Chapter I
Computer systems: the big picture

I.1 Computer architecture

The architecture of modern computers was formed in 1945, when John von Neumann suggested that the program could be stored in memory along with the data, with instructions fetched from memory and decoded for execution. Earlier electronic computers read their programs either from punched tape or from a bank of switches and plug-boards, which made setting up each program time consuming and error-prone. Almost all computers since 1945 have been based on the *von Neumann architecture*: a memory holding data and programs, a processor to fetch and execute instructions that operate on data, and input/output circuitry and devices.

Von Neumann architecture

I.1.1 Technology trends

What has changed since is the number of electronic switches in a computer and their speed. By changing from tubes to transistors, and then to many transistors tightly connected on a single chip, far more components may be packed into a small space, more reliably and cheaply, enabling larger memories, and processors able to do more work in parallel. Also, the smaller components operate faster, so the whole machine is doubly faster: faster operations, and more operations in parallel.

The number of transistors on a single chip has increased from 100 or so in the 1960's to a few billions on the largest chips now, and continues to grow as transistors get smaller. *Moore's law* holds that transistor counts double about every 18 months to two years, and that looks set to continue for a while yet. At the same time, the cost per transistor is also decreasing at a fast rate. Therefore computers are becoming both faster and cheaper.

Moore's law

The most important consequence of this trend is that it enables the creation of new applications. Some applications (e.g. the decoding of human DNA) would not have been possible before computers with high enough speed and low cost came into existence. Furthermore, the explosive growth of computer networks and their convergence with the traditional communications networks have further opened a whole new host of computer applications with no end in sight.

I.1.2 Types of computers

The large number of applications have lead to the development of three main classes of computers with different characteristics and operating system requirements:

- *Servers* are powerful machines with a lot of storage and high speed I/O which are used for handling either few, very computationally intensive tasks (e.g. scientific and engineering applications) or a large number of small tasks (e.g. web server). Servers drive the design of high speed processors and, commonly, contain a few of them on the same board[1]. They are designed to be used by multiple users running a number of applications per user, therefore their operating systems are tuned to achieve this in the most efficient way.

- The *desktop* category includes the common personal computer. This is the type of computer that most people are familiar with. Processors used in these systems always try to strike a balance between speed and cost. Although they are typically used by one person at a time, they allow multiple programs to run concurrently.

- *Embedded* computers do not look like computers at all. They are used in equipment which do not have processing/computing as their main purpose. Examples include mobile phones, cars, TV set-top boxes, etc. They are by far the largest group of computers today and this is likely to hold for a very long time. The design objective for embedded processors is low cost and power consumption (because most of them are in small, hand-held devices). They typically run a single application and are not programmable by their user, therefore the task of their operating system is simpler.

I.1.3 Hardware components

Five hardware parts can always be distinguished in every von Neumann computer:

- The *datapath* is made up of circuits which perform the actual operations on the data. In a sense this is the only useful part of a computer; everything else is there to support the datapath.

- The task of the *Control* is to fetch instructions from memory and control the operation — and sequence of operations — of the datapath, memory and I/O, according to what the instructions command.

[1] or chip, recently.

- The *memory* stores the program instructions and data while it is being executed.
- *Input* devices are used for getting data into the computer for processing. The keyboard, mouse, ... are common input devices for personal computers.
- *Output* hardware is used for getting the results of the computation out of the machine, e.g. screen, printer. There are obviously devices which are both input and output, e.g. the hard disk, network, ...

The datapath and control are collectively called the *processor* or *central processing unit* (CPU).

I.2 The software hierarchy

For someone who has only programmed a computer using a high-level language, the von Neumann model presented above seems quite detached from what the language provides. This is because the language, the *software libraries* and the *operating system* abstract away most the details of the underlying machine, effectively creating a two-layer software organisation. At the bottom layer, one finds the *systems software*, whose raison d'être is to provide services for the *applications software*, that sits at the top layer.

Systems software has two main components: *compilers* and the *operating system*.

Compiler

Compilers translate a program written in a high-level language into machine code that the processor can execute. Because the machine instructions are very simple compared to high-level language constructs and statements, this translation is a complicated task and the subject of other courses. Modern processors rely heavily on optimising compilers to achieve high-speed program execution. It is therefore quite frequent for computer architects to also be compiler experts and vice versa.

Operating system

The operating system (OS) is a collection of software routines that provide basic services, such as basic input/output, memory allocation, to other programs but it also performs supervisory tasks, such as sharing the processor time among multiple programs which are executing simultaneously. In this course we will see what fundamental support the operating system needs from the processor hardware and how the OS handles the processor time-sharing task. As with compilers, a course dedicated to operating systems is offered in the third year.