CS532: Operating Systems Foundations

Winter 2020

Prof. Karen L. Karavanic

Portland State
UNIVERSITY

Operating Systems: Major Trends

PSU CS 533

Prof. Karen L. Karavanic

Portland State
UNIVERSITY

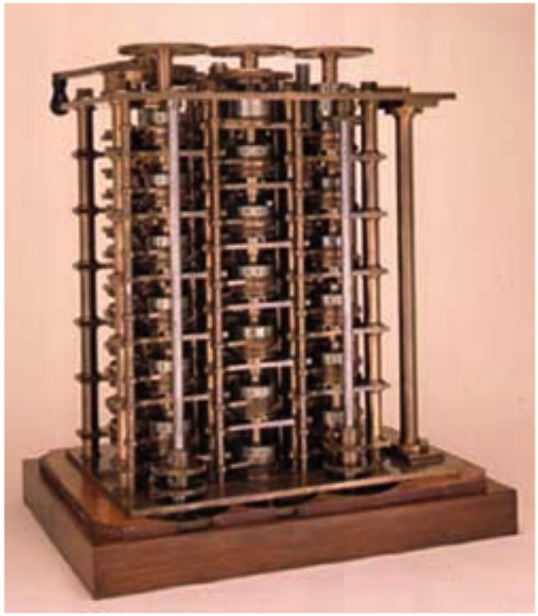
Outline

- The Single Core Era (-> 2006)
  - 0. The Operating System is Human
  - 1. Batch Processing
  - 2. MultiProgramming and Timesharing
  - 3. Personal Computing & Connectivity
- The Multicore Era (2006 ->)
  - 5. Why Multicore?
  - 6. Manycore

The Single Core Era: Early Days

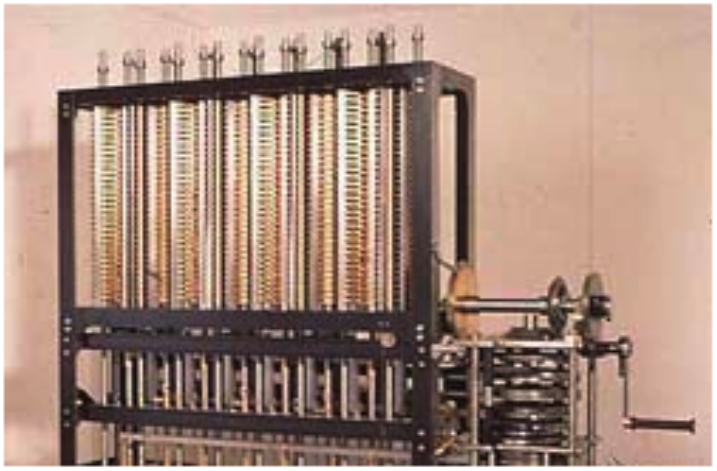
- **The First Computer!**
- **Charles Babbage (1792-1871) and Ada Lovelace: "Analytical Engine"**

- **1821: Difference Engine No. 1: add, subtract, solve polynomial equations**
  - Required 25,000 precision-crafted parts
  - Difference Engine No. 2: Simpler version
  - Analytical Engine: multiplication, division, algebra



The Single Core Era: Early Days

- 1990: Science Museum in London builds Difference Engine No. 2 in one year
  - Cost: $500,000
  - Weight: three tons
  - Size: 11 feet long, 7 feet tall
  - Calculated successive values of seventh-order polynomial equations containing up to 31 digits
  - Proves Babbage's design

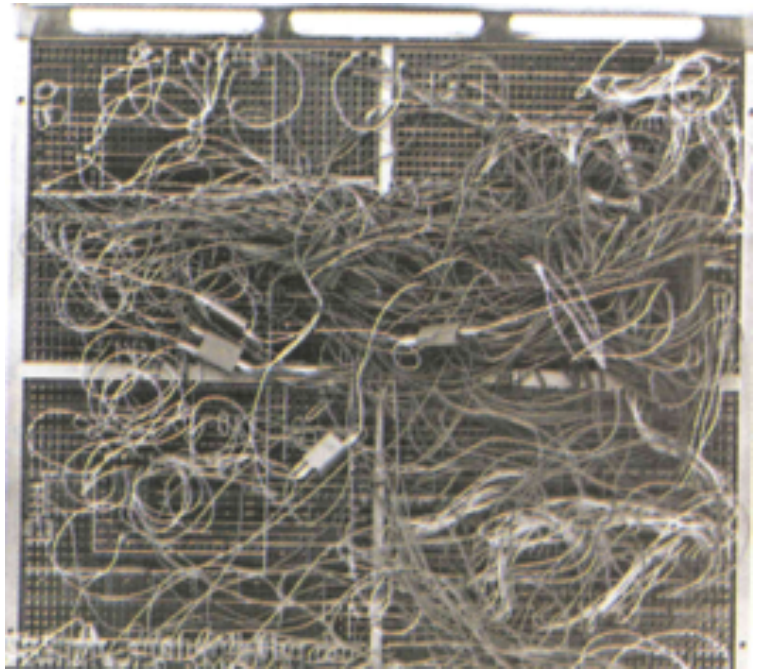


The Single Core Era: Early Days

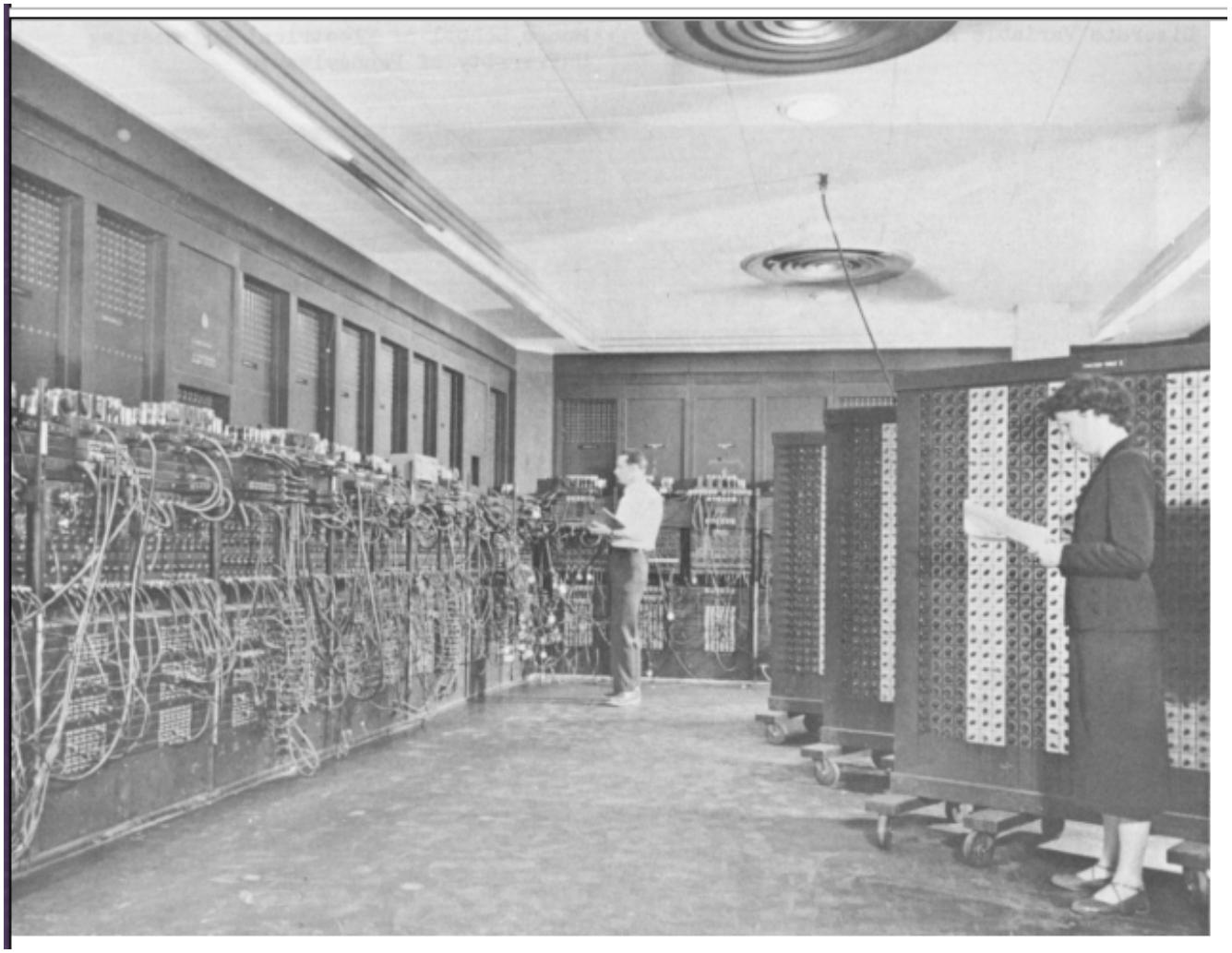
- **"First Generation" Computing (1940-50)**
  - Goal: compute trajectories for warfare
  - Relays/vacuum tubes (about 20,000)

The Single Core Era: Early Days

- Programming: plugboards
- Operating System: none
- Innovation: punch cards



The Eniac (1946)



The Single Core Era: Early Days

- "Second Generation" Computing (1950-64)
- Innovation: transistors, mainframes
- Innovation: first compiler (fortran)
- Batch systems
- Programming: Fortran, assembly language
- Innovation: Operating Systems: FMS, IBSYS

The Single Core Era #1: Batch Systems

Early "OS" was human:

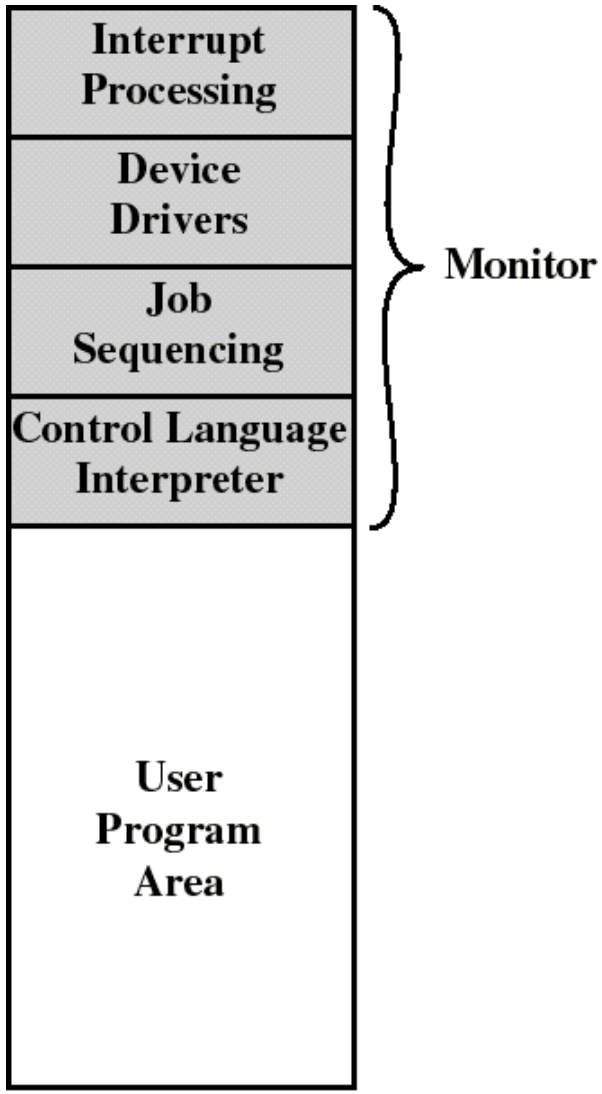
- Operator carries punch cards to an I/O device
- Input device reads cards to tape
- Operator carries tape over to computer
- Single job runs by reading instructions from tape and writing output to tape
- Operator carries tape back to I/O machine which prints output

- Cards include instruction to stop to load tape or compiler

- Switching from one activity to another, loading and saving data were manual, unlike today

The Monitor

- Monitor reads jobs one at a time from the input device
- Monitor places a job in the user program area
- A monitor instruction branches to the start of the user program
- Execution of user program continues until:
  - end-of-pgm occurs
  - error occurs
- Causes the CPU to fetch its next instruction from Monitor



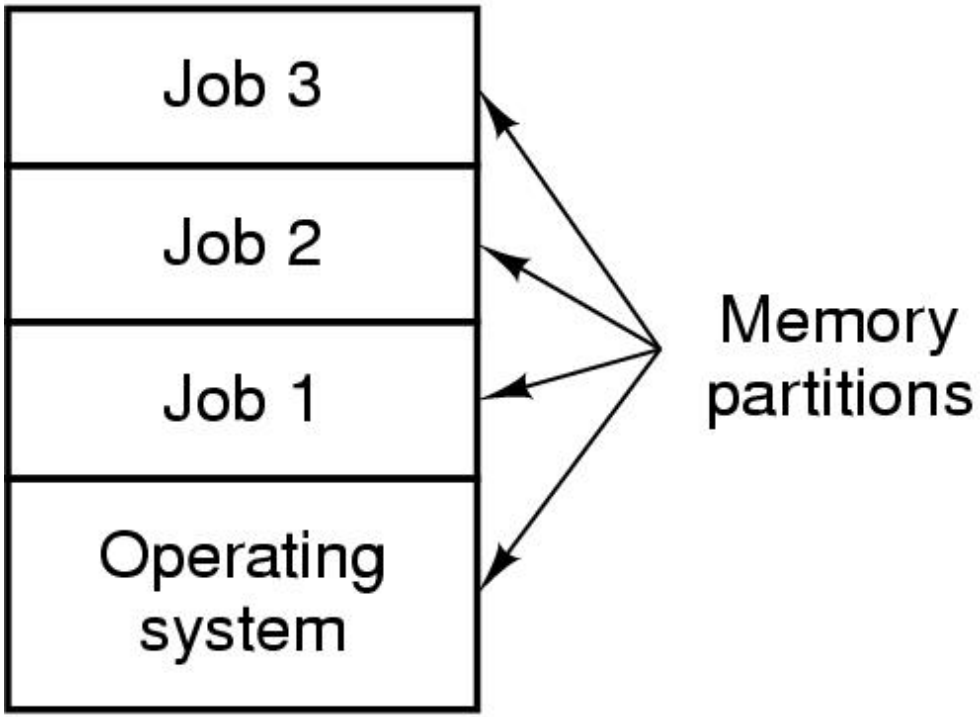
Memory Layout of Resident Monitor

Batch OS: Hardware Requirements

- **Memory protection**
  - Need to protect the monitor code from the user programs
- **Timer**
  - prevents a job from monopolizing the system
  - an interrupt occurs when time expires
- **Privileged instructions**
  - can be executed only by the monitor
  - an interrupt occurs if a program tries these instructions
- **Interrupts**
  - provides flexibility for relinquishing control to and regaining control from user programs

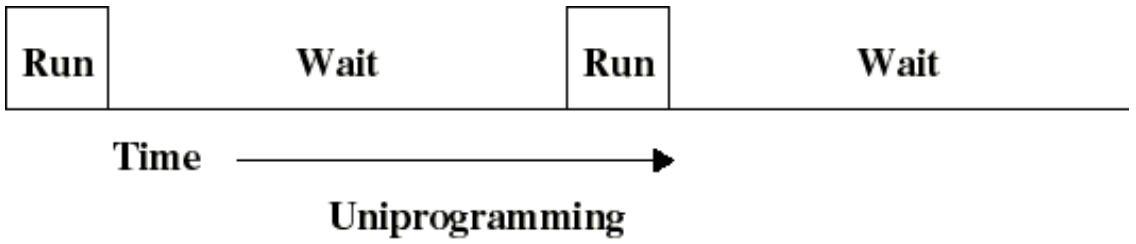
The Single Core Era #2: Multiprogramming

- **Major Innovation: Multiprogramming**
  - three jobs in memory at once



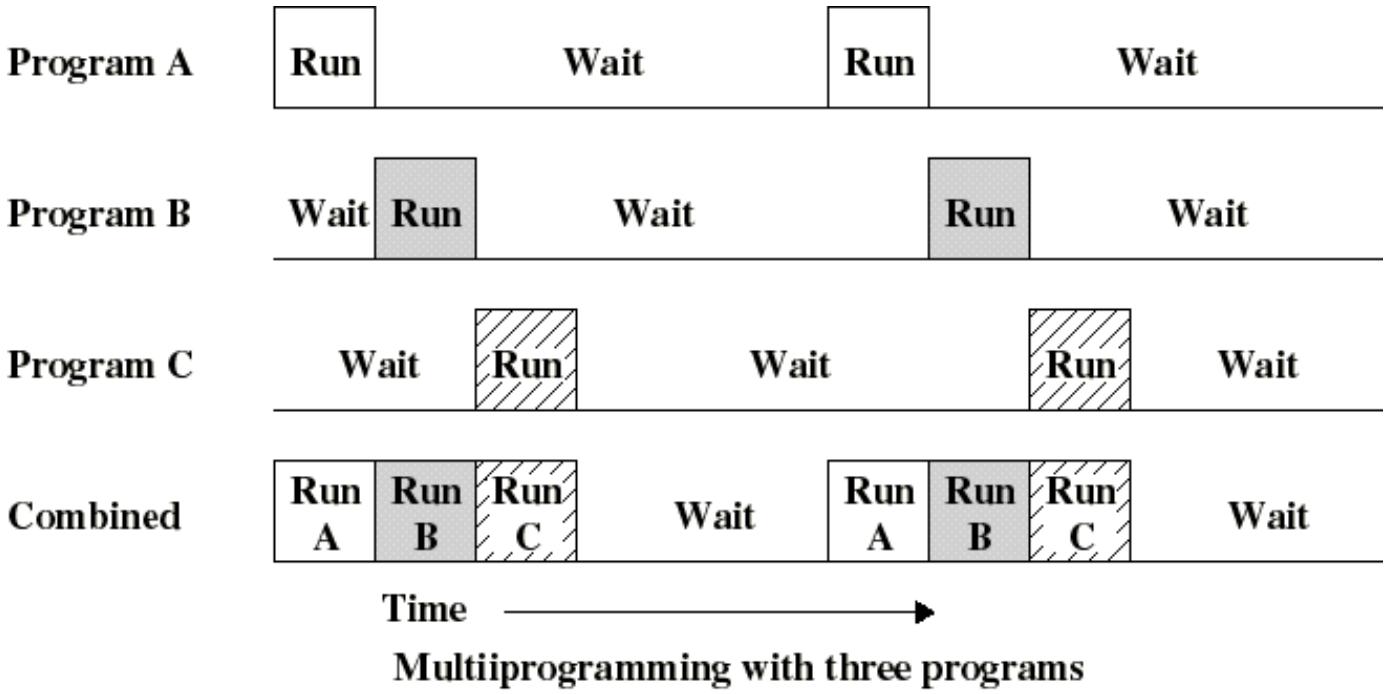
Multiprogrammed Batch Systems: The Insight

- I/O operations are exceedingly slow (compared to instruction execution)
- A program containing even a very small number of I/O ops, will spend most of its time waiting for them
- Hence: poor CPU usage when only one program is present in memory



Multiprogrammed Batch Systems: The Insight

- If memory can hold several programs, then CPU can switch to another one whenever a program is awaiting for an I/O to complete
- This is multitasking (multiprogramming)



Multiiprogramming with three programs

Multiprogramming (cont'd)

- **Multiprogramming is a *virtualization* of the single CPU**
- **Hardware support:**
  - I/O interrupts and (possibly) DMA
    - in order to execute instructions while I/O device is busy
  - Memory management
    - several ready-to-run jobs must be kept in memory
  - Memory protection (data and programs)
- **Software support from the OS:**
  - Scheduling (which program should run next ?)
  - Resource Management (contention, request ordering)

The Single Core Era #3: Time Sharing Systems (TSS)

- Batch multiprogramming does not support interaction with users
- TSS extends multiprogramming to handle multiple *interactive* jobs
- Single core Processor's time is shared among multiple users
- Multiple users simultaneously access the system through terminals
  - Because of slow human reaction time, a typical user needs 2 sec of processing time per minute
  - Then (about) 30 users should be able to share the same system without noticeable delay in the computer reaction time
- Concurrency: two users try to write to same file

The Single Core Era #3: Time Sharing Systems (TSS)

- New OS needs:
  - Shared File Systems: files must be *protected* from unauthorized users (multiple users…)
  - Improvements needed for the user interface, connection / networking speeds
- Operating Systems: Multics, Unix
- Hardware needs: memory capacity
  - What if this program doesn't fit in my memory??
  - HW/SW Solution: *Virtualize* the memory
  - Virtual addresses are used in programs
  - They are translated to physical addresses at runtime

The Single Core Era #4: Personal Computing & Connectivity (1980-1990s)

- Hardware Innovation: [Very] Large Scale Integration ([V]LSI)
- Platform Innovation: microcomputers / PCs
- Innovation: GUI (XEROX), X
- Innovation: fast networks, internet, WWW, client/server model, multithreading
- Operating Systems: DOS, Windows, Linux

The Single Core Era #4: Personal Computing & Connectivity (1980-1990s)

- Operating Systems: DOS, Windows, Linux
- The POSIX API
  - Programmers can write *Posix-compliant* code and it will run across Unix/Linux systems from different vendors
- OS must support: communication, data exchange, disconnection, battery lifetimes
- Security: servers running 24/7 are vulnerable

The Single Core Era #5: The Memory Hierarchy

- Why have a *hierarchy* of memory?
- How does it work?

Storage Trends

SRAM

Metric	1980	1985	1990	1995	2000	2005	2010	*2010:1980*
$/MB	19,200	2,900	320	256	100	75	60	*320*
access (ns)	300	150	35	15	3	2	1.5	*200*

DRAM

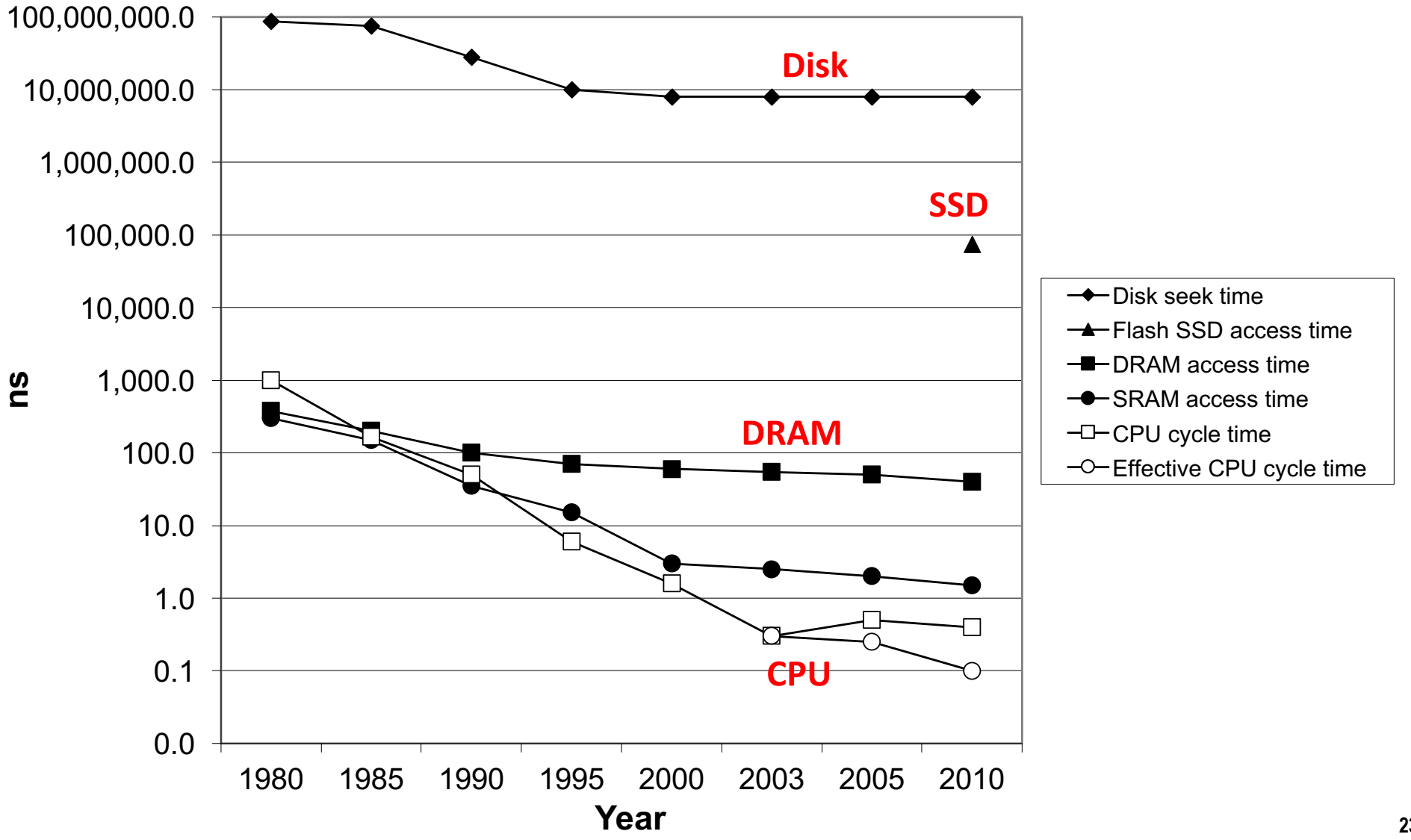
Metric	1980	1985	1990	1995	2000	2005	2010	*2010:1980*
$/MB	8,000	880	100	30	1	0.1	0.06	*130,000*
access (ns)	375	200	100	70	60	50	40	*9*
typical size (MB)	0.064	0.256	4	16	64	2,000	8,000	*125,000*

Disk

Metric	1980	1985	1990	1995	2000	2005	2010	*2010:1980*
$/MB	500	100	8	0.30	0.01	0.005	0.0003	*1,600,000*
access (ms)	87	75	28	10	8	*4*	*3*	*29*
typical size (MB)	1	10	160	1,000	20,000	160,000	1,500,000	*1,500,000*

The CPU-Memory Gap

The gap widens between DRAM, disk, and CPU speeds.

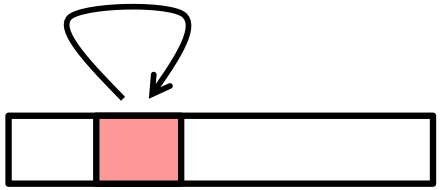


Locality to the Rescue!

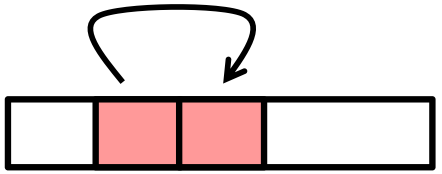
The key to bridging this CPU-Memory gap is a fundamental property of computer programs known as locality

Locality

- **Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently**



- **Temporal locality:**
  - Recently referenced items are likely to be referenced again in the near future



- **Spatial locality:**
  - Items with nearby addresses tend to be referenced close together in time

Locality Example

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

- **Data references**
  - Reference array elements in succession (stride-1 reference pattern). — **Spatial locality**
  - Reference variable `sum` each iteration. — **Temporal locality**
- **Instruction references**
  - Reference instructions in sequence. — **Spatial locality**
  - Cycle through loop repeatedly. — **Temporal locality**

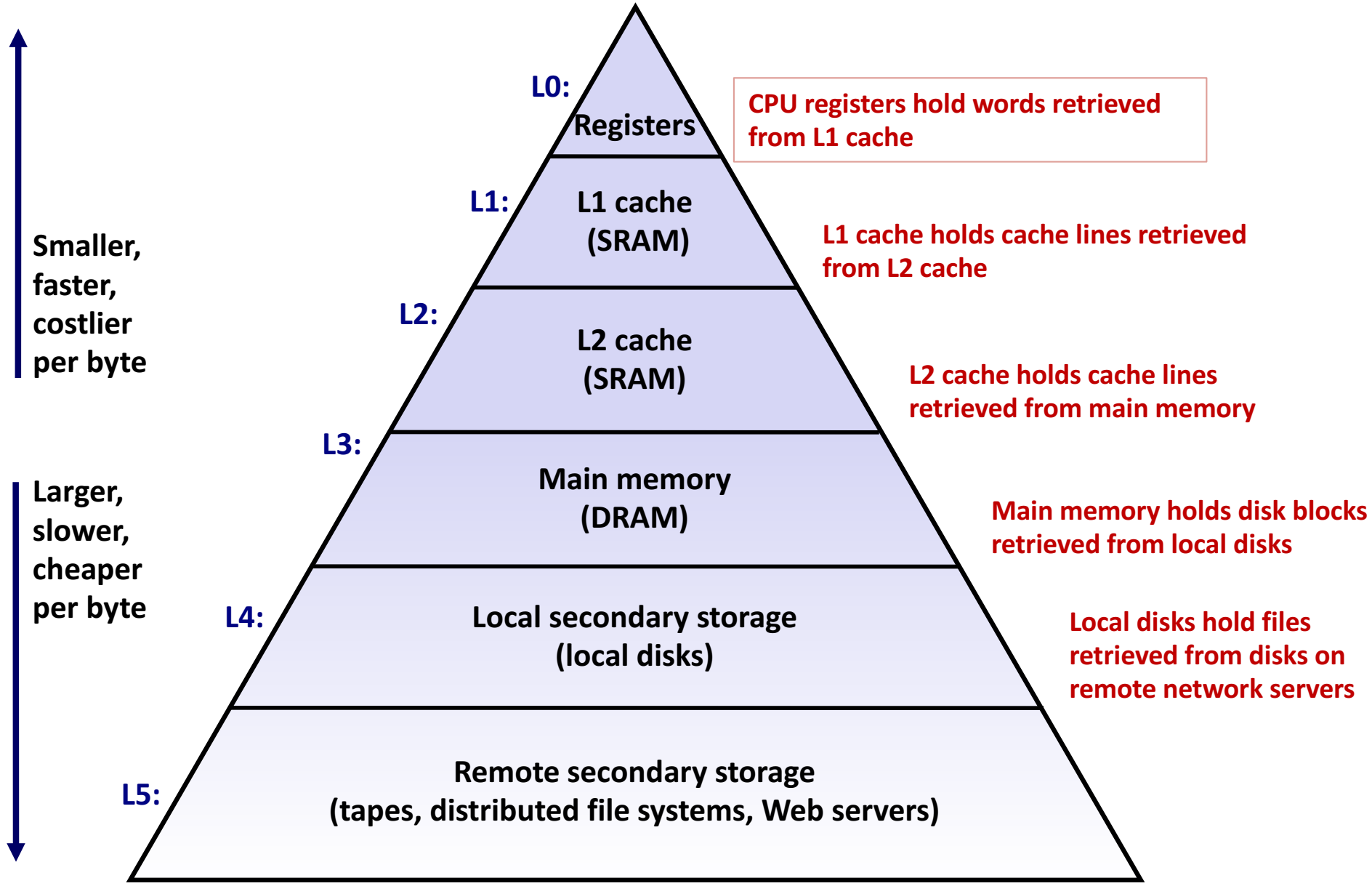
Memory Hierarchies

- **Some fundamental and enduring properties of hardware and software:**
  - Fast storage technologies cost more per byte, have less capacity, and require more power (heat!).
  - The gap between CPU and main memory speed is widening.
  - Well-written programs tend to exhibit good locality.

- **These fundamental properties complement each other beautifully.**

- **They suggest an approach for organizing memory and storage systems known as a memory hierarchy.**

An Example Memory Hierarchy



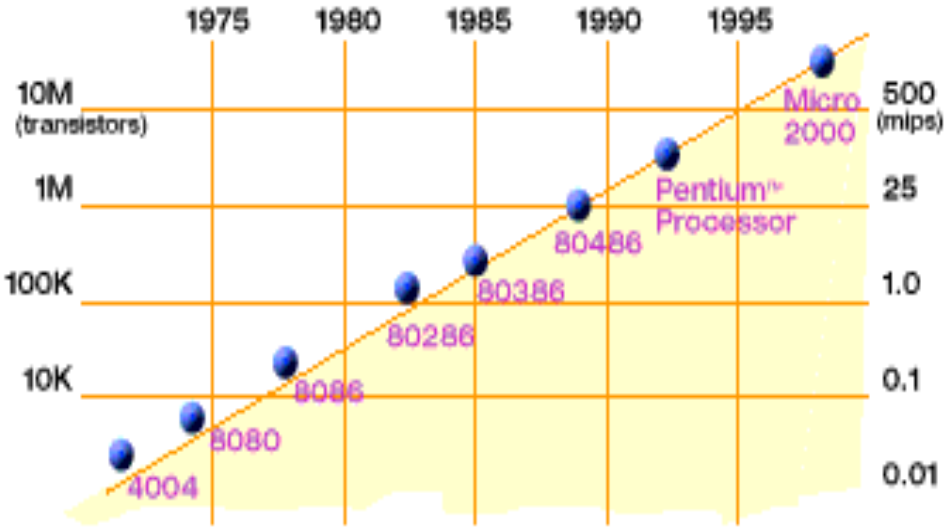
The Single Core Era: Other Advances

- Mobile devices
  - Innovation: Fast Pervasive wireless and cellular
  - Innovation: Power Management
- Embedded Systems
  - Is your car on the internet? Your toaster?
- Open Source
  - Accessibility of OS, rate of upgrades to OS
- Virtualization: including HW support
- Cloud Computing
  - One server farm consumes as much power as a small U.S. city

The Multicore Era

- A single chip now has more than one CPU
- Most systems today are multicore:
  - Servers
  - PCs/laptops
  - Your phone
- Is *everything* multicore??
  - Your toaster may not be
- Software: everything is parallel
- OS: must manage across CPUs

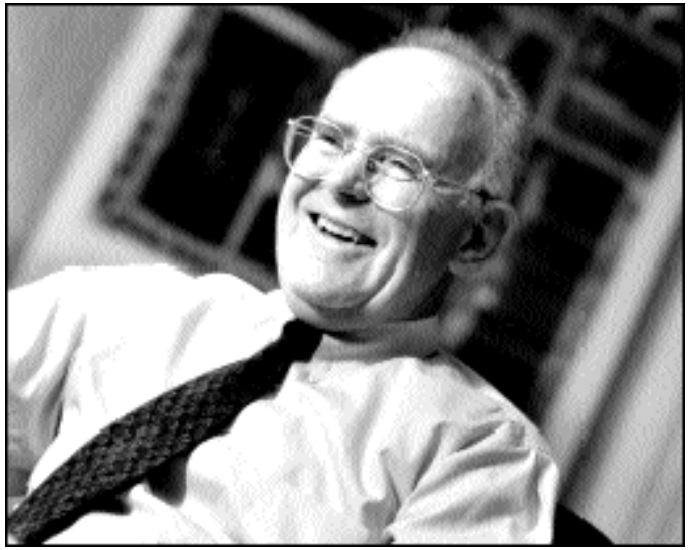
Technology Trends: Microprocessor Capacity



2X transistors/Chip Every 1.5 years

Called "Moore's Law"

Microprocessors have become smaller, denser, and more powerful.

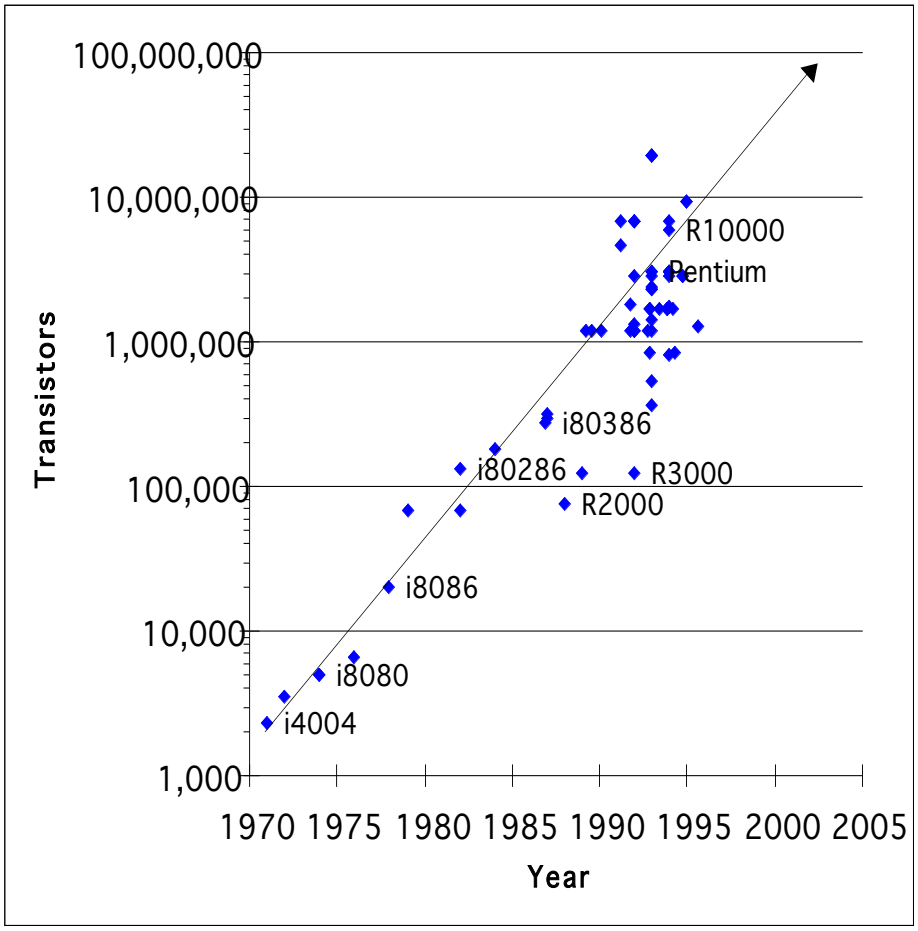


Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

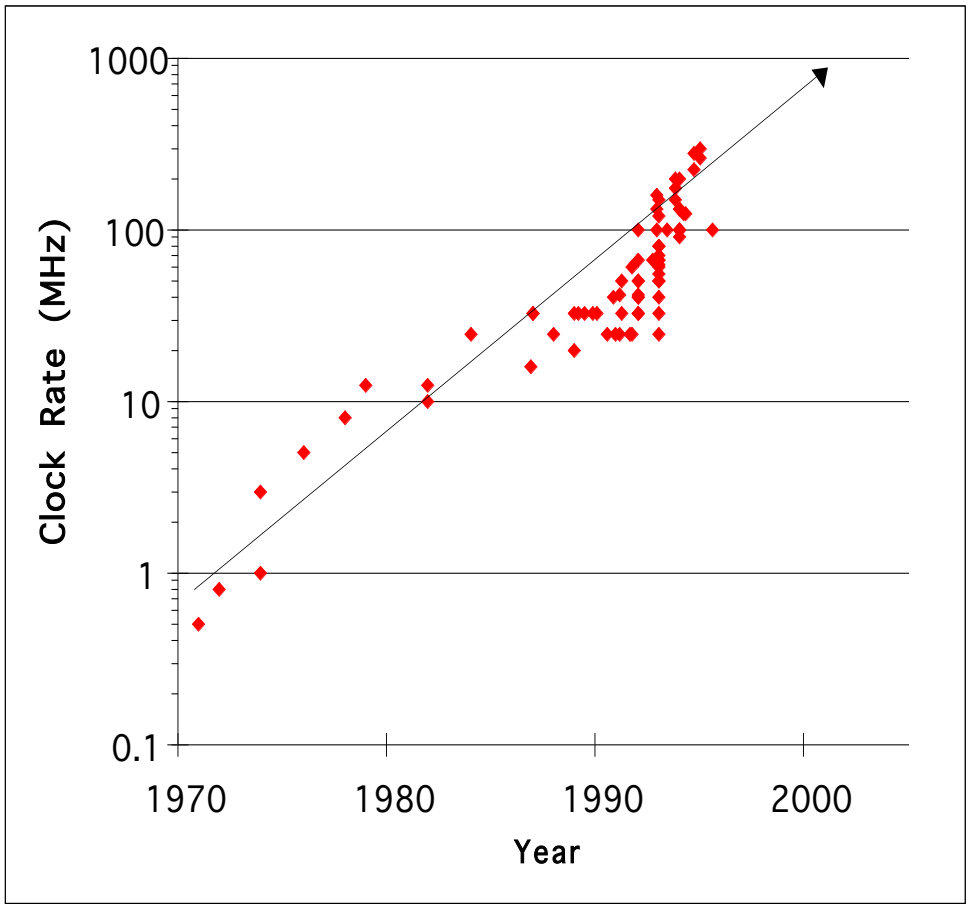
Slide source: Jack Dongarra

Microprocessor Transistors and Clock Rate

Growth in transistors per chip



Increase in clock rate

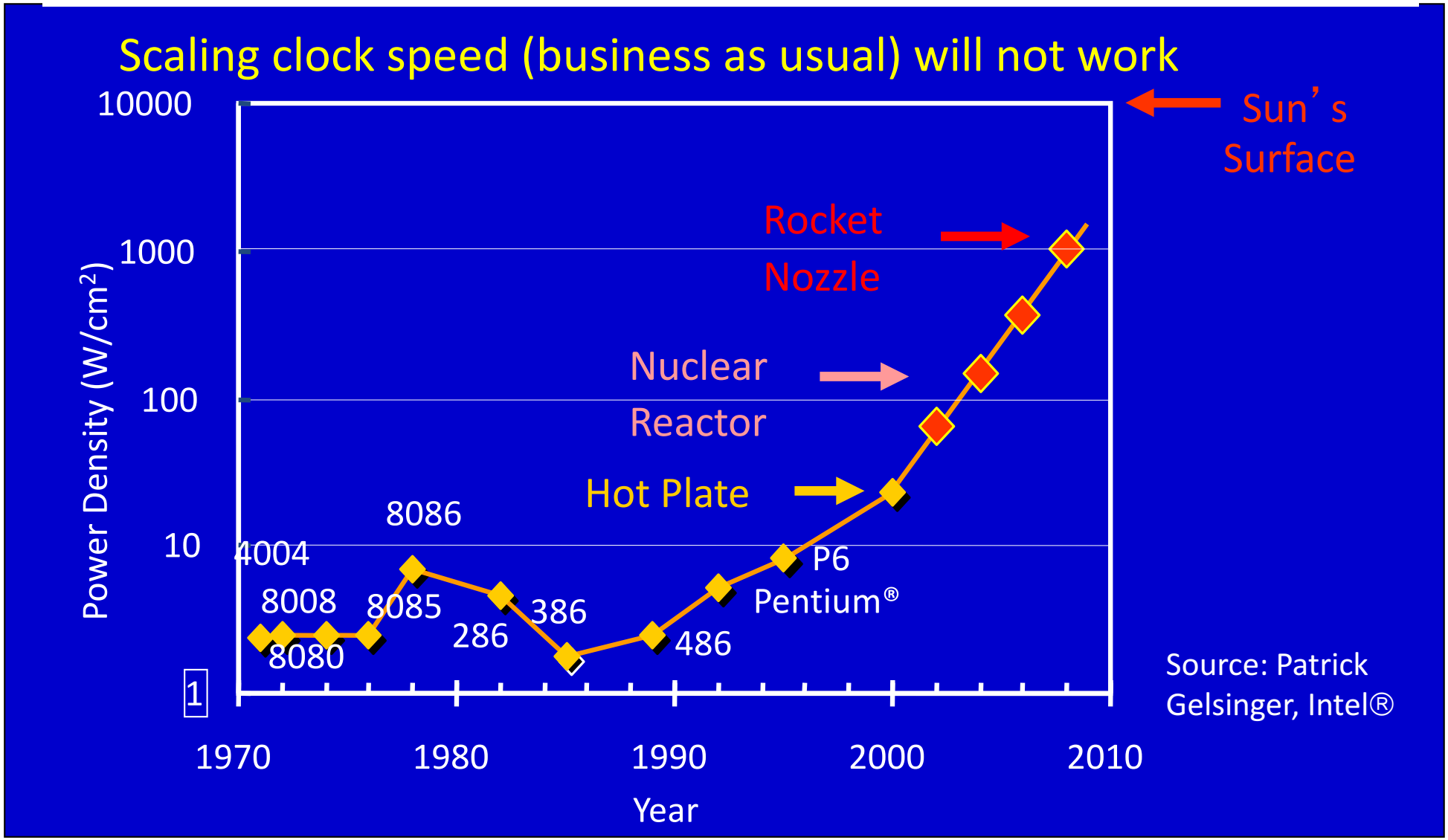


Why bother with parallel programming? Just wait a year or two…

Limit #1: Power density

Can soon put more transistors on a chip than can afford to turn on.

-- Patterson '07



Parallelism Saves Power

- Exploit explicit parallelism for reducing power

  $\text{Power} = (C * V^2 * F)/4$ $\quad$ $\text{Performance} = (\text{Cores} * F)*1$

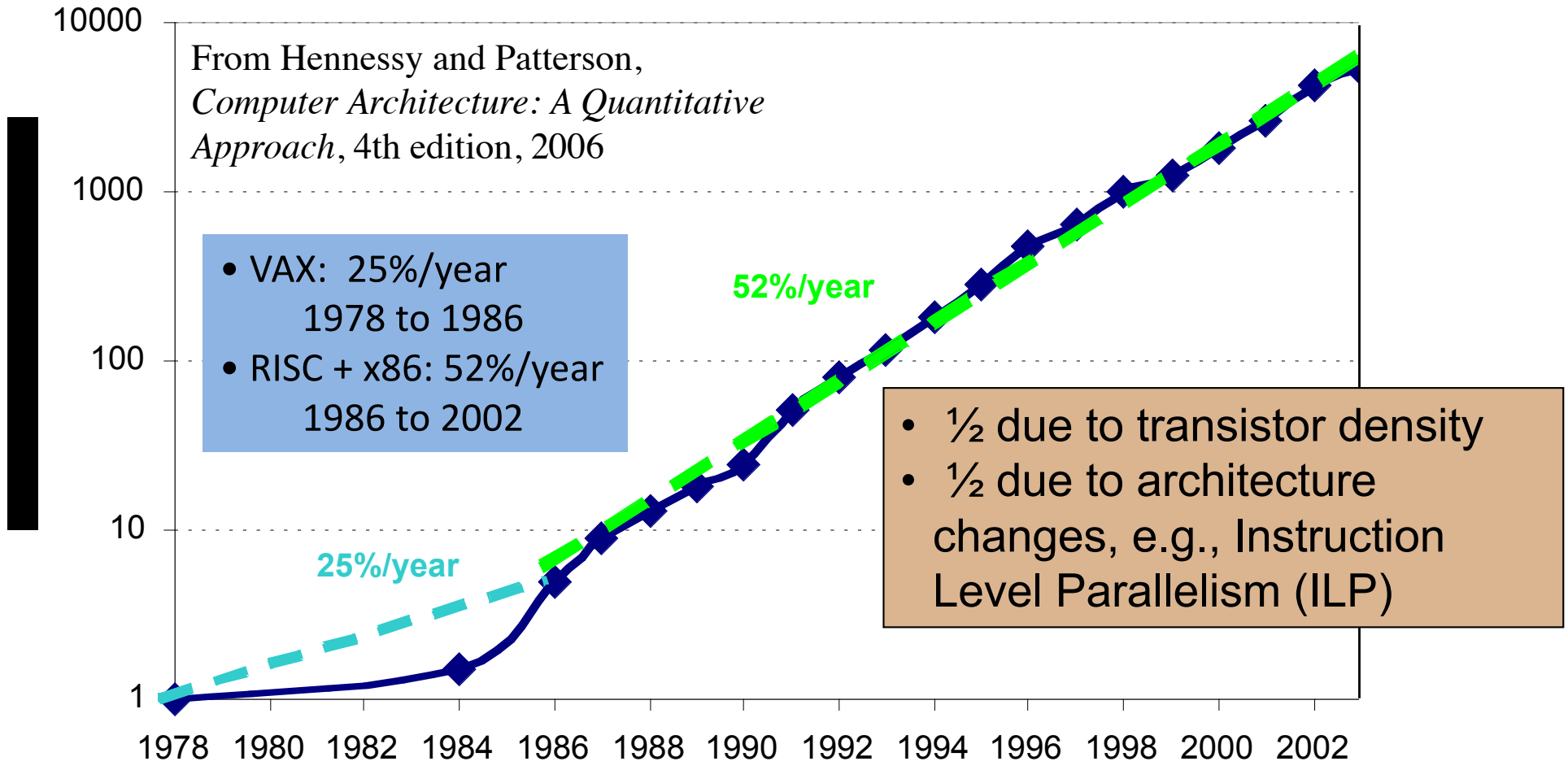
  Capacitance Voltage Frequency

- Using additional cores
  - Increase density (= more transistors = more capacitance)
  - Can increase cores (2x) and performance (2x)
  - Or increase cores (2x), but decrease frequency (1/2): same performance at ¼ the power

- Additional benefits
  - Small/simple cores → more predictable performance

Limit #2: Hidden Parallelism Tapped Out

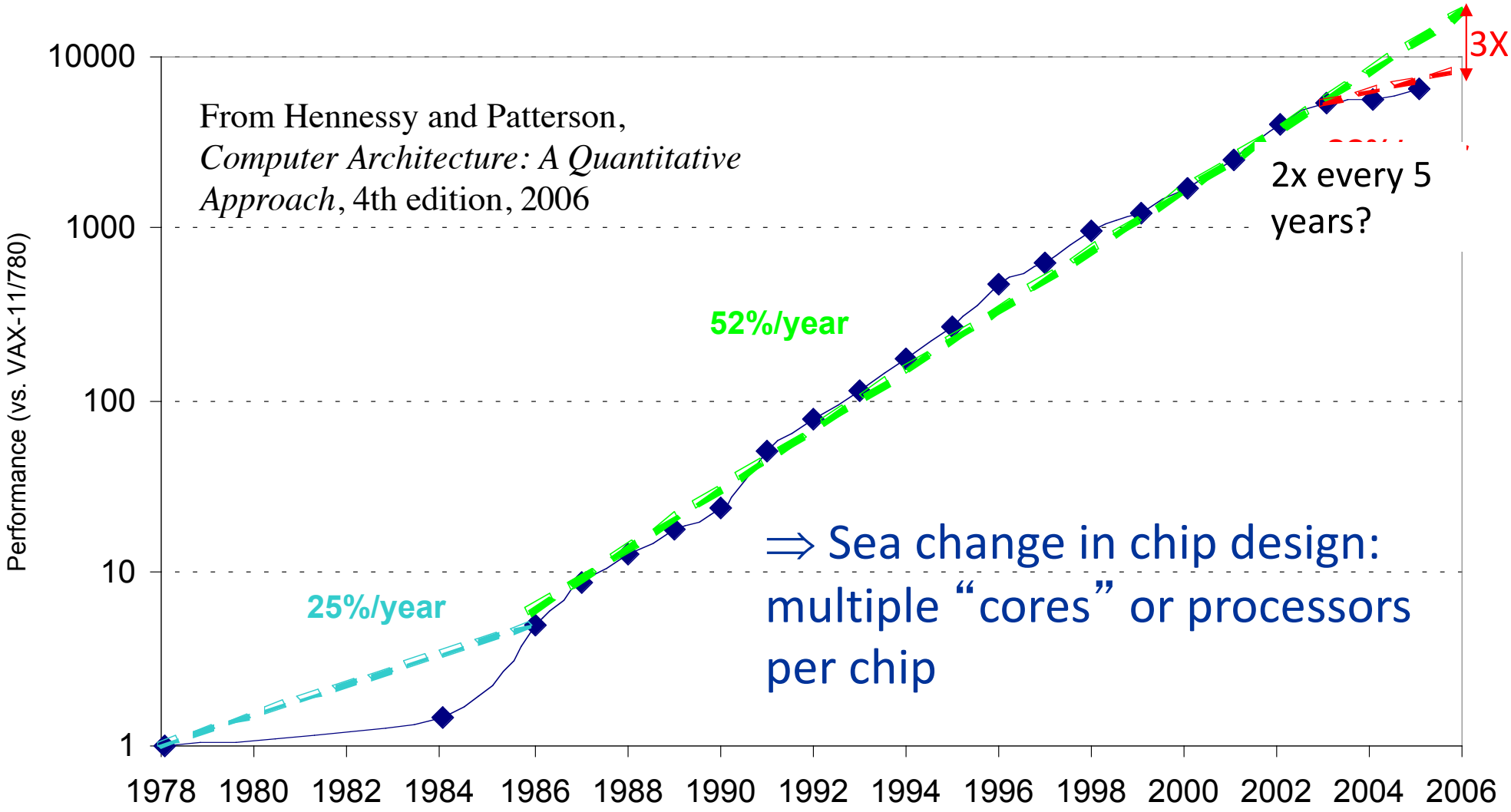
Application performance was increasing by 52% per year as measured by the SpecInt benchmarks here



Limit #2: Hidden Parallelism Tapped Out

- **Superscalar (SS) designs were the state of the art; many forms of parallelism not visible to programmer**
  - **multiple instruction issue**
  - **dynamic scheduling: hardware discovers parallelism between instructions**
  - **speculative execution: look past predicted branches**
  - **non-blocking caches: multiple outstanding memory ops**
- **Unfortunately, these sources have been used up**

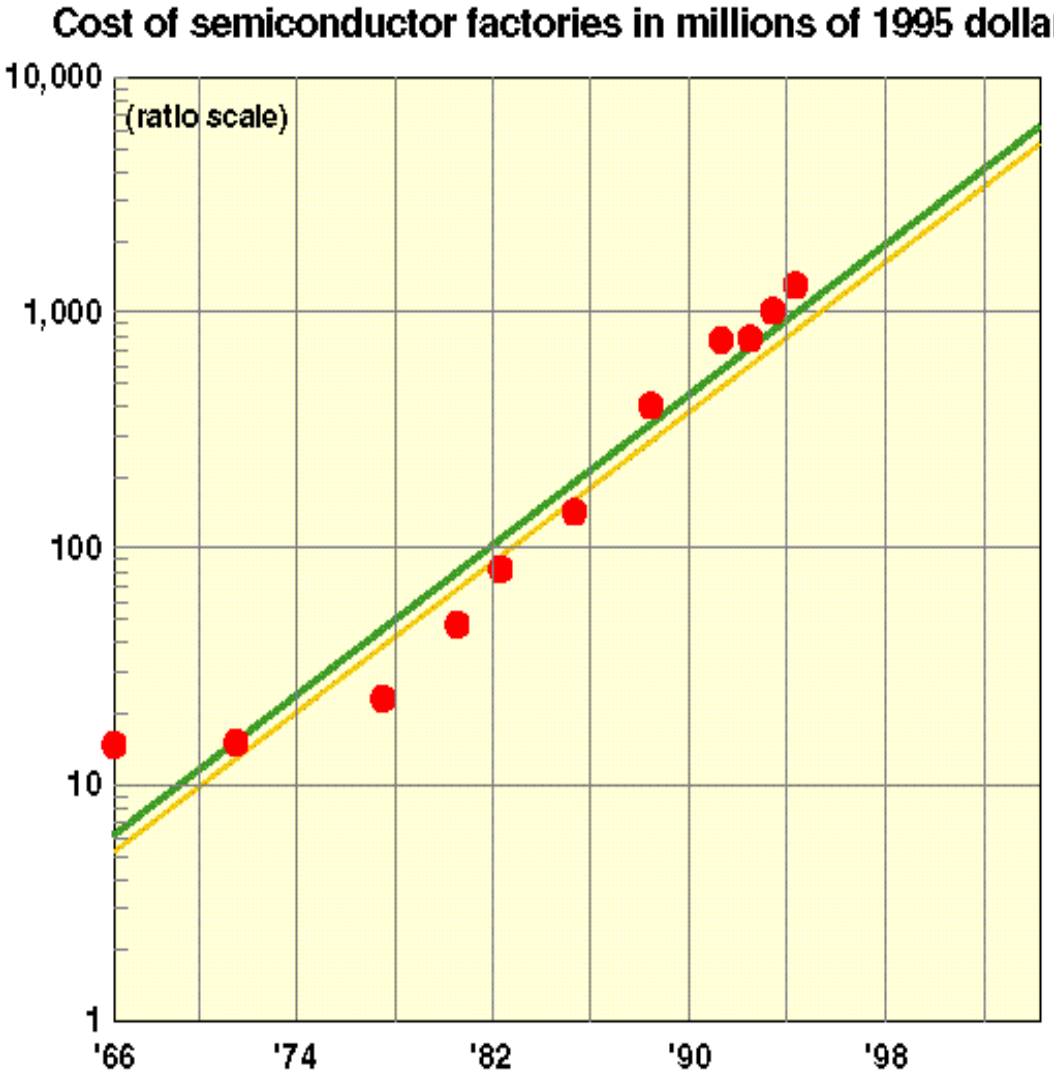
Uniprocessor Performance (SPECint) Today



- VAX : 25%/year 1978 to 1986
- RISC + x86: 52%/year 1986 to 2002
- RISC + x86: ??%/year 2002 to present

Limit #3: Chip Yield

Manufacturing costs and yield problems limit use of density

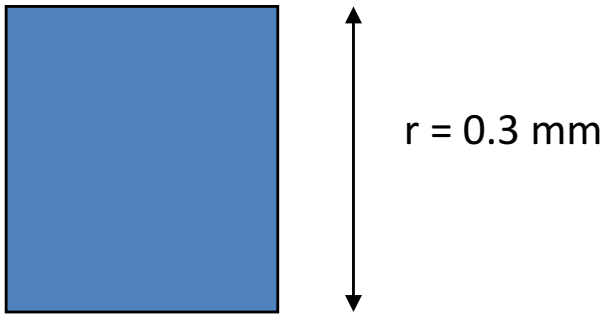


- Moore's (Rock's) 2nd law: fabrication costs go up
- Yield (% usable chips) drops

- Parallelism can help
  - More smaller, simpler processors are easier to design and validate
  - Can use partially working chips:
  - E.g., Cell processor (PS3) is sold with 7 out of 8 "on" to improve yield

Limit #4: Speed of Light (Fundamental)

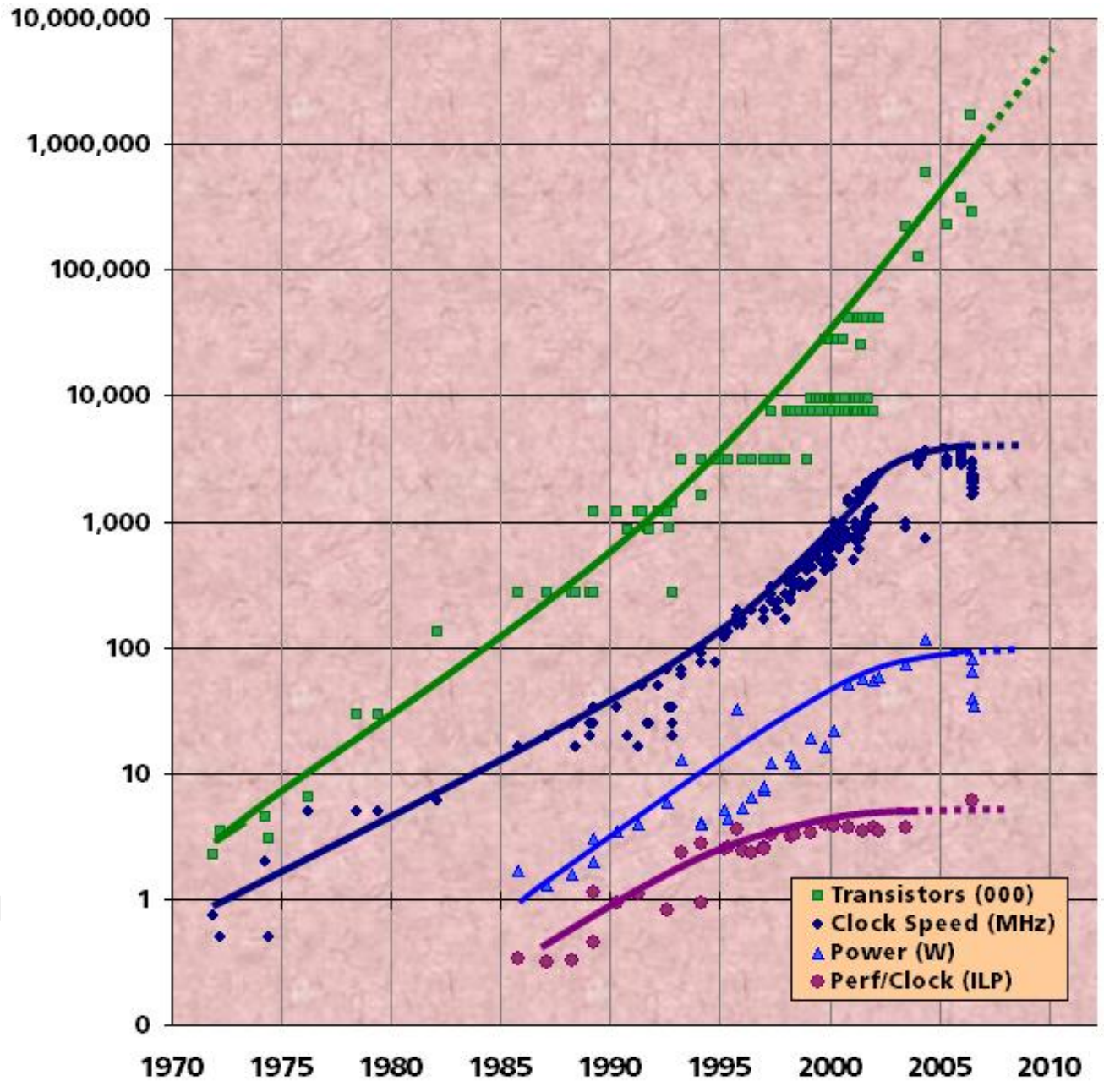
1 Tflop/s, 1 Tbyte sequential machine



- Consider the 1 Tflop/s sequential machine:
  - Data must travel some distance, r, to get from memory to CPU.
  - To get 1 data element per cycle, this means 10^{12} times per second at the speed of light, c = $3x10^8$ m/s. Thus r < $c/10^{12}$ = 0.3 mm.
- Now put 1 Tbyte of storage in a 0.3 mm x 0.3 mm area:
  - Each bit occupies about 1 square Angstrom, or the size of a small atom.
- No choice but parallelism

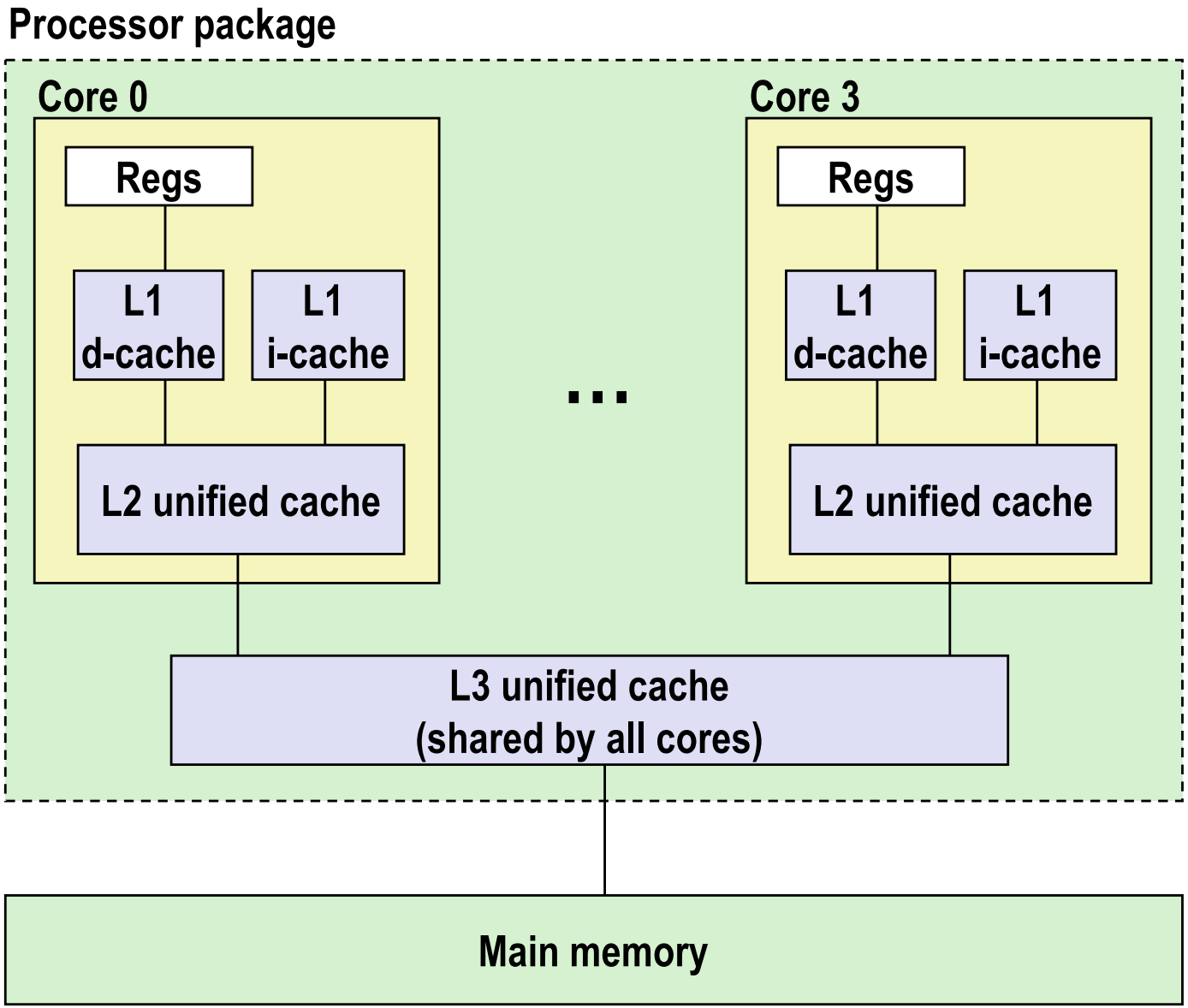
Thus, the Multicore Era

- 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



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

Intel Core i7 Cache Hierarchy



L1 i-cache and d-cache:
32 KB, 8-way,
Access: 4 cycles

L2 unified cache:
256 KB, 8-way,
Access: 11 cycles

L3 unified cache:
8 MB, 16-way,
Access: 30-40 cycles

Block size: 64 bytes for all caches.

What is Manycore ?

- What if we use all of the transisters on a chip for as many cores as we can fit??
- Beyond the edge of number of cores in common "multicore" architectures
- Dividing line is not clearly defined
- Active research, now in embedded & clusters
- Examples:
  - NVIDIA Fermi Graphics Processing Unit (GPU)
    - First model: 32 "CUDA cores" per SM, 16 SMs
      - (SM = "streaming multiprocessor")
    - K20 model: 2496 CUDA cores, peak 3.52 Tflops

Ex: NVIDIA Fermi

What is Manycore ?

- Examples (cont'd)
  - Intel Xeon Phi coprocessor and Knights Landing
    - Up to 61 cores each
  - Example: Tianhe-2 Supercomputer (China)
    - 32,000 multicore CPUs
    - 48,000 coprocessors ("accelerators")
    - peak 33.86 PetaFLOPS
  - Example: Summit Supercomputer (U.S.)
    - IBM Power9 processors
    - accelerated with NVIDIA Volta GPUs
    - Total # cores: 2,414,592

Acknowledgments

- This presentation includes materials and ideas developed by others:
  - 15-213: Introduction to Computer Systems, 2010
    - Randy Bryant and Dave O'Hallaron
  - Jack Dongarra, University of Tennessee
  - Kathy Yelick, UC-Berkeley
  - Andrew S. Tanenbaum, *Modern Operating Systems*
  - Remzi and Andrea C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*