

CSC 320: Computer Organization

Lecture 1: Course Introduction

Spring 2017

Instructor:
Haidar M. Harmanani

Agenda

- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class

Agenda

- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class

1/18/17

Lecture #1

3

The Computer Revolution

- Progress in computer technology
- Makes novel applications feasible
 - Computers in automobiles
 - Cell phones
 - Human genome project
 - World Wide Web
 - Search Engines
- Computers are pervasive

1/18/17

Lecture #1

4

Classes of Computers

- Personal computers
 - General purpose, variety of software
 - Subject to cost/performance tradeoff
- Server computers
 - Network based
 - High capacity, performance, reliability
 - Range from small servers to building sized

1/18/17

Lecture #1

5

Classes of Computers

- Supercomputers
 - High-end scientific and engineering calculations
 - Highest capability but represent a small fraction of the overall computer market
- Embedded computers
 - Hidden as components of systems
 - Stringent power/performance/cost constraints

1/18/17

Lecture #1

6

CSC 320 is NOT about hardware or software ...

- It is about the hardware-software interface
 - What does the programmer need to know to achieve the highest possible performance?
- Languages like C are closer to the underlying hardware, unlike languages like Scheme, Python, Java!
 - We can talk about hardware features in higher-level terms
 - Allows programmer to explicitly harness underlying hardware parallelism for high performance

1/18/17

Lecture #1

7

Your Grand Father's CSC 320



1/18/17

Lecture #1

8

Your Grand Father's CSC 320?



- 6 MHz CPU Clock
- 512 KB RAM
- 20 MB Hard disk
- 1.44 MB 3.5 inch floppy drive
- 640x480 VGA

The PostPC Era

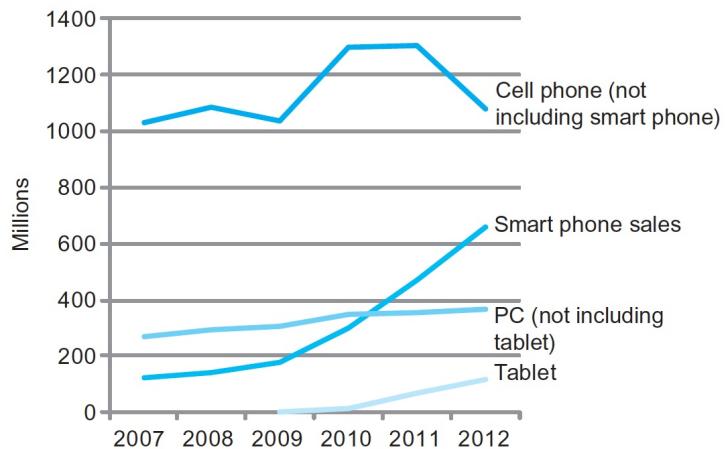


FIGURE 1.2 The number manufactured per year of tablets and smart phones, which reflect the post-PC era, versus personal computers and traditional cell phones. Smart phones represent the recent growth in the cell phone industry, and they passed PCs in 2011. Tablets are the fastest growing category, nearly doubling between 2011 and 2012. Recent PCs and traditional cell phone categories are relatively flat or declining.

1/18/17

Lecture #1

11

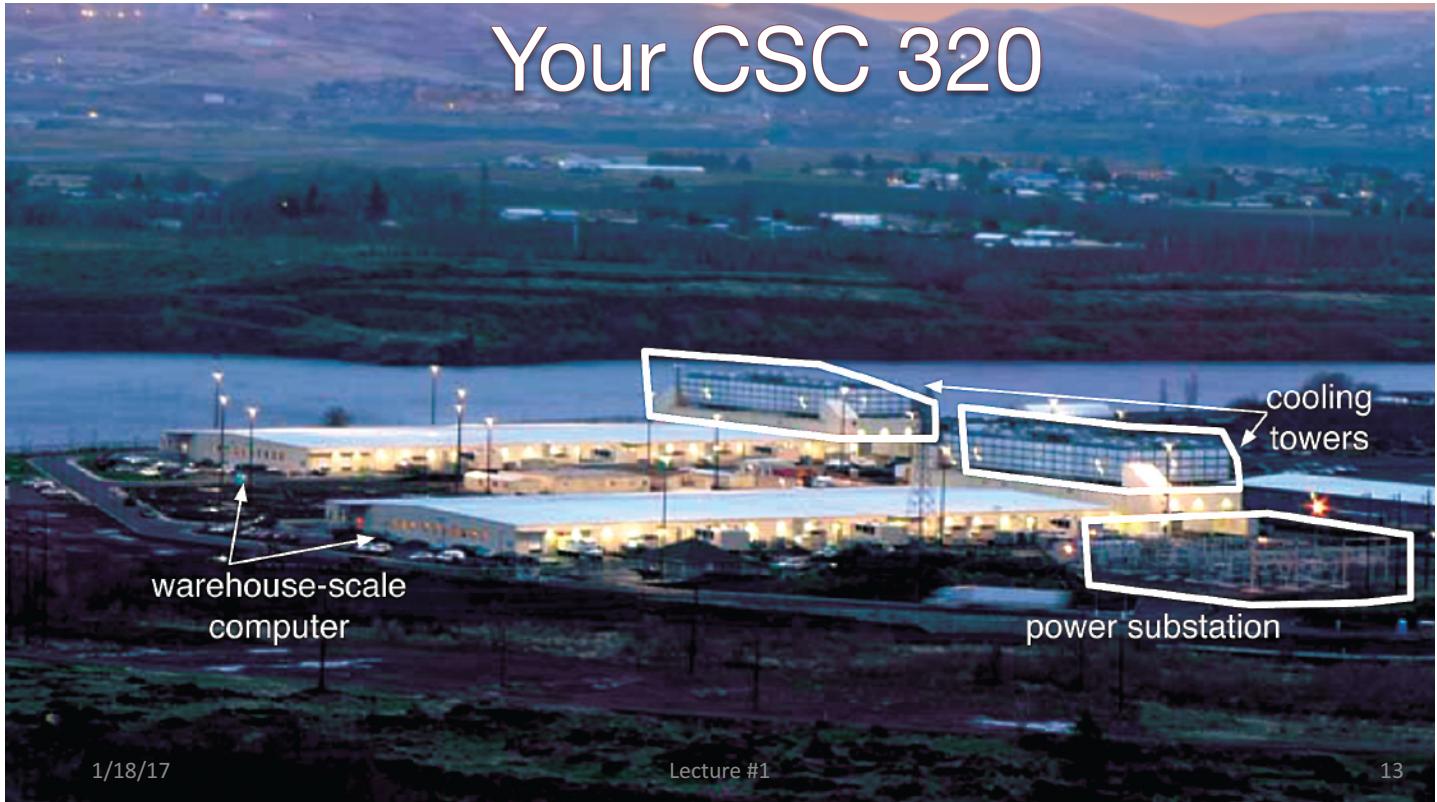
Your CSC 320

Personal
Mobile
Devices

Lecture #1

12

Your CSC 320



1/18/17

Lecture #1

13

Your Kids CSC 320?



1/18/17

Lecture #1

14

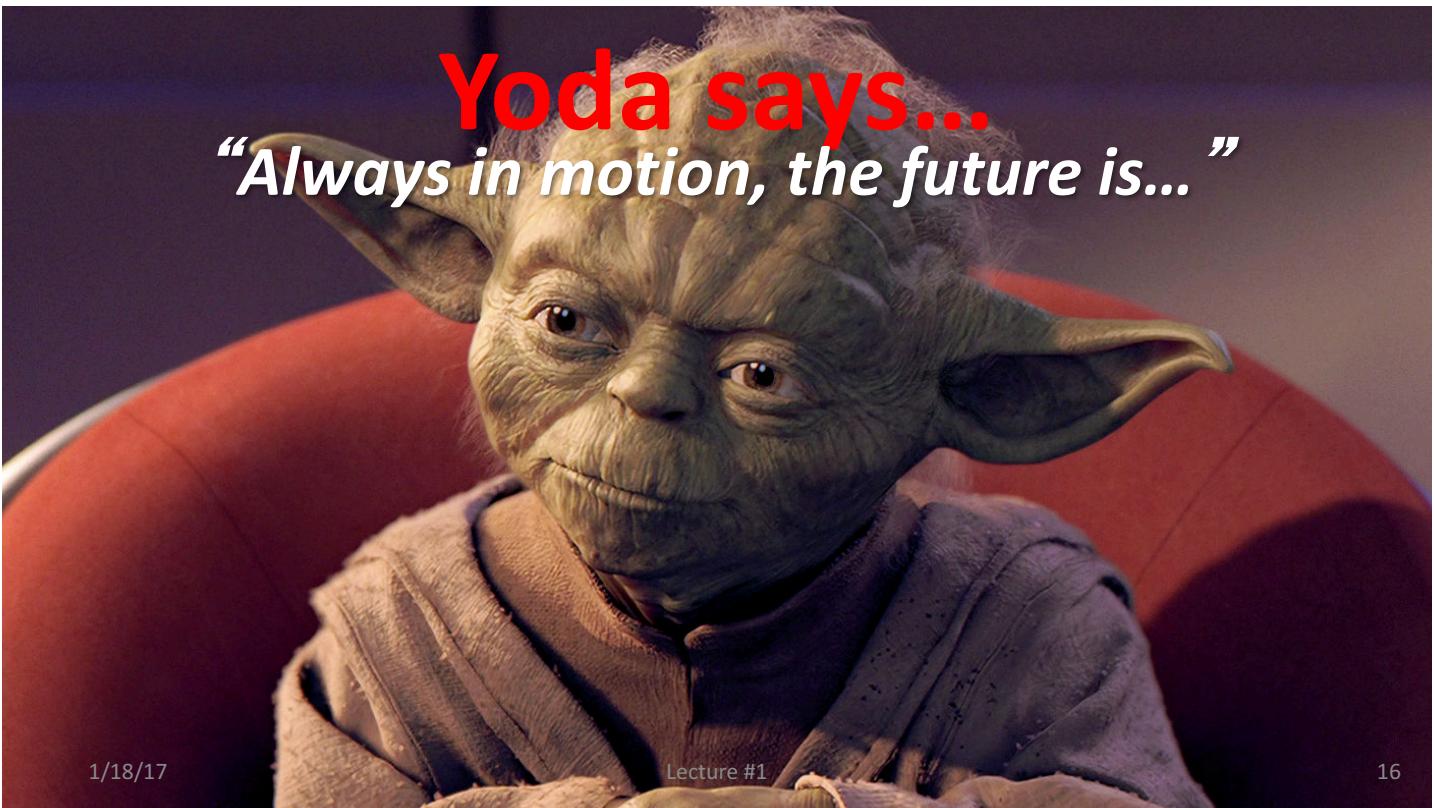
The PostPC Era

- Personal Mobile Device (PMD)
 - Battery operated
 - Connects to the Internet
 - Hundreds of dollars
 - Smart phones, tablets, electronic glasses
- Cloud computing
 - Warehouse Scale Computers (WSC)
 - Software as a Service (SaaS)
 - Portion of software run on a PMD and a portion run in the Cloud
 - Amazon and Google

1/18/17

Lecture #1

15



Yoda says...
“Always in motion, the future is...”

1/18/17

Lecture #1

16

Agenda

- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class

1/18/17

Lecture #1

17

6 Great Ideas in Computer Architecture

1. Abstraction (Layers of Representation/Interpretation) 
2. Moore's Law 
3. Principle of Locality/Memory Hierarchy 
4. Parallelism 
5. Performance Measurement & Improvement 
6. Dependability via Redundancy 

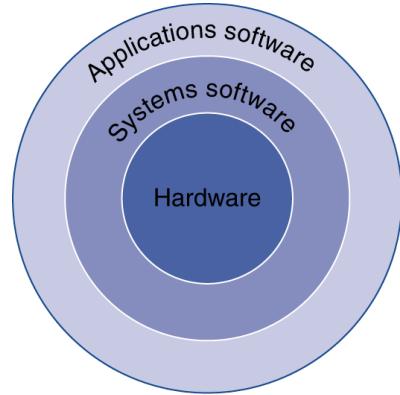
1/18/17

Lecture #1

18

Below Your Program

- Application software
 - Written in high-level language
 - System software
 - Compiler: translates HLL code to machine code
 - Operating System: service code
 - Handling input/output
 - Managing memory and storage
 - Scheduling tasks & sharing resources
 - Hardware
 - Processor, memory, I/O controllers



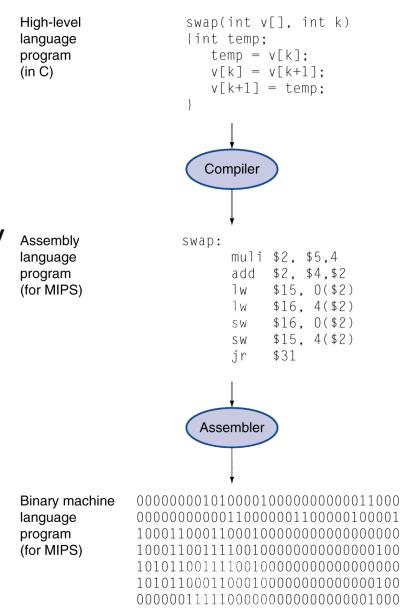
1/18/17

Lecture #1

19

Levels of Program Code

- High-level language
 - Level of abstraction closer to problem domain
 - Provides for productivity and portability
 - Assembly language
 - Textual representation of instructions
 - Hardware representation
 - Binary digits (bits)
 - Encoded instructions and data

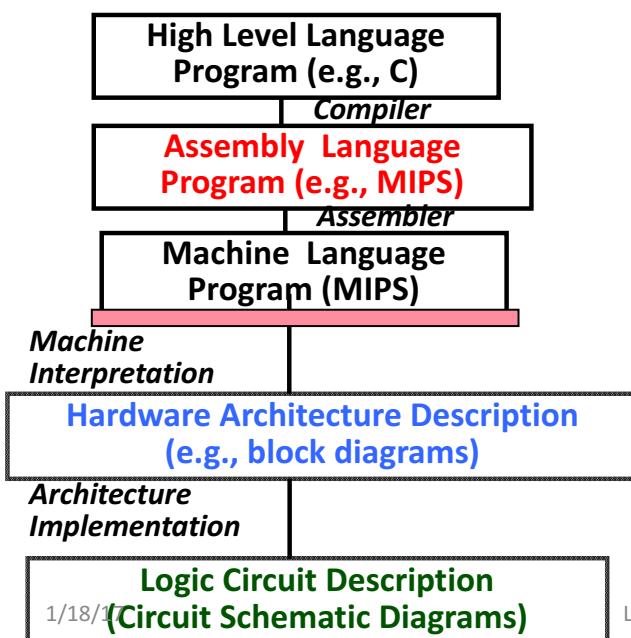


1/18/17

Lecture #1

20

Great Idea #1: Abstraction (Levels of Representation/Interpretation)

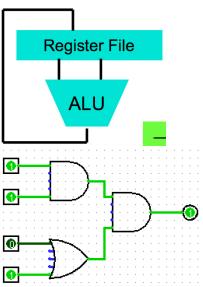


```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw    $t0, 0($2)  
lw    $t1, 4($2)  
sw    $t1, 0($2)  
sw    $t0, 4($2)
```

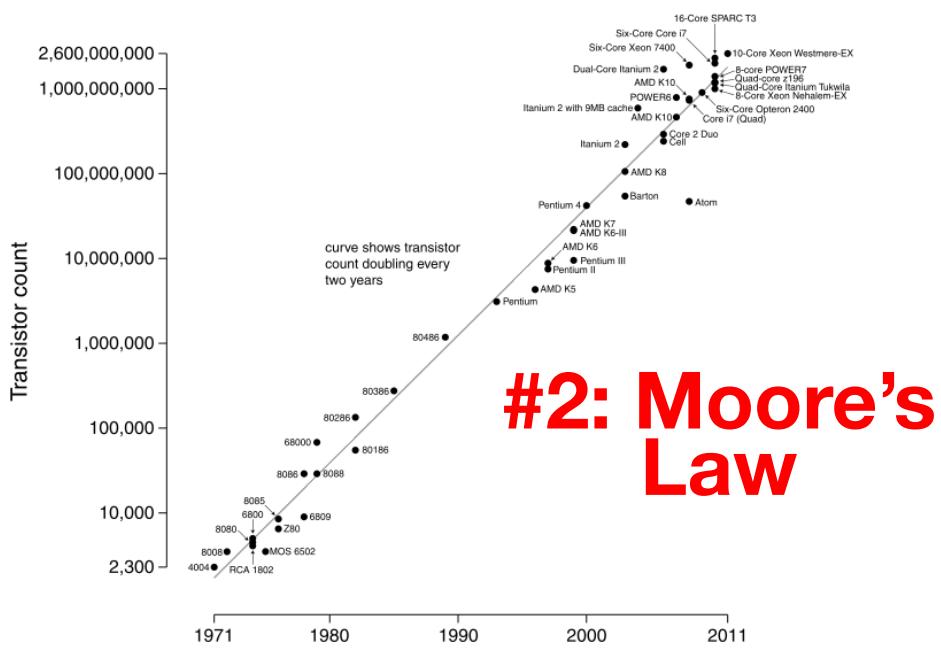
Anything can be represented
as a *number*,
i.e., data or instructions

0000	1001	1100	0110	1010	1111	0101	1000
1010	1111	0101	1000	0000	1001	1100	0110
1100	0110	1010	1111	0101	1000	0000	1001
0101	1000	0000	1001	1100	0110	1010	1111

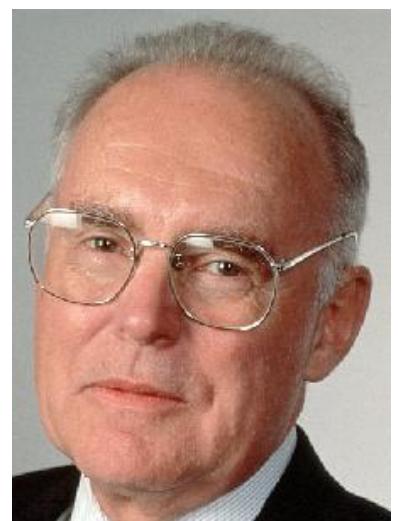


21

Microprocessor Transistor Counts 1971-2011 & Moore's Law



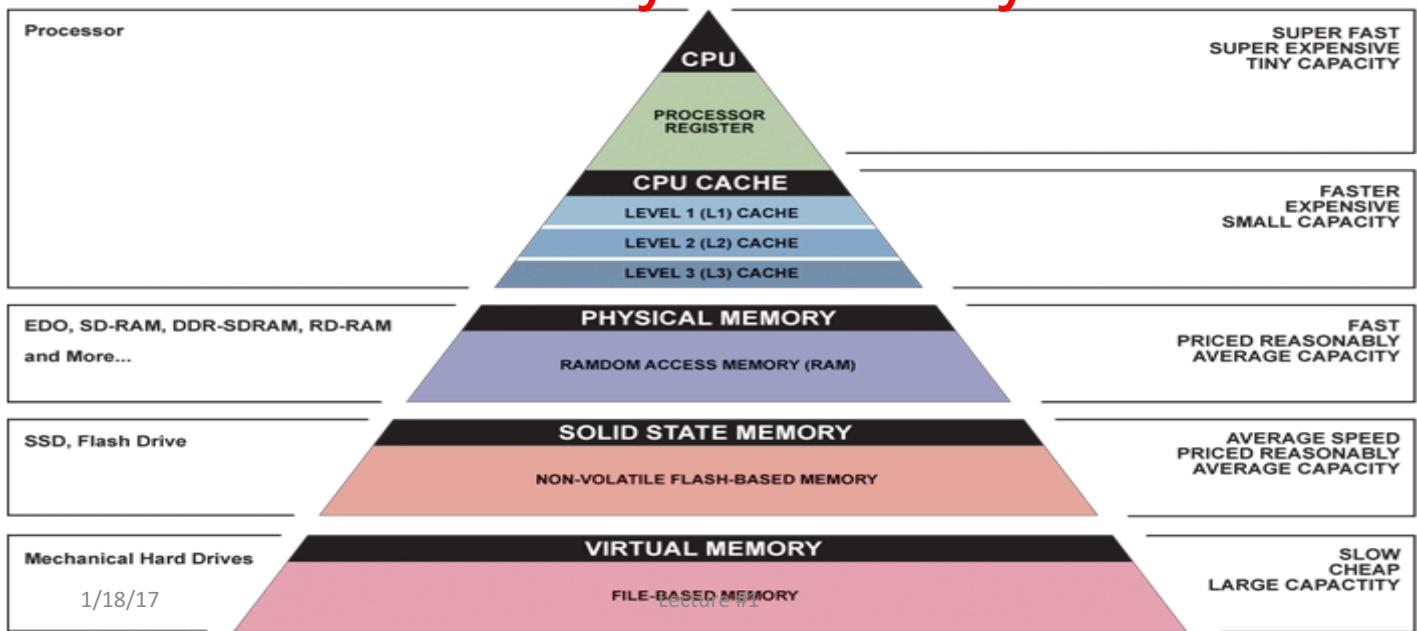
#2: Moore's Law



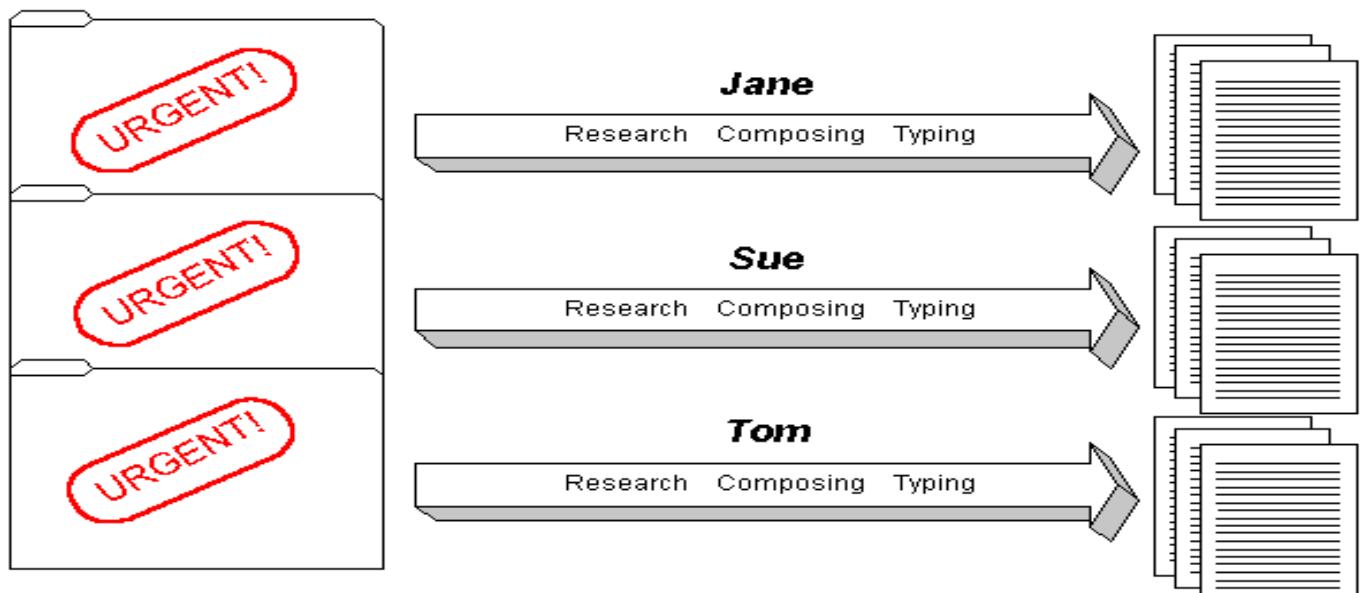
Gordon Moore Intel Cofounder B.S. Cal 1950!

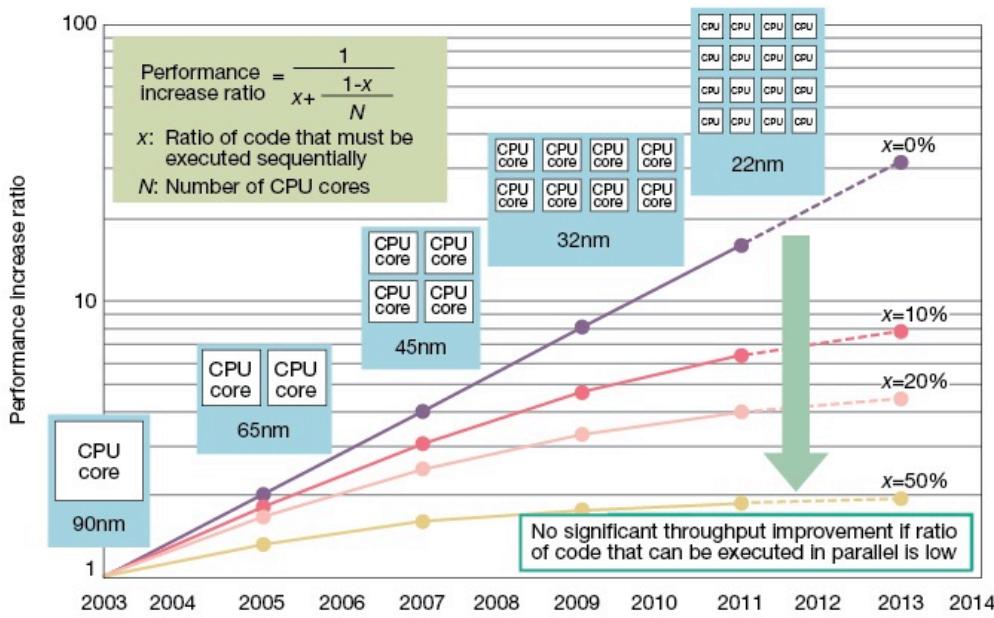
Predicts:
2X Transistors / chip
every 2 years

Great Idea #3: Principle of Locality/ Memory Hierarchy



Great Idea #4: Parallelism





Gene Amdahl
Computer Pioneer

Caveat! Amdahl's Law

25

Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.

Lecture #1

Great Idea #5: Performance Measurement and Improvement

- Matching application to underlying hardware to exploit:
 - Locality
 - Parallelism
 - Special hardware features, like specialized instructions (e.g., matrix manipulation)
- Latency
 - How long to set the problem up
 - How much faster does it execute once it gets going
 - It is all about *time to finish*

Understanding Performance

- Algorithm
 - Determines number of operations executed
- Programming language, compiler, architecture
 - Determine number of machine instructions executed per operation
- Processor and memory system
 - Determine how fast instructions are executed
- I/O system (including OS)
 - Determines how fast I/O operations are executed

1/18/17

Lecture #1

27

Coping with Failures

- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
 - Assume 4% annual failure rate
- On average, how often does a disk fail?
 - a) 1 / month
 - b) 1 / week
 - c) 1 / day
 - d) 1 / hour

1/18/17

Lecture #1

28

Coping with Failures

- 4 disks/server, 50,000 servers
 - Failure rate of disks: 2% to 10% / year
 - Assume 4% annual failure rate
 - On average, how often does a disk fail?
 - a) 1 / month
 - b) 1 / week
 - c) 1 / day
 - d) 1 / hour
- $50,000 \times 4 = 200,000$ disks
- $200,000 \times 4\% = 8000$ disks fail
- $365 \text{ days} \times 24 \text{ hours} = 8760$ hours

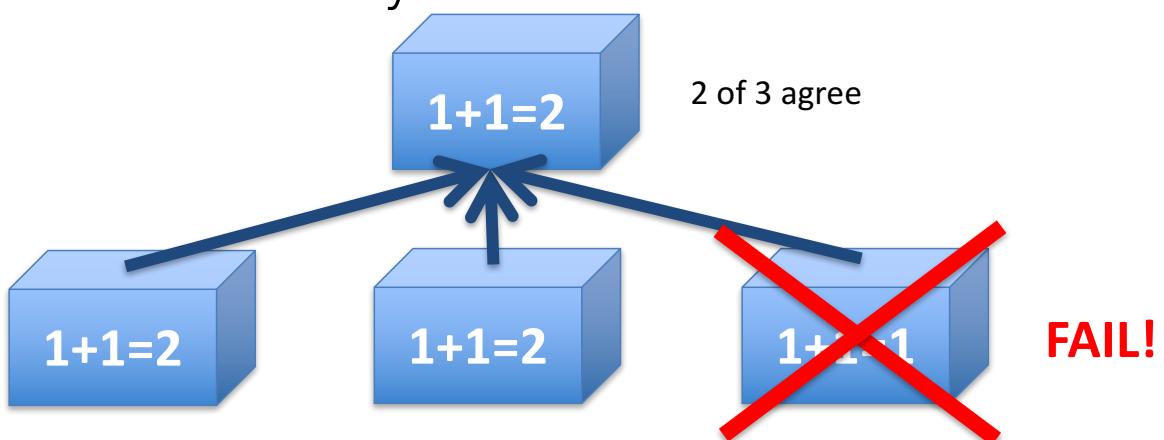
1/18/17

Lecture #1

29

Great Idea #6: Dependability via Redundancy

- Redundancy so that a failing piece doesn't make the whole system fail



Increasing transistor density reduces the cost of redundancy
1/18/17

Lecture #1

30

Great Idea #6: Dependability via Redundancy

- Applies to everything from datacenters to storage to memory to instructors
 - Redundant datacenters so that can lose 1 datacenter but Internet service stays online
 - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
 - Redundant memory bits so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)



1/18/17

Lecture #1

31

Agenda

- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class

1/18/17

Lecture #1

32

Course Information

- Course Web:
 - <http://harmanani.github.io/csc320.html>
 - <http://vlsi.byblos.lau.edu.lb/csc320.html>
- Instructor:
 - **Haidar M. Harmanani, 810 Block A**
- Textbooks:
 - Patterson & Hennessy, Computer Organization and Design, ARM Edition.
 - Kernighan & Ritchie, The C Programming Language, 2nd Edition (Lab Mostly)
- Check **piazza** for announcement, discussion, and clarification
 - Do not send individual emails!

Course Grading

- Midterm (30%)
- Final (40%)
- Homework and Programming Assignments (15%)
 1. Paper and Pencil Textbook Problems (2-4)
 2. MIPS Programming Assignments
 3. Paper and Pencil Computer Design
- Quizzes (15%)
 - 5-7
 - Drop the lowest

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

1/18/17

Lecture #1

35

Peer Instruction



- Increase real-time learning in lecture, test understanding of concepts vs. details
- As complete a “segment” ask multiple choice question
 - 1-2 minutes to decide yourself
 - 2 minutes in pairs/triples to reach consensus.
 - 2 minute discussion of answers, questions, clarifications
- We will be using a free online app



1/18/17

Lecture #1

36

Late Policy ...

- Assignments due at start of the class
- Projects are due by 5 pm
- Every day your project or homework is late, it's 15 points per day.
 - No credit if more than 3 days late

1/18/17

Lecture #1

37

Policy on Assignments and Independent Work

- With the exception of laboratories and assignments (projects and HW) that explicitly permit you to work in groups, all homework and projects are to be YOUR work and your work ALONE.
- It is NOT acceptable to copy solutions from other students.
- It is NOT acceptable to copy (or start your) solutions from the Web.
- PARTNER TEAMS MAY NOT WORK WITH OTHER PARTNER TEAMS
- You are encouraged to help teach other to debug. Beyond that, we don't want you sharing approaches or ideas or code or whiteboarding with other students, since sometimes the point of the assignment is the "algorithm" and if you share that, they won't learn what we want them to learn). We expect that what you hand in is yours.
- **It is NOT acceptable to leave your code anywhere where an unscrupulous student could find and steal it (e.g., public GITHUBs, walking away while leaving yourself logged on, leaving printouts lying around,etc)**
- The first offense is a zero on the assignment and an F in the course the second time
- Both Giver and Receiver are equally culpable and suffer equal penalties

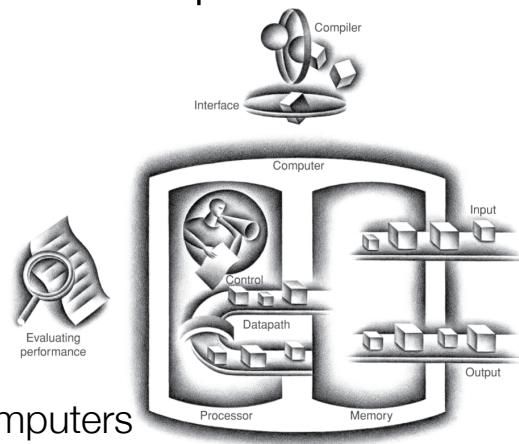
1/18/17

Lecture #1

38

Components of a Computer

- Same components for all kinds of computer
 - Desktop, server, embedded
- Input/output includes
 - User-interface devices
 - Display, keyboard, mouse
 - Storage devices
 - Hard disk, CD/DVD, flash
 - Network adapters
 - For communicating with other computers



1/18/17

Lecture #1

39

Touchscreen

- PostPC device
- Supersedes keyboard and mouse
- Resistive and Capacitive types
 - Most tablets, smart phones use capacitive
 - Capacitive allows multiple touches simultaneously

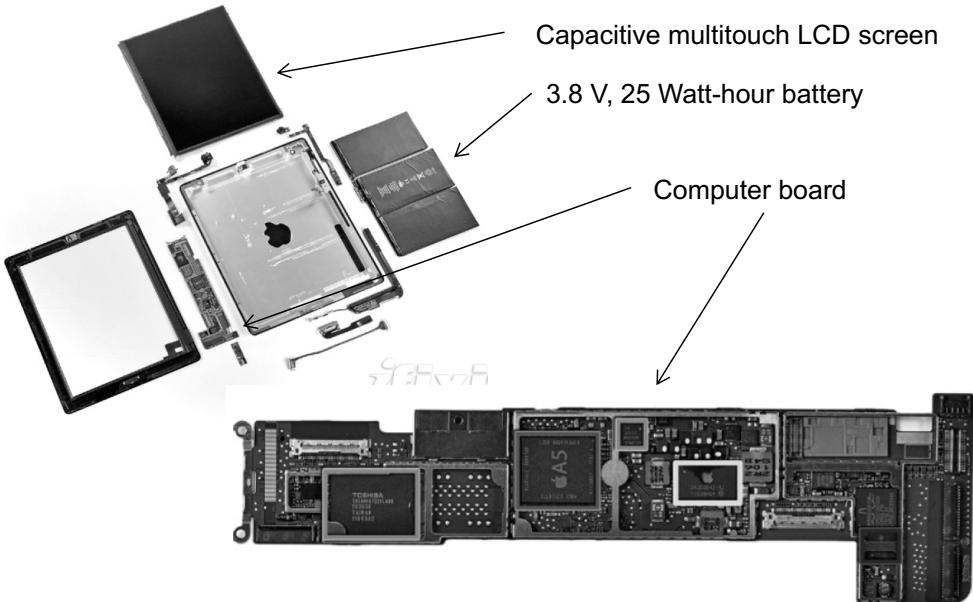


1/18/17

Lecture #1

40

Opening the Box



1/18/17

Lecture #1

41

Inside the Processor (CPU)

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
 - Small fast SRAM memory for immediate access to data

1/18/17

Lecture #1

42

Inside the Processor: Apple A5

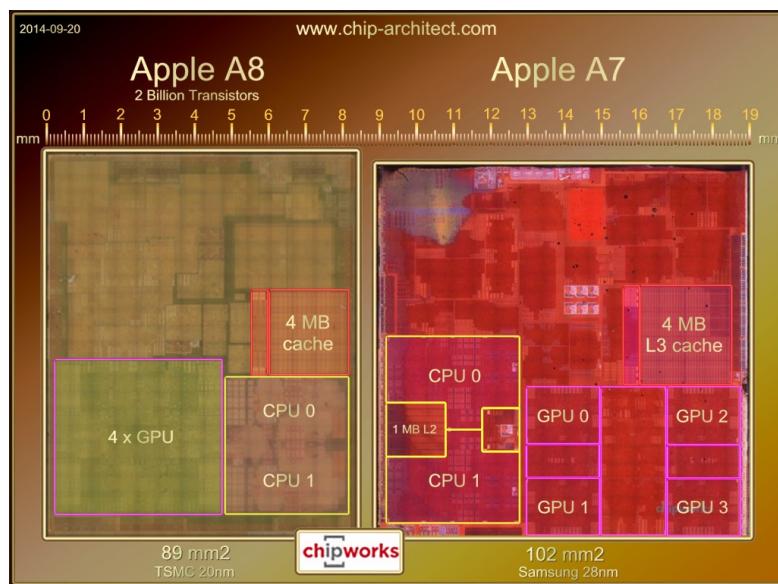


1/18/17

Lecture #1

43

Inside the Processor: Apple A7 and A8



1/18/17

Lecture #1

44

Inside the Processor

Apple A7

- Dual Core
- 64-bit Design
- 28 nm process
- 1 Billion Transistors
- Area = 102 mm²
- Power consumption
 - 1100 mA
- Per Core
 - 64 KB L1 cache for data and 64 KB for instructions
 - 1 MB L2 cache shared by both CPU cores
 - 4 MB L3 cache that services the entire SoC

Apple A8

- Dual Core
- 64-bit Design
- 20 nm process
- 2 Billion Transistors
- Area = 89 mm²
- Power consumption
 - 550 mA
- Per Core
 - 64 KB L1 cache for data and 64 KB for instructions
 - 1 MB L2 cache shared by both CPU cores
 - 4 MB L3 cache that services the entire SoC

1/18/17

Lecture #1

45

Inside the Processor

- The Apple A8 has 25% more CPU performance and 50% more graphics performance while drawing only 50% of the power compared to the Apple A7!

1/18/17

Lecture #1

46

Abstractions

- Abstraction helps us deal with complexity
 - Hide lower-level detail
- Instruction set architecture (ISA)
 - The hardware/software interface
- Application binary interface
 - The ISA plus system software interface
- Implementation
 - The details underlying and interface

1/18/17

Lecture #1

47

A Safe Place for Data

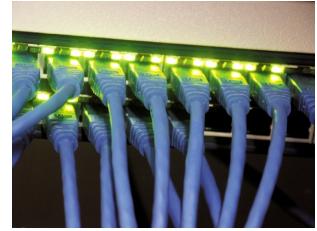
- Volatile main memory
 - Loses instructions and data when power off
- Non-volatile secondary memory
 - Magnetic disk
 - Flash memory
 - Optical disk (CDROM, DVD)



Lecture #1

Networks

- Communication, resource sharing, nonlocal access
- Local area network (LAN): Ethernet
- Wide area network (WAN): the Internet
- Wireless network: WiFi, Bluetooth



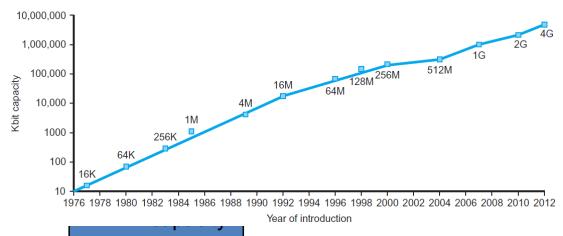
1/18/17

Lecture #1

49

Technology Trends

- Electronics technology continues to evolve
 - 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
2013	Ultra large scale IC	250,000,000,000

1/18/17

Lecture #1

50

Semiconductor Technology

- Silicon: semiconductor
- Add materials to transform properties:
 - Conductors
 - Insulators
 - Switch

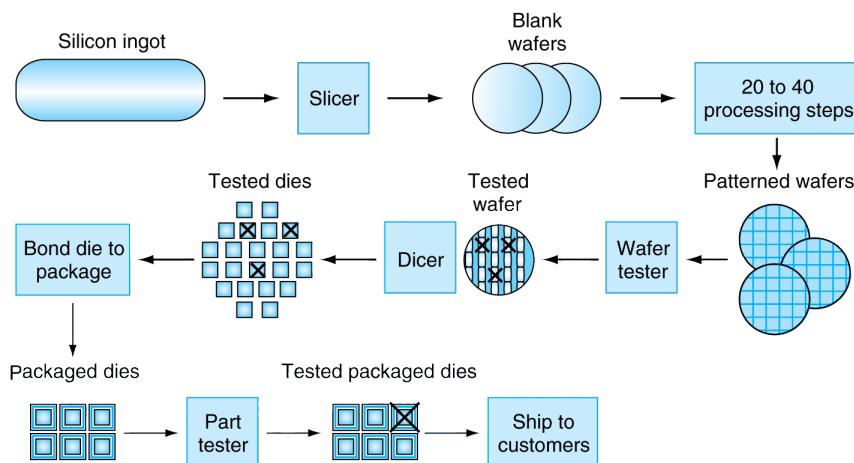
1/18/17

Lecture #1

51

Manufacturing ICs

- Yield: proportion of working dies per wafer

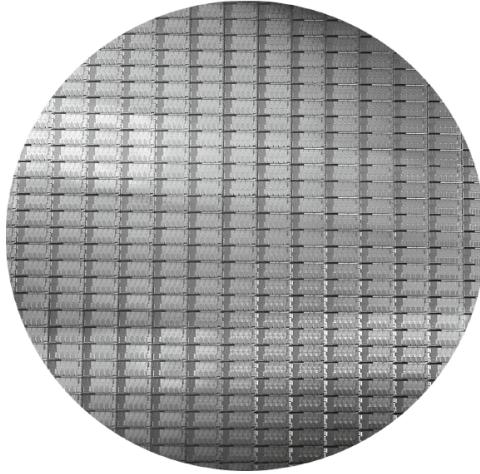


1/18/17

Lecture #1

52

Intel Core i7 Wafer



- 300mm wafer, 280 chips, 32nm technology
- Each chip is 20.7 x 10.5 mm

Integrated Circuit Cost

- Nonlinear relation to area and defect rate
 - Wafer cost and area are fixed
 - Defect rate determined by manufacturing process
 - Die area determined by architecture and circuit design

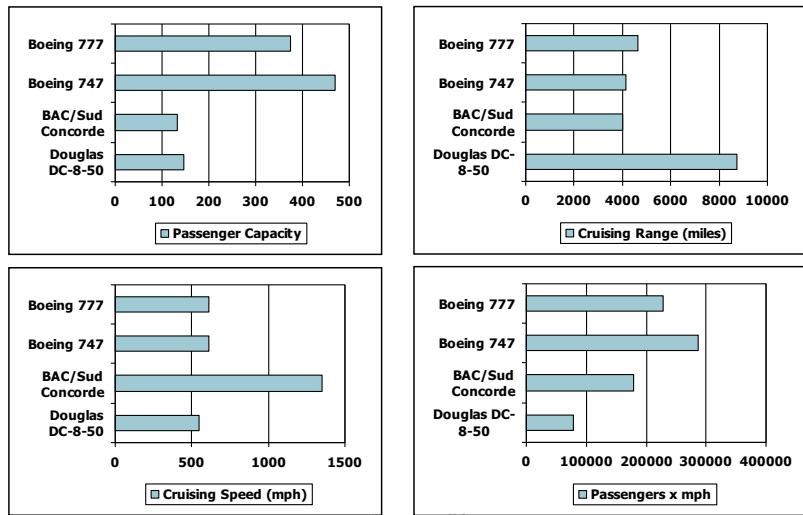
$$\text{Cost per die} = \frac{\text{Cost per wafer}}{\text{Dies per wafer} \times \text{Yield}}$$

$$\text{Dies per wafer} \approx \text{Wafer area}/\text{Die area}$$

$$\text{Yield} = \frac{1}{(1 + (\text{Defects per area} \times \text{Die area}/2))^2}$$

Defining Performance

- Which airplane has the best performance?



1/18/17

Lecture #1

55

Response Time and Throughput

- Response time
 - How long it takes to do a task
- Throughput
 - Total work done per unit time
 - e.g., tasks/transactions/... per hour
- How are response time and throughput affected by
 - Replacing the processor with a faster version?
 - Adding more processors?
- We'll focus on response time for now...

1/18/17

Lecture #1

56

Relative Performance

- Define Performance = 1/Execution Time
- “X is n time faster than Y”

$$\begin{aligned} & \text{Performance}_X / \text{Performance}_Y \\ &= \text{Execution time}_Y / \text{Execution time}_X = n \end{aligned}$$

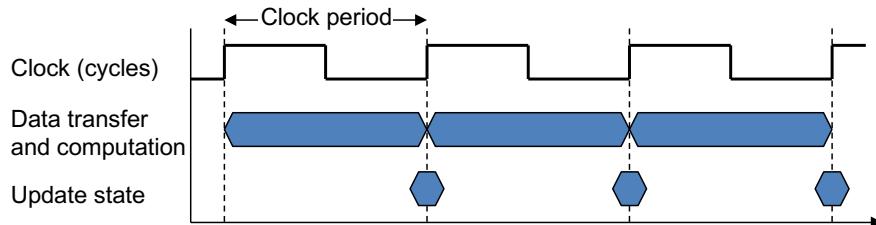
- Example: time taken to run a program
 - 10s on A, 15s on B
 - $\text{Execution Time}_B / \text{Execution Time}_A = 15s / 10s = 1.5$
 - So A is 1.5 times faster than B

Measuring Execution Time

- Elapsed time
 - Total response time, including all aspects
 - Processing, I/O, OS overhead, idle time
 - Determines system performance
- CPU time
 - Time spent processing a given job
 - Discounts I/O time, other jobs' shares
 - Comprises user CPU time and system CPU time
 - Different programs are affected differently by CPU and system performance

CPU Clocking

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



- Clock period = $1/\text{frequency}$ = duration of a clock cycle
 - e.g., $250\text{ps} = 0.25\text{ns} = 250 \times 10^{-12}\text{s}$
- Clock rate or frequency: cycles per second
 - e.g., $4.0\text{GHz} = 4000\text{MHz} = 4.0 \times 10^9\text{Hz}$

CPU Clocking

- If a CPU operates:
 - At 100 Hz, its "clock cycle" is 0.01 second = 10 ms
 - At 100 MHz, its clock cycle is 0.000 000 01 second = 10 ns.
- Recall
 - millisecond (ms): 0.001 second
 - microsecond (μs): 0.000 001 second
 - nanosecond (ns): 0.000 000 001 second
 - picoseconds (ps): 0.000 000 000 001 second
 - 1 kilohertz (kHz) = 1 000 cycles / second
 - 1 megahertz (MHz) = 1 000 000 cycles / second
 - 1 gigahertz (GHz) = 1 000 000 000 cycles / second

CPU Time

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

$$\begin{aligned}\text{CPU Time} &= \text{CPU Clock Cycles} \times \text{Clock Cycle Time} \\ &= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}\end{aligned}$$

1/18/17

Lecture #1

61

CPU Time Example

- Computer A: 2GHz clock, 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?

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

$$\begin{aligned}\text{Clock Cycles}_A &= \text{CPU Time}_A \times \text{Clock Rate}_A \\ &= 10s \times 2\text{GHz} = 20 \times 10^9\end{aligned}$$

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

1/18/17

Lecture #1

62

Instruction Count and CPI

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

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

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

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

1/18/17

Lecture #1

63

CPI Example

- 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?

$$\begin{aligned}\text{CPU Time}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\ &= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps} \quad \text{A is faster...} \\ \text{CPU Time}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\ &= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps} \\ \frac{\text{CPU Time}_B}{\text{CPU Time}_A} &= \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2 \quad \dots \text{by this much}\end{aligned}$$

1/18/17

Lecture #1

64

CPI in More Detail

- If different instruction classes take different numbers of cycles

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

- Weighted average CPI

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

CPI Example

- 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
 $= 2 \times 1 + 1 \times 2 + 2 \times 3$
 $= 10$
- Avg. CPI = $10/5 = 2.0$

- Sequence 2: IC = 6

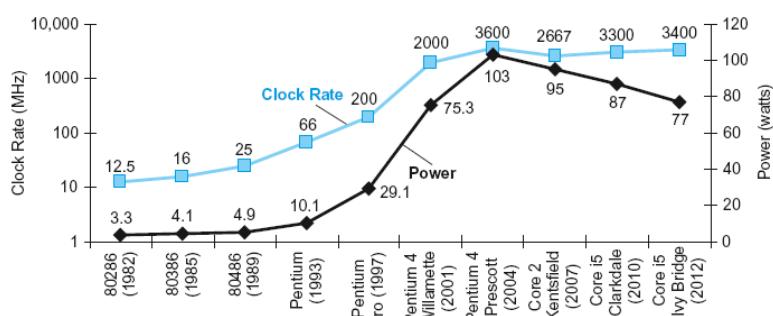
- Clock Cycles
 $= 4 \times 1 + 1 \times 2 + 1 \times 3$
 $= 9$
- Avg. CPI = $9/6 = 1.5$

Performance Summary

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

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

Power Trends



- In CMOS IC technology

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

× 30

5V → 1V

× 1000

Reducing Power

- Suppose a new CPU has
 - 85% of capacitive load of old CPU
 - 15% voltage and 15% frequency reduction

$$\frac{P_{\text{new}}}{P_{\text{old}}} = \frac{C_{\text{old}} \times 0.85 \times (V_{\text{old}} \times 0.85)^2 \times F_{\text{old}} \times 0.85}{C_{\text{old}} \times V_{\text{old}}^2 \times F_{\text{old}}} = 0.85^4 = 0.52$$

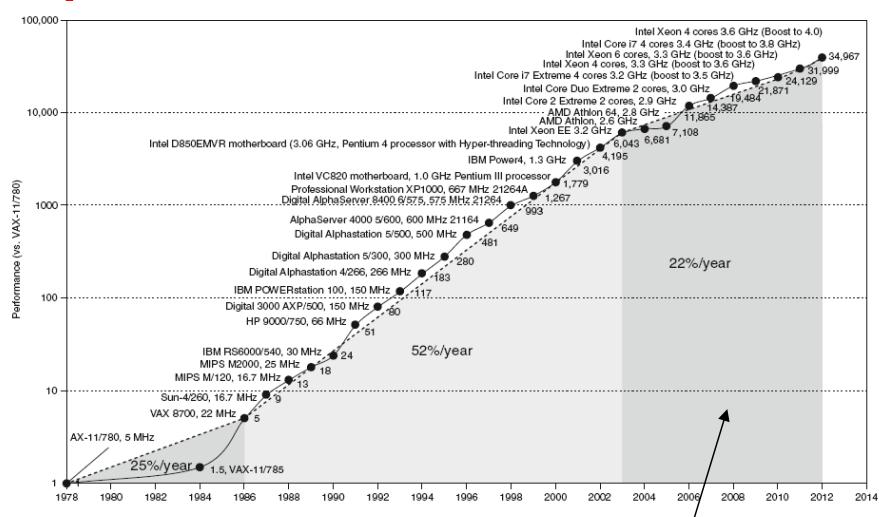
- The power wall
 - We can't reduce voltage further
 - We can't remove more heat
- How else can we improve performance?

1/18/17

Lecture #1

69

Uniprocessor Performance



Constrained by power, instruction-level parallelism,
memory latency

*The Sea Change: The
Switch to Multiprocessors*

1/18/17

Lecture #1

70

Multiprocessors

- Multicore microprocessors
 - More than one processor per chip
- Requires explicitly parallel programming
 - Compare with instruction level parallelism
 - Hardware executes multiple instructions at once
 - Hidden from the programmer
 - Hard to do
 - Programming for performance
 - Load balancing
 - Optimizing communication and synchronization

1/18/17

Lecture #1

71

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)

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

1/18/17

Lecture #1

72

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

SPEC Power Benchmark

- Power consumption of server at different workload levels
 - Performance: ssj_ops/sec
 - Power: Watts (Joules/sec)

$$\text{Overall ssj_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj_ops}_i \right) / \left(\sum_{i=0}^{10} \text{power}_i \right)$$

SPECpower_ssj2008 for Xeon X5650

Target Load %	Performance (ssj_ops)	Average Power (Watts)
100%	865,618	258
90%	786,688	242
80%	698,051	224
70%	607,826	204
60%	521,391	185
50%	436,757	170
40%	345,919	157
30%	262,071	146
20%	176,061	135
10%	86,784	121
0%	0	80
Overall Sum	4,787,166	1,922
$\Sigma \text{ssj_ops} / \Sigma \text{power} =$		2,490

1/18/17

Lecture #1

75

Fallacies and Pitfalls

1/18/17

Lecture #1

76

Pitfall: Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance

$$T_{\text{improved}} = \frac{T_{\text{affected}}}{\text{improvement factor}} + T_{\text{unaffected}}$$

- Example: multiply accounts for 80s/100s
 - How much improvement in multiply performance to get 5x overall?
- Corollary: make the common case fast

Fallacy: Low Power at Idle

- Look back at i7 power benchmark
 - At 100% load: 258W
 - At 50% load: 170W (66%)
 - At 10% load: 121W (47%)
- Google data center
 - Mostly operates at 10% – 50% load
 - At 100% load less than 1% of the time
- Consider designing processors to make power proportional to load

Pitfall: MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
 - Doesn't account for
 - Differences in ISAs between computers
 - Differences in complexity between instructions

$$\begin{aligned} \text{MIPS} &= \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} \\ &= \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6} \end{aligned}$$

- CPI varies between programs on a given CPU

Concluding Remarks

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