18-640 Foundations of Computer Architecture

Lecture 1:
"Introduction To Computer Architecture"

John Paul Shen
August 27, 2014

- Required Reading Assignment:
  - **Chapters 1 and 2 of Shen and Lipasti (SnL).**
- Recommended References:
  - "Amdahl's and Gustafson's Laws Revisited" by Andrzej Karbowski. (2008)
  - "High Performance Reduced Instruction Set Processors" by Tilak Agerwala and John Cocke. (1987)

Electrical & Computer ENGINEERING
Carnegie Mellon University

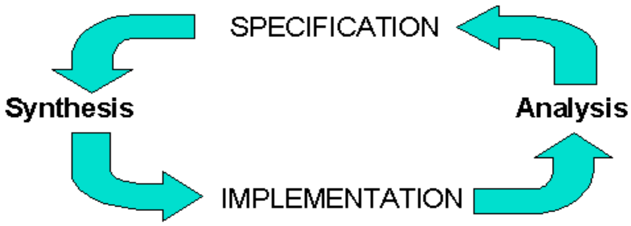
18-640 Foundations of Computer Architecture

Lecture 1:
"Introduction To Computer Architecture"

A. **Instruction Set Architecture (ISA)**
   a. Hardware / Software Interface
   b. Dynamic / Static Interface (DSI)
B. **Historical Perspective on Computing**
   a. Major Epochs
   b. Processor Performance Iron Law (#1)
   c. Course Coverage
C. **"Economics" of Computer Architecture**
   a. Amdahl's Law and Gustafson's Law
   b. Moore's Law and Bell's Law

Electrical & Computer ENGINEERING
Carnegie Mellon University

Anatomy of Engineering Design



Specification: Behavioral description of *"What does it do?"*

Synthesis: Search for possible solutions; pick best one. *Creative process*

Implementation: Structural description of *"How is it constructed?"*

Analysis: Validate if the design meets the specification.
"Does it do the right thing?" + "How well does it perform?"

Lecture 1: "Introduction to Computer Architecture"

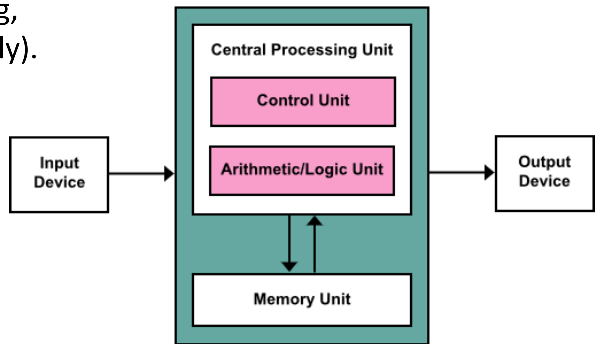
A. Instruction Set Architecture (ISA)

a. Hardware / Software Interface
b. Dynamic / Static Interface (DSI)

Electrical & Computer ENGINEERING

The Classic Von Neumann Computation Model

- Proposed in 1945 by John Von Neumann and others (Alan Turing, J. Presper Eckert and John Mauchly).
- A "**Stored Program Computer**"
  1. **One CPU**
     - One Control Unit
       - Program Counter
       - Instruction Register
     - One ALU
  2. **Monolithic Memory**
     - Data Store
     - Instruction Store
  3. **Sequential Execution Semantics**



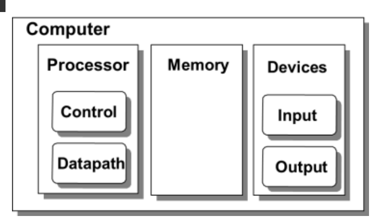
Computer Architecture: Instruction Set Architecture

Instruction-Set Processor Design

Application Software
- Program development
- Program compilation

SPECIFICATION — Instruction Set Architecture (ISA)

Hardware Technology
- Program execution
- Machine performance



IMPLEMENTATION

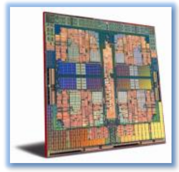
[Gerrit Blaauw & Fred Brooks, 1981]

Art and Science of Instruction Set Processor Design

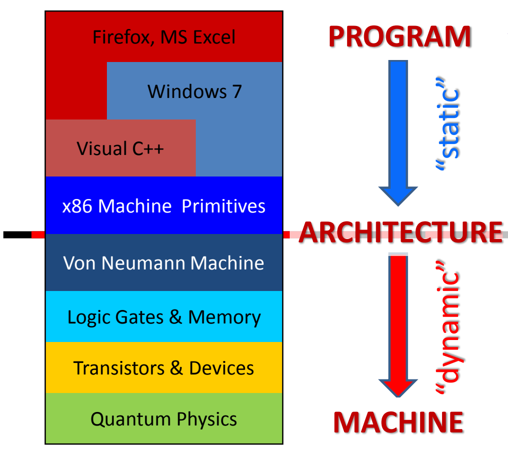
ARCHITECTURE (ISA) <u>programmer/compiler view</u>
- Functional programming model to application/system programmers
- Opcodes, addressing modes, architected registers, IEEE floating point

IMPLEMENTATION (μarchitecture) <u>processor designer view</u>
- Logical structure or organization that performs the ISA specification
- Pipelining, functional units, caches, physical registers, buses, branch predictors

REALIZATION (Chip) <u>chip/system designer view</u>
- Physical structure that embodies the implementation
- Gates, cells, transistors, wires, dies, packaging



Computer Architecture: Dynamic-Static Interface



Architectural state requirements:
- Support sequential instruction execution semantics.
- Support precise servicing of exceptions & interrupts.

Architectural State — Exposed to SW

Dynamic/Static Interface (DSI)=(ISA)

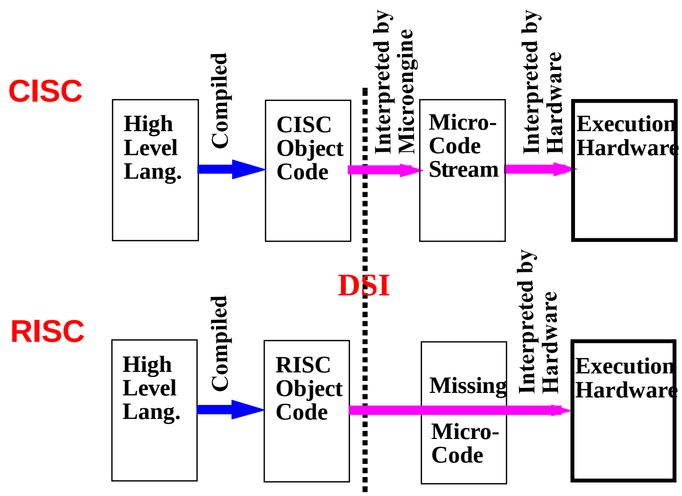
Microarchitecture State — Hidden in HW

Buffering needed between arch and uarch states:
- Allow uarch state to deviate from arch state.
- Able to undo speculative uarch state if needed.

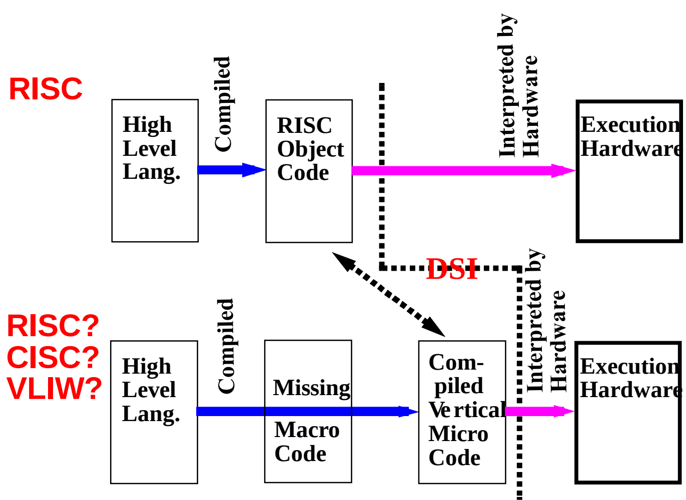
DSI = ISA = a contract between the program and the machine.

RISC vs. CISC *Transition from CISC to RISC:*

[Josh Fisher, HP]

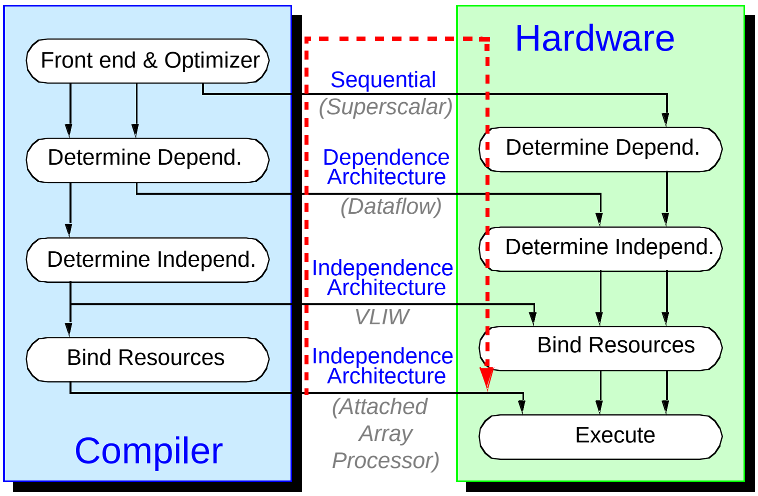


Another way to view RISC



[B. Rau & J. Fisher, 1993]

HW vs. SW and Dynamic vs.Static Design Space



Lecture 1: “Introduction to Computer Architecture”

B. Historical Perspective on Computing

a. Major Epochs
b. Processor Performance Iron Law (#1)
c. Course Coverage

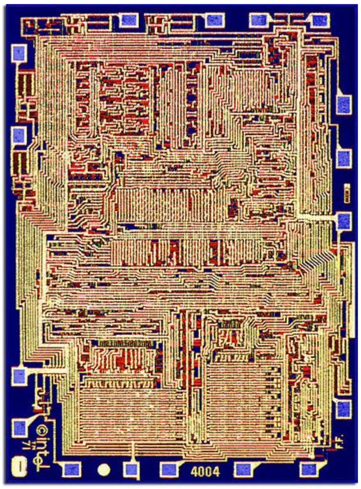
Electrical & Computer ENGINEERING

Almost Seven Decades of Modern Computing . . .

Historical Perspective on the Last Five Decades

- The Decade of the 1960's: *"Computer Architecture Foundations"*
  - Von Neumann computation model, programming languages, compilers, OS's
  - Commercial Mainframe computers, Scientific numerical computers
- The Decade of the 1970's: *"Birth of Microprocessors"*
  - Programmable controllers, bit-sliced ALU's, single-chip processors
  - Emergence of Personal Computers (PC)
- The Decade of the 1980's: *"Quantitative Architecture"*
  - Instruction pipelining, fast cache memories, compiler considerations
  - Widely available Minicomputers, emergence of Personal Workstations
- The Decade of the 1990's: *"Instruction-Level Parallelism"*
  - Superscalar, speculative microarchitectures, aggressive compiler optimizations
  - Widely available low-cost desktop computers, emergence of Laptop computers
- The Decade of the 2000's: *"Mobile Computing Convergence"*
  - Multi-core architectures, system-on-chip integration, power constrained designs
  - Convergence of smartphones and laptops, emergence of Tablet computers

Intel 4004, circa 1971

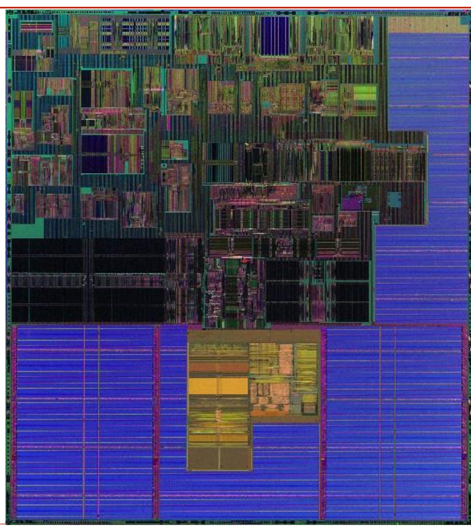


The first single chip CPU

- 4-bit processor for a calculator.
- 1K data memory
- 4K program memory
- 2,300 transistors
- 16-pin DIP package
- 740kHz (eight clock cycles per CPU cycle of 10.8 microseconds)
- ~100K OPs per second

Molecular Expressions: Chipshots

Intel Itanium 2, circa 2002



Performance leader in floating-point apps

- 64-bit processor
- 3 MByte in cache!!
- 221 million transistor
- 1 GHz, issue up to 8 instructions per cycle

In ~30 years, about 100,000 fold growth in transistor count!

http://cpus.hp.com/images/die_photos/McKinley_die.jpg

[John Crawford, Intel, 1993]

Performance Growth in Perspective

- Doubling every 18 months (1982-2000):
  - total of 3,200X
  - Cars travel at 176,000 MPH; get 64,000 miles/gal.
  - Air travel: L.A. to N.Y. in 5.5 seconds (MACH 3200)
  - Wheat yield: 320,000 bushels per acre
- Doubling every 24 months (1971-2001):
  - total of 36,000X
  - Cars travel at 2,400,000 MPH; get 600,000 miles/gal.
  - Air travel: L.A. to N.Y. in 0.5 seconds (MACH 36,000)
  - Wheat yield: 3,600,000 bushels per acre

Unmatched by any other industry!!

Convergence of Key Enabling Technologies

- CMOS VLSI:
  - Submicron feature sizes: 0.3u → 0.25u → 0.18u → 0.13u → 90n → 65n → 45n → 32nm…
  - Metal layers: 3 → 4 → 5 → 6 → 7 (copper) → 12 …
  - Power supply voltage: 5V → 3.3V → 2.4V → 1.8V → 1.3V → 1.1V …
- CAD Tools:
  - Interconnect simulation and critical path analysis
  - Clock signal propagation analysis
  - Process simulation and yield analysis/learning
- Microarchitecture:
  - Superpipelined and superscalar machines
  - Speculative and dynamic microarchitectures
  - Simulation tools and emulation systems
- Compilers:
  - Extraction of instruction-level parallelism
  - Aggressive and speculative code scheduling
  - Object code translation and optimization

"Iron Law" of Processor Performance

$$1/\text{Processor Performance} = \frac{\text{Time}}{\text{Program}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycle}}$$

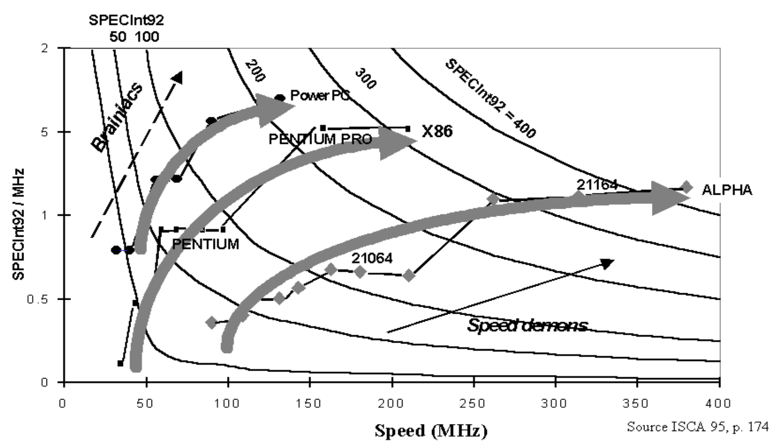
(inst. count) (CPI) (cycle time)

Architecture → Implementation → Realization

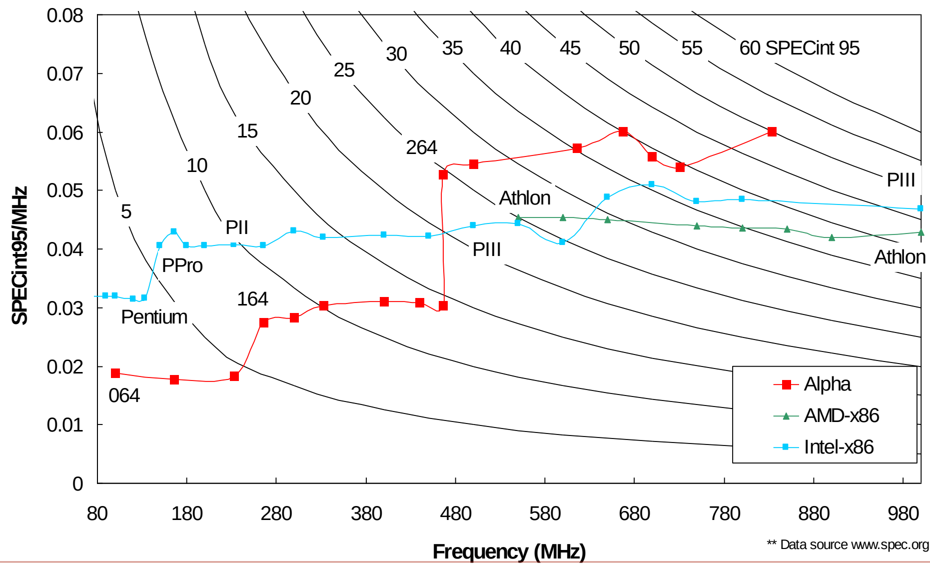
Compiler Designer Processor Designer Chip Designer

- In the 1980's (decade of pipelining):
  - CPI: 5.0 → 1.15
- In the 1990's (decade of superscalar):
  - CPI: 1.15 → 0.5 (best case)
- In the 2000's:
  - we learn the power lesson

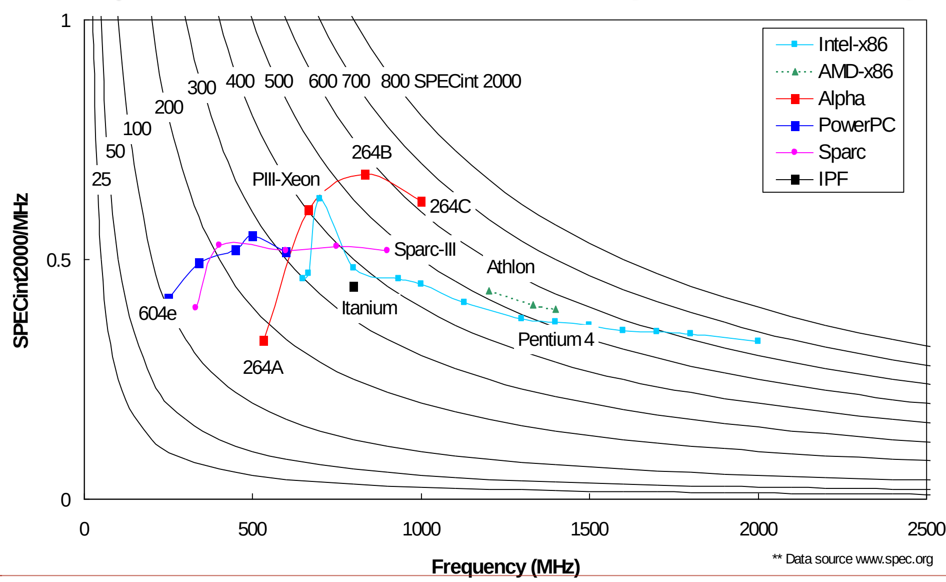
Landscape of Processor Families [SPECint92]



Landscape of Processor Families [SPECint95]



Landscape of Processor Families [SPECint2000]



Iron Law #1 – Processor (Latency) Performance

- Time to execute a program: T (latency)

$$T = \frac{instructions}{program} \times \frac{cycles}{instruction} \times \frac{time}{cycle}$$

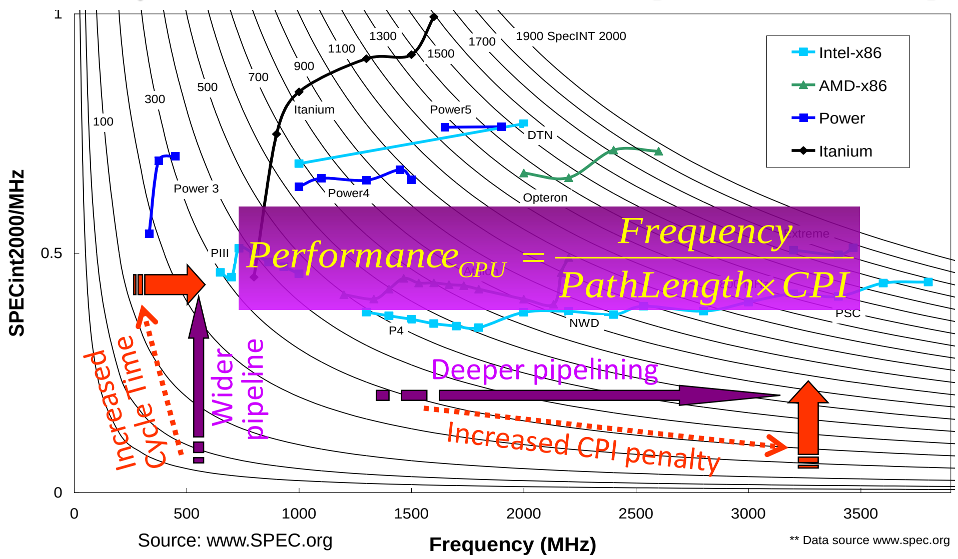
$$T = PathLength \times CPI \times CycleTime$$

- Processor performance: Perf = 1/T

$$Perf_{CPU} = \frac{1}{PathLength \times CPI \times CycleTime} = \boxed{\frac{Frequency}{PathLength \times CPI}}$$

[John DeVale & Bryan Black, 2005]

Landscape of Processor Families [SPECint2000]



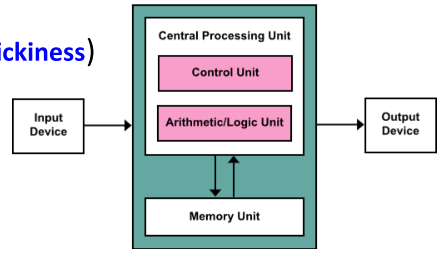
Source: www.SPEC.org

** Data source www.spec.org

18-640 Course Coverage: Processor Designs

Persistence of Von Neumann Model (**Legacy SW Stickiness**)

1. One CPU
2. Monolithic Memory
3. Sequential Execution Semantics



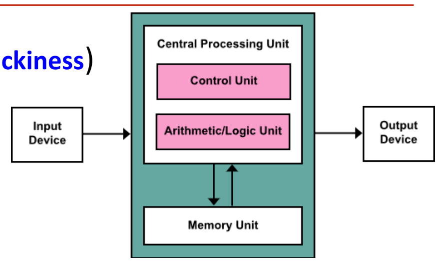
Evolution of Von Neumann Implementations:

- ~~SP: Sequential Processors~~ (direct implementation of sequential execution)
- PP: Pipelined Processors (overlapped execution of in-order instructions)
- SSP: Superscalar Processors (out-of-order execution of multiple instructions)
- MCP: Multi-core Processors = CMP: Chip Multiprocessors (concurrent multi-threads)
- SMP: Symmetric Multiprocessors (concurrent multi-threads and multi-programs)

18-640 Course Coverage: Parallelism Exploited

Persistence of Von Neumann Model (**Legacy SW Stickiness**)

1. One CPU
2. Monolithic Memory
3. Sequential Execution Semantics



Parallelisms for **Performance** (→ for **Power** Reduction → for **Energy** Efficiency)

- **ILP**: Basic Block (exploit ILP in PP, SSP)
- **ILP**: Loop Iteration (exploit ILP in SSP, VLIW)
- **DLP**: Data Set (exploit DLP in VP/SIMD, GPU)
- **TLP**: Task/Thread (exploit TLP in MCP)
- **PLP**: Process/Program (exploit PLP in MCP, SMP)

Lecture 1: "Introduction to Computer Architecture"

C. "Economics" of Computer Architecture

a. Amdahl's Law and Gustafson's Law
b. Moore's Law and Bell's Law

Electrical & Computer ENGINEERING

"Economics" of Computer Architecture

- Exercise in engineering tradeoff analysis
  - Find the fastest/cheapest/power-efficient/etc. solution
  - Optimization problem with 10s to 100s of variables
- All the variables are changing
  - At non-uniform rates
  - With inflection points
  - Only one guarantee: Today's right answer will be wrong tomorrow

➢ Two Persistent high-level "forcing functions":
  - ➢ **Application Demand** (PROGRAM)
  - ➢ **Technology Supply** (MACHINE)

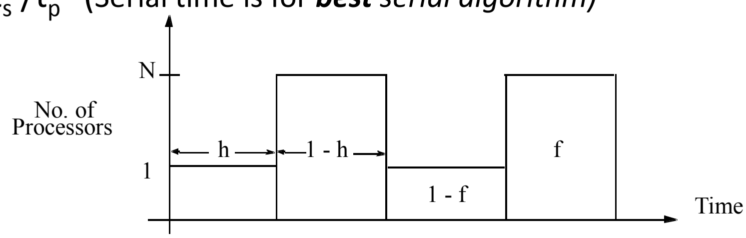
Four Foundational "Laws" of Computer Architecture

- **Application Demand** (PROGRAM)
  - **Amdahl's Law** (1967)
    - Speedup through parallelism is limited by the sequential bottleneck
  - **Gustafson's Law** (1988)
    - With unlimited data set size, parallelism speedup can be unlimited

- **Technology Supply** (MACHINE)
  - **Moore's Law** (1965)
    - (Transistors/Die) increases by 2x every 18 months
  - **Bell's Law** (1971)
    - (Cost/Computer) decreases by 2x every 36 months

Amdahl's Law

- **Speedup** = (Execution time on Single CPU)/(Execution on N parallel processors)
  - t_s / t_p (Serial time is for ***best*** *serial algorithm)*

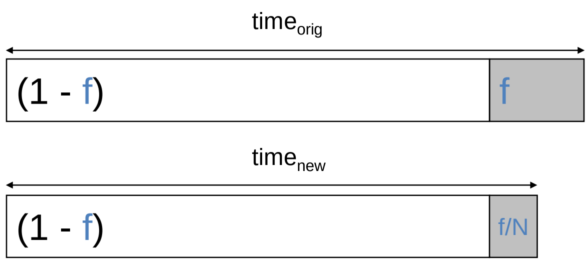


- h = fraction of time in serial code
- f = fraction that is vectorizable or parallelizable
- N = max speedup for f
- Overall speedup → →

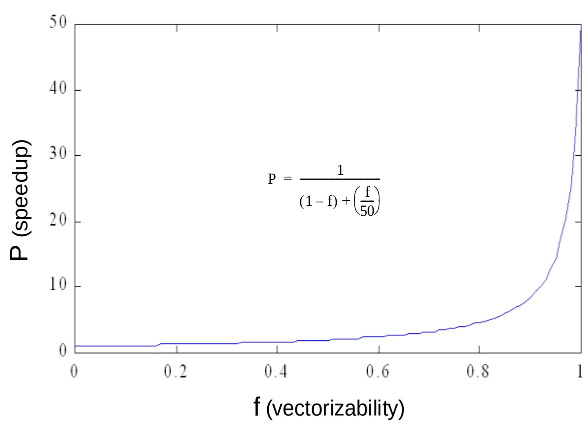
$$Speedup = \frac{1}{(1-f)+\frac{f}{N}}$$

Amdahl's Law Illustrated

- Speedup = $time_{without\ enhancement}$ / $time_{with\ enhancement}$
- If an enhancement speeds up a fraction f of a task by a factor of N
- $time_{new} = time_{orig} \cdot ((1-f) + f/N)$
- $S_{overall} = 1 / ((1-f) + f/N)$



"Tyranny of Amdahl's Law" [Bob Colwell, CMU-Intel-DARPA]



- Suppose that a computation has a 4% serial portion, what is the limit of speedup on 16 processors?
  - 1/((0.04) + (0.96/16)) = 10
  - What is the maximum speedup?
    - 1/0.04 = 25 (with N → ∞)

From Amdahl's Law to Gustafson's Law

- Amdahl's Law works on a *fixed* problem size
  - This is reasonable if your only goal is to solve a problem faster.
  - What if you also want to solve a larger problem?
    - Gustafson's Law (Scaled Speedup)
- Gustafson's Law is derived by fixing the parallel execution time (Amdahl fixed the problem size -> fixed serial execution time)
  - For many practical situations, Gustafson's law makes more sense
    - Have a bigger computer, solve a bigger problem.
- "Amdahl's Law turns out to be too pessimistic for high-performance computing."

Gustafson's Law

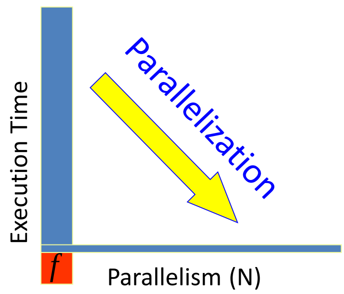
- Fix execution of the computation <u>on a single processor</u> as
  - s + p = serial part + parallelizable part **= 1**
- Speedup(N) = (s + p)/(s + p/N)
  = 1/(s + (1 – s)/N) = 1/((1-p) + p/N) ← Amdahl's law
- Now let 1 = (a + b) = execution time of computation <u>on N processors</u> (fixed) where a = sequential time and b = parallel time on any of the N processors
  - Time for sequential processing = a + (b×N) and Speedup = (a + b×N)/(a + b)
  - Let α = a/(a+b) be the sequential fraction of the parallel execution time
  - $\text{Speedup}_{scaled}(N) = (a + b\times N)/(a + b) = (a/(a+b) + (b\times N)/(a+b)) = \alpha + (1- \alpha)N$
  - If α is very small, the scaled speedup is approximately N, i.e. linear speedup.

The Two Laws on Algorithm and Performance

Amdahl's Law

$$Speedup(N)_{MC} = \frac{1}{\left(\frac{f}{1}\right) + \left(\frac{(1-f)}{N}\right)}$$

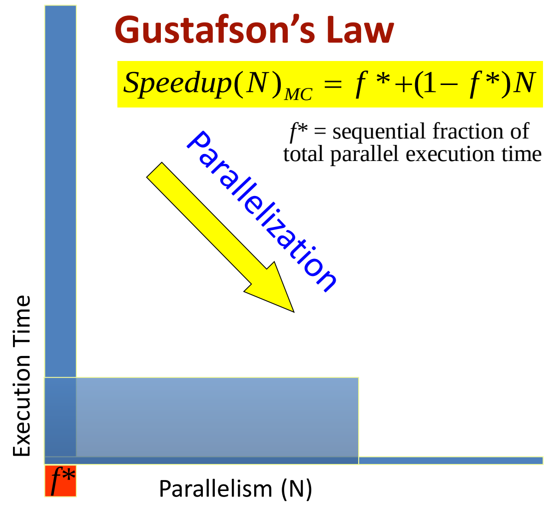
f = sequential %



Gustafson's Law

$$Speedup(N)_{MC} = f^* + (1 - f^*)N$$

f^* = sequential fraction of total parallel execution time



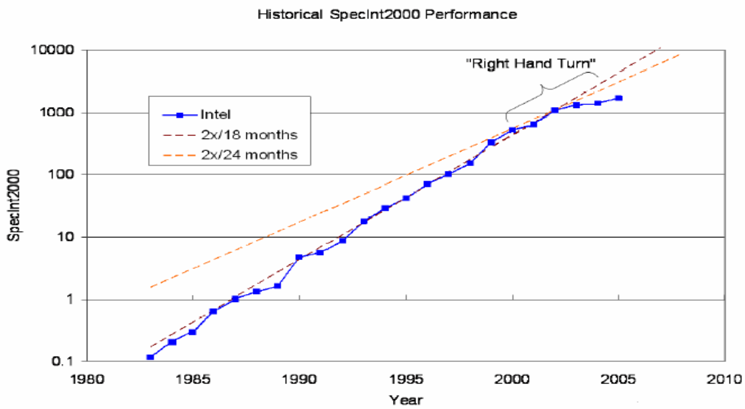
The Two "Gordon" Laws of Computer Architecture

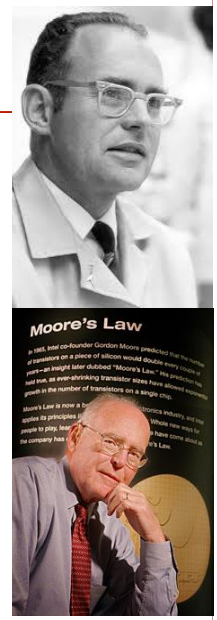
- **Gordon Moore's Law** (1965)
  - (Transistors/Die) increases by 2X every 18 months
  - Constant price, increasing performance
  - Has held for 40+ years, and will continue to hold
- **Gordon Bell's Law** (1971)
  - (Cost/Computer) decreases by 2X every 36 months (~ 10X per decade)
  - Constant performance, decreasing price
  - Corollary of Moore's Law, creation of new computer categories

"In a decade you can buy a computer for less than its sales tax today." – Jim Gray

We have all been living on this exponential curve and assume it…

Moore's Law Trends

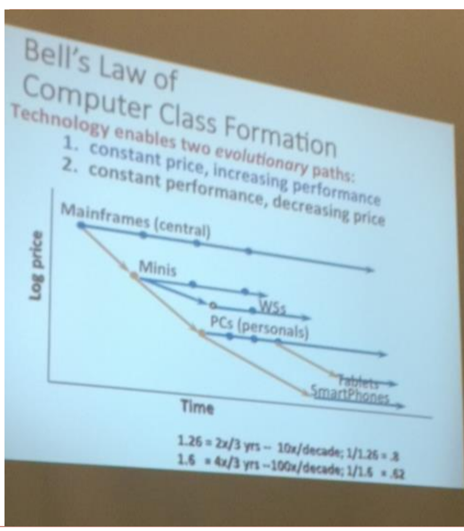


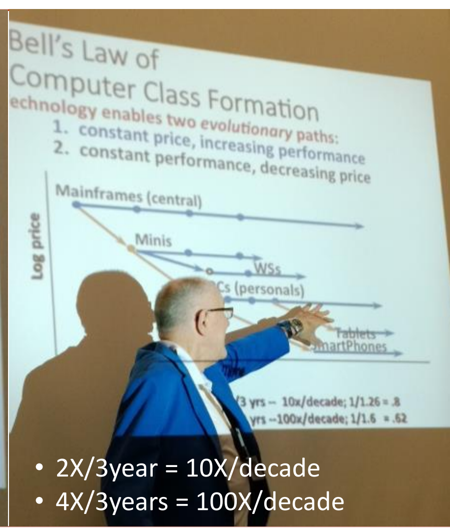


- Moore's Law for device integration
- Chip power consumption
- Single-thread performance trend

[source: Intel]

Bell's Law Trends

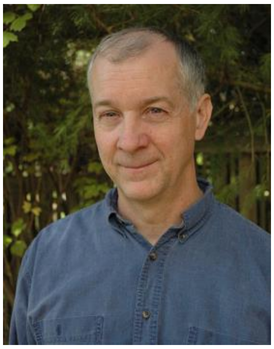


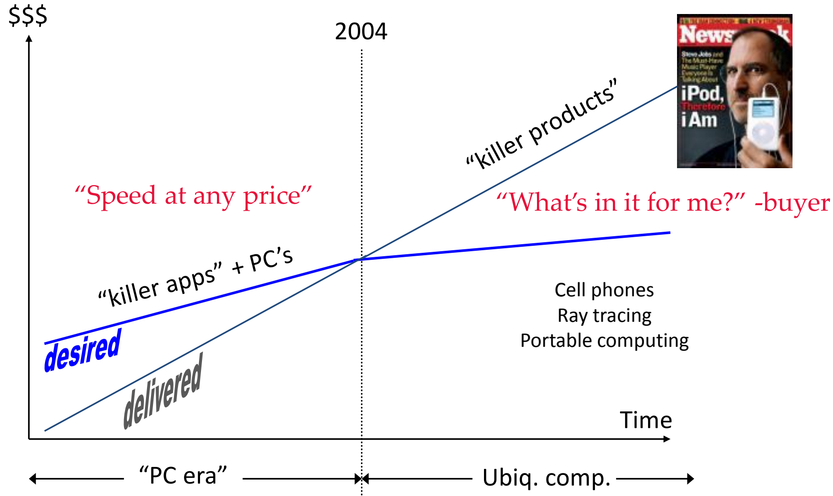


- 2X/3year = 10X/decade
- 4X/3years = 100X/decade

[Bob Colwell CRA Grand Challenges panel 2005]

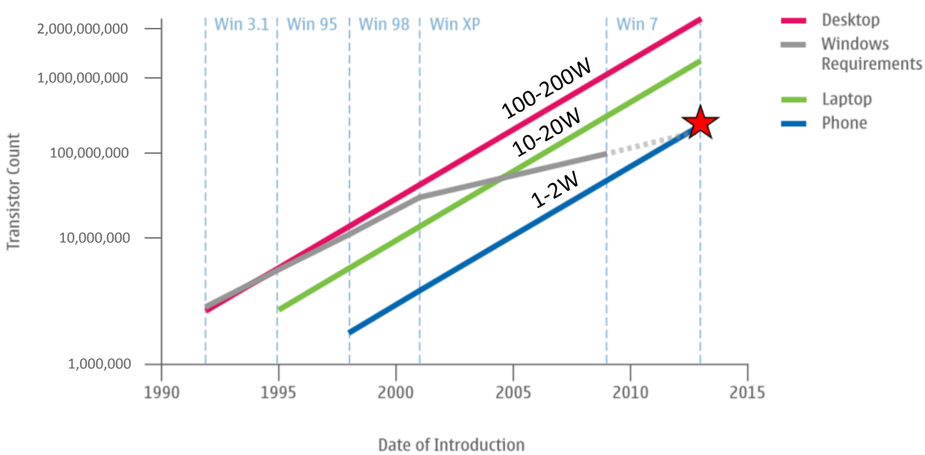
Know Your "Supply & Demand Curves"



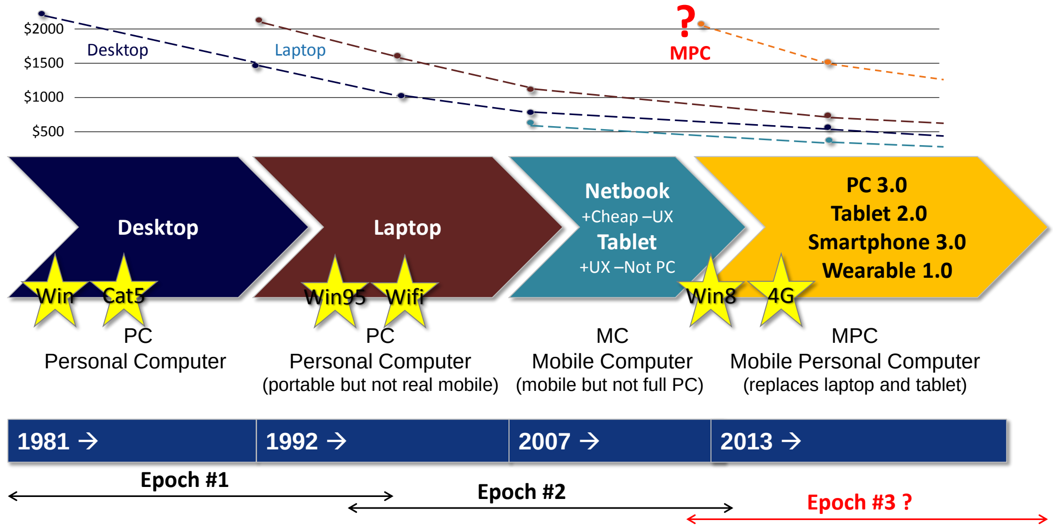


Moore's Law and Bell's Law are Alive and Well

Cross Over Point in 2013 !!!



New Epoch of "Mobile Personal Computers" (MPC)



18-640 Foundations of Computer Architecture

Lecture 2: "Review of Pipelined Processor Design"

John Paul Shen
August 28, 2014

Next Time …

- Required Reading Assignment:
  - **Chapter 2 of Shen and Lipasti (SnL).**
- Recommended Reference:
  - "Optimum Power/Performance Pipeline Depth" by A. Hartstein and Thomas R. Puzak, MICRO 2003.

Electrical & Computer ENGINEERING