# COMP 102: Computers and Computing

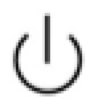
## Lecture 2: Bits&bytes, Switches, and Boolean Logic

Instructor: Kaleem Siddiqi (siddiqi@cim.mcgill.ca)

Class web page: www.cim.mcgill.ca/~siddiqi/102.html

# The Lowly Bit

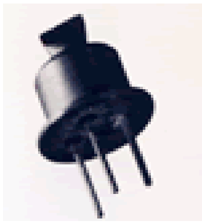
What is the smallest unit of information?

- Chemistry has its molecules.
- Physics has its strings.
- Computer science has its bits:
  - True / False
  - On / Off
  - 1 / 0
- Think of it as a switch: 

# Recall

- The vacuum tube:


- The transistor:



- These are electronic on/off switches.
- The difference engine used mechanical on/off switches (think "lever").

# What's a Bit?

- Word "Bit" is a contraction of "Binary digit"

- What's a **binary digit** ?
  - Base 10: In decimal number system, a digit can be any of the ten values 0, …, 9
  - Base 2: In binary number system, a digit can be any of the two values 0, 1

- Bits are nice because they are:
  - **Simple*:*** There's no smaller discrete distinction to be made.
  - **Powerful**: Sequences of bits can represent seemingly anything.

# Representing numbers

102

100s 10s 1s

- Decimal System uses 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Base 10
- Place-value number system: position of a digit interpreted to give the value

# Representing numbers: Decimal system

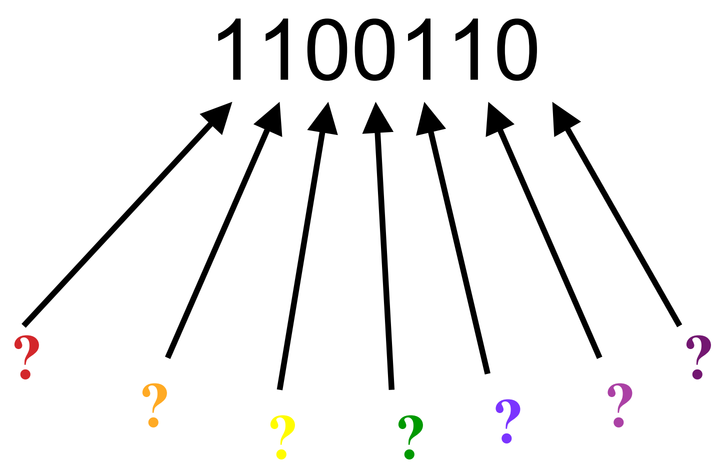
- $102 = 1 \times 100 + 0 \times 10 + 2 \times 1$

  $= 1 \times 10^2 + 0 \times 10^1 + 2 \times 10^0$

- 1 decimal digit produces 10 distinct values
- 2 decimal digits produce 100 distinct values
- 3 decimal digits produce 1000 distinct values

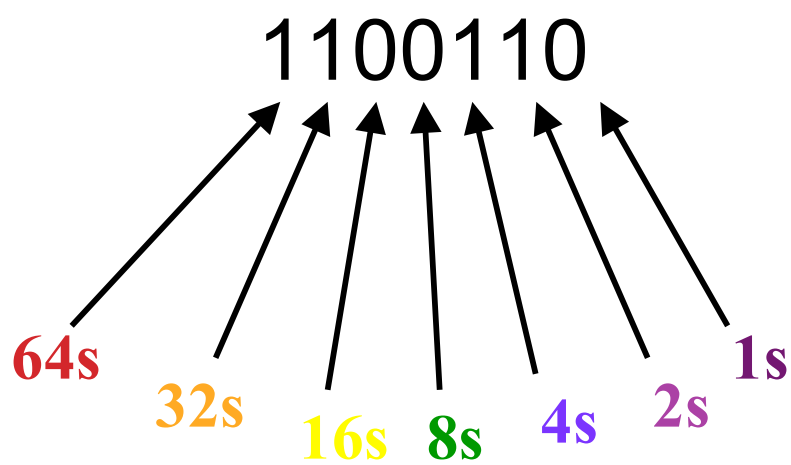
- *n* decimal digits produce $10^n$ distinct values

# Representing numbers: Binary system




- Binary System uses 2 digits: 0, 1
- Base 2

# Representing numbers: Binary system




- Binary System uses 2 digits: 0, 1
- Base 2

# Representing numbers: Binary system

$1{,}100{,}110_2 = 1 \times 64 + 1 \times 32 + 0 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$

$= 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

- 1 binary digit produces 2 distinct values
- 2 binary digits produce 4 distinct values
- 3 binary digits produce 8 distinct values

- *n* binary digits produce $2^n$ distinct values

# Representing numbers: Arbitrary Base

$23641_7$ = ?

$85342_9$ = ?

In general, use positional notation:

A number $a_n a_{n-1} a_{n-2} \ldots a_0$ in base b has the value

$$a_n * b^n + a_{n-1} * b^{n-1} + \ldots + a_0 * b^0$$

# Converting from Base 10 to Base 2

$136_{10} = ?_2$

Keep dividing by 2 and storing the remainder:

136/2 = 68, R = 0

68/2 = 34, R = 0

34/2 = 17, R = 0

17/2 = 8, R = 1

8/2 = 4, R = 0

4/2 = 2, R = 0

2/2 = 1, R = 0

1/2 = 0, R = 1 Answer: $10001000_2$

# Binary Numbers in Computing

- Easy to make fast, reliable, small devices that have only 2 states

- 1/0 represented by
  - hole/no hole in punched card
  - hi/low voltage (memory chips)
  - light bounces off/light doesn't bounce off (CDs/DVDs)
  - magnetic charge present/no magnetic charge (disks)

# Measuring Data

We can group number of binary digits and refer to the group sizes by special names:

- 1 bit(b) = $2^1$ = represents 2 different values
- 1 byte(B) = 8 bits = $2^8$ = 256 values
- 1 kilobyte(KB) = 1024 bytes = $2^{10}$ bytes
- 1 megabyte(MB) = 1024 KB = $2^{20}$ bytes
- 1 gigabyte(GB) = 1024 MB = $2^{30}$ bytes
- 1 terabyte(TB) = 1024 GB = $2^{40}$ bytes

# Combining bits to represent complex information

- Remember bit can only be 0 or 1.

- We can combine multiple bits to represent more complex data.
  - Text
  - Images
  - Sound
  - Video

  Etc.

# Representing Text

- Each character is encoded using 1 byte
- ASCII (***A**merican **S**tandard **C**ode for **I**nformation **I**nterchange*) table

Space: “ ”

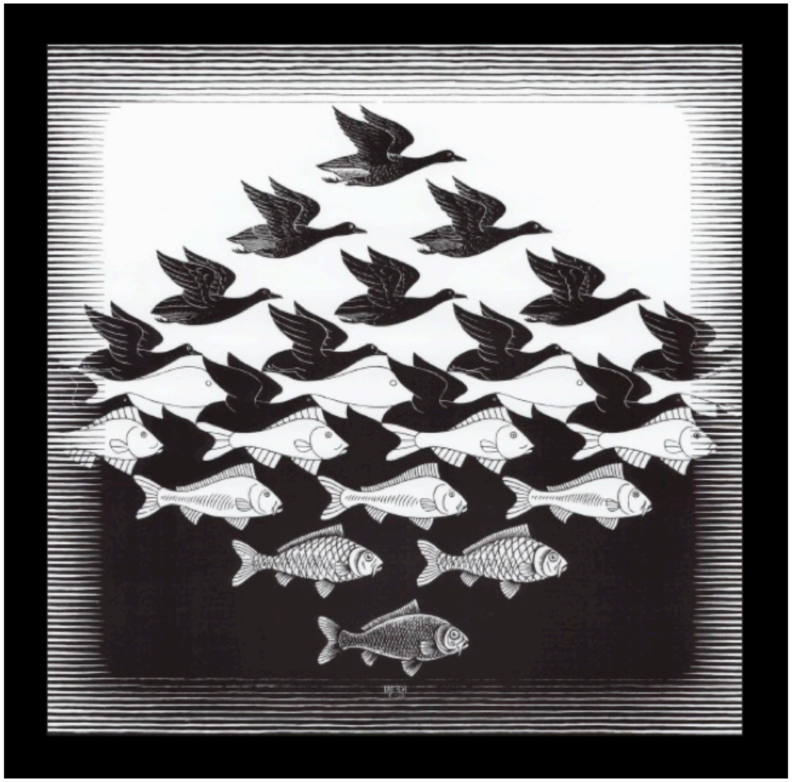
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 00 0000 0000 NUL | 01 0000 0001 SOH | 02 0000 0010 STX | 03 0000 0011 ETX | 04 0000 0100 EOT | 05 0000 0101 ENQ | 06 0000 0110 ACK | 07 0000 0111 BEL | 08 0000 1000 BS | 09 0000 1001 HT | 10 0000 1010 LF | 11 0000 1011 VT | 12 0000 1100 FF | 13 0000 1101 CR | 14 0000 1110 SO | 15 0000 1111 SI | 8 |
| 1 | 16 0001 0000 DLE | 17 0001 0001 DC1 | 18 0001 0010 DC2 | 19 0001 0011 DC3 | 20 0001 0100 DC4 | 21 0001 0101 NAK | 22 0001 0110 SYN | 23 0001 0111 ETB | 24 0001 1000 CAN | 25 0001 1001 EM | 26 0001 1010 SUB | 27 0001 1011 ESC | 28 0001 1100 FS | 29 0001 1101 GS | 30 0001 1110 RS | 31 0001 1111 US | 9 |
| 2 | 32 0010 0000 SP | 33 0010 0001 ! | 34 0010 0010 " | 35 0010 0011 # | 36 0010 0100 $ | 37 0010 0101 % | 38 0010 0110 & | 39 0010 0111 ' | 40 0010 1000 ( | 41 0010 1001 ) | 42 0010 1010 * | 43 0010 1011 + | 44 0010 1100 , | 45 0010 1101 − | 46 0010 1110 . | 47 0010 1111 / | A |
| 3 | 48 0011 0000 0 | 49 0011 0001 1 | 50 0011 0010 2 | 51 0011 0011 3 | 52 0011 0100 4 | 53 0011 0101 5 | 54 0011 0110 6 | 55 0011 0111 7 | 56 0011 1000 8 | 57 0011 1001 9 | 58 0011 1010 : | 59 0011 1011 ; | 60 0011 1100 < | 61 0011 1101 = | 62 0011 1110 > | 63 0011 1111 ? | B |
| 4 | 64 0100 0000 @ | 65 0100 0001 A | 66 0100 0010 B | 67 0100 0011 C | 68 0100 0100 D | 69 0100 0101 E | 70 0100 0110 F | 71 0100 0111 G | 72 0100 1000 H | 73 0100 1001 I | 74 0100 1010 J | 75 0100 1011 K | 76 0100 1100 L | 77 0100 1101 M | 78 0100 1110 N | 79 0100 1111 O | C |
| 5 | 80 0101 0000 P | 81 0101 0001 Q | 82 0101 0010 R | 83 0101 0011 S | 84 0101 0100 T | 85 0101 0101 U | 86 0101 0110 V | 87 0101 0111 W | 88 0101 1000 X | 89 0101 1001 Y | 90 0101 1010 Z | 91 0101 1011 [ | 92 0101 1100 \ | 93 0101 1101 ] | 94 0101 1110 ^ | 95 0101 1111 _ | D |
| 6 | 96 0110 0000 ` | 97 0110 0001 a | 98 0110 0010 b | 99 0110 0011 c | 100 0110 0100 d | 101 0110 0101 e | 102 0110 0110 f | 103 0110 0111 g | 104 0110 1000 h | 105 0110 1001 i | 106 0110 1010 j | 107 0110 1011 k | 108 0110 1100 l | 109 0110 1101 m | 110 0110 1110 n | 111 0110 1111 o | E |
| 7 | 112 0111 0000 p | 113 0111 0001 q | 114 0111 0010 r | 115 0111 0011 s | 116 0111 0100 t | 117 0111 0101 u | 118 0111 0110 v | 119 0111 0111 w | 120 0111 1000 x | 121 0111 1001 y | 122 0111 1010 z | 123 0111 1011 { | 124 0111 1100 \| | 125 0111 1101 } | 126 0111 1110 ~ | 127 0111 1111 DEL | F |

# Representing Text

| “M | A | R | C” |
|---|---|---|---|
| 1st byte | 2nd byte | 3rd byte | 4th byte |
| 77 | 97 | 114 | 99 |
| 01001101 | 01100001 | 01110010 | 01100011 |

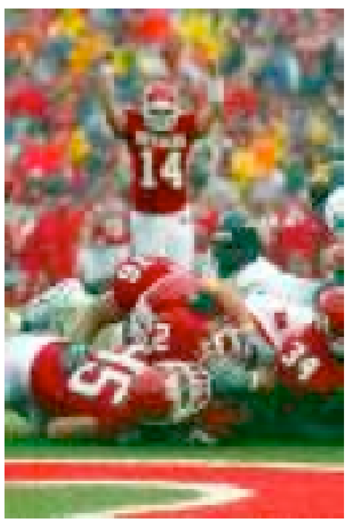
# Almost everything can be represented with bits

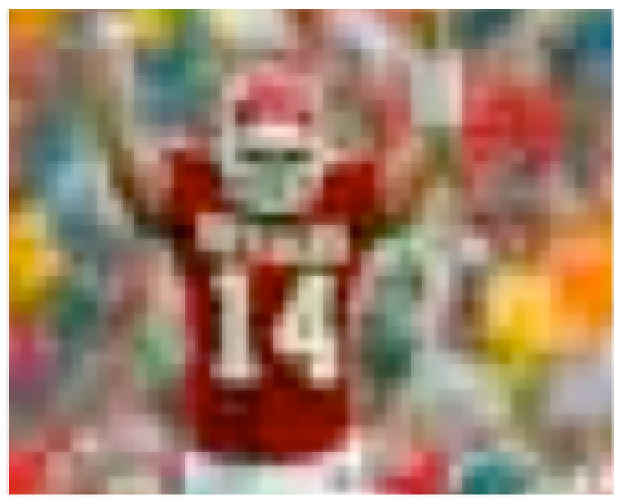
- Escher's drawing:
  - Use one bit to represent the colour (black=0, white=1) at each particular image location.



# Almost everything can be represented with bits

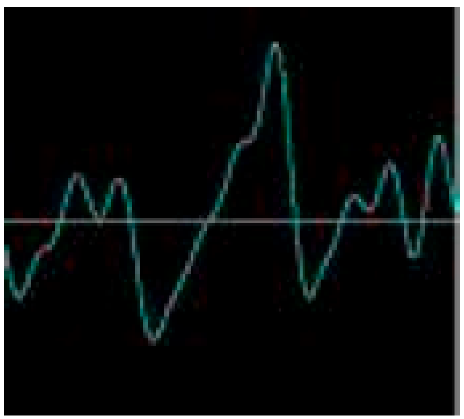
- Digital images:
  - A **group of bits** represents the colour at each particular image location: we call this a **pixel**.
  - An image pixel is one of Red, Blue or Green.
    - How do we encode this information with bits?
    - How many bits do we need?

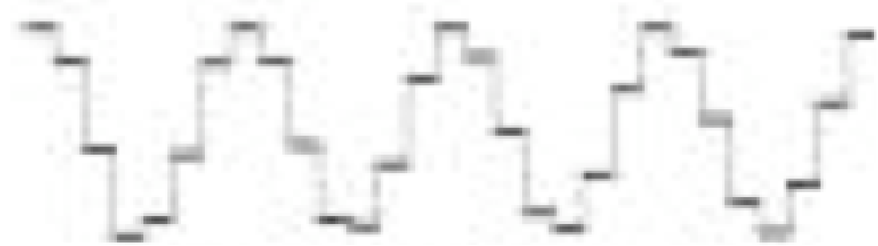


# Almost everything can be represented with bits

- Digital sound:
  - Average sound intensity (= a number) over a short time interval is represented using a group of bits.





# Modern technologies need lots of bits!

- Consider the iPod: 60Gb.
  - 1Gb = one billion bytes (1 byte = 8 bits).
- <u>Sound</u>:
  - 128Kbps per second of sound (Kbps = kilobits).
  - So 62,500 minutes of sound, or 15,000 songs (at 4min. per song).
- <u>Screen</u>:
  - 320x240(=76,800) pixels.
  - Each pixel needs 1 byte of RGB (=Red-Blue-Green) intensity.
  - At 30 frames per second, that’s 55.3 million bits per second.
  - So 144 minutes of (quiet) video.

# Logical variable

- Bits are not just for sound and images.
- Bits can store logical variables.
- A logical variable is something that we can imagine as being True or False.
  - *TodayIsThursday* = True
  - *ItIsDarkOutside* = False
  - *IAmTeachingCOMP102* = True
- *TodayIsThursday*, *ItIsDarkOutside* and *IAmTeachingCOMP102* are logical variables. They can be True or False.
- Logical variables are also sometimes called Boolean variables.

# And, Or, Not

- Logical variables can be combined with <u>logical operations</u>.

- The most important logical operations are **AND**, **OR**, and **NOT**.
  1. *x* **AND** *y* is True only if both *x* is True and *y* is True.
  2. *x* **OR** *y* is True if either *x* or *y* are True.
  3. **NOT** *x* is True only if *x* is False.

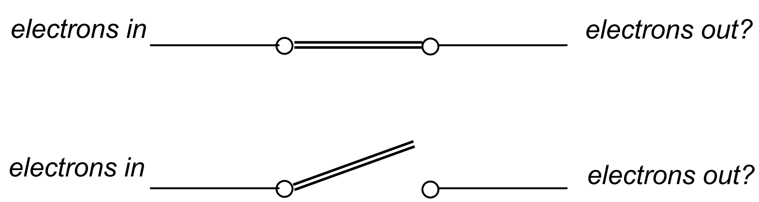
- Logical operations have the intuitive English meaning.

# Logical expressions

- Logical expressions combine logical variables and logical operations into more complex expressions.
  - *IAmTeachingCOMP102* **AND** *ItIsDarkOutside* = False
  - **NOT** *ItIsDarkOutside* = True
  - *IAmTeachingCOMP102* **OR** *TodayIsThursday* = True
  - ( *TodayIsThursday* **OR** *IAmTeachingCOMP102* ) **AND** ( *IAmTeachingCOMP102* **AND** ( **NOT** *ItIsDarkOutside* ) ) = ??

# Implementing logic

- How do we implement a logical variable?
  - Easy!  One switch per logical variable




Closed switch = True

Open switch = False

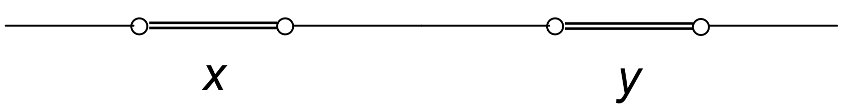
- How do we implement logical operations?

- How do we implement logical expressions?

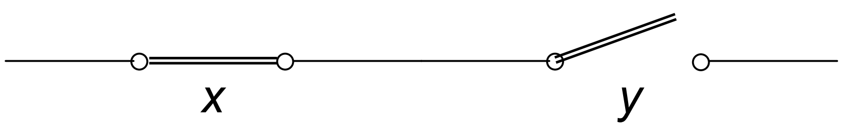
# Implementing logical operations

**Key Idea:** Combining switches

- **AND** operation: Combine switches in series



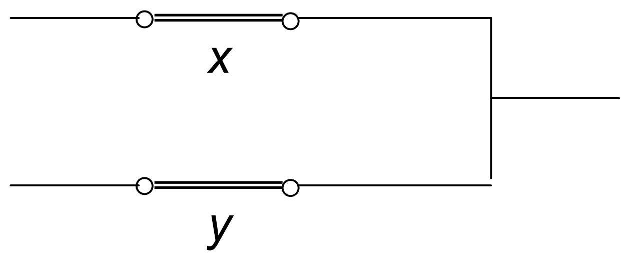

($x$ = True) **AND** ($y$ = True) = True



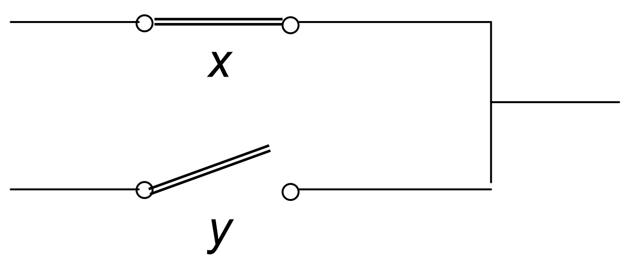

($x$ = True) **AND** ($y$ = False) = False

# Implementing logical operations

- **OR** operation:  Combine switches in parallel




(*x* = True) **OR** (*y* = True) = True




(*x* = True) **OR** (*y* = False) = True

- **NOT** operation is slightly more complicated.

# Implementing logical expressions

- Combine multiple switches.

E.g. ( *TodayIsThursday* **OR** *IAmTeachingCOMP102* ) **AND**

( *IAmTeachingCOMP102* **AND** ( **NOT** *ItIsDarkOutside* ) ) = ??

# Practice example

- Three friends are trying to decide what to do Saturday night (see a movie or go out clubbing). They settle the issue by a vote (everyone gets a single vote, the activity with the most votes wins.)
- Assume you want a computer to automatically compile the votes and declare the winning activity.
- What logical variables would you use?
- Can you write a logical expression, which evaluates whether or not you will go Clubbing (True = Clubbing, False = Movie)?

# Take-home message

- Understand the concept of a **bit**.
- Know how to combine multiple bits to represent complex information (text, images, sound, video).
- Understand what are **logical variables**.
- Know the three basic **logical operations**.
- Be able to evaluate **logical expressions**.

# Final comments

- Some material from these slides was taken from:
  - *http://www.cs.rutgers.edu/~mlittman/courses/cs442-06/*
  - *http://cim.mcgill.ca/~sveta/COMP102/*