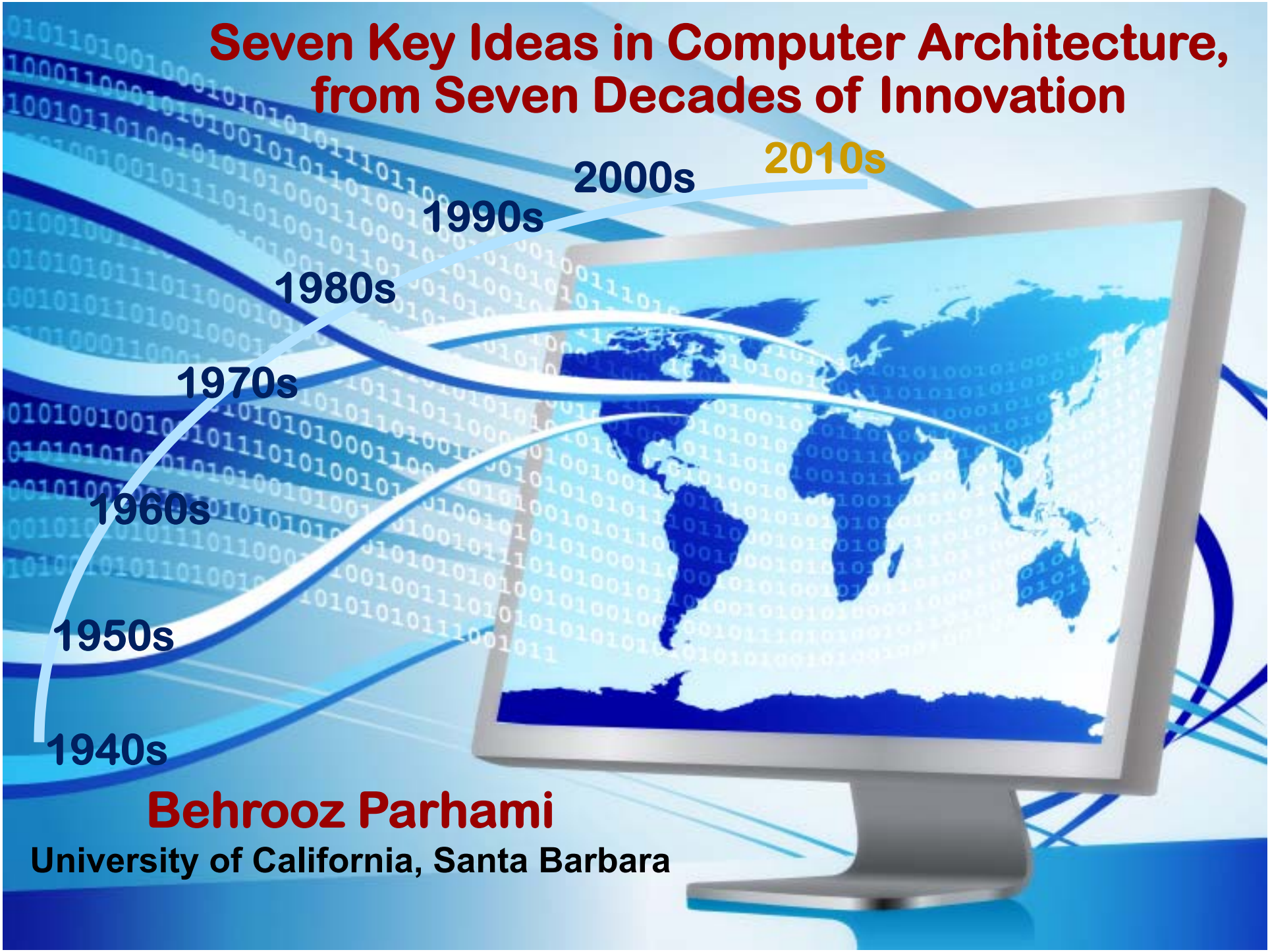
Seven Key Ideas in Computer Architecture, from Seven Decades of Innovation



Behrooz Parhami

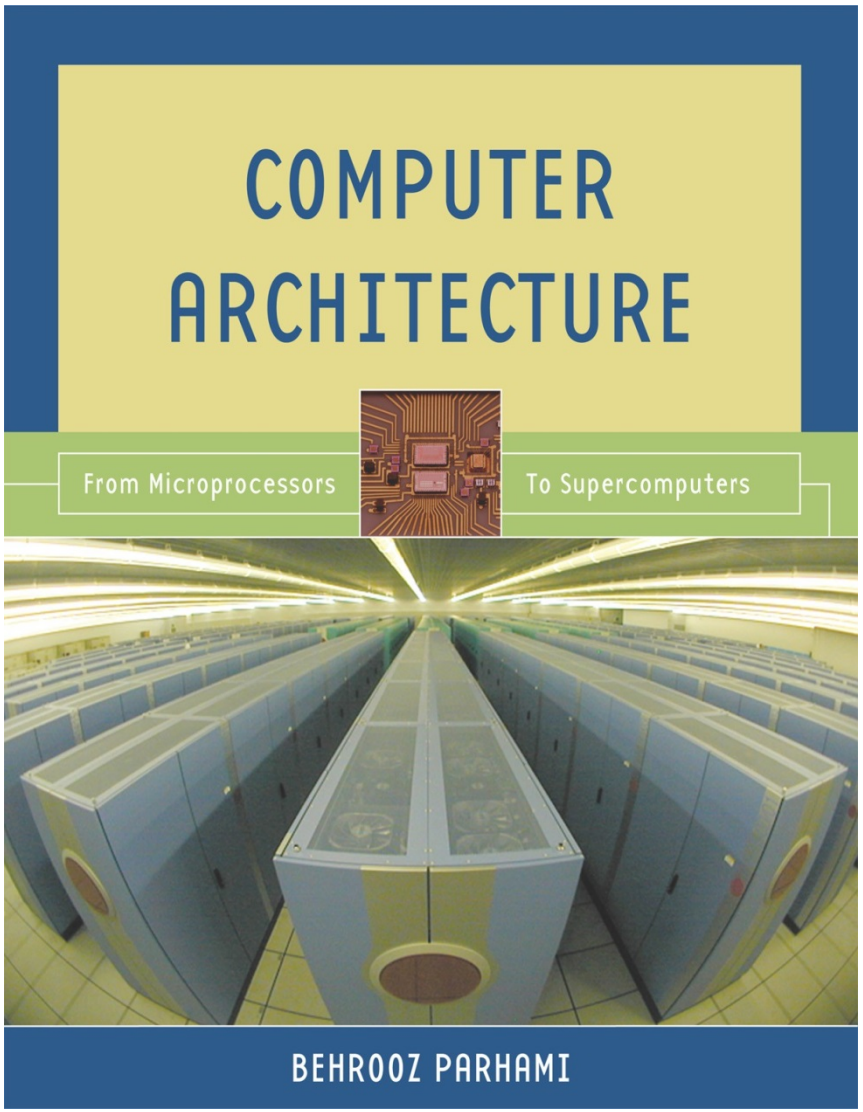
University of California, Santa Barbara

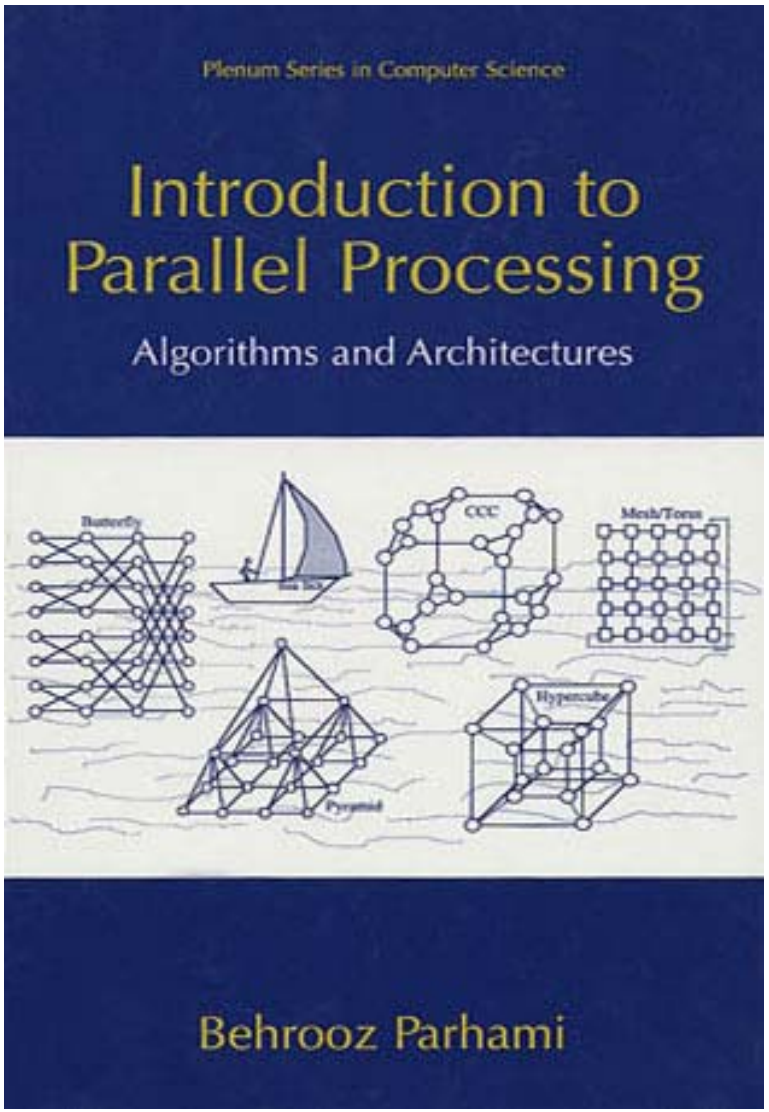
About This Presentation

This slide show was first developed as a keynote talk for remote delivery at CSICC-2016, Computer Society of Iran Computer Conference, held in Tehran on March 8-10. The talk was presented at a special session on March 9, 11:30 AM to 12:30 PM Tehran time (12:00-1:00 AM PST).

Edition	Released	Revised	Revised	Revised
First	March 2016			

Some of the material in this talk come from, or will appear in updated versions of, my two computer architecture textbooks





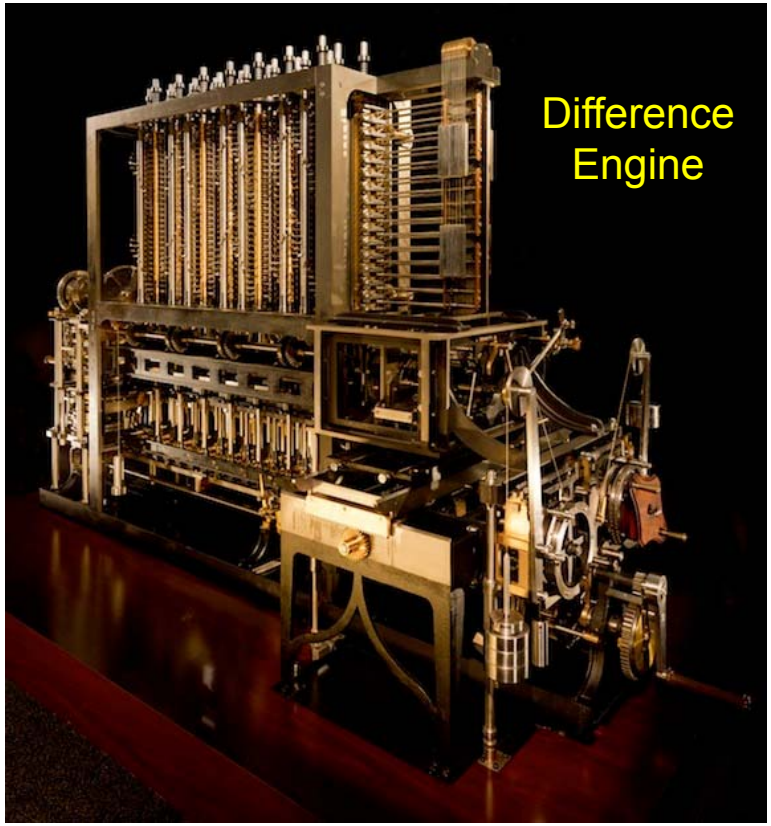
Seven Key Ideas in Computer Architecture, from Seven Decades of Innovation

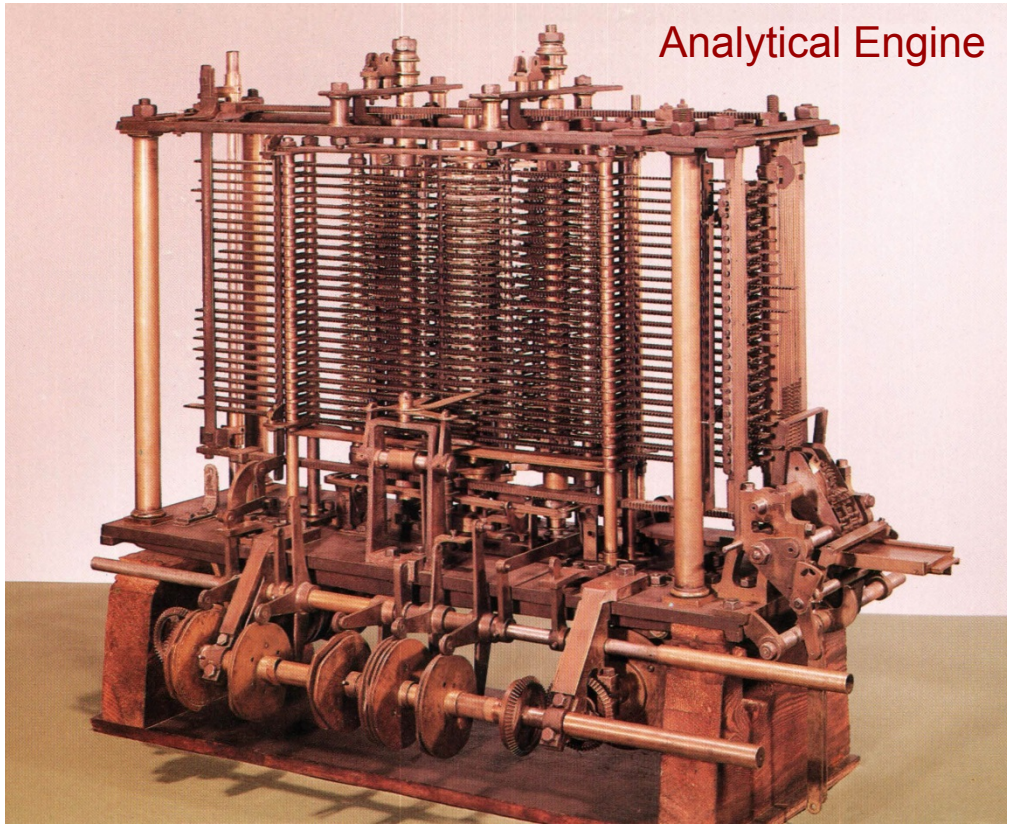
Computer architecture became an established discipline when the stored-program concept was incorporated into bare-bones computers of the 1940s. Since then, the field has seen multiple minor and major innovations in each decade. I will present my pick of the most important innovation in each of the seven decades, from the 1940s to the 2000s, and show how these ideas, when connected to each other and allowed to interact and cross-fertilize, led to the phenomenal growth of computer performance, now approaching exa-op/s (billion billion operations per second) level, as well as to ultra-low-energy and single-chip systems. I will also highlight developments in the current decade which are candidates for the most important idea of the 2010s.

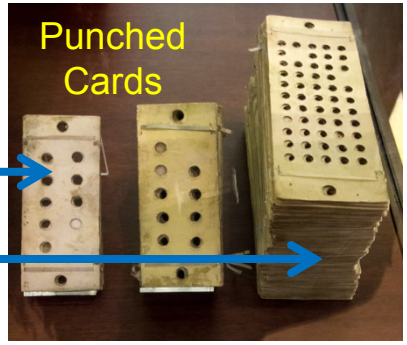
هفت مفهومِ کلیدی در معماریِ کامپیوتر، حاصلِ هفت دهه نوآوری

رشته ی معماریِ کامپیوتر با پذیرش و کاربردِ مفهومِ برنامه ی ذخیره شده در دهه ی ۱۹۴۰ پایه گذاری شد. از آن هنگام تا کنون، هر دهه شاملِ چند نو آوریِ کوچک و بزرگ بوده است. در این سخنرانی، من نو آوریِ برگزیده یِ خود را برایِ هر یک از هفت دهه یِ ۲۰۰۰-۱۹۴۰ مطرح می‌کنم و نشان می دهم که چگونه روابط و کنش‌ها بین این مفاهیم ما را به مرز کار آیی در حدِ میلیارد میلیارد عملِ محاسباتی در ثانیه رسانده و ساختِ سیستم‌هایِ فوقِ کم انرژی و تک تراشه را ممکن ساخته است. در پایان، چند نو آوریِ دهه یِ جاری را، که می‌توانند نامزدِ انتخاب به عنوانِ مفهومِ برگزیده برایِ دهه ی ۲۰۱۰ باشند، بر می‌شمارم.

Background: 1820s-1930s



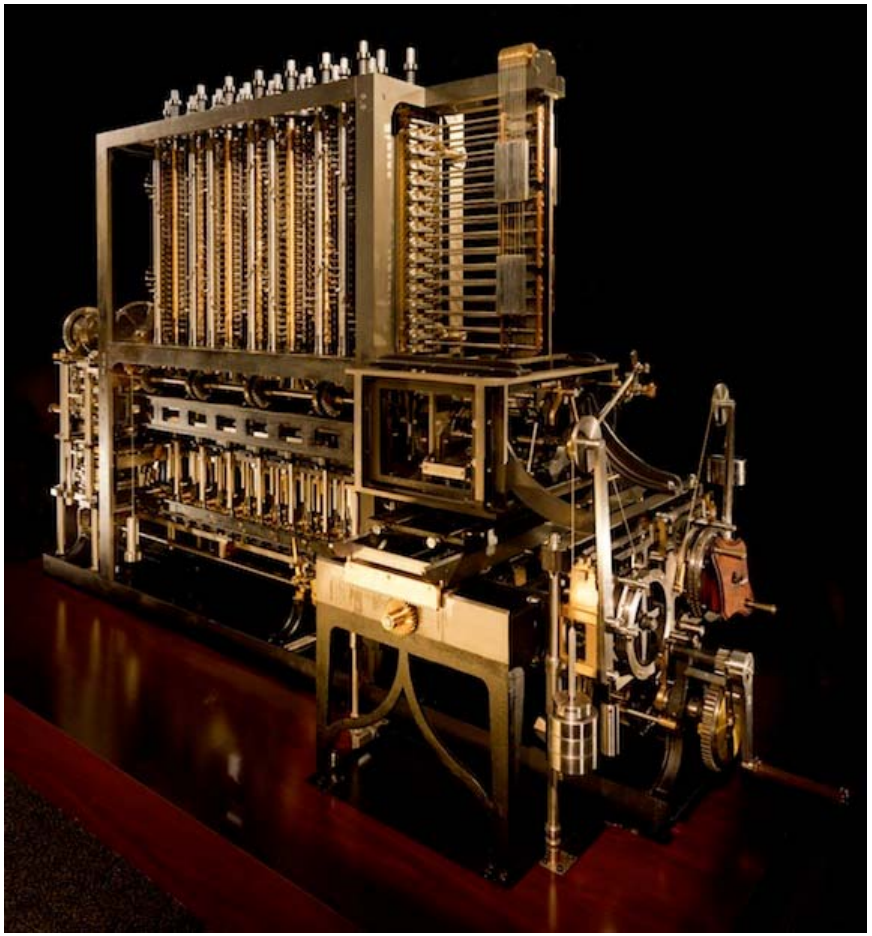




Program (Instructions)

Data (Variable values)

Difference Engine: Fixed Program

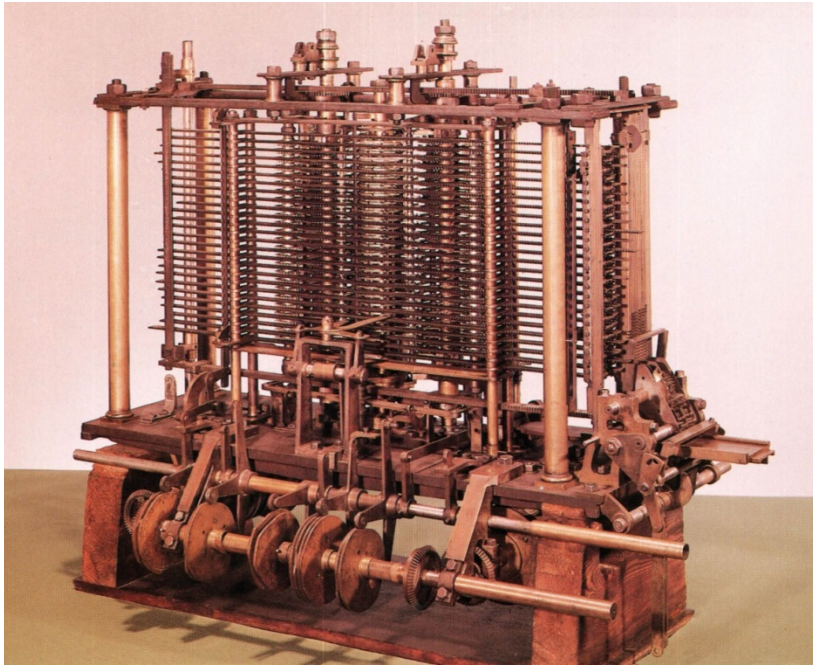


Babbage's Difference Engine 2

$D^{(2)}$	$D^{(1)}$	$f(x)$ x^2+x+41	x
		41	0
	2		
2		43	1
	4		
2		47	2
	6		
2		53	3
	8		
2		61	4
	10		
2		71	5

2nd-degree polynomial evaluation
Babbage used 7th-degree $f(x)$

Analytical Engine: Programmable



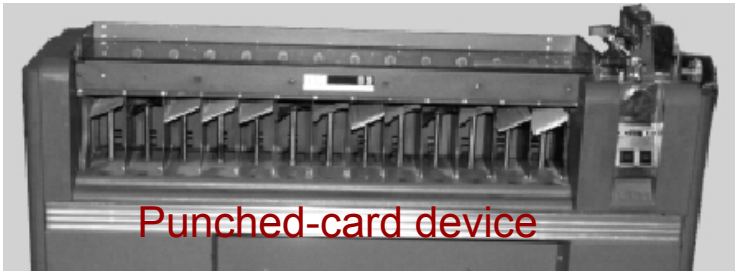
Ada Lovelace,
world's first programmer

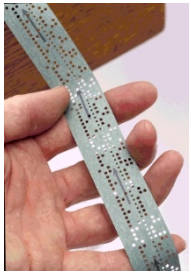
Sample program >

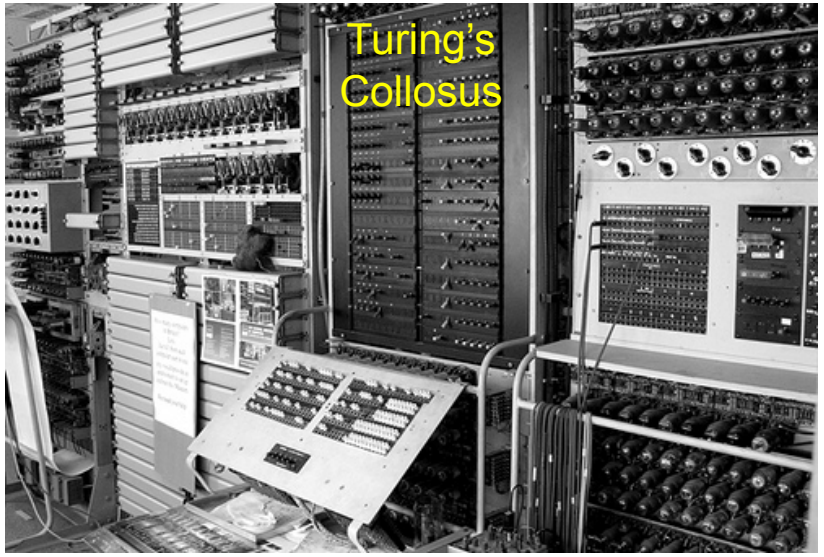
Number of Operation	Nature of Operation	Variables acted upon	Variables receiving results	Indication of change in the value on any Variable	Statement of Results	Data $^1V_3$ 0001	Data $^1V_4$ 0002	Data $^1V_5$ 0004	Data $^0V_4$ 0000	Data $^0V_5$ 0000
						1	2	n		
1	×	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$\{{}^1V_2 = {}^1V_2$ $\{{}^1V_3 = {}^1V_3$	$= 2n$	...	2	n	$2n$	$2n$
2	−	$^1V_4 - {}^1V_1$	$^2V_4$	$\{{}^1V_4 = {}^2V_4$ $\{{}^1V_1 = {}^1V_1$	$= 2n-1$	1	...	...	$2n-1$	
3	+	$^1V_5 + {}^1V_1$	$^2V_5$	$\{{}^1V_5 = {}^2V_5$ $\{{}^1V_1 = {}^1V_1$	$= 2n+1$	1	...	...	...	$2n+1$
4	÷	$^2V_5 \div {}^2V_4$	$^1V_{12}$	$\{{}^2V_5 = {}^0V_5$ $\{{}^2V_4 = {}^0V_4$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0
5	÷	$^1V_{11} \div {}^1V_3$	$^2V_{11}$	$\{{}^1V_{11} = {}^2V_{11}$ $\{{}^1V_3 = {}^1V_3$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...
6	−	$^0V_{13} - {}^2V_{11}$	$^1V_{13}$	$\{{}^2V_{11} = {}^0V_{11}$ $\{{}^0V_{13} = {}^1V_{13}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...
7	−	$^1V_2 - {}^1V_1$	$^1V_{10}$	$\{{}^1V_2 = {}^1V_2$ $\{{}^1V_1 = {}^1V_1$	$= n-1 (=3)$	1	...	n	...	...
8	+	$^1V_1 + {}^0V_7$	$^1V_7$	$\{{}^1V_1 = {}^1V_1$ $\{{}^0V_7 = {}^1V_7$	$= 2+0 = 2$	...	2	...	...	...
9	÷	$^1V_6 \div {}^1V_7$	$^2V_{13}$	$\{{}^1V_6 = {}^1V_6$ $\{{}^0V_{13} = {}^2V_{13}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...
10	×	$^2V_{13} \times {}^2V_{11}$	$^1V_{14}$	$\{{}^1V_{13} = {}^1V_{13}$ $\{{}^2V_{11} = {}^2V_{11}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...
11	+	$^1V_{12} + {}^1V_{13}$	$^2V_{12}$	$\{{}^1V_{12} = {}^0V_{12}$ $\{{}^1V_{13} = {}^2V_{13}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$	...	...	...	...	...
12	−	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$\{{}^1V_{10} = {}^2V_{10}$ $\{{}^1V_1 = {}^1V_1$	$= n-2 (=2)$	1	...	...	...	...
13	−	$^1V_4 - {}^1V_1$	$^2V_6$	$\{{}^1V_4 = {}^2V_4$ $\{{}^1V_1 = {}^1V_1$	$= 2n-1$	1	...	...	...	...
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$\{{}^1V_1 = {}^1V_1$ $\{{}^1V_7 = {}^2V_7$	$= 2+1 = 3$	1	...	...	...	...
15	÷	$^2V_6 \div {}^2V_7$	$^2V_8$	$\{{}^2V_6 = {}^2V_6$ $\{{}^2V_7 = {}^2V_7$	$= \frac{2n-1}{3}$	...	...	...	...	...
16	×	$^1V_8 \times {}^2V_{11}$	$^4V_{11}$	$\{{}^1V_8 = {}^0V_8$ $\{{}^2V_{13} = {}^4V_{13}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...
17	−	$^2V_6 - {}^1V_1$	$^2V_6$	$\{{}^2V_6 = {}^2V_6$ $\{{}^1V_1 = {}^1V_1$	$= 2n-2$	1	...	...	...	...
18	+	$^1V_1 + {}^2V_7$	$^2V_7$	$\{{}^2V_7 = {}^2V_7$ $\{{}^1V_1 = {}^1V_1$	$= 3+1 = 4$	1	...	...	...	...
19	÷	$^2V_6 \div {}^2V_7$	$^1V_9$	$\{{}^2V_6 = {}^2V_6$ $\{{}^2V_7 = {}^2V_7$	$= \frac{2n-2}{4}$	...	...	...	...	...
20	×	$^1V_9 \times {}^2V_{11}$	$^4V_{11}$	$\{{}^1V_9 = {}^0V_9$ $\{{}^4V_{11} = {}^6V_{11}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_2$	...	...	...	...	...
21	×	$^2V_{13} \times {}^2V_{11}$	$^0V_{13}$	$\{{}^2V_{13} = {}^1V_{13}$ $\{{}^2V_{11} = {}^2V_{11}$	$= B_2 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_2 A_2$	...	...	...	...	...
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$\{{}^2V_{12} = {}^0V_{12}$ $\{{}^2V_{13} = {}^3V_{13}$	$= A_0 + B_1 A_1 + B_2 A_2$	...	...	...	...	...
23	−	$^1V_{10} - {}^1V_1$	$^3V_{10}$	$\{{}^1V_{10} = {}^2V_{10}$ $\{{}^1V_1 = {}^1V_1$	$= n-3 (=1)$	1	...	...	...	...
24	+	$^4V_{12} \div {}^4V_{24}$	$^2V_{24}$	$\{{}^4V_{12} = {}^0V_{12}$ $\{{}^4V_{24} = {}^1V_{24}$	$= B_7$	...	...	...	...	...
25	+	$^1V_1 + {}^1V_2$	$^1V_2$	$\{{}^1V_1 = {}^2V_1$ $\{{}^1V_2 = {}^1V_2$ $\{{}^0V_4 = {}^0V_4$ $\{{}^0V_7 = {}^0V_7$	$= n+1 = 4+1 = 5$ ..... by a Variable-card. by a Variable-card.	1	...	$n+1$	...	...

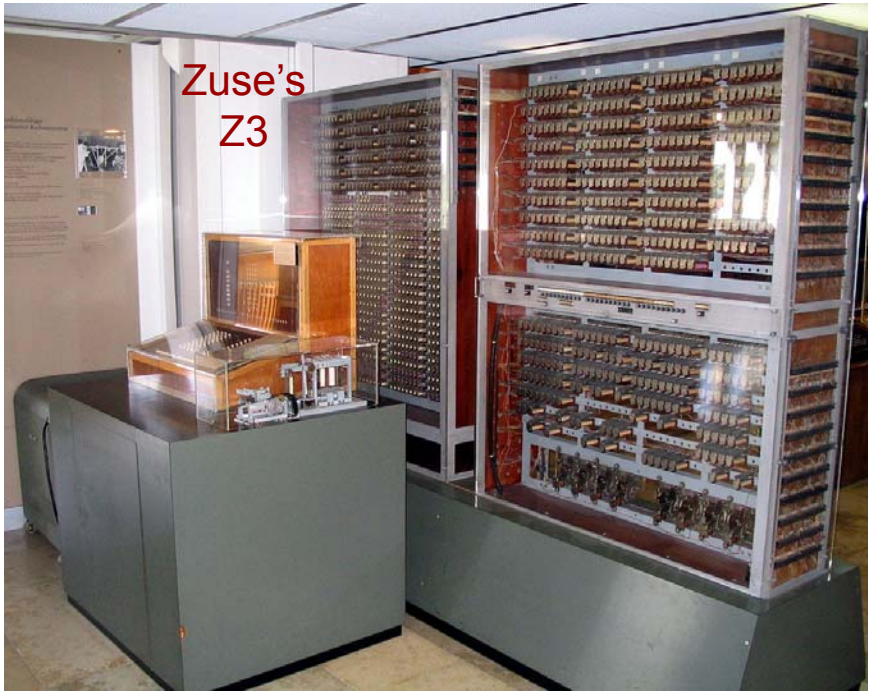
Here follows a repetition

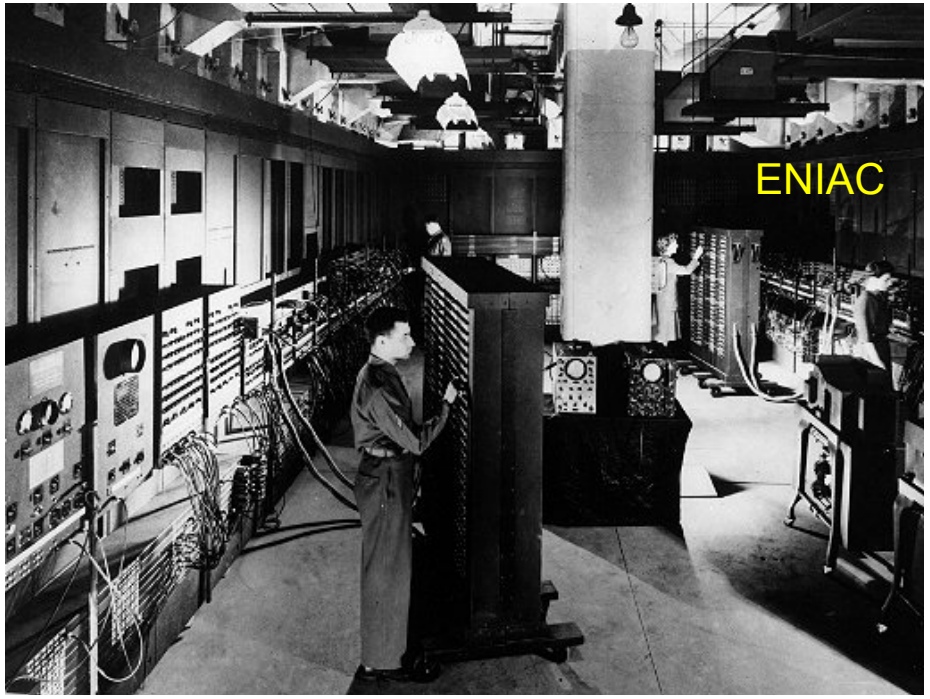
Electromechanical and Plug-Programmable Computing Machines



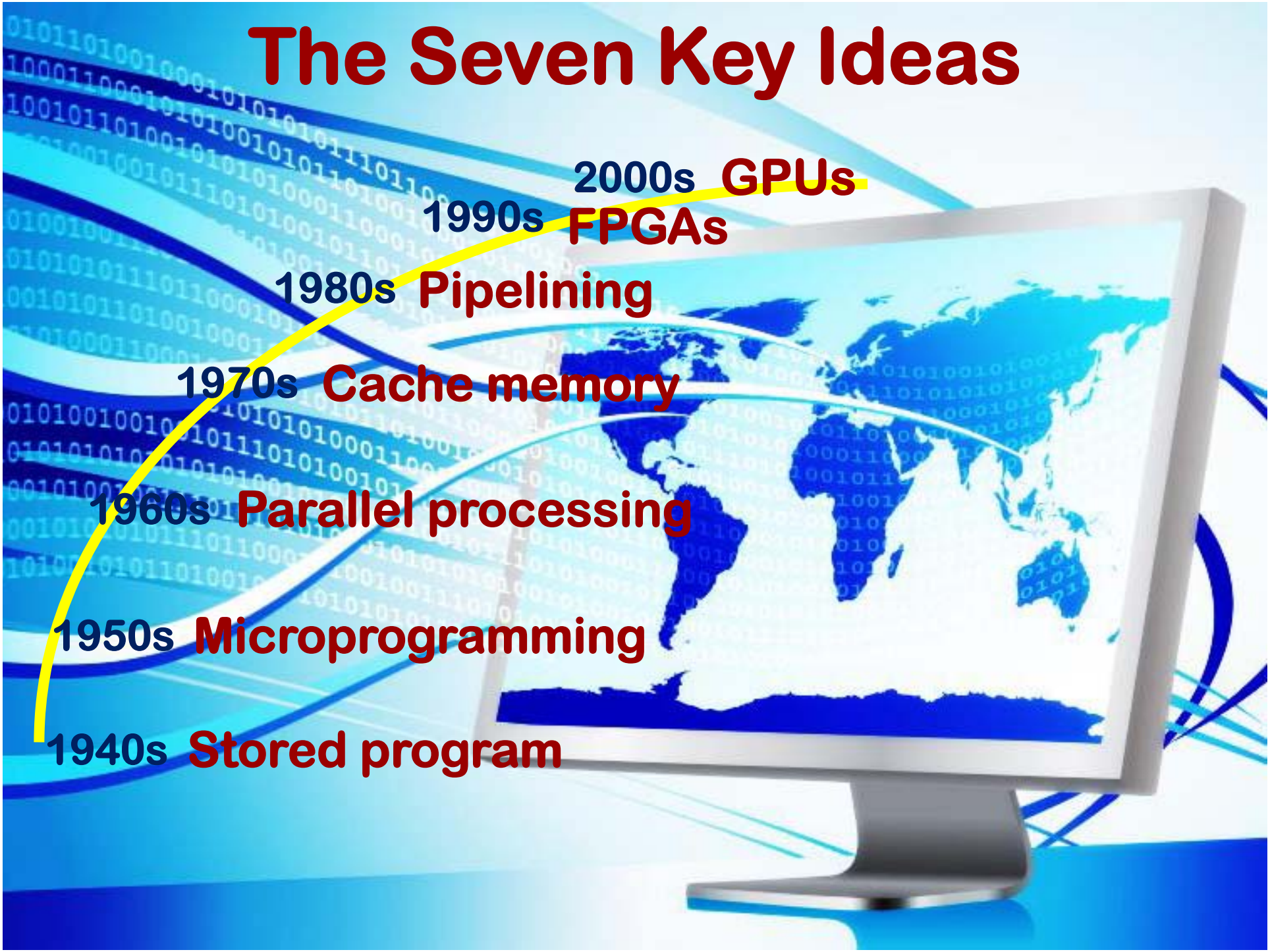








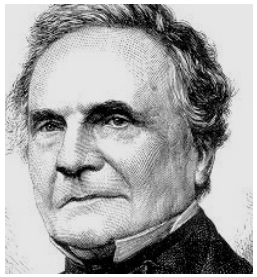
The Seven Key Ideas

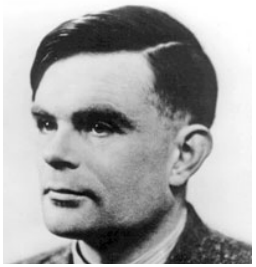


1940s: Stored Program

Exactly who came up with the stored-program concept is unclear

Legally, John Vincent Atanasoff is designated as inventor, but many others deserve to share the credit


Babbage

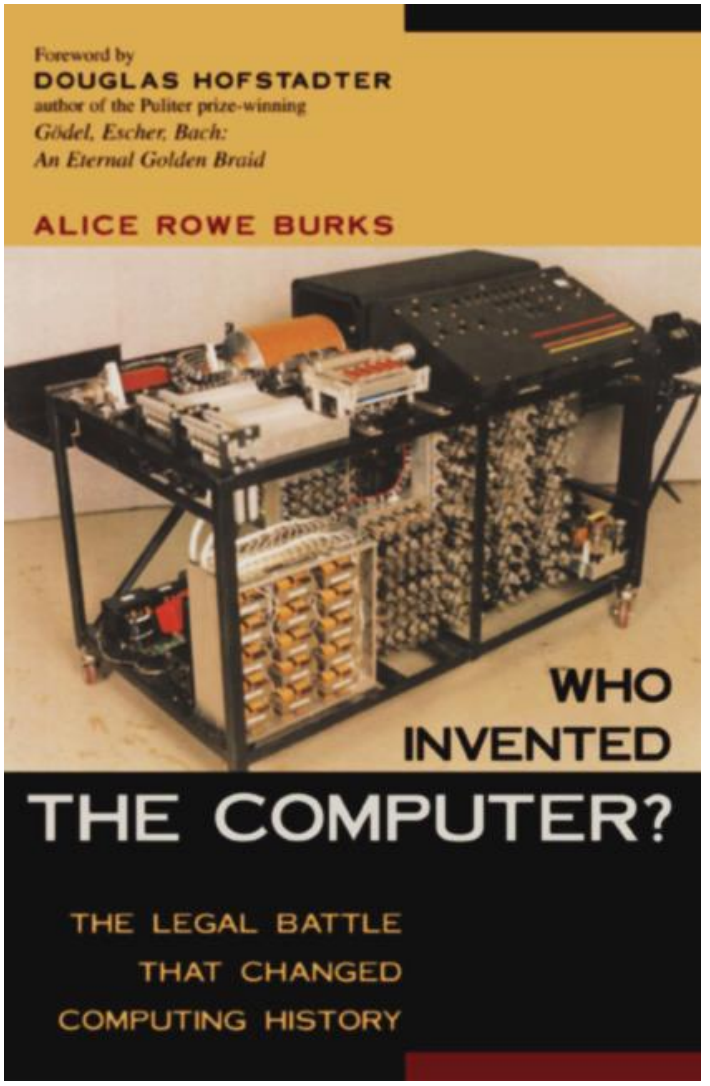

Turing


Atanasoff


Eckert


Mauchly


von Neumann



First Stored-Program Computer

Manchester Small-Scale Experimental Machine
Ran a stored program on June 21, 1948
(Its successor, Manchester Mark 1, operational in April 1949)

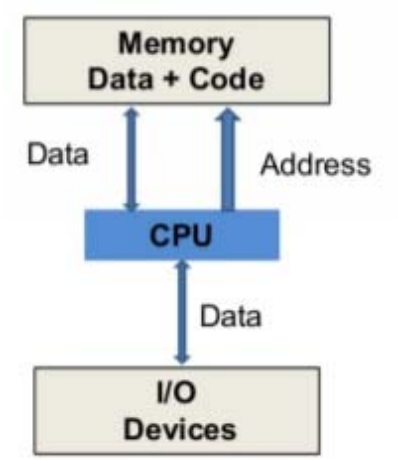
EDSAC (Cambridge University; Wilkes et al.)
Fully operational on May 6, 1949

EDVAC (IAS, Princeton University; von Neumann et al.)
Conceived in 1945 but not delivered until August 1949

BINAC (Binary Automatic Computer, Eckert & Mauchly)
Delivered on August 22, 1949, but did not function correctly

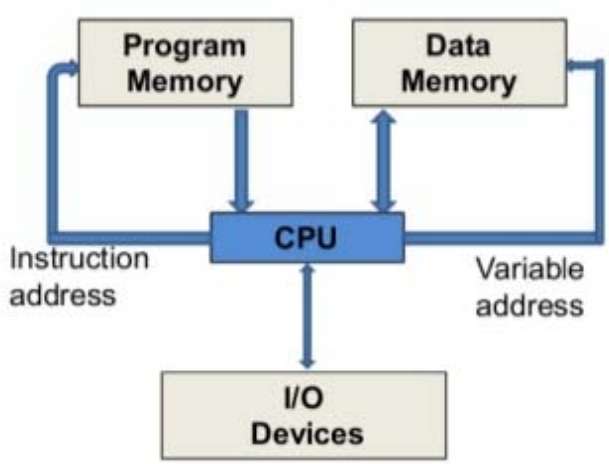
Source: Wikipedia

von Neumann vs. Harvard Architecture



von Neumann architecture
(unified memory for code & data)

Programs can be modified like data
More efficient use of memory space

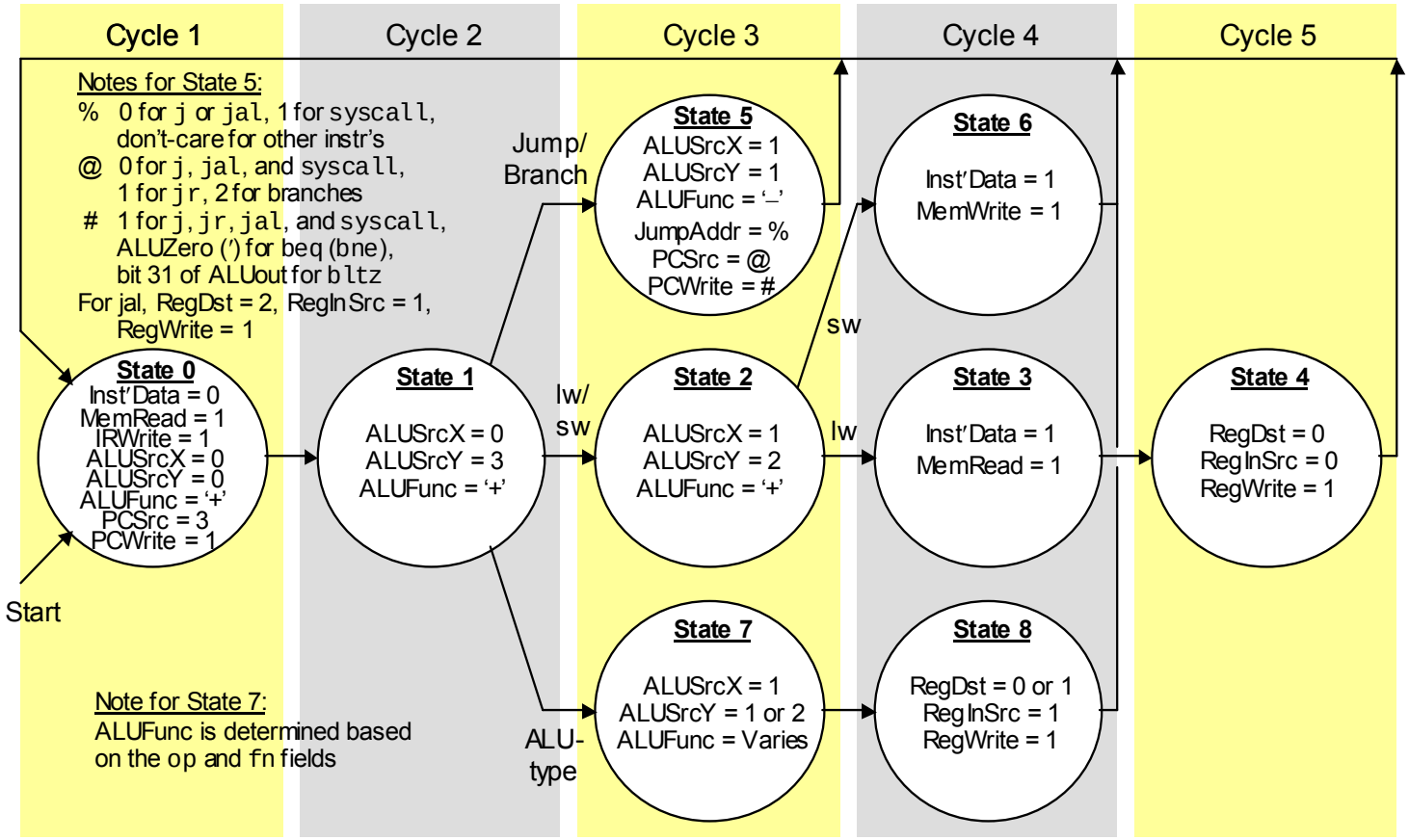


Harvard architecture
(separate memories for code & data)

Better protection of programs
Higher aggregate memory bandwidth
Memory optimization for access type

1950s: Microprogramming

Traditional control unit design (multicycle): Specify which control signals are to be asserted in each cycle and synthesize



Gives rise to random logic

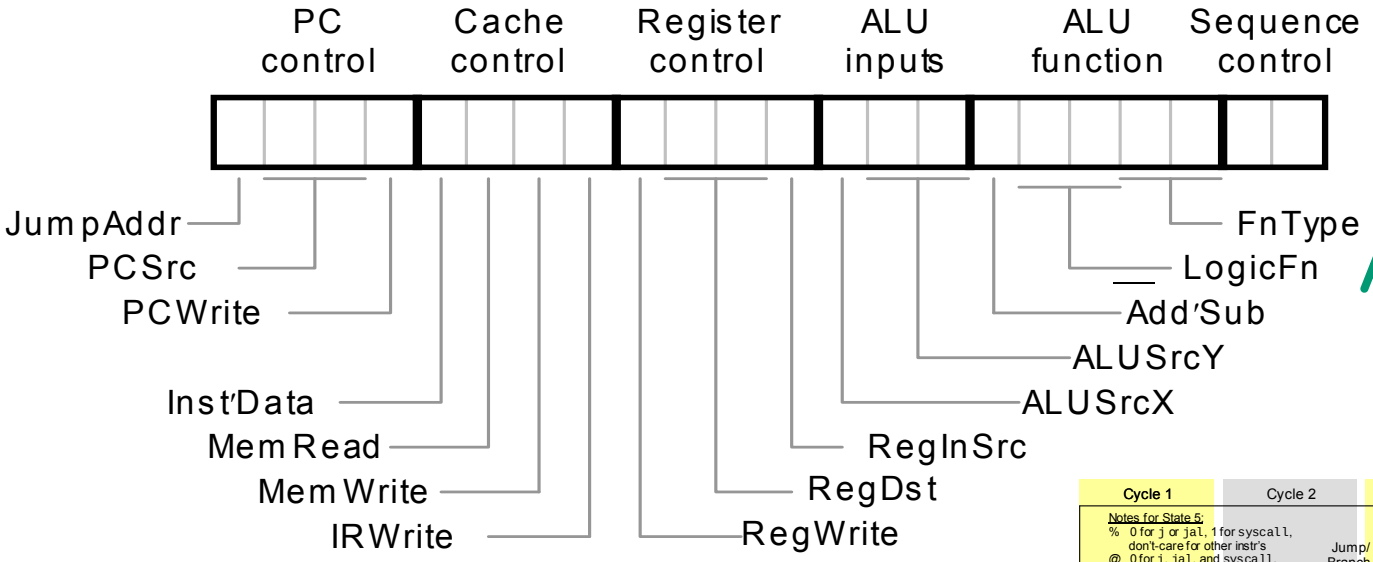
Error-prone and inflexible

Hardware bugs hard to fix after deployment

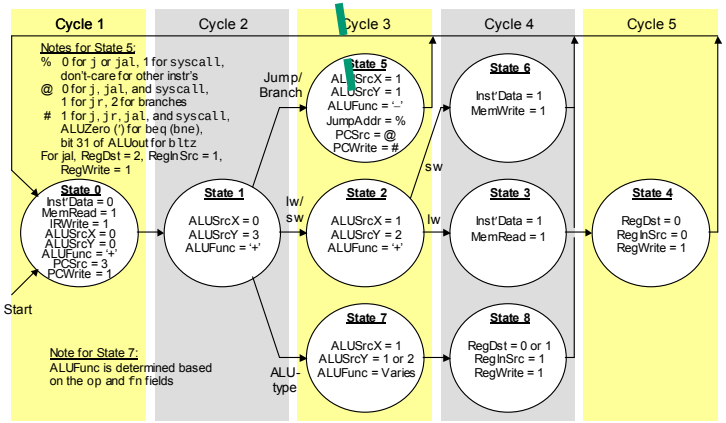
Design from scratch for each system

The Birth of Microprogramming

The control state machine resembles a program (microprogram) comprised of instructions (microinstructions) and sequencing



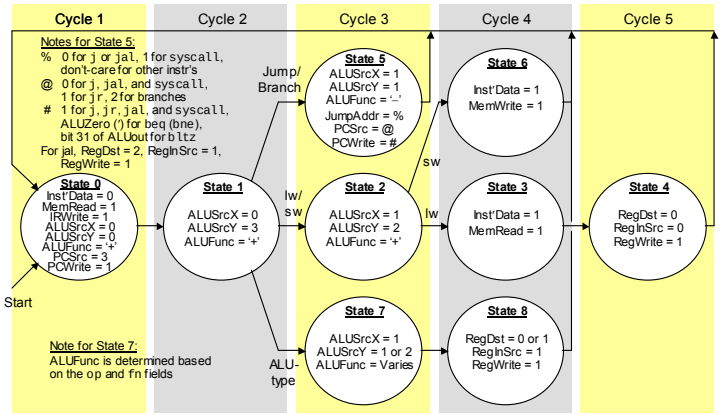
Every μinstruction contains a branch field

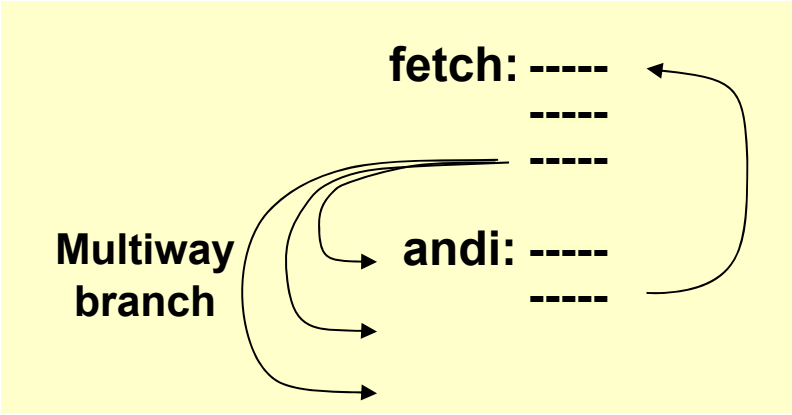


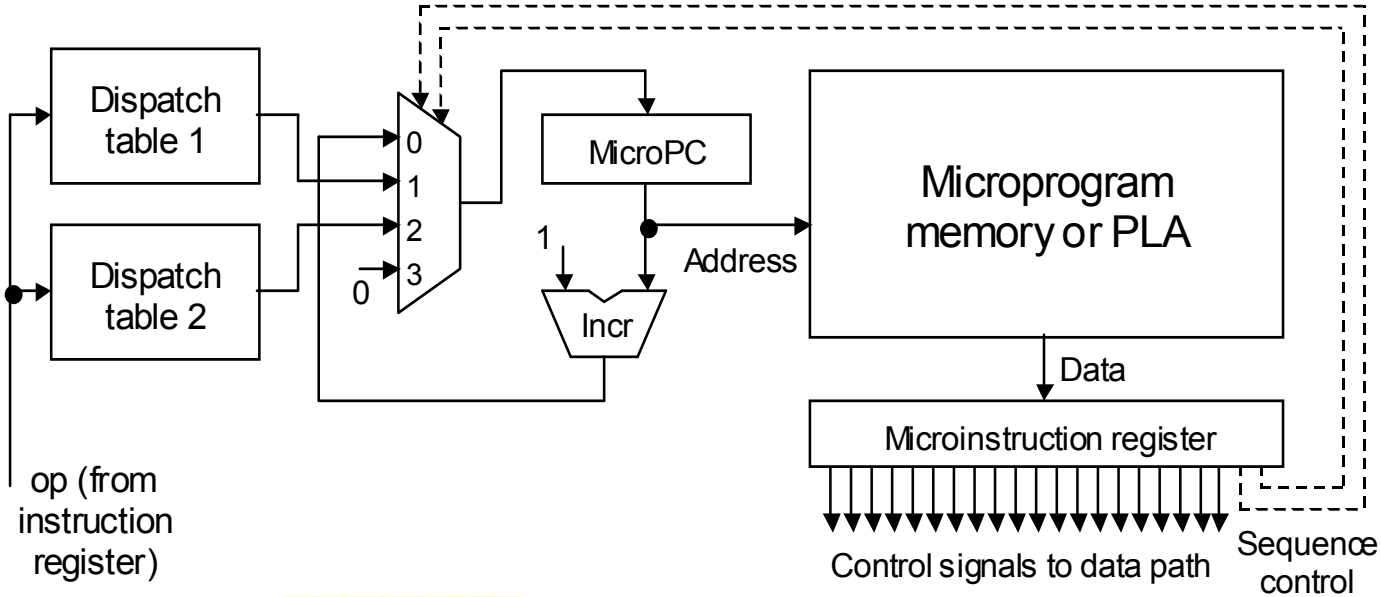
Maurice V. Wilkes
(1913-2010)

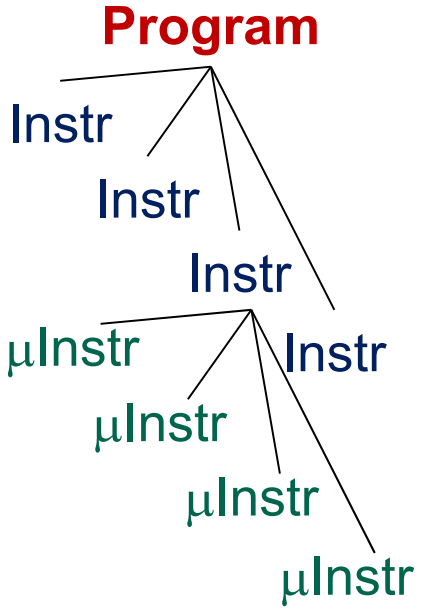
Microprogramming Implementation

Each microinstruction controls the data path for one clock cycle



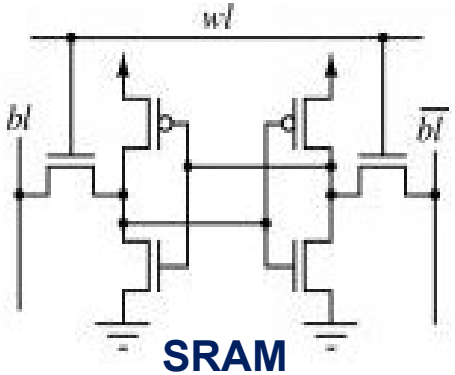




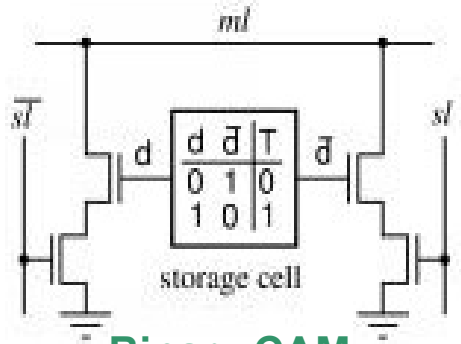


1960s: Parallel Processing

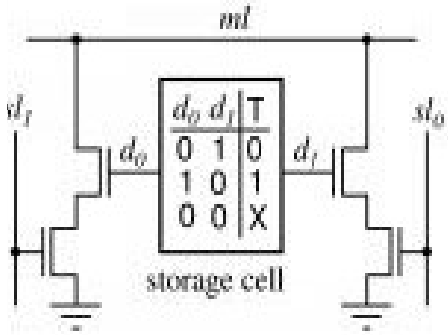
Associative (content-addressed) memories and other forms of parallelism (compute-I/O overlap, functional parallelism) had been in existence since the 1940s



SRAM



Binary CAM



Ternary CAM

Highly parallel machine, proposed by Daniel Slotnick in 1964, later morphed into ILLIAC IV in 1968 (operational in 1975)

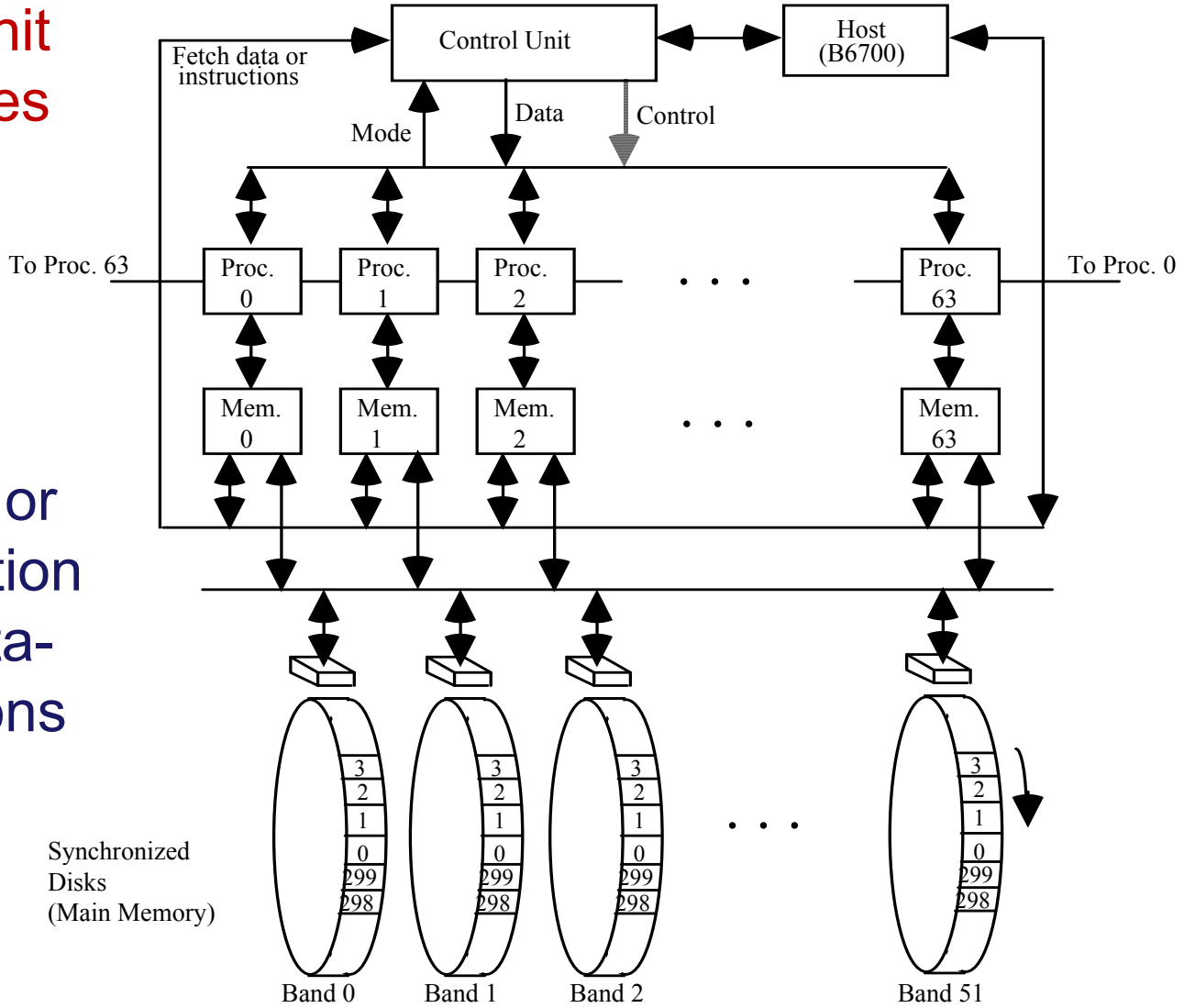
Michael J. Flynn devised his now-famous 4-way taxonomy (SISD, SIMD, MISD, MIMD) in 1966 and Amdahl formulated his speed-up law and rules for system balance in 1967

The ILLIAC IV Concept: SIMD Parallelism

Common control unit fetches and decodes instructions, broadcasting the control signals to all PEs

Each PE executes or ignores the instruction based on local, data-dependent conditions

The interprocessor routing network is only partially shown



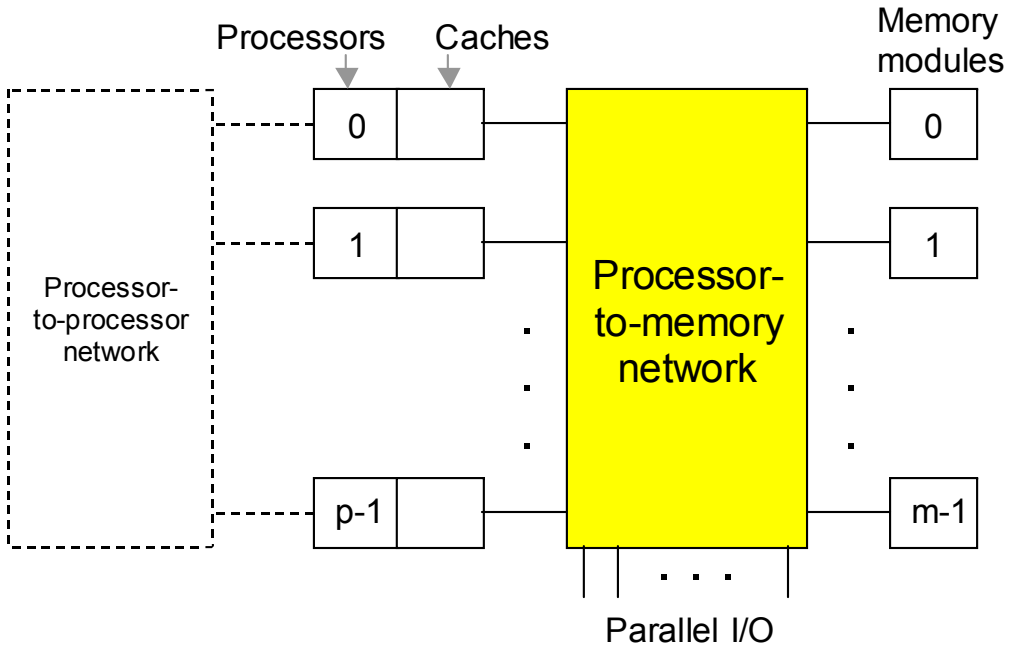
Various Forms of MIMD Parallelism

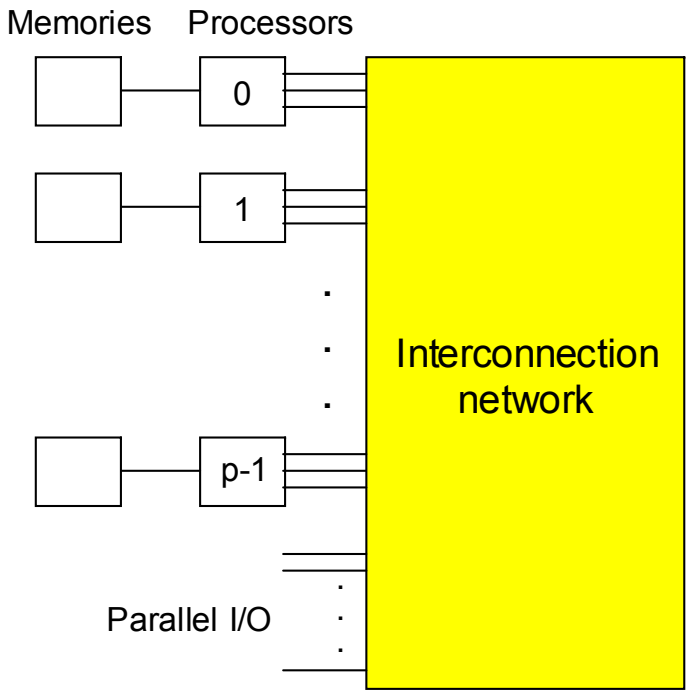
Global shared memory

- Memory latency
- Memory bandwidth

} **Wall**

- Cache coherence





Distributed shared memory or message-passing architecture

- Scalable network performance
- Flexible and more robust
- Memory consistency model

Warehouse-Sized Data Centers

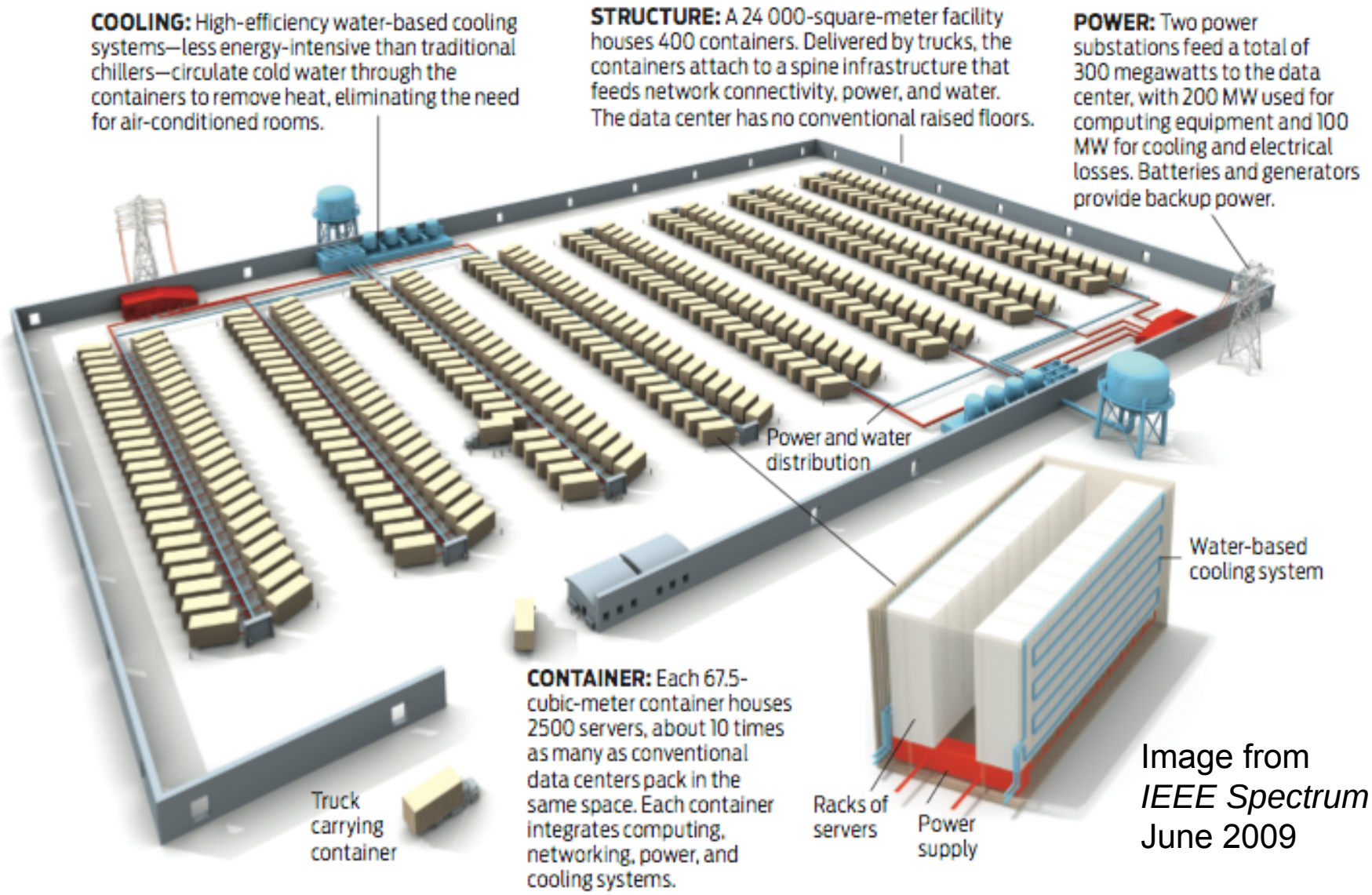
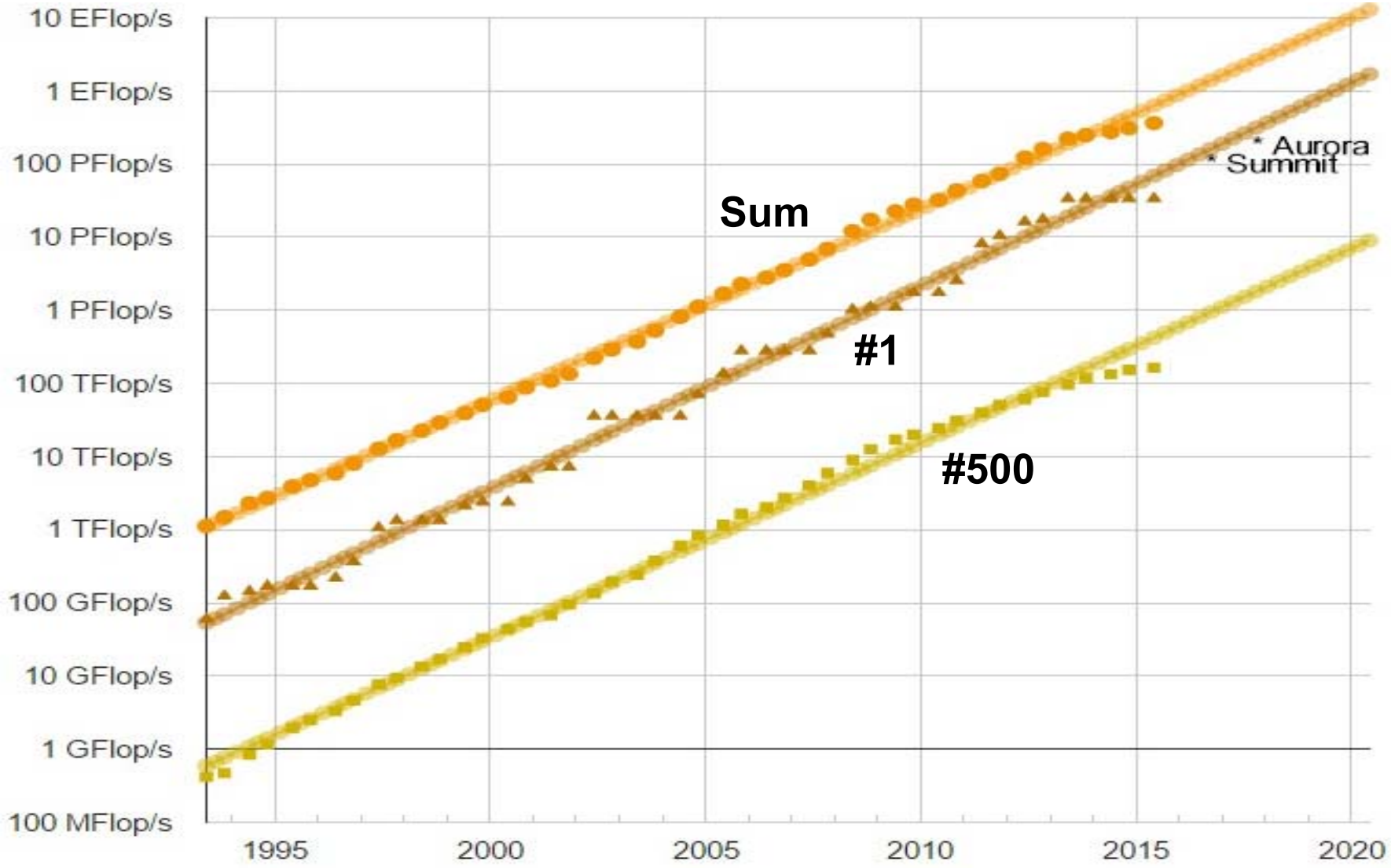
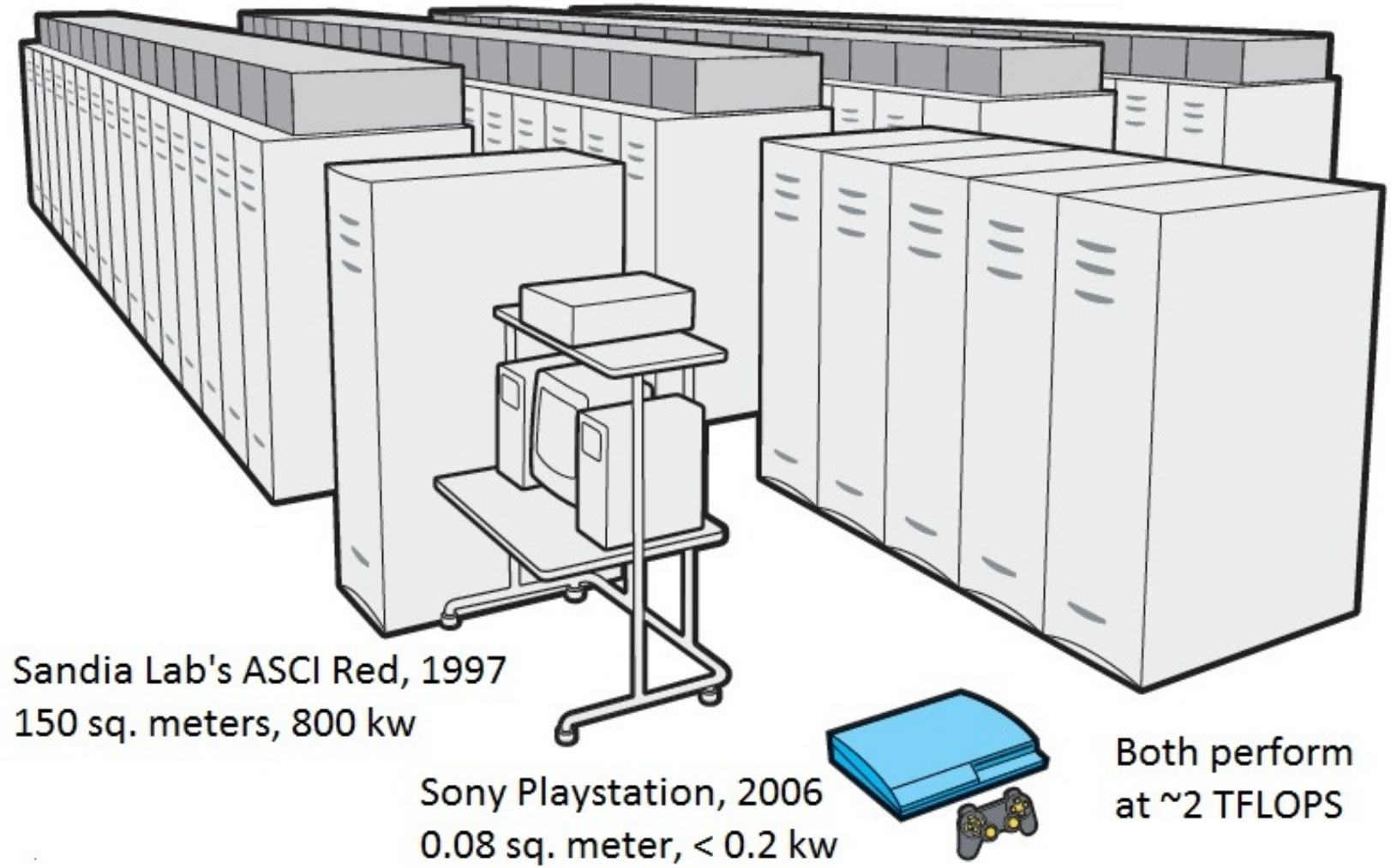


Image from *IEEE Spectrum*, June 2009

Top 500 Supercomputers in the World



The Shrinking Supercomputer



1970s: Cache Memory

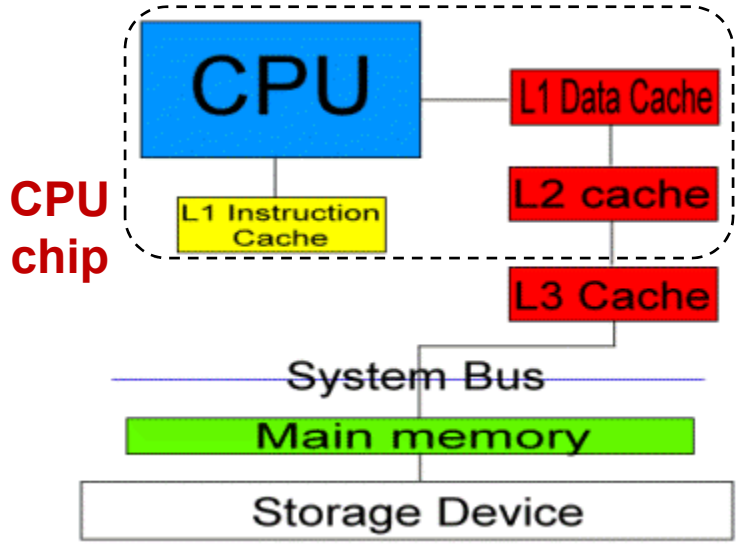
First paper on “buffer” memory: Maurice Wilkes, 1965

First implementation of a general cache memory:
IBM 360 Model 85 (J. S. Liptay; *IBM Systems J.*, 1968)

Broad understanding, varied implementations, and studies of optimization and performance issues in the 1970s

Modern cache implementations

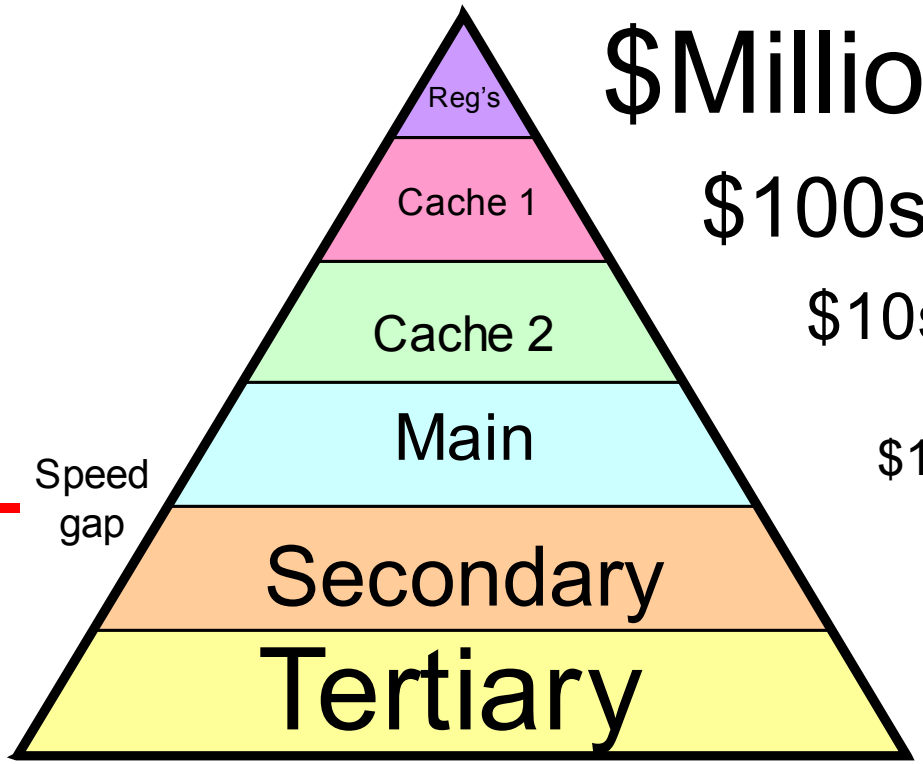
- Harvard arch for L1 cashes
- von Neumann arch higher up
- Many other caches in system besides processor cashes



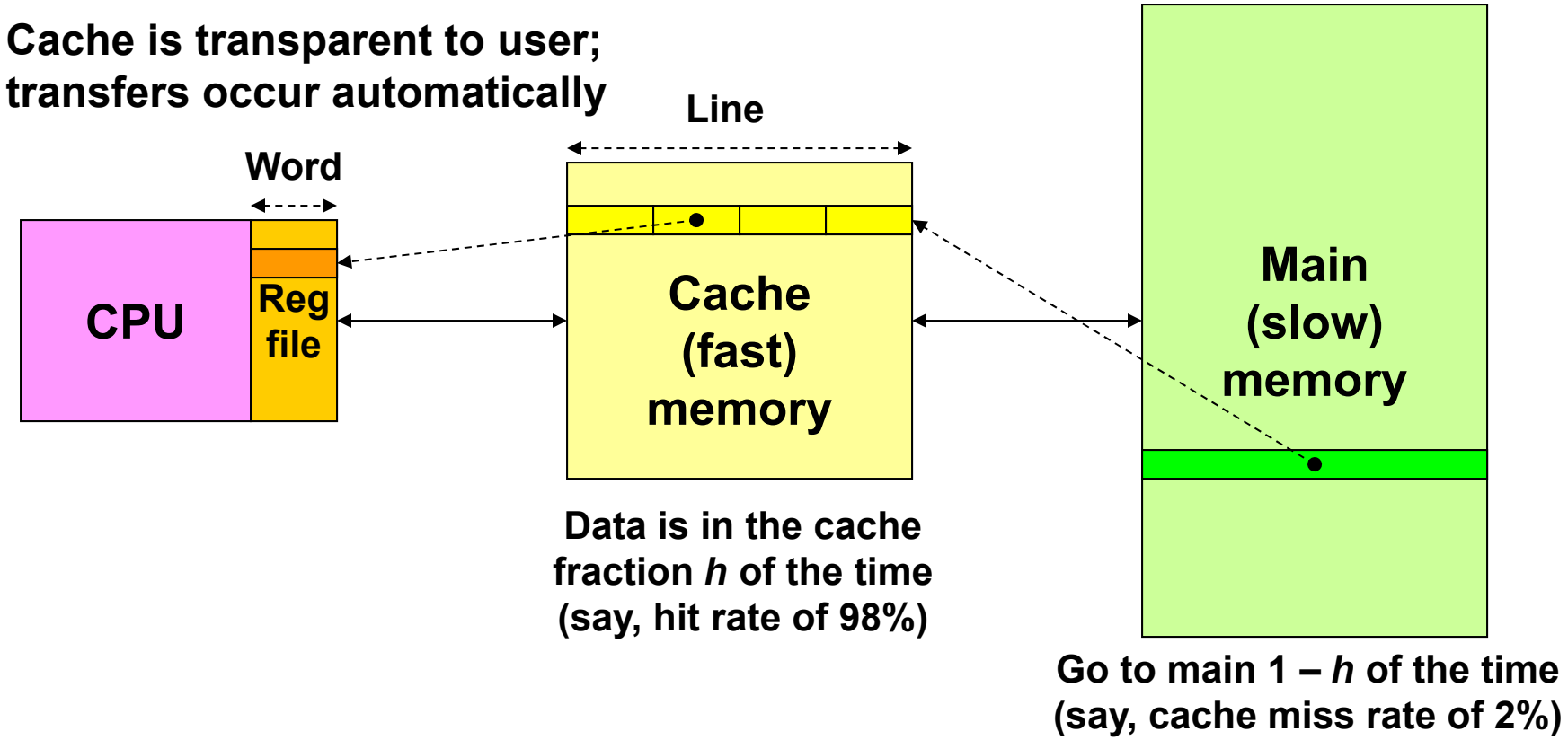
Memory Hierarchy

Hierarchical memory provides the illusion that high speed and large size are achieved simultaneously

Capacity	Access latency		Cost per GB
100s B	ns	Reg's	$Millions
10s KB	a few ns	Cache 1	$100s Ks
MBs	10s ns	Cache 2	$10s Ks
100s MB	100s ns	Main	$1000s
		Speed gap	
10s GB	10s ms	Secondary	$10s
TBs	min+	Tertiary	$1s



Hit/Miss Rate, and Effective Cycle Time



One level of cache with hit rate *h*

$$C_{eff} = hC_{fast} + (1 - h)(C_{slow} + C_{fast}) = C_{fast} + (1 - h)C_{slow}$$

The Locality Principle

From Peter Denning's *CACM* paper, July 2005 (Vol. 48, No. 7, pp. 19-24)

Temporal:
Accesses to the same address are typically clustered in time

Spatial:
When a location is accessed, nearby locations tend to be accessed also

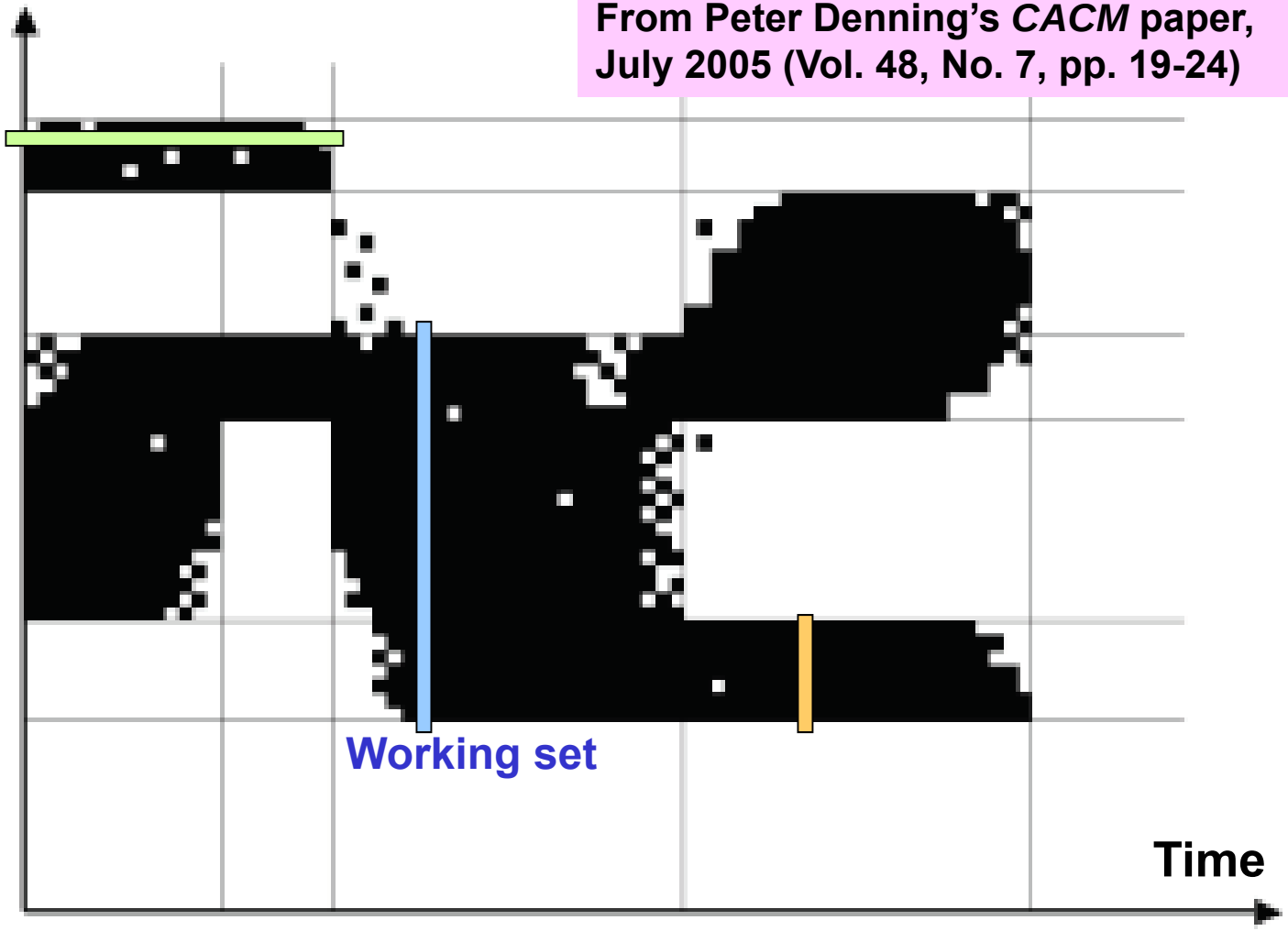


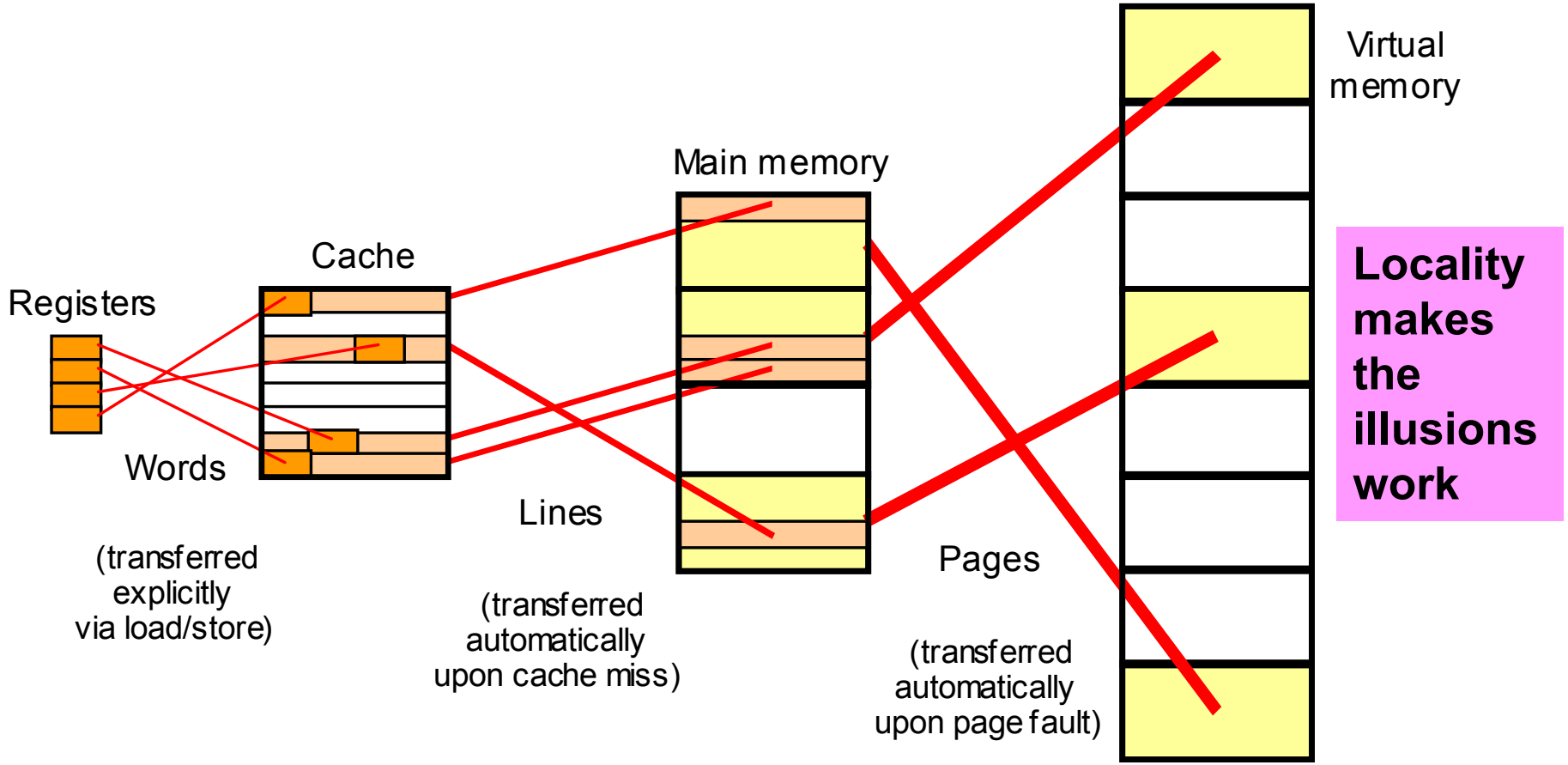
Illustration of temporal and spatial localities

Summary of Memory Hierarchy

Cache memory: provides illusion of very high speed

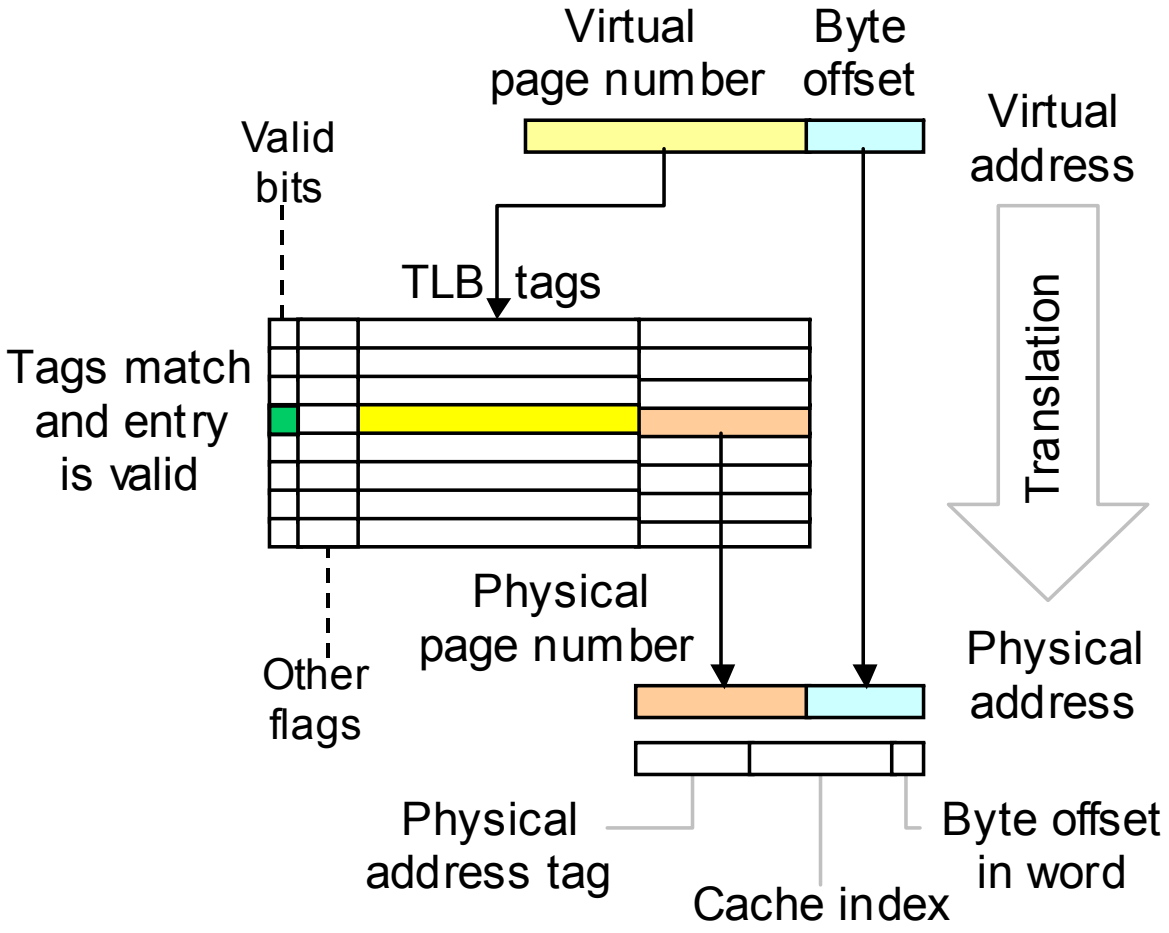
Main memory: reasonable cost, but slow & small

Virtual memory: provides illusion of very large size



Locality makes the illusions work

Translation Lookaside Buffer



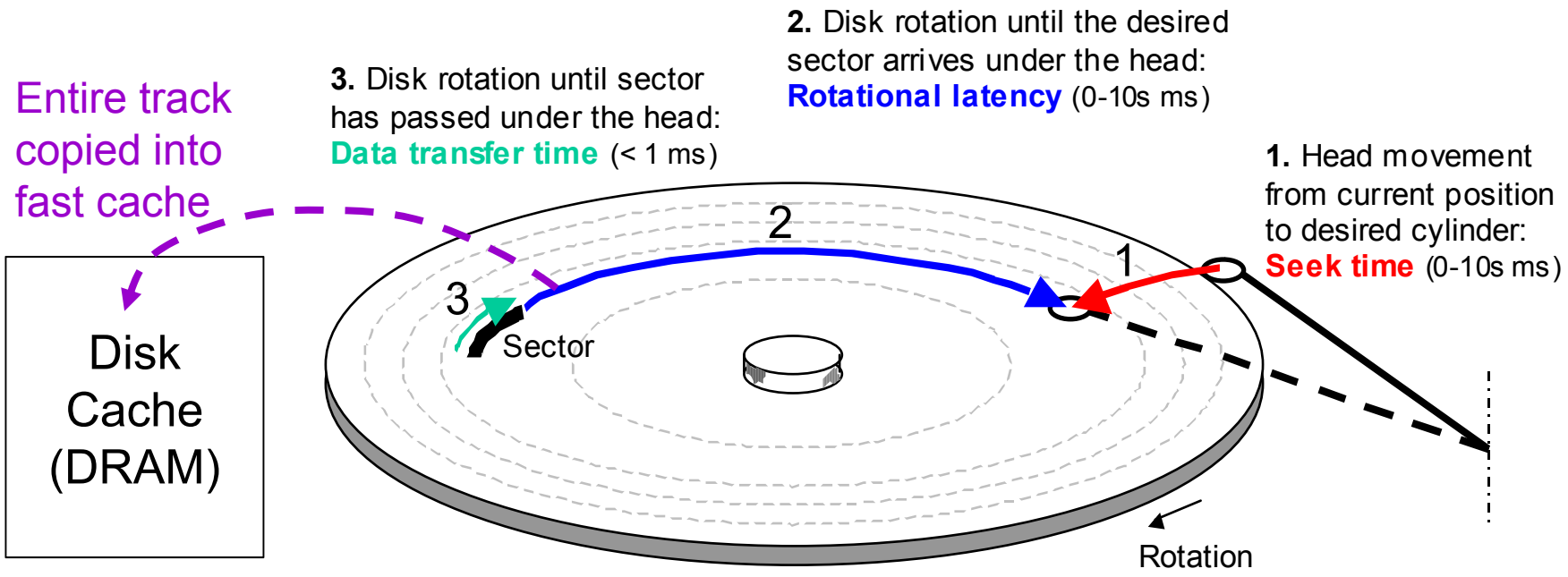
Program page in virtual memory

```
     ...
     lw    $t0,0($s1)
     addi  $t1,$zero,0
L:   add   $t1,$t1,1
     beq   $t1,$s2,D
     add   $t2,$t1,$t1
     add   $t2,$t2,$t2
     add   $t2,$t2,$s1
     lw    $t3,0($t2)
     slt   $t4,$t0,$t3
     beq   $t4,$zero,L
     addi  $t0,$t3,0
     j     L
D:   ...
```

All instructions on this page have the same virtual page address and thus entail the same translation

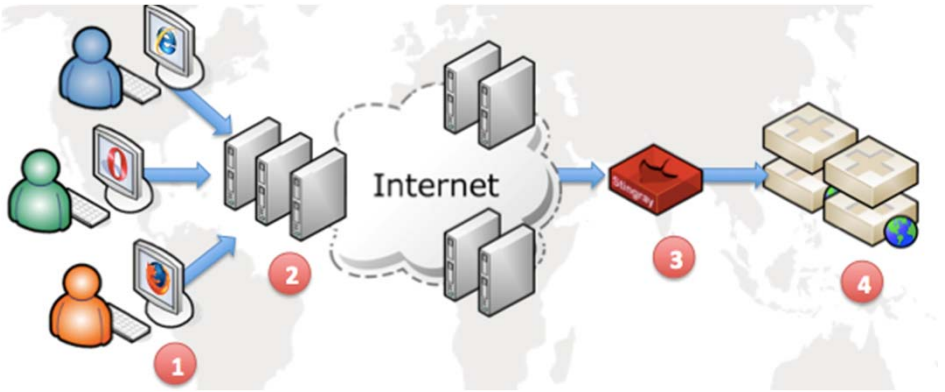
Virtual-to-physical address translation by a TLB and how the resulting physical address is used to access the cache memory.

Disk Caching and Other Applications



Web caching

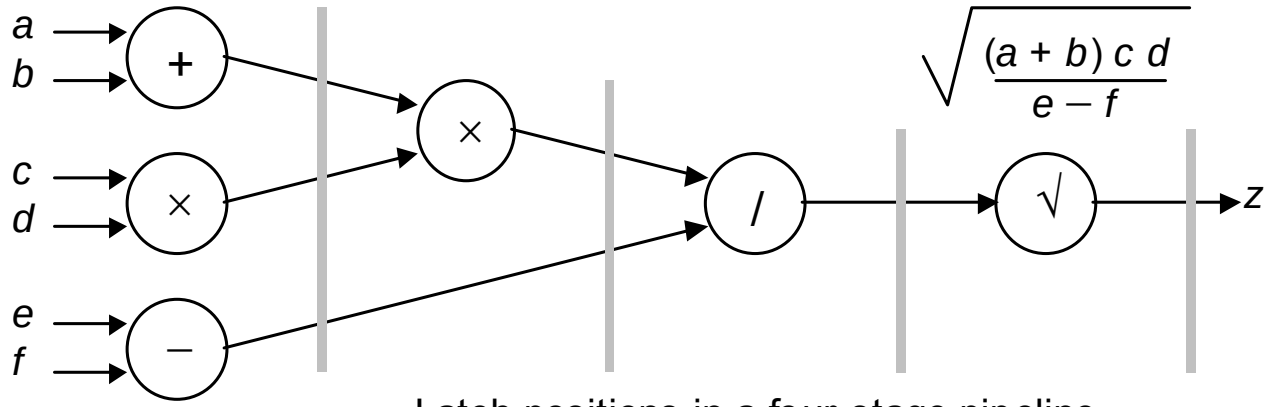
- Client-side caching
- Caching within the cloud
- Server-side caching



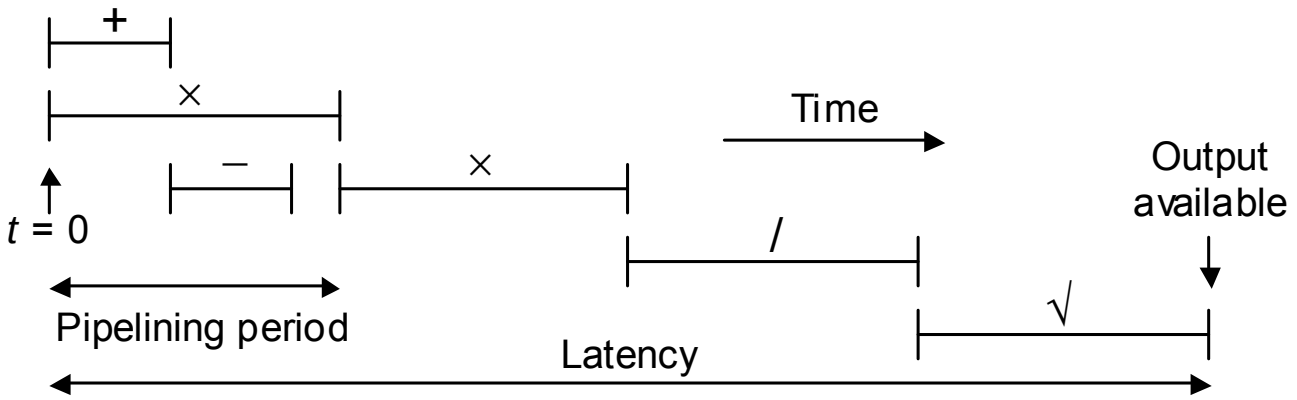
1980s: Pipelining

An important form of parallelism that is given its own name

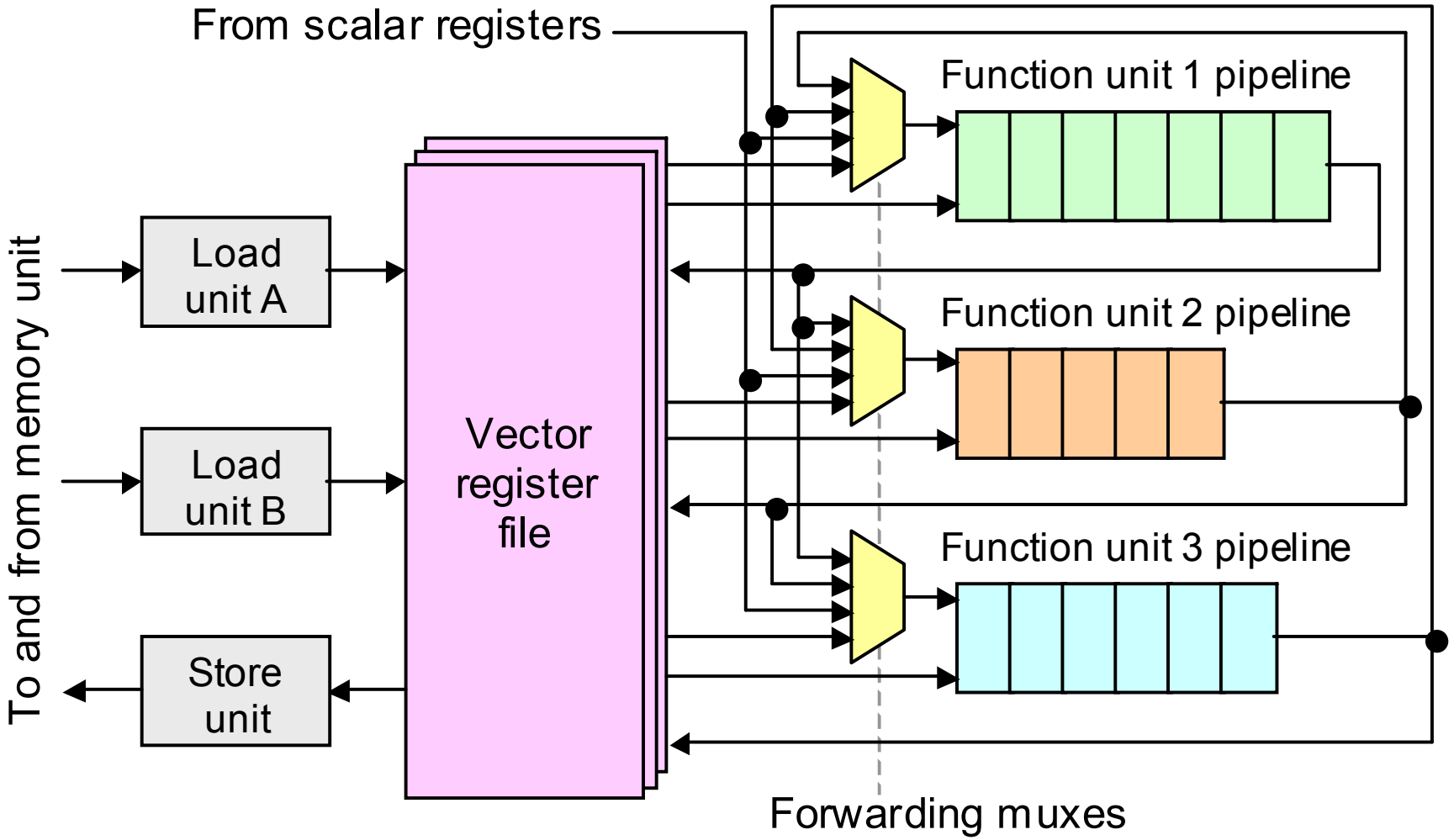
Used from early days of digital circuits in various forms



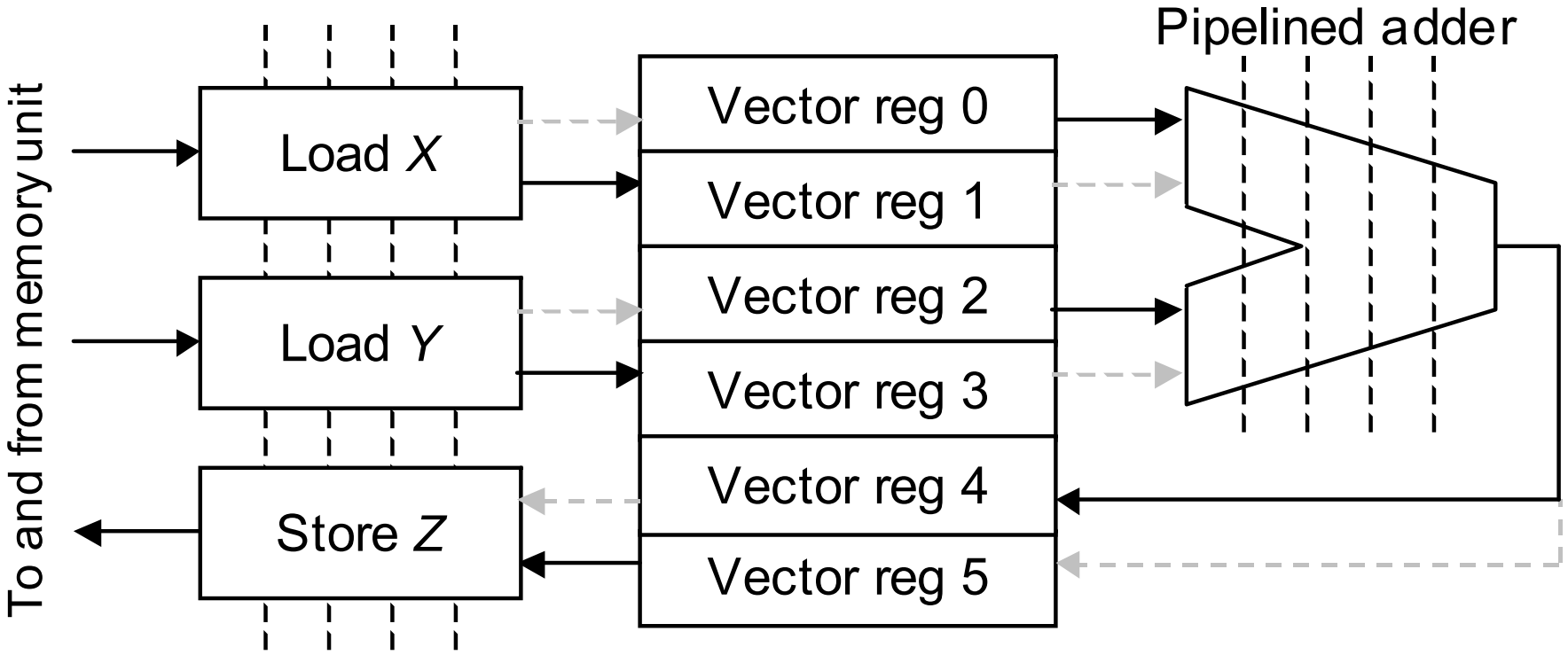
Latch positions in a four-stage pipeline



Vector Processor Implementation

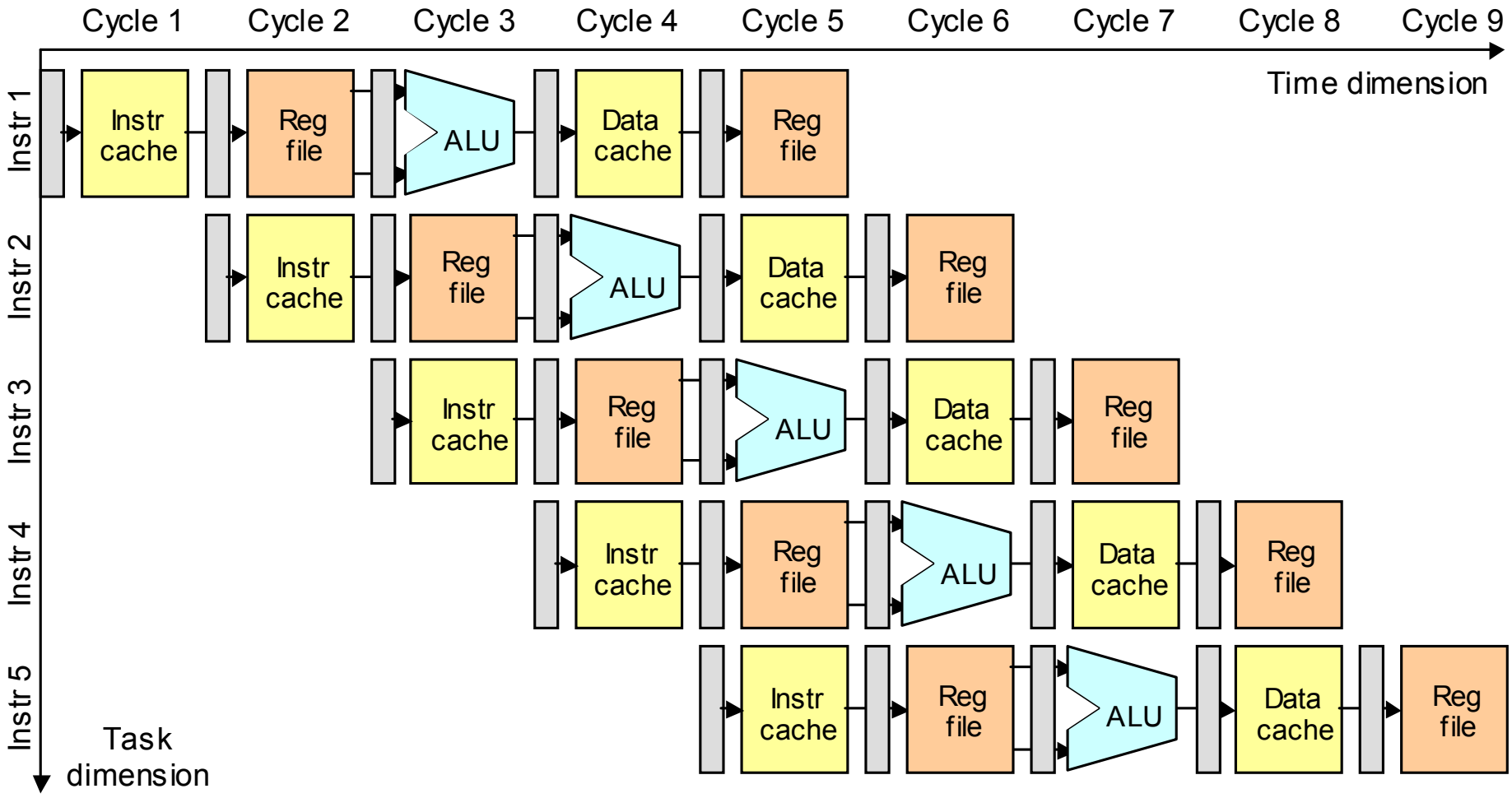


Overlapped Load/Store and Computation



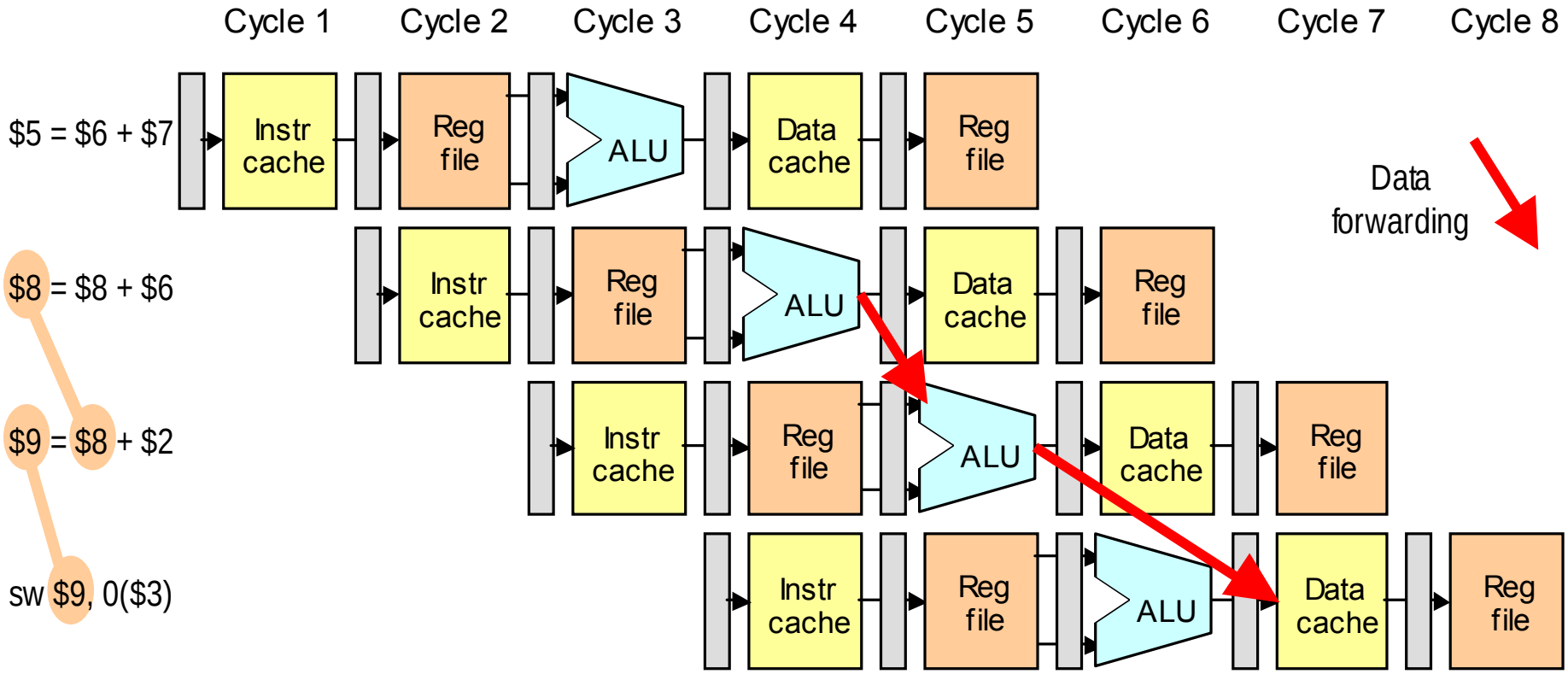
Vector processing via segmented load/store of vectors in registers in a double-buffering scheme. Solid (dashed) lines show data flow in the current (next) segment.

Simple Instruction-Execution Pipeline

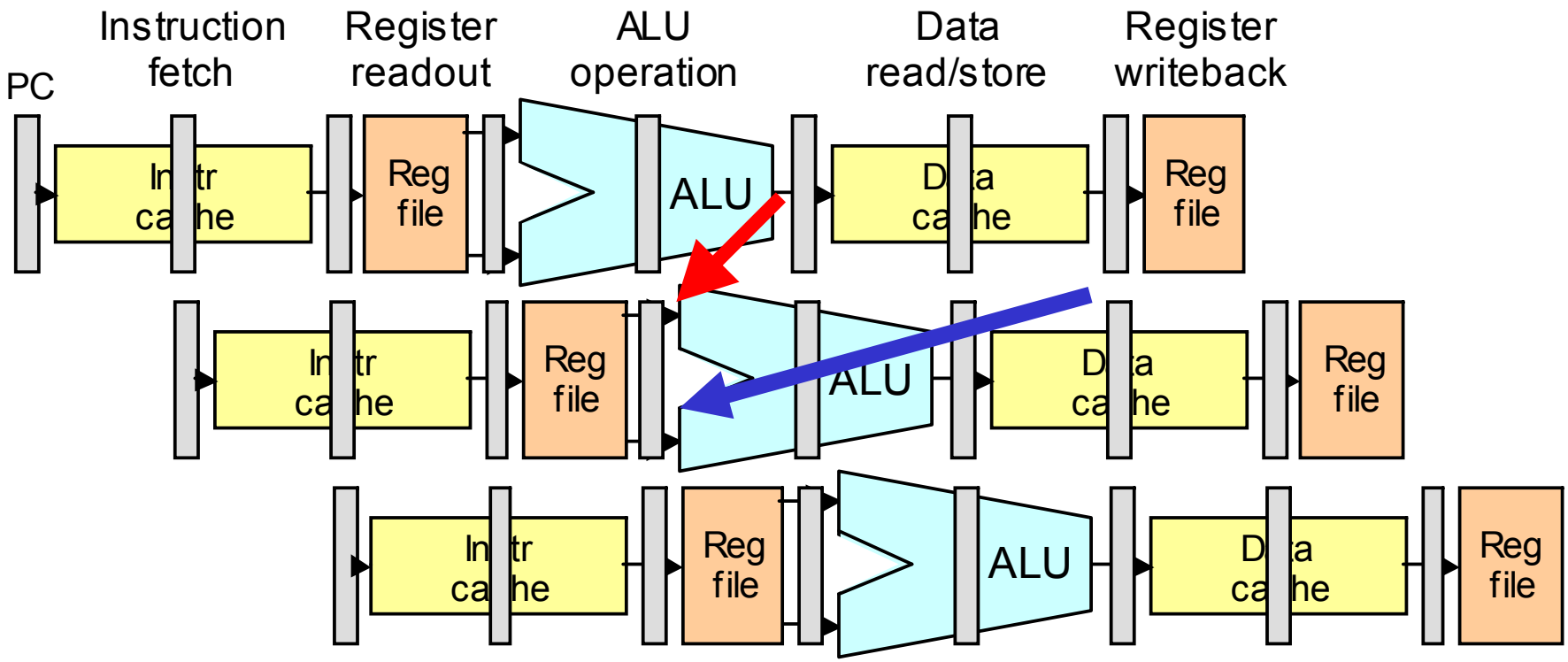


Pipeline Stalls or Bubbles

Data dependency and its possible resolution via forwarding

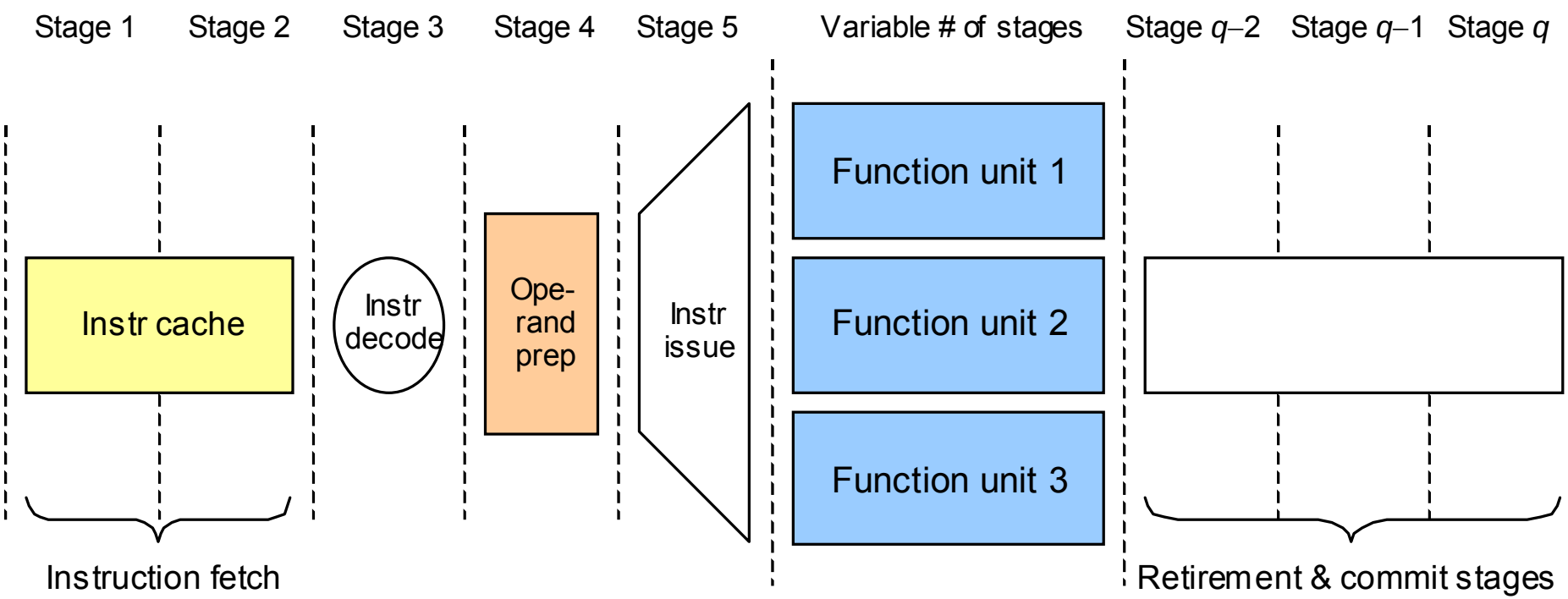


Problems Arising from Deeper Pipelines



Forwarding more complex and not always workable
Interlocking/stalling mechanisms needed to prevent errors

Branching and Other Complex Pipelines



Front end: **In-order or out-of-order**
Instr. issue: **In-order or out-of-order**
Write-back: **In-order or out-of-order**
Commit: **In-order or out-of-order**

The more OoO stages, the higher the complexity

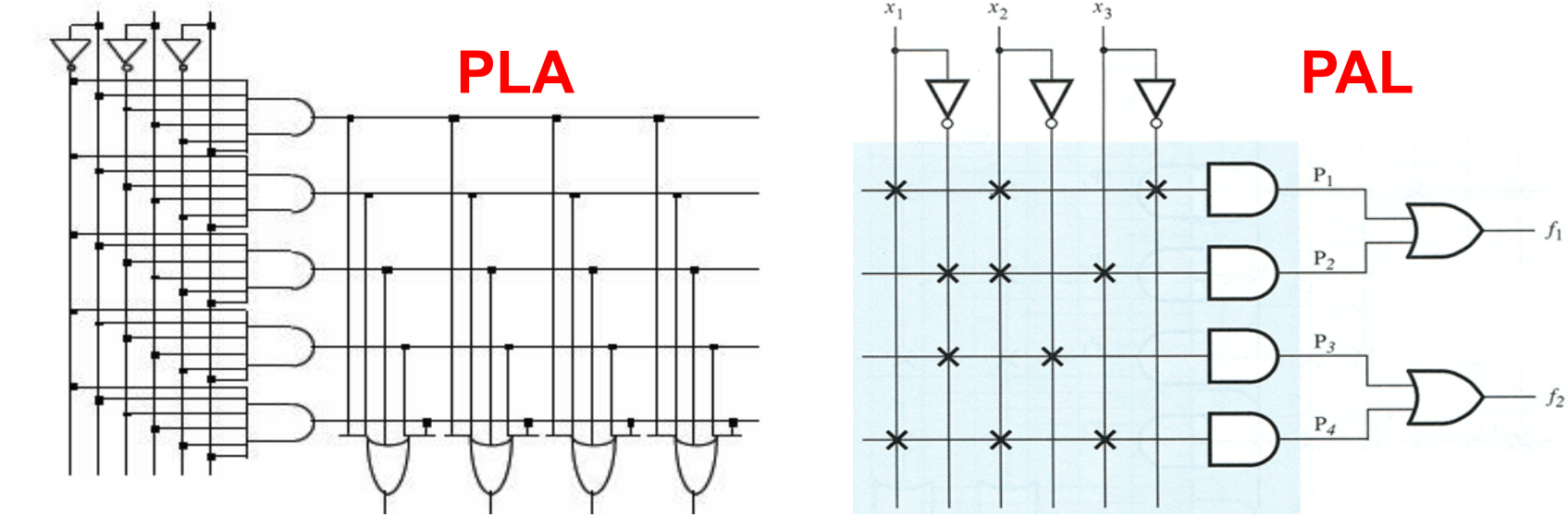
1990s: FPGAs

Programmable logic arrays were developed in the 1970s

PLAs provided cost-effective and flexible replacements for random logic or ROM/PROM

The related programmable array logic devices came later

PALs were less flexible than PLAs, but more cost-effective



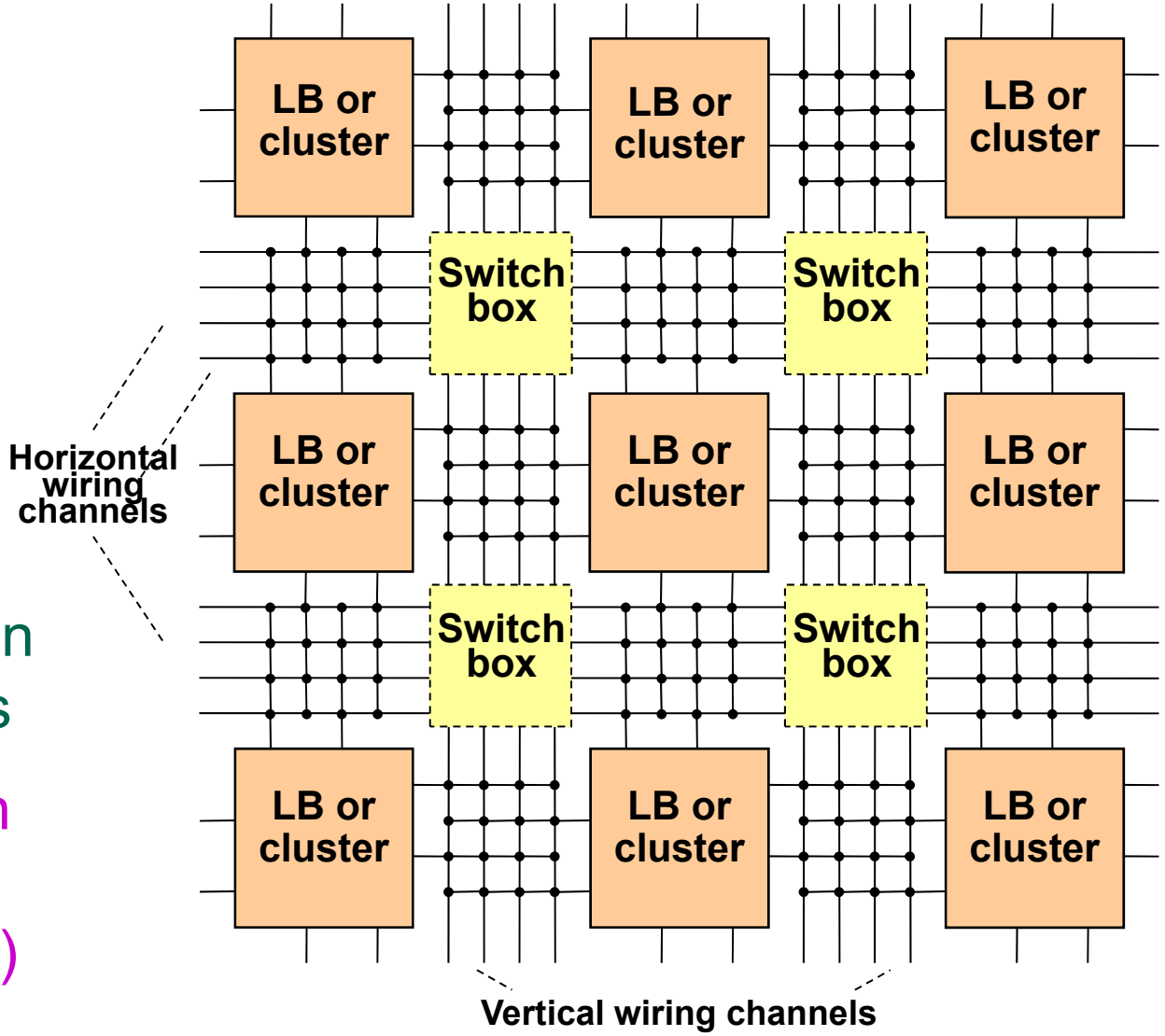
Why FPGA Represents a Paradigm Shift

Modern FPGAs can implement any functionality

Initially used only for prototyping

Even a complete CPU needs a small fraction of an FPGA's resources

FPGAs come with multipliers and IP cores (CPUs/SPs)

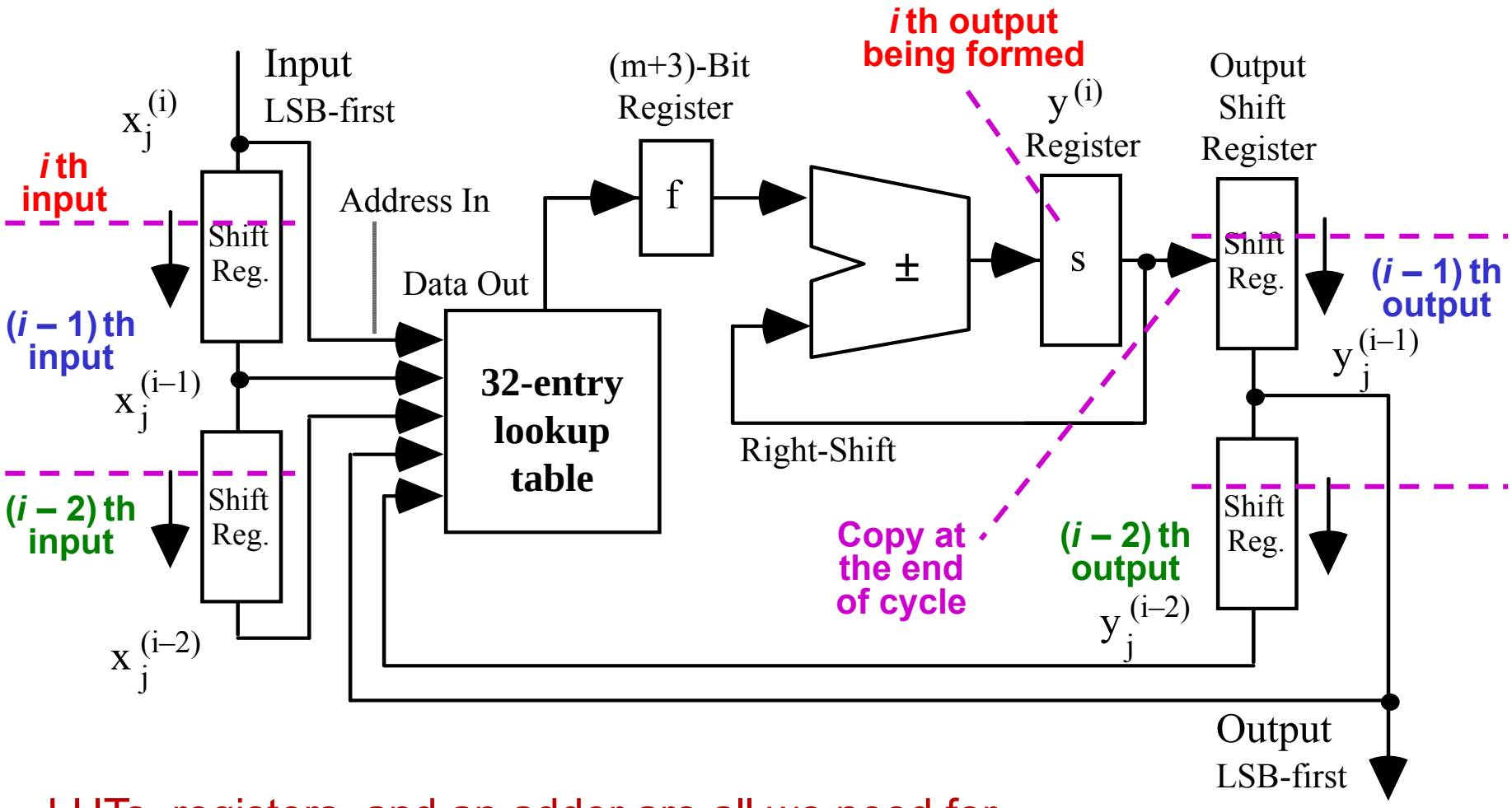


FPGAs Are Everywhere

Applications are found in virtually all industry segments:

- Aerospace and defense
- Medical electronics
- Automotive control
- Software-defined radio
- Encoding and decoding

Example: Bit-Serial 2nd-Order Digital Filter



LUTs, registers, and an adder are all we need for linear expression evaluation: $y^{(i)} = ax^{(i)} + bx^{(i-1)} + cx^{(i-2)} + dy^{(i-1)} + ey^{(i-2)}$

2000s: GPUs

Simple graphics and signal processing units were used since the 1970s

In the early 2000s, the two major players, ATI and Nvidia, produced powerful chips to improve the speed of shading

In the late 2000s, GPGPUs (extended stream processors) emerged and were used in lieu of, or in conjunction with, CPUs in high-performance supercomputers

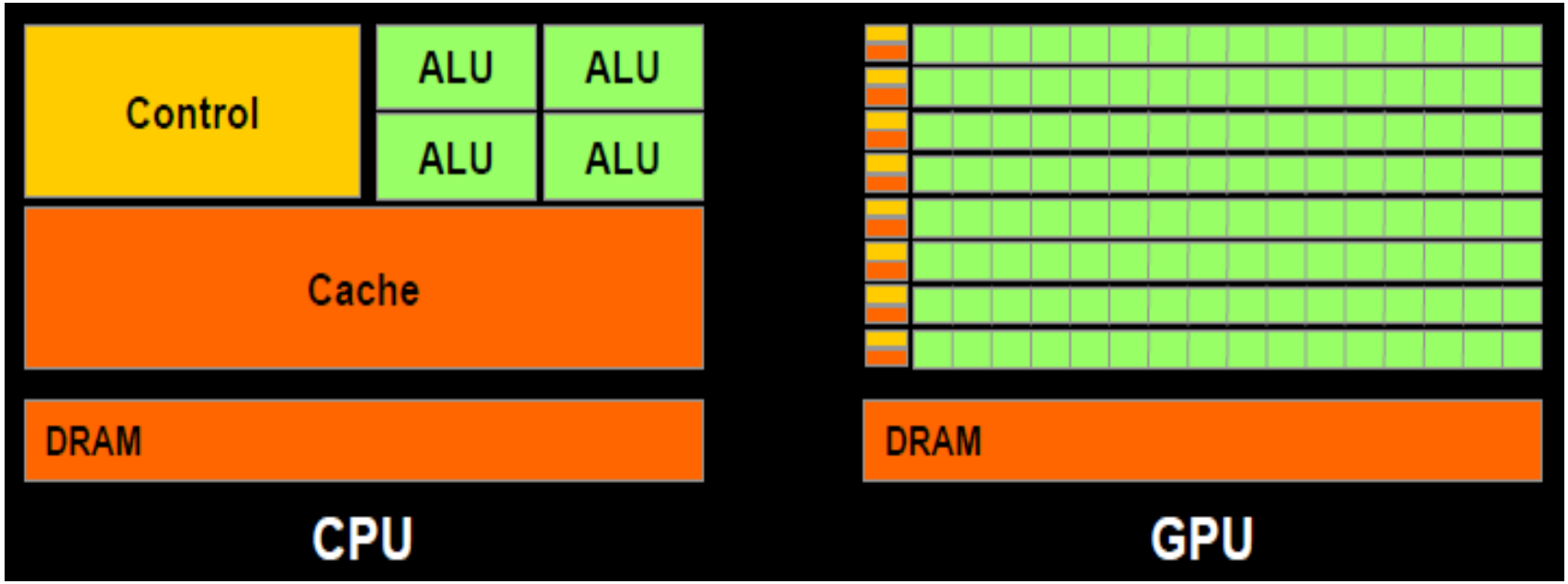
GPUs are faster and more power-efficient than CPUs.

GPUs use a mixture of parallel processing and functional specialization to achieve super-high performance

CPU vs. GPU Organization

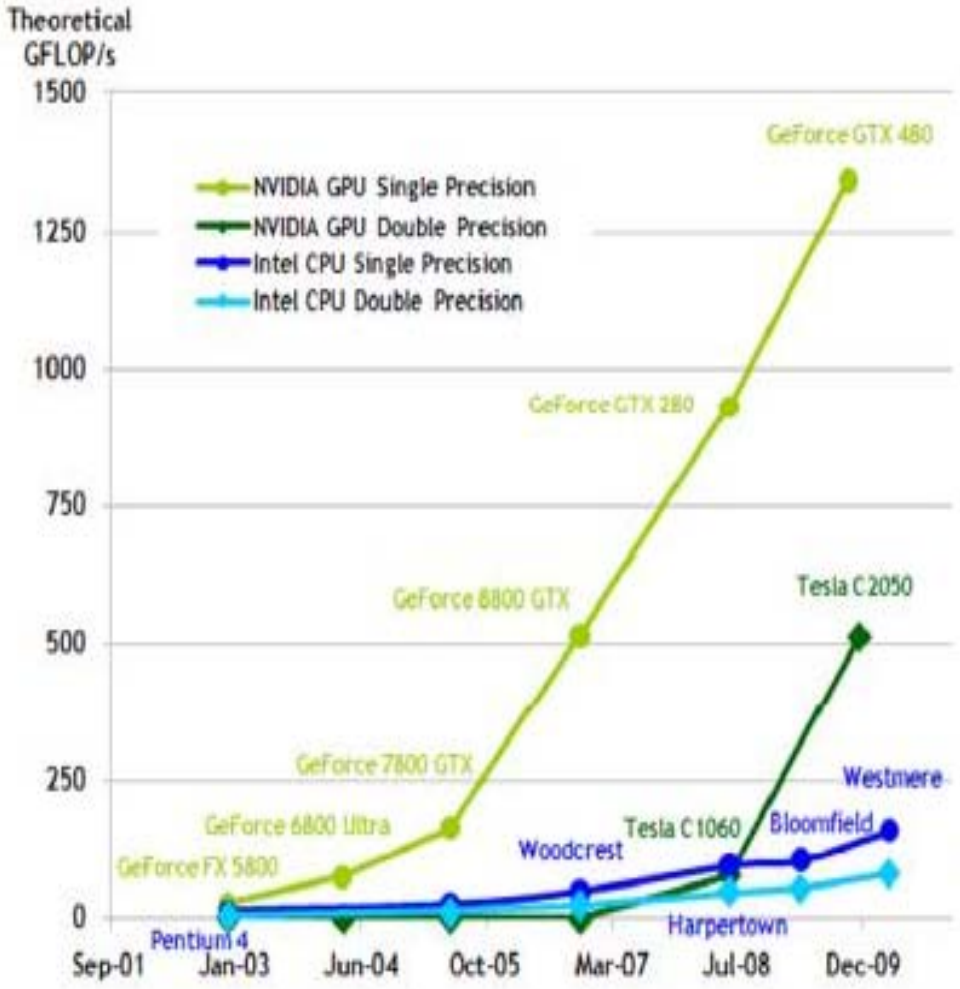
Small number of powerful cores
versus
Very large number of simple stream processors

Demo (analogy for MPP): https://www.youtube.com/watch?v=fKK933KK6Gg



CPU vs. GPU Performance

Peak performance (GFLOPS) and peak data rate (GB/s)



General-Purpose Computing on GPUs

Suitable for numerically intensive matrix computations

First application to run faster on a GPU was LU factorization

Users can ignore GPU features and focus on problem solving
- Nvidia CUDA Programming System
- Matlab Parallel Computing Toolbox
- C++ Accelerated Massive Parallelism

Many vendors now give users direct access to GPU features

Example system (Titan):
Cray XK7 at DOE's Oak Ridge Nat'l Lab used more than ¼ M Nvidia K20x cores to accelerate computations
(energy-efficient: 2+ gigaflops/W)

The Seven Key Ideas

Design advances
Performance improvements

Innovations for Improved Performance

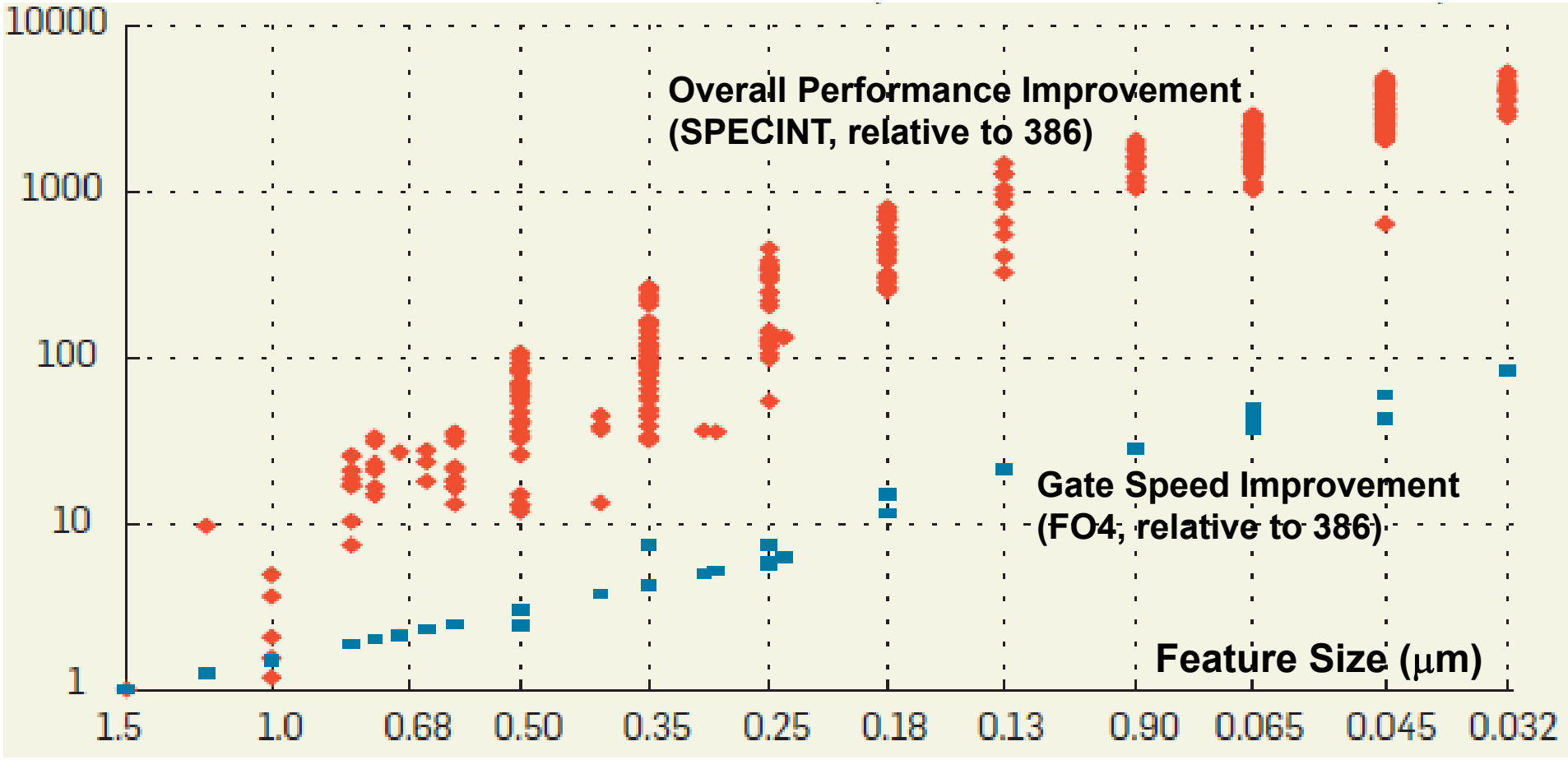
(Parhami: *Computer Architecture*, 2005)

Computer performance grew by a factor of about 10000 between 1980 and 2000
- 100 due to faster technology
- 100 due to better architecture

Available computing power ca. 2000:
- GFLOPS on desktop
- TFLOPS in supercomputer center
- PFLOPS on drawing board

	Architectural method	Improvement factor	
Established methods	1. **Pipelining (and superpipelining)**	3-8 √	Previously discussed
	2. **Cache memory, 2-3 levels**	2-5 √	
	3. RISC and related ideas	2-3 √	
	4. Multiple instruction issue (superscalar)	2-3 √	
	5. ISA extensions (e.g., for multimedia)	1-3 √	Covered in Part VII
Newer methods	6. Multithreading (super-, hyper-)	2-5 ?	
	7. Speculation and value prediction	2-3 ?	
	8. **Hardware acceleration [e.g., GPU]**	2-10 ?	
	9. Vector and array processing	2-10 ?	
	10. **Parallel/distributed computing**	2-1000s ?	

Shares of Technology and Architecture in Processor Performance Improvement



Source: "CPU DB: Recording Microprocessor History," *CACM*, April 2012.

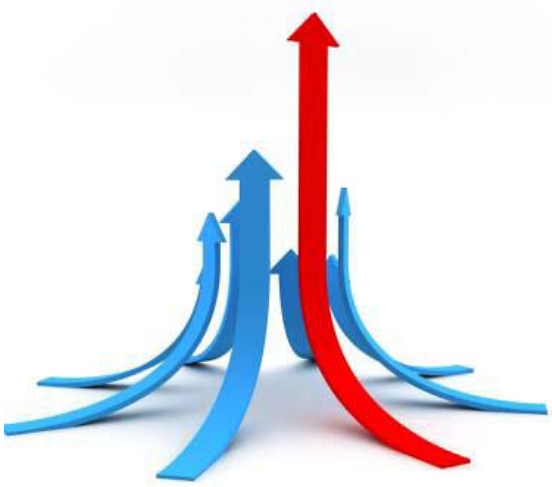
2010s and Beyond: Looking Ahead

Design improvements

- Adaptation and self-optimization (learning)
- Security (hardware-implemented)
- Reliability via redundancy and self-repair
- Logic-in-memory designs (memory wall)
- Mixed analog/digital design style

Performance improvements

- Revolutionary new technologies
- New computational paradigms
- Brain-inspired and biological computing
- Speculation and value prediction
- Better performance per watt (power wall)



We Need More than Sheer Performance

Environmentally responsible design

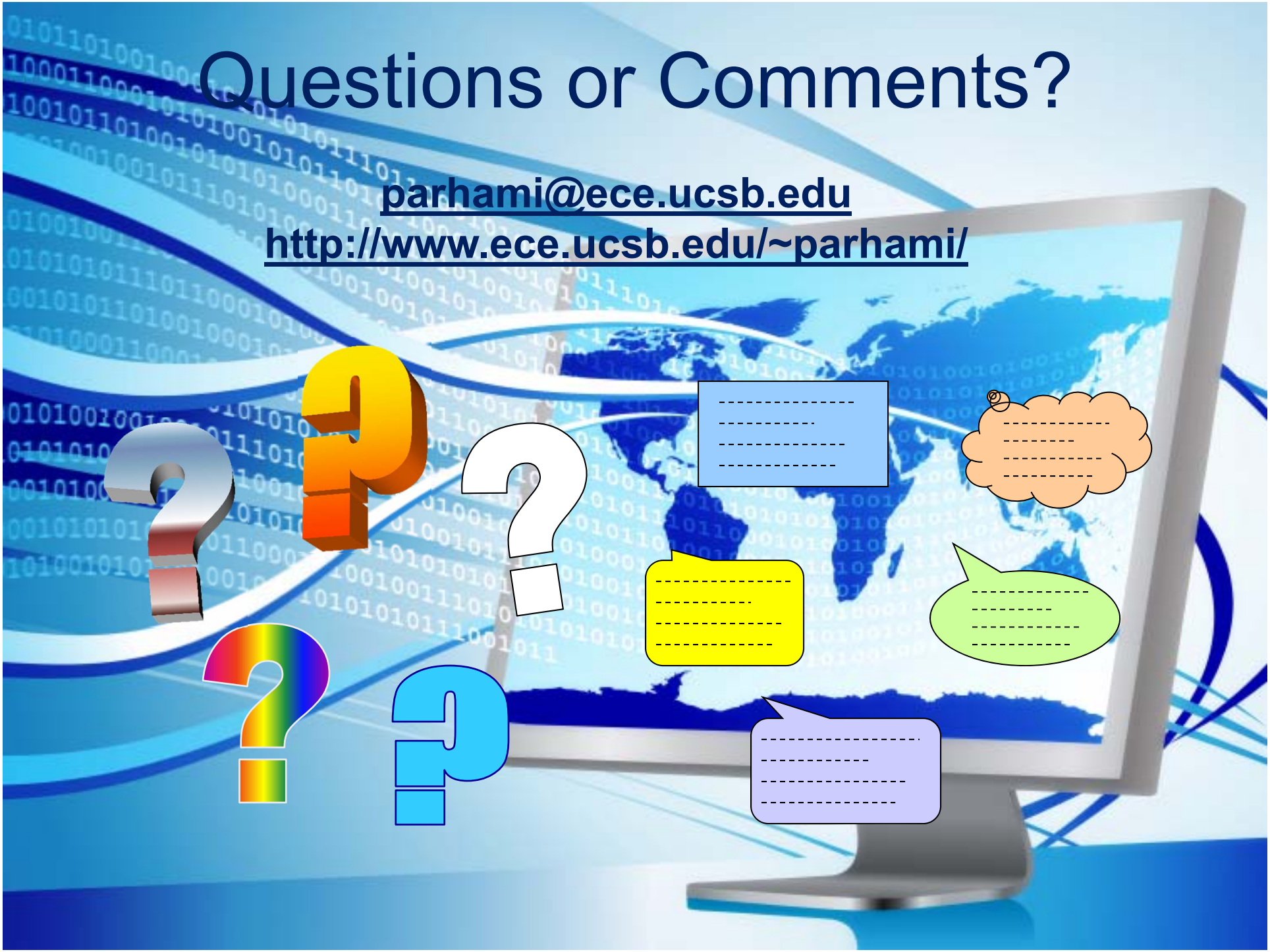
Reusable designs, parts, and material

Power efficiency

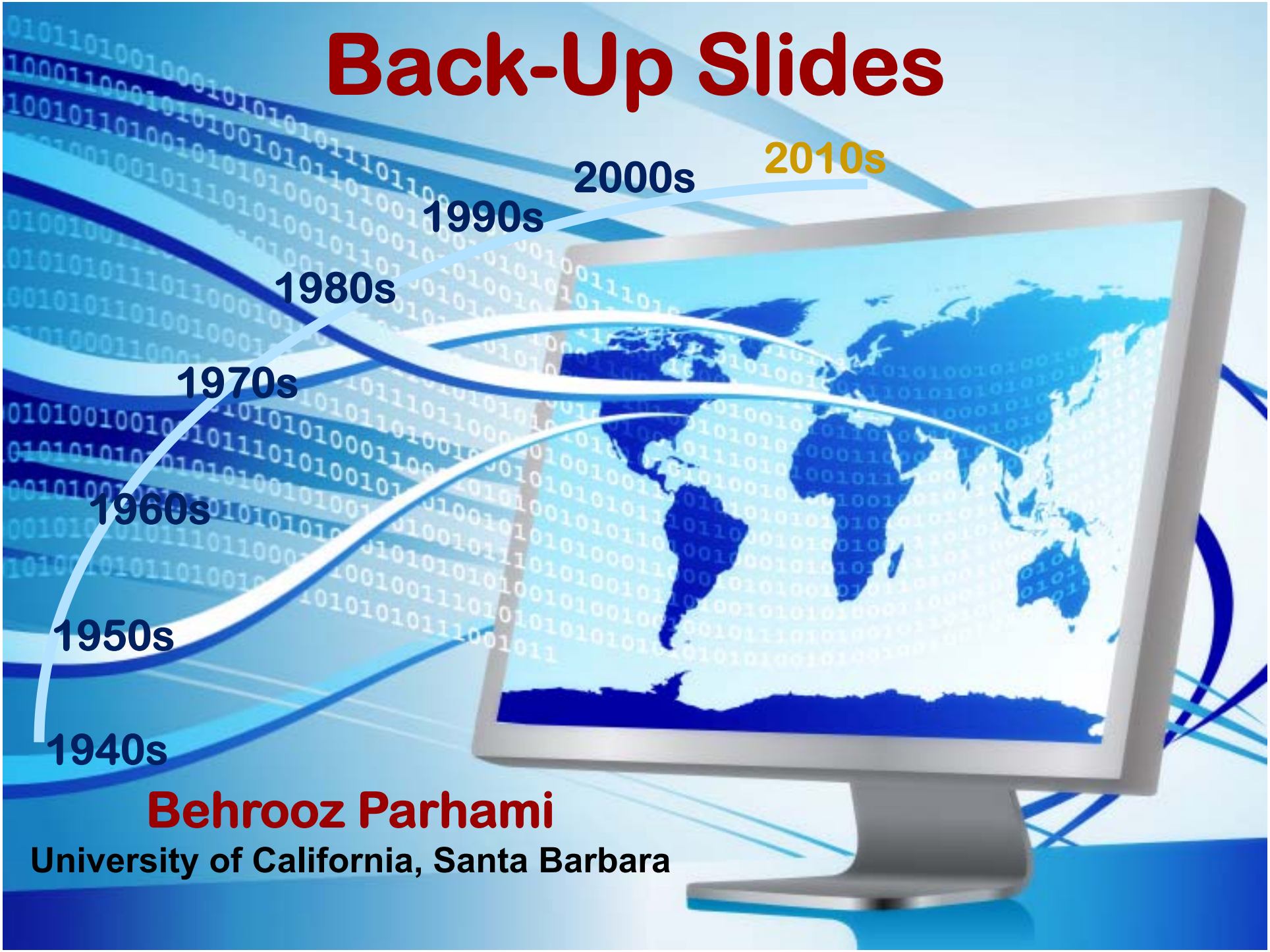
Starting publication in 2016: *IEEE Transactions on Sustainable Computing*

Questions or Comments?

parhami@ece.ucsb.edu
http://www.ece.ucsb.edu/~parhami/



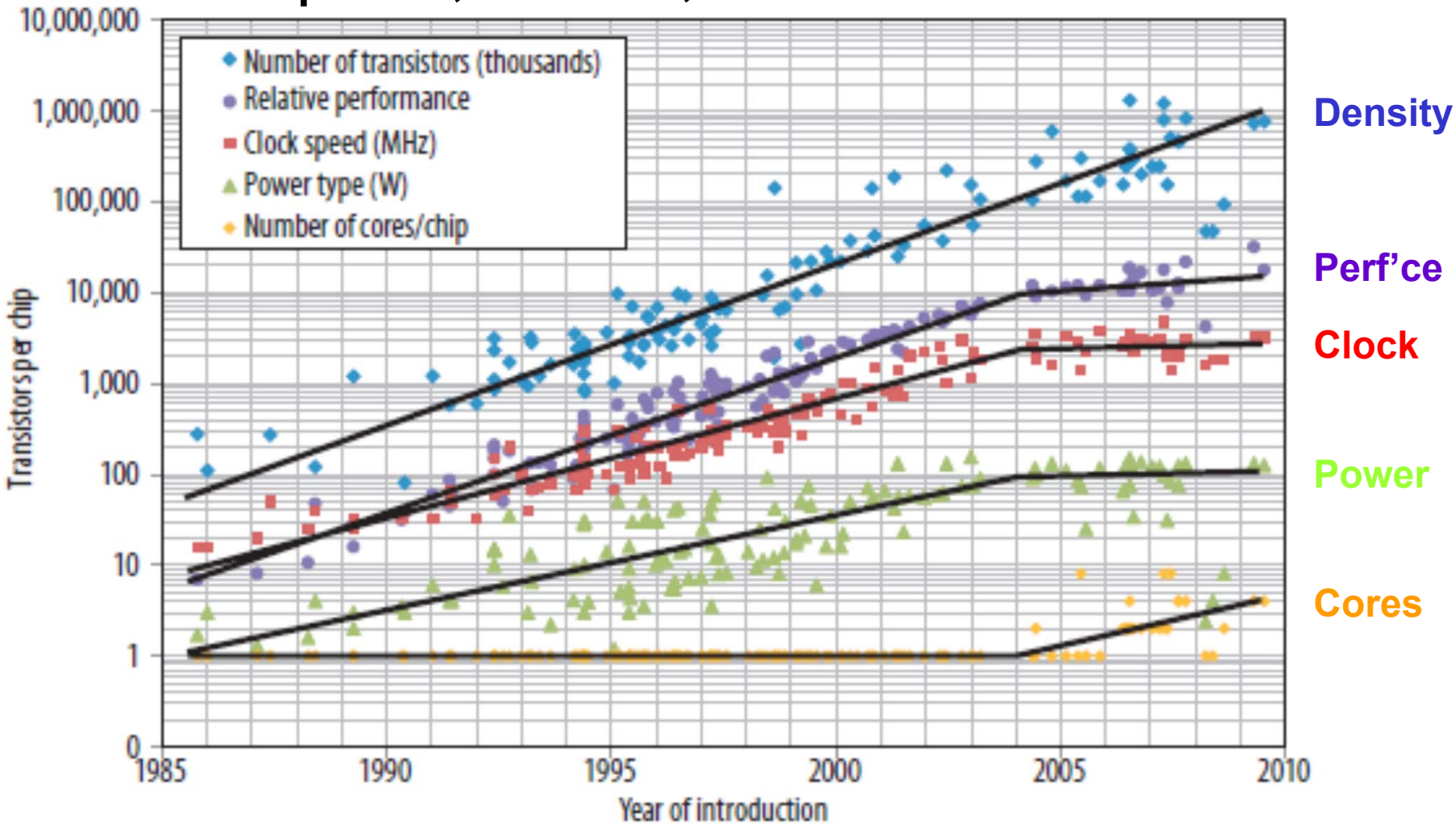
Back-Up Slides



Behrooz Parhami

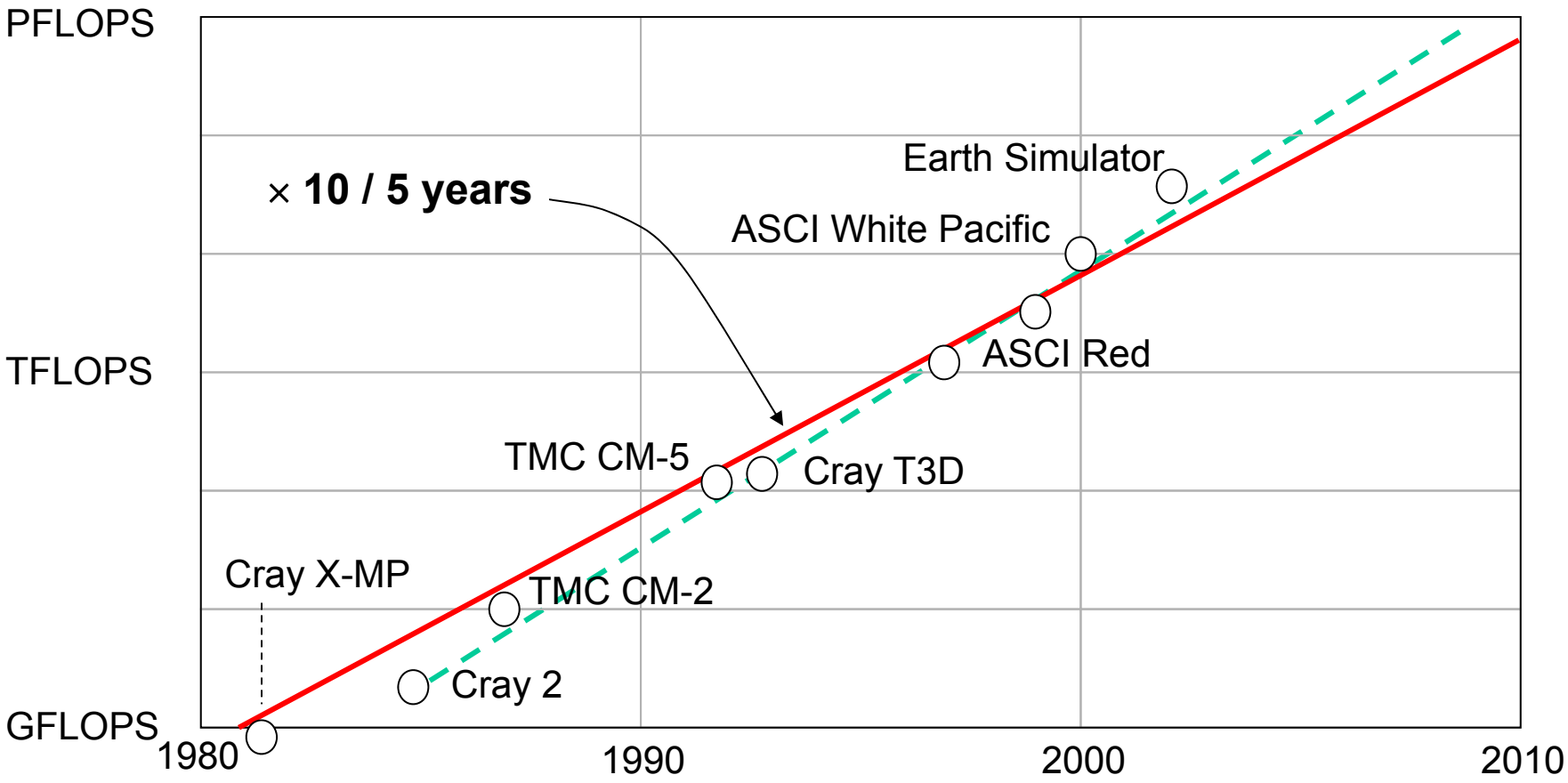
University of California, Santa Barbara

Trends in Processor Chip Density, Performance, Clock Speed, Power, and Number of Cores



NRC Report (2011): The Future of Computing Performance: Game Over or Next Level?

Peak Performance of Supercomputers



Dongarra, J., "Trends in High Performance Computing," *Computer J.*, Vol. 47, No. 4, pp. 399-403, 2004. [Dong04]

The Quest for Higher Performance

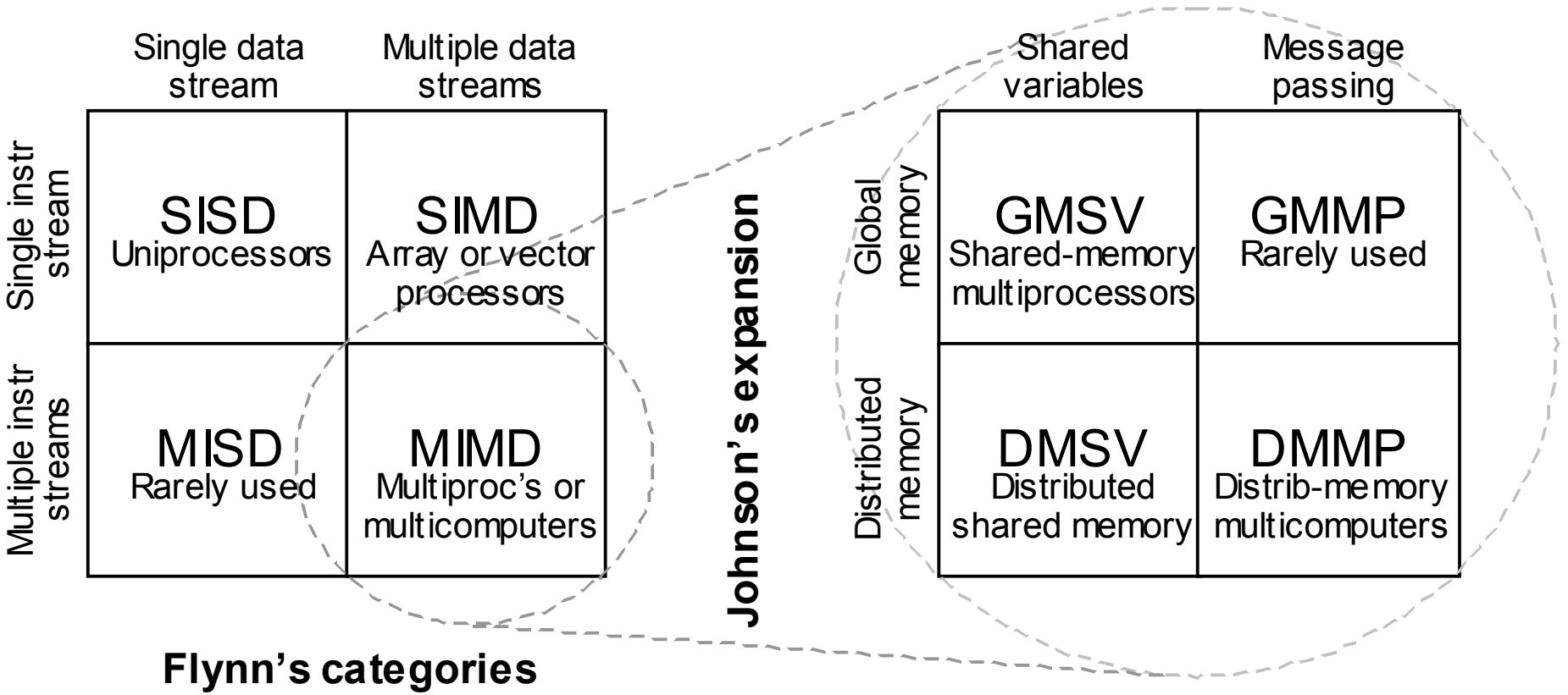
Top Three Supercomputers in November 2012 (http://www.top500.org)

1. Cray Titan	2. IBM Sequoia	3. Fujitsu K Computer
ORNL, Tennessee	LLNL, California	RIKEN AICS, Japan
XK7 architecture	Blue Gene/Q arch	RIKEN architecture
560,640 cores, 710 TB, Cray Linux	1,572,864 cores, 1573 TB, Linux	705,024 cores, 1410 TB, Linux
Cray Gemini interconn't	Custom interconnect	Tofu interconnect
17.6/27.1 PFLOPS*	**16.3/20.1 PFLOPS***	**10.5/11.3 PFLOPS***
AMD Opteron, 16-core, 2.2 GHz, NVIDIA K20x	Power BQC, 16-core, 1.6 GHz	SPARC64 VIIIfx, 2.0 GHz
8.2 MW power	7.9 MW power	12.7 MW power

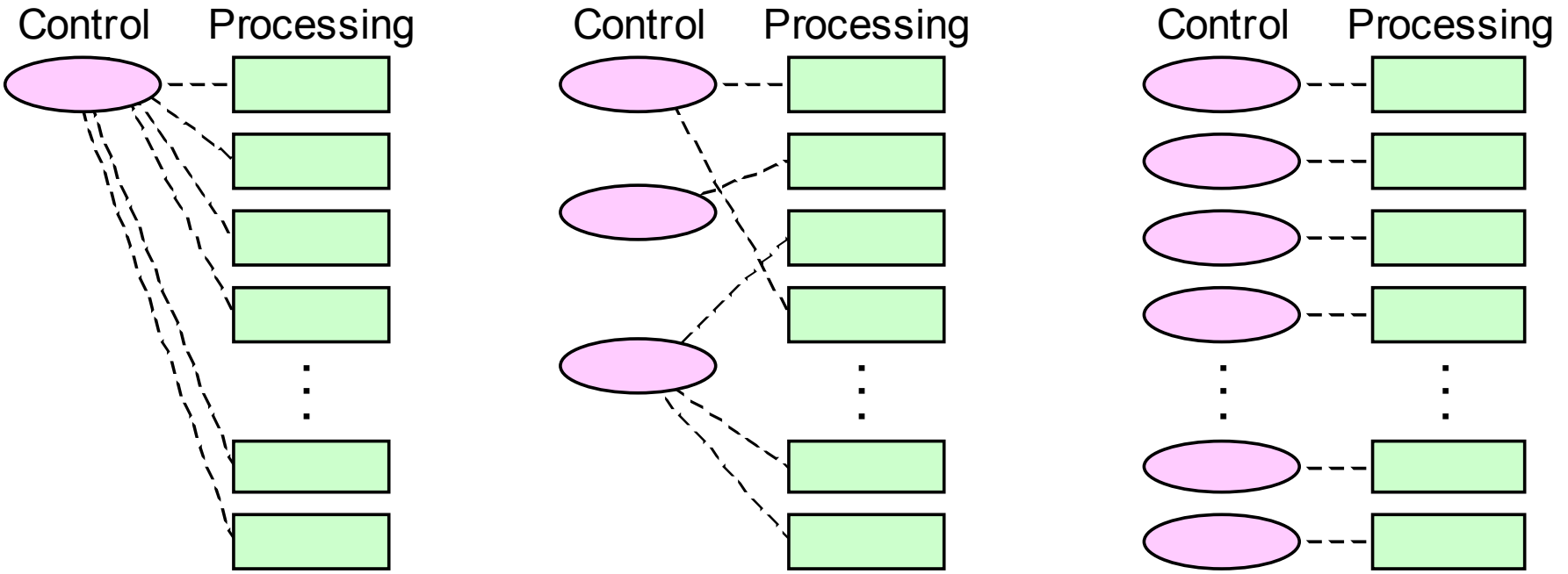
*** max/peak performance**

In the top 10, IBM also holds ranks 4-7 and 9-10. Dell and NUDT (China) hold ranks 7-8.

The Flynn/Johnson Classification



Shared-Control Systems



(a) Shared-control array processor, SIMD

(b) Multiple shared controls, MSIMD

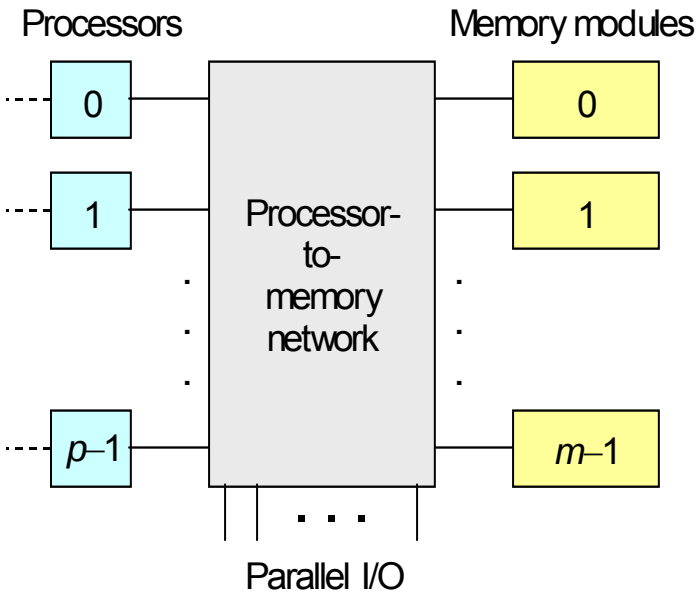
(c) Separate controls, MIMD

From completely shared control to totally separate controls.

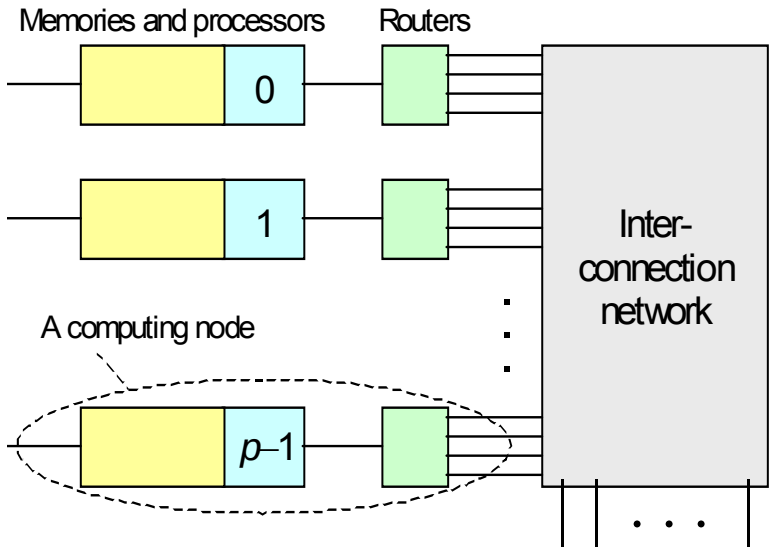
MIMD Architectures

Control parallelism: executing several instruction streams in parallel

GMSV: Shared global memory – symmetric multiprocessors
DMSV: Shared distributed memory – asymmetric multiprocessors
DMMP: Message passing – multicomputers



Centralized shared memory

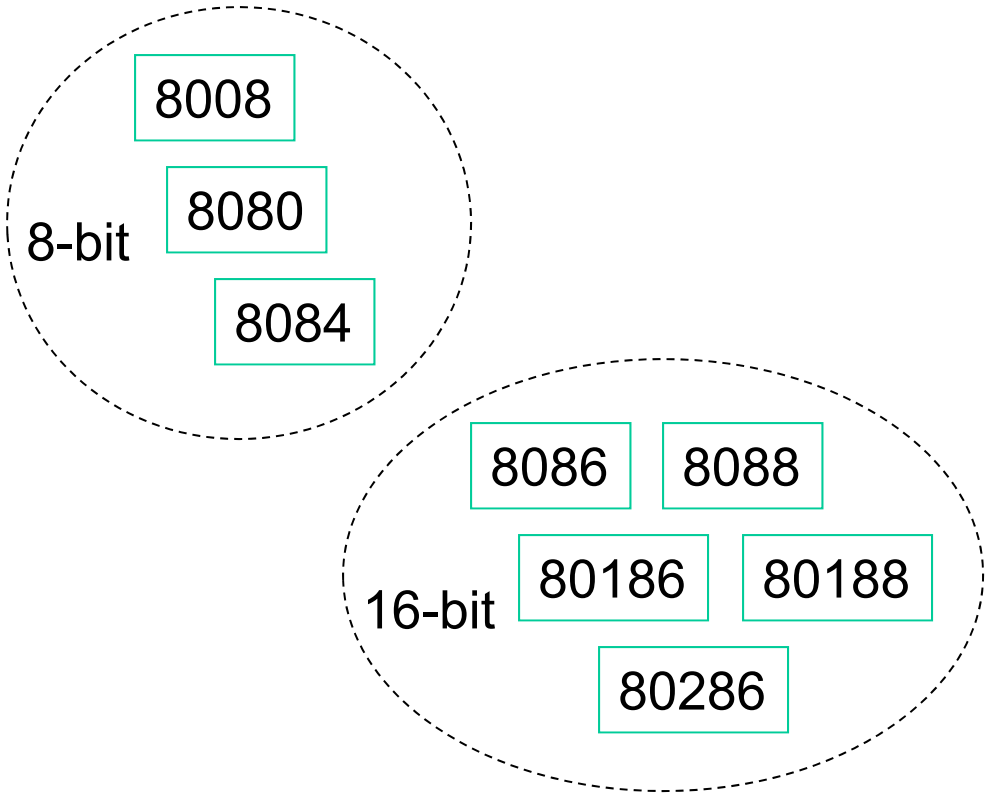


Distributed memory

Past and Current Performance Trends

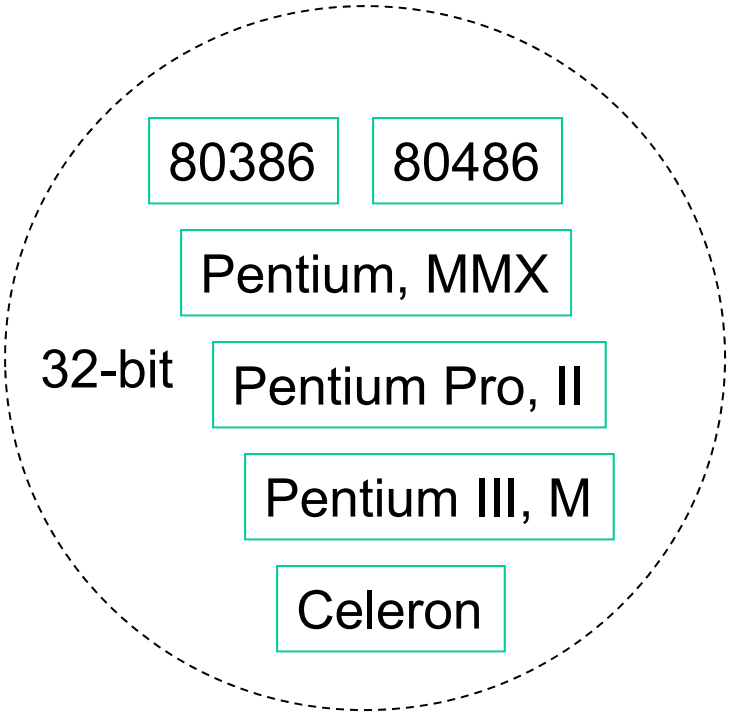
Intel 4004: The first μp (1971)

0.06 MIPS (4-bit processor)

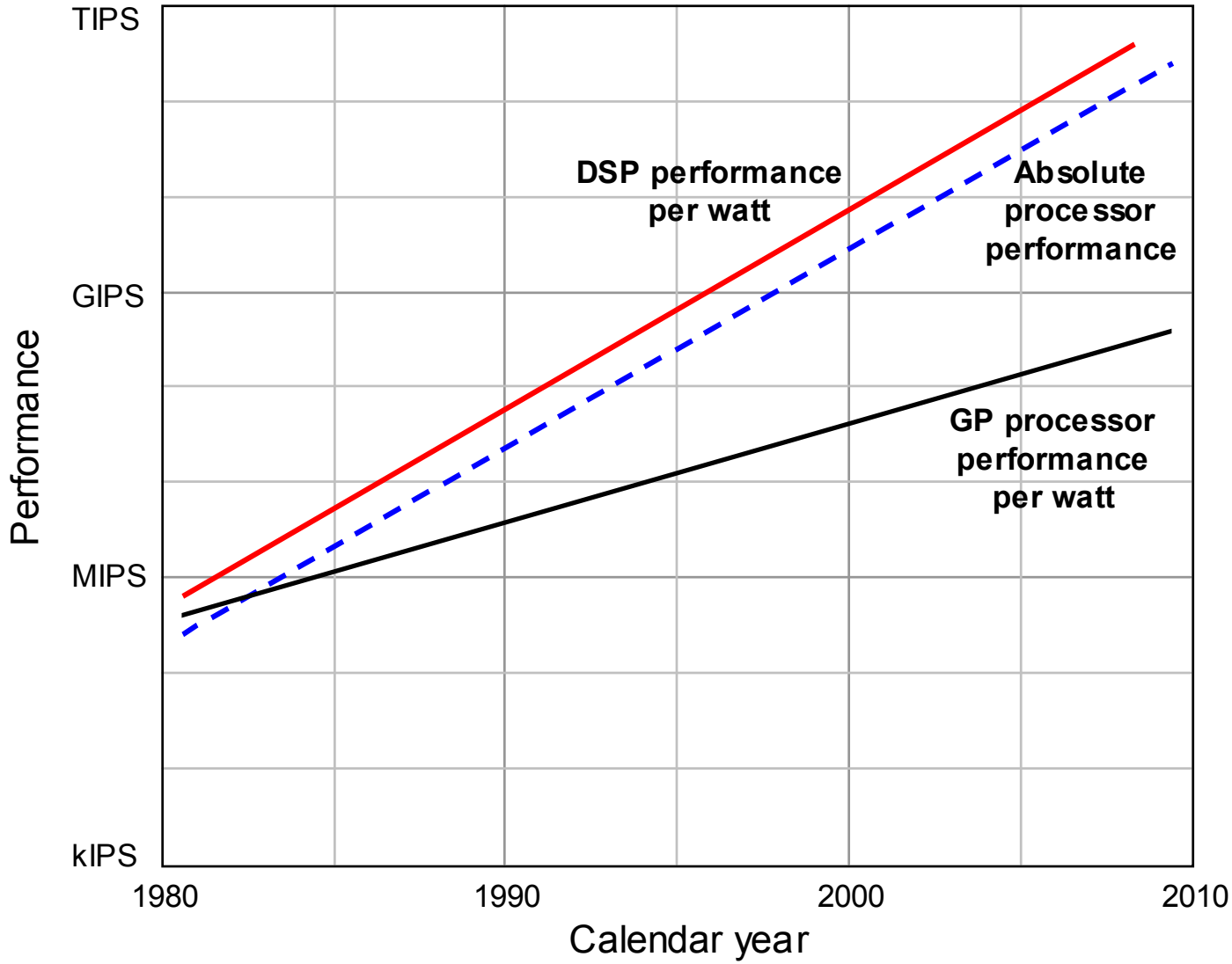


Intel Pentium 4, circa 2005

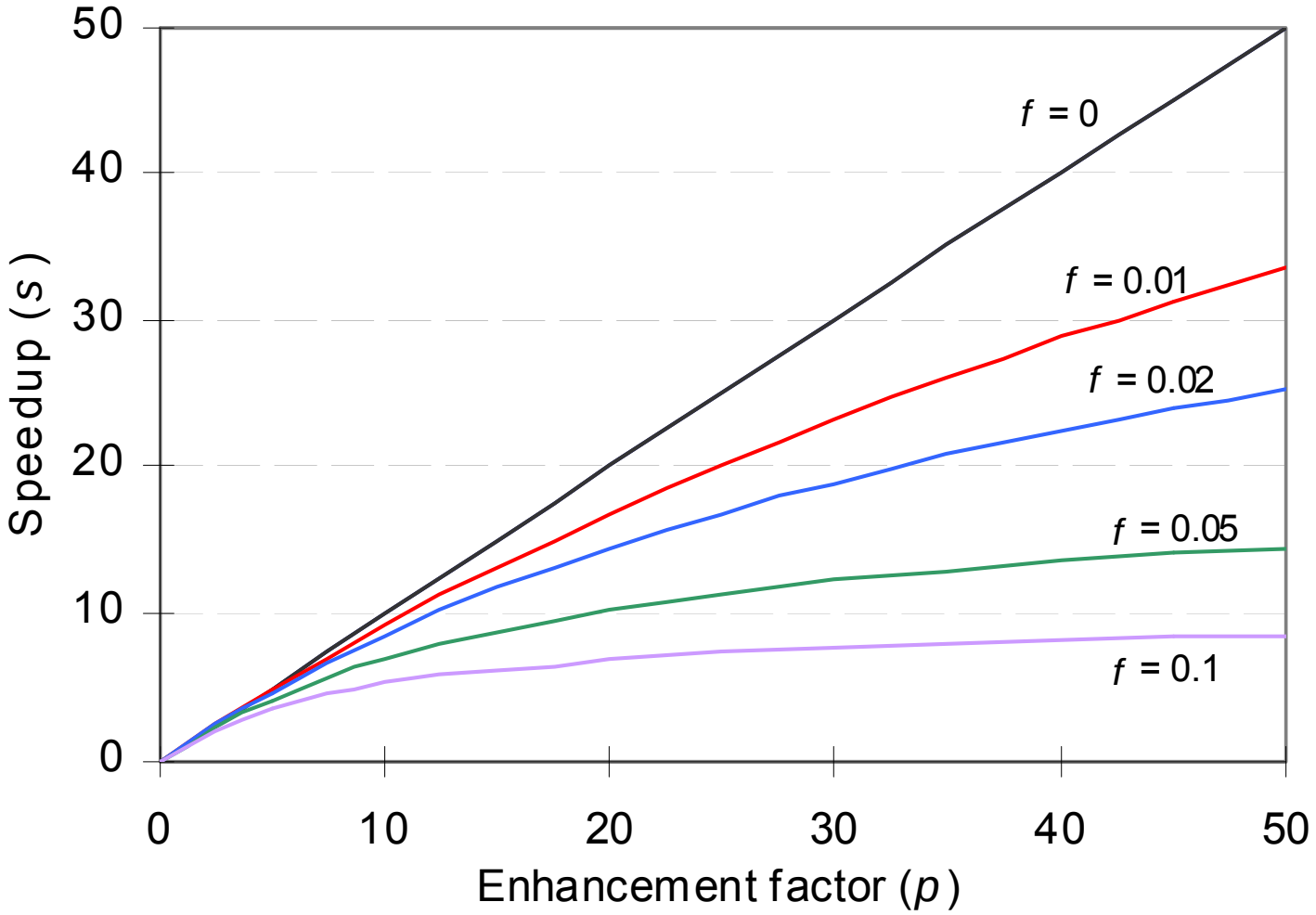
10,000 MIPS (32-bit processor)



Energy Consumption is Getting out of Hand



Amdahl's Law



f **= fraction unaffected**

p **= speedup of the rest**

$$s = \frac{1}{f + (1 - f)/p} \leq \min(p, 1/f)$$

Amdahl's System Balance Rules of Thumb

The need for high-capacity, high-throughput secondary (disk) memory

Processor speed	RAM size	Disk I/O rate	Number of disks	Disk capacity	Number of disks
1 GIPS	1 GB	100 MB/s	1	100 GB	1
1 TIPS	1 TB	100 GB/s	1000	100 TB	100
1 PIPS	1 PB	100 TB/s	1 Million	100 PB	100 000
1 EIPS	1 EB	100 PB/s	1 Billion	100 EB	100 Million

1 RAM byte for each IPS

1 I/O bit per sec for each IPS

100 disk bytes for each RAM byte

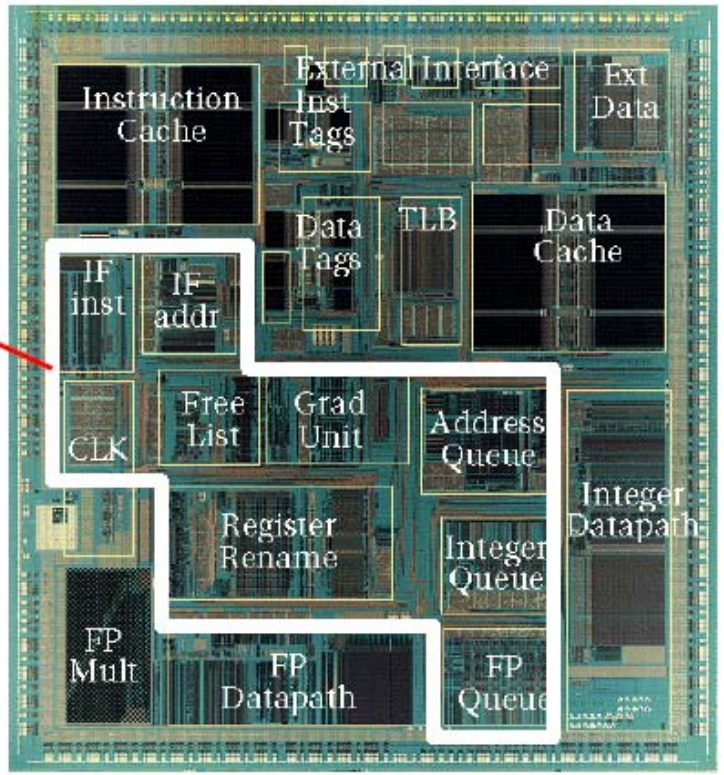
G Giga
T Tera
P Peta
E Exa

Design Space for Superscalar Pipelines

Front end:	**In-order or out-of-order**
Instr. issue:	**In-order or out-of-order**
Writeback:	**In-order or out-of-order**
Commit:	**In-order or out-of-order**

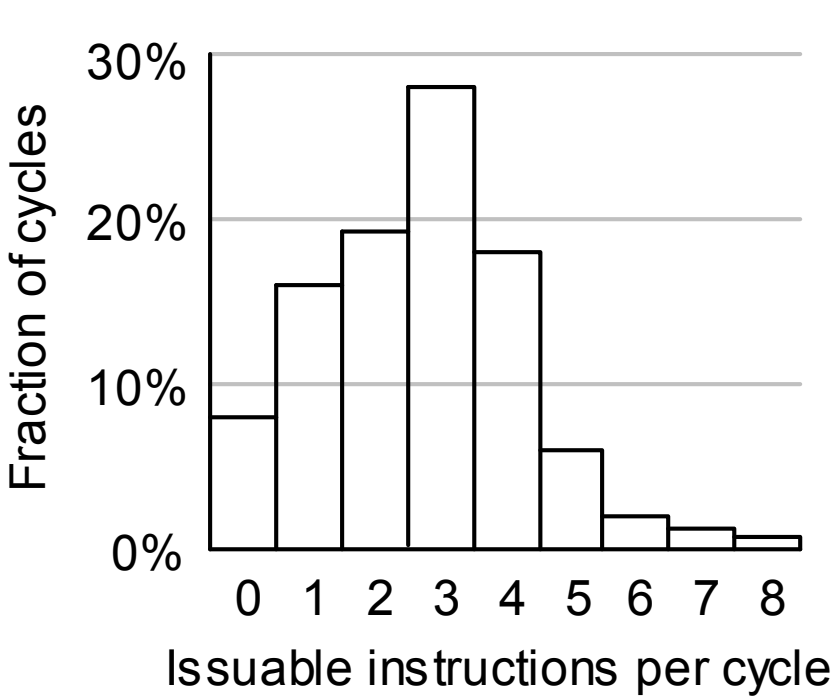
The more OoO stages, the higher the complexity

Example of complexity due to out-of-order processing: MIPS R10000

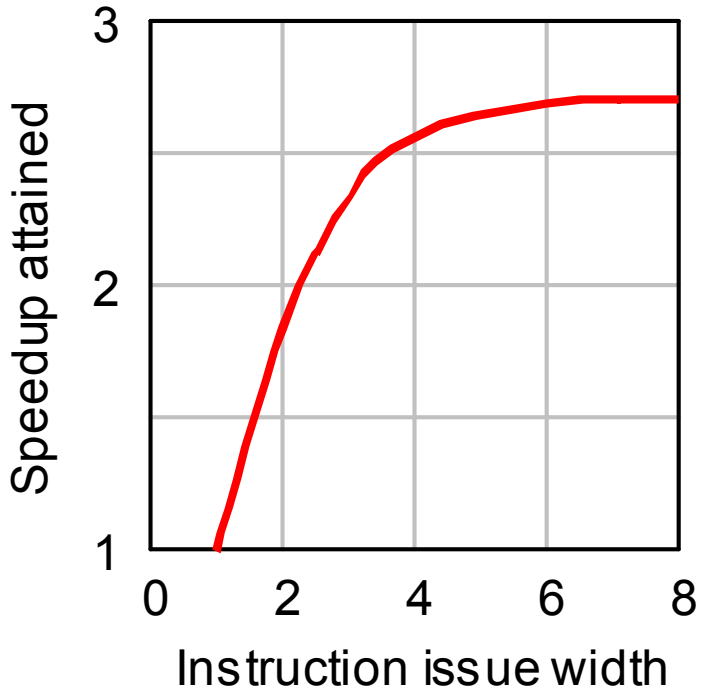


Source: Ahi, A. et al., "MIPS R10000 Superscalar Microprocessor," ***Proc. Hot Chips Conf.*****, 1995.**

Instruction-Level Parallelism



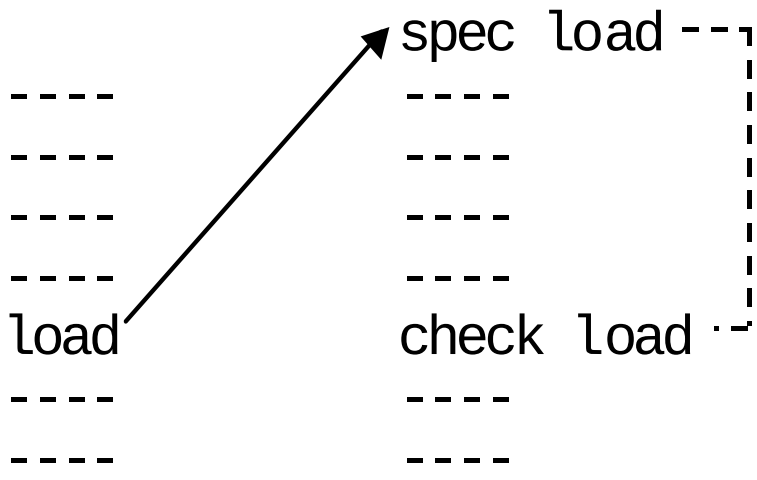
(a)



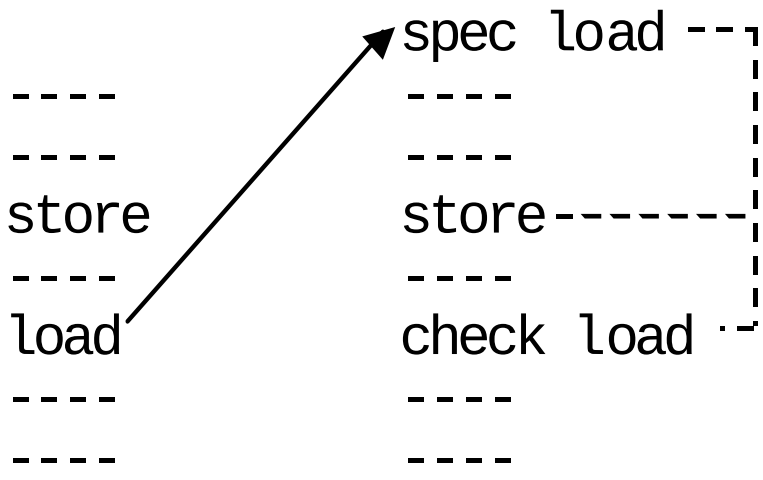
(b)

Available instruction-level parallelism and the speedup due to multiple instruction issue in superscalar processors [John91].

Speculative Loads



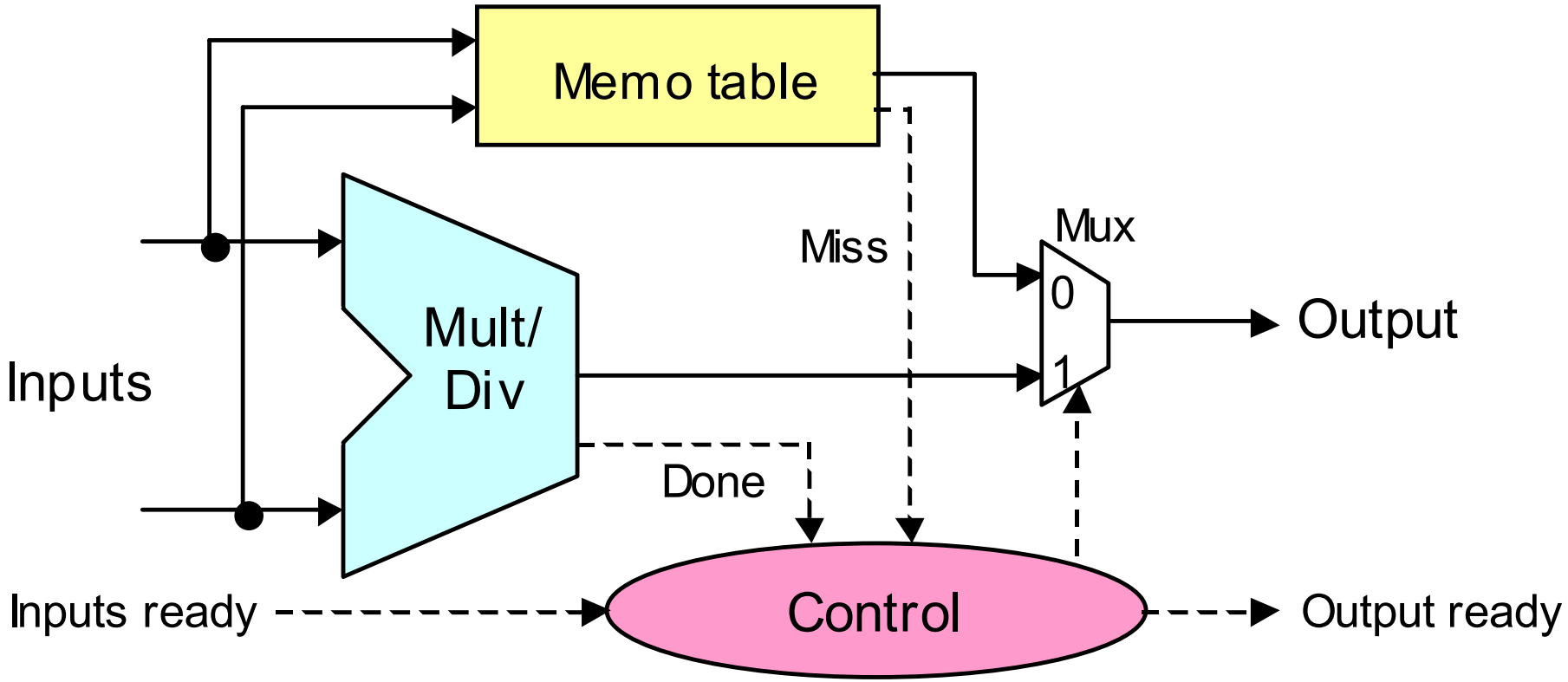
(a) Control speculation



(b) Data speculation

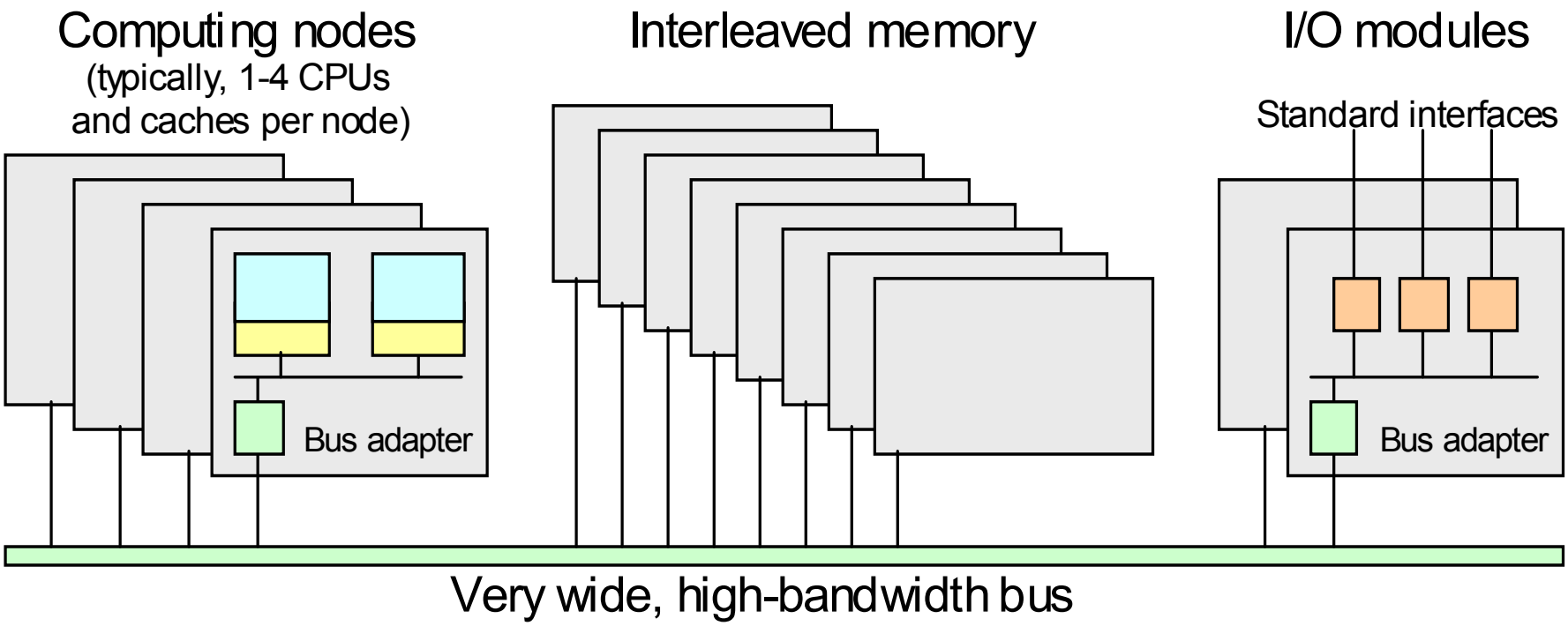
Examples of software speculation in IA-64.

Value Prediction



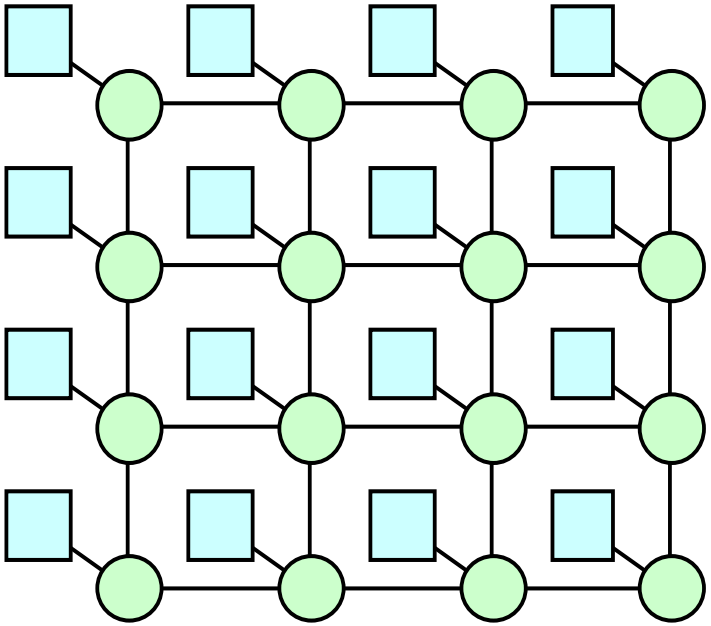
Value prediction for multiplication or division via a memo table.

Implementing Symmetric Multiprocessors

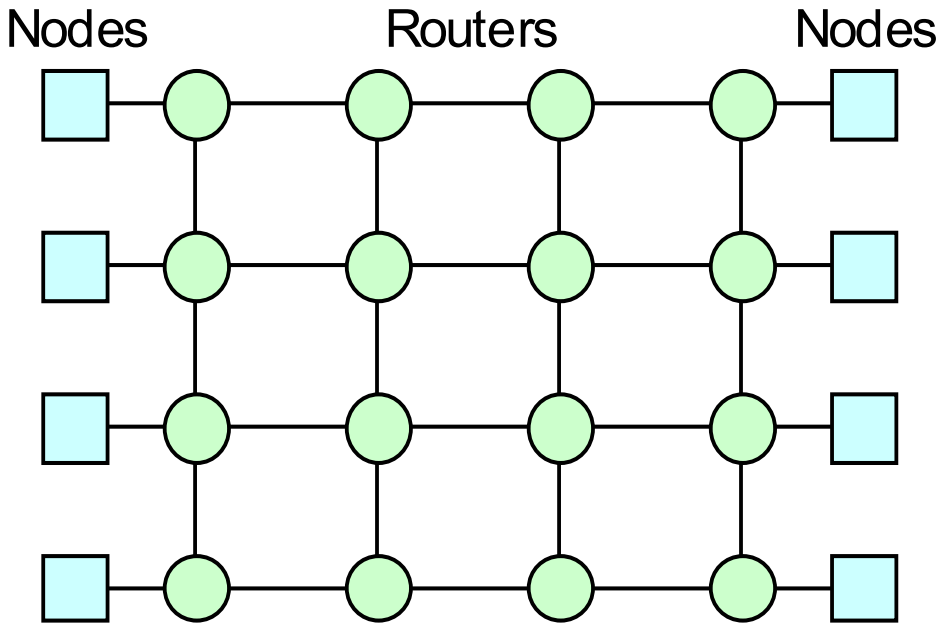


Structure of a generic bus-based symmetric multiprocessor.

Interconnection Networks



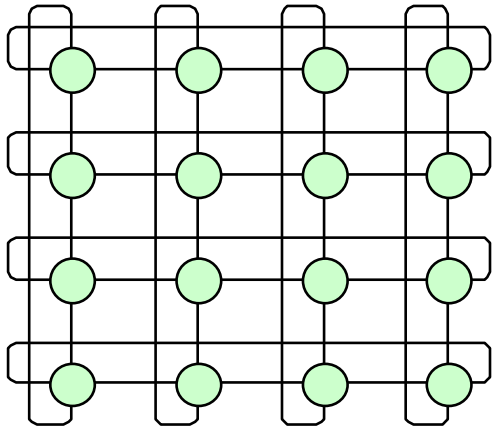
(a) Direct network



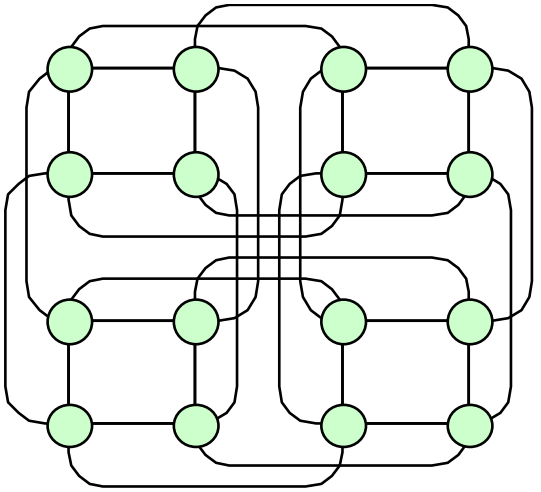
(b) Indirect network

Examples of direct and indirect interconnection networks.

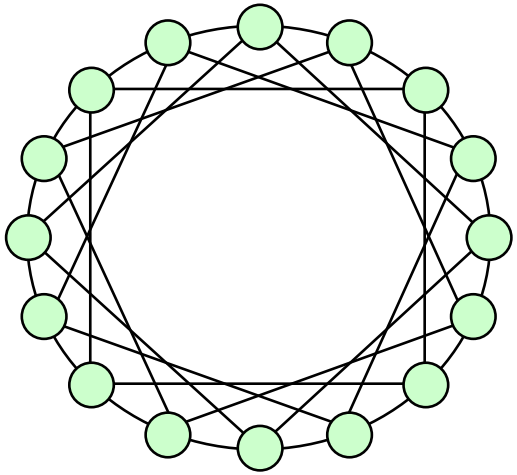
Direct Interconnection Networks



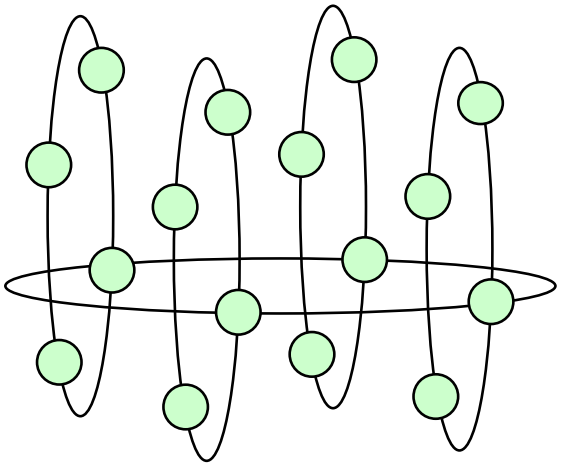
(a) 2D torus



(b) 4D hypercube



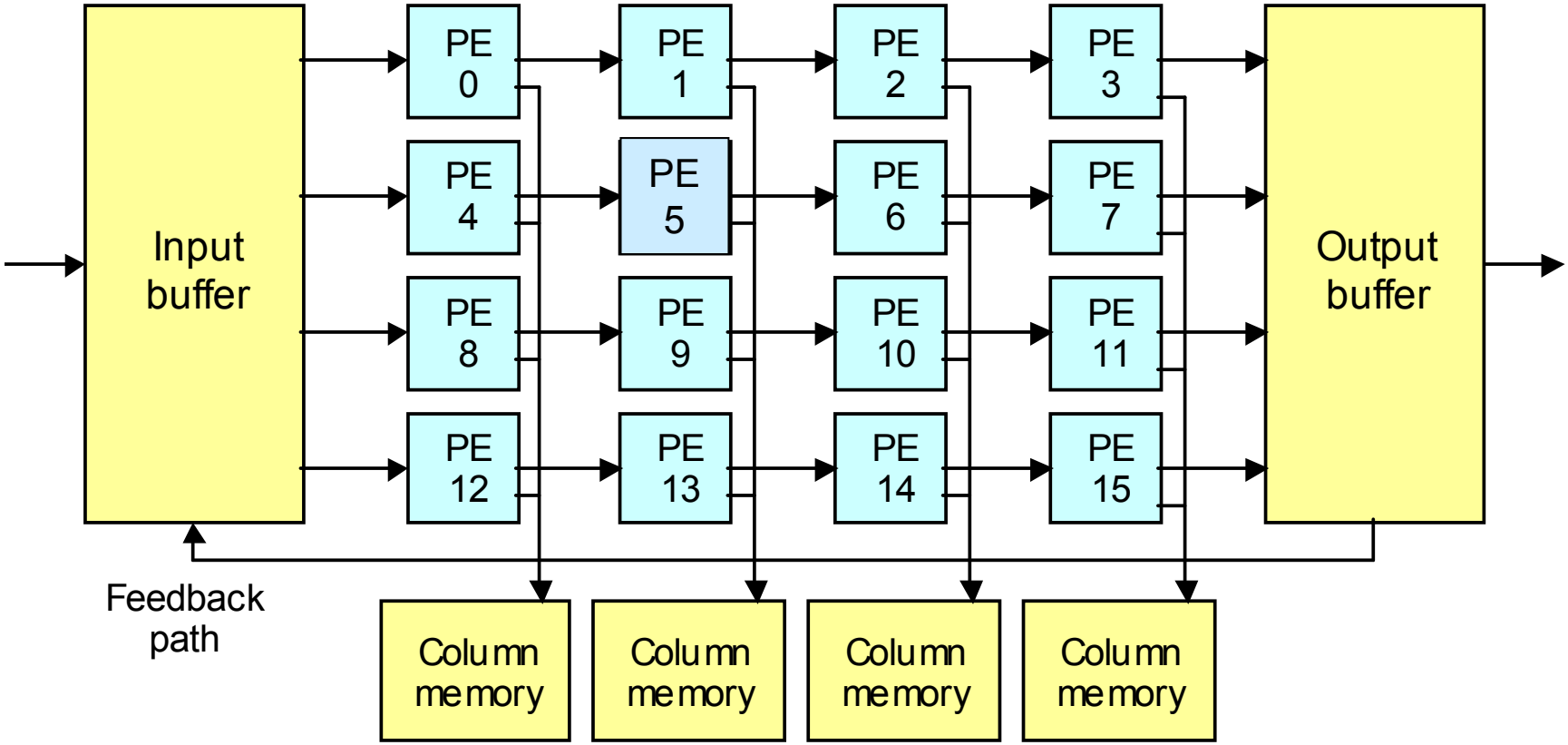
(c) Chordal ring



(d) Ring of rings

A sampling of common direct interconnection networks.
Only routers are shown; a computing node is implicit for each router.

Graphic Processors, Network Processors, …



Simplified block diagram of Toaster2, Cisco Systems' network processor.

Computing in the Cloud

Computational resources, both hardware and software, are provided by, and managed within, the cloud

Users pay a fee for access

Managing / upgrading is much more efficient in large, centralized facilities (warehouse-sized data centers or server farms)

Image from Wikipedia

This is a natural continuation of the outsourcing trend for special services, so that companies can focus their energies on their main business