

Exercise walk-through

1.1

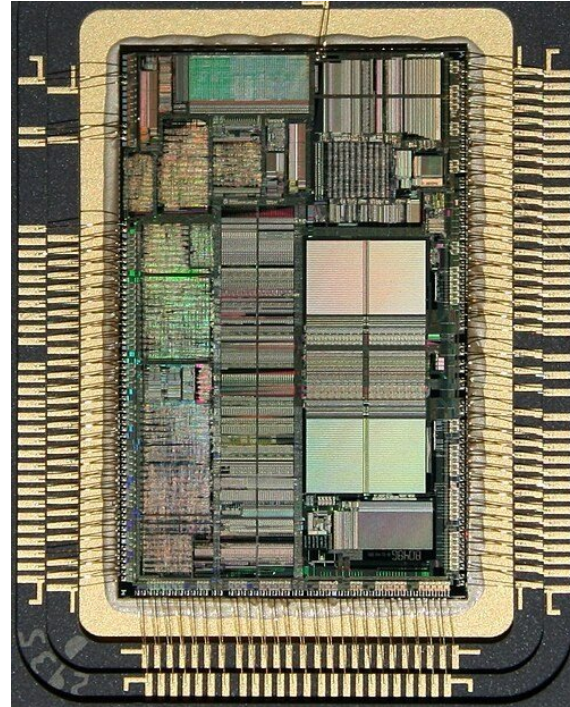
Number systems

IGCSE Computer Science

Why binary?

A computer has only two states: **on** and **off**, written 1 and 0. A system of two digits is **binary** (base 2).

Every kind of data – numbers, text, sound, images – must become binary before the computer can process it.



Three number systems

denary

base 10
digits 0–9

binary

base 2
digits 0 and 1

hexadecimal

base 16
0–9 then A–F

The **base** is how many different digits a system uses. Hex A–F stand for 10–15.



Hard disk storage

Place value

Binary place values **double** from right to left. For 8 bits:

128 64 32 16 8 4 2 1

place value	128	64	32	16	8	4	2	1
bits	1	0	0	1	0	1	1	0

$$150 = 128 + 16 + 4 + 2 \Rightarrow 10010110$$

- 1 **bit** = a single 0 or 1
- 8 bits = 1 **byte**
- 4 bits = 1 **nibble**

Denary $\rightarrow$ binary: **subtract** the place values that fit. $150 = 128 + 16 + 4 + 2$.

Which end of a binary number matters

Most significant bit

The **most significant bit** is the leftmost – worth the most. In two's complement it is also the sign.

Least significant bits

The **least significant bit(s)** are the rightmost – worth the least, and the first to be lost when a number is truncated.

Why it matters

A logical shift left multiplies by two and pushes bits off the most significant end; a shift right divides and loses the least significant.

In hardware

The same convention orders the bits in **registers**, in **transmission** down a wire, and in the outputs of **logic gates**.

Say which end you mean. “The first bit” is ambiguous, and the mark scheme is not.

Worked example – denary to binary and hex

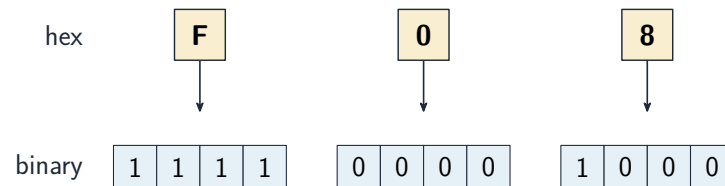
Convert denary 150 to binary, and 100 to hexadecimal.

$150 = 128 + 16 + 4 + 2$, so

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

$$= 10010110$$
$$100 = 64 + 32 + 4 = 0110\ 0100$$

nibbles = 6 and 4, so hex = 64



one hex digit = one nibble (4 bits)

one hex digit = one nibble

Each hex digit maps to its own 4-bit nibble – F08 = 1111 0000 1000

Exercises 2.1-2.2

2.1 Convert the denary number 45 to 8-bit binary.

[2]

2.2 Convert the binary number 11001010 to hexadecimal.

[2]

Exercises 2.1-2.2

2.1 Convert the denary number 45 to 8-bit binary.

[2]

2.1

- denary $\rightarrow$ 8-bit binary: place values 128, 64, 32, 16, 8, 4, 2, 1
- $45 = 32 + 8 + 4 + 1$
- **00101101**

2.2 Convert the binary number 11001010 to hexadecimal.

[2]

2.2

- binary $\rightarrow$ hex: split into nibbles from the right
- $1100 = 12 = \text{C}$ and $1010 = 10 = \text{A}$
- **CA**

Exercise 2.3

2.3 Convert the denary number 200 to 8-bit binary using the division method, showing your working. [3]

Exercise 2.3

2.3 Convert the denary number 200 to 8-bit binary using the division method, showing your working. [3]

2.3

- denary $\rightarrow$ binary: divide by 2, record remainders, read **bottom to top**

```
200 ÷ 2 = 100  rem 0
100 ÷ 2 =  50  rem 0
 50 ÷ 2 =  25  rem 0
25 ÷ 2 =  12  rem 1
12 ÷ 2 =   6  rem 0
 6 ÷ 2 =   3  rem 0
 3 ÷ 2 =   1  rem 1
 1 ÷ 2 =   0  rem 1  ← left-most bit
```

- remainders bottom $\rightarrow$ top: **11001000**
- check: $128 + 64 + 8 = 200$

Exercises 2.4-2.5

2.4 Convert the hexadecimal number 3F to denary.

[2]

2.5 Give two reasons why hexadecimal is used to represent binary numbers, and name two places where it is used.

[4]

Exercises 2.4-2.5

2.4 Convert the hexadecimal number 3F to denary.

[2]

2.4

- hex $\rightarrow$ denary: 3F means $3 \times 16 + F$ and $F = 15$
- $48 + 15 = \mathbf{63}$

2.5 Give two reasons why hexadecimal is used to represent binary numbers, and name two places where it is used.

[4]

2.5

- any **two** reasons: shorter than binary / easier to read and copy / each hex digit is exactly four bits
- any **two** uses: MAC addresses / IPv6 addresses / HTML colour codes / memory dumps / error codes

Exercise 3.1

3.1 An 8-bit register can hold the denary values:

[1]

☐

A 0 to 127

B 0 to 255

C 0 to 256

D 1 to 255

Exercise 3.1

3.1 An 8-bit register can hold the denary values:

[1]

☐

A 0 to 127

B 0 to 255

C 0 to 256

D 1 to 255

3.1 B

- an 8-bit register stores $2^8 = 256$ different values, **0 to 255**
- A is 7-bit unsigned; C goes one past the last code; D misses 0

Exercise 3.2 (a)

3.2 Two 8-bit binary numbers are added: $01101100 + 00111010$.

(a) Work out the binary result.

[2]

Exercise 3.2 (a)

3.2 Two 8-bit binary numbers are added: $01101100 + 00111010$.

(a) Work out the binary result.

[2]

(a)

- add column by column from the right

```
  0110 1100
+ 0011 1010
-----
  1010 0110
```

- 10100110

Exercise 3.2 (b)

3.2 Two 8-bit binary numbers are added: $01101100 + 00111010$.

(b) Convert your answer to denary, and state whether overflow has occurred.

[2]

Exercise 3.2 (b)

3.2 Two 8-bit binary numbers are added: $01101100 + 00111010$.

(b) Convert your answer to denary, and state whether overflow has occurred.

[2]

(b)

- 1-bits at $128 + 32 + 4 + 2 = \mathbf{166}$
- $166 \leq 255$ (no ninth bit), so **no overflow**

Exercise 3.3

3.3 A register holds 00110100. Perform a logical left shift of 2 places and state the effect on the denary value. [2]

Exercise 3.3

3.3 A register holds 00110100. Perform a logical left shift of 2 places and state the effect on the denary value. [2]

3.3

- 00110100 shifted left 2 places, zeros fill the right
- 11010000; a left shift of 2 multiplies by $2^2 = 4$

Exercise 3.4 (a)

3.4 A microprocessor uses 8-bit registers.

(a) Convert the denary number 172 to 8-bit binary.

[2]

Exercise 3.4 (a)

3.4 A microprocessor uses 8-bit registers.

(a) Convert the denary number 172 to 8-bit binary.

[2]

(a)

- $172 = 128 + 32 + 8 + 4$
- **10101100**

Exercise 3.4 (b)

3.4 A microprocessor uses 8-bit registers.

(b) Convert your answer to (a) to hexadecimal.

[2]

Exercise 3.4 (b)

3.4 A microprocessor uses 8-bit registers.

(b) Convert your answer to (a) to hexadecimal.

[2]

(b)

- split 1010 1100
- $1010 = 10 = A$ and $1100 = 12 = C$, so **AC**

Exercise 3.4 (c)

3.4 A microprocessor uses 8-bit registers.

(c) The register holds 10110101. This binary number is added to 01011100. Complete the addition and give the 8-bit answer. [3]

Exercise 3.4 (c)

3.4 A microprocessor uses 8-bit registers.

(c) The register holds 10110101. This binary number is added to 01011100. Complete the addition and give the 8-bit answer. [3]

(c)

- add column by column from the right

```
  1011 0101
+ 0101 1100
-----
1 0001 0001
```

- stored 8-bit result **00010001**, with a carry out of bit 7

Exercise 3.4 (d)

3.4 A microprocessor uses 8-bit registers.

(d) Explain whether overflow has occurred in part (c), and state what overflow means.[2]

Exercise 3.4 (d)

3.4 A microprocessor uses 8-bit registers.

(d) Explain whether overflow has occurred in part (c), and state what overflow means.[2]

(d)

- **yes, overflow**
- the true sum needs 9 bits, but the 8-bit register cannot store the ninth, so the stored answer is wrong

Exercise 3.4 (e)

3.4 A microprocessor uses 8-bit registers.

(e) The register 00011010 has a logical left shift of 3 places applied. Give the result, and calculate the effect on the denary value. [3]

Exercise 3.4 (e)

3.4 A microprocessor uses 8-bit registers.

(e) The register 00011010 has a logical left shift of 3 places applied. Give the result, and calculate the effect on the denary value. [3]

(e)

- 00011010 shifted left 3 places $\rightarrow$ **11010000**
- denary was $16 + 8 + 2 = 26$ and becomes $128 + 64 + 16 = 208$
- a left shift of 3 multiplies by $2^3 = 8$, and $26 \times 8 = 208$

Exercise 3.4 (f)

3.4 A microprocessor uses 8-bit registers.

(f) The same register is used to hold two's complement numbers. Give the denary value of 11101100. [2]

Exercise 3.4 (f)

3.4 A microprocessor uses 8-bit registers.

(f) The same register is used to hold two's complement numbers. Give the denary value of 11101100. [2]

(f)

- two's complement: the MSB is worth -128
- $-128 + 64 + 32 + 8 + 4 = -\mathbf{20}$