COEN-4730/EECE-5730 Computer Architecture

Lecture 1
Introduction
(Ch.1)

Cristinel Ababei
***D**ept. of Electrical and Computer Engineering*
MARQUETTE UNIVERSITY
BE THE DIFFERENCE.

Credits: Slides adapted from presentations of Sudeep Pasricha and others: Kubiatowicz, Patterson, Mutlu, Elsevier

Administrative

- Discussion of Syllabus
- Grading policies, attendance, HW, etc.
- Course websites:
  - D2L:
    - https://d2l.mu.edu/d2l/login
  - Public:
    - http://dejazzer.com/coen4730/index.html

Outline

- What is and what does a computer architect?
- Classes of computers
- Trends in technology
- Defining computer architecture
- New definition of computer architecture

"You can't really understand what is going on now unless you understand what came before." --SJ

3

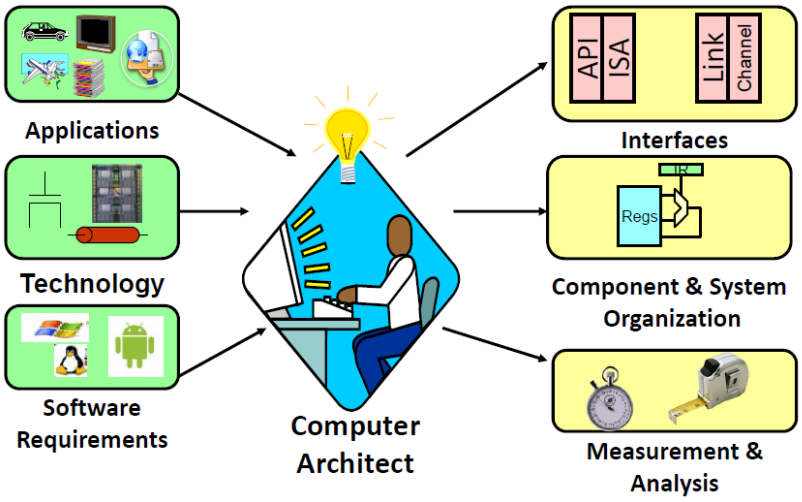
3

Why Study Computer Architecture?

- Understand why computers work the way they do
- Make computers faster, cheaper, smaller, more reliable
  - By exploiting architectural advances and changes in underlying technology/circuits
- Adapt the computing stack to technology trends
  - Innovation in software is built into trends and changes in computer architecture
- Enable new applications
  - Life-like 3D visualization 20 years ago?
  - Virtual/augmented reality?
  - High performance mobile computing?

4

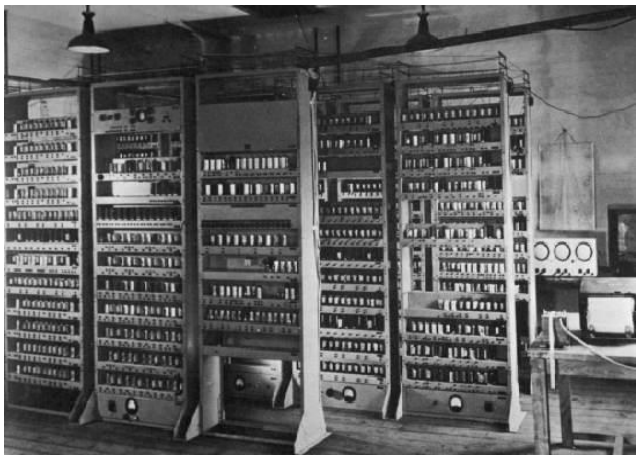
What is a Computer Architect?



5

5

Computing Devices Then…



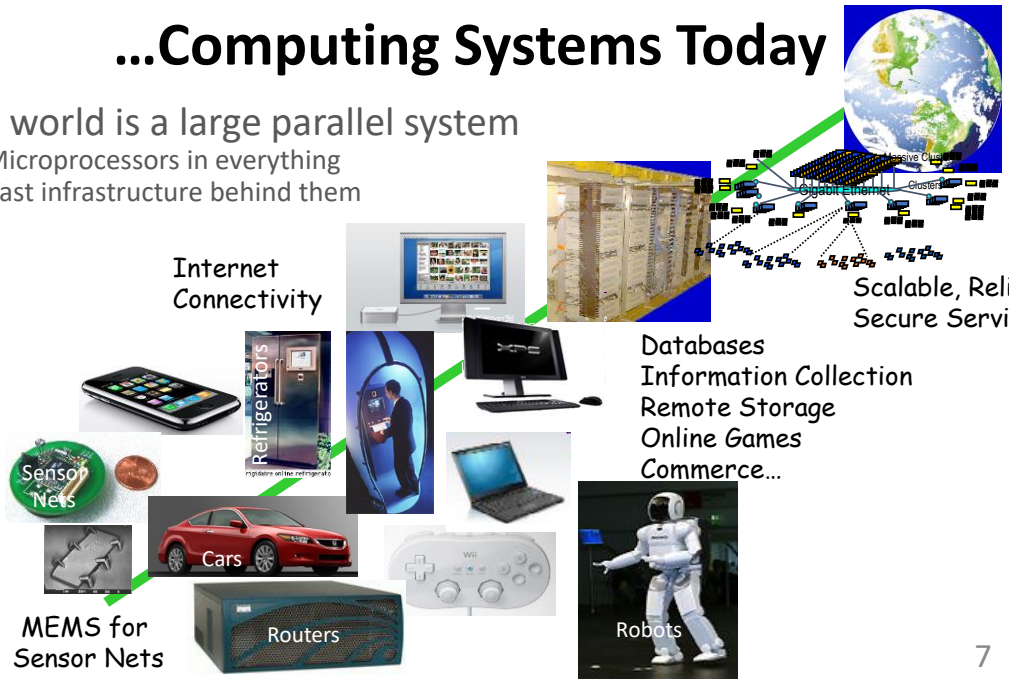
Electronic Delay Storage Automatic Calculator (EDSAC) - early British computer
University of Cambridge, UK, 1949
Considered to be one of the first stored program electronic computers

6

6

…Computing Systems Today

- The world is a large parallel system
  - Microprocessors in everything
  - Vast infrastructure behind them



Classes of Computers

- Embedded systems; Personal Mobile Devices (PMD)
  - e.g., smart phones, tablet computers
  - Emphasis on energy efficiency, cost, and real-time
- Desktop Computing
  - Emphasis on price-performance
- Servers
  - Emphasis on availability, scalability, throughput
- Clusters/Datacenters/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

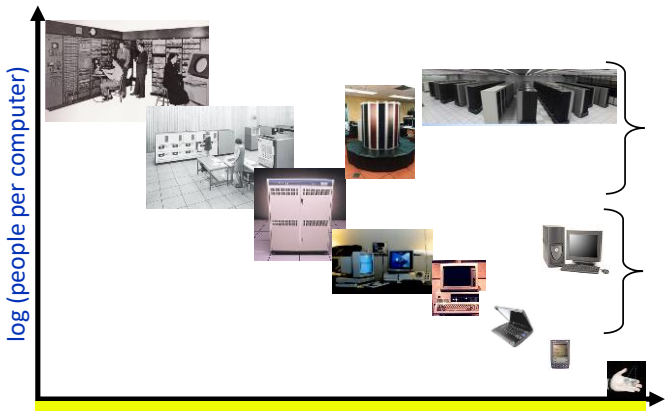
Bell's Law of Computer Classes

- Definition
  - Roughly every decade a new, lower priced computer class forms based on a new programming platform, network, and interface resulting in new usage and the establishment of a new industry
- Evolution
  - mainframes (1960s)
  - minicomputers (1970s); essentially replaced by clusters of PCs
  - personal computers and workstations evolving into a network enabled by Local Area Networking or Ethernet (1980s)
  - web browser client-server structures enabled by the Internet (1990s)
  - cloud computing, e.g., Amazon Web Services or Microsoft's Azure (2000s)
  - small form-factor devices such as cell phones and other cell phone sized devices, e.g., Smartphones (c. 2000)
  - Wireless Sensor Networks (WSN), Internet of Things (IoT) (c. >2005)

9

9

Bell's Law – new class per decade

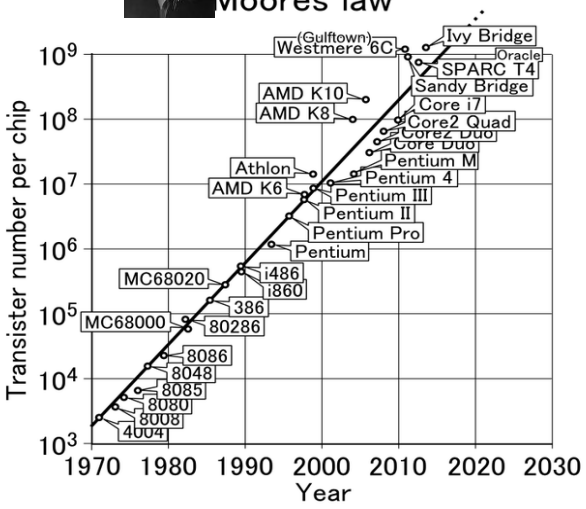


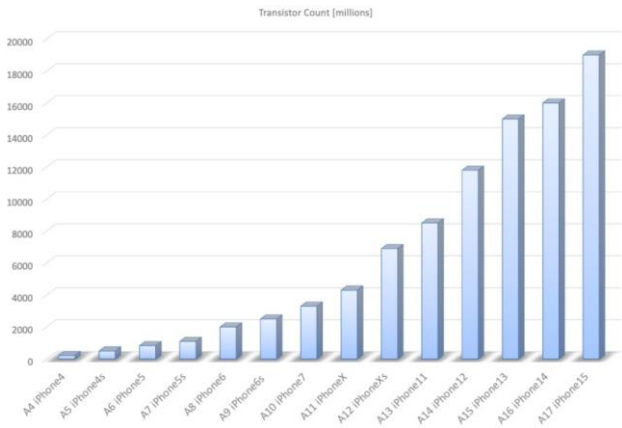
- Enabled by technological opportunities
- Smaller, more numerous and more intimately connected
- Brings in a new kind of application
- Used in many ways not previously imagined

10

10

Moores law

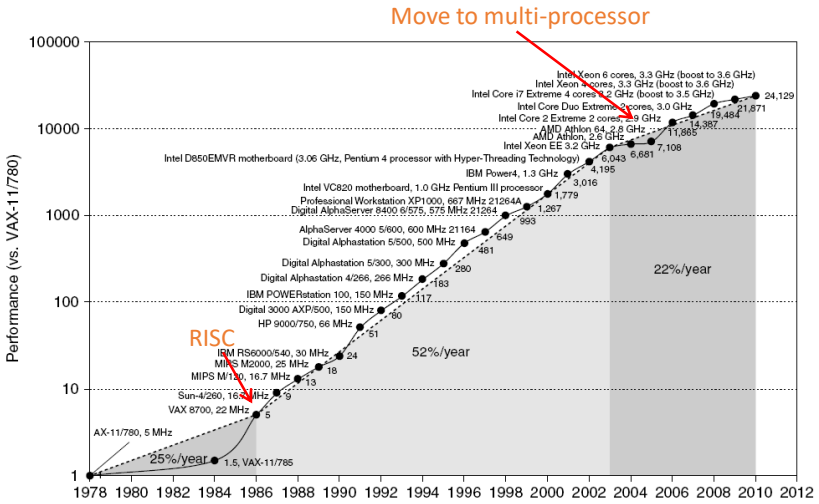




2023

11

Single Processor Performance

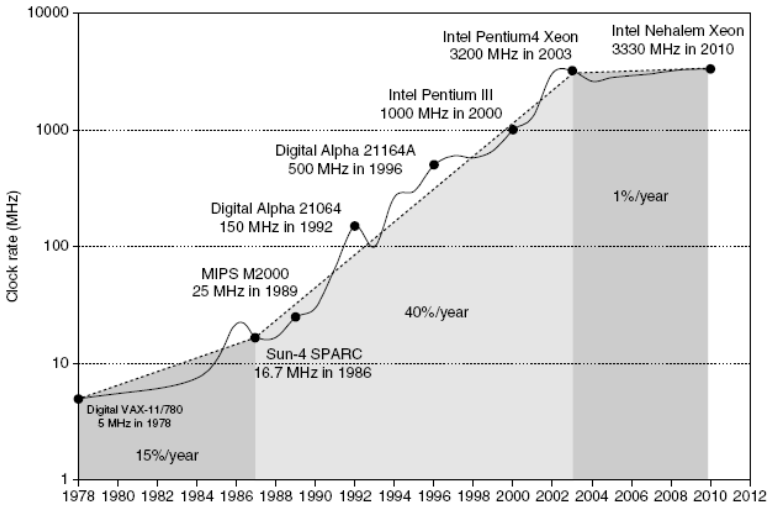


What has been the main method to improve performance?

12

12

Clock Frequency

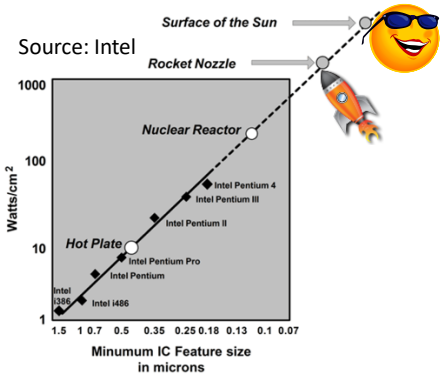


13

13

Power (density)

Source: Intel



- **Intel 80386 consumed ~2 W**
- **3.3 GHz Intel Core i7 consumes 130 W**
- **Heat must be dissipated from 1.5x1.5 cm chip**
- **This is the limit of what can be cooled by air**

Power-consumption = Dynamic-power + Static-power

(Includes due to short-circuit) (Due to Leakage)

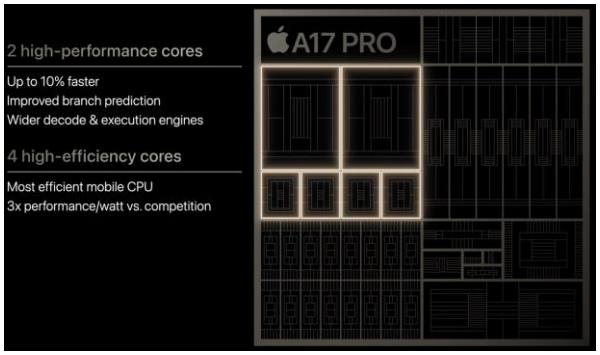
$$P = \frac{1}{2}\alpha C_L V_{dd}^2 f + V_{dd} I_{leak}$$

14

14

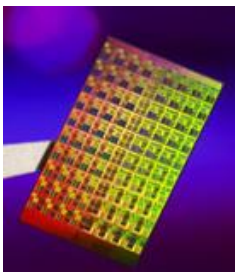
Bottom-line: Technology constantly on the move!

- Num of transistors not limiting factor
  - Currently ~19 billion transistors/chip
  - Problems:
    - Too much Power, Heat, Latency
    - Not enough Parallelism
- 3-dimensional chip technology?
  - Sandwiches of silicon
  - "Through-Silicon-Vias" for comm.
- On-chip optical connections?
  - Power savings for large packets
- Apple A17-Pro chip in iPhone 15
  - 6=2 (high perf.)+4 (high-efficiency) cores/chip
  - 3 nm
  - 19 billion transistors

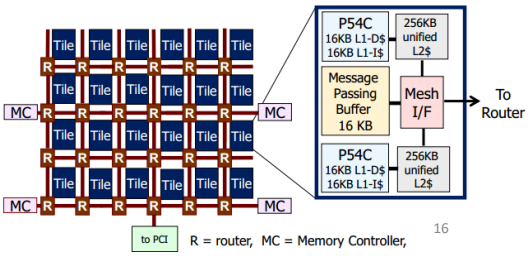


Manycore Chips: the future is here!

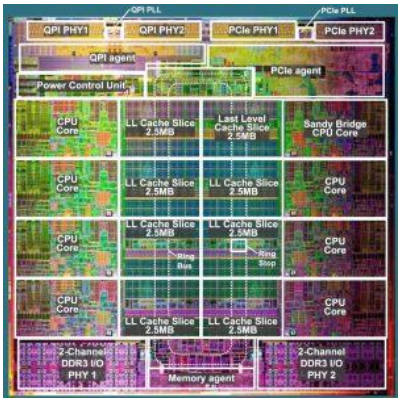
- Intel 80-core multicore chip (Feb.2007)
  - 80 simple cores
  - Two FP-engines / core
  - Mesh-like network
  - 100 million transistors
  - 65nm feature size
- Intel Single-Chip Cloud Computer (Aug.2010)
  - 24 "tiles"
  - 24-router mesh network with 256 GB/s bisection
  - 4 integrated DDR3 memory controllers
  - Hardware support for message-passing
- "Manycore" refers to many processors/chip
  - 64? 128? Hard to say exact boundary
- How to program these?
  - Use 2 CPUs for video/audio
  - Use 1 for word processor, 1 for browser
  - 76 for virus checking?

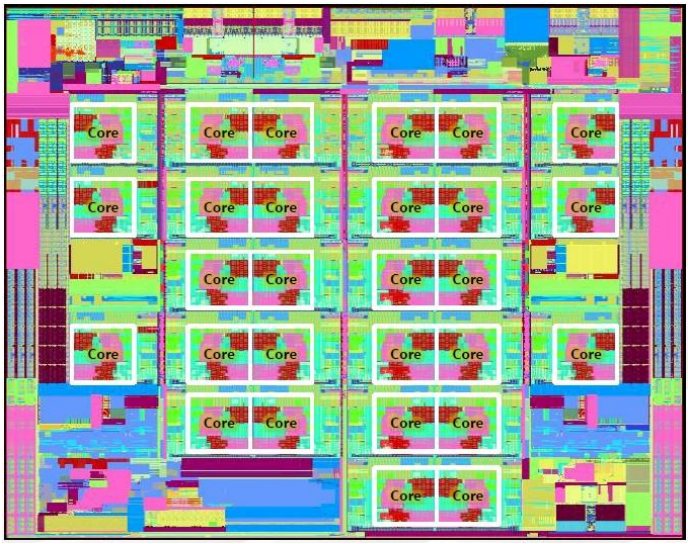


Intel 80-core chip
https://en.wikichip.org/wiki/intel/microarchitectures/polaris



Example: Intel Sandy Bridge, Xeon, Ice Lake… (i3,i5,i7; 2011…)



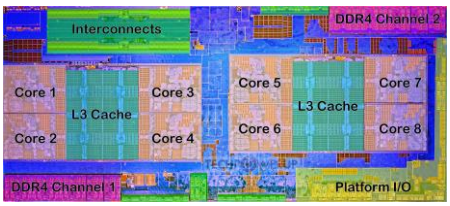


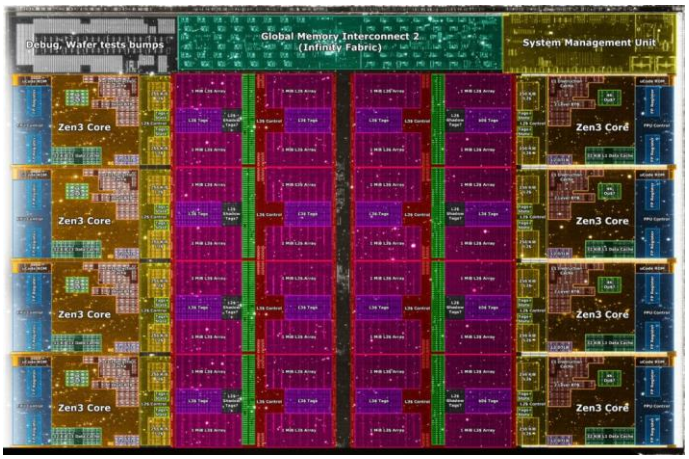
https://en.wikipedia.org/wiki/List_of_Intel_CPU_microarchitectures
https://www.nextplatform.com/2021/04/19/deep-dive-into-intels-ice-lake-xeon-sp-architecture/

17

17

Example: AMD Ryzen (2017-)



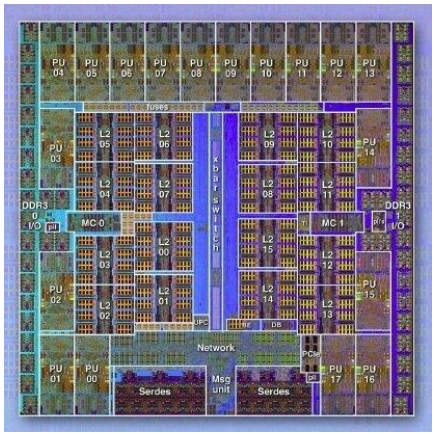


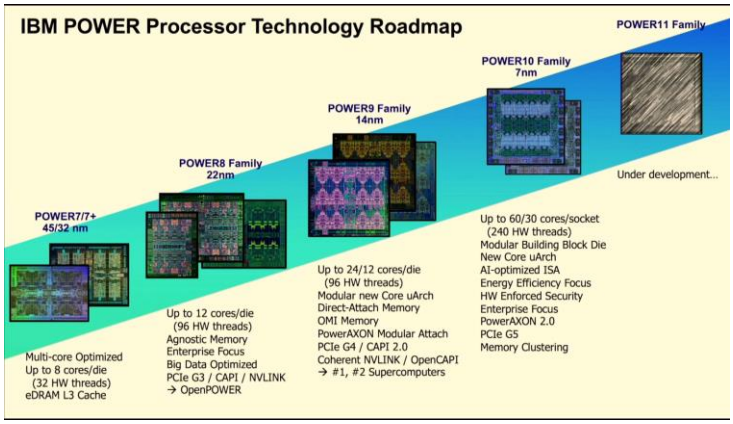
https://wccftech.com/amd-ryzen-5000-zen-3-vermeer-undressed-high-res-die-shots-close-ups-pictured-detailed/

18

18

IBM Blue Gene A2, Power Family (2010-)



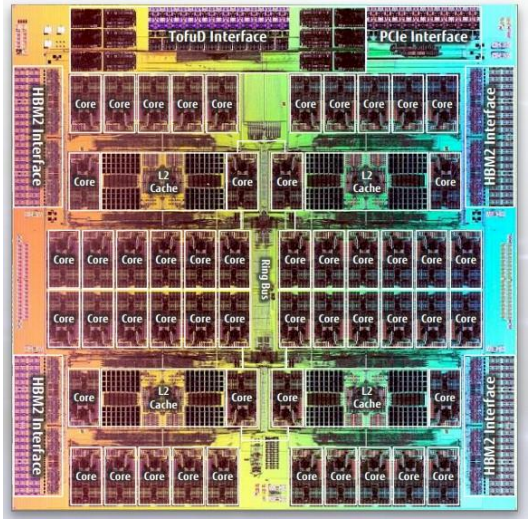


https://www.anandtech.com/show/15985/hot-chips-2020-live-blog-ibms-power10-processor-on-samsung-7nm-1000am-pt

19

19

Fujitsu A64FX (2020-)



https://en.wikipedia.org/wiki/Fujitsu_A64FX#:~:text=Each%20A64FX%20processor%20has%20four,cores%20for%20non%2Dcomputational%20purposes.
https://smist08.wordpress.com/2020/07/17/challenges-for-many-core-processors/

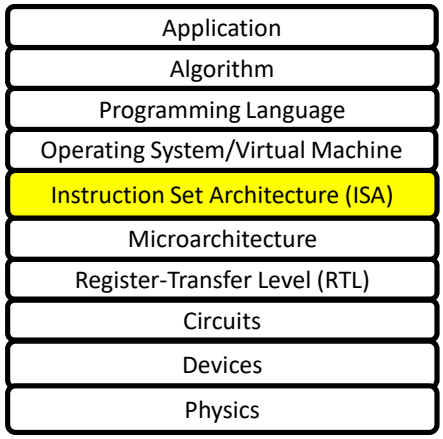
20

20

What is Computer Architecture?

Gap too large to bridge in one step

(but there are exceptions, e.g., magnetic compass)



At each abstraction layer, optimizations can be done to impact:
--Performance
--Power
--Area
--Cost
--Reliability
--Security
…

In its broadest definition, **computer architecture** is the design of the abstraction layers that allow us to implement information processing **applications** efficiently using available **manufacturing technologies.**

21

Computer Architecture's Changing Definition

- 1950s to 1960s: Computer Architecture Course: Computer Arithmetic
- 1970s to mid 1980s: Computer Architecture Course: Instruction Set Design, especially ISA appropriate for compilers
- 1990s: Computer Architecture Course:
  Design of CPU, memory system, I/O system, Multiprocessors, Networks
- 2000s: Computer Architecture Course: Multi-core design, on-chip networking, parallel programming, power reduction, instruction level parallelism
- 2013: Computer Architecture Course: Data and thread level parallelism. Self adapting systems. Security and reliability. Datacenters/Warehouse scale computing. GPUs.
- 2023: More datacenters? Domain specific architecture? AI/ML applications? Near-memory computing?

22

22

Outline

- What is and what does a computer architect?
- Classes of computers
- Trends in technology
- Defining computer architecture
- New definition of computer architecture

Crossroads: Conventional Wisdom in Comp. Arch.

- Old Conventional Wisdom (CW): Power is free, Transistors expensive
- New CW: "Power wall" Power expensive, Transistors free (Can put more on chip than can afford to turn on)
- Old CW: Sufficiently increasing Instruction Level Parallelism via compilers, innovation (Out-of-order, speculation, VLIW, …)
- New CW: "ILP wall" law of diminishing returns on more HW for ILP
- Old CW: Multiplies are slow, Memory access is fast
- New CW: "Memory wall" Memory slow, multiplies fast (200 clock cycles to DRAM memory, 4 clocks for multiply)
- Old CW: Uniprocessor performance 2X / 1.5 yrs
- New CW: Power Wall + ILP Wall + Memory Wall = "Brick Wall"
  - Uniprocessor performance now 2X / 5(?) yrs

Change in chip design - multiple cores: 2X processors per chip / ~2 years
  - More power efficient to use a large number of simpler processors rather than a small number of complex processors

Computer Architecture is an Integrated Approach

- **Old definition** of computer architecture:
  - Instruction set design
- **New definition**: design the **organization** (memory design, memory interconnect, internal CPU, multicore) **and hardware** (detailed logic design, packaging) to meet goals and functional requirements
- What really matters is the functioning of the complete system: application, operating system, compiler, hardware
- It is very important to think across all hardware/software boundaries
  - New Technology ⇒ New Capabilities ⇒ New Architectures ⇒ New Tradeoffs

25

25

Part 2: Computer Architecture – Design Principles and Analysis

26

Outline

- Quantitative **principles of design**
  1. Take Advantage of Parallelism
  2. Principle of Locality
  3. Focus on the Common Case
  4. Amdahl's Law
  5. The Processor Performance Equation
- Culture of anticipating and exploiting advances in technology - technology performance trends
- Careful, quantitative comparisons
  - Define and quantify power, cost, dependability, relative performance

27

27

1) Taking Advantage of Parallelism

- Detailed HW design
  - Carry look-ahead adders use parallelism to speed up computing sums from linear to logarithmic in number of bits per operand
  - Multiple memory banks searched in parallel in set-associative caches
- Pipelining: overlap instruction execution to reduce the total time to complete an instruction sequence
  - Not every instruction depends on immediate predecessor ⇒ executing instructions completely/partially in parallel possible
  - Classic 5-stage pipeline:
    1) Instruction Fetch (Ifetch),
    2) Register Read (Reg),
    3) Execute (ALU),
    4) Data Memory Access (Dmem),
    5) Register Write (Reg)
- Increasing throughput of server computer via multiple processors or multiple disks

28

28

2) 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., straight line code, array access)
- Last 30+ years, HW relied on locality for speed

29

29

3) Focus on the Common Case - "Make Frequent Case Fast"

- Common sense guides computer design
- In making a design trade-off, favor the frequent case over the infrequent case
  - E.g., Instruction fetch and decode unit used more frequently than multiplier, so optimize it first
- Frequent case is often simpler and can be done faster than the infrequent case
  - E.g., overflow is rare when adding 2 numbers, so improve performance by optimizing more common case of no overflow
  - May slow down overflow, but overall performance improved by optimizing for the normal case
- What is frequent case? How much is performance improved by making frequent case faster?

  => Amdahl's Law

30

30

4) Amdahl's Law

$$\text{ExTime}_{new} = \text{ExTime}_{old} \times \left[\left(1 - \text{Fraction}_{enhanced}\right) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}} \right]$$

$$\text{Speedup}_{overall} = \frac{\text{ExTime}_{old}}{\text{ExTime}_{new}} = \frac{1}{\left(1 - \text{Fraction}_{enhanced}\right) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}}}$$

31

Amdahl's Law example

- New CPU 10X faster
- I/O bound server, so 60% time waiting for I/O

$$\text{Speedup}_{overall} = \frac{1}{\left(1 - \text{Fraction}_{enhanced}\right) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}}}$$

$$= \frac{1}{\left(1 - 0.4\right) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

- Apparently, it is human nature to be attracted by 10X faster, vs. keeping in perspective it is just 1.6X faster

32

5) Processor Performance Equation

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	Inst Count	CPI	Cycle time
Program	X		
Compiler	X	(X)	
Inst. Set.	X	X	
Organization		X	X
Technology			X

33

Cycles Per Instruction (CPI)

$$\text{CPU time} = \text{Cycle Time} \times \sum_{j=1}^{n} \text{CPI}_j \times \text{I}_j$$

$$\text{CPI} = \sum_{j=1}^{n} \text{CPI}_j \times \text{F}_j \qquad \text{where } \text{F}_j = \frac{\text{I}_j}{\text{Instruction Count}}$$

"Instruction Frequency"

34

Outline

- Quantitative principles of design
  1. Take Advantage of Parallelism
  2. Principle of Locality
  3. Focus on the Common Case
  4. Amdahl's Law
  5. The Processor Performance Equation
- Culture of anticipating and exploiting advances in technology - technology performance trends
- Careful, quantitative comparisons
  - Define and quantify power, cost, dependability, relative performance

35

35

Tracking Technology Performance Trends

- Detailed comparisons for four technologies over 20 years:
  - **Disks**
  - **Memory**
  - **Network**
  - **Processors**
- Compare ~1980s Archaic (Nostalgic) vs. ~2000s Modern (Newfangled)
  - Performance Milestones in each technology
- Compare Bandwidth vs. Latency improvements in performance over 20 years
- **Bandwidth:** number of events per unit time
  - E.g., Mbits/second over network, Mbytes/second from disk
- **Latency:** elapsed time for a single event
  - E.g., one-way network delay in microseconds, average disk access time in milliseconds

36

36

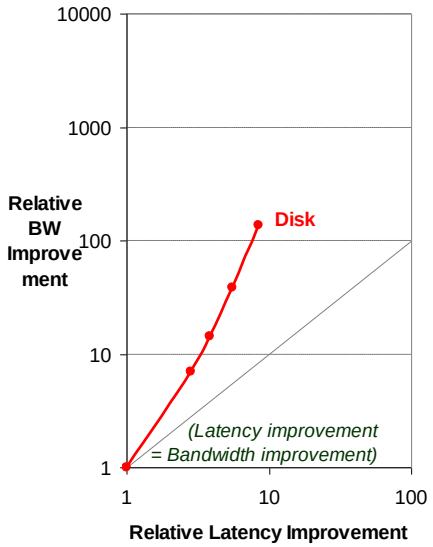
Disks: Archaic vs. Modern

• CDC Wren I, 1983	• Seagate 373453, 2003	
• 3600 RPM	• 15000 RPM	(4X)
• 0.03 GBytes capacity	• 73.4 GBytes	(2500X)
• Tracks/Inch: 800	• Tracks/Inch: 64,000	(80X)
• Bits/Inch: 9,550	• Bits/Inch: 533,000	(60X)
• Three 5.25" platters	• Four 2.5" platters (in 3.5" form factor)	
• Bandwidth: 0.6 MBytes/sec	• Bandwidth: 86 MBytes/sec	(140X)
• Latency: 48.3 ms	• Latency: 5.7 ms	(8X)
• Cache: none	• Cache: 8 MBytes	

37

37

Latency Lags Bandwidth (for last ~20 years)



- Performance Milestones:
- Disk: 3600, 5400, 7200, 10000, 15000 RPM

38

38

Memory: Archaic vs. Modern

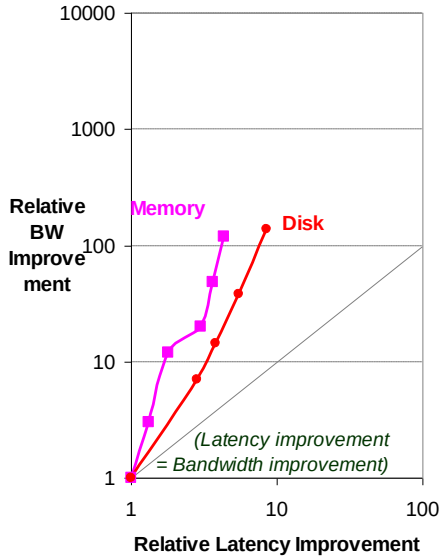
- 1980 DRAM (asynchronous)
- 0.06 Mbits/chip
- 64,000 xtors, 35 mm²
- 16-bit data bus per module, 16 pins/chip
- 13 Mbytes/sec
- Latency: 225 ns
- (no block transfer)

- 2000 Double Data Rate Synchr. (clocked) DRAM
- 256.00 Mbits/chip (4000X)
- 256,000,000 xtors, 204 mm²
- 64-bit data bus per DIMM, 66 pins/chip (4X)
- 1600 Mbytes/sec (120X)
- Latency: 52 ns (4X)
- Block transfers (page mode)

39

39

Latency Lags Bandwidth (for last ~20 years)



- Performance Milestones:
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM
- Disk: 3600, 5400, 7200, 10000, 15000 RPM

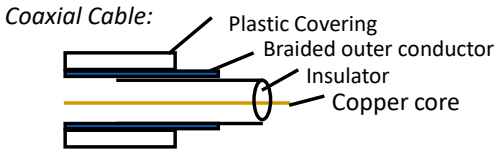
40

40

LANs: Archaic vs. Modern

- Ethernet 802.3
- Year of Standard: 1978
- 10 Mbits/s link speed
- Latency: 3000 µsec
- Shared media
- Coaxial cable

- Ethernet 802.3ae
- Year of Standard: 2003
- 10,000 Mbits/s link speed (1000X)
- Latency: 190 µsec (15X)
- Switched media
- Category 5 copper wire



"Cat 5" is 4 twisted pairs in bundle

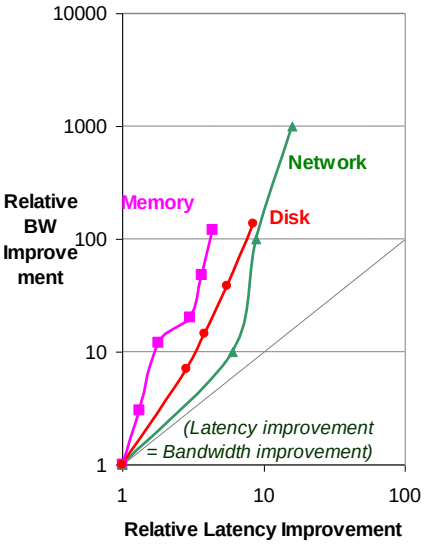
Twisted Pair:

Copper, 1mm thick,
twisted to avoid antenna effect

41

41

Latency Lags Bandwidth (for last ~20 years)



- Performance Milestones:
- Ethernet: 10Mb, 100Mb, 1000Mb, 10000 Mb/s
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM
- Disk: 3600, 5400, 7200, 10000, 15000 RPM

42

42

CPUs: Archaic vs. Modern

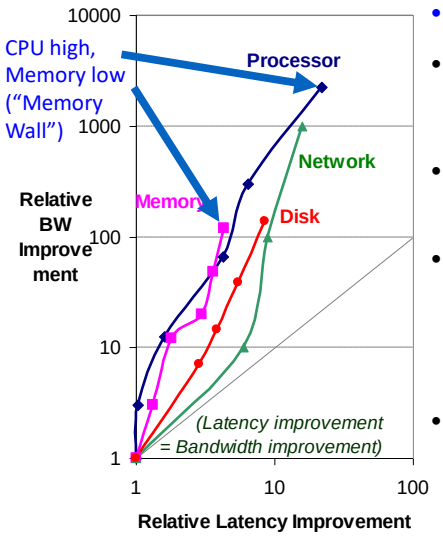
- 1982 Intel 80286
- 12.5 MHz
- 2 MIPS (peak)
- Latency 320 ns
- 134,000 xtors, 47 mm^2
- 16-bit data bus, 68 pins
- Microcode interpreter, separate FPU chip
- (no caches)

- 2001 Intel Pentium 4
- 1500 MHz = 1.5 GHz (120X)
- 4500 MIPS (peak) (2250X)
- Latency 15 ns (20X)
- 42,000,000 xtors, 217 mm^2
- 64-bit data bus, 423 pins
- 3-way superscalar, Dynamic translate to RISC, Superpipelined (22 stage), Out-of-Order execution
- On-chip 8KB Data caches, 96KB Instr. Trace cache, 256KB L2 cache

43

43

Latency Lags Bandwidth (for last ~20 years)



- Performance Milestones:
- Processor: '286, '386, '486, Pentium, Pentium Pro, Pentium 4 (21x,2250x)
- Ethernet: 10Mb, 100Mb, 1000Mb, 10000 Mb/s (16x,1000x)
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x,120x)
- Disk : 3600, 5400, 7200, 10000, 15000 RPM (8x,143x)

(Latency = simple operation w/o contention, BW = best-case)

44

44

6 Reasons for "Latency Lags Bandwidth"

1. Moore's Law helps BW more than latency

- Faster transistors, more transistors, more pins help Bandwidth
  - MPU Transistors: 0.130 vs. 42 M xtors (300X)
  - DRAM Transistors: 0.064 vs. 256 M xtors (4000X)
  - MPU Pins: 68 vs. 423 pins (6X)
  - DRAM Pins: 16 vs. 66 pins (4X)
- Smaller, faster transistors but latency has not reduced as dramatically with successive generations
  - Feature size: 1.5 to 3 vs. 0.18 micron (8X,17X)
  - MPU Die Size: 35 vs. 204 mm^2 (ratio sqrt ⇒ 2X)
  - DRAM Die Size: 47 vs. 217 mm^2 (ratio sqrt ⇒ 2X)

45

45

6 Reasons for "Latency Lags Bandwidth"

2. Distance limits latency

- Size of DRAM block ⇒ long bit and word lines ⇒ most of DRAM access time
- 1. & 2. explains linear latency vs. square BW

3. Bandwidth easier to sell ("bigger=better")

- E.g., 10 Gbits/s Ethernet ("10 Gig") vs. 10 µsec latency Ethernet
- 4400 MB/s DIMM ("PC4400") vs. 50 ns latency
- Even if it is just marketing, customers are now trained
- Since bandwidth sells, more resources thrown at bandwidth, which further tips the balance

46

46

6 Reasons for "Latency Lags Bandwidth"

4. Latency helps BW, but not vice versa

- Spinning disk faster improves both bandwidth and rotational latency
- Lower DRAM latency ⇒
  More access/second (higher bandwidth)
- Higher linear density helps disk BW
  (and capacity), but not disk Latency
- More cores help BW (throughput), not latency

47

47

6 Reasons for "Latency Lags Bandwidth"

5. Bandwidth hurts latency

- Queues help Bandwidth, hurt Latency (Queuing Theory)
- Adding chips to widen a memory module increases Bandwidth but higher fan-out on address lines may increase Latency

6. Operating System overhead hurts Latency more than Bandwidth

- Long messages amortize overhead;
  overhead bigger part of short messages
  It takes longer to create and to send a long message,
  which is needed instead of a short message to lessen
  average cost per data byte of fixed size message overhead

48

48

Sum-up of Technology Trends

- For disk, LAN, memory, and microprocessor, bandwidth improves by more than the square of latency improvement
  - In the time that bandwidth doubles, latency improves by no more than 1.2X to 1.4X
- Lag of gains for latency vs. bandwidth probably even larger in real systems, as bandwidth gains multiplied by replicated components
  - Multiple processors in a cluster or even on a chip
  - Multiple disks in a disk array
  - Multiple memory modules in a large memory
  - Simultaneous communication in switched local area networks (LANs)
- HW and SW developers should innovate assuming Latency lags Bandwidth

49

49

Outline

- Quantitative principles of design
  1. Take Advantage of Parallelism
  2. Principle of Locality
  3. Focus on the Common Case
  4. Amdahl's Law
  5. The Processor Performance Equation
- Culture of anticipating and exploiting advances in technology - technology performance trends
- Careful, quantitative comparisons
  - Define and quantify power, cost, dependability, relative performance

50

50

Metrics used to Compare Designs

- Energy and Power
  - Also peak power and peak switching current
- Cost
  - Die cost and system cost
- Reliability
  - Resiliency to electrical noise, part failure
  - Robustness to bad software, operator error
- Execution Time
  - Average and worst-case
  - Latency vs. Throughput
- Maintainability
  - System administration costs
- Compatibility
  - Software costs dominate

51

Performance: What to Measure

- Usually rely on benchmarks vs. real workloads
- To increase predictability, collections of benchmark applications, called benchmark suites, are popular
- SPECCPU: popular desktop benchmark suite
  - CPU only, split between integer and floating point programs
  - SPECint2000 had 12 integer, SPECfp2000 had 14 integer codes
  - SPEC CPU2006 has 12 integer benchmarks (CINT2006) and 17 floating-point benchmarks (CFP2006)
  - SPECSFS (NFS file server) and SPECWeb (WebServer) have been added as server benchmarks
- Transaction Processing Council measures server performance and cost-performance for databases
  - TPC-C Complex query for Online Transaction Processing
  - TPC-H models ad hoc decision support
  - TPC-W a transactional web benchmark
  - TPC-App application server and web services benchmark

52

52

CINT2006 for Opteron X4 2356

Name	Description	IC×10^9	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.4	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.4	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.4	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.4	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.4	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.4	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.4	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.4	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.4	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.4	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.4	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.4	1,143	6,900	6.0
Geometric mean							11.7

53

How to Mislead with Performance Reports

1. Select pieces of workload that work well on your design, ignore others
2. Use unrealistic data set sizes for application (too big or too small)
3. Report throughput numbers for a latency benchmark
4. Report latency numbers for a throughput benchmark
5. Report performance on a kernel and claim it represents an entire application
6. Use 16-bit fixed-point arithmetic (because it's fastest on your system) even though application requires 64-bit floating-point arithmetic
7. Use a less efficient algorithm on the competing machine
8. Report speedup for an inefficient algorithm (bubblesort)
9. Compare hand-optimized assembly code with unoptimized C code
10. Compare your design using next year's technology against competitor's year old design (1% performance improvement per week)
11. Ignore the relative cost of the systems being compared
12. Report averages and not individual results
13. Report speedup over unspecified base system, not absolute times
14. Report efficiency not absolute times
15. Report MFLOPS not absolute times (use inefficient algorithm)

[David Bailey, "Twelve ways to fool the masses when giving performance results for parallel supercomputers"]

54

Summary

- Computer Architecture is much more than just ISA
- Computer Architecture skill sets:
  1. Quantitative approach to design
  2. Technology tracking and anticipation
  3. Appropriate comparison metrics
- Computer Engineering at the crossroads from sequential to parallel computing
  - Requires innovation in many fields, including computer architecture
- Read Chapter 1, Appendix A, B, C of textbook

55