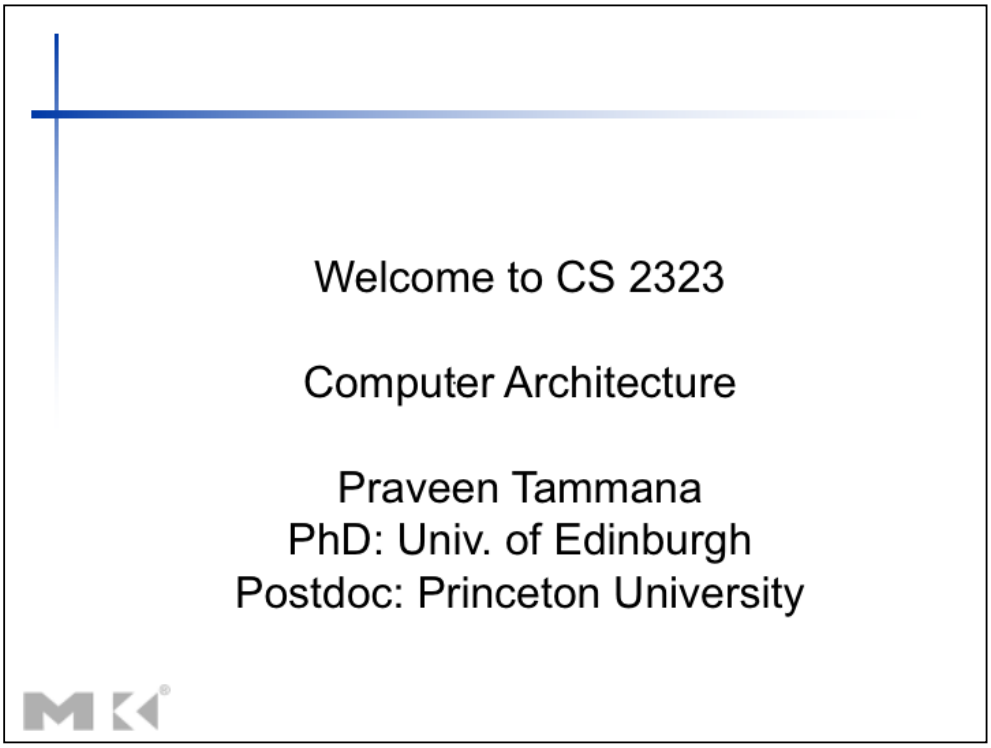
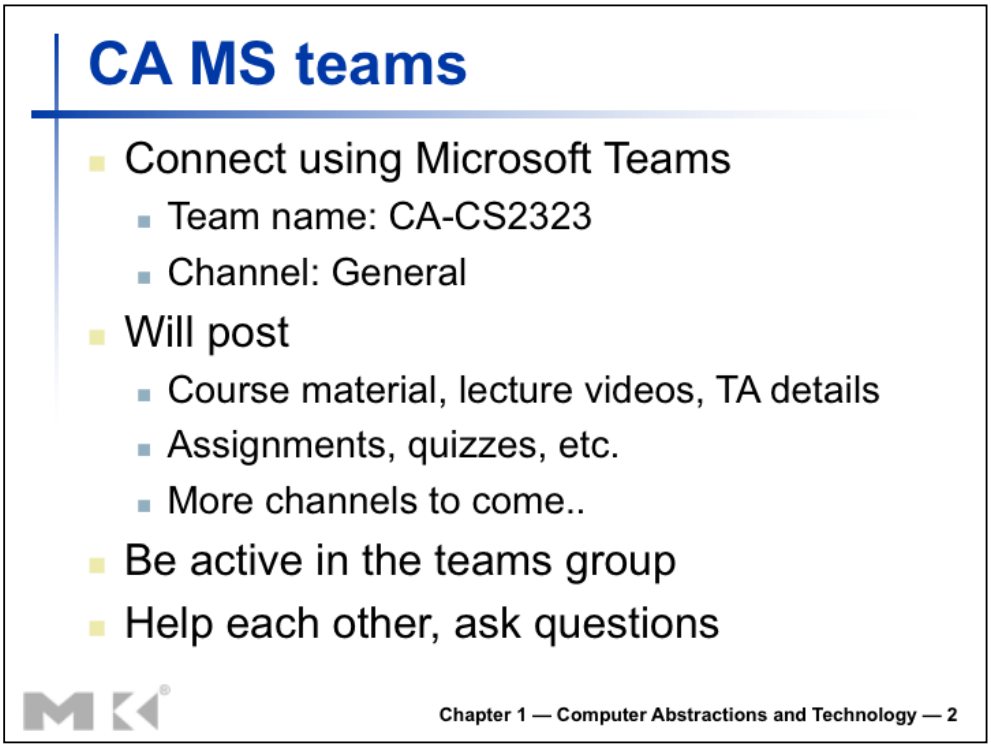

Welcome to CS 2323
Computer Architecture
Praveen Tammana
PhD: Univ. of Edinburgh
Postdoc: Princeton University


CA MS teams
Connect using Microsoft Teams
Team name: CA-CS2323
Channel: General
Will post
Course material, lecture videos, TA details
Assignments, quizzes, etc.
More channels to come..
Be active in the teams group
Help each other, ask questions
Chapter 1 — Computer Abstractions and Technology — 2

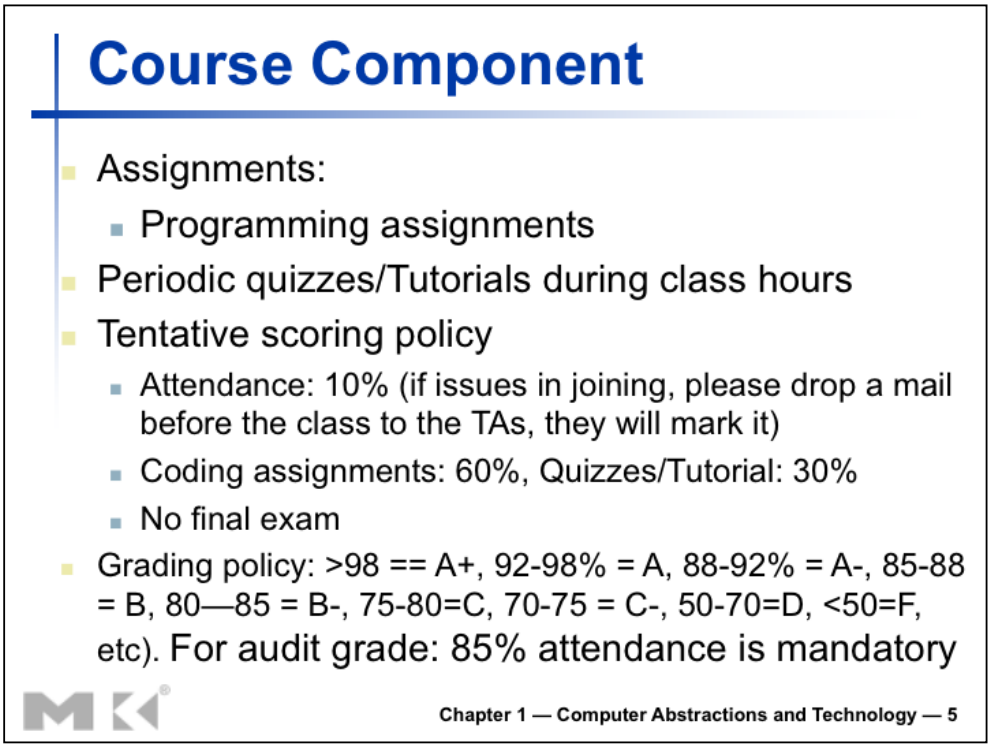
CA initial hiccups

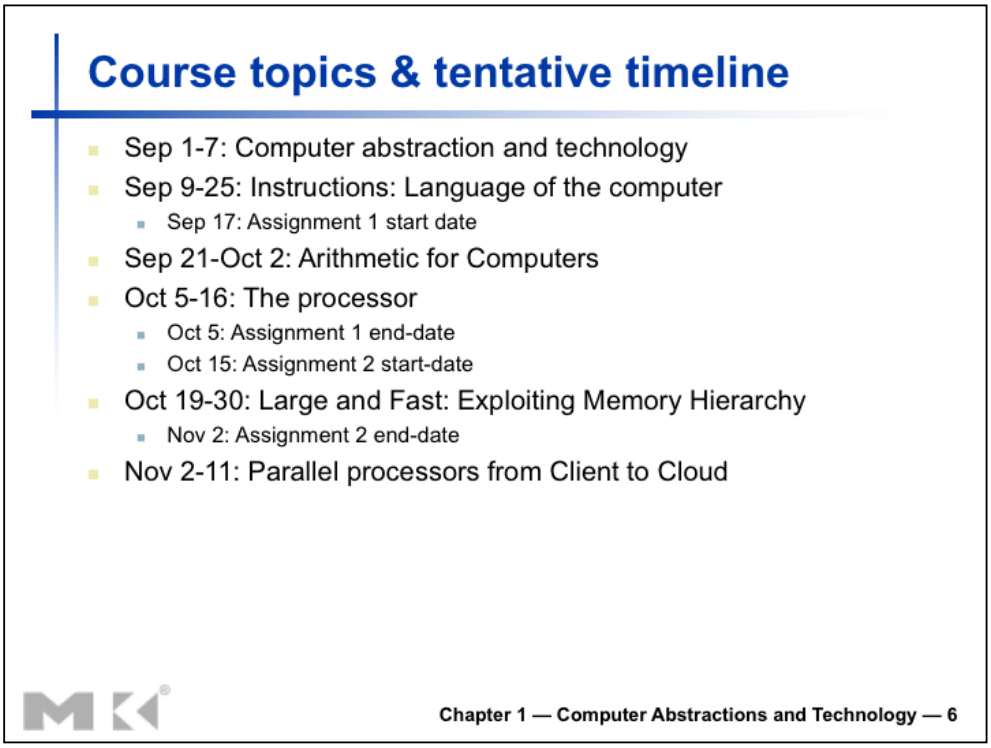
- Overlap with other courses?
  - Please de-register and take the course next year
- Pre-requisites:
  - CS1353: Introduction to Data structure
  - ID1303: Introduction to Programming
  - If not done with both courses, please de-register
- Any trouble accessing teams account?
  - Contact Vijay (vijay.chakry@cse.iith.ac.in)

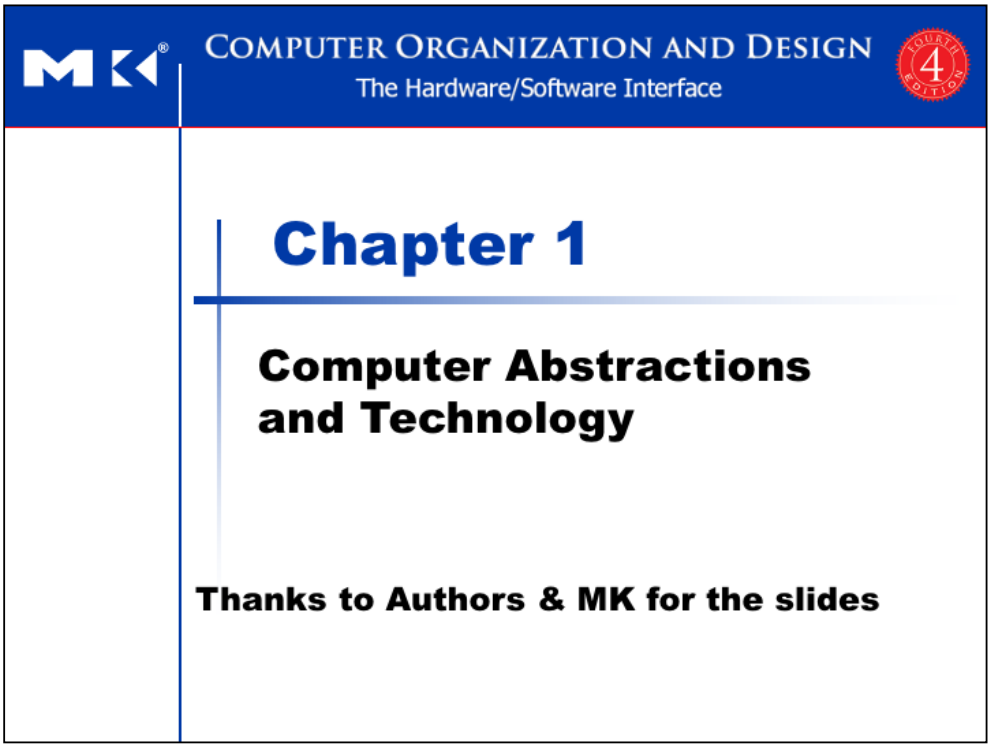
Chapter 1 — Computer Abstractions and Technology — 3

Reference books

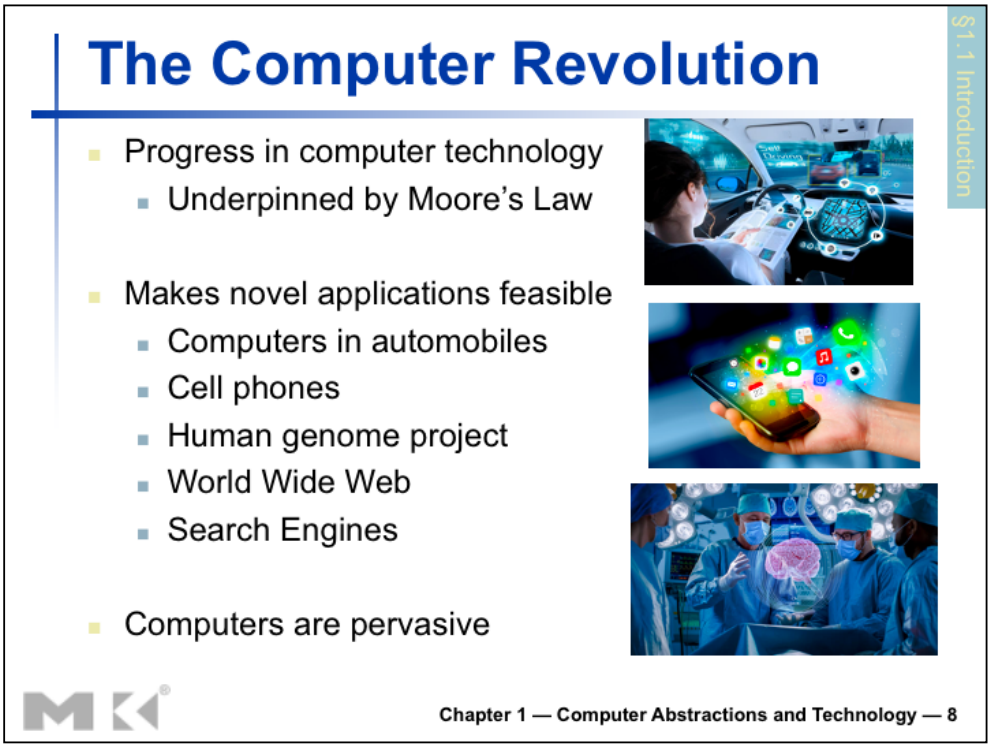
- Mostly: Computer Organization and Design, The hardware/Software Interface 5th Edition (MIPS)
- Authors:
  - David A. Patterson
  - Joh L. Hennessy
- Optional read:
  - Computer Organization and Architecture, by Smriti Ranjan Sarangi


Course Component
Assignments:
Programming assignments
Periodic quizzes/Tutorials during class hours
Tentative scoring policy
Attendance: 10% (if issues in joining, please drop a mail before the class to the TAs, they will mark it)
Coding assignments: 60%, Quizzes/Tutorial: 30%
No final exam
Grading policy: >98 == A+, 92-98% = A, 88-92% = A-, 85-88 = B, 80—85 = B-, 75-80=C, 70-75 = C-, 50-70=D, <50=F, etc). For audit grade: 85% attendance is mandatory
Chapter 1 — Computer Abstractions and Technology — 5


Course topics & tentative timeline
Sep 1-7: Computer abstraction and technology
Sep 9-25: Instructions: Language of the computer
Sep 17: Assignment 1 start date
Sep 21-Oct 2: Arithmetic for Computers
Oct 5-16: The processor
Oct 5: Assignment 1 end-date
Oct 15: Assignment 2 start-date
Oct 19-30: Large and Fast: Exploiting Memory Hierarchy
Nov 2: Assignment 2 end-date
Nov 2-11: Parallel processors from Client to Cloud
Chapter 1 — Computer Abstractions and Technology — 6



- Our economy is becoming highly dependent on online services, computers, cloud, etc.
- In this course, students learn about low-level details of computer that made all application built on top of it possible.

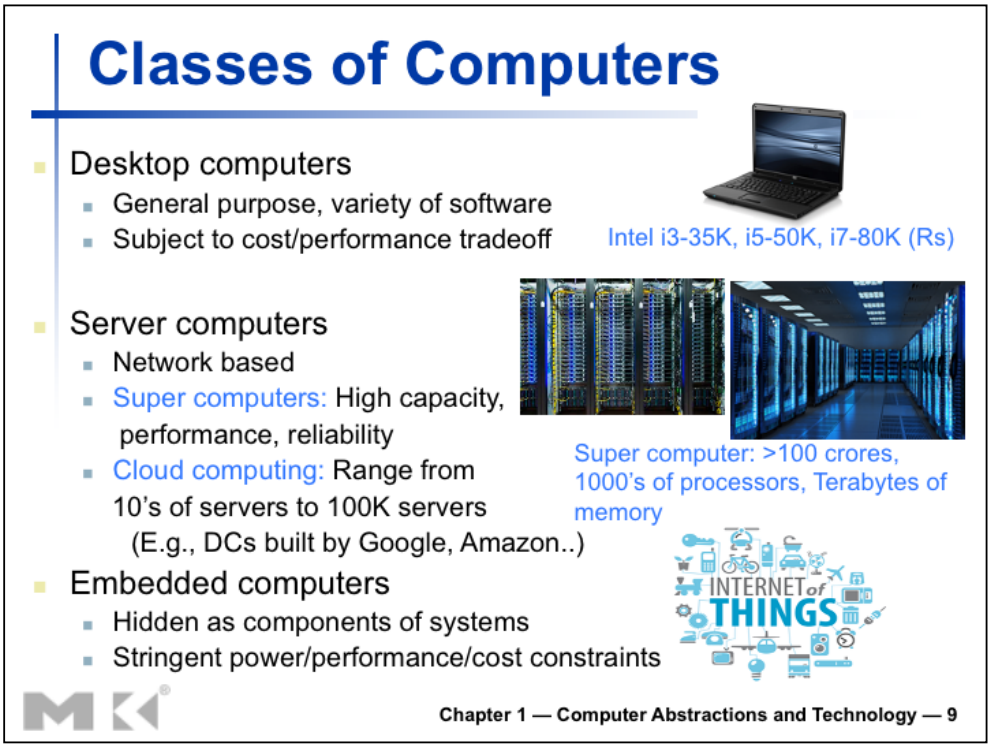


- Moore's Law refers to Moore's perception that the number of transistors on a microchip doubles every two years, though the cost of computers is halved.
- Each time the cost of computing improves by order of 10, the opportunities multiply.

- Computers reduce pollution, improve fuel efficiency, air bag inflation to protect occupants in a crash
- connects ½ of the population on planet from almost every where
- The cost of computer equipment to map and analyze human DNA sequences was hundreds of millions of dollars, now it is significantly reduced and made once science fiction a reality. *DNA sequencing,* determines the order of nucleotides in a DNA strand, allows scientists to read the genetic code so they can study the normal versions of genes. After they know the order of nucleotides in normal and disease versions of a gene, they can identify which changes in the gene cause the disease.
- Web has replaced libraries and news papers
- Finding relevant information became increasingly important. It is a hardship to go without them

- Clearly, all this technology is possible because of computers the we rely on every aspect of our society.

- In this course you will learn the fundamentals of hardware/software of a computer that enable killer applications like glasses with augment reality, cashless society, and cars that can drive themselves is possible.



Although a common set of hardware technologies is used in computers ranging from smart home appliances to cell phones to the largest supercomputers, these different applications have different design requirements and employ the core hardware technologies in different ways.

Desktop computers

- emphasize delivery of good performance to single users at low cost
- usually execute third-party software.
- Intel I3-7 cost in Rs.

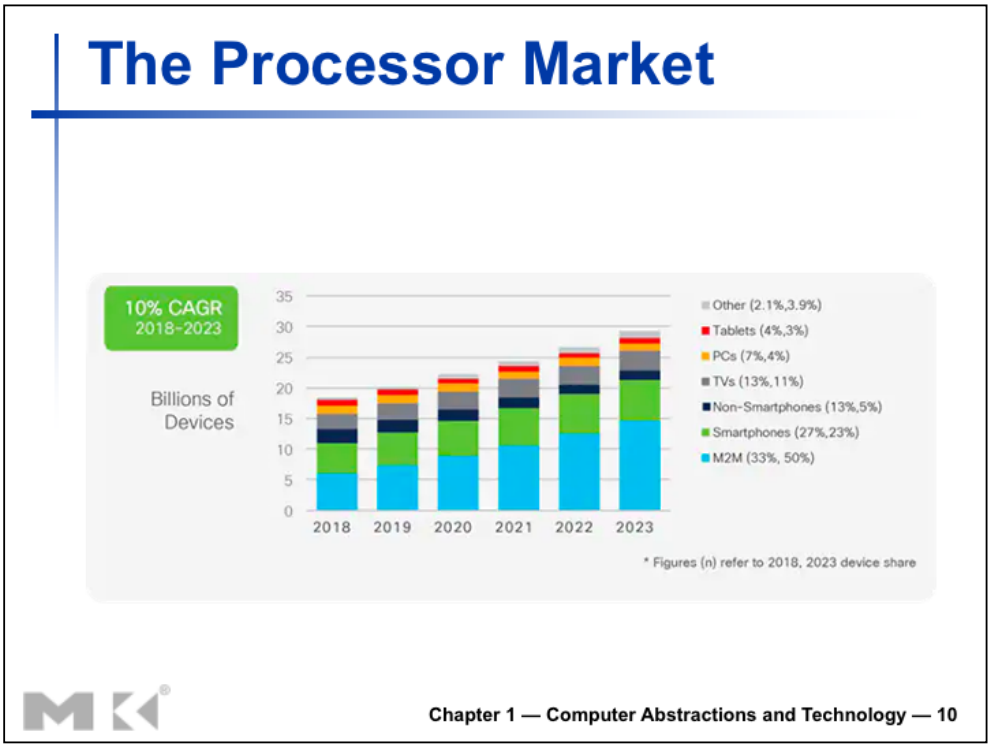
Servers are the modern form of what were once mainframes, minicomputers, and supercomputers, and are usually accessed only via a network.

- Servers are oriented to carrying large workloads - a single complex applications—usually a scientific or engineering application—or handling many small jobs, such as would occur in building a large Web server.
- In general, servers also place a greater emphasis on dependability, since a crash is usually more costly than it would be on a single-user desktop computer.
- Datacenter (DC) is a farm of servers that could be as large as 300 times cricket field.

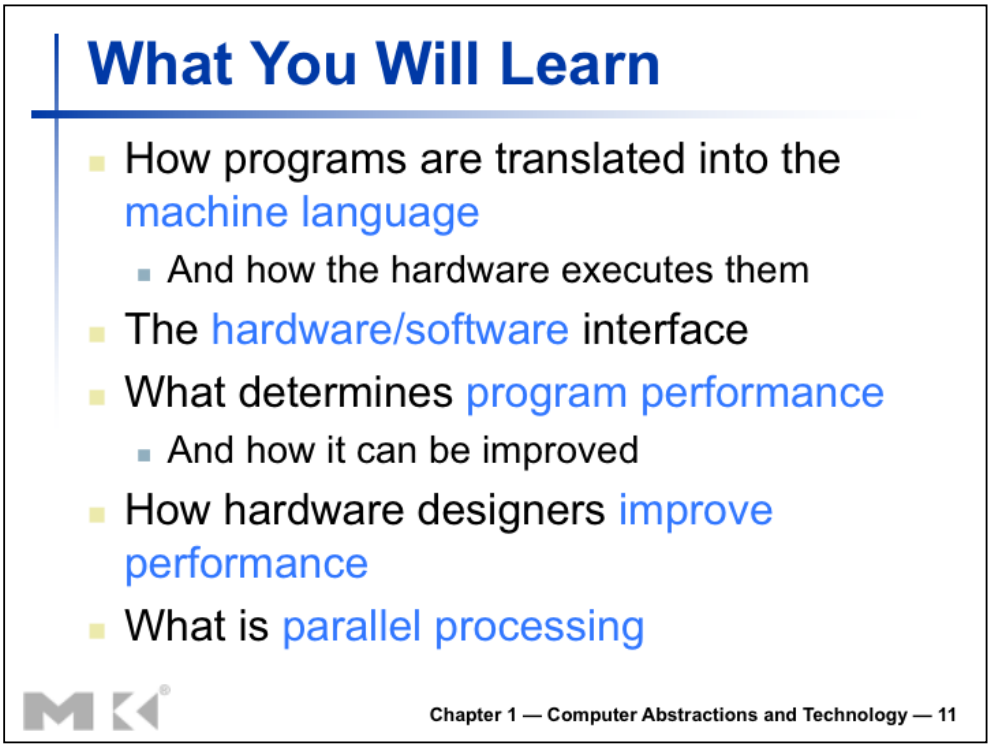
Embedded computers: Widest range of apps and performance

- includes microprocessors in cars
- computers in television sets
- Amazon echo, smart bulbs
- processors that control modern airplane or cargoship

- + Personal mobile devices

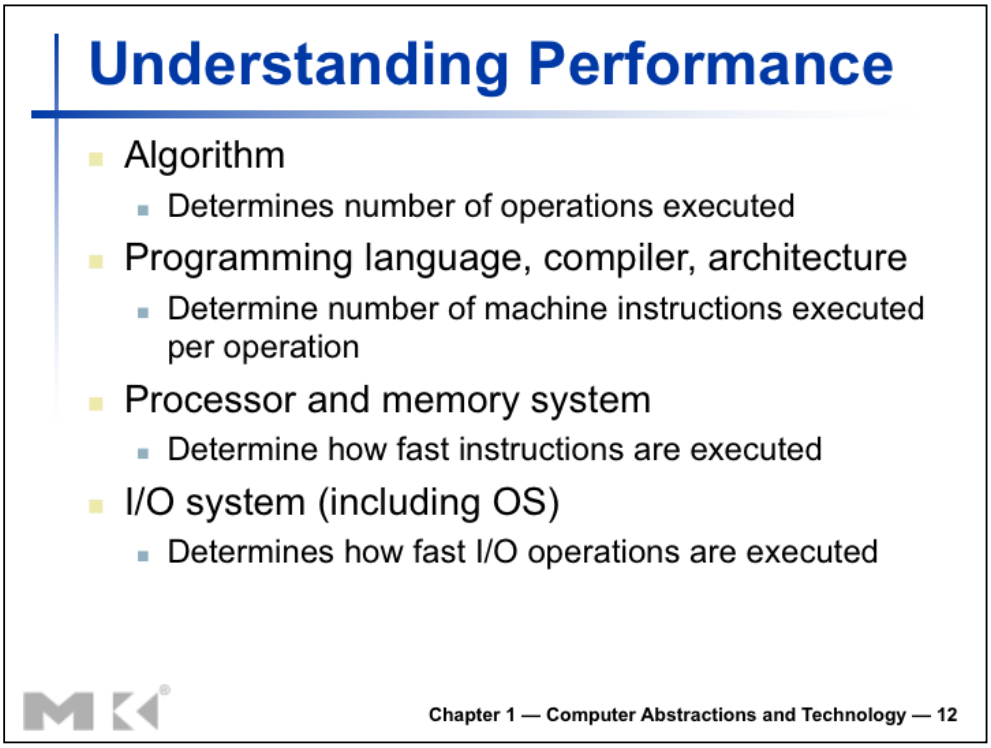


- Globally, devices and connections are growing faster (10 percent CAGR) than both the population (1.0 percent CAGR) and the Internet users (6 percent CAGR).
- A growing number of M2M applications, such as smart meters, video surveillance, healthcare monitoring, transportation, and package or asset tracking, are contributing in a major way to the growth of devices and connections.
- By 2023, M2M connections will be half or 50 percent of the total devices and connections.

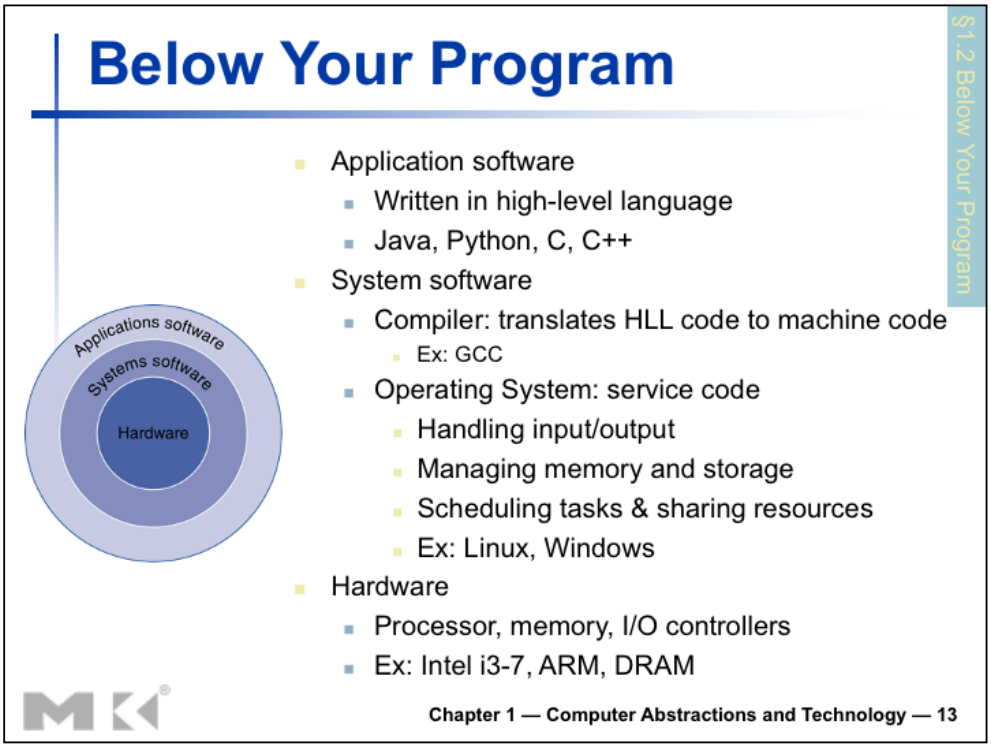


- How are programs written in a high-level lang, such as C or Java translated into the language of the hardware, and the hardware execute those programs?
- What is the interface between s/w and h/w? How s/w instruct hardware?
- What determines a performance & how it can be improved? program -> machine language -> h/w executes the program!
- What techniques h/w designers use to improve performance?
- Reasons for moving from sequential processing to parallel processing?

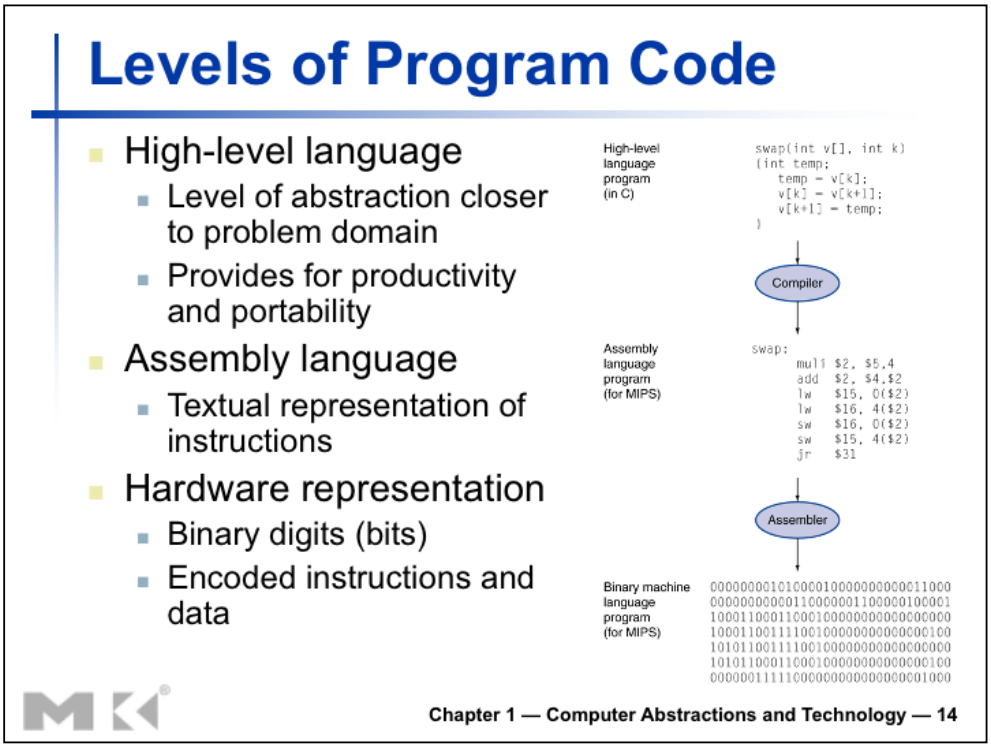
- Why one has to understand answers to these questions?
    - Without that improving program performance or evaluate what features make one computer better than another will be a complex process of trial and error!



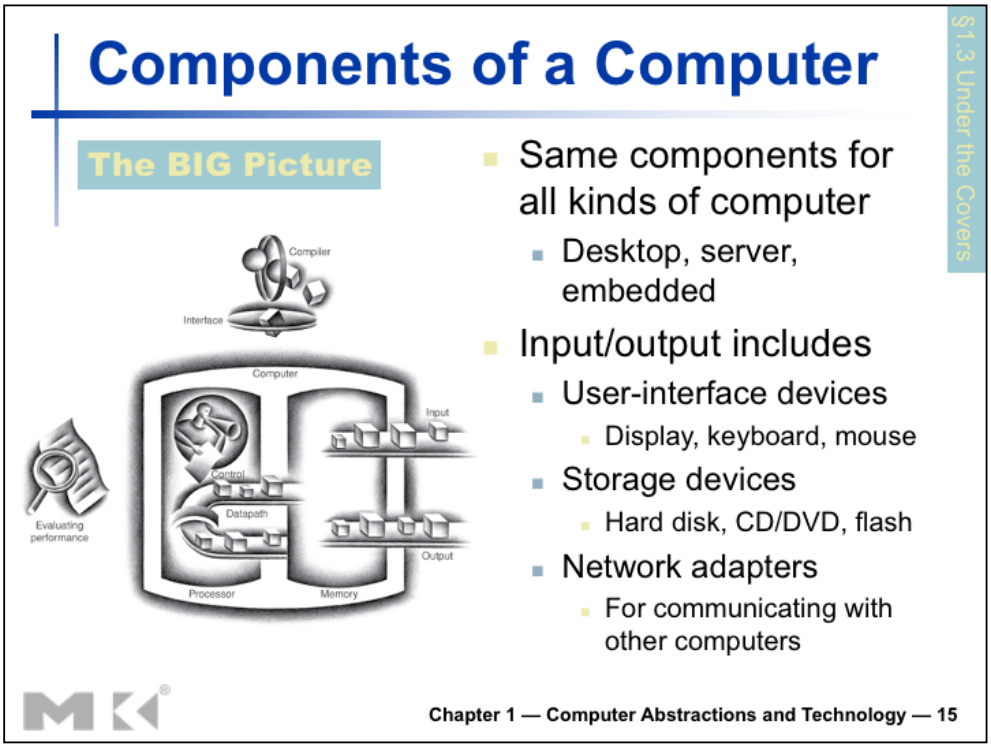
- Algorithm determines #operations
- PL, comp, arch determine #machine instructions per operation
- Processor and mem system determine how fast each instruction can be executed
- I/O system determine how fast I/O operations are executed



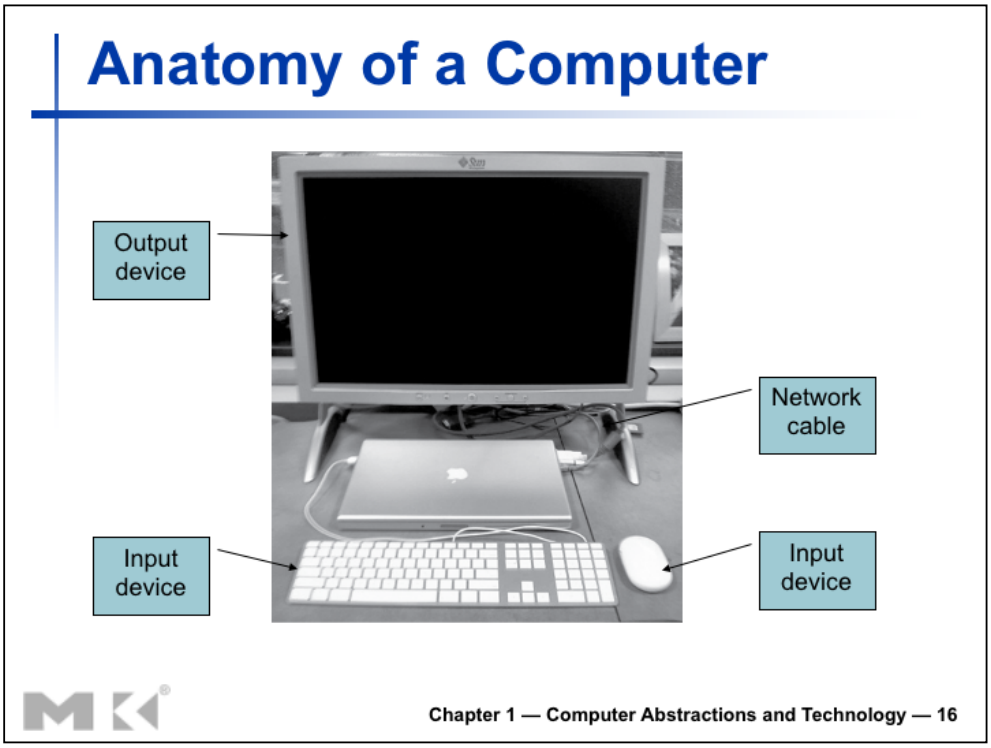
- App software (whatapp to banking apps to webrowser)
- underlying libraries both kernel/OS and user apps - compilers, s/w that talks to h/w, memory management, process/thread scheduling code, etc.
- CPU, DRAM, I/O controllers (PCI)

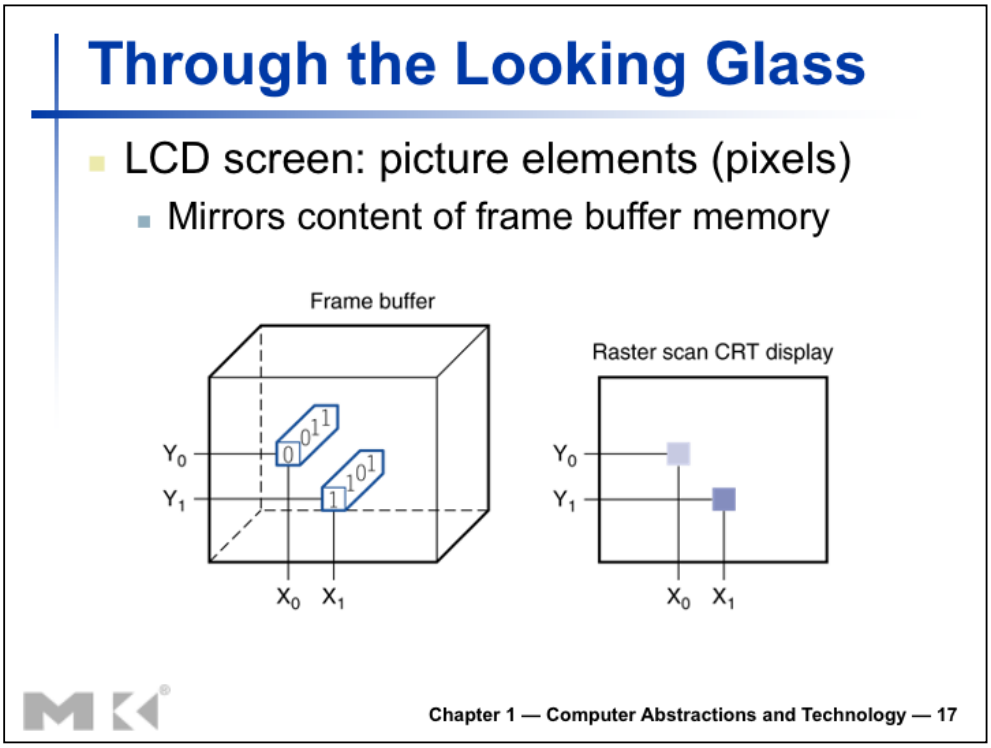


- Computers are slaves to commands
- However it understands only 1's and 0's – a tedious job for programmers to give instructions in 1's and 0's
    - Like alphabets
- So we invented assembler that translates textual instructions to binary
    - Like dictionary ☺
    - Assembly language requires programmer to write one line for every instruction, forcing programmer to think like the computer.
    - Although this is better than binary, still far from the human language, say normal English
- Today creation of HLL like c, java, python, are one of great breakthroughs
    - the creation increased productivity, and
    - make humans think in a more natural language.
- HLL → compilers → assembly language → assembly → binary code ☺

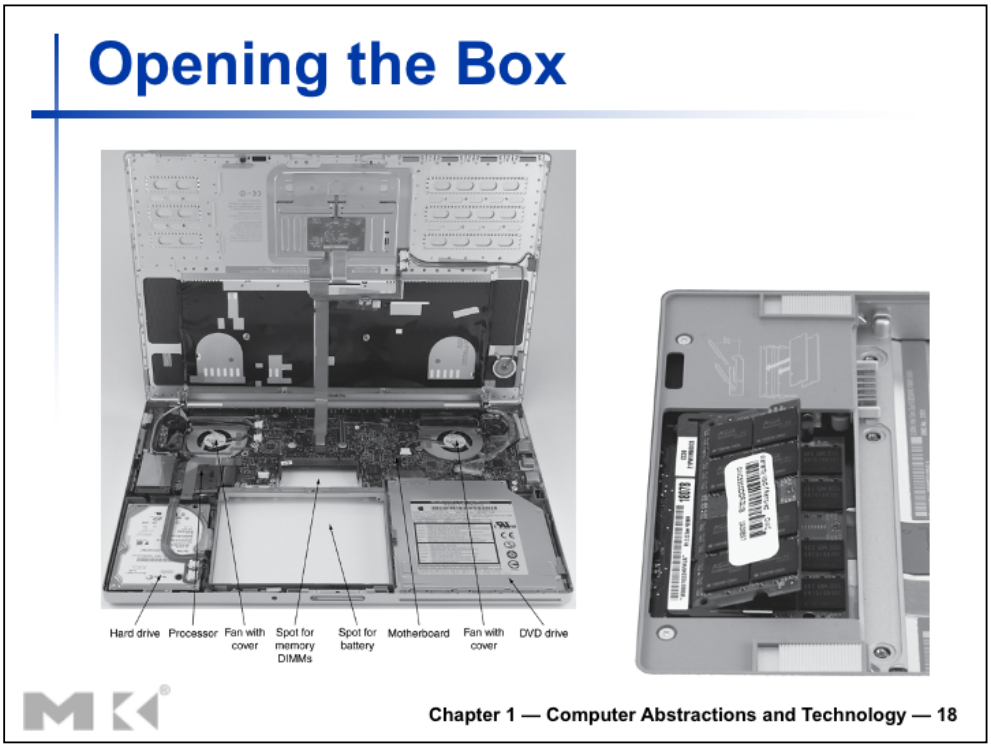


- Computer do mainly four basic operations are input, output, process and store. They are same for all kinds of computers – laptop to server to mobile to smart bulb.
- The processor (CPU) gets instructions from memory (DRAM)
- Input writes data to memory (Mouse,keyboard)
- Output reads data from memory (Harddisk, Monitor, N/w adapters)
- Control sends the signals that determine the operations of the data path, memory, input and output.


Anatomy of a Computer
Output device
Network cable
Input device
Input device
Chapter 1 — Computer Abstractions and Technology — 16

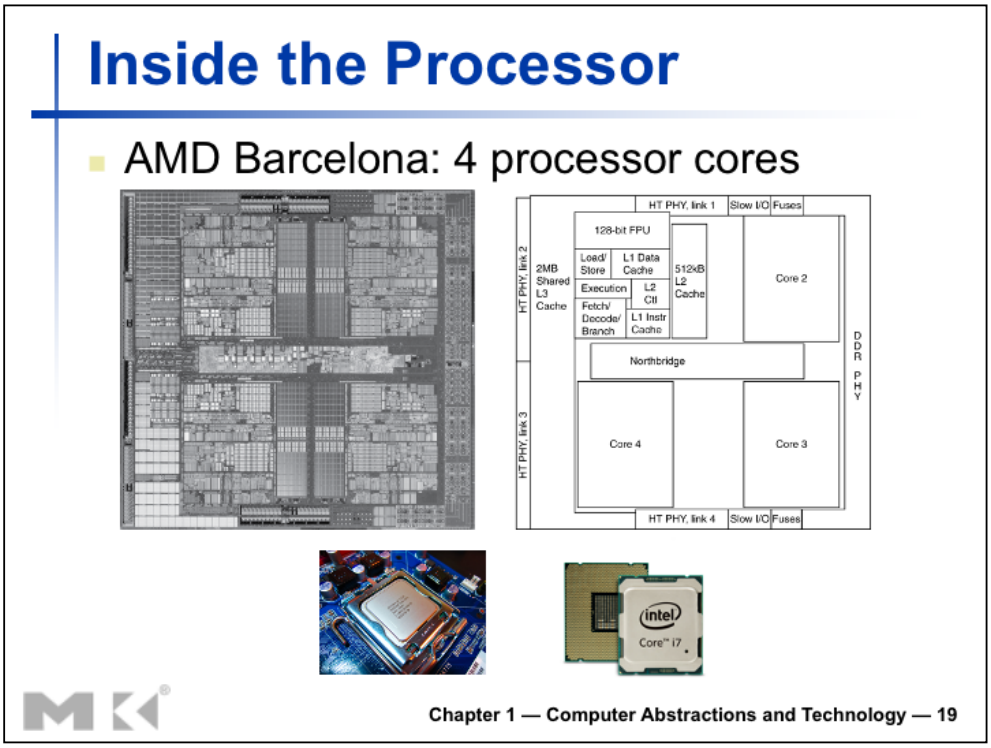


- 640 * 480, 2560 * 1600 matrix of bits also called pixels
- The computer hardware support for graphics consists mainly of a *frame buffer*, to store the bit map.
- The image to be represented onscreen is stored in the frame buffer, and the bit pattern per pixel is read out to the graphics display at the refresh rate.
    - shows a frame buffer with a simplified design of just 4 bits per pixel.
- Pixel x0,y0 contains the bit pattern 0011, which is a lighter shade on the screen than the bit pattern 1101 in pixel x1,y1

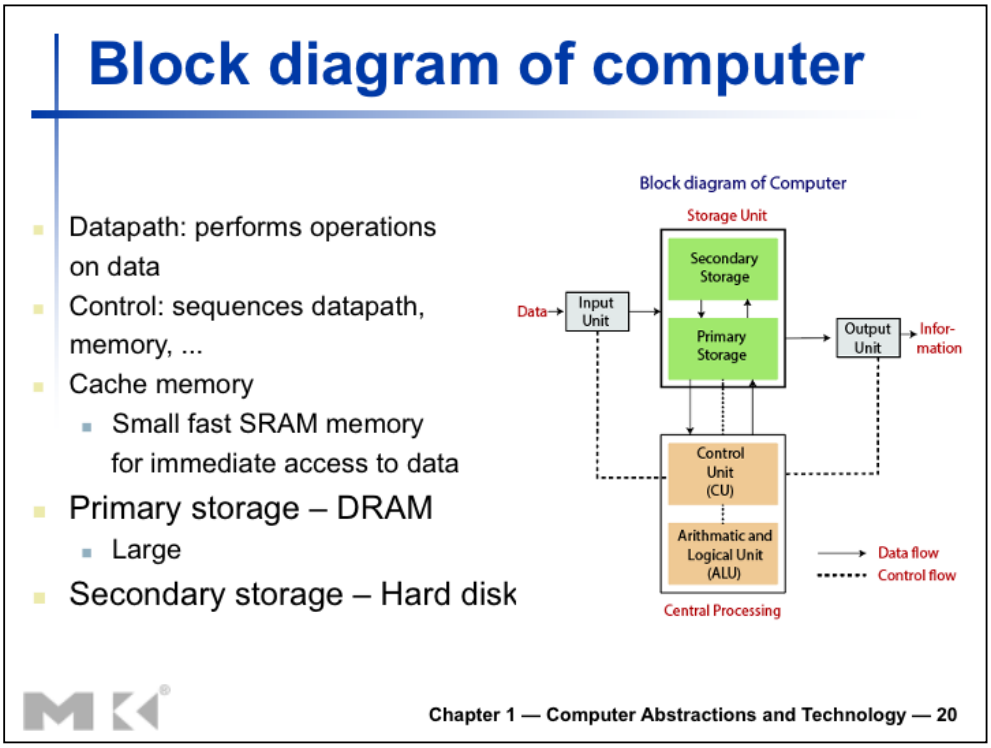


- The shiny box with the white label on the lower left is a 100 GB SATA hard disk drive,
- and the shiny metal box on the lower right side is the DVD drive.
- The hole between them is where the laptop battery would be located.
- The small hole above the battery hole is for memory DIMMs. Right figure is a closeup of the DIMMs, which are inserted from the bottom in this laptop.
- Above the battery hole and DVD drive is a printed circuit board (PC board), called the *motherboard*, which contains most of the electronics of the computer. The board is composed of three pieces: the piece connecting to the I/O devices mentioned earlier, the memory, and the processor.

- The two shiny circles in the upper half of the picture are two fans with covers.
- The processor is the large raised rectangle just below the left fan.



- The lefthand side is a microphotograph of the AMD Barcelona processor chip, and the righthand side shows the major blocks in the processor.
- This chip has four processors or “cores”. The microprocessor in the laptop has two cores per chip, called an Intel Core 2 Duo.



- The **datapath** performs the arithmetic operations, and **control** tells the datapath, memory, and I/O devices what to do according to the wishes of the instructions of the HLL program.
- **Cache memory** consists of a small, fast memory that acts as a buffer for the DRAM memory.
- Cache is built using a different memory technology, **static random access memory** (**SRAM**). SRAM is faster but less dense, and hence more expensive, than DRAM (more in next lectures)

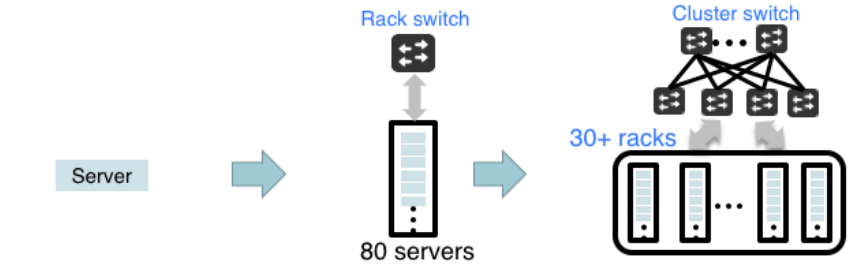
- volatile memory/main memory used to hold data and programs while they are running and
- this nonvolatile memory/ secondary memory used to store data and programs between runs
- **Flash memory**, a nonvolatile semiconductor tor memory, is used instead of disks in mobile devices such as cell phones and is increasingly replacing disks in music players and even laptops.
- Access times for magnetic disks are much slower than for DRAMs: disks typically take 5–20 milliseconds, while DRAMs take 50–70 nanoseconds—making DRAMs about 100,000 times faster.
- Thus, there are three primary differences between magnetic disks and main memory: disks are nonvolatile because they are magnetic; they have a slower access time because they are mechanical devices; and they are cheaper per gigabyte because they have very high storage capacity at a modest cost.
- Many have tried to invent a technology cheaper than DRAM but faster than disk to fill that gap, but many have failed.
- Flash memory, however, is a serious challenger. This semiconductor memory is nonvolatile like disks and has about the same bandwidth, but latency is 100 to 1000 times faster than disk.

Networks

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

Networks enable compute and storage at massive scales

The network has thousands of switches and millions of links
(Datacenters built by Google, facebook, Amazon, MS..)



- NIC: 10Gbps
- CPUs : 64
- DRAM: 16GB, 100ns
- Disk: 2TB, 10ms

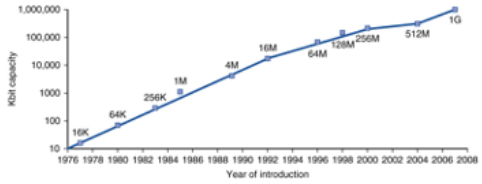
- Bandwidth: 800Gbps
- CPUs: 5K
- DRAM: 1TB, 300us
- Disk: 160TB, 11ms

- Bandwidth: 24Tbps
- CPUs : 153K
- DRAM: 30TB, 500us
- Disk: 4.80PB, 12ms

Reference: Jeff Dean. Designs, Lessons and Advice from Building Large Distributed Systems.

Technology Trends

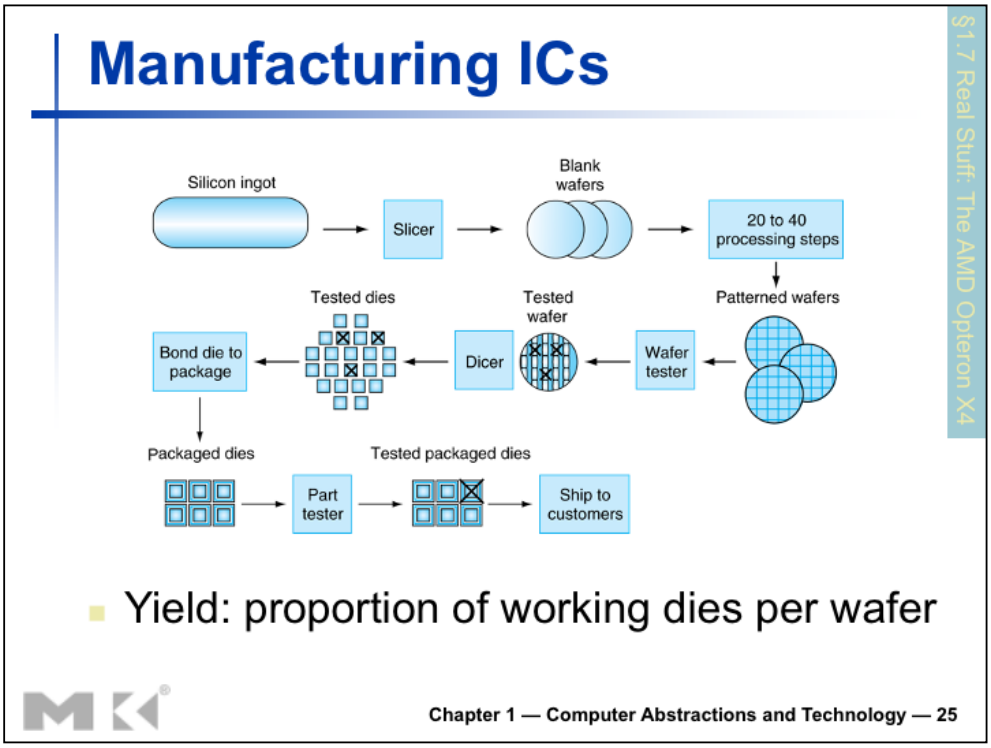
- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost



DRAM capacity

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	6,200,000,000

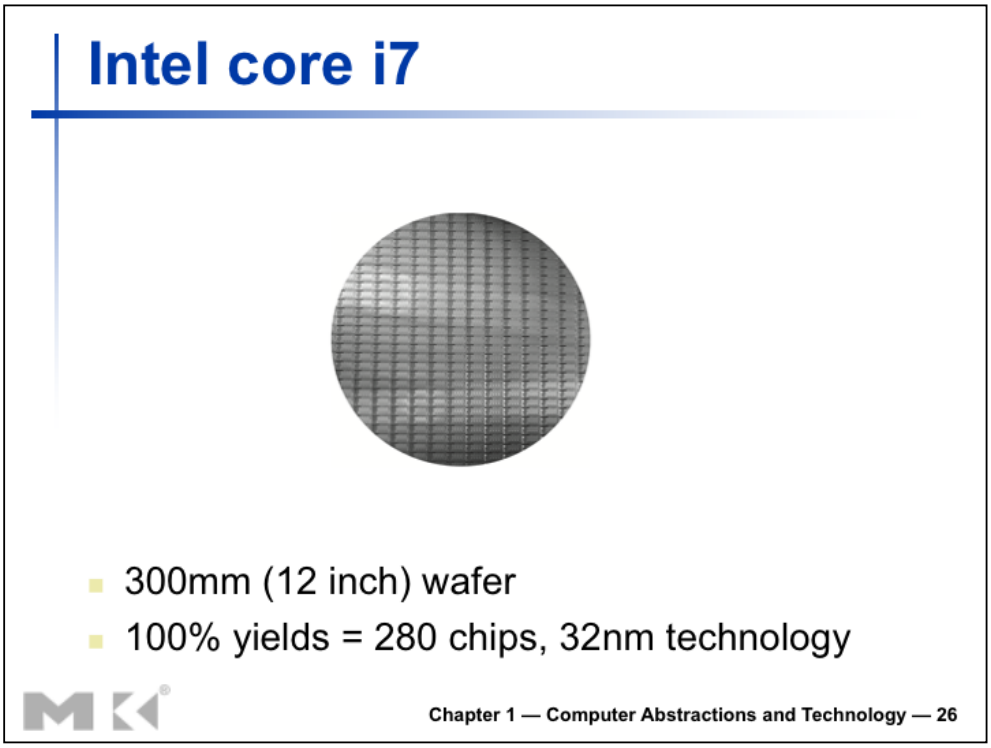
- Moore's law -- that the number of transistors on a microchip doubles every two years, though the cost of computers is halved.
- This table estimates relative performance per unit cost for each technology
  - A transistor is simply an on/off switch controlled by electricity
  - The IC combine dozens to hundreds of transistors into a single chip
  - VLSI : hundreds to millions
- The DRAM shows the growth in DRAM capacity since 1977 - quadrapuled capacity every 3 year



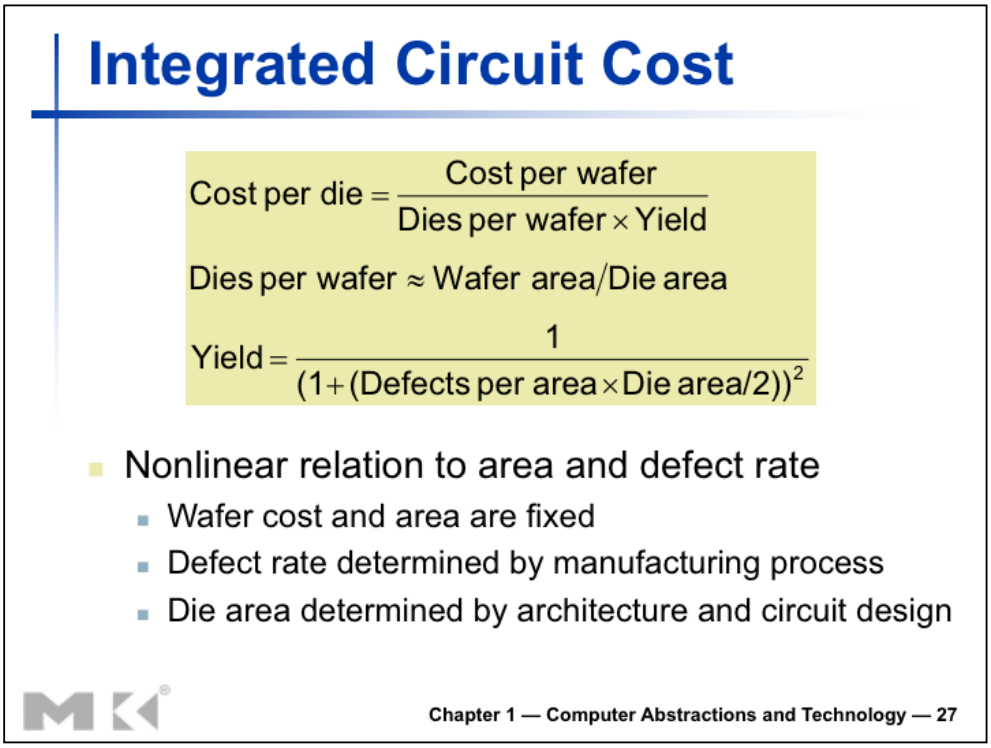
- Let's understand how ICs are manufacture
- Silicion is a substance found in sand, because it does not conduct electricity well, it is called a semiconductor
- With special chemical process and adding some materials to silicion, tiny areas transform into one of three devices
    - Excellent conductors of electricity (using either microscopic copper or aluminum wire)
    - Excellent insulators from electricity (like plastic sheathing or glass)
    - Areas that can conduct or insulate under special conditions (as a switch)
- Transistors fall in the last category. VLSI is just billions of combinations of conductors, insulators, and switches manufacture in a small packaging

- Processing:
    - Silicon Ingot: 8-12 inches in diameter and 12-24 long
    - Sliced into wafers
    - Each wafer goes through a series of chemical processing steps, during which chemical patterns create transistor, insulator, and conductors
- A single flaw in the wafer or in one of the dozens processing steps can result in that area of the wafer area failing

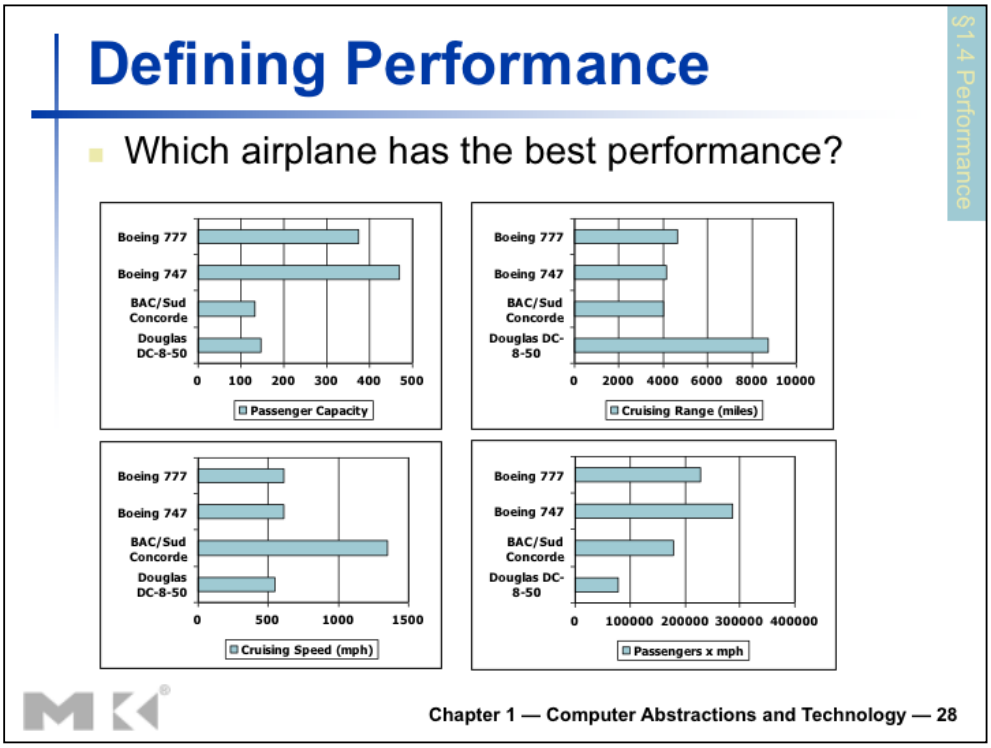
- It is impossible to make process perfect, so a best way to work around imperfection is to place many components on a single wafer
- The partiotioned wafer is diced into components called dies or informally called chips
- Dicing enables to discard dies, instead of whole wafer
- yield of a process: % of good dies from total number of dies
- Once you've found good dies, they are connected to the input/output pins of a package, using a process called *bonding*.
- These packaged parts are tested a final time, since mistakes can occur in packaging, and then they are shipped to customers.



- The number of dies on this 300 mm (12 inch) wafer at 100% yield is 280 chips/dies, each 20.7 by 10.5 mm.
- This die uses a 32-nanometer technology, which means that the smallest features are approximately 32 nm in size (e.g., transistor size)



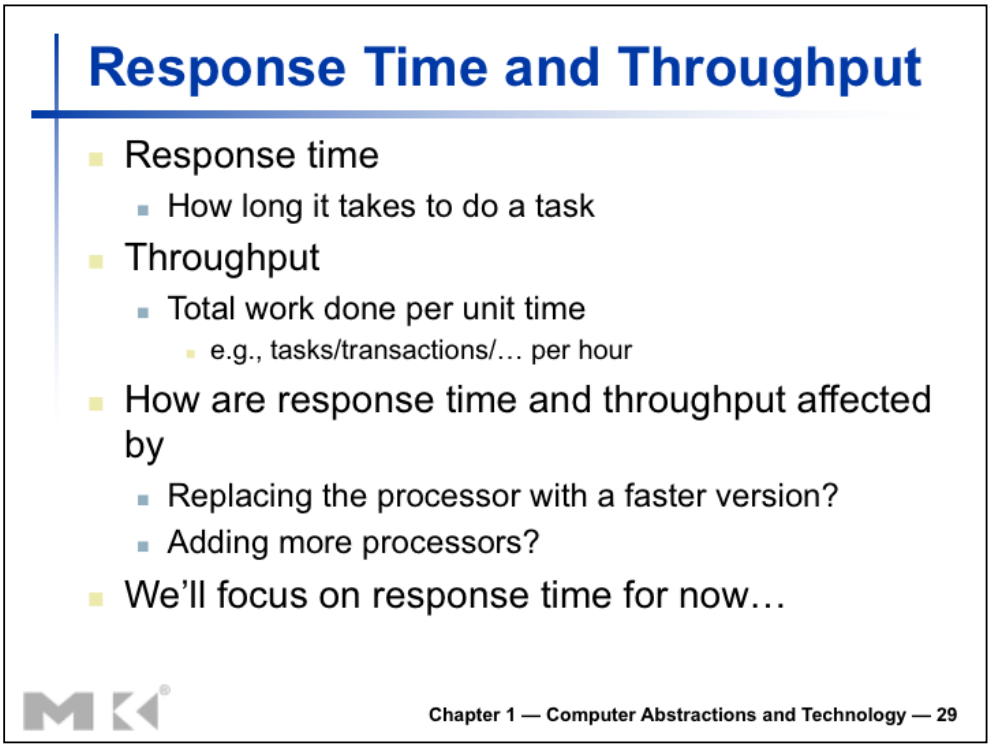
- The cost of an integrated circuit rises quickly as the die size increases, due both to the lower yield and the smaller number of dies that fit on a wafer.
- The first equation is straightforward to derive.
- The second is an approximation, since it does not subtract the area near the border of the round wafer that cannot accommodate the rectangular dies
- The final equation is based on empirical observations of yields at integrated circuit factories, with the exponent related to the number of critical processing steps.



How do you define performance?

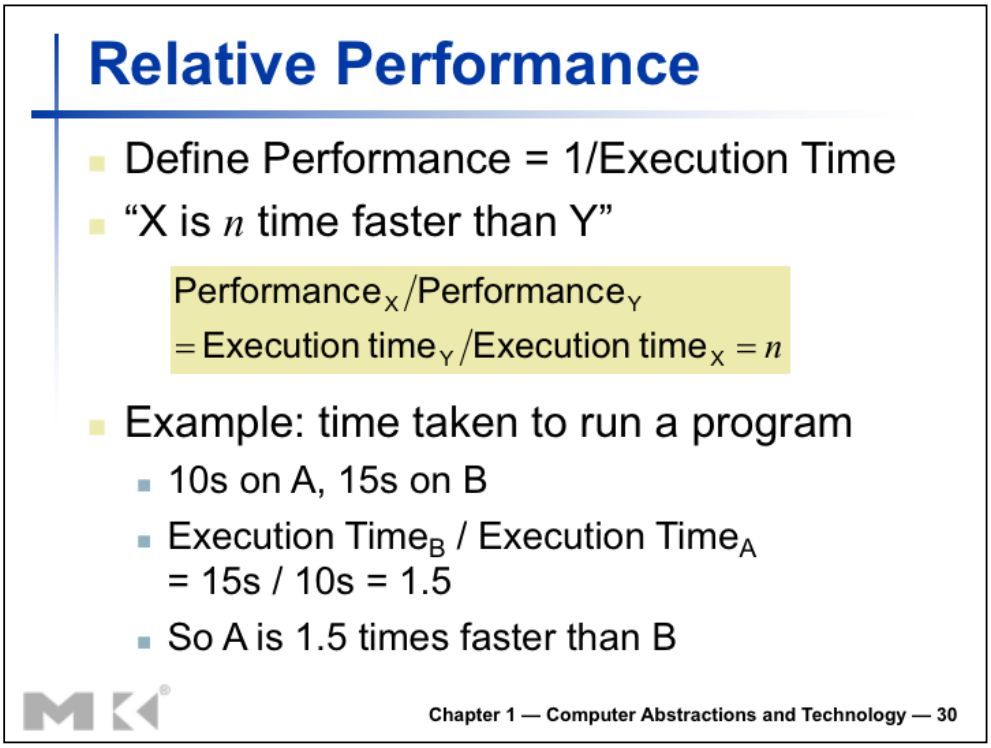
- speed? highest cruising speed ➔ concorde
- range? longest range → DC-8
- Capacity? Largest capacity → Boeing 747
- Which is faster?:
    - For one passenger? → concorde is the winner
    - For many passengers? → boeing 747 is the winner (#passenger x mph)

- Running a same program on two diff desktop comp? which is faster?
    - Desktop computer that gets job done first
- Running a datacenter with several serveres running jobs submitted by webusers
    - fast datacenter is the one which complets most jobs during a day

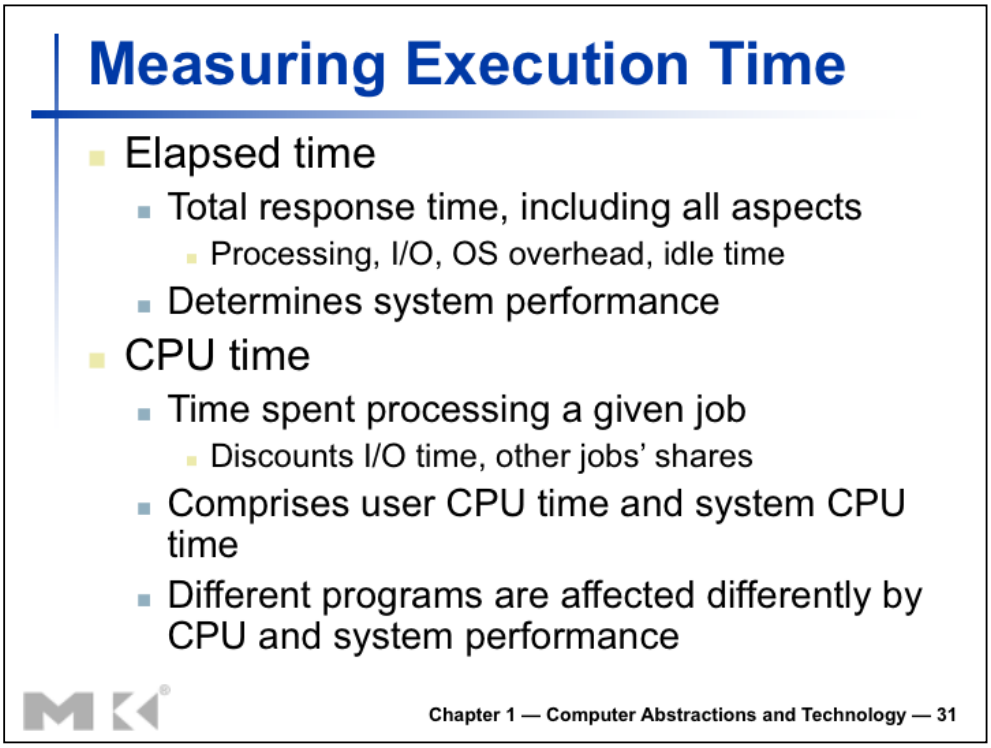


- As an individual computer user, you are interested in reducing **response time**—the time between the start and completion of a task—also referred to as a execution time
- Datacenter managers are often interested in increasing **throughput** or **bandwidth**—the total amount of work done in a given time.
- Hence, in most cases, we will need different performance metrics to benchmark personal mobile devices, which are more focused on response time, versus servers, which are more focused on throughput.

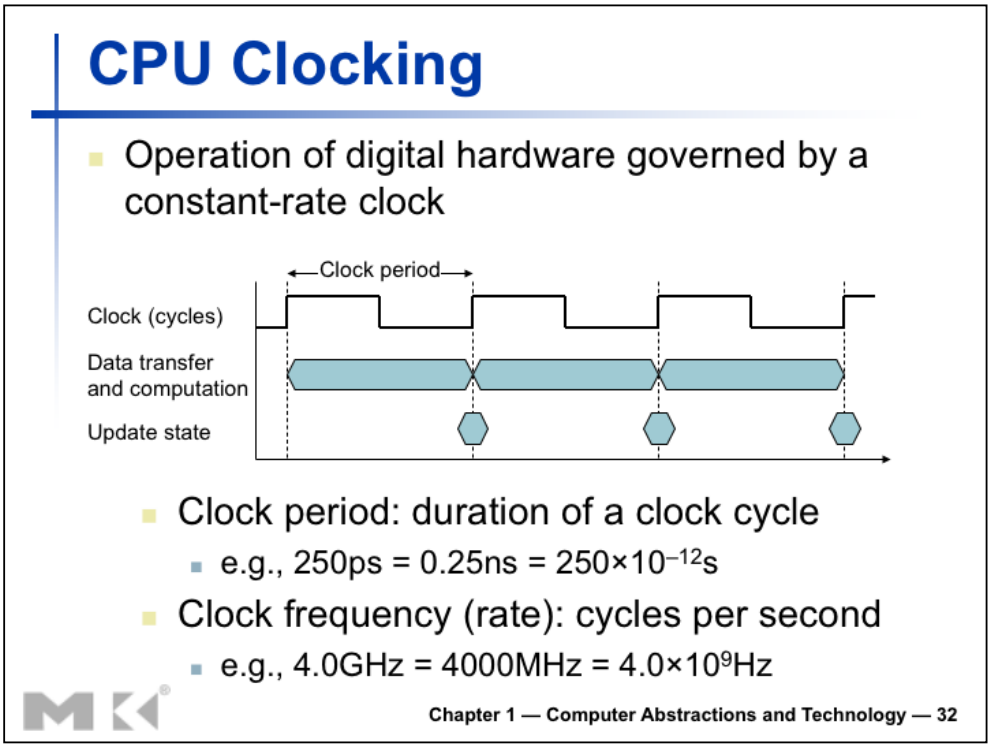
- How are time and throughput affected by
    - replacing faster processor
    - adding more processor
- Ans: Decreasing response time almost always improves throughput. Hence, in case 1, both response time and throughput are improved. In case 2, no one task gets work done faster, so only throughput increases.
- when it comes In comp perf, we primarliy focus on response time..



- To maximize performance, we want to minimize response time or execution time for some task.
- So performance is inverse of execution time
- To compare performance of two computers, e..g, how many times X is fast compared to Y, to get this we take the ratio: perf-x/perf-y or exectime-y/exectime-x
- See example…

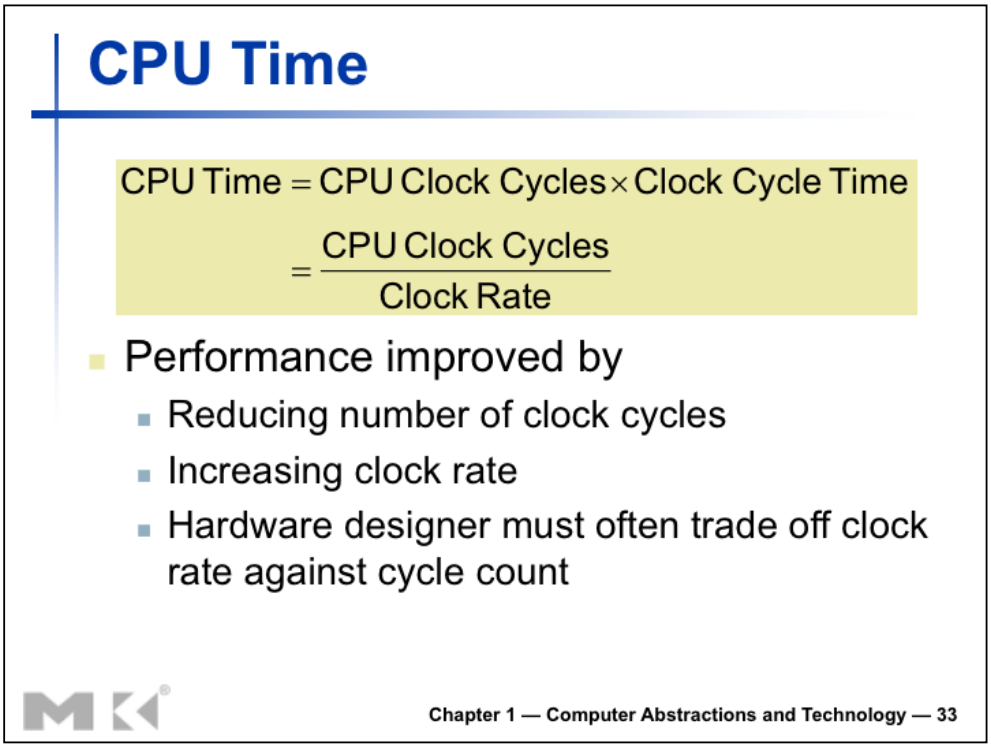


- Program execution time is measured in seconds per program.
- Response time or elapsed time includes all aspects – CPU processing, disk accesses, memory accesss, I/O delays, OS overhead
- We want to distinguish between the elapsed time and the time over which the processor is working on our behalf, called CPU time (eg., no waiting time for I/O or running other programs)
- CPU time can be further divided into CPU time spent on program, called user CPU time and the CPU time spend in the OS performing tasks on behalf of the program, called system CPU time
- CPU performance: refer to user CPU time
- System performance: refer to elapsed time on an unloaded system
- We will focus on CPU performance in this chapter



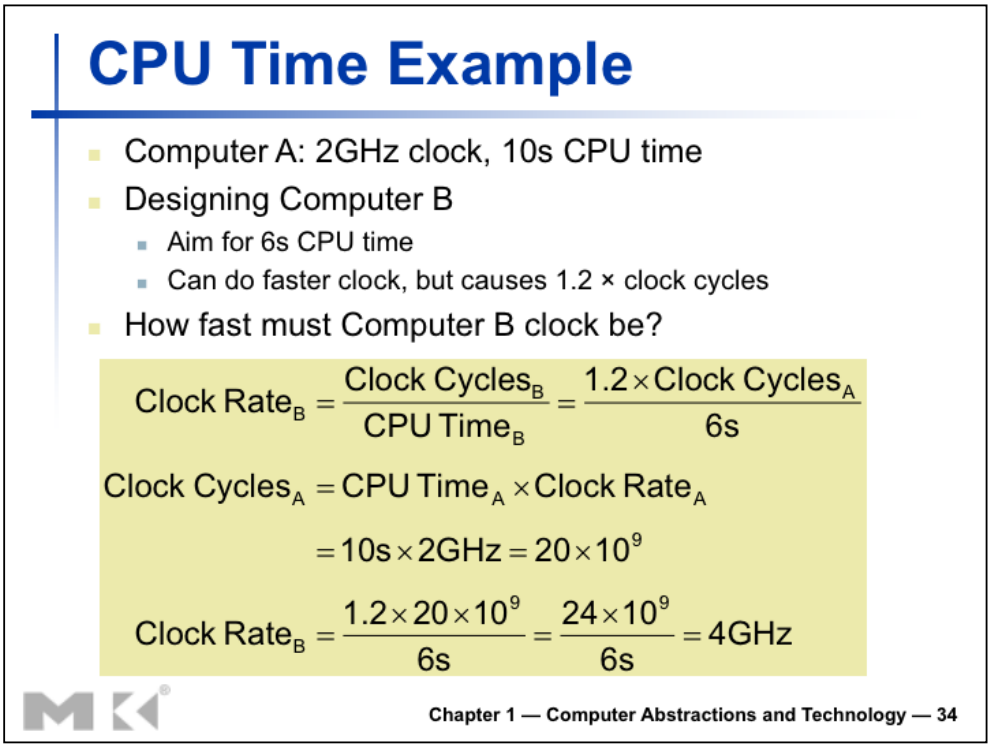
- Although time is what we care about, computer designers use a measure that relates to how fast the hardware can perform basic functions.

- All computers are constructed using a clock that determines when events take place in the hardware
- These discrete time intervals are called clock cycles or clock periods or clock ticks- the time for one clock period, usually of the processor clock, which runs at a constant time
- Clock period: length of each clock cycle
- Clock frequence (rate): #cycles per second – inverse of clock period

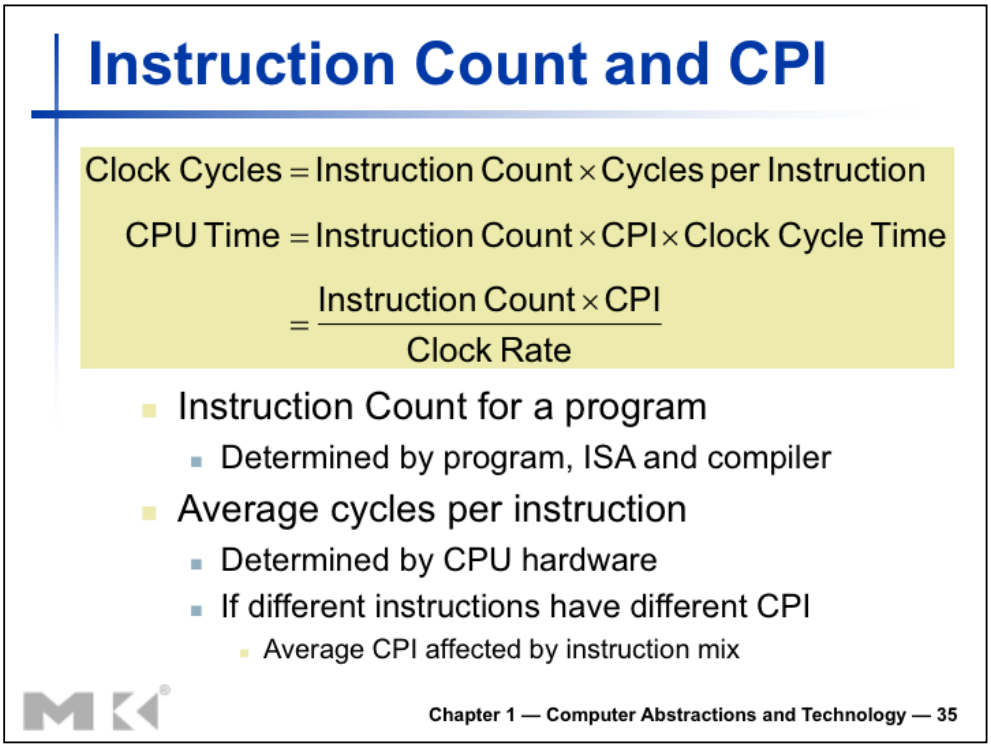


- Users and designers often examine performance using different metrics. If we could relate these different metrics, we could determine the effect of a design change on the performance as experienced by the user.

- CPU performance = CPU execution time → relates to basic metrics – clock cycles and cycle time
- A hardware designer can improve performance either by reducing the #clock cycles required for a program or length of the clock cycle (cycle time)
- Later chapters discuss the tradeoff between cycle rate vs cycle count



Our favorite program runs in 10 seconds on computer A, which has a 2 GHz clock. We are trying to help a computer designer build a computer, B, which will run this program in 6 seconds. The designer has determined that a substantial increase in the clock rate is possible, but this increase will affect the rest of the CPU design, causing computer B to require 1.2 times as many clock cycles as computer A for this program. What clock rate should we tell the designer to target?



- We haven't discusses how many instructions needed for the program
- Compiler generates instructions to execute, and the computer had to execute the instructions (C-progam → Assembly instructions → Instructions in byte code)
- So the execution time depends on the #instructions in a program
- CPU clock cycles = instruction count x average clock cycles per instruction (CPI)
- Each instruction may take different number of clock cycles -- Same instruction take different time on different hardware implementations, but the instruction count could be the same.
- therefore, CPU time= instruction count x CPI x clock cycle time or (instruction count x CPI) / clock rate
- These formulas are particularly useful because they separate the three key factors that affect performance. We can use these formulas to compare two different implementations or to evaluate a design alternative if we know its impact on these three parameters.

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?

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

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

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

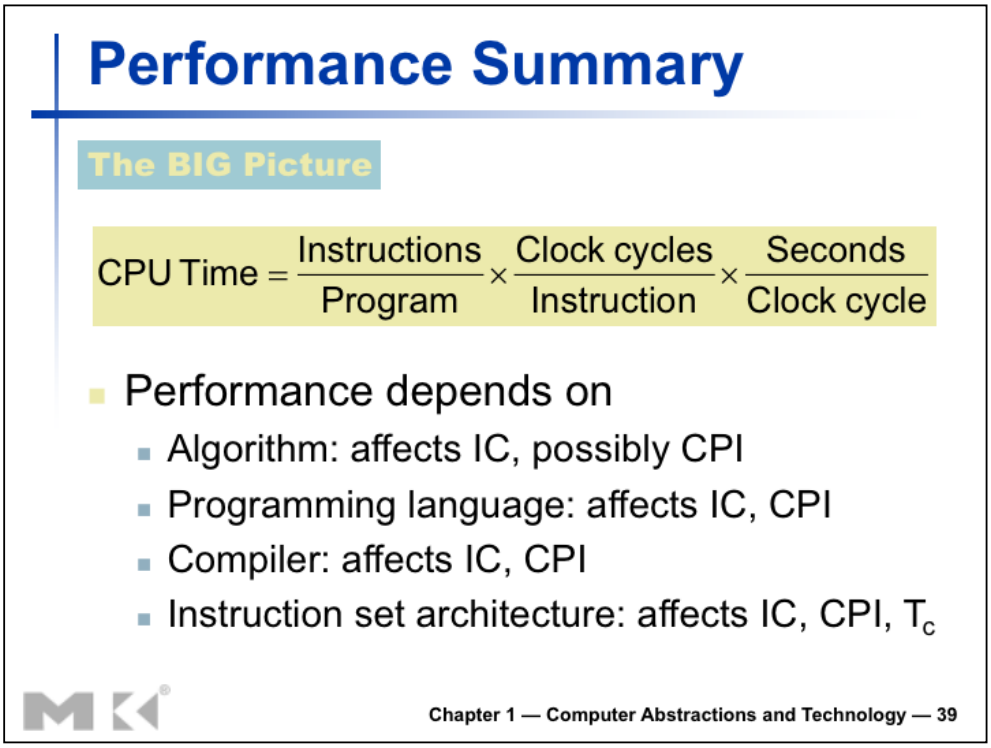
CPI Example

- 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
    = 2×1 + 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

Chapter 1 — Computer Abstractions and Technology — 38

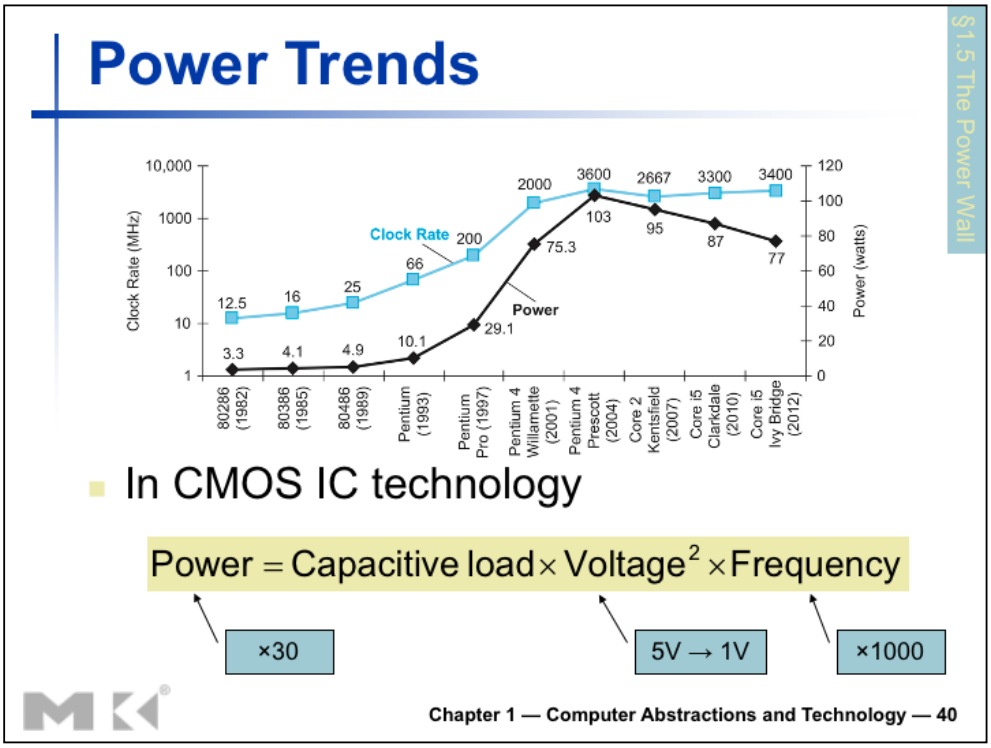


- The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. For example, if the algorithm uses more divides, it will tend to have a higher CPI.

- The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example, a language with heavy support for data abstraction (e.g., Java) will require indirect calls, which will use higher CPI instructions.

- The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler's role can be very complex and affect the CPI in complex ways.

- The instruction set architecture affects all three aspects of CPU performance, since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor.



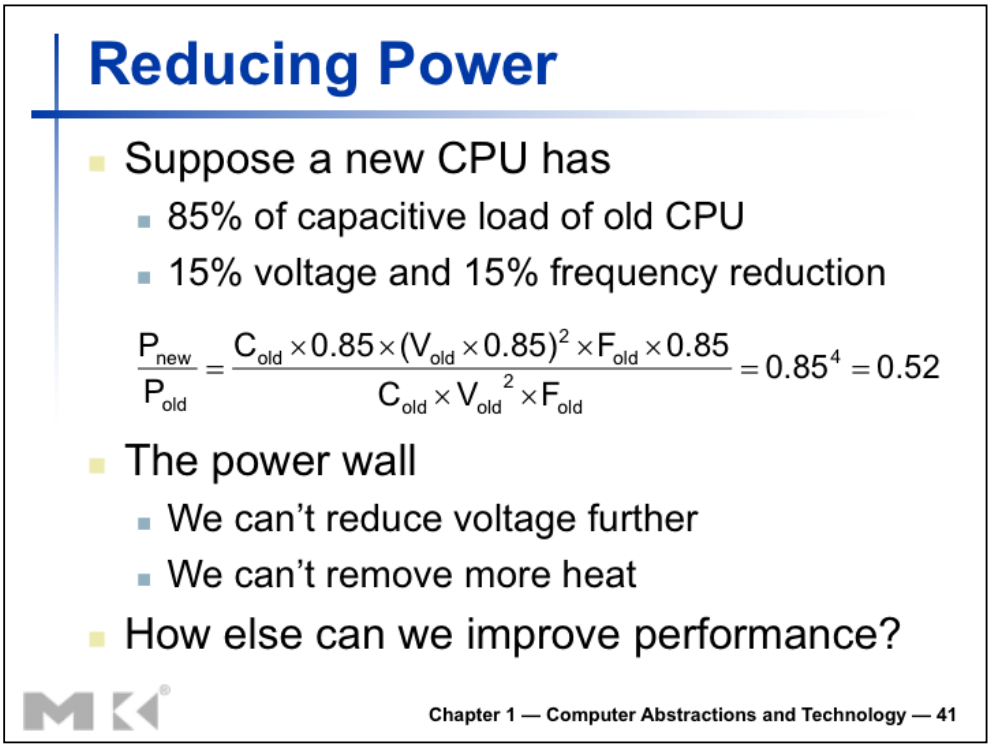
- Shows increase in clock rate and power of eight generations of intel processors
  - The Pentium 4 made a dramatic jump in clock rate and power but less so in performance. The Prescott thermal problems led to the abandonment of the Pentium 4 line. The Core 2 line reverts to a simpler pipeline with lower clock rates and multiple processors per chip. The Core i5 pipelines follow in its footsteps.

- Power and clock rate are corelated
- The reason for slowing down is we ran into practical power limit for cooling commodity processors
  - Battery life can trump performance in the personal mobile devices (laptop, mobiles)
  - Architects of servers, try to reduce costs of powering and cooling 100k servers as the costs are high at this scale
- Like CPU time for measuring performance, joules (joules/sec) is a better measure than a power rate like watts

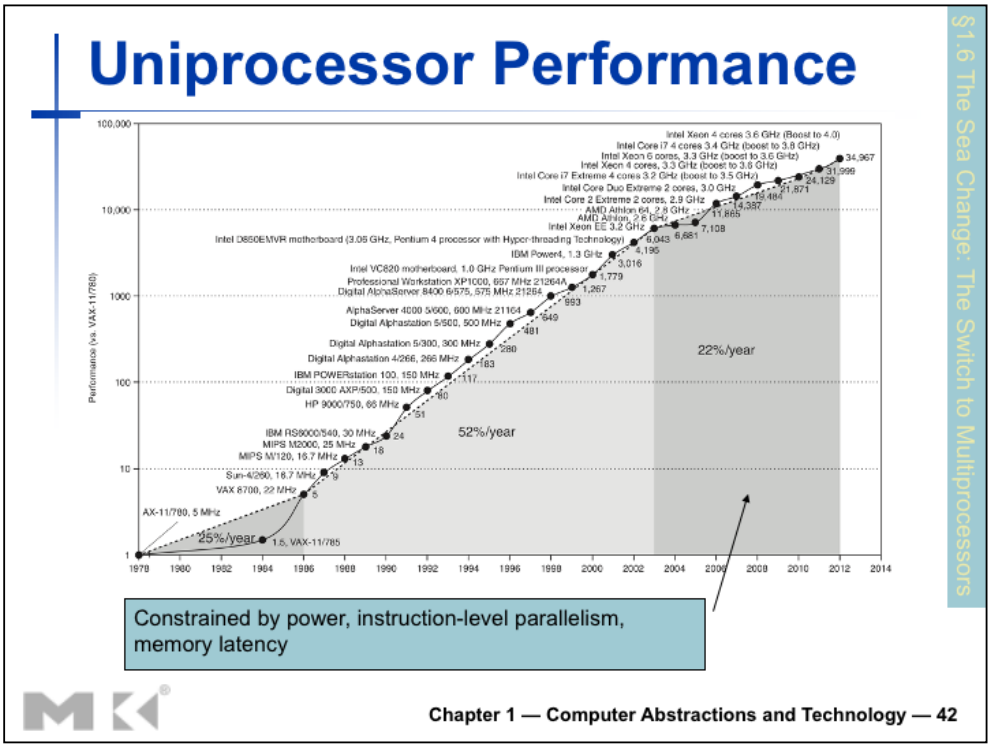
- CMOS is the dominant technology for Integrated circuits in processor
- For CMOS, primary source of energy consumption is so-called dynamic energy – which is energy consumed when transistors switch states from 0 to 1 and viceversa

- Energy is proportional to capacitive load x voltage^
- Power required per transistor is: power is proportional to capacitive load x voltage^2 x frequency switched
    - Frequency switched is a function of clock rate
    - Capacitive load per transistor is a function of both #transistors connected to output (fanout) and the technology

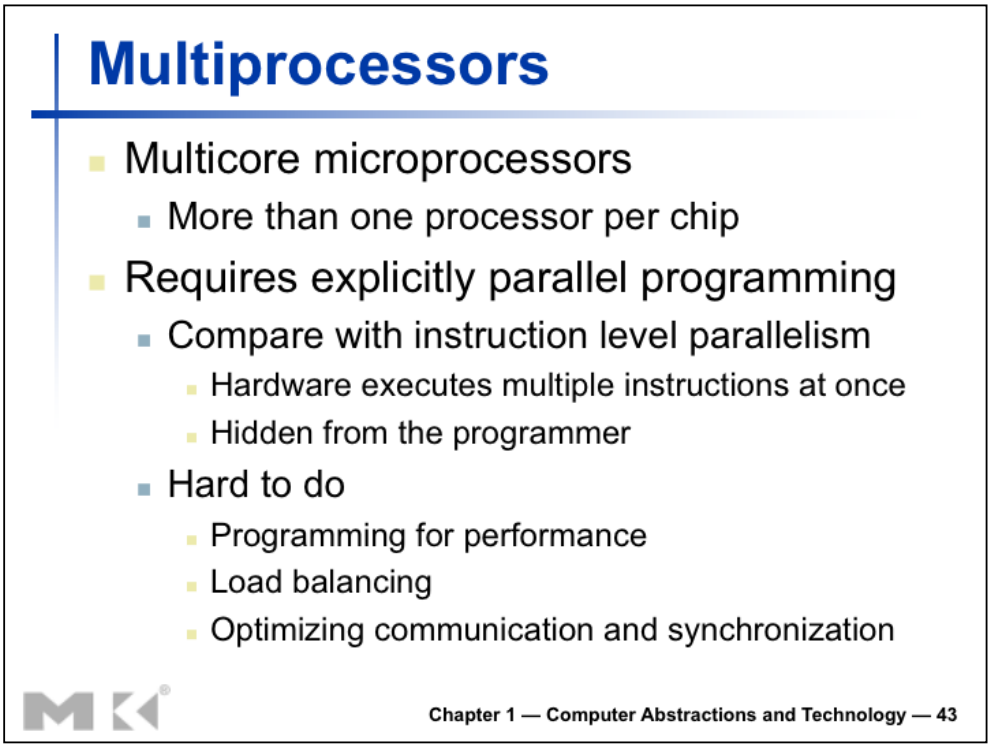
- How clock rate grow by a factor of 1000 while power grew by only a factor of 30?
    - Energy and thus power can be reduced by lowering the voltage, possible with new generation of technology
    - Since power is a function of the voltage squared, it worked
    - Voltage was reduced about 15% per generation, In 20 years, voltages have gone from 5V to 1V, power increase is only 30 times



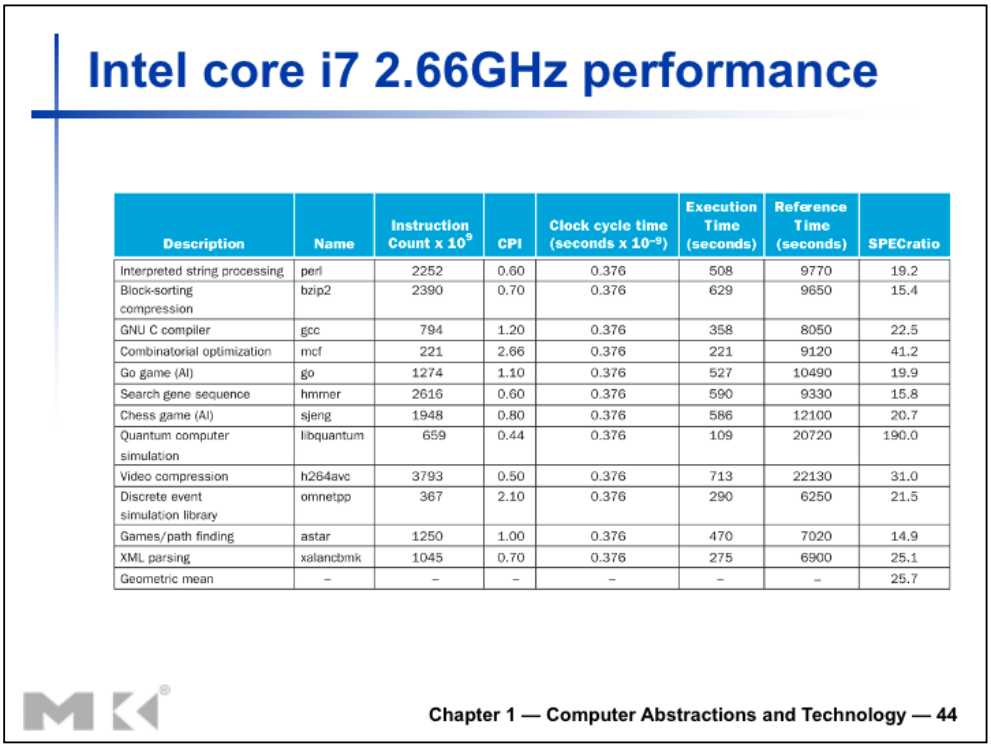
- Suppose we developed a new, simpler processor that has 85% of the capacitive load of the more complex older processor. Further, assume that it has adjustable voltage so that it can reduce voltage 15% compared to processor B, which results in a 15% shrink in frequency. What is the impact on dynamic power?
- Hence, the new processor uses about half the power of the old processor



- Prior to the mid-1980s, processor performance growth was largely technology-driven and averaged about 25% per year.
- The increase in growth to about 52% since then is attributable to more advanced architectural and organizational ideas. The higher annual performance improvement of 52% since the mid-1980s meant performance was about a factor of seven higher in 2002 than it would have been had it stayed at 25%.
- Since 2002, the limits of power, available instruction-level parallelism, and long memory latency have slowed uniprocessor performance recently, to about 22% per year.



- - In the past, programmers could rely on innovations in hardware, architecture, and compilers to double performance of their programs every 18 months without having to change a line of code. Today, for programmers to get significant improvement in response time, they need to rewrite their programs to take advantage of multiple processors.
- a “quadcore” microprocessor is a chip that contains four processors or four cores.
- Two types of parallelism:
    - Instruction-level parallelism already used heaviliy in uniprocessors (no changes are required to programs, hidden from the programmer)
    - Multicore/multiprocessor parallelism (changes required to programs to improve performance, programmers should rewrite their programs for multiprocessors)
- Parallel programming, hard because
    - increases difficulty of programming, divide application so each processor get same amount of work to do, overhead of scheduling and coordination
-

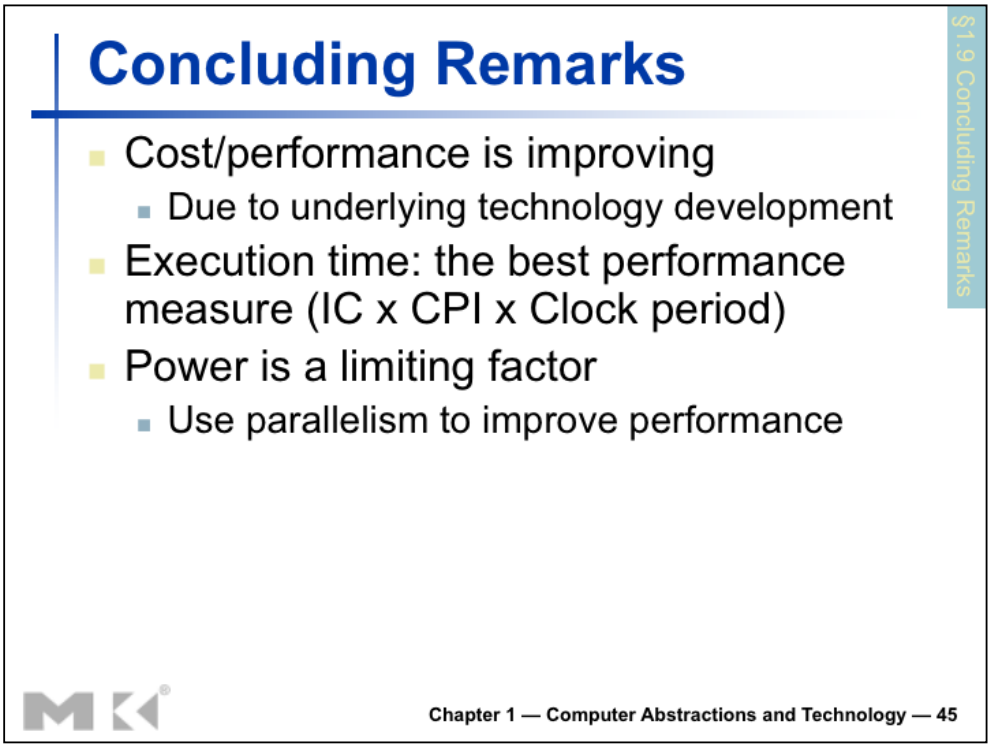


Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)	SPECratio
Interpreted string processing	perl	2252	0.60	0.376	508	9770	19.2
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650	15.4
GNU C compiler	gcc	794	1.20	0.376	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

- Benchmark – A program selected for use in comparing computer performance

- SPEC: System performance evaluation cooperative – created standard sets of benchmarks for modern computer systems

- Table: Execution time is the product of the three factors in this table: instruction count in billions, clocks per instruction (CPI), and clock cycle time in nanoseconds. SPECratio is simply the reference time, which is supplied by SPEC, divided by the measured execution time. The single number quoted as SPECINTC2006 is the geometric mean of the SPECratios.



- it's a safe bet that cost/performance will be much better than they are today.
- using the execution time of real programs as the metric is a reliable method of determining and reporting performance
- Conserving power while trying to increase performance has forced the hardware industry to switch to multicore microprocessors, thereby forcing the software industry to switch to programming parallel hardware. **Parallelism** is now required for performance.