



# Computer Architecture

---

CSCI 3121

Malcolm Heywood

Course Text: Patterson D.A., Hennessy J.L.,  
Computer Architecture: A Quantitative Approach (4th Ed)



# Course overview

---

- Objectives and performance measurement in computer design
- ISA
  - the design decision perspective
- Instruction Level Parallelism
  - Or how to maximize CPU utilization
- Basic Memory Architecture
- Virtual Memory
  - Or the H/W - O/S interface



# Some Definitions

Instruction Set Architecture (Architecture) or  
ISA

Features of a processor which are visible to  
software.

Organization (Micro-architecture)

Resources and organization used to realize the  
ISA.

Hardware (Logical design or implementation)

Actual logic design used to realize a micro  
architecture



# Packaging and Technology

Pentium Pro	Pentium II (Klamath)	Pentium III (Katmai)	Pentium IV (Willamette)
35 watts	43 watts	34.5 watts	54.7 watts
166 MHz	300 MHz	600 MHz	1.5 GHz
Socket 8	Slot 1	Slot 1	Socket 423
0.35	0.35	0.25	0.18



# ISA design from scratch

---

- Identify Functional Requirements
  - application
- Design optimization
  - What is processor performance ?!
- Cost performance trade off
  - Everything costs
- Future proofing



# Future Proofing

---

- Computer Utilization Trends
  - SW program size
  - High level language development
  - Compiler – processor synergy
- Predicting Technology
  - Processor IC technology
  - DRAM technology
  - Magnetic storage



# Costing an IC before fabrication

---

- Constraints

Wafer shape Wafer size	Determines volume & wastage
Process resolution Process cleanness	Determines die yield



# Implications

---

- Designer only controls die area
- Manufacturing process controls wafer yield, number of layers ( $\alpha$ ) and defects
- Die cost  $\geq f$  ( Die area<sup>4</sup> )
- √ Overall IC costs

$$\frac{\text{Die Cost} + \text{Total Test Cost} + \text{Packaging cost}}{\text{Final Test Yield}}$$



# Computing Effectiveness

---

- Objective – Qualify which computers are more effective
- Consider
  - MIPS - millions of instructions per second
  - Clock frequency
  - Pipeline depth
  - Cache size
- Do any of these performance indicators capture what it means to compute effectively?



# Execution time on task 'x'

---

- What is the execution time for a task?
- Consider
  - UNIX 'time' command
  - Typical response
    - 22.3u 1.2s 1:01 39%
  - Implication – execution time is a function of
    - Operating system; number of concurrent tasks; integration of the peripheral devices



# Execution time (UNIX)

metric	Description
Wall clock time Response time Elapsed time	Latency to complete task including peripherals, any multitasking and operating System calls
CPU time	No peripheral devices, so no I/O
User CPU time	Only user tasked program contribution
System CPU time	Time spent by operating system on actions requested by user program



# Over what tests do we measure computational effectiveness

---

- Consider
  - Real Programs
  - Modified Applications
  - Kernels
  - Toy Benchmarks
  - Synthetic Benchmarks

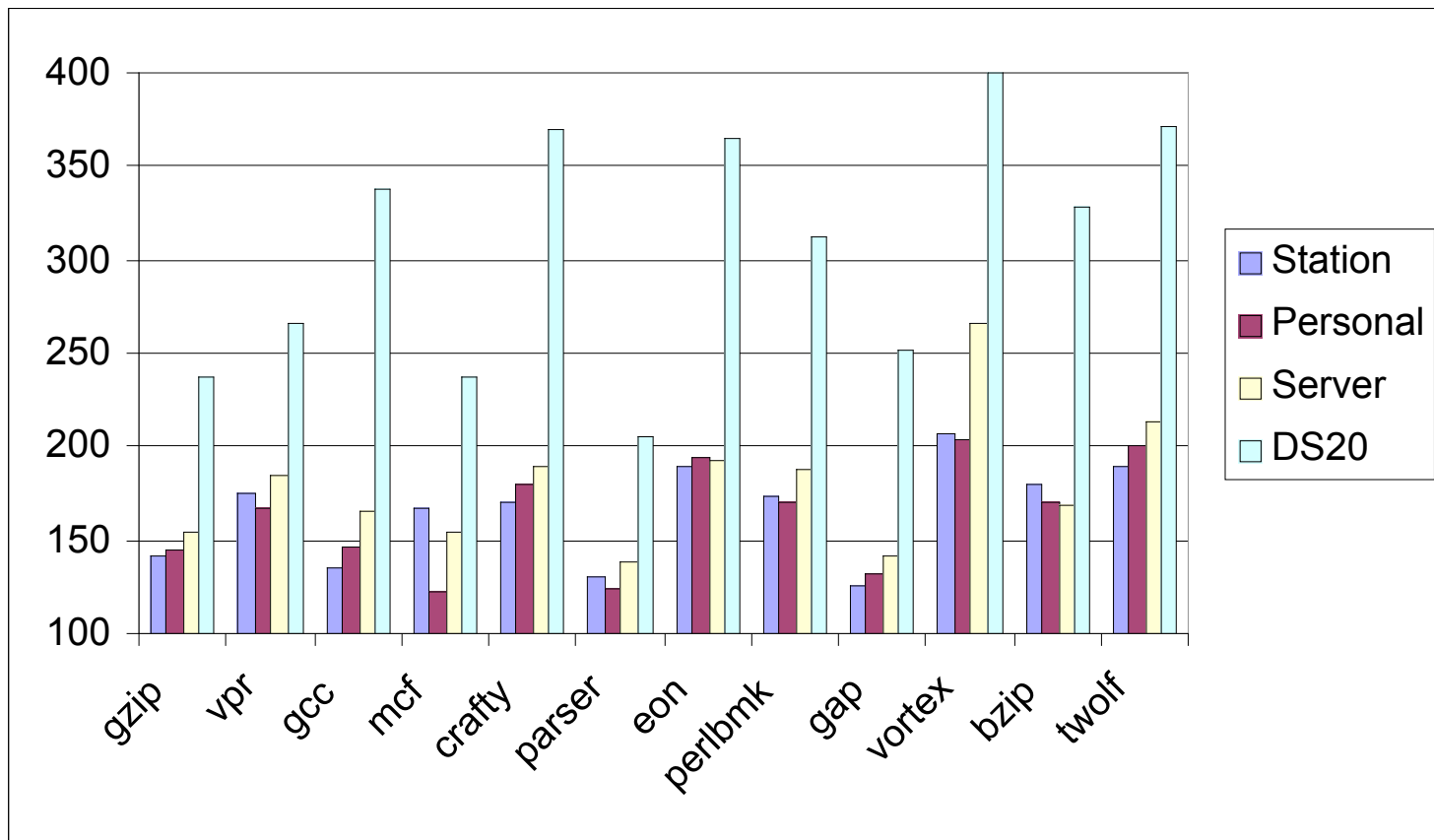


# Benchmarking Case Study

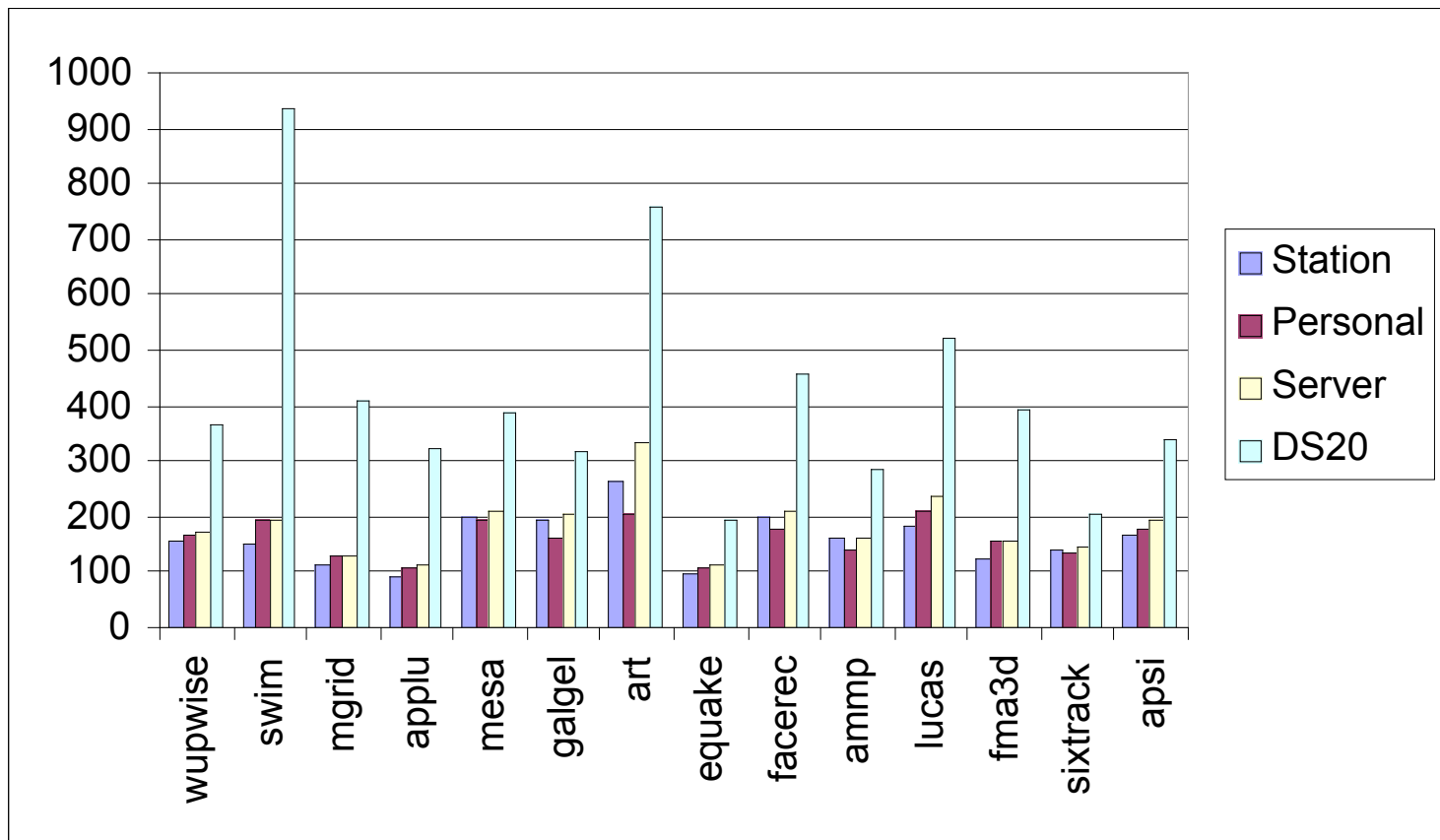
---

- SPEC CPU 2000
- See: John L. Henning, SPEC CPU 2000: Measuring CPU Performance in the New Millennium, IEEE Computer, pp 28-35, July 2000.
- Following figures further summarize figure 1.

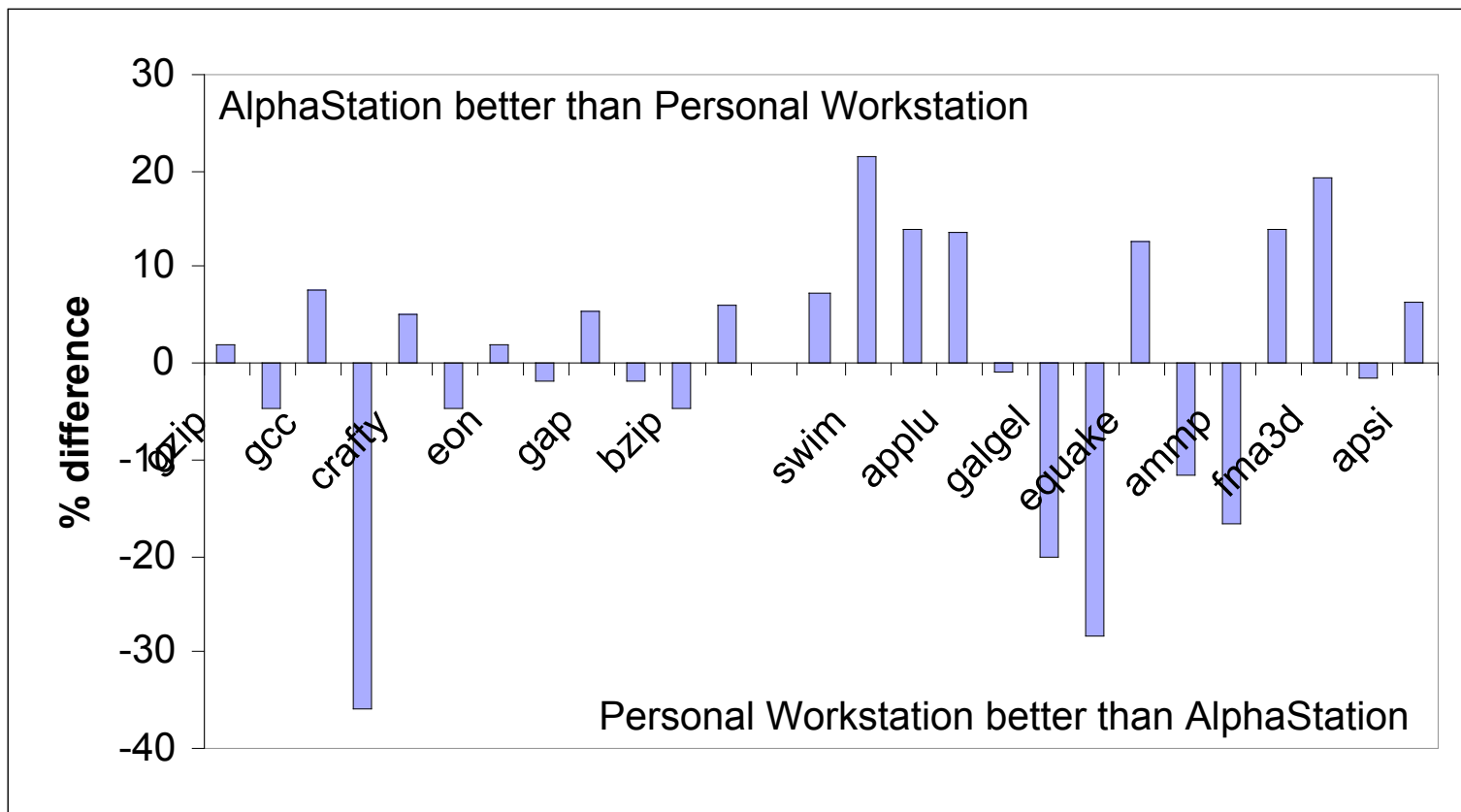
# Performance of Alpha systems on SPECint2000



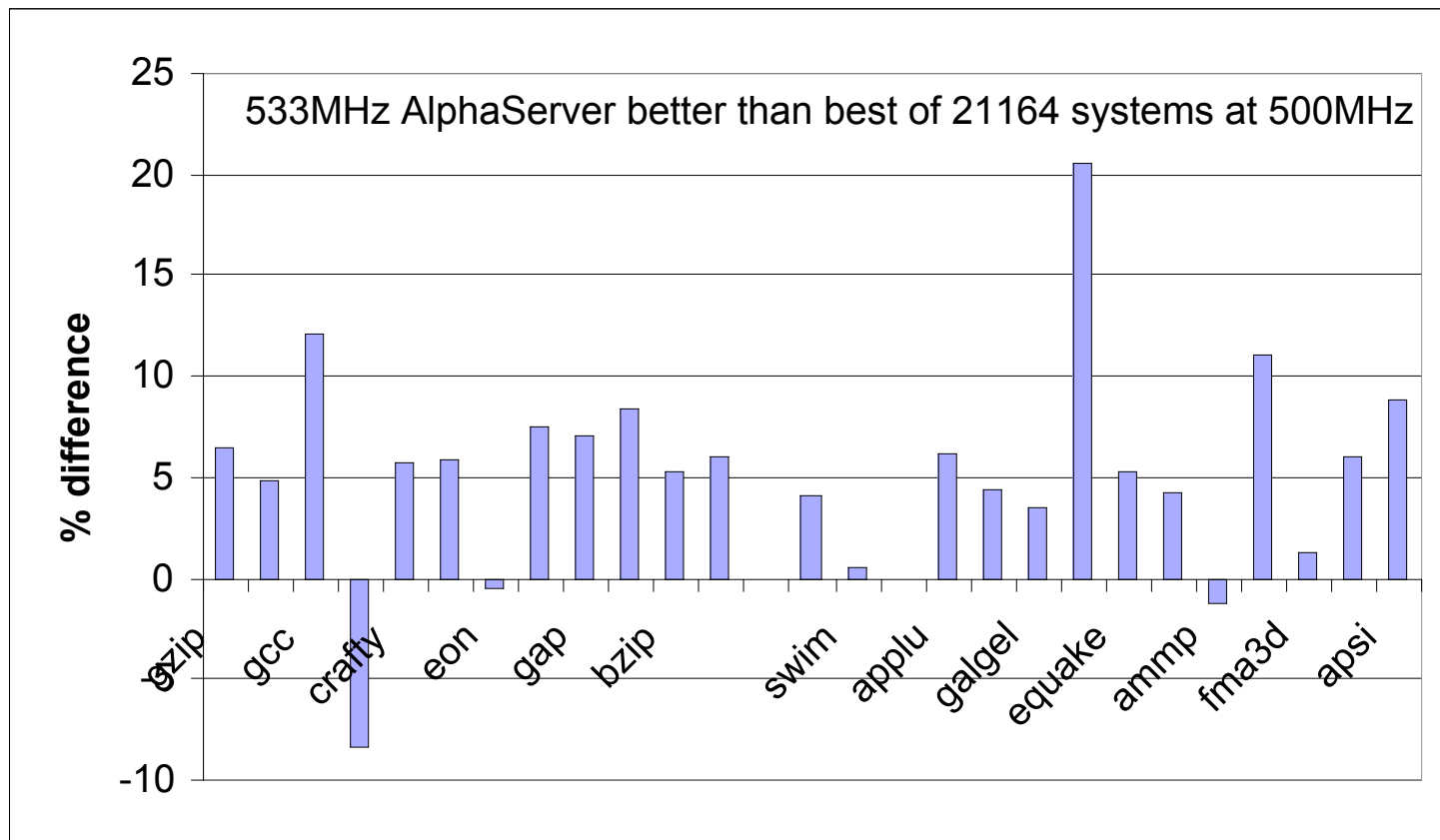
# Performance of Alpha systems on SPECfp2000



# AlphaStation against Personal Workstation alone



# 533MHz AlphaServer against the best of AlphaStation or Personal Workstation





# Quantitative Computer Design

---

- Make the typical cases most effective
- Amdahl's law
  - Improvements to the overall computational effectiveness is limited to the fraction of time the improvement can be used.

Speedup =  $\frac{\text{Execution time for entire task without enhancement}}{\text{Exe. Time when using enhancement when possible}}$



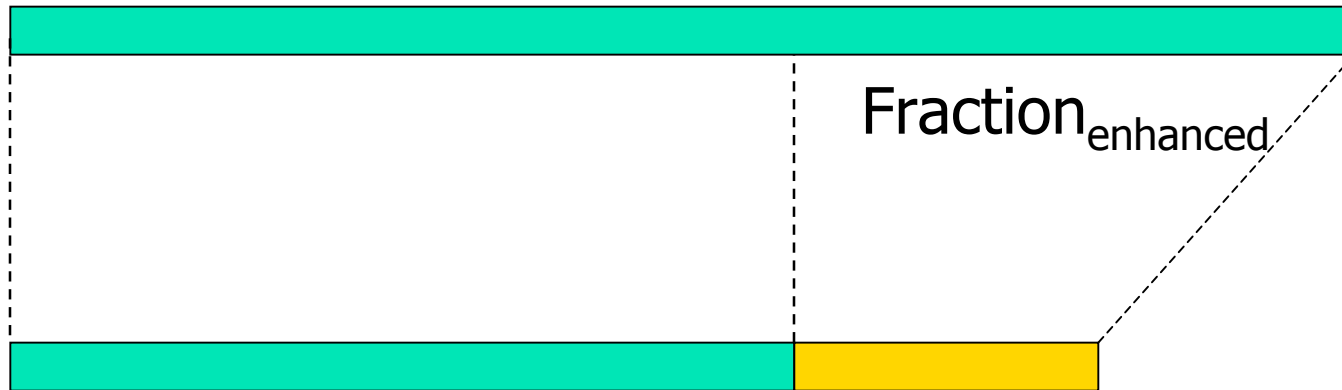
# Interpreting Amdahl's law (1)

---

- Fraction of time in the original scheme converted to take advantage of the enhancement
  - $\text{Fraction}_{\text{enhanced}} \leq 1$
- The improvement if all the task could utilize the enhancement
  - $\text{Speedup}_{\text{enhanced}} > 1$

# Interpreting Amdahl's law (2)

Time to perform task under original configuration,  $t(\text{old})$



fraction of time at the old rate + Fraction of time in the enhanced

$$1 - \text{Fraction}_{\text{enhanced}} + \text{Fraction}_{\text{enhanced}} / \text{Speedup}_{\text{enhanced}}$$



## Interpreting Amdahl's law (3)

---

- Execution time of enhanced system ( $t_{\text{new}}$ ) with respect to old rate ( $t_{\text{old}}$ )

$$\begin{aligned} t_{\text{new}} &= t_{\text{old}} \left\{ (1 - \text{Fraction}_{\text{enhanced}}) \right. \\ &\quad \left. + \text{Fraction}_{\text{enhanced}} / \text{Speedup}_{\text{enhanced}} \right\} \\ &= t_{\text{old}} / \text{Speedup}_{\text{overall}} \end{aligned}$$



# CPU performance Equation

---

- CPU computational efficiency is a function of 3 parameters
  - Clock Cycle Time – clock period
  - Clock Cycles per Instruction – CPI
  - Instruction Count – IC
- Interested in the interaction of these three parameters from the perspective of CPU operation



# Clock Period

---

- Discrete steps at which computational events take place.
  - Ticks, clock ticks, clock periods, clock cycle, ( or inverse: clock frequency, clock rate)
- More interested in CPU time to execute a program,

$$\begin{aligned}\text{CPU time} &= \text{CPU clk cycles in program} \times \text{clk period} \quad \textcircled{1} \\ &= \frac{\text{CPU clk cycles in program}}{\text{Clk freq}}\end{aligned}$$



# Clock Cycles per Instruction

---

- 1 to many relationship between clk cycles and instruction execution.
- LET **instruction count (IC)** be the number of program instructions

$$\text{CPI} = \frac{\text{Num CPU clk cycles to run program}}{\text{Instruction Count}} \quad \textcircled{2}$$

- Where CPI is an average



# CPU Performance Equation

---

- Substituting (2) in (1) leaves,

$$\begin{aligned}\text{CPU time} &= \text{CPU clk cycles in program} \times \text{clk period} \\ &= \text{Instruction Count} \times \text{CPI} \times \text{clk period} \\ &= \text{IC} \times \text{CPI} \times \text{clk(tpd)}\end{aligned}$$

- How easy are these parameters to estimate?

# Factors effecting CPI & IC

Factor	IC	CPI
Algorithm	# instructions to describe task	Instruction type I.e. precision
Language	One to many relation to IC	Data abstraction
Compiler	Efficiency of compiler	Sensitivity to sequencing and mix
ISA	RISC versus CISC ?	



# Pragmatics

---

- All three features are inter-related and EQUALLY significant
  - Cannot effect one feature independently of others
- Pipeline stalls and Cache misses make reliable estimation of CPI **impossible** without benchmarking



# Conclusion

---

- Must be careful when estimating computational efficiency
  - Appropriateness of the metric
  - Pipelining and cache technology complicate any performance estimation
- 3 inter-related elements central to CPU performance
  - Clock Cycle Time
  - Clock Cycles per Instruction
  - Instruction Count
- Compiler design now a significant contributor to CPU 'performance'