Questions for discussion:

- How is the information encoded in the computer?
- Give examples from life in which there are two states.
- How many bits are needed to encode traffic light status information?
- Determine how many bytes the word
- "Goding schemes"

Coding schemes

describe standard coding systems for coding character data (ASCII, Unicode).

Expected results (Success criteria)

- know and understand the purpose of the coding system (ASCII, Unicode);
- know and understand the advantages and disadvantages of coding systems;

Coding schemes

Character coding schemes use binary patterns to represent character data (text).

A common code in all computers ensures that information can easily be transferred between machines.

ASCII

символ	10-li код	2-li код	символ	10-li код	2-li код	символ	10-li код	2-li код	символ	10-li код	2-li код
	32	00100000	8	56	00111000	P	80	01010000	h	104	01101000
!	33	00100001	9	57	00111001	Q	81	01010001	i	105	01101001
"	34	00100010	:	58	00111010	R	82	01010010	j	106	01101010
#	35	00100011	;	59	00111011	S	83	01010011	k	107	01101011
$	36	00100100	<	60	00111100	T	84	01010100	l	108	01101100
%	37	00100101	=	61	00111101	U	85	01010101	m	109	01101101
&	38	00100110	>	62	00111110	V	86	01010110	n	110	01101110
'	39	00100111	?	63	00111111	W	87	01010111	o	111	01101111
(	40	00101000	@	64	01000000	X	88	01011000	p	112	01110000
)	41	00101001	A	65	01000001	Y	89	01011001	q	113	01110001
*	42	00101010	B	66	01000010	Z	90	01011010	r	114	01110010
+	43	00101011	C	67	01000011	[	91	01011011	s	115	01110011
,	44	00101100	D	68	01000100	\	92	01011100	t	116	01110100
-	45	00101101	E	69	01000101	]	93	01011101	u	117	01110101
.	46	00101110	F	70	01000110	^	94	01011110	v	118	01110110
/	47	00101111	G	71	01000111	_	95	01011111	w	119	01110111
0	48	00110000	H	72	01001000	`	96	01100000	x	120	01111000
1	49	00110001	I	73	01001001	a	97	01100001	y	121	01111001
2	50	00110010	J	74	01001010	b	98	01100010	z	122	01111010
3	51	00110011	K	75	01001011	c	99	01100011	{	123	01111011
4	52	00110100	L	76	01001100	d	100	01100100	|	124	01111100
5	53	00110101	M	77	01001101	e	101	01100101	}	125	01111101
6	54	00110110	N	78	01001110	f	102	01100110	~	126	01111110
7	55	00110111	O	79	01001111	g	103	01100111	□	127	01111111

символ	10-й код	2-й код	символ	10-й код	2-й код	символ	10-й код	2-й код	символ	10-й код	2-й код
Ђ	128	10000000		160	10100000	А	192	11000000	а	224	11100000
Ѓ	129	10000001	Ў	161	10100001	Б	193	11000001	б	225	11100001
‚	130	10000010	ў	162	10100010	В	194	11000010	в	226	11100010
ѓ	131	10000011	Ј	163	10100011	Г	195	11000011	г	227	11100011
„	132	10000100	¤	164	10100100	Д	196	11000100	д	228	11100100
…	133	10000101	Ґ	165	10100101	Е	197	11000101	е	229	11100101
†	134	10000110	¦	166	10100110	Ж	198	11000110	ж	230	11100110
‡	135	10000111	§	167	10100111	З	199	11000111	з	231	11100111
€	136	10001000	Ё	168	10101000	И	200	11001000	и	232	11101000
‰	137	10001001	©	169	10101001	Й	201	11001001	й	233	11101001
Љ	138	10001010	Є	170	10101010	К	202	11001010	к	234	11101010
‹	139	10001011	«	171	10101011	Л	203	11001011	л	235	11101011
Њ	140	10001100	¬	172	10101100	М	204	11001100	м	236	11101100
Ќ	141	10001101	-	173	10101101	Н	205	11001101	н	237	11101101
Ћ	142	10001110	®	174	10101110	О	206	11001110	о	238	11101110
Џ	143	10001111	Ї	175	10101111	П	207	11001111	п	239	11101111
ђ	144	10010000	°	176	10110000	Р	208	11010000	р	240	11110000
‘	145	10010001	±	177	10110001	С	209	11010001	с	241	11110001
’	146	10010010	І	178	10110010	Т	210	11010010	т	242	11110010
“	147	10010011	і	179	10110011	У	211	11010011	у	243	11110011
”	148	10010100	ґ	180	10110100	Ф	212	11010100	ф	244	11110100
•	149	10010101	µ	181	10110101	Х	213	11010101	х	245	11110101
–	150	10010110	¶	182	10110110	Ц	214	11010110	ц	246	11110110
—	151	10010111	·	183	10110111	Ч	215	11010111	ч	247	11110111
□	152	10011000	ё	184	10111000	Ш	216	11011000	ш	248	11111000
™	153	10011001	№	185	10111001	Щ	217	11011001	щ	249	11111001
љ	154	10011010	є	186	10111010	Ъ	218	11011010	ъ	250	11111010
›	155	10011011	»	187	10111011	Ы	219	11011011	ы	251	11111011
њ	156	10011100	ј	188	10111100	Ь	220	11011100	ь	252	11111100
ќ	157	10011101	Ѕ	189	10111101	Э	221	11011101	э	253	11111101
ћ	158	10011110	ѕ	190	10111110	Ю	222	11011110	ю	254	11111110
џ	159	10011111	ї	191	10111111	Я	223	11011111	я	255	11111111

ASCII Coding schemes

ASCII normally uses 8 bits (1 byte) to store each character.

ASCII values can take many forms:

- Numbers
- Letters (capitals and lower case are separate)
- Punctuation (?/|\£$ etc.)
- non-printing commands (enter, escape, F1)

The symbols are represented by 16 pieces per line. From the top you can see a hexadecimal number from 0 to 16. On the left are similar numbers in hexadecimal form from 0 to FFF.

Unicode

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
004	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
005	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
006	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
007	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
008																
009																
00A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
00B	°	±	^{2}	^{3}	´	µ	¶	·	¸	^{1}	º	»	¼	½	¾	¿

By connecting the number on the left with the number on top, you can find out the symbol code.
For example: the English letter F is located on line 004, in the column 6: 004 + 6 = symbol code 0046.

http://foxtools.ru/Unicode

There are several versions of unicode, each with using a different number of bits to store data:

Name	Descriptions
UTF-8	8-bit is the most common unicode format. Characters can take as little as 8-bits, maximizing compatibility with ASCII. But it also allows for variable-width encoding expanding to 16, 24, 32, 40 or 48 bits when dealing with larger sets of characters
UTF-16	16-bit, variable-width encoding, can expand to 32 bits.
UTF-32	32-bit, fixed-width encoding. Each character takes exactly 32-bits

ASCII/8859-1 Text

A	0100 0001
S	0101 0011
C	0100 0011
I	0100 1001
I	0100 1001
/	0010 1111
8	0011 1000
8	0011 1000
5	0011 0101
9	0011 1001
-	0010 1101
1	0011 0001
	0010 0000
t	0111 0100
e	0110 0101
x	0111 1000
t	0111 0100

Unicode Text

A	0000 0000 0100 0001
S	0000 0000 0101 0011
C	0000 0000 0100 0011
I	0000 0000 0100 1001
I	0000 0000 0100 1001
	0000 0000 0010 0000
天	0101 1001 0010 1001
地	0101 0111 0011 0000
	0000 0000 0010 0000
س	0000 0110 0011 0011
ل	0000 0110 0100 0100
ا	0000 0110 0010 0111
م	0000 0110 0100 0101
	0000 0000 0010 0000
α	0000 0011 1011 0001
≰	0010 0010 0111 0000
γ	0000 0011 1011 0011

- *What do you think is the encoding system used in our computers? Why? Explain your answer*

- *Advantages and disadvantages of coding systems*

Fixed point numbers

understand how binary can be used to represent negative and fractional numbers using floating and fixed point.

Expected results (Success criteria)

- are able to convert negative numbers from decimal to binary system and back;
- are able to convert fractional numbers with a fixed point from decimal to binary system and back;

Questions for discussion:

- How are whole decimal numbers converted to a binary number system?
- How are binary numbers translated into the decimal system?
- How do you think, how can you translate negative numbers?

Method: Converting a Negative Denary Number into Binary Twos Complement

Let's say you want to convert **-35** into Binary Twos Complement. First, find the binary equivalent of 35 (the positive version)

32	16	8	4	2	1
1	0	0	0	1	1

Now add an extra bit **before** the MSB, make it a zero, which gives you:

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

Now 'flip' all the bits: if it's a 0, make it a 1; if it's a 1, make it a 0:

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

This new bit represents -64 (minus 64). Now **add 1**:

```
64 32  16   8   4   2   1
 1  0   1   1   1   0   0
                      + 1

 1  0   1   1   1   0   1
```

If we perform a quick binary -> denary conversion, we have: -64 + 16 + 8 + 4 + 1 = -64 + 29 = **-35**

Signed binary numbers

0000 0101 (positive)
1111 1011 (negative)

Method 1: converting twos complement to denary

To find the value of the negative number we must find and keep the right most 1 and all bits to its right, and then flip everything to its left. Here is an example:

```
1111 1011 note the number is negative
```

```
1111 1011 find the right most one

1111 1011
0000 0101 flip all the bits to its left
```

We can now work out the value of this new number which is:

```
128  64  32  16   8   4   2   1
  0   0   0   0   0   1   0   1
                      4   +   1 = -5   (remember the sign you worked out earlier!)
```

Method 2: converting twos complement to denary

To find the value of the negative number we must take the MSB and apply a negative value to it. Then we can add all the heading values together

```
1111 1011 note the number is negative
-128  64  32  16   8   4   2   1
   1   1   1   1   1   0   1   1
-128 +64 +32 +16  +8      +2  +1 = -5
```

Fixed point numbers

Example: converting decimal to binary decimal using fixed point notation

We are going to convert the number 6.125 into a binary fraction by using the grid below

8	4	2	1		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
0	1	1	0	.	0	0	1	0

This seems simple enough as 6.125 = 4 + 2 + 0.125, but what about this more interesting number: 6.4

8	4	2	1		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$
0	1	1	0	.	0	1	1	0

But this doesn't look right?! This number isn't correct as it only reaches 4 + 2 + 0.25 + 0.125 = 6.375, we need more bits for the binary fraction places. However, a computer might restrict you to the number of bits you can use, so we'll use the number closest to the one we were aiming for. You could feel a bit annoyed at this, but don't worry, you make this compromise every time you try to represent $\frac{1}{3}$ with the decimal factions, 0.33333333.

Tasks

Convert a Negative Denary Number into Binary Twos Complement -12

Convert the following two's complement number into denary 0001 1011

Converting from denary to binary fractions 7.5

Convert these binary fractions into denary: 0111.0100

Additional tasks

Converting from denary to binary fractions -34.5

Converting from denary to binary fractions 4.5625

Convert the following two's complement number into denary 1111 1111

Convert these binary fractions into denary: 1011.1001

0000 1100 = +12 -> 1111 0100 = -12

(positive number) 27

0111.1000

7.25

0010 0010 = +34 -> 1101 1110 = -34

0100.1001

(negative number) 0000 0001 = -1

11.5625