Curriculum coverage
This chapter covers part or all of the requirements for the Computing Programme of Study (age 11–14) for England:

- understand simple Boolean logic (e.g. AND, OR and NOT) and some of its uses in circuits
- understand how instructions are stored and executed within a computer system.

This chapter also covers these main requirements for the Computing at School (CAS) Progression Pathways (for a full list of requirements met, see pages 9–10 of this handbook):

- recognise and understand the function of the main internal parts of basic computer architecture
- understand the concepts behind the fetch-execute cycle
- understand the von Neumann architecture in relation to the fetch-execute cycle, including how data is stored in memory
- understand the basic function and operation of location addressable memory
- understand the relationship between binary and electrical circuits, including Boolean logic.

Preparation
There is little requirement for specialist software in this chapter. Students will need to use a spreadsheet in Lesson 3.6. The Student Book gives examples using Microsoft Excel. If your school uses a different spreadsheet, you will need to make sure the spreadsheet operates in a similar way. Students also need access to a presentation package to complete the extension activity in the review section at the end of the chapter. The Student Book does not suggest a specific presentation package.

Each ‘Before the lesson’ section suggests ideas for your own introductory presentations.

Learning outcomes
By completing this chapter students will be able to:

- identify that the CPU contains switches
- explain how the CPU stores data in its memory
- understand simple Boolean logic using AND, OR and NOT logic gates
- understand that a computer uses the fetch-decode-execute cycle to process data
- explain how the CPU uses AND, OR and NOT gates to test data
- create a simple logic test using spreadsheet software
- create a simple truth table to test data.

Let us be logical

Talk about...
You can do the discussion activity offline. You could use this activity any time to vary the pace of lessons and encourage students to reflect on their learning.

It can be difficult to make topics such as Boolean logic and digital processors relevant to everyday life. They become more real and easier to understand if you can make them relevant to students’ interests. Here are some more discussion ideas.

- Many companies are testing driverless cars. Would you trust a car that is driven by a computer?
- Can you think of examples of Boolean logic a computer would use when driving a car? Here is an example to get you started: IF speed limit is 30 kph AND current speed of car is > 30 kph THEN reduce speed.
- You are playing a high-definition video game with realistic characters and multiple players. You have learned that millions of switches make up a computer processor. That is all a computer can use to run programs. Can you believe that a processor built with switches can really be running your game?
Computer processors are in many devices around the home. They control your washing machine, intruder alarm system, and your heating or air-conditioning system. Choose a device in your home. Suggest an example of an if... then... else statement the device’s processor might use. For example, IF wash cycle = ended AND spin cycle selected = yes THEN start spin cycle ELSE stop.

FACT
The Fact box explains that Apollo 11 made the first moon landing on 20 July 1969. Computers made sure the astronauts on board made it back to Earth safely, even though the early Apollo 11 project began without computers. Ask students whether they can imagine any activity today that does not involve computers.

Word cloud
The Word cloud contains all the key words that are highlighted and defined in Key words boxes in the lessons. The key words for this chapter are: ALU, computer memory, registers, fetch-decode-execute, sub-routines, Boolean logic, truth table, AND statement, OR statement, AND gate, NOT gate, OR gate and IF statement.
Learning outcomes
When they have completed this lesson students should be able to:

- explain how the CPU stores data in its memory.

More-confident students will:
- describe the difference between RAM and registers.

Overview
In this lesson students learn about the role that a CPU plays in a computer. They learn about the four main parts of a CPU: control unit, arithmetic and logic unit, registers and clock. Students also learn about the different types of memory that a computer uses, from small registers through to large disk drives. In the first part of the lesson students revise what they learned about the input, output and storage devices of a computer system.

Language development
In computer science we use the word ‘register’ to mean a small area of memory where data are recorded so that they can be accessed quickly. In the English language the word ‘register’ has other meanings. Schools use the word ‘register’ to mean a record of student attendance at classes. Most uses of the word ‘register’ are related to recording information. We register births and marriages in an official register. A football team can register a win or if the players are very good they might register a fifth consecutive win. We also send registered mail. The progress of a registered letter is recorded until it meets its final destination.

Before the lesson
Can you identify any old computer equipment in your school that can be made available for demonstration purposes? Talk to your school IT technician about any old equipment you may be able to use. If there is a decommissioned computer, can you open the case to reveal the motherboard? If so, you can point out where the CPU sits. You may be able to remove a CPU and RAM chips so that students can see what these look like. If there is a decommissioned disk drive, can the case be opened so that students can see the disk and the read/write head assembly inside?

If you cannot identify real examples of equipment in your school, find images online of a CPU, RAM chips and a disk drive. Use these images to illustrate the components discussed in the lesson. You may find a suitable animation or video that demonstrates how a disk drive works on YouTube. Try a Google search, such as ‘YouTube How a disk drive works’.

In the first part of the lesson students revise the knowledge they gained about the elements of a computer system in Matrix 2, Chapter 3, Data and the CPU. If you schedule this lesson in a computer room, students can identify whether each component in the room is an input, output or storage device. You may want to bring other devices into the room that are not normally located there. A scanner, headphones, microphone or webcam are some items you might use.

Read the Student Book to make sure you understand the content. This can be a difficult subject area for some students, as it is theoretical and, at times, abstract. You may need to provide additional support to some students to help their understanding.

The key words for this lesson are: ALU, computer memory and registers. The words are highlighted in the text the first time they appear. Their definitions are included in the Key words box at the end of the lesson. You may want to review these words before the lesson.

Learn about...
You will lead the first part of the lesson. Make sure students understand these ideas. You may ask them to make notes. You may use directed questioning to check understanding.

Inside the CPU
The Student Book describes the four important parts inside the CPU.
● The control unit receives instructions from a program and carries them out.
● The arithmetic and logic unit (ALU) is the calculator inside the CPU.
● The registers are small areas of memory that store an instruction and the data that instruction will use.
● The clock is a tiny chip that sends out regular electrical pulses just like the tick of a clock.

The Student Book shows these on page 65 in a highlighted box with detailed descriptions. The Student Book also shows how the parts of the CPU fit together. Remind students of these descriptions and ask them to study the diagram, which explains how buses (wire connections that move data around quickly) connect the parts.

Computer memory

The Student Book describes and compares the four main types of memory in a computer.

● The registers are small memory stores that are placed next to the CPU.
● The cache holds data that will be sent to the CPU in the near future.
● The random access memory (RAM) holds the program currently in use on a computer and the data the program is using.
● The disk drive stores large amounts of data permanently.

The Student Book shows these on page 66 in a highlighted box with detailed descriptions. The Student Book also uses a diagram to show how the further away memory is located from the CPU, the slower and cheaper it gets. It also shows a table listing the typical storage sizes for different types of memory. Remind students of these descriptions and ask them to study the diagram and table (on page 66).

Now you do it...

Students think about the different methods they can use to remember facts. Students create a list of the various ways in which they can remember a birthday. They order the methods by the amount of time it takes to recall the information.

What success looks like: Students recognise that there are many ways of remembering a birthday, some of which mean they record information. They make a list of the methods, ordered from fastest to slowest.

If you have time...

Students conduct research using the Internet to find the difference between RAM and registers in a computer.

What success looks like: Students use the Internet search skills they have developed elsewhere in the Matrix series to find sites that identify the difference between RAM and registers. Students identify several key differences, including the following.

● Registers can be accessed directly by the CPU, RAM cannot.
● Registers are built directly onto the CPU board.
● RAM contains more memory than registers.
● Registers are quicker to access than RAM.
Test yourself...

FOUNDATION QUESTIONS
1 List the four main parts of the CPU. Answer: Control unit, arithmetic and logic unit (ALU), registers, clock.
2 What part of the CPU carries out calculations? Answer: The ALU carries out all calculations and logical operations.

EXTENSION QUESTIONS
3 Why does the CPU need memory? Answer: The CPU needs memory to store the instructions that it uses, to store data temporarily while calculations take place and to store the results of calculations.
4 Why doesn’t the CPU get data directly from the computer’s disk drive? Answer: It would take too long. A disk drive is much slower than the cache or RAM. The registers in the CPU are the fastest memory of all.

Fetch-decode-execute cycle

Learning outcomes
When they have completed this lesson students should be able to:
agog understand that the computer uses the fetch-decode-execute cycle to process data.
More-confident students will:
agog describe how multi-core processors can make computers operate faster.

Overview
This lesson describes how the CPU uses the fetch-decode-execute cycle to carry out instructions. Students learn what takes place during each of the fetch, decode and execute phases of the cycle. Students also learn about different types of memory used in a computer system. They learn that the registers in a CPU are small areas of memory that can be accessed very quickly. Disk drives are slower than registers but provide more memory capacity for long-term storage of data files.
Before the lesson

Prepare a slide using a presentation package on the structure of the CPU diagram shown in the ‘Learn about... ’ section on page 68 of the Student Book. Project the image onto the board to support a class demonstration of the fetch-decode-execute cycle during the ‘How to... ’ part of this lesson. Annotating the diagram using marker pens during the lesson will support a more interactive demonstration. As an alternative, draw the CPU diagram on the board before the lesson or use a large sheet of paper.

Read the content of the ‘How to... ’ section before the lesson. Your demonstration should follow the steps laid out in the Student Book so that students are prepared for the ‘Now you do it... ’ activity. The key words for this lesson are: fetch-decode-execute cycle and sub-routines. The words are highlighted in the text the first time they appear. Their definitions are included in the Key words box at the end of the lesson. You may want to review these words before the lesson.

Learn about...

You will lead the first part of the lesson. Make sure students understand these ideas. You may ask them to make notes. You may use directed questioning to check understanding.

- The CPU: The main components of the CPU are the control unit, ALU, registers and clock. Data are loaded into the registers from cache memory and are moved around the CPU along buses.

- Fetch-decode-execute cycle: When an instruction is carried out it is first fetched from the cache along with any data that are needed for the instruction to be completed. The instruction is then decoded. That is, the CPU converts the instruction into a format it can use. It then stores the instruction and data in the registers. Finally, the instruction is executed by the ALU and the result is sent back to a register for storage.

- Machine code: The language the CPU can understand is machine code. Machine code is written entirely in binary. Any instruction must be decoded into machine code before it can be executed.

How to...

In the second part of the lesson students complete an exercise under your guidance.

This section in the Student Book provides a step-by-step demonstration of a simplified fetch-decode-execute cycle. The demonstration prepares students for the ‘Now you do it... ’ activity in which pairs of students carry out a simulation. One student plays the role of the control unit and the other plays the role of the ALU. Students must understand the fetch-decode-execute cycle and the roles of the control unit and ALU during the cycle.

Run a whole-class demonstration of the fetch-decode-execute cycle to help students understand the activity’s requirements. Use a diagram such as the one on page 68 of the Student Book. Prepare two simple addition instructions to demonstrate. Demonstrate the three steps of the fetch-decode-execute cycle as detailed on pages 69 and 70 of the Student Book. Lead students through the first demonstration. Run a second demonstration using a session of questions and answers so that you can correct any misunderstanding.

Now you do it...

Students work in pairs to run a simulation of an instruction moving through the fetch-decode-execute cycle in a CPU. One student carries out the tasks of the control unit, while the other student plays the role of the ALU. Students swap roles for a second run of the simulation.

What success looks like: Students demonstrate an understanding of the role of the control unit, ALU and registers. They demonstrate this understanding by correctly running a simulation of the fetch-decode-execute cycle. Each student explains what his or her role was in the simulation and identifies any improvements needed for another performance of the simulation.

If you have time...

Students use the Internet to research what is meant by dual core and quad core processor. Students will discover why multi-core processors make computers run faster.
What success looks like: Students explain that a dual core processor has two CPUs working in parallel on the same chip. A quad core processor has four CPUs working in parallel. Students explain that the computer using a dual core processor runs faster than one with a single CPU. This is because, with a dual core processor, two CPUs share the work.

Test yourself...

FOUNDATION QUESTIONS

1. What is machine code? Answer: Machine code is a computer program that is written in the computer’s own language, which is binary. The computer recognises some binary codes as instructions that the CPU can execute.

2. What parts of the fetch-decode-execute cycle are performed by the ALU? The ALU performs the execute part of the cycle after the control unit has fetched from memory and decoded the data.

EXTENSION QUESTIONS

3. Why don’t programmers normally write programs in machine code? Answer: Machine code is written in the computer’s language, which is binary. Binary is hard for people to understand and makes it difficult to write code and to identify where there are problems.

4. Describe the fetch-decode-execute cycle using an everyday example, such as preparing a meal. Answer: When preparing a meal (example initial steps):
   - open a recipe book and chose the recipe you are going to use (load program into RAM)
   - read the first instruction—chop a medium onion (fetch)
   - work out what is needed to carry out the instruction—chopping board, knife and onion (decode)
   - carry out the instruction—chop the onion (execute).
Learning outcomes
When they have completed this lesson students should be able to:
- identify that the CPU contains switches
- understand simple Boolean logic using AND
- create a simple truth table to test data.
More-confident students will:
- identify everyday events that can be expressed in Boolean logic.

Overview
In this lesson students learn that computers are able to process logic statements as well as solve arithmetical problems. Computers can process logic because the false or true state of a Boolean logic statement is binary and can be represented by the 0 or 1 of the binary number system. Students learn how to write simple logic statements and how to lay out the statements in the form of a logic table.

Language development
The word ‘then’ is an adverb that is frequently used in the English language. In logic statements ‘then’ is used to mean ‘therefore’ or ‘in that case’. We might say, ‘If we don’t leave now, then we will be late for school’.

One of the most common uses of ‘then’ is when we want to say ‘at that time’. We say, ‘I was not available then’. A similar use for ‘then’ is when we want to say ‘after that’. We say, ‘I made a cup of tea then took it to my mother’.

Before the lesson
Think of some examples of logic statements that you can use in your introduction to the ‘How to...’ part of this lesson. Realistic logic statements can be difficult to think of quickly. Therefore, having a few already prepared will help the lesson run smoothly.
The activities ask for examples from computer games or the school day. Having examples prepared from those areas will help students who are struggling to create their own examples.
The key words for this lesson are: Boolean logic and truth table. The words are highlighted in the text the first time they appear. Their definitions are included in the Key words box at the end of the lesson. You may want to review these words before the lesson.

Learn about...
You will lead the first part of the lesson. Make sure students understand these ideas. You may ask them to make notes. You may use directed questioning to check understanding.
- **ALU:** The ALU carries out two types of operation, arithmetical and logical. Students learned about arithmetic operations in Lesson 3.2. In this lesson students learn about logical operations. They may be less familiar with logic than the type of basic arithmetic used to demonstrate arithmetical operations in Lesson 3.2. Computer games provide an excellent source of examples of logical and arithmetical operations, which should engage students.
- **Boolean logic:** The word ‘Boolean’ is used in computing to describe a statement that can either be true or false. This is significant in computing because logic problems are stated in a binary form (false or true). This means logic problems can be represented by 0 or 1 in the binary number system. That explains how logic can be processed by a computer.

How to...
In the second part of the lesson students complete an exercise under your guidance. Prepare two or three examples you can use to demonstrate the process of defining a logic statement and creating a truth table. Use a session of questions and answers to involve students and make sure that the class understands the process. You can then move on to the ‘Now you do it...’ activity.
Now you do it...

Students think about a computer game they play and identify two statements about the game that are logically linked. Students are given an example that they replace with a statement that reflects their own experience of a game. If students prefer, they can choose an example from any aspect of their lives. It need not be a computer game. When they have a suitable example, students combine their statements into a single logic statement and create a truth table.

What success looks like: Students create a realistic logic statement in which there is a clear logical relationship between the proposition and conclusion, such as ‘Crossed the finish line first THEN won the race’. Students create a truth table with the proposition in the left-hand column and the conclusion in the right-hand column. The columns are correctly labelled and true/false values are correctly entered.

If you have time...

In this activity students think about their school day and write three logic statements that describe events that occur within the day, such as ‘Time is 3:30 THEN go home’.

What success looks like: Successful students are able to analyse everyday events and write them as logic statements. The logic statements they create are clearly written and there is a link between the proposition and conclusion.

Test yourself...

FOUNDATION QUESTIONS

1. What does ALU stand for? Answer: Arithmetic and logic unit.
2. Where would you find an ALU? The ALU is a part of the computer’s central processing unit (CPU).

EXTENSION QUESTIONS

3. Why do we use a truth table to describe a logic problem? Answer: A truth table makes it easier to read and understand logic statements. All the possible outcomes are clearly laid out.
4. Why can we use Boolean logic with computer processors? Answer: Boolean logic can be processed by computers because the false or true value of a Boolean statement can be represented by the value 0 or 1 in binary.
Overview
In this lesson students learn to write logic statements that include two parts in the proposition part of the statement. Students learn to use AND and OR operators to join two statements in the proposition. They develop skills in creating truth tables for compound logic statements.

Before the lesson
Think of some examples of logic statements that you can use in your introduction to the ‘How to...’ part of this lesson. Realistic logic statements can be difficult to think of quickly, so having a few prepared will help the lesson run smoothly. The activities in the lesson ask for examples from computer games or the school day. Having examples prepared from those areas will help students who are struggling to create their own examples.

The key words for this lesson are: AND statement and OR statement. The words are highlighted in the text the first time they appear. Their definitions are included in the Key words box at the end of the lesson. You may want to review these words before the lesson.

Learn about...
You will lead the first part of the lesson. Make sure students understand these ideas. You may ask them to make notes. You may use directed questioning to check understanding.

● **Compound logic statements:** Simple logic statements are easy to write and understand, but in real life they are not very useful. Logic statements become useful when they describe situations where there is more than one factor to consider. While logic statements always have only one conclusion they can have two or more parts in the proposition.

  - If there are two parts to a proposition the parts must be joined together with either an AND or an OR operator. AND is used if both statements need to be true before the conclusion is also true. OR is used if either statement needs to be true before the conclusion is also true.

How to...
In the second part of the lesson students complete an exercise under your guidance. Have two or three examples prepared that you can use on the board in a class demonstration. Make sure some examples use the AND operator and others use the OR operator. Students need to be clear about these areas.

  - They need to understand when to use the AND operator and when to use OR. If AND is used, both statements in the proposition need to be true before the conclusion is true. If OR is used, the conclusion is true if either of the statements in the proposition is true.

  - When creating a truth table, students need to ensure that they enter all combinations of false/true. In your demonstration, draw a parallel between false/true and the binary values 0 and 1. A two-part statement will always have four lines in its truth table. This becomes clear if you draw a parallel with binary. A two-digit binary number can hold four values. Those values are the binary equivalents of 0 to 3 in decimal: 00, 01, 10 and 11. If the zeros are replaced with ‘false’ and the ones with ‘true’ you have all the possible combinations for your truth table: false/false, false/true, true/false and true/true.

Learning outcomes
When they have completed this lesson students should be able to:

- understand simple Boolean logic using AND and OR logic gates
- create a truth table for a logic statement where the proposition contains either AND or OR.

More-confident students will:

- work independently to create a truth table.
Now you do it...

In this activity students work in pairs to play the game called Twenty Questions. A student thinks of the name of an object or person. The partner can ask 20 questions in order to guess the object or person.

What success looks like: Successful students complete the game by guessing the object or person. They recognise that they have demonstrated the use of logic to solve a practical problem.

If you have time...

Students consider two examples of logic statements that describe everyday situations. They draw a truth table for each statement.

What success looks like: Successful students create a truth table that correctly describes both logic statements provided. The truth table for the OR statement in the first statement and the AND statement in the second statement are correctly completed. The tables are neatly laid out with a clear heading for each column.

Test yourself...

FOUNDATION QUESTIONS

1. What is the missing word in this logic statement?
   Monday _____ not a holiday THEN go to school.
   Answer: AND

2. What is the missing word in this logic statement?
   Lives is zero _____ player quits THEN game over.
   Answer: OR

EXTENSION QUESTIONS

3. What is the part of a logic statement that follows THEN called? Answer: It is the conclusion.

4. If you have a logic statement with three parts to the proposition (e.g. a AND b AND c THEN d), how many possible combinations of true/false will there be?
   Answer: There are eight possible combinations for a three-part proposition. (Think of the problem as a three-digit binary number that has eight possible values.)
Overview
In this lesson students take their knowledge about Boolean logic and apply it to logic gates. Students find out how to recognise logic gate symbols. They also learn how to construct truth tables to describe how those gates operate.

Language development
In the English language, a gateway is an opening in a wall or fence that allows us to pass through. A gate is a barrier that closes the opening and prevents us from passing. This can be confusing. Does a gate allow us to pass through an obstacle or prevent us from passing? The answer is that a gate allows us to pass through an obstacle if we know how to open that gate. In this lesson students discover what conditions open a logic gate in a CPU. If both inputs to an AND gate are set to 1, then the gate is opened and the output is also 1. Understanding this process is like knowing the code to a security gate.

Before the lesson
In this lesson students apply the skills they developed in Lesson 3.4 to a new problem: logic gates. The skills they need are exactly the same. Prepare a presentation to use in your introduction to the ‘How to...’ part of the lesson. Emphasise the similarity between the tasks.

Your presentation could start with the AND truth table you created for a Boolean logic statement in Lesson 3.4. As an alternative, you can prepare a slide containing a truth table similar to that on page 77 of the Student Book (the umbrella example).

A second slide could contain an image of an AND gate above the same truth table. The truth table could be modified with the original headings crossed out and replaced with: ‘input a’; ‘input b’ and ‘output’. In the body of the table show each instance of ‘false’ crossed out and replaced with a 0. Also show each instance of ‘true’ replaced with a 1.

The key words for this lesson are: AND gate, NOT gate and OR gate. The words are highlighted in the text the first time they appear. Their definitions are included in the Key words box at the end of the lesson. You may want to review these words before the lesson.

Learn about...
You will lead the first part of the lesson. Make sure students understand these ideas. You may ask them to make notes. You may use directed questioning to check understanding.

- Logic gates: Individual switches in a CPU can be combined to make logic gates. It is logic gates that allow the CPU to perform logical operations on data.
- Types of logic gate: The three basic types of logic gate are AND, OR and NOT. All logic gates have just one output. All logic gates have two inputs with the exception of NOT, which only has one.

How to...
In the second part of the lesson students complete an exercise under your guidance.

Use the slides you prepared before the lesson to demonstrate how the truth tables students constructed in Lesson 3.4 apply to logic gates in a CPU. Emphasise how the statements in a proposition are replaced by the two inputs to an AND or OR gate. The conclusion is replaced by the output of the gate. Drawing this comparison shows
that the logic gates in a computer can be used to solve everyday logic problems. Use a session of questions and answers in your demonstration to involve students and to check their understanding.

**Now you do it...**

Students draw a truth table for a NOT gate. In pairs, students then play the game called Opposites to demonstrate their understanding of the action of a NOT gate.

**What success looks like:** Students create a truth table for a NOT gate. The truth table is correctly drawn and neatly laid out. When playing the game, students demonstrate they can work together to build the longest number of opposites they can. Success in the game depends on the first student in the pair providing a prompt that has a binary response. For example, ‘start’ has the opposite response ‘stop’.

**If you have time...**

Students use the Internet to explore three more logic gates: XOR, NOR and NAND. Students find a diagram and provide a brief explanation of what each gate does. Students also provide an everyday example of how XOR is different from OR.

**What success looks like:** Students provide a correct diagram and brief description for XOR, NOR and NAND similar to these points.

- **XOR:** This is a logic gate that will output 1 only if one or the other (not both) inputs are 1.
- **NOR:** This is a logic gate that gives the value 1 if both inputs have a value of 0. Any other combination of inputs has a value of 0.
- **NAND:** This is a logic gate that gives the value 0 if both inputs have a value of 1. Any other combination of inputs has a value of 1.

**Note:** In the case of NAND and NOR, students may say that NAND, for example, is the opposite of AND or that it is NOT AND. That is an acceptable answer. In the case of XOR, students may say that it is like OR, but where both inputs are 1 the output is 0.

An example of an XOR gate in real life is a room with two light switches. If the lights are off, flicking either of the switches will turn the lights on. Flick both and the lights stay off.

**Test yourself...**

**FOUNDATION QUESTIONS**

1. How many outputs do gates have? Answer: All gates have just one output.

2. If the input of a NOT gate is 0, what is the output? Answer: NOT reverses the input so if the input is 0, the output is 1.

**EXTENSION QUESTIONS**

3. If both inputs to an OR gate are 0, what is the output? Answer: If both inputs to an OR gate are 0 then the output is also 0.

4. Sketch the flowchart symbols for the AND, OR and NOT gates. The shapes of the three symbols have something in common. What is it and why do you think they share that feature? Answer: Students draw the AND, OR and NOT gates correctly. Refer to the Student Book pages 81 and 82 if students are unsure. What the shapes have in common is that they all have the inputs entering from the left and the output exiting to the right. Students might say they all ‘point’ to the right, which is correct. All gates flow to the right because the output from a gate can be the input to another gate.
Overview
In this lesson students learn how Boolean logic is applied in practical situations. So far students have found out how to describe real-life problems in Boolean logic. They have also discovered how to draw truth tables to show the relationship between two logical statements. In this lesson students learn how computers use logic in ‘if... then... else’ statements to make decisions that affect real processes. Students use a spreadsheet program to demonstrate the practical application of logic through an ‘if... then... else’ statement.

Before the lesson
Students need to use a spreadsheet in this lesson. The Student Book uses Microsoft Excel for demonstration purposes. If Microsoft Excel software is available on your school network, check that students have permission to use the application before the lesson starts. If your school uses a different spreadsheet application, run through the ‘How to...’ exercise yourself and identify any differences in the way the application operates. Tell students of any differences that affect the way they will need to work to complete the lesson.

Prepare the spreadsheet that students will build in the ‘How to...’ part of the lesson. Enter a text box beneath the table created in the ‘How to...’ part of the lesson that contains an example of the formula =IF(AND(B6 = 1,C6 = 1), 1, 0). Put this formula in as large a font size as you can. Refer to this formula when you demonstrate the spreadsheet during the ‘How to...’ part of the lesson.

The key word for this lesson is: IF statement. The word is highlighted in the text the first time it appears. The definition is included in the Key words box at the end of the lesson. You may want to review the meaning of IF statement before the lesson.

Learn about...
You will lead the first part of the lesson. Make sure students understand these ideas. You may ask them to make notes. You may use directed questioning to check understanding.

● ‘If... then... else’: This is a statement used in programming languages, spreadsheets and other application programs that support complex programming. An IF statement uses a Boolean logic test to decide between two actions. The two alternative actions are represented in the ‘then’ and ‘else’ parts of the statement. The syntax of an ‘if... then... else’ statement can vary. Programming languages use the words if before a logical test and then and else before two actions. Spreadsheet syntax replaces ‘then’ and ‘else’ with commas.

How to...
In the second part of the lesson students complete an exercise under your guidance.

Students complete a well-structured exercise using a spreadsheet. The Student Book uses Microsoft Excel for demonstration purposes. If students use another spreadsheet tell them if the package they are using varies from Microsoft Excel.

You may want to include a demonstration in your introduction to this section. It may be that students have some experience of using spreadsheets on other courses. You will need to judge how much detail to include in your introduction based on students’ previous experience with spreadsheets. This is what you can do at the very least.

● Show students what the final spreadsheet looks like and how it works. It is important that they know what they are aiming to achieve. Seeing the finished spreadsheet will help.

Learning outcomes
When they have completed this lesson students should be able to:

➔ create a simple logic test using spreadsheet software
➔ explain how the CPU uses AND, OR and NOT gates to test data.
More-confident students will:
➔ explain how the CPU uses a NAND gate to test data.

3.6 Logic gate simulator

Logic gate simulator

pages 84–87

3.6

Learning outcomes
When they have completed this lesson students should be able to:

➔ create a simple logic test using spreadsheet software
➔ explain how the CPU uses AND, OR and NOT gates to test data.

More-confident students will:
➔ explain how the CPU uses a NAND gate to test data.

Overview
In this lesson students learn how Boolean logic is applied in practical situations. So far students have found out how to describe real-life problems in Boolean logic. They have also discovered how to draw truth tables to show the relationship between two logical statements. In this lesson students learn how computers use logic in ‘if... then... else’ statements to make decisions that affect real processes. Students use a spreadsheet program to demonstrate the practical application of logic through an ‘if... then... else’ statement.

Before the lesson
Students need to use a spreadsheet in this lesson. The Student Book uses Microsoft Excel for demonstration purposes. If Microsoft Excel software is available on your school network, check that students have permission to use the application before the lesson starts. If your school uses a different spreadsheet application, run through the ‘How to...’ exercise yourself and identify any differences in the way the application operates. Tell students of any differences that affect the way they will need to work to complete the lesson.

Prepare the spreadsheet that students will build in the ‘How to...’ part of the lesson. Enter a text box beneath the table created in the ‘How to...’ part of the lesson that contains an example of the formula =IF(AND(B6 = 1,C6 = 1), 1, 0). Put this formula in as large a font size as you can. Refer to this formula when you demonstrate the spreadsheet during the ‘How to...’ part of the lesson.

The key word for this lesson is: IF statement. The word is highlighted in the text the first time it appears. The definition is included in the Key words box at the end of the lesson. You may want to review the meaning of IF statement before the lesson.

Learn about...
You will lead the first part of the lesson. Make sure students understand these ideas. You may ask them to make notes. You may use directed questioning to check understanding.

● ‘If... then... else’: This is a statement used in programming languages, spreadsheets and other application programs that support complex programming. An IF statement uses a Boolean logic test to decide between two actions. The two alternative actions are represented in the ‘then’ and ‘else’ parts of the statement. The syntax of an ‘if... then... else’ statement can vary. Programming languages use the words if before a logical test and then and else before two actions. Spreadsheet syntax replaces ‘then’ and ‘else’ with commas.

How to...
In the second part of the lesson students complete an exercise under your guidance.

Students complete a well-structured exercise using a spreadsheet. The Student Book uses Microsoft Excel for demonstration purposes. If students use another spreadsheet tell them if the package they are using varies from Microsoft Excel.

You may want to include a demonstration in your introduction to this section. It may be that students have some experience of using spreadsheets on other courses. You will need to judge how much detail to include in your introduction based on students’ previous experience with spreadsheets. This is what you can do at the very least.

● Show students what the final spreadsheet looks like and how it works. It is important that they know what they are aiming to achieve. Seeing the finished spreadsheet will help.
• Explain the format of the IF function. Explain the three parts (‘if’, ‘then’ and ‘else’). Use the formula in the demonstration as your example.
• Explain the format of the AND function. Once again, use the formula in the demonstration as your example.

Now you do it...
Students use a spreadsheet package to create a logic gate simulator to create a truth table for an OR gate. The simulator uses an IF statement to calculate the correct outputs for two given inputs. Students model their answer on the AND gate simulator created in the ‘How to... ’ part of the lesson.

What success looks like: Students correctly create a logic gate simulator to create a truth table for an OR gate. The Student Book shows the completed OR gate truth table.

If you have time...
Students are challenged to adapt the AND simulator they built in a spreadsheet in the ‘How to... ’ part of the lesson. They adapt it so that it becomes a NAND simulator. This shows that they understand how a NAND gate works. Students who carried out the extension activity in Lesson 3.5 will be prepared for this challenge.

What success looks like: Students adapt the AND simulator they built previously so that it simulates a NAND gate. They use the Help function in the spreadsheet to establish how to use the NOT function. Students adapt their AND spreadsheet correctly and test it. Testing requires them to draw a manual NAND truth table to check the results displayed in the spreadsheet.

Test yourself...

FOUNDATION QUESTIONS
1. How would you find cell C5 in a spreadsheet? Answer: Cell C5 can be found where column C meets row 5.
2. What tells us that an entry in a spreadsheet is a formula? Answer: A spreadsheet formula always starts with an equals sign ‘=’.

EXTENSION QUESTIONS
3. You used this formula in your spreadsheet: =IF(AND(B6=1,C6=1),1,0). Explain what each part of the formula means. Answer:
   ◦ =IF: Checks if the logic statement is true or false.
   ◦ AND(B6=1, C6=1): This is the logic statement. The statement is true if cell B6=1 and cell C6=1. Otherwise the logic statement is false.
   ◦ ,1: This is the action taken if the logic statement is true. A 1 is entered in the cell containing the formula.
   ◦ ,0: This is the action taken if the logic statement is false. A 0 is entered in the cell containing the formula.
4. How does a spreadsheet simulation help us understand logic gates? Answer: A spreadsheet simulation of a logic problem allows us to enter data to test the problem. We see the outputs as we enter the input data.
The test questions and assessment activities give you an opportunity to evaluate students’ understanding. The questions are shown here with possible answers.

**Model answers to test questions**

1. What do the letters ALU stand for? Answer: Arithmetic and logic unit.

2. Why does the CPU need memory? Answer: The CPU needs memory to store the instruction it is currently executing and to store the data needed to carry out the instruction.

3. What is the role of the control unit in the fetch-decode-execute cycle? Answer: The control unit receives instructions from a program, decodes the instructions and carries them out. It controls all the other parts of the CPU. The control unit is the engine of the computer.

4. Where are instructions stored before being loaded into the CPU? Answer: Instructions are stored in an area of memory called the cache before being loaded to the CPU.

5. Why do we use Boolean logic with computer processors? Answer: In Boolean logic a statement can be either false or true. That can be represented in binary by 0 and 1 in the CPU.

6. How many inputs are there to the AND, OR and NOT gates? Answer: The AND and OR gates have two inputs. The NOT gate has only one input.

7. Draw a truth table for an AND gate.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

8. Draw a truth table for an OR gate.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

9. Describe what a NOT gate does. Answer: a NOT gate reverses the input. If the input to a NOT gate is 0 the output is 1.

10. Describe the fetch-decode-execute cycle. Answer:

   - The control unit of the CPU fetches an instruction from the cache.
   - The control unit decodes the instruction so that it can be processed and stores the instruction, and the data the instruction needs, in the CPU registers.
   - The control unit instructs the ALU to execute the instruction.
   - The ALU executes the instruction and stores any result in a register.
   - The control unit fetches the next instruction.

**Model answers to assessment activities**

**Starter activity**

All students should be able to complete this activity. Students draw a gate symbol and truth table for AND, OR and NOT gates. They use the tables to answer the questions ‘When is the output 1 for each gate?’ and ‘How could you reverse the output for the AND gate to create a NAND gate?’

**What success looks like:** Students draw the correct gate symbol for each gate. They construct a truth table that is correct and neatly laid out. If the truth tables are correct, successful students provide these responses to the questions: ‘When is the output a 1 for each gate?’ Answer:

- AND gate output is 1: where both inputs a and b are 1.
- OR gate output is 1: where inputs a and b are 10, 01, or 11.
- NOT gate output is 1: where the input is 0.

‘How could you reverse the output for the AND gate to create a NAND gate?’ Answer: You pass the output of the AND gate to the input of a NOT gate.

**Intermediate activity**

Students compare two computers, one a cheaper low-specification PC and the other a more expensive high-specification PC. Students compare several
factors that influence the performance of a computer (clock speed, cache size, processor specification). Students also compare the two specifications. They highlight the factors that will make one computer perform better than the other.

Do some research before this activity to identify an online site that presents the information students are asked to identify. Pointing students to a recommended site will make the activity run smoothly.

**What success looks like:** Students identify two suitable devices to compare. They gather the information they are asked to find and lay it out clearly in a table format. Their conclusions refer to the findings in the table. Students correctly identify at least three factors that are likely to make one computer perform better than the other. Those factors include: the number of processors in the CPU (the more expensive device is likely to have four processors, for example); clock speed is likely to be faster in the more expensive device; the more expensive device is likely to have more cache memory and more RAM. Students identify at least one general performance factor besides those they are specifically asked to identify. For example, they comment on screen specification or range of storage devices.

**Extension activity**

In this activity students prepare a short presentation that explains the steps in the fetch-decode-execute cycle. The presentation format is specified in terms of the number and content of the pages.

**What success looks like:** Students create a presentation that meets the specification in terms of number of pages and the content of those pages. They demonstrate their understanding of the fetch-decode-execute cycle in a clear and concise way.