

Welcome! *Do now:*

1. Finish readings before you continue (1.0-1.1, 1.4, 4.0-4.2)
2. Register your iClicker:
forms.gle/uDx8Ejc7DWdh5bdd7

CS31: Introduction to Computer Systems

Week 2, Class 1
Data Representation
01/30/24

Dr. Sukrit Venkatagiri
Swarthmore College



Announcements

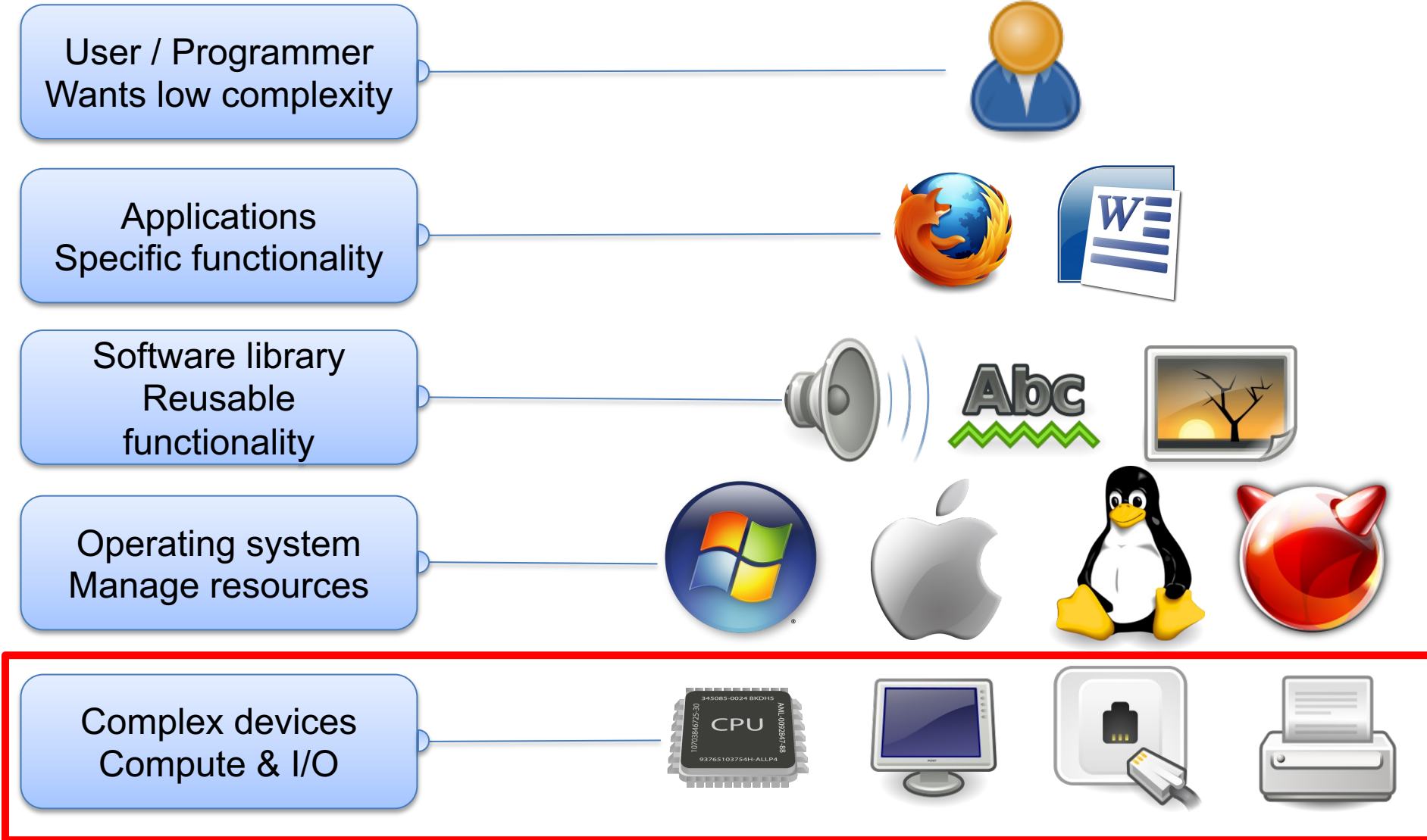
- Lab 1 is due Thursday, 11:59pm
- HW1 is due Friday, 11:59pm
 - up to groups of four
 - invitations sent from gradescope
- Clickers will count for credit from Tuesday, February 6th

Agenda

Data representation

- number systems + conversion
- data types, storage
- sizes, representation
- signedness

Abstraction



Data Storage

- Lots of technologies out there:
 - Magnetic (hard drive, floppy disk)
 - Optical (CD / DVD / Blu-Ray)
 - Electronic (RAM, registers, ...)
- Focus on electronic for now
 - We'll see (and build) digital circuits soon
- Relatively easy to differentiate two states
 - Voltage present
 - Voltage absent

Bits and Bytes

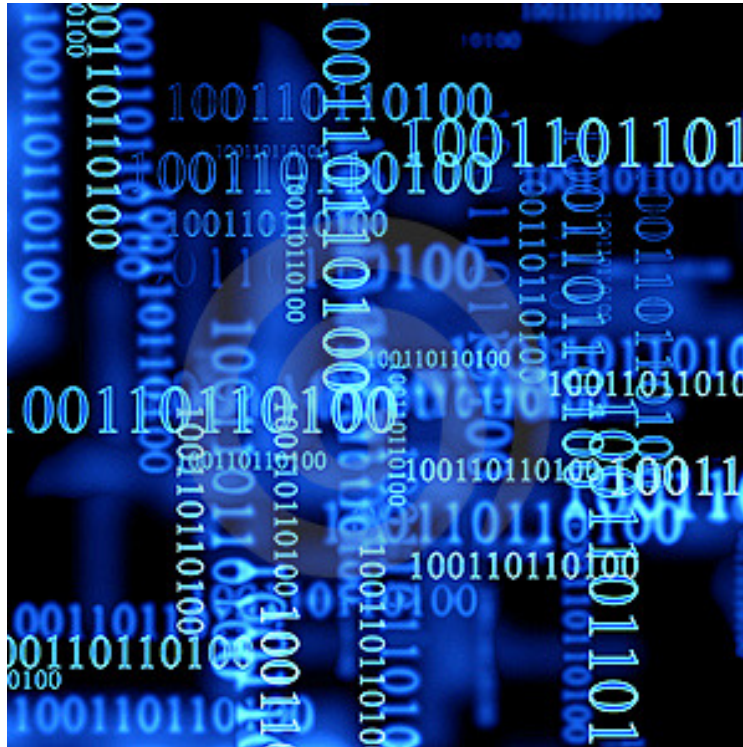
- Bit: a 0 or 1 value (binary)
 - HW represents as two different voltages
 - 1: the presence of voltage (**high voltage**)
 - 0: the absence of voltage (**low voltage**)
- **Byte**: 8 bits, the smallest addressable unit

Memory:	01010101	10101010	00001111	...
(address)	[0]	[1]	[2]	...

- Other names:
 - 4 bits: Nibble
 - “Word”: Depends on system, often 4 bytes

Files

Sequence of bytes... nothing more, nothing less



Binary Digits (BITS)

- One bit: two values (0 or 1)
- Two bits: four values (00, 01, 10, or 11)
- Three bits: eight values (000, 001, ..., 110, 111)



How many unique values can we represent with 9 bits? Why?

- One bit: two values (0 or 1)
 - Two bits: four values (00, 01, 10, or 11)
 - Three bits: eight values (000, 001, ..., 110, 111)
- A. 18
- B. 81
- C. 256
- D. 512
- E. Some other number of values.

How many unique values can we represent with 9 bits? Why?

- One bit: two values (0 or 1)
- Two bits: four values (00, 01, 10, or 11)
- Three bits: eight values (000, 001, ..., 110, 111)

A. 18

B. 81

C. 256

D. 512

E. Some other number of values.

How many values?

1 bit:

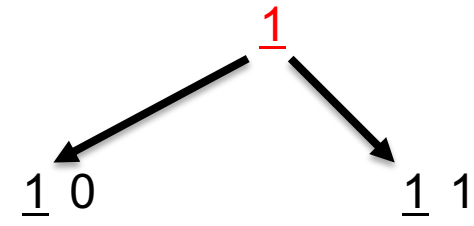
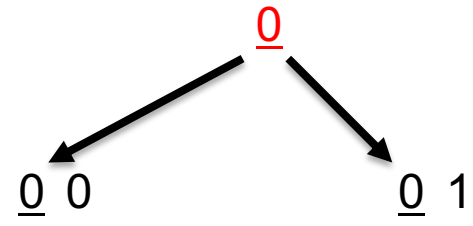
0

1

How many values?

1 bit:

2 bits:



How many values?

1 bit:

0

1

2 bits:

0 0

0 1

1 0

1 1

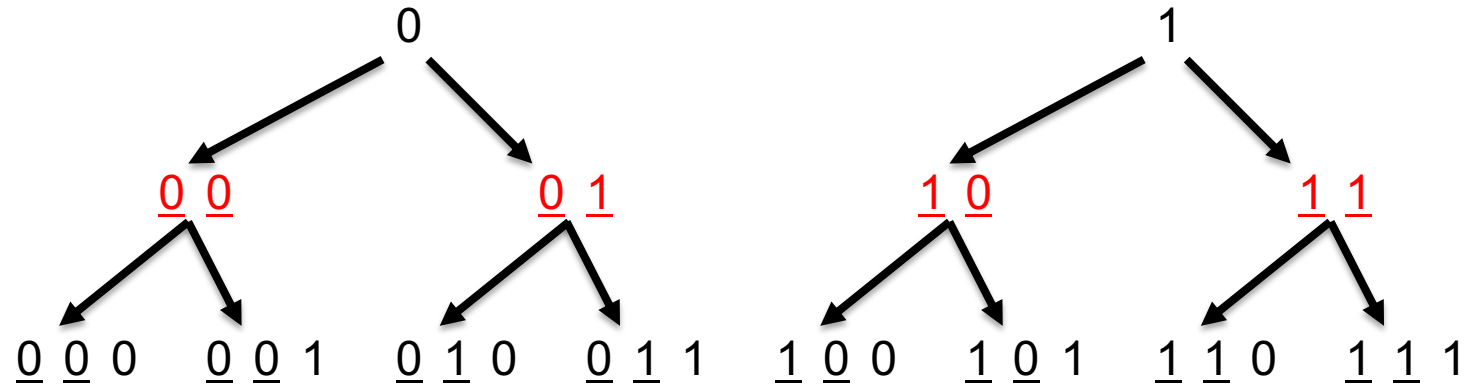
3 bits:

0 0 0 0 0 1

0 1 0 0 1 1

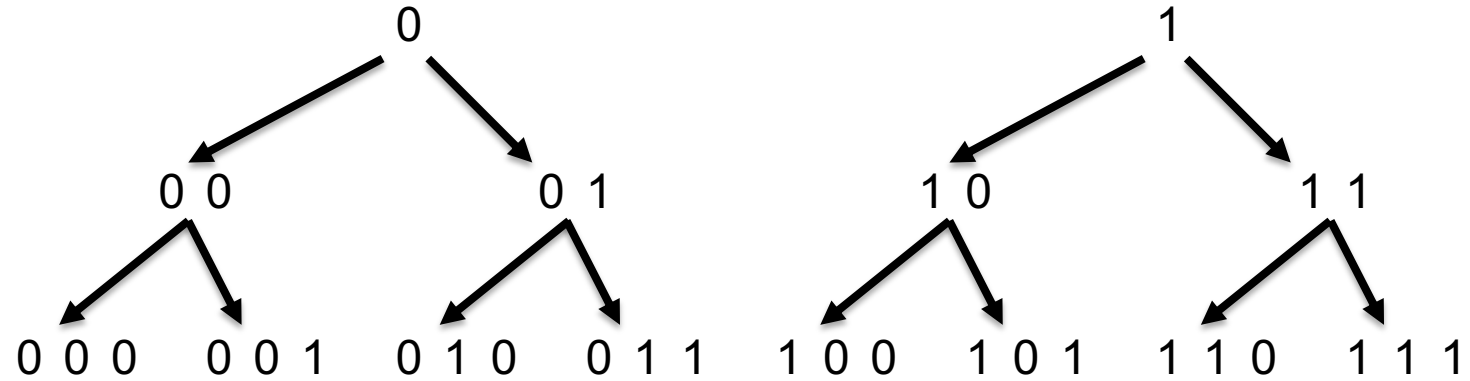
1 0 0 1 0 1

1 1 0 1 1 1



How many values?

1 bit:



2 bits:

3 bits:

4 bits:

0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 16 values
0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 1

1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1
1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1

N bits: 2^N values

C types and their (typical!) sizes

- 1 byte: `char`, `unsigned char`
- 2 bytes: `short`, `unsigned short`
- 4 bytes: `int`, `unsigned int`, `float`
- 8 bytes: `long long`, `unsigned long long`, `double`
- 4 or 8 bytes: `long`,

```
unsigned long v1;  
short s1;  
long long ll;
```

```
// prints out number of bytes  
printf("%lu %lu %lu\n", sizeof(v1), sizeof(s1), sizeof(ll));
```

WARNING: These sizes are NOT a guarantee. Don't always assume that every system will use these values!

How do we use this storage space (bits) to represent a value?

Let's start with what we know...

- Digits 0-9
- Positional numbering
- Digits are composed to make larger numbers
- Known as **Base 10** representation



Decimal number system (Base 10)

- Sequence of digits in range [0, 9]

64025



Digit #0: 1's place, "least significant digit"

Digit #1: 10's place

Digit #4: "most significant digit"

Decimal: Base 10

A number, written as the sequence of N digits,

$$d_{n-1} \dots d_2 d_1 d_0$$

where d is in {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, represents the value:

$$[d_{n-1} * 10^{n-1}] + [d_{n-2} * 10^{n-2}] + \dots + [d_1 * 10^1] + [d_0 * 10^0]$$

64025 =

$$6 * 10^4 + 4 * 10^3 + 0 * 10^2 + 2 * 10^1 + 5 * 10^0$$

$$60000 + 4000 + 0 + 20 + 5$$

Binary: Base 2

- Used by computers to store digital values.
- Indicated by prefixing number with **0b**
- A number, written as the sequence of N digits, $d_{n-1} \dots d_2 d_1 d_0$, where d is in $\{0,1\}$, represents the value:

$$[d_{n-1} * 2^{n-1}] + [d_{n-2} * 2^{n-2}] + \dots + [d_2 * 2^2] + [d_1 * 2^1] + [d_0 * 2^0]$$

Converting Binary to Decimal

Most significant bit \longrightarrow 10001111 \longleftarrow Least significant bit
7 6 5 4 3 2 1 0

Representation: $1 \times 2^7 + 0 \times 2^6 + \dots + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

128 + + 8 + 4 + 2 + 1

10001111 = 143

Hexadecimal: Base 16

Indicated by prefixing number with 0x

A number, written as the sequence of N digits,

$$d_{n-1} \dots d_2 d_1 d_0,$$

where d is in {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}, represents:

$$[d_{n-1} * 16^{n-1}] + [d_{n-2} * 16^{n-2}] + \dots + [d_2 * 16^2] + [d_1 * 16^1] + [d_0 * 16^0]$$

Generalizing: Base **b**

The meaning of a digit depends on its position in a number.

A number, written as the sequence of N digits,

$$d_{n-1} \dots d_2 d_1 d_0$$

in base **b** represents the value:

$$[d_{n-1} * b^{n-1}] + [d_{n-2} * b^{n-2}] + \dots + [d_2 * b^2] + [d_1 * b^1] + [d_0 * b^0]$$

Base 10: $[d_{n-1} * 10^{n-1}] + [d_{n-2} * 10^{n-2}] + \dots + [d_1 * 10^1] + [d_0 * 10^0]$

Other (common) number systems.

- Base 2: How data is stored in hardware.
- Base 8: Used to represent file permissions.
- Base 10: Preferred by people.
- Base 16: Convenient for representing memory addresses.
- Base 64: Commonly used on the Internet, (e.g. email attachments).

It's all stored as binary in the computer.

Different representations (or visualizations) of the **same information!**

What is the value of 0b110101 in decimal?

A number, written as the sequence of N digits $d_{n-1} \dots d_2 d_1 d_0$ where d is in $\{0, 1\}$, represents the value:

$$[d_{n-1} * 2^{n-1}] + [d_{n-2} * 2^{n-2}] + \dots + [d_2 * 2^2] + [d_1 * 2^1] + [d_0 * 2^0]$$

- A. 26
- B. 53
- C. 61
- D. 106
- E. 128

What is the value of 0b110101 in decimal?

A number, written as the sequence of N digits $d_{n-1} \dots d_2 d_1 d_0$ where d is in $\{0, 1\}$, represents the value:

$$[d_{n-1} * 2^{n-1}] + [d_{n-2} * 2^{n-2}] + \dots + [d_2 * 2^2] + [d_1 * 2^1] + [d_0 * 2^0]$$

- A. 26
- B. 53**
- C. 61
- D. 106
- E. 128

What is the value of 0x1B7 in decimal?

$$[d_{n-1} * 16^{n-1}] + [d_{n-2} * 16^{n-2}] + \dots + [d_2 * 16^2] + [d_1 * 16^1] + [d_0 * 16^0]$$

(Note: $16^2 = 256$)

- A. 397
- B. 409
- C. 419
- D. 437
- E. 439

DEC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

What is the value of 0x1B7 in decimal?

$$[d_{n-1} * 16^{n-1}] + [d_{n-2} * 16^{n-2}] + \dots + [d_2 * 16^2] + [d_1 * 16^1] + [d_0 * 16^0]$$

(Note: $16^2 = 256$)

A. 397

B. 409

C. 419

D. 437

E. 439

$$1 * 16^2 + 11 * 16^1 + 7 * 16^0 =$$

$$256 + 176 + 7 = \underline{439}$$

DEC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Important Point...

- You can represent the same value in a variety of number systems or bases.
- It's **all** stored as binary in the computer.
 - Presence/absence of voltage.

Hexadecimal: Base 16

- Fewer digits to represent same value
 - Same amount of information!
- Like binary, the base is power of 2
- Each digit is a “nibble”, or half a byte.

Each hex digit is a “nibble”

- One hex digit: 16 possible values (0-9, A-F)
- $16 = 2^4$, so each hex digit has exactly four bits worth of information.
- We can map each hex digit to a four-bit binary value.
(helps for converting between bases)

Each hex digit is a “nibble”

Example value: 0x1B7

Four-bit value: 1

Four-bit value: B (decimal 11)

Four-bit value: 7

In binary: 0001 1011 0111

1 B 7

Converting Decimal -> Binary

- Two methods:
 - division by two remainder
 - powers of two and subtraction

Method 1: decimal value D , binary result b (b_i is i th digit):

```
i = 0
while (D > 0)
    if D is odd
        set  $b_i$  to 1
    if D is even
        set  $b_i$  to 0
    i++
    D = D/2
```

Example: Converting
105

idea:

example: $D = 105$

$b_0 = 1$

Method 1: decimal value D , binary result b (b_i is i th digit):

```
i = 0
while (D > 0)
    if D is odd
        set  $b_i$  to 1
    if D is even
        set  $b_i$  to 0
    i++
    D = D/2
```

Example: Converting
105

idea:	D	example: D = 105	$b_0 = 1$
	D = D/2	D = 52	$b_1 = 0$

Method 1: decimal value D, binary result b (b_i is ith digit):

```
i = 0
while (D > 0)
    if D is odd
        set  $b_i$  to 1
    if D is even
        set  $b_i$  to 0
    i++
    D = D/2
```

Example: Converting
105

idea:	D	example: D = 105	$b_0 = 1$
	D = D/2	D = 52	$b_1 = 0$
	D = D/2	D = 26	$b_2 = 0$
	D = D/2	D = 13	$b_3 = 1$
	D = D/2	D = 6	$b_4 = 0$
	D = D/2	D = 3	$b_5 = 1$
	D = D/2	D = 1	$b_6 = 1$
	D = 0 (done)	D = 0	$b_7 = 0$

105 = 01101001



Method 2

- $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 64$, $2^7 = 128$

-

To convert 105:

- Find largest power of two that's less than 105 (64)
- Subtract 64 ($105 - 64 = \underline{41}$), put a 1 in d_6
- Subtract 32 ($41 - 32 = \underline{9}$), put a 1 in d_5
- Skip 16, it's larger than 9, put a 0 in d_4
- Subtract 8 ($9 - 8 = \underline{1}$), put a 1 in d_3
- Skip 4 and 2, put a 0 in d_2 and d_1
- Subtract 1 ($1 - 1 = \underline{0}$), put a 1 in d_0 (Done)

$$\frac{1}{d_6} \quad \frac{1}{d_5} \quad \frac{0}{d_4} \quad \frac{1}{d_3} \quad \frac{0}{d_2} \quad \frac{0}{d_1} \quad \frac{1}{d_0}$$

What is the value of 357 in binary?

8 7 6 5 4 3 2 1 0

→ digit position

A. 1 0110 0011

B. 1 0110 0101

C. 1 0110 1001

D. 1 0111 0101

E. 1 1010 0101

$$2^0 = 1, \quad 2^1 = 2, \quad 2^2 = 4, \quad 2^3 = 8, \quad 2^4 = 16,$$

$$2^5 = 32, \quad 2^6 = 64, \quad 2^7 = 128, \quad 2^8 = 256$$

What is the value of 357 in binary?

8 7 6 5 4 3 2 1 0

→ digit position

A. 1 0110 0011

B. 1 0110 0101

C. 1 0110 1001

D. 1 0111 0101

E. 1 1010 0101

$$357 - 256 = 101$$

$$101 - 64 = 37$$

$$37 - 32 = 5$$

$$5 - 4 = 1$$

$$\frac{1}{d_8} \quad \frac{0}{d_7} \quad \frac{1}{d_6} \quad \frac{1}{d_5} \quad \frac{0}{d_4} \quad \frac{0}{d_3} \quad \frac{1}{d_2} \quad \frac{0}{d_1} \quad \frac{1}{d_0}$$

$$2^0 = 1, \quad 2^1 = 2, \quad 2^2 = 4, \quad 2^3 = 8, \quad 2^4 = 16,$$

$$2^5 = 32, \quad 2^6 = 64, \quad 2^7 = 128, \quad 2^8 = 256$$

So far: Unsigned Integers

With N bits, can represent values: 0 to 2^n-1

We can always add 0's to the front of a number without changing it:

$$10110 = \underline{0}10110 = \underline{000}10110 = \underline{00000}10110$$

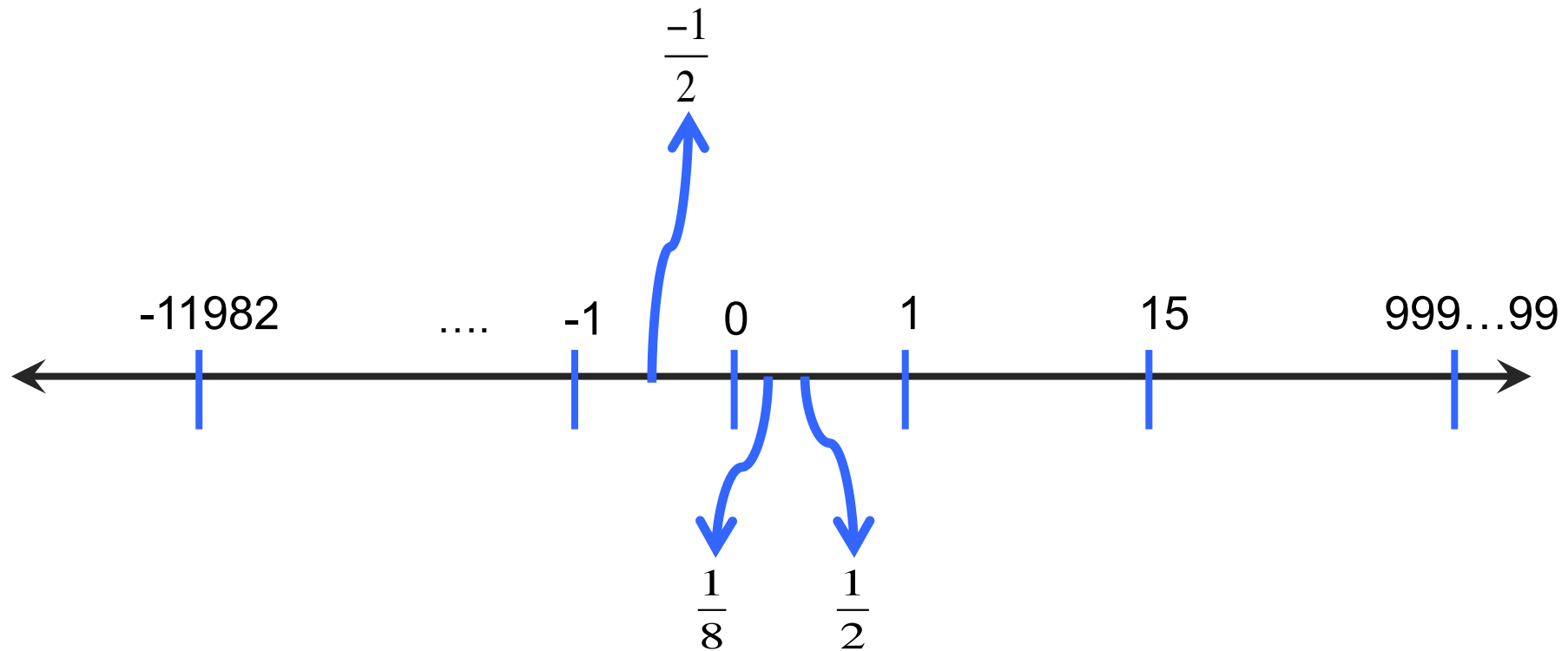
So far: Unsigned Integers

With N bits, can represent values: 0 to 2^n-1

- 1 byte: char, unsigned char
- 2 bytes: short, unsigned short
- 4 bytes: int, unsigned int, float
- 8 bytes: long long, unsigned long long, double
- 4 or 8 bytes: long, unsigned long

Additional Info: Fractional binary numbers

How do we represent fractions in binary?



Additional Info: Representing Signed Float Values

- One option (used for floats, NOT integers)
 - Let the first bit represent the sign
 - 0 means positive
 - 1 means negative
- For example:
 - 0101 → 5
 - 1101 → -5
- Problem with this scheme?

Additional Info: Floating Point Representation

1 bit for sign sign | exponent | fraction |
8 bits for exponent
23 bits for precision

$$\text{value} = (-1)^{\text{sign}} * 1.\text{fraction} * 2^{(\text{exponent}-127)}$$

let's just plug in some values and try it out

$$\begin{aligned} 0x40ac49ba: & \quad 0 \ 10000001 \quad 01011000100100110111010 \\ & \quad \text{sign} = 0 \ \text{exp} = 129 \quad \text{fraction} = 2902458 \\ & \quad = 1 * 1.2902458 * 2^2 = 5.16098 \end{aligned}$$

I don't expect you to memorize this