University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

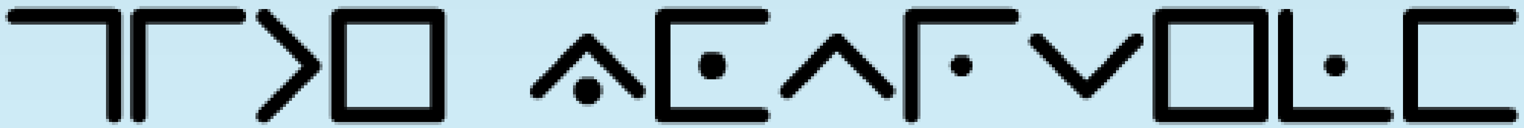
Representations and Bits

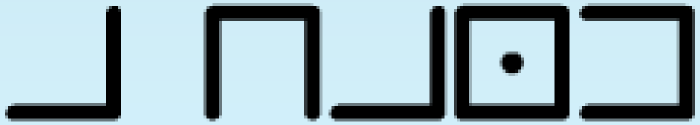
Today's Random Topic: History!

MIT students created the Big Screw Award, which is given to "whoever [sic] has screwed over the most students over the past year."

One professor taught in French to win it.

The rest of my lectures will use this code:



Good luck! 

Represent One Type of Information with Another

We often represent one type of information with other patterns, physical quantities, and so forth.

examples
- English letters represented by drawn patterns
- colors represented by variations in radio signal amplitude

The **mapping from one form to another** is called a **representation**.

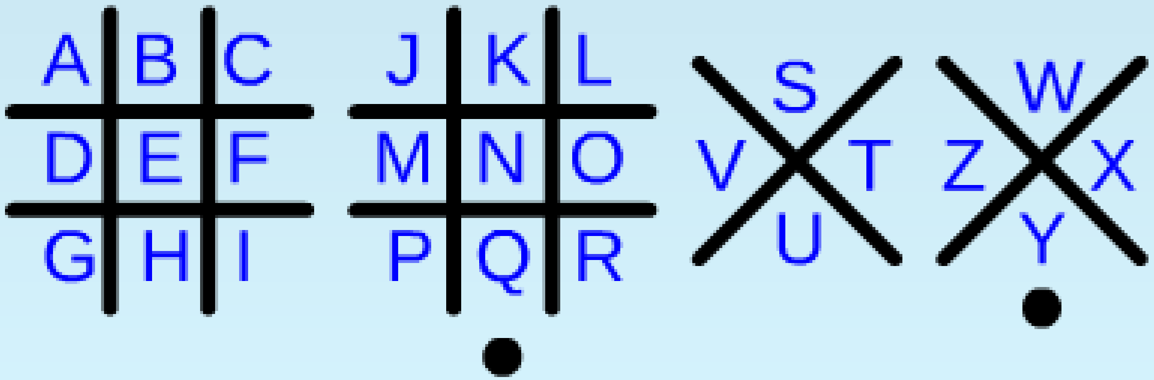
Follow These Simple Instructions

Please do so for the rest of today’s lecture.

You thought I was kidding?

Knowing the Representation May Help You

The code above is called a tic-tac-toe code: each letter (information) is represented by a drawing (pattern).



What Do We Need to Make a Representation Useful?

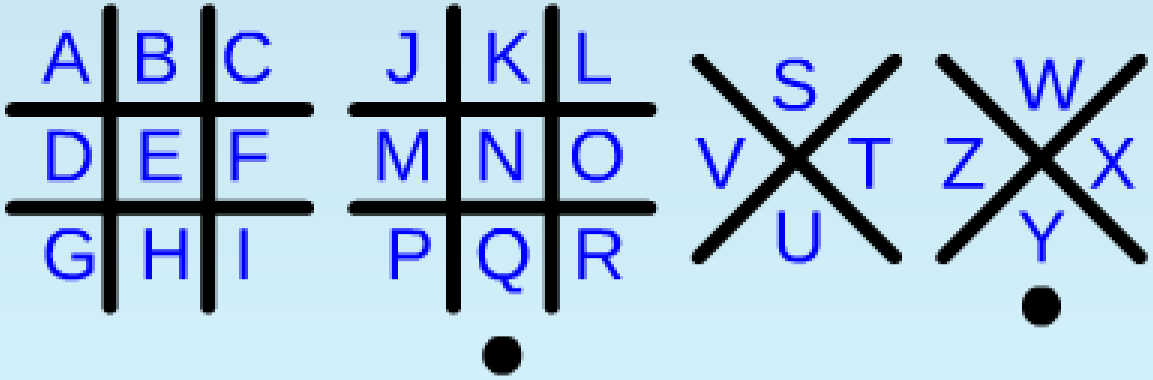
What properties are necessary for a representation to be useful?

Hints:
- Think about the tic-tac-toe code.
- Think about algorithm properties.

First Answer: Representations Must be Well-Defined

All users must **know the translation in advance**.

Our goal is communication, not obfuscation.



Some Mappings May Not be Usable by Computers

0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z				

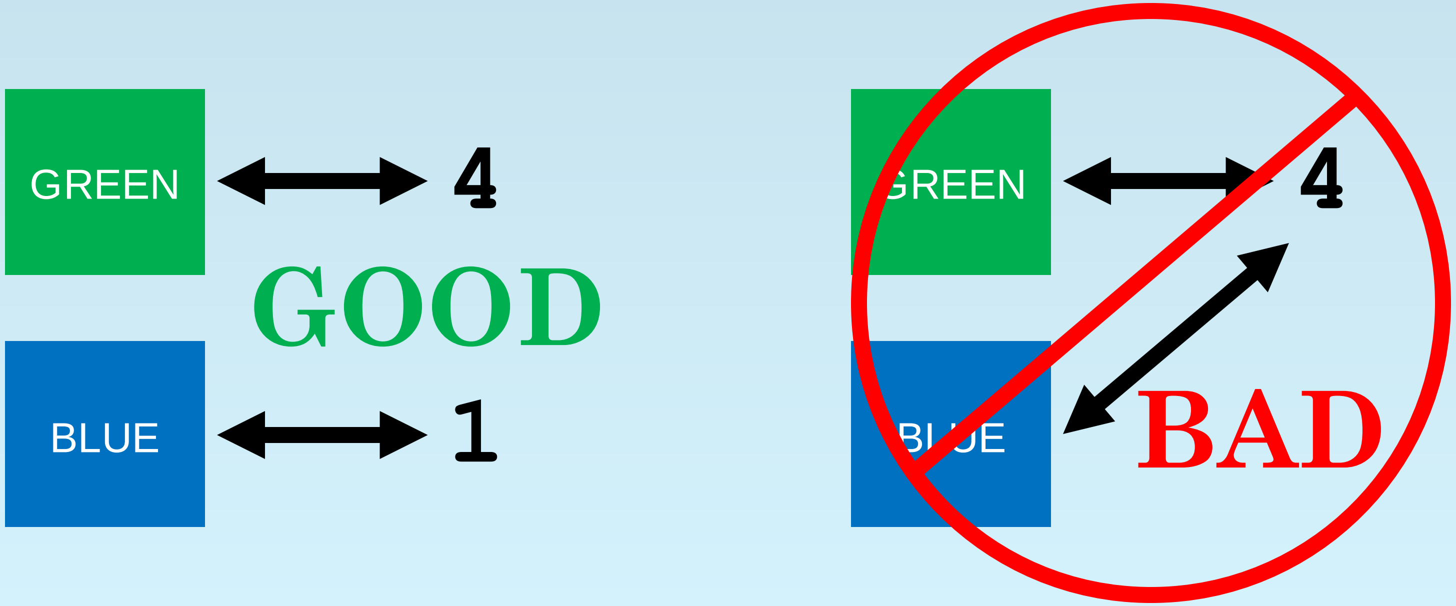
If we use 10 digits to represent 26 letters as shown above, what does "143" mean?

BED? **BOX**? **VYN**?

Computers are dumb—they cannot guess.

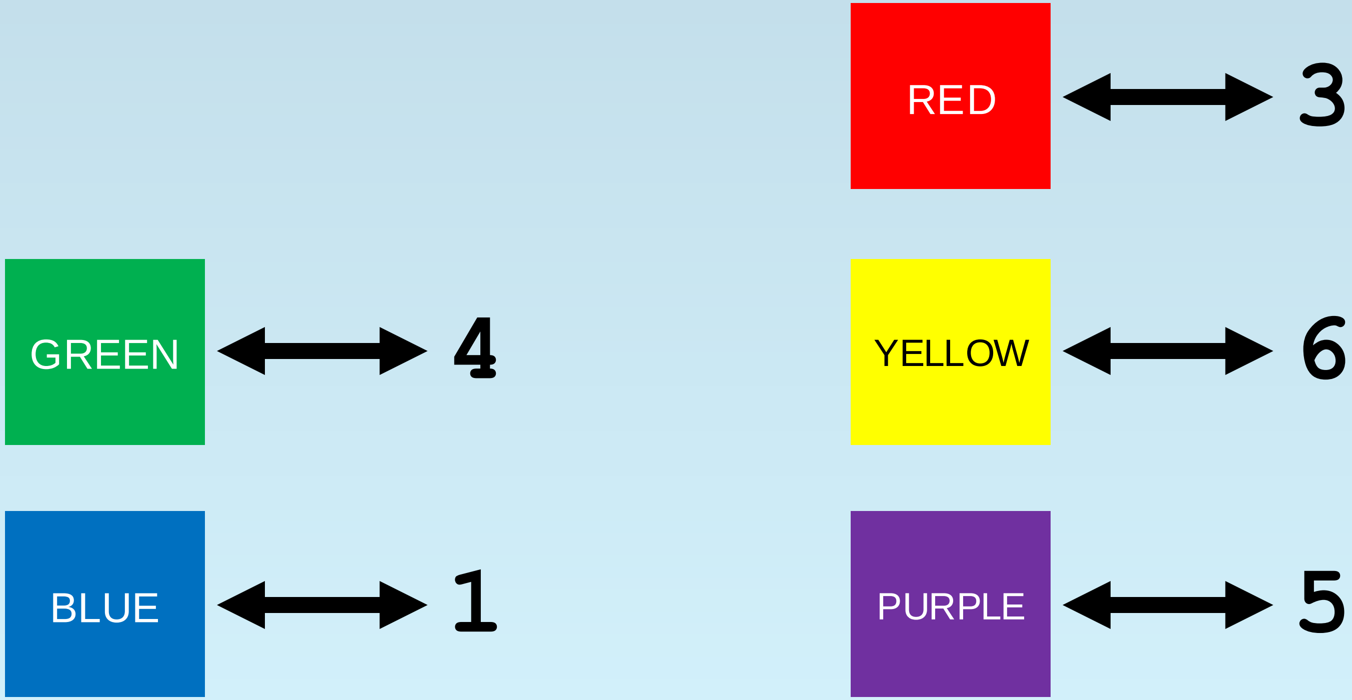
Second Answer: Representations Must be Unambiguous

Each pattern must represent
at most one thing.



But Some Patterns May Represent Nothing

In the representation below, the digits 0, 2, 7, 8, and 9 represent no color.



Computers are Based on Electrons

In digital systems, electrons are **all we have** to represent information!

What can you ask about electrons?

How many electrons are in a certain place? (related closely to voltage)

So…

- Choose a ground: 0V by definition.
- Pick a higher voltage (called V_{dd}).

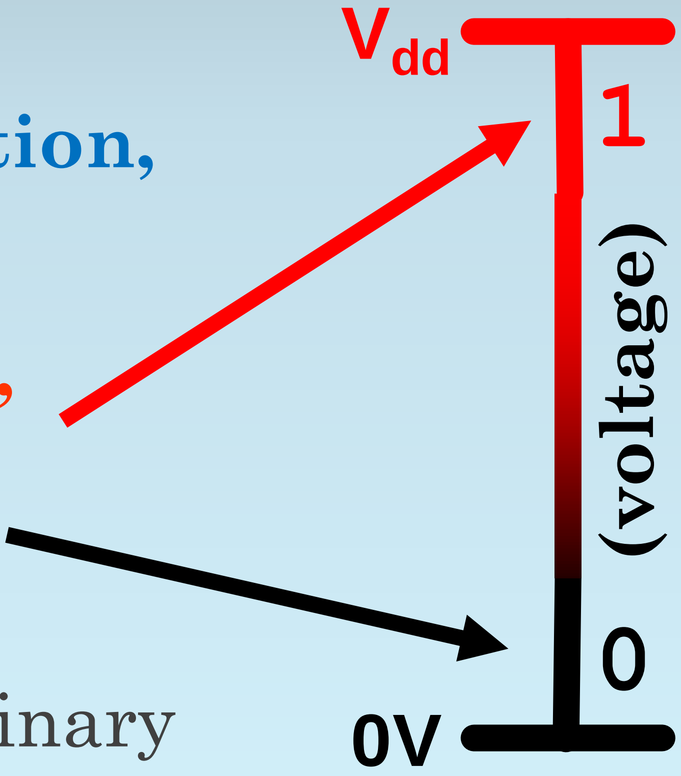
Computer Representations are Based on BInary digiTs

Now ask:

At a given physical location, what is the voltage?

Voltage near V_{dd} is a “1.”

Voltage near 0V is a “0.”

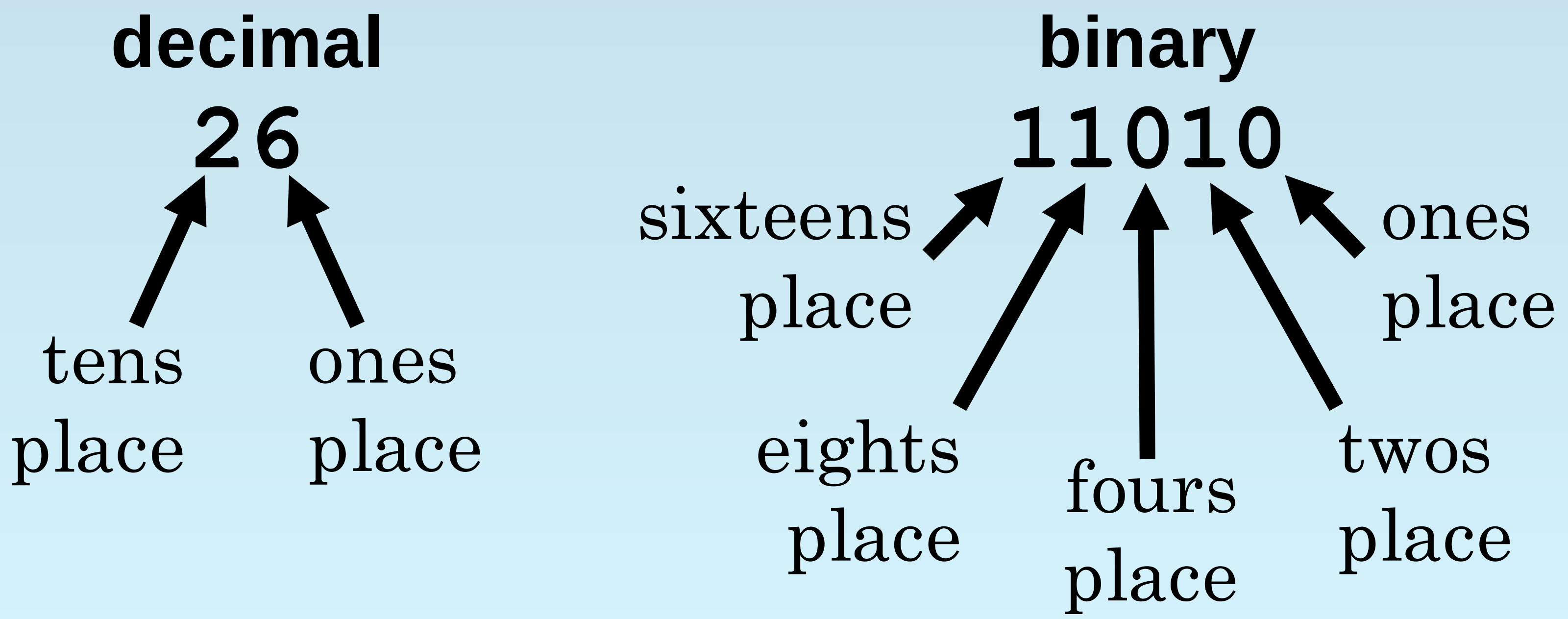


The location thus holds a binary digit, which we call a **bit**.

Physical Locations Enable Place Value

Each bit is somewhere on a computer chip.

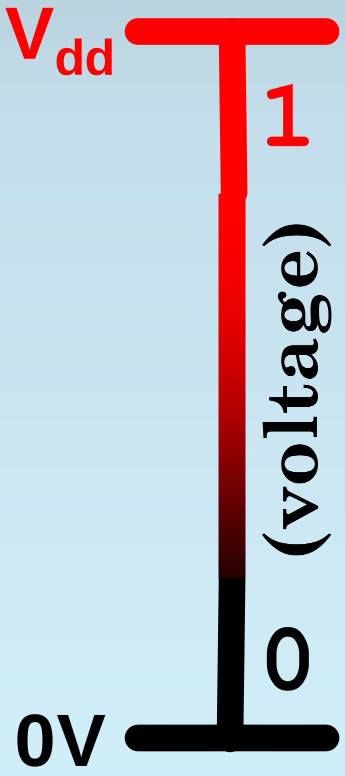
So using positional / **place value** is natural.



Represented by What? The Answer is Always "Bits"

Remember:

- Electrons are all we have inside computers.
- No decimal, no hexadecimal, no letters, no real numbers, no colors.
- **ALL computer representations are based on bits.**



A Question for You: How Many Bits do We Need?

How many bits do we need to represent a whole number in the range…

- from **0 to 31**?
  - 32 different integers
  - so **we need 5 bits** ($2^5 = 32$ bit patterns)
- from **0 to 100**?
  - 101 different integers
  - so **we need 7 bits** ($2^7 = 128$ bit patterns)

We Need One Bit Pattern for Each Possible Thing

Trick question: How many bits do we need to represent two books?

- **The Collected Works of Shakespeare**
- **Our textbook by Patt & Patel**

  - 2 different books
  - so **we need only 1 bit!** ($2^1 = 2$ bit patterns)

What matters is the **number of things**, not what those things are.

How Many Bits Do We Need to Represent N Things?

Let's test your understanding (and generalize)!

How many bits do we need to represent…

- a whole number from **1000 to 1100**?
  101 different integers, so **7 bits** ($2^7 = 128$)
- one of **199 flavors of ice cream**?
  199 different flavors, so **8 bits** ($2^8 = 256$)
- **a living person**?
  7-8 billion people, so **33 bits** ($2^{33} > 8$ billion)
- **N things**?
  $\lceil \log_2 N \rceil$ (ceiling / integer at least as large as log base 2 of N)

University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

The Unsigned Representation

We Can Represent Anything with Bits

Recall: All information in a computer is **represented with bits**.

We can represent anything with bits.*

useful examples: integers
real numbers
human language characters
(alphabet, digits, punctuation)

Important: **Computers do not "know" the meaning of the bits!**

* A computer only stores a finite number of bits, of course!

How Do We Decide What to Represent?

Let's think about integer (whole number) representations.

What numbers should we represent?

- Some random set?
- Everyone in our class' favorite number (mine is 42!)?
- A contiguous set starting with 0?

Does the Representation Matter?

We want computers to do arithmetic.

How does a representation affect arithmetic?

- Imagine that we represent numbers in the range **[100, 131]**.
- We need **5 bits** (32 different numbers).
- What happens if we add two numbers?
- Can we represent the sum using the same representation?

Choose a contiguous range including 0.

Human Representations are Good Choices

Let's borrow a human representation, **base 2** from mathematics.

For example,

$$17_{10} = \mathbf{\texttt{10001}}_2$$

$$42_{10} = \mathbf{\texttt{101010}}_2$$

$$1000_{10} = \mathbf{\texttt{1111101000}}_2$$

The subscripts indicate the base.

But computers have no "blank" bits!

The Unsigned Representation: Base 2 with Leading 0s

Use leading 0s to fix the number of bits (to **N**).

Result: the **N-bit unsigned representation**.

Using the 8-bit unsigned representation,

$17_{10} =$ `00010001`

$42_{10} =$ `00101010`

$1000_{10} =$ Cannot be represented!

What Can the Unsigned Representation Represent?

What range of integers can be represented with the ***N*-bit unsigned representation**?

- smallest value… all 0s
- largest value … all 1s

Note that $\mathtt{100...000}_2$ (**N** 0s after a 1) is $\mathbf{2^N}$.

The range is thus $\mathbf{[0, 2^N - 1]}$.

Use a Polynomial to Convert to Decimal

How can we **calculate the decimal number represented by a bit pattern** in an unsigned representation?

Remember the place values.

Let's name the bits of the bit pattern:

$$\mathbf{a_5\ a_4\ a_3\ a_2\ a_1\ a_0}$$

Multiply each bit by its place value, then sum:

$$a_5 32 + a_4 16 + a_3 8 + a_2 4 + a_1 2 + a_0 1$$

$$= a_5 2^5 + a_4 2^4 + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0$$

What about Converting from Decimal?

What about finding the bit pattern that represents a decimal number **D** using an unsigned representation?

Seem harder?

Again, name our bits $\mathbf{a_i}$.

In the unsigned representation, every bit pattern represents a different number.

Thus the $\mathbf{a_i}$ that represent **D** are unique.

Use the Same Polynomial to Convert from Decimal

The decimal number is given by

$$\mathbf{D = a_5 2^5 + a_4 2^4 + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0}$$

All terms in the sum except for the last are even (they are multiples of 2).

So, if **D** is odd, $\mathbf{a_0 = 1}$.

And if **D** is even, $\mathbf{a_0 = 0}$.

We subtract out $\mathbf{a_0}$, divide by 2, and use the same reasoning until we run out of digits.

Example: the Unsigned Bit Pattern for D = 37.

$\mathbf{37 = a_5 2^5 + a_4 2^4 + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0}$

37 is odd, so $\mathbf{a_0 = 1}$.

$\mathbf{(37–1)/2 = (a_5 2^5 + a_4 2^4 + a_3 2^3 + a_2 2^2 + a_1 2^1)/2}$

$\mathbf{18 = a_5 2^4 + a_4 2^3 + a_3 2^2 + a_2 2^1 + a_1 2^0}$

18 is even, so $\mathbf{a_1 = 0}$.

$\mathbf{(18 – 0)/2 = (a_5 2^4 + a_4 2^3 + a_3 2^2 + a_2 2^1)/2}$

$\mathbf{9 = a_5 2^3 + a_4 2^2 + a_3 2^1 + a_2 2^0}$

Example: the Unsigned Bit Pattern for D = 37.

$\mathbf{9 = a_5 2^3 + a_4 2^2 + a_3 2^1 + a_2 2^0}$

9 is odd, so $\mathbf{a_2 = 1}$.

$\mathbf{(9 - 1)/2 = (a_5 2^3 + a_4 2^2 + a_3 2^1)/2}$

$\mathbf{4 = a_5 2^2 + a_4 2^1 + a_3 2^0}$

4 is even, so $\mathbf{a_3 = 0}$.

$\mathbf{(4 - 0)/2 = (a_5 2^2 + a_4 2^1)/2}$

$\mathbf{2 = a_5 2^1 + a_4 2^0}$

Example: the Unsigned Bit Pattern for D = 37.

$\mathbf{2 = a_5 2^1 + a_4 2^0}$

2 is even, so $\mathbf{a_4 = 0}$.

$\mathbf{(2 - 0)/2 = (a_5 2^2)/2}$

$\mathbf{1 = a_5 2^0}$

Putting the bits together, we obtain

$$37_{10} = \mathtt{100101}$$

Note: be sure to put the bits in the right order!

Example: the Unsigned Bit Pattern for D = 137.

We don't need to write the polynomial…

137 (odd)		$\rightarrow$ 1
(137 – 1) / 2	= 68	$\rightarrow$ 0
(68 – 0) / 2	= 34	$\rightarrow$ 0
(34 – 0) / 2	= 17	$\rightarrow$ 1
(17 – 1) / 2	= 8	$\rightarrow$ 0
(8 – 0) / 2	= 4	$\rightarrow$ 0
(4 – 0) / 2	= 2	$\rightarrow$ 0
(2 – 0) / 2	= 1	$\rightarrow$ 1
(1 – 1) / 2	= 0	(done)

137_{10} = **10001001**

Read the bits from bottom to top (and add leading 0s if needed).