Lecture 01: Introduction

CSE 564 Computer Architecture
Summer 2017

Department of Computer Science and Engineering

Yonghong Yan

yan@oakland.edu

www.secs.oakland.edu/~yan

Copyright and Acknowledgement

- Most slides were adapted from lectures notes of the two textbooks with copyright of publisher or the original authors including Elsevier Inc, Morgan Kaufmann, David A. Patterson and John L. Hennessy.
- Some slides were adapted from the following courses:
  - UC Berkeley course "Computer Science 252: Graduate Computer Architecture" of David E. Culler Copyright 2005 UCB
    - http://people.eecs.berkeley.edu/~culler/courses/cs252-s05/
  - Great Ideas in Computer Architecture (Machine Structures) by Randy Katz and Bernhard Boser
    - http://inst.eecs.berkeley.edu/~cs61c/fa16/
- I also refer to the following courses and lecture notes when preparing materials for this course
  - Computer Science 152: Computer Architecture and Engineering, Spring 2016 by Dr. George Michelogiannakis from UC Berkeley
    - http://www-inst.eecs.berkeley.edu/~cs152/sp16/
  - Computer Science 252: Graduate Computer Architecture, Fall 2015 by Prof. Krste Asanović from UC Berkeley
    - http://www-inst.eecs.berkeley.edu/~cs252/fa15/
  - Computer Science S 250: VLSI Systems Design, Spring 2016 by Prof. John Wawrzynek from UC Berkeley
    - http://www-inst.eecs.berkeley.edu/~cs250/sp16/
  - Computer System Architecture, Fall 2005 by Dr. Joel Emer and Prof. Arvind from MIT
    - http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-823-computer-system-architecture-fall-2005/
  - Synthesis Lectures on Computer Architecture
    - http://www.morganclaypool.com/toc/cac/1/1

-

Contents

- **Computers and computer components**
- Computer architectures and great ideas in history and now
- Performance

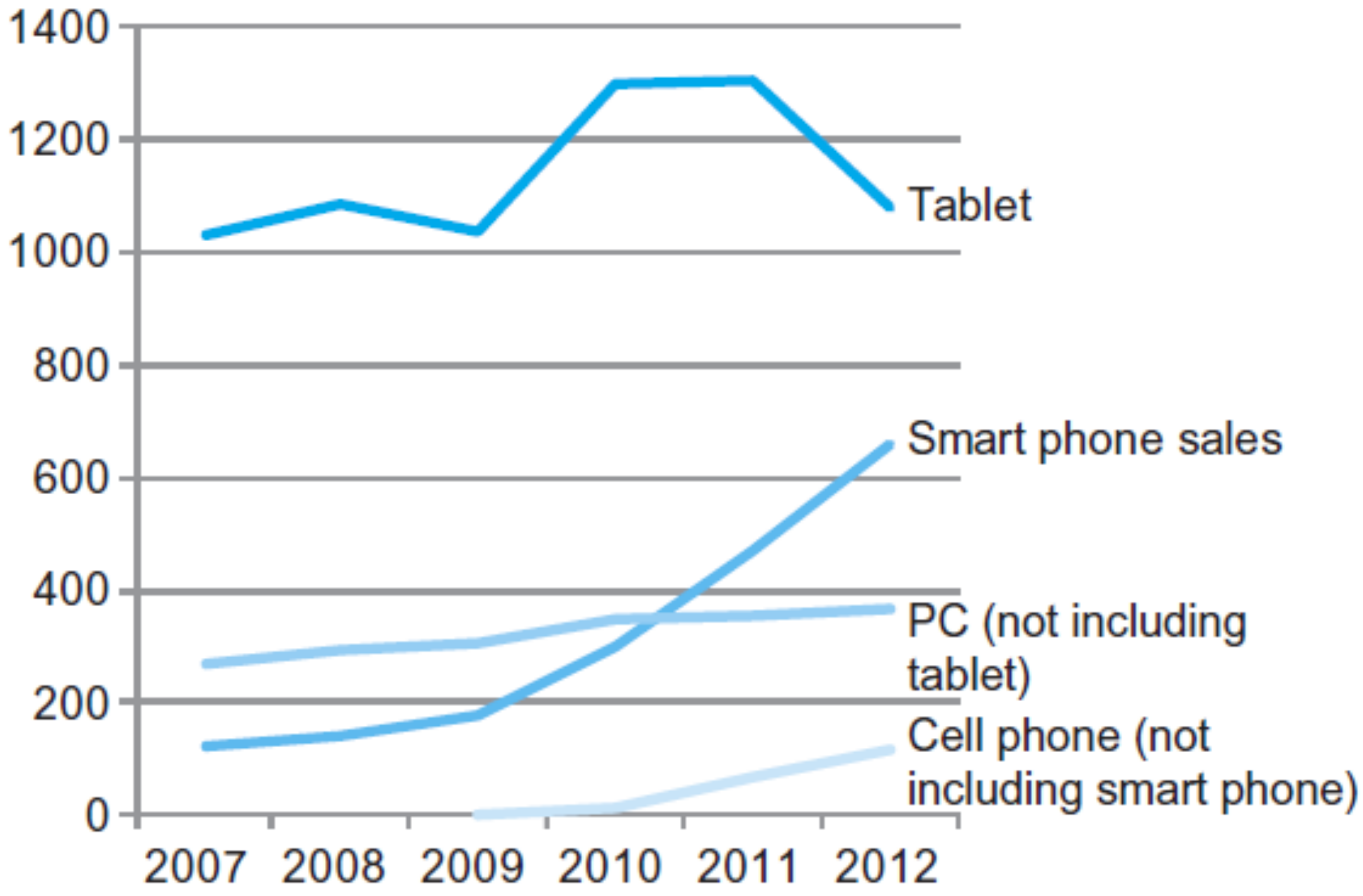
The Computer Revolution

- Progress in computer technology
  - Underpinned by Moore’s Law
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive

Classes of Computers

- Personal Mobile Device (PMD)
  - e.g. smartphones, tablet computers
  - Emphasis on energy efficiency and real-time
- Desktop Computing
  - Emphasis on price-performance
- Servers
  - Emphasis on availability, scalability, throughput
- Clusters / Warehouse Scale Computers
  - Used for "Software as a Service (SaaS)"
  - Emphasis on availability and price-performance
  - Sub-class: Supercomputers, emphasis: floating-point performance and fast internal networks
- Embedded Computers
  - Emphasis: price

The PostPC Era

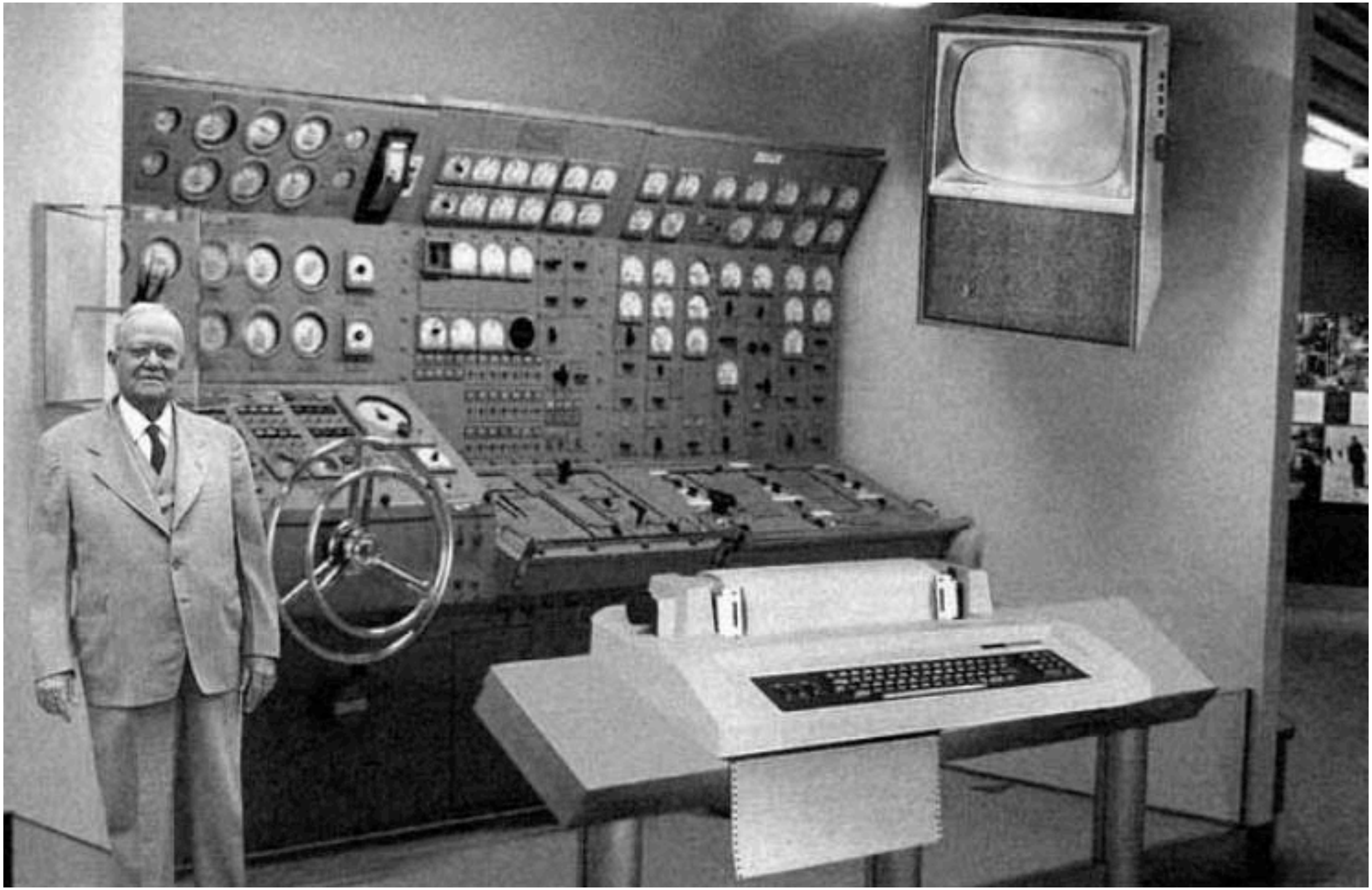


The PostPC Era

- Personal Mobile Device (PMD)
  - Battery operated
  - Connects to the Internet
  - Hundreds of dollars
  - Smart phones, tablets, electronic glasses

- Cloud computing
  - Warehouse Scale Computers (WSC)
  - Software as a Service (SaaS)
  - Portion of software run on a PMD and a portion run in the Cloud
  - Amazon and Google

Old School Computer

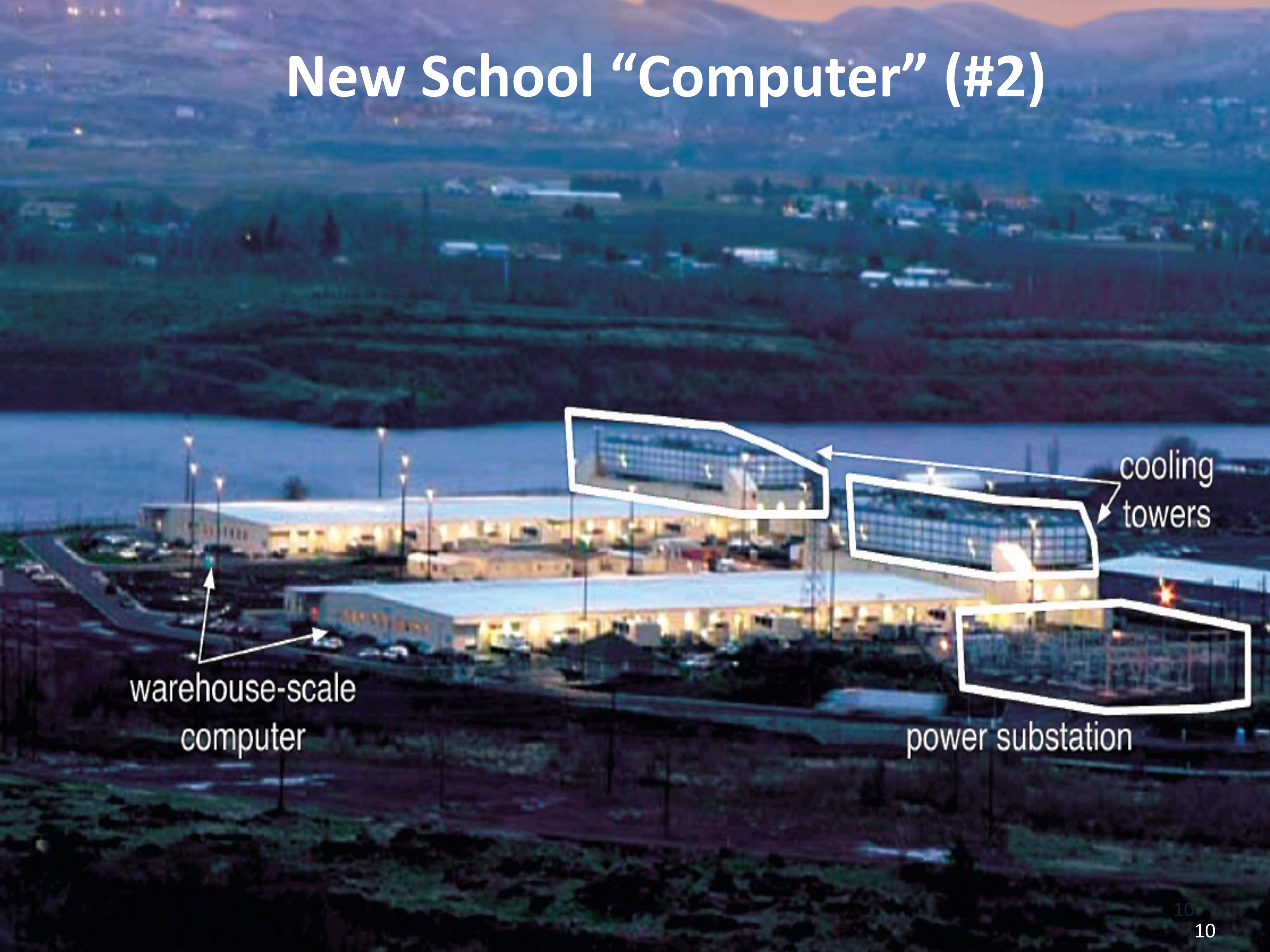


New School Computer (#1)

Personal
Mobile
Devices

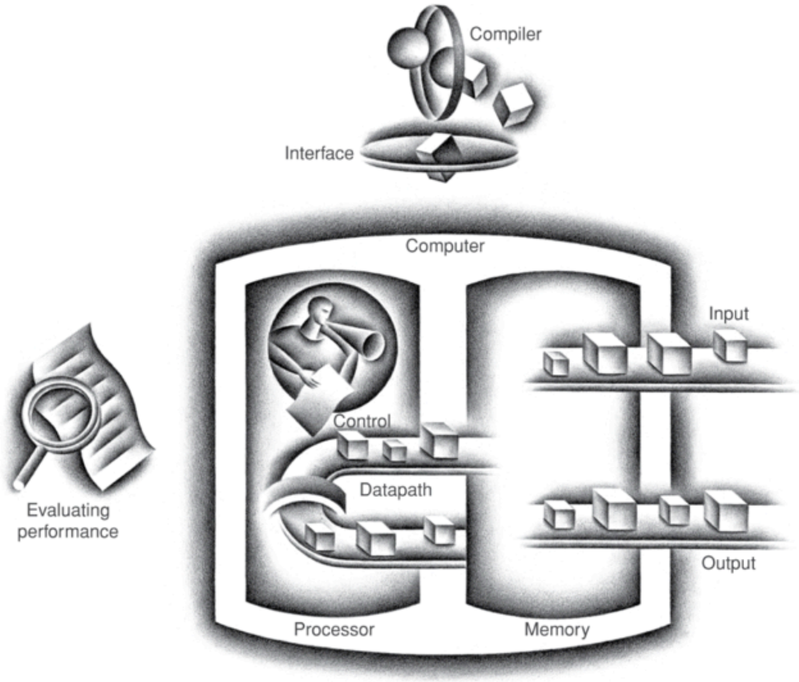
9

New School “Computer” (#2)



Components of a Computer

The BIG Picture

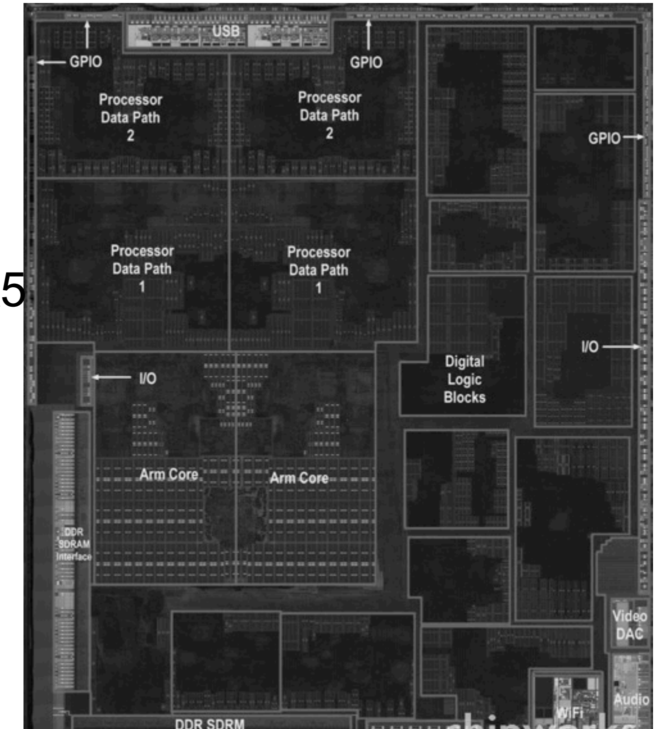


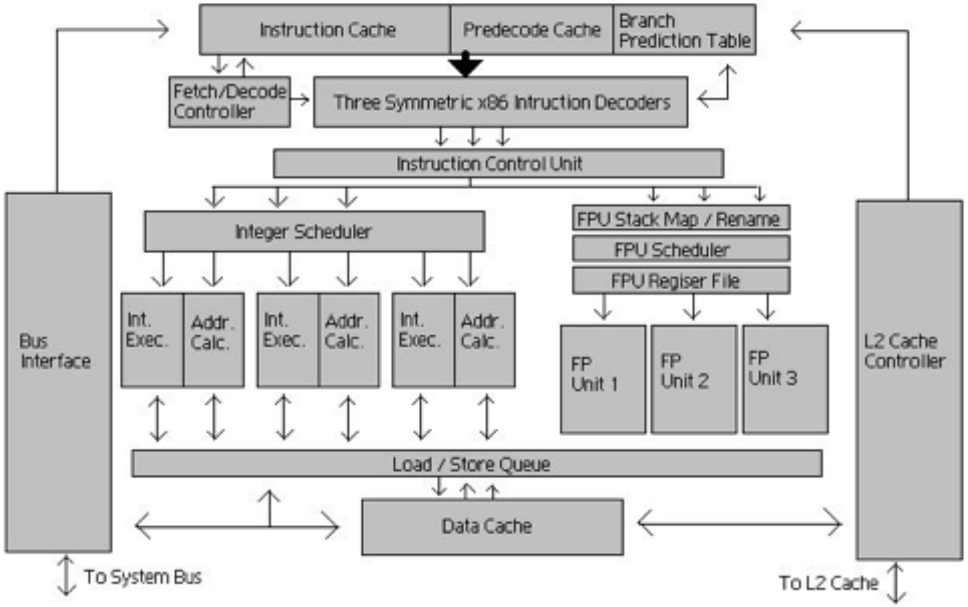
- Same components for all kinds of computer
  - Desktop, server, embedded
- Input/output includes
  - User-interface devices
    - Display, keyboard, mouse
  - Storage devices
    - Hard disk, CD/DVD, flash
  - Network adapters
    - For communicating with other computers

Inside the Processor (CPU)

- Functional units: performs computations
- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data

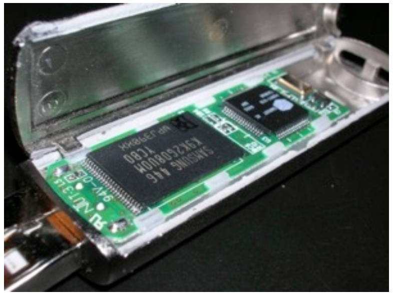
Apple A5





A Safe Place for Data

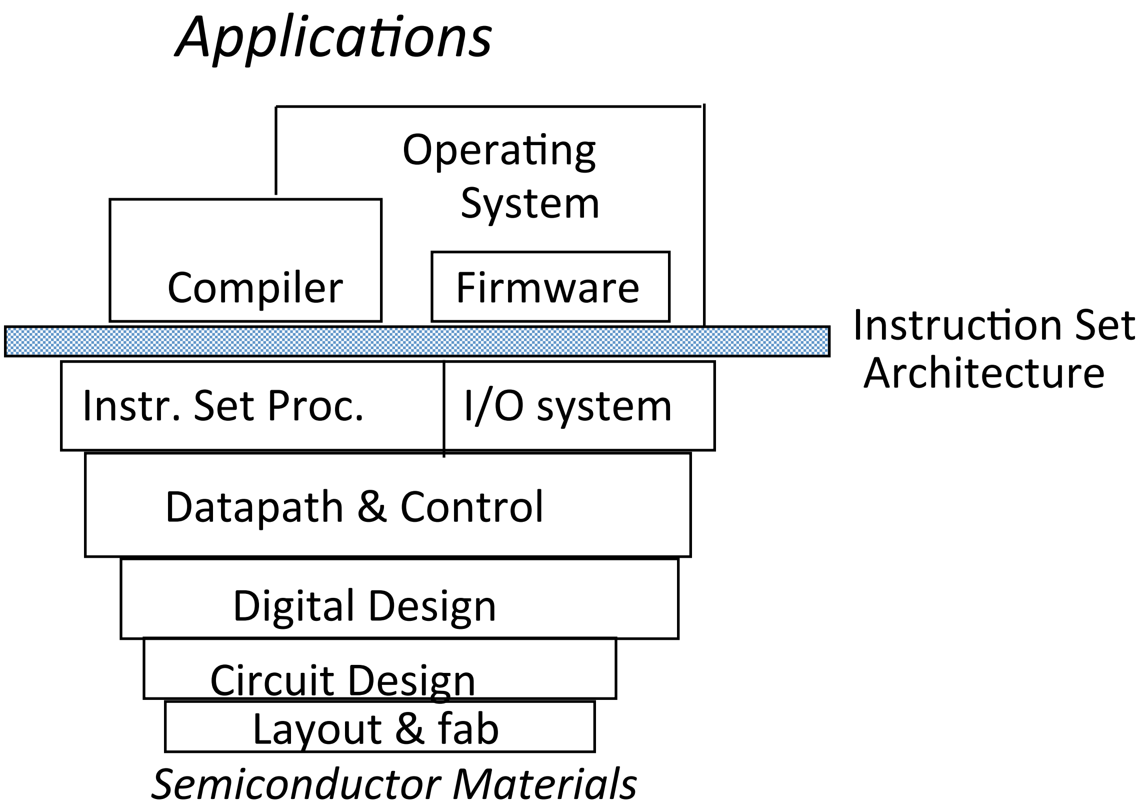
- Volatile main memory
  - Loses instructions and data when power off
- Non-volatile secondary memory
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)



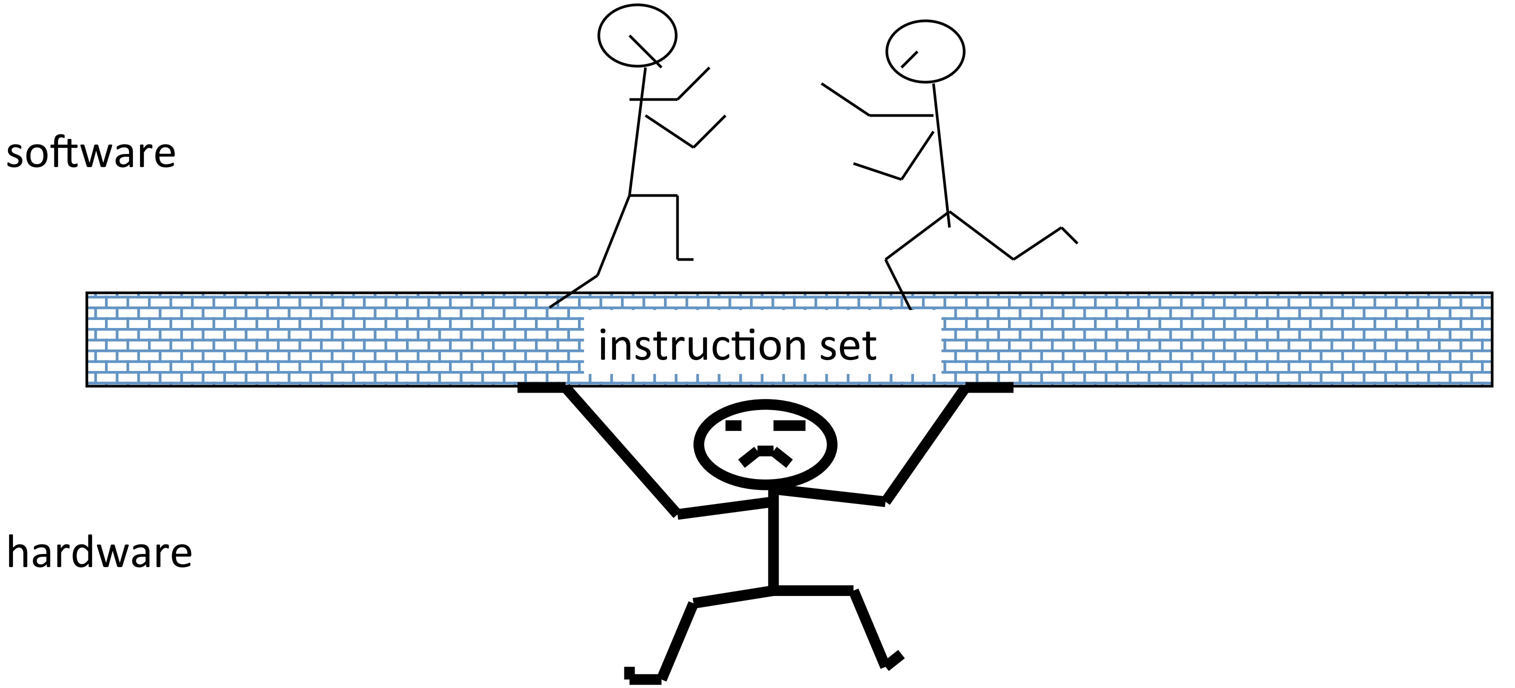
Contents

- Computers and computer components
- **Computer architectures and great ideas in history and now**
- Performance

What is “Computer Architecture”?



The Instruction Set: a Critical Interface



- Properties of a good abstraction
  - Lasts through many generations (portability)
  - Used in many different ways (generality)
  - Provides convenient functionality to higher levels
  - Permits an efficient implementation at lower levels

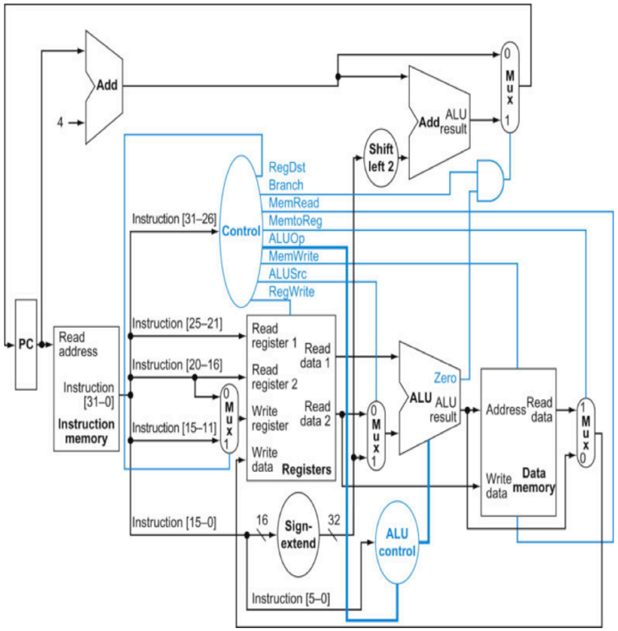
Elements of an ISA

- Set of machine-recognized data types
  - bytes, words, integers, floating point, strings, . . .
- Operations performed on those data types
  - Add, sub, mul, div, xor, move, ….
- Programmable storage
  - regs, PC, memory
- Methods of identifying and obtaining data referenced by instructions (addressing modes)
  - Literal, reg., absolute, relative, reg + offset, …
- Format (encoding) of the instructions
  - Op code, operand fields, …

Computer Architecture

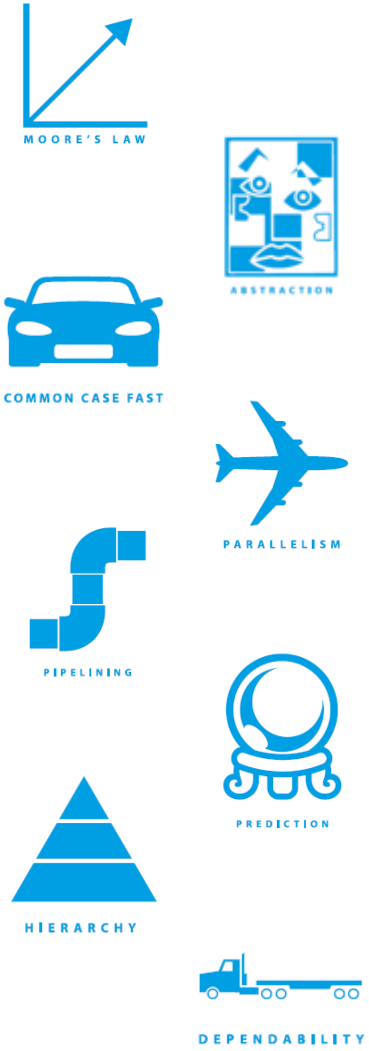
How things are put together in design and implementation

- Capabilities & Performance Characteristics of Principal Functional Units
  - (e.g., Registers, ALU, Shifters, Logic Units, ...)
- Ways in which these components are interconnected
- Information flows between components
- Logic and means by which such information flow is controlled.
- Choreography of FUs to realize the ISA



Great Ideas in Computer Architectures

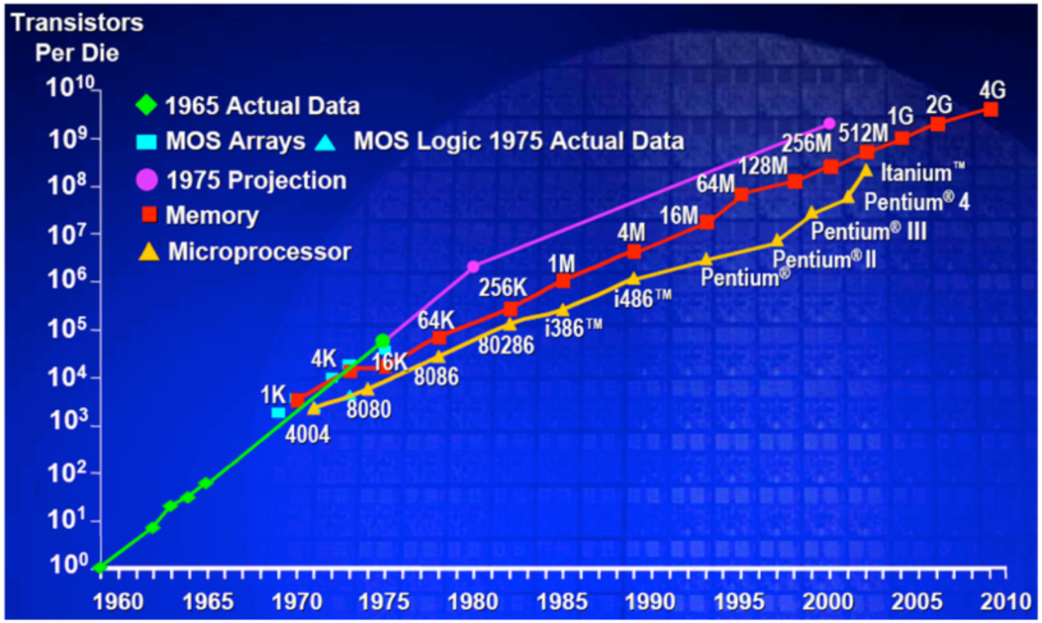
1. Design for ***Moore’s Law***
2. Use ***abstraction*** to simplify design
3. Make the ***common case fast***
4. Performance *via* ***parallelism***
5. Performance *via* ***pipelining***
6. Performance *via* ***prediction***
7. ***Hierarchy*** of memories
8. ***Dependability*** *via* redundancy



Great Idea: "Moore's Law"

Gordon Moore, Founder of Intel

- 1965: since the integrated circuit was invented, the number of transistors/inch^2 in these circuits roughly doubled every year; this trend would continue for the foreseeable future
- 1975: revised - circuit complexity doubles every two years



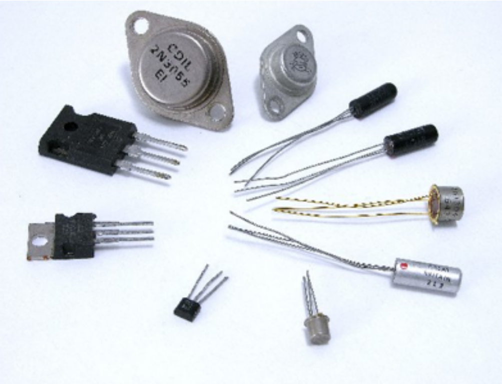
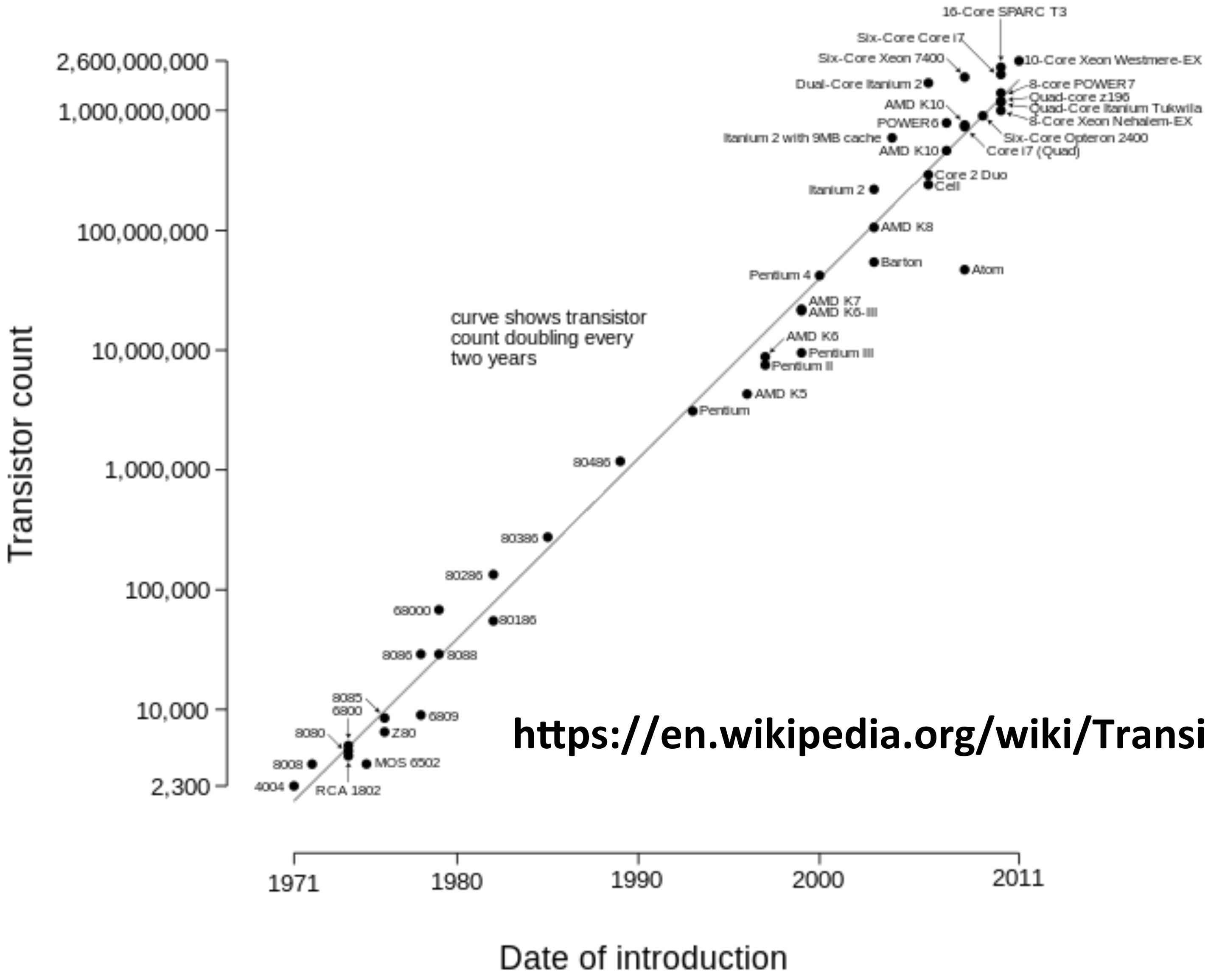


Image credit: Intel

Microprocessor Transistor Counts 1971-2011 & Moore's Law



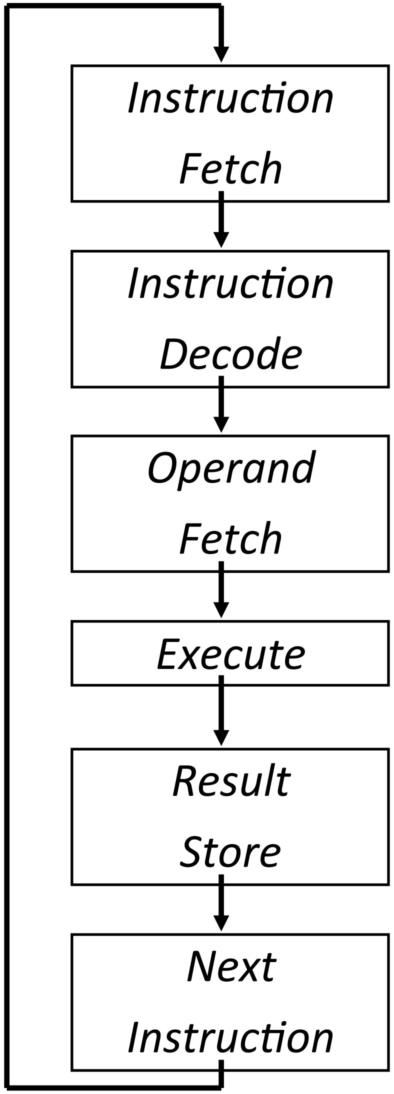
https://en.wikipedia.org/wiki/Transistor_count

Moore's Law trends

- More transistors = ↑ opportunities for exploiting parallelism in the instruction level (ILP)
  - Pipeline, superscalar, VLIW (Very Long Instruction Word), SIMD (Single Instruction Multiple Data) or vector, speculation, branch prediction
- General path of scaling
  - Wider instruction issue, longer piepline
  - More speculation
  - More and larger registers and cache
- **Increasing circuit density ~= increasing frequency ~= increasing performance**
- Transparent to users
  - An easy job of getting better performance: buying faster processors (higher frequency)
- **We have enjoyed this free lunch for several decades, however (TBD) …**

Great Idea: Pipeline

Fundamental Execution Cycle



Instruction Fetch

Obtain instruction from program storage

Instruction Decode

Determine required actions and instruction size

Operand Fetch

Locate and obtain operand data

Execute

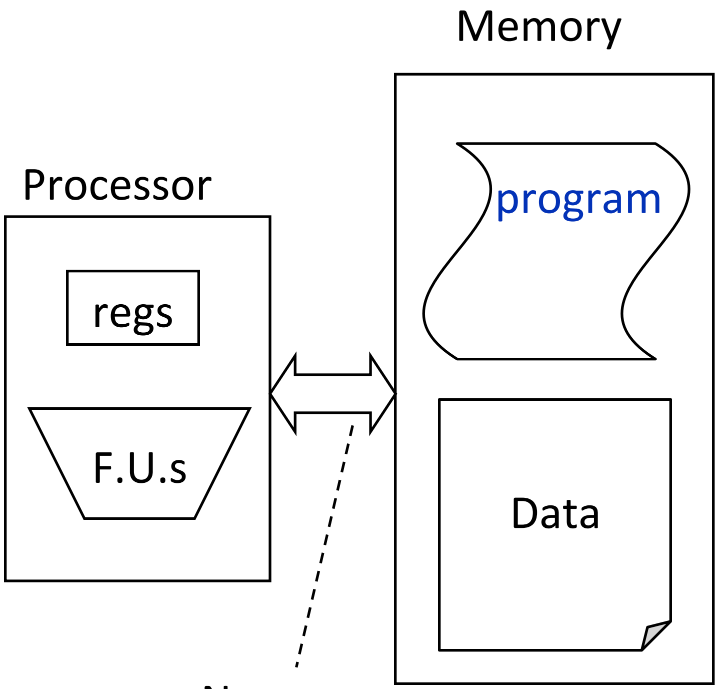
Compute result value or status

Result Store

Deposit results in storage for later use

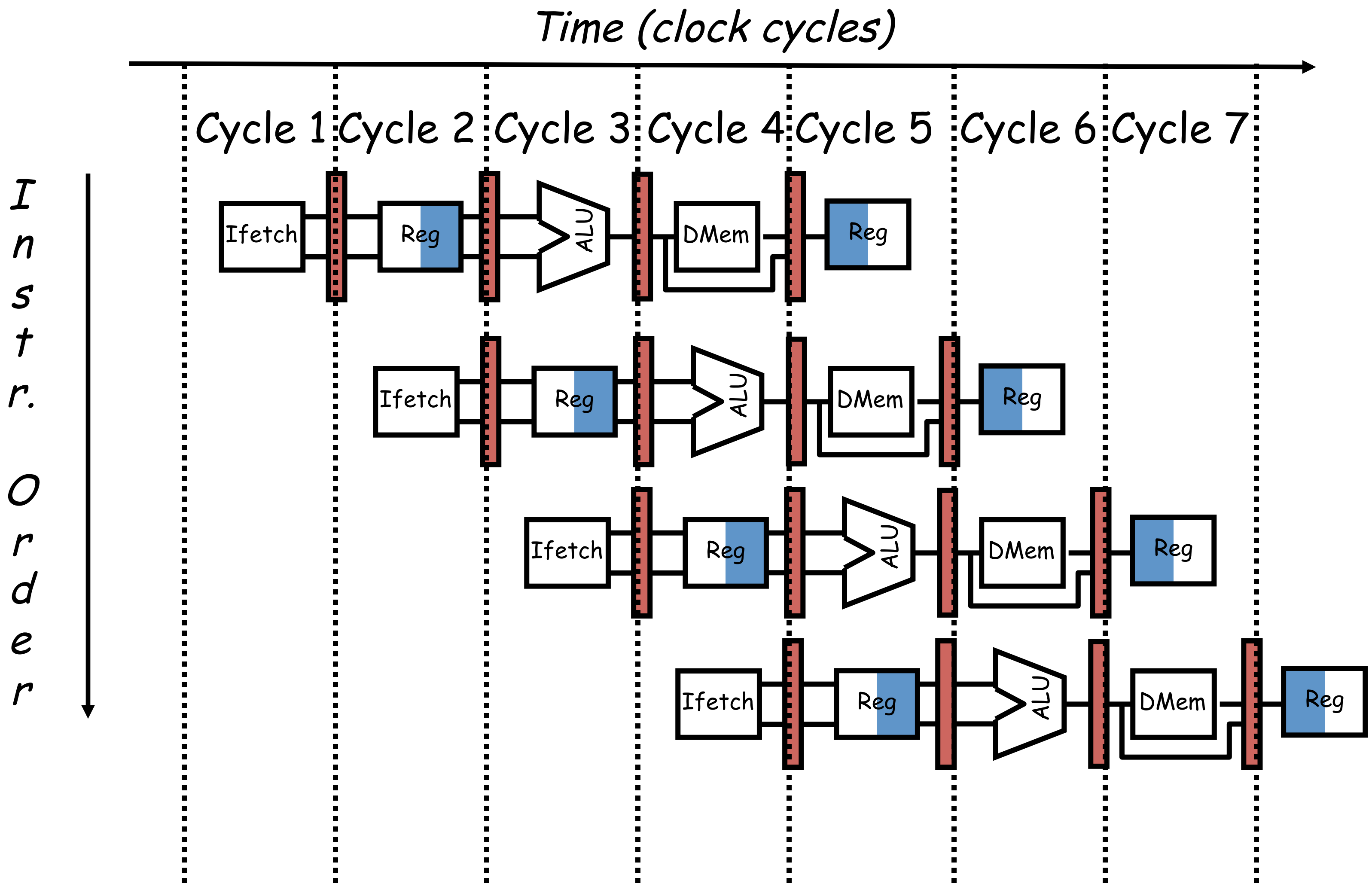
Next Instruction

Determine successor instruction

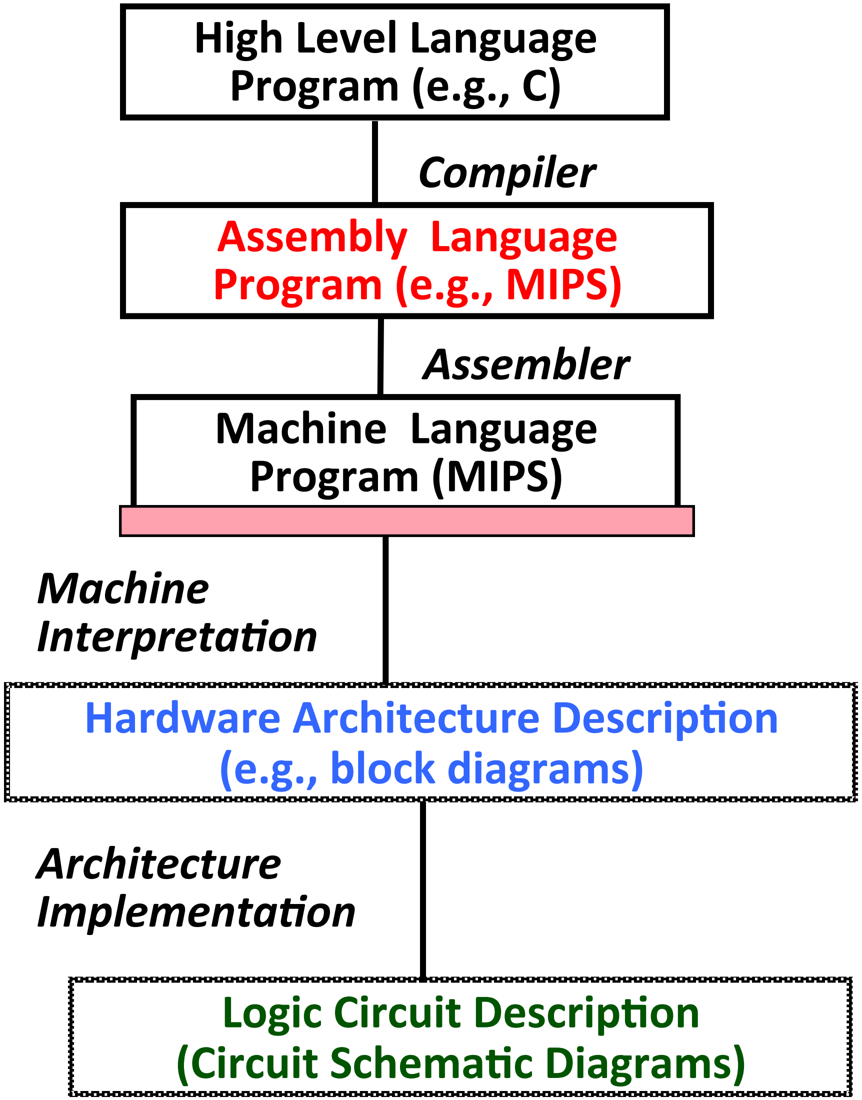


von Neuman bottleneck

Pipelined Instruction Execution



Great Idea: Abstraction
(Levels of Representation/Interpretation)



High Level Language Program (e.g., C)

Compiler

Assembly Language Program (e.g., MIPS)

Assembler

Machine Language Program (MIPS)

Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)

Architecture Implementation

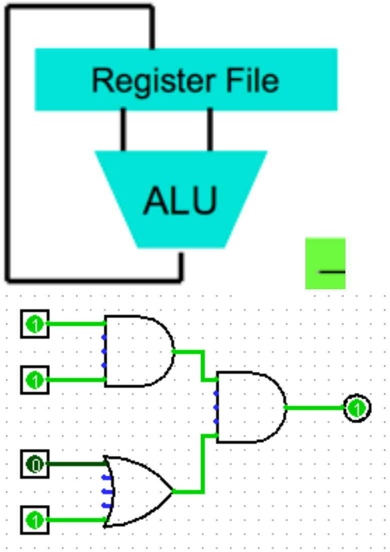
Logic Circuit Description (Circuit Schematic Diagrams)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw    $t0, 0($2)
lw    $t1, 4($2)
sw    $t1, 0($2)
sw    $t0, 4($2)
```

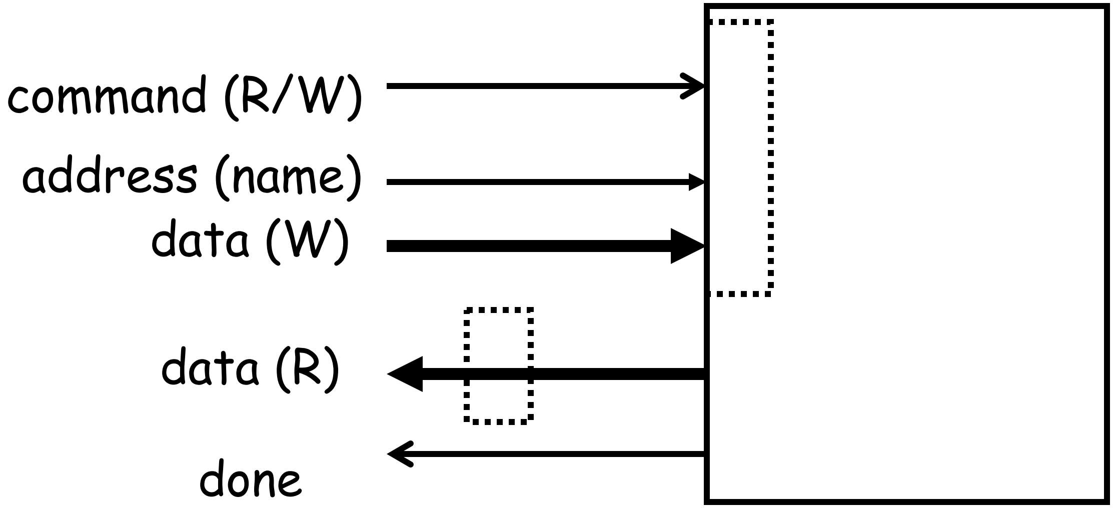
Anything can be represented as a *number*, i.e., data or instructions

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

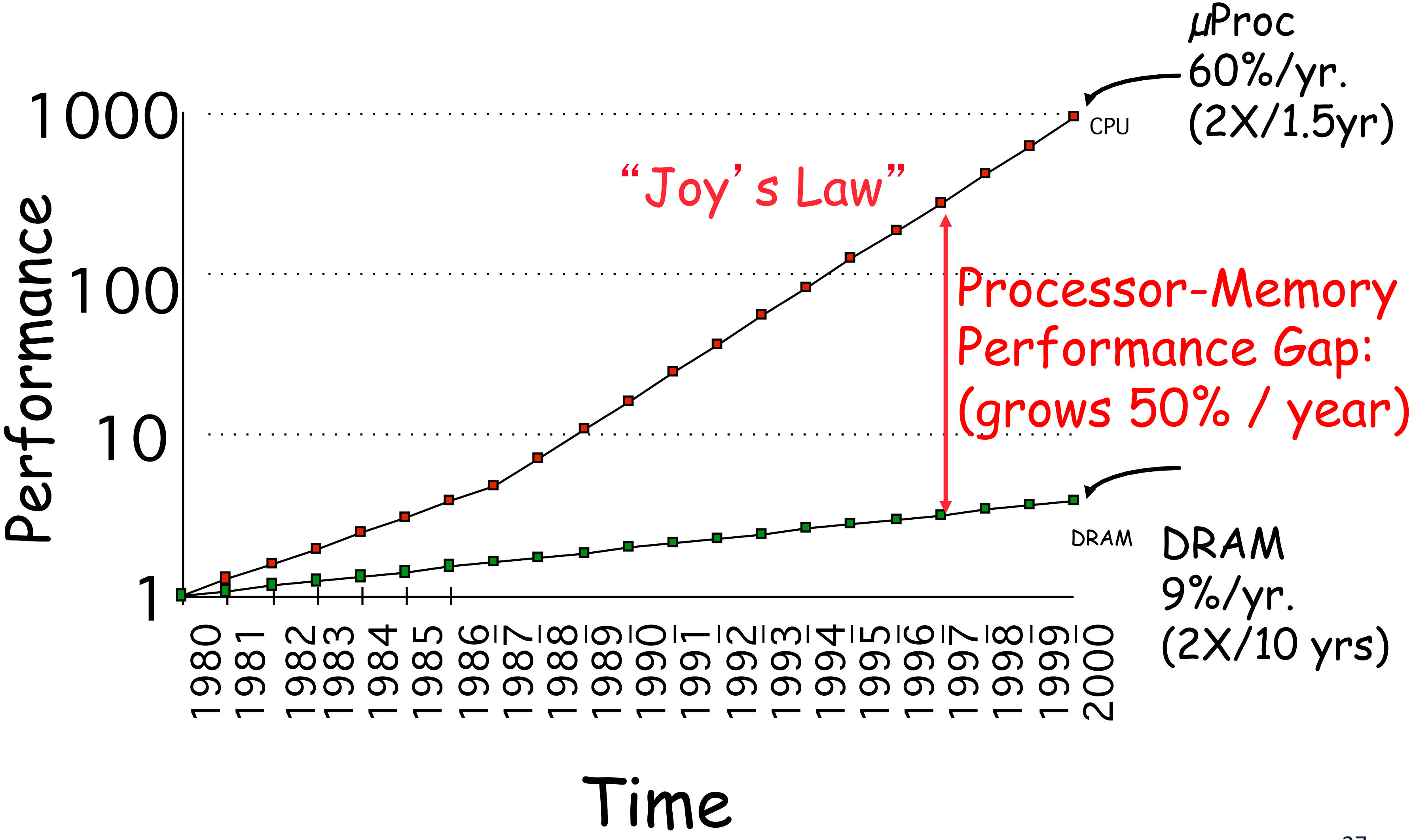


The Memory Abstraction

- Association of <name, value> pairs
  - typically named as byte addresses
  - often values aligned on multiples of size
- Sequence of Reads and Writes
- Write binds a value to an address
- Read of addr returns most recently written value bound to that address

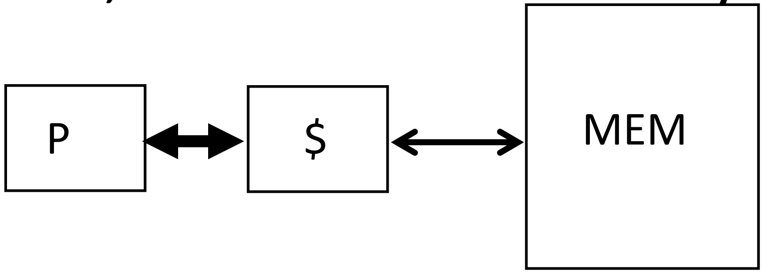


Processor-DRAM Memory Gap (latency)



The Principle of Locality

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon
    (e.g., straightline code, array access)
- Last 30 years, HW relied on locality for speed



Great idea: Memory Hierarchy

Levels of the Memory Hierarchy

Capacity
Access Time
Cost

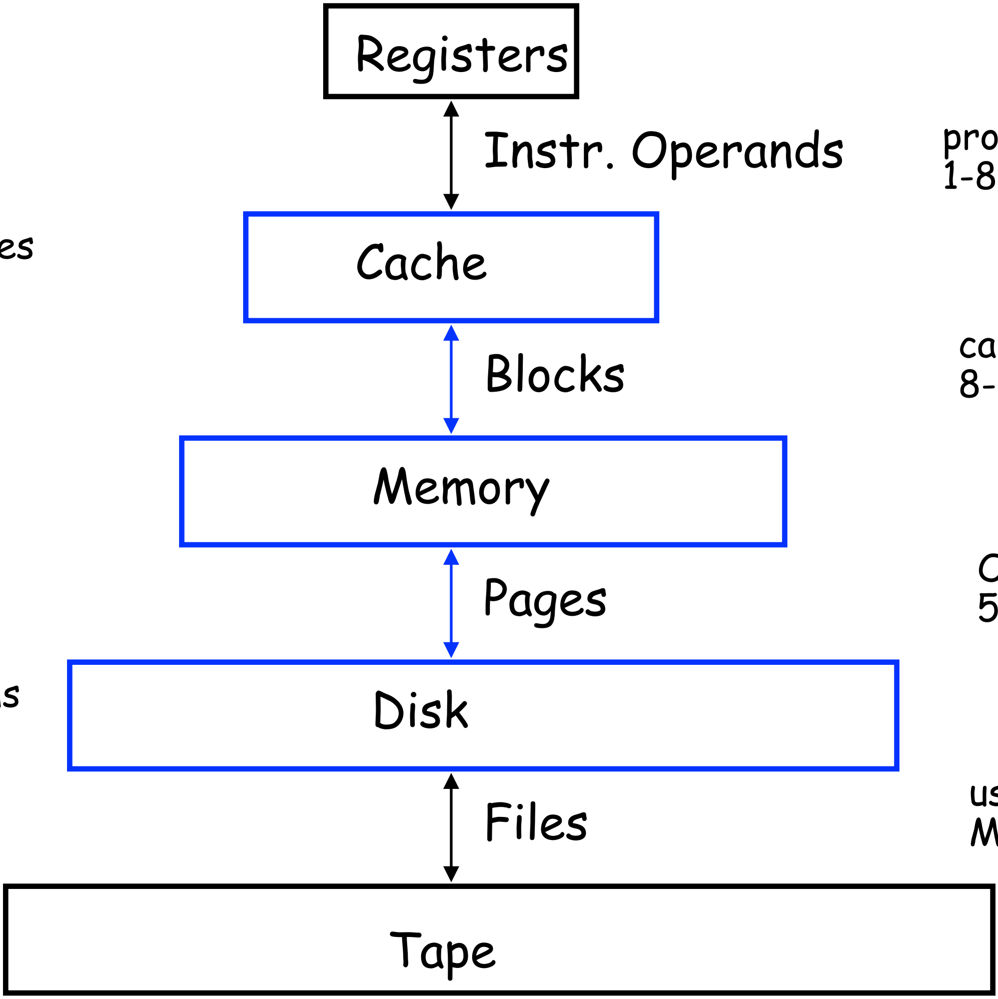
CPU Registers
100s Bytes
<< 1s ns

Cache
10s-100s K Bytes
~1 ns
$1s/ MByte

Main Memory
M Bytes
100ns- 300ns
$< 1/ MByte

Disk
10s G Bytes, 10 ms
(10,000,000 ns)
$0.001/ MByte

Tape
infinite
sec-min
$0.0014/ MByte



Staging
Xfer Unit

prog./compiler
1-8 bytes

cache cntl
8-128 bytes

OS
512-4K bytes

user/operator
Mbytes

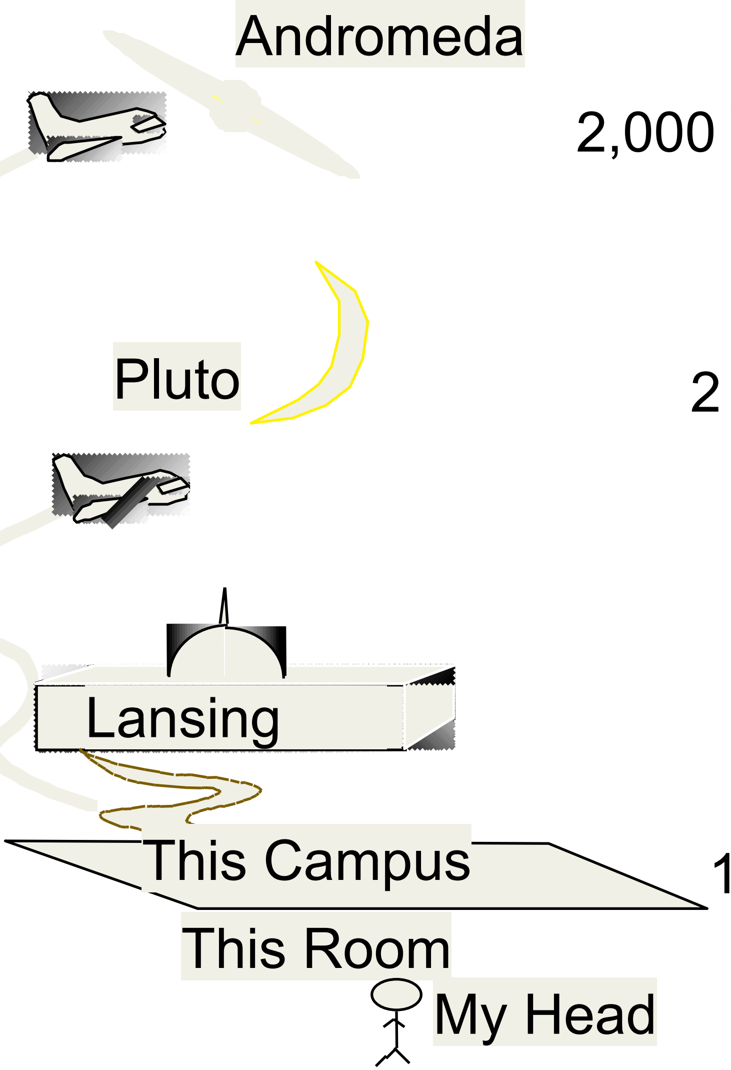
Upper Level

faster

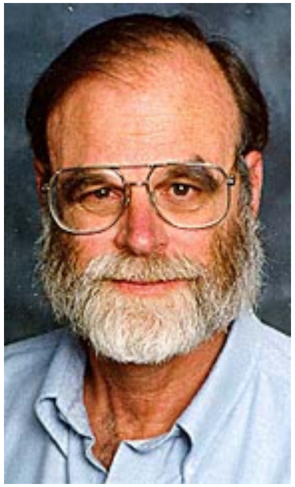
Larger

Lower Level

Jim Gray's Storage Latency Analogy: How Far Away is the Data?



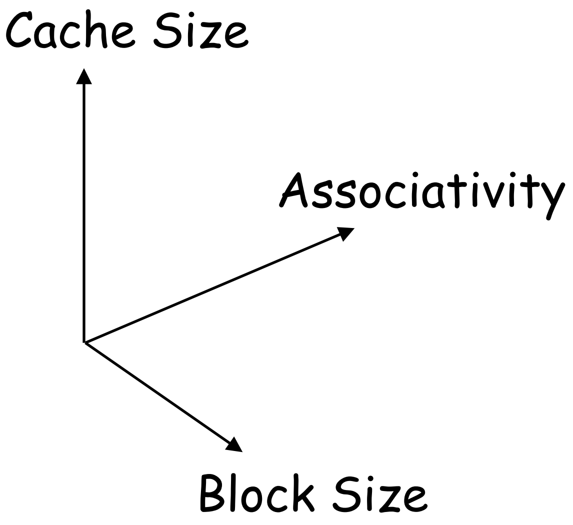
10^9	Tape /Optical Robot	Andromeda	2,000 Years
10^6	Disk	Pluto	2 Years
100	Main Memory	Lansing	1.5 hr
10	On Board Cache	This Campus	10 min
2	On Chip Cache	This Room	
1	Registers	My Head	1 min
(ns)			



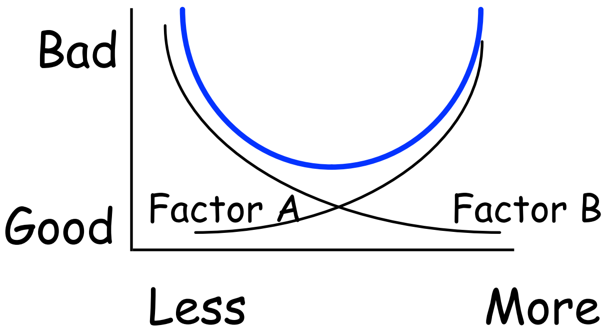
Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969!

The Cache Design Space

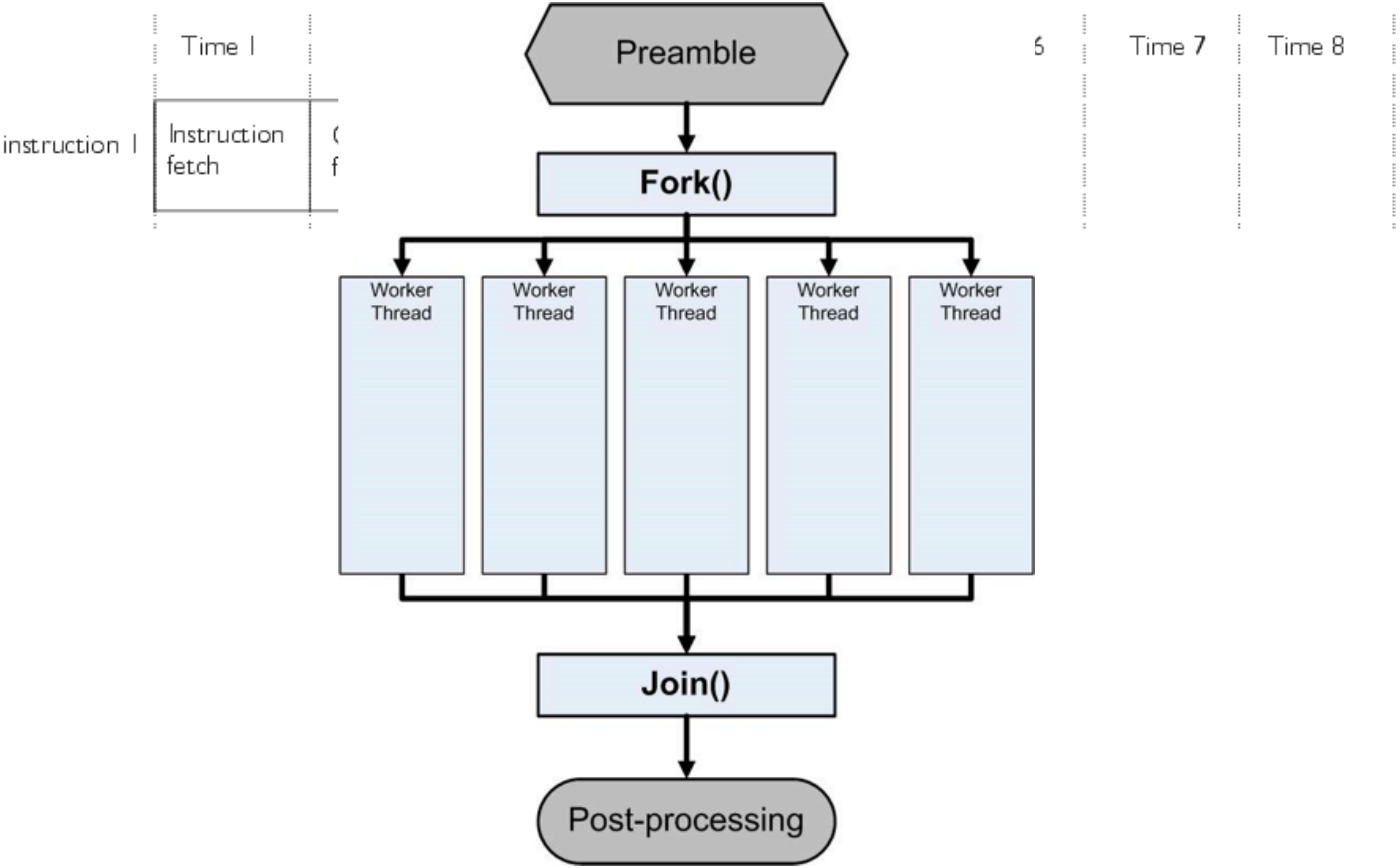
- Several interacting dimensions
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back



- The optimal choice is a compromise
  - depends on access characteristics
    - workload
    - use (I-cache, D-cache, TLB)
  - depends on technology / cost
- Simplicity often wins



Great Idea: Parallelism

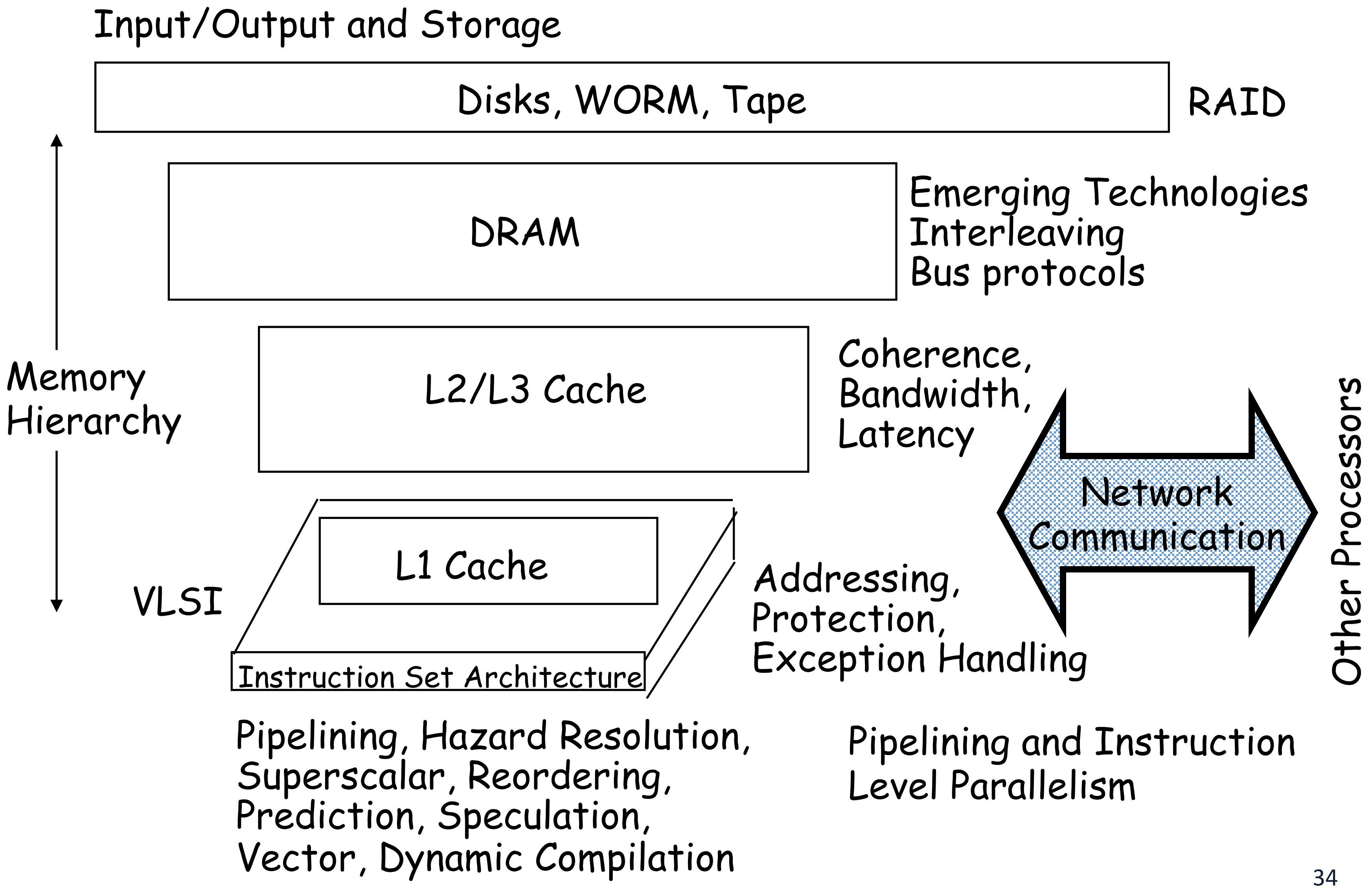


Defining Computer Architecture

- “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:
  - Specific requirements of the target machine
  - Design to maximize performance within constraints: cost, power, and availability
  - Includes ISA, microarchitecture, hardware

Computer Architecture Topics



Input/Output and Storage

Disks, WORM, Tape

RAID

DRAM

Emerging Technologies
Interleaving
Bus protocols

L2/L3 Cache

Coherence,
Bandwidth,
Latency

Memory
Hierarchy

L1 Cache

VLSI

Instruction Set Architecture

Addressing,
Protection,
Exception Handling

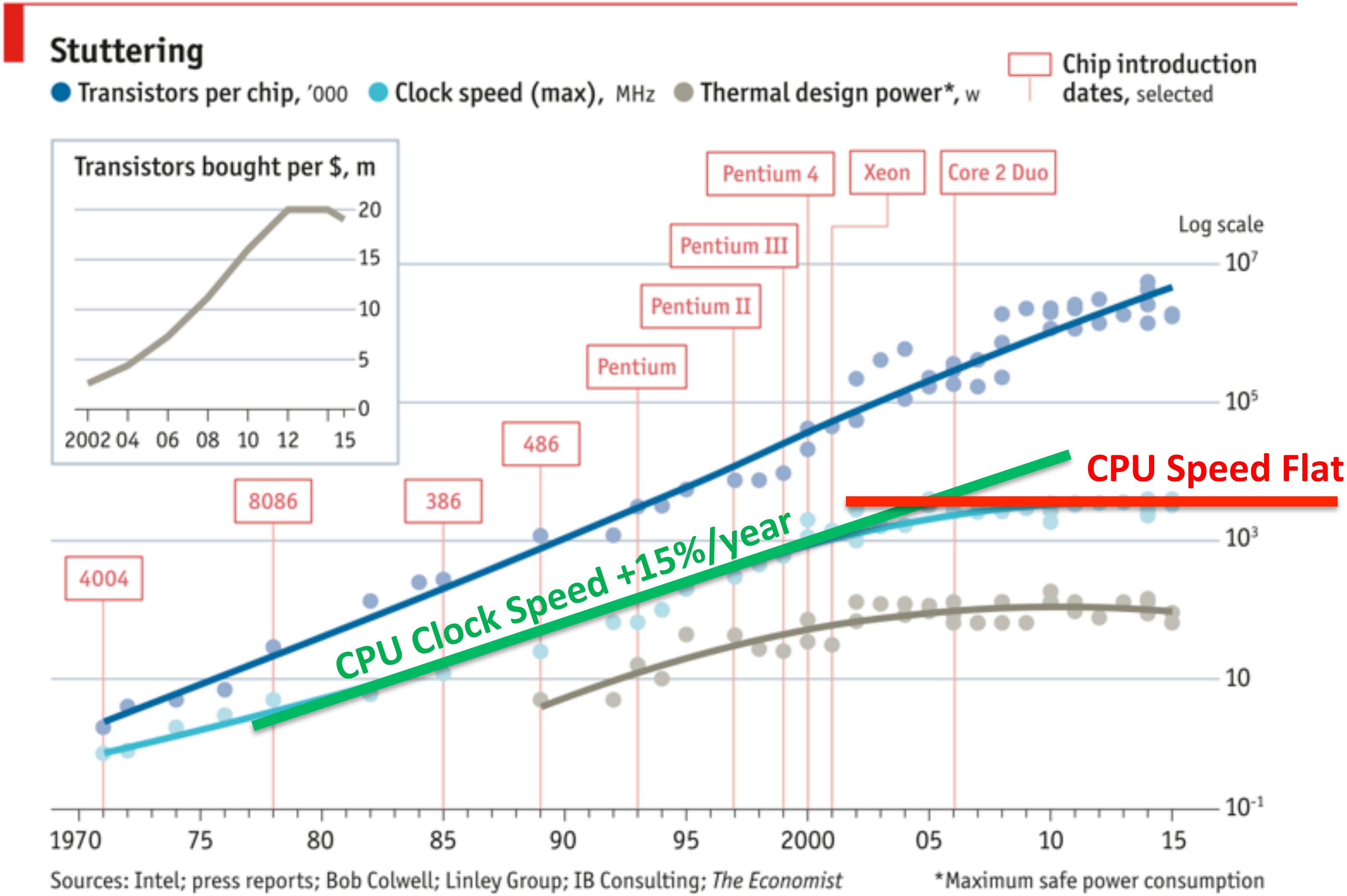
Network
Communication

Other Processors

Pipelining, Hazard Resolution,
Superscalar, Reordering,
Prediction, Speculation,
Vector, Dynamic Compilation

Pipelining and Instruction
Level Parallelism

Why is Architecture Exciting Today?



Sources: Intel; press reports; Bob Colwell; Linley Group; IB Consulting; *The Economist*

*Maximum safe power consumption

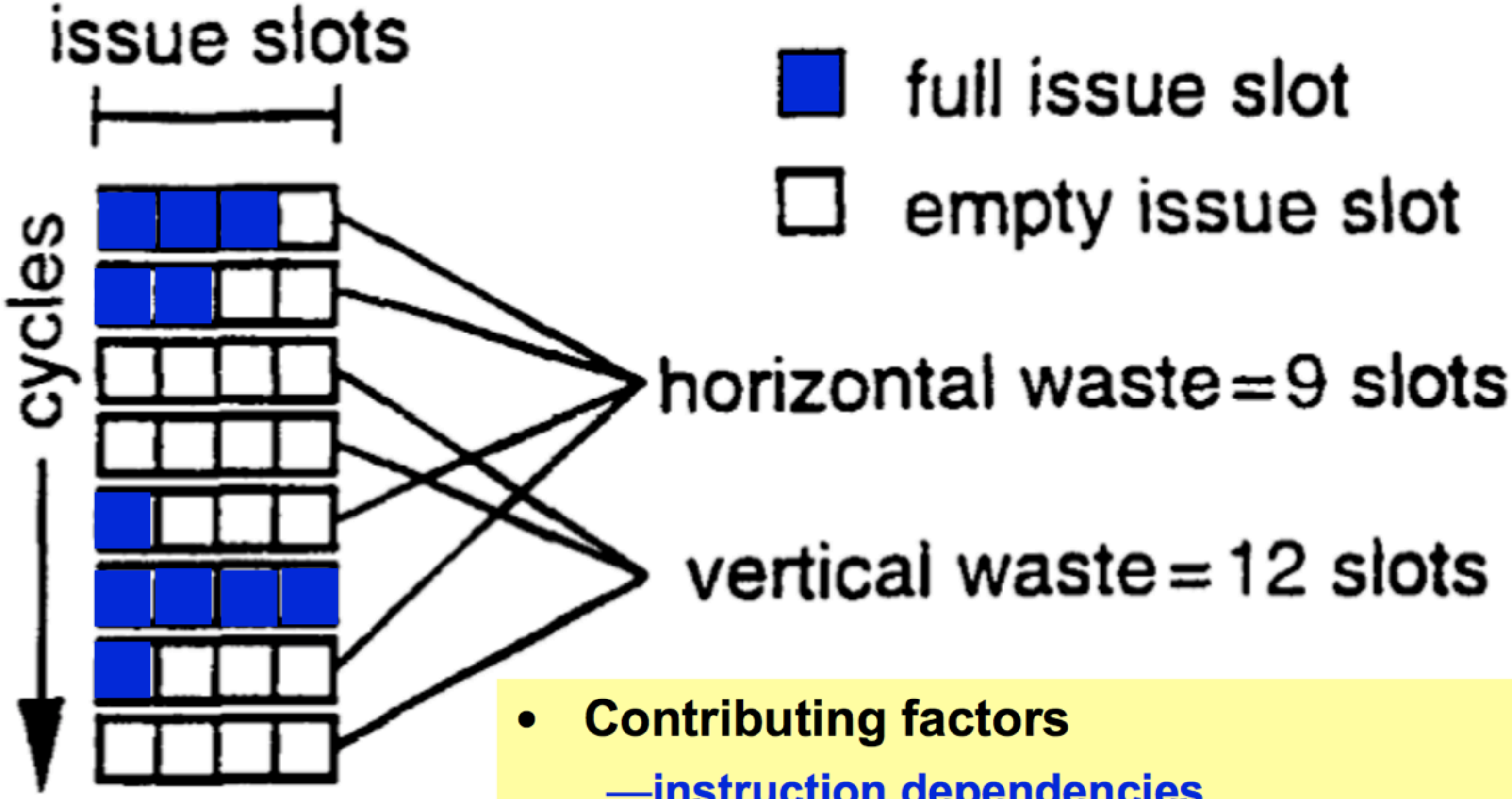
Problems of traditional ILP scaling

- Fundamental circuit limitations[1]
  - delays ⇑ as issue queues ⇑ and multi-port register files ⇑
  - increasing delays limit performance returns from wider issue
- Limited amount of instruction-level parallelism[1]
  - inefficient for codes with difficult-to-predict branches
- Power and heat stall clock frequencies

[1] The case for a single-chip multiprocessor, K. Olukotun, B. Nayfeh, L. Hammond, K. Wilson, and K. Chang, ASPLOS-VII, 1996.

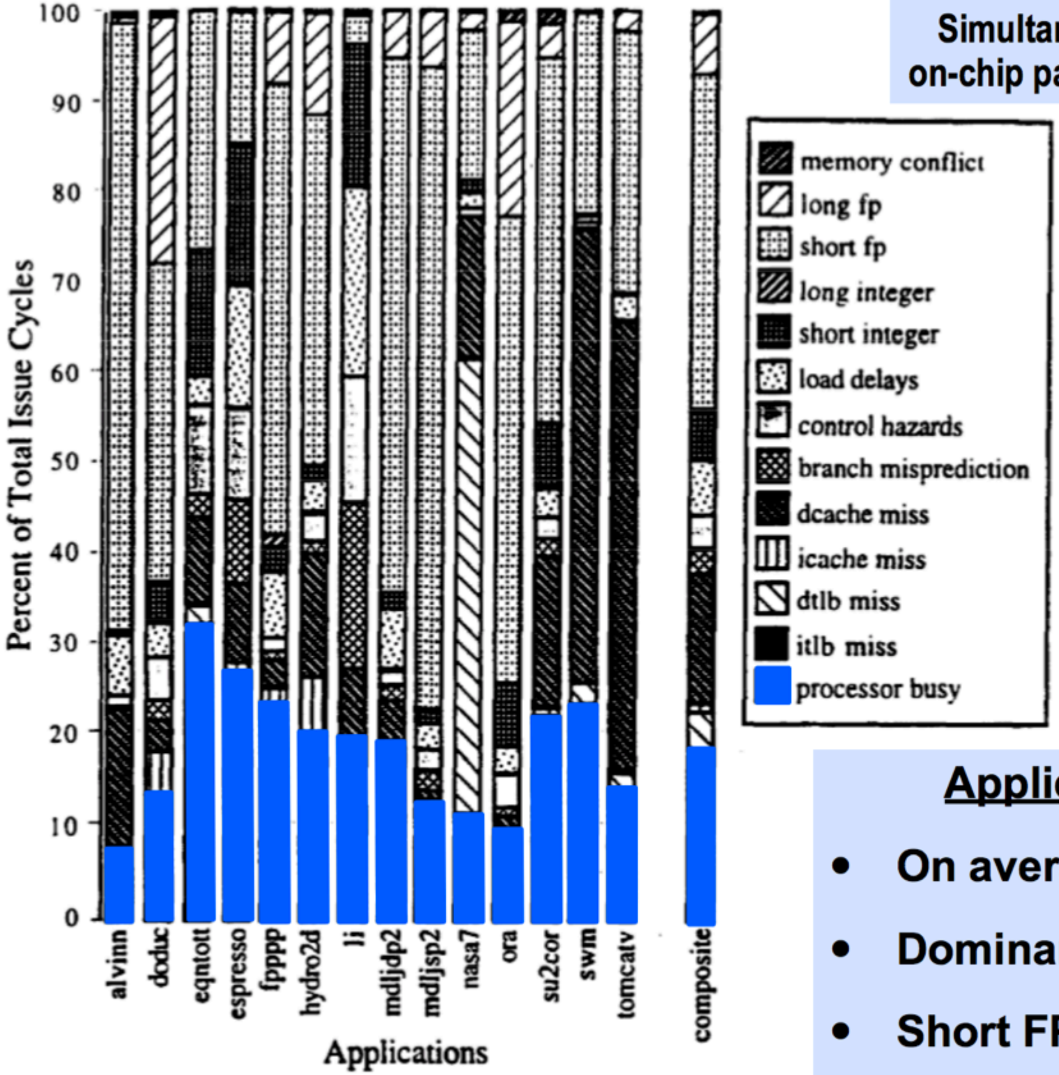
ILP impacts

Issue Waste



- **Contributing factors**
  - —**instruction dependencies**
  - —**long-latency operations within a thread**

Simulations of 8-issue Superscalar



Simultaneous multithreading: maximizing on-chip parallelism, Tullsen et. al. ISCA, 1995.

Summary: Highly underutilized

Applications: most of SPEC92

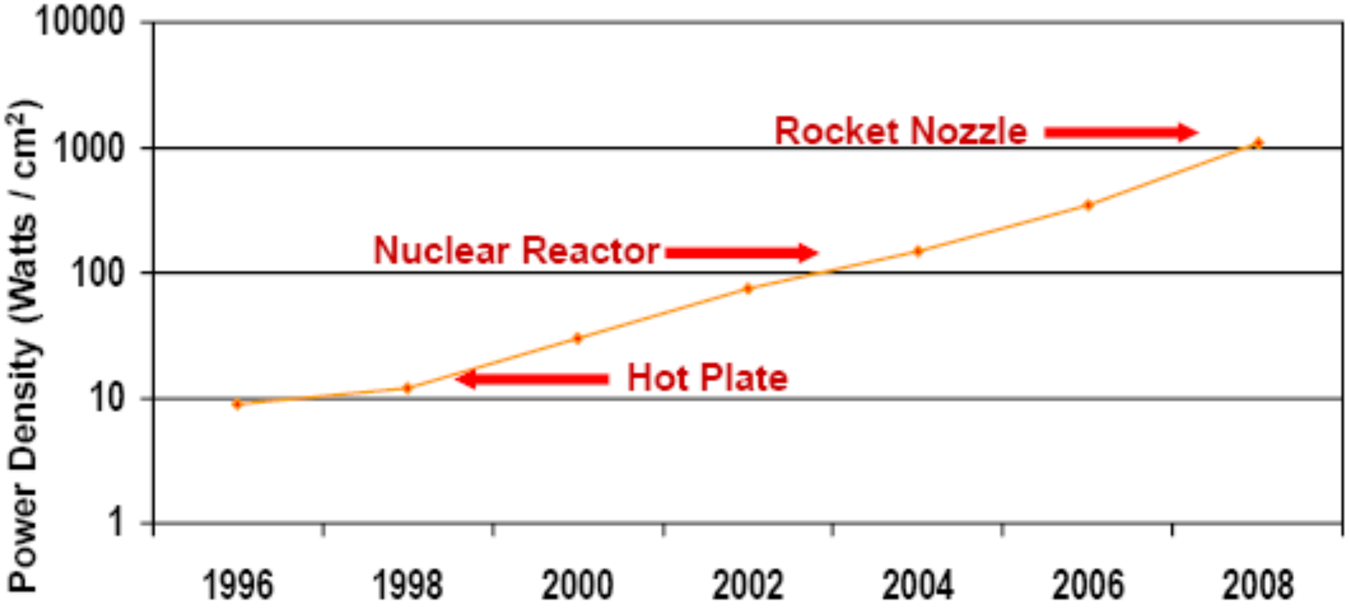
- On average < 1.5 IPC (19%)
- Dominant waste differs by application
- Short FP dependences: 37%

Power/heat density limits frequency

- Some fundamental physical limits are being reached

Moore's Law Extrapolation:

Power Density for Leading Edge Microprocessors



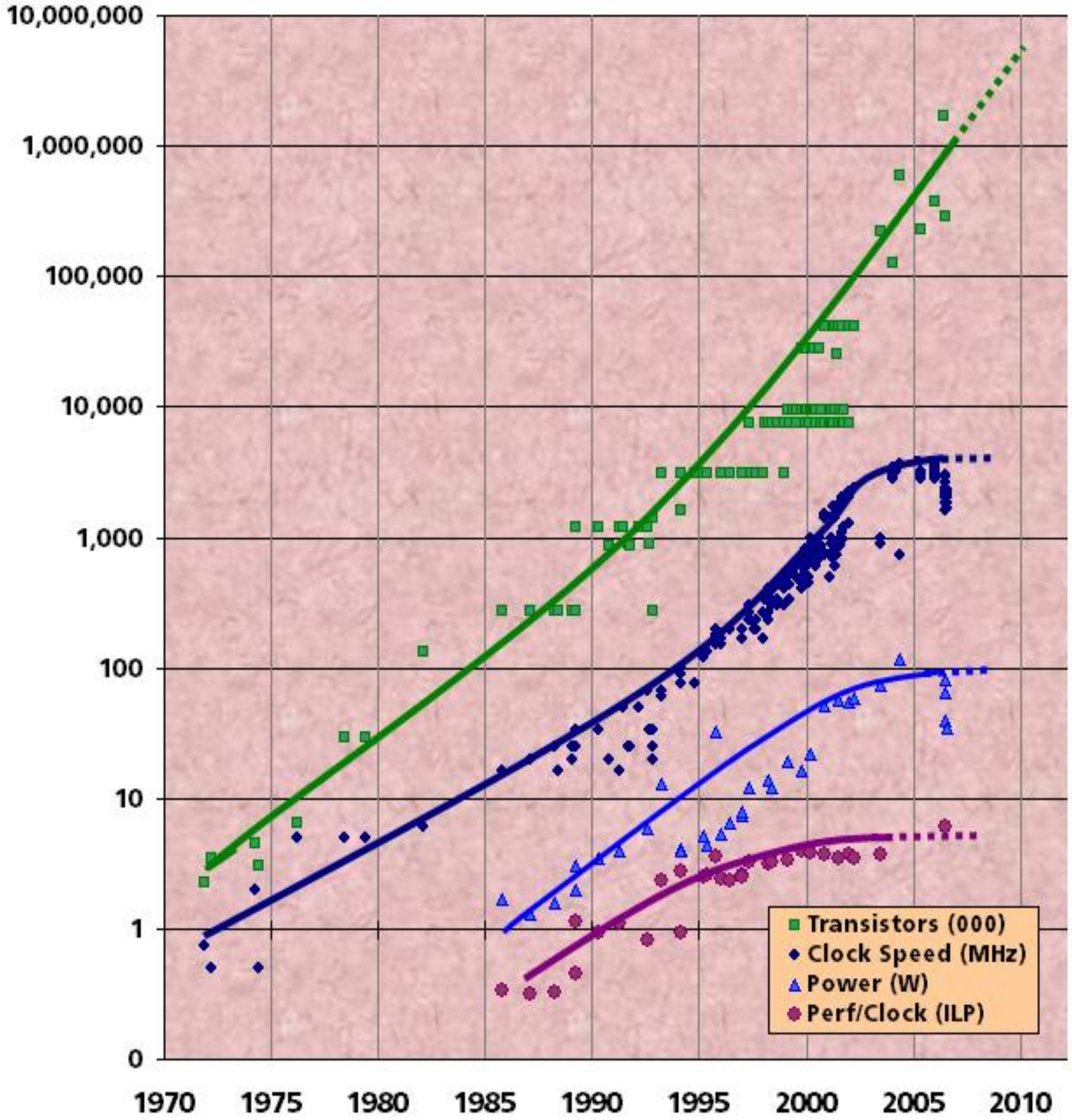
Power Density Becomes Too High to Cool Chips Inexpensively

Source: Shekhar Borkar, Intel Corp

We will have this …

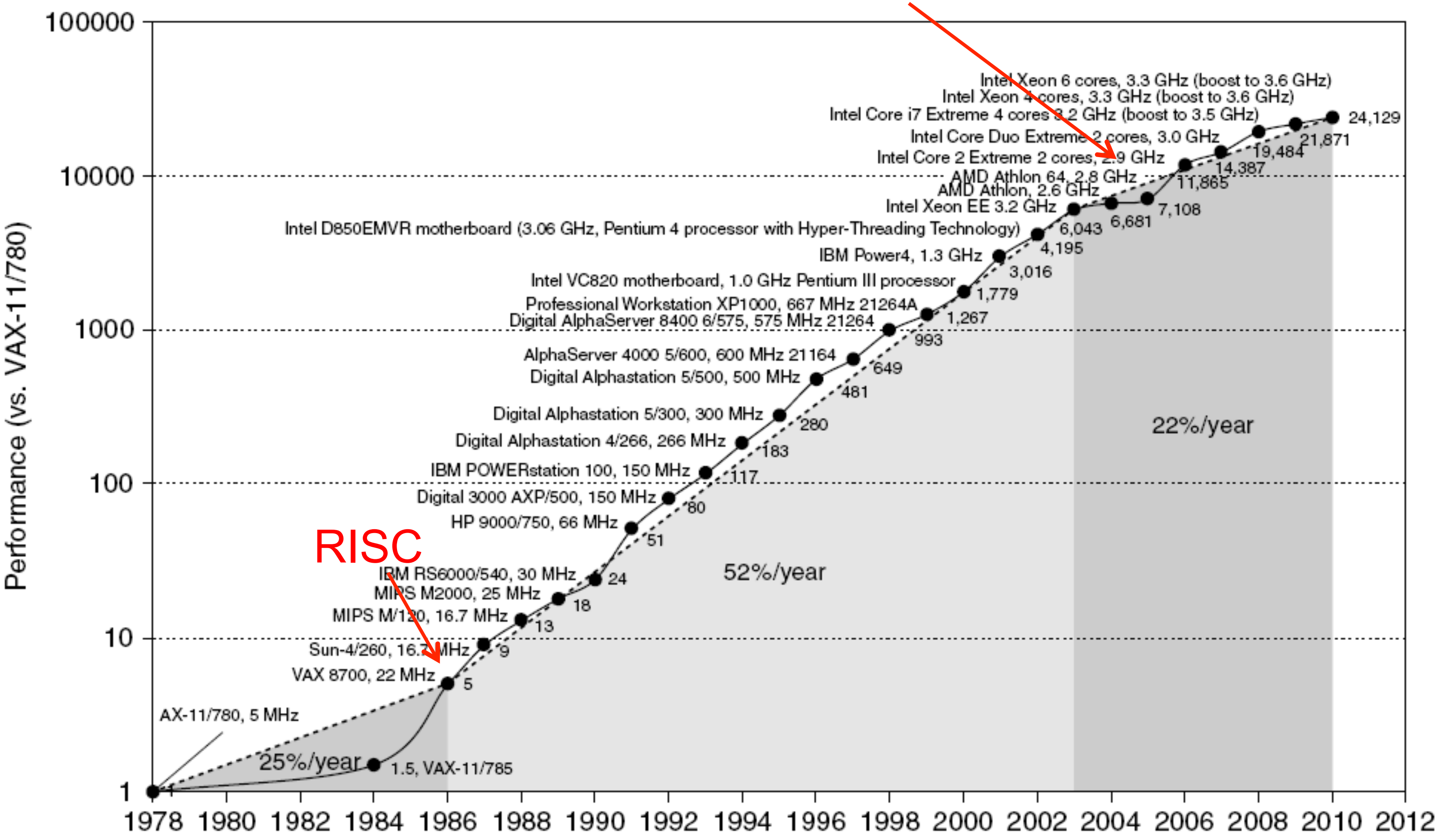
Revolution is happening now

- Chip density is continuing increase ~2x every 2 years
  - Clock speed is not
  - Number of processor cores may double instead
- There is little or no hidden parallelism (ILP) to be found
- Parallelism must be exposed to and managed by software
  - No free lunch



Source: Intel, Microsoft (Sutter) and Stanford (Olukotun, Hammond)

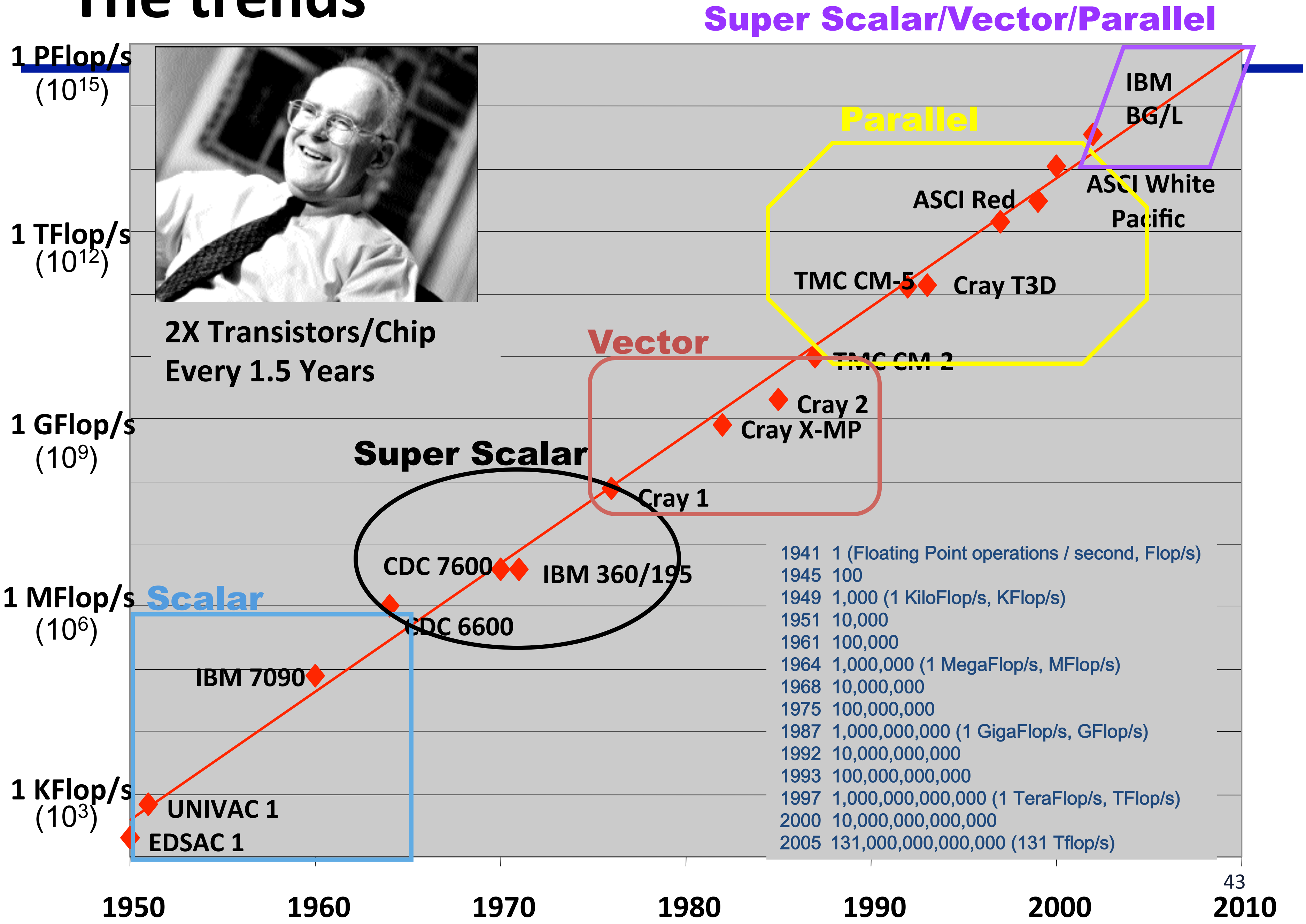
Single Processor Performance



Move to multi-processor

RISC

The trends



2X Transistors/Chip
Every 1.5 Years

1941	1 (Floating Point operations / second, Flop/s)
1945	100
1949	1,000 (1 KiloFlop/s, KFlop/s)
1951	10,000
1961	100,000
1964	1,000,000 (1 MegaFlop/s, MFlop/s)
1968	10,000,000
1975	100,000,000
1987	1,000,000,000 (1 GigaFlop/s, GFlop/s)
1992	10,000,000,000
1993	100,000,000,000
1997	1,000,000,000,000 (1 TeraFlop/s, TFlop/s)
2000	10,000,000,000,000
2005	131,000,000,000,000 (131 Tflop/s)

Recent multicore processors

- Sept 13: Intel Ivy Bridge-EP Xeon E5-2695 v2
  - 12 cores; 2-way SMT; 30MB cache
- March 13: SPARC T5
  - 16 cores; 8-way fine-grain MT per core
- May 12: AMD Trinity
  - 4 CPU cores; 384 graphics cores
- Nov 12: Intel Xeon Phi coprocessor
  - ~60 cores
- Feb 12: Blue Gene/Q
  - 17 cores; 4-way SMT
- Q4 11: Intel Ivy Bridge
  - 4 cores; 2 way SMT;
- November 11: AMD Interlagos
  - 16 cores
- Jan 10: IBM Power 7
  - 8 cores; 4-way SMT; 32MB shared cache
- Tilera TilePro64

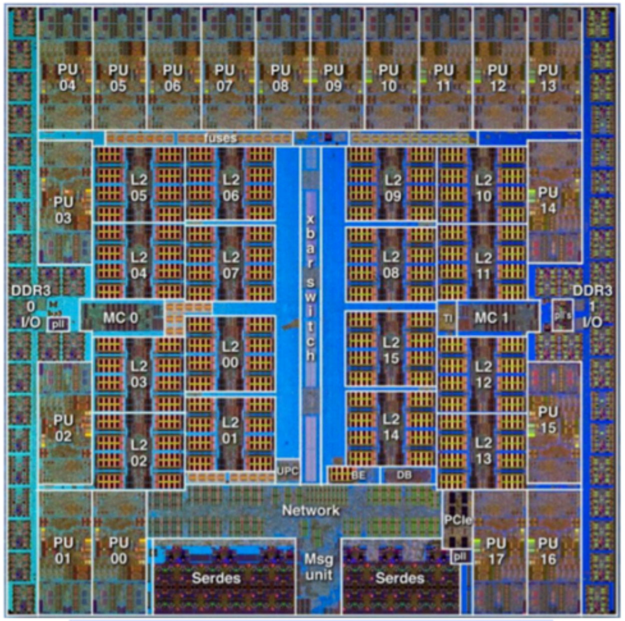
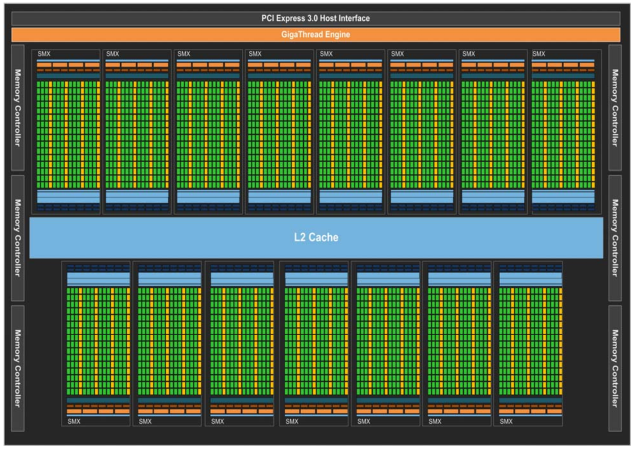
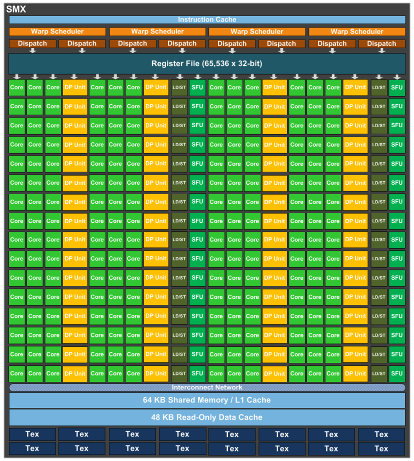


Figure credit: Ruud Haring, Blue Gene/Q compute chip, Hot Chips 23, August, 2011.

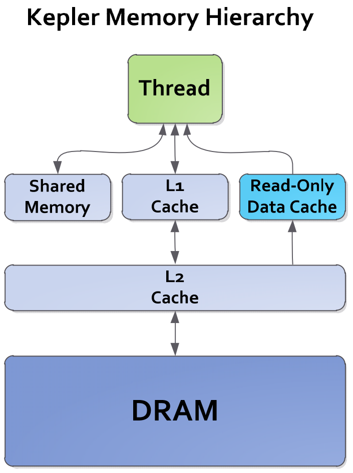
Recent manycore GPU processors

- ~3k cores




SMX
Instruction Cache
Warp Scheduler
Dispatch
Register File (65,536 x 32-bit)
Core
DP Unit
LD/ST
SFU
Interconnect Network
64 KB Shared Memory / L1 Cache
48 KB Read-Only Data Cache
Tex

SMX: 192 single-precision CUDA cores, 64 double-precision units, 32 special function units (SFU), and 32 load/store units (LD/ST).



Kepler Memory Hierarchy

Thread

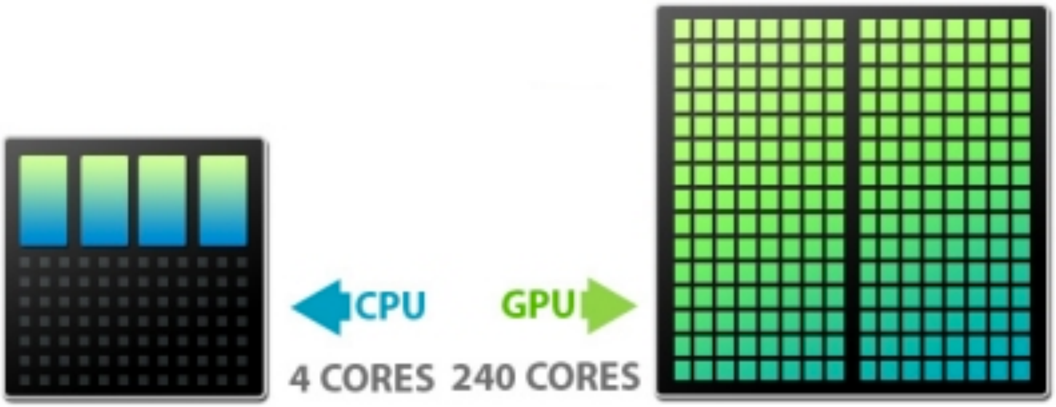
Shared Memory

L1 Cache

Read-Only Data Cache

L2 Cache

DRAM



Current Trends in Architecture

- Cannot continue to leverage Instruction-Level parallelism (ILP)
  - Single processor performance improvement ended in 2003

- New models for performance:
  - Data-level parallelism (DLP)
  - Thread-level parallelism (TLP)
  - Heterogeneity

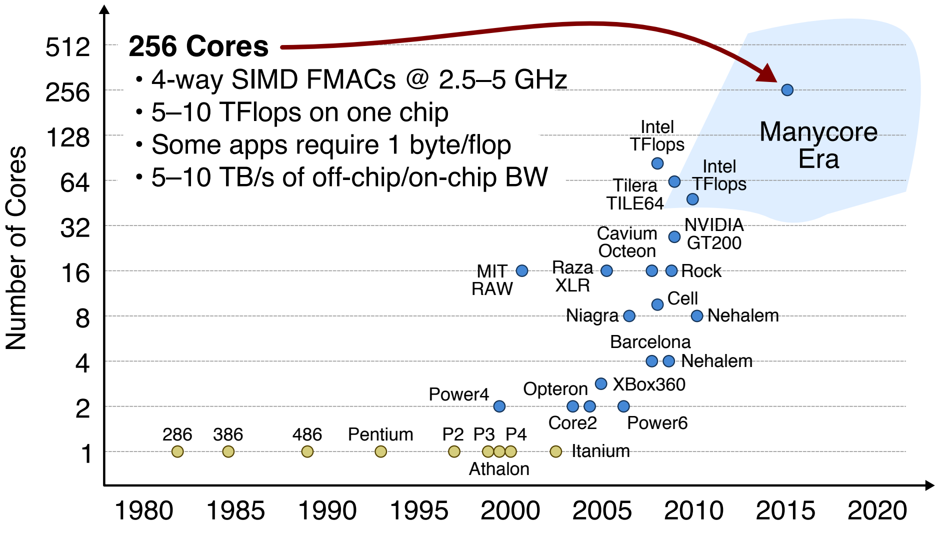
- These require explicit restructuring of the application

Parallelism

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

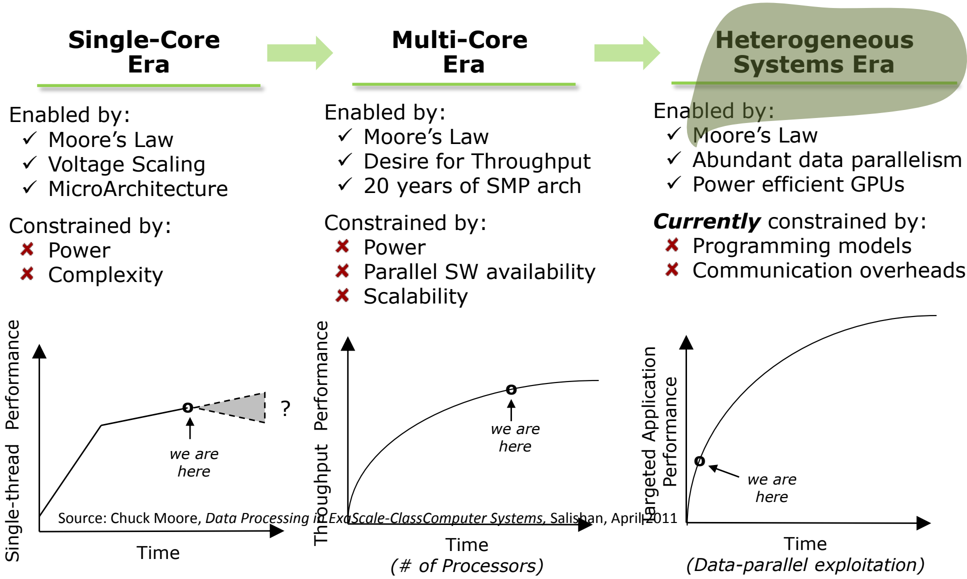
- Classes of architectural parallelism:
  - Instruction-Level Parallelism (ILP)
  - Vector architectures/Graphic Processor Units (GPUs)
  - Thread-Level Parallelism
  - Heterogeneity

Architectural Challenges



256 Cores

- 4-way SIMD FMACs @ 2.5–5 GHz
- 5–10 TFlops on one chip
- Some apps require 1 byte/flop
- 5–10 TB/s of off-chip/on-chip BW



Single-Core Era

Enabled by:
- ✓ Moore's Law
- ✓ Voltage Scaling
- ✓ MicroArchitecture

Constrained by:
- ✗ Power
- ✗ Complexity

Multi-Core Era

Enabled by:
- ✓ Moore's Law
- ✓ Desire for Throughput
- ✓ 20 years of SMP arch

Constrained by:
- ✗ Power
- ✗ Parallel SW availability
- ✗ Scalability

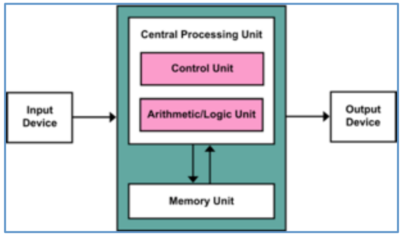
Heterogeneous Systems Era

Enabled by:
- ✓ Moore's Law
- ✓ Abundant data parallelism
- ✓ Power efficient GPUs

Currently constrained by:
- ✗ Programming models
- ✗ Communication overheads

Single-thread Performance
?
we are here
Time
Throughput Performance
we are here
Time
(# of Processors)
Targeted Application Performance
we are here
Time
(Data-parallel exploitation)

Source: Chuck Moore, *Data Processing in ExaScale-ClassComputer Systems*, Salishan, April 2011



- Massive (ca. 4X) increase in concurrency
  - Multicore (4 - <100) → Manycores (100s – 1ks)
- Heterogeneity
  - System-level (accelerators) vs chip level (embedded)
- Compute power and memory speed challenges (two walls)
  - *500x compute power and 30x memory of 2PF HW*
  - *Memory access time lags further behind*

Exercise: Inspect ISA for sum

- cp ~yan/sum.c ~ (copy sum.c file from my home folder to your home folder)
- gcc -save-temps sum.c –o sum
- ./sum 102400

- vi sum.c
- vi sum.s
- Or check from:
  - https://passlab.github.io/CSE564/exercises/sum/

- View them from H drive
- Other system commands:
  - cat /proc/cpuinfo to show the CPU and #cores
  - top command to show system usage and memory

Backup

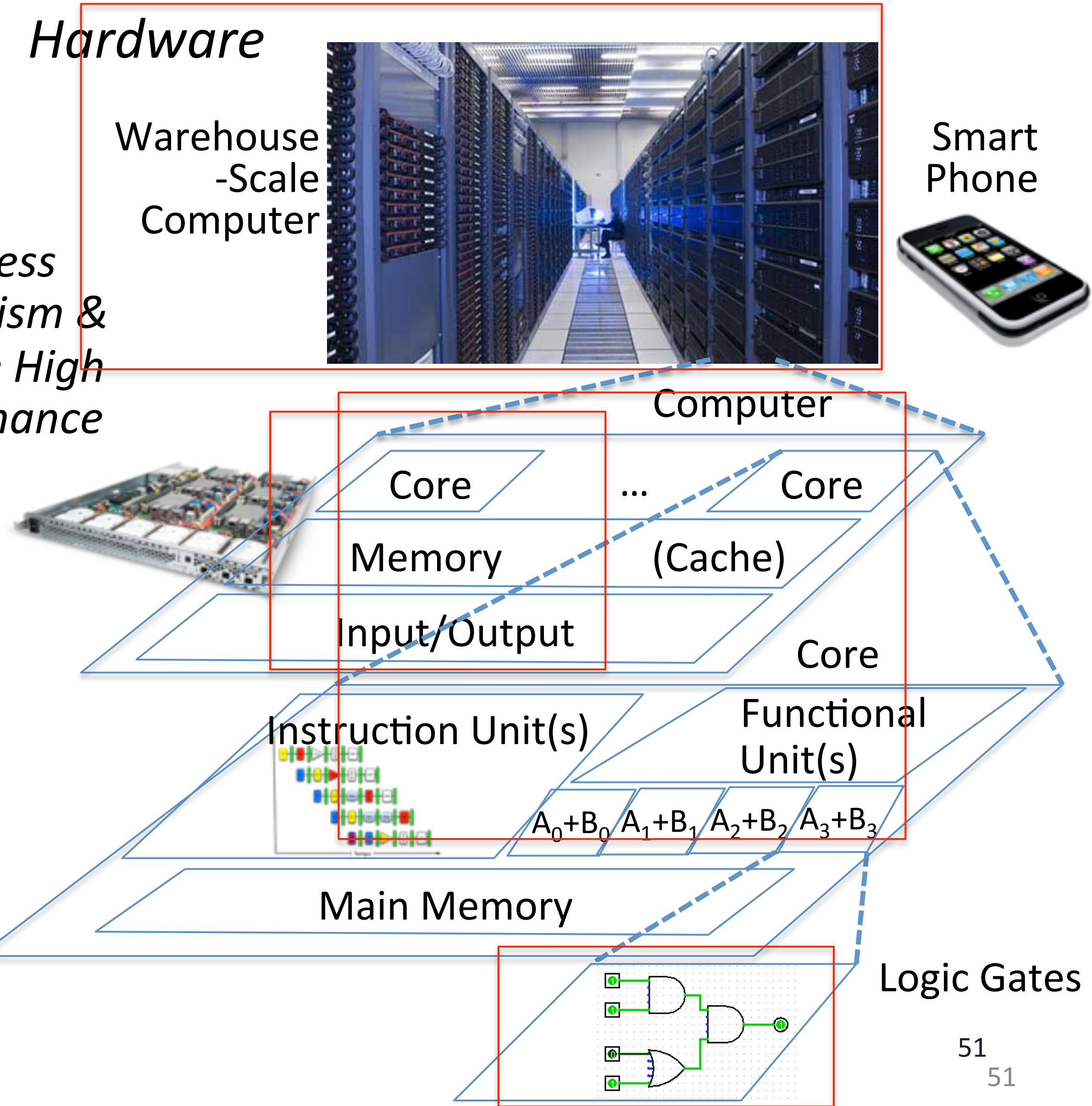
New-School Machine Structures

Software

- Parallel Requests
  Assigned to computer
  e.g., Search "cats"
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions
  All gates functioning in parallel at same time

Harness Parallelism & Achieve High Performance

Hardware



Warehouse -Scale Computer

Smart Phone

Computer

Core … Core

Memory (Cache)

Input/Output

Core

Instruction Unit(s)

Functional Unit(s)

A_0+B_0 A_1+B_1 A_2+B_2 A_3+B_3

Main Memory

Logic Gates

Coping with Failures

- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour

Coping with Failures

- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour

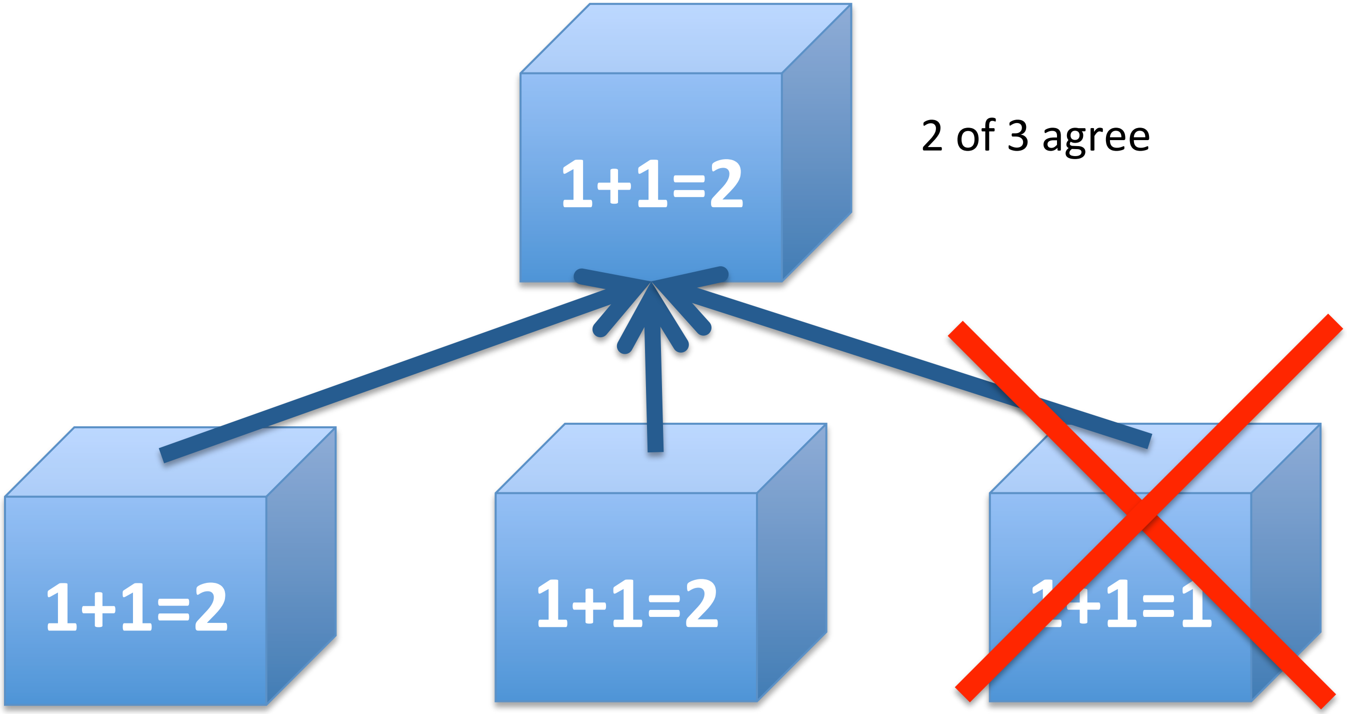
50,000 x 4 = 200,000 disks

200,000 x 4% = 8000 disks fail

365 days x 24 hours = 8760 hours

Great Idea:
Dependability via Redundancy

- Redundancy so that a failing piece doesn’t make the whole system fail



2 of 3 agree

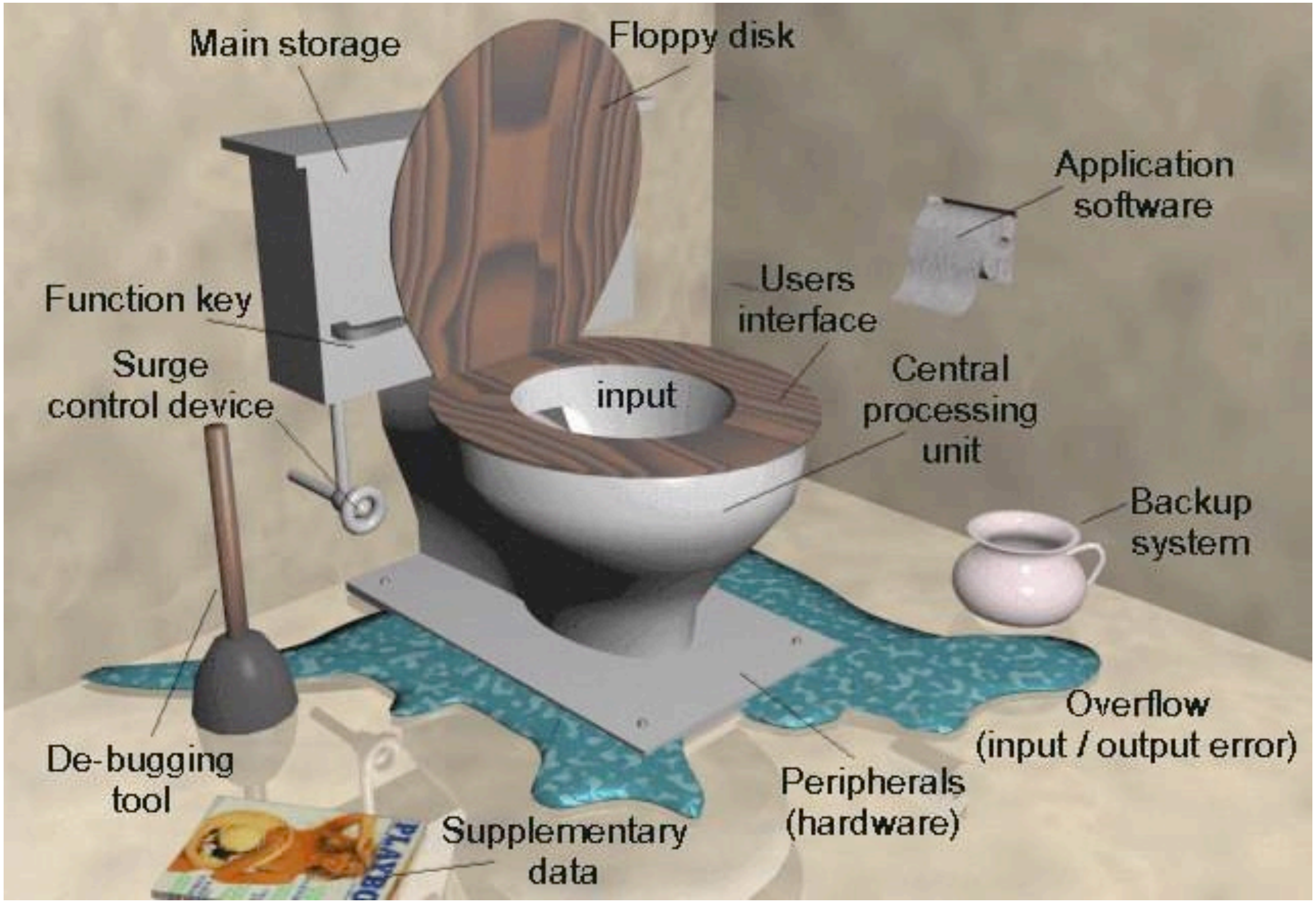
FAIL!

Increasing transistor density reduces the cost of redundancy

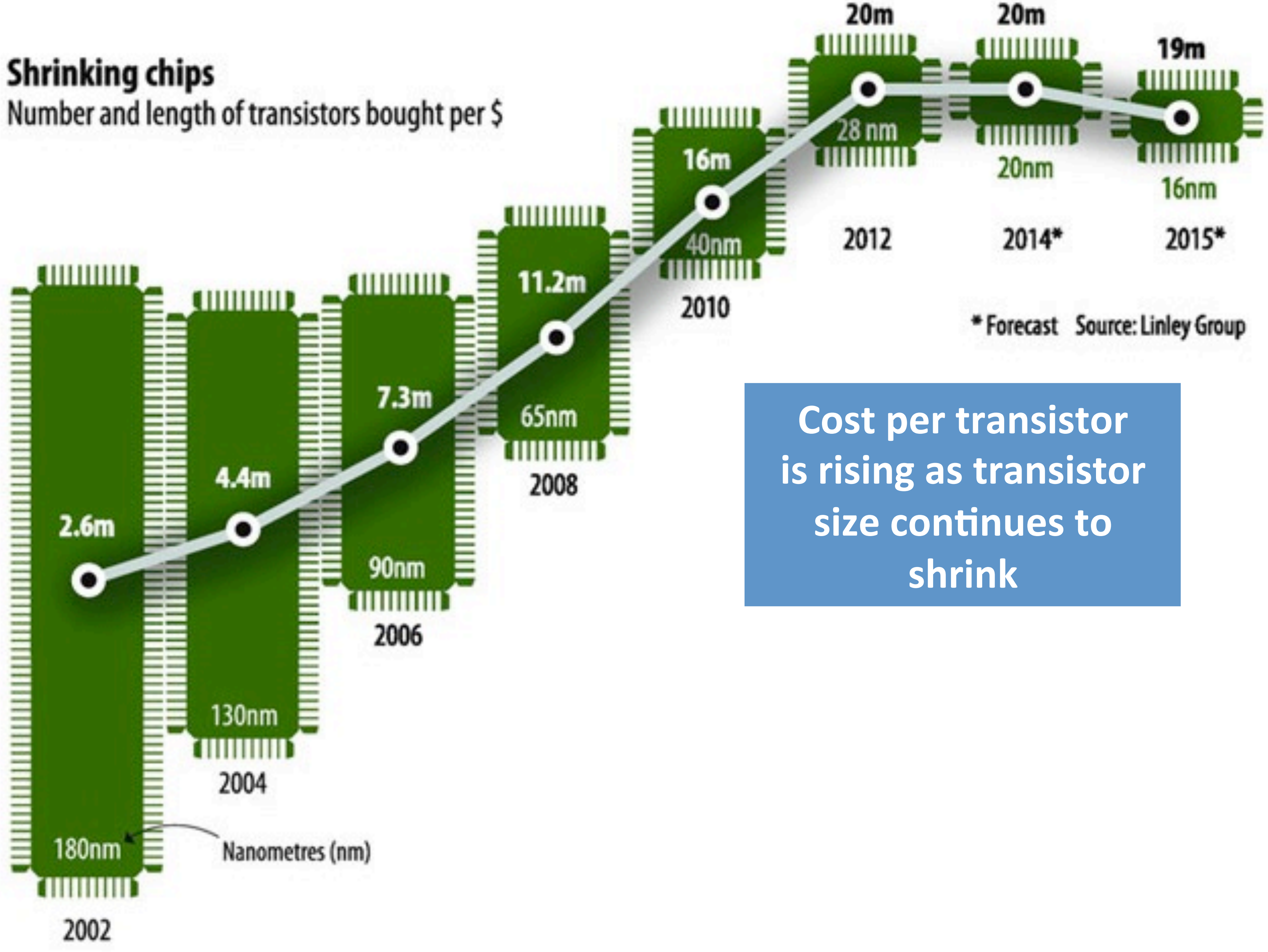
Great Idea: Dependability via Redundancy

- Applies to everything from datacenters to storage to memory to instructors
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
  - Redundant memory bits of so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)

Understanding Computer Architecture



de.pinterest.com



Shrinking chips

Number and length of transistors bought per $

* Forecast Source: Linley Group

Cost per transistor is rising as transistor size continues to shrink