technoteach

technocamps

UNDEB EWROPEAIDD
EUROPEAN UNION

Llywodraeth Cymru
Welsh Government

Cronfa Gymdeithasol Ewrop
European Social Fund

Prifysgol Abertawe
Swansea University

Cardiff University
Prifysgol Caerdydd

Prifysgol Bangor University

Cardiff Metropolitan University
Prifysgol Metropolitan Caerdydd

i.t.wales

Prifysgol Aberystwyth University

Prifysgol Glyndŵr Wrecsam
Wrexham Glyndŵr University

University of South Wales
Prifysgol De Cymru

Binary, Decimal and Hex

Learning Objectives

From the new GCSE Computer Science specification (2017):

1. **Use and convert between denary, binary (up to 16 bits) and hexadecimal counting systems.**
   - (Direct conversion or using an intermediate base are both acceptable)
2. **Explain the use of hexadecimal notation as shorthand for binary numbers.**
   - Understand that a binary number is far easier to use as the shorter hexadecimal notation. E.g. 110110101011110_2 = $6D5E_{16}$
3. **Use arithmetic shift functions and explain their effect.**
   - Understand the effect of shifts both left and right.
   - i.e. Shifting one place to the left: 00001100 -> 00011000 equiv. to multiplying by 2

Learning Objectives

From the new GCSE Computer Science specification (2017):

4. Apply binary addition techniques.

- Use the add/divide/remainder method to add binary numbers.

5. Explain the concept of Overflow

- Understand that if the result of an addition or shift process results in a number that is too large to fit in the space available then an overflow has occurred.
- For example if we tried to store the addition of the following two 8 bit numbers in an 8 bit register.

```
 11011011
 11111011 +
111010110
```

Introducing Binary

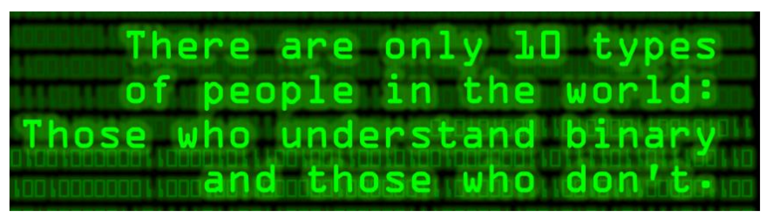
Computers use binary – the digits 0 and 1 to store data.

A binary digit, or bit, is the smallest unit of data in computing.

Binary numbers are made up of binary digits (bits), eg the binary number 1001

This is a 9 in our familiar everyday decimal number system.

Computer processors are made up of billions of tiny switches called transistors.



Transistors

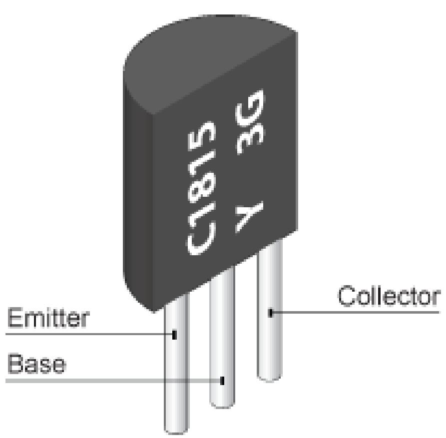
Transistors are essentially "on and off" switches. If the switch is on it has a binary value of 1, if the switch is off it has a value of 0.

With only one transistor we can produce a single bit – either 1 or 0.

But we can build up longer strings of binary numbers with more.

i.e. the binary number 11010010.

This is 210 in our familiar everyday decimal number system.



Linking to Other Subjects

When computers were first being developed, data was entered by the use of punched cards or punched tape, as well as by flicking switches.

Some famous examples include: Charles Babbage's Analytical Engine (in 1837) and the Colossus (used during the Second World War.

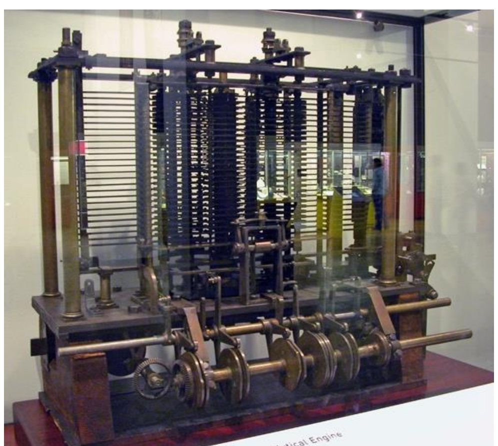


Figure 1. "Babbage's Analytical Engine, 1834-1871. (Trial model)". Science Museum. Retrieved 2017-08-23.

Figure 2. originally posted to **Flickr** as Punched cards for programming the Analytical Engine, 1834-71 , Karol Lorentey

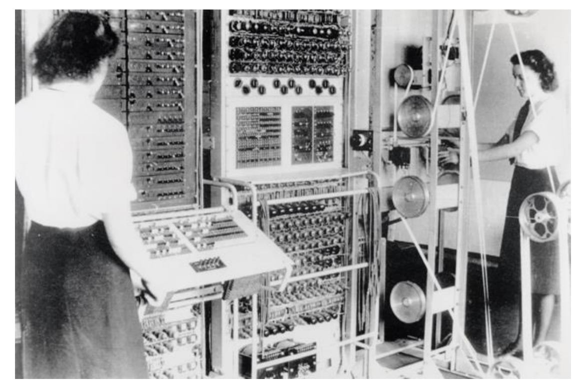


Figure 3. A Colossus Mark 2 computer being operated by Wrens Dorothy Du Boisson (left) and Elsie Booker. Wikipedia. Retrieved 2017-09-01

Bits and Bytes

Bits can be grouped together to make them easier to work with. A group of 8 bits is called a byte.

Nibble - 4 bits (half a byte)

Byte - 8 bits

Kilobyte (KB) - 1024 bytes (or 1024 x 8 bits)

Megabyte (MB) - 1024 kilobytes (or 1048576 bytes)

Gigabyte (GB) - 1024 megabytes

Terabyte (TB) - 1024 gigabytes

Amount of Storage Space Required

Data	Storage
One extended-ASCII character in a text file (eg 'A')	1 byte
The word 'Monday' in a document	6 bytes
A plain-text email	2 KB
64 pixel x 64 pixel GIF	12 KB
Hi-res 2000 x 2000 pixel RAW photo	11.4 MB
Three minute MP3 audio file	3 MB
One minute uncompressed WAV audio file	15MB
One hour film compressed as MPEG4	4 GB

Figure 4. Table of data types and storage required (from BBC Bitesize. Retrieved 17-09-01)

Binary and Denary

As we know, computers use binary – 0 and 1.

In everyday life, we use numbers based on combinations of the digits between 0 and 9. This is known as decimal, denary or base 10.

A number base tells us how many digits are available in a numerical system.

Denary/decimal is known as base 10 as there are 10 digits available (0 – 9).

Binary is known as base 2 as there are only 2 digits available (0, 1).

Converting Between Binary and Denary

All denary numbers have a binary equivalent and it is possible to convert between denary and binary.

To do this we break down the numbers into their ''place values''

Place Values: Denary

Using the denary system, 7242 reads as seven thousand, two hundred and forty two. We can break this down:

- Seven thousands
- Two hundreds
- Four tens
- Two ones

Each number has a place value which we can arrange in columns. Each column is a power of ten in the base 10 system.

Thousands 1000s (10^3)	Hundreds 100s (10^2)	Tens 10s (10^1)	Ones 1s (10^0)
7	2	4	2

Place Values: Denary

Thousands 1000s (10^3)	Hundreds 100s (10^2)	Tens 10s (10^1)	Ones 1s (10^0)
7	**2**	**4**	**2**

We can think of this as:

(**7** x 1000) + (**2** x 100) + (**4** x 10) + (**2** x 1) = **7242**

Similarly, All number systems can be thought of in the same way.

Binary Place Values

With binary, we break the number down into columns, but each column is a power of two instead of a power of 10. For example the number 1001.

Eights 8s (2^3)	Fours 4s (2^2)	Twos 2s (2^1)	Ones 1s (2^0)
1	**0**	**0**	**1**

(**1** x 8) + (**0** x 4) + (**0** x 2) + (**1** x 1) = **9**

Converting Binary to Denary

Calculating larger numbers e.g. 10101000

We need more place values of multiples 2.

128 (2^7)	64 (2^6)	32 (2^5)	16 (2^4)	8 (2^3)	4 (2^2)	2 (2^1)	1 (2^0)
1	0	1	0	1	0	0	0

In denary the sum is calculated as:

(**1** x 128) + (**0** x 64) + (**1** x 32) + (**0** x 16) + (**1** x 8) + (**0** x 4) +

+ (**0** x 2) + (**0** x 1) = **128 + 32 + 8** = **168**

Questions 1

Try converting these binary numbers into denary:

1. 1110
2. 01010
3. 11101
4. 1000001
5. 11111111

Converting Denary to Binary: 2 Methods

There are two methods for converting a denary (base 10) number to binary (base 2).

The first method involves repeatedly dividing by 2 and using the remainder.

The second method involves repeatedly subtracting the biggest 2^n value.

Remember: To check if your calculations are right, just convert your answer back to denary!

Method 1 Example: Dividing by 2

To convert 83 to a binary number:

83 ÷ 2 = **41** remainder	**1**	(**Divide the starting number by 2**)
41 ÷ 2 = **20** remainder	**1**	(**Divide the answer by 2**)
20 ÷ 2 = **10** remainder	**0**	(**and again**)
10 ÷ 2 = **5** remainder	**0**	(**and again**)
5 ÷ 2 = **2** remainder	**1**	(**and again**)
2 ÷ 2 = **1** remainder	**0**	(**and again**)
1 ÷ 2 = 0 remainder	**1**	(**last time as this equals 0)**

The final step is to reverse the order of the remainders:

So 83 in binary is: 1010011

Questions 2

Using the divide by 2 method, convert these numbers into binary:

1. 28
2. 43
3. 99

Method 2 Example
Subtracting the biggest 2^n value

~~128 (2^7)~~	64 (2^6)	32 (2^5)	16 (2^4)	8 (2^3)	4 (2^2)	2 (2^1)	1 (2^0)
	1	0	1	0	1	0	0

To convert 84 to a binary number:

$84 - 64 = \mathbf{20}$ **(Subtract by the largest place value + mark the place value with a 1)**

$\mathbf{20} - 16 = \mathbf{4}$ **(subtract answer by largest place value)**

$\mathbf{4} - 4 = \mathbf{0}$ **(last time as it equals 0)**

Fill in any remaining place values with 0s:

So 84 in binary is: 1010100

Question 3 – Slide 20

Using the subtraction of highest 2^n method, convert these numbers into binary.

1. 37
2. 65
3. 134

Hexadecimal (Base 16)

Hexadecimal is another number system, this time in base 16.

Its 16 symbols are made up of the numbers 0-9 (like denary) followed by the letters A-F.

Denary	Hexadecimal
0	0
1	1
…	…
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Converting from Denary to Hex

To convert from denary to Hex you need to remember the equivalent binary numbers for the first 16 Hex digits.

Denary	Binary	Hex
13 = (1 x 8) + (1 x 4) +(0 x 2) + (1 x 1)	1101	D
19 = (1 x 16) + (0 x 8) + (0 x 4) + (1 x 2) + (1 x 1)	10011	??

Since 19 is more than the value of F in Hex, we have to combine hex digits.

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010

Hex	Binary
B	1011
C	1100
D	1101
E	1110
F	1111

Converting from Denary to Hex

So we know $\mathbf{19_{10}}$ is $\mathbf{10011_2}$ in binary. This is more than a 4-bit binary number and is therefore more than a single digit in Hex.

If we split the binary digits into groups of 4 like so:

0001 | 0011

We tend to add 0s to the front to make it a group of 4 digits.

We can now assign these groups individual hex values. $\mathbf{0001_2 = 1_{16}}$ and $\mathbf{0011_2 = 3_{16}}$ so combining we get :

$$\mathbf{19_{10} = 13_{16}}$$

More Examples

1 B

$27_{10} = 11011_2 = 0001 \mid 1011 = 1B_{16}$

E B

$235_{10} = 11101011_2 = 1110 \mid 1011 = EB_{16}$

3 1 6 A

$11000101101010_2 = 0011 \mid 0001 \mid 0110 \mid 1010$

$= 316A_{16}$

Questions 4 – Slide 25

Convert these numbers into Hexadecimal.

1. 12_{10}
2. 37_{10}
3. 10101010_2
4. 11011010_2
5. 1010111110010001_2

Converting from Hex to Denary (Via Binary)

First we convert hex to binary and then from binary to denary:

$2D_{16}$ = 0010 | 1101 = 00101101_2

128 (2^7)	64 (2^6)	32 (2^5)	16 (2^4)	8 (2^3)	4 (2^2)	2 (2^1)	1 (2^0)
0	0	1	0	1	1	0	1

(1 x 32) + (1 x 8) + (1 x 4) + (1 x 1) = 45_{10}

Questions 5 – Slide 27

Try Converting these Hexadecimal values into Denary (via Binary):

1. $4E_{16}$
2. $7A_{16}$
3. $1EB_{16}$

So Why Use Hexadecimal?

Simply because Hexadecimal is more convenient due to it being easier to read and more compact. Compare this value in both bases.

$$\mathbf{1101111000111010110100 1101_2 = 378EB4D_{16}}$$

Which is easier to decipher and use for humans?

If you wanted to input this value into a computer, it is much easier to input the Hex value and would lead to fewer mistakes.

Example: Using Colours

When using a certain colour on a screen we have to tell the computer which colour we want to render.

If we want red we could use the binary code for it:

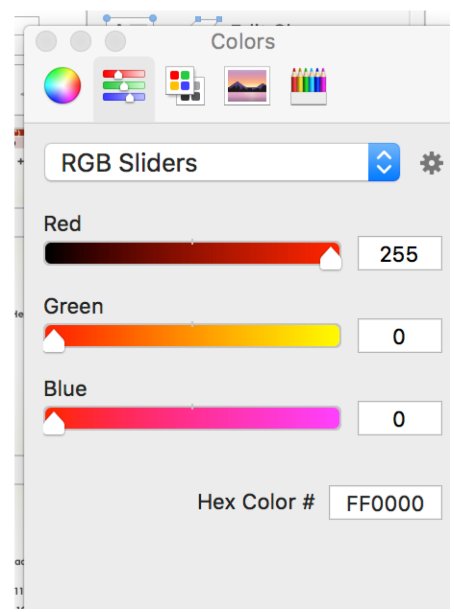
111111110000000000000000

But imagine typing that in, it's very easy to make a mistake.

For convenience, we use Hex colour codes instead.

This same colour in Hex is

1111|1111|0000|0000|0000|0000 = #FF0000



Binary Shift Functions

A shift function means shifting all digits in a binary number either to the left or the right.

Example: If I shift **00001011** to the left once, it becomes:

00010110 **(all of the digits move to the left)**

The original number $\mathbf{00001011_2 = (1 \times 8) + (1 \times 2) + (1 \times 1) = 11_{10}}$

Shifted number $\mathbf{00010110_2 = (1 \times 16) + (1 \times 4) + (1 \times 2) = 22_{10}}$

So we can see that shifting once to the left is equivalent to multiplying by 2.

What About Shifting to the Left Twice?

Original number **00001011_2 = (1 x 8) + (1 x 2) + (1 x 1) =** 11_{10}

x4 x4 x4 x4

Shifted left twice: **00101100_2 = (1 x 32) + (1 x 8) + (1 x 4) =** 44_{10}

So shifting left twice is equivalent to multiplying by 4.

Shifting left 3 times is equivalent to multiplying by 8.

Shifting left 4 times is equivalent to multiplying by 16.

What About Shifting to the Right?

This has the opposite effect. Shifting to the right once is equivalent to dividing by 2.

Shifting to the right twice is equivalent to dividing by 4.

3 times -> dividing by 8.

So Shifting Left by n Amount Leads to Multiplying by 2^n.

$n =$	Shift left: Multiply number by	Shift right: Divide number by
1	$2^1 = 2$	$2^1 = 2$
2	$2^2 = 4$	$2^2 = 4$
3	$2^3 = 8$	$2^3 = 8$
4	$2^4 = 16$	$2^4 = 16$
5	$2^5 = 32$	$2^5 = 32$
6	$2^6 = 64$	$2^6 = 64$
7	$2^7 = 128$	$2^7 = 128$

Arithmetic Shift

Shifting Left:

- The least significant bit is always a 0

10010110

100101100

Shifting Right:

- The most significant bit for the answer is copied from the original binary value's most significant bit

10010110

11001011

Two's Complement

The first bit is a signed bit:

- 0 = positive
- 1 = negative

Overflow and Underflow

Say we only have 8 bits to store an integer.

What's the issue with this left shift?

10010110

Overflow and Underflow

Say we only have 8 bits to store an integer.

What's the issue with this left shift?

10010110

00101100

Overflow and Underflow

Say we only have 8 bits to store an integer.

What's the issue with this left shift?

10010110

00101100

What about this subtraction?

00000001

00000010

Overflow and Underflow

Say we only have 8 bits to store an integer.

What's the issue with this left shift?

10010110

00101100

What about this subtraction?

00000001

00000010

11111111

Binary Addition

Binary addition is much like normal addition we did in primary school, except in base 2 instead of base 10. The difference being that if the numbers add to make 2 or more, then we must carry a 1 to the next place value.

Example:

1 + 1 = 2_{10} in decimal
1 + 1 = 10_2 in binary

1 + 1 + 1 = 3_{10} in decimal
1 + 1 + 1 = 11_2 in binary

```
  00110011   +
  10101000
  --------
  11011011
  --------
   1
```

Example

```
   10011010
   10011001   +
 __________
  100110011
 __________
  1     1 1
```

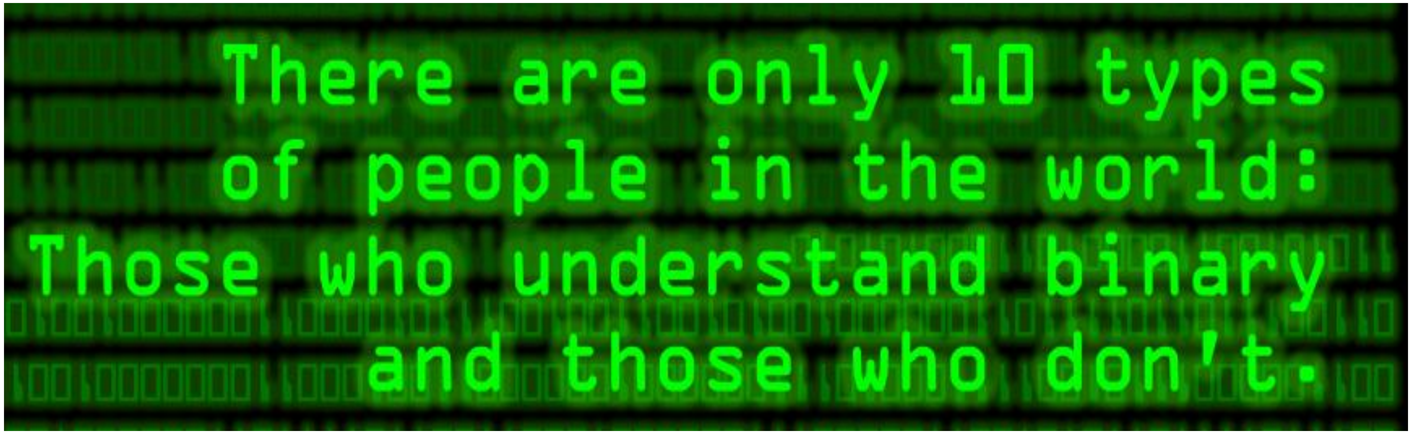
Question 6 – Slide 36

Add the following binary numbers and then convert the numbers and your answers to decimal to see if they match up:

1. 1011 + 0110
2. 1101 + 1100
3. 11001011 + 00111011

Awful Joke

Hopefully you're in the group of those 10_2 that do!



Q1 Answers – Slide 15

Try converting these binary numbers into denary:

1. 1110
2. 01010
3. 11101
4. 1000001
5. 11111111

1. (1 x 8) + (1 x 4) + (1 x 2) = 14
2. (1 x 8) + (1 x 2) = 10
3. (1 x 16) + (1 x 8) + (1 x 4) + (1 x 1) = 29
4. (1 x 64) + (1 x 1) = 65
5. (1 x 128) + (1 x 64) + (1 x 32) + (1 x 16) + (1 x 8) + (1 x 4) + (1 x 2) + (1 x 1) = 255

Q2.1 Answer – Slide 18:

To convert 28 to a binary number:

28 ÷ 2 = **14** remainder	**0**	(**Divide the starting number by 2**)
14 ÷ 2 = **7** remainder	**0**	(**Divide the answer by 2**)
7 ÷ 2 = **3** remainder	**1**	(**and again**)
3 ÷ 2 = **1** remainder	**1**	(**and again**)
1 ÷ 2 = 0 remainder	**1**	**(last time as this equals 0)**

The final step is to reverse the order of the remainders:

So 28 in binary is: 11100

Q2.2 Answer – Slide 18:

To convert 43 to a binary number:

43 ÷ 2 = **21** remainder **1** (**Divide the starting number by 2**)

21 ÷ 2 = **10** remainder **1** (**Divide the answer by 2**)

10 ÷ 2 = **5** remainder **0** (**and again**)

5 ÷ 2 = **2** remainder **1** (**and again**)

2 ÷ 2 = **1** remainder **0** (**and again**)

1 ÷ 2 = 0 remainder **1** (**last time as this equals 0**)

The final step is to reverse the order of the remainders:

So 43 in binary is: 101011

Q2.3 Answer – Slide 18:

To convert 99 to a binary number:

99 ÷ 2 = **49** remainder **1** (**Divide the starting number by 2**)

49 ÷ 2 = **24** remainder **1** (**Divide the answer by 2**)

24 ÷ 2 = **12** remainder **0** (**and again**)

12 ÷ 2 = **6** remainder **0** (**and again**)

6 ÷ 2 = **3** remainder **0** (**and again**)

3 ÷ 2 = 1 remainder **1** (**and again**)

1 ÷ 2 = 0 remainder **1** (**last time as this equals 0**)

The final step is to reverse the order of the remainders:

So 99 in binary is: 1100011

Q3.1 Answer – Slide 20:

~~128~~	~~64~~	32 (2^5)	16 (2^4)	8 (2^3)	4 (2^2)	2 (2^1)	1 (2^0)
		1	0	0	1	0	1

To convert 37 to a binary number:

$37 - 32 = \mathbf{5}$ (**Subtract by the largest place value + mark the place value with a 1**)

$\mathbf{5} - 4 = 1$ (**subtract answer by largest place value**)

$\mathbf{1} - 1 = \mathbf{0}$ (**last time as it equals 0**)

Fill in any remaining place values with 0s:

So 37 in binary is: 100101

Q3.2. Answer – Slide 20:

128 (2^7)	64 (2^6)	32 (2^5)	16 (2^4)	8 (2^3)	4 (2^2)	2 (2^1)	1 (2^0)
	1	0	0	0	0	0	1

To convert 65 to a binary number:

$65 - 64 = \mathbf{1}$ **(Subtract by the largest place value + mark the place value with a 1)**

$\mathbf{1} - 1 = \mathbf{0}$ **(last time as it equals 0)**

Fill in any remaining place values with 0s:

So 65 in binary is: 1000001

Q3.3 Answer – Slide 20

128 (2^7)	64 (2^6)	32 (2^5)	16 (2^4)	8 (2^3)	4 (2^2)	2 (2^1)	1 (2^0)
1	0	0	0	0	1	1	0

To convert 134 to a binary number:

$134 - 128 = 6$ (**Subtract by the largest place value + mark the place value with a 1**)

$6 - 4 = 2$ (**subtract answer by largest place value**)

$2 - 2 = 0$ (**last time as it equals 0**)

Fill in any remaining place values with 0s:

So 134 in binary is: 10000110

Q4 Answers – Slide 25:

Convert these numbers into Hexadecimal.

1. 12_{10} $= 1100_2 = C_{16}$
2. 37_{10} $= 100101_2 = 0010|0101 = 25_{16}$
3. $10101010_2 = 1010|1010 = AA_{16}$
4. $11011010_2 = 1101|1010 = DA_{16}$
5. $1010111110010001_2 = 1010|1111|1001|0001 = AF91_{16}$

Q5 Answers – Slide 27

Try converting these Hexadecimal values into Denary (via Binary):

1. **$4E_{16}$ = 0100 | 1110 = 01001110 = 64 + 8 + 4 + 2 = 78**
2. **$7A_{16}$ = 0111 | 1010 = 01111010 = 64 + 32 + 16 + 8 + 2 = 122**
3. **$1EB_{16}$ = 0001 | 1110 | 1011 = 000111101011 =**
   = 256 + 128 + 64 + 32 + 8 + 2 + 1 = 491

Q6 Answers – Slide 36

1. 1011 + 0110

11	1011
6	0110

17	10001
	1 1 1

2. 1101 + 1100

13	1101
12	1100

25	11001
	1 1

3. 11001011 + 00111011

203	11001011
59	00111011

262	100000110
	1 1 1 1 1 1 1

You can check by working out the decimal equivalents.