Computer Abstractions and Technology

The computer revolution

- Computers in automobiles:computers reduce pollution, improve fuel efficiency via engine controls, and increase safety through nearly automated driving and air bag inflation to protect occupants in a crash
- Cell phones: person-to-person communication to almost anyone anywhere in the world?
- Human genome project:The cost of computer equipment to map and analyze human DNA sequences was hundreds of millions of dollars

- World Wide Web:the web has replaced libraries and newspapers.
- Search engines:As the content of the web grew in size and in value, finding relevant information became increasingly important.

Traditional Classes of Computing Applications and Their Characteristics

- personal computer (PC) A computer designed for use by an individual, usually incorporating a graphics display, a keyboard, and a mouse
- server A computer used for running larger programs for multiple users, often simultaneously, and typically accessed only via a network

Servers are built from the same basic technology as desktop computers, but provide for greater computing, storage, and input/output capacity. In general, servers also place a higher emphasis on dependability, since a crash is usually more costly than it would be on a single user PC

- supercomputer A class of computers with the highest performance and cost; they are configured as servers and typically cost tens to hundreds of millions of dollars
- embedded computer A computer inside another device used for running one predetermined application or collection of software

common size terms

Decimal term	Abbreviation	Value	Binary term	Abbreviation	Value	% Larger
kilobyte	KB	10^3	kibibyte	KiB	2^{10}	2%
megabyte	MB	10^6	mebibyte	MiB	2^{20}	5%
gigabyte	GB	10^9	gibibyte	GiB	2^{30}	7%
terabyte	TB	10^{12}	tebibyte	TiB	2^{40}	10%
petabyte	PB	10^{15}	pebibyte	PiB	2^{50}	13%
exabyte	EB	10^{18}	exbibyte	EiB	2^{60}	15%
zettabyte	ZB	10^{21}	zebibyte	ZiB	2^{70}	18%
yottabyte	YB	10^{24}	yobibyte	YiB	2^{80}	21%
ronnabyte	RB	10^{27}	robibyte	RiB	2^{90}	24%
queccabyte	QB	10^{30}	quebibyte	QiB	2^{100}	27%

- Personal mobile devices (PMDs) are small wireless devices to connect to the Internet; they rely on batteries for power, and software is installed by downloading apps. Conventional examples are smart phones and tablets.
- Cloud Computing refers to large collections of servers that provide services over the Internet; some providers rent dynamically varying numbers of servers as a utility.
- Software as a Service (SaaS) delivers software and data as a service over the Internet, usually via a thin program such as a browser that runs on local client devices, instead of binary code that must be installed, and runs wholly on that device. Examples include web search and social networking

Program Performance

Hardware or software component	How this component affects performance
Algorithm	Determines both the number of source-level statements and the number of I/O operations executed
Programming language, compiler, and architecture	Determines the number of computer instructions for each source-level statement
Processor and memory system	Determines how fast instructions can be executed
I/O system (hardware and operating system)	Determines how fast I/O operations may be executed

Seven Great Ideas in Computer Architecture

- Use Abstraction to Simplify Design- characterize the design at different levels of representation; lower-level details are hidden to offer a simpler model at higher levels.
- Make the Common Case Fast-Making the common case fast will tend to enhance performance better than optimizing the rare case. Ironically, the common case is often simpler than the rare case and hence is usually easier to enhance.
- Performance via Parallelism-more performance by computing operations in parallel

- Performance via Pipelining-A particular pattern of parallelism is so prevalent in computer architecture that it merits its own name: pipelining
- Performance via Prediction-In some cases, it can be faster on average to guess and start working rather than wait until you know for sure, assuming that the mechanism to recover from a misprediction is not too expensive and your prediction is relatively accurate

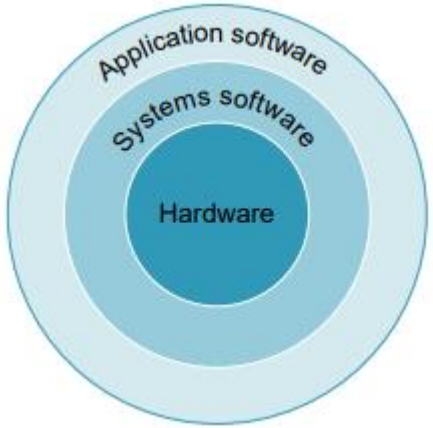
- Hierarchy of Memories-a hierarchy of memories, with the fastest, smallest, and the most expensive memory per bit at the top of the hierarchy and the slowest, largest, and cheapest per bit at the bottom.
- Dependability via Redundancy-Computers not only need to be fast; they need to be dependable. Since any physical device can fail, we make systems dependable by including redundant components that can take over when a failure occurs and to help detect failures

Below Your Program

- A typical application, such as a word processor or a large database system, may consist of millions of lines of code and rely on sophisticated software libraries that implement complex functions in support of the application.
- the hardware in a computer can only execute extremely simple low-level instructions. To go from a complex application to the primitive instructions involves several layers of software that interpret or translate high-level operations into simple computer instructions, an example of the great idea of abstraction.

A simplified view of hardware and software as hierarchical layers

- systems software Software that provides services that are commonly useful, including operating systems, compilers, loaders, and assembler
- operating system Supervising program that manages the resources of a computer for the benefit of the programs that run on that computer.



- An operating system interfaces between a user’s program and the hardware and provides a variety of services and supervisory functions. Among the most important functions are:

■ Handling basic input and output operations

■ Allocating storage and memory

■ Providing for protected sharing of the computer among multiple applications using it simultaneously

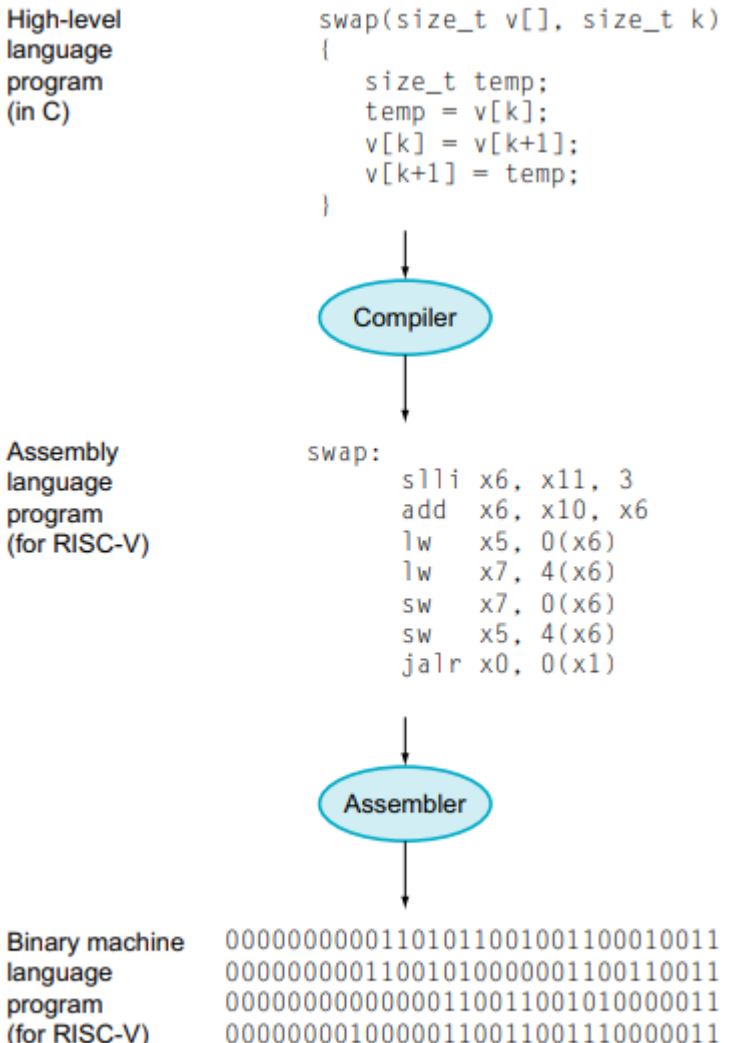
Ex:Linux, iOS, Android, and Windows

From a High-Level Language to the Language of Hardware

- compiler A program that translates high-level language statements into assembly language statements.
- binary digit Also called a bit. One of the two numbers in base 2 (0 or 1) that are the components of information
- instruction A command that computer hardware understands and obeys
- assembler A program that translates a symbolic version of instructions into the binary version

- assembly language A symbolic representation of machine instructions
- machine language A binary representation of machine instructions.
- high-level programming language A portable language such as C, C++, Java, or Visual Basic that is composed of words and algebraic notation that can be translated by a compiler into assembly language.

C program compiled into assembly language and then assembled into binary machine language



- central processor unit (CPU) Also called processor. The active part of the computer, which contains the datapath and control and which adds numbers, tests numbers, signals I/O devices to activate, and so on.
- datapath The component of the processor that performs arithmetic operations.
- control The component of the processor that commands the datapath, memory, and I/O devices according to the instructions of the program

- memory The storage area in which programs are kept when they are running and that contains the data needed by the running programs
- dynamic random access memory (DRAM) Memory built as an integrated circuit; it provides random access to any location. Access times are 50 nanoseconds and cost per gigabyte in 2012 was $5 to $10.
- static random access memory (SRAM). SRAM is faster but less dense, and hence more expensive, than DRAM

- cache memory A small, fast memory that acts as a buffer for a slower, larger memory.
- instruction set architecture Also called architecture. An abstract interface between the hardware and the lowest-level software that encompasses all the information necessary to write a machine language program that will run correctly, including instructions, registers, memory access, I/O, and so on.
- application binary interface (ABI) The user portion of the instruction set plus the operating system interfaces used by application programmers. It defines a standard for binary portability across computers.

- implementation Hardware that obeys the architecture abstraction.
- volatile memory Storage, such as DRAM, that retains data only if it is receiving power.
- nonvolatile memory A form of memory that retains data even in the absence of a power source and that is used to store programs between runs. A DVD disk is nonvolatile
- main memory Also called primary memory. Memory used to hold programs while they are running; typically consists of DRAM in today's computers
- secondary memory Nonvolatile memory used to store programs and data between runs; typically consists of flash memory in PMDs and magnetic disks in servers.
- flash memory A nonvolatile semiconductor memory. It is cheaper and slower than DRAM but more expensive per bit and faster than magnetic disks. Access times are about 5 to 50 microseconds and cost per gigabyte in 2020 was $0.06 to $0.12

Communicating with Other Computers

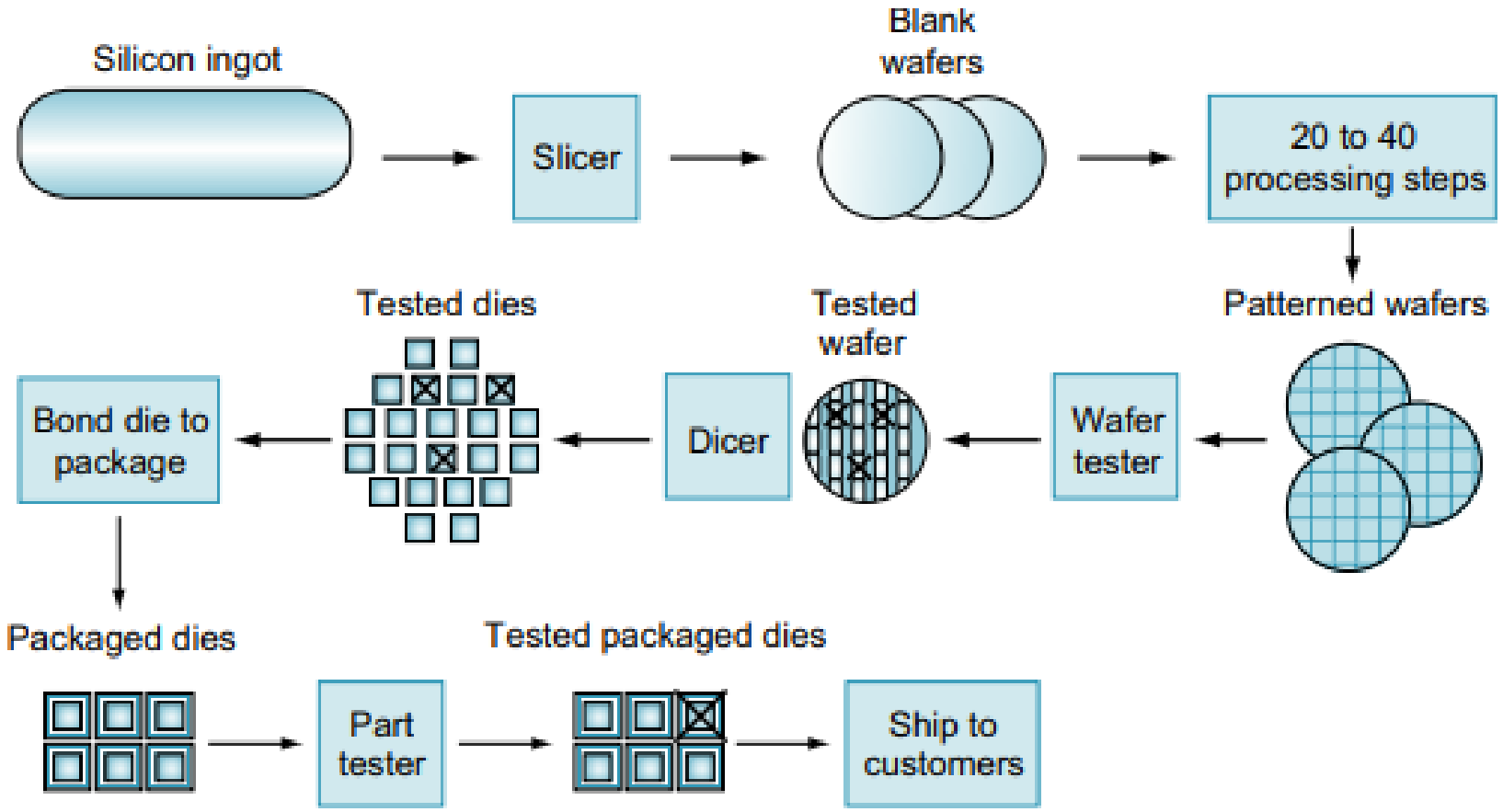
- Communication: Information is exchanged between computers at high speeds.
- Resource sharing: Rather than each computer having its own I/O devices, computers on the network can share I/O devices.
- Nonlocal access: By connecting computers over long distances, users need not be near the computer they are using

- local area network (LAN) A network designed to carry data within a geographically confined area, typically within a single building
- wide area network (WAN) A network extended over hundreds of kilometers that can span a continent.

Technologies for Building Processors and Memory

- transistor An on/off switch controlled by an electric signal.
- very large-scale integrated (VLSI) circuit A device containing hundreds of thousands to millions of transistors.
- silicon A natural element that is a semiconductor.
- semiconductor A substance that does not conduct electricity well.
- silicon crystal ingot A rod composed of a silicon crystal that is between 8 and 12 inches in diameter and about 12 to 24 inches long.
- wafer A slice from a silicon ingot no more than 0.1 inches thick, used to create chips

The chip manufacturing process



- defect A microscopic flaw in a wafer or in patterning steps that can result in the failure of the die containing that defect
- die The individual rectangular sections that are cut from a wafer, more informally known as chips.
- yield The percentage of good dies from the total number of dies on the wafer.

- : The cost of an integrated circuit can be expressed in three simple equations:

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

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

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

Performance

Airplane	Passenger capacity	Cruising range (miles)	Cruising speed (m.p.h.)	Passenger throughput (passengers × m.p.h.)
Boeing 737	240	3000	564	135,360
BAC/Sud Concorde	132	4000	1350	178,200
Boeing 777-200LR	301	9395	554	166,761
Airbus A380-800	853	8477	587	500,711

- response time Also called execution time. The total time required for the computer to complete a task, including disk accesses, memory accesses, I/O activities, operating system overhead, CPU execution time, and so on
- throughput Also called bandwidth. Another measure of performance, it is the number of tasks completed per unit time.

$$\text{Performance}_X = \frac{1}{\text{Execution time}_X}$$

$$\text{Performance}_X > \text{Performance}_Y$$

$$\frac{1}{\text{Execution time}_X} > \frac{1}{\text{Execution time}_Y}$$

$$\text{Execution time}_Y > \text{Execution time}_X$$

Relative Performance

$$\frac{\text{Performance}_X}{\text{Performance}_Y} = n$$

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

- If computer A runs a program in 10 seconds and computer B runs the same program in 15 seconds, how much faster is A than B?

We know that A is n times as fast as B if

$$\frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = n$$

Thus the performance ratio is

$$\frac{15}{10} = 1.5$$

and A is therefore 1.5 times as fast as B.

In the above example, we could also say that computer B is 1.5 times *slower than* computer A, since

$$\frac{\text{Performance}_A}{\text{Performance}_B} = 1.5$$

Measuring Performance

- CPU execution time Also called CPU time. The actual time the CPU spends computing for a specific task.
- user CPU time The CPU time spent in a program itself.
- system CPU time The CPU time spent in the operating system performing tasks on behalf of
- clock cycle Also called tick, clock tick, clock period, clock, or cycle. The time for one clock period, usually of the processor clock, which runs at a constant rate.
- clock period The length of each clock cycle.

CPU Performance and Its Factors

- the bottom-line performance measure is CPU execution time

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

Alternatively, because clock rate and clock cycle time are inverses,

$$\text{CPU execution time for a program} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

- Our favorite program runs in 10 seconds on computer A, which has a 2GHz 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?

Let's first find the number of clock cycles required for the program on A:

$$\text{CPU time}_A = \frac{\text{CPU clock cycles}_A}{\text{Clock rate}_A}$$

$$10\,\text{seconds} = \frac{\text{CPU clock cycles}_A}{2 \times 10^9 \,\frac{\text{cycles}}{\text{second}}}$$

$$\text{CPU clock cycles}_A = 10\,\text{seconds} \times 2 \times 10^9 \,\frac{\text{cycles}}{\text{second}} = 20 \times 10^9 \,\text{cycles}$$

CPU time for B can be found using this equation:

$$\text{CPU time}_B = \frac{1.2 \times \text{CPU clock cycles}_A}{\text{Clock rate}_B}$$

$$6 \text{ seconds} = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{\text{Clock rate}_B}$$

$$\text{Clock rate}_B = \frac{1.2 \times 20 \times 10^9 \text{ cycles}}{6 \text{ seconds}} = \frac{0.2 \times 20 \times 10^9 \text{ cycles}}{\text{second}} = \frac{4 \times 10^9 \text{ cycles}}{\text{second}} = 4 \text{ GHz}$$

To run the program in 6 seconds, B must have twice the clock rate of A.

Instruction Performance

- The performance equations above did not include any reference to the number of instructions needed for the program.
- clock cycles per instruction (CPI) Average number of clock cycles per instruction for a program or program fragment.

$$\text{CPU clock cycles} = \text{Instructions for a program} \times \frac{\text{Average clock cycles}}{\text{per instruction}}$$

- Suppose we have two implementations of the same instruction set architecture. Computer A has a clock cycle time of 250ps and a CPI of 2.0 for some program, and computer B has a clock cycle time of 500ps and a CPI of 1.2 for the same program. Which computer is faster for this program and by how much?

- We know that each computer executes the same number of instructions for the program; let's call this number I. First, find the number of processor clock cycles for each computer:

$$\text{CPU clock cycles}_A = I \times 2.0$$
$$\text{CPU clock cycles}_B = I \times 1.2$$

Now we can compute the CPU time for each computer:

$$\text{CPU time}_A = \text{CPU clock cycles}_A \times \text{Clock cycle time}$$
$$= I \times 2.0 \times 250 \text{ ps} = 500 \times I \text{ ps}$$

Likewise, for B:

$$\text{CPU time}_\text{B} = I \times 1.2 \times 500\,\text{ps} = 600 \times I\,\text{ps}$$

Clearly, computer A is faster. The amount faster is given by the ratio of the execution times:

$$\frac{\text{CPU performance}_\text{A}}{\text{CPU performance}_\text{B}} = \frac{\text{Execution time}_\text{B}}{\text{Execution time}_\text{A}} = \frac{600 \times I\,\text{ps}}{500 \times I\,\text{ps}} = 1.2$$

We can conclude that computer A is 1.2 times as fast as computer B for this program.

The Classic CPU Performance Equation

- instruction count The number of instructions executed by the program

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Clock cycle time}$$

or, since the clock rate is the inverse of clock cycle time:

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

Comparing Code Segments

- A compiler designer is trying to decide between two code sequences for a computer. The hardware designers have supplied the following facts:

A compiler designer is trying to decide between two code sequences for a computer. The hardware designers have supplied the following facts:

	CPI for each instruction class		
	A	B	C
CPI	1	2	3

For a particular high-level language statement, the compiler writer is considering two code sequences that require the following instruction counts:

Code sequence	Instruction counts for each instruction class		
	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? Which will be faster? What is the CPI for each sequence?

- Sequence 1 executes 2 + 1 + 2 = 5 instructions. Sequence 2 executes 4 + 1 + 1 = 6 instructions. Therefore, sequence 1 executes fewer instructions.

$$\text{CPU clock cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{C}_i)$$

$$\text{CPU clock cycles}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ cycles}$$
$$\text{CPU clock cycles}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ cycles}$$

- The CPI values can be computed by

$$\text{CPI} = \frac{\text{CPU clock cycles}}{\text{Instruction count}}$$

$$\text{CPI}_1 = \frac{\text{CPU clock cycles}_1}{\text{Instruction count}_1} = \frac{10}{5} = 2.0$$

$$\text{CPI}_2 = \frac{\text{CPU clock cycles}_2}{\text{Instruction count}_2} = \frac{9}{6} = 1.5$$

The basic components of performance and how each is measured

Components of performance	Units of measure
CPU execution time for a program	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

- All factors are combined to yield execution time measured in seconds per program:
- Always bear in mind that the only complete and reliable measure of computer performance is time

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

Hardware or software component	Affects what?	How?
Algorithm	Instruction count, CPI	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.
Programming language	Instruction count, 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.
Compiler	Instruction count, CPI	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 varied ways.
Instruction set architecture	Instruction count, clock rate, CPI	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.

- A given application written in Java runs 15 seconds on a desktop processor. A new Java compiler is released that requires only 0.6 as many instructions as the old compiler. Unfortunately, it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler? Pick the right answer from the three choices below:

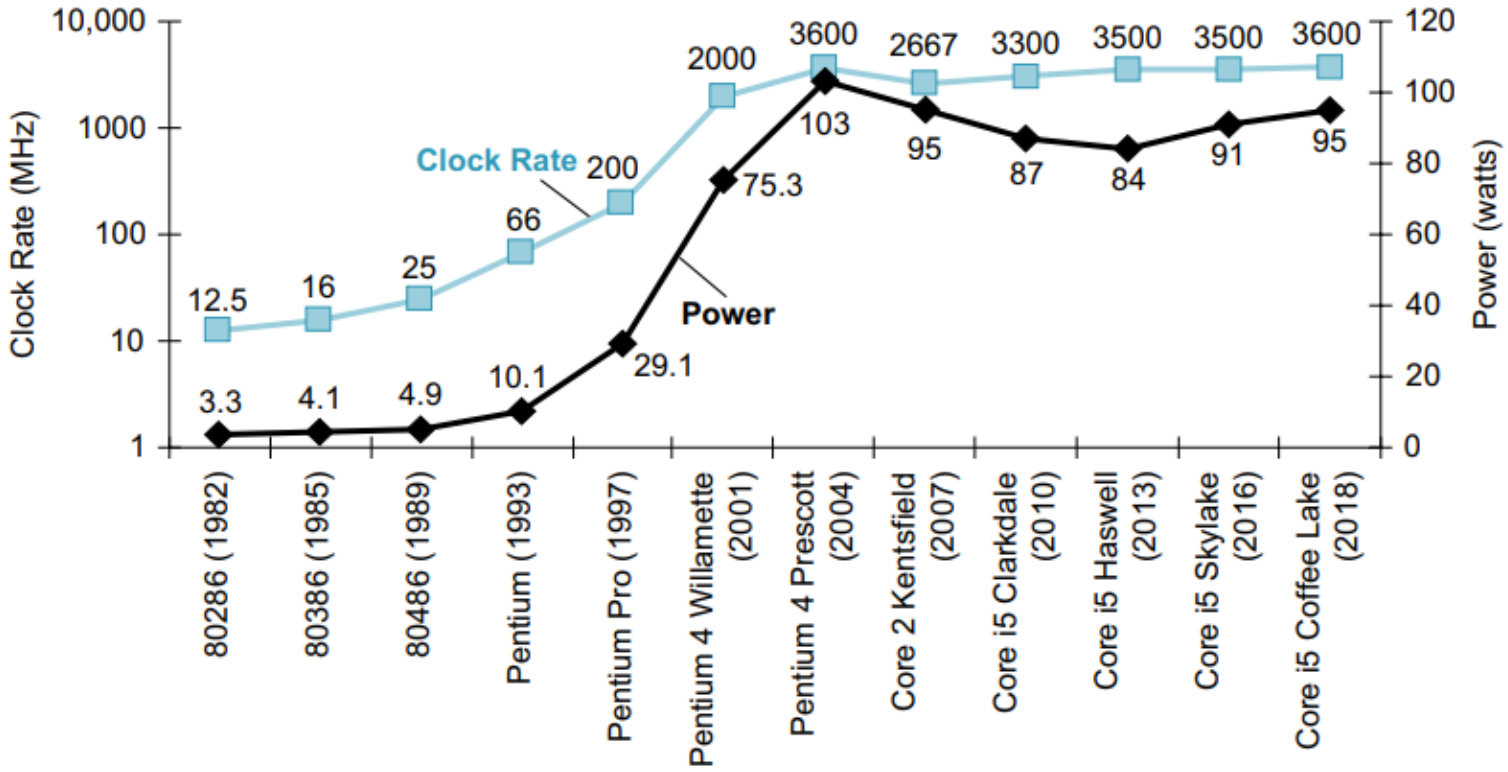
a. $\dfrac{15 \times 0.6}{1.1} = 8.2 \text{ sec}$

b. $15 \times 0.6 \times 1.1 = 9.9 \text{ sec}$

c. $\dfrac{1.5 \times 1.1}{0.6} = 27.5 \text{ sec}$

The Power Wall

- Clock rate and power for Intel x86 microprocessors over nine generations and 36 years



- Although power provides a limit to what we can cool, in the post-PC era the really valuable resource is energy
- The dominant technology for integrated circuits is called CMOS (complementary metal oxide semiconductor)
- The dynamic energy depends on the capacitive loading of each transistor and the voltage applied:

$$Energy \propto Capacitive\ load \times Voltage^2$$

$$Energy \propto 1/2 \times Capacitive\ load \times Voltage^2$$

$$Power \propto 1/2 \times Capacitive\ load \times Voltage^2 \times Frequency\ switched$$

- Frequency switched is a function of the clock rate.
- The capacitive load per transistor is a function of both the number of transistors connected to an output (called the fanout) and the technology, which determines the capacitance of both wires and transistors

- Suppose we developed a new, simpler processor that has 85% of the capacitive load of the more complex older processor. Further, assume that it can adjust 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?

$$\frac{\text{Power}_{\text{new}}}{\text{Power}_{\text{old}}} = \frac{\text{Capacitive load} \times 0.85 \times \text{Voltage} \times 0.85^{2} \times \text{Frequency switched} \times 0.85}{\text{Capacitive load} \times \text{Voltage}^{2} \times \text{Frequency switched}}$$

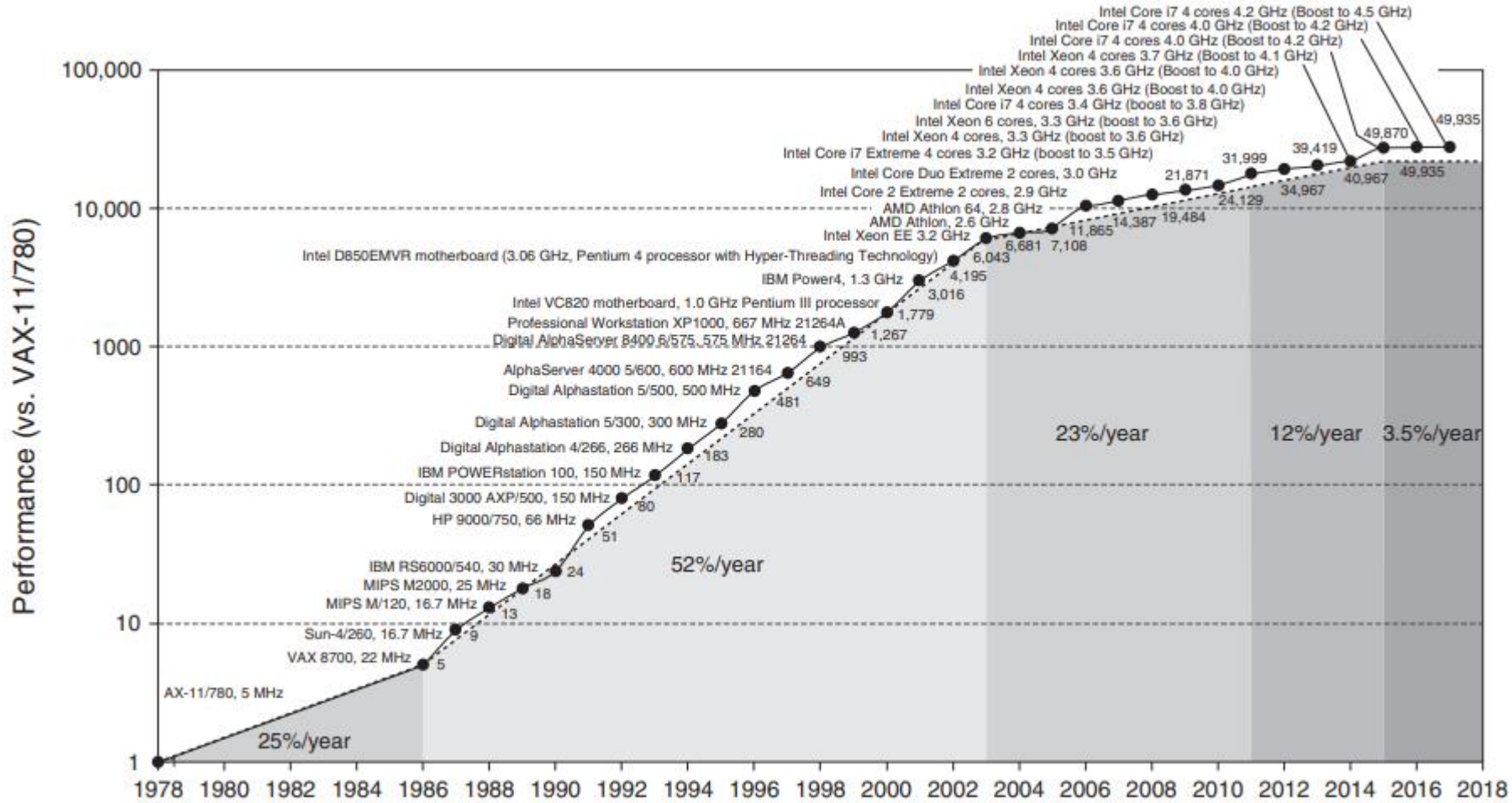
Thus the power ratio is

$$0.85^{4} = 0.52$$

The Sea Change: The Switch from Uniprocessors to Multiprocessors

- Parallelism has always been crucial to performance in computing, but it was often hidden
- Pipelining is one example of instruction-level parallelism, where the parallel nature of the hardware is abstracted away so the programmer and compiler can think of the hardware as executing instructions sequentially

Growth in processor performance since the mid-1980



- Parallelism and Instructions: Synchronization. Usually, independent parallel tasks need to coordinate at times, such as to say when they have completed their work.
- Parallelism and Computer Arithmetic: Subword Parallelism. Perhaps the simplest form of parallelism to build involves computing on elements in parallel, such as when multiplying two vectors. Subword parallelism relies on wider arithmetic units that can operate on many operands simultaneously

- Parallelism via Instructions. Given the difficulty of explicitly parallel programming, tremendous effort was invested in the 1990s in having the hardware and the compiler uncover implicit parallelism, initially via pipelining
- Parallelism and Memory Hierarchies: Cache Coherence. One way to lower the cost of communication is to have all processors use the same address space, so that any processor can read or write any data. Given that all processors today use caches to keep a temporary copy of the data in faster memory near the processor, it's easy to imagine that parallel programming would be even more difficult if the caches associated with each processor had inconsistent values of the shared data.

- Parallelism and Memory Hierarchy: Redundant Arrays of Inexpensive Disks. This section describes how using many disks in conjunction can offer much higher throughput, which was the original inspiration of Redundant Arrays of Inexpensive Disks (RAID). The real popularity of RAID proved to be the much greater dependability offered by including a modest number of redundant disks. The section explains the differences in performance, cost, and dependability between the various RAID levels

Real Stuff: Benchmarking the Intel Core i7

- workload A set of programs run on a computer that is either the actual collection of applications run by a user or constructed from real programs to approximate such a mix. A typical workload specifies both the programs and the relative frequencies

SPEC CPU Benchmark

- SPEC (System Performance Evaluation Cooperative) is an effort funded and supported by a number of computer vendors to create standard sets of benchmarks for modern computer systems.

SPEC speed 2017 Integer benchmarks running 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

Amdahl’s Law

- A rule stating that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used. It is a quantitative version of the law of diminishing returns.

- A simple design problem illustrates it well. Suppose a program runs in 100 seconds on a computer, with multiply operations responsible for 80 seconds of this time. How much do I have to improve the speed of multiplication if I want my program to run five times faster?
- Execution time after improvement

$$= \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}$$

For this problem:

$$\text{Execution time after improvement} = \frac{80\,\text{seconds}}{n} + (100 - 80\,\text{seconds})$$

Since we want the performance to be five times faster, the new execution time should be 20 seconds, giving

$$20\ \text{seconds} = \frac{80\ \text{seconds}}{n} + 20\ \text{seconds}$$

$$0 = \frac{80\ \text{seconds}}{n}$$

Computers at low utilization use little power

- Designing for performance and designing for energy efficiency are unrelated goals
- Since energy is power over time, it is often the case that hardware or software optimizations that take less time save energy overall even if the optimization takes a bit more energy when it is used. One reason is that all the rest of the computer is consuming energy while the program is running, so even if the optimized portion uses a little more energy, the reduced time can save the energy of the whole system.

- Pitfall: Using a subset of the performance equation as a performance metric
- We have already warned about the danger of predicting performance based on simply one of the clock rate, instruction count, or CPI. Another common mistake is to use only two of the three factors to compare performance

- million instructions per second (MIPS)
- A measurement of program execution speed based on the number of millions of instructions. MIPS is computed as the instruction count divided by the product of the execution time and 10^6 .

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

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