Today

- Binary addition
- Representing negative numbers

Binary Addition

Consider the following binary numbers:

0 0 1 0 0 1 1 0

0 0 1 0 1 0 1 1

How do we add these numbers?

Binary Addition

0 0 1 0 0 1 1 0

0 0 1 0 1 0 1 1

1

Binary Addition

0 0 1 0 0 1 1 0

0 0 1 0 1 0 1 1

0 1

And we have a carry now!

Binary Addition

0 0 1 0 0 1 1 0

0 0 1 0 1 0 1 1

0 0 1

And we have a carry again!

Binary Addition

0 0 1 0 0 1 1 0

0 0 1 0 1 0 1 1

0 0 0 1

and again!

Binary Addition

0 0 1 0 0 1 1 0

0 0 1 0 1 0 1 1

1 0 0 0 1

Binary Addition

0 0 1 0 0 1 1 0

0 0 1 0 1 0 1 1

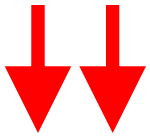
0 1 0 0 0 1

One more carry!

Binary Addition

0 0 1 0 0 1 1 0

0 0 1 0 1 0 1 1



0 1 0 1 0 0 0 1

Binary Addition

Behaves just like addition in decimal, but:

- We carry to the next digit any time the sum of the digits is 2 (decimal) or greater

Negative Numbers

So far we have only talked about representing non-negative integers

- What can we add to our binary representation that will allow this?

Representing Negative Numbers

One possibility:

- Add an extra bit that indicates the sign of the number
- We call this the “sign-magnitude” representation

Sign Magnitude Representation

+12 0 0 0 0 1 1 0 0

Sign Magnitude Representation

+12 0 0 0 0 1 1 0 0

-12 1 0 0 0 1 1 0 0

Sign Magnitude Representation

+12 0 0 0 0 1 1 0 0

-12 1 0 0 0 1 1 0 0

What is the problem with this approach?

Sign Magnitude Representation

What is the problem with this approach?

- Some of the arithmetic operators that we have already developed do not do the right thing

Sign Magnitude Representation

Operator problems:

- For example, we have already designed a counter (that implements an ‘increment’ operation)

-12 1 0 0 0 1 1 0 0

Sign Magnitude Representation

Operator problems:

-12 1 0 0 0 1 1 0 0

Increment

Sign Magnitude Representation

Operator problems:

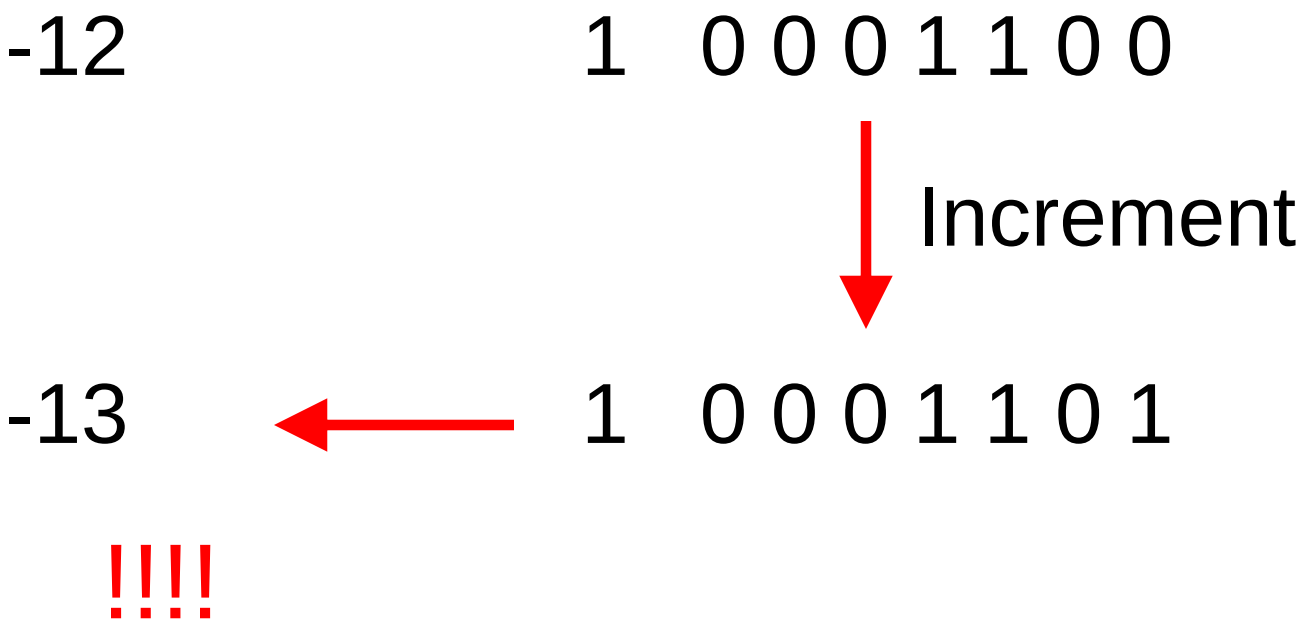
-12 1 0 0 0 1 1 0 0

Increment

1 0 0 0 1 1 0 1

Sign Magnitude Representation

Operator problems:



Representing Negative Numbers

An alternative:

- When taking the additive inverse of a number, invert all of the individual bits
- The leftmost bit still determines the sign of the number

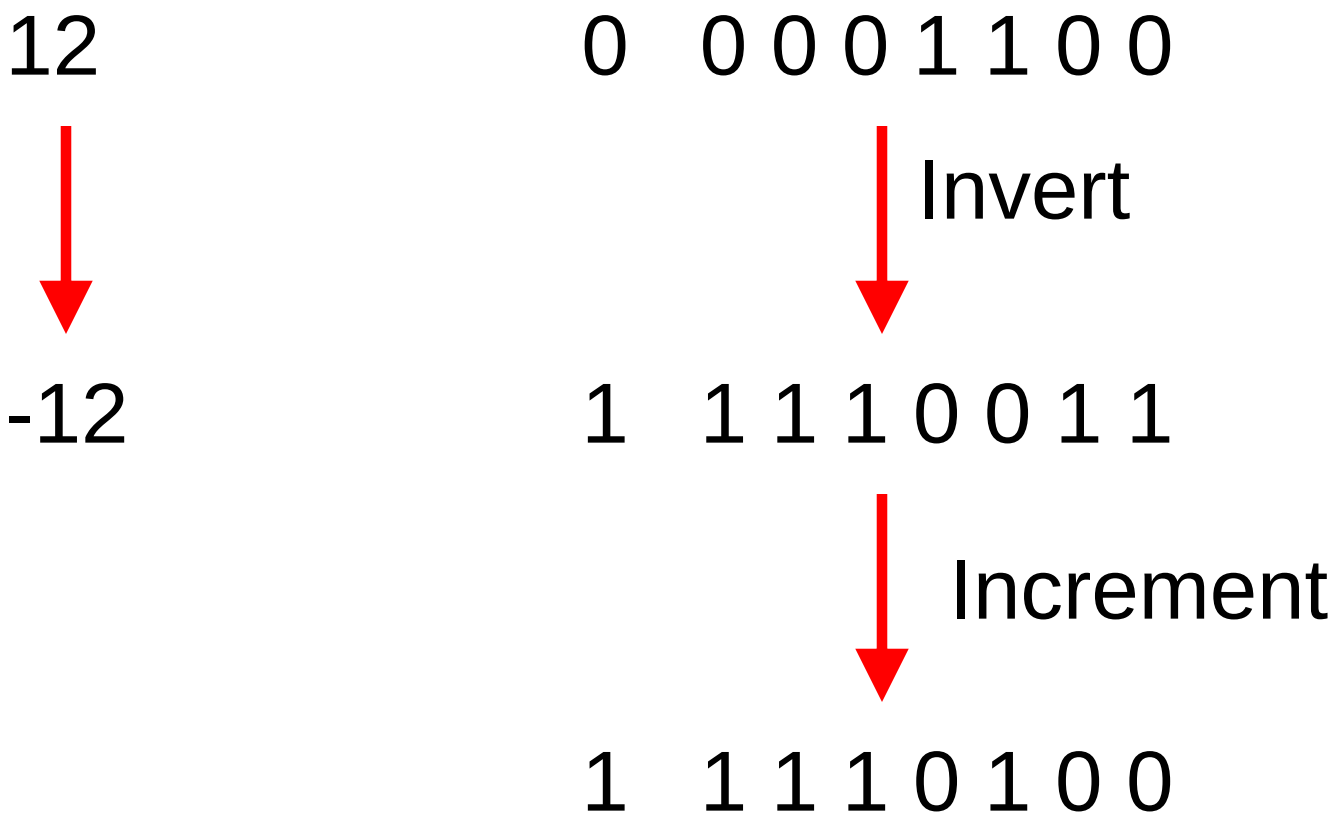
One’s Complement Representation

12 0 0 0 0 1 1 0 0

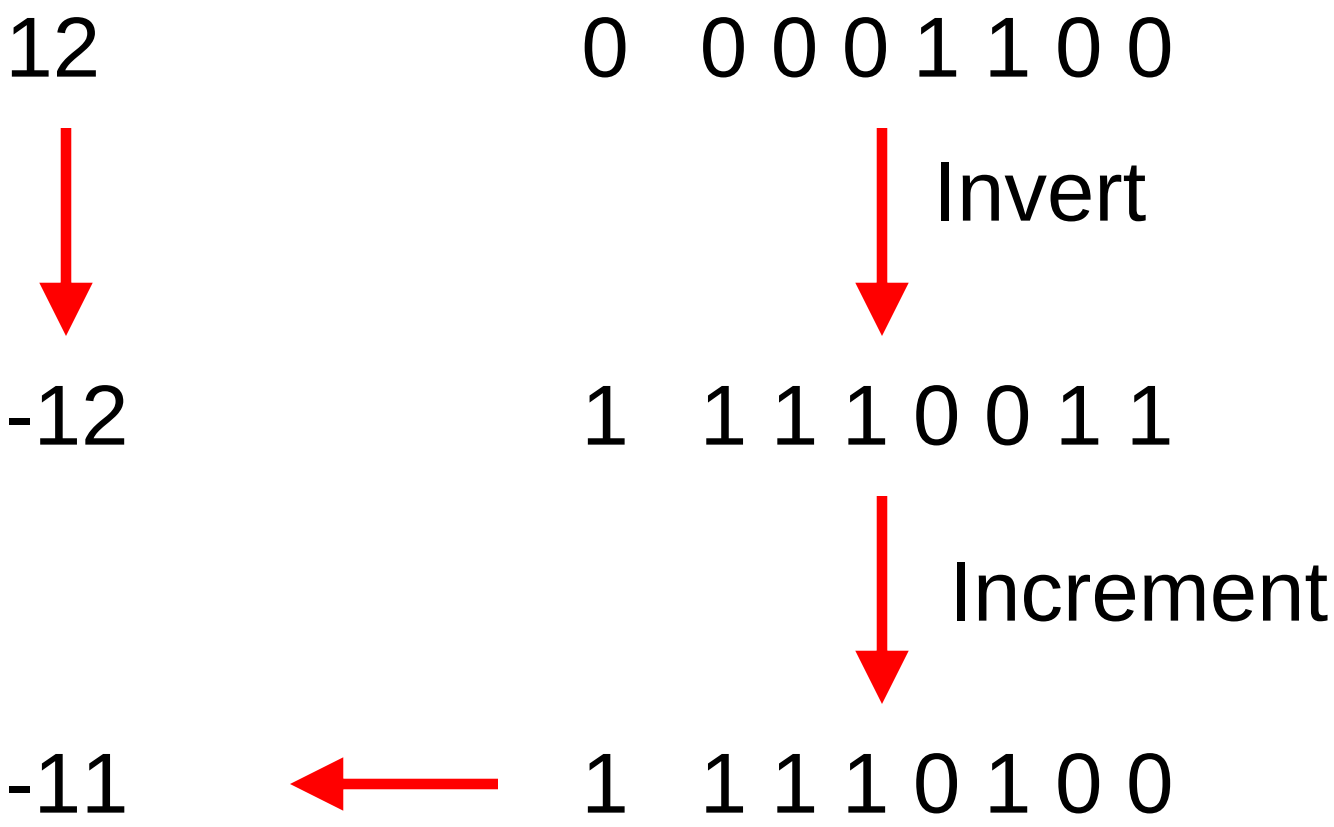
Invert

-12 1 1 1 1 0 0 1 1

One’s Complement Representation



One’s Complement Representation



One’s Complement Representation

What problems still exist?

One’s Complement Representation

What problems still exist?

- We have two distinct representations of ‘zero’:

0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0

One’s Complement Representation

What problems still exist?

- We can’t directly add a positive and a negative number:

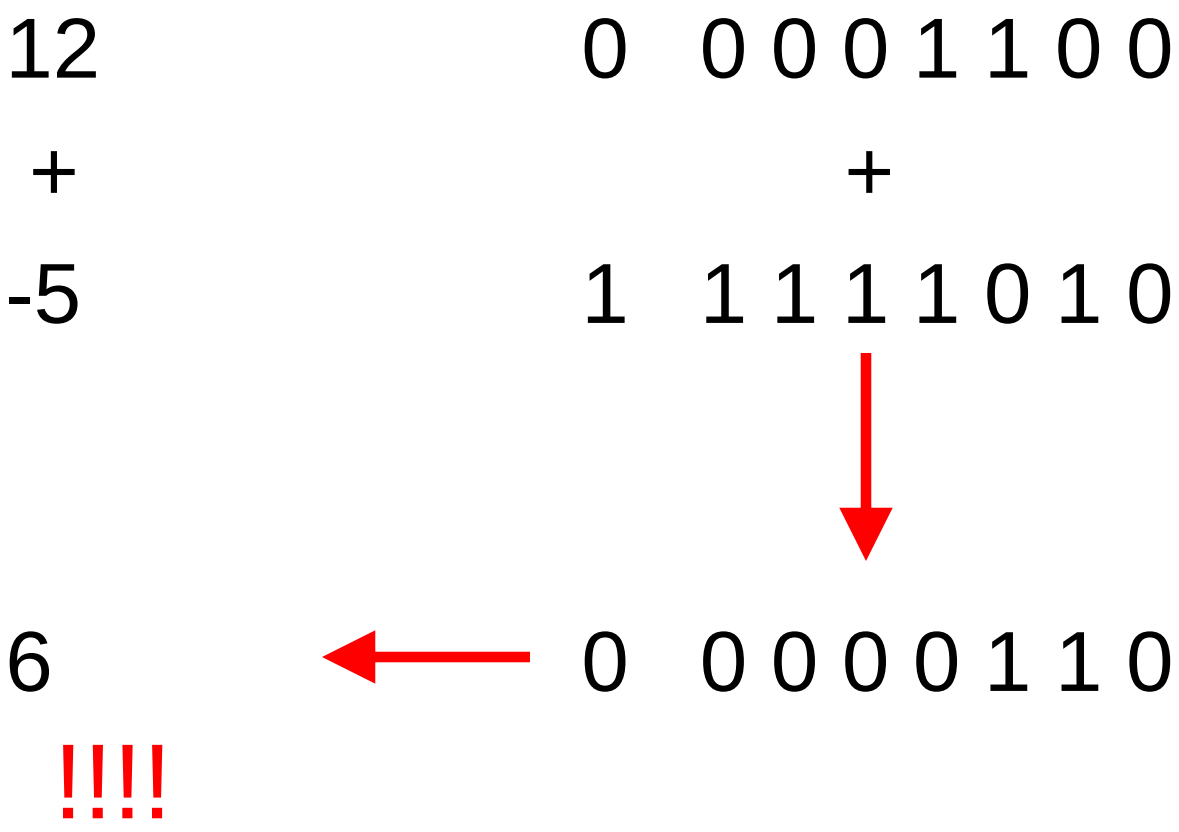
12	0	0 0 0 1 1 0 0
+		+
-5	1	1 1 1 1 0 1 0

One’s Complement Representation

12	0 0 0 0 1 1 0 0
+	+
-5	1 1 1 1 1 0 1 0

0 0 0 0 0 1 1 0

One’s Complement Representation



Representing Negative Numbers

An alternative:

(a little intuition first)

0 0 0 0 0 0 0 0

Decrement

Representing Negative Numbers

An alternative:

(a little intuition first)

0 0 0 0 0 0 0 0 0

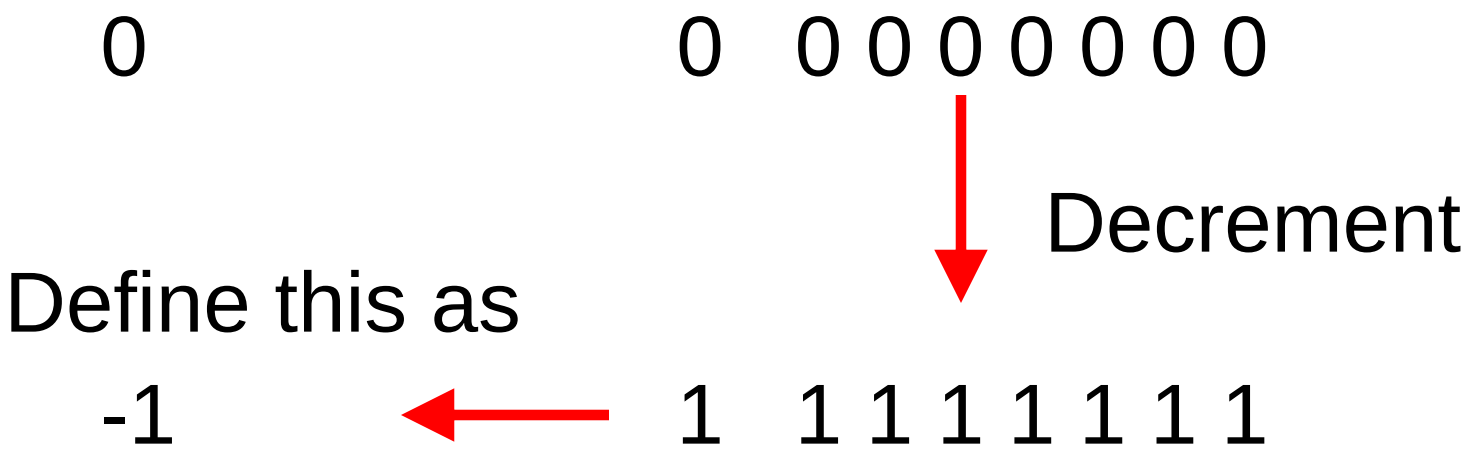
Decrement

1 1 1 1 1 1 1 1

Representing Negative Numbers

An alternative:

(a little intuition first)



Representing Negative Numbers

A few more numbers:

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

Two’s Complement Representation

In general, how do we take the additive inverse of a binary number?

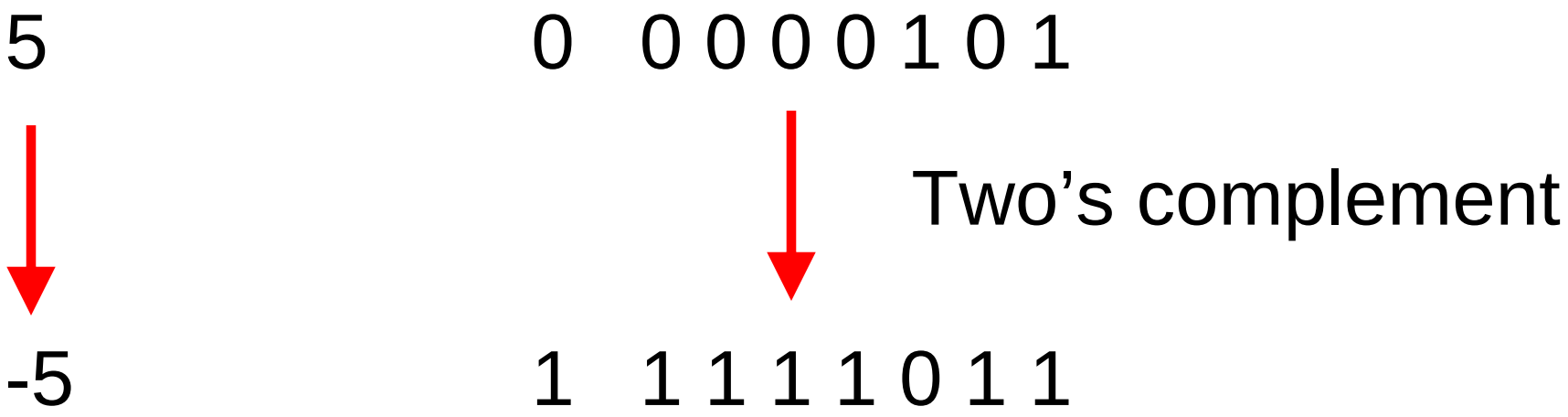
Two’s Complement Representation

In general, how do we take the additive inverse of a binary number?

- Invert each bit and then add ‘1’

Two’s Complement Representation

Invert each bit and then add ‘1’



Two's Complement Representation

Now: let's try adding a positive and a negative number:

12	0 0 0 0 1 1 0 0
+	+
-5	1 1 1 1 1 0 1 1

Two's Complement Representation

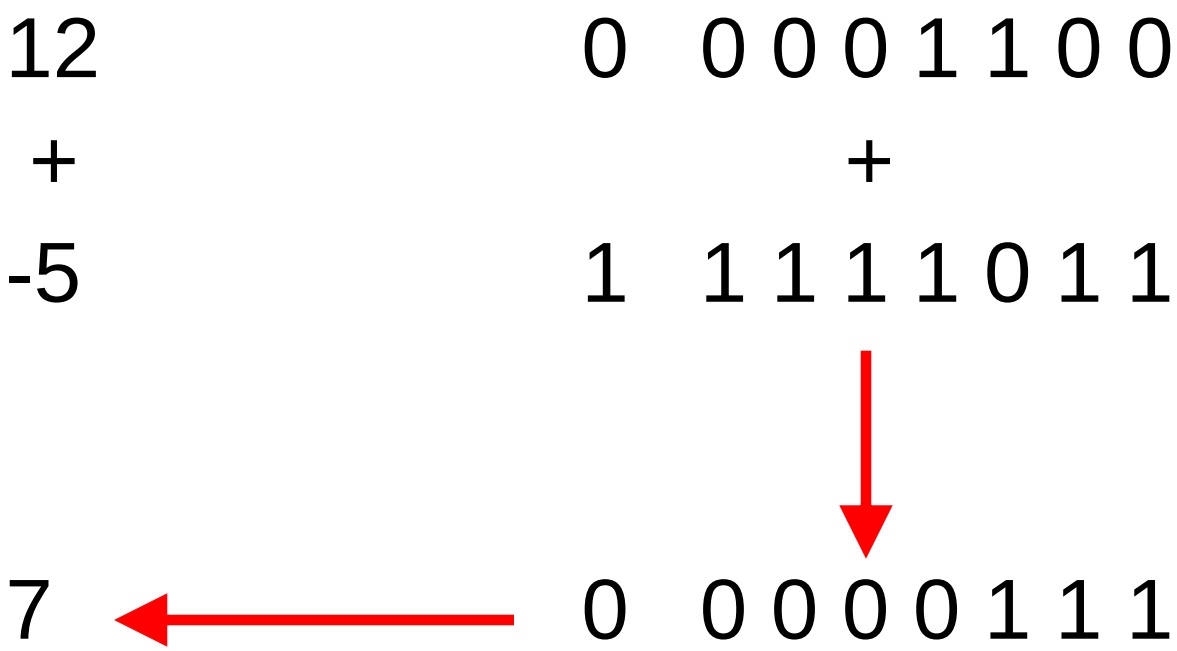
Now: let's try adding a positive and a negative number:

12	0 0 0 0 1 1 0 0
+	+
-5	1 1 1 1 1 0 1 1

0 0 0 0 0 1 1 1

Two's Complement Representation

Now: let's try adding a positive and a negative number:



Two’s Complement Representation

Two’s complement is used for integer representation in today’s processors

Two’s Complement Representation

Two’s complement is used for integer representation in today’s processors

One oddity: we can represent one more negative number than we can positive numbers

Implementing Subtraction

How do we implement a ‘subtraction’ operator?

(e.g., A – B)

Implementing Subtraction

How do we implement a ‘subtraction’ operator?

(e.g., A – B)

- Take the 2s complement of B
- Then add this number to A