Introduction

The phrase *computer science* has a very broad meaning today. However,this introduction first tries to find out what a computer is, then investigates other issues directly related to computers. We look first at the **Turing model** as a mathematical and philosophical definition of computation. We then show how today's computers are based on the **von Neumann model**.

Objectives

After studying this introduction, the student should be able to:

- Define the Turing model of a computer.
- Define the von Neumann model of a computer.
- Describe the three components of a computer: hardware, data, and software.
- List topics related to computer hardware.
- List topics related to data.
- List topics related to software.
- Give a short history of computers.

1.1 TURING MODEL

The idea of a universal computational device was first described by Alan Turing in 1936. He proposed that all computation could be performed by a special kind of a machine, now called a **Turing machine**. Although Turing presented a mathematical description of such a machine, he was more interested in the philosophical definition of computation than in building the actual machine. He based the model on the actions that people perform when involved in computation. He abstracted these actions into a model for a computational machine that has really changed the world.

1.1.1 Data processors

Before discussing the Turing model, let us define a computer as a **data processor**. Using this definition, a computer acts as a black box that accepts input data, processes the data, and creates output data (Figure 1.1). Although this model can define the functionality of a computer today, it is too general. In this model, a pocket calculator is also a computer (which it is, in a literal sense).

Figure 1.1 *A single-purpose computing machine*

Another problem with this model is that it does not specify the type of processing, or whether more than one type of processing is possible. In other words, it is not clear how many types or sets of operations a machine based on this model can perform. Is it a specific-purpose machine or a general-purpose machine?

This model could represent a specific-purpose computer (or processor) that is designed to do a single job, such as controlling the temperature of a building or controlling the fuel usage in a car. However, computers, as the term is used today, are *general-purpose* machines. They can do many different types of tasks. This implies that we need to change this model into the Turing model to be able to reflect the actual computers of today.

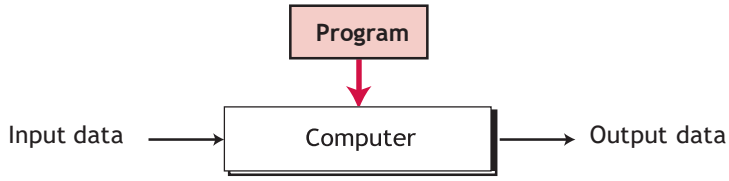
1.1.2 Programmable data processors

The Turing model is a better model for a general-purpose computer. This model adds an extra element to the specific computing machine: the *program*. A **program** is a set of instructions that tells the computer what to do with data. Figure 1.2 shows the Turing model.

In the Turing model, the **output data** depends on the combination of two

factors: the **input data** and the program. With the same input data, we can generate different output if we change the program. Similarly, with the same program, we can generate different

Figure 1.2 *A computer based on the Turing model: programmable data processor*

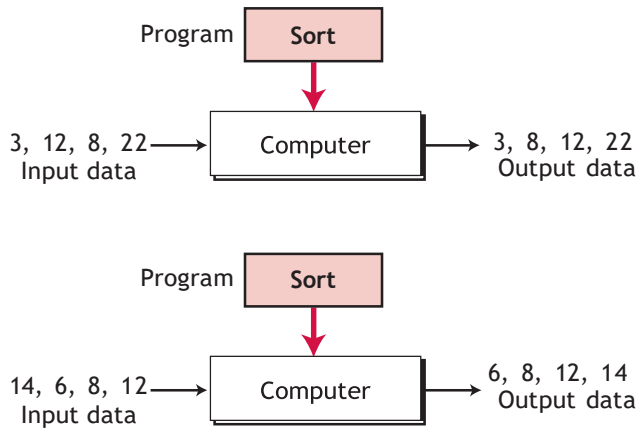


outputs if we change the input data. Finally, if the input data and the program remain the same, the output should be the same. Let us look at three cases.

Same program, different input data

Figure 1.3 shows the same sorting program with different input data. Although the program is the same, the outputs are different, because different input data is processed.

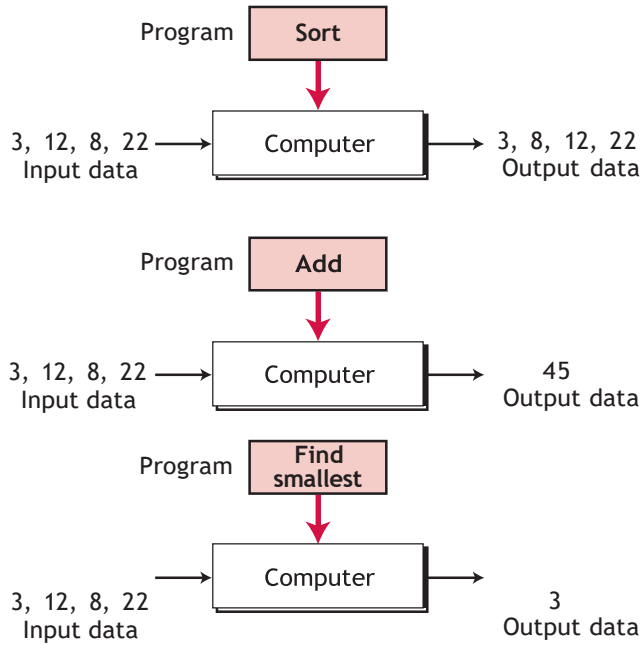
Figure 1.3 *The same program, different data*



Same input data, different programs

Figure 1.4 shows the same input data with different programs. Each program makes the computer perform different operations on the input data. The first program sorts the data, the second adds the data, and the third finds the smallest number.

Figure 1.4 *The same data, different programs*



Same input data, same program

We expect the same result each time if both input data and the program are the same, of course. In other words, when the same program is run with the same input data, weexpect the same output.

1.1.3 The universal Turing machine

A *universal Turing machine*, a machine that can do any computation if the appropriate program is provided, was the first description of a modern computer. It can be proved that a very powerful computer and a universal Turing machine can compute the same thing. We need only provide the data and the program—the description of how to do the computation—to either machine. In fact, a universal Turing machine is capable of computing anything that is computable.

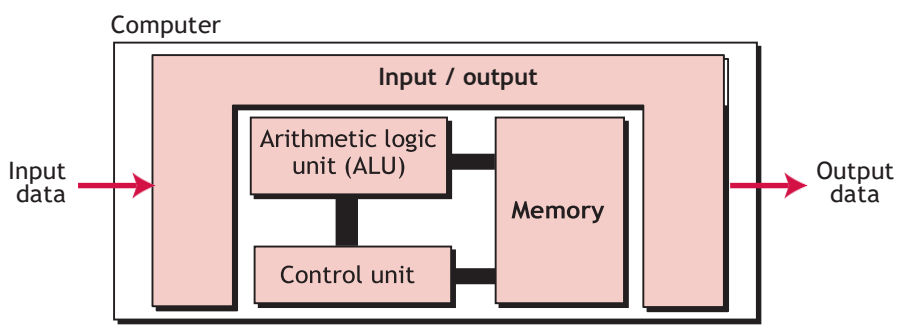
1.1 VON NEUMANN MODEL

Computers built on the Turing universal machine store data in their memory. Around 1944–1945, John von Neumann proposed that, since program and data are logically the same, programs should also be stored in the memory of a computer.

1.1.1 Four subsystems

Computers built on the von Neumann model divide the computer hardware into four subsystems: memory, arithmetic logic unit, control unit, and input/output (Figure 1.5).

Figure 1.5 *The Von Neumann model*



Memory

Memory is the storage area. This is where programs and data are stored during processing.

Arithmetic logic unit

The **arithmetic logic unit (ALU)** is where calculation and **logical operations** take place. For a computer to act as a data processor, it must be able to do arithmetic operations on data (such as adding a list of numbers). It should also be able to do logical operations on data.

Control unit

The **control unit** controls the operations of the **memory**, ALU, and the input/output subsystem.

Input / output

The **input subsystem** accepts input data and the program from outside the computer, while the **output subsystem** sends the result of processing to the outside world. The definition of the input/output subsystem is very broad: it also includes secondary storage devices such as disk or tape that stores data and programs for processing. When a disk stores data that results from processing, it is considered an output device: when it reads data from the disk, it is considered an input device.

1.1.2 The stored program concept

The von Neumann model states that the program must be stored in memory. This istotally different from the architecture of early computers in which only the data was stored in memory: the programs for their task were implemented by manipulating a set ofswitches or by changing the wiring system.

The memory of modern computers hosts both a program and its corresponding data. This implies that both the data and programs should have the same format, because they are stored in memory. In fact, they are stored as *binary* patterns in memory—a sequence of 0s and 1s.

1.1.3 Sequential execution of instructions

A program in the von Neumann model is made of a finite number of **instructions**. In this model, the control unit fetches one instruction from memory, decodes it, then executesit. In other words, the instructions are executed one after another. Of course, one instruction may request the control unit to jump to some previous or following instruction, but this does not mean that the instructions are not executed sequentially. Sequential execution of a program was the initial requirement of a computer based onthe von Neumann model. Today's computers execute programs in the order that is the most efficient.

1.2 COMPUTER COMPONENTS

We can think of a computer as being made up of three components: computer hardware, data, and computer software.

1.2.1 Computer hardware

Computer hardware today has four components under the von Neumann model, although we can have different types of memory, different types of input/output subsystems, andso on. We discuss computer hardware in more detail in Chapter 5.

1.2.2 Data

The von Neumann model clearly defines a computer as a data processing machine that accepts the input data, processes it, and outputs the result.

Storing data

The von Neumann model does not define how data must be stored in a computer. If a computer is an electronic device, the best way to store data is in the form of an electrical signal, specifically its presence or absence. This implies that a computer can store data in one of two states.

Obviously, the data we use in daily life is not just in one of two states. For example, our numbering system uses digits that can take one of ten states (0 to 9). We cannot (as yet) store this type of information in a computer: it needs to be changed to another systemthat uses only two states (0 and 1). We also need to be able to process other types of data(text, image, audio, video). These also cannot be stored in a computer directly, but need to be changed to the appropriate form (0s and 1s).

Organizing data

Although data should be stored only in one form inside a computer, a binary pattern, data outside a computer can take many forms. In addition, computers (and the notion of data processing) have created a new field of study known as *data organization*, which asks the question: can we organize our data into different entities and formats before storing them inside a computer? Today, data is not treated as a flat sequence of information. Instead, data is organized into small units, small units are organized into larger units, and so on.

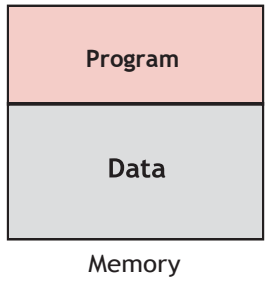
1.2.3 Computer software

The main feature of the Turing or von Neumann models is the concept of the *program*. Although early computers did not store the program in the computer's memory, they did use the concept of programs. *Programming* those early computers meant changing the wiring systems or turning a set of switches on or off. Programming was therefore a task done by an operator or engineer before the actual data processing began.

Programs must be stored

In the von Neumann model programs are stored in the computer's memory. Not only do we need memory to hold data, but we also need memory to hold the program (Figure 1.6).

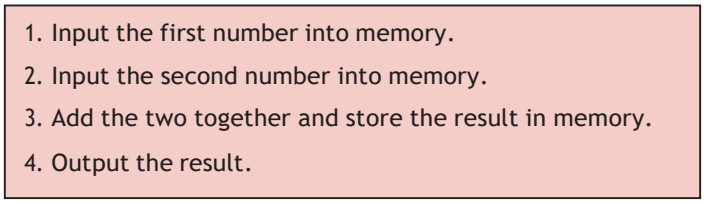
Figure 1.6 *Program and data in memory*



A sequence of instructions

Another requirement of the model is that the program must consist of a sequence of instructions. Each instruction operates on one or more data items. Thus, an instruction can change the effect of a previous instruction. For example, Figure 1.7 shows a program that inputs two numbers, adds them, and prints the result. This program consists of four individual instructions.

Figure 1.7 *A program made of instructions*



We might ask why a program must be composed of instructions. The answer is reusability. Today, computers do millions of tasks. If the program for each task was an independent entity without anything in common with other programs, programming would be difficult. The Turing and von Neumann models make programming easier by defining the different instructions that can be used by computers. A programmer can then combine these instructions to make any number of programs. Each program can be a different combination of different instructions.

Algorithms

The requirement for a program to consist of a sequence of instructions made programmingpossible, but it brought another dimension to using a computer. A programmer must not only learn the task performed by each instruction but also learn how to combine these instructions to do a particular task. Looking at this issue differently, a programmer must first solve the problem in a step-by-step manner, then try to find the appropriate instruction (or series of instructions) to implement those steps. This step-by-step solution is called an **algorithm**. Algorithms play a very important role in computer science.

Languages

At the beginning of the computer age there was only one computer language, *machine language*. Programmers wrote instructions (using binary patterns) to solve a problem. However, as programs became larger, writing long programs using these patterns became tedious. Computer scientists came up with the idea of using symbols to represent binary patterns, just as people use symbols (words) for commands in daily life. Of course, the symbols used in daily life are different from those used in computers. So the concept of **computer**

languages was born. A natural language such as English is rich and has many rules to combine words correctly: a computer language, on the other hand, has a more limited number of symbols and also a limited number of words.

Software engineering

Something that was not defined in the von Neumann model is **software engineering**, which is the design and writing of **structured programs**. Today it is not acceptable just to write a program that does a task: the program must follow strict rules and principles.

Operating systems

During the evolution of computers, scientists became aware that there was a series of instructions common to all programs. For example, instructions to tell a computer where to receive data and where to send data are needed by almost all programs. It is more efficient to write these instructions only once for the use of all programs. Thus the conceptof the **operating system** emerged. An operating system originally worked as a manager to facilitate access to the computer's components by a program, although today operating systems do much more.

1.4 HISTORY

In this section we briefly review the history of computing and computers. We divide this history into three periods.

1.4.1 Mechanical machines (before 1930)

During this period, several computing machines were invented that bear little resemblance to the modern concept of a computer.

- In the seventeenth century, Blaise Pascal, a French mathematician, and philosopher, invented Pascaline, a mechanical calculator for addition and subtraction operations.In the twentieth century, when Niklaus Wirth invented a structured programming language, he called it Pascal to honor the inventor of the first mechanical calculator.
- In the late seventeenth century, German mathematician Gottfried Leibniz invented a more sophisticated mechanical calculator that could do multiplication and division as well as addition and subtraction. It was called the Leibniz Wheel.
- The first machine that used the idea of storage and programming was the Jacquard loom, invented by Joseph-Marie Jacquard at the beginning of the nineteenth century. The loom used punched cards

(like a stored program) to control the raising of the warp threads in the manufacture of textiles.

- In 1823, Charles Babbage invented the Difference Engine, which could do more than simple arithmetic operations—it could solve polynomial equations, too. Later, he in- vented a machine called the Analytical Engine that, to some extent, parallels the idea of modern computers. It had four components: a mill (corresponding to a modern ALU), a store (memory), an operator (control unit), and output (input/output).
- In 1890, Herman Hollerith, working at the US Census Bureau, designed and builta programmer machine that could automatically read, tally, and sort data stored on punched cards.

1.4.2 The birth of electronic computers (1930-1950)

Between 1930 and 1950, several computers were invented by scientists who could be considered the pioneers of the electronic computer industry.

Early electronic computers

The early computers of this period did not store the program in memory—all were programmed externally. Five computers were prominent during these years:

- The first special-purpose computer that encoded information electrically was invented by John V. Atanasoff and his assistant Clifford Berry in 1939. It was called the ABC (Atana- soff Berry Computer) and was specifically designed to solve a system of linear equations.
- At the same time, a German mathematician called Konrad Zuse designed a general- purpose machine called Z1.
- In the 1930s, the US Navy and IBM sponsored a project at Harvard University under the direction of Howard Aiken to build a huge computer called Mark I. This comput-er used both electrical and mechanical components.
- In England, Alan Turing invented a computer called Colossus that was designed to break the German Enigma code.
- The first general-purpose, totally electronic computer was made by John Mauchly and J. Presper Eckert and was called ENIAC (Electronic Numerical Integrator and Calcu- lator). It was completed in 1946. It used 18 000 vacuum tubes, was 100 feet long by 10 feet high, and weighed 30 tons.

Computers based on the von Neumann model

The preceding five computers used memory only for storing data and were programmed externally using wires or switches. John von Neumann proposed that the program and the data should be stored in memory. That way, every time we use a computer to do a new task, we need only change the program instead of rewiring the machine or turning hundreds of switches on and off.

The first computer based on von Neumann's ideas was made in 1950 at the University of Pennsylvania and was called EDVAC. At the same time, a similar computer called EDSAC was built by Maurice Wilkes at Cambridge University in England.

1.4.3 Computer generations (1950-present)

Computers built after 1950 more or less follow the von Neumann model. They have become faster, smaller, and cheaper, but the principle is almost the same. Historians divide this period into generations, with each generation witnessing some major change in hardware or software (but not in the model).

First generation

The first generation (roughly 1950–1959) is characterized by the emergence of commercial computers. During this time, computers were used only by professionals. They were locked in rooms with access limited only to the operator or computer specialist. Computers were bulky and used vacuum tubes as electronic switches. At this time, computers were affordable only by big organizations.

Second generation

Second-generation computers (roughly 1959–1965) used transistors instead of vacuum tubes. This reduced the size of computers, as well as their cost, and made them affordable to small and medium-size corporations. Two high-level programming languages, FORTRAN and COBOL, were invented and made programming easier. These two languages separated the programming task from the computer operation task. A civil engineer, for example could write a FORTRAN program to solve a problem without being involved in the electronic details of computer architecture.

Third generation

The invention of the **integrated circuit** (transistors, wiring, and other components on a single chip) reduced the cost and size of computers even further. *Minicomputers* appeared on the market. Canned programs, popularly known as software packages, became available. A small corporation could buy a package, for example for accounting, instead

of writingits own program. A new industry, the software industry, was born. This generation lasted roughly from 1965 to 1975.

Fourth generation

The fourth generation (approximately 1975–1985) saw the appearance of **microcomputers**. The first desktop calculator, the Altair 8800, became available in 1975. Advances in the electronics industry allowed whole computer subsystems to fit on a single circuit board. This generation also saw the emergence of computer networks .

Fifth generation

This open-ended generation started in 1985. It has witnessed the appearance of laptop and palmtop computers, improvements in secondary storage media (CD-ROM, DVD, and so on), the use of multimedia, and the phenomenon of virtual reality.

COMPUTER SCIENCE AS A DISCIPLINE

With the invention of computers, a new discipline has evolved: *computer science*. Like any other discipline, computer science has now divided into several areas. We can divide these areas into two broad categories: *systems areas* and *applications areas*. Systems areas cover those areas that directly related to the creation of hardware and software, such as *computer architecture*, *computer networking*, *security issues*, *operating systems*, *algorithms*, *programming languages*, and *software engineering*. Applications areas cover those that are related to the *use* of computers, such as *databases* and *artificial intelligence*.