A2 Computer Science

AQA 7516/2 (Paper 2)

- **Number Bases & Systems**

AS COMPUTER SCIENCE

UNIT 4

- Number Systems:
- Binary, Denary, Hexadecimal

Number Bases:

In this chapter you will learn:

- the function of bits and bytes and how they are combined to form larger units
- how number bases work including binary, decimal and hexadecimal
- how to convert binary to decimal and vice versa
- how to convert binary to hexadecimal and vice versa
- how to convert decimal to hexadecimal and vice versa.

Number Bases:

Objectives:

- Understand how **Binary, Denary and Hexadecimal numbers** are represented and why it is a core feature of computing.

Success Criteria:

- To have a solid understanding of data representation of **Binary and Hexadecimal numbers**.
- Be able to explain how the **Binary and Hex numbers** work and how they can be used.
- Be able to complete simple processes using **Binary, Denary and Hex numbers** effectively in your work.

Key Vocabulary

- Number-base
- Binary
- Hexadecimal
- Bit
- Byte
- LSB / MSB
- Overflow

Bits and Bytes

- All digital computers use the binary number system for representing data of all types –numbers, characters, sound, pictures etc.
- **Binary digit** (known as a **bit**) can be either 0 or 1.
- The processor can only handle electricity in a relatively simple way – either electricity is flowing, or it is not. Hence, **0 or 1**.
- Bits are normally grouped in **8-bit bytes**.
  - One byte can hold one character, part of a sound, part of a number, etc.
  - A byte can hold 2^8 combinations of 0s and 1s.
  - E.g. 256 different characters can be represented.

Key Terms:

Binary number system (Base 2):

- Is based on a number system that uses two digits, 0 and 1.

Denary number system (Base 10):

- Number system is based on the decimal system and uses the digits 0 to 9.

Bits:

- A single binary digit that can have a value that is either 0 or 1.
- 0 (low voltage) and 1 (high voltage) in a computer circuit.

Bytes:

- A group of **8 bits** e.g. 10101011.
- The maximum value it can store is 255.

Nibble:

- Is a group of **4 bits** or 'half a byte'.

Denary Numbers (base 10)

- '134' represents 1 hundred, 3 tens and 4 units.

100	10	1
1	3	4

= 100 + 30 + 4 = 134

Binary Numbers (base 2)

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

Add the numbers in the table if they have a 1 in their column, to find the number in Denary (base 10).

Therefore,

$$= 128 + 4 + 2$$

$$= 134$$

Represent Integers using Binary:

To begin with we will work with denary positive whole numbers between 0 and 255, which fits into a byte (8-bit binary number).

Denary	Binary Data							
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
13	0	0	0	0	1	1	0	1
47	0	0	1	0	1	1	1	1
166	1	0	1	1	0	0	0	0
255	1	1	1	1	1	1	1	1

BINARY AND DENARY NUMBERS

Converting to and from Binary…

Convert Binary To Denary

a) 0100
b) 0101
c) 1010
d) 01000010
e) 01011001

a) 4
b) 5
c) 10
d) 66
e) 89

Convert Denary To Binary

a) 3
b) 9
c) 19
d) 28
e) 76
f) 129

a) 0011
b) 1001
c) 00010011
d) 00011100
e) 01001100
f) 10000001

Denary → Binary:

- To convert the **denary** number of **21** to **binary**.
- Write down the binary place values in a table.
- Find the largest place value that is less than or equal to your number.
- Write a 1 under that column header (16).
- Subtract that number (16) from your original number (**21**)
  - and you get 5.
- Repeat the above process until you have no numbers left, you will either have a 1 as a result or you will have finished before then with no leftovers.

21 =

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

Denary → Binary:

- Try out these below examples, fill out one byte only:

128	64	32	16	8	4	2	1

1. 179 → 10110011
2. 29 → 00011101
3. 148 → 10010100
4. 164 → 10100100
5. 13 → 00001101
6. 255 → 11111111
7. 71 → 01000111

Quantities of Bytes (base 10)

- Storage capacity is measured in thousand-byte units called kilobytes (KB), megabytes(MB) and gigabytes (GB).
- **1 Kilo is exactly 1,000.**

1 KB = 1000 = 10^3 bytes

1 MB = 1000 x 1000 = 10^6 bytes *or* 1,000 KB

1 GB = 1000 x 1000 x 1000 = 10^9 bytes *or* 1,000 MB

What is the capacity of your RAM?
Use command:

```
wmic memorychip get capacity
```

Memory Addressing (base-2)

- Each byte in memory has its own address, and memory capacity is measured in different base-2 units called **kibibytes (KiB)**, **mebibytes (MiB)** and **gibibytes (GiB)**.
- **1 Kibi is exactly 1,024.**

International Electrotechnical Commission (IEC) standard

1 KiB = 1024 = 2^{10} bytes

1 MiB = 1024 x 1024 = 2^{20} bytes *or* 1,000 KiB

1 GiB = 1024 x 1024 x 1024 = 2^{30} bytes *or* 1,000 MiB

HEXADECIMAL NUMBERS

What is Hexadecimal

&

Converting to and from Hex…

Hexadecimal (base 16):

- You may notice from the table that **one hexadecimal digit** can represent **exactly 4 binary bits (Nibble)**.

- Hexadecimal is useful to us as a shorthand way of writing binary, and makes it easier to work with long binary numbers.

- Hexadecimal is a **base-16** number system which means we will have 16 different characters (numbers and letters) to represent our digits.

- The only problem being that we run out of numbers after 9, and knowing that 10 is counted as two digits we need to use letters instead:

  - **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**

Hexadecimal:

- Hexadecimal is a **base-16 number system** which means we will have 16 different ‘numbers’ to represent our digits.

  - **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**

- Hexadecimal is used in computers for representing numbers for human consumption, having uses for things such as memory addresses, error codes and Hex colour codes (Fireworks).

- **NOTE:** *Hexadecimal is used as it is shorthand for binary and easier for people to read and remember. It **DOES NOT** take up less space in computer memory, only on paper!*

- Computers still have to store everything as binary, whatever it appears as on the screen as.

Number Systems:

Binary	Denary	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

- When using the different number systems available in computing it is very useful to take note of the table to the left.
- This details some of the key numbers used across the different number systems.

- I recommend that you copy this table.
- So that you can refer to it throughout this chapter to check your answers.

Hexadecimal (base 16)

8	**4**	**2**	**1**
1	0	1	0

= 8 + 2 = 10 = A

8	**4**	**2**	**1**
1	1	1	1

= 8 + 4 + 2 + 1 = 15 = F

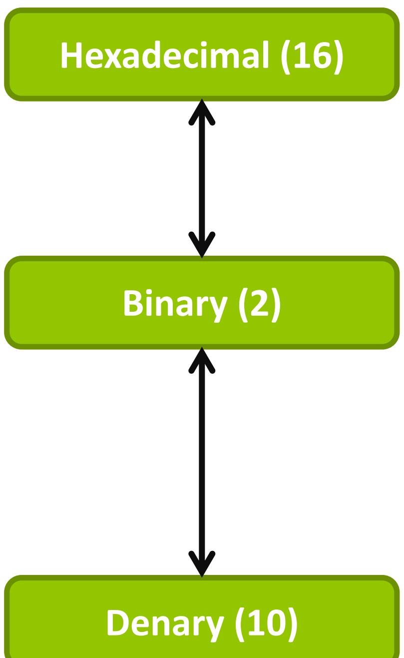
CONVERSIONS:
USING ALL NUMBER SYSTEMS.

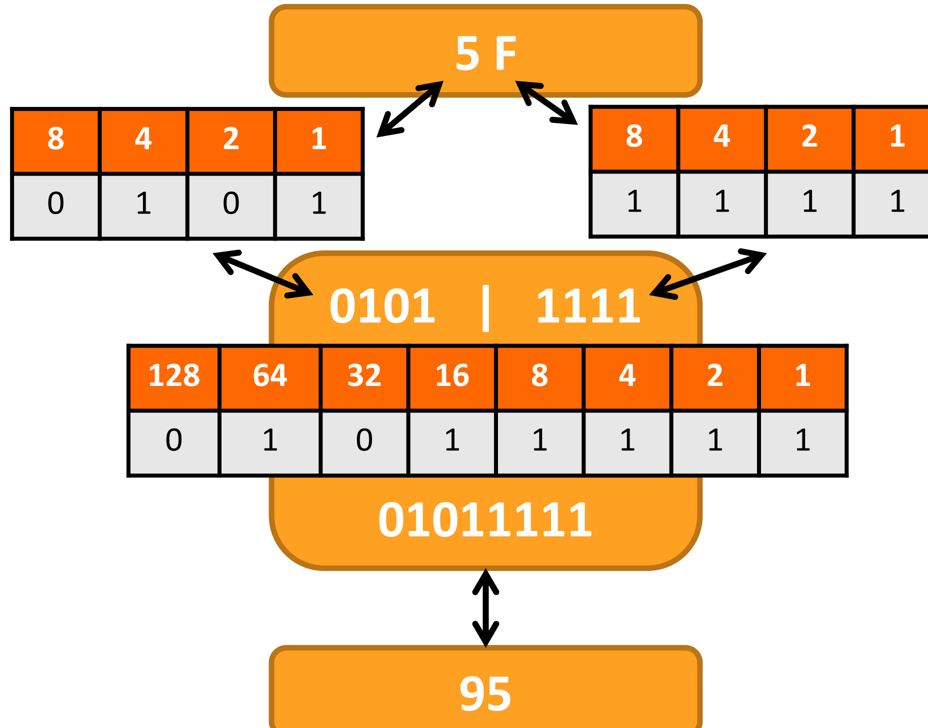
Converting numbers using:

- Binary
- Hexadecimal
- Denary

Hex→ Binary → Denary :

Method 1 (suitable for all conversions)





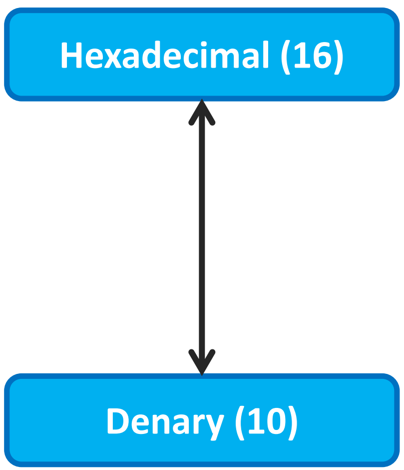
Hexadecimal conversions:

Use your preferred Method 1 or 2:

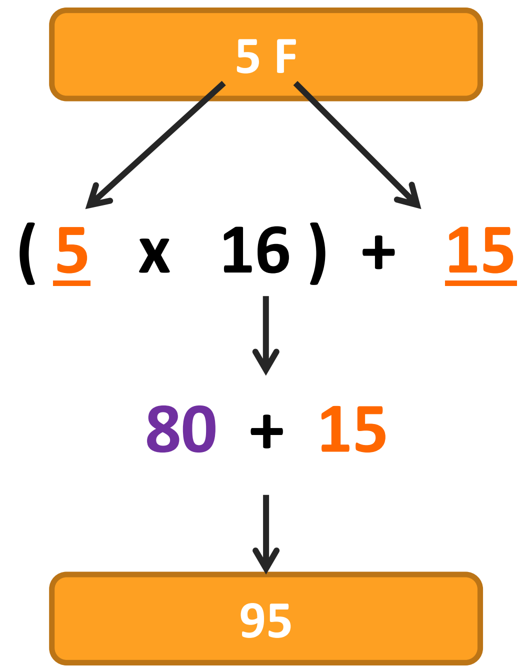
1. A1	1. 161	1. 10100001
2. 7D	2. 125	2. 01111101
3. C4	3. 196	3. 11000100
4. E3	4. 227	4. 11100011
5. F9	5. 249	5. 11111001
6. B2	6. 178	6. 10110010
7. 3D	7. 61	7. 00111101
8. 2D	8. 45	8. 00101101
9. 4A	9. 74	9. 01001010
10. 8B	10. 139	10. 10001011

Hex→ Denary :

Method 2 (suitable for 2 digit Hex conversions)



Binary	Denary	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F



Hexadecimal conversions:

Use your preferred Method 1 or 2:

	Hex	Decimal	Binary
1.	A1	161	10100001
2.	7D	125	01111101
3.	C4	196	11000100
4.	E3	227	11100011
5.	F9	249	11111001
6.	B2	178	10110010
7.	3D	61	00111101
8.	2D	45	00101101
9.	4A	74	01001010
10.	8B	139	10001011

What have you learnt?

1. A binary pattern `0110 1010` can be interpreted in different ways.
    a. State its hexadecimal representation
    b. State its denary value as a pure binary integer.
    c. State its denary value as a signed integer in two's complement notation.
    d. State its denary value as an unsigned fixed point number with four digits after the point.

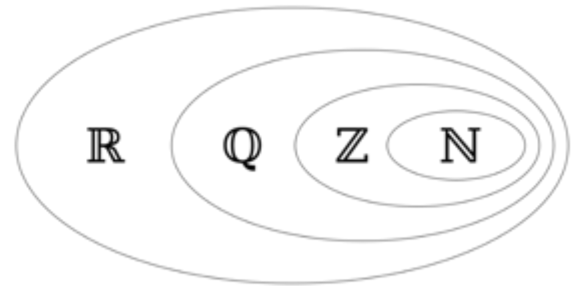
2. Same question for bit pattern `1011 0011`

NUMBER SETS

Useful to cover but will most likely need recapping later on.

Number Sets

$\mathbb{N}$	Naturals, e.g. 0, 1, 2, …
$\mathbb{Z}$	Integers, e.g. -2
$\mathbb{Q}$	Rationals, e.g. ½
	Irrationals, e.g. $\sqrt{2}$, π
$\mathbb{R}$	Reals, e.g. -2, 0, 1, ½, π



$\mathbb{Z}$ is for Zahlen (German for numbers)
$\mathbb{Q}$ is for Quotient (quantity produced by division)
$\mathbb{R} \setminus \mathbb{Q}$ defines the set $\mathbb{R}$ minus all members of the set $\mathbb{Q}$

For x to be a rational number, it must be expressible in the form:
$x = {}^{m}/_{n}$
Where m and n are integers, excluding zero for n.

The only numbers that can be truly represented as a binary number are integers.
Rationals and irrationals are approximated.
What data types does C# use to represent:

- Naturals?
- Integers?
- Reals (union of rationals and irrationals)?

Sets

- A set is an unordered collection of values or symbols in which each value/symbol occurs at most once.
- An empty set Ø has no elements, i.e.

```
A = {}
```

Defining By Listing Each Number

- A set can be defined by listing each number, e.g.

  A = {2, 4, 6, 8}

- A set can be infinite, e.g.

  $\mathbb{N}$ = {0, 1, 2, 3, …}

  $\mathbb{Z}$ = {…, -2, -1, 0, 1, 2, …}

- An infinite set can be countable, e.g. $\mathbb{Z}$

  or uncountable, e.g. $\mathbb{R}$

Extension

- What is the sum of the first 100 natural numbers?

```
  1+  2+  3+...+ 98+ 99+100
100+ 99+ 98+...+  3+  2+  1
```

$$\frac{n\ (n + 1)}{2}$$

Helpful links...

- Computing WikiBooks ***(very useful on most topics)***:
  - https://en.wikibooks.org/wiki/A-level_Computing/AQA
- C Sharp.net Tutorials:
  - http://csharp.net-tutorials.com/basics/visual-csharp-express/
- Microsoft .NET and C# Tutorials:
  - https://www.microsoft.com/net/tutorials/csharp/getting-started
- Tutorials Point website – helpful tutorials:
  - https://www.tutorialspoint.com/csharp/
- AS Computing – Revision Flash Cards:
  - http://quizlet.com/23924940/aqa-as-computing-flash-cards/