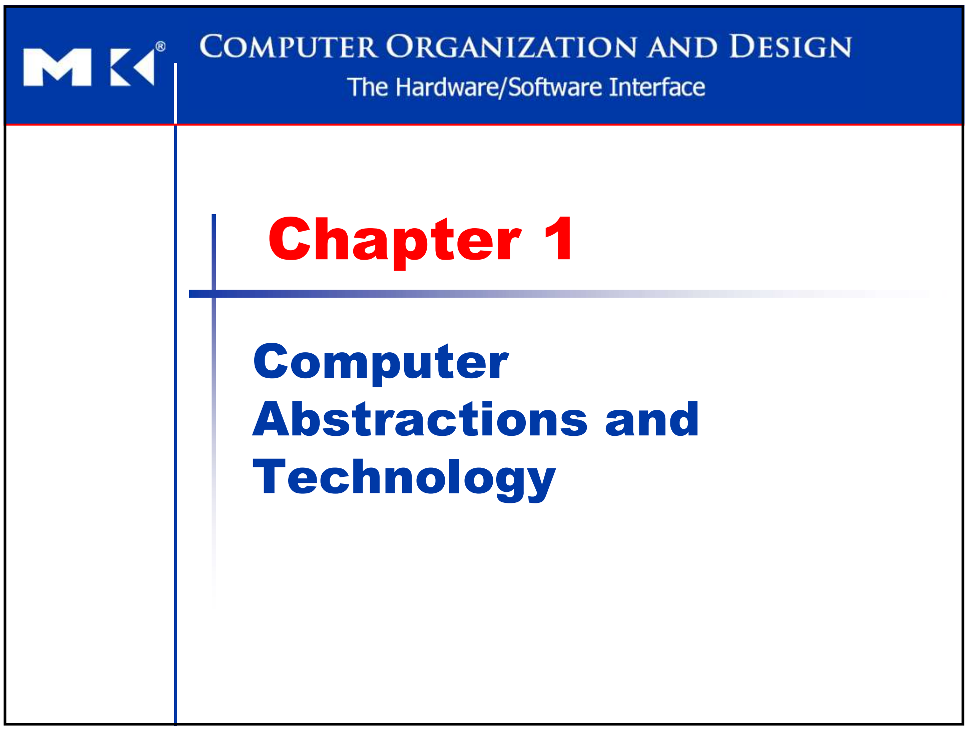
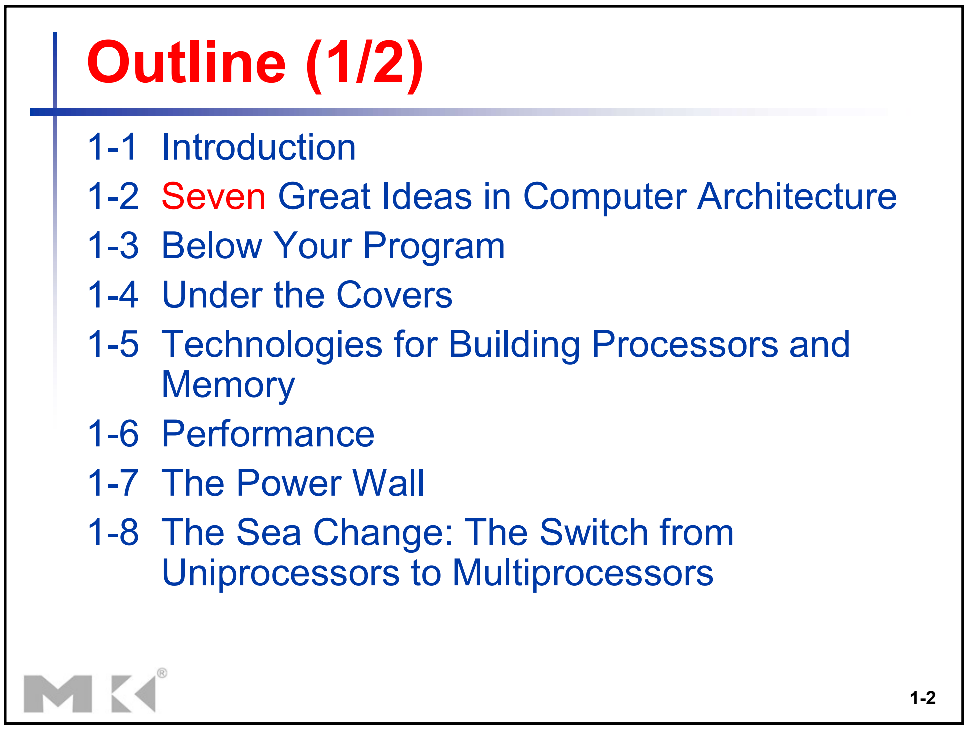


COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface

Chapter 1

Computer Abstractions and Technology

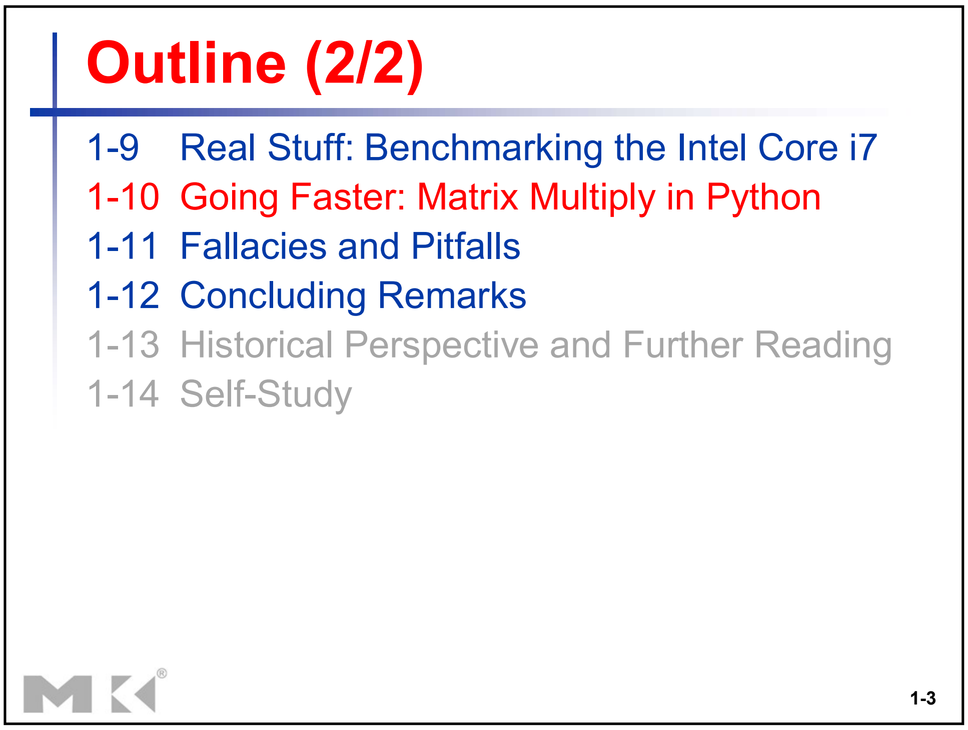
1



Outline (1/2)

1-2

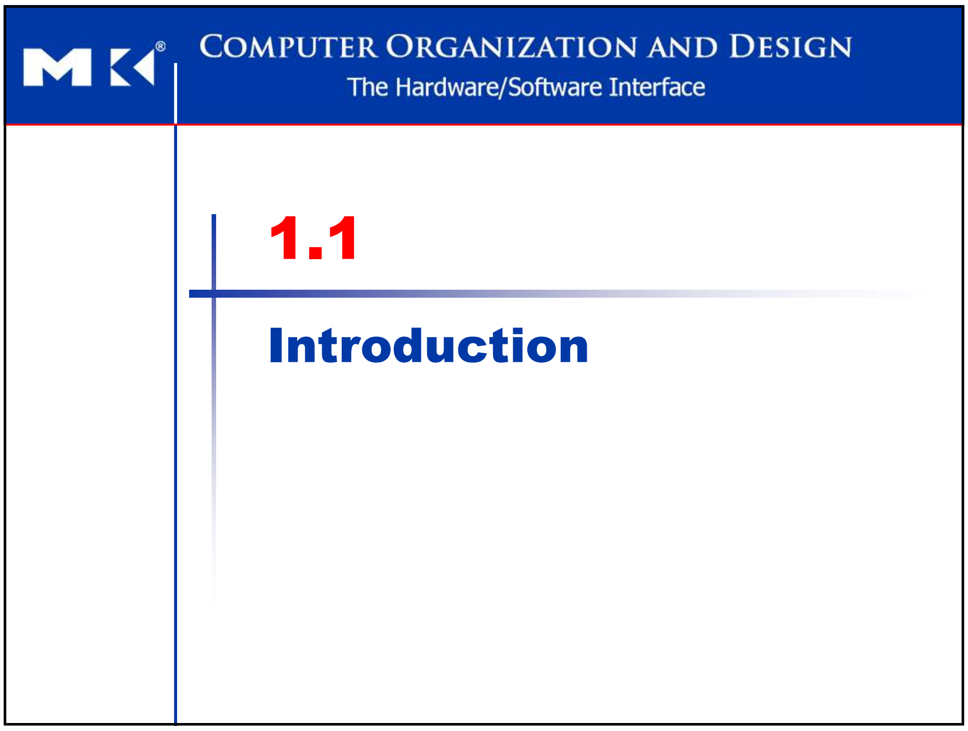
2



Outline (2/2)

1-3

3



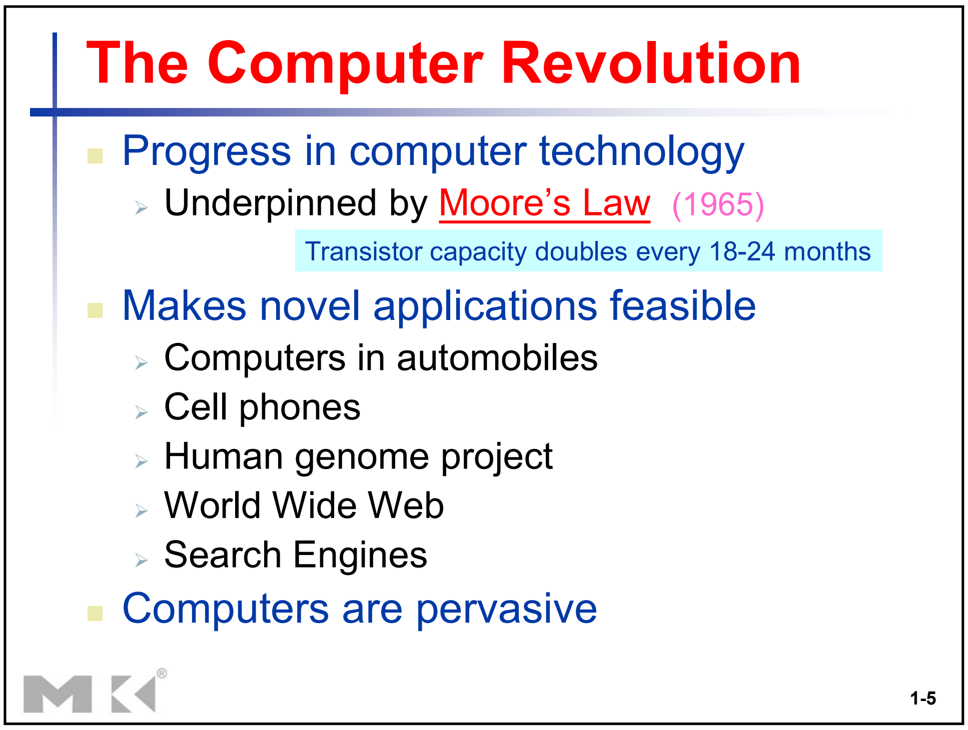
COMPUTER ORGANIZATION AND DESIGN

The Hardware/Software Interface

1.1

Introduction

4



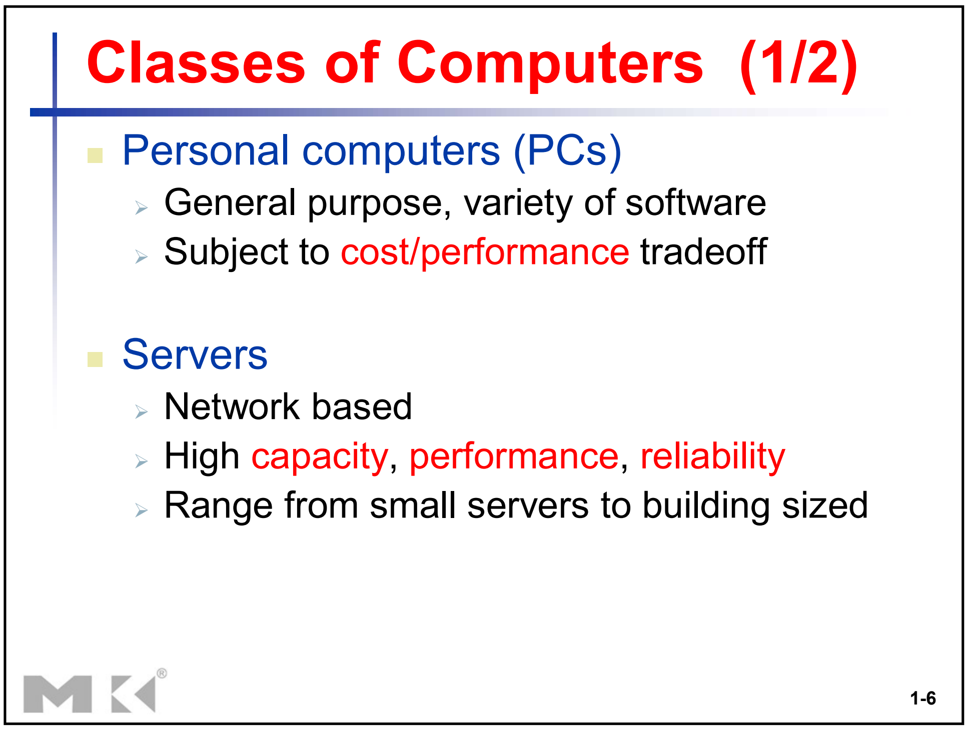
The Computer Revolution

- Progress in computer technology
  - Underpinned by Moore's Law (1965)

    Transistor capacity doubles every 18-24 months
- Makes novel applications feasible
  - Computers in automobiles
  - Cell phones
  - Human genome project
  - World Wide Web
  - Search Engines
- Computers are pervasive

1-5

5

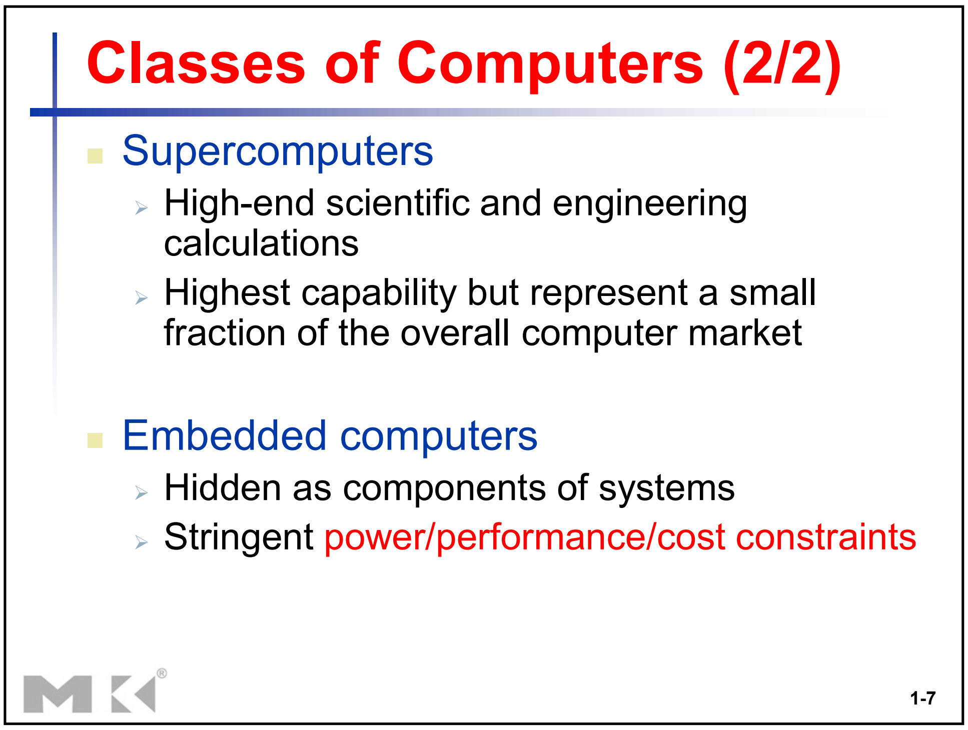


Classes of Computers (1/2)

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

1-6

6

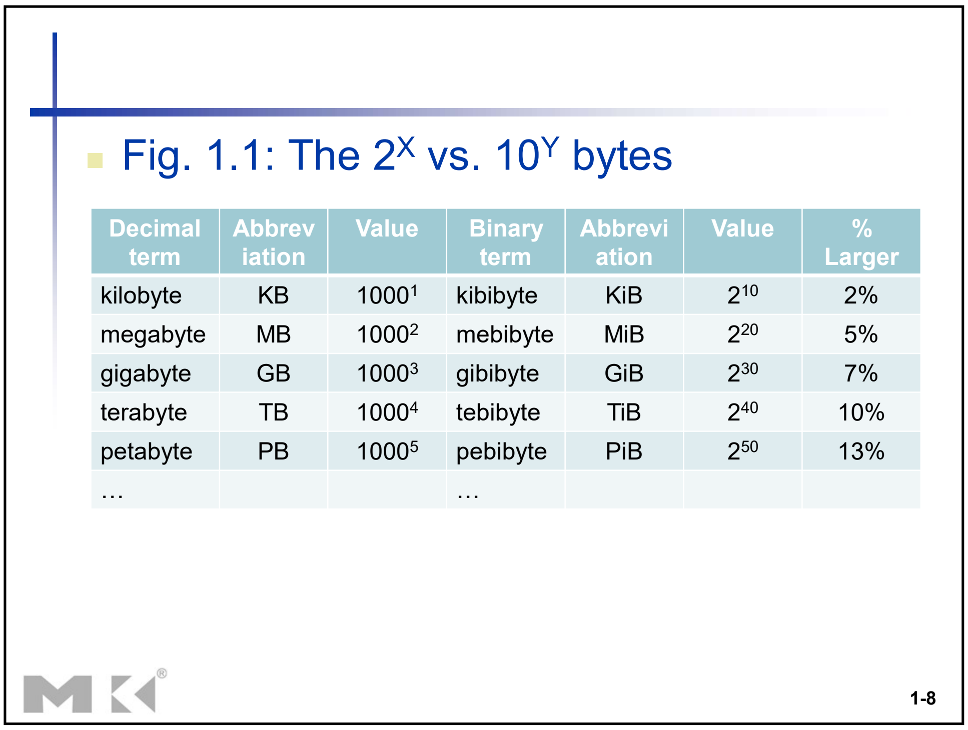


Classes of Computers (2/2)

- 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

1-7

7



- Fig. 1.1: The 2^X vs. 10^Y bytes

Decimal term	Abbreviation	Value	Binary term	Abbreviation	Value	% Larger
kilobyte	KB	1000^1	kibibyte	KiB	2^{10}	2%
megabyte	MB	1000^2	mebibyte	MiB	2^{20}	5%
gigabyte	GB	1000^3	gibibyte	GiB	2^{30}	7%
terabyte	TB	1000^4	tebibyte	TiB	2^{40}	10%
petabyte	PB	1000^5	pebibyte	PiB	2^{50}	13%
…			…			

1-8

8

The PostPC Era

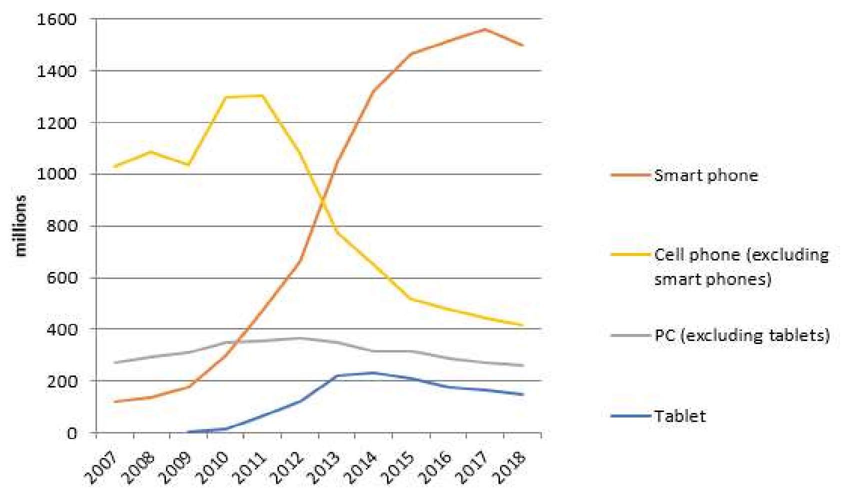


FIGURE 1.2 The number manufactured per year of tablets and smart phones, which reflect the PostPC era, versus personal computers and traditional cell phones. Smart phones represent the recent growth in the cell phone industry, and they passed PCs in 2011.

1-9

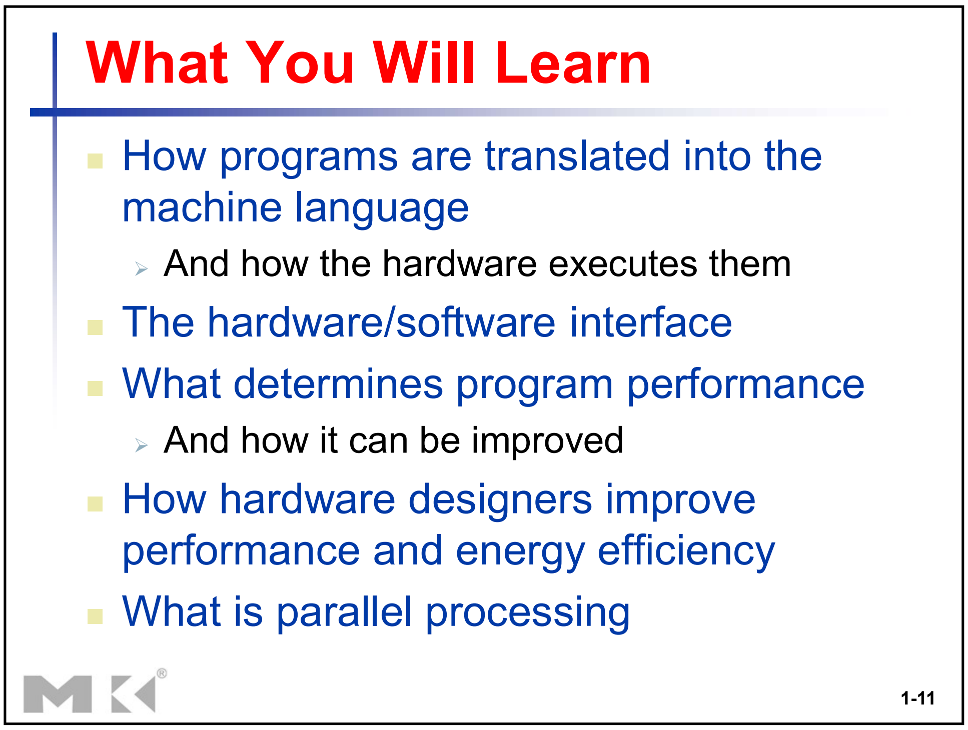
9

The PostPC Era (2/2)

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

Chapter 1 — Computer Abstractions and Technology — 10

10

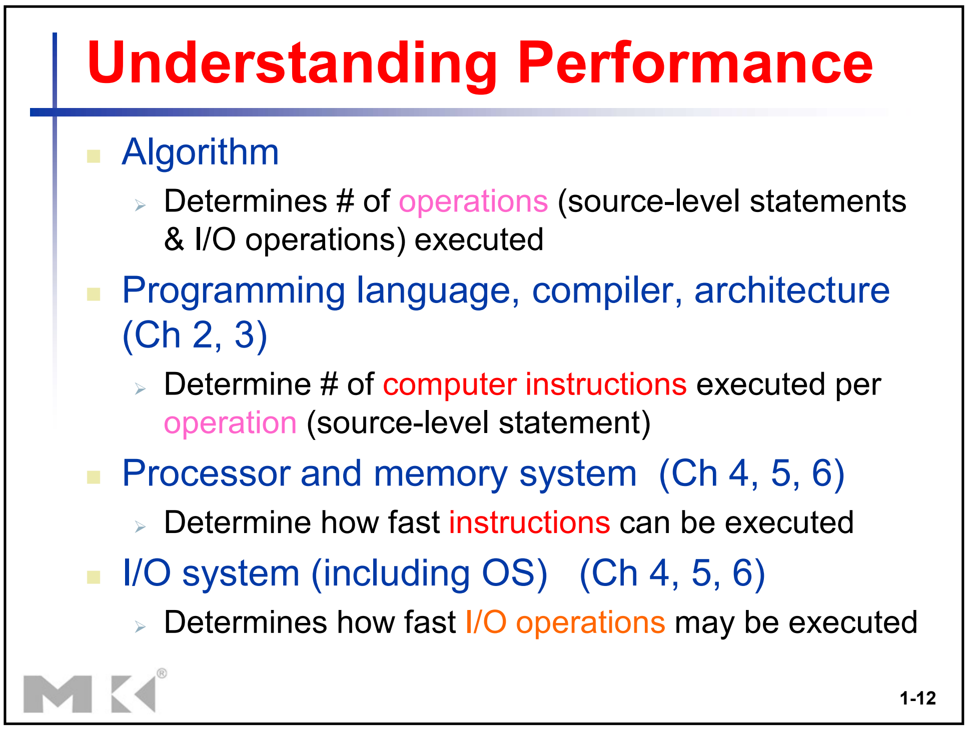


What You Will Learn

- 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 and energy efficiency
- What is parallel processing

1-11

11

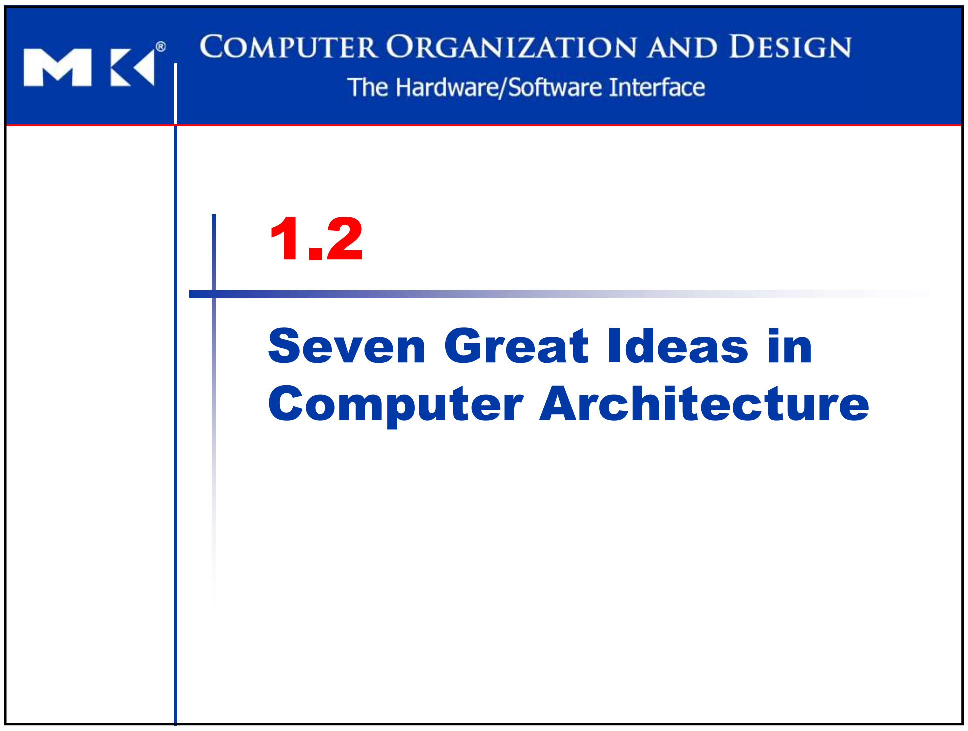


Understanding Performance

- Algorithm
  - Determines # of operations (source-level statements & I/O operations) executed
- Programming language, compiler, architecture (Ch 2, 3)
  - Determine # of computer instructions executed per operation (source-level statement)
- Processor and memory system (Ch 4, 5, 6)
  - Determine how fast instructions can be executed
- I/O system (including OS) (Ch 4, 5, 6)
  - Determines how fast I/O operations may be executed

1-12

12



MK®

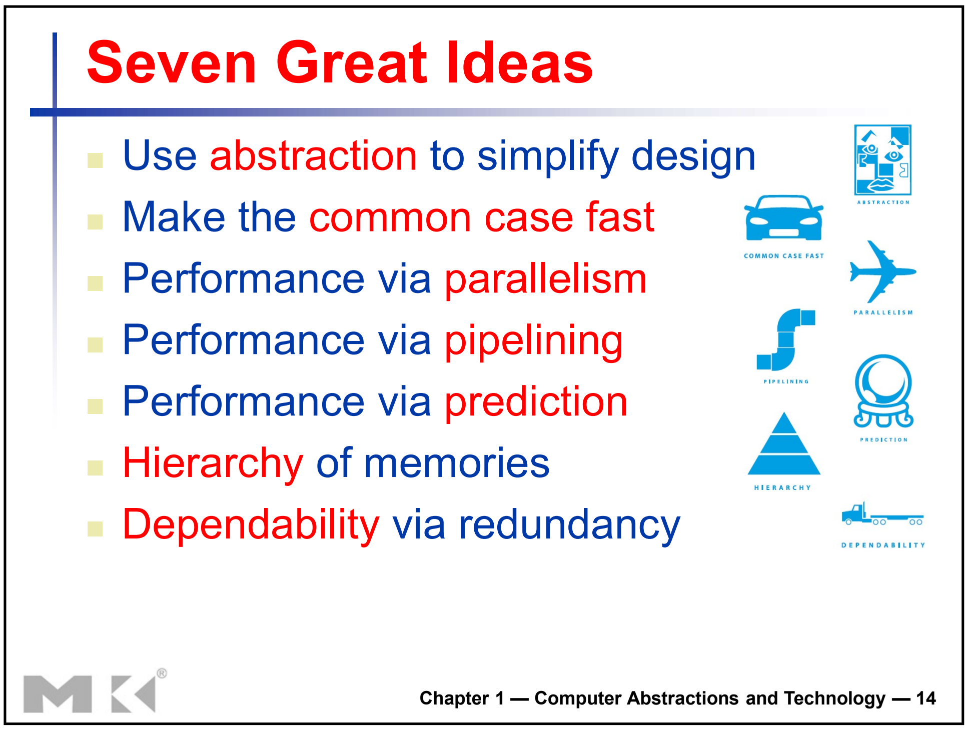
COMPUTER ORGANIZATION AND DESIGN

The Hardware/Software Interface

1.2

Seven Great Ideas in Computer Architecture

13



Seven Great Ideas

- Use abstraction to simplify design
- Make the common case fast
- Performance via parallelism
- Performance via pipelining
- Performance via prediction
- Hierarchy of memories
- Dependability via redundancy

ABSTRACTION
COMMON CASE FAST
PARALLELISM
PIPELINING
PREDICTION
HIERARCHY
DEPENDABILITY

MK®

Chapter 1 — Computer Abstractions and Technology — 14

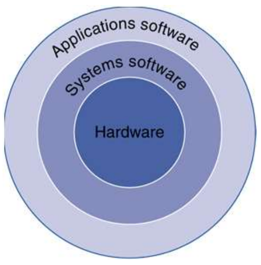
14

1.3

Below Your Program

15

Below Your Program

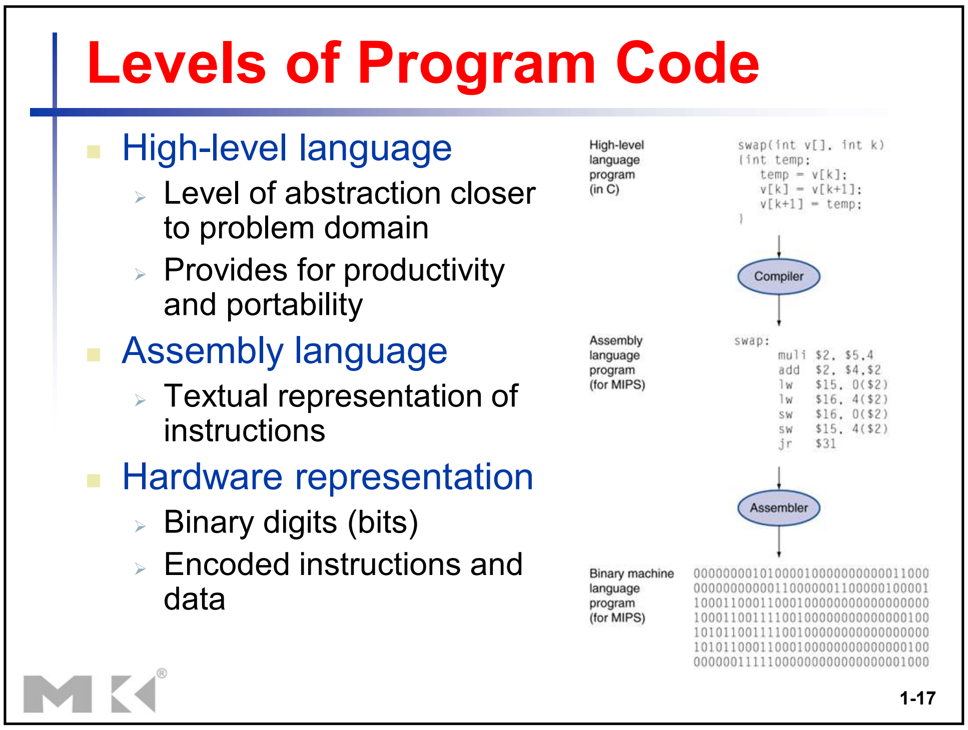


Hierarchical layers of SW & HW

- 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

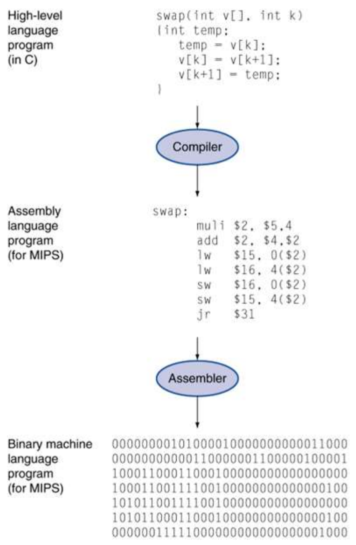
1-16

16



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


High-level language program (in C)
Compiler
Assembly language program (for MIPS)
Assembler
Binary machine language program (for MIPS)

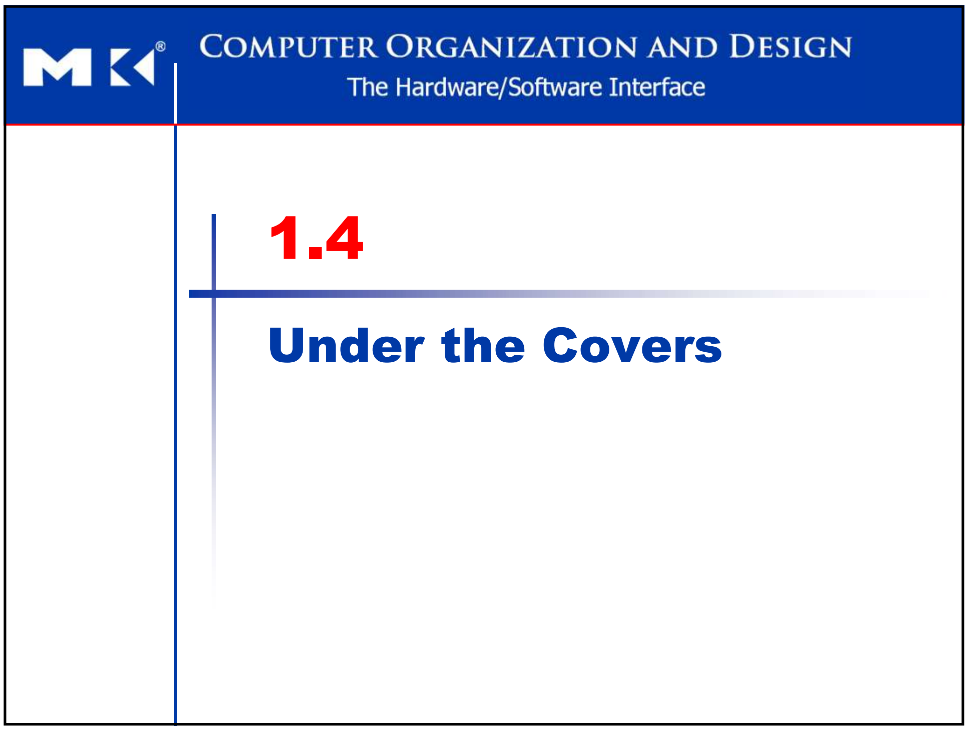
```
swap(int v[]. int k)
(int temp:
   temp = v[k]:
   v[k] = v[k+1]:
   v[k+1] = temp:
)
```

```
swap:
      muli $2. $5.4
      add  $2. $4.$2
      lw   $15. 0($2)
      lw   $16. 4($2)
      sw   $16. 0($2)
      sw   $15. 4($2)
      jr   $31
```

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

1-17

17



COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface

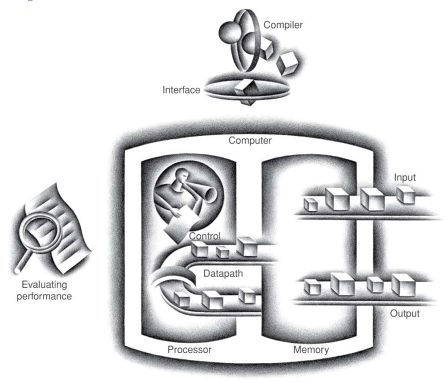
1.4

Under the Covers

18

Components of a Computer

The BIG Picture



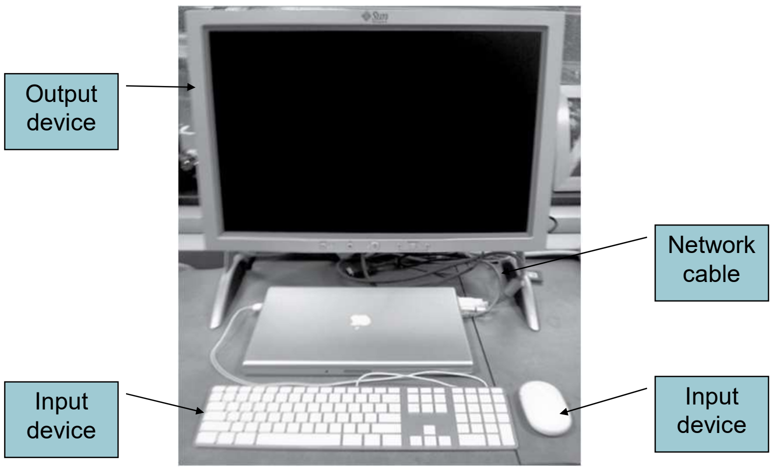
5 classic components of a computer

- 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

1-19

19

Anatomy of a Computer

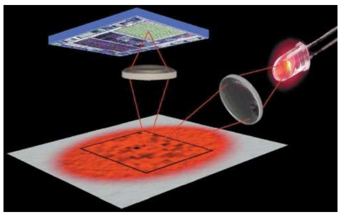


1-20

20

Mouse

- Optical mouse
  - LED illuminates desktop
  - Small low-res camera
  - Basic image processor
    - Looks for x, y movement
  - Buttons & wheel
- Supersedes roller-ball electromechanical mouse

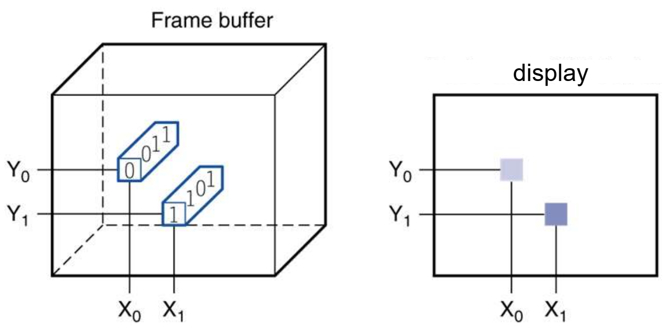


1-21

21

LCD Screen

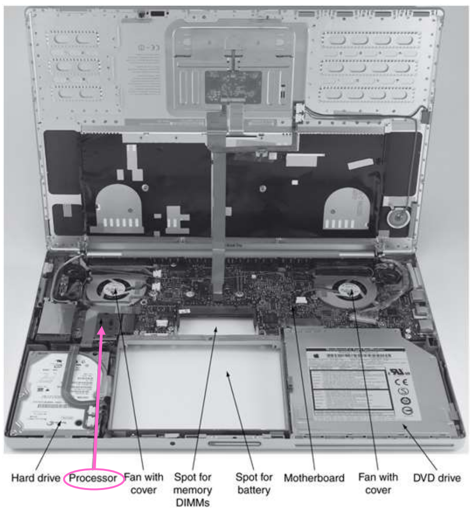
- LCD screen: picture elements (pixels)
  - a thin, low-power display
  - Mirrors content of frame buffer memory



1-22

22

Opening the Box



1-23

23

Inside the Processor (CPU)

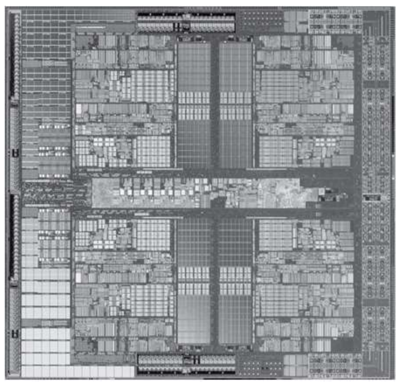
- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory
  - Small fast SRAM memory for immediate access to data

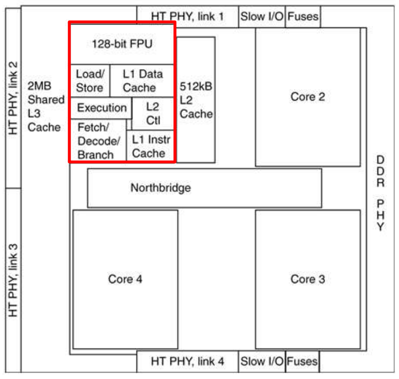
1-24

24

Inside the Processor

- AMD Barcelona: 4 processor cores



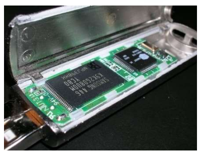


1-25

25

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)

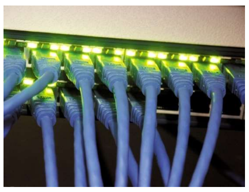


1-26

26

Networks

- Communication and resource sharing
- Local area network (LAN): Ethernet
  - Within a building
- Wide area network (WAN): Internet
- Wireless network: WiFi, Bluetooth



1-27

27

Anatomy of a Tablet

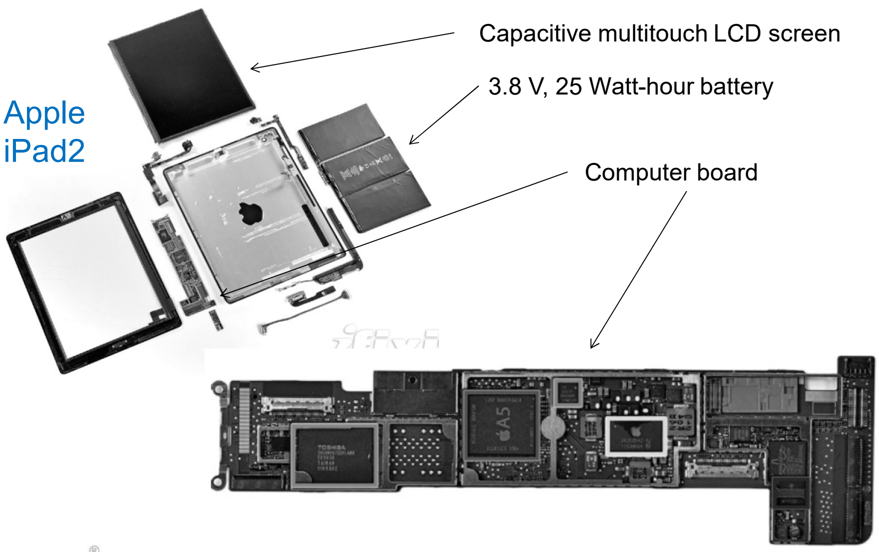
- PostPC device
- Touchscreen
  - Supersedes keyboard and mouse
  - Resistive and Capacitive types
    - Most tablets, smart phones use capacitive
    - Capacitive allows multiple touches simultaneously

1-28

28

Opening the Box

Apple iPad2

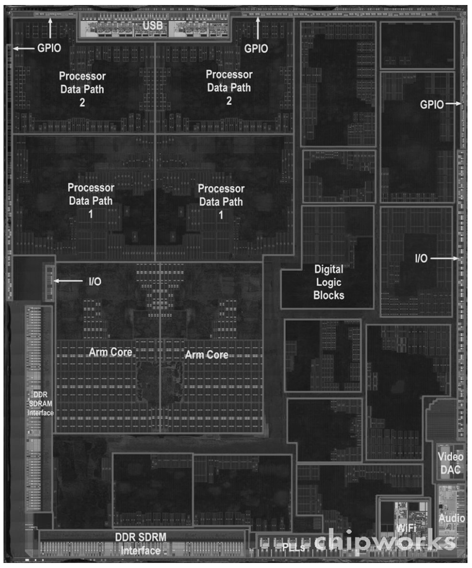


1-29

29

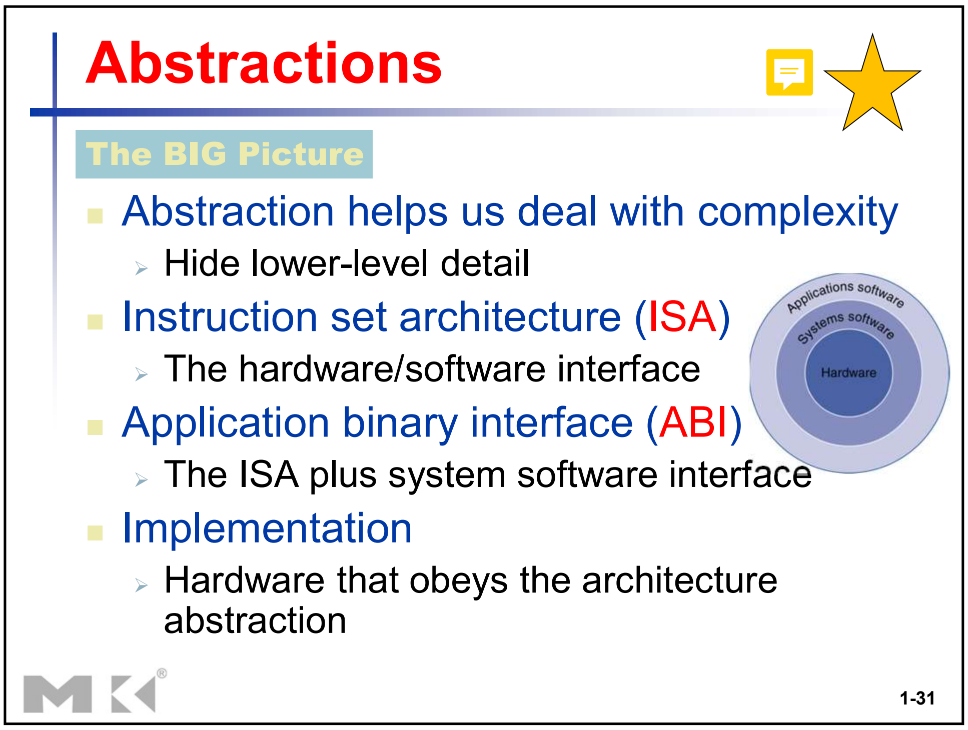
Inside the Processor

- Apple A5
  - 2 ARM cores
  - 1 PowerVR GPU (graphical processing unit) with 4 datapath



1-30

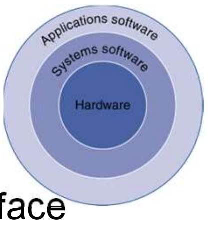
30



Abstractions

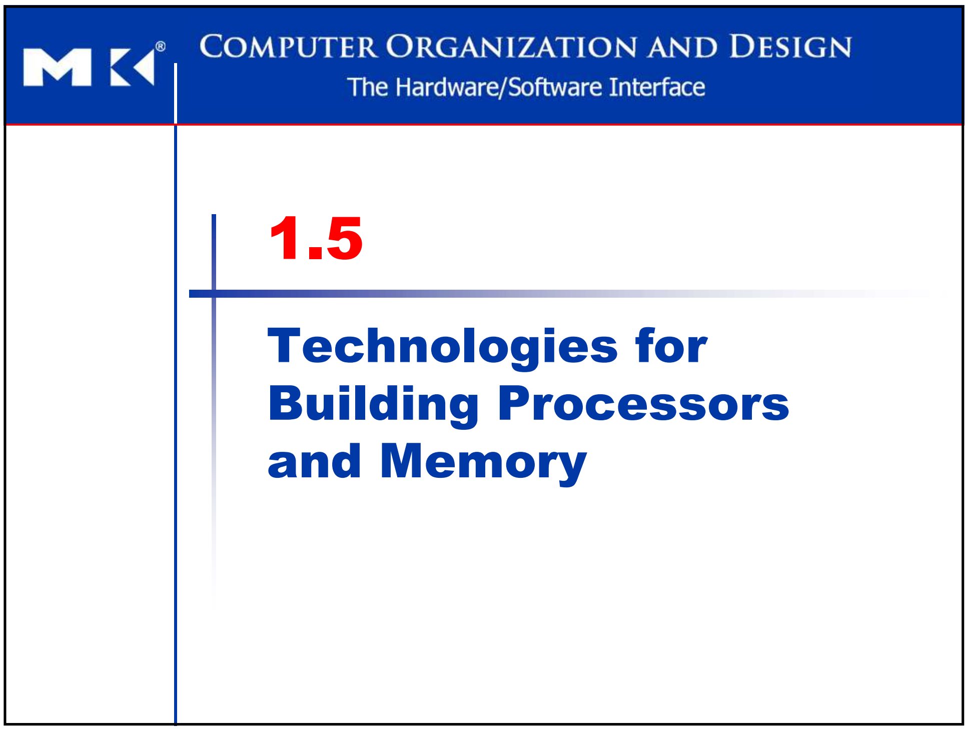
The BIG Picture

- Abstraction helps us deal with complexity
  - Hide lower-level detail
- Instruction set architecture (ISA)
  - The hardware/software interface
- Application binary interface (ABI)
  - The ISA plus system software interface
- Implementation
  - Hardware that obeys the architecture abstraction



1-31

31

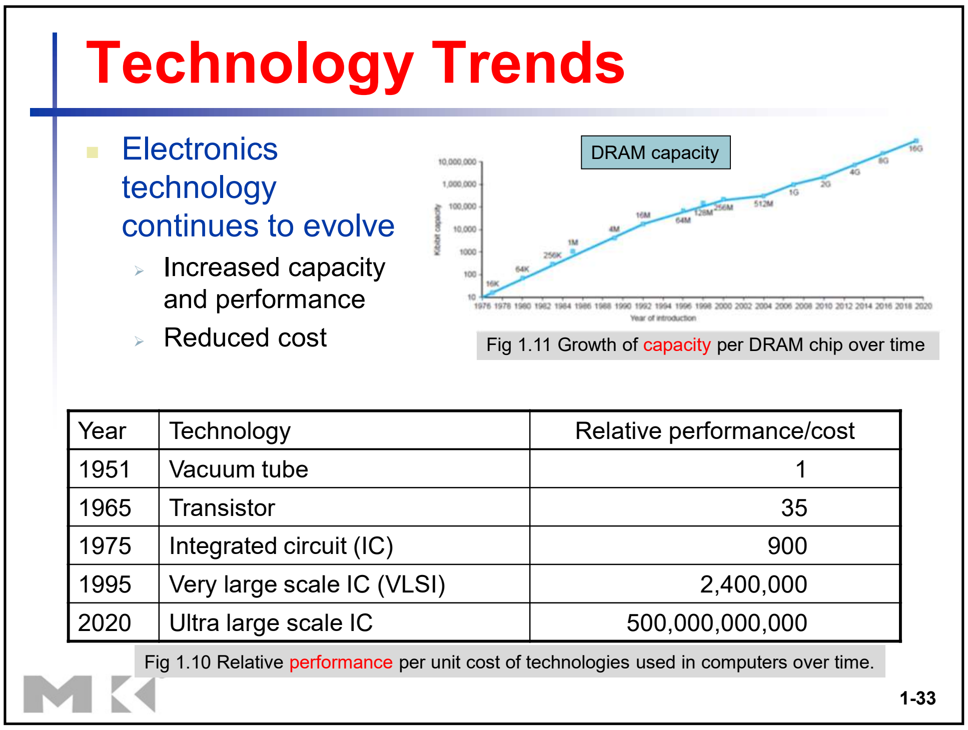


COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface

1.5

Technologies for Building Processors and Memory

32



Technology Trends

- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost

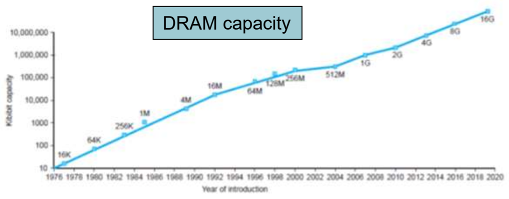


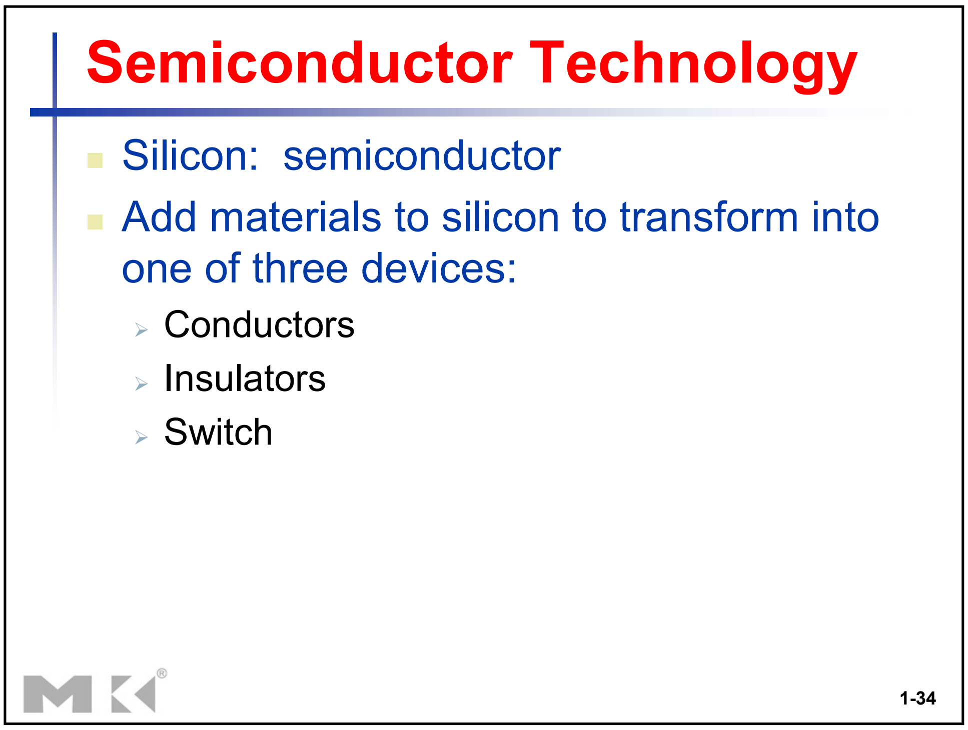
Fig 1.11 Growth of capacity per DRAM chip over time

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
2020	Ultra large scale IC	500,000,000,000

Fig 1.10 Relative performance per unit cost of technologies used in computers over time.

1-33

33

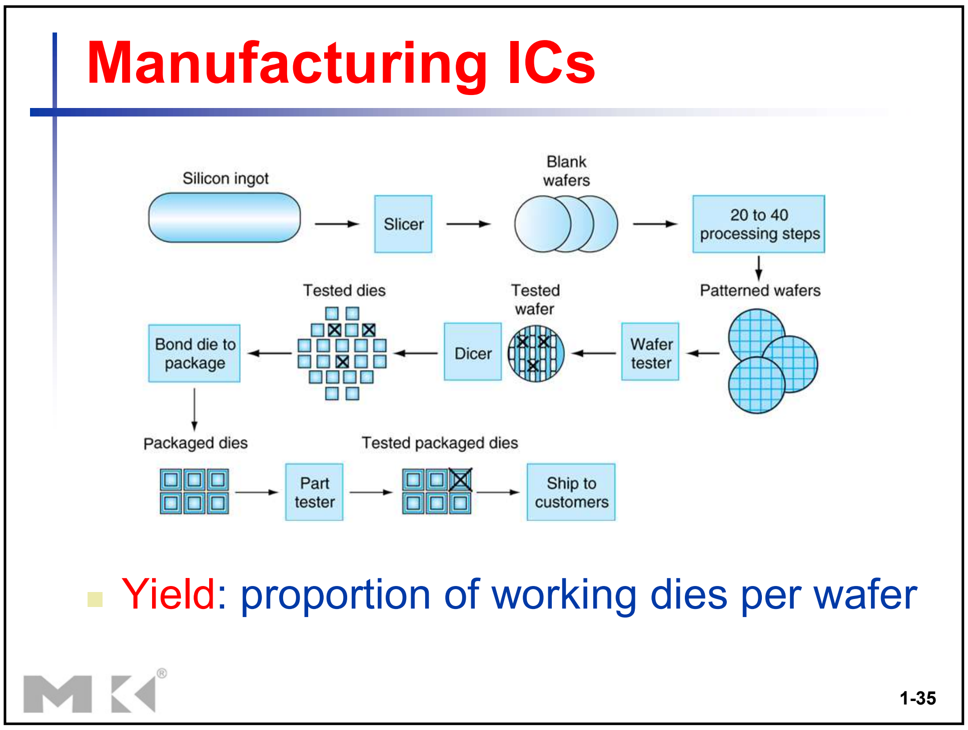


Semiconductor Technology

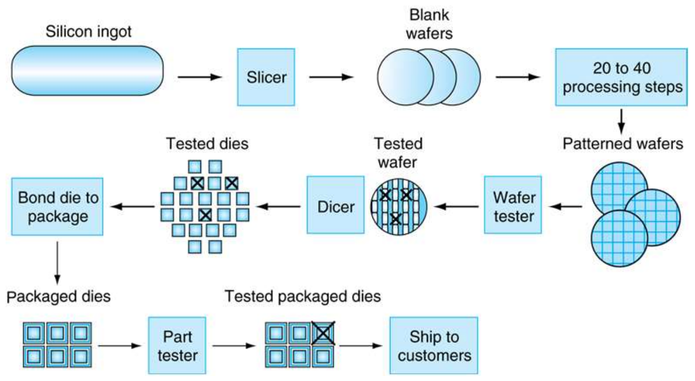
- Silicon: semiconductor
- Add materials to silicon to transform into one of three devices:
  - Conductors
  - Insulators
  - Switch

1-34

34



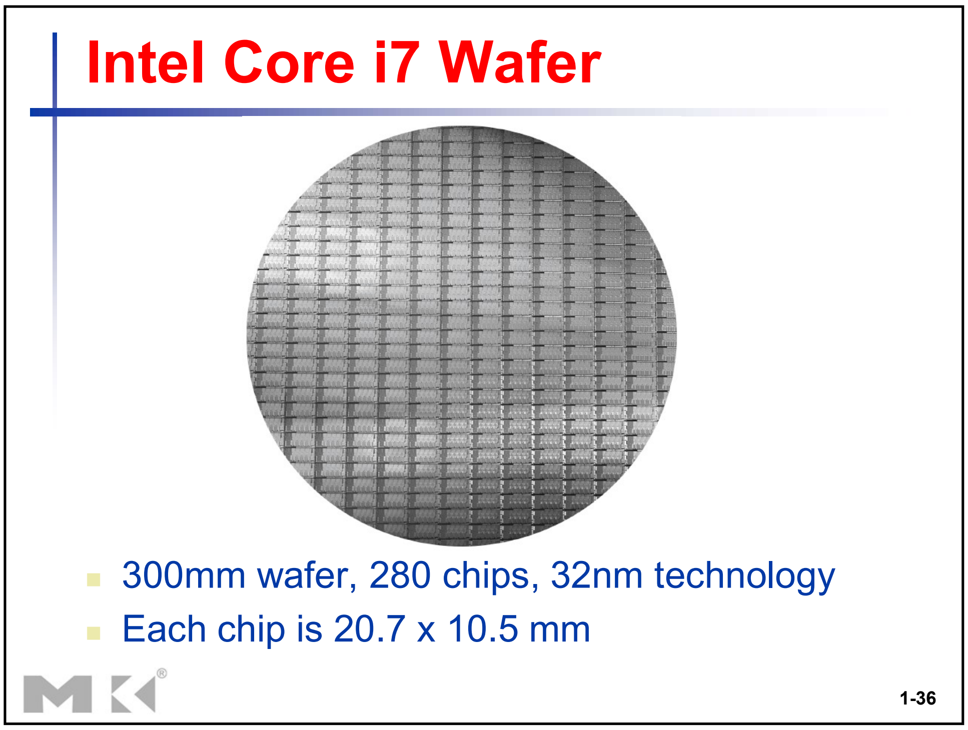
Manufacturing ICs



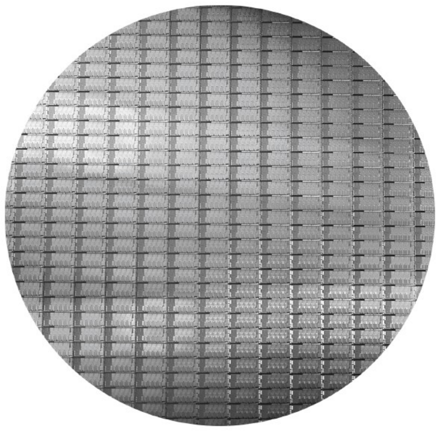
- Yield: proportion of working dies per wafer

1-35

35



Intel Core i7 Wafer



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

1-36

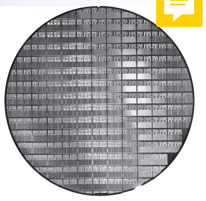
36

Integrated Circuit Cost

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

$$Dies\ per\ wafer \approx \frac{Wafer\ area}{Die\ area}$$

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



⇒ empirical observation

- Wafer cost & area: fixed
- Die area:
  - determined by architecture and circuit design
- Defect rate: nonlinear relation to area
  - determined by manufacturing process

1-37

37

COMPUTER ORGANIZATION AND DESIGN

The Hardware/Software Interface

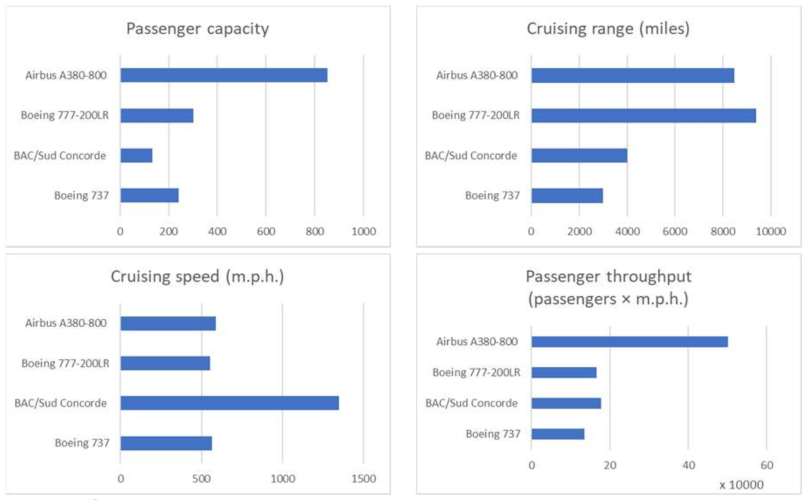
1.6

Performance

38

Defining Performance

- Which airplane has the best performance?



Chapter 1 — Computer Abstractions and Technology — 39

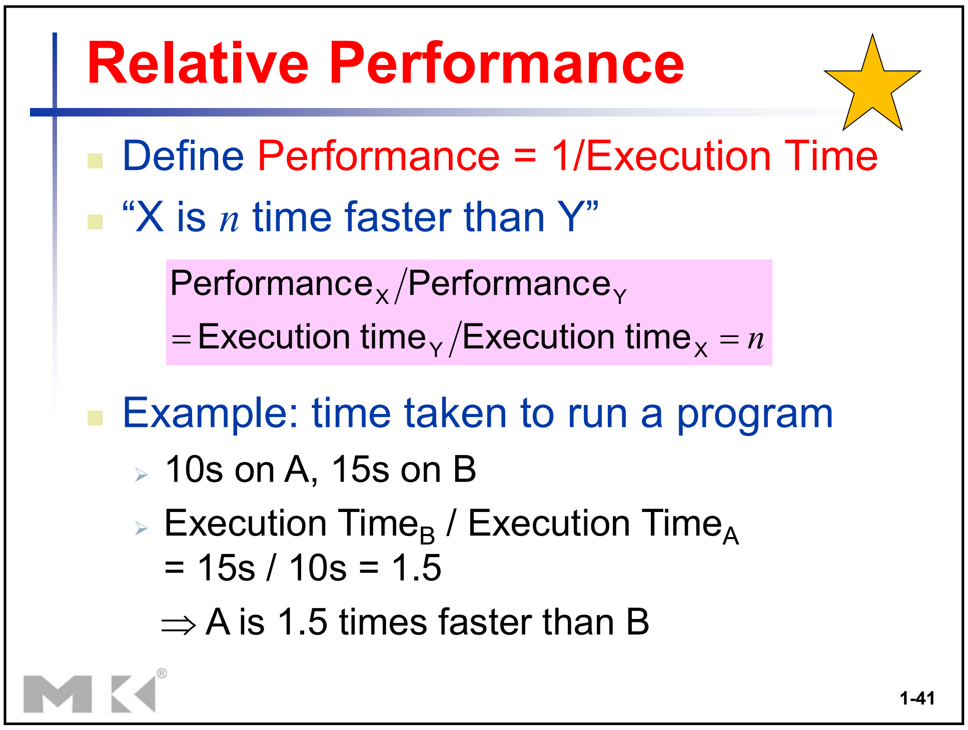
39

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…

1-40

40



Relative Performance

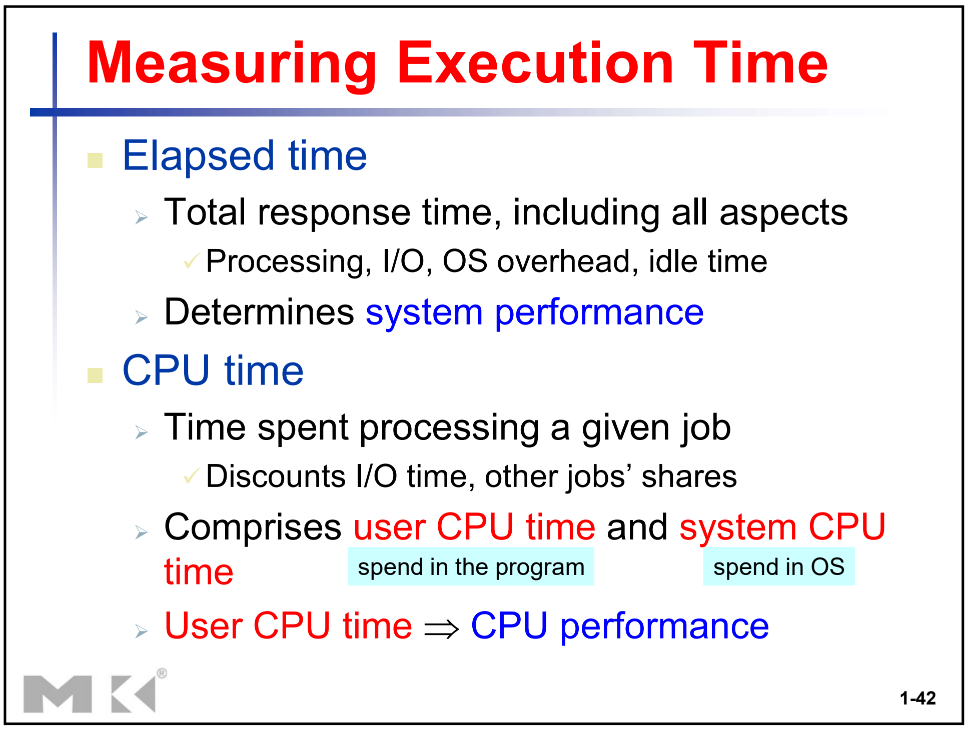
- Define Performance = 1/Execution Time
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$$

- Example: time taken to run a program
  - 10s on A, 15s on B
  - Execution Time_B / Execution Time_A = 15s / 10s = 1.5
  - $\Rightarrow$ A is 1.5 times faster than B

1-41

41

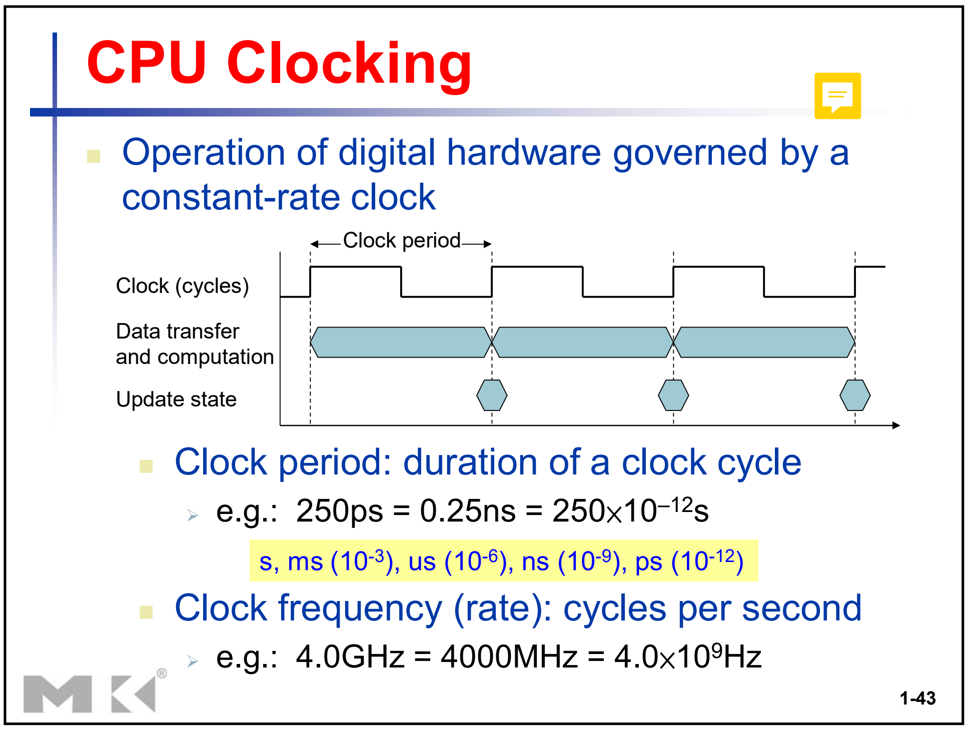


Measuring Execution Time

- 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
    spend in the program spend in OS
  - User CPU time $\Rightarrow$ CPU performance

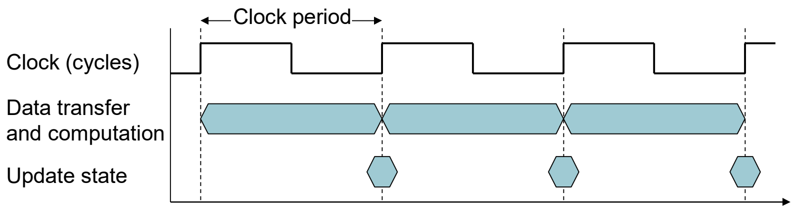
1-42

42



CPU Clocking

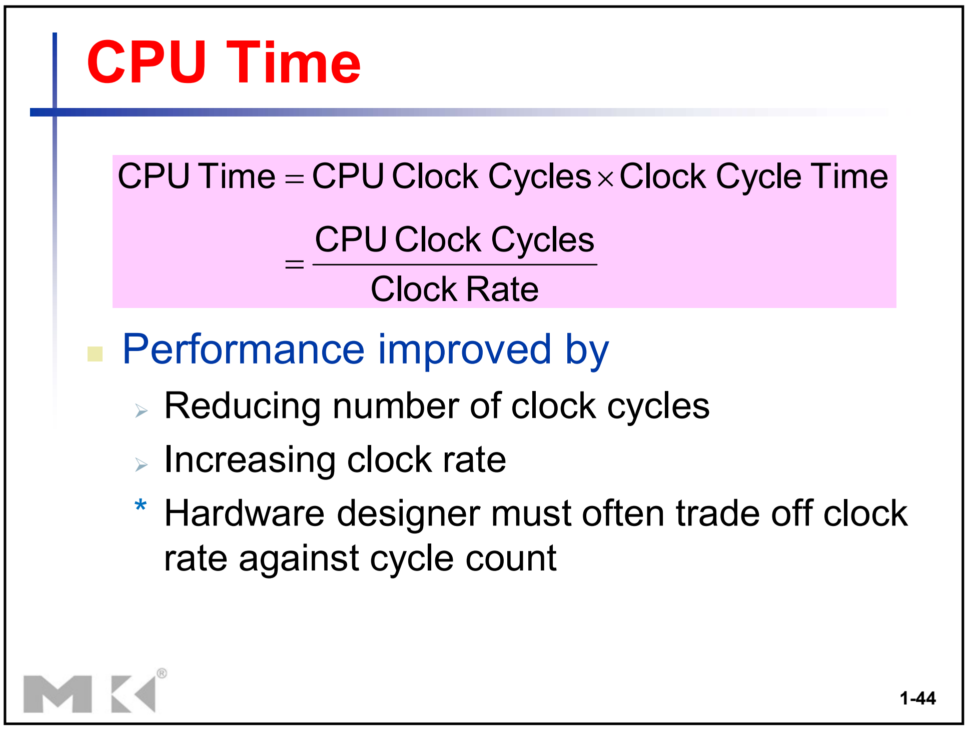
- Operation of digital hardware governed by a constant-rate clock



- Clock period: duration of a clock cycle
  - e.g.: 250ps = 0.25ns = 250×10^{-12}s
  - s, ms (10^{-3}), us (10^{-6}), ns (10^{-9}), ps (10^{-12})
- Clock frequency (rate): cycles per second
  - e.g.: 4.0GHz = 4000MHz = 4.0×10^{9}Hz

1-43

43



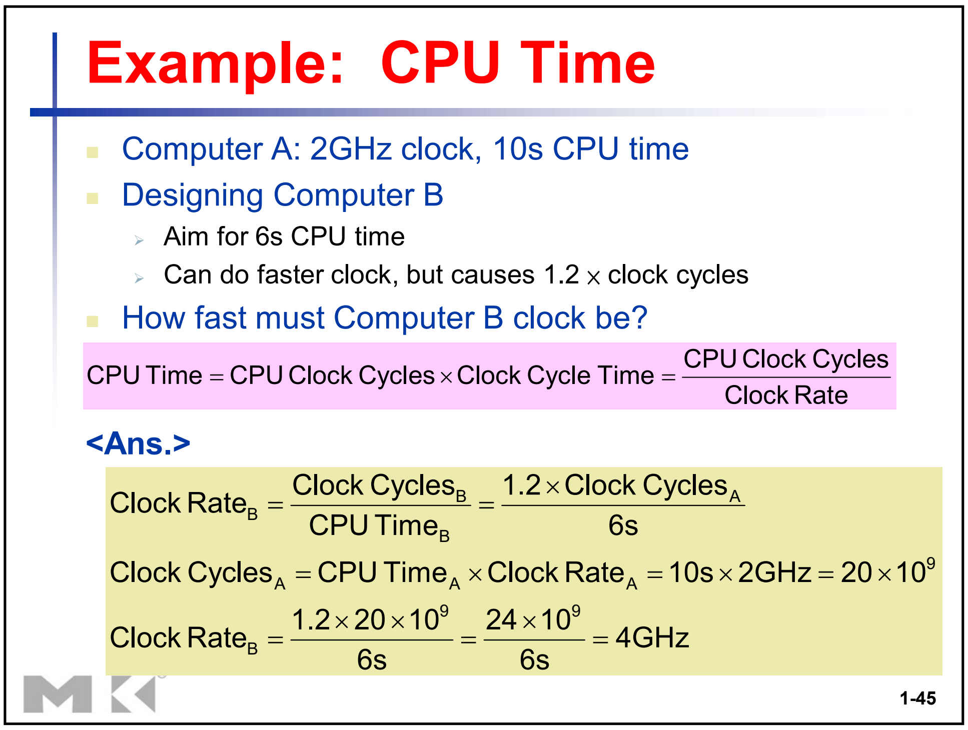
CPU Time

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

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

1-44

44



Example: CPU Time

- 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?

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

<Ans.>

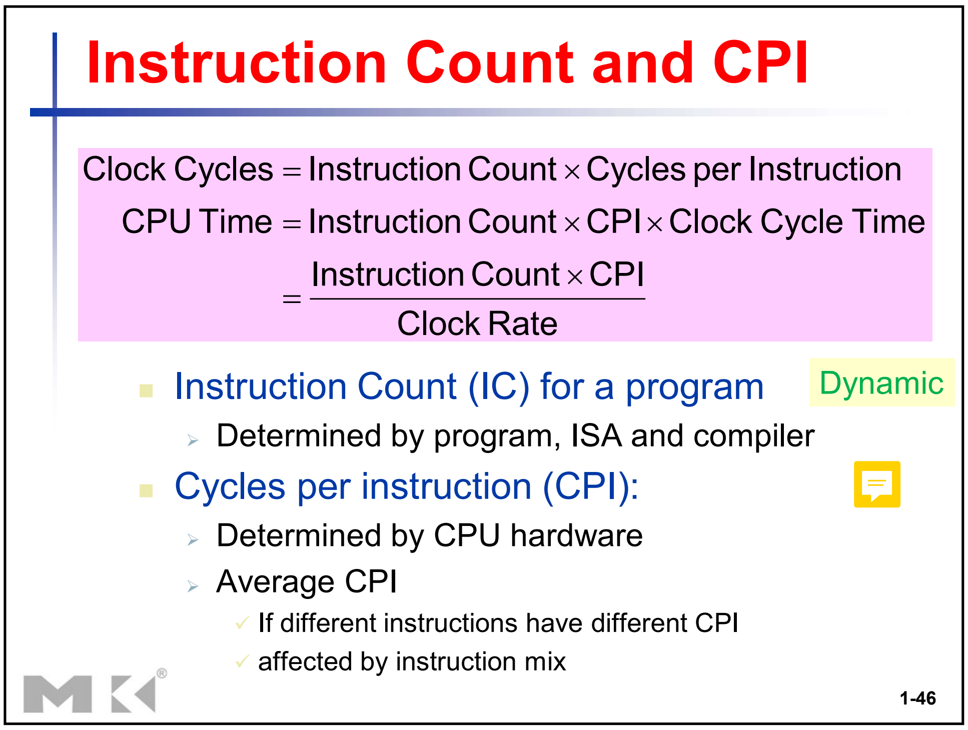
$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

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

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

1-45

45



Instruction Count and CPI

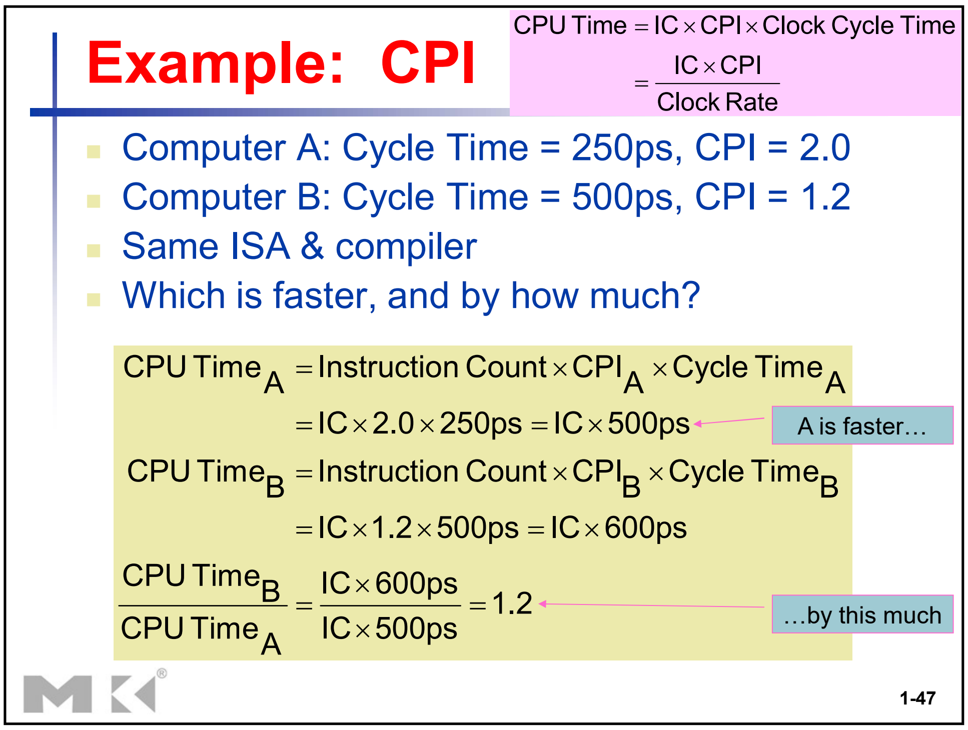
$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

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

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

1-46

46



Example: CPI

$$\text{CPU Time} = \text{IC} \times \text{CPI} \times \text{Clock Cycle Time} = \frac{\text{IC} \times \text{CPI}}{\text{Clock Rate}}$$

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

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= \text{IC} \times 2.0 \times 250\text{ps} = \text{IC} \times 500\text{ps}$$

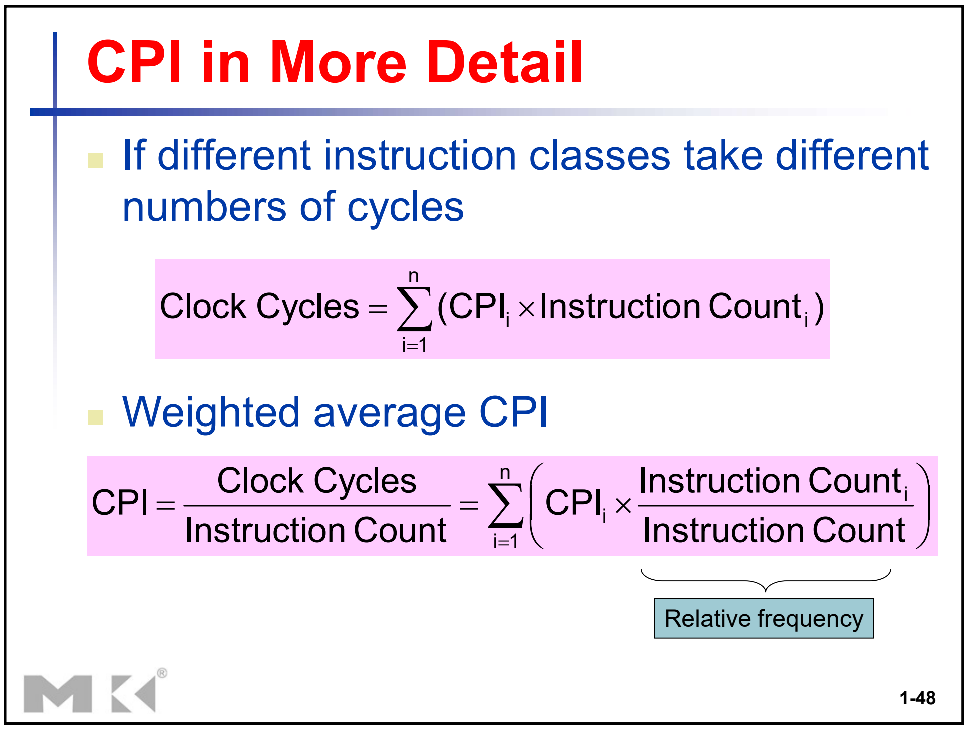
A is faster…

$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= \text{IC} \times 1.2 \times 500\text{ps} = \text{IC} \times 600\text{ps}$$
$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{\text{IC} \times 600\text{ps}}{\text{IC} \times 500\text{ps}} = 1.2$$

…by this much

1-47

47



CPI in More Detail

- If different instruction classes take different numbers of cycles

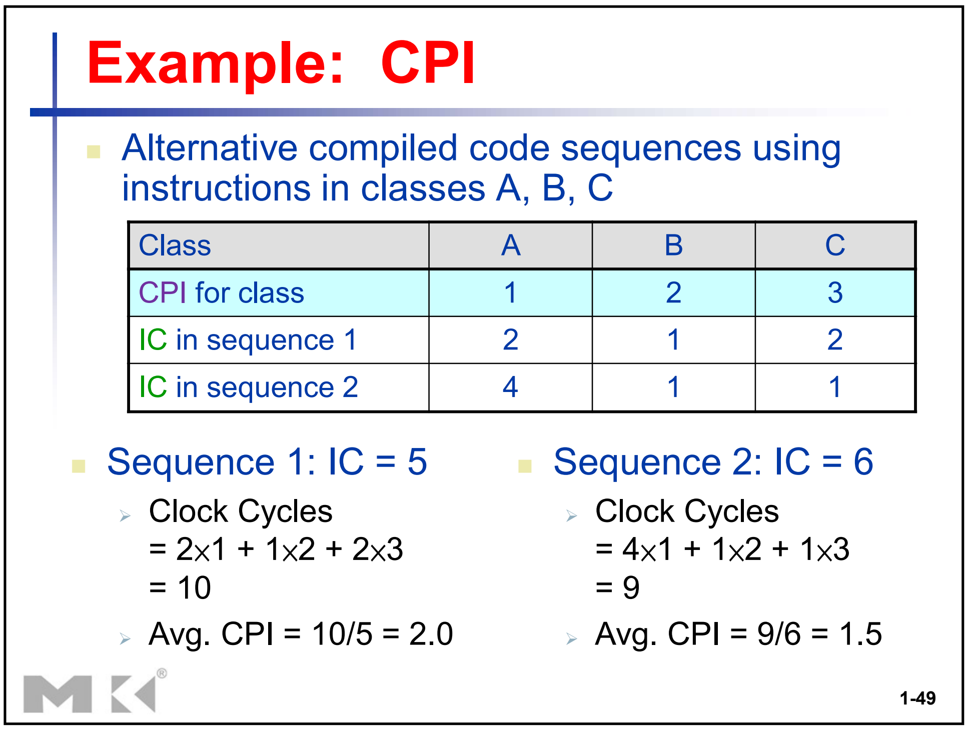
$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Weighted average CPI

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

1-48

48



Example: CPI

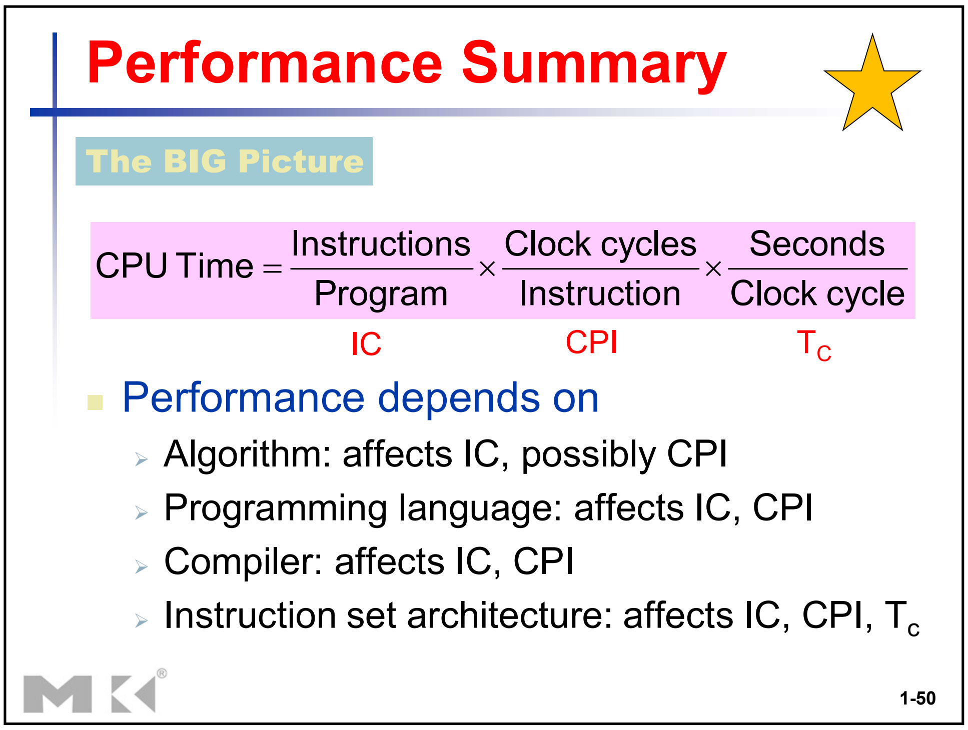
- Alternative compiled code sequences using instructions in classes A, B, C

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 $= 2\times1 + 1\times2 + 2\times3 = 10$
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles $= 4\times1 + 1\times2 + 1\times3 = 9$
  - Avg. CPI = 9/6 = 1.5

1-49

49



Performance Summary

The BIG Picture

$$\text{CPU Time} = \underbrace{\frac{\text{Instructions}}{\text{Program}}}_{\text{IC}} \times \underbrace{\frac{\text{Clock cycles}}{\text{Instruction}}}_{\text{CPI}} \times \underbrace{\frac{\text{Seconds}}{\text{Clock cycle}}}_{T_C}$$

- 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

1-50

50

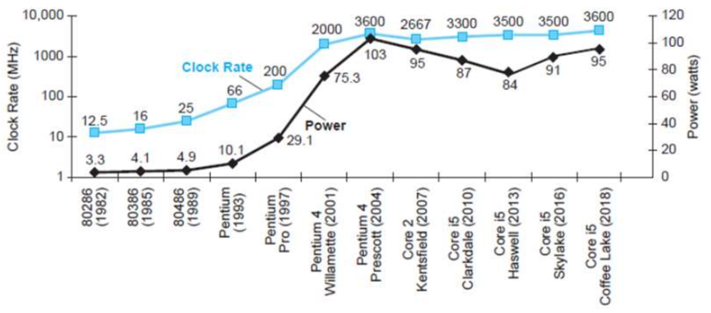
COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface

1.7

The Power Wall

51

Power Trends



over 8 generations and 25 years.

- In CMOS IC technology

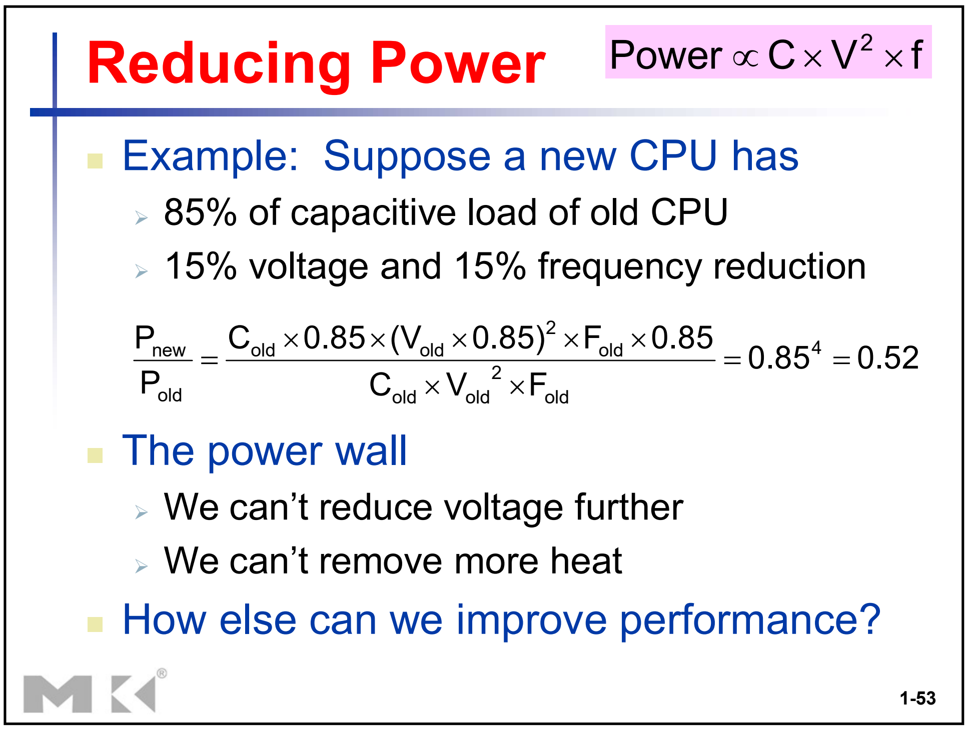
$$\text{Energy} \propto \text{Capacitive load} \times \text{Voltage}^2$$

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

×30 5V → 1V ×1000

1-52

52



Reducing Power

$\text{Power} \propto C \times V^2 \times f$

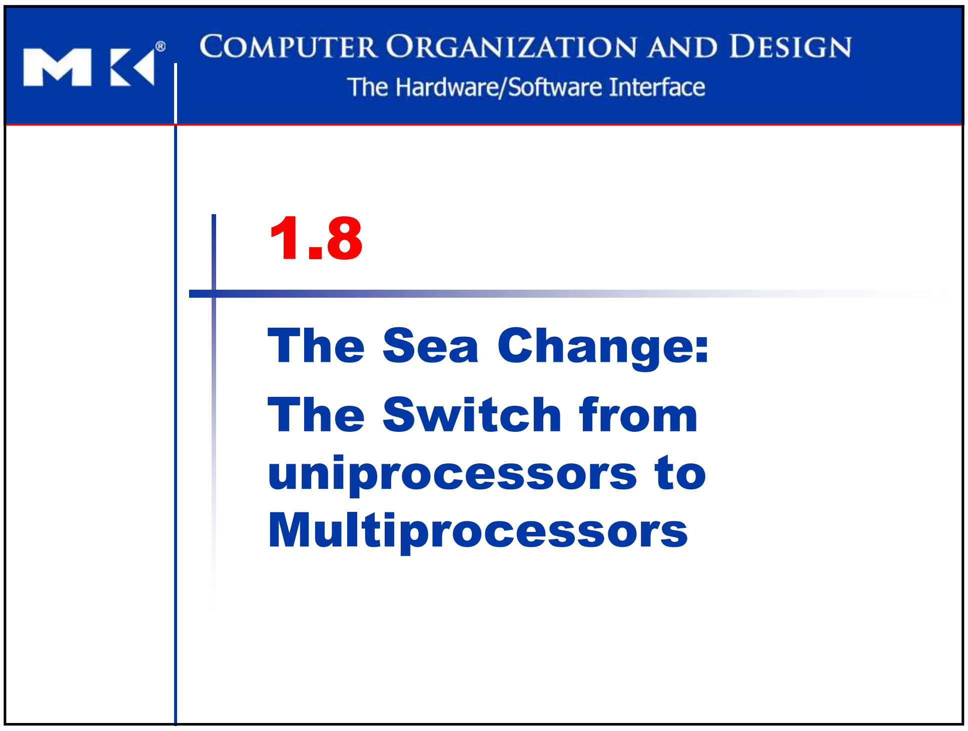
- Example: Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

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

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

1-53

53



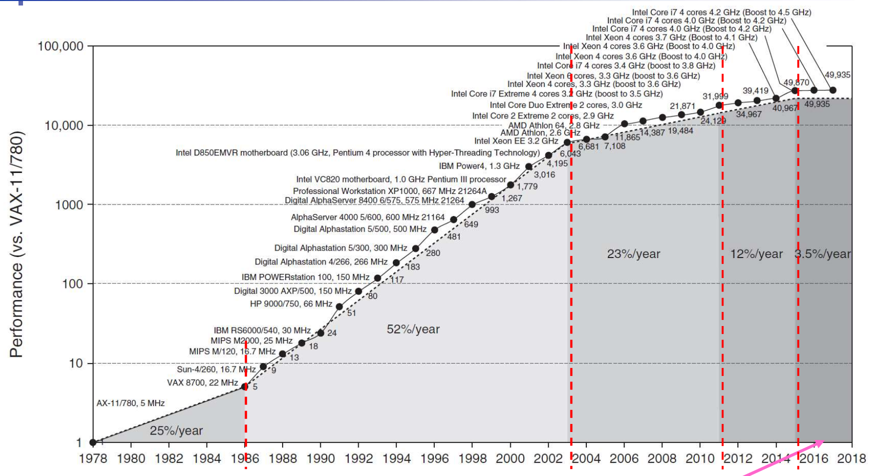
COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface

1.8

The Sea Change: The Switch from uniprocessors to Multiprocessors

54

Uniprocessor Performance



Constrained by power, instruction-level parallelism (ILP), memory latency

1-55

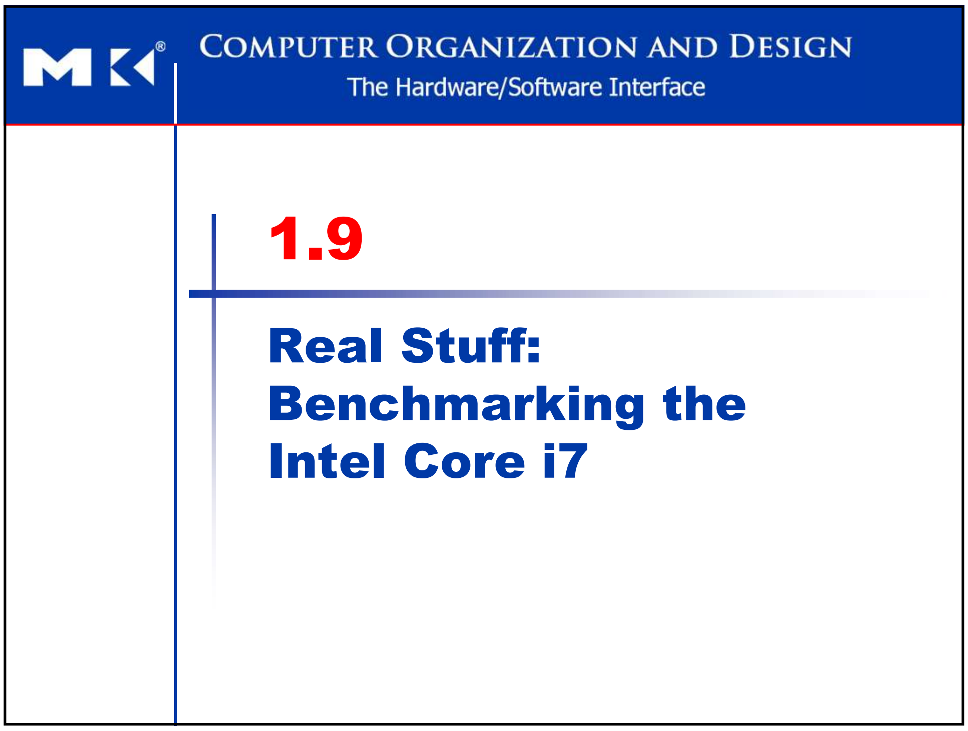
55

Multiprocessors

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

1-56

56



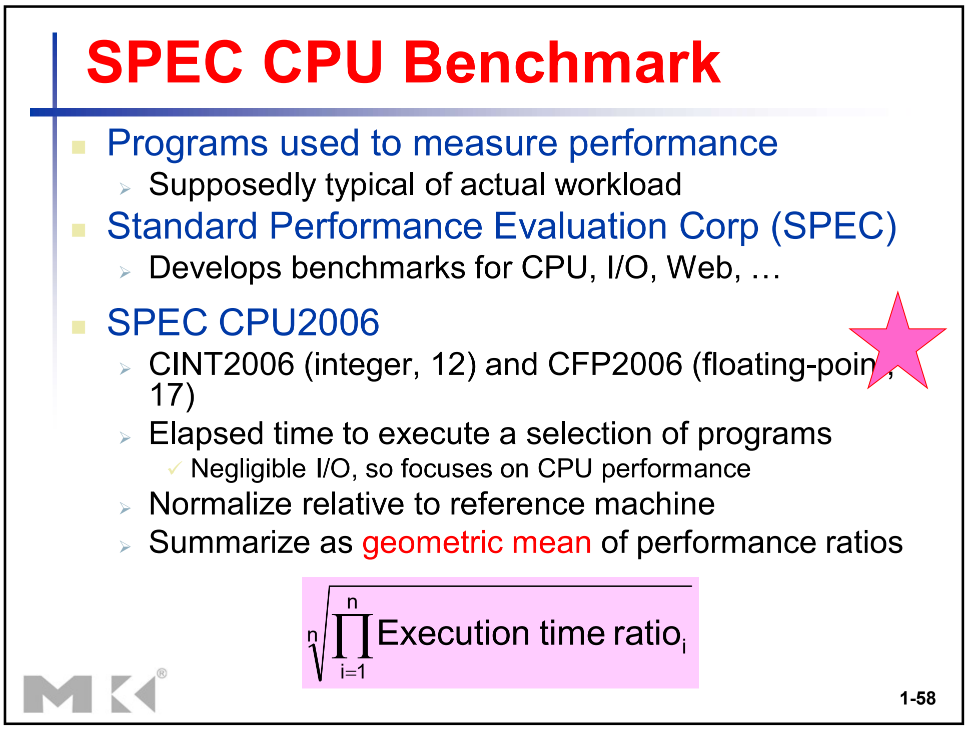
COMPUTER ORGANIZATION AND DESIGN

The Hardware/Software Interface

1.9

Real Stuff: Benchmarking the Intel Core i7

57



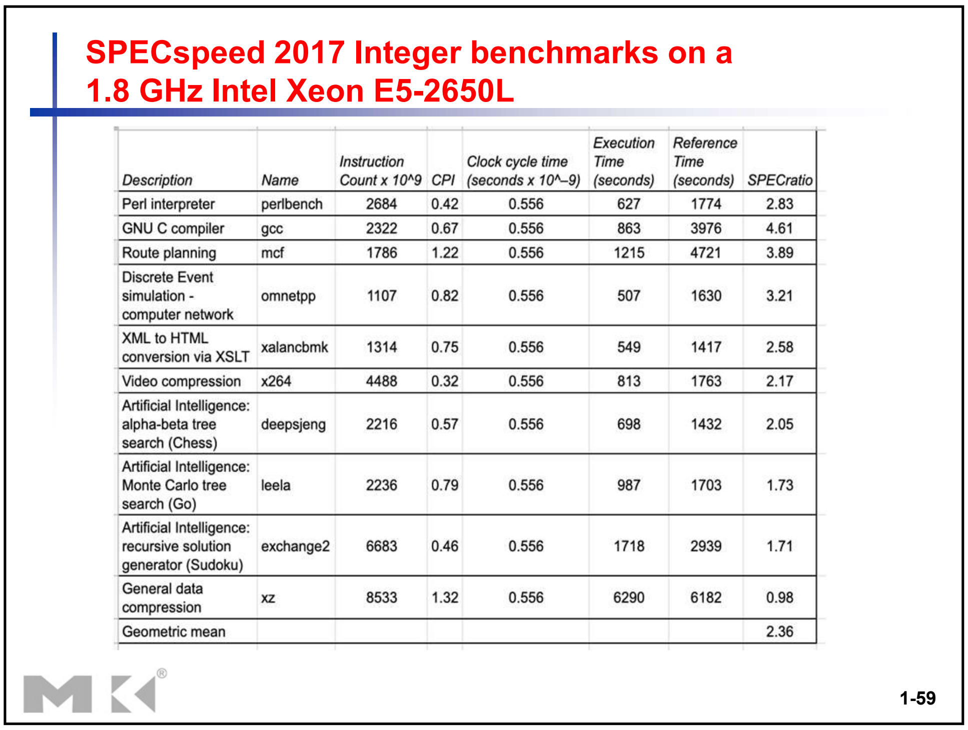
SPEC CPU Benchmark

- Programs used to measure performance
  - Supposedly typical of actual workload
- Standard Performance Evaluation Corp (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2006
  - CINT2006 (integer, 12) and CFP2006 (floating-poin, 17)
  - 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

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

1-58

58

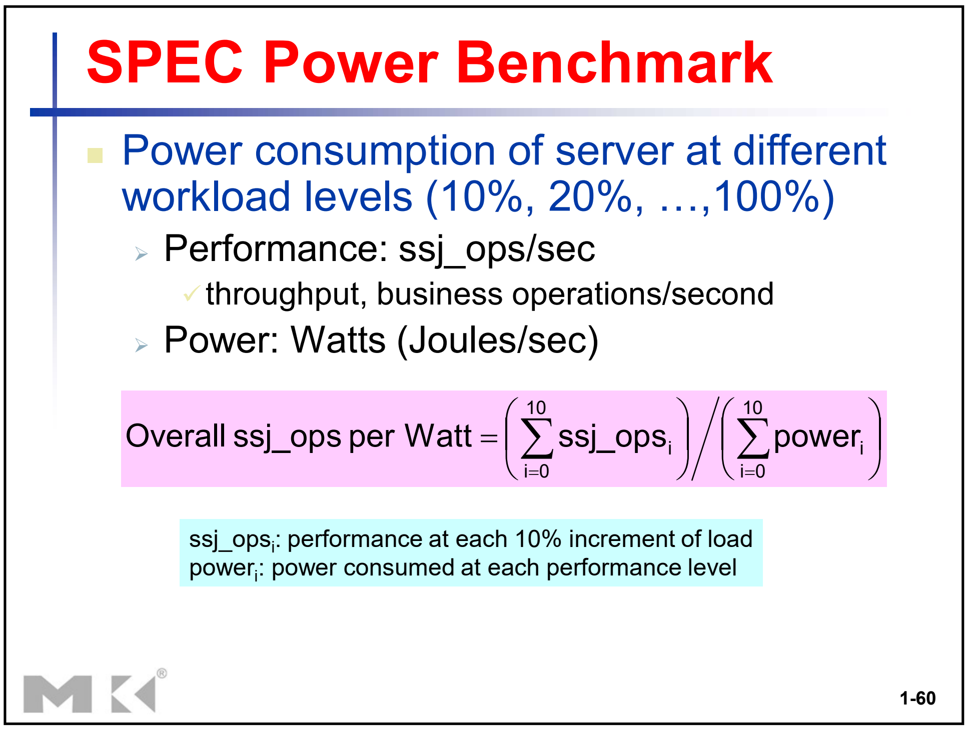


SPECspeed 2017 Integer benchmarks on a 1.8 GHz Intel Xeon E5-2650L

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^–9)	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Perl interpreter	perlbench	2684	0.42	0.556	627	1774	2.83
GNU C compiler	gcc	2322	0.67	0.556	863	3976	4.61
Route planning	mcf	1786	1.22	0.556	1215	4721	3.89
Discrete Event simulation - computer network	omnetpp	1107	0.82	0.556	507	1630	3.21
XML to HTML conversion via XSLT	xalancbmk	1314	0.75	0.556	549	1417	2.58
Video compression	x264	4488	0.32	0.556	813	1763	2.17
Artificial Intelligence: alpha-beta tree search (Chess)	deepsjeng	2216	0.57	0.556	698	1432	2.05
Artificial Intelligence: Monte Carlo tree search (Go)	leela	2236	0.79	0.556	987	1703	1.73
Artificial Intelligence: recursive solution generator (Sudoku)	exchange2	6683	0.46	0.556	1718	2939	1.71
General data compression	xz	8533	1.32	0.556	6290	6182	0.98
Geometric mean							2.36

1-59

59



SPEC Power Benchmark

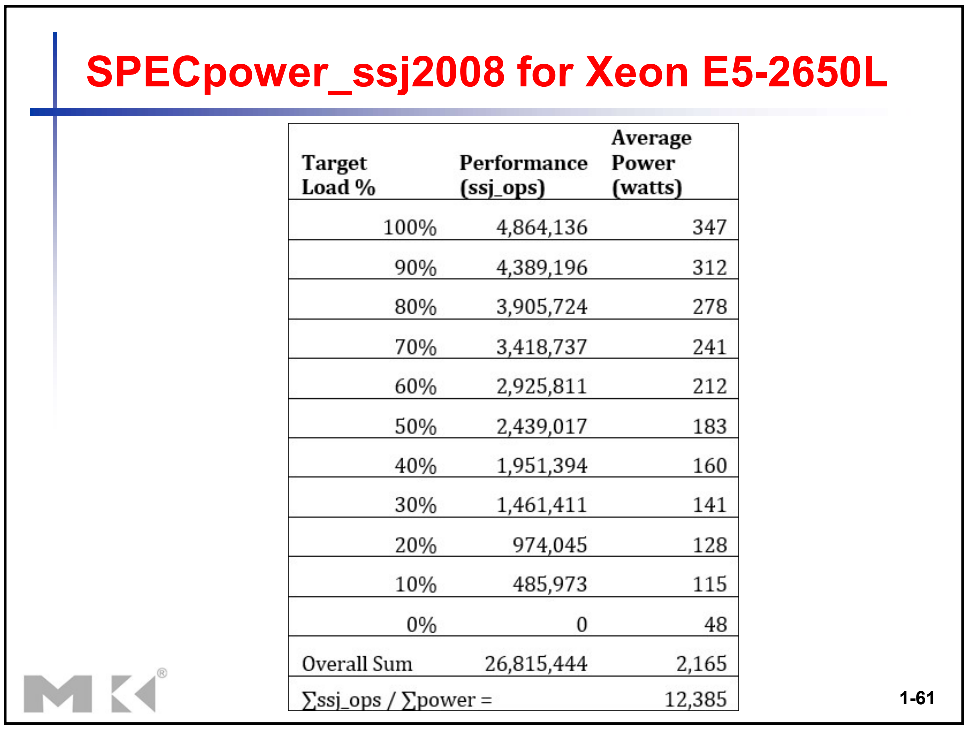
- Power consumption of server at different workload levels (10%, 20%, …,100%)
  - Performance: ssj_ops/sec
    - throughput, business operations/second
  - Power: Watts (Joules/sec)

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

ssj_ops_i: performance at each 10% increment of load
power_i: power consumed at each performance level

1-60

60

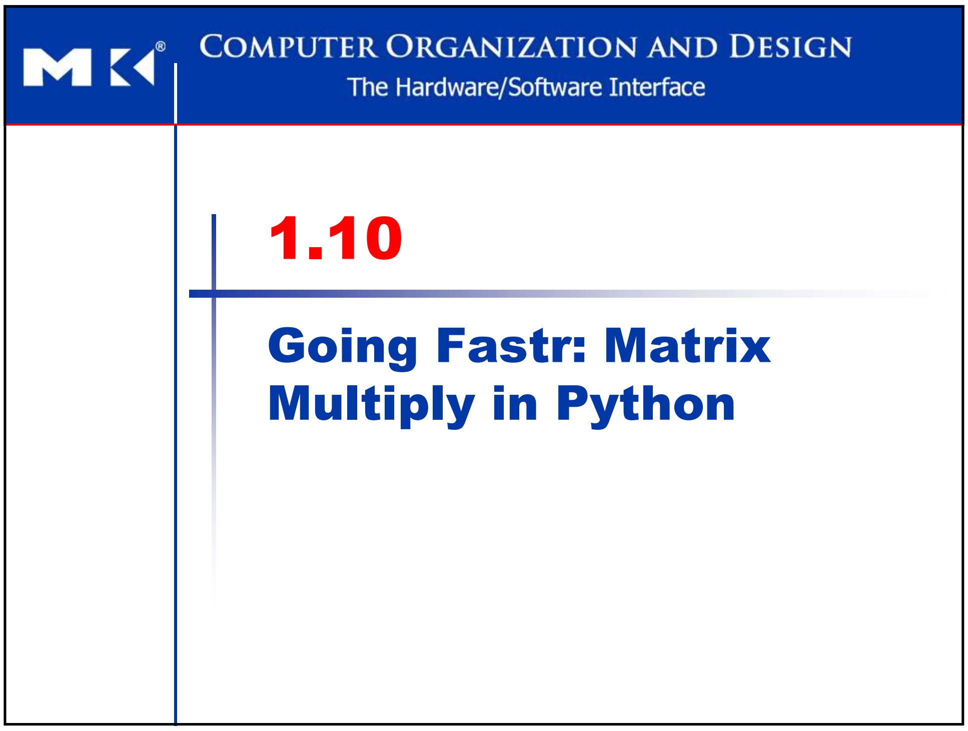


SPECpower_ssj2008 for Xeon E5-2650L

Target Load %	Performance (ssj_ops)	Average Power (watts)
100%	4,864,136	347
90%	4,389,196	312
80%	3,905,724	278
70%	3,418,737	241
60%	2,925,811	212
50%	2,439,017	183
40%	1,951,394	160
30%	1,461,411	141
20%	974,045	128
10%	485,973	115
0%	0	48
Overall Sum	26,815,444	2,165
∑ssj_ops / ∑power =		12,385

1-61

61

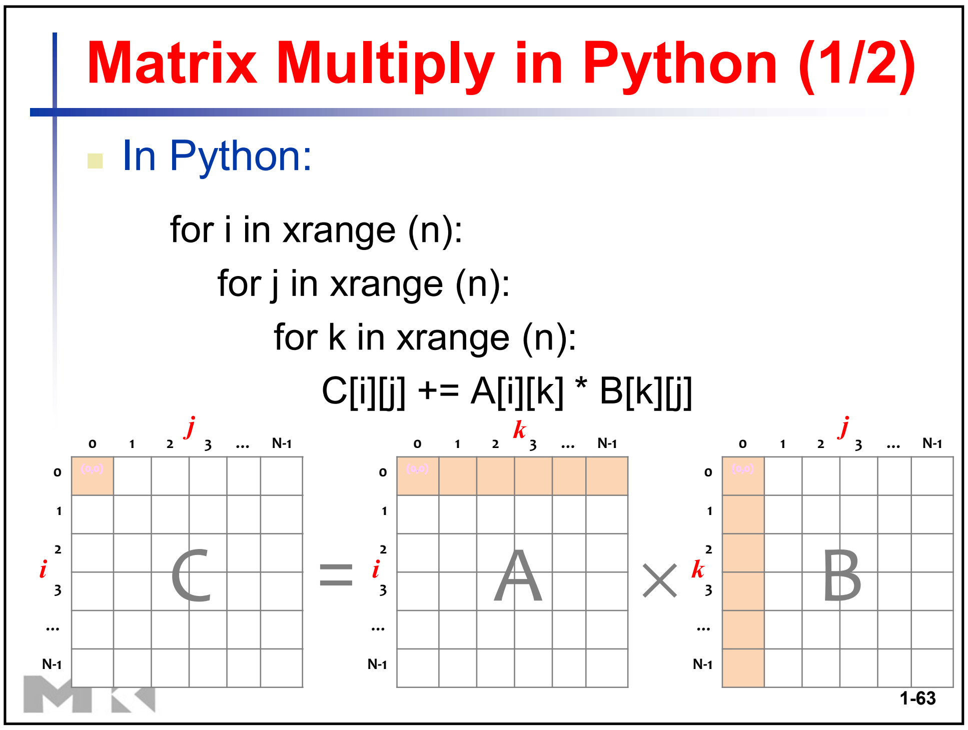


COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface

1.10

Going Fastr: Matrix Multiply in Python

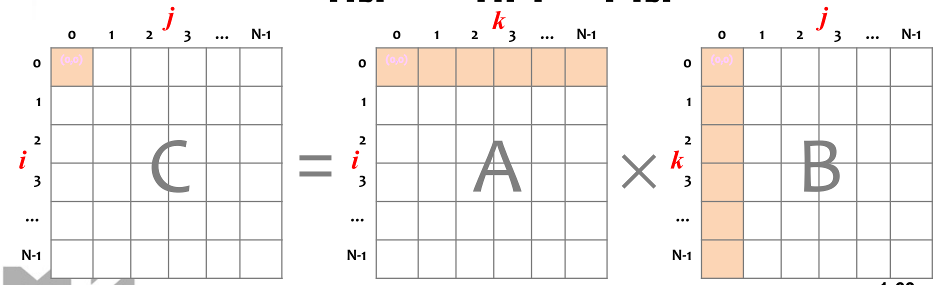
62



Matrix Multiply in Python (1/2)

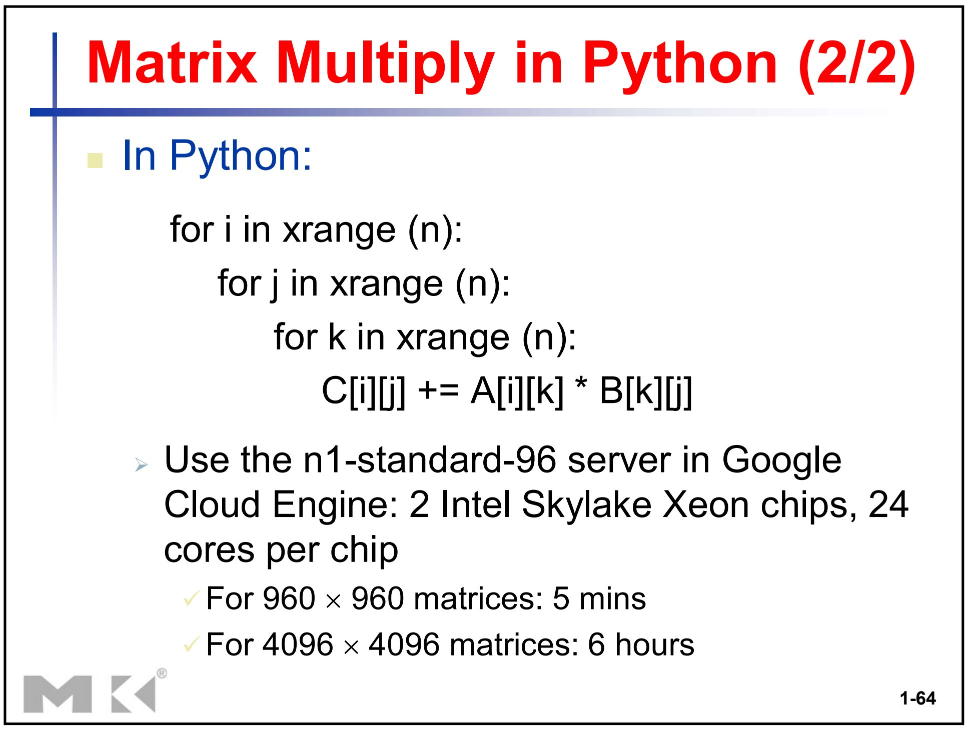
- In Python:

```
for i in xrange (n):
    for j in xrange (n):
        for k in xrange (n):
            C[i][j] += A[i][k] * B[k][j]
```



1-63

63



Matrix Multiply in Python (2/2)

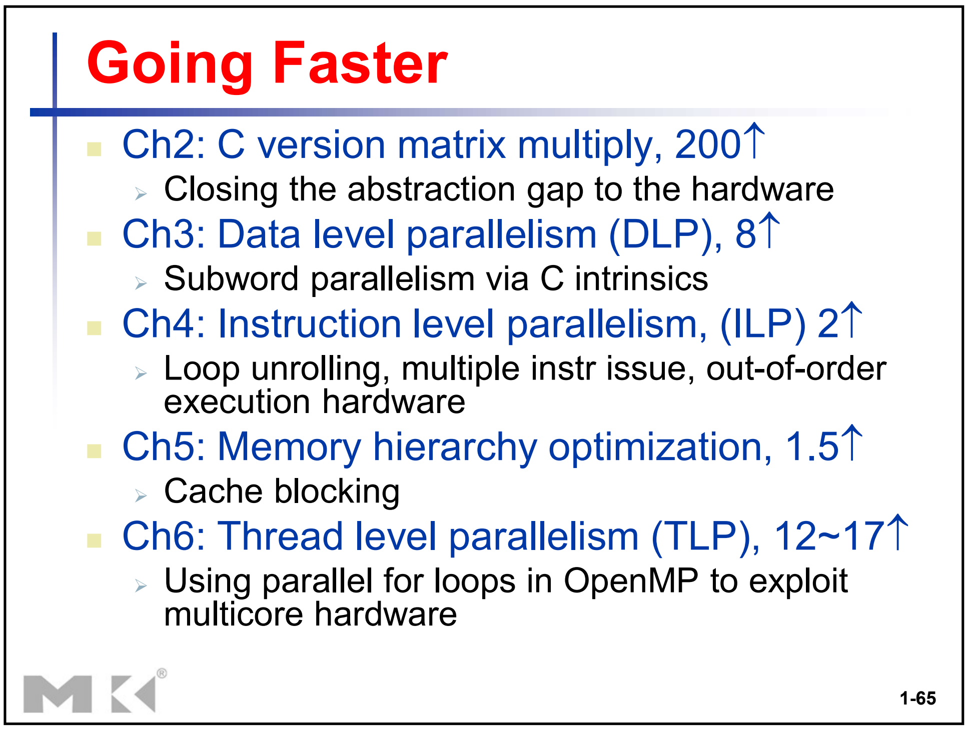
- In Python:

```
for i in xrange (n):
    for j in xrange (n):
        for k in xrange (n):
            C[i][j] += A[i][k] * B[k][j]
```

  - Use the n1-standard-96 server in Google Cloud Engine: 2 Intel Skylake Xeon chips, 24 cores per chip
    - ✓ For 960 × 960 matrices: 5 mins
    - ✓ For 4096 × 4096 matrices: 6 hours

1-64

64

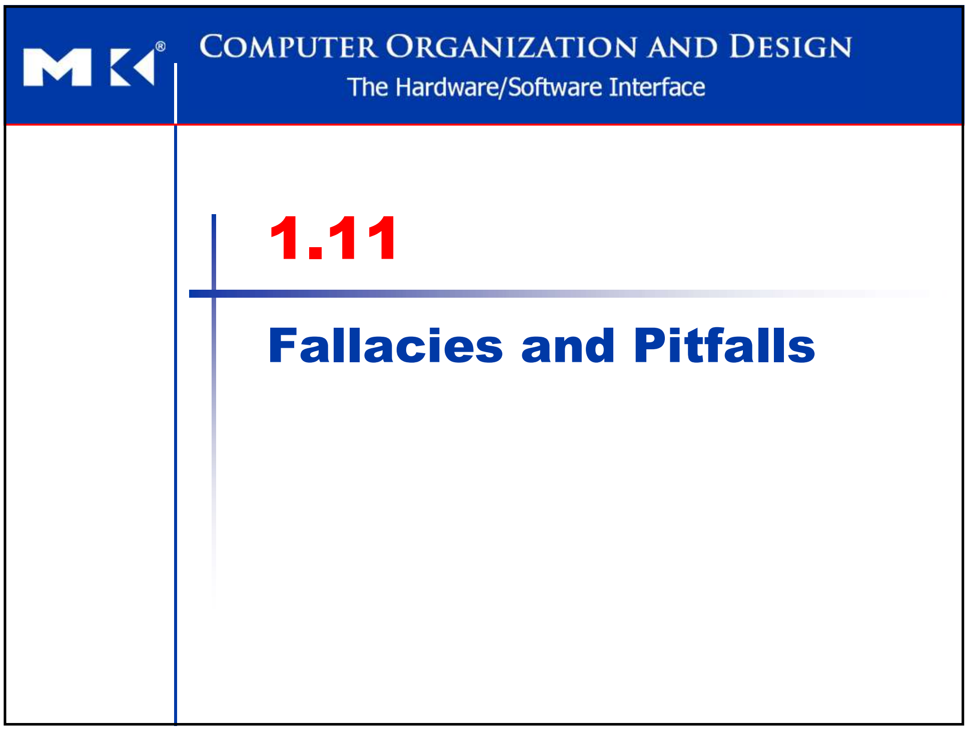


Going Faster

- Ch2: C version matrix multiply, 200↑
  - Closing the abstraction gap to the hardware
- Ch3: Data level parallelism (DLP), 8↑
  - Subword parallelism via C intrinsics
- Ch4: Instruction level parallelism, (ILP) 2↑
  - Loop unrolling, multiple instr issue, out-of-order execution hardware
- Ch5: Memory hierarchy optimization, 1.5↑
  - Cache blocking
- Ch6: Thread level parallelism (TLP), 12~17↑
  - Using parallel for loops in OpenMP to exploit multicore hardware

1-65

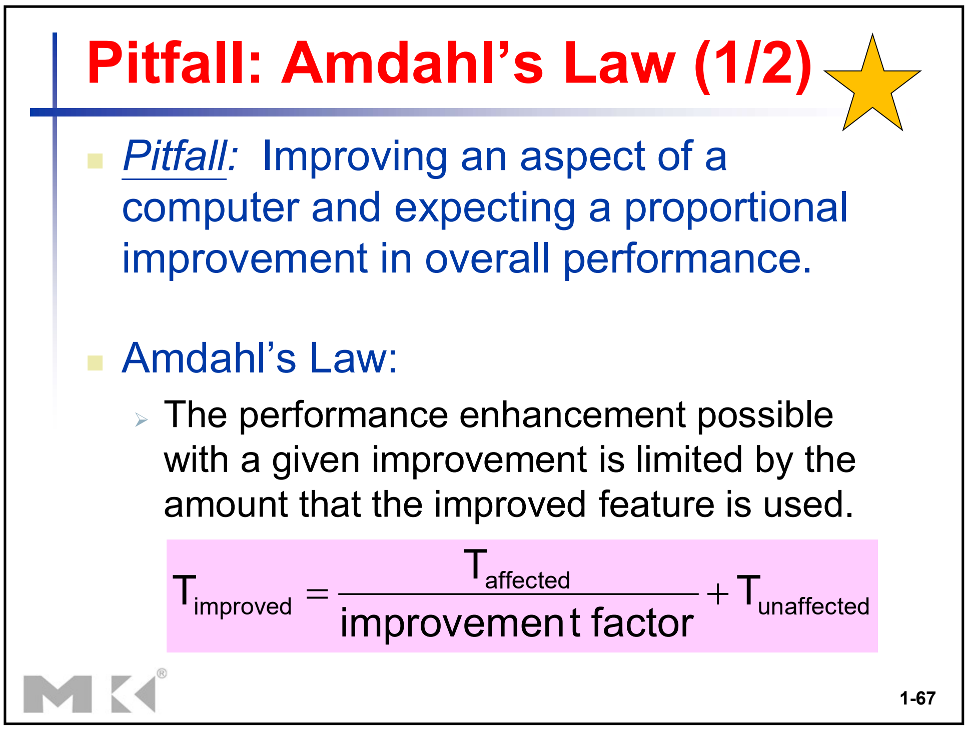
65



COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface

1.11 Fallacies and Pitfalls

66



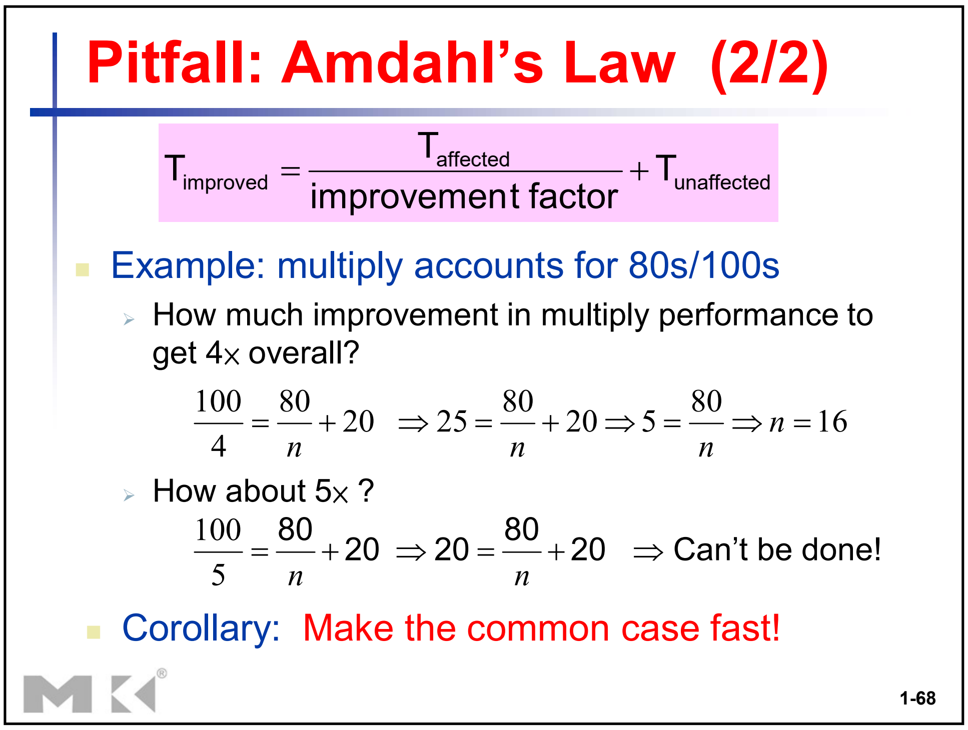
Pitfall: Amdahl's Law (1/2)

- *Pitfall:* Improving an aspect of a computer and expecting a proportional improvement in overall performance.
- Amdahl's Law:
  - The performance enhancement possible with a given improvement is limited by the amount that the improved feature is used.

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

1-67

67



Pitfall: Amdahl's Law (2/2)

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

- Example: multiply accounts for 80s/100s
  - How much improvement in multiply performance to get 4× overall?

$$\frac{100}{4} = \frac{80}{n} + 20 \Rightarrow 25 = \frac{80}{n} + 20 \Rightarrow 5 = \frac{80}{n} \Rightarrow n = 16$$

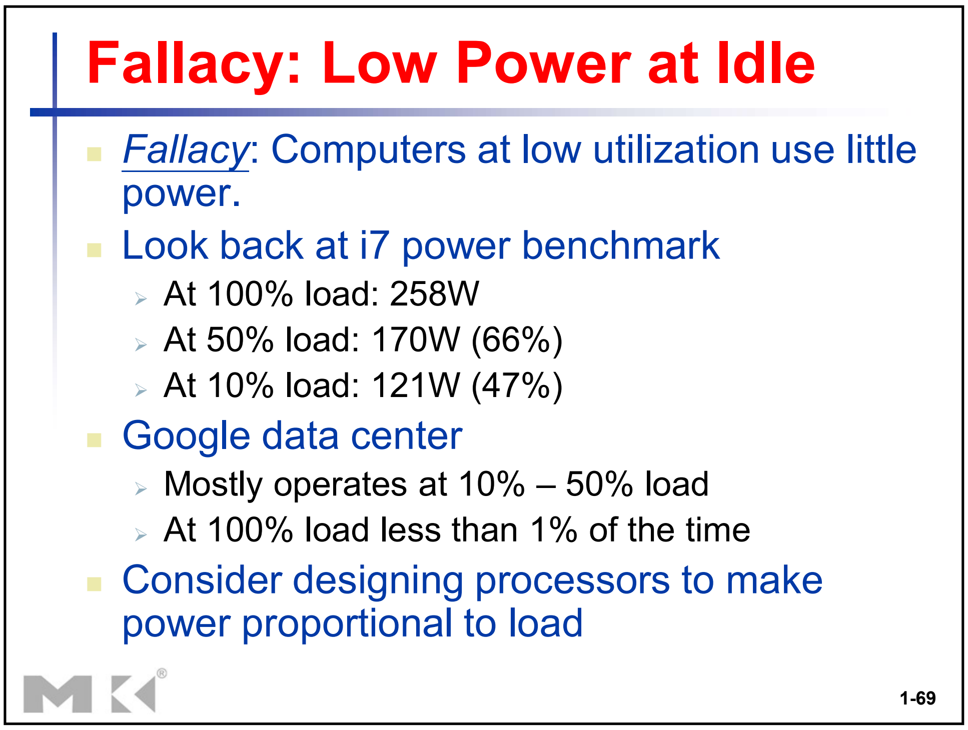
  - How about 5× ?

$$\frac{100}{5} = \frac{80}{n} + 20 \Rightarrow 20 = \frac{80}{n} + 20 \quad \Rightarrow \text{Can't be done!}$$

- Corollary: Make the common case fast!

1-68

68

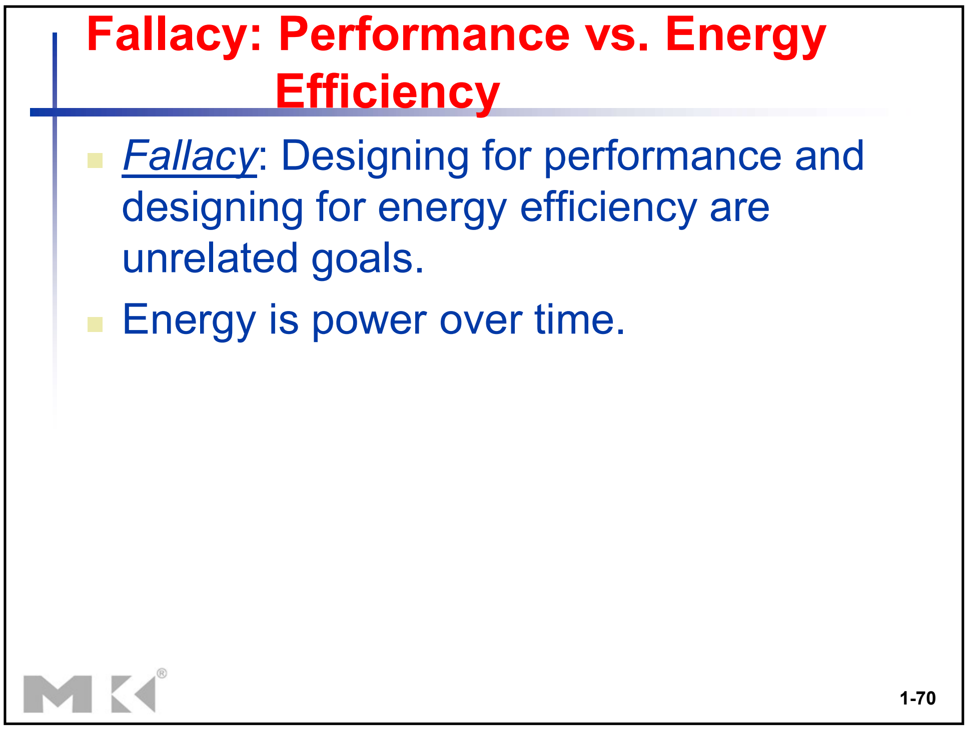


Fallacy: Low Power at Idle

- *Fallacy*: Computers at low utilization use little power.
- 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

1-69

69

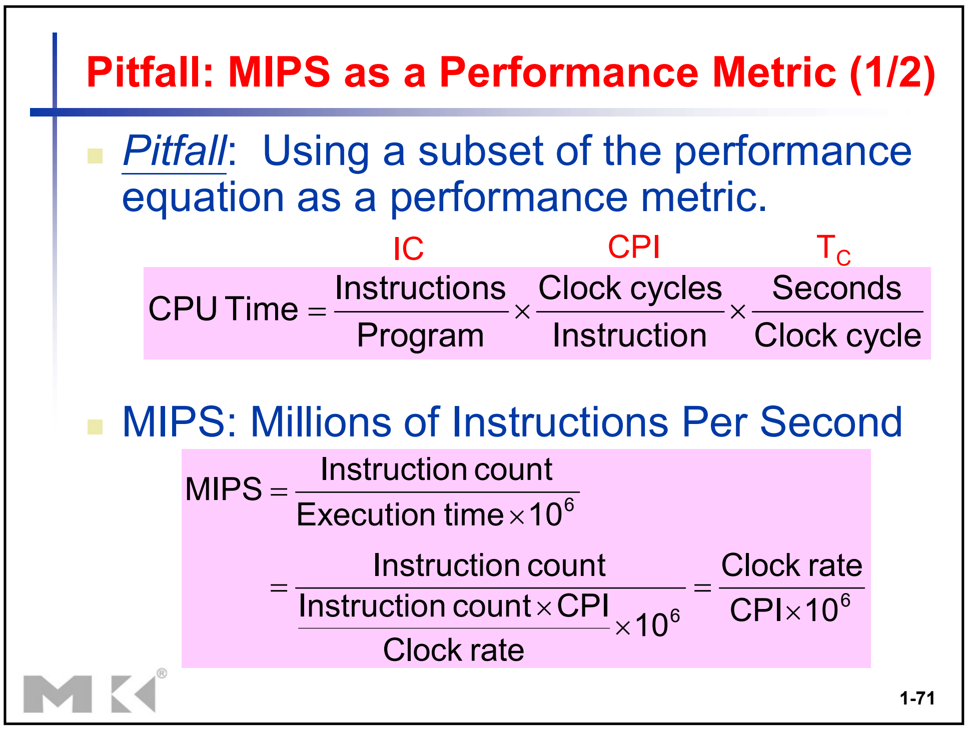


Fallacy: Performance vs. Energy Efficiency

- *Fallacy*: Designing for performance and designing for energy efficiency are unrelated goals.
- Energy is power over time.

1-70

70



Pitfall: MIPS as a Performance Metric (1/2)

- *Pitfall*: Using a subset of the performance equation as a performance metric.

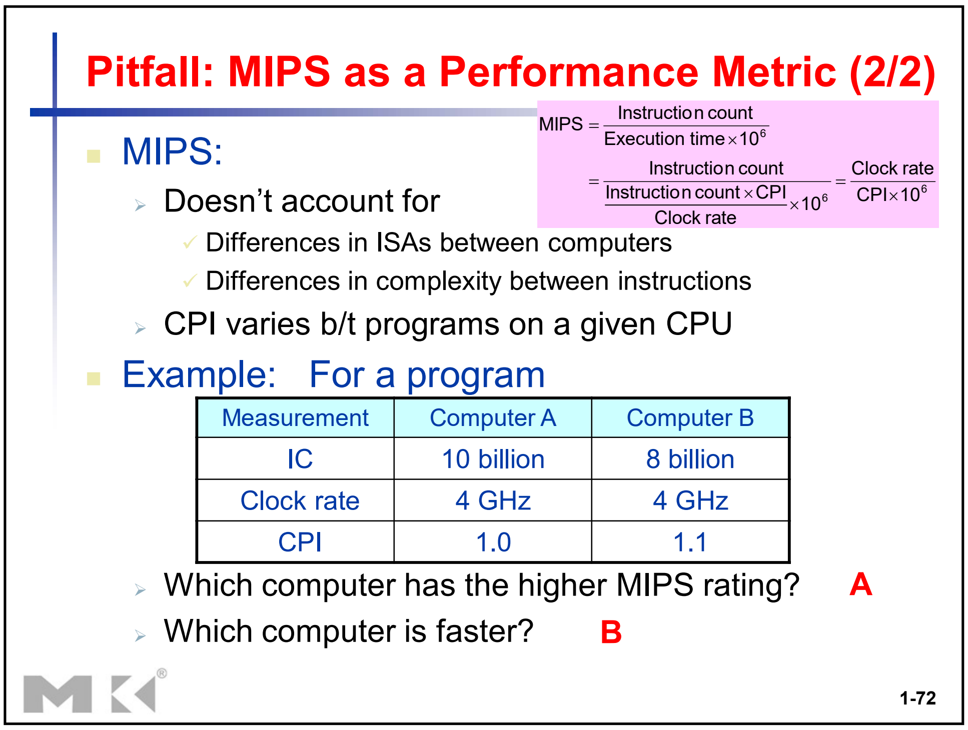
$$\text{CPU Time} = \overset{\text{IC}}{\frac{\text{Instructions}}{\text{Program}}} \times \overset{\text{CPI}}{\frac{\text{Clock cycles}}{\text{Instruction}}} \times \overset{T_C}{\frac{\text{Seconds}}{\text{Clock cycle}}}$$

- MIPS: Millions of Instructions Per Second

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

1-71

71



Pitfall: MIPS as a Performance Metric (2/2)

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

- MIPS:
  - Doesn't account for
    - Differences in ISAs between computers
    - Differences in complexity between instructions
  - CPI varies b/t programs on a given CPU
- Example: For a program

Measurement	Computer A	Computer B
IC	10 billion	8 billion
Clock rate	4 GHz	4 GHz
CPI	1.0	1.1

  - Which computer has the higher MIPS rating? **A**
  - Which computer is faster? **B**

1-72

72

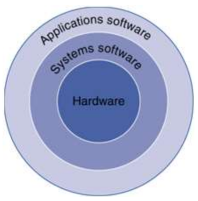
1.12

Concluding Remarks

73

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 (IC × CPI × CT)
- Power is a limiting factor
  - Use parallelism to improve performance



1-74

74