Eight Key Ideas in Computer Architecture, from Eight 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). An updated version of the talk was prepared and used within B. Parhami’s suite of lectures for IEEE Computer Society’s Distinguished Visitors Program, 2021-2023. All rights reserved for the author. ©2021 Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>Mar. 2016</td>
<td>July 2017</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
My personal academic journey:

- Graduated 1968
- PhD in 1973
- Children ages 28-37
- Fifty years in academia (2023)
Some of the material in this talk come from, or will appear in updated versions of, my two computer architecture textbooks.
Eight Key Ideas in Computer Architecture, from Eight 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 eight decades, from the 1940s to the 2010s, and show how these ideas, when connected to each other and allowed to interact and cross-fertilize, produced 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 offer predictions for what to expect in the 2020s and beyond.
We Are Fond of Making Top-\(n\) Lists

and even more fond of arguing over them!
The Eight Key Ideas

- Stored program
- Parallel processing
- Cache memory
- Pipelining
- Microprogramming
- FPGAs
- GPUs
- Specialization

Come back in 2040, for my top-10 list!
Background: 1820s-1930s

Difference Engine

Analytical Engine

Program (Instructions)

Data (Variable values)

Punched Cards
Difference Engine: Fixed Program

Babbage’s Difference Engine 2

2nd-degree polynomial evaluation
Babbage used 7th-degree $f(x)$
### Analytical Engine: Programmable

**Ada Lovelace,** world’s first programmer

**Sample program >**

<table>
<thead>
<tr>
<th>Number of Operation</th>
<th>Nature of Operation</th>
<th>Variables acted upon</th>
<th>Variables receiving results</th>
<th>Statement of Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>×</td>
<td>V₁₂ × V₁₃</td>
<td>V₄, V₅, V₆, V₇</td>
<td>= 2n × 1</td>
</tr>
<tr>
<td>2</td>
<td>−</td>
<td>V₁₄ − V₁₅</td>
<td>V₄</td>
<td>= 2n − 1</td>
</tr>
<tr>
<td>3</td>
<td>+</td>
<td>V₁₆ + V₁₇</td>
<td>V₈</td>
<td>= 2n + 1</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>V₁₈ + V₁₉</td>
<td>V₈, V₁₀</td>
<td>= 2n − 1 + 1</td>
</tr>
<tr>
<td>5</td>
<td>+</td>
<td>V₂₁ + V₂₂</td>
<td>V₈, V₁₀</td>
<td>= 2n − 1 + 1</td>
</tr>
<tr>
<td>6</td>
<td>−</td>
<td>V₂₃ − V₂₄</td>
<td>V₈, V₁₀</td>
<td>= 2n − 1 − 1</td>
</tr>
<tr>
<td>7</td>
<td>−</td>
<td>V₂₅ − V₂₆</td>
<td>V₈, V₁₀</td>
<td>= n − 1 (n = 5)</td>
</tr>
<tr>
<td>8</td>
<td>+</td>
<td>V₂₇ + V₂₈</td>
<td>V₈</td>
<td>= 2n + 0 = 2</td>
</tr>
<tr>
<td>9</td>
<td>+</td>
<td>V₂₉ + V₃₀</td>
<td>V₈</td>
<td>= 2n + 1 = A₃</td>
</tr>
<tr>
<td>10</td>
<td>×</td>
<td>V₃₁ × V₃₂</td>
<td>V₈, V₁₀</td>
<td>= n × n = BₙAₙ</td>
</tr>
<tr>
<td>11</td>
<td>+</td>
<td>V₃₃ + V₃₄</td>
<td>V₈, V₁₀</td>
<td>= 2(2n − 1) + 1 = Aₙ</td>
</tr>
<tr>
<td>12</td>
<td>−</td>
<td>V₃₅ − V₃₆</td>
<td>V₈, V₁₀</td>
<td>= n − 2(n = 2)</td>
</tr>
<tr>
<td>13</td>
<td>−</td>
<td>V₃₇ − V₃₈</td>
<td>V₈, V₁₀</td>
<td>= 2n − 1</td>
</tr>
<tr>
<td>14</td>
<td>+</td>
<td>V₃₉ + V₄₀</td>
<td>V₈</td>
<td>= 2n + 1 = 3</td>
</tr>
<tr>
<td>15</td>
<td>×</td>
<td>V₴₁ × V₴₂</td>
<td>V₈</td>
<td>= 2n − 1</td>
</tr>
<tr>
<td>16</td>
<td>+</td>
<td>V₴₃ + V₴₄</td>
<td>V₈</td>
<td>= 2n − 1</td>
</tr>
<tr>
<td>17</td>
<td>−</td>
<td>V₴₅ − V₴₆</td>
<td>V₈</td>
<td>= 2n − 1</td>
</tr>
<tr>
<td>18</td>
<td>+</td>
<td>V₴₇ + V₴₈</td>
<td>V₈</td>
<td>= 3n + 1 − 4</td>
</tr>
<tr>
<td>19</td>
<td>×</td>
<td>V₴₉ × V₵₀</td>
<td>V₈</td>
<td>= 2n − 2</td>
</tr>
<tr>
<td>20</td>
<td>+</td>
<td>V₵₁ + V₵₂</td>
<td>V₈</td>
<td>= 4n − 2</td>
</tr>
<tr>
<td>21</td>
<td>×</td>
<td>V₵₃ × V₵₄</td>
<td>V₈</td>
<td>= 2n − 2 + 1 = Aₙ</td>
</tr>
<tr>
<td>22</td>
<td>+</td>
<td>V₵₅ + V₵₆</td>
<td>V₈</td>
<td>= 3n + 1 − 4</td>
</tr>
<tr>
<td>23</td>
<td>−</td>
<td>V₵₇ − V₵₈</td>
<td>V₈</td>
<td>= 2n − 2 + 1 = Aₙ</td>
</tr>
</tbody>
</table>

*Here follows a repetition*
The Programs of Charles Babbage

Electromechanical and Plug-Programmable Computing Machines

Punched-card device

Turing's Colossus

Zuse's Z3

ENIAC
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

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

Notes for State 5:
- % 0 for j or jal, 1 for syscall, don't-care for other instr's
- @ 0 for j, jal, and syscall, 1 for jr, 2 for branches
- # 1 for j, jr, jal, and syscall, ALUZero() for beq (bne), bit 31 of ALUout for bltz

For jal, RegDst = 2, RegInSrc = 1, RegWrite = 1

Start

State 0
- Inst’Data = 0
- MemRead = 1
- IRWrite = 1
- ALUSrcX = 0
- ALUSrcY = 0
- ALUFnc = ‘+’
- PCSrc = 3
- PCWrite = 1

State 1
- ALUSrcX = 0
- ALUSrcY = 3
- ALUFnc = ‘+’

State 2
- ALUSrcX = 1
- ALUSrcY = 2
- ALUFnc = ‘+’

State 3
- Inst’Data = 1
- MemRead = 1

State 4
- RegDst = 0
- RegInSrc = 0
- RegWrite = 1

State 5
- ALUSrcX = 1
- ALUSrcY = 1
- ALUFnc = ‘-’
- JumpAddr = %
- PCSrc = @
- PCWrite = #

State 6
- Inst’Data = 1
- MemWrite = 1

State 7
- ALUSrcX = 1
- ALUSrcY = 1 or 2
- ALUFnc = Varies

State 8
- RegDst = 0 or 1
- RegInSrc = 1
- RegWrite = 1

Notes for State 7:
- ALUFnc is determined based on the op and fn fields

For jal, RegDst = 2, RegInSrc = 1, RegWrite = 1
The Birth of Microprogramming

The control state machine resembles a program (microprogram) comprised of instructions (microinstructions) and sequencing. Every \( \mu \)instruction contains a branch field.

Maurice V. Wilkes (1913-2010)
Microprogramming Implementation

Each microinstruction controls the data path for one clock cycle

Diagram showing the sequence of states and transitions for different instructions and operations.
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.

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
- Cache coherence

**Distributed shared memory or message-passing architecture**
- Scalable network performance
- Flexible and more robust
- Memory consistency model
Warehouse-Sized Data Centers

**COOLING:** High-efficiency water-based cooling systems—less energy-intensive than traditional chillers—circulate cold water through the containers to remove heat, eliminating the need for air-conditioned rooms.

**STRUCTURE:** A 24,000-square-meter facility houses 400 containers. Delivered by trucks, the containers attach to a spine infrastructure that feeds network connectivity, power, and water. The data center has no conventional raised floors.

**POWER:** Two power substations feed a total of 300 megawatts to the data center, with 200 MW used for computing equipment and 100 MW for cooling and electrical losses. Batteries and generators provide backup power.

**CONTAINER:** Each 67.5-cubic-meter container houses 2,500 servers, about 10 times as many as conventional data centers pack in the same space. Each container integrates computing, networking, power, and cooling systems.

Image from *IEEE Spectrum*, June 2009
The Shrinking Supercomputer

Sandia Lab's ASCI Red, 1997
150 sq. meters, 800 kw

Both perform at ~2 TFLOPS

Sony Playstation, 2006
0.08 sq. meter, < 0.2 kw

iPhone 13, 2021
6 CPU + 5 GPU + 16 NN
0.01 sq. meter, < 0.5 w

Giga = 10^9
GFLOPS on desktop: TFLOPS PFLOPS
Apple Macintosh, with G4 processor

Tera = 10^{12}
TFLOPS in supercomputer center: PFLOPS EFLOPS
1152-proc IBM RS/6000 SP; Cray T3E

Peta = 10^{15}
PFLOPS on the drawing board: Exa = 10^{18} EFLOPS Zeta = 10^{21}
1M-processor IBM Blue Gene (2005?)
32 proc/chip, 64 chips/board, 8 boards/tower, 64 towers
Processor: 8 threads, on-chip memory, no data cache
Chip: defect-tolerant, row/column rings in a 6 × 6 array
Board: 8 × 8 chip grid organized as 4 × 4 × 4 cube
Tower: Boards linked to 4 neighbors in adjacent towers
System: 32×32×32 cube of chips, 1.5 MW (water-cooled)
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 caches
- von Neumann arch higher up
- Many other caches in system besides processor cashes
Hierarchical memory provides the illusion that high speed and large size are achieved simultaneously.

<table>
<thead>
<tr>
<th>Capacity</th>
<th>Access latency</th>
<th>Cost per GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>100s B</td>
<td>ns</td>
<td>$Millions</td>
</tr>
<tr>
<td>10s KB</td>
<td>a few ns</td>
<td>$100s Ks</td>
</tr>
<tr>
<td>MBs</td>
<td>10s ns</td>
<td>$10s Ks</td>
</tr>
<tr>
<td>100s MB</td>
<td>100s ns</td>
<td>$1000s</td>
</tr>
<tr>
<td>10s GB</td>
<td>10s ms</td>
<td>$10s</td>
</tr>
<tr>
<td>TBs</td>
<td>min+</td>
<td>$1s</td>
</tr>
</tbody>
</table>
Cache Memory Analogy

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

Example: Tax docs
Temporal locality
Spatial locality

Main memory

Register file

Access cabinet in 30 s

Access drawer in 5 s

Access desktop in 2 s

Cache memory
Hit/Miss Rate, and Effective Cycle Time

Cache is transparent to user; transfers occur automatically

Data is in the cache fraction $h$ of the time (say, hit rate of 98%)

Go to main $1 - h$ of the time (say, cache miss rate of 2%)

One level of cache with hit rate $h$

$$C_{\text{eff}} = hC_{\text{fast}} + (1 - h)(C_{\text{slow}} + C_{\text{fast}}) = C_{\text{fast}} + (1 - h)C_{\text{slow}}$$
The Locality Principle

Addresses

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

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

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

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.
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

1. Head movement from current position to desired cylinder: Seek time (0-10s ms)
2. Disk rotation until the desired sector arrives under the head: Rotational latency (0-10s ms)
3. Disk rotation until sector has passed under the head: Data transfer time (< 1 ms)

Entire track copied into fast cache
1980s: Pipelining

An important form of parallelism that is given its own name

Used from early days of digital circuits in various forms

![Diagram of pipelining process]

Latch positions in a four-stage pipeline

\[
\sqrt{\frac{(a + b)\ c\ d}{e - f}}
\]

Pipelining period

Latency

Output available

\[ t = 0 \]

Time
Vector Processor Implementation

From scalar registers

Function unit 1 pipeline

Function unit 2 pipeline

Function unit 3 pipeline

Forwarding muxes

To and from memory unit

Load unit A
Load unit B
Store unit

Vector register file
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

Task dimension

Instr 1
Instr 2
Instr 3
Instr 4
Instr 5

Instr cache
Reg file
ALU
Data cache
Reg file

Cycle 1
Cycle 2
Cycle 3
Cycle 4
Cycle 5
Cycle 6
Cycle 7
Cycle 8
Cycle 9

Time dimension

Sep. 2022
Eight Key Ideas in Computer Architecture
Pipeline Stalls or Bubbles

Data dependency and its possible resolution via forwarding

$5 = \text{Instr cache} + \text{Reg file} + \text{ALU}$

$8 = \text{Instr cache} + \text{Reg file} + \text{ALU}$

$9 = \text{Instr cache} + \text{Reg file} + \text{ALU}$

sw $9, 0(\text{Reg file})$
Problems Arising from Deeper Pipelines

Forwarding more complex and not always workable Interlocking/stalling mechanisms needed to prevent errors
Branching and Other Complex Pipelines

Stage 1  Stage 2  Stage 3  Stage 4  Stage 5  Variable # of stages  Stage q−2  Stage q−1  Stage q

Instr cache  
Instruction fetch  

Instr decode  
Operand prep  

Instr issue  

Function unit 1

Function unit 2

Function unit 3

Retirement & commit stages

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

Sep. 2022

Eight Key Ideas in Computer Architecture
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 Eight Key Ideas

- Stored program
- Microprogramming
- Parallel processing
- Cache memory
- Pipelining
- FPGAs
- GPUs
- Specialization

Methodological advances
Performance improvements
## Innovations for Improved Performance


Computer performance grew by a factor of about 10000 between 1985 and 2010. 100 due to faster technology. 100 due to better architecture.

Available computing power ca. 2010:
- TFLOPS on desktop
- PFLOPS in supercomputer center
- EFLOPS on drawing board

<table>
<thead>
<tr>
<th>Architectural method</th>
<th>Improvement factor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Established methods</strong></td>
<td></td>
</tr>
<tr>
<td>1. Pipelining (and superpipelining)</td>
<td>3-8 √</td>
</tr>
<tr>
<td>2. Cache memory, 2-3 levels</td>
<td>2-5 √</td>
</tr>
<tr>
<td>3. RISC and related ideas</td>
<td>2-3 √</td>
</tr>
<tr>
<td>4. Multiple instruction issue (superscalar)</td>
<td>2-3 √</td>
</tr>
<tr>
<td>5. ISA extensions (e.g., for multimedia)</td>
<td>1-3 √</td>
</tr>
<tr>
<td><strong>Newer methods</strong></td>
<td></td>
</tr>
<tr>
<td>6. Multithreading (super-, hyper-)</td>
<td>2-5 ?</td>
</tr>
<tr>
<td>7. Speculation and value prediction</td>
<td>2-3 ?</td>
</tr>
<tr>
<td>8. Hardware acceleration [e.g., GPU]</td>
<td>2-10 ?</td>
</tr>
<tr>
<td>9. Vector and array processing</td>
<td>2-10 ?</td>
</tr>
<tr>
<td>10. Parallel/distributed computing</td>
<td>2-1000s ?</td>
</tr>
</tbody>
</table>

Established methods: Previously discussed

Newer methods: Covered in Part VII

Available computing power ca. 2010:
- TFLOPS on desktop
- PFLOPS in supercomputer center
- EFLOPS on drawing board

Computer performance grew by a factor of about 10000 between 1985 and 2010. 100 due to faster technology. 100 due to better architecture.
Shares of Technology and Architecture in Processor Performance Improvement

Overall Performance Improvement (SPECINT, relative to 386)

Gate Speed Improvement (FO4, relative to 386)

Feature Size (μm)

~1985

--------- 1995-2000 ---------

~2005

~2010

Much of arch. improvements already achieved

Source: “CPU DB: Recording Microprocessor History,” CACM, April 2012.
2010s: Specialization

Specialization entails high initial cost and creates a danger of obsolescence.

Hennessy/Patterson’s Turing Lecture

Recipients of 2017 ACM Turing Award heralded in their 2018 joint lecture “A New Golden Age of Computer Architecture” in which domain-specific hardware dominates

Further progress in “conventional” architectures impeded by slowing Moore scaling, end of Dennard scaling (power wall), diminishing return at high energy cost for ILP, multi-core, etc., sorry state of security (software-only solutions not working)

- Domain-specific languages along with domain-specific arch’s

- Agile hardware-development methodologies (for, otherwise, developing many domain-specific systems would be costly)

- Free open architectures and open-source implementations
Shades of Domain-Specificity

Domain-specificity is achieved via ASIC design

ASIC types
- Full-custom
- Semi-custom
- Programmable

Nothing new: ASICs have been around since the 1970s

Controller chips for DVD players, automotive electronics, phones

Generality lowers per-unit cost but hurts performance

ASIC’s popularity: Standards & high-volume products
Higher volume $\rightarrow$ We can afford to make it more specific
Google’s Tensor-Processing Unit

AI-accelerator ASIC aimed at machine-learning applications
Developed in 2015 and released for third-party use in 2018
Pixel Neural Core announced in 2019 for Pixel 4 phone

<table>
<thead>
<tr>
<th>Google TPU</th>
<th>Integer</th>
<th>FLP</th>
<th>Table from Wikipedia</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TPUv1</td>
<td>TPUv2</td>
<td>TPUv3</td>
</tr>
<tr>
<td>Date Introduced</td>
<td>2016</td>
<td>2017</td>
<td>2018</td>
</tr>
<tr>
<td>Process Node</td>
<td>28 nm</td>
<td>16 nm</td>
<td>16 nm</td>
</tr>
<tr>
<td>Die Size (mm²)</td>
<td>331</td>
<td>&lt; 625</td>
<td>&lt; 700</td>
</tr>
<tr>
<td>On chip memory (MiB)</td>
<td>28</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Clock Speed (MHz)</td>
<td>700</td>
<td>700</td>
<td>940</td>
</tr>
<tr>
<td>Memory (GB)</td>
<td>8GB DDR3</td>
<td>16GB HBM</td>
<td>32GB HBM</td>
</tr>
<tr>
<td>TDP (W)</td>
<td>75</td>
<td>280</td>
<td>450</td>
</tr>
<tr>
<td>TOPS</td>
<td>23</td>
<td>45</td>
<td>90</td>
</tr>
</tbody>
</table>

NVIDIA: Tensor Core → 2x → 2x → 2x?
Tensor-Processing Unit Architecture

Optimized for matrix multiplication, a key computation in ML
Faster than CPU/GPU, 15-30x; More energy-frugal, 30-80x

Figures from: https://research.google/pubs/pub46078/
Perils of Technology Forecasting

Nobel Laureate Physicist Niels Bohr:
“Prediction is difficult, especially if it’s about the future.”
[Paraphrased by Yogi Berra in his famous version]

Anonymous quotes about forecasting:
“Forecasting is the art of saying what will happen, and then explaining why it didn’t.”
“There are two kinds of forecasts: lucky and wrong.”
“A good forecaster is not smarter than everyone else; he merely has his ignorance better organized.”

Henri Poincare was more positive:
“It is far better to foresee even without certainty than not to foresee at all.”
2020s 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)
Three Directions for Future Systems

Survival

Agility

Agency

Self-Organizing
Self-Improving

Managed
Autonomous

Self-Healing
Self-Sustaining
Trends in Processor Chip Density, Performance, Clock Speed, Power, and Number of Cores

Original data up to 2010 collected/plotted by M. Horowitz et al.; Data for 2010-2017 extension collected by K. Rupp
### The Quest for Higher Performance

**Top-Five Supercomputers in November 2020**

<table>
<thead>
<tr>
<th>Rank (previous)</th>
<th>Rmax Rpeak (PFLOPS)</th>
<th>Name</th>
<th>Model</th>
<th>CPU cores</th>
<th>Accelerator (e.g. GPU)</th>
<th>Interconnect</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>442.010 537.212</td>
<td>Fugaku</td>
<td>Supercomputer Fugaku</td>
<td>158,976 × 48 A64FX @2.2 GHz</td>
<td>0</td>
<td>Tofu interconnect D</td>
<td>Fujitsu</td>
</tr>
<tr>
<td>2 (1)</td>
<td>148.600 200.795</td>
<td>Summit</td>
<td>IBM Power System AC922</td>
<td>9,216 × 22 POWER9 @3.07 GHz</td>
<td>27,648 × 80 Tesla V100</td>
<td>InfiniBand EDR</td>
<td>IBM</td>
</tr>
<tr>
<td>3 (2)</td>
<td>94.640 125.712</td>
<td>Sierra</td>
<td>IBM Power System S922LC</td>
<td>8,640 × 22 POWER9 @3.1 GHz</td>
<td>17,280 × 80 Tesla V100</td>
<td>InfiniBand EDR</td>
<td>IBM</td>
</tr>
<tr>
<td>4 (3)</td>
<td>93.015 125.436</td>
<td>Sunway TaihuLight</td>
<td>Sunway MPP</td>
<td>40,960 × 260 SW26010 @1.45 GHz</td>
<td>0</td>
<td>Sunway[26]</td>
<td>NRCPC</td>
</tr>
<tr>
<td>5 (7)</td>
<td>63.460 79.215</td>
<td>Selene</td>
<td>Nvidia</td>
<td>1,120 × 64 Epyc 7742 @2.25 GHz</td>
<td>4,480 × 108 Ampere A100</td>
<td>Mellanox HDR Infiniband</td>
<td>Nvidia</td>
</tr>
</tbody>
</table>
The Quest for Higher Performance
June 2022 update (http://www.top500.org)

Top Supercomputer: 1+ exaflops performance
- Frontier system at US Oak Ridge National Lab.
- Based on the latest HPE Cray EX235a architecture
- Equipped with AMD EPYC 64C 2 GHz processors
- Number of Cores ~8.7 million
- Power efficiency rating of ~52 gigaflops/W

Top "Green" Supercomputer: ~20 petaflops
- Frontier Test & Development System at ORNL
- A subset of the top supercomputer above
- Number of cores ~120K
- Power efficiency rating of ~63 gigaflops/W
- The top supercomputer above is #2 on the Green500 list
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/
Peak Performance of Supercomputers

Past and Current Performance Trends

Intel 4004: The first muP (1971)
0.06 MIPS (4-bit processor)

8008
8080
8084

8-bit

Intel Pentium 4, circa 2005
10,000 MIPS (32-bit processor)

80386
80486

Pentium, MMX

Pentium Pro, II

Pentium III, M

Celeron

32-bit

8086
8088

80186
80188

80286

16-bit
Energy Consumption is Getting out of Hand
Amdahl’s Law

\[ f = \text{fraction unaffected} \]
\[ p = \text{speedup of the rest} \]
\[ s = \frac{1}{f + (1 - f)/p} \leq \min(p, 1/f) \]
## Amdahl’s System Balance Rules of Thumb

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

<table>
<thead>
<tr>
<th>Processor speed</th>
<th>RAM size</th>
<th>Disk I/O rate</th>
<th>Number of disks</th>
<th>Disk capacity</th>
<th>Number of disks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GIPS</td>
<td>1 GB</td>
<td>100 MB/s</td>
<td>1</td>
<td>100 GB</td>
<td>1</td>
</tr>
<tr>
<td>1 TIPS</td>
<td>1 TB</td>
<td>100 GB/s</td>
<td>1000</td>
<td>100 TB</td>
<td>100</td>
</tr>
<tr>
<td>1 PIPS</td>
<td>1 PB</td>
<td>100 TB/s</td>
<td>1 Million</td>
<td>100 PB</td>
<td>100 000</td>
</tr>
<tr>
<td>1 EIPS</td>
<td>1 EB</td>
<td>100 PB/s</td>
<td>1 Billion</td>
<td>100 EB</td>
<td>100 Million</td>
</tr>
</tbody>
</table>

1 RAM byte for each IPS  
1 I/O bit per sec for each IPS  
100 disk bytes for each RAM byte
The Flynn/Johnson Classification

Flynn’s categories

<table>
<thead>
<tr>
<th>Single instr stream</th>
<th>Multiple instr streams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single data stream</td>
<td>Multiple data streams</td>
</tr>
<tr>
<td>SISD Uniprocessors</td>
<td>SIMD Array or vector processors</td>
</tr>
<tr>
<td>MISD Rarely used</td>
<td>MIMD Multiproc’s or multicomputers</td>
</tr>
</tbody>
</table>

Johnson’s expansion

<table>
<thead>
<tr>
<th>Shared variables</th>
<th>Message passing</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMSV Shared-memory multiprocessors</td>
<td>GMMP Rarely used</td>
</tr>
<tr>
<td>DMSV Distributed shared memory</td>
<td>DMMP Distrib-memory multicomputers</td>
</tr>
</tbody>
</table>
Shared-Control Systems

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
Implementing Symmetric Multiprocessors

Computing nodes
(typically, 1-4 CPUs and caches per node)

Interleaved memory

I/O modules

Very wide, high-bandwidth bus

Structure of a generic bus-based symmetric multiprocessor.
Interconnection Networks

(a) Direct network
(b) Indirect network

Examples of direct and indirect interconnection networks.
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

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 for multiplication or division via a memo table.
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).

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