

3

Hardware

In this chapter you will learn about:

- ★ computer architecture
 - the Central Processing Unit (CPU)/microprocessor
 - von Neumann architecture
 - arithmetic and logic unit (ALU), control unit (CU) and registers
 - control bus, address bus, data bus
 - cores, cache and the internal clock
 - Fetch-Decode-Execute cycle
 - instruction set for a CPU
 - embedded systems
- ★ input and output devices
 - the following input devices: barcode and QR code scanners, digital cameras, keyboards, microphones, mouse, 2D/3D scanners and touch screens
 - the following output devices: actuators, light projectors, inkjet and laser printers, LED and LCD screens, speakers and 3D printers
 - sensors and their use in control and monitoring
- ★ data storage
 - primary storage (RAM and ROM)
 - secondary storage (magnetic, optical and solid state)
 - virtual memories
 - cloud storage
- ★ network hardware
 - network systems (NIC, MAC address, IP address and routers).

This chapter considers the hardware found in many computer systems. The hardware that makes up the computer itself and the various input and output devices will all be covered.

3.1 Computer architecture

洗衣机也有

3.1.1 The central processing unit (CPU)

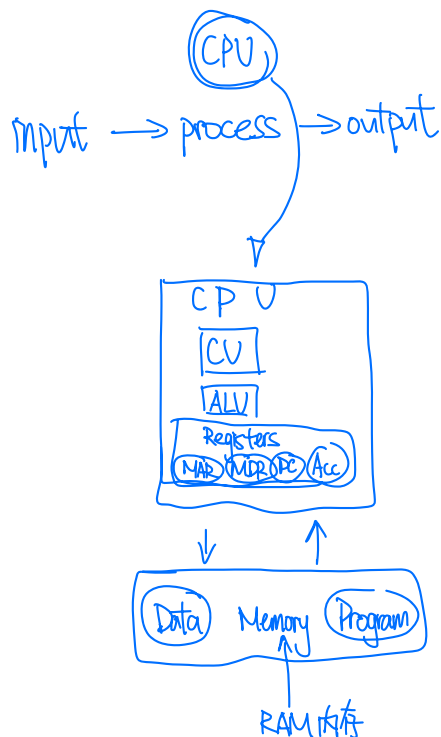
The **central processing unit (CPU)** (also known as a microprocessor or processor) is central to all modern computer systems (including tablets and smartphones). The CPU is very often installed as an **integrated circuit** on a single microchip. The CPU has the responsibility for the execution or processing of all the instructions and data in a computer application. As Figure 3.1 shows, the CPU consists of:

- » control unit (CU)
- » arithmetic and logic unit (ALU)
- » registers and buses.

3.1.2 Von Neumann architecture

Early computers were fed data while the machines were actually running; it wasn't possible to store programs or data, which meant they couldn't operate without

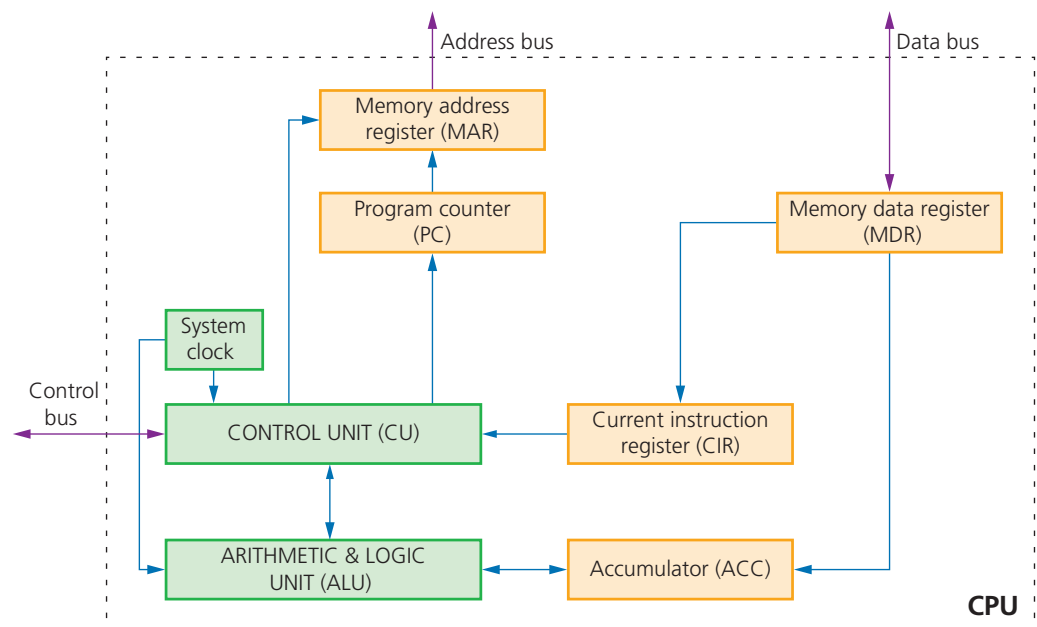
stored program concept



considerable human intervention. In the mid-1940s, John von Neumann developed the concept of the 'stored program computer', which has been the basis of computer architecture ever since. The von Neumann architecture had the following main novel features (none of which were available in computers prior to the mid-1940s):

- » the concept of a central processing unit (CPU or processor)
- » the CPU was able to access the memory directly
- » computer memories could store programs as well as data
- » stored programs were made up of instructions which could be executed in sequential order.

There are many diagrams of von Neumann CPU architecture in other textbooks and on the internet. The following diagram is one example of a simple representation of von Neumann architecture:



▲ **Figure 3.1** Von Neumann architecture

Components of the central processing unit (CPU)

The main components of the CPU are the Control Unit (CU), Arithmetic & Logic Unit (ALU) and system clock.

- do logic operation like AND or NOT
- contains ACC register

Arithmetic & Logic Unit (ALU)

The **Arithmetic & Logic Unit (ALU)** allows the required arithmetic (e.g. +, - and shifting) or logic (e.g. AND, OR) operations to be carried out while a program is being run; it is possible for a computer to have more than one ALU to carry out specific functions. Multiplication and division are carried out by a sequence of addition, subtraction and left or right logical shift operations.

Control Unit (CU)

The **control unit** reads an instruction from memory. The address of the location where the instruction can be found is stored in the Program Counter (PC). This instruction is then interpreted using the Fetch-Decode-Execute cycle (see later in this section). During that process, signals are generated along the control bus to tell the other components in the computer what to do. The control unit ensures synchronisation of data flow and program instructions throughout the computer. A

Link

For more arithmetic operations, please refer to Chapter 1.

Link

For more details of the system clock, see Section 3.1.3.

Link

For more details on RAM, see Section 3.3.

system clock is used to produce timing signals on the control bus to ensure this vital synchronisation takes place – without the clock the computer would simply crash!

The RAM holds all the data and programs needed to be accessed by the CPU. The RAM is often referred to as the **Immediate Access Store (IAS)**. The CPU takes data and programs held in **backing store** (e.g. a hard disk drive) and puts them into RAM temporarily. This is done because read/write operations carried out using the RAM are considerably faster than read/write operations to backing store; consequently, any key data needed by an application will be stored temporarily in RAM to considerably speed up operations.

Registers 寄存器

One of the most fundamental components of the von Neumann system are the **registers**. Registers can be general or special purpose. We will only consider the special purpose registers. A full list of the registers used in this textbook are summarised in Table 3.1. The use of these registers is explained more fully in the Fetch–Decode–Execute cycle (see later in this section).

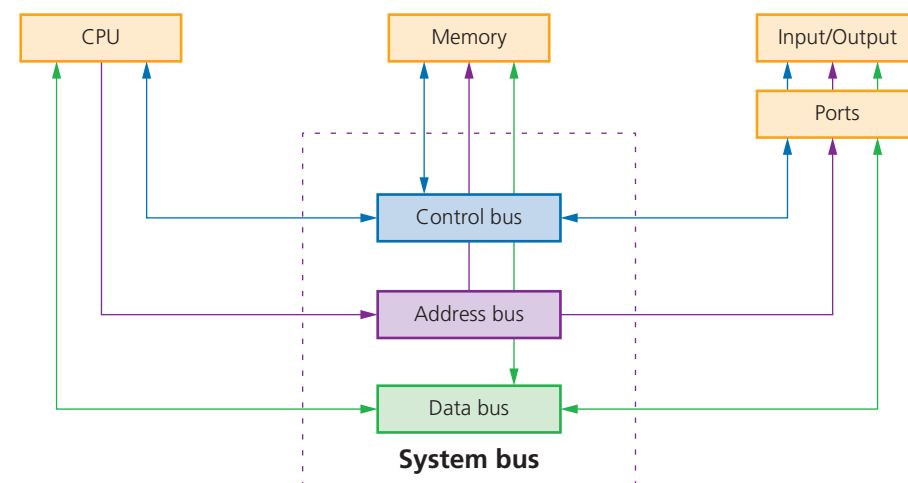
▼ **Table 3.1** Specific purpose registers

Register	Abbreviation used	Function/purpose of register
current instruction register	Stores... CIR data (instruction)	this register stores the current <u>instruction</u> being decoded and executed
accumulator 累加器	ACC	this register is used when carrying out ALU calculations; it stores data temporarily during the calculations
memory address register	MAR Address	this register stores the <u>address</u> of the memory location currently being read from or written to
memory data/buffer register	MDR data	this register stores <u>data</u> which has just been read from memory or data which is about to be written to memory
program counter	PC address	this register stores the <u>address</u> where the next instruction to be read can be found

PC
↓
MAR
↓
MDR
↓
CIR
↓
CU

System buses and memory

Earlier on, Figure 3.1 referred to some components labelled as buses. Figure 3.2 shows how these buses are used to connect the CPU to the memory and to input/output devices.



▲ **Figure 3.2** System buses and memory

Eg 8G + 512G
 8G = memory / 内存 / 主存 primary memory / RAM
 512G = 次级储存 secondary storage / 硬盘
 需要被CPU用

▼ **Table 3.2** Section of computer memory

Address	Contents
1111 0000	0111 0010
1111 0001	0101 1011
1111 0010	1101 1101
1111 0011	0111 1011
1111 1100	1110 1010
1111 1101	1001 0101
1111 1110	1000 0010
1111 1111	0101 0101

Memory

The computer memory is made up of a number of partitions. Each partition consists of an **address** and its contents. Table 3.2 uses 8 bits for each address and 8 bits for the content. In a real computer memory, the address and its contents are actually much larger than this.

The address will uniquely identify every **location** in the memory and the contents will be the binary value stored in each location.

Let us now consider two examples of how the MAR and MDR registers can be used when carrying out a read and write operation to and from memory:

First, consider the **READ** operation. We will use the memory section shown in Table 3.2. Suppose we want to read the contents of memory location **1111 0001**; the two registers are used as follows:

- » the address of location 1111 0001 to be read from is first written into the MAR (memory address register):

MAR:	1	1	1	1	0	0	0	1
------	---	---	---	---	---	---	---	---

- » a 'read signal' is sent to the computer memory
- » the contents of memory location 1111 0001 are then put into the MDR (memory data register):

MDR:	0	1	0	1	1	0	1	1
------	---	---	---	---	---	---	---	---

Now let us now consider the **WRITE** operation. Again, we will use the memory section shown in Table 3.2. Suppose this time we want to show how the value 1001 0101 was written into memory location 1111 1101:

- » the data to be stored is first written into the MDR (memory data register):

MDR:	1	0	0	1	0	1	0	1
------	---	---	---	---	---	---	---	---

- » this data has to be written into location with address: 1111 1101; so this address is now written into the MAR:

MAR:	1	1	1	1	1	1	0	1
------	---	---	---	---	---	---	---	---

- » finally, a 'write signal' is sent to the computer memory and the value 10010101 will then be written into the correct memory location.

Input and output devices

The input and output devices will be covered in more detail in Section 3.2. They are the main method of entering data into and getting data out of computer systems. Input devices convert external data into a form the computer can understand and can then process (e.g. keyboards, touch screens and microphones). Output devices show the results of computer processing in a human understandable form (e.g. printers, monitors and loudspeakers).

(System) buses

(System) buses are used in computers as parallel transmission components; each wire in the bus transmits one bit of data. There are three common buses used in the von Neumann architecture known as: address bus, data bus and control bus.

Address bus

As the name suggests, the **address bus** carries addresses throughout the computer system. Between the CPU and memory, the address bus is **unidirectional** (i.e. bits can travel in one direction only); this prevents addresses being carried back to the CPU, which would be an undesirable feature.

The width of a bus is very important. The wider the bus, the more memory locations that can be directly addressed at any given time, e.g. a bus of width 16 bits can address 2^{16} (65 536) memory locations whereas a bus width of 32 bits allows 4 294 967 296 memory locations to be **simultaneously** addressed. However, even this isn't large enough for modern computers but the technology behind even wider buses is outside the scope of this book.

Data bus

The **data bus** is **bidirectional** (allowing data to be sent in both directions along the bus). This means **data can be carried** from CPU to memory (and vice versa) and to and from input/output devices. It is important to point out that data can be an address, an instruction or a numerical value. As with the address bus, the width of the data bus is important; the wider the bus the larger the **word length** that can be transported. (A **word** is a group of bits which can be regarded as a single unit e.g. 16-bit, 32-bit or 64-bit word lengths are the most common.) Larger word lengths can improve the computer's overall performance.

Control bus

The **control bus** is also bidirectional. It carries signals from the control unit (CU) to all the other computer components. It is usually 8-bits wide. There is no real need for it to be any wider since it only **carries control signals**.

Fetch–Decode–Execute cycle

To carry out a set of instructions, the CPU first of all **fetches** some data and instructions from memory and stores them in suitable registers. Both the address bus and data bus are used in this process. Once this is done, each instruction needs to be **decoded** before finally being **executed**. This is all known as the **Fetch–Decode–Execute cycle**.

Fetch (instruction)

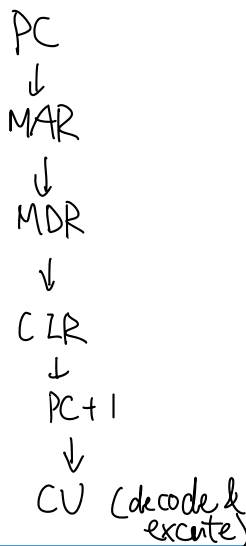
Both data and instruction can be stored in MDR. In the **Fetch–Decode–Execute cycle**, the next instruction is **fetched** from the memory address currently stored in the MAR and the instruction is stored in the MDR. The contents of the MDR are then copied to the Current Instruction Register (CIR). The PC is then incremented (increased by 1) so that the next instruction can be then be processed.

Decode

The instruction is then decoded so that it can be interpreted in the next part of the cycle.

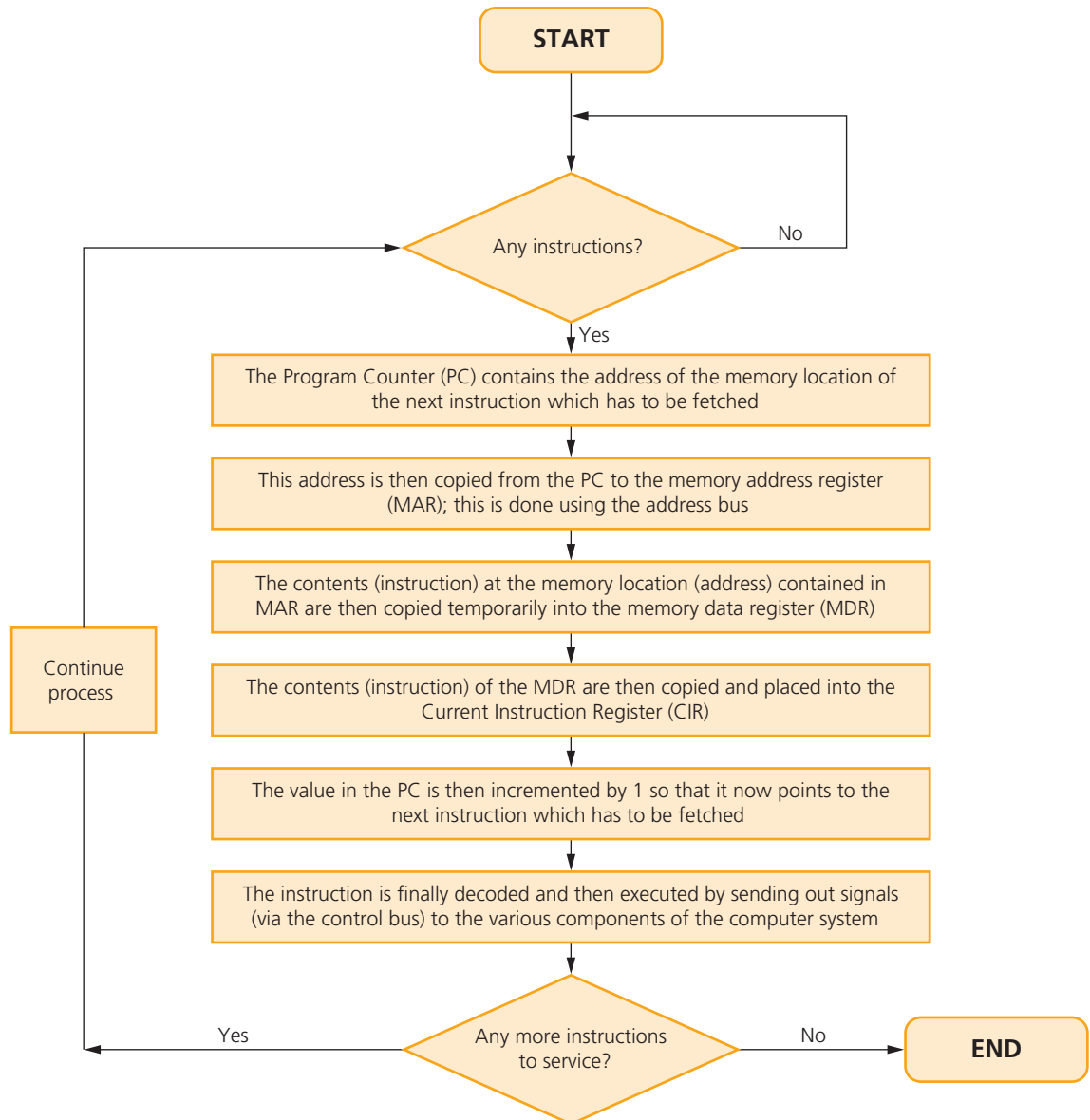
Execute

The CPU passes the decoded instruction as a set of control signals to the appropriate components within the computer system. This allows each instruction to be carried out in its logical sequence.

**Link**

See Section 4.2.2 for more details about these instructions.

Figure 3.3 shows how the Fetch–Decode–Execute cycle is carried out in the von Neumann computer model.



▲ **Figure 3.3** Fetch–Decode–Execute cycle flowchart

3.1.3 Cores, cache and internal clock

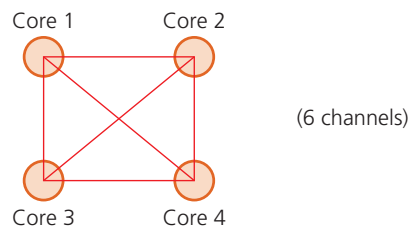
We will now consider the factors that determine the performance of a CPU. The first thing to consider is the role of the **system clock**. The clock defines the **clock cycle** that synchronises all computer operations. As mentioned earlier, the control bus transmits timing signals ensuring everything is fully synchronised. By increasing clock speed, the processing speed of the computer is also increased (a typical current value is 3.5 GHz – which means 3.5 billion clock cycles a second). Although the speed of the computer may have been increased, it isn't possible to say that a computer's overall *performance* is necessarily increased by using a higher clock speed. Other factors need to be considered, for example:

- 1 The width of the address bus and data bus (as mentioned earlier) can also affect computer performance and needs to be taken into account.

- 2 **Overclocking** is a factor to consider. The clock speed can be changed by accessing the **BIOS (Basic Input/Output System)** and altering the settings. However, using a clock speed higher than the computer was designed for can lead to problems, for example:
- i execution of instructions outside design limits can lead to seriously unsynchronised operations (i.e. an instruction is unable to complete in time before the next one is due to be executed) – the computer would frequently crash and become unstable
 - ii overclocking can lead to serious overheating of the CPU again leading to unreliable performance.
- 3 The use of **cache memories** can also improve CPU performance. Unlike RAM, cache memory is located within the CPU itself, which means it has much faster data access times than RAM. Cache memory stores frequently used instructions and data that need to be accessed faster, which improves CPU performance. When a CPU wishes to read memory, it will first check out the cache and then move on to main memory/RAM if the required data isn't there. The **larger the cache memory size the better the CPU performance.**
- 4 The use of a different **number of cores** can improve computer performance. One core is made up of an ALU, a control unit and the registers. Many computers are dual core (the CPU is made up of two cores) or quad core (the CPU is made up of four cores). The idea of using more cores alleviates the need to continually increase clock speeds. However, doubling the number of cores doesn't necessarily double the computer's performance since we have to take into account the need for the CPU to communicate with each core; this will reduce overall performance. For example, with a **dual core** the CPU communicates with both cores using one channel reducing some of the potential increase in its performance:

▲ **Figure 3.4**

while, with a **quad core** the CPU communicates with all four cores using six channels, considerably reducing potential performance:

▲ **Figure 3.5**

So all these factors need to be taken into account when considering computer performance. Summarising these points:

- ① » increasing bus width (data and address buses) increases the performance and speed of a computer system
- ② » increasing clock speed will potentially increase the speed of a computer

- 3 » a computer's performance can be changed by altering bus width, clock speed and use of multi-core CPUs
- 4 » use of cache memories can also speed up a CPU's performance.

Activity 3.1

- 1 **a** Name three buses used in the von Neumann architecture.
b Describe the function of each named bus.
c Describe how bus width and clock speed can affect computer performance.
- 2 Complete the following paragraph by using terms from this chapter:
 The CPU data and instructions required for an application and temporarily stores them in the until they can be processed. The is used to hold the address of the next instruction to be executed. This address is copied to the using the The contents at this address are stored in the Each instruction is then and finally by sending out using the Any calculations carried out are done using the During any calculations, data is temporarily held in a special purpose register known as the

3.1.4 Instruction set 指令集

In a computer system, instructions are a set of operations which are decoded in sequence. Each operation will instruct the ALU and CU (which are part of the CPU). An operation is made up of an **opcode** and an **operand**.

The opcode informs the CPU what operation needs to be done

The operand is the data which needs to be acted on or it can refer to a register in the memory

Link

See Section 4.2.2 for examples of opcodes and operands in assembly language.

Link

For more on interpreters and compilers see Section 4.2.3.

Since the computer needs to understand the operation to be carried out, there is actually a limited number of opcodes that can be used; this is known as the **instruction set**. All software running on a computer will contain a set of instructions (which need to be converted into binary). The Fetch–Decode–Execute cycle is the sequence of steps used by the CPU to process each instruction in sequence.

One example of an instruction set is the X86, a common CPU standard used in many modern computers. Although different computer manufacturers will adopt their own internal electronic design, if the computer is based on the X86 CPU then all designs will share almost identical instruction sets. For example, Intel Pentium and AMD Athlon CPUs use almost identical X86 instruction sets even though they are based on very different electronic designs.

(Note of caution: do not confuse instruction sets with programming code; instruction sets are the **low-level language instructions that instruct the CPU how to carry out an operation**. Program code needs interpreters or compilers to convert the code into the instruction set understood by the computer. Some examples of instruction set operations include: ADD, JMP, LDA, and so on.)

computer system which is used to perform a dedicated function inside a larger mechanical unit.

3.1.5 Embedded systems

An embedded system is a combination of hardware and software which is designed to carry out a specific set of functions. The hardware is electronic, electrical or electro-mechanical. Embedded systems can be based on:

microcontrollers:

this has a CPU in addition to some RAM and ROM and other peripherals all embedded onto one single chip (together they carry out a specific task)

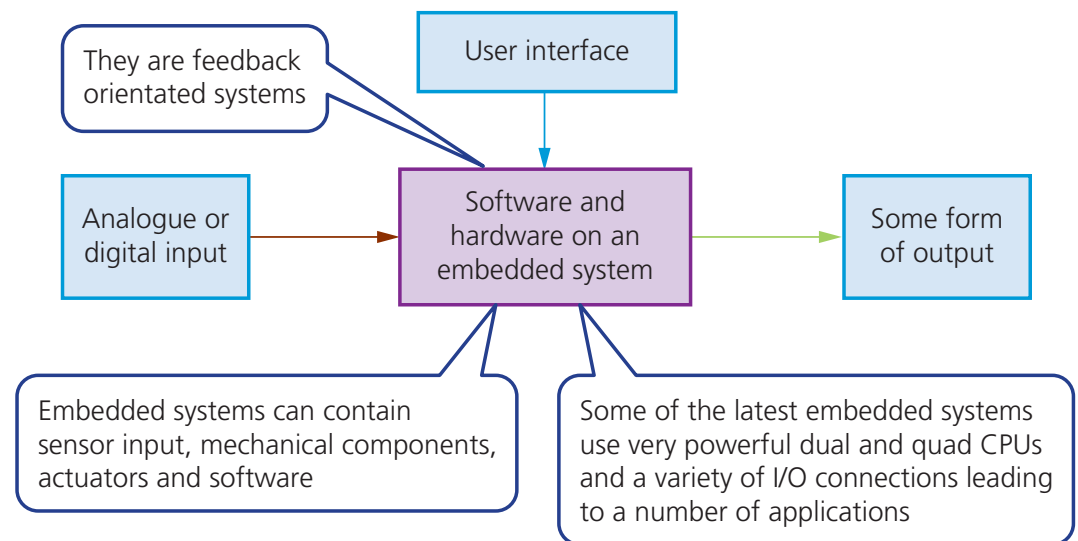
microprocessor:

integrated circuit which only has a CPU on the chip (there is no RAM, ROM or peripherals – these need to be added)

system on chips (SoC):

this may contain a microcontroller as one of its components (they almost always will include CPU, memory, input/output (I/O) ports and secondary storage on a single microchip)

An embedded system will have a **specific set of tasks**; Figure 3.6 summarises how embedded systems work in general:



▲ **Figure 3.6** Embedded systems

When installed in a device, either an operator can input data manually (for example, select a temperature from a keypad or turn a dial on an oven control panel) or the data will come from an automatic source, such as a sensor. This sensor input will be analogue or digital in nature, for example, inputs such as oxygen levels or fuel pressure in a car's engine management system. The output will then carry out the function of the embedded system by sending signals to the components that are being controlled (for example, increase the power to the heating elements in an oven or reduce fuel levels in the engine).

Depending on the device, embedded systems are either programmable or non-programmable. Non-programmable devices need, in general, to be replaced if they require a software upgrade. Programmable devices permit upgrading by two methods:

- » connecting the device to a computer and allowing the download of updates to the software (for example, this is used to update the maps on a GPS system used in a vehicle)

- » automatic updates via a Wi-Fi, satellite or cellular (mobile phone network) link (for example, many modern cars allow updates to engine management systems and other components via satellite link).

There are definite benefits and drawbacks of devices being controlled using embedded systems:

▼ **Table 3.3** Benefits and drawbacks of using embedded systems

Benefits	Drawbacks
they are small in size and therefore easy to fit into devices	it can be difficult to upgrade some devices to take advantage of new technology
compared to other systems, they are relatively low cost to make	troubleshooting faults in the device becomes a specialist task
they are usually dedicated to one task allowing simple interfaces and often no requirement for an operating system	although the interface can appear to be more simple (e.g. a single knob) in reality it can be more confusing (e.g. changing the time on a cooker clock can require several steps!)
they consume very little power	any device that can be accessed over the internet is also open to hackers, viruses, etc.
they can be controlled remotely using a mobile phone, for example	due to the difficulty in upgrading and fault finding, devices are often just thrown away rather than being repaired (very wasteful)
very fast reaction to changing input (operate in real time and are feedback orientated)	can lead to an increase in the 'throw away' society if devices are discarded just because they have become out-of-date
with mass production comes reliability	

Because embedded systems can be connected to the internet, it is possible to control them remotely using a smartphone or computer. For example, setting the central heating system to switch on or off while away from home or remotely instructing a set top box to record a television programme. Since embedded systems are dedicated to a specific set of tasks, engineers can optimise their designs to reduce the physical size and cost of the devices. The range of applications are vast, ranging from a single microcontroller (for example, in an MP3 player) to a complex array of multiple units (for example, in a medical imaging system).

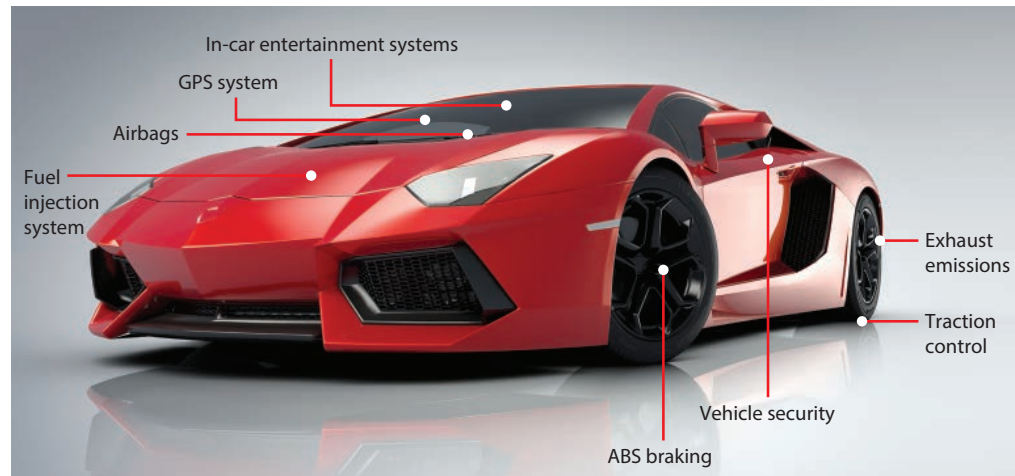
It is worth mentioning here that a computer is **not** an example of an embedded system. Computers are multi-functional (that is, they can carry out many different tasks which can be varied by using different software) which means they can't be classed as embedded systems.

Examples of the use of embedded systems

Motor vehicles

Modern cars have many parts that rely on embedded systems to function correctly. Figure 3.7 shows some of the many components that are controlled in this way.

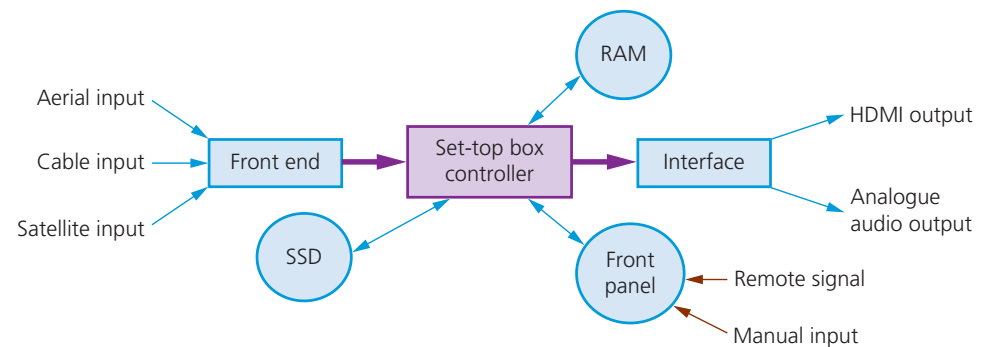
hospital equipment
coffee machines
traffic lights
...



▲ **Figure 3.7** Embedded systems found in a car

Set-top box

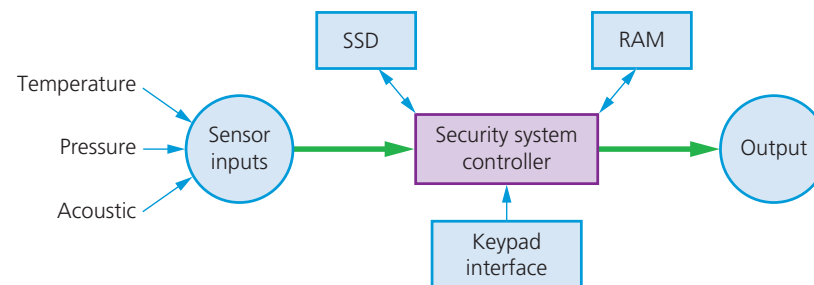
In this example, a set-top box uses an embedded system to allow, for example, recording and playback of television programmes. This can be operated remotely by the user when not at home using an internet-enabled device or by using the interface panel when at home. The embedded system will look after many of the functions involving inputs from a number of sources such as a Solid State Device (SSD) (where television programmes can be stored or retrieved) or a satellite signal (where it will be necessary to decode the incoming signal).



▲ **Figure 3.8** Embedded system found in a set-top box

Security systems

Embedded systems are used in many security devices:



▲ **Figure 3.9** Embedded system found in a security system

The security code is set in RAM and the alarm activated or deactivated using the keypad. Data from sensors is sent to the controller which checks against values stored on the SSD (these settings are on SSD rather than RAM in case

Link

See Section 3.2.3 for sensors used in security systems.

the sensitivity needs to be adjusted). An output can be a signal to flash lights, sound an alarm or send a message to the home owner via their mobile phone. Again, the home owner can interface with the system remotely if necessary.

Lighting systems

Embedded systems are used in modern sophisticated lighting systems from simple home use to major architectural lighting systems. We will concentrate here on a lighting system used in a large office. The system needs to control the lighting taking into account:

- » the time of day or day of the week
- » whether the room is occupied
- » the brightness of the natural light.

An embedded system can automatically control the lighting using a number of inputs (such as light sensors) and key data stored in memory. Again, this fits very well into the system described in Figure 3.6.

The time of day or day of the week is important data in an office environment since energy is saved if the system switches to low lighting levels when unoccupied. This can be over-ridden by the second bullet point (above); if there is movement in the office then correct lighting levels will be automatically restored. On a very bright sunny day, the system could automatically dim the lights, only increasing the light output if natural light levels fall below a set value. There are many internal and external lighting systems that could be controlled by embedded systems (e.g. a fountain light display or a light show on a building to commemorate a special occasion). They are also used to trigger emergency lighting in, for example, aeroplanes in case of an emergency.

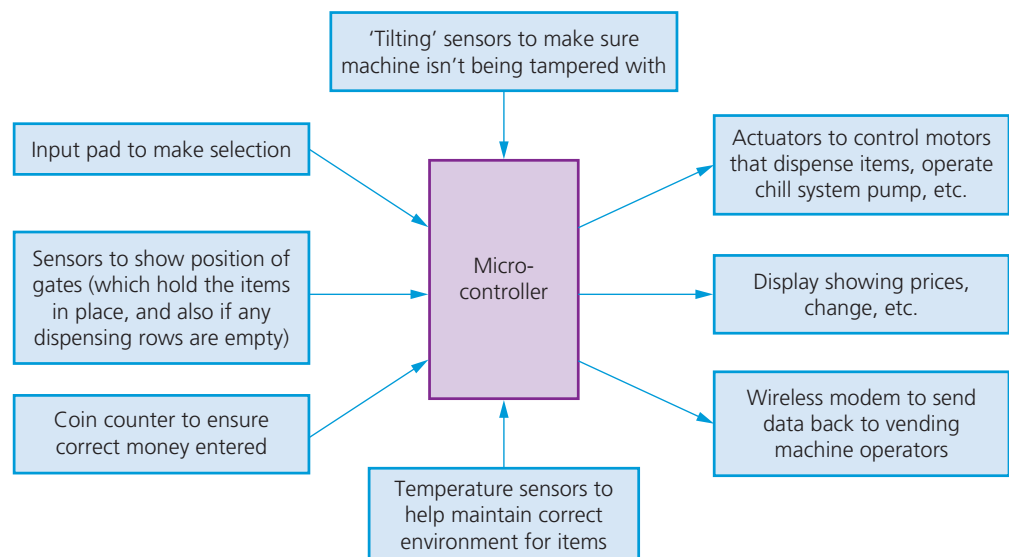
Some lighting systems use Bluetooth light bulbs. This allows the embedded system to control each bulb independently. Many of the bulbs available today use LEDs and many come in a number of colours to change the mood.



▲ Figure 3.10 LED light bulb

Vending systems

Vending machines make considerable use of embedded systems. They usually use microcontrollers to control a number of functions that we all associate with vending machines:



▲ Figure 3.11 Embedded system found in a vending machine

At the heart of the vending machine is an embedded system in the form of a microcontroller. Inputs to this system come from the keypad (item selection) and from sensors (used to count the coins inserted by the customer, the temperature inside the machine and a 'tilt sensor' for security purposes). The outputs are:

- » actuators to operate the motors, which drive the helixes (see figure below) to give the customers their selected item(s)
- » signals to operate the cooling system if the temperature is too high
- » item description and any change due shown on an LCD display panel
- » data sent back to the vending machine company so that they can remotely check sales activity (which could include instructions to refill the machine) without the need to visit each machine.



▲ **Figure 3.12** Helix used in a typical vending machine

All of this is controlled by an embedded system which makes the whole operation automatic but also gives immediate sales analysis which would otherwise be very time consuming.

Washing machines

Many 'white goods' (such as refrigerators, washing machines, microwave ovens, and so on) are controlled by embedded systems. They all come with a keypad or dials that are used to select the temperature, wash cycle or cooking duration. This data forms the input to the embedded system, which then carries out the required task without any further human intervention. As with other devices, these 'white goods' can also be operated remotely using an internet-enabled smartphone or computer.

Activity 3.2

- 1 **a** Explain how it is possible to increase the performance of a CPU/microprocessor. In your explanation, include some of the risks associated with your suggestions to improve performance.
- b** What is meant by the term *instruction set*?
- 2 A car is fitted with the latest GPS navigation system. This device is controlled by an embedded system in the form of a microcontroller.
 - a** Describe the inputs needed by the embedded system and describe which outputs you would expect it to produce.
 - b** Since updates to the GPS device are required every six months, explain how the device is updated without the need to take the car to the garage every six months.