

Understand
how computers
calculate

Know when
computers fail

Understand
how computers
represent
numbers



Integers

- ① Overflow
- ② Half / full adder
- ③ Signed / unsigned integers

Perform the following number conversions:

A. 0x39A7F8 to binary

0011 1001 1010 0111 1111 1000

B. Binary 1100 1001 0111 1011 to hexadecimal

C 9 7 B

C. 0xD5E4C to binary

1101 0101 1110 0100 1100

D. Binary 10 0110 1110 0111 1011 0101 to hexadecimal

2 6 E 7 B 5

Pentium FDIV bug: \$475 million



$$\frac{4,195,835}{3,145,727} = 1.333739068902037589$$

Ariane 5 Overflow bug: \$7 billion



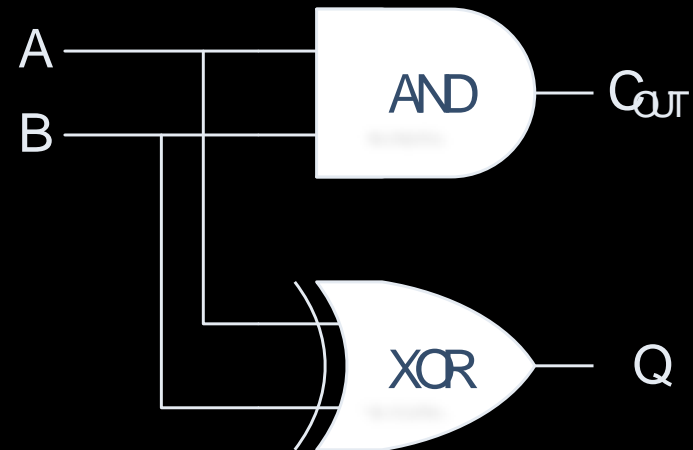
Half Adder

0	+	0	=	00
0	+	1	=	01
1	+	1	=	01
1	+	0	=	10

A	B	Q	C _{OUT}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

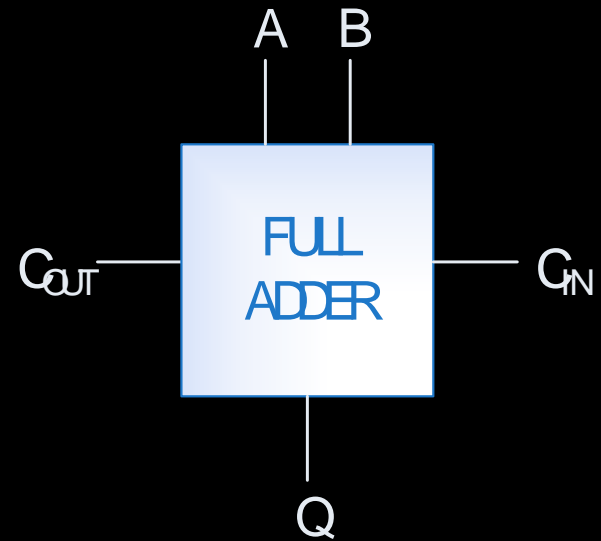
XOR

AND

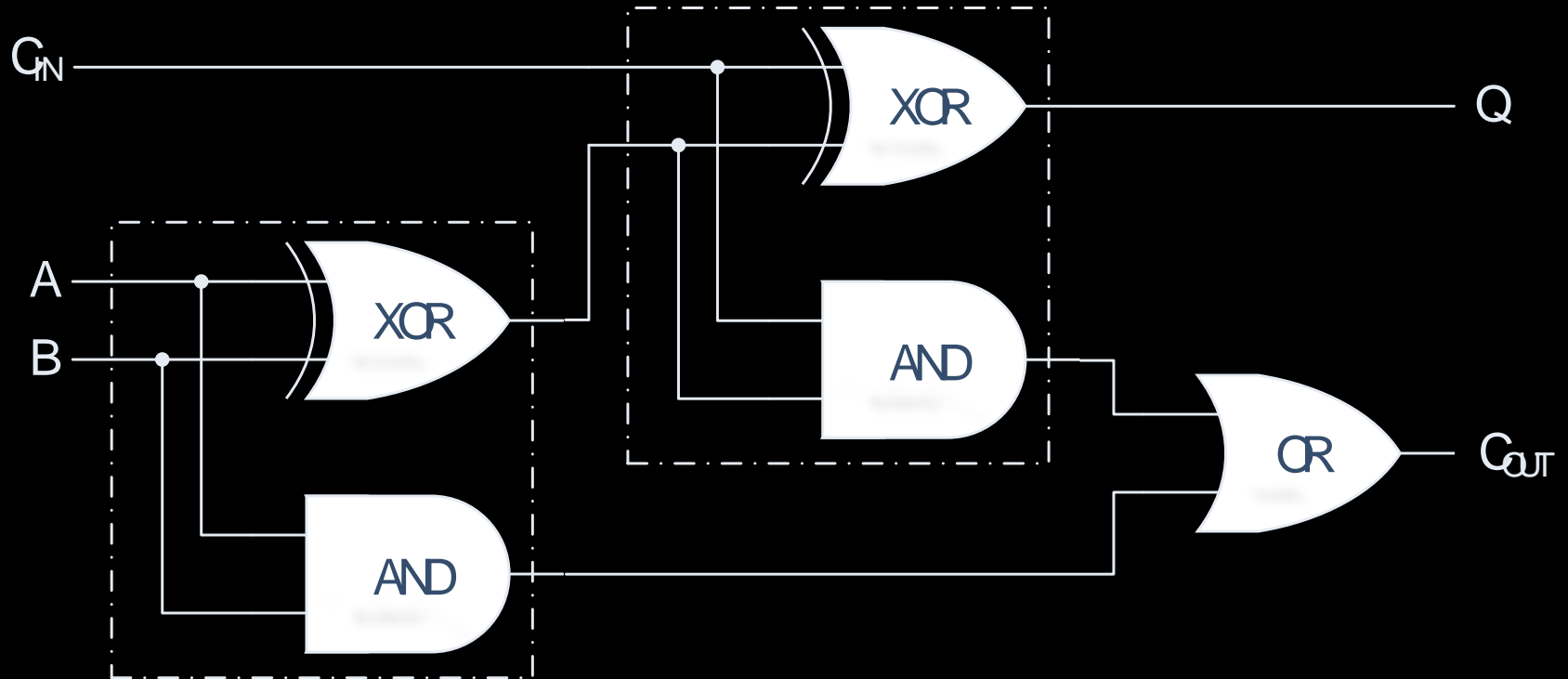


Full Adder

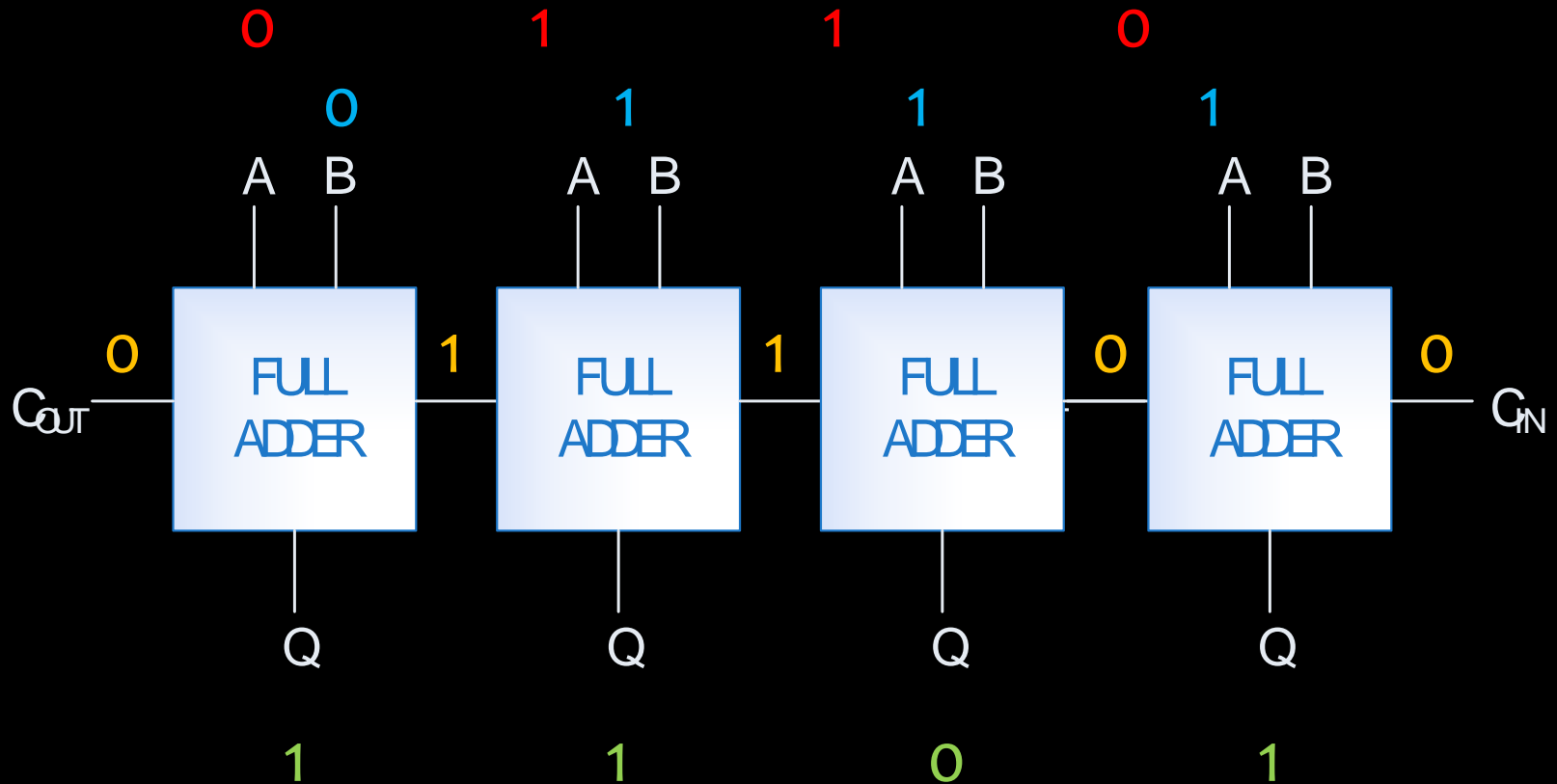
C_{IN}	A	B	Q	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Full Adder



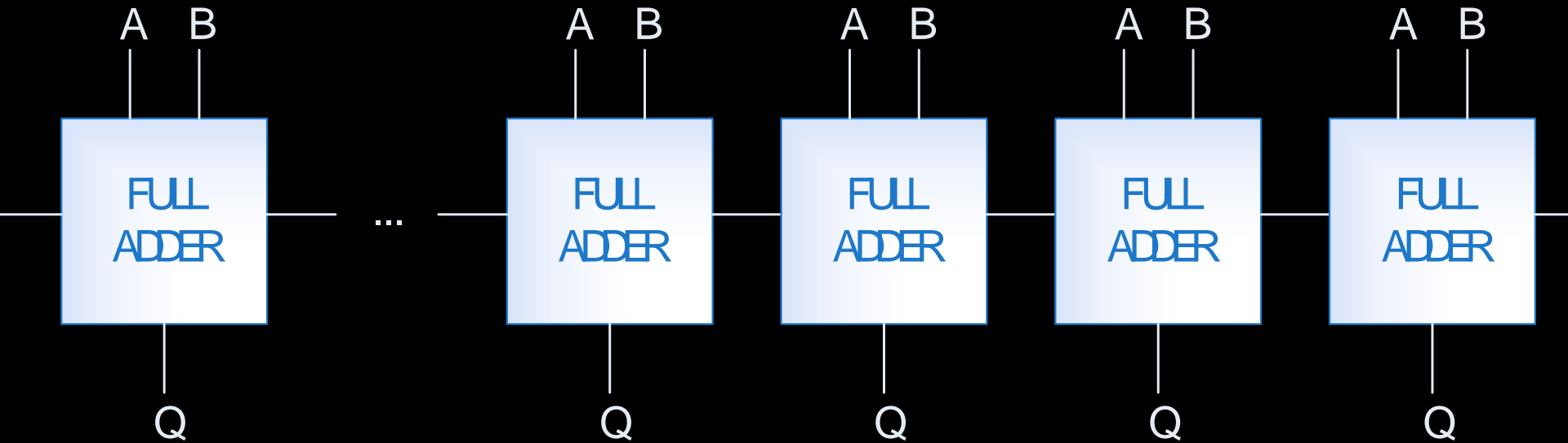
N-bit Full Adder



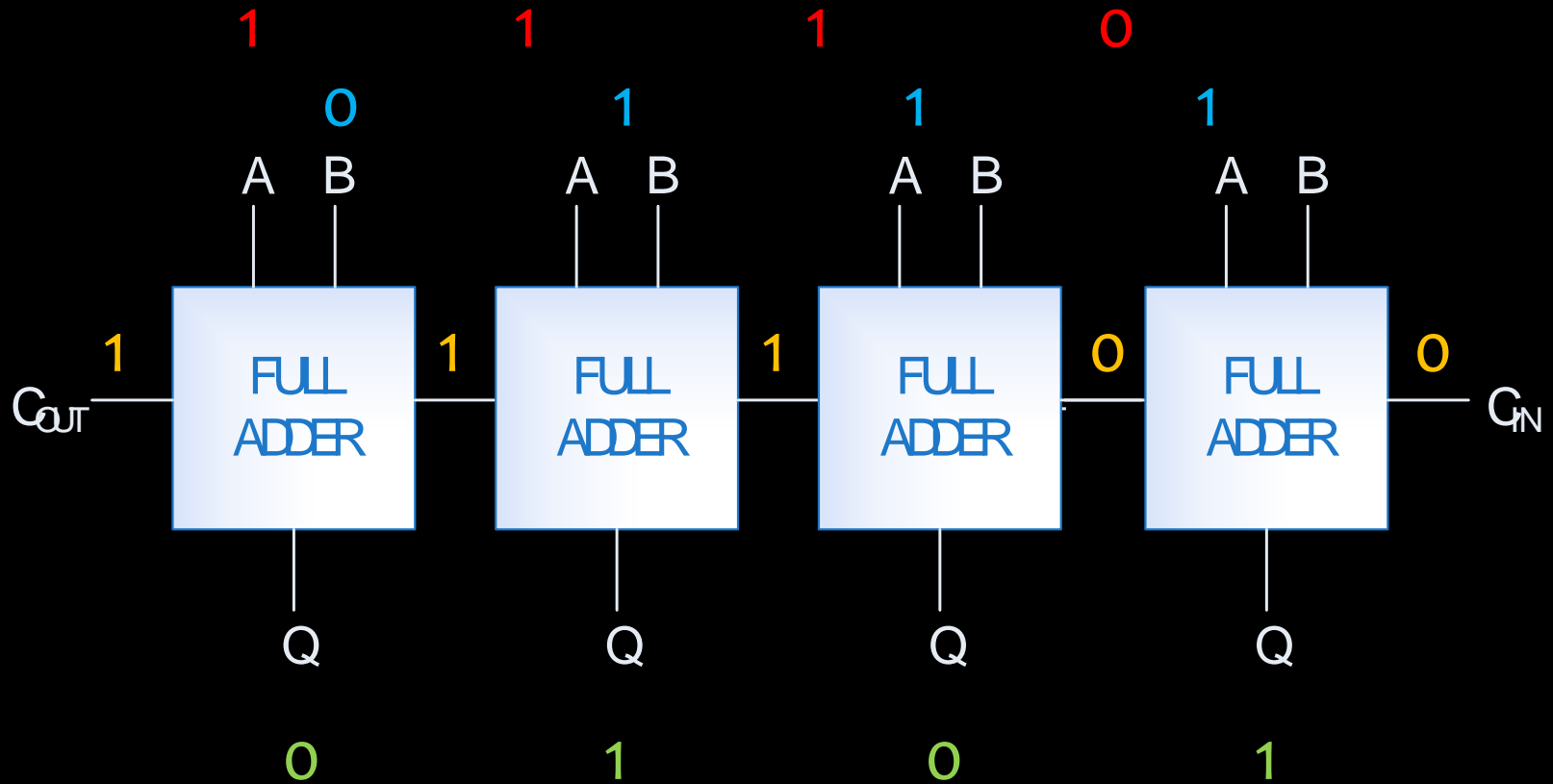
$$\begin{array}{r} 0110 \\ + 0111 \\ \hline 1101 \end{array}$$

6 + 7 = 13

Ripple-Carry Adder



Arithmetic overflow



$$\begin{array}{r}
 1110 \\
 + 0111 \\
 \hline
 0101
 \end{array}$$

14 + 7 = 5



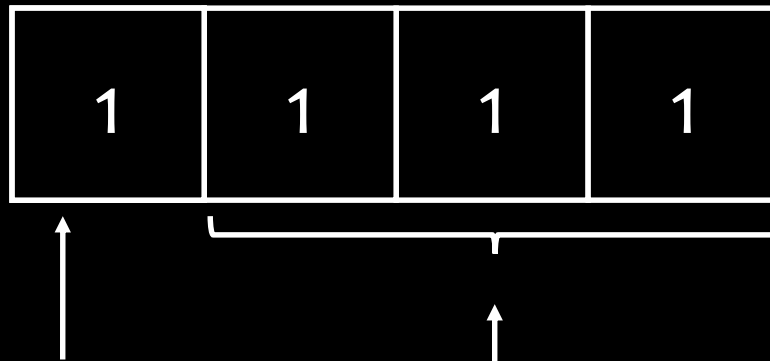
Register width

	Maximum representable value		2005
8 bits	2^8-1	255	←
16 bits	$2^{16}-1$	65,535	
32 bits	$2^{32}-1$	4,294,967,295	
64 bits	$2^{64}-1$	18,446,744,073,709,551,615	←
128 bits	$2^{128}-1$	340 billion billion billion billion	
			2012

Negative numbers?

7

-7



Sign-and-Magnitude

0x00 = 0 0 0 0 0 0 0 0₂ is non-negative, because the sign bit is 0

0x7F = 0 1 1 1 1 1 1 1₂ is non-negative

0x85 = 1 0 0 0 0 1 0 1₂ is negative

0x80 = 1 0 0 0 0 0 0 0₂ is negative



Binary				Decimal
0	0	0	0	+0
0	0	0	1	+1
0	0	1	0	+2
0	0	1	1	+3
0	1	0	0	+4
0	1	0	1	+5
0	1	1	0	+6
0	1	1	1	+7
1	0	0	0	-0
1	0	0	1	-1
1	0	1	0	-2
1	0	1	1	-3
1	1	0	0	-4
1	1	0	1	-5
1	1	1	0	-6
1	1	1	1	-7

$$\begin{array}{r}
 0100 \\
 + 1011 \\
 \hline
 1111
 \end{array}
 \qquad
 \begin{array}{r}
 4 \\
 + -3 \\
 \hline
 -7
 \end{array}$$



math is cumbersome

Binary				Decimal
0	0	0	0	0
0	0	0	1	+1
0	0	1	0	+2
0	0	1	1	+3
0	1	0	0	+4
0	1	0	1	+5
0	1	1	0	+6
0	1	1	1	+7
1	0	0	0	-8
1	0	0	1	-7
1	0	1	0	-6
1	0	1	1	-5
1	1	0	0	-4
1	1	0	1	-3
1	1	1	0	-2
1	1	1	1	-1

$$\begin{array}{r}
 0100 \\
 + 1101 \\
 \hline
 0001
 \end{array}$$

$$\begin{array}{r}
 4 \\
 + -3 \\
 \hline
 1
 \end{array}$$

Two's Complement Negatives

MSB have same value, but negative weight

$$1 \ 0 \ 1 \ 0 \quad -1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = -6_{10}$$

$$0 \ 0 \ 1 \ 0 \quad -0*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 2_{10}$$

$$1 \ 1 \ 1 \ 0 \quad -1*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = -2_{10}$$

$$0x00 = 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0_2 \quad 0$$

$$0x7F = 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1_2 \quad 127$$

$$0x85 = 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1_2 \quad -123$$

$$0x80 = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0_2 \quad -128$$

$$\underbrace{\sim X + 1}_{\text{one's complement}} = -X$$

one's complement

two's complement

Two's Complement

8 bits

$$\begin{array}{r}
 \\
 0 0 0 0 0 1 \\
 + 1 1 1 1 1 1 \\
 \hline
 1 0 0 0 0 0 0
 \end{array}
 \qquad
 \begin{array}{r}
 \\
 1 \\
 \hline
 = 0
 \end{array}$$

$$\begin{array}{r}
 \\
 0 0 0 0 1 0 \\
 + 1 1 1 1 1 0 \\
 \hline
 1 0 0 0 0 0 0
 \end{array}
 \qquad
 \begin{array}{r}
 \\
 2 \\
 \hline
 + -2 \\
 \hline
 = 0
 \end{array}$$

$$\begin{array}{r}
 \\
 0 0 0 0 1 1 \\
 + 1 1 1 1 1 0 1 \\
 \hline
 1 0 0 0 0 0 0
 \end{array}
 \qquad
 \begin{array}{r}
 \\
 3 \\
 \hline
 + -3 \\
 \hline
 = 0
 \end{array}$$

C data type	No. of bit	Minimum	Maximum
char	8	-128	127
unsigned char	8	0	255
short	16	-32,768	32,767
unsigned short	16	0	65,535
int	32	-2.147 Billion	2.147 Billion
unsigned	32	0	4.295 Billion
long	64	-9.2 Quintillion	9.2 Quintillion
unsigned long	64	0	18.4 Quintillion
long long	64	-9.2 Quintillion	9.2 Quintillion
unsigned long long	64	0	18.4 Quintillion

1 1 1 1 1 1 1 1



Bits are unchanged, just interpreted differently!

If you mix unsigned and signed in a single expression,
then signed values implicitly cast to unsigned

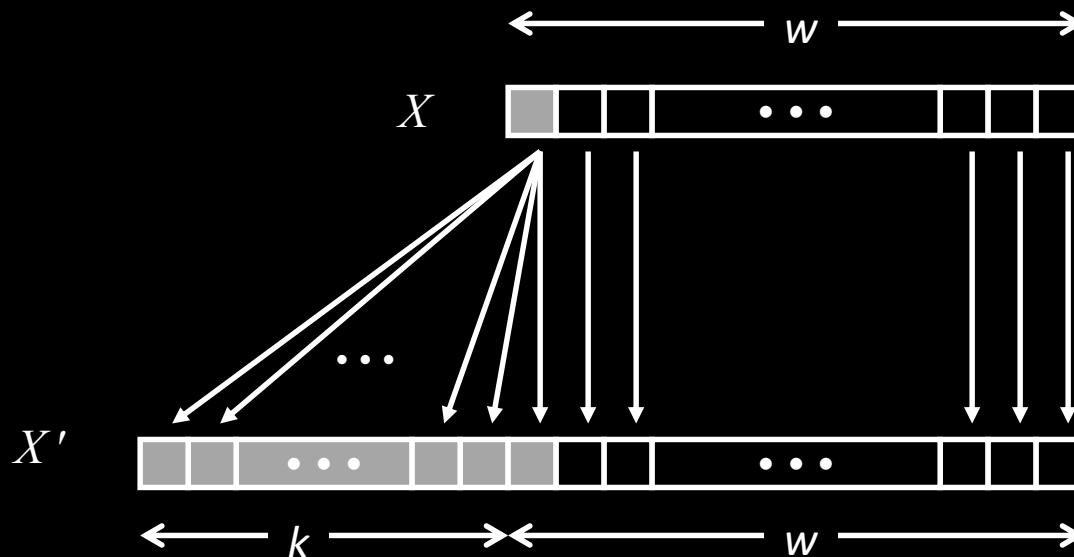
Constant ₁	Constant ₂	Relation	Evaluation
0	0U	==	unsigned
-1	0	<	signed
-1	0U	>	unsigned
2,147,483,647	-2,147,483,648	>	signed
2,147,483,647U	-2,147,483,648	<	unsigned
-1	-2	>	signed
(unsigned) -1	-2	>	unsigned
2,147,483,647	2,147,483,648U	<	unsigned
2,147,483,647	(int) 2,147,483,648U	>	signed

Sign Extension

Task:

Given w -bit signed integer x ,
convert it to $w+k$ -bit integer *with same value*

Rule: Make k copies of sign bit:



Sign Extension Example

Converting from smaller to larger integer data type

C automatically performs sign extension

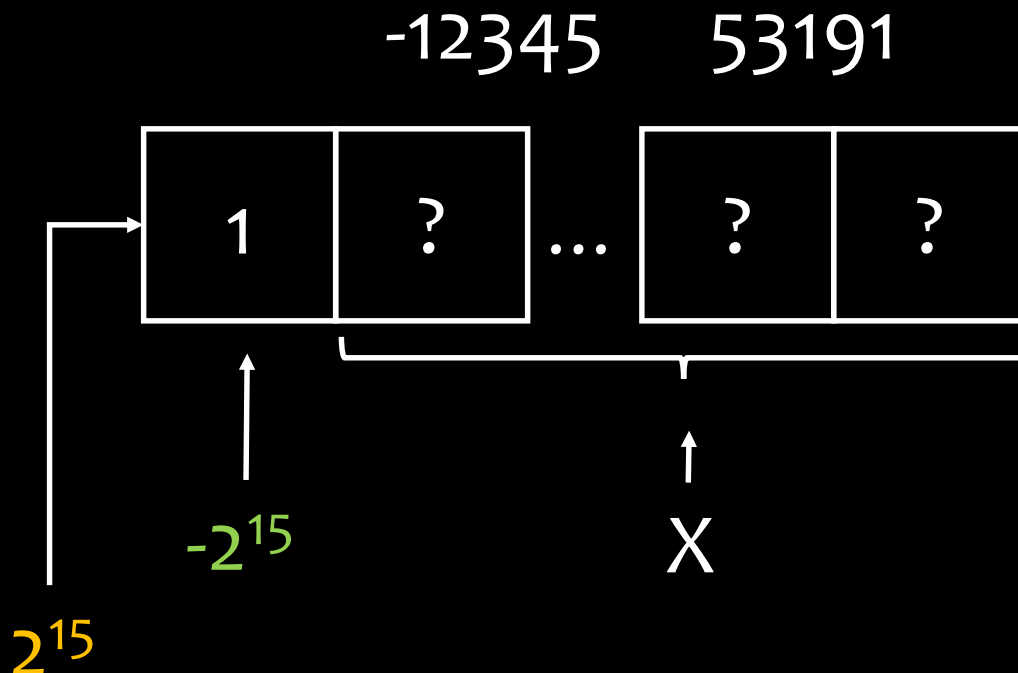
```
short int  x = 12345;  
int       ix = (int)x;  
short int  y = -12345;  
int       iy = (int)y;
```

x	Decimal	Hex	Binary
x	12345	30 39	00110000 01101001
ix	12345	00 00 30 39	00000000 00000000 00110000 01101101
y	-12345	CF C7	11001111 11000111
iy	-12345	FF FF CF C7	11111111 11111111 11001111 11000111

```

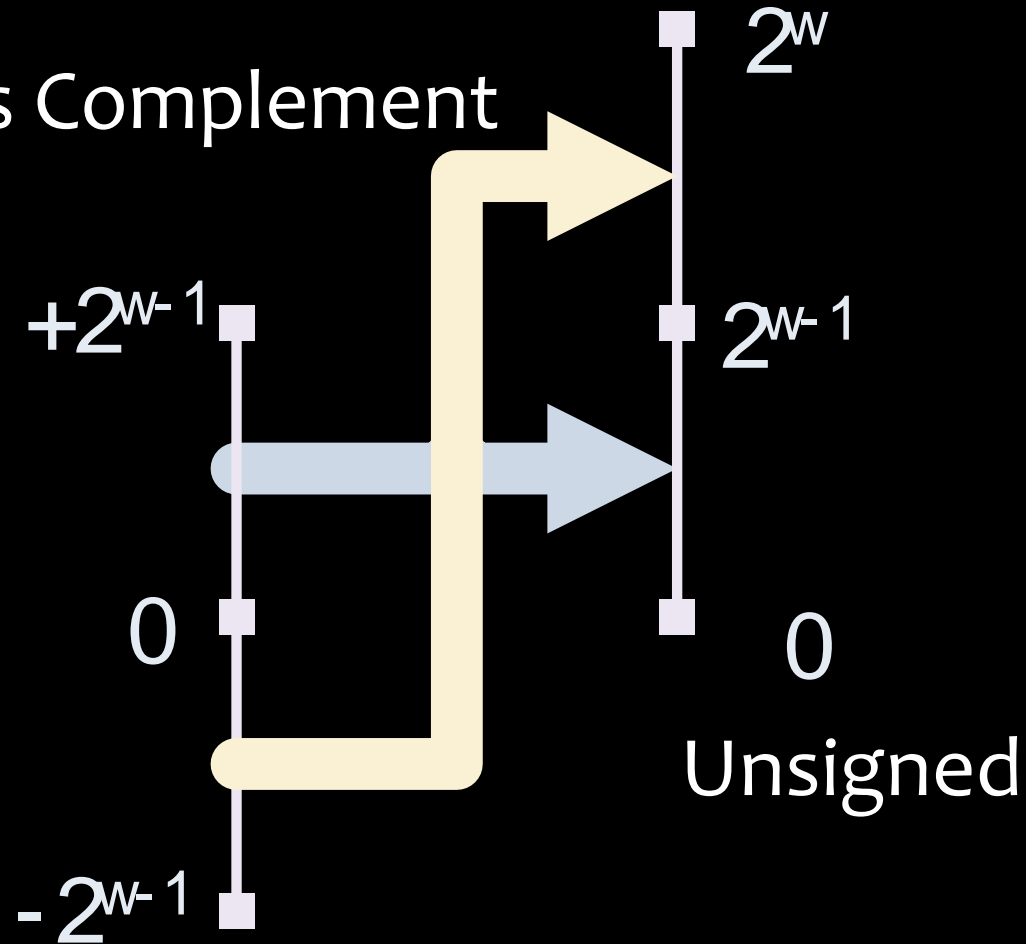
short int v = -12345;
unsigned short uv = (unsigned short)v;
printf("v = %d, uv = %u\n", v, uv);

```



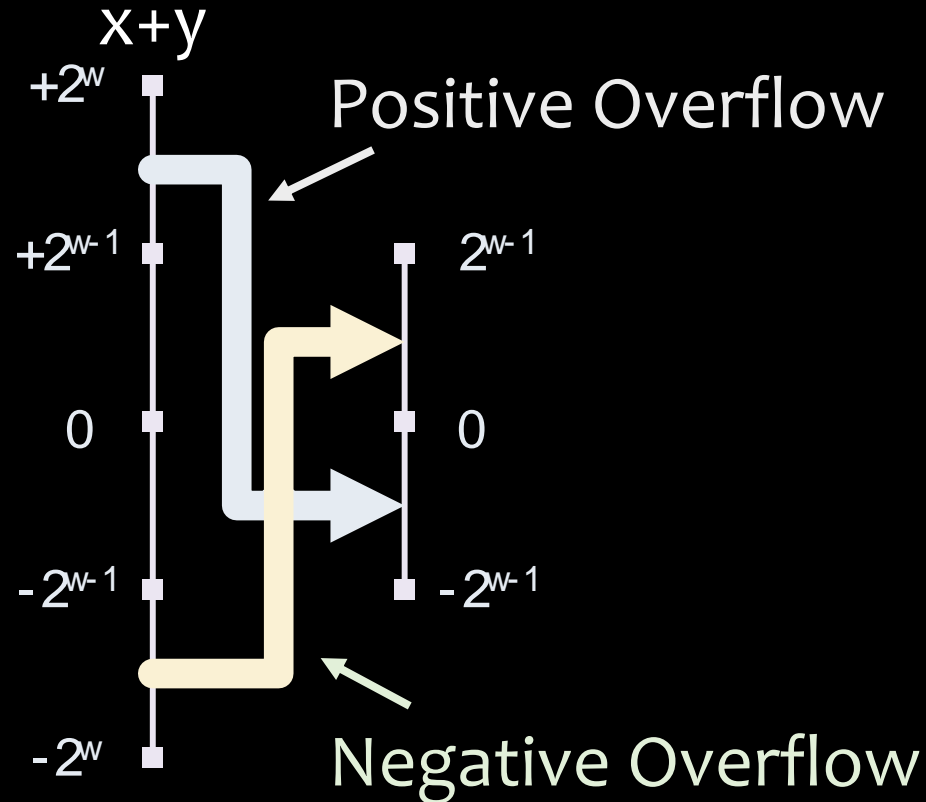
$$\begin{aligned}
 v &= -2^{15} + X \\
 uv &= 2^{15} + X \\
 &= 2^{15} + v - (-2^{15}) \\
 &= v + 2^{16}
 \end{aligned}$$

Two's Complement



Overflow

exceed the range of the representation



Underflow?

Summary

- Signed / Unsigned integers
- Adder
- Overflow

“ There are 10 kinds of people in this world:
those who can count in binary and those
who can't. ”