FACTFILE: GCSE DIGITAL TECHNOLOGY

Unit 4

DIGITAL AUTHORING CONCEPTS

Rewarding Learning

Digital Data

Learning Outcomes

Students should be able to:

- describe how a number is converted to a binary pattern for storage in a computer;
- demonstrate understanding of the following units of data;
  - Bit;
  - Nibble;
  - Byte;
  - Kilobyte;
  - Megabyte:
  - Gigabyte; and
  - Terabyte;
- demonstrate understanding of the following types of character representation:
  - ASCII (7-bit and 8-bit); and
  - Unicode;
- demonstrate understanding of and use number representation and convert between denary, binary and hexadecimal;
- perform the addition of two bytes and explain the meaning of overflow.

Content

- How a number is converted to a binary pattern
- Description of the following units of data:
  - Bit, nibble, byte, kilobyte, megabyte, gigabyte, and terabyte
- Description of the following types of character representation:
  - ASCII and Unicode
- Number representation and converting between denary, binary and hexadecimal
- Addition of two bytes
- Overflow

Converting to a Binary Pattern

In all modern computers memory can either be ON or OFF. On is indicated by 1 and off is indicated by 0. Data is stored character by character with each character being represented by a code. The code will consist of a string of 1's and 0's. Each 1 or 0 is called a bit – **B**inary dig**IT**. Therefore if the character A is typed, the code 01000001 is sent to the CPU.

To understand the binary system it is important to think about how the decimal (denary) system works. For example the number 134 represents one hundred, three tens and four ones

100	10	1
1	3	4

100 + 30 + 4 = 134

Moving from left to left each digit is worth ten times the previous one, this is a base ten number system. However the binary system uses a base two system, therefore as we move from right to left each digit is worth twice as much as the previous one.

Again use the example of 134:

128 + 4 + 2 = 134

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

Units of Data

Bit

The word bit stands for Binary digit. It is a single digit in a binary number. It will either be 0 or 1. Bits are the smallest unit of storage on a computer. All data is stored as binary.

Byte

This is a group of 8 bits, which is made up of a series of 0's and 1's. Each byte represents a single character. The storage capacity of all computers is measured in multiples of ytes – Kilobytes, Megabytes, Gigabytes and Terabytes.

Nibble

A nibble is half a byte, only consisting of 4 binary digits. It can be represented by one hexadecimal digit.

Kilobyte (KB)

This is larger than a Byte. 1 Kilobyte is equal to 1, 024 Bytes.

Megabyte (MB)

This is larger than a Kilobyte. 1 Megabyte is equal to 1, 024 Kilobytes or 1, 048, 576 Bytes

Gigabyte (GB)

This is larger than a Megabyte. 1 Gigabyte is equal to 1, 024 Megabytes

Terabyte (TB)

This is larger than a Gigabyte. 1 Terabyte is equal to 1,024 Bigabytes or one trillion Bytes

Character Representation

ASCII (7-bit and 8-bit)

ASCII stands for American Standard Code for Information Interchange. It's a 7-bit character code where every single bit represents a unique character. The first 32 characters in ASCII are unprintable and used for controlling peripheral devices such as printers. Characters 32 to 127 represent upper and lower case letters, numbers, punctuation marks and symbols.

The 8-Bit ASCII codes only cover the unaccented characters of the English language.

Below is a partial table for the ASCII codes for common symbols and letters of the alphabet:

DEC	OCT	HEX	BIN	Symbol	HTML Number	HTML Name	Description
32			00100000				Space
33	041	21	00100001	!	&#33;		Exclamation mark
34	042	22	00100010	"	&#34;		Speech marks
35	043	23	00100011	#	&#35;		Hash Tag
36	044	24	00100100	$	&#36;		Dollar
37	045	25	00100101	%	&#37;		Percentage
38	046	26	00100110	&	&#38;	&amp;	Ampersand
39	047	27	00100111	'	&#39;		Single quote
40	050	28	00101000	(	&#40;		Open parenthesis (or open bracket)
41	051	29	00101001	)	&#41;		Close parenthesis (or close bracket)
The cells above show the ASCII codes for some of the most common used symbols, below show the ASCII codes for the letters of the alphabet.							
65	101	41	01000001	A	&#65;		Uppercase A
66	102	42	01000010	B	&#66;		Uppercase B
67	103	43	01000011	C	&#67;		Uppercase C
68	104	44	01000100	D	&#68;		Uppercase D
69	105	45	01000101	E	&#69;		Uppercase E
70	106	46	01000110	F	&#70;		Uppercase F
71	107	47	01000111	G	&#71;		Uppercase G
72	110	48	01001000	H	&#72;		Uppercase H
73	111	49	01001001	I	&#73;		Uppercase I
74	112	4A	01001010	J	&#74;		Uppercase J
75	113	4B	01001011	K	&#75;		Uppercase K
76	114	4C	01001100	L	&#76;		Uppercase L

This table shows some of the ASCII code for typical characters on a keyboard.

Unicode

Unicode is a world-wide character encoding standard. Compared to older encoding systems, Unicode simplifies character and string manipulation. Unicode enables universal data exchange by using a single binary file for every possible character code. Remember other langauges have a further variety of characters they use and Unicode takes this into account.

Number Representation

Any number no matter how big or small can be represented in various ways. Numbers can cause problems for computer scientists because they behave in unpredictable ways. They can be stored as denary/decimal numbers, binary, octal and hexadecimal. (Octal will not be discussed in this course)

Denary

This is the system we are all familiar with using the digits 0 to 9. It uses a base ten number sytem. It is commonly referred to as decimal notation.

Binary

This is the system using the digits 0 and 1. It is the most convenient way of representing numbers on a computer using a base two number system. Computers complete arithmetic using binary numbers.

Whether a number is a whole, positive, negative or a fraction it must be converted to a binary pattern for storage and manipulation on a computer. For each number there are a limited number of binary patterns available. The table below demonstrates the patterns available:

1 bit	Example 1	2 patterns (0 and 1)
2 bits	Example 11	4 patterns (00, 01,10, 11)
3 bits	Example 101	8 patterns (000, 001, ...111)
4 bits	Example 1101	16 patterns (0000, 0001,... 1111)
8 bits	Example 11011001	128 patterns
32 bits		4, 294, 967,296 patterns

Hexadecimal

This system is a base 16 number system. To account for all 16 possibilities it uses the digits 0 – 9 and characters A–F. The advantage of this system is that it is easy to convert from binary to hexadecimal and it is easy to read and write down the hexadecimal equivalent of each number.

Converting between Denary, Binary and Hexadecimal

Denary to Binary conversion

123 (denary)	1	2	3
	↓	↓	↓
Binary Codes	0001	0010	0011
123 = 000100100011			

How to convert decimal to binary

To convert decimal to binary, divide the decimal number by 2 and get the binary digits from the remainders:

Example

Convert 25 to binary:

Division by 2 Quotient	Division by 2 Remainder	Bit Number
25	–	–
12	1	0
6	0	1
3	0	2
1	1	3
0	1	4

So $25_{10} = 11001_2$

Binary to Denary conversion

How to convert binary to decimal

For binary number with n digits:

$$d_{n-1} \ldots d_3\, d_2\, d_1\, d_0$$

The decimal number is equal to the sum of binary digits (d_n) times their power of 2 (2^n):

$$\text{decimal} = d_0 \times 2^0 + d_1 \times 2^1 + d_2 \times 2^2 + \ldots$$

Example: find the decimal value of 111001_2:

binary number:	1	1	1	0	0	1
power of 2:	2^5	2^4	2^3	2^2	2^1	2^0

$$111001_2 = 1.2^5+1.2^4+1.2^3+0.2^2+0.2^1+1.2^0 = 57_{10}$$

Denary to hex

How to convert from decimal to hex

To convert decimal to hex, divide the decimal number by 16 and get the hex digits from the remainders:

Example
Convert 7562 to hex:

Divison by 16 Quotient	Divison by 16 Remainder	Hex Digit Number
7562	–	–
472	10	0
29	8	1
1	13	2
0	1	

So 7562_{10} = $1D8A_{16}$

Hex to denary

How to convert from hex to decimal

A regular decimal number is the sum of the digits multiplied with power of 10.
137 in base 10 is equal to each digit multiplied with its corresponding power of 10:

$$137_{10} = 1\times10^2+3\times10^1+7\times10^0 = 100+30+7$$

Hex numbers are read the same way, but each digit counts power of 16 instead of power of 10.
For hex number with n digits:

$$d_{n-1} \dots d_3\, d_2\, d_1\, d_0$$

Multiply each digit of the hex number with its corresponding power of 16 and sum:

$$\text{decimal} = d^{n-1}\times16^{n-1}+\dots+d_3\times16^3+d_2\times16^2+d_1\times16^1+d_0\times16^0$$

Example #1
3B in base 16 is equal to each digit multiplied with its corresponding 16^n:

$$3B_{16} = 3\times16^1+11\times16^0 = 48+11 = 59_{10}$$

Example #2
E7A9 in base 16 is equal to each digit multiplied with its corresponding 16^n:

$$E7A9_{16} = 14\times16^3+7\times16^2+10\times16^1+9\times16^0 =$$
$$57344+1792+160+9 = 59305_{10}$$

Example #3
0.8 in base 16:

$$0.8_{16} = 0\times16^{0}+8\times16^{=1} = 0+0.5 = 0.5_{10}$$

Binary to Hexadecimal conversion

Binary	0111	1011
	$\downarrow$	$\downarrow$
Split into 4 bits	0111	1011
Hexadecimal codes	7	B
So 01111011 binary is equal to 7B Hexadecimal		

Addition of Bytes (Binary Addition)

Binary addition is an easy calculation to perform and very similar to how decimal numbers are added together.

Two numbers are lined up; one under the other, then starting from the right add each column, recording the result and carry as you go along.

There are 4 possibilities when adding:

0+0 = 1
0+1 = 1
1+1 = 2 which is 10 in binary, which is 0 with a carry 1
1+1+1 (carried over) = 3 which is 11 in binary, which is 1 with a carry of 1

For example to add 26 plus 12

26		1	1	0	1	0
12	1	1	1	1	0	0
38	1	0	0	1	1	0

Overflow
Overflow happens when computerised calculations produce an answer that is too big to be represented. Overflow usually happens during multiplication of binary. When overflow happens double length registers are used to hold the data.

A CPU with a capacity of 8 bits can only hold 11111111 in binary, if another 1 was added an overflow message would appear. The error message demonstrates that the register has exceeded its maximum number. Modern computers can now have 64-bit CPU's which means they can handle much larger numbers.

An example of an 8 bit overflow: when adding (denary 255 + 1), We would expect this answer to give us 256.

However an 8 bit overflow means that there is only room to store 8 bits and the 9th is therefore the overflow. In the example below the 1 to the LHS (left hand side) is the overflow so the computer would say the answer is 00000000 unless it is told that it needed more room – hence the overflow bit is considered.

	1	1	1	1	1	1	1	1
+	0 1	0 1	0 1	0 1	0 1	0 1	0 1	1
1	0	0	0	0	0	0	0	0

Exam Questions

1. Convert the following binary numbers to decimal:
   0011 0110 1010 01000001 01000101 (5)

2. Convert the following numbers to binary:
   5 7 1 26 68 137 (6)

3. Convert the following binary numbers to hexadecimal:
   01110101 00001110 11001111 (3)

4. Convert the following hex numbers to binary:
   12 2A BC FF (4)

Bibliography

Books

BCS Glossary of Computing and ICT, 13th ed., BCS Academy Glossary Working Party: Pages – 324, 332, 336, 337, 433,

Computing 3rd Edition, P M Heathcote: Pages 245, 264

Websites

http://ryanstutorials.net/binary-tutorial/binary-arithmetic.php

http://www.bbc.co.uk/education/guides/zjfgjxs/revision/3

http://www.rapidtables.com/convert/number/decimal-to-binary.htm

Images

http://www.ascii-code.com/