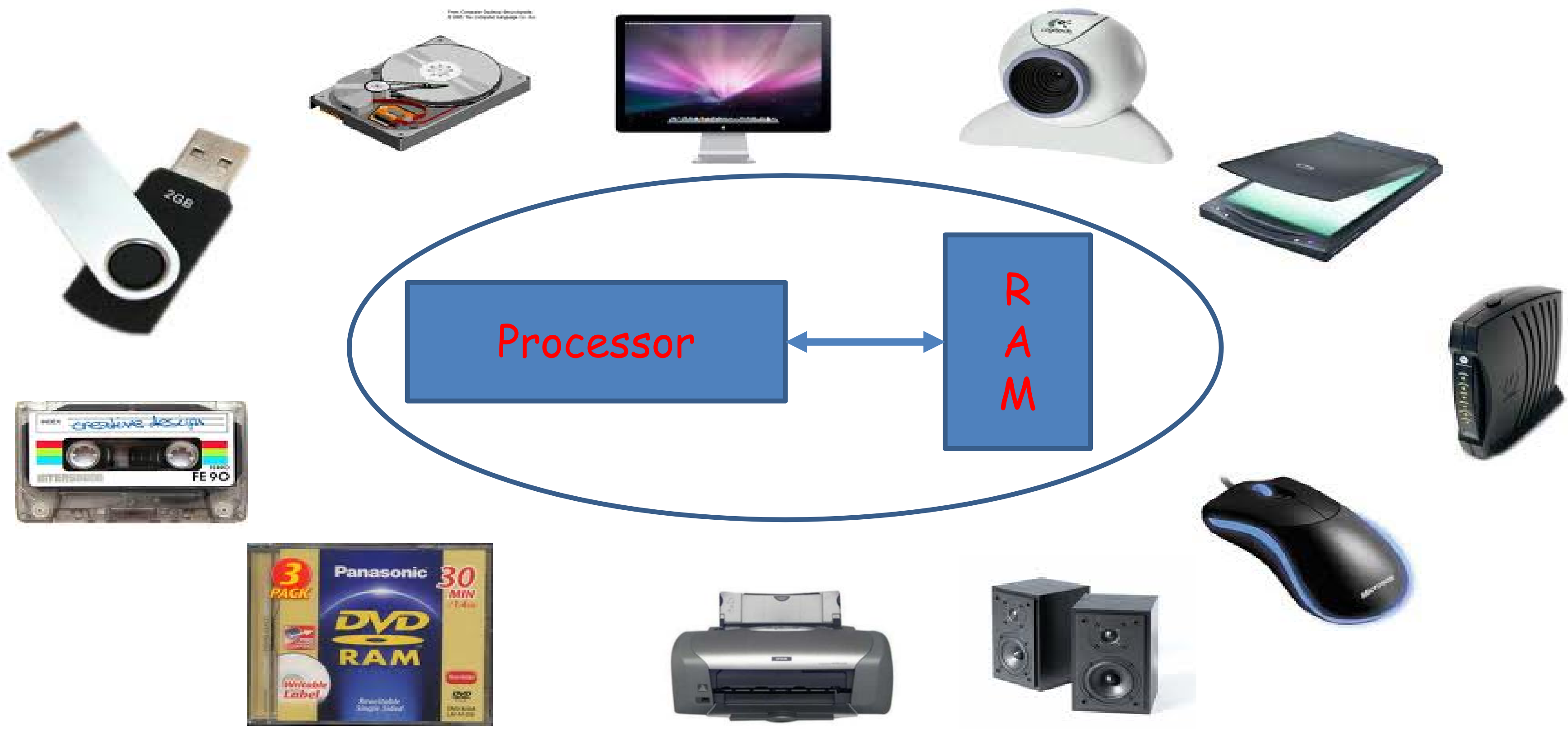
CS227: Digital Systems

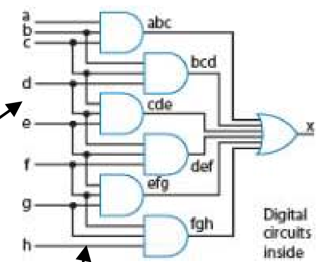
Introduction

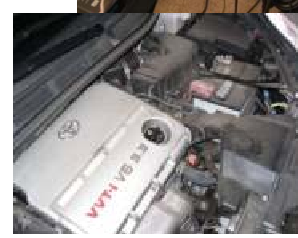


- Computer Systems
  - Internal (processor + memory (RAM))
  - Peripheral (Disk, Display, Audio, Eth,..)

Digital Design?

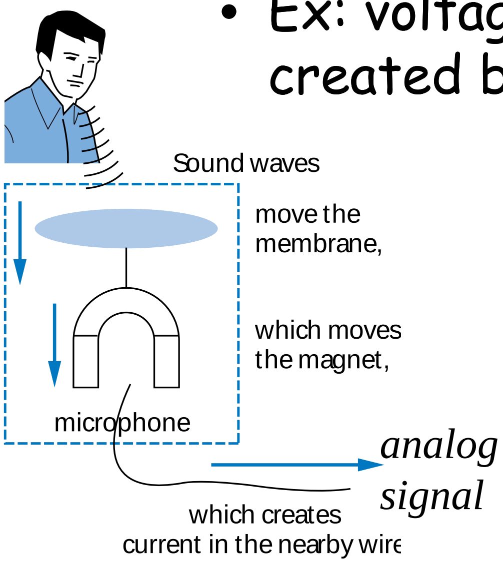
- Look “under the hood” of computers
  - Solid understanding --> confidence, insight, even better programmer when aware of hardware resource issues
- Electronic devices becoming digital
  - Enabled by shrinking and more capable chips
  - Enables:
    - Better devices: Better sound recorders, cameras, cars, cell phones, medical devices,...
    - New devices: Video games, PDAs, ...
  - Known as “embedded systems”
    - Thousands of new devices every year
    - Designers needed: Potential career direction

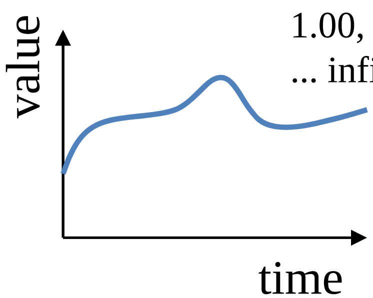






What Does "Digital" Mean?

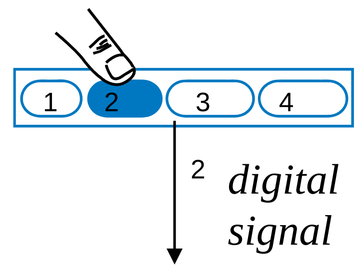
- Analog signal
  - Inifinite possible values
    - Ex: voltage on a wire created by microphone

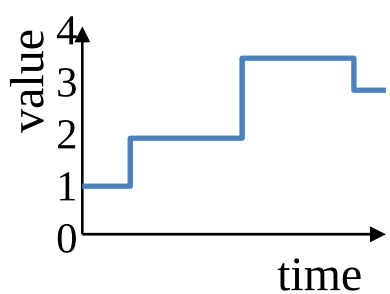




Possible values:
1.00, 1.01, 2.0000009,
... infinite possibilities

- Digital signal
  - Finite possible values
    - Ex: button pressed on a keypad

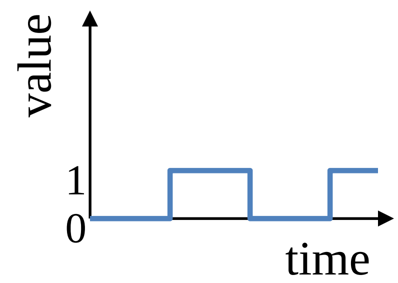




Possible values:
0, 1, 2, 3, or 4.
That's it.

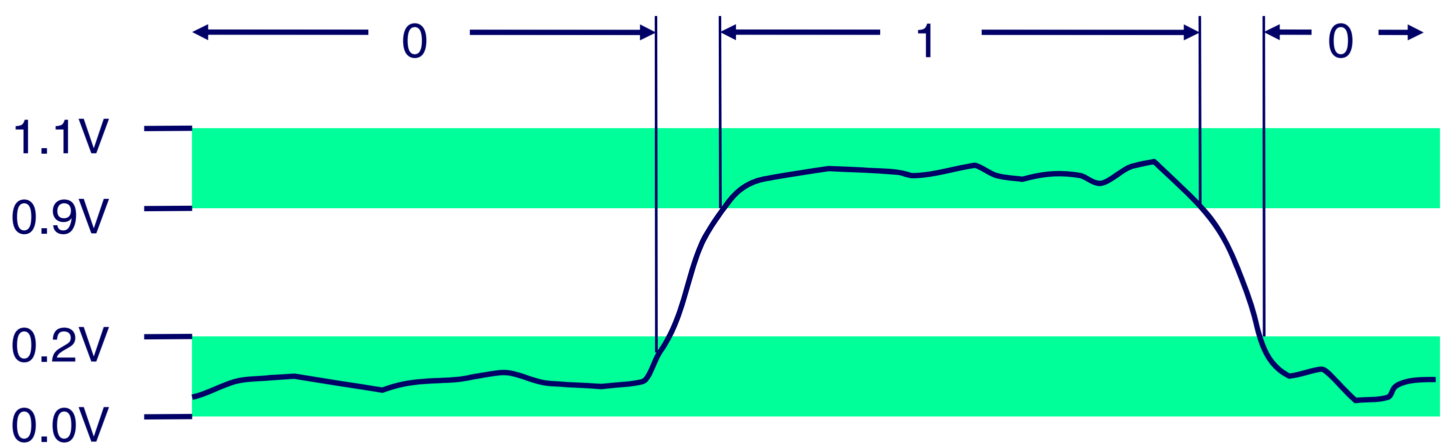
Digital Signals with Only Two Values: Binary

- ***Binary*** digital signal -- only *two* possible values
  - Typically represented as **0** and **1**
  - One *b*inary dig*it* is a ***bit***
  - We'll only consider *binary* digital signals
  - Binary is popular because
    - Transistors, the basic digital electric component, operate using *two* voltages Storing/transmitting one of *two* values is easier than three or more (e.g., loud beep or quiet beep, reflection or no reflection)



Everything is bits

- Each bit is 0 or 1
- By encoding/interpreting sets of bits in various ways
  - Computers determine what to do (instructions)
  - … and represent and manipulate numbers, sets, strings, etc…
- Why bits? Electronic Implementation
  - Easy to store with bistable elements
  - Reliably transmitted on noisy and inaccurate wires

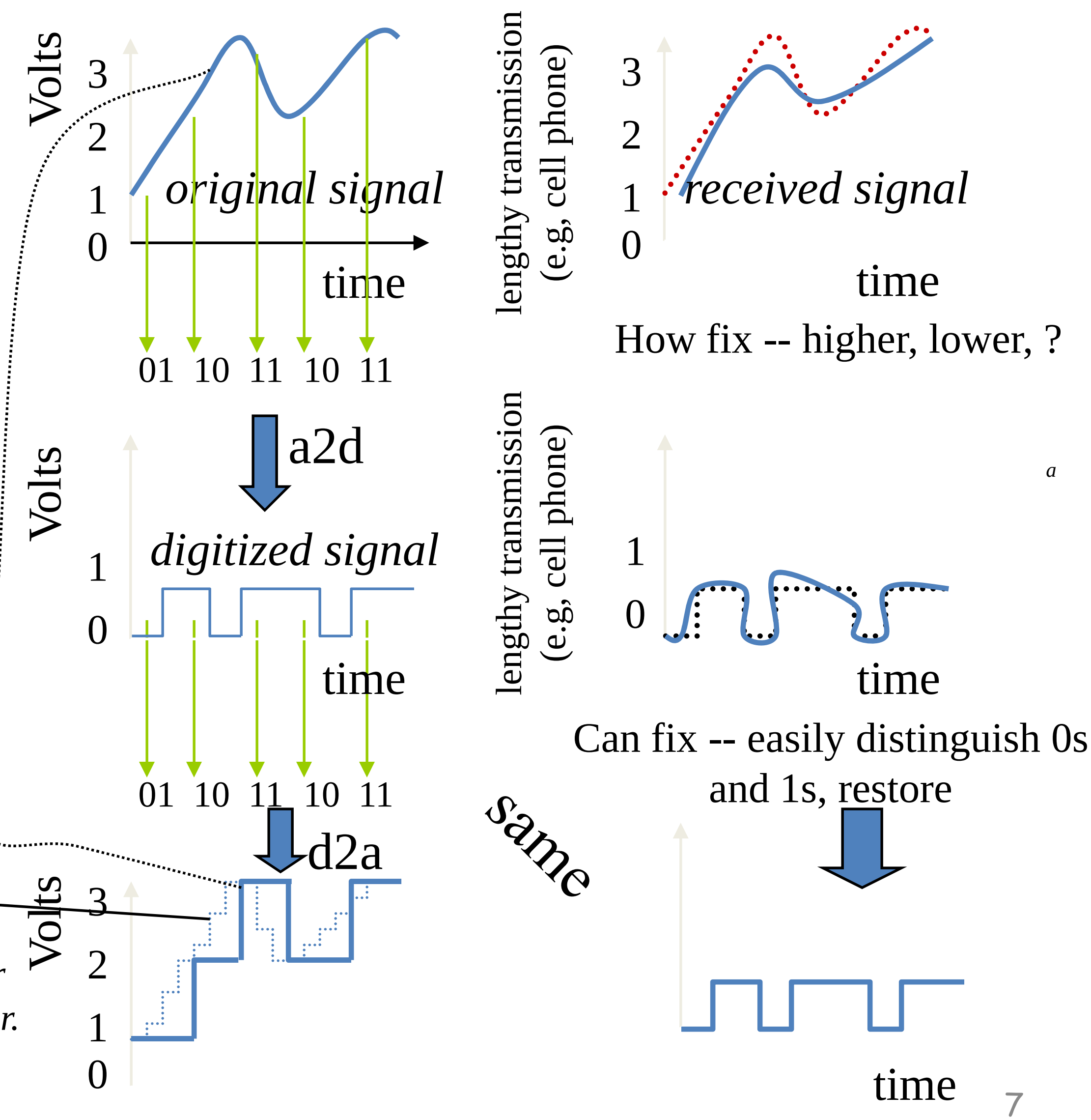


Real World Example -Digitization Benefit

- Analog signal (e.g., audio) may lose quality
  - Voltage levels not saved/copied/transmitted perfectly
- Digitized version enables near-perfect save/cpy/trn.
  - "Sample" voltage at particular rate, save sample using bit encoding
  - Voltage levels still not kept perfectly
  - But we can distinguish 0s from 1s

Let bit encoding be:
1 V: "01"
2 V: "10"
3 V: "11"

Digitized signal not perfect re-creation, but higher sampling rate and more bits per encoding brings closer.



Digitized Audio: Compression Benefit

- Digitized audio can be compressed
  - e.g., MP3s
  - A CD can hold about 20 songs uncompressed, but about 200 compressed
- Compression also done on digitized pictures (jpeg), movies (mpeg), and more
- Digitization has many other benefits too

Example compression scheme:

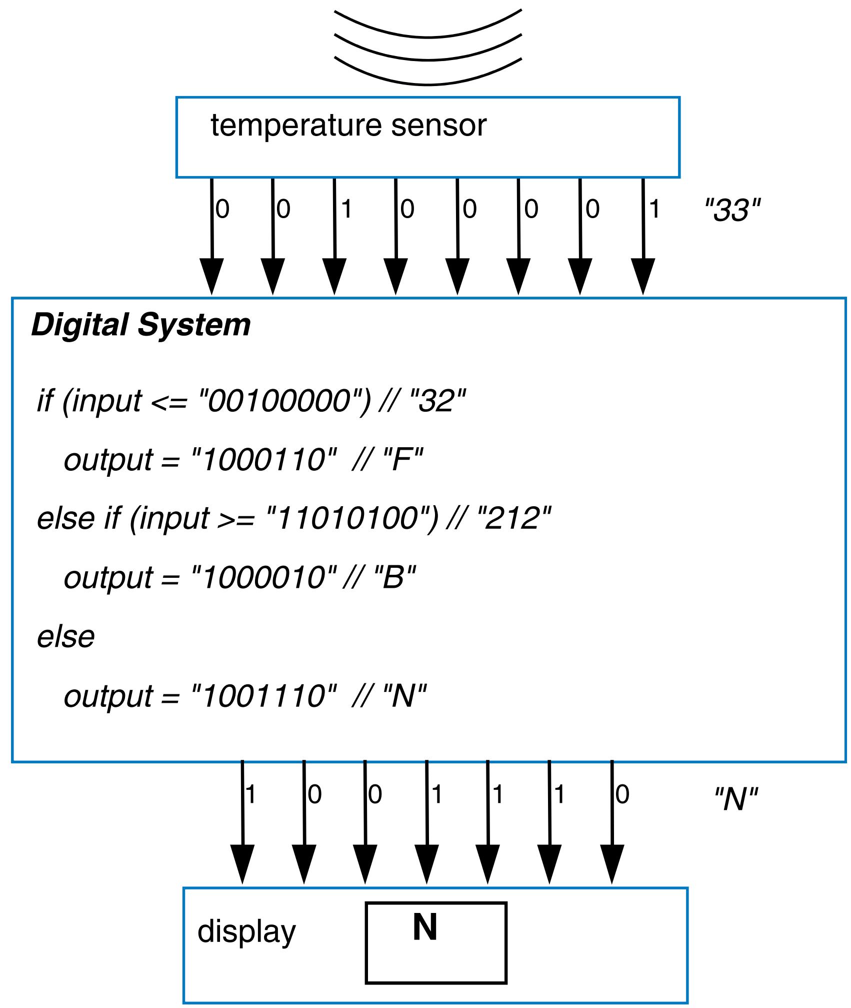
00 --> 0000000000
01 --> 1111111111
1X --> X

0000000000 0000000000 0000001111 1111111111

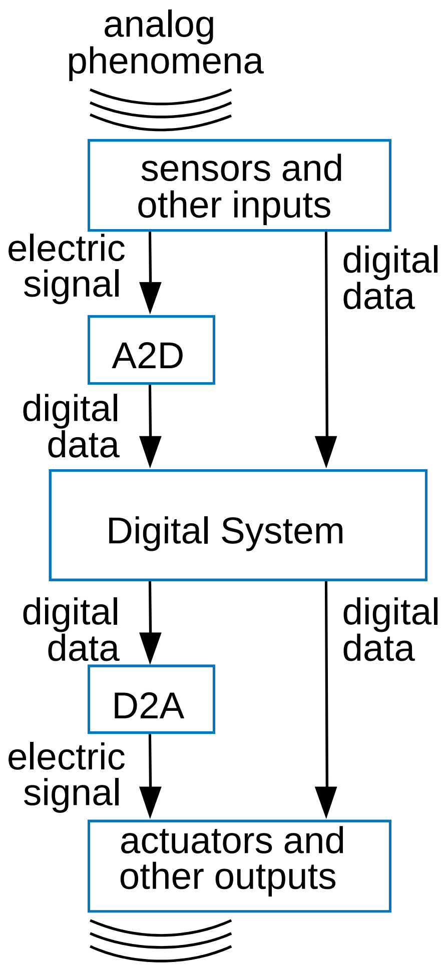
00 00 10000001111 01

Using Digital Data in a Digital System

- A temperature sensor outputs temperature in binary
- The system reads the temperature, outputs ASCII code:
  - "F" for freezing (0-32)
  - "B" for boiling (212 or more)
  - "N" for normal
- A display converts its ASCII input to the corresponding letter

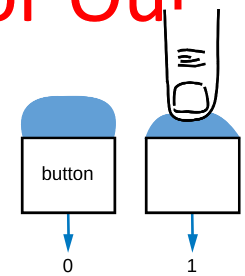


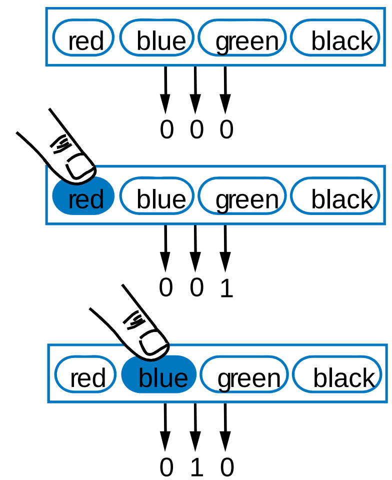
How Do We Encode Data as Binary for Our Digital System?

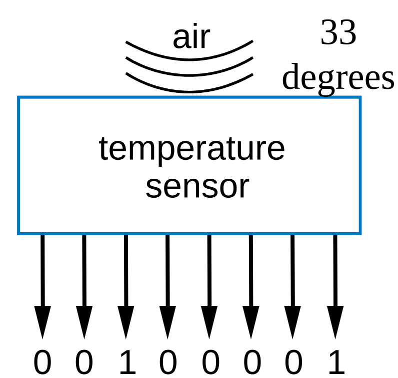


- Some inputs inherently binary
  - Button: not pressed (0), pressed (1)
- Some inputs inherently digital
  - Just need encoding in binary
  - e.g., multi-button input: encode red=001, blue=010, ...
- Some inputs analog
  - Need analog-to-digital conversion
  - As done in earlier slide -- sample and encode with bits

A2D -Analog to digital converter
D2A-Digital to Analog converter

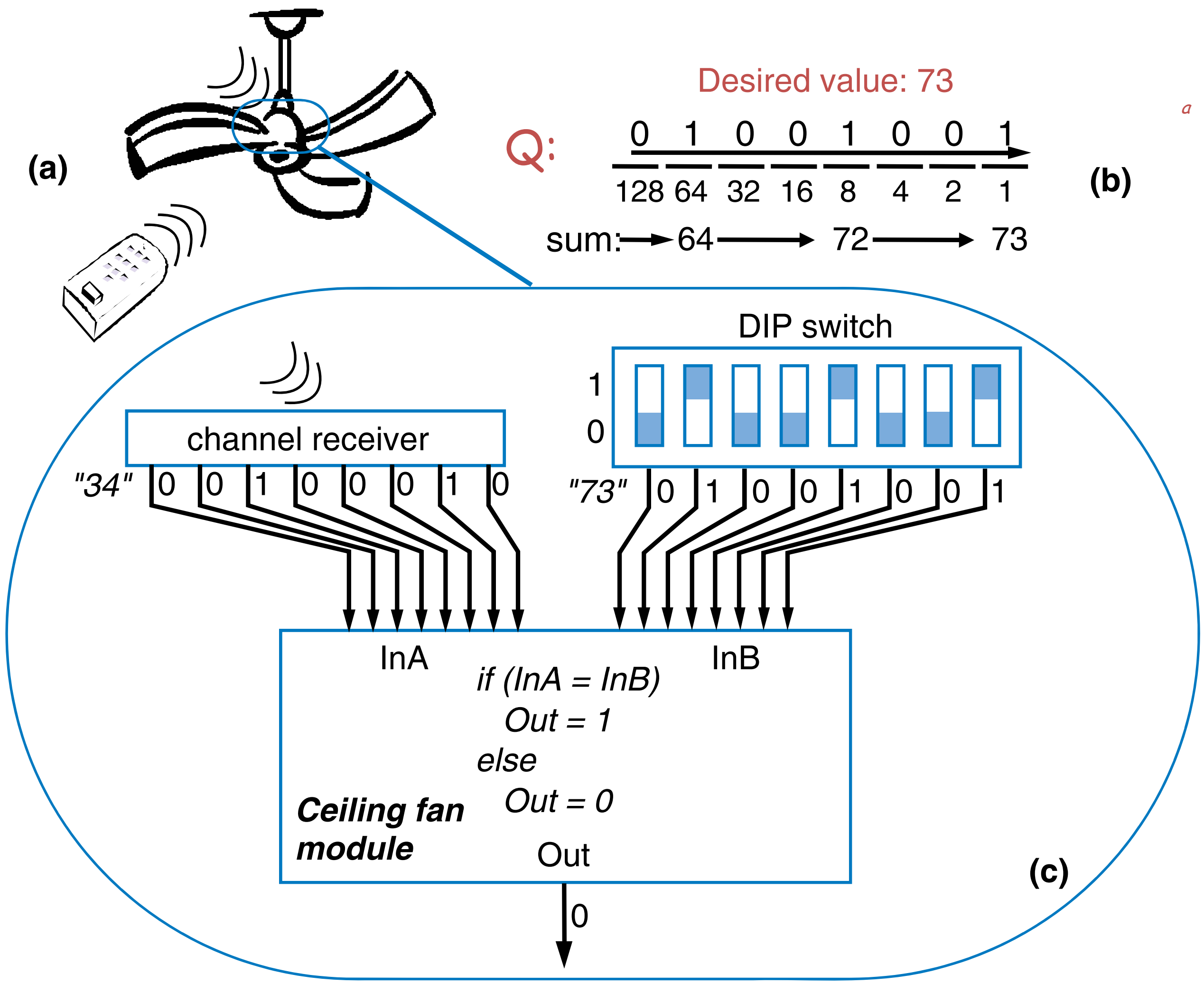






Example: DIP-Switch Controlled Channel

- Ceiling fan receiver should be set in factory to respond to channel "73"
- Convert 73 to binary, set DIP switch accordingly



Home work? (Hw1)

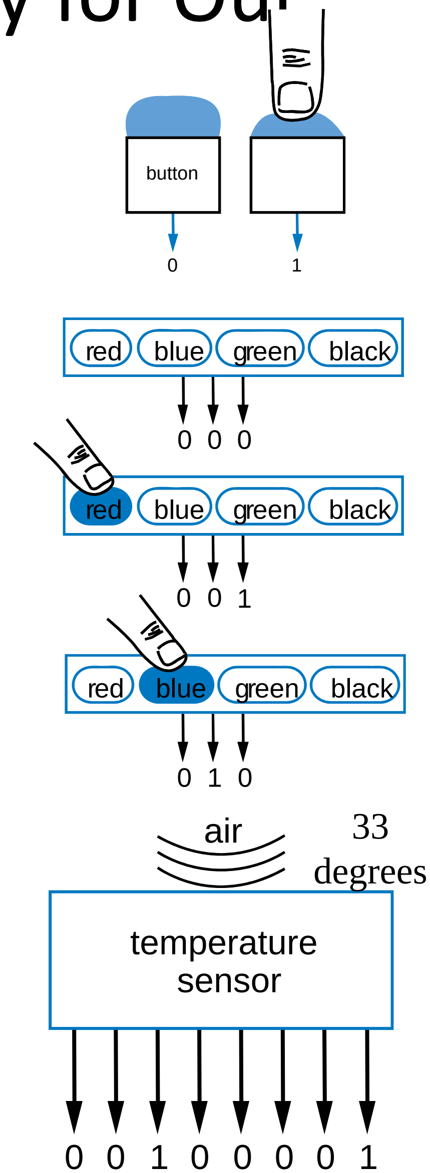
Prepare a list of all real world Sensors-
with one example

Record in your note book

How Do We Encode Data as Binary for Our Digital System?

(encoded)

- Button: not pressed (0000) 000
- Button: Red pressed (1000) 100
- Button: Blue pressed (0100) 010
- Button: Green pressed (0010) 110
- Button: black pressed (0001) 001



Data Representation

Numbering System Characteristics:

- **The number of characters in the number system is equal to the radix of the number system.**
- **Example:**
  - **There are 10 characters in the decimal number system.**

    (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) : $(13)_{10}$
  - **There are 2 characters in the binary number system. (0, 1). : $(1101)_2$**
- **Compact representation**
  - **Octal : $(15)_8$**
  - **Hexadecimal : $(D)_{16}$**

Binary Data Representation

Computers use binary numbers:

- **Binary numbers correspond directly with values in Boolean logic.**
- **Computers combine multiple digits to form a single data value to represent large numbers.**

Binary Data Representation

	Binary system (Base 2)					Decimal system (Base 10)			
Place Values	2^3 8	2^2 4	2^1 2	2^0 1		10^3 1000	10^2 100	10^1 10	10^0 1
	0	0	0	0	=	0	0	0	0
	0	0	0	1	=	0	0	0	1
	0	0	1	0	=	0	0	0	2
	0	0	1	1	=	0	0	0	3
	0	1	0	0	=	0	0	0	4
	0	1	0	1	=	0	0	0	5
	0	1	1	0	=	0	0	0	6
	0	1	1	1	=	0	0	0	7
	1	0	0	0	=	0	0	0	8
	1	0	0	1	=	0	0	0	9
	1	0	1	0	=	0	0	1	0

Data Representation

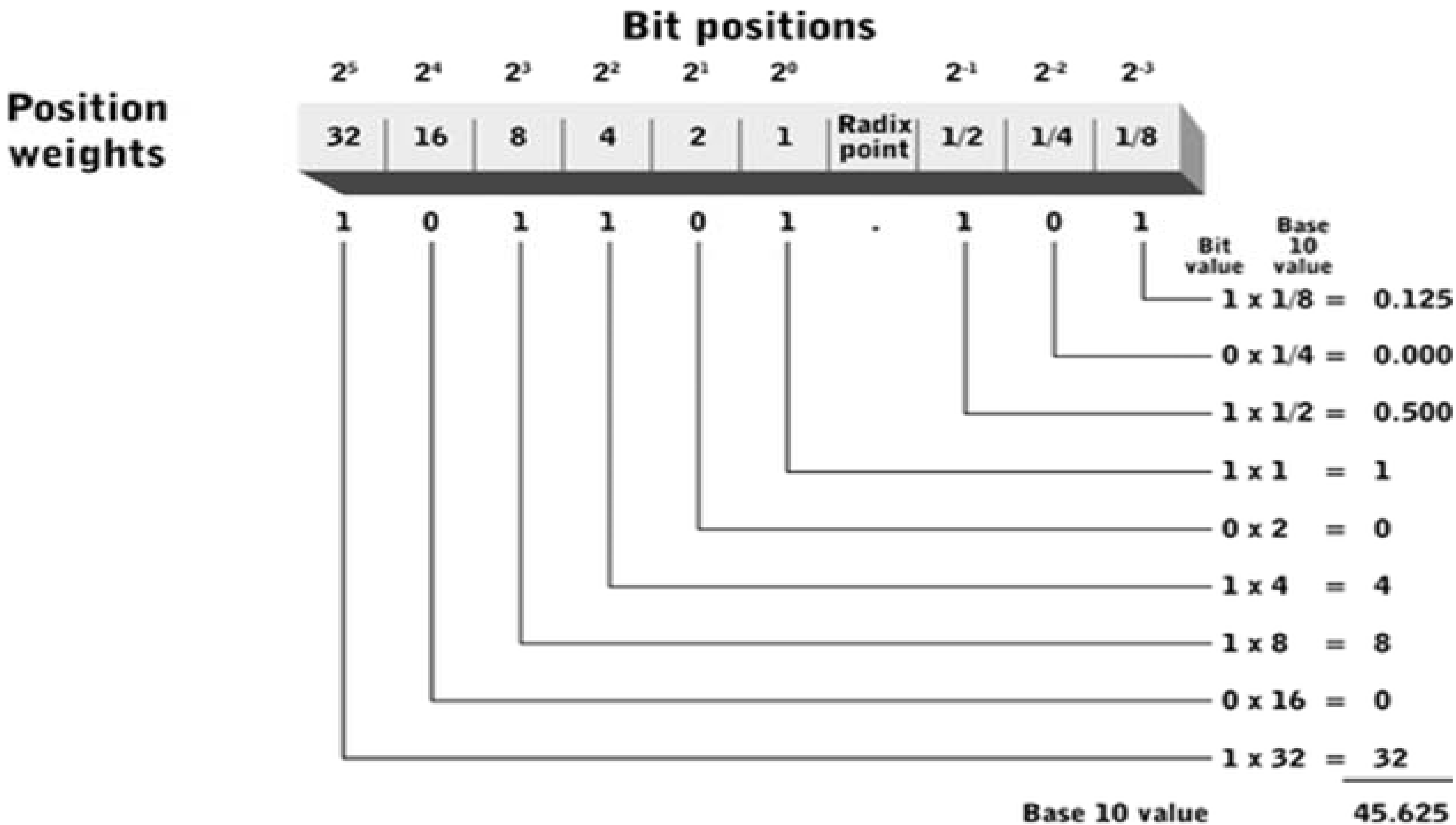
The fractional part of a numeric value is separated from the whole number by a period (radix point)

For Example: 5,689.368

(3 x .1) + (6 x .01) + (8 x .001) =

0.3 + 0.06 + 0.008 = 0.368

Binary Data Representation



Binary Data Representation

Number of bits (n)	Number of values (2^n)	Numeric range (decimal)
1	2	0...1
2	4	0...3
3	8	0...7
4	16	0...15
5	32	0...31
6	64	0...63
7	128	0...127
8	256	0...255
9	512	0...511
10	1024	0...1023
11	2048	0...2047
12	4096	0...4095
13	8192	0...8191
14	16384	0...16383
15	32768	0...32767
16	65536	0...65535

Converting Decimal to Binary

- **To convert from decimal to any radix/base we divide the number by the radix/base and record the remainder. This process is repeated until the number is 0 and then the final remainder is recorded. We shall see this in the following examples.**
- **To convert decimal to binary - radix=2**
- **To convert decimal to hexadecimal- radix=16**
- **To convert decimal to octal – radix =8**

Converting Decimal to Binary

Converting 207 to Binary..

- **207/2 = 103 remainder is 1 (LSB)**
- **103/2 = 51 remainder is 1**
- **51/2 = 25 remainder is 1**
- **25/2 = 12 remainder is 1**
- **12/2 = 6 remainder is 0**
- **6/2 = 3 remainder is 0**
- **3/2 = 1 remainder is 1**
- **1/2 = 0 remainder is 1 (MSB)**

The binary representation is the remainders read from the bottom to top. So, 207 = 11001111

Converting Binary to Decimal

- **We can just sum the values according to their positions e.g.**

$$(101001101)_2 = 2^8 + 2^6 + 2^3 + 2^2 + 2^0$$
$$= 256 + 64 + 8 + 4 + 1$$
$$= 333_{10}$$

- **Although this can become difficult as the length of the binary number increases.**

Decimal-to-binary Conversions: Fractional Part

- Successively multiply number by 2, taking integer part as result and chopping off integer part before next iteration.

- Converting 0.625 to binary

.625 * 2 = 1.25 integer part = 1

.25 * 2 = 0.5 integer part = 0

.5 * 2 = 1 integer part = 1

Ans = 0.101

Decimal-to-binary Conversions: Fractional Part

- Successively multiply number by 2, taking integer part as result and chopping off integer part before next iteration.
- May be unending!
- Example: convert 0.3 to binary.

.3 * 2 = .6 integer part = 0

.6 * 2 = 1.2 integer part = 1

.2 * 2 = .4 integer part = 0

.4 * 2 = .8 integer part = 0

.8 * 2 = 1.6 integer part = 1

.6 * 2 = 1.2 integer part = 1, etc.

Ans = 0.0100110 …

Octal Data Representation

- **Some operating systems and machine language programs use octal notation.**
- **The base (radix) of an Octal number system is 8.**
- **There are 8 characters in the octal number system. (0, 1, 2, 3, 4, 5, 6, 7)**

Eg. $(25)_{10} = (31)_8$

Hexadecimal Data Representation

- **The base (radix) of a hexadecimal number system is 16.**
- **There are 16 characters in the hexadecimal number system.**
- **There are only 10 characters in the Arabic number system that can be used to represent some of the 16 characters in the hexadecimal number system.**
- **The letters A, B, C, D, E, F are used to represent the last 6 characters in the hexadecimal number system.**

Octal/Decimal/Hexa Decimal Values

Octal	Decimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
	8
	9

Hexadecimal	Decimal
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Encoding Byte Values

- Byte = 8 bits
  - Binary 00000000_2 to 11111111_2
  - Decimal: 0_{10} to 255_{10}
  - Hexadecimal 00_{16} to FF_{16}
    - Base 16 number representation
    - Use characters '0' to '9' and 'A' to 'F'
    - Write $FA1D37B_{16}$ in C as
      - 0xFA1D37B
      - 0xfa1d37b

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

15213: $\underbrace{0011}_{3}\ \underbrace{1011}_{B}\ \underbrace{0110}_{6}\ \underbrace{1101}_{D}$

Binary to Hexadecimal

Notice the Pattern:

- **Largest 4 digit binary is 1111**
- **1 hex digit will represent a 4 digit binary number**
- **Highest hex digit is F**

Binary to Hexadecimal

Hexadecimal	Binary	Hexadecimal	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Converting Hex to Binary

Steps:

- **Convert Hex number to groups of powers of 2.**
- **Convert to Binary number (Remember to drop leading zeros for first set of binary numbers - i.e. first left set)**

Converting Hex to Binary

$11F6_{16}$

=	1	1	F	6
=	000(1)	000(1)	(8)(4)(2)(1)	0(4)(2)0
=	0001	0001	1111	0110

$= (1000111110110)_2$

Convert Binary to Hex

Steps:

- **Separate into 4 bit groups starting from the right.**
- **Calculate decimal equivalent (in placeholders in powers of 2)**
- **Convert to Hexadecimal number**

Convert Binary to Hex

1000111110110_2

$= 1 \quad 0001 \quad 1111 \quad 0110$

$= 0001 \quad 0001 \quad 1111 \quad 0110$

$= 1 \quad 1 \quad (8)(4)(2)(1) \quad 0(4)(2)0$

$= 1 \quad 1 \quad 15 \quad 6$

$= 11F6_{16}$

Converting Octal to Hex

- **The easiest method to convert between Octal and Hexadecimal is to convert to binary as an intermediate step**

- **Regroup binary in groups of 4 for hexadecimal and 3 for octal**

Converting Decimal to Hex

Converting 207 to Hexadecimal..

207/16 = 12 remainder is 15 = F
12/16 = 0 remainder is 12 = C

- **Again we read the remainders from the bottom to the top. So, $(207)_{10}$ = $(CF)_{16}$**

- **We usually convert the decimal number to binary and then covert the binary number to hexadecimal.**

Converting Hex to Decimal

- **Given $5D2A1_{16}$ convert it to decimal. We could just sum the values according to their positions.**

  $5 = 5 \times 16^4 = 5 \times 65536 = 327680$

  $D = 13 \times 16^3 = 13 \times 4096 = 53248$

  $2 = 2 \times 16^2 = 2 \times 256 = 512$

  $A = 10 \times 16^1 = 10 \times 16 = 160$

  $1 = 1 \times 16^0 = 1 \times 1 = 1$

- **Summing the values we get**

  $327680 + 53248 + 512 + 160 + 1 = 381601_{10}$

Binary Addition

- **Binary addition is very simple.**
- **This is best shown in an example of adding two binary numbers…**

```
  1 1  1  1  1  1 <------ carries
    1  1  1  1  0  1
+      1  0  1  1  1
--------------------
  1 0  1  0  1  0  0
```

Binary Multiplication

- **Binary multiplication is much the same as decimal multiplication, except that the multiplication operations are much simpler…**

```
              1  0  1  1  1
X                1  0  1  0
-------------------------
              0  0  0  0  0
           1  0  1  1  1
        0  0  0  0  0
     1  0  1  1  1
-------------------------
     1  1  1  0  0  1  1  0
```

How To Represent Signed Numbers

- Plus and minus sign used for decimal numbers:

  25 (or +25), -16, etc.
- For computers, desirable to represent everything as *bits*.
- Three types of signed binary number representations:

  sign magnitude, *1's complement*, *2's complement*.
- In each case: left-most bit indicates sign:

  positive (0) or negative (1).

Consider *sign magnitude*:

$00001100_2 = 12_{10}$

Sign bit Magnitude

$10001100_2 = -12_{10}$

Sign bit Magnitude

One's Complement Representation

- The one's complement of a binary number involves inverting all bits.

  1's comp of 00110011 is 11001100

  1's comp of 10101010 is 01010101

- For an n bit number N the 1's complement is $(2^n-1) - N$.

Example. 12_{10} One's complement is 243

-

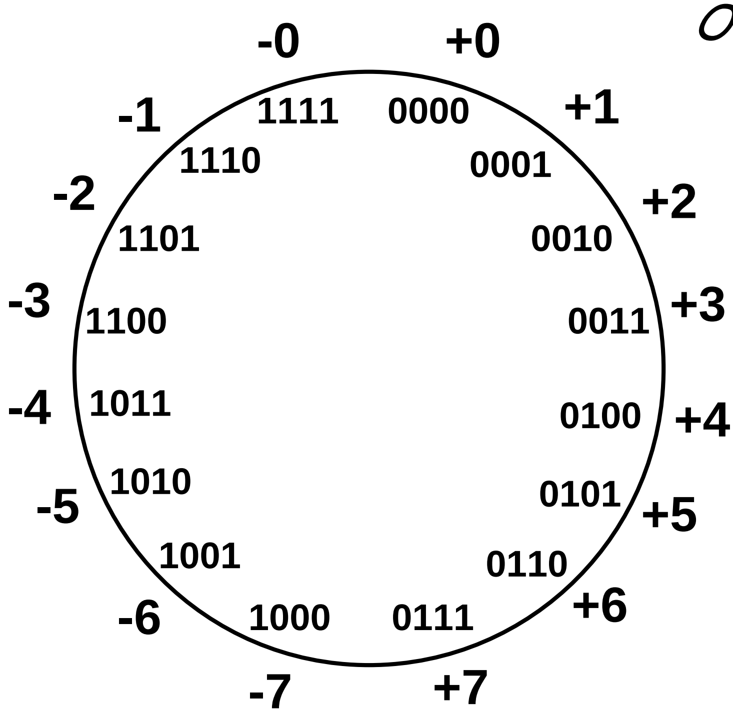
$0\underbrace{0001100}_{\text{Magnitude}}{}_2 = 12_{10}$

Sign bit Magnitude

$1\underbrace{1110011}_{\text{Magnitude}}{}_2 = -12_{10}$

Sign bit Magnitude

Ones Complement



+

0 100 = + 4

1 011 = - 4

-

Subtraction implemented by addition & 1's complement

Still two representations of 0! This causes some problems

Some complexities in addition

Two’s Complement Representation

- The two’s complement of a binary number involves inverting all bits and adding 1.

  2’s comp of 00110011 is 11001101

  2’s comp of 10101010 is 01010110

- For an n bit number N the 2’s complement is $(2^n - N)$.
- Eg. 12 , Two’s complement is 244
- To find negative of 2’s complement number take the 2’s complement.

$$00001100_2 = 12_{10}$$

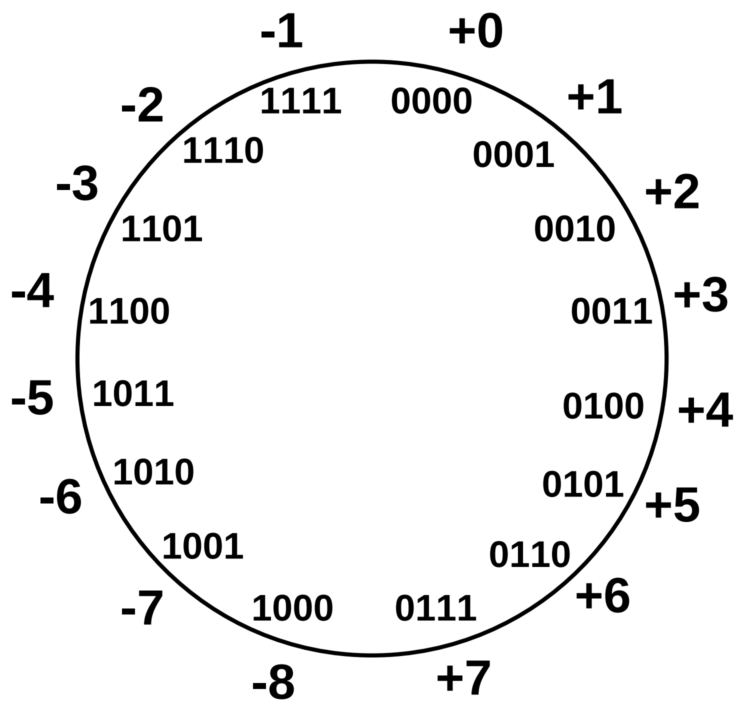
Sign bit Magnitude

$$11110100_2 = -12_{10}$$

Sign bit Magnitude

Two's Complement

like 1's comp except shifted one position clockwise



+

0 100 = + 4

1 100 = - 4

-

Only one representation for 0

One more negative number than positive number

Two's Complement Shortcuts

- ***Algorithm 1*** ***– Simply complement each bit and then add 1 to the result.***
  - **Finding the 2's complement of $(01100101)_2$ and of its 2's complement…**

```
N  =  01100101                 N =     10011011
      10011010                         01100100
 +           1                      +        1
     ---------------                  ---------------
      10011011                                01100101
```

- ***Algorithm 2 – Starting with the least significant bit, copy all of the bits up to and including the first 1 bit and then complementing the remaining bits.***

```
Eg1.  N      = 0 1 1 0 0 1 0 1        Eg2.    N       = 0 1 1 0 0 1 0 0
      [N]    = 1 0 0 1 1 0 1 1                [N]     = 1 0 0 1 1 1 0 0
```

Finite Number Representation

- Machines that use 2's complement arithmetic can represent integers in the range
  $$\underline{-2^{n-1} <= N <= 2^{n-1}-1}$$
  where n is the number of bits available for representing N. Note that
  $2^{n-1}-1 = (011..11)$ and $-2^{n-1} = (100..00)$
- For 2's complement more negative numbers than positive.
- For 1's complement two representations for zero.
- For an n bit number in base (radix) z there are z^n different unsigned values. $(0, 1, \ldots z^{n-1})$
  - Eg. For 2 digit : binary (4) decimal (100) , hex (256)

1's Complement Arithmetic

Let A and B are the two operands, then

Addition

A+B = Add (A +B)

Subtraction

A-B =

- Step 1: Take 1's complement of 2nd operand B
- Step 2: Add binary numbers (A+B')
- Step 3: Add carry to low order bit

1's Complement Addition

Step 1: Add binary numbers

Step 2: Add carry to lower-order bit

- Using 1's complement numbers, adding numbers is easy.
- For example, suppose we wish to add $+(1100)_2$ and $+(0001)_2$.
- Let's compute $(12)_{10} + (1)_{10}$.

$(12)_{10} = +(1100)_2 = (01100)_2$ in 1's comp.

$(1)_{10} = +(0001)_2 = (00001)_2$ in 1's comp.

```
      0 1 1 0 0
+     0 0 0 0 1
---------------
    0 0 1 1 0 1
              0
Add carry
---------------
Final Result
      0 1 1 0 1
```

2's Complement Addition

Step 1: Add binary numbers

Step 2: Ignore carry bit

- Using 2's complement numbers, adding numbers is easy.
- For example, suppose we wish to add $+(1100)_2$ and $+(0001)_2$.
- Let's compute $(12)_{10} + (1)_{10}$.

  $(12)_{10} = +(1100)_2 = (01100)_2$ in 2's comp.

  $(1)_{10} = +(0001)_2 = (00001)_2$ in 2's comp.

```
     0 1 1 0 0
+    0 0 0 0 1
--------------
 0   0 1 1 0 1
 ^
 |
 |  Ignore
```

Example Data Representations

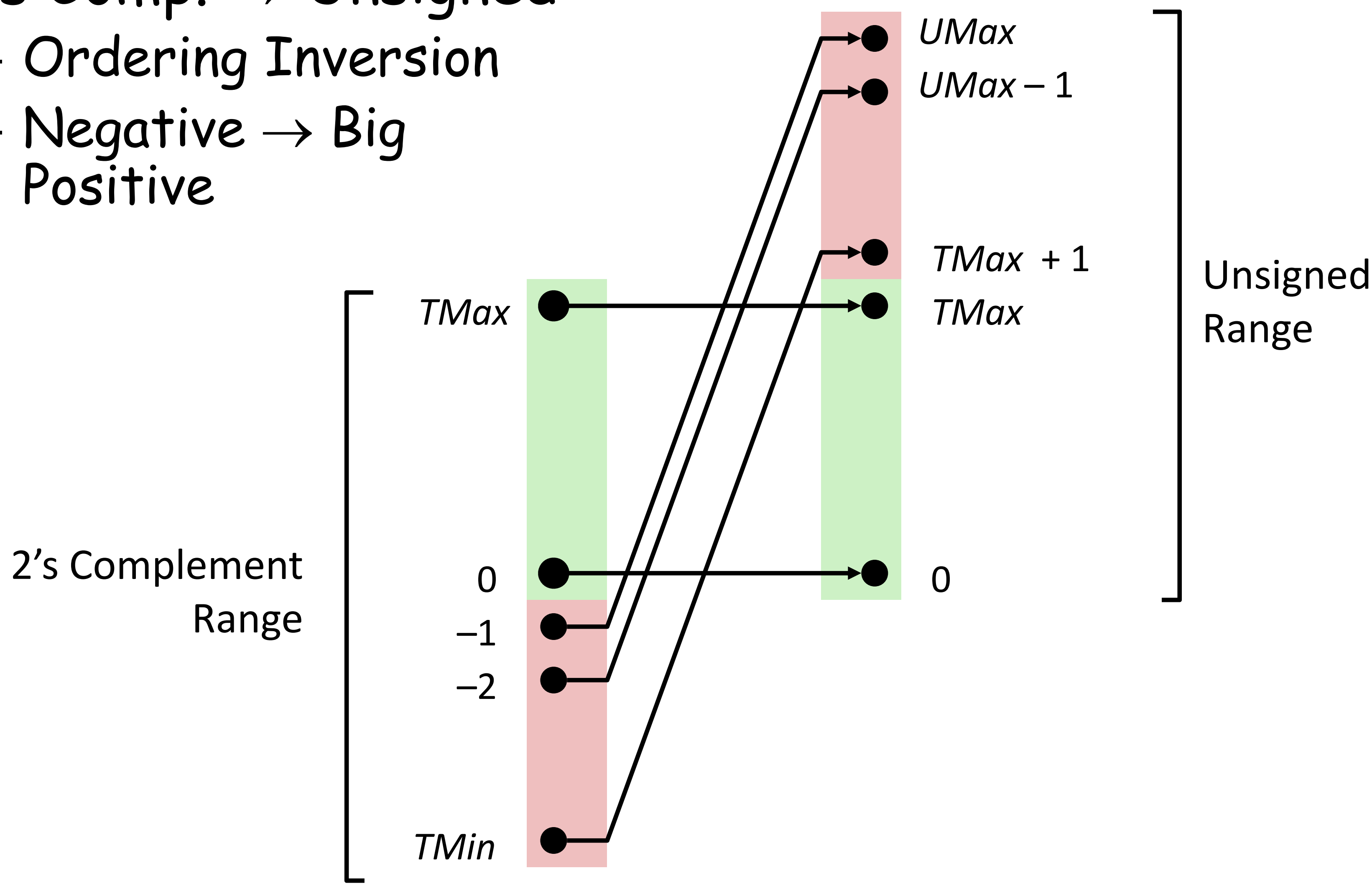
C Data Type	Typical 32-bit	Typical 64-bit	x86-64
`char`	1	1	1
`short`	2	2	2
`int`	4	4	4
`long`	4	8	8
`float`	4	4	4
`double`	8	8	8
pointer	4	8	8

Mapping Signed ↔ Unsigned

Bits	Signed		Unsigned
0000	0		0
0001	1		1
0010	2		2
0011	3		3
0100	4	=	4
0101	5		5
0110	6		6
0111	7		7
1000	-8		8
1001	-7		9
1010	-6		10
1011	-5	+/- 16	11
1100	-4		12
1101	-3		13
1110	-2		14
1111	-1		15

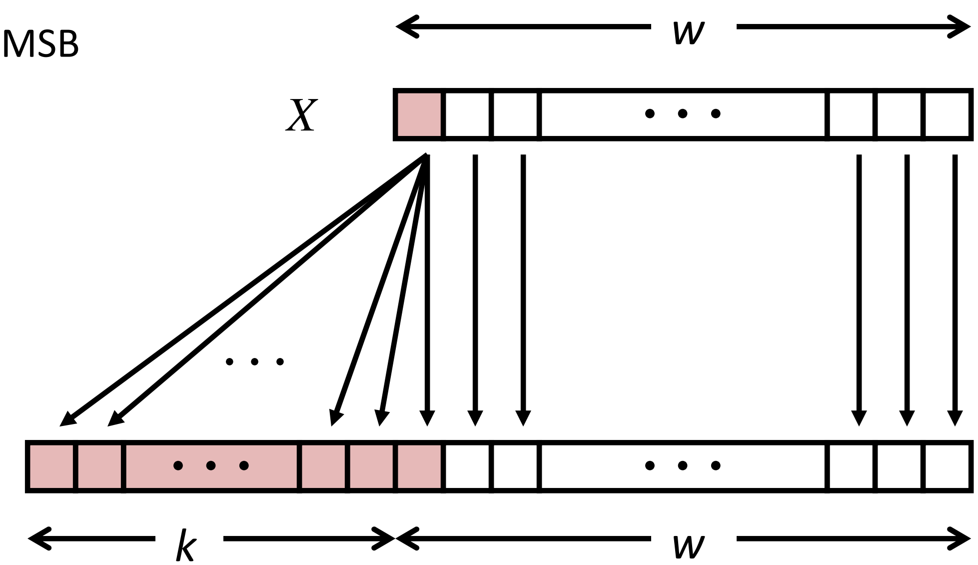
Conversion Visualized

- 2's Comp. → Unsigned
  - Ordering Inversion
  - Negative → Big Positive



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:
  - $X' = \underbrace{x_{w-1}, \ldots, x_{w-1}}_{k \text{ copies of MSB}}, x_{w-1}, x_{w-2}, \ldots, x_0$



Sign Extension: Simple Example

Positive number

	-16	8	4	2	1
10 =	0	1	0	1	0

	-32	16	8	4	2	1
10 =	0	0	1	0	1	0

Negative number

	-16	8	4	2	1
-10 =	1	0	1	1	0

	-32	16	8	4	2	1
-10 =	1	1	0	1	1	0

Larger Sign Extension Example

```
short int x =  15213;
int       ix = (int) x;
short int y = -15213;
int       iy = (int) y;
```

	Decimal	Hex	Binary
x	15213	3B 6D	00111011 01101101
ix	15213	00 00 3B 6D	00000000 00000000 00111011 01101101
y	-15213	C4 93	11000100 10010011
iy	-15213	FF FF C4 93	11111111 11111111 11000100 10010011

- Converting from smaller to larger integer data type
- C automatically performs sign extension

Summary

- Digital systems surround us
  - Inside computers
  - Inside huge variety of other electronic devices (embedded systems)
- Digital systems use 0s and 1s
  - Encoding analog signals to digital can provide many benefits
    - e.g., audio -- higher-quality storage/transmission, compression, etc.
  - Encoding integers as 0s and 1s: Binary numbers

- Signed numbers represented in signed magnitude, 1's complement, and 2's complement

- 2's complement most important (only 1 representation for zero).

- Important to understand treatment of sign bit for 1's and 2's complement.