5.2 Computer languages
Programming languages
Low-level languages
High-level languages
Exam preparation

6 Security
6.1 Security threats
Data security
Security threats
Malpractice and crime
Online attacks

6.2 Security protection
Proof of identity
Firewalls
Security protocols
Encryption
Security examples
Exam preparation

7 Ethics
Ethics
Copyright
Free software
Hackers and crackers
Exam preparation

8 Programming
8.1 Introduction to programming
Introduction to Python
Algorithms

8.2 Begin coding
Output
Sequence
Input
Assign a value
Calculated values
Data types
Variables in pseudocode
Variables in flowcharts

8.3 Selection
Logical decision
Python if
Python if...else...
Python elif
Selection in pseudocode

8.4 Repetition
Loops
for loops

9 Solution development
9.1 Worked examples
Count how many
Calculate a total
Calculate an average
Verification
Validation
Validation algorithms

9.2 Testing and evaluation
Test data
Evaluation

9.3 Developing algorithms
Trace tables
Trace tables (loops)
Trace tables (flowcharts)
Analyse algorithms
Find errors in algorithms
Create an algorithm

9.4 Development methods
Top-down programming
Structure diagrams
Code libraries
Exam preparation

10 Databases
10.1 Database design
Records and fields
Data types
Primary key

10.2 Database queries
Select fields
Select records
Exam preparation

Appendices
Answers to knowledge test questions
Answers to exam preparation and exam-style questions
Data representation

This page summarises what you will learn about data representation. Tick the boxes on this page when you are confident you have learned each item.

1.1 Binary systems

**TICK THE BOX WHEN YOU HAVE LEARNED:**
- what binary data and binary numbers are
- what "bits" and "bytes" are
- what denary numbers are
- how position changes the value of a digit in a number
- how to count upwards in binary
- how to convert binary numbers into denary numbers
- how to convert denary numbers into binary numbers.

1.2 Hexadecimal

**TICK THE BOX WHEN YOU HAVE LEARNED:**
- what hexadecimal numbers are
- the 16 hexadecimal digits
- how to convert hexadecimal to denary
- how to convert denary to hexadecimal
- how to convert binary to hexadecimal
- how to convert hexadecimal to binary
- the advantages of hexadecimal
- the main uses of hexadecimal in computer science.

1.3 Data storage

**TICK THE BOX WHEN YOU HAVE LEARNED:**
- how text and numbers are stored on the computer
- the different ways to store image files
- the relationship between image quality and file size
- how sounds and video are stored on the computer
- file formats used for images, sounds and video
- the factors that improve the quality of audio and video
- the difference between lossless and lossy compression
- ways of compressing files and the effect on quality.
1.1 Binary systems

Binary data

A computer is an electronic machine. The data inside the computer is stored as electronic on/off signals. These binary signals can be used to represent many different forms of data. These signals are dynamic – they are always changing. By changing the electronic signals the computer transforms the data. This transformation is called processing. Everything that you will learn on a computer science course arises from these key facts.

Fact check

What is a computer? A computer is a machine for data processing. The computer processes data electronically. A computer stores and processes data using electrical switches. The switches can be on (conduct electricity) or off (no electricity).

Binary data: "Binary" means anything that can be in exactly two states, for example electrical switches that can be on or off. All data inside the computer is stored in binary form. We represent this data using 1s and 0s to stand for "on" and "off".

Forms of data: Binary can be used to represent many different forms of data:
- numbers, text
- images
- sound
- video.

Check your readiness

You should understand why computers are important and useful to us. What does binary mean? Can you explain why computer science students learn about binary numbers?

Bits and bytes

Data is held inside the computer in binary form. We represent the on/off signals using 1s and 0s (bits). Bits are organised into groups of eight called bytes. Computer memory and storage are measured in bytes. Active computer memory is also called RAM. A computer with large RAM can process more data, more quickly than one with less RAM.

Fact check

What are bits and bytes? A 1 or 0 is a "binary digit". This is shortened to "bit". Inside the computer, bits are organised in groups of eight. A group of eight bits is called a byte. When we write a binary number we write it as a complete byte. If there aren’t enough bits we add 0s at the start to make a complete byte:

0 0 0 1 1 1 0 1
Main memory: All data is held as electrical signals inside the computer. The main memory that holds data is called random access memory (RAM). We measure the size of RAM in bytes.

RAM and computer power: RAM is the active memory of the computer. Data can be processed quickly if it is held in RAM. In general, more RAM means a faster and more powerful computer.

Units of measurement: RAM and storage are measured in bytes. A kilobyte is 1024 bytes. A megabyte is 1024 kilobytes. A gigabyte is 1024 megabytes.

Registers: Data processing means data is changed and transformed. This is done by changing the electronic signals that make the data. At the time it is processed, this data is stored in a small area of memory called a register. A register holds just a few bits of data (e.g. 8 bit, 32 bit).

Check your readiness
Without looking at your notes or this page, answer these questions. How are bits organised inside the computer? How do we measure the capacity of RAM? What computer applications or games need really fast processing speed? Why does bigger RAM mean a faster computer?

Binary and denary

There are many different number systems. The number system we use in everyday life is called denary or base 10. The denary system has ten digits. Computers use the binary system, which has two digits. The value of a digit in either binary or denary depends on its position in the number.

Fact check
Denary numbers: Something that can be in ten different states is known as denary. Our normal number system is denary. We also call our number system "base 10" and "decimal". Denary numbers use ten digits:

0 1 2 3 4 5 6 7 8 9

Place value in denary: The same digit can represent different values in a denary number. The value of a digit depends on its position in the number. The value on the right is units. As you move left, each position is ten times bigger. In the denary number 3086 the 3 stands for three times a thousand.

<table>
<thead>
<tr>
<th>Thousands</th>
<th>Hundreds</th>
<th>Tens</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>

Place value in binary: Binary has only two digits (bits). The value of a bit depends on its position in the number. The value on the right is units. As you move left, each position is two times bigger.
In the binary number 1000, the 1 stands for one times eight.

<table>
<thead>
<tr>
<th>Eights</th>
<th>Fours</th>
<th>Twos</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Check your readiness**

Without looking at your notes or this page, answer these questions.

Can you explain the difference between denary and binary? Does your explanation include the number of digits and the place value of digits? Can you identify the value of a binary or denary digit by multiplying the digit by the place value?

**Fact check**

**Counting in denary:** There are ten denary digits (from 0 to 9). When you have counted up to 9 you have run out of digits.

9

Reset the units column to 0 and put 1 in the tens column.

10

Now continue in this way.

99

When you reach 99 you have run out of digits. Reset the tens and digits to 0 and put 1 in the hundreds column.

100

The largest number you can make with three digits is 999. What happens next?

**Counting in binary:** This is just the same as denary, except there are only two digits.

- Count 0
- Count 1

Now you have run out of digits. Reset the units column to 0 and put a 1 in the next column (the twos column).

- Count 1 0
- Count 1 1

You have run out of digits. Reset the units and twos to zero and put a 1 in the next column (the fours column).

**You need to know:**
- the rules of binary counting
- how to count upwards in binary.
**Writing binary number as bytes:** Remember that we often add 0s at the left of a binary number to make it up to eight bits (one byte).

**Check your readiness**
Without looking at your notes or this page, write out the complete series of 8-bit binary numbers starting at 0 0 0 0 0 0 0 0 and ending at 1 1 1 1 1 1 1 1.

**Convert binary to denary**

Each bit in a byte has a set value depending on its position. The position on the far right is worth 1. Work out the value of each position by multiplying by 2. This creates a table called the binary grid. To work out the value of a binary number put the bits into the binary grid. Add up all values that have a 1 underneath.

**Fact check**

**The binary grid:** This shows the position values of each bit in a byte. The bit on the right is worth 1. As you move to the left through the byte, each position is worth double the previous position.

<table>
<thead>
<tr>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
</table>

**Convert binary to denary:** Put the eight bits of a binary number into the eight columns of the binary grid. Note every value that has a 1 underneath it. Add these values together.

**If there are fewer than eight bits in the number:** Make sure the number is at the right of the table. Fill up the empty columns on the left with 0s.

**16-bit numbers:** You can convert a number with more than eight bits. The method is the same. Use this 16-column table.

<table>
<thead>
<tr>
<th>32 768</th>
<th>16384</th>
<th>8192</th>
<th>4096</th>
<th>2048</th>
<th>1024</th>
<th>512</th>
<th>256</th>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
</table>

You don’t have to memorise all the position values in this table. Just start on the right with 1 and multiply each column by 2 as you move to the left.

**Check your readiness**
Work on your own or with a friend. Write a series of eight bits, mixing 1s and 0s. Use the binary grid to work out the value of the number you have just written. Repeat this as many times as you can until conversion comes easily.
Convert denary to binary

You need to know:
- how to convert denary numbers to binary.

Fact check

The binary grid: To convert a denary number to binary we use the binary grid (position table). This eight-bit grid will convert a denary number from 0 to 255:

<table>
<thead>
<tr>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For larger numbers use the 16-bit grid.

Subtraction method: Start at the left of the binary grid (largest values). Find the largest value you can subtract from the number (without going below 0). Put a 1 into the grid below that value and subtract it from the number.

Keep subtracting values and putting 1s into the grid. Stop when you reach 0. Put 0s into any empty columns of the grid. Finally, show the bits without the grid.

Worked example: Convert 40 to binary

\[
\begin{align*}
40 &- 32 = 8 \\
8 &- 8 = 0
\end{align*}
\]

You have subtracted 32 and 8. Put 1 under these values. Fill in the rest of the grid with 0s.

<table>
<thead>
<tr>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Show the bits without the grid. The result is 0 0 1 0 0 0 0.

Check your readiness

Work on your own or with a friend. Write any denary number from 0 to 255. Convert it to binary, then back to denary, using the methods you have learned. You should get back to the number you started with. Repeat this until you can do it easily without errors.
1.1 Knowledge test
1. Computers transform data. What do they transform it into and why is this important to us?
2. What does “binary” mean?
3. How do binary numbers differ from denary numbers? Give as many differences as you can.
4. Why do computer science students learn about binary numbers?
5. How is a bit represented inside of RAM?
6. How are bits organised inside the computer?
7. What units do we use to measure the capacity of RAM?
8. Why does more RAM mean a faster computer?
9. Write out the sequence of binary numbers starting at 0 0 0 1 0 0 0 0 and ending at 0 0 0 1 0 1 1 0.
10. Fill in the missing position values in this binary grid.

<table>
<thead>
<tr>
<th>128</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>2</th>
<th>1</th>
</tr>
</thead>
</table>

11. Fill in the missing position values in this binary grid.

<table>
<thead>
<tr>
<th>32768</th>
<th>8192</th>
<th>4096</th>
<th>2048</th>
<th>512</th>
<th>256</th>
<th>128</th>
<th></th>
<th>2</th>
<th>1</th>
</tr>
</thead>
</table>

12. Translate these binary numbers into denary. Show your working.
   0 1 1 0 0 0 1 1
   0 0 1 1 1 1 0 0
   1 0 1 1 1 1 0 1
   1 1 0 0 1 1 0 1
   1 0 1 0 1 0 1 0 0 (this is more than one byte).

13. Translate these denary numbers into binary. Show your working.
   81
   106
   217
   290 (this is more than one byte)
   1010 (this is more than one byte).