## Layers of a Computing System

## Communication

## Application

## Operating System

## Programming

## Hardware

## Information

## Abstraction

* A way to think about something that removes or hides complex details, i.e., only the information necessary to accomplish our goal
* When we are dealing with a computer on one layer, we don’t need to be thinking about the details of the other layers
* We rely on abstractions every day…
  + How many know the details of the car’s engine??
  + Dashboard is an abstraction of the engine
* When running a program we don’t have to be aware of the operating system which gives our program memory and processor time, or the programming involved, or the hardware, or how data is represented

## Analog and Digital Information

* Natural World – continuous and infinite
* Computers – finite
  + Memory and other devices have only so much room to store and manipulate a certain amount of data
  + Always fail to represent an infinite world on a finite machine
  + Digitize information by breaking it into pieces and representing those pieces separately
  + Goal:
* All data stored in a computer is represented using only two symbols – 0 and 1!



0

10

20

30

40

50

60

70

80

90

100



0

10

20

30

40

50

60

70

80

90

100

## Data and Computers

* Electrical Devices

DD00943_DD00943_DD00943_DD00943_

* Magnetic Devices

N

N

N

N

N

N

N

N

## Encoding Information with 1s and 0s

* Bit – unit capable of storing one of two values, 0 or a 1

Pass or fail course



Suppose we want to store letter grade – A, B, C, D or F?

We can encode information in different ways – some better than others.

Letter grade in course (stored in 5 bits)

0 0 0 1 0

0 1 0 0 0

0 0 1 0 0

1 0 0 0 0

0 0 0 0 1

Can you come up with a different encoding, a different pattern of 1s and 0s, to represent your grade that uses fewer bits?

## Representing Natural Numbers

* Base 10 (decimal)

2 1 9 3

* Base 2 (binary)

1 0 1 1 1

*There are 10 kinds of people in the world: those who understand binary, and those who don't.*

* Use same ideas to represent any base
  + Base n: 0 🡪 n-1 digits
  + Positional value: n0, n1, n2, n3, …
* Conversions:
  + base 2 🡪 base 10
  + what about base 10 to base 2??

### Let’s convert 47 (base 10) to base 2

20 = 1

21 = 2

22 = 4

23 = 8

24 = 16

25 = 32

26 = 64

**Practice Problems and Thought Questions**

In your groups, get some practice with base 2 and other numbering systems by answering the following questions. Make sure everyone in your group understands the problems and the answers you come up with.

1) Convert the following binary (base 2) numbers to decimal (base 10)

1101

10

10101011

2) Convert the decimal value 53 to binary.

3) If you have only four bits to store a number, what is the largest value you can store?

4) Challenge: Test your understanding of the base 10 and base 2 numbering systems by answering the following questions about the base 8 numbering system.

a) How many digits would you use in base 8?

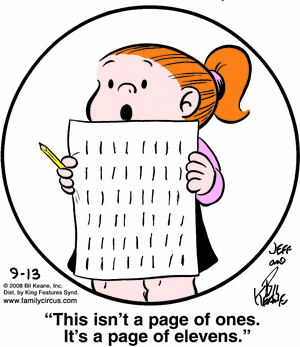
b) The rightmost digit in a base 8 number is the 80 = 1s place. What is the place of the second digit from the right? What about the third?

c) Convert 205 (base 8) to base 10?

## Other Numbering Systems

* Base 8 (octal)

4 0 1 7 3



* Base 16 (hexadecimal)

#### 4 F A 7 2

*Question: Why do computer nerds confuse Christmas and Halloween?*

*Answer: Because* ***DEC 25= OCT 31!***

## Storage Units

* Bit
* Byte
* Whole numbers are typically stored in 16 or 32 bits
* Largest value you can store in a 32 bit integer

**…**

* Smallest value you can store in a 32 bit integer

**…**

## Signed Magnitude Notation

* Leftmost bit is used for sign: 1 for negative, 0 for positive
* Need to know number of bits for each number:
  + ex: to represent +5 with 6 bit signed magnitude notation:

0 0 0 1 0 1

* + how about +53 with 8 bit signed magnitude notation?
  + how about -53 with 8 bit signed magnitude notation?
  + how about -1 with 8 bit signed magnitude notation?
* How many different numbers can we represent with 3 bits using signed magnitude notation?
* How about 32 bits? (size of integer in many machines)

## Storing Text on a Computer

* Text is stored as a series of characters. Each character is represented by its own special pattern of 0s and 1s.

01001000011010010010000001001101

011011110100110100100001

## Text Compression

* *Keyword Encoding*

“The system must not only work independently, they must interact and cooperate as well.”

%

well

$

that

#

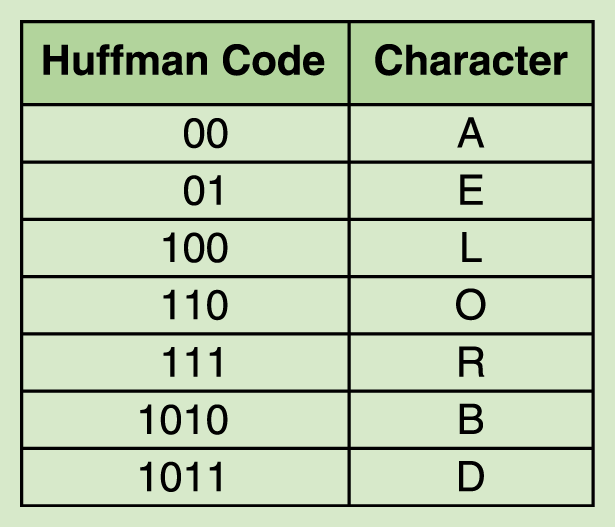
those

&

must

+

and

* *Huffman Encoding*
  + Why should the character “X”, which is seldom used in text, take up the same number of bits as the blank, which is used very frequently?
  + Huffman codes using variable-length bit strings to represent each character.
  + A few characters may be represented by five bits, and another few by six bits, and yet another few by seven bits, and so forth…
  + If we use only a few bits to represent characters that appear often and reserve longer bit strings for characters that don’t appear often, the overall size of the document being represented is smaller.
  + **For example:
  + DOORBELL would be encode in binary as

1011110110111101001100100

* + If we used a fixed-size bit string to represent each character (say, 8 bits), then the binary from of the original string would be 64 bits.
  + The Huffman encoding for that string is 25 bits long, giving a compression ratio of 25/64, or approximately 0.39.
  + An important characteristic of any Huffman encoding is that no bit string used to represent a character is the prefix of any other bit string used to represent a character.

## Representing Color

* Human retinas have 3 types of color receptors
* Painters Colors
* RGB Colors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **R** | **G** | **B** | **Color** | **Hexidecimal** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Images

## Macintosh HD:Users:meadmin:Pictures:Rebuilt Library:Previews:2011:07:04:20110704-163933:IMG_0446.jpg

## Audio

## MP3 Format

* In addition to regular compression techniques, MP3 format reduces data by:
  + Eliminating sounds humans cannot hear
  + Eliminating a soft sound if it is playing at the same time as a loud sound
* Compression ratio:

What happens when integers overflow?

https://www.youtube.com/watch?v=z-r9cYp3tTE