CHAPTER 1

Computer Abstractions and Technology

1.1 Introduction 3

- Modern computer technology requires professionals of every computing specialty to understand **both** hardware and software.

Classes of Computing Applications and Their Characteristics

- Personal computers
    - A computer designed for use by an individual, usually incorporating a graphics display, a keyboard, and a mouse.
    - Personal computers emphasize delivery of good performance to **single** users at low cost and usually execute third-party software.
    - This class of computing drove the evolution of many computing technologies, which is only about 35 years old!
- Server computers
    - 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.
- Supercomputers
    - A class of computers with the highest performance and cost
    - Supercomputers consist of tens of thousands of processors and many terabytes of memory, and cost tens to hundreds of millions of dollars.
    - Supercomputers are usually used for high-end scientific and engineering calculations
- Embedded computers
    - A computer **inside** another device used for running one predetermined application or collection of software.
    - Embedded computers include the microprocessors found in your car, the computers in a television set, and the networks of processors that control a modern airplane or cargo ship.
    - Embedded computing systems are designed to run one application or one set of related applications that are normally integrated with the hardware and delivered as a single system; thus, despite the large number of embedded computers, most users **never** really see that they are using a computer!
    - Elaboration: Many embedded processors are designed using processor cores, a version of a processor written in a hardware description language, such as **Verilog or VHDL**. The core allows a designer to integrate other application-specific hardware with the processor core for fabrication on a single chip.

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%

FIGURE 1.1 The 2^X vs. 10^Y bytes ambiguity was resolved by adding a binary notation for all the common size terms. These prefixes work for bits as well as bytes, so *gigabit* (Gb) is 10^9 bits while *gibibits* (Gib) is 2^{30} bits.

Welcome to the PostPC Era

- **Personal Mobile Device (PMD)**
  - Replacing the PC is the personal mobile device (PMD).
  - PMDs are battery operated with wireless connectivity to the Internet.
  - Figure 1.2 shows the rapid growth time of tables and smart phones versus that of PCs and tradition cell phones.

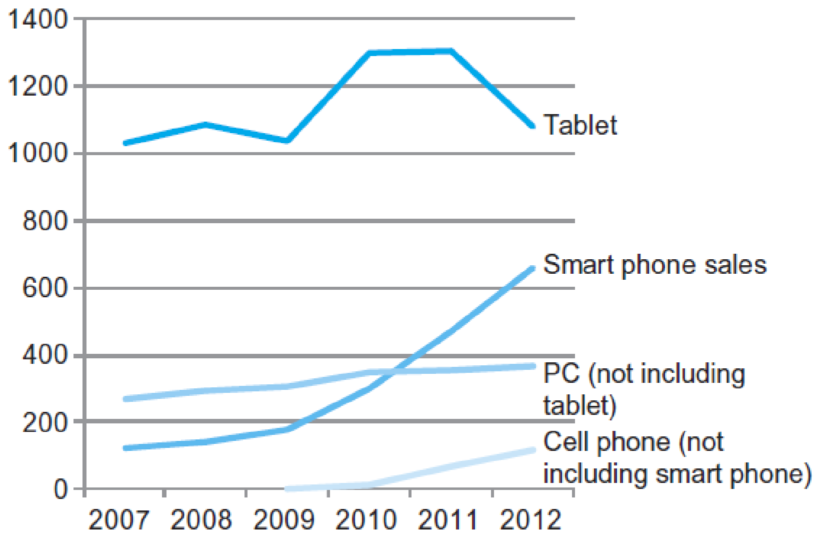


FIGURE 1.1 The number manufactured per year of tablets and smart phones, which reflect the PostPC era, versus personal computers and traditional cell phones. Smart phones represent the recent growth in the cell phone industry, and they passed PCs in 2011. Tablets are the fastest growing category, nearly doubling between 2011 and 2012. Recent PCs and traditional cell phone categories are relatively flat or declining.

- **Cloud Computing**
  - Warehouse Scale Computers (WSC): Companies like Amazon and Google build these WSCs containing 100,000 servers and then let companies rent portions of them so that they can provide software services to PMDs with having to build WSCs of their own.

    - Software as a Service (SaaS): It 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. SaaS deployed via the cloud is revolutionizing the software industry just PMDs and WSCs are revolutionizing the hardware industry.
- Today's software developers will often have a portion of software that runs on a PMD and a portion that run in the Cloud.

What You Can Learn in This Book

- How programs written in a high-level language, such as C or Java are **translated** into the machine language and how the hardware executes them?
- What is the interface between the software and the hardware, and how does software **instruct** the hardware to **perform** needed function?
- What determines the **performance** of a program, and how can a **programmer** improve the performance?
- What techniques can be used by hardware **designers** to improve performance?
- What is the reasons for and the consequences of the recent switch from sequential processing to **parallel processing**?

Understanding Program Performance

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

1.2 Eight Great Ideas in Computer Architecture 11

- We now introduce **eight** great ideas that computer architects have been invented in the last 60 years of computer design.
- These ideas are so powerful they have lasted long after the first computer that used them, with newer architects demonstrating their admiration by imitating their predecessors.

1. Design for **Moore's Law**
    - It states that integrated circuit resources **double** every **18-24** months.

2. Use **Abstraction** to Simplify Design
    - A major productivity technique for hardware and software is to use **abstractions** to represent the design at different levels of representation; lover-level details are **hidden** to offer a simpler model at higher levels.

3. 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 often easier to enhance.

4. Performance via **Parallelism**
    - Since the dawn of computing, computer architects have offered designs that get more performance by performing operations in **parallel**.

5. Performance via **Pipelining**

- A particular pattern of parallelism is so prevalent in computer architecture that it merits its own name: **pipelining**.

6. Performance via **Prediction**

- The next great idea is **prediction**. In some cases, it can be faster on average to **guess** and start working rather than wait until you know for sure.

7. Hierarchy of Memories

- Architects have found that they can address these **conflicting** demands with a **hierarchy of memories** with the fastest, smallest, and most expensive memory per bit at the **top** of the hierarchy and the slowest, largest, and cheapest per bit at the bottom.

8. 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 help detect failures. **Restoring** redundancy!

1.3 Below Your Program 13

- To go from a complex application to the simple 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**.
- Application software: written in high-level language
- System software sitting between the hardware and application software. Two type of systems software are central to every computer today: an operating system and a complier
    - Compiler: Translates HLL code to machine code
    - Operating System: Provide a variety of service and supervisory functions.
        - Handling input and output operations
        - Allocating storage and memory
        - Scheduling tasks & sharing resources: Providing for protected sharing of the computer among multiple applications using it simultaneously
- Hardware
    - Processor, memory, I/O controllers

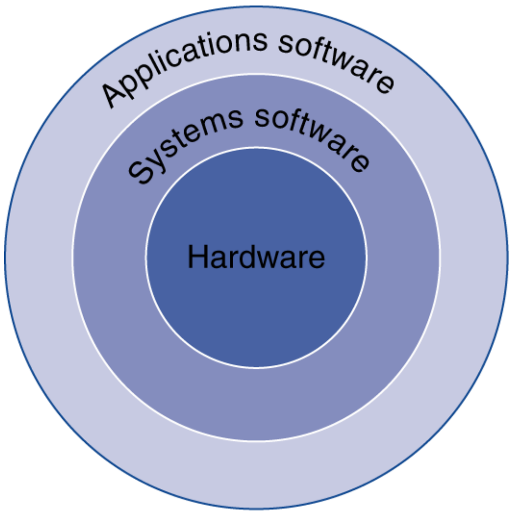
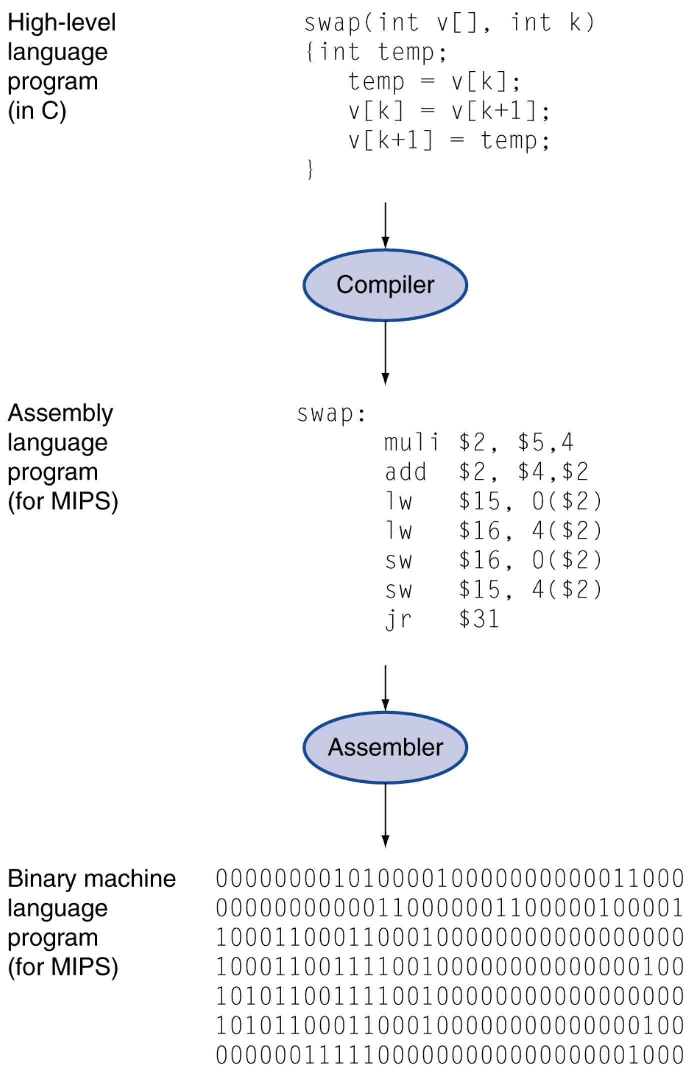


FIGURE 1.3 A simplified view of hardware and software as hierarchical layers, shown as concentric circles with hardware in the center and applications software outermost. In complex applications, there are often multiple layers of application software as well. For example, a database system may run on top of the systems software hosting an application, which in turn runs on top of the database.

From a High-Level Language to the Language of Hardware

- **High-level programming language**: A **portable** language such as C, C++, Java, or Visual Basic that is composed of words and algebraic nation that can be translated by a compiler into assembly language
- **Complier**: A program that **translates** high level language statements into assembly language statements.
- **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 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
```

FIGURE 1.4 C program compiled into assembly language and then assembled into binary machine language. Although the translation from high-level language to binary machine language is shown in two steps, some compilers cut out the middleman and produce binary machine language directly. These languages and this program are examined in more detail in Chapter 2.

1.4 Under the Covers 16

- The **five** classic components of a computer are **input**, **output**, **memory**, **datapath**, and **control**, with the last two (datapath and control) sometimes combined and called the processor.
    - **Input device**: A mechanism through which the computer is fed information, such as a keyboard.
    - **Output device**: A mechanism that conveys the result of a computation to a user, such as a display, or to anther computer such as network adapters.
    - **Memory**: The storage area in which program are kept when they are running and that contains the data needed by the running programs such as hard disk, CD/DVD, flash memory.
    - **Central processor unit (CPU)**: Also processor. The active part the computer, which contains the datapath and control and which add numbers, test number, signals I/O device to activate, and so on.
    - **Datapath**: The component of the processor that performs arithmetic operations
    - **Control**: The component of the processor tat commands the datapath memory, and, and I/O devices according to the instructions of the program

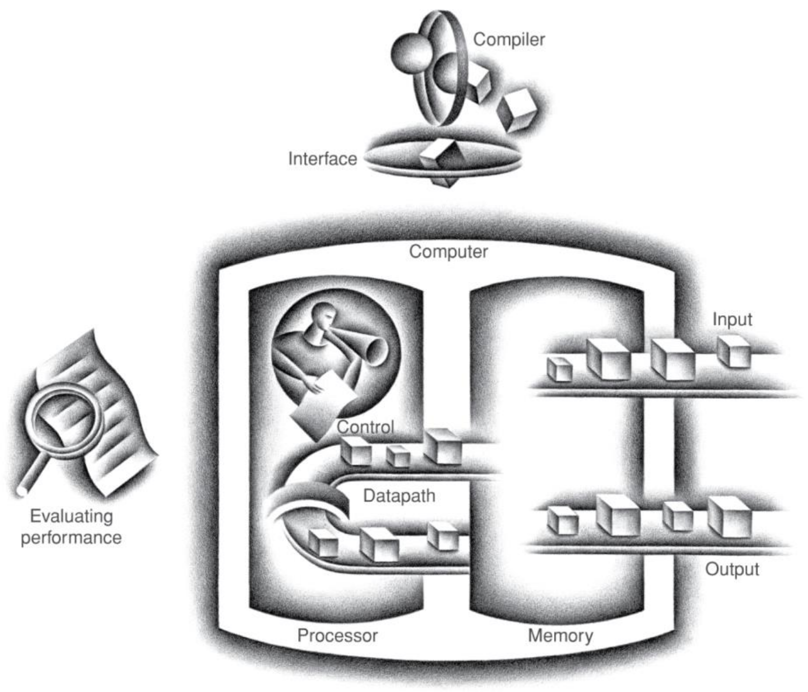


FIGURE 1.5 The organization of a computer, showing the five classic components. The processor gets instructions and data from memory. Input writes data to memory, and output reads data from memory. Control sends the signals that determine the operations of the datapath, memory, input, and output.

Through the Looking Glass

- LCD screen: A display technology using a thin layer of liquid polymers that can be used to transmit or block light according to whether a c charge is applied.
- **Pixel**: The smallest individual picture element. Screen are composed of hundreds of thousands to millions of pixels, organized in a matric.
- The image is composed of a matrix of picture elements, or **pixels**, which can be represented a matrix of bits, call a **bit map**. A typical tablet ranges in size from 1024 X 768 to 2048 X 1536.
- A color display might use 8 bits for each of the three colors (read, blue, and green), for 24 bits per pixel, permitting millions of different colors to be displayed. Figure 1.6 shows a fram buffer with a simplified design of just 4 bits per pixel.

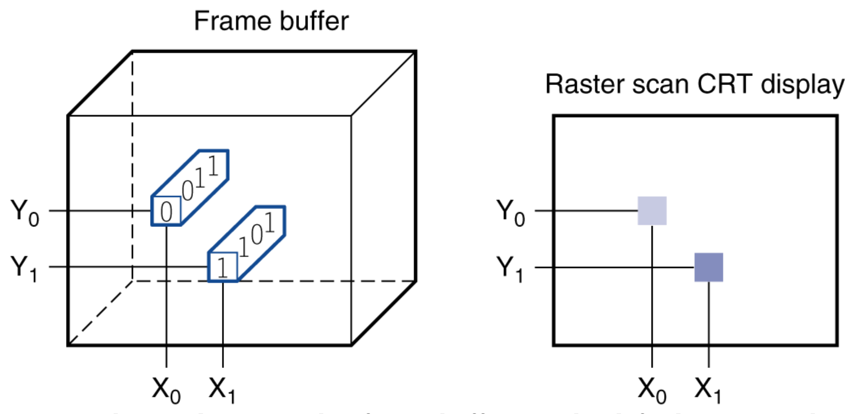


FIGURE 1.6 Each coordinate in the frame buffer on the left determines the shade of the corresponding coordinate for the raster scan CRT display on the right. Pixel (X_0, Y_0) contains the bit pattern 0011, which is a lighter shade on the screen than the bit pattern 1101 in pixel (X_1, Y_1).

Touchscreen

- While PCs also use LCD displays, the **tablets** and **smartphones** of the PostPC era have replaced the keyboard and mouse with touch sensitive display, which has the wonderful user interface advantage of user pointing directly what they are interested in rat that indirectly with a mouse.

Opening the Box

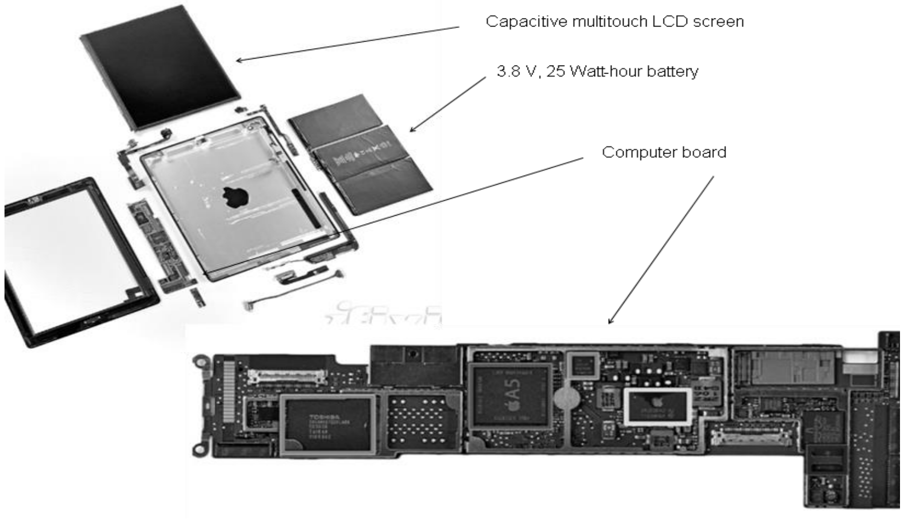


FIGURE 1.7 Components of the Apple iPad 2 A1395. The metal back of the iPad (with the reversed Apple logo in the middle) is in the center. At the top is the capacitive multitouch screen and LCD display. To the far right is the 3.8 V, 25 watt-hour, polymer battery, which consists of three Li-ion cell cases and offers 10 hours of battery life. To the far left is the metal frame that attaches the LCD to the back of the iPad. The small components surrounding the metal back in the center are what we think of as the computer; they are often L-shaped to fit compactly inside the case next to the battery. Figure 1.8 shows a close-up of the L-shaped board to the lower left of the metal case, which is the logic printed circuit board that contains the processor and the memory. The tiny rectangle below the logic board contains a chip that provides wireless communication: Wi-Fi, Bluetooth, and FM tuner. It fits into a small slot in the lower left corner of the logic board. Near the upper left corner of the case is another L-shaped component, which is a front-facing camera assembly that includes the camera, headphone jack, and microphone. Near the right upper corner of the case is the board containing the volume control and silent/screen rotation lock button along with a gyroscope and accelerometer. These last two chips combine to allow the iPad to recognize 6-axis motion. The tiny rectangle next to it is the rear-facing camera. Near the bottom right of the case is the L-shaped speaker assembly. The cable at the bottom is the connector between the logic board and the camera/volume control board. The board between the cable and the speaker assembly is the controller for the capacitive touchscreen. (Courtesy iFixit, www.ifixit.com)

- **Integrated circuit**: Also called a chip. A device combining dozens to millions of transistors.
- **Dynamic random access memory (DRAM):** Memory built as an integrated circuit; it provides random access to any location. Access times are50 nanoseconds and cost per gigabyte in 2012 was $5 to $10
- The memory is where the programs are kept when they are running; it also contains the data needed by running program. The memory is built form DRAM chips.
- Multiple DRAMs are used together to contain the instructions and data of a program.

FIGURE 1.8 The logic board of Apple iPad 2 in Figure 1.7. The photo highlights five integrated circuits. The large integrated circuit in the middle is the **Apple A5 chip**, which contains a dual ARM processor cores that run at 1 GHz as well as 512 MB of main memory inside the package. Figure 1.9 shows a photograph of the processor chip inside the A5 package. The similar sized chip to the left is the 32 GB flash memory chip for non-volatile storage. There is an empty space between the two chips where a second flash chip can be installed to double storage capacity of the iPad. The chips to the right of the A5 include power controller and I/O controller chips. (Courtesy iFixit, www.ifixit.com)

Inside the Processor

- Datapath: performs operations on data
- Control: sequences datapath, memory, ...
- Cache memory: Consists of a small, fast memory that acts as a buffer for the DRAM memory.
    - **Static random access memory (SRAM)**: Also memory built as an integrated circuit, but faster and less dense than DRAM

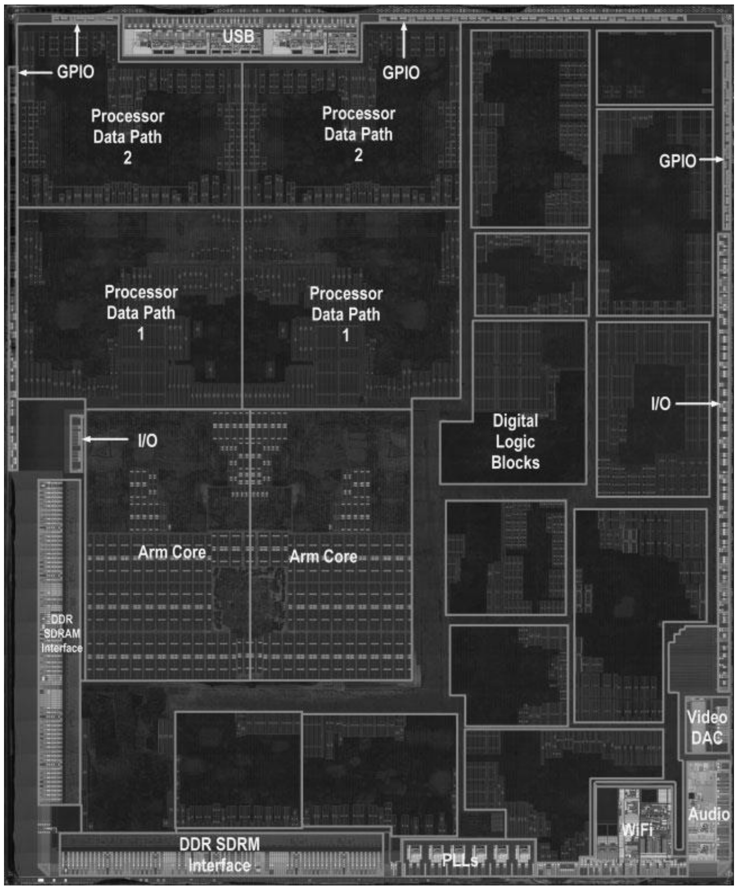


FIGURE 1.9 The processor integrated circuit inside the A5 package. The size of chip is 12.1 by 10.1 mm, and it was manufactured originally in a 45-nm process (see Section 1.5). It has two identical ARM processors or cores in the middle left of the chip and a PowerVR graphical processor unit (GPU) with four datapaths in the upper left quadrant. To the left and bottom side of the ARM cores are interfaces to main memory (DRAM). (Courtesy Chipworks, www.chipworks.com)

A Safe Place for Data

- **Volatile memory**: Storage, such as DRAM that retains data only if it is receiving power.
    - Volatile main memory (primary memory): loses instructions and data when power off.
- **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.
    - Non-volatile secondary memory:
        - Magnetic disk
        - Flash memory
        - Optical disk (CDROM, DVD)

Communicating with Other Computers

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

1.5 Technologies for Building Processors and Memory 24

- Electronics technology continues to evolve
  - Increased capacity and performance
  - Reduced cost
- A **transistor** is simply an on/off switch controlled by electricity.
- The integrated circuit (**IC**) combined dozens to hundreds of transistors into a simple chip.
- Very large-scale integrated (**VLSI**) circuit: A device containing hundreds of thousands to millions of transistors.

Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2013	Ultra large scale IC	250,000,000,000

FIGURE 1.10 Relative performance per unit cost of technologies used in computers over time. Source: Computer Museum, Boston, with 2013 extrapolated by the authors.

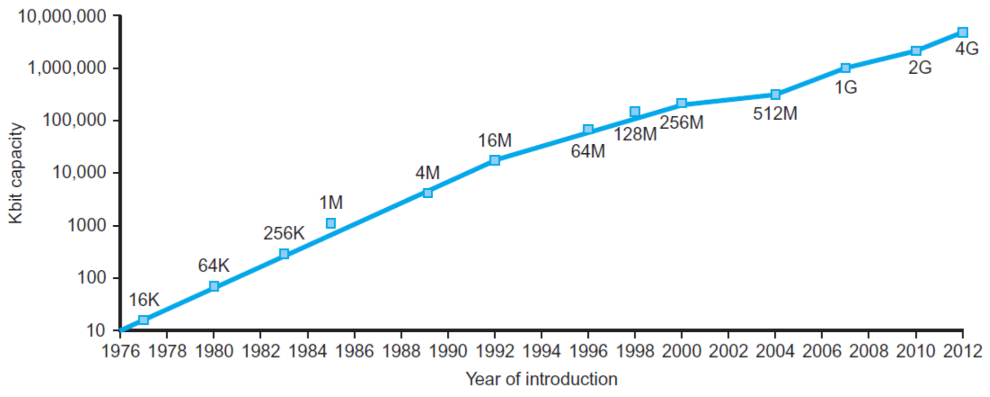


FIGURE 1.11 Growth of capacity per **DRAM** chip over time. The y-axis is measured in kibibits (210 bits). The DRAM industry quadrupled capacity almost every three years, a 60% increase per year, for 20 years. In recent years, the rate has slowed down and is somewhat closer to doubling every two years to three years.

Manufacturing ICs

- Silicon crystal igot: A rod composed of a silicon crystal that is between 8 and 12 inches in diameter and about 12 to24 inches long
- Wafer: A slice from a silicon igot no more than 0.1 inches thick, used to create chips.

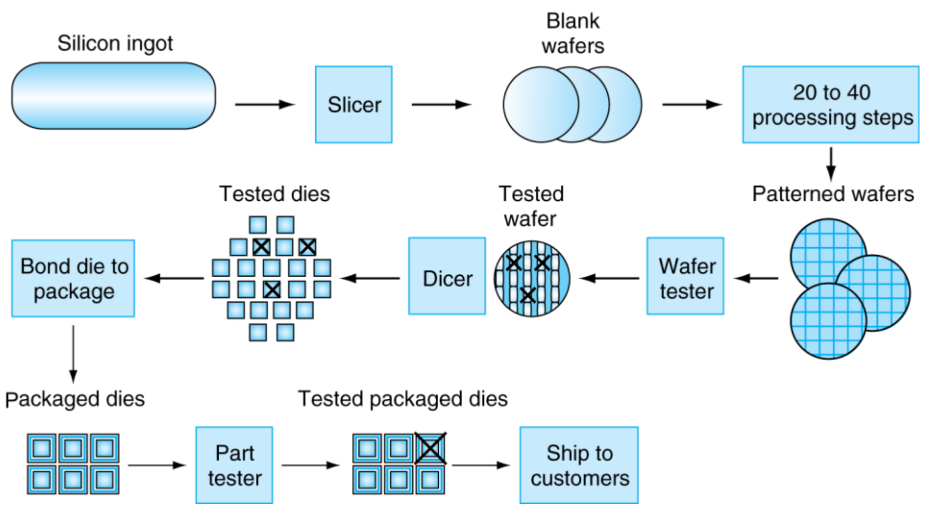


FIGURE 1.12 The chip manufacturing process. After being sliced from the silicon ingot, blank wafers are put through 20 to 40 steps to create patterned wafers (see Figure 1.13). These patterned wafers are then tested with a wafer tester, and a map of the good parts is made. Then, the wafers are diced into dies (see Figure 1.9). In this figure, one wafer produced 20 dies, of which 17 passed testing. (X means the die is bad.) The yield of good dies in this case was 17/20, or 85%. These good dies are then bonded into packages and tested one more time before shipping the packaged parts to customers. One bad packaged part was found in this final test

- **Intel Core i7 Wafer**
  - 300mm wafer, **280 chips**, 32nm technology
  - Each chip is 20.7 x 10.5 mm

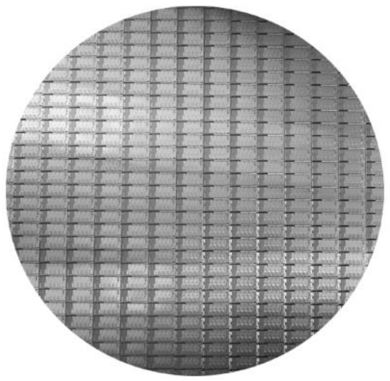


FIGURE 1.13 A 12-inch (300 mm) **wafer of Intel Core i7** (Courtesy Intel). The number of dies on this 300 mm (12 inch) wafer at 100% yield is 280, each 20.7 by 10.5 mm. The several dozen partially rounded chips at the boundaries of the wafer are useless; they are included because it's easier to create the masks used to pattern the silicon. This die uses a 32-nanometer technology, which means that the smallest features are approximately 32 nm in size, although they are typically somewhat smaller than the actual feature size, which refers to the size of the transistors as "drawn" versus the final manufactured size.

1.6 Performance 28

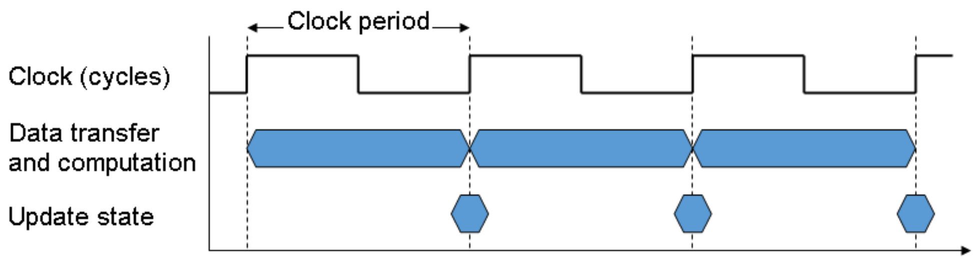
- Response Time and Throughput
  - **Response time**: Also called execution time for the computer to **complete a task**, including disk access, memory access, I/O activities, operating system overhead, CPU execution time, and so on. How long it takes to do a task
  - **Throughput**: Also called bandwidth. Another measure of performance. It is the **number of tasks** completed per unit time. Total work done per unit time
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- Relative Performance
  - Define **Performance = 1/Execution Time**
  - If X is n time fast as Y, then the execution time on Y is n times as long as it is on X.

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

  - Example: 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?

    Execution Time_B / Execution Time_A = 15s / 10s = 1.5
    So A is **1.5** times faster than B

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

CPU Performance and its Factors

- CPU Time

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

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count
- CPU Time Example: 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 with 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 time as many clock cycles as computer A for this program. What clack rate should we tell the designer to target?
  - Computer A: 2 GHz clock, 10sec CPU time
  - Designing Computer B
    - Aim for 6sec CPU time
    - Can do faster clock, but causes 1.2 × clock cycles
  - How fast must Computer B clock be?

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

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

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

  - **Answer: 4GHz**

Instruction Performance

- Instruction Count and CPI

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

- **Clock cycles per instruction (CPI)**: Average number of clock cycles per instruction for a program or program fragment.
  - Instruction Count for a program
    - Determined by program, ISA and compiler
  - Average cycles per instruction
    - Determined by CPU hardware
    - If different instructions have different CPI
    - Average CPI affected by instruction mix

- CPI Example: 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?
  - Computer A: Cycle Time = 250ps, CPI = 2.0
  - Computer B: Cycle Time = 500ps, CPI = 1.2
  - Same instruction set architecture (ISA)
  - Which is faster, and by how much?

$$\text{CPU Time}_A = \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A$$
$$= \text{I} \times 2.0 \times 250\text{ps} = \text{I} \times 500\text{ps}$$
$$\text{CPU Time}_B = \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B$$
$$= \text{I} \times 1.2 \times 500\text{ps} = \text{I} \times 600\text{ps}$$
$$\frac{\text{CPU Time}_B}{\text{CPU Time}_A} = \frac{\text{I} \times 600\text{ps}}{\text{I} \times 500\text{ps}} = 1.2$$

  - **Answer: 1.2**

The Classic CPU Performance Equation

- 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

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

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

Class	A	B	C
CPI for class	1	2	3

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

Class	A	B	C
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

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

  - Alternative compiled code sequences using instructions in classes A, B, C
  - Sequence 1:
    - IC = 2 + 1 + 2 = 5 inst.
    - Clock Cycles = 2×1 + 1×2 + 2×3 = 10
    - Avg. CPI = 10 / 5 = 2.0
  - Sequence 2:
    - IC = 4 + 1 + 1 = 6 inst.
    - Clock Cycles = 4×1 + 1×2 + 1×3 = 9
    - Avg. CPI = 9 / 6 = 1.5

Performance Summary

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

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

1.7 The Power Wall 40

- Clock rate and Power

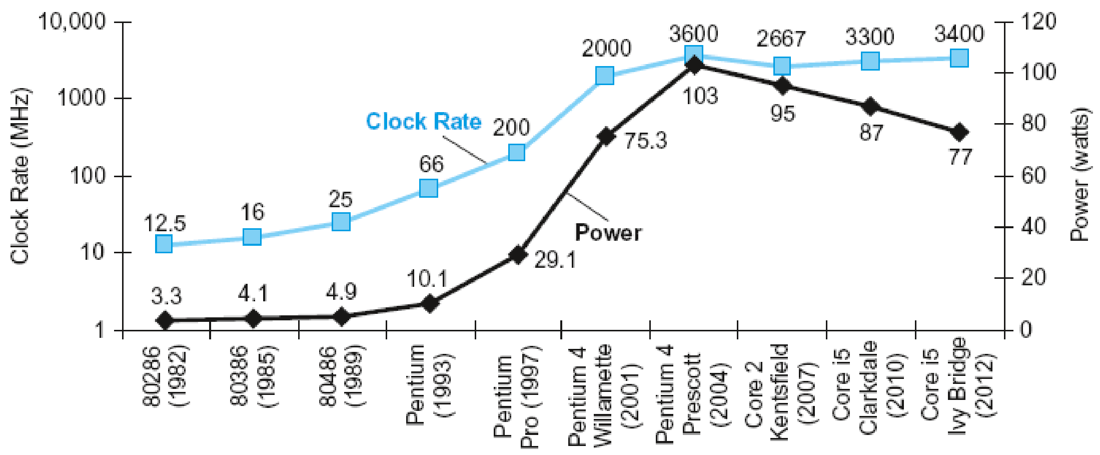


FIGURE 1.16 **Clock rate and Power** for Intel x86 microprocessors over eight generations and 25 years. The Pentium 4 made a dramatic jump in clock rate and power but less so in performance. The Prescott thermal problems led to the abandonment of the Pentium 4 line. The Core 2 line reverts to a simpler pipeline with lower clock rates and multiple processors per chip. The Core i5 pipelines follow in its footsteps.

- In CMOS (complementary metal oxide semiconductor) IC technology

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

- Reducing Power Example: 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 15% shrink in frequency. What is the impact on dynamic power?

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

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

  - **Answer:** the new processor uses about half (**0.52)** the power of the old processor

1.8 The Sea Change: The Switch from Uniprocessors to Multiprocessors 43

- Uniprocessor: A single program running on the single processor
- Multicore microprocessors
  - More than one processor per chip
  - A "quadcore" microprocessor is a chip that contains four processor or four cores.

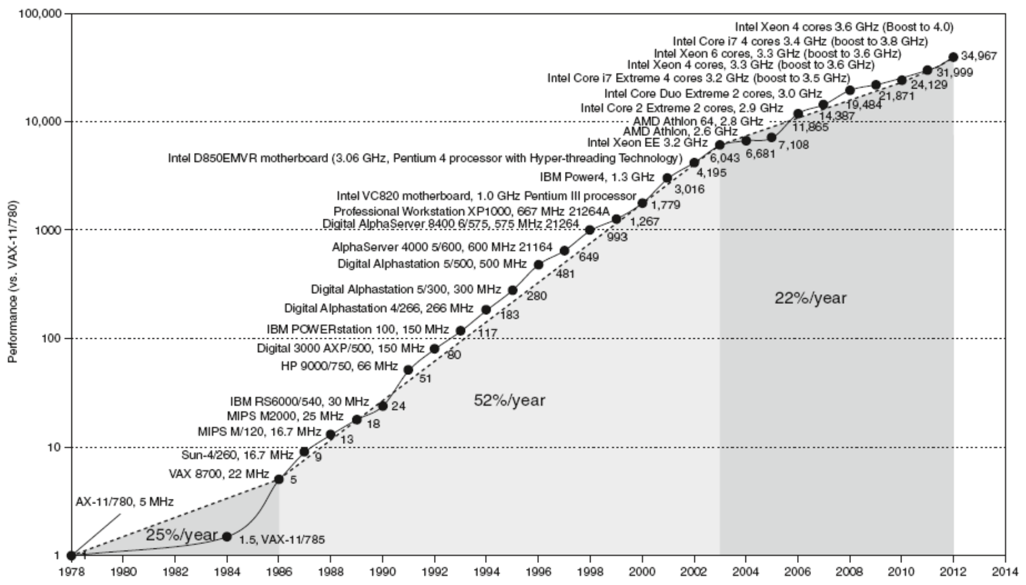
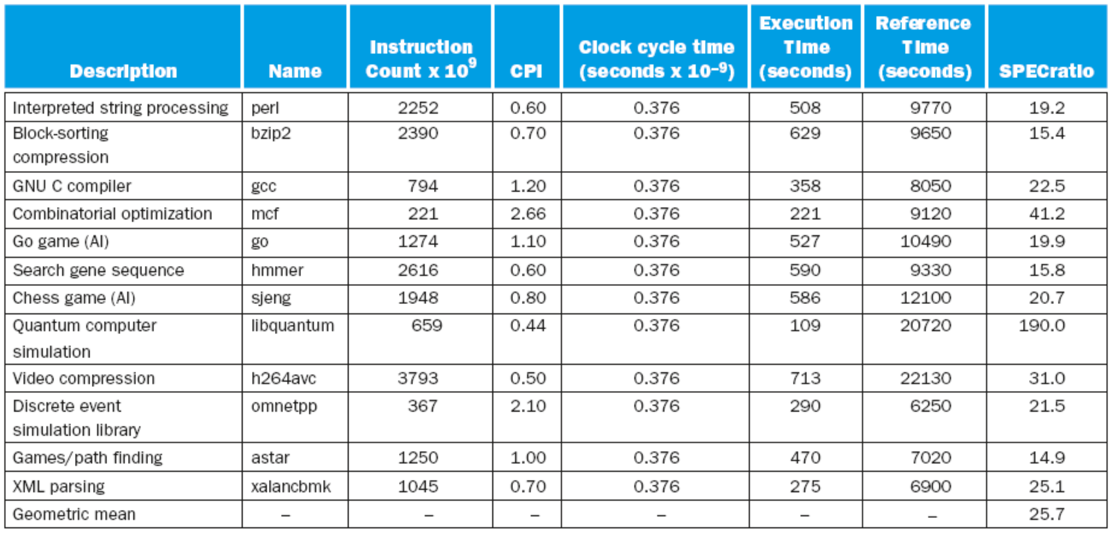


FIGURE 1.17 Growth in processor performance since the mid-1980s. This chart plots performance relative to the VAX 11/780 as measured by the SPECint benchmarks (see Section 1.10). Prior to the mid-1980s, processor performance growth was largely technology-driven and averaged about 25% per year. The increase 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.

1.9 Real Stuff: Benchmarking the Intel Core i7 46

SPEC CPU Benchmark

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



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

FIGURE 1.18 SPECINTC2006 benchmarks running on a 2.66 GHz Intel Core i7 920. As the equation on page 35 explains, 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.

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

- Intel Core i7 Geometric mean is **25.7**

1.11 Concluding Remarks 52

- **Eight** Great Ideas in Computer Architecture
  1. Design for **Moore's Law**
  2. Use **Abstraction** to Simplify Design
  3. Make the **Common Case Fast**
  4. Performance via **Parallelism**
  5. Performance via **Pipelining**
  6. Performance via **Prediction**
  7. **Hierarchy of Memories**
  8. Dependability via **Redundancy**
- The **execution time** is related to other important measurements we can make by the following equation:

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