Higher
Computing Science

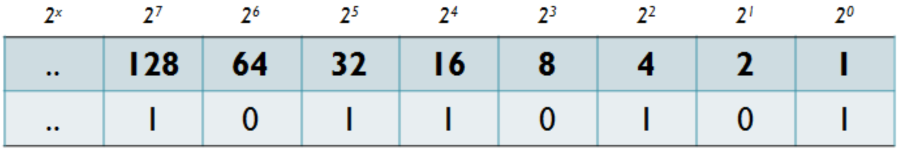
Computer Systems
Data Representation

Name:____________________________

Contents

Integers: Positive Numbers

Numbers are represented in the computer system using binary



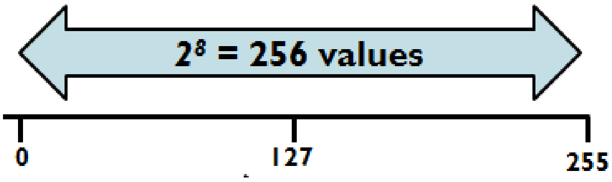
2^x	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
..	128	64	32	16	8	4	2	1
..	1	0	1	1	0	1	0	1

128 + 32 + 16 + 4 + 1 = 181

The column headings with a one beneath them are added together to give the number in denary.

Range of Integers

An 8 bit number can represent the numbers from **00000000** to **11111111** (0 to 255)



The range of integers can be worked out as **0** to **(2^{bits}) -1**

So:

Bits	**Calcualtion**	**Range of Integers**
5 bits	0 to (2^5) -1	0 to 31
10 bits	0 to (2^{10}) -1	0 to 1023
16 bits	0 to (2^{16}) -1	0 to 65534

Integers: Two’s Complement

In two’s complement, the **most significant bit** (largest column) is negative and all other columns are positive.

Example 1 (8 bit)

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

-128 + 32 + 16 + 4 + 1 = -75

-128	64	32	16	8	4	2	1	
1	0	1	1	0	0	1	1	-77
1	0	0	0	0	0	1	1	-125
1	1	1	1	1	1	1	1	-1

In two’s complement, the largest column value is negative and all other columns are positive.

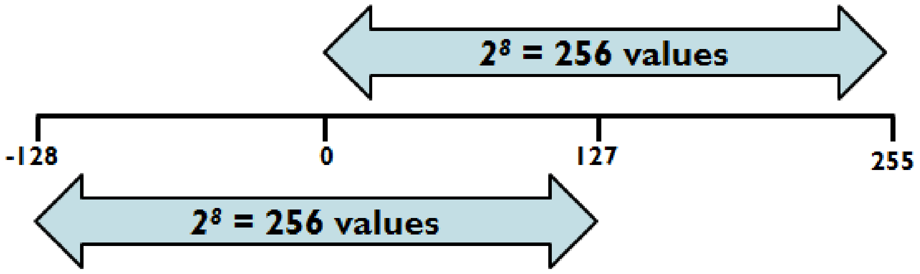
When the most significant bit is a 1, the number is negative. When it is a 0 the number is positive. For this reason it is called the **sign bit.**

Example 2 (5 bit)

-16	8	4	2	1
1	0	1	0	1

-16 + 4 = -12

-16	8	4	2	1	
1	1	1	0	1	-3



Range of Integers

An 8 bit number can represent the numbers from **00000000** to **11111111** (0 to 255)

An 8 bit two's complement number can represent the numbers from **10000000** to **01111111** (-128 to 127)

The range of integers can be worked out as:

negative of (2^{bits-1}) to **positive of (2^{bits-1}) - 1**

So:

Bits	**Calcualtion**	**Range of Integers**
3 bits	minus(2^2) to (2^2) -1	-4 to 3
5 bits	minus(2^4) to (2^4) -1	-16 to 15
8 bits	minus(2^7) to (2^7) -1	-128 to 127

Real Numbers

Real Numbers are numbers that contain a decimal point – they are stored using **floating point notation**.

Example 1

Take the following real number. The aim is to move the point until you end up with 0.1!!!!!!!

10110.001 **0.10110001** x 2^{101}

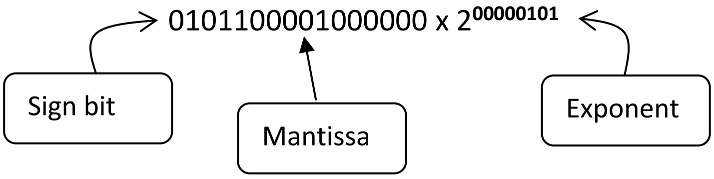
*The point has moved **5** places (hence 101)*

In floating point representation, the number of bits for the mantissa and exponent are usually fixed.

Sign Bit (1 bit)	Mantissa (15bit)	Exponent (8 bit)
0	**101100001**000000	00000101

The sign bit is a 0 if the mantissa is positive and a 1 if it is negative.

This gives the final answer of:



Example 2

Take the following real number. The aim is to move the point until you end up with 0.1!!!!!!!

-0.00011 **0.11** x 2^{-11}

The point has moved -3 places (hence -11)

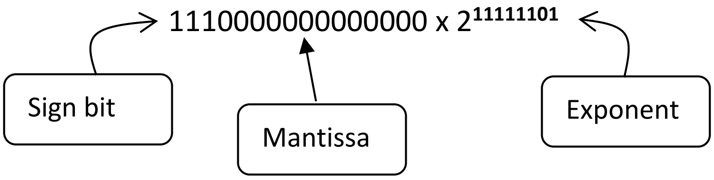
In floating point representation, a **negative mantissa** is represented using a **sign bit**. A **negative exponent** is represented using **two's complement**.

Sign Bit (1 bit)	Mantissa (15bit)	Exponent (8 bit)
1	**11**0000000000000	11111101

11111101 is two's complement for -3

The sign bit is a 0 if the mantissa is positive and a 1 if it is negative.

This gives the final answer of:



The number of bits allocated to the **mantissa** affects the **precision** of the number

The number of bits allocated to the **exponent** affects the **range** of numbers

Increasing bits for MANTISSA = **Increase Precision** = **Reduced bits for EXPONENT** = **Reduced Range**

Reducing bits for MANTISSA = **Reduce Precision** = **Increased bits for EXPONENT** = **Increased Range**

If 16 bits are allocated to a floating point number, they could be allocated as:

$\mathbf{11111111} \times 2^{\mathbf{11111111}}$

The allocation of bits could be changed:

$\mathbf{1111} \times 2^{\mathbf{111111111111}}$

Increased range, reduced precision

$\mathbf{111111111111} \times \mathbf{2}^{\mathbf{1111}}$

Reduced range, increased precision

Revision Questions - Integers

1. Convert the following 16-bit two's complement number into denary.

   1111 1110 1110 1011

2. Two's complement can be used to represent positive and negative integers.

   (a) Convert the denary number −9 into 8-bit two's complement.

(b) State the range of denary values that can be represented using 8-bit two's complement.

3. Write the binary number −0.0011 using floating-point representation. There are 16 bits for the mantissa (including the sign bit) and 8 bits for the exponent.

4. The decimal number 6.125 converted to binary is 110.001. Convert 110.001 to floating-point representation. There are 16 bits for the mantissa (including the sign bit) and 8 bits for the exponent.

5. Convert the following 8-bit two's complement number into denary. 1001 1010

Characters (Text)

The computer can only represent data using binary (1 or 0). To represent characters, they have to be converted into numbers which can then be stored as binary values.

Example:

A = 65 01000001

B = 66 01000010

C = 67 01000011

Text can be encoded like this using:

- **ASCII**
- **Unicode**

ASCII

ASCII (American Standard Code for Information Interchange) is a standard format for encoding text.

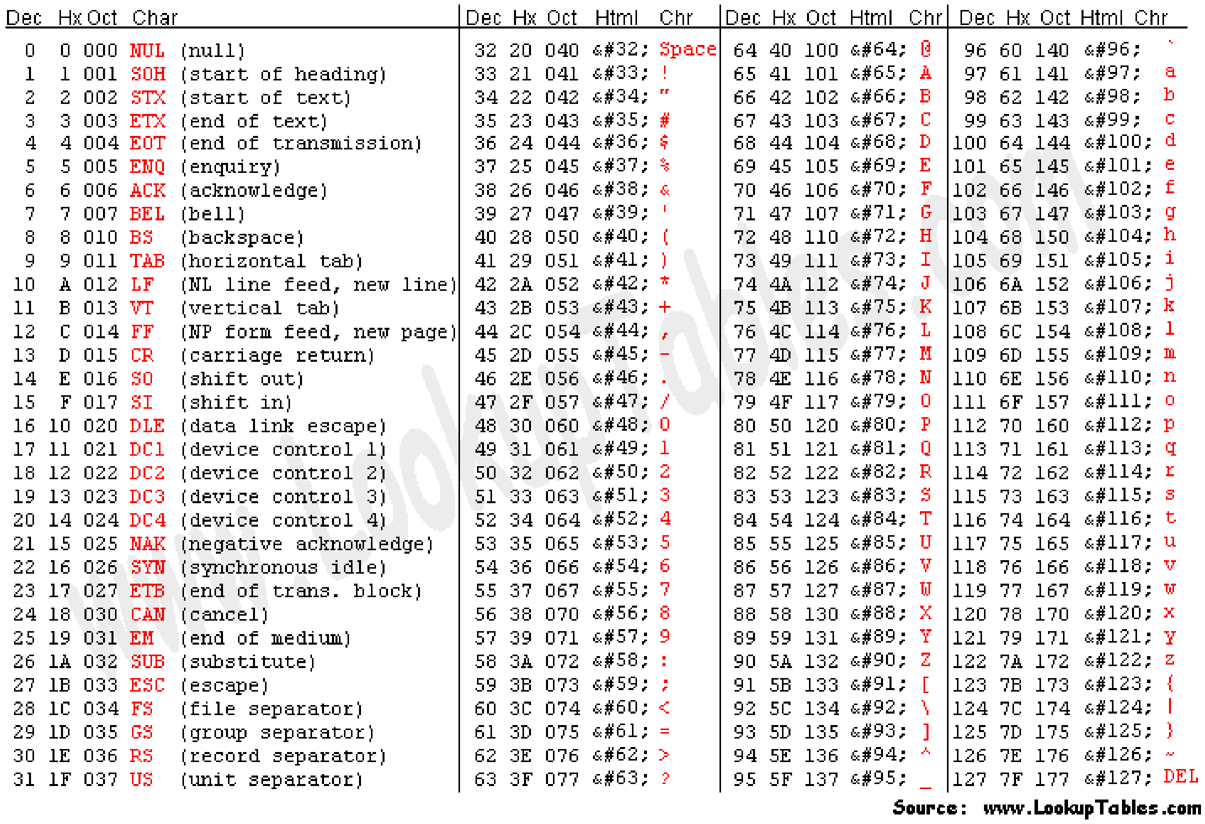
ASCII uses 8 bits (1 byte) to store each character meaning a total of 256 characters can be represented.

> *The word "Computing" has 9 characters, so an ASCII file containing only this word would take up 9 bytes of storage*

256 characters is not enough to store the symbols from all the different languages and fonts we use.

This is a major limitation of the ASCII format.

ASCII Table



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Unicode

Unicode deals with the limitation of ASCII by using **16 bits** (2 bytes) to **store** each character.

This means a total of 65,536 characters can be represented.

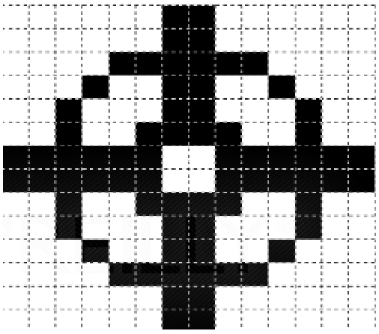
The word "Computing" has 9 characters, so a Unicode file containing only this word would take up 18 bytes of storage

Although Unicode overcomes the limitations of ASCII, its files requires double the storage capacity.

Revision Questions – Characters (Text)

1. State one advantage of using Unicode characters rather than ASCII characters in web pages.

Graphics: Bit-mapped

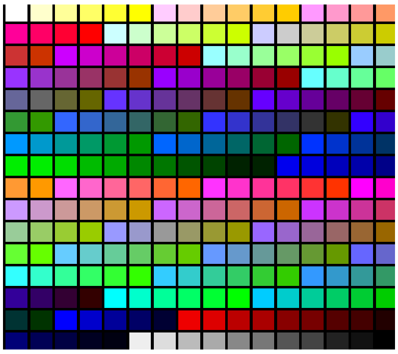


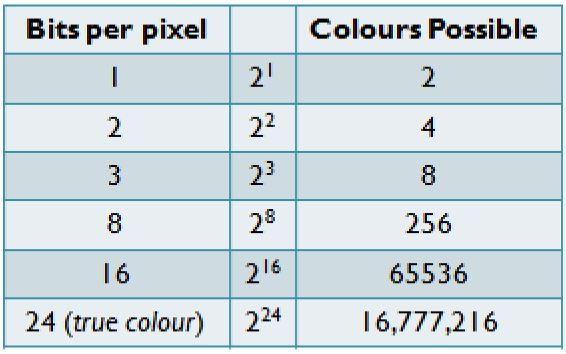
Black and white images are represented by a 2D array of pixels.

Each pixel is represented by a **1 bit** binary number: **1** for black, **0** for white.

Count the number of pixels to determine the **resolution.**

Colour images have to use more than 1 bit per pixel in order to represent more than 2 colours.



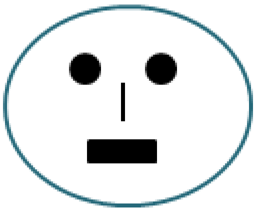


Bits per pixel		Colours Possible
1	2^1	2
2	2^2	4
3	2^3	8
8	2^8	256
16	2^{16}	65536
24 (*true colour*)	2^{24}	16,777,216

The number of bits per pixel is known as the **colour depth** and applies to **all** pixels in an image.

Vector Graphics

Vector graphics are used to create images made up of shapes and lines. They cannot be used to store detailed life-like images.



Vector graphics are stored as a detailed description of the **objects** and **attributes** used in the image.

Some objects and attributes for this shape are given below.

<**circle:** centre x, centre y, radius, line colour, fill colour>

<**rectangle:** x, y, width, height, line colour, fill colour>

<**line** x start, y start, x end, y end, line colour, line width>

Features of Vector Graphics

- Storing objects and attributes in a text file takes up very little space
- The more objects in an image, the larger the file size becomes.
- Images do not lose quality when they are scaled up.
- Shapes that make up an image remain separate and can be moved around individually

Vector Graphics v Bitmap Graphics

Vector Graphics	Bitmap Graphics
Objects can overlap other objects without rubbing out the one below	Objects that overlap **overwrite** the one below.
Increasing the size of objects does not alter the file size (size attribute is simply altered). Adding more objects increases file size.	Adding more details to the image does not affect the file size (same number of pixels used).
Object descriptions are saved as a series of attributes in a text file so **little storage space** required	Entire screen is saved (details of each pixel) so **file size is very large** regardless of the picture.
Object is **resolution independent**. Resolution it is created in has no effect on it at higher resolutions	**Resolution is fixed** when the picture is created. Bitmap will not take advantage of using a higher resolution later.
Individual pixels **cannot** be edited	Bitmap can be zoomed in so individual pixels **can** to be edited
Increasing the size of an object can be done **without** loss of quality. Attributes are simply re-written.	Increasing size of object will result in image becoming **pixelated.**

Revision Questions - Graphics

1. LottoScot has a logo shown below in diagram 1. They want to change the logo to the one in diagram 2.

Diagram 1

Diagram 2

In diagram 2 the rectangle has been moved forward. Explain the advantage of making this change using a vector graphic application compared to a bit-mapped graphic application.

2. A web page uses bit-mapped graphic files for the book covers. State one advantage of using bit-mapped graphic files rather than vector graphic files on this web page.