COEN-2710 Microprocessors - Lecture 1

Computer Abstractions and Technology (Ch.1)

Cris Ababei
Marquette University
Dept. of Electrical and Computer Engineering (ECE)

1

Outline

- Historical Perspective
- Technology Trends
- The Computing Stack: Layers of Abstraction
- Performance Evaluation
- Benchmarking

2

2

Historical Perspective

- **ENIAC of WWII - first general purpose computer**
  - **Used for computing artillery firing tables**
  - **80 feet long by 8.5 feet high; used 18,000 vacuum tubes**
  - **Performed 1900 additions per second**

3

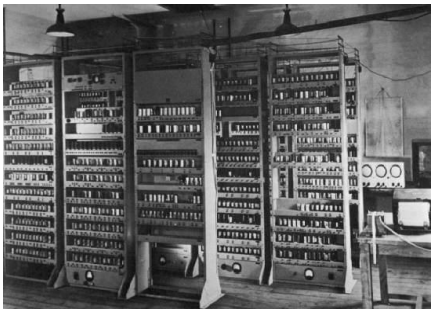
3

Computing Devices Then…

Manchester Baby, also called the Small-Scale Experimental Machine (SSEM)

University of Manchester, UK, 1948.

Considered 1st electronic stored-program computer.



Electronic Delay Storage Automatic Calculator (EDSAC)

University of Cambridge, UK, 1949

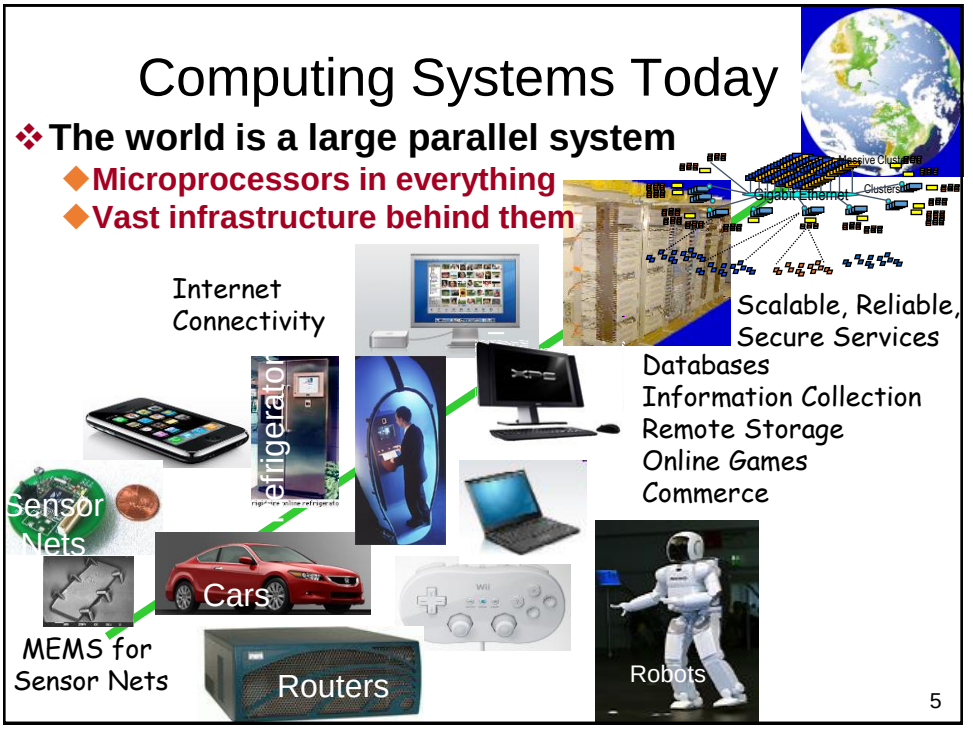
Considered to be 2nd electronic stored-program computer.

4

4

Computing Systems Today

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



5

5

Technology Trends

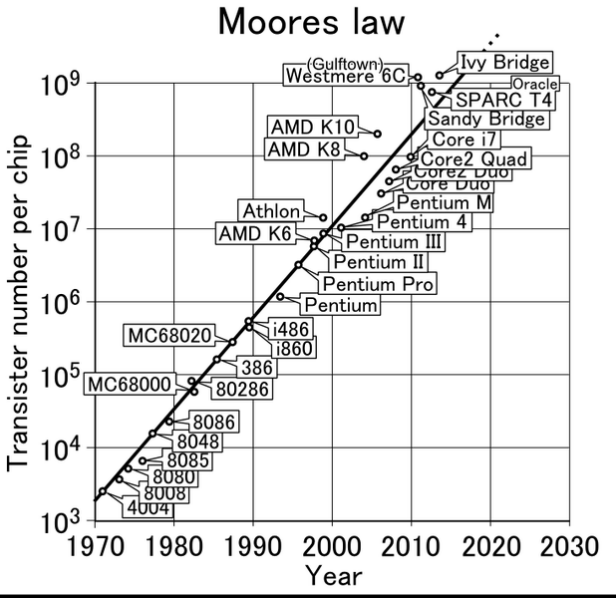
- **Electronics technology evolution**
  - **Increased capacity and performance**
  - **Reduced cost**

Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2005	Ultra large scale IC (ULSI)	6,200,000,000

6

6

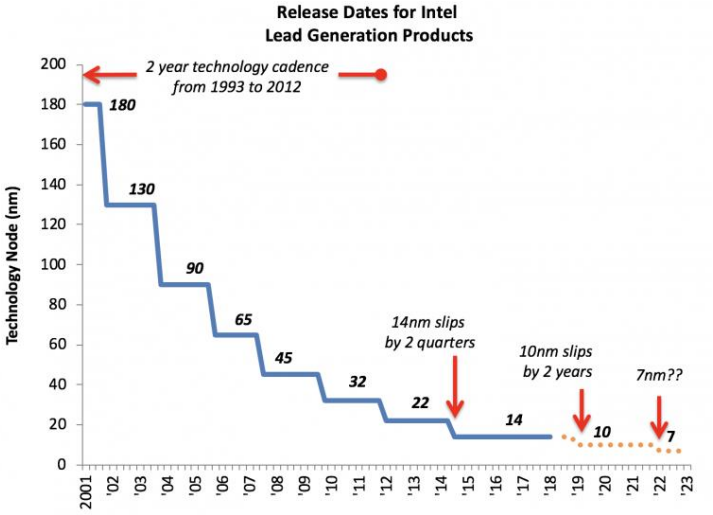
Moore's Law



7

7

Feature Size Decreasing

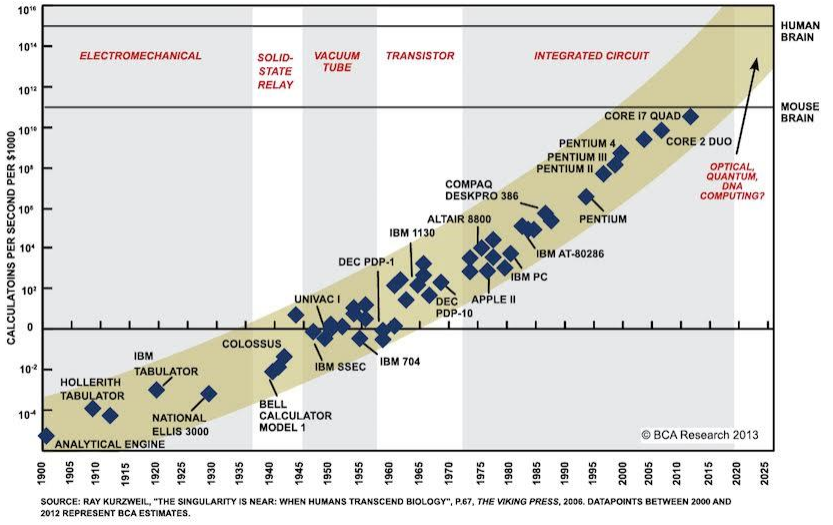


Source: ACM SIGARCH

8

8

Computation speed increasing

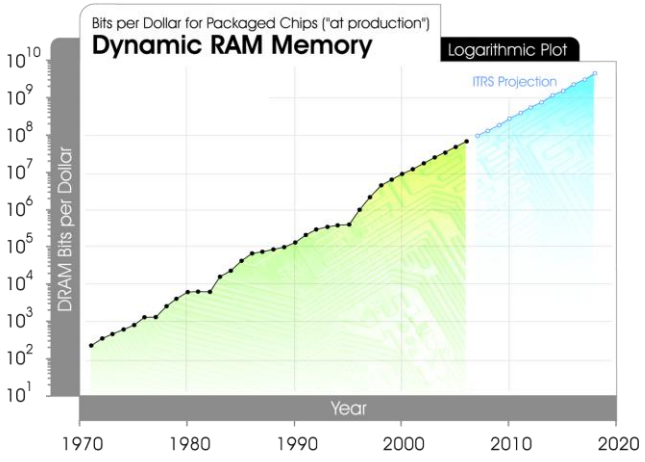


SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

9

9

Memory Capacity increasing

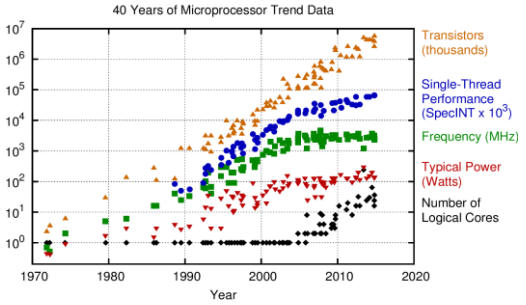


Source: https://www.kurzweilai.net/sin-charts

10

10

Power Trends



$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

×30 | 5V → 1V | ×1000

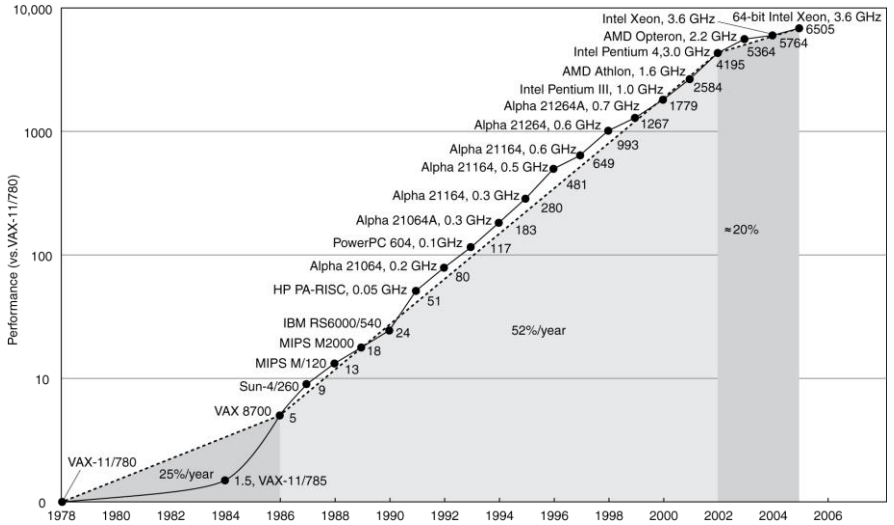
The **power wall**:

- Can't reduce voltage or remove heat more
- How else can we improve performance?

11

11

Uniprocessor Performance



Solution: Increase **number** of processors!

12

12

Types of Parallelism

- **Instruction Level Parallelism (ILP)**
  - Multiple instructions executing within processor, e.g. pipelining, superscalar design
- **Thread Level Level Parallelism (TLP)**
  - Simultaneous, temporal multithreading
- **Multiprocessing**
  - More than one processor per chip
  - May be tightly or loosely coupled

Flynn's Taxonomy:

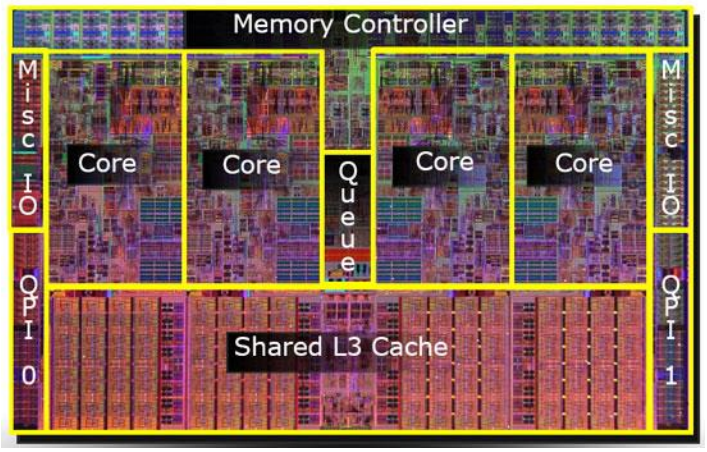
	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

13

13

Example: Intel i7

Intel i7 Nehalem Architecture, 4 cores

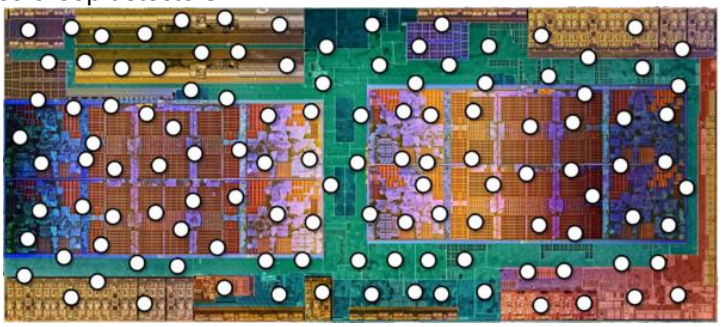


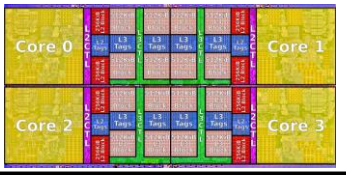
14

14

Example: AMD Zen

- Over 1300 sensors to monitor the state of the die over all critical paths
- 48 high-speed power supply monitors, 20 thermal diodes, and 9 high-speed droop detectors



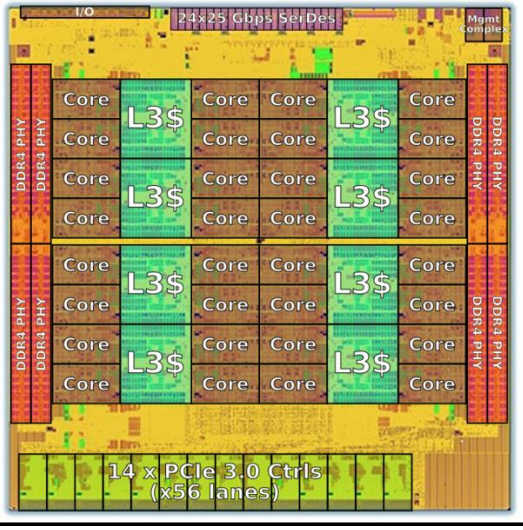


[Images source: wikichip.org]

15

Example: (ARM) Vulcan

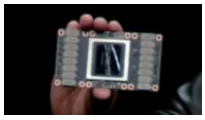
- Vulcan is a 16 nm high-performance 64-bit ARM microarchitecture designed by Broadcom and later introduced by Cavium for the server market.



[Images source: wikichip.org]

16

Example: Nvidia V100

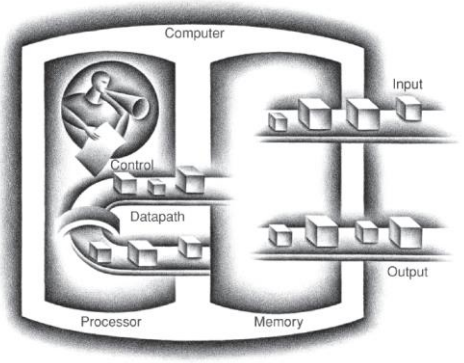


- Tesla Volta V100 graphics processor has 5,120 CUDA / Shader cores and is based upon 21 Billion transistors.

17

Still the Same Basics!

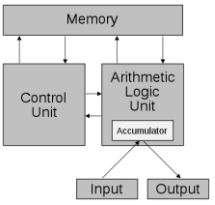
1. **Memory**
2. **Output**
3. **Datapath**
4. **Input**
5. **Control**



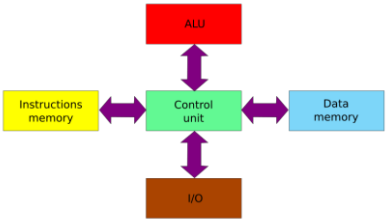
18

Hardware Architectures

- **Von Neumann Architecture**
  - Single memory for both Programs and Data



- **Harvard Architecture**
  - Separate memories for Programs and Data

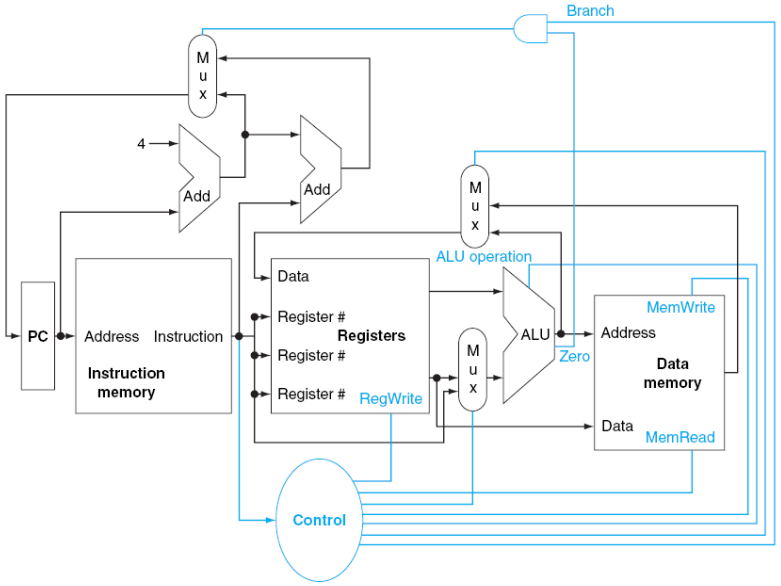


- **Modified Harvard Architecture**
  - Single physical memory, but separate CPU pathways for Programs and Data

19

19

Datapath and Control



20

20

Instruction Set Architectures (ISAs)

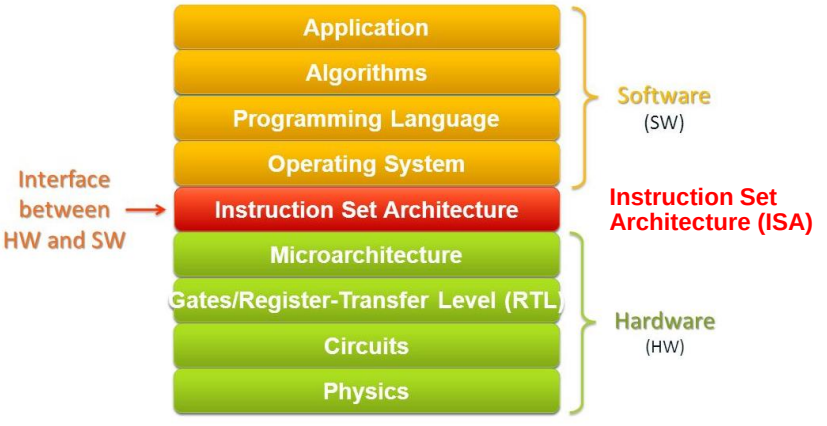
- CISC: Complex Instruction Set Chip
  - Large number of core instructions
  - Leads to larger, slower hardware
  - Good for special purpose processors (or necessary backwards compatibility, such as Intel)
- RISC: Reduced Instruction Set Chip
  - Small number of general instructions
  - Leads to more compact, faster hardware
  - Good for general purpose processors

21

21

The Computing Stack: Hierarchy

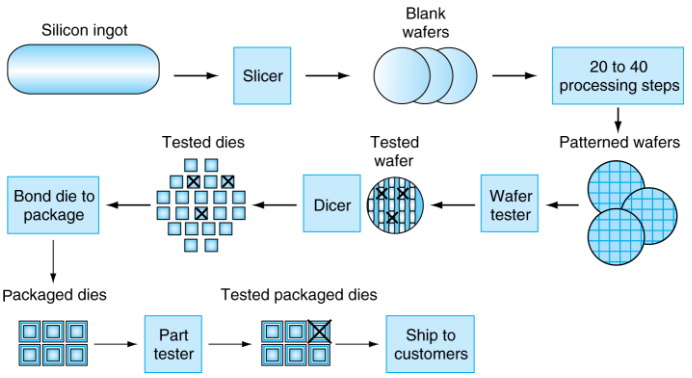
- **Must focus on one "layer" or "abstraction"**



22

22

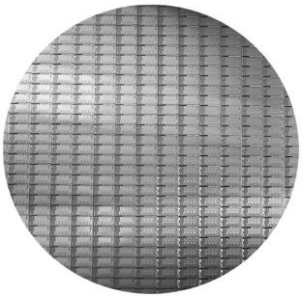
Manufacturing ICs



- **Yield**: proportion of working dies per wafer

23

Intel Core i7 Wafer



- 300mm wafer, 280 chips, 32nm technology
- Each chip is 20.7 x 10.5 mm
- Gates constructed of transistor circuits
- Circuits etched directly into silicon
  - Original Pentium: 4.5 million transistors
  - Pentium IV: 42 million transistors
  - Pentium IV Itanium: 592 million transistors
  - i7: 731 million transistors

24

How to Measure Performance?

- **Execution time (response time, latency)**
  - — How long does it take for my job to run?
  - — How long does it take to execute a job?
  - — How long must I wait for the database query?
- **Throughput**
  - — How many jobs can the machine run at once?
  - — What is the average execution rate?
  - — How much work is getting done?

If we upgrade a machine with a new processor what do we increase?

If we add a new machine to the lab what do we increase?

25

Definition of Performance

- **Performance**
  (larger number means better performance)

$$\text{performance}(x) = \frac{1}{\text{execution time}(x)}$$

So "X is n times faster than Y" means

$$n = \frac{\text{performance}(x)}{\text{performance}(y)} = \frac{\text{execution time}(y)}{\text{execution time}(x)}$$

26

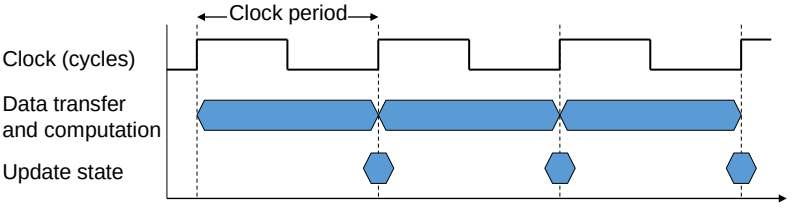
Measuring Performance

- "Wall clock time" or "Elapsed time" or "Response time"
  - Total time to complete task
  - Includes everything
    - Processing, I/O, OS overhead, idle time
  - Determines System Performance
- "CPU time" or "CPU execution time" or "EXE time"
  - Time spent processing a given job only
    - Discounts I/O time, other jobs' shares
  - Comprises (but, difficult to separate the two):
    - "User CPU time": time spent in the program
    - "System CPU time": time spent in the OS performing tasks on behalf of the program
  - Different programs are affected differently by CPU and system performance
  - Determines CPU Performance

27

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



- Clock period (clock cycle time OR cycle time): duration of a clock cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz
- Clock period is the inverse of clock frequency

28

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}} \quad \textbf{(1)}$$

- **Performance improved by**
  - **Reducing number of clock cycles**
  - **Increasing clock rate**
  - **Hardware designer must often trade off clock rate against cycle count**

29

29

CPU Time: Example 1

- **Computer A: 2GHz clock rate, 10s CPU time**
- **Designing Computer B**
  - **Aim for 6s CPU time**
  - **Can do faster clock, but causes 1.2 × clock cycles**
- **How fast must Computer B clock be?**

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

30

30

Instruction Count (IC) and CPI

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI}_{Avg} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}_{Avg}}{\text{Clock Rate}} \quad \textbf{(2)}$$

- **Instruction Count for a program: IC**
  - **Determined by program, ISA and compiler**
- **Average cycles per instruction: CPI**
  - **Determined by CPU hardware**
  - **If different instructions have different CPI**
    - **Average CPI affected by instruction mix**

31

31

CPI: Example 2

- **Computer A: Cycle Time = 250ps, CPI = 2.0**
- **Computer B: Cycle Time = 500ps, CPI = 1.2**
- **Same ISA**
- **Which is faster, and by how much?**

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$

$$= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps}$$ ← A is faster…

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$

$$= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps}$$

$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2$$ ← …by this much

32

32

CPI in More Detail

- **If different instruction classes take different numbers of cycles**

$$CPU\ \text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

  - Weighted Average CPI

Relative frequency

$$\text{CPI} = \frac{CPU\ \text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right)$$

$$\text{CPU}_{\text{Time}} = [\ \textstyle\sum_{i=1}^{n} (CPI_i * IC_i)\ ] * \text{T}_{\text{CLK}} \quad \textbf{(3)}$$

CPI: Example 3

- **Alternative compiled code sequences using instructions in classes A, B, C**

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles
    = 2×1 + 1×2 + 2×3
    = 10
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    = 4×1 + 1×2 + 1×3
    = 9
  - Avg. CPI = 9/6 = 1.5

Performance Summary

Gold Formula:

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

(4)

- **Performance depends on**
  - **Algorithm: affects IC, possibly CPI**
  - **Programming language: affects IC, CPI**
  - **Compiler: affects IC, CPI**
  - **Instruction set architecture: affects IC, CPI, T_c**

35

35

SPEC CPU Benchmark

- **Programs used to measure performance**
  - **Supposedly typical of actual workload**
- **Standard Performance Evaluation Corp (SPEC)**
  - **Develops benchmarks for CPU, I/O, Web, …**
- **SPEC CPU2006**
  - **Elapsed time to execute a selection of programs**
    - **Negligible I/O, so focuses on CPU performance**
  - **Normalize relative to reference machine**
  - **Summarize as geometric mean of performance ratios**
    - **CINT2006 (integer) and CFP2006 (floating-point)**

$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

36

36

CINT2006 for Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	358	8050	22.5
Combinatorial optimization	mcf	221	2.66	0.376	221	9120	41.2
Go game (AI)	go	1274	1.10	0.376	527	10490	19.9
Search gene sequence	hmmer	2616	0.60	0.376	590	9330	15.8
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100	20.7
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720	190.0
Video compression	h264avc	3793	0.50	0.376	713	22130	31.0
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250	21.5
Games/path finding	astar	1250	1.00	0.376	470	7020	14.9
XML parsing	xalancbmk	1045	0.70	0.376	275	6900	25.1
Geometric mean	–	–	–	–	–	–	25.7

What You Will Learn

- How programs are translated into the machine language
  - And how the hardware executes them
- The hardware/software interface
- What determines program performance
  - And how it can be improved
- How hardware designers improve performance
- What is parallel processing

Eight Great Ideas Invented by Computer Architects

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

39

39

Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction Set Architecture (ISA)
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance

40

40