# EEL 6764 Principles of Computer Architecture

Dr Hao Zheng
Computer Sci. & Eng.
U of South Florida

# About This Course

- Principles of modern computer architectures
  - → All the “things” behind modern computers
- Quantitative analysis approaches to design tradeoffs in terms of
  - → Cost, performance, power/energy efficiency etc.
- Why this course?
  - → Have better sense of future by learning from the past
  - → Design better computers to solve real-world problems
  - → Write better software
  - → Get a good grade in order to continue your grad. study
    - ➤ A more tangible motivation

# Course Objectives

- Understand various aspects of modern computer architecture design
  - → Instruction set design, mem hierarchy, pipelining, multi-processing, etc
- Understand quantitative approaches to analysis and evaluation of design tradeoffs
  - → Cost, performance, power/energy, etc
- Understand the interplays between the computer architecture & software
  - → How they affect each other

# Instructor

- Dr. Hao Zheng
  - → Office: ENB 312
  - → Office Hours: 10-11:30 and 2:00 – 3:30 pm, Wed
  - → Drop email for appointment
  - → Phone: 974-4757
  - → Email: haozheng@usf.edu

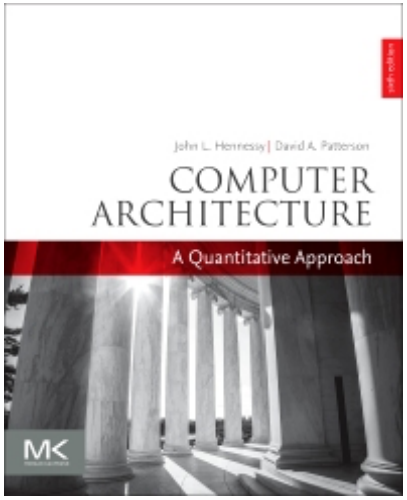
- Research interests – design of STAR systems
  - → **S**ecure
  - → **T**rustworthy
  - → **A**utonomous
  - → **R**esilient

# Course Administration

- TA/Grader: Shamaria Engram
- Email: sengram@mail.usf.edu
- Office: ENB 327

  → Office Hours: 4-6pm Tue. & Thr

# Course Material




- Text:
  - → Computer Architecture: A Quantitative Approach by H ennessy & Patterson, 6th Edition, Morgan Kaufmann ( **Required**).
  - → Computer Organization and Design, 5th Edition: The Hardware/Software Interface, Morgan Kaufmann by Hennessy & Patterson (**Recommended**)
- Additional material will be posted
- Partial lectures and reading material will be posted before class

# Evaluation and Grading Information

| | Weights | Dates |
|---|---|---|
| Homework | 15% | |
| Quizzes | 10% | |
| Mid-term Exam | 25% | October 1st, 2018 |
| Final Exam (comprehensive) | 35% | Dec. 3, 3 – 5pm |
| Term paper | 15% | TBD |

- Let me know about midterm exam conflicts ASAP
- For homework/quiz grades talk to the TA.
  → All HW must be submitted electronically to Canvas.
- Other queries: TA and Instructor

# Final Grading Scale

| | | | | | |
|---|---|---|---|---|---|
| $x \geq 95\%$ | **A+** | $90\% \leq x < 95\%$ | **A** | $87\% \leq x < 90\%$ | **A-** |
| $84\% \leq x < 87\%$ | **B+** | $80\% \leq x < 84\%$ | **B** | $77\% \leq x < 80\%$ | **B-** |
| $74\% \leq x < 77\%$ | **C+** | $70\% \leq x < 74\%$ | **C** | $65\% \leq x < 70\%$ | **C-** |
| $x < 65\%$ | **F** | | | | |

*Incomplete (I) grades will NOT be given.*

No curves!

# What You Should Know

- Course prerequisites
  - → CDA 3201 Logic Design
  - → CDA 4205 Computer Architecture
- Basic logic design & computer organization
  - → Information representation, Boolean logic
  - → Logic building blocks: flip-flops, multiplexers, decoders, shift registers
  - → Datapaths, register transfer level design
  - → Knowledge of digital systems in terms of computer organization and logic level design
  - → Instruction sets: opcodes, operands, memory addressing, etc
  - → Input/output and communication; memory hierarchy.
- Basic knowledge of assembly programming.

# Expectations from You

- Do all homeworks (start early!),
- Do not miss quizzes
- Attend lectures (come **on time**),
- Take notes whenever necessary.
- Prepare for classes/quizzes.
- Ask questions, participate in class.
- For exams, read notes, book and supplementary materials.
- This will be a heavy course – work hard!
- Be honest to yourselves!

# More on Homework Assignments

- Do them to truly understand the material, not just to get the grade.
- Content from lectures, readings, discussions.
- All homework write-ups *must* be your own work, written up individually and independently.
  - → Peer discussions are encouraged.
  - → Copying each others' work or from previous work is strictly forbidden!
- There is a fine line between **plagiarism** and **collaboration**.
- **No late homework submissions are accepted.**

# Cheating and Academic Dishonesty

- **Absolutely no form of cheating will be tolerated**
- You are all adults and we will treat you so
- See syllabus, USF Policy, and CSE Academic Integrity Policy
- Cheating → failing grade (no exceptions)
  - → You may get **FF**, resulting in dismissal from the program.

# Course Content

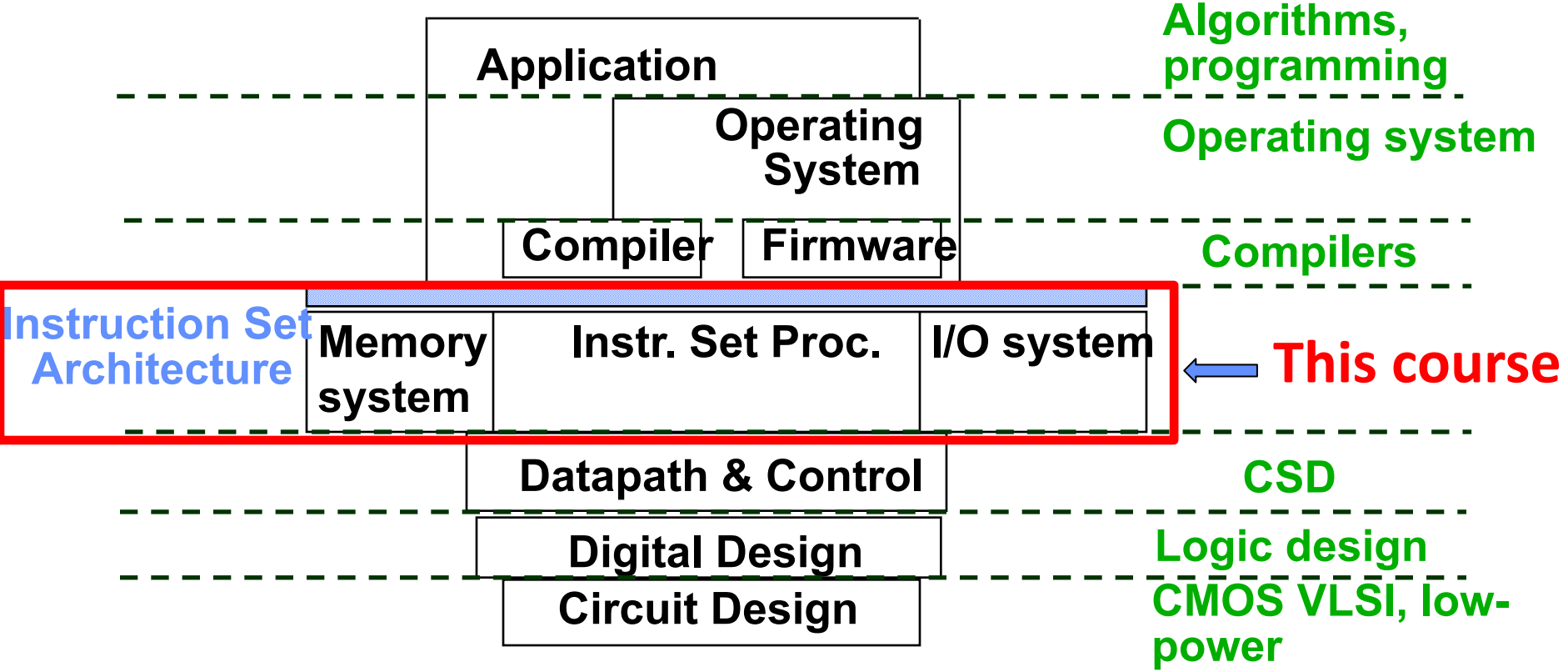
| | |
|---|---|
| Introduction | Ch. 1 |
| Memory hierarchy design | Ch. 2, App. B |
| Instruction set principles | App. A |
| Instruction level parallelism and pipelining | Ch. 3, App. C |
| Data level parallelism and vector processing | Ch 4, App. G |
| Thread-level parallelism and multiprocessors | Ch. 5 |
| Interconnection networks | App. F |

Other current topics may be introduced if time permits.

# Reading

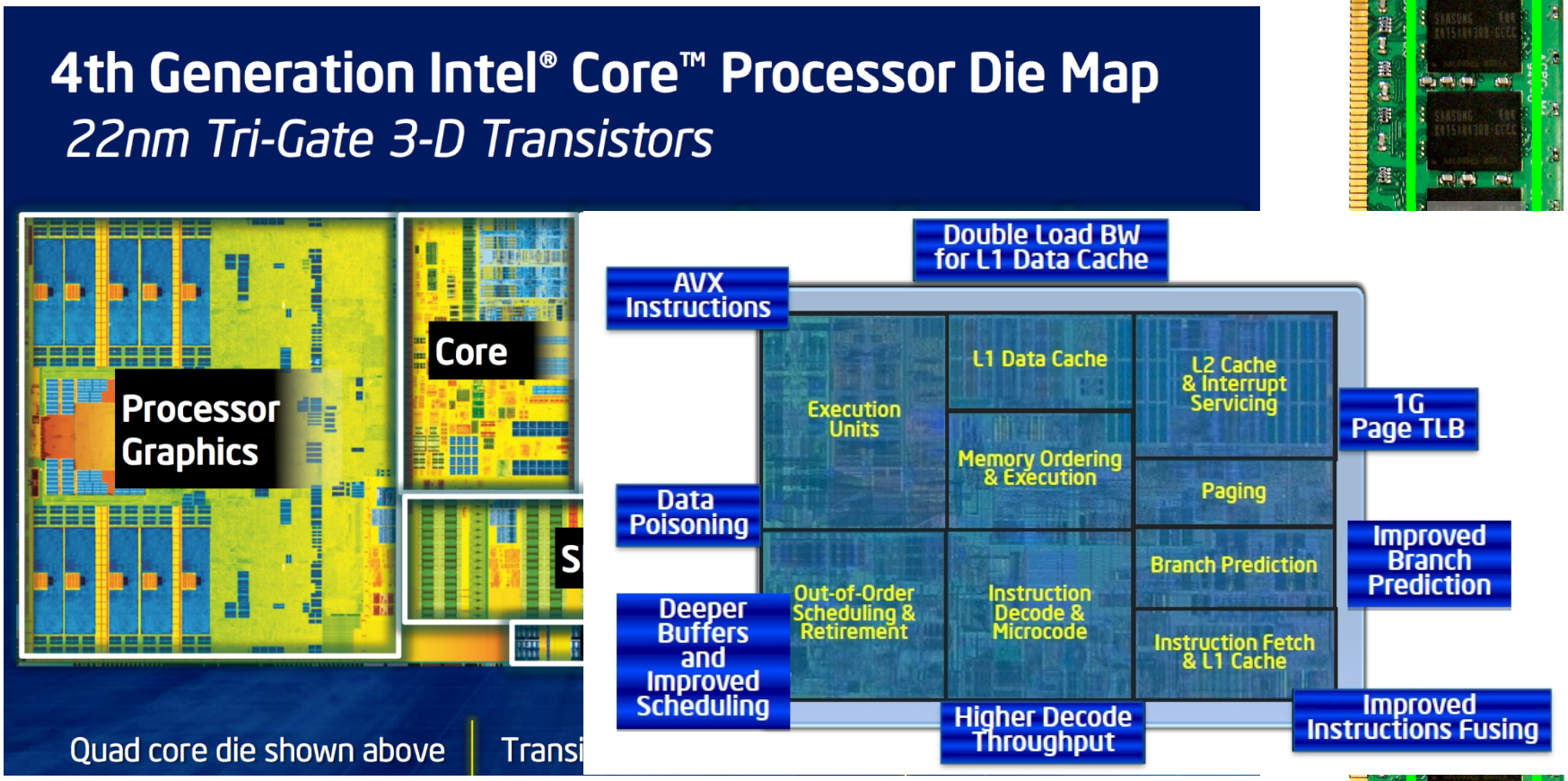
- Chapter 1

# How Do the Pieces Fit Together?

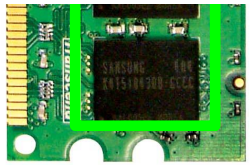



- Coordination of many *levels of abstraction*
- Under a rapidly changing set of forces
- Design, measurement, *and* evaluation

# An Example

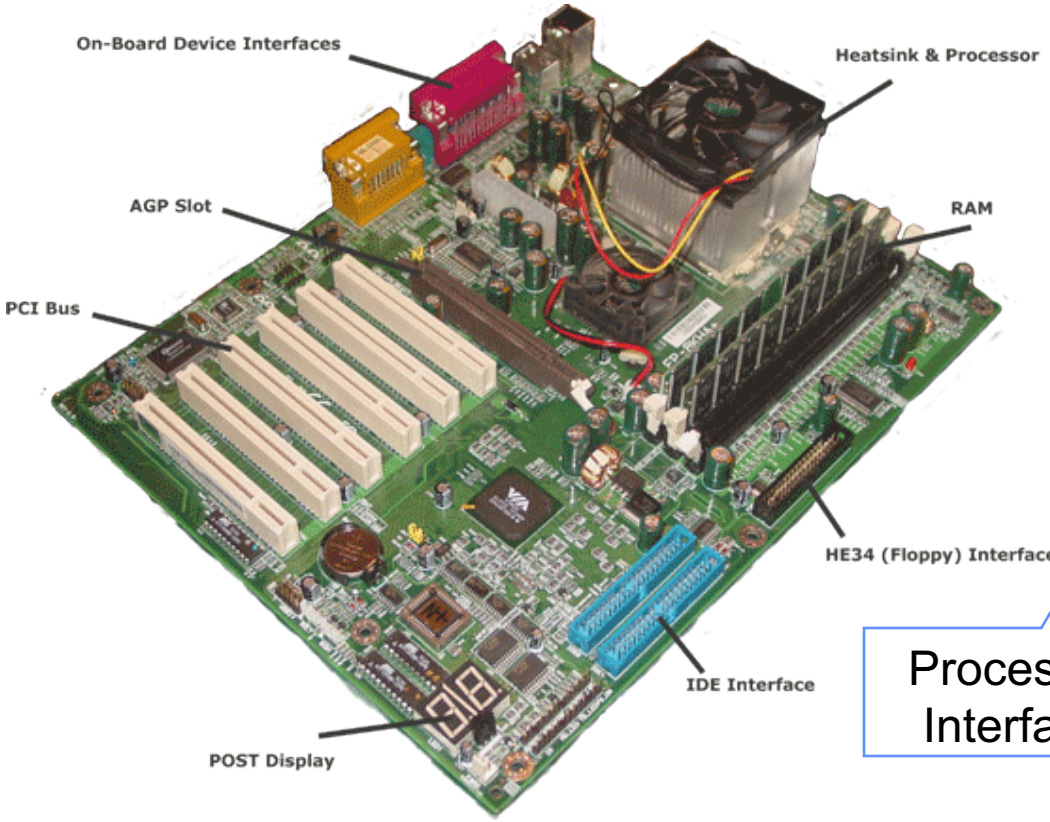


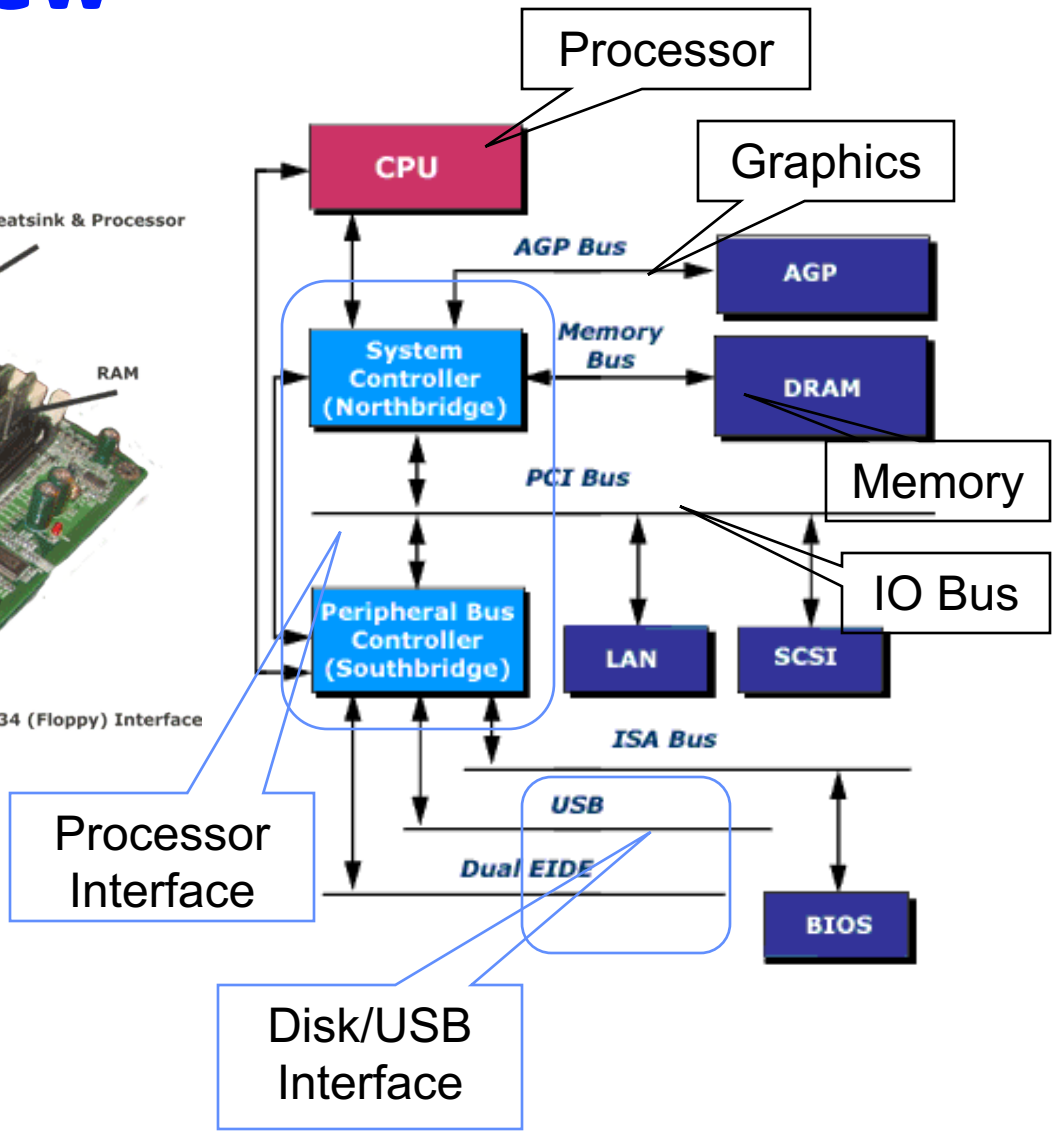


- The objective of this course is to touch each of these components

# A System-Level View



# Computer Architecture

- What is it?
- Why is it important?

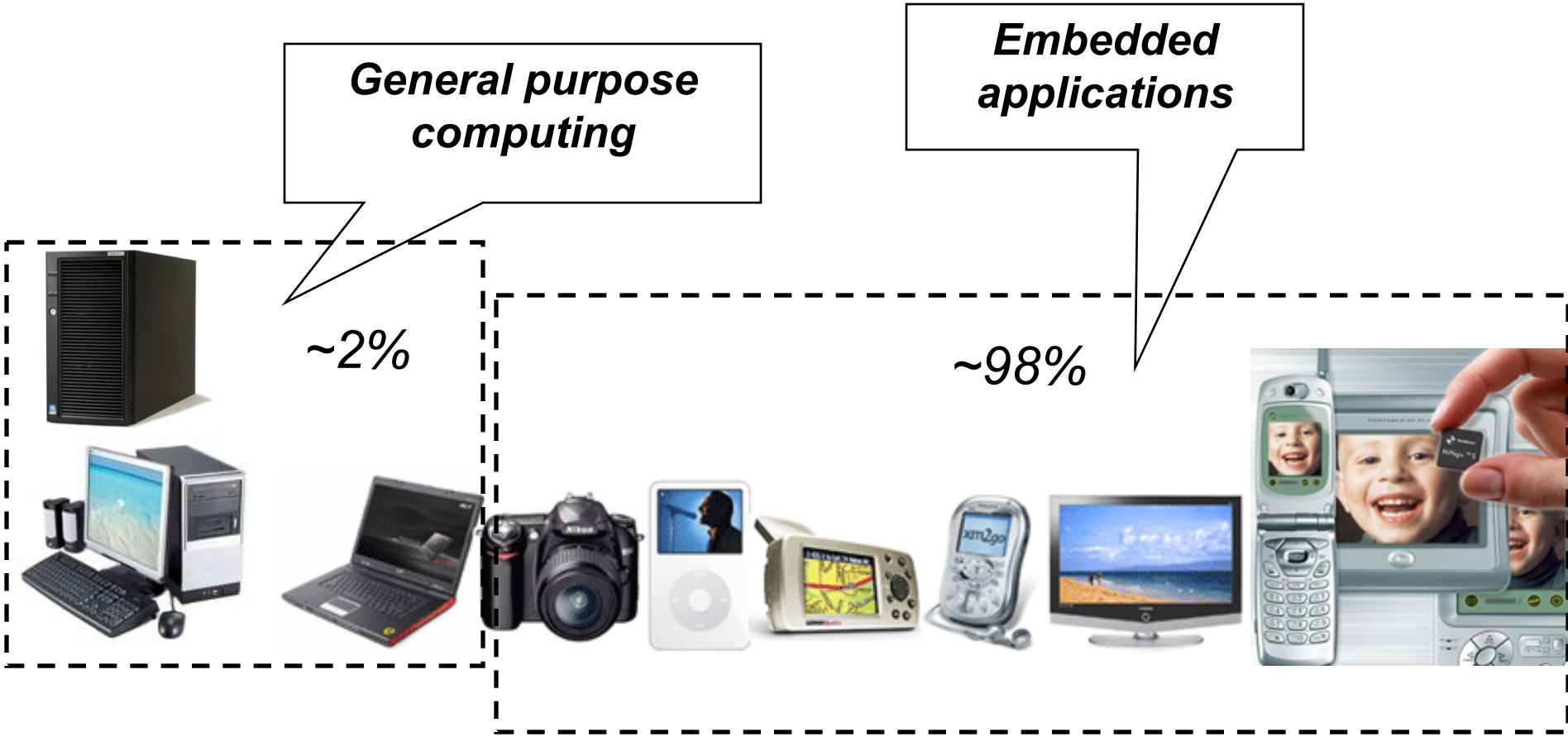
# What is Computer Architecture?

*"Computer architecture is the science and art of selecting and interconnecting hardware components to create a computer that meets functional, performance, and cost goals."* *

* G. Sohi, UW

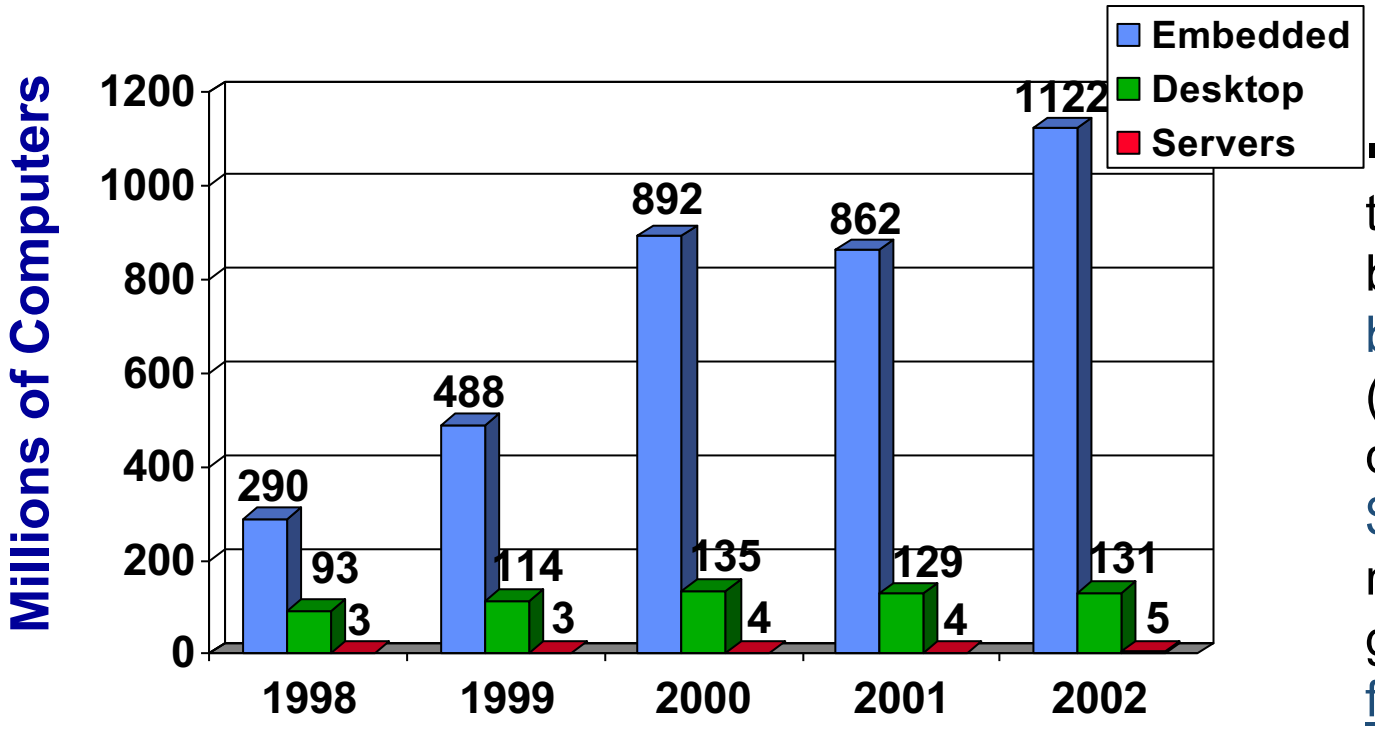
Think about architecture of buildings!

# Ubiquitous Usage of Computers




- Today's high-end computers have
  - → Billions of transistors in a chip
  - → Tera-flops (10^12 FPOS) processing power

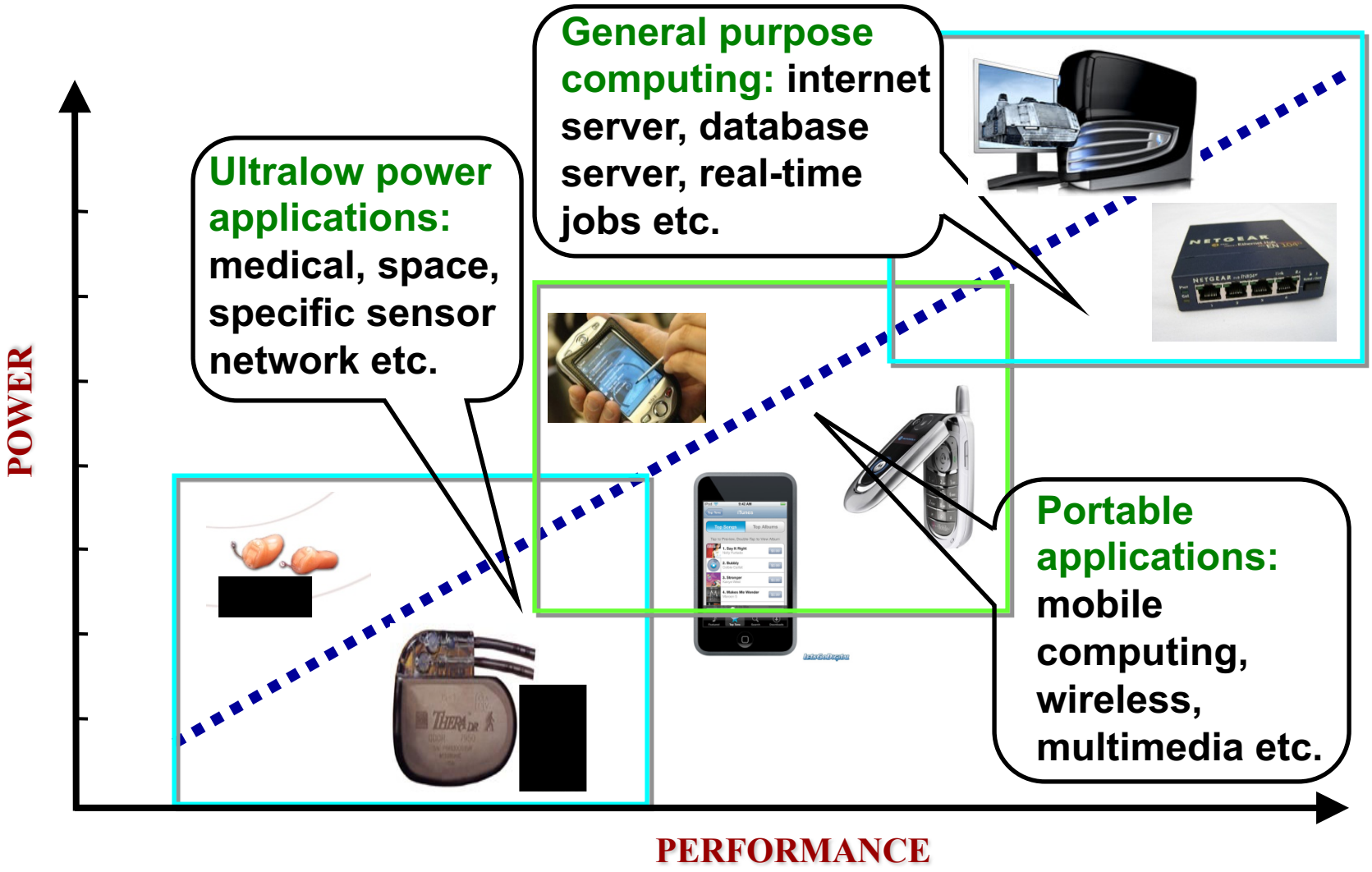
# Where is the Market?




- It is estimated that there will be between 15 and 20 billion devices (small embedded devices) with a $900 Billion USD market, which is growing twice as fast as the PC market.

- Embedded applications: mobile electronics, automobile electronics, home electronics, communications and networking, healthcare products, sensor networks, gaming hardware, defense applications and many others!

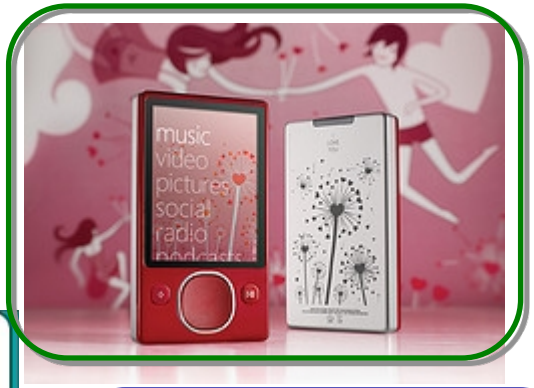
# Computer Applications



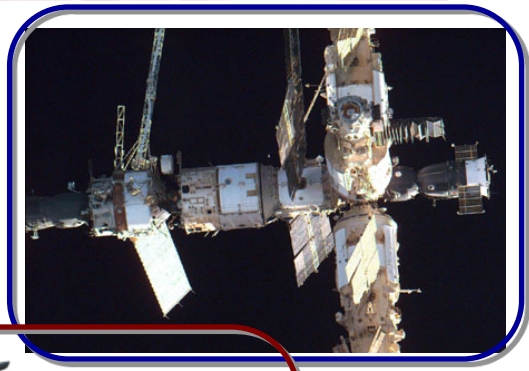

Different applications have different power-performance demands

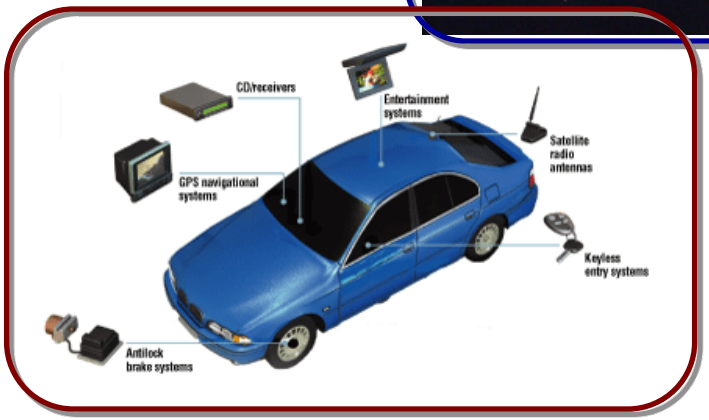
# System Design Parameters

- Performance (Speed)
- Cost
- Power (static + dynamic)
  - → Peak power
  - → Average power
- Robustness
  - → Noise-tolerance
  - → Radiation-hardness
- Testability
- Reconfigurability
- Time-to-market etc.







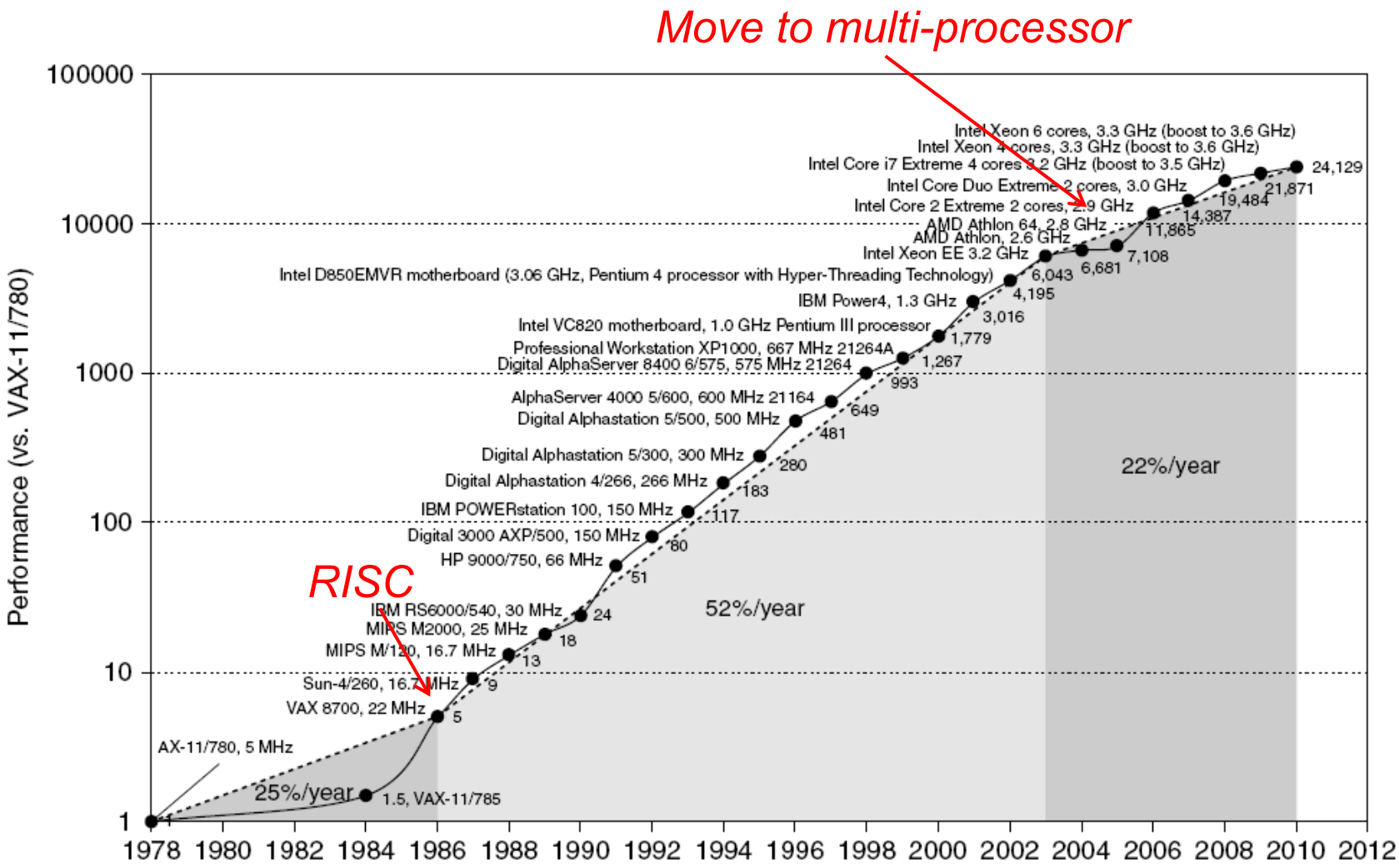


# Classes of Computers

| Feature | Personal mobile device (PMD) | Desktop | Server | Clusters/warehouse-scale computer | Embedded |
|---|---|---|---|---|---|
| Price of system | $100–$1000 | $300–$2500 | $5000–$10,000,000 | $100,000–$200,000,000 | $10–$100,000 |
| Price of micro-processor | $10–$100 | $50–$500 | $200–$2000 | $50–$250 | $0.01–$100 |
| Critical system design issues | Cost, energy, media performance, responsiveness | Price-performance, energy, graphics performance | Throughput, availability, scalability, energy | Price-performance, throughput, energy proportionality | Price, energy, application-specific performance |

- Shipments: 1.9B PMD, 350M  Desktop, 19B Embedded.
- Desktop has largest $ market share.

# Single Processor Performance



# Computer Technology Driving Forces

- Improvements in semiconductor technology
  → Feature size, clock speed, cost
- Improvements in computer architectures
  → Enabled by high-level language compilers, UNIX
  → Lead to RISC architectures

- Together have enabled:
  → More powerful and efficient computers.
  → New classes of computers, i.e. mobile devices, etc.
  → Penetration of GP CPUs into many applications.
  → Tradeoff between performance and productivity in SW development.

# Current Trends in Architecture

- Power Wall
- Memory Wall
- Lack of Instruction-Level parallelism (ILP) to exploit
  → Single processor performance improvement ended in 2003

- New models for performance:
  → Data-level parallelism (DLP)
  → Thread-level parallelism (TLP)
  → Request-level parallelism (RLP)

- These require explicit restructuring of the applications
  → Applications must expose parallelism explicitly.

# Classes of Parallelism

- Exploitation of parallelism --> performance

- Classes of parallelism in applications:
  - → *Data-Level Parallelism* (DLP)
  - → *Task-Level Parallelism* (TLP)

- Classes of architectural parallelism:
  - → *Instruction-Level Parallelism* (ILP)
  - → *Vector architectures/Graphic Processor Units* (GPUs)
  - → *Thread-Level Parallelism (TLP)*
  - → *Request-Level Parallelism (RLP)*

# Flynn's Taxonomy

- *Single instruction stream, single data stream (SISD)*
  - → Exploit ILP and TLP in some degree
- *Single instruction stream, multiple data streams (SIMD)*
  - → Targets DLP
  - → Vector architectures
  - → Multimedia extensions
  - → Graphics processor units
- *Multiple instruction streams, single data stream (MISD)*
  - → No commercial implementation
- *Multiple instruction streams, multiple data streams (MIMD)*
  - → Targets TLP and RLP
  - → Tightly-coupled MIMD - TLP
  - → Loosely-coupled MIMD - RLP

| | SD | MD |
|---|---|---|
| SI | SISD | SIMD |
| MI | X | MIMD |

# Computer Architecture – Past and Now

- "Old" view of computer architecture:
  - → Instruction Set Architecture (ISA) design
  - → i.e. decisions regarding:
    - ➤ registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding

- "Real" computer architecture:
  - → ISA design is less of a focus
  - → Meet specific requirements of the target machine
  - → Design to find a best tradeoff among performance, cost, power, and availability, etc, optimized for target applications
  - → Consider **ISA**, **microarchitecture**, logic/circuit design, implementation, etc.

Impacts from requirements of applications and technology.

# Technology Trends

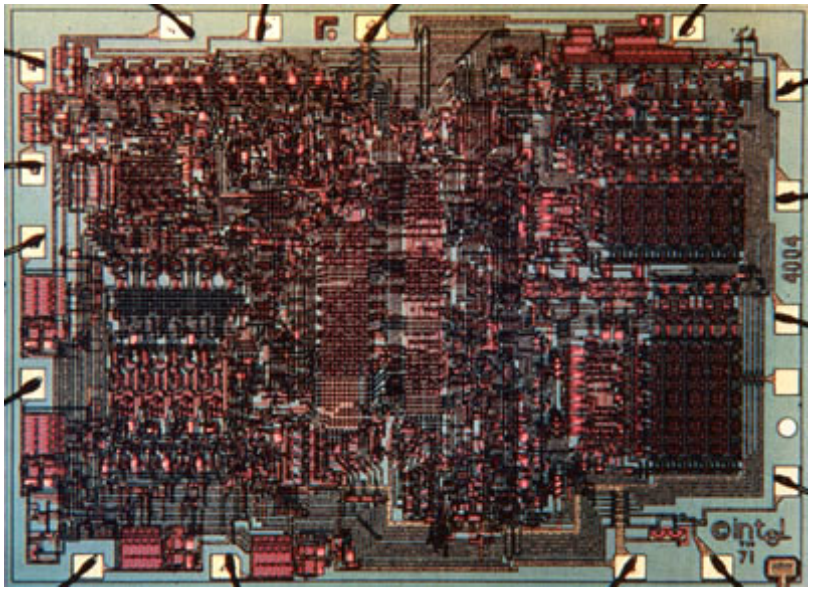
- Integrated circuit technology
  - → Transistor density: +35%/year
  - → Die size: +10-20%/year
  - → Integration overall: +40-55%/year
- DRAM capacity: +25-40%/year (slowing)
  - → Foundation of main memory.
- Flash capacity: +50-60%/year
  - → 8-10X cheaper/bit than DRAM
  - → An order of magnitude slower than DRAM
- Magnetic disk technology: +40% -> 5%/yr
  - → 15-25X cheaper/bit then Flash
  - → 300-500X cheaper/bit than DRAM
  - → Main storage for server or WSC.

*Important to design for the next generation of technology!*

*Current technologies approaching their limits; new technologies being researched*

# First Microprocessor

- Intel 4004 (1971)
  - Application: calculators
  - Technology: 10000 nm
  - 2300 transistors
  - 13 $mm^2$
  - 108 KHz
  - 12 Volts
  - 4-bit data
  - Single-cycle datapath



# Height of Single-Core Processor

- Intel Pentium4 (2003)
  - Application: desktop/server
  - Technology: 90nm (1/100th of 4004)

  - 55M transistors (20,000x)
  - 101 $mm^2$ (10x)
  - 3.4 GHz (10,000x)
  - 1.2 Volts (1/10th)

  - 32/64-bit data (16x)
  - 22-stage pipelined datapath
  - 3 instructions per cycle (superscalar)
  - Two levels of on-chip cache
  - data-parallel vector (SIMD) instructions, hyperthreading

# Transistors and Wires

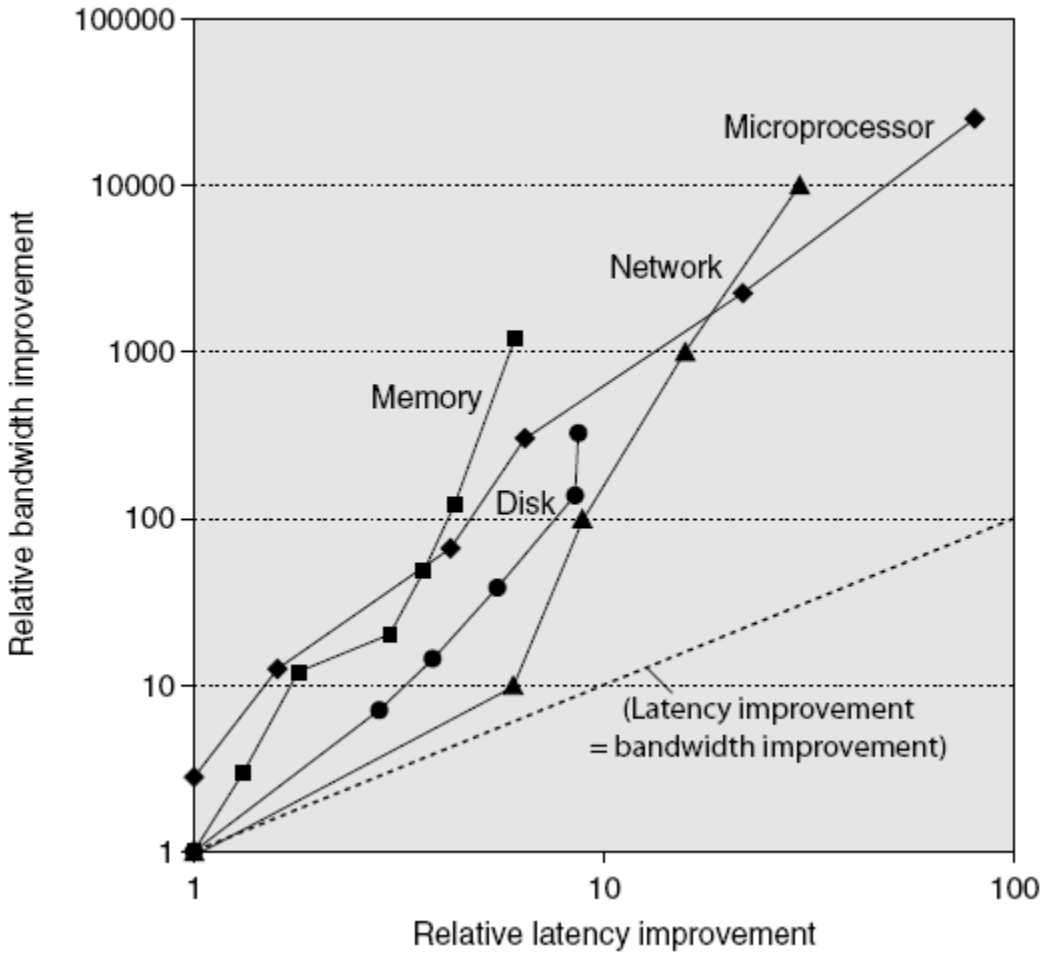
- Transistor feature size
  - → Minimum size of transistor or wire in x or y dimension
  - → 10 microns in 1971 to .032 microns in 2011
  - → Now in 2017/2018, seeing 10nm – 7nm.
- Transistor density grows exponentially.
  - → Moore's law – has been slowing down
- Transistor performance scales linearly
- Wire delay does not improve with feature size!
  - → In fact, it is getting worse!
  - → Make on-chip interconnect design an important task!

# Bandwidth and Latency

- Bandwidth or throughput
  - → Total work done in a given time
  - → Important for servers and data center operators

- Latency or response time
  - → Time between start and completion of an event
  - → Important for individual users

# Bandwidth and Latency




*Log-log plot of bandwidth and latency milestones*

- Bandwidth
  - → 10,000-25,000X improvement for processors
  - → 300-1200X improvement for memory and disks
- Latency
  - → 30-80X improvement for processors
  - → 6-8X improvement for memory and disks
- Improvement in Bandwidth = square of improvement in latency

# Power and Energy

- Problem:  Get power in, distribute it, get it out

- Thermal Design Power (TDP)
  - → Characterizes sustained power consumption
  - → Used as target for power supply and cooling system
  - → Lower than peak power, higher than average power consumption

- Clock frequency can be reduced dynamically to limit power consumption

- Energy efficiency is often a better measurement
  - → Power = a design constraint

# Dynamic Energy and Power

- Dynamic energy
  - → Transistor switch from 0 -> 1 or 1 -> 0
  - → E = ½ x Capacitive load x Voltage$^2$
- Dynamic power
  - → P = ½ x Capacitive load x Voltage$^2$ x Frequency switched
- Relation between energy and power consumption

$$P \approx E \times f$$

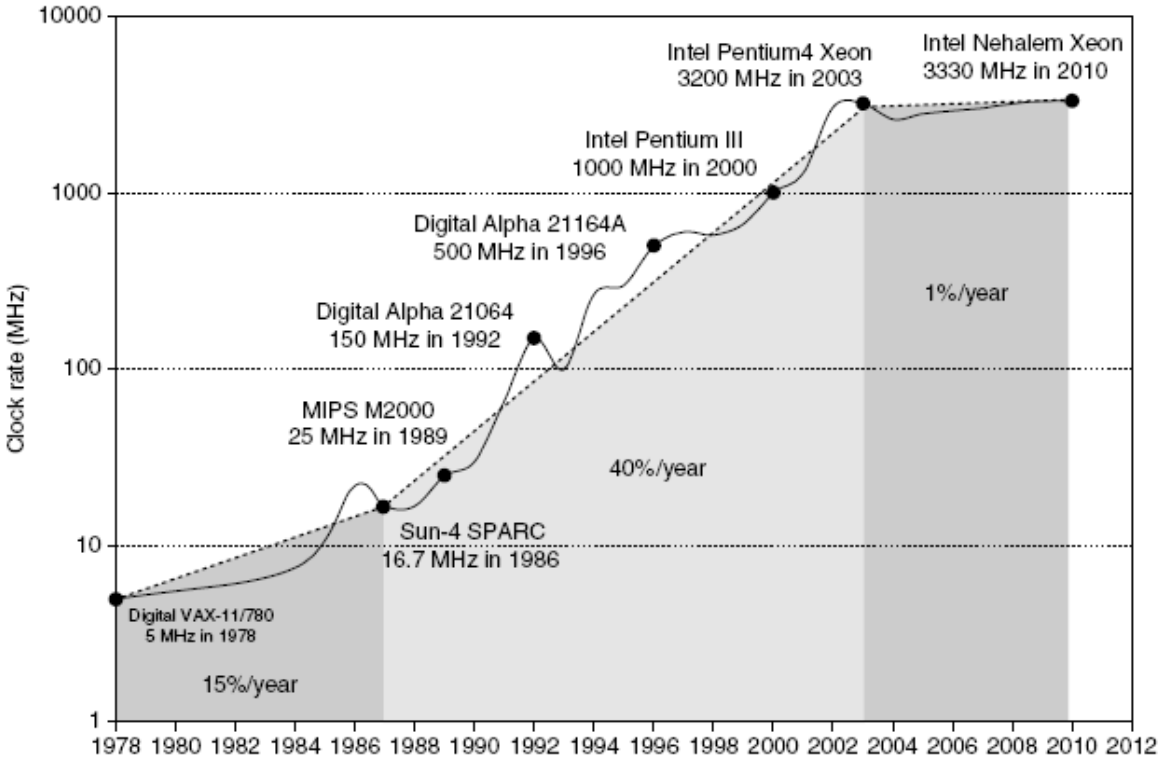
  - → *P*: power, *E*: energy, *f*: switching frequency
- Reducing clock frequency reduces power, not energy

# Power and Energy

- Suppose two designs have the same clock frequency. Design 1 consumes, on average, 20% less than energy per operation than Design 2.
  - → Design 1's power is 20% less than design 2.
  - → Design 1 dissipates less heat with the same performance of design 2.
  - → Or, design 1's clock frequency can be improved by 25% while consuming the same power as design 2.

# Power

- Intel 80386 consumed ~2 W
- 3.3 GHz Intel Core i7 consumes 130 W
- Heat must be dissipated from 1.5 x 1.5 cm chip
- This is the limit of what can be cooled by air.
- Performance improvement by increasing *f* is over!



# Reducing Power

- Techniques for reducing power:
  - → Turn off components not being used
  - → Dynamic Voltage-Frequency Scaling
  - → Low power state for DRAM, disks
  - → Overclocking, turning off all but one core

Energy efficiency has become a critical quality metric

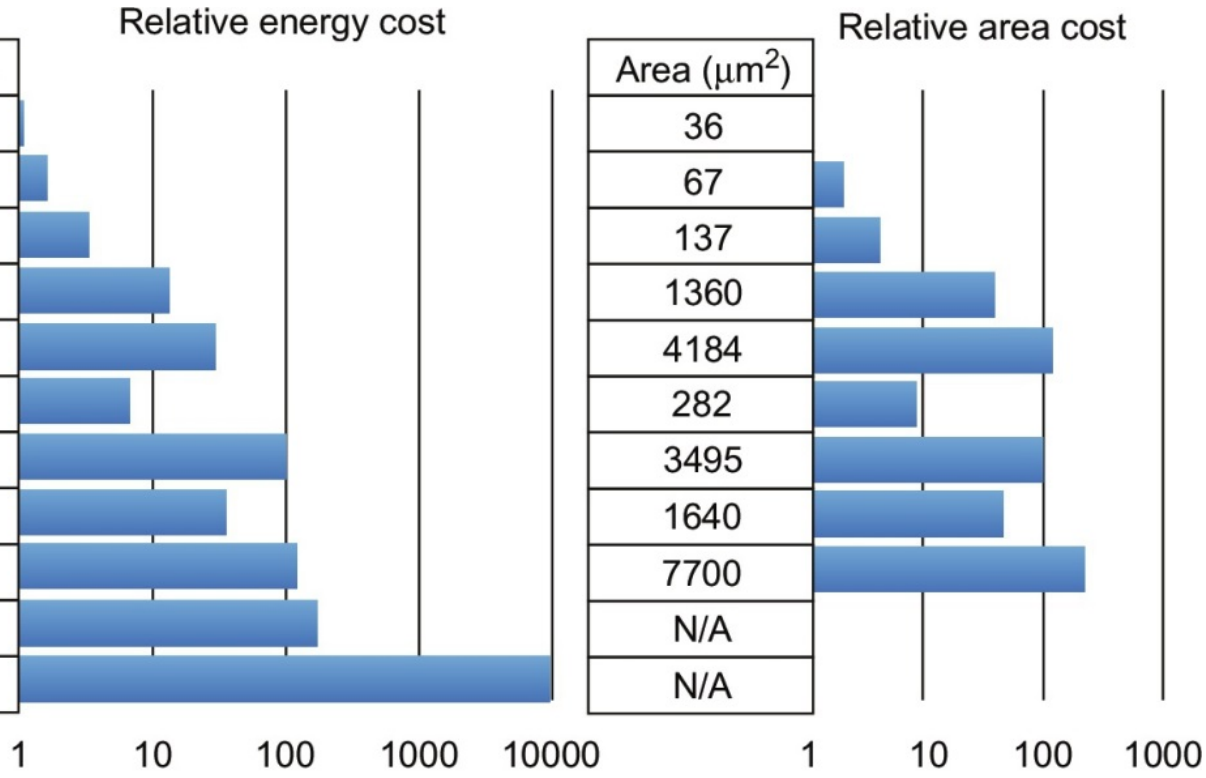
- Measured by tasks/joule or performance/watt

# Static Power

- Transistors are not perfect switches
  → leakage current even when no switching
- Static power consumption
  → $\text{Current}_{\text{static}}$ x Voltage
  → Scales with number of transistors
  → To reduce: power gating
- Static power
  → Increasing share of overall system power.

# Where are Energy/Area Consumed?

| Operation: | Energy (pJ) | Area ($\mu m^2$) |
|---|---|---|
| 8b Add | 0.03 | 36 |
| 16b Add | 0.05 | 67 |
| 32b Add | 0.1 | 137 |
| 16b FB Add | 0.4 | 1360 |
| 32b FB Add | 0.9 | 4184 |
| 8b Mult | 0.2 | 282 |
| 32b Mult | 3.1 | 3495 |
| 16b FB Mult | 1.1 | 1640 |
| 32b FB Mult | 3.7 | 7700 |
| 32b SRAM Read (8KB) | 5 | N/A |
| 32b DRAM Read | 640 | N/A |

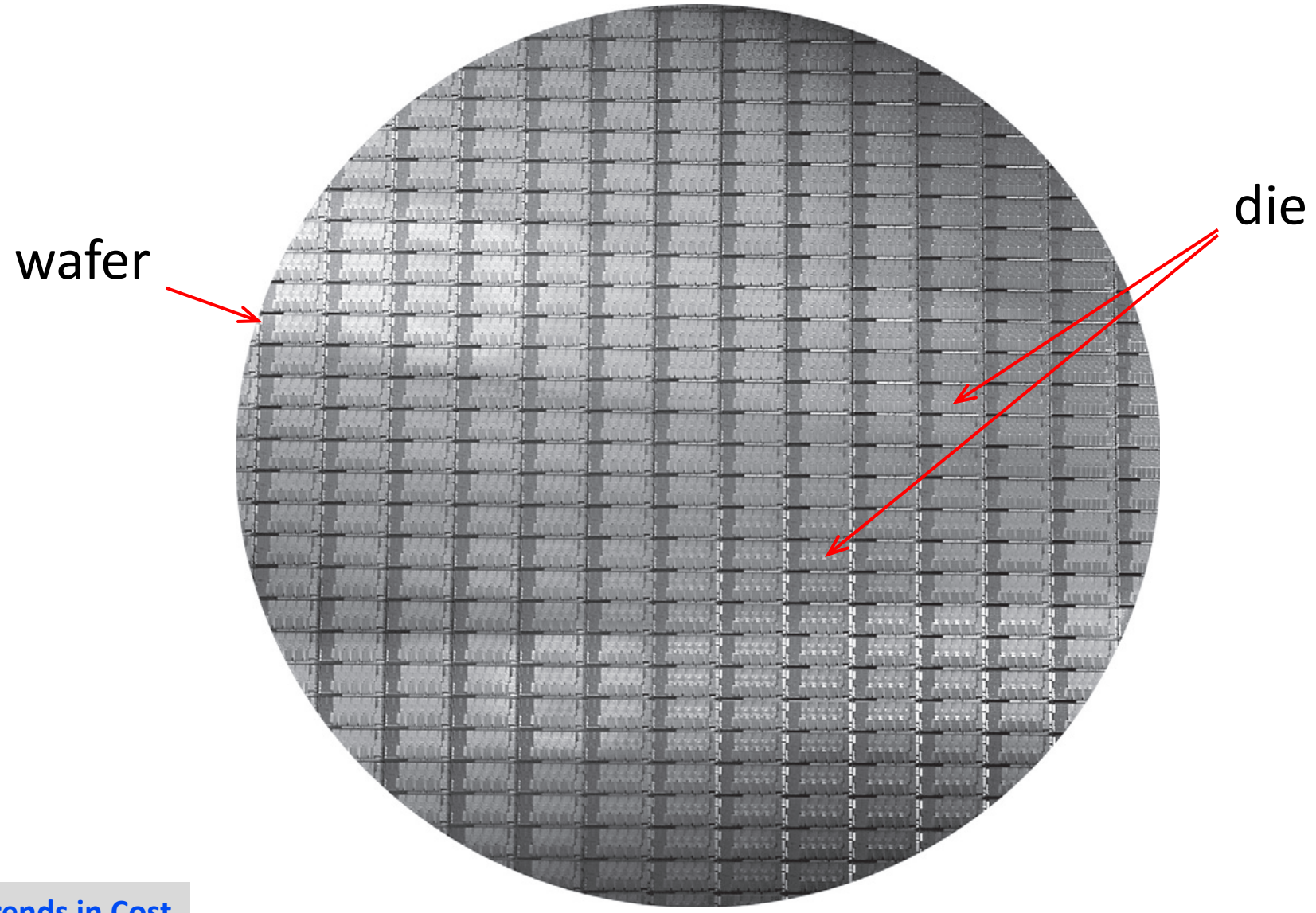



Energy numbers are from Mark Horowitz *Computing's Energy problem (and what we can do about it)*. ISSCC 2014
Area numbers are from synthesized result using Design compiler under TSMC 45nm tech node. FP units used DesignWare Library.

# Factors Affecting Cost

- Cost: design, manufacturing, material, etc.
  → Design – one time cost
  → Manufacturing ... – recurring cost
- Cost driven down by learning curve
  → Yield: ratio between good products among all.
- Increase in volume -> reduced unit cost
  - 10% less for each doubling of volume

# Integrated Circuit: Wafer and Die



# Cost of Integrated Circuit Cost

- Cost of Integrated circuit

$$\text{Cost of integrated circuit} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging and final test}}{\text{Final test yield}}$$

$$\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2 \times \text{Die area}}}$$

- Bose-Einstein formula:

$$\text{Die yield} = \text{Wafer yield} \times 1/(1 + \text{Defects per unit area} \times \text{Die area})^N$$

→ Defects per unit area = 0.016-0.057 defects per square cm (2010)

→ N = process-complexity factor = 11.5-15.5 (40 nm, 2010)

# Dependability

- As feature size shrinks, computers fail more often.

- Module reliability
  - → Mean time to failure (MTTF)
  - → Mean time to repair (MTTR)
  - → Mean time between failures (MTBF) = MTTF + MTTR
  - → Availability = MTTF / MTBF
  - → = a ratio between service time and total life span

- To improve reliability: redundancy.

# Measuring Performance

- Typical performance metrics:
  → **Response (execution) time** & **throughput**
- Speedup of X relative to Y
  → Execution $time_Y$ / Execution $time_X$
- Execution time
  → **Wall clock time**: includes all system overheads
  → **CPU time**: only computation time
- Benchmarks
  → Kernels (e.g. matrix multiply)
  → Toy programs (e.g. sorting)
  → Synthetic benchmarks (e.g. Dhrystone)
  → Benchmark suites (e.g. SPEC06fp, TPC-C)

Can be mis-leading

# Principles of Computer Design

- **Take Advantage of Parallelism**
  → e.g. multiple processors, disks, memory banks, pipelining, multiple functional units
- **Principle of Locality**
  → A property of programs: data and instructions typical reused
- **Focus on the Common Case**
  → **Amdahl's Law**

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

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

# Amdahl's Law - Example

Suppose a new processor is 10 times faster than the current one. Assume that the current processor is 40% time busy with computation and 60% of the time idle waiting for IO. What is the overall speedup with the new processor?

# Principles of Computer Design

- **The Processor Performance Equation**

$$\text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time}$$

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time}$$

ISA compiler

ISA arch

technology arch

# Principles of Computer Design

- Different instruction types having different CPIs

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

$$\text{CPU time} = \left( \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$

→ $IC_i$ : count of instruction *i* in a program.

→ $CPI_i$ : average cycle count for instruction *i*

Knowledge of $IC_i$ and $CPI_i$ is helpful for optimization.

Suppose the following measurements

freq of FP ops = 25%

Average CPI of FP ops = 4 cycles

Average CPI of other inst = 1.33 cycles

freq of FPSQR = 2%

Average CPI of FPSQR = 20 cycles

Design #1: decrease CPI of FPSQR to 2

Design #2: decrease CPI of all FP to 2.5

Compare these two design alternatives.

# Summary

- Computer architecture involves ISA, microarchitecture, HW technologies.
  - → Is about making tradeoffs among design parameters optimized for target applications.
  - → Should be mindful about technology trends, and their impacts on comp. design.
- Classification of comp. arch. wrt different types of parallelism they exploit.
- Reviews of trends of various system parameters.
- Overviews of computer design principles
  - → Exploitation of parallelism, locality, optimization for common cases, etc.