2

Introduction to Computer Systems

“Don't Panic.”

Douglas Adams, *The Hitch Hiker's Guide to the Galaxy*

Aims

The aims of this chapter are to introduce the following:

- The components of a computer — the hardware.
- The components of a complete computer system — the other devices that you need to do useful work with a computer.
- The software needed to make the hardware do what you want it do.

2 Introduction to Computer Systems

A computer is an electronic device and can be thought of as a tool like a lever or a wheel, which can be made to do useful work. At the fundamental level it works with *bits* (binary digits or sequences of zeros and ones). Bits are generally put together in larger configurations, e.g., 8, 16, 32, or 64. Hence computers are often referred to as 8-bit, 16-bit, 32-bit, or 64-bit machines.

2.1 The core of a computer system

The heart of most computer systems comprises a motherboard, CPU, memory, one or more busses and a power supply. We will look at the CPU, memory and bus in more depth below.

2.1.1 Central processor unit — CPU

This is the brains of the computer. All of the work that the computer does is organised here.

2.1.2 Memory

The computer also has a memory. Memory on a computer is a solid state device that comprises an ordered collection of bits/bytes/words that can be read or written by the CPU. A byte is generally 8 bits (as in *8-bit byte*), and a word is most commonly accepted as the minimum number of bits that can be referenced by the CPU. This referencing is called *addressing*. The memory typically contains programs and data.

The two most common word sizes are 32 and 64 bits.

A computer memory is often called random access memory, or RAM. This simply means that the access time for any part of the memory is the same; in order to examine location (say) 97, it is not necessary to first look through locations 1 to 96. It is possible to go directly to location 97. A slightly better term might have been *access at random*. The memory itself is highly ordered.

2.1.3 Bus

A *bus* is a set of connections between the CPU and other components. The bus is used for a variety of purposes. These include address signals, which tell the memory which words are wanted next and data lines, which are used to transfer data to and from memory and to and from other parts of the computer system. This is typical of many systems, but systems do vary considerably; so while the information above may not be true in specific cases, it provides a general model.

2.2 Other components of a computer system

So far the computer we have described is not sufficiently versatile. We have to add on other pieces of electronics to make it really useful.

2.2.1 Disks

These are devices for storing collections of *bits*, which are inevitably organised in reality into bytes and files. One advantage of adding these to our computer system is that we can switch the machine off, go away and come back at a later time and continue with what we were doing.

Memory is expensive and fast, whereas disks are slower but cheaper. Most computer systems balance speed against cost, and have a small memory in relation to disk capacity.

Many people will be familiar with the two main types of disks on early personal computers (PCs) or microcomputers: floppy disks and hard disks. Floppy disks now come in one main physical size, 3-1/2 inch, but smaller ones are also used. Hard disks are inside the system, and most people do not see them.

Optical drives are an essential part of present day systems. They exist in a variety of flavours including simple read-only CD, rewritable CD and DVD forms.

2.2.2 Others

There are a large number of other input and output devices. These vary considerably from system to system, depending on the work being carried out. They include:

- Network (ethernet or wifi) cards for access to local and wide area networks.
- Modems for access from home, mainly to the Internet.
- Printers of a variety of types.
- Colour plotters.
- Phototypesetters.
- Pens.
- Sound interfaces, both for speech recognition and sound production.
- Scanners.
- Digital cameras.
- Joy sticks.
- Zip drives.

- Memory sticks.

The most important I/O devices are the keyboard and the screen, whether you use a terminal, PC or workstation. This book has been written assuming that most of your work will be done at one of these devices.

Terminals fall into two categories, character-based devices (and the DEC VT series is a very popular one) and graphical devices (the X-Windows terminals are the most popular). Terminal access to remote systems is often provided on PCs using terminal emulation software, e.g., Telnet, WinQVT and X-Windows access to UNIX systems via software like Vista Exceed.

PCs provide the opportunity for cheap and powerful desktop computing facilities, where the processing is done locally.

Workstations are more powerful than microcomputers but this division is becoming rather blurred with the recent generations of processors. Screens on these devices are graphically oriented. Access to these systems is via a graphical or windows interface.

This means that the device we use looks rather like an ordinary typewriter keyboard, although some of the keys are different. However, the location of the letters, numbers and common symbols is fairly standard. Don't panic if you have never met a keyboard before. You don't have to know much more than where the keys are. Few programmers, even professionals, advance beyond the stage of using two index fingers and a thumb for typing. You will find that speed in typing is rarely important; it's accuracy that counts.

One thing that people unfamiliar with keyboards often fail to realise is that what you have typed in is not sent to the computer until you press the *carriage return* key. To achieve any sort of communication you must press that key; it will be somewhere on the right-hand side of the keyboard, and will be marked *return, c/r, send, enter*, or something similar.

2.3 Software

So far we have not mentioned software. Software is the name given to the programs that *run* on the hardware. Programs are written in *languages.* Computer languages are frequently divided into two categories: *high level* and *low level.* A low-level language (e.g., assembler) is closer to the hardware, whereas a high-level language (e.g., Fortran) is closer to the problem statement. There is typically a one-to-one correspondence between an assembly language statement and the actual hardware instruction. With a high-level language there is a one-to-many correspondence; one high-level statement will generate many machine-level instructions.

A certain amount of general purpose software will have been provided by the manufacturer. This software will typically include the basic operating system, one or more *compilers*, an *assembler*, an *editor*, and a *loader* or *link editor*.

- A *compiler* translates high-level statements into machine instructions.
- An *assembler* translates low-level or assembly language statements into machine instructions.
- An *editor* makes changes to text files, e.g., program sources.
- A *loader* or *link editor* takes the output from the compiler and completes the process of generating something that can be executed on the hardware.

These programs will vary considerably in size and complexity. Certain programs that make up the operating system will be quite simple and small (like copying utilities), whereas certain others will be relatively large and complex (like a compiler).

In this book we concentrate on software or programs that you write for your research or course work. As the book progresses you will be introduced to ways of building on what other people have produced and how to take advantage of the vast amount of software already written, tested and documented.

2.4 Problems

1. Distinguish between a memory address and memory contents.

2. What does RAM stand for?

3. What would a WOM (write only memory) do? How would you use it?

4. What does CPU stand for? What does it do?

5 What does a compiler do?

6 What does a linker do?

2.5 Bibliography

Baer J.L., *Computer Systems Architecture*, Computer Science Press.

Extremely readable coverage of this whole area. The version could do with an update, but it is still a very impressive coverage. Highly recommended.

Bhandarkar D.P., *Alpha Implementation and Architecture: Complete Reference and Guide*, Digital Press, 1996.

Excellent source of information on the Alpha architecture.

Intel currently make a lot of material available on their web site. Two useful URLs are:

- http://www.intel.com/

and

- http://developer.intel.com/

Well worth a look. Many publications are available in Adobe Acrobat Portable Document Format — PDF.

Reeves C.M., *An Introduction to Logical Design of Digital Circuits*, CUP, 1972.

This book provides coverage of the construction of the very simple electronic building blocks from which most modern computer systems are made. Relatively theoretical.

Tannenbaum A.S., *Structured Computer Organisation*, Prentice-Hall, 1976.

Very good coverage looking at a computer system in terms of a hierarchy of levels. An easy read.