Logistics

- Discord: 95/128 students
- Go to OH and check-in with the TA. This is graded. Deadline: 1/27/25
  - 29/128 students
- Pre-project: check the videos posted on Discord -> Announcements
  submissions: 48/128 students
  - submit at least "Task 1" to get an extension.
- Zybooks: check the instructions on the website
  - If you cannot access the textbook yet, please email me
  - Average Score 84%
  - 119/128 students

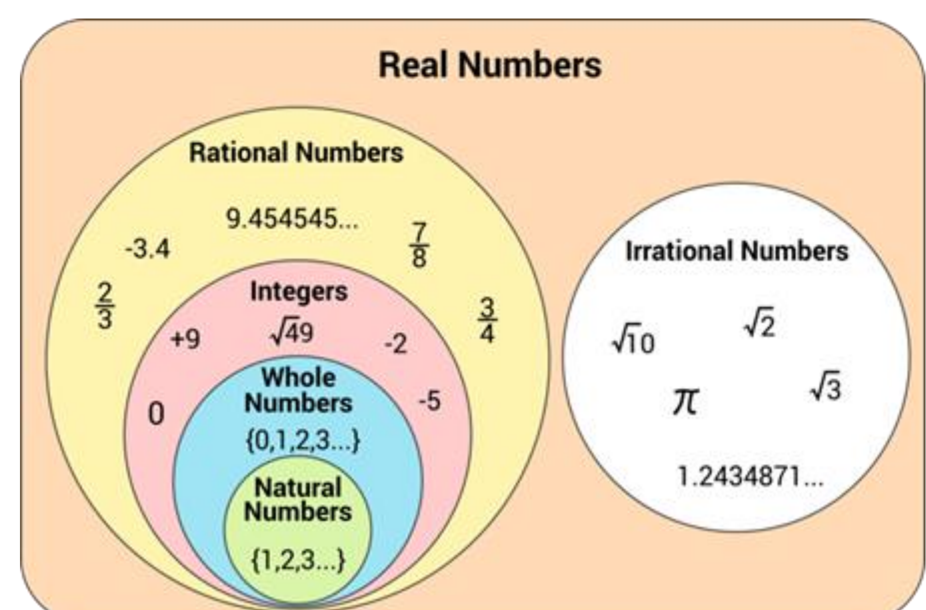
Recap

- What is a computer?
- The goal of this class is to understand how the hardware works
- Everything in the computer is ones and zeros (binary)
- We can add two binary numbers the same way we add two decimal numbers

Let’s store a number in the computer

Which numbers?

1. Write whole numbers
2. Write integer numbers
3. Write rational numbers



You can see references in the notes

write in Java a line of code to declare a variable var1 which can contain only a whole number and assign a value to it.

unsigned int var1 = 1;

Natural numbers are counting numbers starting at 1.
Whole numbers are natural numbers together with 0.

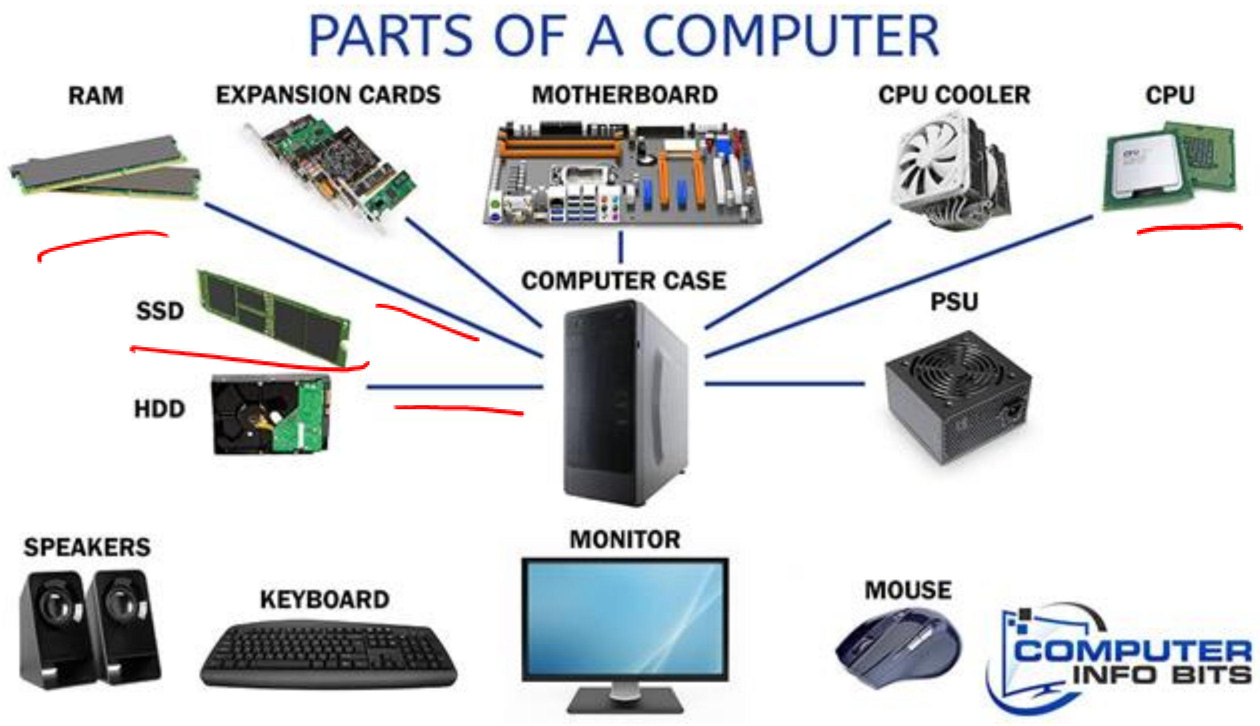
write in Java a line of code to declare a variable var1 which can contain only a whole number and assign a value to it.

```
(int var1 = 1;)
```

Natural numbers are counting numbers starting at 1.
Whole numbers are natural numbers together with 0.

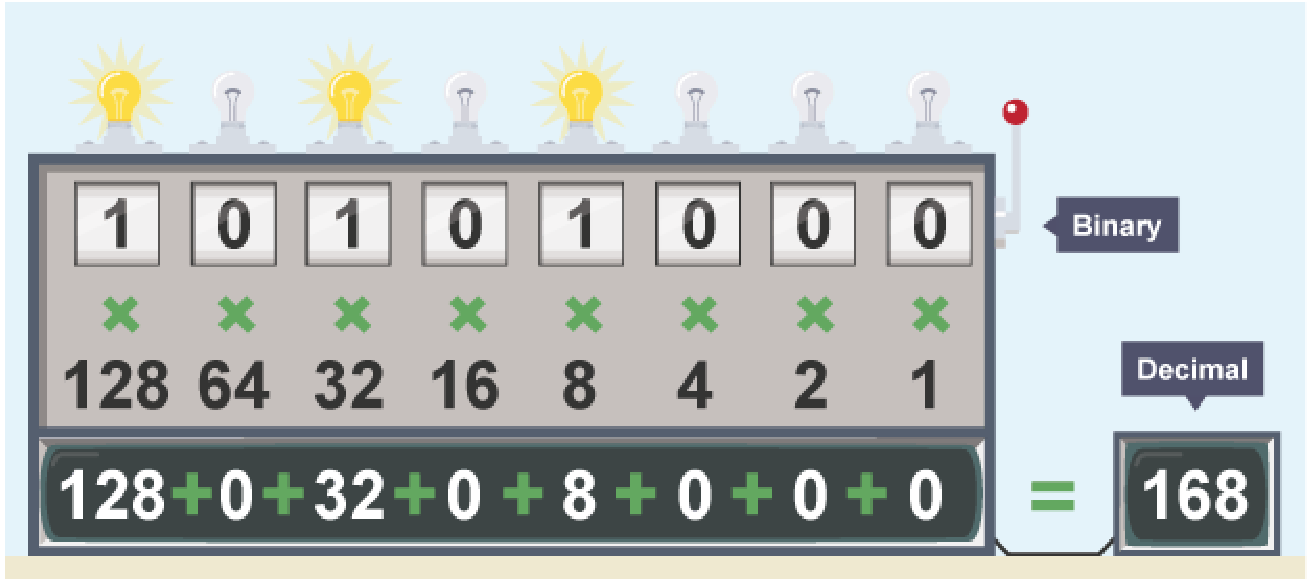
Where is this store in the computer?

- Memory
- Disks
- CPU



Where exactly?

- BITS = wire
- Number of bits is always **finite**



Binary

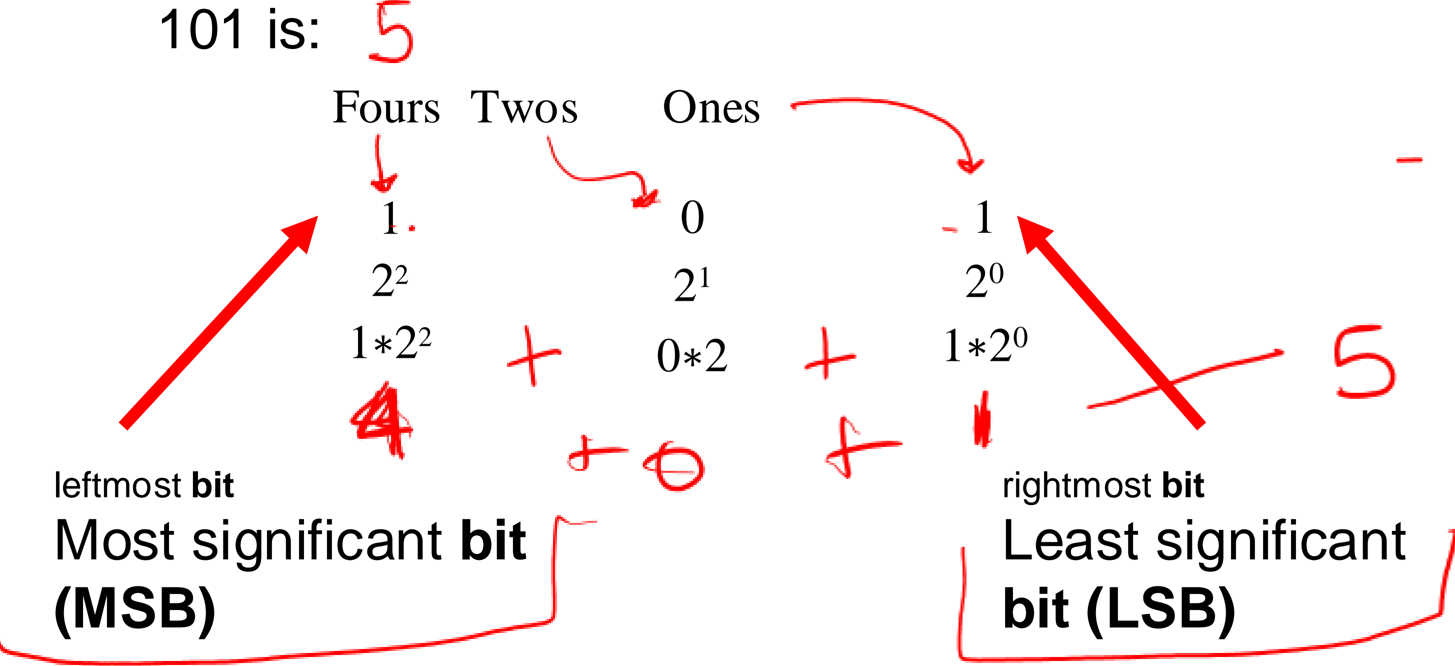
in Math:

infinite numbe

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

Binary System

- Base-2 uses powers of 2 for each column



You need to know how to convert from Binary to decimal and decimal to binary

- 93% of the class already know how to do this.
- Supplementary material is provided

In A Computer

- *bit* — A binary digit (one wire)
- *nibble* — 4 bits
- *byte* — 8 bits
- *half-word* — *16 bits (*defined by the CPU)
- *word* — *32 bits (*defined by the CPU)

Word – Depends on the processor.

- Intel 4004 – 4-bit words
- Intel 8008 – 8-bit words
- Intel 8086, Intel 80286 – 16-bit words
- Intel 80386, Intel 80486 – 32-bit words
- Intel Pentium – 64-bit bus, 32-bit words
- Intel Core 2 Duo, Intel Core i3/i5/i7 – 64-bit words
- Graphics Processors – 128-bit words

- In general, a larger word size is faster, but more expensive to implement.

How Do We Store Numbers?

- Stored using a **finite, fixed** number of bits (eg. 64-bits)
- Pad extra digits with leading 0s
- A **byte** representing 4 = **00000**100

A Byte

- Binary: Each column is worth a power-of-2

128 's	64 's	32 's	16 's	8 's	4 's	2 's	1 's
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	1

128 + 0 + 32 + 0 + 8 + 0 + 2 + 1 = 171

How many dogs do you see in this picture?

write your answer in **a byte**

$6 - 4 = 2$

$2^3 \quad 2^2 \quad 2^1 \quad 2^0$

8 4 2 1

0 0 0 0 0 1 1 0

What is the largest number we can write in a byte?

Write your answer in binary

1111 11111

What is the largest number we can write in a byte?

in decimal

7-bit word

$2^k - 1$

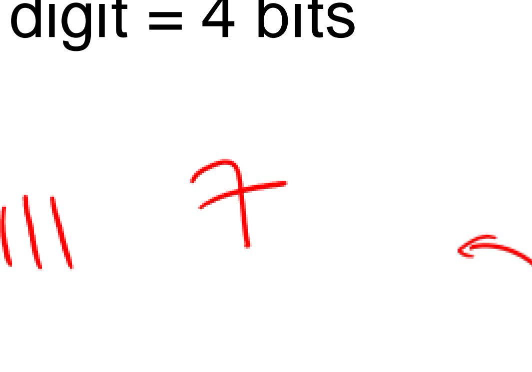
1111111
+ 1
10000000 → $2^7 - 1$

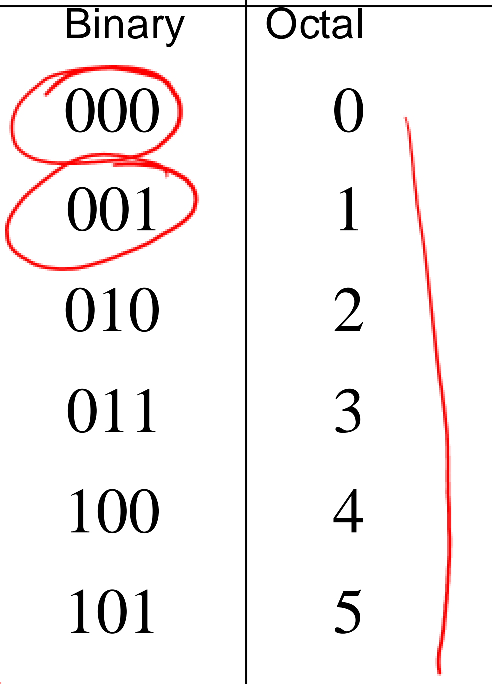
Reading and writing binary is difficult for humans

Hex and Octal

- Hexadecimal (base 16) and octal (base 8) are

- 1 octal digit = 3 bits
- 1 hex digit = 4 bits





Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6

Converting to/from Octal

Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

- Group bits into packs of 3 (**starting at the right**)
- Convert each group to octal

Convert to Octal:

110111_2 67₈

10000000_2 → 200_8 (byte)

1010101011100111_2 (half-word)

Converting to/from Octal

- Group bits into packs of 3 (**starting at the right**)
- Convert each group to octal

Convert to Octal:	
110111$_2$	67$_8$
10**000**000$_2$	200$_8$
1010**101**011**100**111$_2$	125347$_8$

- Unfortunately, octal digits don't line up with bytes!

Hexadecimal

- Base 16: digits 0-9, then a-f (or A-F)

Binary	Hex	Binary	Hex
0000	0	1000	8
0001	1	1001	9
0010	2	1010	a
0011	3	1011	b
0100	4	1100	c
0101	5	1101	d
0110	6	1110	e

Hexadecimal to Binary

1. Expand each hexadecimal digit into the corresponding 4 binary digits:
   - 1 2 3 4 A F 0 C
   - 0001 0010 0011 0100 1010 1111 0000 1100

Binary to Hexadecimal

1. Group 4 binary digits from right to left and convert every nibble to hexadecimal

   – 0001 0010 0011 0100 1010 1111 0000 1100
   – 1 2 3 4 A F 0 C

Why Hexadecimal?

- Dense, easy to type &
- read

  2 hex digits = 1 byte

  1 nibble

- In in many languages, hex constants start with `0x`

```
  Example (Java):
System.out.println(String.format("0x%08X", 1));
```

Why Hexadecimal?

- Dense, easy to type &
- read

  2 hex digits = 1 byte
- Great constants

  `0xdeadbeef`

  `0xbaadf00d`

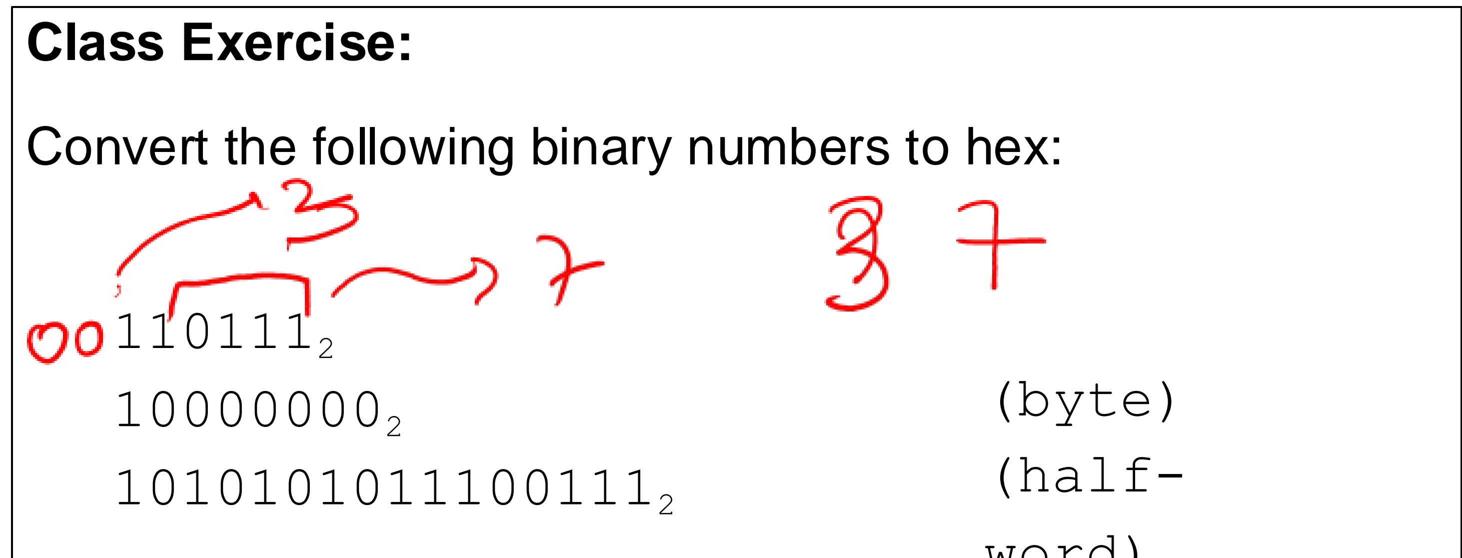
  `0x0000000ff1ce` (MS Office)

  `0xc00010ff` (iOS error - overheat)

  `face:b00c` (part of an IPv6 addr)

Decimal to/from Octal, Hex

- Possible to go direct, but annoying
- Easier: go to binary, then to octal/hex



Gradescope Question 2.

Class Exercise:

Convert the following binary numbers to hex:

110111_2

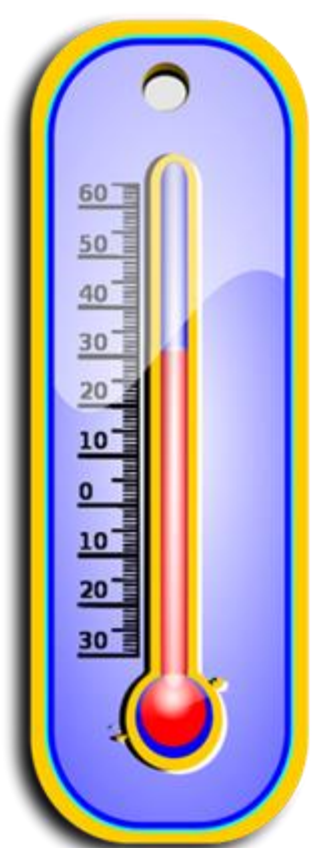
10000000_2 (byte)

1010101011100111_2 (half-word)

Temperature check

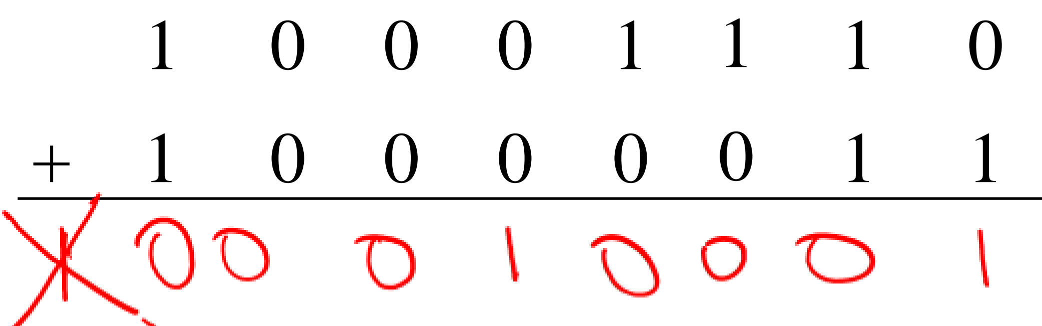
How are you feeling?

A. Very confused
B. Need a lot more practice
C. Need a little more practice
D. Just have a couple of questions
E. Feeling good



Let’s sum two whole numbers in the computer

Add these two bytes



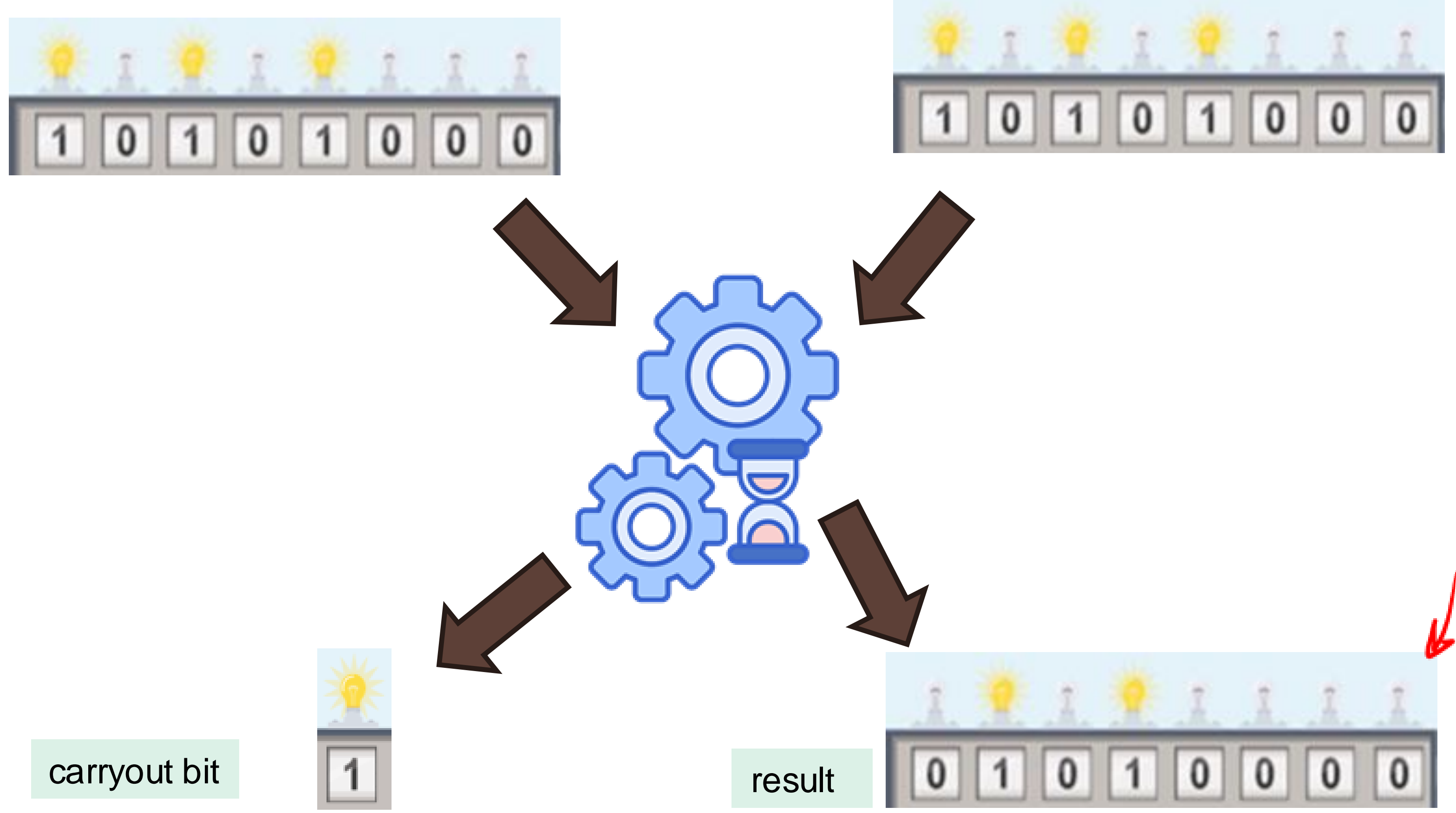
On whiteboard:

- Add using binary (watch for carries)
- The answer should be 1 byte

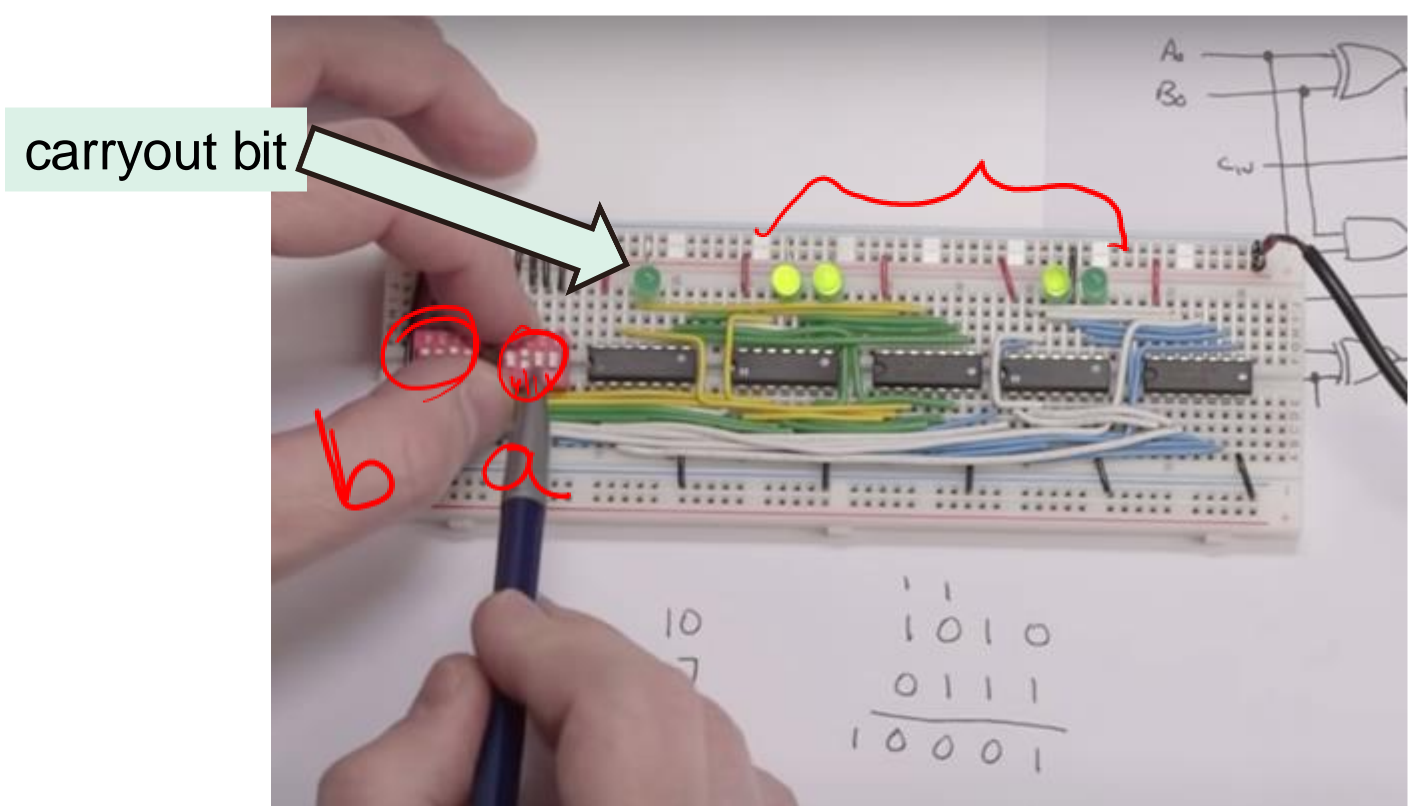
Add these two bytes - is this the correct answer in decimal?

	1	0	0	0	0	0	0	0
+	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0

Intro to the carryout bit

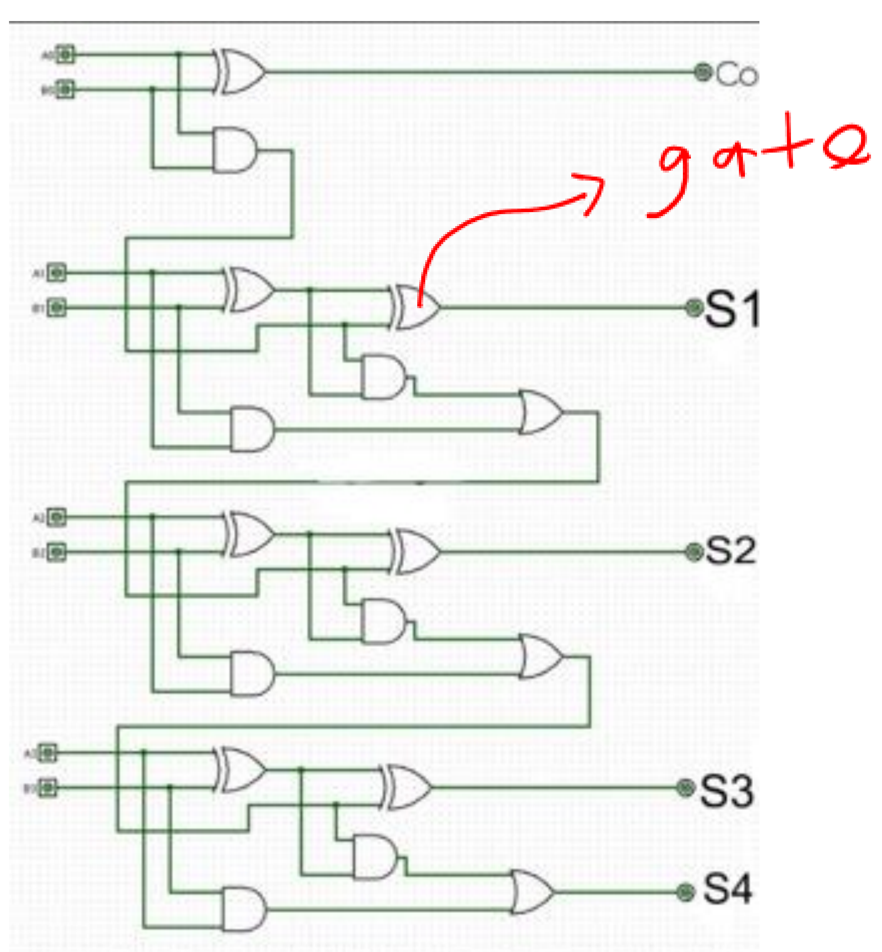


You could build your own 4-bit adder

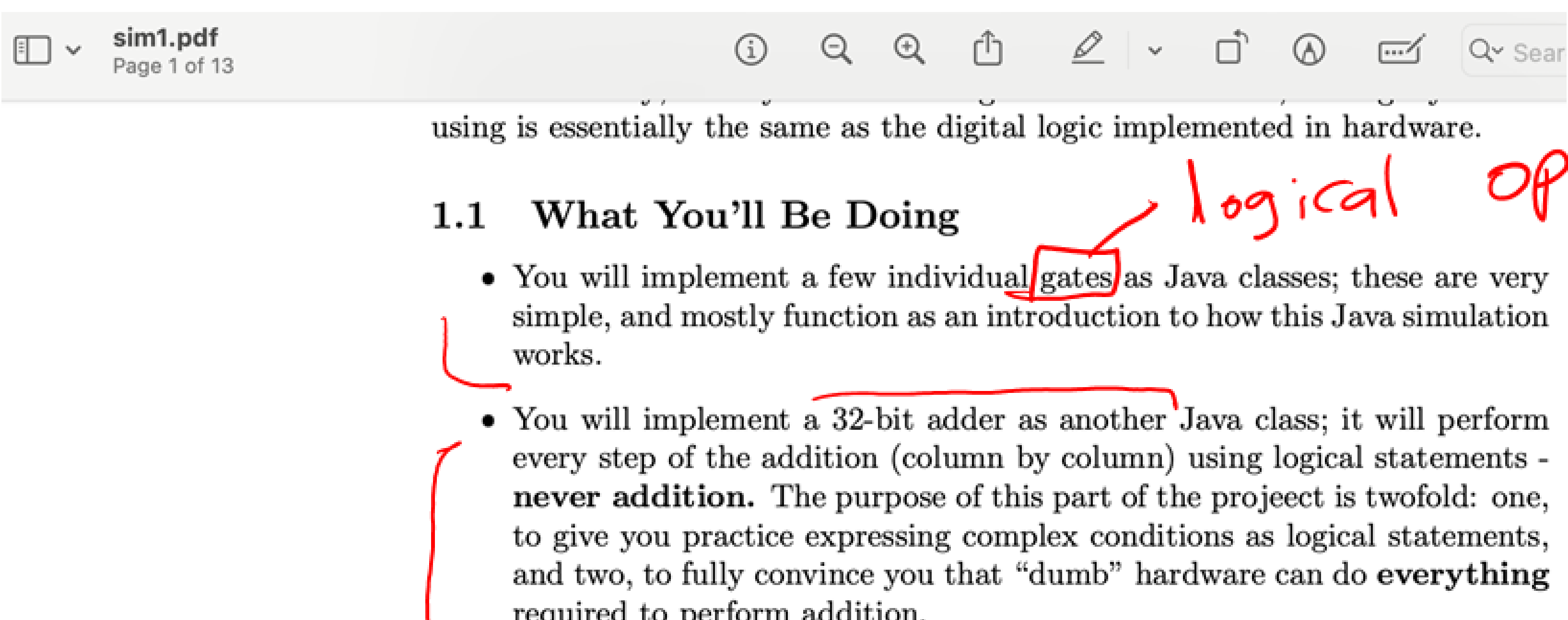


https://www.youtube.com/watch?v=wvJc9CZcvBc

Circuit Schematic



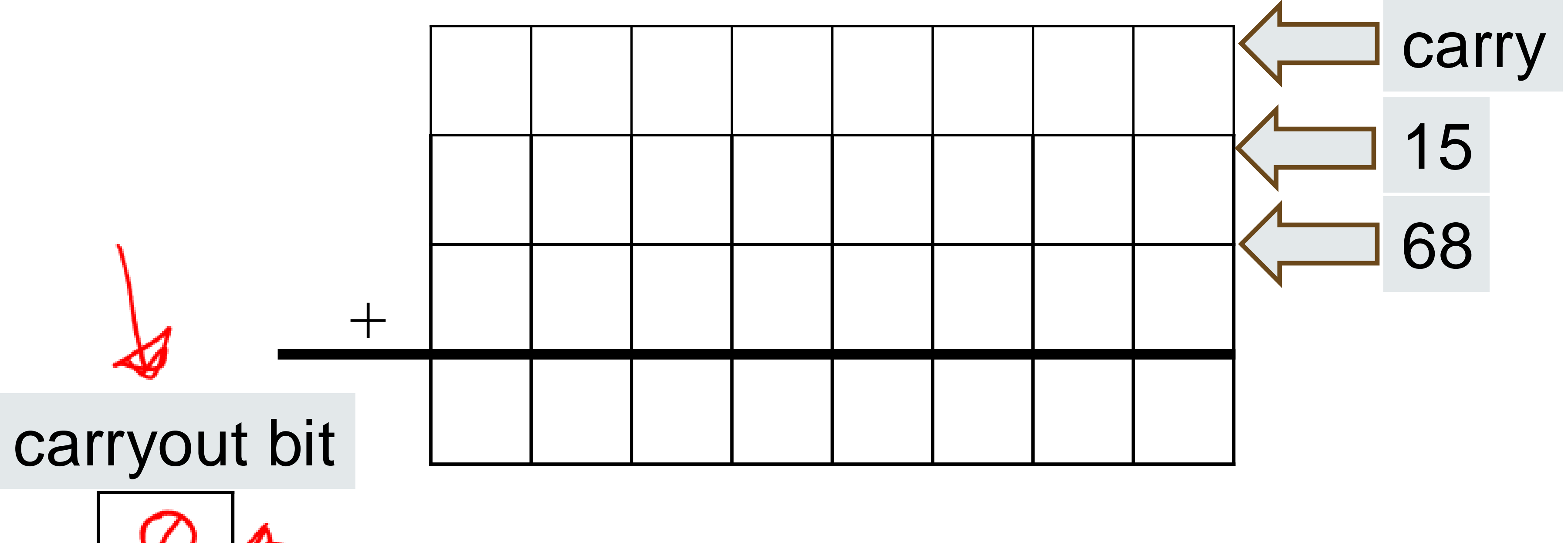
Simulation Project 1



Gradescope Question 3

Show how an 8-bit circuit adds the numbers 15 + 68

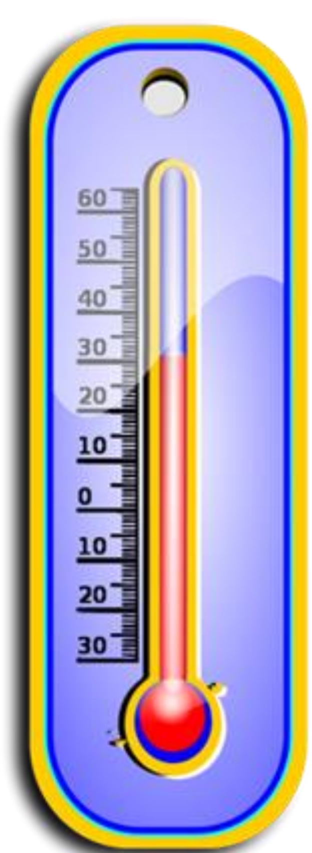
Fill in the blanks:



Temperature check

How are you feeling?

A. Very confused
B. Need a lot more practice
C. Need a little more practice
D. Just have a couple of questions
E. Feeling good

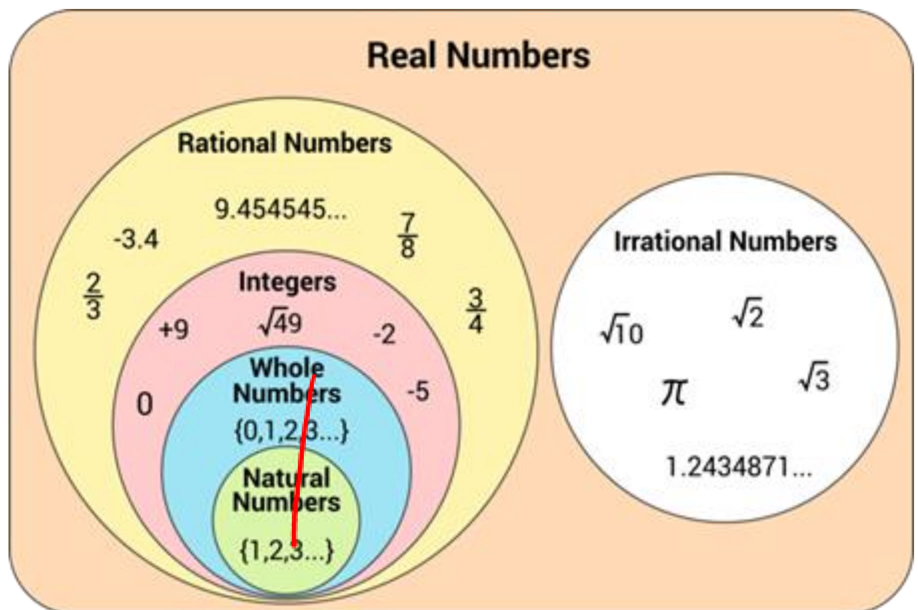


Let’s store an INTEGER number in the computer

Which numbers?

1. ~~Write whole numbers~~ ✓
2. Write integer numbers
3. Write rational numbers

Integers are all whole numbers and their opposites (negatives), as well as 0.



write in Java a line of code to declare a variable var1 which can contain only an integer number and assign a value to it.

```
int var1 = 5;
```

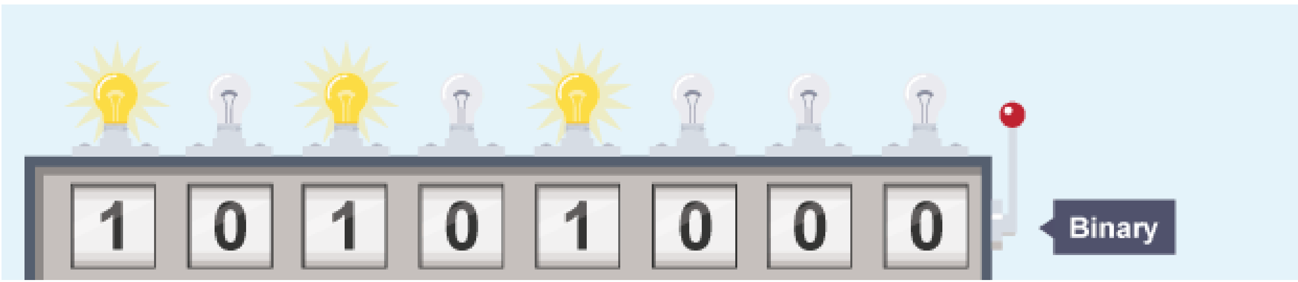
Integers are all whole numbers and their opposites (negatives), as well as 0.

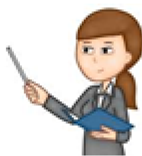
Signed vs. Unsigned

- Some variables are naturally unsigned
  - Memory addresses
  - Some counters
- Some variables are naturally signed
  - Counters that can go negative
  - Physical properties
  - Differences between addresses

Brainstorm

If you were in charge of designing the representation of negative numbers using only zeros and ones, what would you do?





Some ways to store negative numbers

- Sign and magnitude
- Biased number
- One’s complement
- Two’s complement

Sign Bit **(FAIL)**

- Idea: Use the most left bit for sign (1=negative)

0000 0000	0
0000 0001	1
1000 0001	-1

- Problems: 1 + -1 != 0
  - Separate hardware for signed vs. unsigned
  - Separate hardware for add & subtract
  - Negative zero ???? 1000 0000

1's Complement **(FAIL)**

- Idea: Use the top bit for sign (1=negative)

```
0000 0000        0
0000 0001        1
1111 1110       -1
```

- Problems:
  - The same ones!

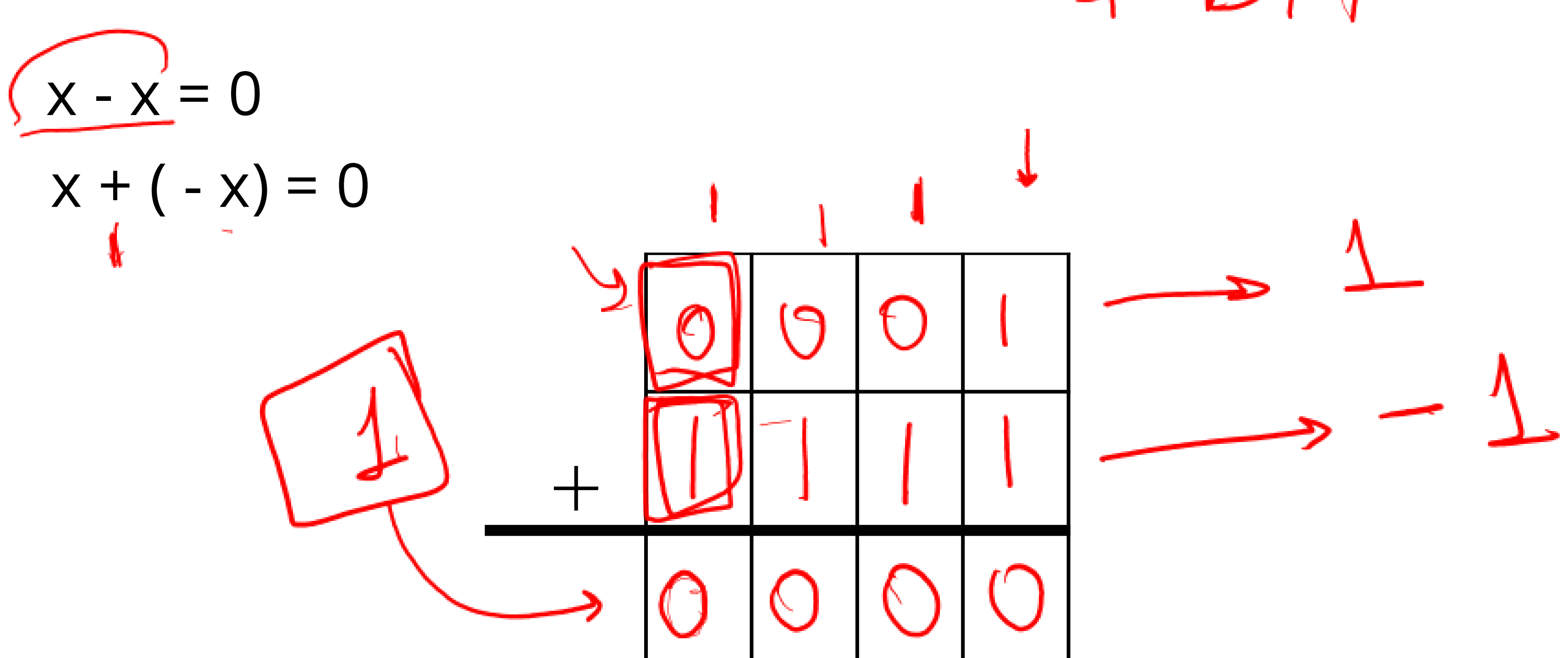
2's Complement

(SUCCESS!)

Design to be a real complement system:

$x - x = 0$

$x + (-x) = 0$

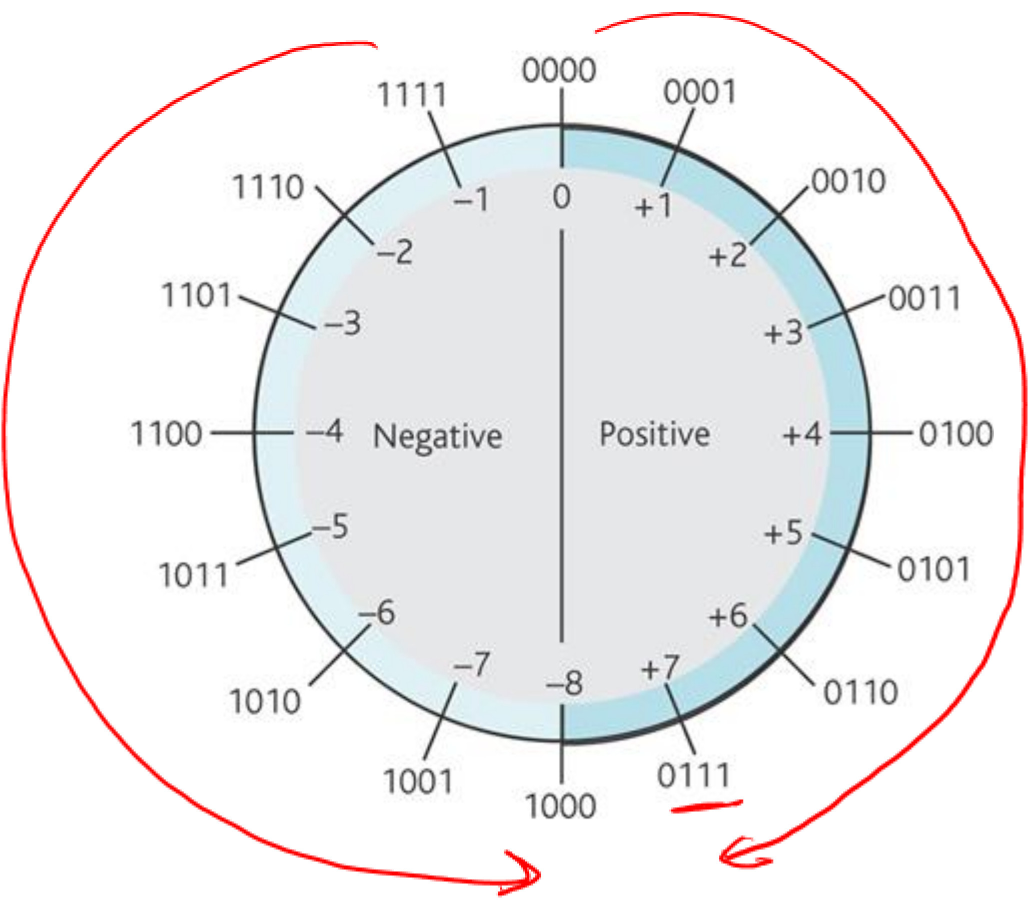


2's Complement advantages

- Use the most left bit also can tell the sign (1=negative)
- We do not need to build a circuit for subtraction
  - x + (-x) = x - x
- Only one zero (no more negative zero)

When we declare and integer variable, the computer uses two’s complement representation

4-bit Two's Complement

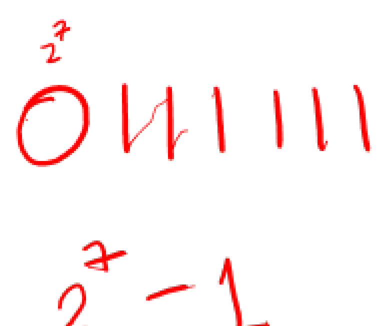


4-bit two's compleme

Binary Number	Unsigned Value	Signed Value
0000	0	0
0001	1	1
0010	2	2
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

we can write with 7 bits using 2's complement?

write your answer in binary first and then in decimal



Steps to write a **positive** decimal number in 2's complement 4-bit

1. Convert the decimal number to binary =
2. Pad with zeros to the left
3. Did the number fit? In other words, it is a positive number? MSB = 0

4-bit integer.

Write the binary representation of **9** in a 8-bit integer.

Steps to write a **negative** decimal number in 2's complement

1. Convert the decimal number to binary
2. Pad with zeros to the left
3. Invert the bits (bitwise negation)
4. Add 1
5. Did the number fit? In other words, it is a negative number? MSB = 1

-5

Example: write **-3** in 4-bit 2's complement

Base 2:

Complement: invert bits, add 1.

0011 (3) has complement: $(0011)' + 1$

$1100 + 1$

1101

Write the binary representation of -**9** in a
4-bit integer.

Write the binary representation of -**9** in a
8-bit integer.

write with 7 bits using 2’s complement?

write your answer in binary first and then in decimal

2's Complement

- Four magic values (no matter how many bits)

$$0000_00 \ldots 00 = 0$$

$$1111_11 \ldots 11 = -1$$

$$0111_11 \ldots 11 = 2^k - 1$$ (max value)

$$1000_00 \ldots 00 = (-1) * 2^k$$ (min value)

Memorize these!

A Note on Terminology

- 2's Complement is both a format an a process
  - “Write the n-bit 2's complement encoding of X” means write X, using the 2's complement format
  - “Take the n-bit 2's complement of X” means to **negate** X using 2's complement (next slide)

This distinction will probably be important on the exam!

Taking the 2's Complement

How to negate a number using 2's complement?

1. Invert the bits (bitwise negation)
2. Add 1

Negating Integers

$-n = n * (-1)$

- To negate any number (times -1)
  1. flip all the bits
  2. add 1

Let's say the following number is an 8-bit integer.

Is this a positive or negative number?

Take the two's complement of this number

Leave your answer in binary

Sign Extension

- We often want to convert from a small format to a larger one.
  - This is trivial for positive numbers. All of these numbers are equal to 6_{ten} :

8-bit							0000	0110
16-bit					0000	0000	0000	0110
32-bit	0000	0000	0000	0000	0000	0000	0000	0110

Sign Extension

- Use **sign extension** (copy the 1)
  - All of these numbers are equal to -8_{ten}:

8-bit							1111	1000
16-bit					1111	1111	1111	1000
32-bit	1111	1111	1111	1111	1111	1111	1111	1000

- This works because of the "subtract from -1" trick

Sign Extension

- Generally, to convert a **signed integer** to a larger signed integer, simply copy the sign bit (MSB) into all of the new bits
  - Positive numbers get more zeros
  - Negative numbers get more ones
- This is called **sign extension**

What is the output?

```c
#include <stdio.h>

int main()
{
    int x = 1;
    printf("%d", x);
    return 0;
}
```

Why?

```
#include <stdio.h>

int main()
{
    int x = -1;
    printf("%#x",x);

    return 0;
}
```



```
0xffffffff
```

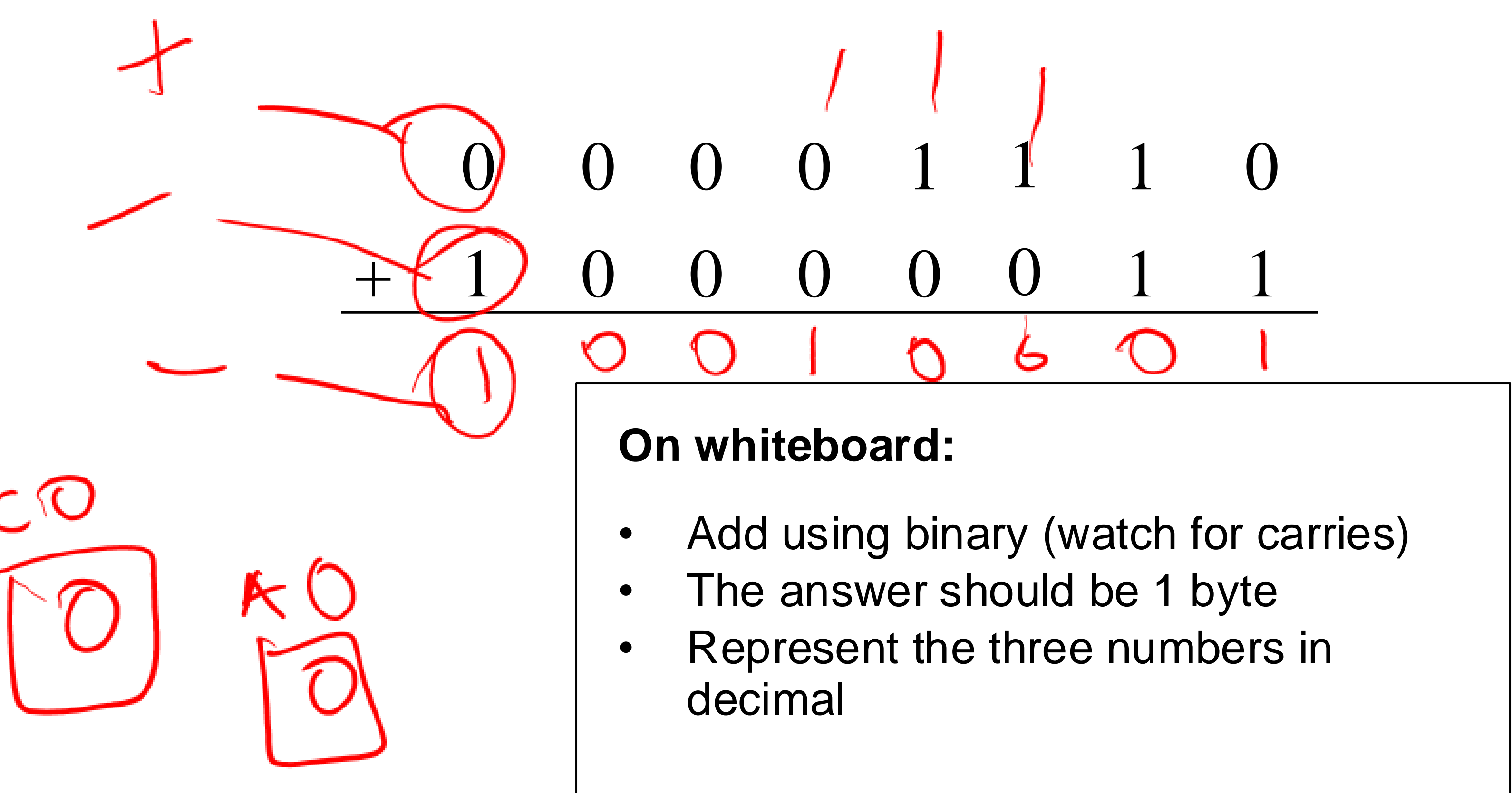
Gradescope Question 4

Write the 7-bit 2's complement encoding of the following numbers. If the number does not fit, explain the range that can be written with 7 bits:

- 0
- 26
- -128
- -15
- 236

Let’s sum INTEGER numbers in the computer

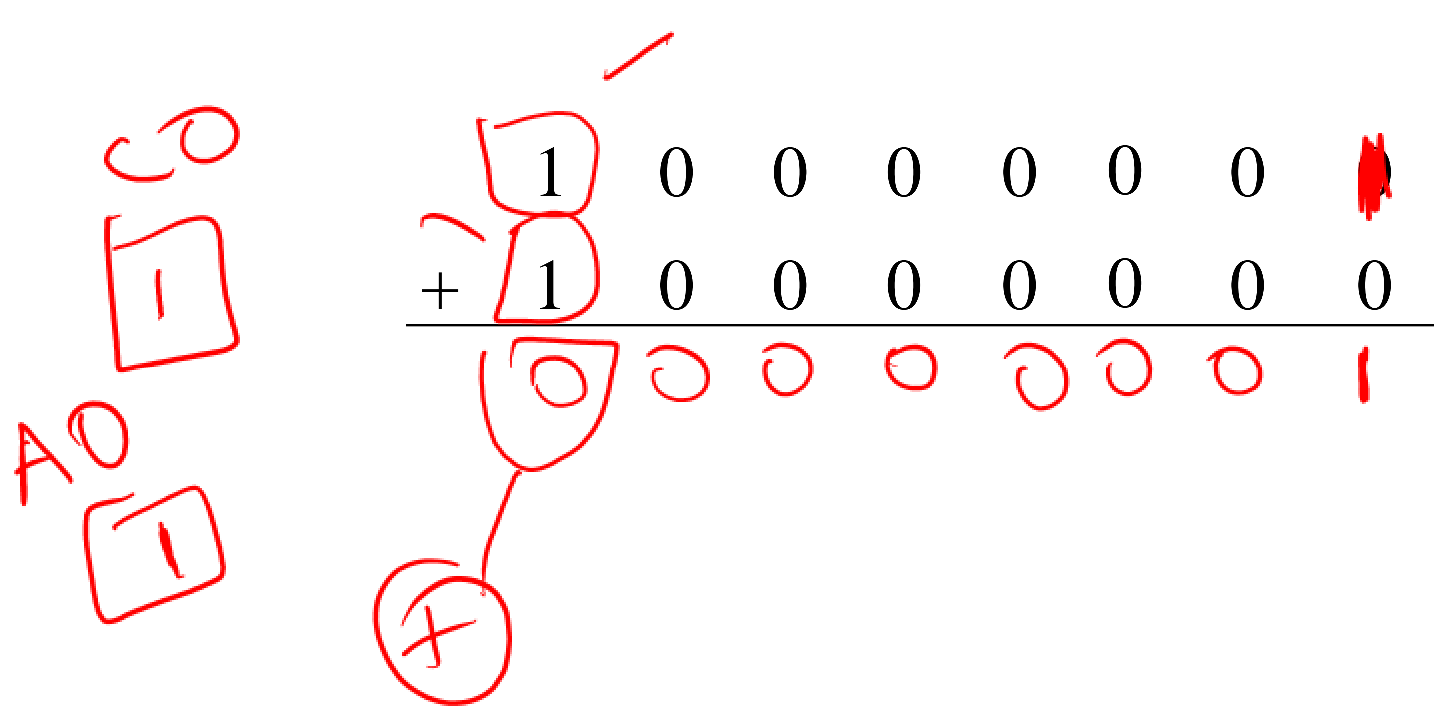
Add these two 8-bit integers



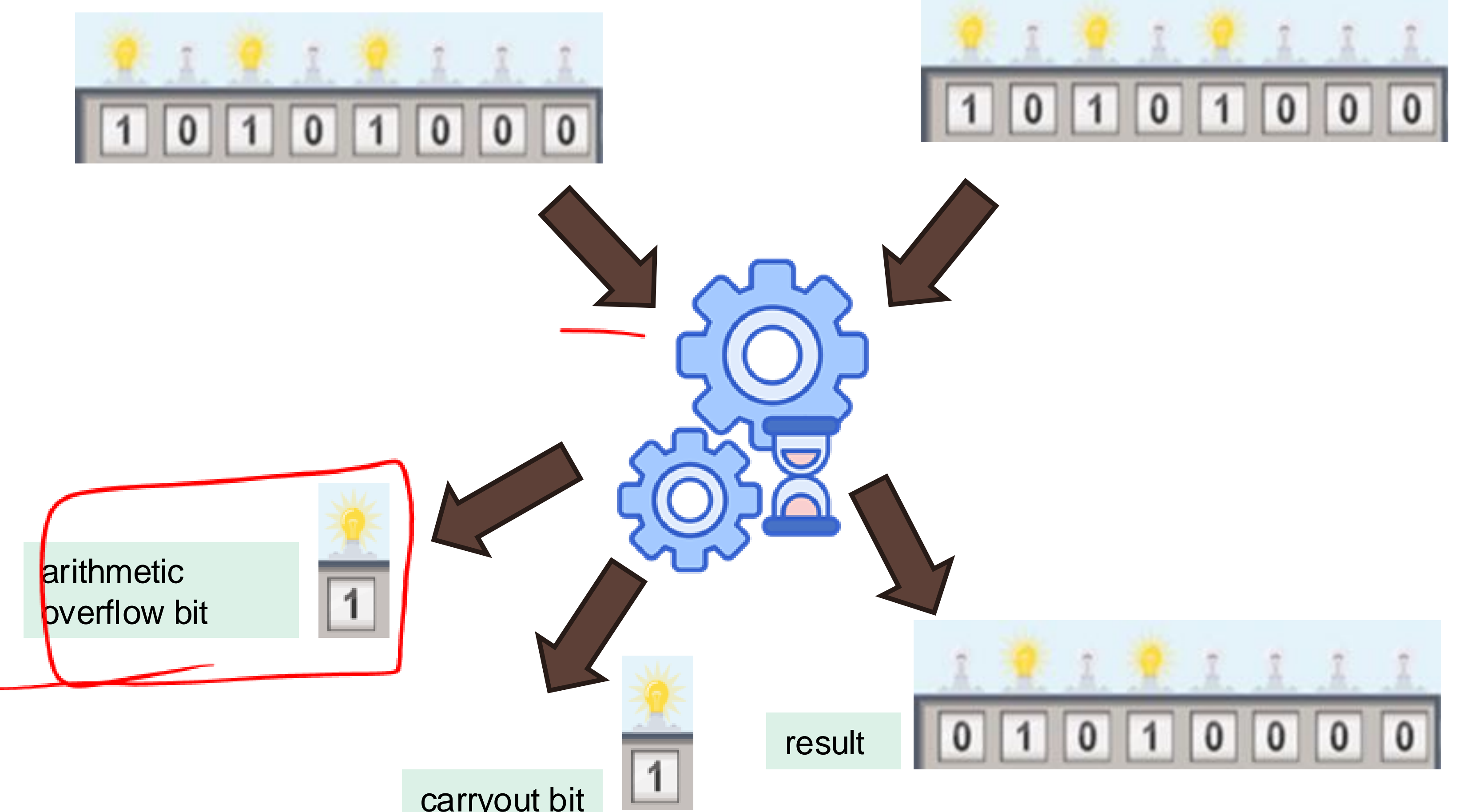
On whiteboard:

- Add using binary (watch for carries)
- The answer should be 1 byte
- Represent the three numbers in decimal

Add these two integers - is this the correct answer in decimal?



Intro to the arithmetic overflow bit



Arithmetic Overflow

Adding two positives
May overflow

0010 + 0011 = 0101 (ok)
2 + 3 = 5

0111 + 0001 = 1000 (overflow)
7 + 1 ≠ -8

Adding two negatives
May overflow

1111 + 1111 = (1)1110 (ok)
-1 + -1 = -2

1000 + 1111 = (1)0111 (overflow)
-8 + -1 ≠ 7

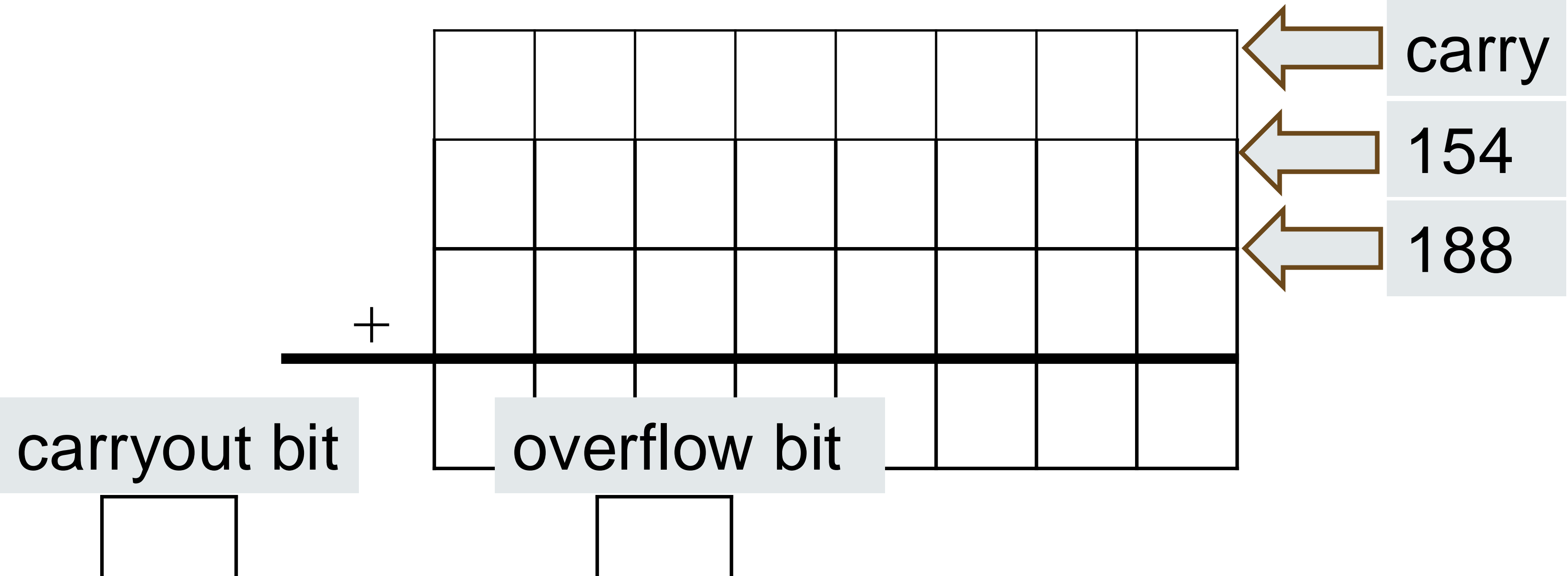
Adding positive and negative
Cannot overflow

1000 + 0111 = 1111 (ok)
-8 + 7 = -1

0010 + 1000 = 1010 (ok)
2 + -8 = -6

Gradescope Question 5

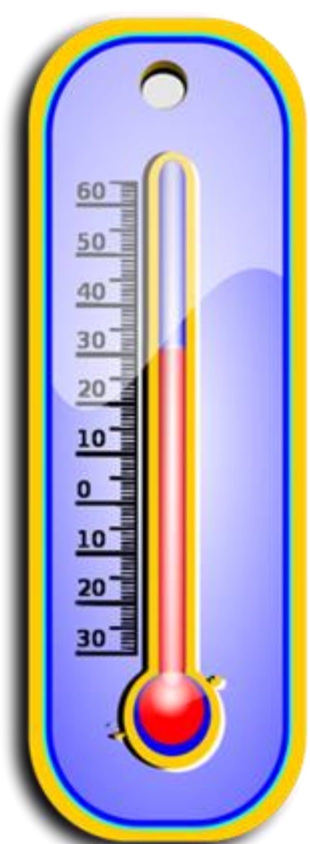
Show how a computer adds adds the 8-bit integers 154 + 188
Fill in the blanks:



Temperature check

How are you feeling?

A. Very confused
B. Need a lot more practice
C. Need a little more practice
D. Just have a couple of questions
E. Feeling good



Recap

Unsigned Numbers

- Some types of numbers, such as memory addresses, will <u>never be negative</u>
- Some programming languages reflect this with types such as “unsigned int”, which only hold positive numbers.

Steps to write an integer number in n-bit 2's complement

1. Does the number fit?
2. Convert the decimal number to binary
3. Pad with zeros to the left
4. If the number is negative
   a. Invert the bits
   b. Add 1

n-bit Two's Complement Properties

- Two's complement is a true complement system.

  $$X + (-X) = 0.$$

- There is one unique zero: a string of all zeroes.
  - eg. for n=8, 00000000 is 0
- For n bits, the range is from $-(2^{n-1})$ to $2^{n-1} - 1$.
  - eg, for n=8, we can represent values from -128 to +127
  - 10000000 is -128, the smallest number that can be represented

n-bit Two's Complement properties

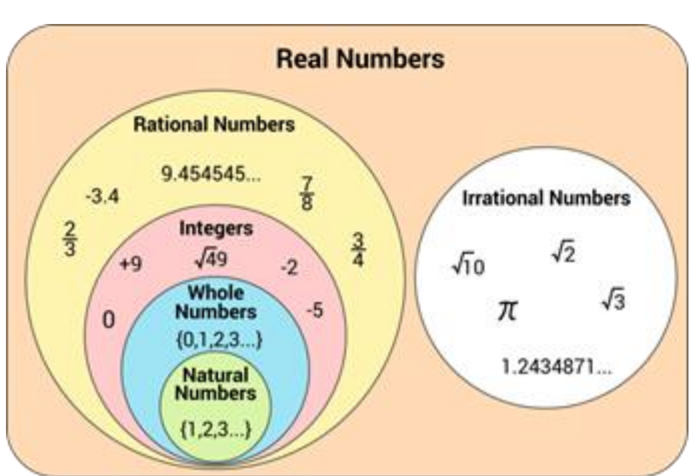
- The most significant bit holds the sign: a 0 means positive and a 1 means negative.
- For an n-bit integer N, we have '$-N$' => $2^n - N$. Let $n = 8$
  - $N = 5 = 00000101$
  - $2^8 - 5 = 100000000 - 00000101 = 11111011$, which is -5

Terminology!

- Bit – The smallest unit of storage; has a value of 0 or 1.
- Nibble – The logical unit of 4 bits form a nibble.
- Byte – The logical unit of 8 bits form a byte.
- Word – Depends on the processor.
  - Intel 4004 – 4-bit words
  - Intel 8008 – 8-bit words
  - Intel 8086, Intel 80286 – 16-bit words
  - Intel 80386, Intel 80486 – 32-bit words
  - Intel Pentium – 64-bit bus, 32-bit words
  - Intel Core 2 Duo, Intel Core i3/i5/i7 – 64-bit words
  - Graphics Processors – 128-bit words
- Note: In general, a larger word size is faster, but more expensive to implement.

Steps to write an **integer** number in n-bit 2’s complement

1. Does the number fit?
2. If the number is negative, ignore the negative sign for now. Convert the decimal number to binary
3. Pad with zeros to the left
4. If the number is negative
   a. flip the bits
   b. sum 1 (+1)



Steps to write a binary **integer** (n-bit 2’s complement) as a decimal number

1. If the number is negative
   a. flip the bits
   b. add 1
2. Convert the binary number to decimal
3. If the number was negative affix the sign

Steps to write an integer number in n-bit 2’s complement

1. Does the number fit?
2. Convert the decimal number to binary
3. Pad with zeros to the left
4. If the number is negative
    a. Invert the bits
    b. Add 1

Steps to write an n-bit 2's complement number in decimal form

1. Is the number positive or negative?
2. Positive:
   a. Convert the binary number to decimal

2. Negative:
   a. Invert the bits
   b. Add 1
   c. Convert the binary number to decimal
   d. Afix the sign

Subtraction

How does a computer compute 9 - 6 ?

EXIT POLL