1.1 Number systems

Signed integers using two's complement

How can we represent a negative number?

From the previous video, we know the smallest number you can represent in binary is 0.

If we are using 8 bits:

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

How can we represent a negative number?

From the previous video, we know the smallest number you can represent in binary is 0.

If we are using 8 bits:

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

We also know that if we add a 1 in any of the columns, the number becomes positive and increases in value.

How can we represent a negative number?

From the previous video, we know the smallest number you can represent in binary is 0.

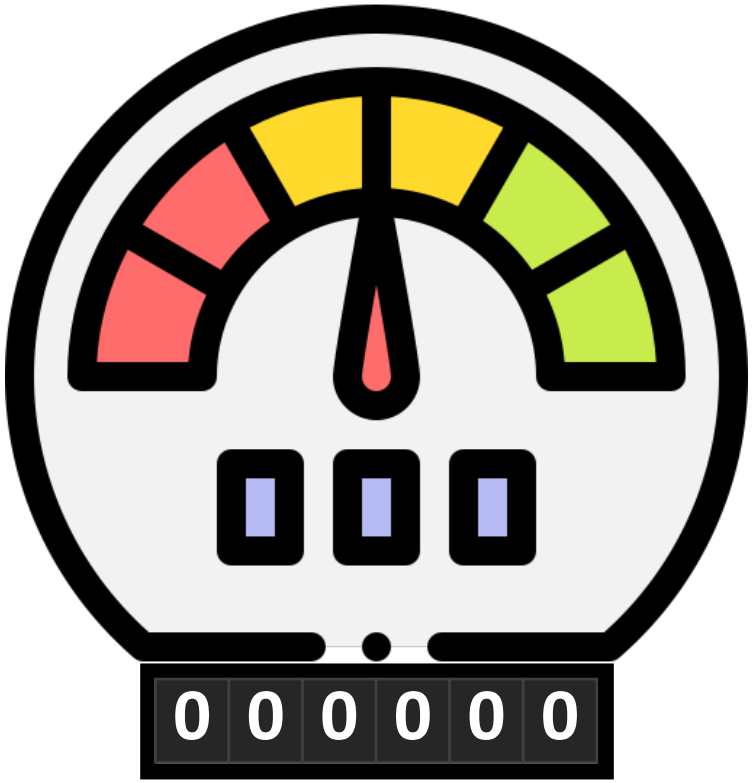
If we are using 8 bits:

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

We also know that if we add a 1 in any of the columns, the number becomes positive and increases in value.

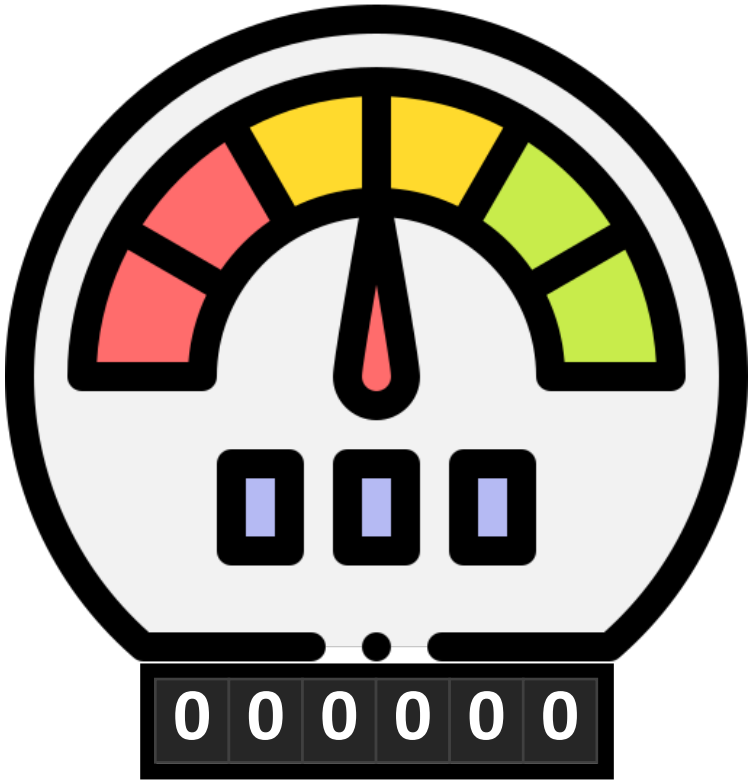
So, if adding 1s increases the number's value, how can we represent negative numbers like -10?

Think about a car's milometer



A car leaves the factory with all digits set to 0.

Think about a car's milometer

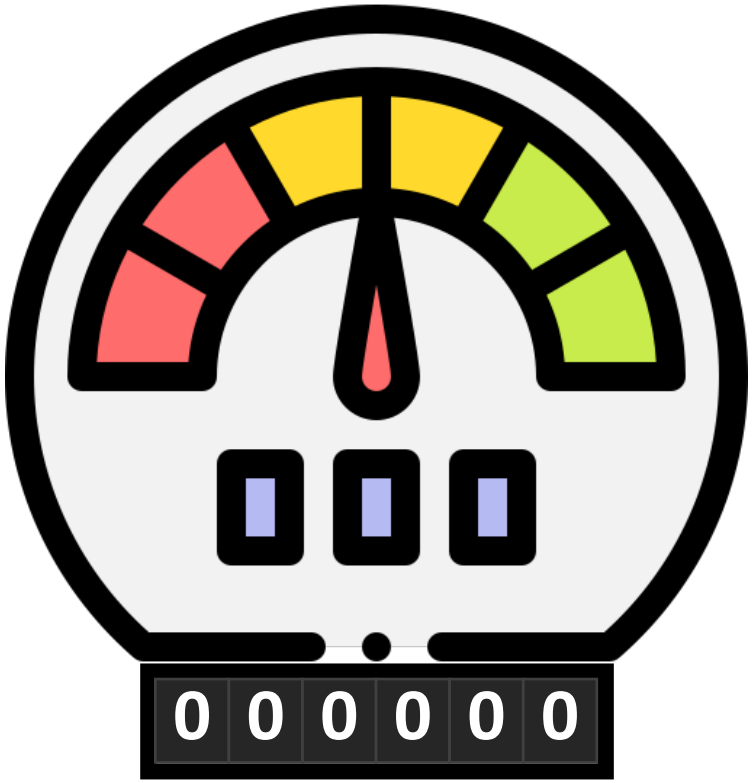


A car leaves the factory with all digits set to 0.

If we drive the car for one mile, we end up with 000001 on the meter.

We can think of this as +1 mile.

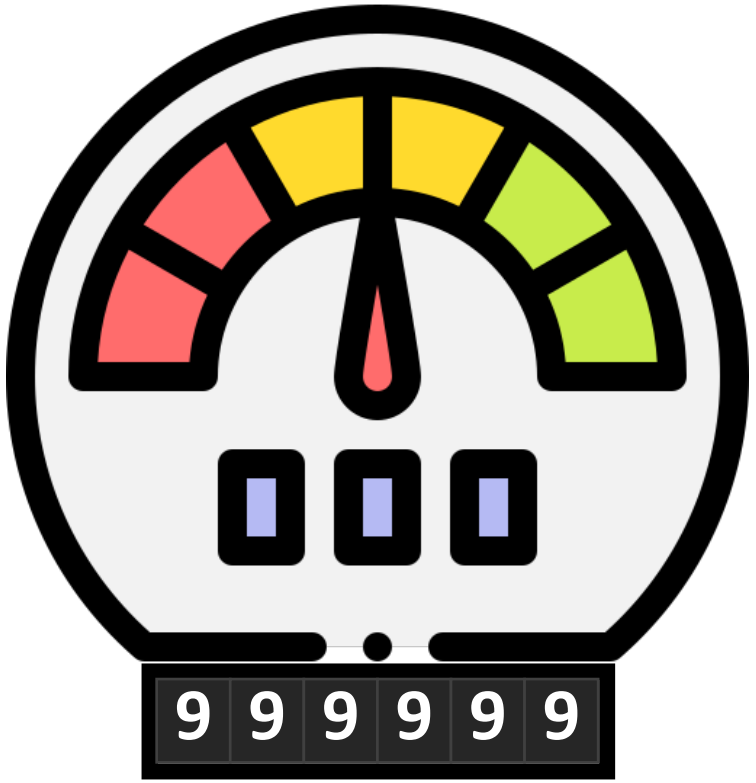
Think about a car's milometer



A car leaves the factory with all digits set to 0.

If we drive the car for one mile, we end up with 000001 on the meter.

We can think of this as +1 mile.



Imagine if we could turn the meter back one mile to 999999.

We can think of this as -1 mile.

Computers can use a similar concept.

Two's complement

1	1	1	1	1	1	0	1	=	**-3**
1	1	1	1	1	1	1	0	=	**-2**
1	1	1	1	1	1	1	1	=	**-1**
0	0	0	0	0	0	0	0	=	**0**
0	0	0	0	0	0	0	1	=	**1**
0	0	0	0	0	0	1	0	=	**2**
0	0	0	0	0	0	1	1	=	**3**

Two's complement

1	1	1	1	1	1	0	1	=	**-3**
1	1	1	1	1	1	1	0	=	**-2**
1	1	1	1	1	1	1	1	=	**-1**
0	0	0	0	0	0	0	0	=	**0**
0	0	0	0	0	0	0	1	=	**1**
0	0	0	0	0	0	1	0	=	**2**
0	0	0	0	0	0	1	1	=	**3**

When using the two's complement method of representing binary numbers, you will notice that:

- All positive numbers **always** start with a **zero**.

Two's complement

1	1	1	1	1	1	0	1	=	-3
1	1	1	1	1	1	1	0	=	-2
1	1	1	1	1	1	1	1	=	-1
0	0	0	0	0	0	0	0	=	0
0	0	0	0	0	0	0	1	=	1
0	0	0	0	0	0	1	0	=	2
0	0	0	0	0	0	1	1	=	3

When using the two's complement method of representing binary numbers, you will notice that:

- All positive numbers **always** start with a **zero**.
- All negative numbers **always** start with a **one**.

Two's complement

Notice that the left-hand bit – known as **the most significant bit (MSB)** – is now representing a negative value.

-128	64	32	16	8	4	2	1

Two's complement

Let's start by representing **positive** 117 (+117):

-128	64	32	16	8	4	2	1

Two's complement

Let's start by representing **positive** 117 (+117):

-128	64	32	16	8	4	2	1
0							

Two's complement

Let's start by representing **positive** 117 (+117):

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

= +117

Two's complement

Now let's try representing **negative** 117 (-117):

-128	64	32	16	8	4	2	1	
0	1	1	1	0	1	0	1	**= +117**

Two's complement

Now let's try representing **negative** 117 (-117):

-128	64	32	16	8	4	2	1	
0	1	1	1	0	1	0	1	**= +117**
1								

Two's complement

Now let's try representing **negative** 117 (-117):

-128	64	32	16	8	4	2	1	
0	1	1	1	0	1	0	1	**= +117**
1	0	0	0	1	0	1	1	
-128				+ 8				

= -120

Two's complement

Now let's try representing **negative** 117 (-117):

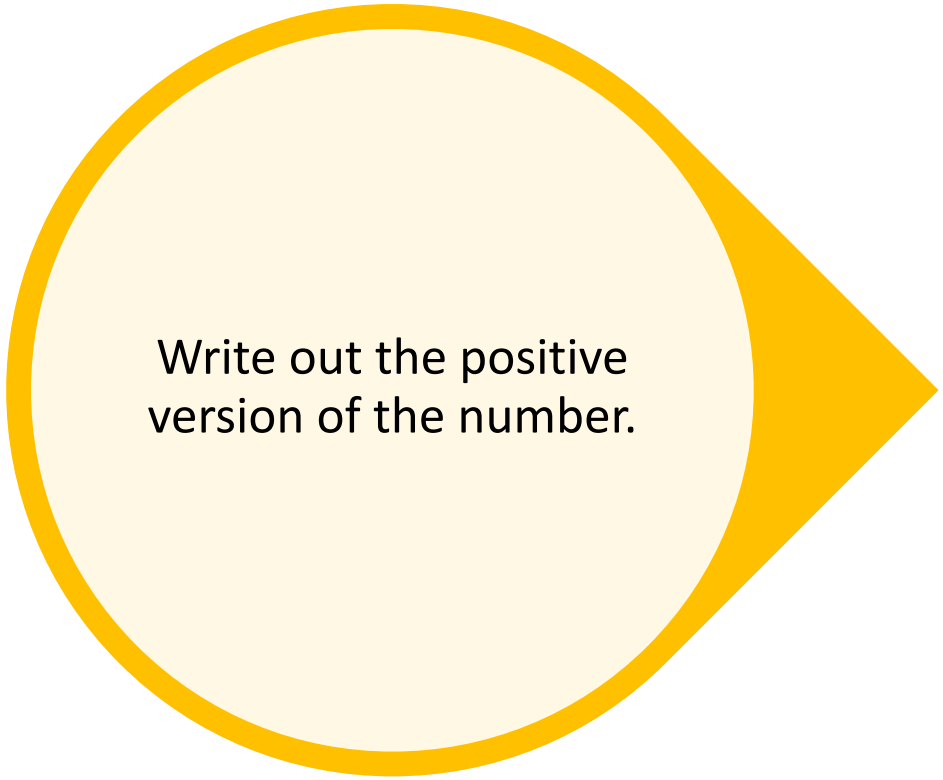
-128	64	32	16	8	4	2	1	
0	1	1	1	0	1	0	1	**= +117**
1	0	0	0	1	0	1	1	
-128				+ 8		+ 2		
						= -118		

Two's complement

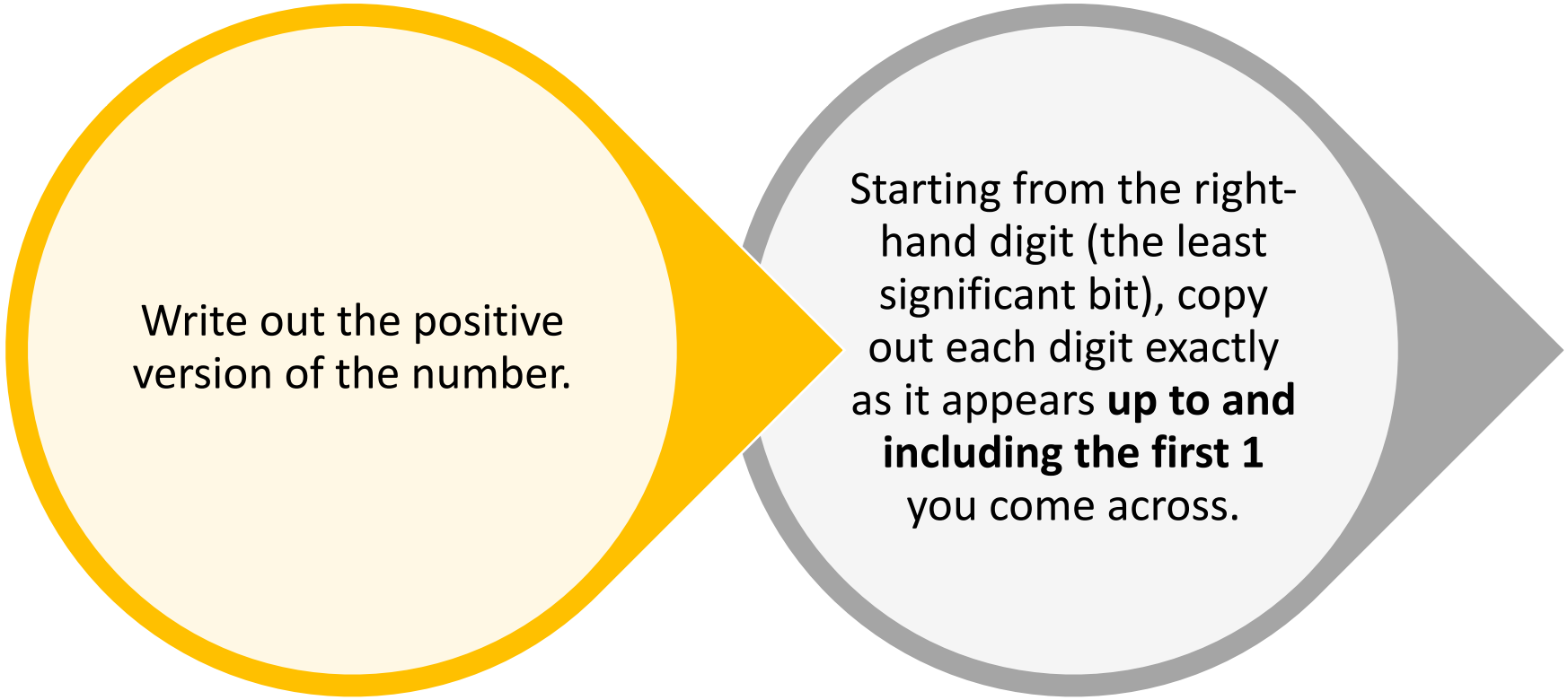
Now let's try representing **negative** 117 (-117):

-128	64	32	16	8	4	2	1	
0	1	1	1	0	1	0	1	**= +117**
1	0	0	0	1	0	1	1	
-128				+ 8		+ 2	+ 1	**= -117**

How to easily turn a positive number into its two's complement negative version



How to easily turn a positive number into its two's complement negative version



How to easily turn a positive number into its two's complement negative version

How to easily turn a positive number into its two's complement negative version

Let's convert the number +12 into -12.

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

How to easily turn a positive number into its two's complement negative version

Let's convert the number +12 into -12.

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

1. Write out the positive version of the number.
2. Starting from the right-hand digit (the least significant bit), copy out each digit exactly as it appears **up to and including the first 1** you come across.
3. From the next digit onwards, **swap every 1 for a 0** and **swap every 0 for a 1**.

How to easily turn a positive number into its two's complement negative version

Let's convert the number +12 into -12.

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

1. Write out the positive version of the number.
2. Starting from the right-hand digit (the least significant bit), copy out each digit exactly as it appears **up to and including the first 1** you come across.
3. From the next digit onwards, **swap every 1 for a 0** and **swap every 0 for a 1**.

How to easily turn a positive number into its two's complement negative version

Let's convert the number +12 into -12.

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

1. Write out the positive version of the number.
2. Starting from the right-hand digit (the least significant bit), copy out each digit exactly as it appears **up to and including the first 1** you come across.
3. From the next digit onwards, **swap every 1 for a 0** and **swap every 0 for a 1**.

How to easily turn a positive number into its two's complement negative version

Let's convert the number +12 into -12.

-128	64	32	16	8	4	2	1	
0	0	0	0	1	1	0	0	**= +12**
1	1	1	1	0	1	0	0	**= -12**

1. Write out the positive version of the number.
2. Starting from the right-hand digit (the least significant bit), copy out each digit exactly as it appears **up to and including the first 1** you come across.
3. From the next digit onwards, **swap every 1 for a 0** and **swap every 0 for a 1**.

1.1 Number systems | Logical binary shifts