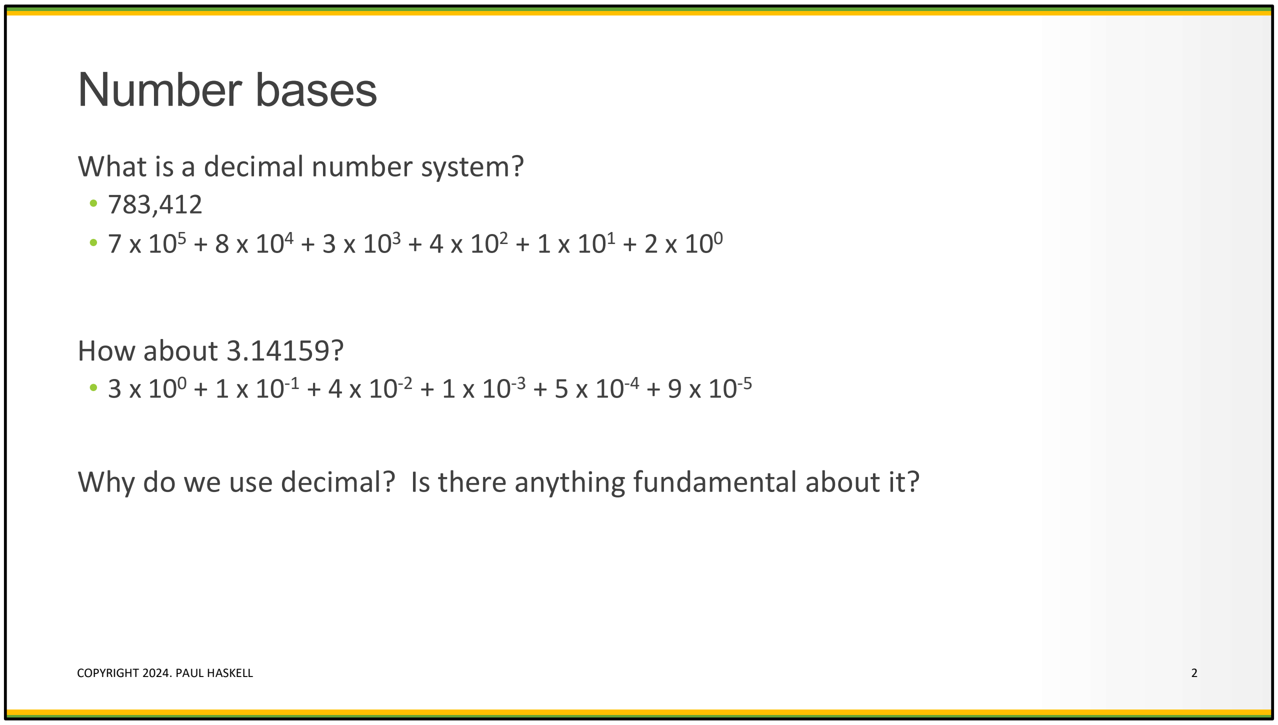
CS221
C and Systems Programming



Number bases

What is a decimal number system?

- 783,412
- $7 \times 10^5 + 8 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 1 \times 10^1 + 2 \times 10^0$

How about 3.14159?

- $3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2} + 1 \times 10^{-3} + 5 \times 10^{-4} + 9 \times 10^{-5}$

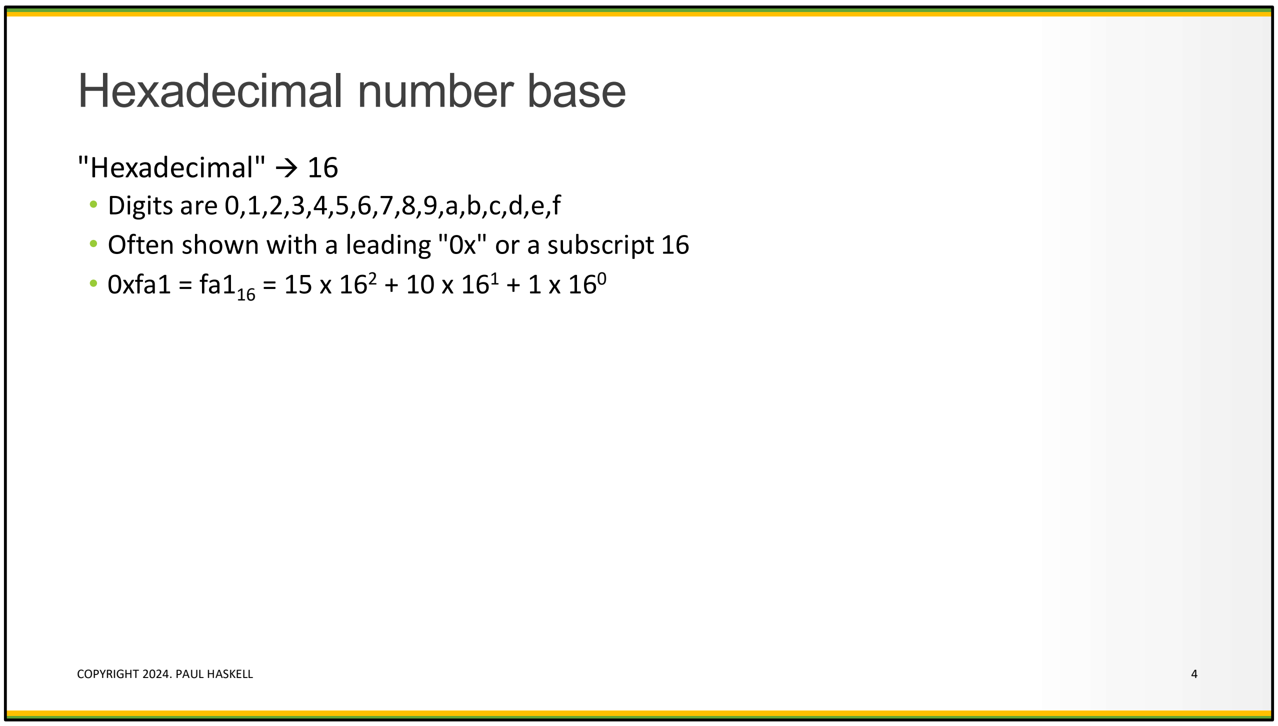
Why do we use decimal? Is there anything fundamental about it?

Mathematically, nothing fundamentally important about decimal

Octal number base

"Octal" → 8

- Digits are 0,1,2,3,4,5,6,7
- Often shown with a leading "0" or a subscript 8
- 0376 = $376_8 = 3 \times 8^2 + 7 \times 8^1 + 6 \times 8^0$ (= 254_{10})



Hexadecimal number base

"Hexadecimal" → 16

- Digits are 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f
- Often shown with a leading "0x" or a subscript 16
- 0xfa1 = $fa1_{16} = 15 \times 16^2 + 10 \times 16^1 + 1 \times 16^0$

a=10, b=11, …f=15

Binary number base

"Binary" → 2

- Digits are 0,1
- Often shown with a leading "b" or a subscript 2
- b1101 = $1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

Computer circuits (memory, CPU) work with binary – why?

Octal and hexadecimal are useful to humans

- More compact than binary
- Easily convert 3 or 4 binary digits to/from an octal or hex digit

Examples

Decimal	Octal	Hex	Binary
15			
	0300		
		0xfa	
			0111 1010
128			
	077		
		0x100	
			0101 1110

How are numbers stored in computers?

"bit" = "binary digit"

"byte" = 8 "bits"

How do we store a char, short, int, long?

- char = 1 byte
- short = 2 bytes
- int = 4 bytes (sometimes 2)
- long = 4 bytes

`sizeof()` operator

How are numbers stored in computers?

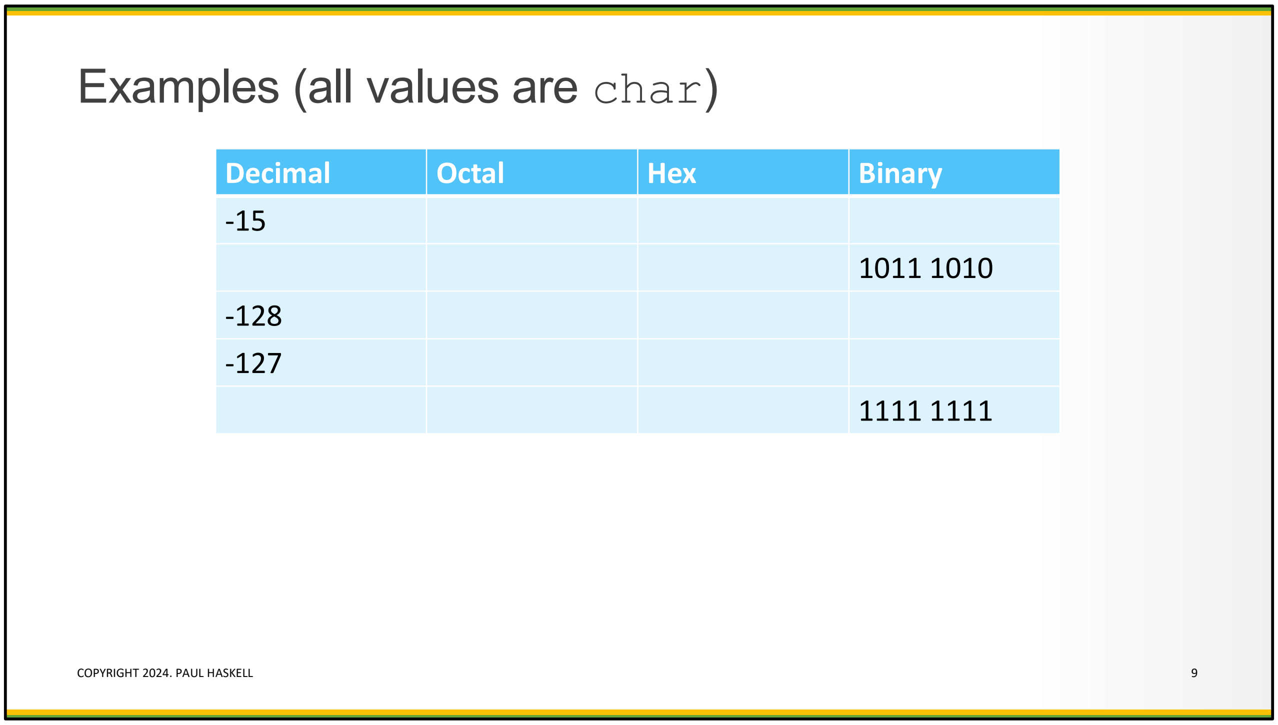
How do we store -42?

Computers use "2's complement"

- Most significant digit is not 2^7, 2^{15}, 2^{31} in a char, short, long
- Most significant digit is -2^7, -2^{15}, -2^{31}
- In a char, 1000 0101 = $-2^7 + 2^2 + 2^0 = -123_{10}$

If first digit is '1', value is <0

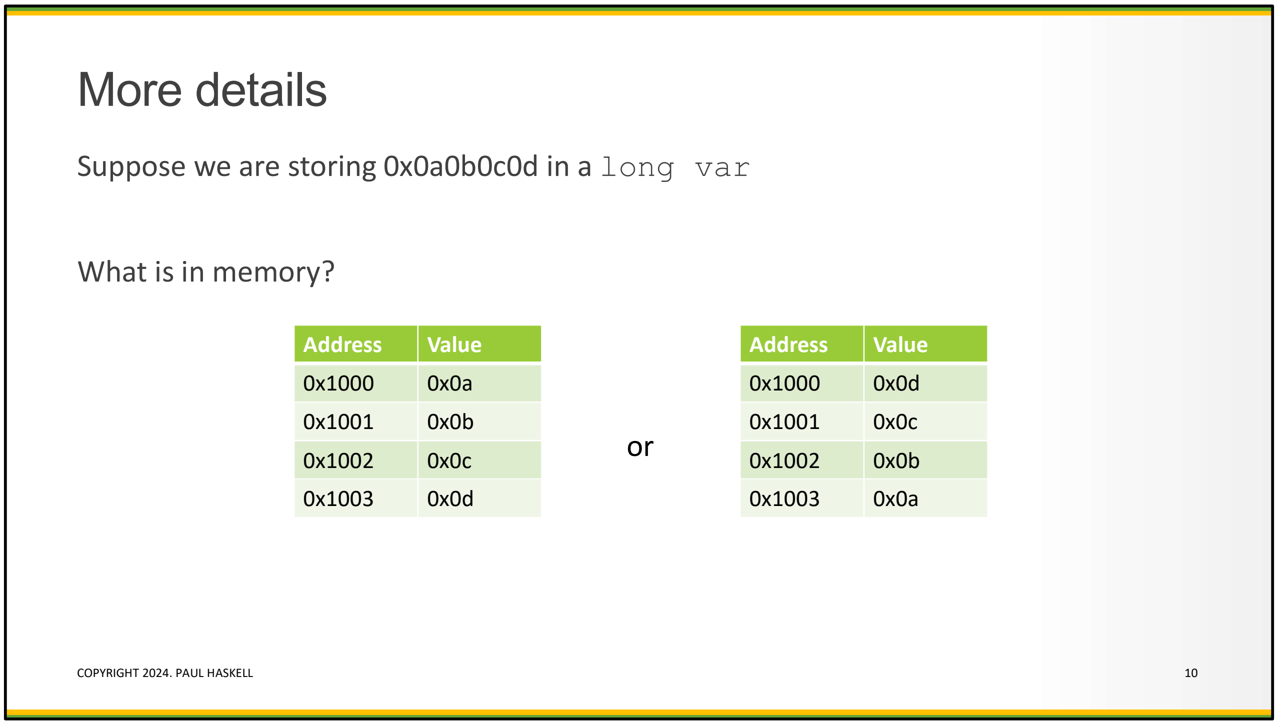
To convert a number to its negative, flip all bits and add +1



Examples (all values are `char`)

Decimal	Octal	Hex	Binary
-15			
			1011 1010
-128			
-127			
			1111 1111

Do -15 two ways: counting up values bit by bit, and as 2's complement of +15
Do 1111 1111 two ways



More details

Suppose we are storing 0x0a0b0c0d in a `long var`

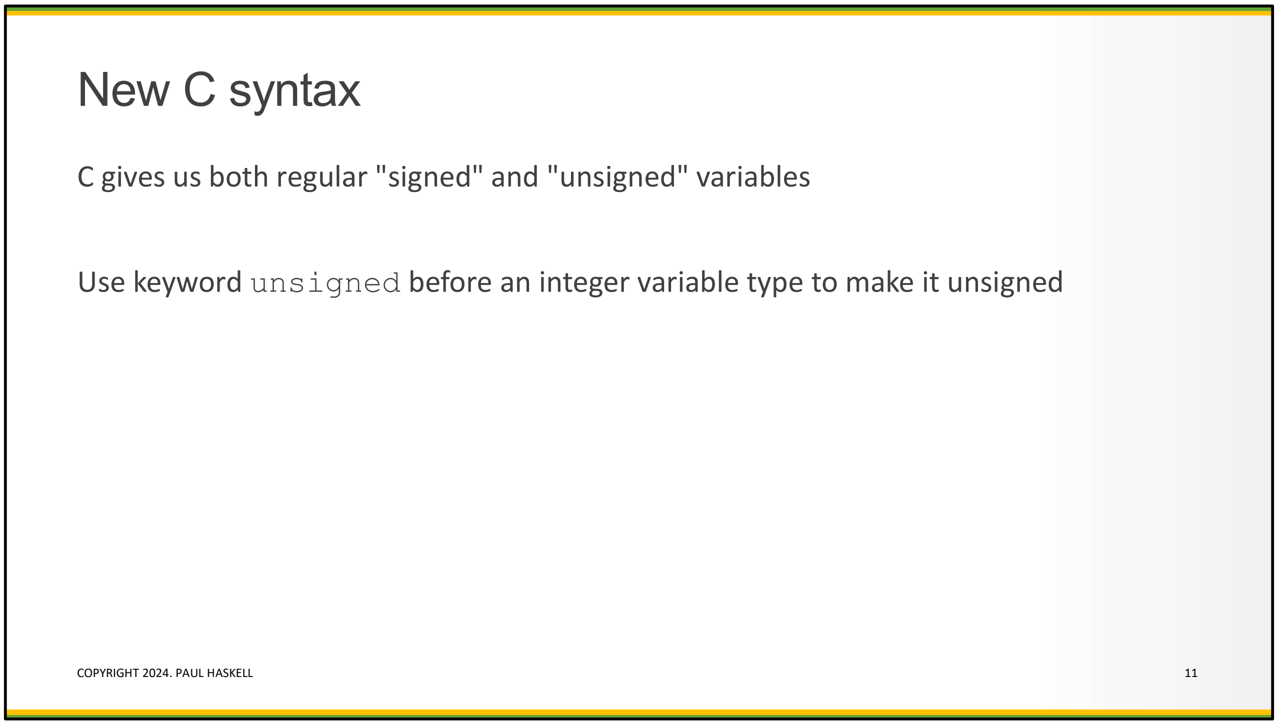
What is in memory?

Address	Value
0x1000	0x0a
0x1001	0x0b
0x1002	0x0c
0x1003	0x0d

or

Address	Value
0x1000	0x0d
0x1001	0x0c
0x1002	0x0b
0x1003	0x0a

Answer: either of em, depending on CPU type. "little endian" vs "big endian"
Important to know answers if exchanging numbers between computers.
Good reasons for each, and the world could not settle on a single answer quickly enough.



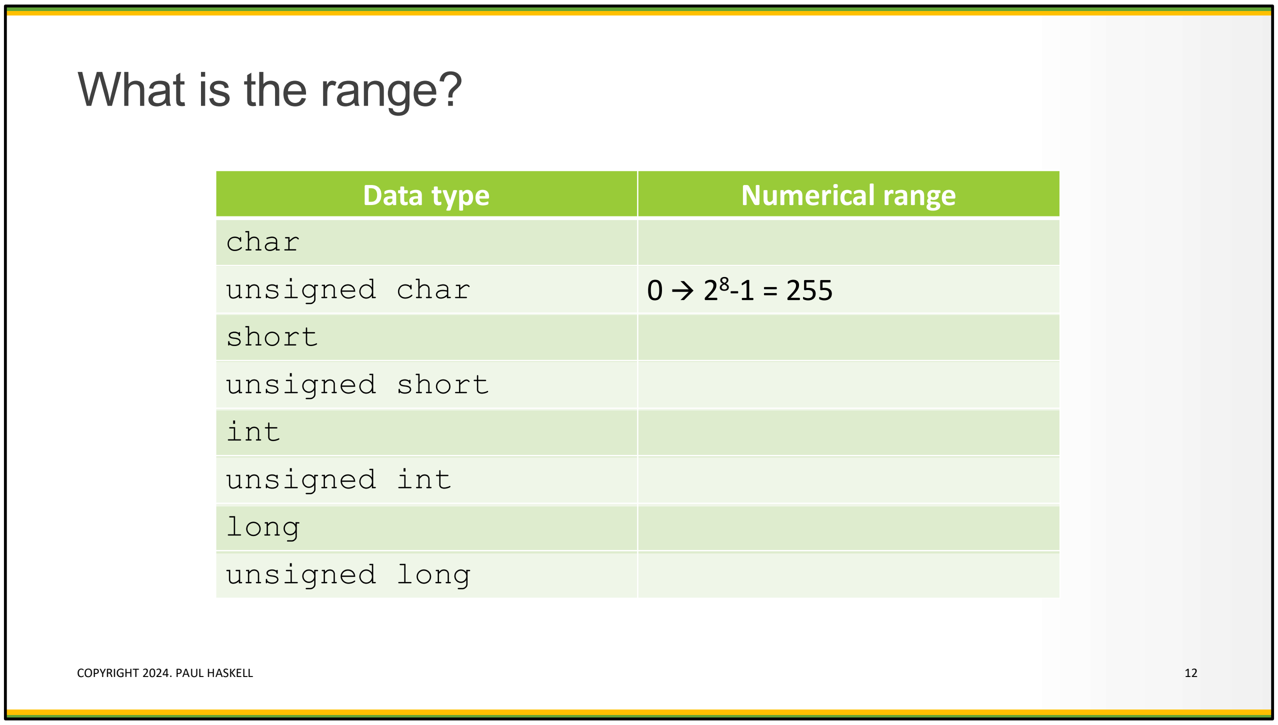
New C syntax

C gives us both regular "signed" and "unsigned" variables

Use keyword `unsigned` before an integer variable type to make it unsigned

Do sample program. Do casting between them. See sample1.c

How to printf() unsigned values?
"u" for unsigned decimal ("d" for signed decimal, "x" for unsigned hex, "o" for unsigned octal)
"l" for long
"h" for short ("half")



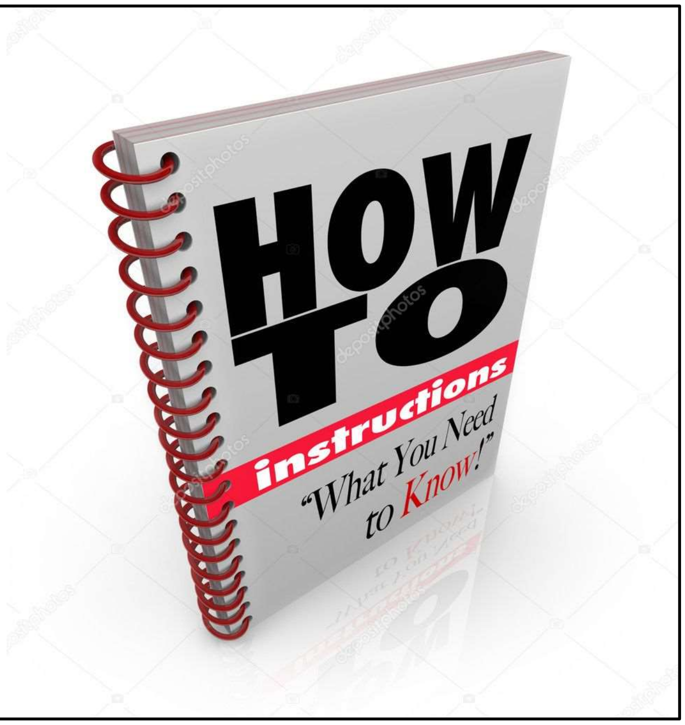
What is the range?

Data type	Numerical range
`char`	
`unsigned char`	$0 \rightarrow 2^8-1 = 255$
`short`	
`unsigned short`	
`int`	
`unsigned int`	
`long`	
`unsigned long`	

12

Do code-along: print lots of char and uchar values. Lots of short and ushort values. Print sizeof()

Manual pages



Available built into Linux and MacOS. On Windows must get from Internet.

Look some up! ls, printf