Practice

What is the decimal equivalent of the bit pattern shown below if it represents an unsigned fixed-point binary value with eight bits before the binary point and four bits after the binary point?

0110 1000 1010

Practice

What is the decimal equivalent of the bit pattern shown below if it represents an unsigned fixed-point binary value with eight bits before the binary point and four bits after the binary point?

0110 1000 1010

$$104\frac{5}{8}$$

Practice

Convert the decimal number below into an unsigned fixed-point binary value with 2 bytes, assuming 4 bits after the binary point.

150.375

Practice

Convert the decimal number below into an unsigned fixed-point binary value with 2 bytes, assuming 4 bits after the binary point.

150.375

0000 1001 0110 0110

Practice

What is the decimal equivalent of the bit pattern below if it represents a two’s complement binary value assuming 4 bits after the point?

1010 0101 0110

Practice

What is the decimal equivalent of the bit pattern below if it represents a two's complement binary value assuming 4 bits after the point?

1010 0101 0110

–90.625

Practice

Convert the decimal number below into a two’s complement binary value using 1 byte assuming 4 bits after the point.

−2.75

Practice

Convert the decimal number below into a two’s complement binary value using 1 byte assuming 4 bits after the point.

−2.75

1101 0100

Practice

Convert the decimal number below into standard form.

1230

Practice

Convert the decimal number below into standard form.

1230

1.23×10^3

Practice

Convert the decimal number below from standard form into ordinary numbers

$$6.25 \times 10^{-2}$$

Practice

Convert the decimal number below from standard form into ordinary numbers

$$6.25 \times 10^{-2}$$

0.0625

Floating Point Binary Numbers

Specification

4.5.4.4 Numbers with a fractional part

Content	Additional information
Know how numbers with a fractional part can be represented in: • fixed point form in binary in a given number of bits • floating point form in binary in a given number of bits.	Students are not required to know the Institute of Electrical and Electronic Engineers (IEEE) standard, only to know, understand and be able to use a simplified floating representation consisting of mantissa + exponent.
Be able to convert for each representation from: • decimal to binary of a given number of bits • binary to decimal of a given number of bits.	Exam questions on floating point numbers will use a format in which both the mantissa and exponent are represented using two's complement.

4.5.4.8 Normalisation of floating point form

Content	Additional information
Know why floating point numbers are normalised and be able to normalise un-normalised floating point numbers with positive or negative mantissas.	

Standard form

Standard/scientific form

$$-819\ 000\ 000 = -8.19 \times 10^{8}$$

- The **mantissa** (–8.19) represents the significant digits of the number. It also holds the sign
- The **exponent** (8) defines where the decimal point needs to be if the number shown is in its ordinary decimal form

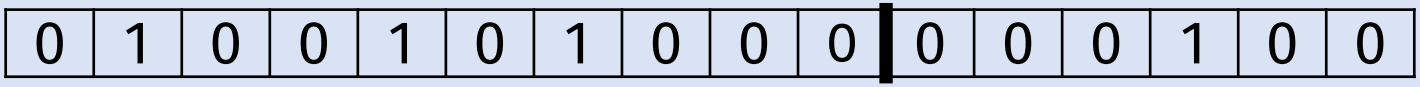
Real numbers in binary

- Binary **floating point** uses the same idea as standard form
- Both the **mantissa** and **exponent** are in two's complement format
- If the most significant (first) bit in the **mantissa** is a 1, then the number is negative
- What can we say if the most significant bit in the **exponent** is a 1?

Real numbers in binary

- All our examples will use 16 bits
- These 16 bit numbers will use **10 bits for the mantissa** and **6 bits for the exponent**
- In practice, real numbers are stored using a minimum of 32 bits – typically 64

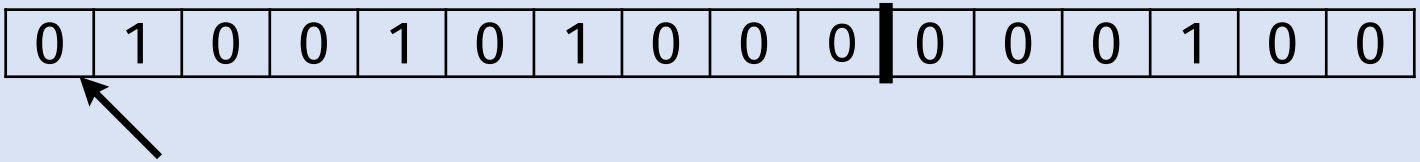
Converting floating point binary to decimal



$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

$$V = M \times 2^{E}$$

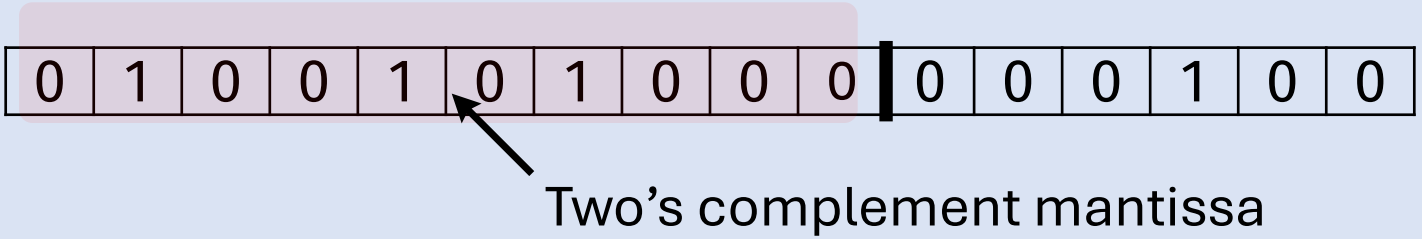
Converting floating point binary to decimal



$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

$$V = M \times 2^{E}$$

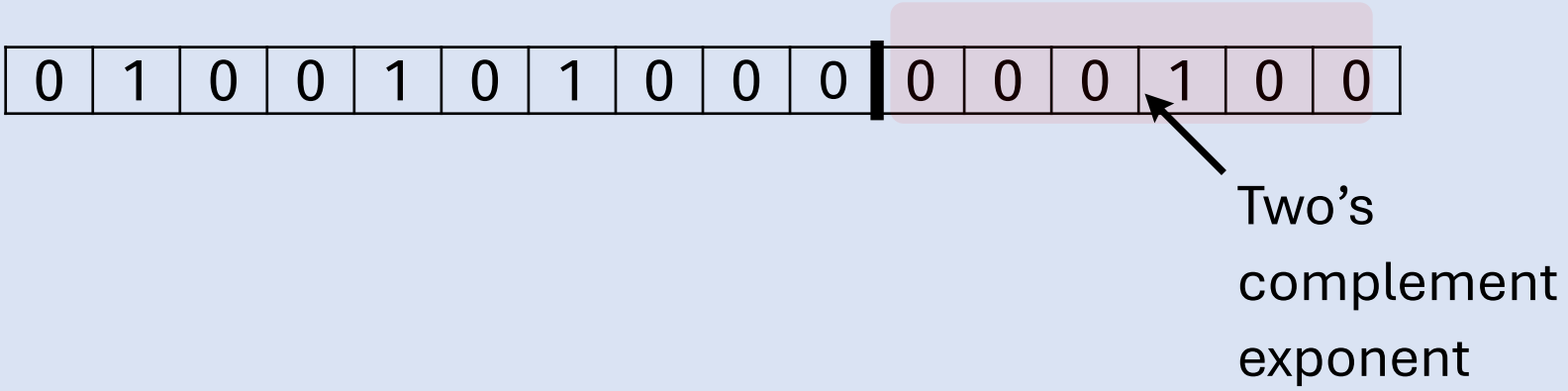
Converting floating point binary to decimal



$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

$$V = M \times 2^{E}$$

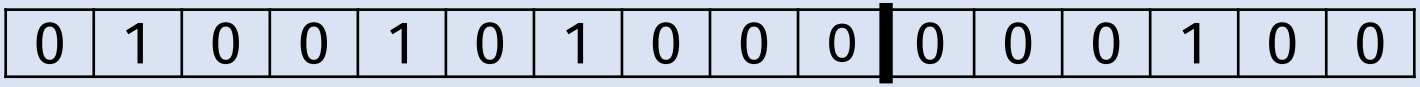
Converting floating point binary to decimal



$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

$$V = M \times 2^{E}$$

Method & Example 1



1. Convert the mantissa to decimal
2. Convert the exponent to decimal
3. Apply the formula

$$V = M \times 2^{E}$$

Example 2

Convert 1011100000000101 from floating point binary, with a 10 bit mantissa and 6 bit exponent, to decimal.

Practice

What is the decimal representation of the bit pattern below if it represents a **floating point** binary number with a 8-bit mantissa and a 4-bit exponent?

0110 1000 0110

Practice

What is the decimal representation of the bit pattern below if it represents a **floating point** binary number with a 8-bit mantissa and a 4-bit exponent?

0110 1000 0110

52

Practice

What is the decimal representation of the bit pattern below if it represents a **floating point** binary number with a 8-bit mantissa and a 4-bit exponent?

0111 0100 1010

Practice

What is the decimal representation of the bit pattern below if it represents a **floating point** binary number with a 8-bit mantissa and a 4-bit exponent?

0111 0100 1010

$$\frac{29}{2048}$$

Practice

What is the decimal representation of the bit pattern below if it represents a **floating point** binary number with a 8-bit mantissa and a 4-bit exponent?

1011 0100 0110

Practice

What is the decimal representation of the bit pattern below if it represents a **floating point** binary number with a 8-bit mantissa and a 4-bit exponent?

1011 0100 0110

−38

Practice

What is the decimal representation of the bit pattern below if it represents a **floating point** binary number with a 8-bit mantissa and a 4-bit exponent?

1011 0000 1100

Practice

What is the decimal representation of the bit pattern below if it represents a **floating point** binary number with a 8-bit mantissa and a 4-bit exponent?

1011 0000 1100

$$-\frac{5}{128}$$

Arrays

Arrays

Discuss

1. What does the word ‘array’ mean in English?
2. What is an array in programming?

Arrays

This is an array.

A fixed-size sequential collection of elements of the same type.

Indices

0	1	2	3	4	5	6	7
12	34	67	23	68	95	36	3

- Each element in an array is accessed with an **index**, which is a number that refers to a position in the array
- Indices start at 0

Example 1

The following algorithm uses an array called `Values` containing four numbers.

```
Result ← 0
Index ← 0
Repeat
  If Result < Values[Index]
    Result ← Values[Index]
  EndIf
  Index ← Index + 1
Until Index = 4
Print Result
```

Values

0	24
0	13
0	57
0	45

Trace table

Result	Index

Example 1

The following algorithm uses an array called `Values` containing four numbers.

```
Result ← 0
Index ← 0
Repeat
  If Result < Values[Index]
    Result ← Values[Index]
  EndIf
  Index ← Index + 1
Until Index = 4
Print Result
```

Values

0	24
0	13
0	57
0	45

Trace table

Result	Index
0	0
24	1
	2
57	3
	4

Example 2

- The operators `DIV` and `MOD` perform integer arithmetic
  - `x DIV y` calculates the maximum number of times `y` fits into `x` (`7 DIV 3 = 2`)
  - `x MOD Y` calculates the remainder from the division (`7 MOD 3 = 1`)

```
X = 5
Index = 0
Repeat
  Y = X MOD 2
  X = X DIV 2
  Result[Index] = Y
  Index = Index + 1
Until X = 0
```

Y	X	Index	Result[2]	Result[1]	Result[0]

Example 2

- The operators `DIV` and `MOD` perform integer arithmetic
  - `x DIV y` calculates the maximum number of times `y` fits into `x` (`7 DIV 3 = 2`)
  - `x MOD Y` calculates the remainder from the division (`7 MOD 3 = 1`)

```
X = 5
Index = 0
Repeat
  Y = X MOD 2
  X = X DIV 2
  Result[Index] = Y
  Index = Index + 1
Until X = 0
```

Y	X	Index	Result[2]	Result[1]	Result[0]
	5	0			
1	2	1			1
0	1	2		0	
1	0	3	1		


WHO
DUNNIT?


TWO
DUNNIT?


OOPS I
DUNNIT
AGAIN

Character Coding

- How many different characters can we type using a keyboard?
- How many bits do we need to represent that number of options?

ASCII

- American Standard Code of Information Interchange
- 7 bits
- Examples
  - A is coded 65 (1000001)
  - B is coded 66 (1000010)
  - 0 is coded 48 (0110000)
  - 1 is coded 49 (0110001)

ASCII

- American Standard Code of Information Interchange
- 7 bits
- Examples
  - A is coded 65 (1000001)
  - B is coded 66 (1000010)
  - 0 is coded 48 (0110000)
  - 1 is coded 49 (0110001)
  - a is coded 97 (1100001)
  - b is coded 98 (1100010)

ASCII

	Ch	ASCII		Ch	ASCII		Ch	ASCII		Ch	ASCII
0	NULL	0000000	32	sp	0100000	64	@	1000000	96	'	1100000
1	SOH	0000001	33	!	0100001	65	A	1000001	97	a	1100001
2	STX	0000010	34	"	0100010	66	B	1000010	98	b	1100010
3	ETX	0000011	35	£	0100011	67	C	1000011	99	c	1100011
4	EOT	0000100	36	$	0100100	68	D	1000100	100	d	1100100
5	ENQ	0000101	37	%	0100101	69	E	1000101	101	e	1100101
6	ACK	0000110	38	&	0100110	70	F	1000110	102	f	1100110
7	BEL	0000111	39	`	0100111	71	G	1000111	103	g	1100111
8	BS	0001000	40	(	0101000	72	H	1001000	104	h	1101000
9	HT	0001001	41	)	0101001	73	I	1001001	105	i	1101001
10	LF	0001010	42	*	0101010	74	J	1001010	106	j	1101010
11	VT	0001011	43	+	0101011	75	K	1001011	107	k	1101011
12	SF	0001100	44	,	0101100	76	L	1001100	108	l	1101100
13	CR	0001101	45	-	0101101	77	M	1001101	109	m	1101101
14	SO	0001110	46	.	0101110	78	N	1001110	110	n	1101110
15	SI	0001111	47	/	0101111	79	O	1001111	111	o	1101111
16	DLE	0010000	48	0	0110000	80	P	1010000	112	p	1110000
17	DC1	0010001	49	1	0110001	81	Q	1010001	113	q	1110001
18	DC2	0010010	50	2	0110010	82	R	1010010	114	r	1110010
19	DC3	0010011	51	3	0110011	83	S	1010011	115	s	1110011
20	DC4	0010100	52	4	0110100	84	T	1010100	116	t	1110100
21	NAK	0010101	53	5	0110101	85	U	1010101	117	u	1110101
22	SYN	0010110	54	6	0110110	86	V	1010110	118	v	1110110
23	ETB	0010111	55	7	0110111	87	W	1010111	119	w	1110111
24	CAN	0011000	56	8	0111000	88	X	1011000	120	x	1111000
25	EM	0011001	57	9	0111001	89	Y	1011001	121	y	1111001
26	SUB	0011010	58	:	0111010	90	Z	1011010	122	z	1111010
27	ESC	0011011	59	;	0111011	91	[	1011011	123	{	1111011
28	FS	0011100	60	<	0111100	92	\	1011100	124	|	1111100
29	GS	0011101	61	=	0111101	93	]	1011101	125	}	1111101
30	RS	0011110	62	>	0111110	94	^	1011110	126	~	1111110
31	US	0011111	63	?	0111111	95	_	1011111	127	del	1111111

Extended ASCII

- 8 bits
- Includes characters such as ©, ®, *é*

Unicode

- Unicode provides a unique number for every character
  - No matter the platform
  - No matter the program
  - No matter the language
- Originally intended to be 16 bits
- Now is generally 1-4 bytes (UTF-8)
- Allows for 1,112,064 characters

ASCII in C#

- Casting between `int` and `char` lets us convert between characters and their respective ASCII values

```
int asciiCode;
string userChar;
char character;

Console.Write("Enter code: ");
asciiCode = int.Parse(Console.ReadLine());

Console.WriteLine((char)asciiCode);

Console.Write("Enter character: ");
userChar = Console.ReadLine();
```

```
character = userChar[0];

Console.WriteLine((int)character);
Console.ReadKey();
```

ASCII in C#

- What if we wanted to make `a` convert to `0` and `b` convert to `1` etc.?

ASCII in C#

- What if we wanted to make `a` convert to `0` and `b` convert to `1` etc.?

```
string word = "bark";
int ascii = word[0] - 'a';
```

Programming

- Finish the built-in functions worksheet from last lesson
  - Focus on the ASCII questions
- Then work on your adventure game