my revision notes

WJEC and EDUQAS GCSE

COMPUTER SCIENCE

Ian Paget
Robert Wicks

HODDER EDUCATION
LEARN MORE

The Publishers would like to thank the following for permission to reproduce copyright material.

Photo credits
P.42 (left) taken from Greenfoot software, **(right)** Copyright 1991-1995 by Stichting Mathematisch Centrum, Amsterdam, The Netherlands; **p.61** © NASA Photo/Alamy Stock Photo; **p.80** taken from Greenfoot software.

Acknowledgements
Every effort has been made to trace all copyright holders, but if any have been inadvertently overlooked, the Publishers will be pleased to make the necessary arrangements at the first opportunity.

Although every effort has been made to ensure that website addresses are correct at time of going to press, Hodder Education cannot be held responsible for the content of any website mentioned in this book. It is sometimes possible to find a relocated web page by typing in the address of the home page for a website in the URL window of your browser.

Hachette UK's policy is to use papers that are natural, renewable and recyclable products and made from wood grown in sustainable forests. The logging and manufacturing processes are expected to conform to the environmental regulations of the country of origin.

Orders: please contact Bookpoint Ltd, 130 Park Drive, Milton Park, Abingdon, Oxon OX14 4SE. Telephone: (44) 01235 827720. Fax: (44) 01235 400401. Email education@bookpoint.co.uk. Lines are open from 9 a.m. to 5 p.m., Monday to Saturday, with a 24-hour message answering service. You can also order through our website: www.hoddereducation.co.uk

ISBN: 978 1 5104 5493 4

First published in 2018 by
Hodder Education,
An Hachette UK Company
Carmelite House
50 Victoria Embankment
London EC4Y 0DZ
www.hoddereducation.co.uk

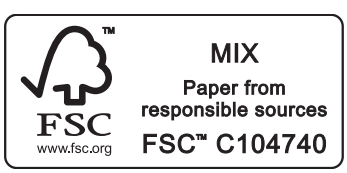
Impression number 10 9 8 7 6 5 4 3 2 1

Year 2023 2022 2021 2020 2019

Typeset in Bembo Std Regular 11/13pt by Aptara Inc.

Printed in Spain

A catalogue record for this title is available from the British Library.



Get the most from this book

Everyone has to decide his or her own revision strategy, but it is essential to review your work, learn it and test your understanding. These Revision Notes will help you to do that in a planned way, topic by topic. Use this book as the cornerstone of your revision and don't hesitate to write in it — personalise your notes and check your progress by ticking off each section as you revise.

Tick to track your progress

Use the revision planner on pages iv, v and vi to plan your revision, topic by topic. Tick each box when you have

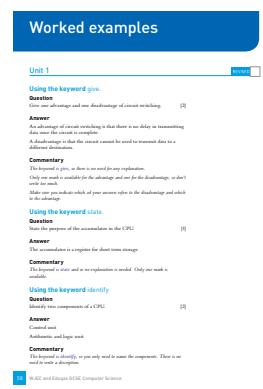
- revised and understood a topic
- tested yourself
- practised the exam questions and gone online to check your answers and complete the quick quizzes

You can also keep track of your revision by ticking off each topic heading in the book. You may find it helpful to add your own notes as you work through each topic.


My revision planner

Introduction
viii Using this book
viii How to revise
viii Examination technique
x Finally...

Unit 1
1 Hardware
01 The Von Neumann model
01 Central processing unit
03 The fetch–decode–execute cycle
04 Factors affecting processor performance
05 Input and output
08 Data capacity and storage requirements
2 Logical Operations
10 Logical operators
11 Truth tables
13 Boolean logic
13 Simplifying Boolean expressions using Boolean identities and rules
3 Communication
15 Networks
18 Servers
18 Wired and wireless networks
18 Network hardware
20 Protocols
21 Layers
22 Routing
22 IP addresses
4 Organisation and structure of data
24 Representation of numbers
27 Arithmetic shift functions
28 Overflow
28 Adding binary numbers
28 Representation of graphics and sound
29 Metadata
30 Storage of characters
30 Data types
30 Data structures
32 Data validation and verification
33 Algorithms
5 System Software
34 The operating system
36 Utility software
6 Principles of programming
39 Computer languages
7 Software engineering
8 Program construction
44 Assembler
44 Compiler
45 Interpreter
45 The process of compilation
44 Programming errors
9 Security and data management
47 Data Security
47 Data management
48 Compression
49 Network security
49 Policies
50 Cybersecurity
51 Protecting software during production
51 Internet cookies
10 Ethical, legal and environmental impacts
52 Computer crime
52 E-commerce
53 Social Impact of digital technology
54 Standards
54 Legislation
56 Environmental issues
Worked examples


Worked examples

Unit 1

Using the keyword give.
Question
Give one advantage and one disadvantage of circuit switching. [2]
Answer
An advantage of circuit switching is that there is no delay in transmitting data once the circuit is complete.
A disadvantage is that the circuit cannot be used to transmit data to a different destination.
Commentary
The keyword is give, so there is no need for any explanation.
Only one mark is available for the advantage and one for the disadvantage, so don't write too much.
Make sure you indicate which of your answers refers to the disadvantage and which to the advantage.

Using the keyword state.
Question
State the purpose of the accumulator in the CPU. [1]
Answer
The accumulator is a register for short term storage.
Commentary
The keyword is state and so no explanation is needed. Only one mark is available.

Using the keyword identify
Question
Identify two components of a CPU. [2]
Answer
Control unit
Arithmetic and logic unit
Commentary
The keyword is identify, so you only need to name the components. There is no need to write a description.

Features to help you succeed

Key terms

A number of key terms are used and defined for you as you work through this guide. Make sure you learn them so that you can use the terms accurately and precisely in the exams in order to gain top marks. A glossary of programming terms can be found on page 96 whilst the full glossary can be found at **www.hoddereducation.co.uk/myrevisionnotesdownloads.**

Exam tips

Exam tips are given throughout the book to help you polish your exam technique in order to maximise your chances in the exam.

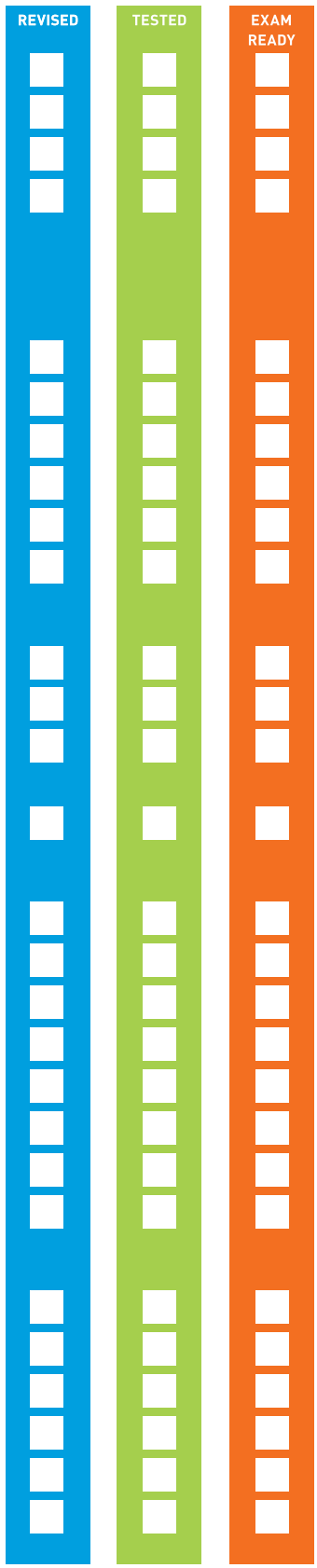
Now test yourself

These short, knowledge-based questions are the first step in testing your learning. Answers are available online.

My revision planner

Introduction

Unit 1



Introduction

Using this book

You can use this book to revise for WJEC GCSE in Computer Science.

- This is a **revision** book.
- It does not attempt to delve deeply into any of the topics.
- You should study individual sections using other available text books, lesson notes, attending classes and so on.
- It is to remind you of the essential vocabulary needed.
- It gives summaries of advantages and disadvantages of various computer science systems and practices.

This book is to be used **after** you have studied the subject.

The book is divided into Unit 1 and Unit 2 covering the topics in the specification. Answers for the 'Now test yourself' questions as well as a glossary listing key terms can be found online at www.hoddereducation.co.uk/myrevisionnotesdownloads.

Unit 1

REVISED ☐

- Understanding Computer Science
- Written examination: 1 hour 45 minutes
- 50% of the qualification
- 100 marks

Unit 2

REVISED ☐

- Computational Thinking and Programming
- On-screen examination: 2 hours
- 30% of the qualification
- 60 marks

How to revise

REVISED ☐

Revision is an **active** undertaking. You cannot revise successfully using reading skills alone. There are several ways of making sure that the content you are revising stays in your long-term memory.

- After you have read a section put the book down and summarise the essential points in writing.
- Use past examination questions and answer them. Always check on the kind of answers the examiners are looking for.
- Revise with a friend or get a member of the family to test you.
- Revise at a desk or table, not lying in bed!

Examination technique

REVISED ☐

The first and foremost technique is to **be prepared**. The examination is not a magic wand that will fill your head with answers on the day of the exam. It is a test of what you have learned in the past year or two years. If you are properly prepared you will more easily relax.

1 Hardware

Computer hardware consists of the physical parts of the computer, including

- **central processing unit (CPU)**
- **soundcard**
- **motherboard**
- **graphics processing unit (GPU)**
- **input** and **output units**
- **storage** units (primary and secondary).

The internal organisational structure of a computer is known as the **computer architecture**. There are many designs, including the

- **Von Neumann** model
- **Harvard** model.

The Von Neumann model

REVISED

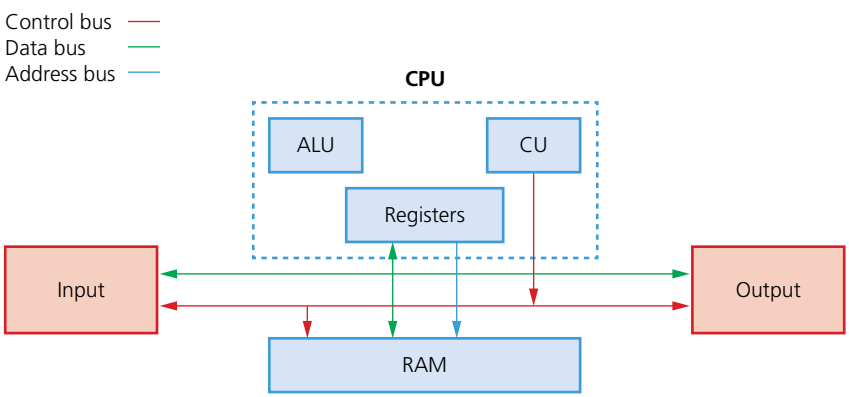


Figure 1.1.1 The Von Neumann model

Central processing unit (CPU)

REVISED

The **central processing unit** consists of the components of the computer that process the instructions. These instructions may be

- simple arithmetic
- logical operations.

There is a CPU at the heart of every computer. It is regarded as the 'brain' of the computer. The main CPU architecture used today is the Von Neumann architecture.

Every program instruction is interpreted and executed within the CPU. To process the data a computer must have facilities to

- input instructions and data
- store the data and program instructions (a memory)
- output the results of the processing
- control and interpret the machine language and send appropriate signals to each of the other components.

Components of the CPU

- Control unit (CU)
- Arithmetic and logic unit (ALU)
- Registers
- Buses
- Internal memory

The control unit

- Supervises the **fetch-decode-execute cycle.**
- Sends and receives signals from all parts of the computer.
- Decodes instructions in the current instruction register (CIR).
- Selects machine resources.
- Selects the particular mathematical operation to be used.
- Ensures that all processes take place at the right time and in the correct order.

The arithmetic and logic unit

- Processes and manipulates data.
- Carries out simple arithmetic operations, such as addition and subtraction.
- Carries out logical operations, such as comparing two values.

The registers

The CU needs somewhere to store details of the operations being dealt with by the fetch-decode-execute cycle. The ALU needs somewhere to put the results of any operations it carries out. These are called **registers** and, although they have limited storage capacity, they play a vital role in the operation of the computer. Registers are usually much faster to access than internal memory since they must be accessed so often.

Registers that are used by the processor as part of the fetch-execute cycle include the

- **current instruction register** (CIR) that stores the instruction that is currently being executed by the processor
- **program counter** (PC) that stores the memory location of the next instruction that will be needed by the processor
- **memory address register** (MAR) that stores the memory location where data is currently being written to or read from
- **accumulator** (ACC) that stores the results of calculations made by the ALU.

Exam tip

A register is a temporary, fast access storage location found in the CPU where data or control information is temporarily stored.

Buses

Buses are pathways using groups of parallel wires that connect the processor to the various input and output controllers being used by the computer. They are also used to connect the internal components of the CPU, known as registers, and to connect the CPU to memory. They are the

- data bus
- address bus
- control bus.

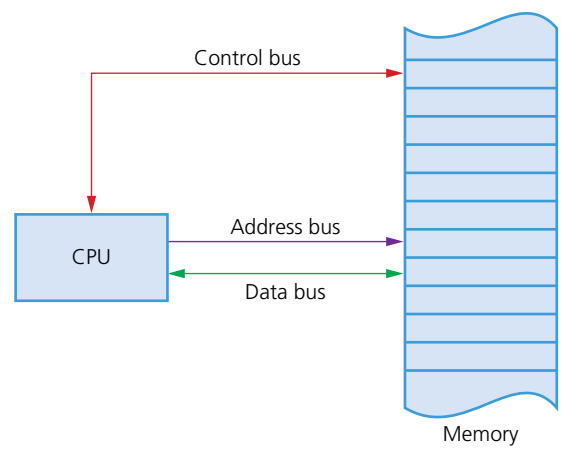


Figure 1.1.2 Buses connecting the CPU to the memory

The data bus

- carries instructions and data between the processor and memory as the program is run
- transfers the data both to and from memory
- moves the data to and from the I/O controllers.

The address bus

- only carries data in one direction – from the processor into memory
- is used by the processor and carries the memory address of the next instruction or data item
- is used to access anything that is stored in memory, not just instructions, to the processor.

The control bus

- is used by the CPU to communicate with other units in the computer
- ensures that the control information reaches the right place at the right time
- usually only sends data one-way.

Internal memory

- Memory built into the computer such as
  - random access memory (RAM)
  - read only memory (ROM)
  - registers.
- Fast access memory is directly accessed by the processor.

Check your understanding

REVISED

1 Write down **five** items of hardware.
2 Name **two** different types of computer architecture.
3 Give the full name to these registers:

CIR	
MAR	
ACC	
PC	

The fetch-decode-execute cycle

REVISED

There are three steps to processing instructions given by a currently running program:

1 **The fetch cycle**
   - The PC holds the address of the next instruction to be executed.
   - A copy of the address in the PC is sent to the MAR.
   - The processor uses this address to find the instruction in the memory.
   - The contents of the data in that memory location are sent to the CIR.
   - The PC moves on one to point at the next instruction to be carried out.
   - The cycle is repeated until all instructions have been carried out.
2 **Decode**
   - The CU checks the instruction in the CIR.
   - The instruction is decoded to determine the action that needs to be carried out.

3 **Execute**
- The processor carries out the instruction.
- The next instruction is fetched.

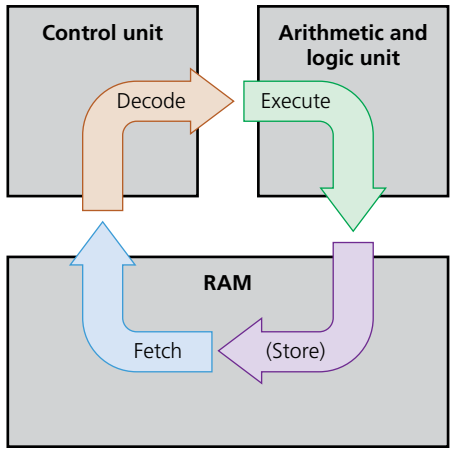


Figure 1.1.3 Fetch-decode-execute cycle

Factors affecting processor performance

REVISED

Cache memory

- Fast access temporary storage on the CPU.
- Used to store instructions and data that are needed frequently.
- Most commonly used functions or data used in a program are placed into the cache.

Cache memory has the disadvantage of being more expensive than random access memory (RAM), but the advantage is that the cache can be accessed much more quickly than main memory, so programs run faster.

Clock speed

- Indicates how fast each instruction will be executed.
- Increasing the clock speed should increase the speed at which the processor executes instructions.

Multiple cores

A **core** contains an ALU, CU and registers. Multiple processors can be incorporated onto a single chip to make a multi-core processor. A dual-core processor, therefore, has two cores on one chip and will run much faster than a single-core system.

- In a single-core CPU, each instruction is processed one after the other.
- In a dual-core CPU, two instructions may be processed at the same time, meaning the instructions should be processed twice as fast as a single core.
- Faster speeds, in theory, can be gained using four cores (quad cores) or more.

Performance may be affected by one core waiting for the result of another and, therefore, cannot carry out any more instructions, leading to the performance being no better than a single-core processor.

Multiple cores increase the processor cost, though this is offset by the increase in processing speed.

Exam tip

Factors affecting speed should be considered as a whole rather than choosing one aspect to fix the problem. There is no point fitting a faster clock to a computer if you do not change the components that are going to make good use of the clock pulse.

Glossary of programming terms

Simplifying the problem

abstraction: changing a human problem into one that can be solved by a computer

decomposition: splitting a complex problem into smaller parts

programming: creating an algorithm in program code

pseudocode: an algorithm not using a specific programming language

Program structure

constant: a value that does not change in the lifetime of the program

documentation: explanation of the program code

function: a subroutine than returns a value

global variable: a variable that is recognised throughout a program

iteration: a process that is repeated

local variable: a variable that is only recognised in a particular subroutine

module: a discrete part of a computer program

repetition: a process that is repeated

selection: following a particular route depending on the answer to a question

sequence: instructions that are written to be executed in a predetermined order

subroutine: a section of program code that can be called many times

variable lifetime: a variable exists only as long as the program code that uses it

Object-oriented programming (OOP)

class: the definition of the method and properties of a group of similar objects

encapsulation: wrapping code and data together into a single unit

inheritance: allows a class to use the properties and methods of an existing class

instance variable: variables that are bound to class instances

method: the behaviour of an object in object-oriented programming

object: a particular instance of a class

object-oriented programming: using objects rather than actions and data rather than logic

Finding errors in the code – debugging

break point: interrupts a program at a specific line

debugging: finding and correcting errors in computer code

error diagnostics: error messages displayed to help the programmer

memory inspector: displays the contents of all the stores in a section of memory

single stepping: executes a program one line at a time when debugging

store dump: displays the contents of all variables used by the program

trace: displays the order in which the program is being executed

variable watch: shows values of variables while program is running