COMPUTER SYSTEMS

National 5 Computing Science

Contents

[Data representation 3](#_Toc482281627)

[Using binary to represent data 3](#_Toc482281628)

[Integers 4](#_Toc482281629)

[Real numbers 5](#_Toc482281630)

[Characters 6](#_Toc482281631)

[Vector graphics 7](#_Toc482281632)

[Bitmapped graphics 7](#_Toc482281633)

[Computer structure 10](#_Toc482281634)

[Computer architecture 10](#_Toc482281635)

[Program instructions 13](#_Toc482281636)

[Translator software 13](#_Toc482281637)

[Environmental impact 15](#_Toc482281638)

[Security precautions 16](#_Toc482281639)

[Firewalls 16](#_Toc482281640)

[Encryption 16](#_Toc482281641)

## Data representation

### Using binary to represent data

The number 3862 in decimal means 3 x 103 + 8 x 102 + 6 x 101 + 2 x 100

The number 110101 in binary means 1 x 25 + 1 x 24 + 0 x 23 + 1 x 22 + 0 x 21 + 1 x 20

**Decimal** means numbers are written in base 10 using the digits 0 to 9

**Binary** means numbers are written in base 2 using the digits 0 and 1

For example, a binary piece of data could be 10111001 or 10000010100101000100010110010001

Almost all data within computer systems is held in binary form. This is often referred to as **digital data**

Computer systems use binary numbers to encode all information because

1. Electronically it is easier to represent just two values
2. It is easier to build electronic circuitry for carrying out calculations when only two values are involved
3. The significant advances in hardware mean that it is possible to store and process enormous quantities of digital information (information that is held in codes made up of 1s and 0s)

How does a computer system know what 10000010100101000100010110010001 represents? Is it a piece of text? Is it a number? Is it a piece of audio, video or graphic?

It depends largely on what the computer is *expecting*! Each piece of data will carry a code indicating what type of data it is

Each type of data is stored in different ways

### Integers

An integer is a number with no fractional part e.g. 76, 8945, 100006, 2

(n.b. National 5 does not cover negative integers)

How to turn an integer from decimal to binary

 e.g. What is 73 in binary?

 The column headings are always (right to left) 1, 2, 4, 8, 16, 32, and so on. These are powers of 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Put 1s and 0s in the right columns so that the numbers add up to 73

 Answer: 73 in decimal = 01001001 in binary

How to turn an integer from binary to decimal

 e.g. What is 10011100 in decimal?

 Put the 1s and 0s under the column headings and add up the numbers where there are 1s

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

 Answer: 10011100 in binary = 156 in decimal

Range of values

For example, if 8 bits are used then

the smallest number would be 00000000 (this is 0 in decimal)

the largest number would be 11111111 (this is 255 in decimal. 255 = 28 – 1)

So the range of possible values is 00000000 to 11111111 (in binary)

0 to 255 or 0 to 28 – 1 (in decimal)

In general, with n bits, the range of values possible with n bits is 0 to 2n-1

### Real numbers

A real number is a number that has a fractional part e.g. 67.341, 0.0005, 100.3

The number is stored using *floating point representation* using a *mantissa* and an *exponent*

 e.g. 6527.802 = 0.6527802 x 104

 The decimal point has “floated” to the front of the number

 The 6527802 is called the *mantissa* and the 4 is called the *exponent*

This is similar to the scientific notation that you might meet in mathematics or science – the point floats to a different place but it is the same idea

The mantissa and the exponent would be stored as separate numbers. The computer recreates the original number for calculations when it needs to

### Characters

**character** – any symbol or letter found on a keyboard. Each character has a unique numerical value inside the computer e.g. ‘A’ is 01000001 (= 65).

How does the computer know whether it should be the number 65 or the letter ‘A’ ? *See page 3*

**character set** **of a computer** – a list of all characters a computer can process and store. Different computers have different character sets

**control characters** – do not print on the screen in the normal way. They control certain operations of the computer such as ‘cursor up’, cursor down’, ‘clear screen’, ‘tab’, ‘return (enter)’, ‘delete’ etc. If you look at the table below, they are characters with ASCII codes 0 to 31

**(EXTENDED ) ASCII**

* American Standard Code for Information Interchange
* An ASCII code is stored using one byte (8 bits)
* 28 = 256 therefore ASCII can represent 256 different characters
* In ASCII, codes 0 to 31 are used for control characters e.g. 0001001 is the tab key



**Aside:** There are other coding systems for characters. For example, Unicode is designed to represent all the characters from all of the world’s major languages. It was originally set up as a 16-bit code.

216 = 65536 different characters. Nowadays even more bits are used to give an even greater range of characters

### Image result for example vector graphicVector graphics

**Representing graphics using vector graphics**

* + - A vector graphic is made up of **objects** each having a set of **attributes**
		- Examples of **objects:** rectangle, ellipse, line, polygon
		- Examples of **attributes:** co-ordinates (position), fill colour, line colour
		- e.g. a red rectangle. The *object* is a ‘rectangle, its *attribute* of fill colour has the value red
		- an example of a vector graphic package is CorelDRAW
		- the file size increases as the complexity of the graphic increases. The more objects that are added, the more data that must be stored

### Bitmapped graphics

**Representing graphics using bitmaps**

* **Pixel** - short for **pic**ture **el**ement
* **Black and white bitmapped graphics**
	+ one bit per pixel (usually set to 0 for white, 1 for black)
	+ This means that the number of bits = number of pixels
* **Bitmap** – a grid of bits (1s and 0s) is used to represent the colour of the pixels of the graphic
* **Colour bitmapped graphics**
	+ - Several bits are assigned to each pixel to represent the colour of the pixel
		- e.g. 10 bits per pixel means 1024 possible colours for each pixel
		- **True colour** means 24 bits per pixel (the limit at which the human eye can distinguish between colours). This gives a total range of 16777216 colours
* **Resolution** of a graphic
* the number of pixels making up the graphic
* determines the quality (accuracy) of the graphic
* the smaller the pixels, the higher the resolution and the better the quality
* **Screen resolution** e.g. 1024 x 768 (horizontal x vertical)
* **Capture resolution** e.g. digital camera with a total resolution of 8 megapixels

**Bit depth**

If more than one bit is used for each pixel, then shades of grey or even colour can be represented. For example, if two bits are used for each pixel then there are four possible combinations (00, 01, 10, 11) which can be used to represent the colour of a pixel (e.g. white, light grey, dark grey, black). The greater the number of bits used for each pixel, the greater the variety of colours)

***Definition*: Bit depth** (colour depth) is the number of bits used to represent colour or shades of grey in a graphic

How many bits would 128 colours require?

Ans: 128 = 27 so 7 bits required

What is the bit depth of a graphic allowing up to 1024 colours?

Ans: 1024 = 210 therefore bit depth = 10

A graphic has a bit depth of 6. How many colours does this allow? Answer: 26 (64)

**Storage requirements of a bitmapped graphic**

* Changing the colours in a bit-mapped graphic does not alter the storage requirements. If the bit depth or resolution of a bit-mapped graphic changes, the file size will change
* As bit depth and resolution increases, the quality of the image will increase but the file size will too, meaning **increased storage requirements** and **slower transmission times**

**Comparison of bit mapped and vector graphics**

**EDITING**: When two shapes overlap in a bit mapped package, the shape on top will “rub out” the shape underneath whereas, with a vector graphic, individual objects can be selected and layered

**EDITING at pixel level**: This is possible with a bit-mapped graphic

To edit an object in a vector graphics file, the object needs to be selected and then the values of its attributes are changed e.g. colour (attribute) red (value) might be changed to colour green

**FILE SIZE**: The file size of a bit mapped graphic stays the same regardless of the amount of detail in the graphic because each pixel is stored regardless of its colour, whereas a vector graphic file size increases as objects are added

The description of each object making up a vector graphic is called its “set of attributes”

**RESOLUTION INDEPENDENCE**: This is when the resolution of the graphic on the screen does not affect the resolution of the printed graphic. This is a characteristic of **vector** **graphics** files because **the processor sends the file of object attributes** that represent the graphic to the printer and the printer prints the graphic off using its own resolution settings

When a **bit map** is printed out, the resolution of the printout is the same as the resolution of the bit map on the screen regardless of the resolution of the printer. This is because **the processor sends the bit map** as it is to the printer and it is printed as it is

## Computer structure

### Computer architecture

Inside a computer…

Memory

RAM

ROM

Processor

Registers

Data bus

ALU

Control unit

Address bus

Clock

#### The processor (sometimes called the CPU or Central Processing Unit)

 **ALU – Arithmetic Logic Unit**

 Carries out calculations and makes decisions using logical operations

e.g. if a program needs to add 2.73 and 5.76, these would be brought into the ALU where they would be added together

e.g. if a program needs to compare the values of variables x and y, this would be done in the ALU

 **Control unit**

Controls all the parts of the processor and makes sure that program instructions are synchronized to be executed in the correct order

Makes sure that all events in the processor happen in the correct place and at the correct time

**Registers**

 Registers are storage locations inside the processor chip that hold data while the processor is using it

**MAR** – Memory Address Register. This holds the *address* of the location in memory that is currently being accessed

 **MDR** – Memory Data Register. This holds the *contents* of the location currently being accessed

#### Memory

Memory is organised as a **series of storage locations**, each with its own **unique address**. The processor uses the address to place data in a specific location and to fetch data from a specific location

**RAM – Random Access Memory**. This is called **volatile** memory which means that it holds its data as long as the computer is switched on, but when the computer is switched off any data in RAM is lost. RAM can be edited by the user

Backing storage such as hard disk drives (HDD), solid state drives (SSD) and flash ROM are used to hold data whilst the computer is switched off. Flash ROM can be reprogrammed whilst inside the computer and is popular at present e.g. USB flash drives and digital camera memory cards.

**ROM – Read Only** Memory. This is called **non-volatile** memory which means that it holds its data permanently even when the computer is switched off. It is often used to hold the start-up operating system files. ROM is not normally editable

#### Buses

Buses are sets of wires connecting parts of the computer together

 **Data bus**

 Bi-directional. This bus transfers data between main memory and the processor

 The number of wires making up the bus determines the *quantity of data* the bus can carry

 e.g. if the bus has 8 wires

 then the number of possible numbers it can carry is 28

 and the greatest integer it can carry is 28 – 1 (assuming no negative integers)

**Address bus**

Uni-directional. The address bus indicates which address in memory is being *read from* or *written to*

The number of wires making up the bus determines the *number of storage locations* the processor can access

e.g. if the bus has 32 wires then the number of possible addresses it can access is 232 and the range of addresses is from 0 to 232 - 1

**Fetch-execute cycle**

If a program is running, its instructions (the lines of code that have been translated into machine code) will be held in a series of storage locations in memory

The processor starts with the first instruction. It specifies the instruction’s address (where the instruction is) using the address bus

The instruction is transferred to the processor via the data bus

The processor **executes** the instruction and then **fetches** the next instruction. A program counter keeps track of which instruction the processor is on

### Program instructions

High level language instructions need to be translated into machine code instructions (in binary) in order for the processor to be able to execute them. This means that, in the end, even the most complicated programs like a computer game or an operating systems are just a huge collection of 1s and 0s

Every processor has its own instruction set

Portable software: A piece of software that can run on different computer systems i.e. it can be translated into the machine code of a variety of processors e.g. for an Intel processor, for an AMD processor etc.

A short machine code program might look like this

 Operator Operand *What the program means*

 01001000 00001100 *Load accumulator (in the ALU) with the contents of address 12*

 01110110 00000101 *Add to the accumulator the number 5*

 01011001 00001100 *Store the value of the accumulator in location with address 12*

Obviously, programming in binary machine code is time-consuming and hugely error-prone. Coding in machine code is usually only done in situations where the programmer needs to refer directly to the individual parts of the processor chip or refer directly to locations in main memory

### Translator software

High-level languages are **designed to be read by humans**. They try to use simple, common language to form instructions for the processor to carry out. For example, LiveCode

**on** placeFlags

 **repeat** with flagNumber = 1 to 4

 **set** the location of image id flagData[flagNumber]["imageID"] to flagData[flagNumber]["current position"] \* 100, 200

 **end** **repeat**

**end** placeFlags

Other examples of high level languages are C#, C++, Java, PHP, HTML, JavaScript, Perl, Ruby, Python, Scratch

However, before these instructions can be executed by the processor, they must be **translated into low-level** instructions. Each high-level instruction will translate into several low-level instructions

High-level language translators come in two forms:

#### Interpreters

The program (source code) is in memory along with the interpreter which **translates and executes each line of the program in turn**

**Advantage:** Often used during development time. It is easier to locate errors since the interpreter can highlight lines in the source code where the program fails

**Disadvantages:** The run (execution) is slower because of the simultaneous translation and because any repeated line will be re-translated. In addition, the interpreter software takes up memory

#### Compilers

The program (source code) is translated into **machine object code** that can be executed by the processor independently of the original source code. The only software that needs to be in main memory to run the program is the object code

**Advantages:** The run (execution) will be fast because the translation has been done separately. In addition, the original program source code is not available for people to view and possibly plagiarise.

**Disadvantage:** All errors need to have been corrected before source code can be compiled to run independently

## Environmental impact

A computing-related carbon footprint measures the amount of greenhouse gases emitted by computer equipment during its

* **manufacture**
* **use**
* **disposal**

In particular, the use of computer equipment can require large amounts of energy

* *Problem:* Some energy sources: oil, gas and coal are non-renewable. This means that the use of them as energy sources is unsustainable. In the long run, the supplies will run out – and mining/drilling for such energy sources can harm the environment
* *Problem:* Energy production e.g. by a coal-fired power station, releases CO2 and other gases that contribute to global warming

If the user of a computer system is accessing the Internet, then the carbon footprint also involves the electricity usage (and therefore carbon emissions) of the companies that run the websites that the user views

We should try to reduce carbon emissions while we use computers by making equipment more efficient. Modern manufacturers of IT equipment are constantly looking for ways to reduce the energy consumption of devices.

Some solutions:

Use low-power computing devices e.g. low–power-consumption processors, low-energy-rated LED monitors

Make the computing device more energy efficient by

1. adjusting the monitor settings (e.g. brightness, sleep)
2. power-down settings (e.g. set the hard drive to stop spinning after a period of time, set the Minimum Processor State which allows you to choose a percentage of power to allot to the processor when it is inactive or performing minimal tasks
3. Computers can also be left on standby when not in use to cut power consumption

## Security precautions

### Firewalls

A firewall is a network security system designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software, or a combination of both. Network firewalls are frequently used to prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets. All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.

Hardware and Software Firewalls

Firewalls can be either hardware or software but the ideal configuration will consist of both. In addition to limiting access to your computer and network, a firewall is also useful for allowing remote access to a private network through secure authentication certificates and logins.

Hardware firewalls can be purchased as a stand-alone product but are also typically found in broadband routers, and should be considered an important part of your system and network set-up. Most hardware firewalls will have a minimum of four network ports to connect other computers, but for larger networks, business networking firewall solutions are available.

Software firewalls are installed on your computer (like any software) and you can customize it; allowing you some control over its function and protection features. A software firewall will protect your computer from outside attempts to control or gain access your computer.

Some firewall software is free to download off the Internet e.g. ZoneAlarm

### Encryption

In cryptography, encryption is when data held on a computing system is put into a code. This means that if the data is hacked into, it is meaningless to the hacker unless they have the security key to unlock or decipher the encrypted data - only authorized parties who have the key can access it. Encryption does not of itself prevent interference, but denies the intelligible content to a would-be interceptor. In an encryption scheme, the intended information or message is encrypted using an encryption algorithm, generating information that can only be read if decrypted. It is theoretically possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, considerable computing power and skills would be required.