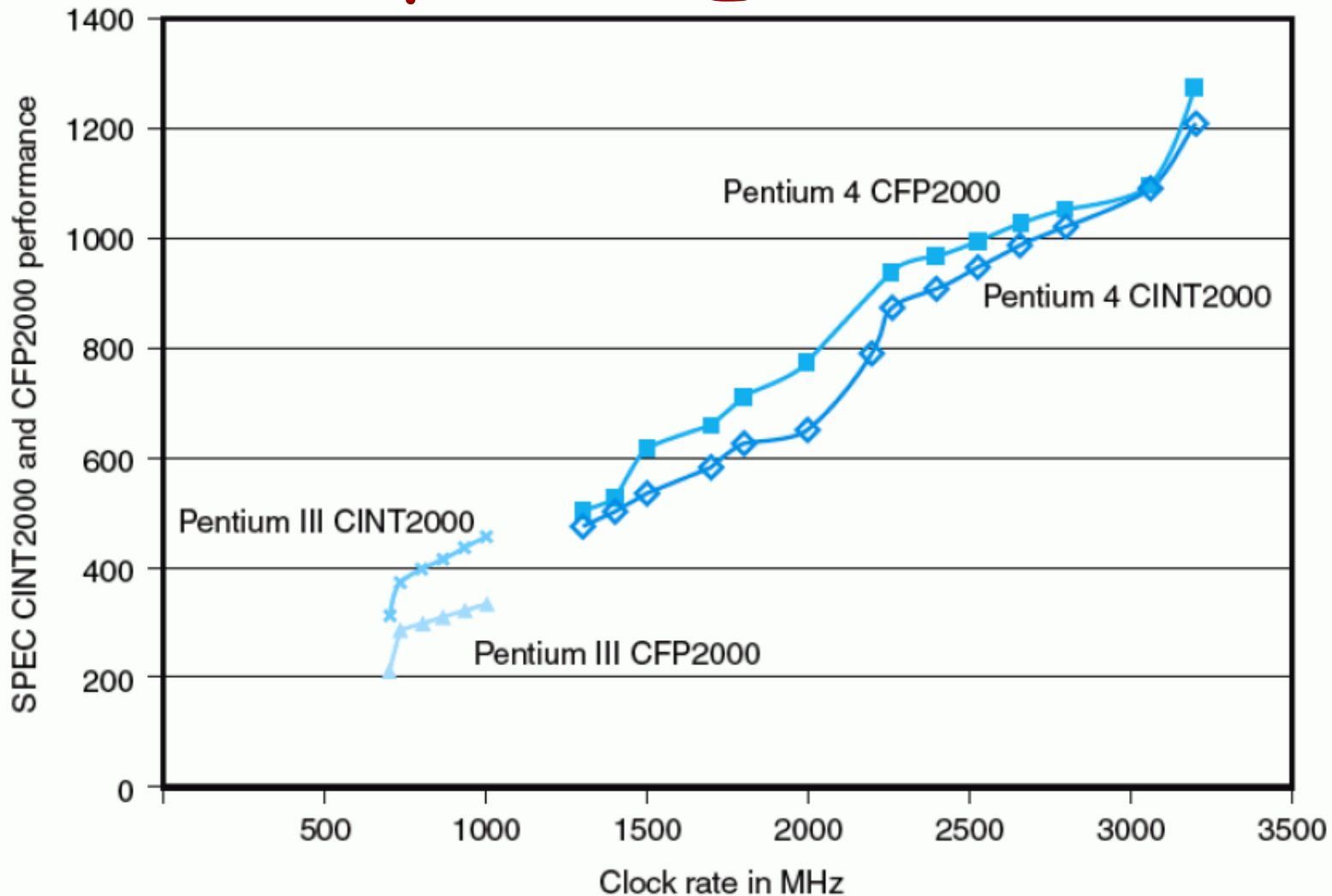
# Easy Practice



- Question:
  - Can microprocessor designers optimize their architecture to optimize performance of SPEC2000 or other benchmark program?



# Performance of PIII and P4 for SPEC2000



# Experiment Setup



- How to Report Experiment Setup?
  - CPU frequency is NOT enough!
  - Need to Report both HW & SW config.
- Software
  - Operating system: WinXP prof. SP1
  - Compiler: Microsoft Visual Studio.NET
    - 7.0.xxx
  - File system type: NTFS
  - System state: default
  - Benchmark, Program, Input to program



# Experiment Setup (cont.)



- Hardware
  - Hardware vender: Dell
  - Model number: Precision WorkStation 360
  - CPU: Intel Pentium 4 (800 MHz system bus)
  - CPU MHz: 3200
  - FPU: Integrated
  - Primary cache: 12KB (I), 8KB (D), both on-chip
  - Secondary cache: 512KB (I+D), on-chip
  - L3 cache: 2048KB (I+D), on-chip
  - Memory: 4x512MB ECC DDR400 SDRAM CL3
  - Disk subsystem: 1x80GB ATA/100 7200 RPM



# CPU/Memory/IO Intensive Benchmarks



- SPEC Benchmarks
  - CPU/Memory intensive benchmarks
  - Some stress CPU
  - Some stress memory subsystem
- SPEC-Web
  - I/O intensive benchmarks
  - Mostly stress I/O subsystem
    - Disk subsystem, network connections, ...





# SPECWeb99 Performance for Variety of Systems



System	Processor	Number of disk drives	Number of CPUs	Number of networks	Clock rate (GHz)	Result
1550/1000	Pentium III	2	2	2	1	2765
1650	Pentium III	3	2	1	1.4	1810
2500	Pentium III	8	2	4	1.13	3435
2550	Pentium III	1	2	1	1.26	1454
2650	Pentium 4 Xeon	5	2	4	3.06	5698
4600	Pentium 4 Xeon	10	2	4	2.2	4615
6400/700	Pentium III Xeon	5	4	4	0.7	4200
6600	Pentium 4 Xeon MP	8	4	8	2	6700
8450/700	Pentium III Xeon	7	8	8	0.7	8001



# Speedup

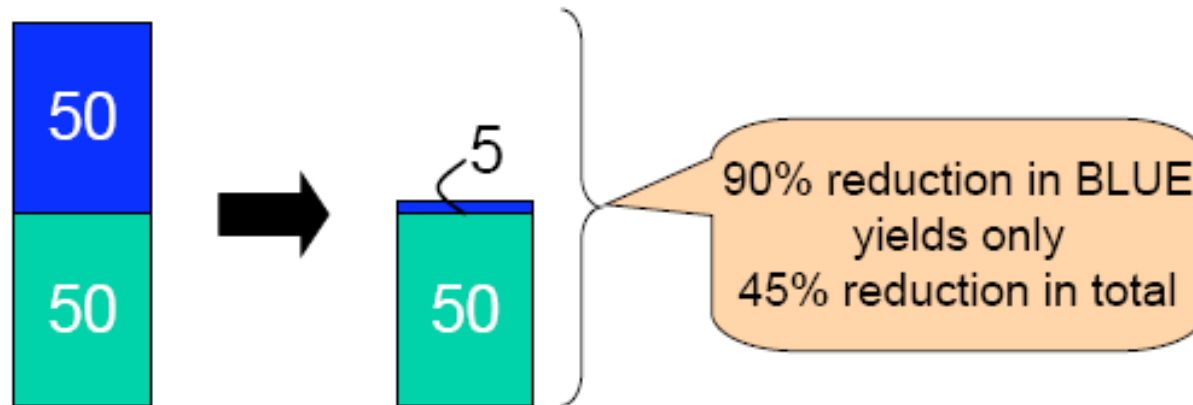


- $\text{Speedup} = \text{time}_{\text{original}} / \text{time}_{\text{improved}}$
- Example
  - $\text{time}_{\text{original}} = 100 \text{ s}$
  - $\text{time}_{\text{improved}} = 98 \text{ s}$
  - $\text{Speedup} = 100/98 = 1.02$



# Amdahl's Law

- Amdahl's Law Says
  - Speedup limited to fraction improved
    - Obvious, but fundamental observation



# Amdahl's Law (cont.)



- Obvious but Common Mistakes
  - CPU upgraded from 1.5Ghz to 3Ghz
    - But not seeing 100% improvement in performance
  - Combinational upgrade 1
    - CPU from 1.5Ghz to 3Ghz
    - Memory from 1GB-90nm to 2GB-45nm
  - Combinational upgrade 2
    - CPU from 1.5Ghz to 3Ghz
    - Memory from 1GB-90nm to 2GB-45nm
    - L1 from 16KB to 32KB & L2 from 1MB to 2MB



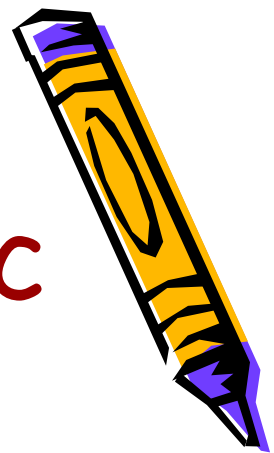
# Speedup (cont.)

**ExecTime<sub>new</sub>**

$$= \text{ExecTime}_{\text{old}} \times \{(1 - \text{Frac}_{\text{enhanced}}) + (\text{Frac}_{\text{enhanced}} / \text{Speedup}_{\text{enhanced}})\}$$

$$\text{Speedup}_{\text{overall}} = \text{ExecTime}_{\text{old}} / \text{ExecTime}_{\text{new}} = \frac{1}{(1 - \text{Frac}_{\text{enhanced}}) + (\text{Frac}_{\text{enhanced}} / \text{Speedup}_{\text{enhanced}})}$$





# MIPS: Performance Metric

- MIPS
  - Million Instructions Per Second =  
$$\text{Instruction Count} / (\text{Execution time} \times 10^6)$$
- Drawbacks?



# MIPS:

## Performance Metric (cont.)

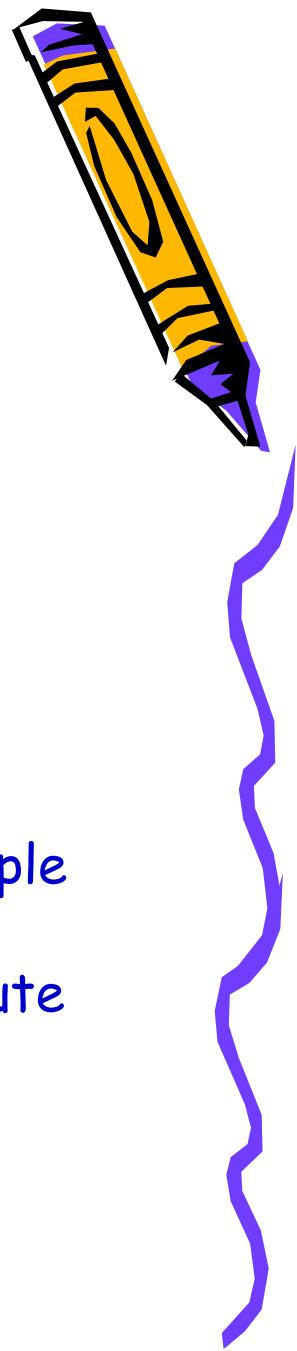


- Drawbacks
  - Not taking into account capabilities of instructions
    - Comparing two different ISAs not fair
  - Not realistic metric even on same CPU
    - Two programs: program A and program B
    - $MIPS(A) > MIPS(B) \Rightarrow Perf.(A) > Perf.(B) ???$
  - Some optimization tech. add more code



# MIPS:

## Performance Metric (cont.)



- Example:
  - Machine A
    - Special instruction for performing square root
    - It takes 100 cycles to execute
  - Machine B
    - Doesn't have special instruction
    - must perform square root in software using simple instructions
    - e.g, Add, Mult, Shift each take 1 cycle to execute
  - Clock cycle = 1 $\mu$ s
  - Machine A: 1/100 MIPS = 0.01 MIPS
  - Machine B: 1 MIPS





# MFLOPS: Performance Metric



- MFLOPS  
= (FP ops/program) × (program/time) × 10<sup>-6</sup>
- Popular in scientific computing
  - FP ops were previously much slower than regular instructions (i.e., off-chip, sequential execution)
- Not great for “predicting” performance
  - Ignores other instructions (e.g., load/store)
  - Not all FP ops have common format
  - Depends on how FP-intensive program is



# Backup

