Representing Binary in Computers:

- In a base two system, there are only two numbers available (0 and 1). When a digit higher than a binary 1 is required, an increment resets the 1 to a 0 and an increment of the next digit to the left is produced.
- Each digit moving from left to right in binary represents a higher power of 2 when comparing to the base 10 system.
- The denary is converted to binary using the standard system of dividing by the base 2's and writing in the 1/0s to a grid.

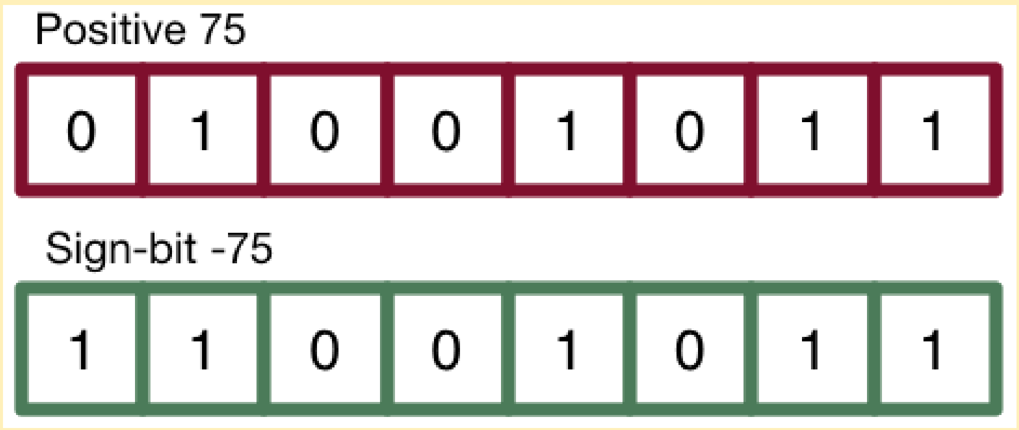
i.e. The index of 2 by which each digit represents an increase by 1 successively from left to right.

Base 10 representation	128	64	32	16	8	4	2	1
Index representation	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary	1	1	1	1	1	1	1	1

Write 01111001 in decimal	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary value	0	1	1	1	1	0	0	1
Calculation	0	1*64	1*32	1*16	1*8	0	0	1
Denary	121							
Write 176 in binary	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary calculation	1	0	1	1	0	0	0	0
Binary	10110000							

Sign and Magnitude

This method uses the most significant bit (the leftmost digit) to represent the sign and no magnitude
(e.g. 0 = positive number 1 = negative number)
The digits to the right of this will only represent magnitude.
This means the largest number that can be represented with 8 bits is now 127 and -127.



Computers are told to interpret the binary in the sign-magnitude form by the data type in the code rather than just positive 8 bits so 11111111 = -127 not 255 and 01111111 = 127. In exam questions' you are told what form to interpret the binary in.

Two's Complement method

This is the most common way computers use to interpret negative numbers.

To get the two's complement negative notation of an integer, the decimal number is expressed in an ordinary binary form. You then invert the digits and add binary one to the binary result. Remember that the most significant digit of the binary number is now a negative e.g. 0 lots of 128 becomes 1 lots of 128 but in a negative form (-128). This means we can only represent up to -128 in this form 10000000 but up to 127 positive form 01111111 using only 8 bits.

For example: Write -28 in binary form

	+/- 128	64	32	16	8	4	2	1
28=	0	0	0	1	1	1	0	0
Invert>>>	1	1	1	0	0	0	1	1
Add binary one to this to get -28	1	1	1	0	0	1	0	0

	+/- 128	64	32	16	8	4	2	1
53=	0	0	1	1	0	1	0	1
Invert>>	1	1	0	0	1	0	1	0
Add binary one to this to get - 53	1	1	0	0	1	0	1	1

How to add binary numbers:

Adding binary numbers in a normal form 8 bit form is simple.

- 0 + 0 = 0
- 1 + 0 = 1
- 1 + 1 = 0 (and then carry one to the next significant column)
- 1+1+1 = 1 (and then carry one to the next significant column)

For example, what is 10111010 + 01011011 (*Subscripts = carried numbers)*

	^{1}	$^{1}1$	$^{1}0$	$^{1}1$	$^{1}1$	1	$^{1}0$	1	0
+		0	1	0	1	1	0	1	1
=	1	0	0	0	1	0	1	0	1

The result is longer than 8 bits (1 byte) since an overflow occurred - a binary digit was carried over into the 9 bit column. The computer requires an extra bit in order to store this result. If it attempts to store this as an 8 bit binary value then the computer will crash with a runtime error.

How to subtract binary numbers:

Subtracting binary numbers follows a similar system to subtracting decimal numbers. The difference being that when a 1 is subtracted from a 0, 2 lots of 1 are carried over from the next significant column. This happens since each binary digit has a value double that of the digit less significant than it to the left.
If there is no binary 1 available in the next column (to the left) to carryover then we look to the next digit on the left where a 1 is present. This is carried over and keeps on getting carried over to the appropriate column until the subtraction can take place – remembering it brings two 1's each time but one is removed when a further two are taken to the next column to the left.

For example, what is 10110110 - 00111001 (*Subscripts = carried numbers)*

		~~1~~	$^{1,\cancel{1}}$0	^{1,1}~~1~~	^{1,1}~~1~~	^{1,1}0	1	~~1~~	$^{1, 1}$0
-		0	0	1	1	1	0	0	1
=		0	1	1	1	1	1	0	1

Example 2: what is 01001101 – 00110101 (*Subscripts = carried numbers)*

		0	~~1~~	$^{1,\cancel{1}}$0	^{1,1}0	1	1	0	1
-		0	0	1	1	0	1	0	1
=		0	0	1	1	1	0	0	0

Hexadecimal

Hexadecimal is a system of numerical notation that uses base 16. Hexadecimal digits are represented by numbers 0 to 9 – numbers greater than 9 then use the alphabetical letters A to F. This means there are 16 different numbers to represent the digits (including 0).

In Hexadecimal, the digits represent the number of 16's we have in that particular place. Like with binary, moving from left to right consecutively will increment the index of 16 by each time. The least significant digit always represents 16^0 (= 1) and so the value of the digit for this is the same as in decimal. The next digit to the left of this represents the number of 16^1's present (i.e. the digit * 16). Likewise, the next digit along will represent a higher power of 16 (16^2 = 256).
This pattern continues however since each digit in hexadecimal represents a specified power of 16, less digits are required to store larger numbers than with a base 2 or base 10 system. Therefore, exam questions tend to only have up to two hexadecimal digits.

- **Hexadecimal is often used to represent binary numbers** (particularly for long values representing RGB **colours**) because **each Hexadecimal digit represents 4 bits** and **so fewer digits are required** to represent longer binary values. This means it is **easier to remember**, **enter in/type** and **reduces the chance of errors** or **loss of integrity**. They are also used since converting to and from Hexadecimal and binary is easy.

Converting Hexadecimal to Denary

E.g. 0 1 2 3 4 5 6 7 8 9 A B C D E F (16 values all together)

	16^5	16^4	16^3	16^2	16^1	16^0
HEX #	0	0	F	C	E	C
Decimal:			($15*16^3$)	($12*16^2$)	(14*16)	(12*1)
Decimal:	6,368,780					

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

More appropriate example for an exam:

What is F4 in decimal?

	16^1	6^0
HEX #	F	4
Calculation:	(15*16) +	(4*1)
Decimal =	244	

```
  X   10    6
10 100    60
5    50     30      Sum = 240 + 4 = 244
```

(In the real world, Hexadecimal numbers start with a pound sign (or hashtag), and are followed by six letters or numbers)

Converting Denary to Hexadecimal

For denary numbers less than 255 (since 2 Hex digits in questions can only represent up to 8 bits).

If the number is greater than 16 then we will divide it by 16 and ignore any remainders. The result will be the most significant digit of the Hex value. The remainder will then form the least significant digit.

Example 1) 77 (77/16 = 4 remainder 13 (13 = D)   SO ***4D***
Example 2) 124 (124/16 = 7 remainder 12 (12 = C) SO ***7C***
*Example 3) 11 (11/16 does not go (hence 0*16 = 0) remainder 11) SO =* ***0B***

Note: Since conversion between Hex and binary is so easy, hard arithmetic can be avoided by converting the denary to binary then to Hex. **(refer to the below)**

Converting Binary to hexadecimal

If there is a binary value (say 8 bits), it is split into 2 nibbles (4 bits each). Each nibble is then converted to a denary equivalent, and then to a hexadecimal value - but ignoring that the most significant digit represents a given number of 16's (i.e. all digits are treated as $16^{0)}$.

For example, convert 10010110 to Hexadecimal

1	0	0	1		0	1	1	0
8	0	0	1		0	4	2	0
= 9 = HEX #9					= 6 = HEX #6			
# Hexadecimal = 96								

Example 2) 01001011   = 0100 and 1011 so 0100 = ***4*** *and 1011 = 11 (**B**) so when joined together =* ***4B***

*Example 3) 11001001 = 1100 and 1001 so 1100 = 12 (**C**) and 1001 =* ***9*** *so when joined =* ***C9***

Hexadecimal to binary

For a simple digit hex value, split into the two digits. Then convert each of the digits in to a denary value and then into a nibble (4 bit binary value). Once the two nibbles are obtained, join them together remembering to put the most significant before the least significant.
For example, convert DE into binary
DE → D & E.   D = 13   E = 14

2^3	2^2	2^1	2^0		2^3	2^2	2^1	2^0
1	1	0	1		1	1	1	0
Joining nibbles:	= 1101110							