



# Computer Science

KS3 → GCSE

## Transition

# Summer Workbook

This workbook belongs to: \_\_\_\_\_



## Introduction

So, you've chosen Computer Science and are about to start the course...!?

Firstly, well done for making an excellent decision!

GCSE Computer Science is a fantastic course, which will open up a whole new world for you.

Learning how computer systems work will enable you to become 'the wizards of the future'.

Compared to your non-computing counterparts, you will be able to make computers do things that will 'blow people's mind'! Whether it's to revive a 'dead' machine, build a money-making website or design the latest 'world changing' app, studying Computer Science will give you the tools to make this a reality...and it all begins today!

Your course will cover many exciting topics and this workbook seeks to introduce you to some of them.

Below is a list of the various areas of study on your course and those highlighted will be 'touched on' in this workbook.

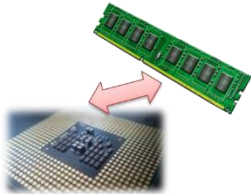
Topics:
Hardware
Data Representation
Networking
Software
CS Issues
Algorithms
Programming Techniques
Robust Programs
Logic
Translators & IDEs

So, engage and enjoy as you begin the GCSE, which is going to take you places!

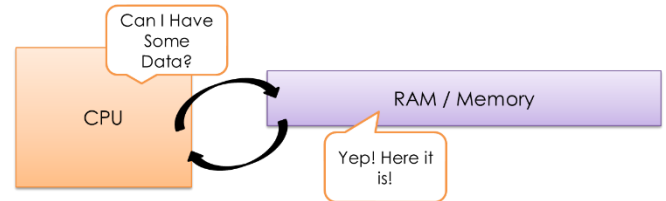
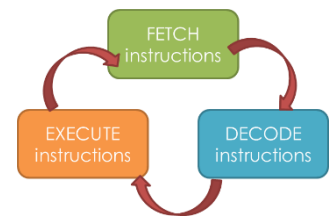
## Section 1: CPUs, The F-D-E Cycle & Clock Speed

The CPU is often known as the 'brain of the computer'. Its job is to process data. And by processing we mean things like searching, sorting, calculating and decision making. Whenever you are on working on your computer, it is the CPU which is at the heart of everything.

The CPU follows three steps in order to process data. It is known as the Fetch - Decode - Execute cycle (aka Fetch-Execute Cycle).



Whenever you open and work with a program, its data and instructions are loaded onto the RAM. As the RAM is accessed directly by the CPU, the CPU can get to work!

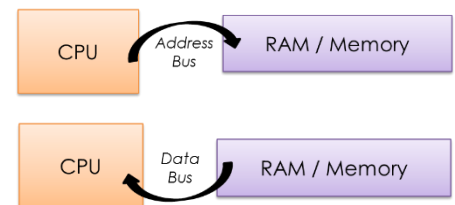


### The Fetch Stage

In this step the CPU fetches some data and instructions from main memory (RAM) and then stores them in its own temporary memory called 'registers'. For this to happen, the CPU uses a piece of hardware called the 'address bus'.

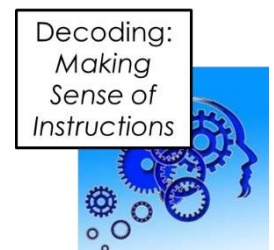
The address of the next item that the CPU wants is put onto the 'address bus'.

Requested data then travels from the RAM to the CPU on another piece of hardware called the 'Data Bus'.



### The Decode Stage

The decode step is where the CPU understands / works out what the instruction it has just fetched actually means. The CPU 'decodes' the instruction and gets things ready for the next step.



### The Execute Stage

The Execute stage is where data processing happens. Instructions are carried out on the data. Once a cycle has been completed, another begins.

### Clock Speed

The speed of the Fetch-Decode-Execute cycle is determined by the CPU's clock chip. This chip uses a vibrating crystal that maintains a constant rate. The speed of the clock is measured in hertz (Hz) which is the amount of cycles per second. A clock speed of 500Hz means 500 cycles per second. Current computers have CPU clock speeds of 3GHz which means 3 Billion cycles per second.

### Overclocking

It is possible to increase the clock speed for a CPU. This is known as overclocking. In theory, if the clock is faster, then the CPU can perform more calculations and therefore perform faster. The problem is that CPUs get hotter the more work they do – so overclocking is dangerous without the appropriate heat management.



**Questions:**

1. Name the steps that the CPU carries out in order to process instructions. [3]

---

---

---

---

---

2. Describe what happens during each of the stages written in your last answer. [3]

---

---

---

---

---

---

---

---

---

---

---

---

3. Bob's PC struggles to perform when playing the latest computer games, due to the CPU being unable to process instructions quick enough. Instead of upgrading his CPU, what could he try to do to improve the performance of his CPU and what issues may he be faced with? [3]

---

---

---

---

---

---

---

---

---

---

---

---

## Section 2: IPs, URLs and DNS

### Web Hosting and DNS

When visiting websites, we will usually type a website's address into a browser's address bar. However, our computer can only connect to these websites if it knows the websites IP address. So how do we get our computers to connect to websites, if we don't know their IP addresses? Well, computers use a system called DNS and in order to understand how DNS works we first need to understand a few other important acronyms.

### IP Address

This means INTERNET PROTOCOL ADDRESS. It is a unique number (e.g. 324.45.321.23) given to every computer on the internet – no two computers can have the same address. It's just like a postal address – used to identify a house – no two houses have the same address! Interestingly, a device's IP address will not necessarily remain the same each time it joins a network.

### ISP

This means INTERNET SERVICE PROVIDER. This is simply the company who provide you with your internet connection.

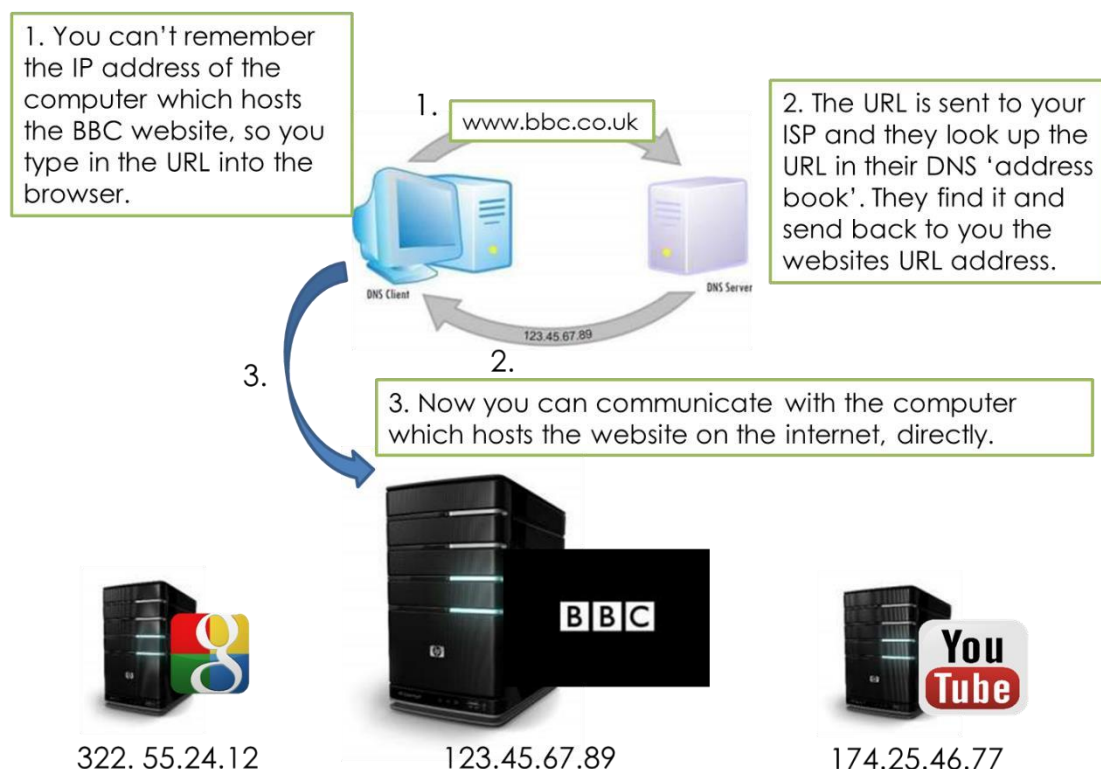
### URL

This means UNIFORM RESOURCE LOCATOR. This is simply a fancy name for a web address, such as "http://www.bbc.co.uk" or "http://www.google.com".

### DNS

This means DOMAIN NAME SYSTEM. This is the system used to find the computer which hosts the website you are looking for.

### How DNS works:





**Questions:**

1. Explain, using examples, the following acronyms: IP address, ISP and URL. [3]

---

---

---

---

---

---

---

2. What is DNS for and how does it work? [5]

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

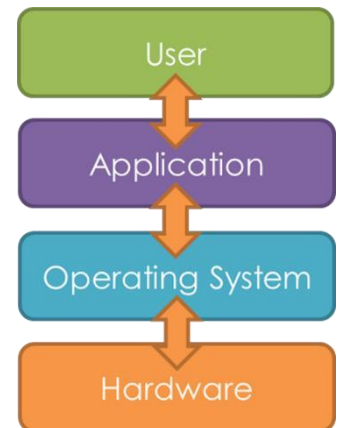
---

## Section 3: The Role of the Operating System

The operating system is the most important piece of software on any computer. Without it, no programs will run. This is because an operating system is responsible for controlling / communicating with the computer hardware. It provides a platform on which games, browsers, music players etc., can all work.

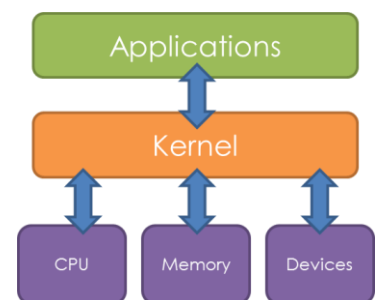
If you were to run an everyday program (e.g. a word processor), without an operating system, nothing would be displayed on the screen, nothing could be sent to the printer, nothing could be typed. This is because application software does not know how to TALK to hardware devices. However, the operating system does. The operating system can also talk to the application that is running. So when you print a document, the application talks to the OS, which in turn talks to the printer.

The operating system sits between the user's applications and the hardware. It enables applications to use the hardware resources.



### The Kernel

The kernel is the heart of the operating system and is responsible for looking after "the most low-level hardware operations". It is the kernel that applications make use of when they want to operate the computer's hardware.



### Questions:

1. Using an example, explain why an Operating System is essential when running applications on a computer. [4]

---

---

---

---

2. What is the role of the Operating System's Kernel? [2]

---

---

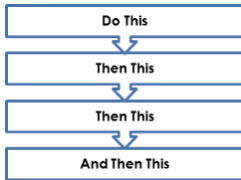
---

---

## Section 4: Programming Constructs

Programming languages have a set of statements to determine how to reach a goal. The FLOW of these statements is CONTROLLED by 3 different structures:

**Sequencing** (Performing one instruction after another)



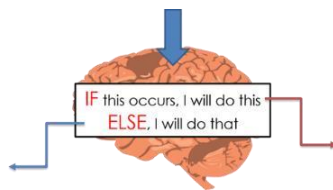
```

number1 = integer
number2 = integer
answer = integer

INPUT number1
INPUT number2
answer = number1 + number2

PRINT answer
  
```

**Selection** (The program making decisions)



```

month = input("Please enter your month number")
month = int(month)

if month == 2:
    print("Your month has 28 days")
elif (month == 4) or (month == 6) or (month == 9) or (month == 11):
    print("month has 30 days")
else:
    print("Your month has 31 day")

input()
  
```

**Iteration** (The program repeating: for a set number or whilst a condition is true)



```

count = 10

while count > 0:
    print("Hello Word")
    count = count - 1

input()
  
```

```

Hello Word
Hello Word
Hello Word
Hello Word
Hello Word
Hello Word
Hello Word
Hello Word
Hello Word
Hello Word
  
```

**Question:**

Label the code to identify where the following control structures are:

- Sequencing [1]
- Selection [1]
- Iteration [1]

```

import time

menuchoice = 0

print('*****Menu*****')
print('')
print('1. Display my name')
print('2. Display my age')
print('3. Display my address')
print('')

while True:
    try:
        while(menuchoice < 1) or (menuchoice > 3):
            menuchoice = int(input("What is your menu option?"))
        break
    except ValueError:
        print("Please type in a number not a word")

if menuchoice == 1:
    print("Mr Wickins")
elif menuchoice == 2:
    print("29 years old")
elif menuchoice == 3:
    print("Sidmouth College")

print("Goodbye")
  
```





## Section 5: Translators

### Introduction

CPUs are very impressive but they are actually quite simple when it comes to processing. They can only process 1's and 0's (machine code). They therefore do not understand how to process high level programming code in the form in which we write it (e.g. Python).

### High Level Language Translators

Translators are programs that convert high level language commands, such as PRINT, IF and FOR, into a set of machine code commands such as 1011, 11001 and 11000011110, so that the CPU can process the data.

There are 2 ways in which translators work:

- Take the whole code and convert it into machine code before running it (known as compiling).
- Take the code one instruction at a time, translate and run the instruction, before translating the next instruction (known as interpreting).

#### Compiler

Translates the whole code into one executable file (often a .exe file).

The file can then be run on any computer without the translator needing to be present.

Can take a long time to compile source code as the translator will often have to convert the instructions into various sets of machine code as different CPUs will understand instructions with different machine code from one another.

#### Interpreter

Converts the source code into machine code 1 line at a time. Program therefore runs very slowly.

An interpreter is often used at the testing / development stage as it highlights any coding errors, when it gets to them. Programmers can quickly identify errors and fix them.

The translator must be present on the computer for the program to be run.

Interpreters do not generate machine code directly (they call appropriate machine code subroutines within their own code to carry out commands).

### Questions:

1. Can computers understand high level programming code? *Explain your answer.* [2]

---

---

2. Explain the similarity and differences between 'Compilers' and 'Interpreters'. [3]

---

---

---

---



## Section 6: Binary Numbers

Because humans have 10 fingers, they learnt to count using a 'base 10' number system (the denary/decimal number system).

When we count we start at zero and keep adding 1. We have digits to represent each number up to 9. But after that something interesting happens. When we get to ten there is no single digit to represent that number. Instead we record that we have counted to ten by placing a 1 in the 10s column and then we simply restart counting from zero up to 9 again in the units column. When we get to 100, we make a record of it by placing a 1 in the 100s column...and so on!

Computers do not have ten fingers on which to count. Computers are made up of switches and as such can only represent 2 digits. Computers use the binary number system.

The binary system is very similar to our denary number system, however, instead of the columns representing (from right to left) ones (units), tens, hundreds etc. the binary system columns represent ones, twos, fours, eights etc.

So, when we count in binary we only use 2 digits (1 and 0). We start at zero and keep adding 1. But as we have no digit for 2, we place a 1 in the twos column and restart counting from zero in the ones column. Study the image to the right to see how we count up to 5 in binary.

Fours	Twos	Ones	
0	0	1	= 1 (in denary)
Fours	Twos	Ones	
0	1	0	= 2 (in denary)
Fours	Twos	Ones	
0	1	1	= 3 (in denary)
Fours	Twos	Ones	
1	0	0	= 4 (in denary)
Fours	Twos	Ones	
1	0	1	= 5 (in denary)

### Converting from Binary to Denary (easy)

If you are given a binary number which is to be converted into denary do the following:

1. Above each bit, write the value of that bit.

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

2. Then simply add the values where there is a 1 underneath, together.

1	2	8
		4
		1
	1	
1	3	3

**Questions:**

Convert the following binary numbers into denary. Show your workings.

1) 01100011

128	64	32	16	8	4	2	1

**Workings:**

Answer = \_\_\_\_\_

2) 11101010

128	64	32	16	8	4	2	1

**Workings:**

Answer = \_\_\_\_\_

3) 00101001

128	64	32	16	8	4	2	1

**Workings:**

Answer = \_\_\_\_\_

4) 10101010

128	64	32	16	8	4	2	1

**Workings:**

Answer = \_\_\_\_\_