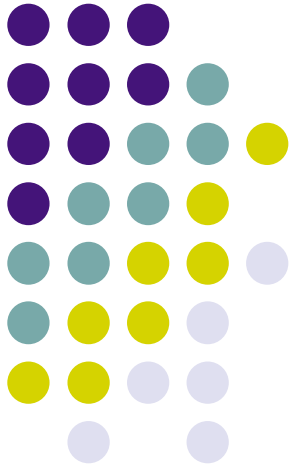
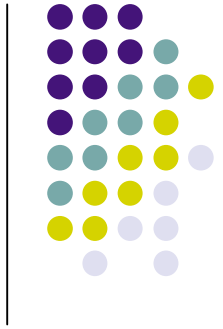
Bits and Bytes

Computer Literacy Lecture 4

29/09/2008



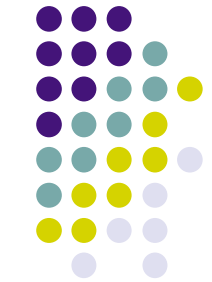
Lecture Overview



- Lecture Topics
  - How computers encode information
  - How to quantify information and memory
  - How to represent and communicate binary data

  The aim is to be able to reason quantitatively about computer systems

Computers encode information

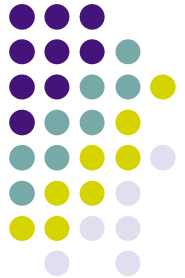


- What is information?

➔ Very difficult to define!

  - communication that has value because it informs?
  - just raw data?
  - semantical **content**?

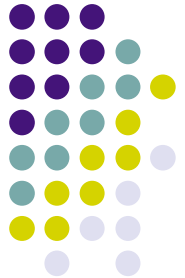
Syntax versus Semantics



- Language has two levels:

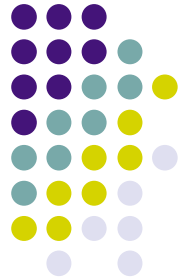
1. syntax: notation, sign, symbol
2. semantics: meaning, content, interpretation

Relation between the Two



- The sign denotes or refers to the content
- Normally the meaning or content itself can’t be displayed just using syntax - which is why dictionaries often use pictures
- The same meaning can be denoted by many different signs: e.g.”2” and “II” both refer to the same number as the English word “two”

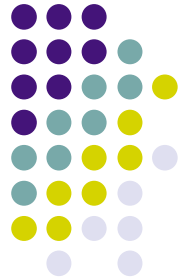
What is Information?



- The information carried by a sign is its interpretation or meaning, i.e. the **semantics**
- But the semantical level itself is quite ellusive
- So computers really just encode and manipulate **syntax** - we give it a meaning and call it information

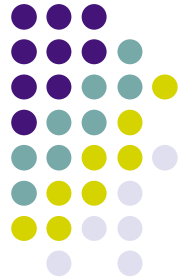
➔ How do they do it?

Modern Computers are Digital



- Computers are built out of many electric switches
- Each switch is “on” or “off” at any time
- Advantages
  - Robust to errors
  - Fast
- Alternative is analogue codes e.g. along phone lines

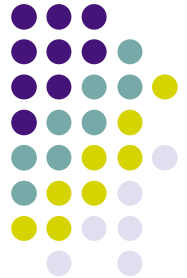
Analogue vs Digital



- Analogue values are continuous, and are often represented using gauges and dials: e.g. weight, temperature, time of day, voltage
- There is always a margin of error or degree of approximation when reading analogue values.
  - When does the scale read exactly 3 pounds, 2 ounces?

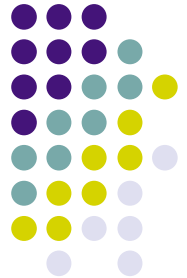
Because continuous, analogue can be read to an arbitrary level of precision

Analogue vs Digital



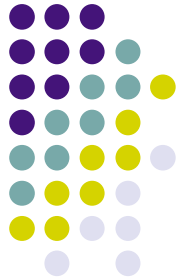
- Digital values are discrete, as in numerical digits.
  - Hence can be represented in simple numerical terms.
    - On a digital watch display the time increases by discrete steps of 1 second - between seconds there is no way of measuring what fraction of a second has passed

Analogue vs Digital



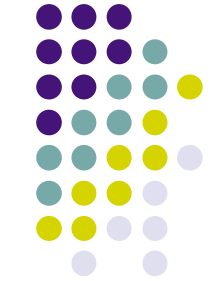
- There is always a degree of idealisation in reading digital values, since physical magnitudes are actually continuous (above the quantum level!)
  - There is a range of real weight values that will all cause the digital scale to read “exactly” 3 pounds, 2 ounces.

Analogue vs Digital



- Many of the early computing devices used analogue representations
  - e.g. differential analyzers for solving differential equations, slide rules (pre-digital pocket calculators)
- But virtually all contemporary computers are based on manipulating voltages represented in digital terms, i.e. discrete “bits”

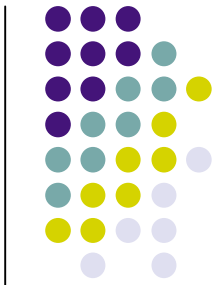
Analogue vs Digital



- Advantages of digital over analogue:

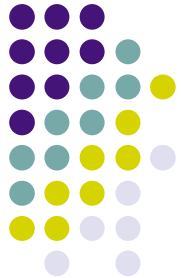
1. **It is fast**: much quicker to decide if a switch is “on” or “off” than to decide how much it is on or off.
2. **Robust to errors**: small errors at each switch in the computer are not propagated. If a digital switch is still a little bit on when it should be off, then the signal coming from it will still be treated as “off”

Bits of information



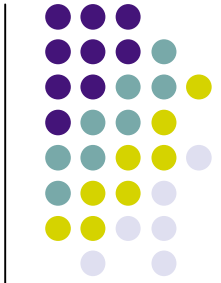
- A ***bit*** is one unit of “information”
- Comes from **bi**nary digi**t** ~1948
  - bigit and binit were considered!
- Each bit has one of two values e.g. 1 or 0
- The bit is represented by the number, it is not the number
- Could be represented by True or False, Yes or No, George or Saddam, any binary scheme

Bytes



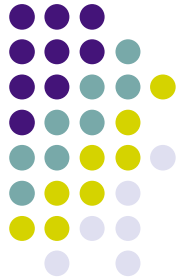
- ***Byte***
  - short for **b**inar**y** **te**rm
  - coined 1956
- Mutation from bite to byte ~1956
- 1 Byte = 8 bits
- 1 Byte is the minimum unit of memory that can be accessed by standard computers
- Used to measure memory, size of files capacity of file-space

Byte Size



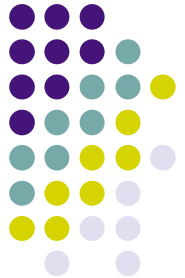
- One byte can express 256 different possibilities, because
- Each bit can have one of two values:
  - for 2 bits there are 2x2 possibilities (00, 01, 10, 11).
  - With 3 bits there are 2x2x2 possibilities (000, 001, 010, 011...100, 101, 110, 111).
  - With 8 bits there are 2x2x2x2x2x2x2x2 = 256 possibilities

Big Bytes



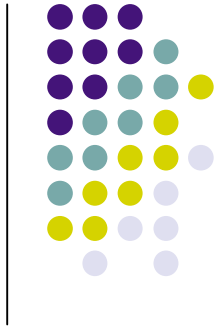
- 2^{10} bytes = 1024 bytes ~1000 bytes
- 2^{10} bytes = 1 kilobyte = 1KB
- 1 KB **is not** 1000 bytes

Bigger Bytes



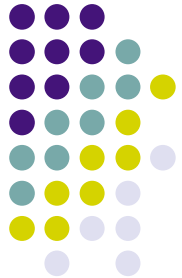
- 1 Megabyte (MB) = 1,048,576 bytes = 2^{20}bytes ≈ 10^{6}bytes = 1024 KB
- 1 Gigabyte (GB) = 1,073,741,824 bytes = 2^{30}bytes ≈ 10^{9}bytes
- 1 Terabyte (TB) = 2^{40}bytes ≈ 10^{12}bytes
- 1 Petabyte (PB) = 2^{50}bytes ≈ 10^{15}bytes
- Followed by exabyte, zettabyte and yottabyte!

How big is big?



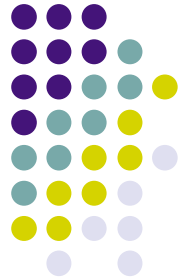
- 300 page novel
- Floppy Disk
- 3 minute MP3 track
- Edinburgh telephone directory
- CD
- DVD
- Corporate customer database
- Video of your life
- Human brain

How big is big?



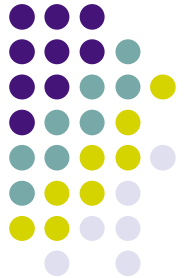
300 page novel	→ 1 MB
Floppy Disk	→ 1.44 MB
3 minute MP3 track	→ 2 MB
Edinburgh telephone directory	→ 20 MB
CD	→ 600 - 700 MB
DVD	→ 4 GB
Corporate customer database	→ 1 TB
Video of your life	→ 1 PB
Human brain	→ > 10 PB

Expressing characters



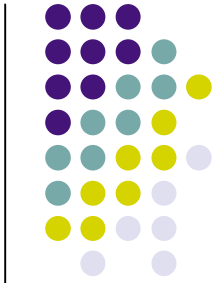
- Computers process bits of information
- We process language through characters or notation
- Conventions define how characters are expressed in bits, i.e. how computers encode syntax
- E.g. ASCII, Unicode (used by Java), ANSI, ISO Latin

ASCII



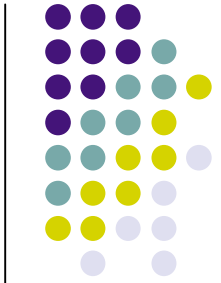
- **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
- A → **0**1000001 ? → **0**0111111
  a → **1**1100001 9 → **0**0111001
- Represents 128 characters
- 8th bit in front allows some error detection
  - Codes “parity” of the byte, whether byte is odd or even
  - If one bit is corrupted, parity will change

Other conventions



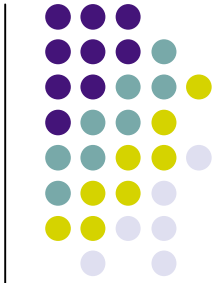
- ASCII (7 bits) - 128 characters
- ISO Latin 1 (8 bits) - 256 characters
- Unicode (16 bits) - 65536 characters
- E.g. In Japanese, 1,945 ideogram characters in standard use
- Unicode designed to encode any language character
  - But doubles memory demands
  - Requires compatibility of software

Hexadecimal (Hex)



- Binary is counting in 2s (twos)
- Normally we count in 10s (tens)
- Hexadecimal is counting in 16s (sixteens)
- Bits can be expressed in hexadecimal with no loss of generality
- *Easier for humans to read!*
- Because bytes can be expressed using two digits in hex (one digit for the first 4 bits, one for the next 4 bits) it's a very widely used system for binary coding

Counting in Hex

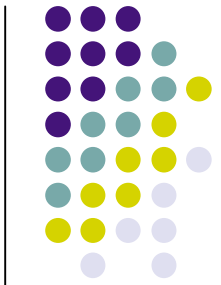


- The sixteen digits of Hex are

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- When you get to the numeral for ten, if you used “10”, it would be interpreted in Hex as one unit of sixteen plus zero ones, which equals 16 in decimal notation.
- The original choice of using letters for the numbers from ten to fifteen was arbitrary, but this is now the convention

Keep Counting!



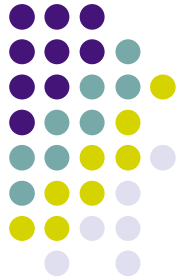
- "10" in Base 16 equals "16" in Base 10. If we keep going in Hex we get: **10,11,12,13,14,15,16,17,18,19,1A,1B,1C,1D, 1E,1F**
- Meaning:

1A (Base 16) = **1**x16 + **10**x1 (Base 10) = 26

1B (Base 16) = **1**x16 + 11x1 (Base 10) = 27

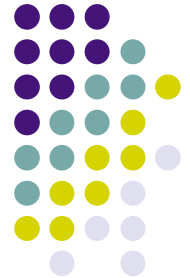
AB3 (Base 16) = **10**x16^2+ **11**x16^1+ **3**x1 (Base 10) = 2733 (Base 10)

Coding Codes in Base 16



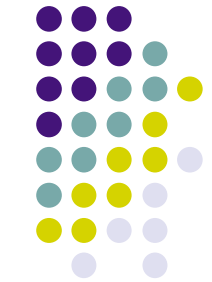
- As we saw before, ASCII uses binary notation to code standard keyboard characters
- In turn, Hex is used to code binary numbers
- Thus we can go from keyboard characters to Hex via ASCII coding

Example



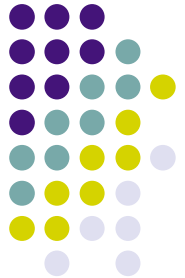
- The ASCII character code for “A” is **0**1000001
- Hex coding uses one digit for the number denoted in binary by the first four bits and another for the number denoted by the second four bits
- By breaking it up we first get 0100, which equals four in binary (one unit of two squared), which is just denoted by “4” in Hex
- The second half is 0001, which is 1 in binary and Hex
- The ASCII code **0**1000001 for “A” is coded as 41 in Hex

Another Example



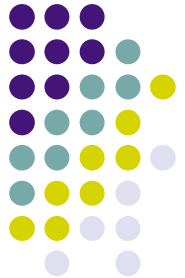
- The ASCII code for “?” is 00111111
- Breaking this up we first get 0011, which equals three in binary (two + one), which is denoted by “3” in Hex
- The second half, 1111, denotes fifteen in binary (8+4+2+1), which is denoted by “F” in Hex
- So the ASCII code 00111111 for “?” is coded as 3F in Hex

Color Codes



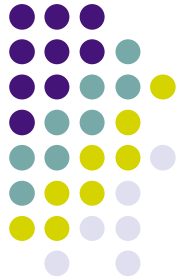
- An example of Hex in practice is the “non-dithering RGB color codes”
- “Dithering colors” look different on different web browsers and are a common fault of many websites
  - http://en.wikipedia.org/wiki/Dithering
- Colors defined by their red green blue (RGB) code look the same using any browser

Non-dithering



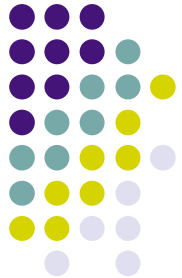
- The “non-dithering RGB color codes” are binary codes expressed in Hex
- Each code is 3 bytes long, and each byte is expressed using two digits in Hex
- E.g. the code 0xCC3399 expresses one byte of information about the level of red (0xCC), one byte about the level of green (0x33), and one byte about the level of blue (0x99).
- http://www.computerhope.com/htmcolor.htm#02

MIME



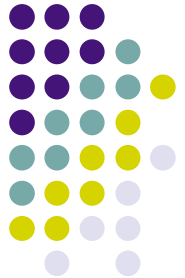
- **M**ultipurpose **I**nternet **M**ail **E**xtensions
- Uses Base64 to convert binary data into “safe” ASCII
- Industry standard that describes how
  - emails must be formatted so the receiver can interpret the email
  - how non-text (pictures, audiofiles, etc) are converted into ASCII

MIME



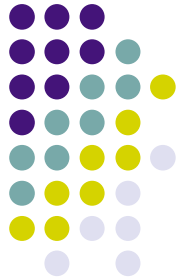
- Takes 3 bytes at a time (24 bits), expresses each block of 6 bits in ASCII, result is four bytes of “safe” ASCII code
- File size is increased by 33% as each block of 6 bits is expressed by 8 bits, but encoding benefits from ASCII parity checks.

Key Points



- Computers process bits very quickly
- Hex allows the user to talk in binary
- Characters are encoded by the computer as numbers, using e.g. ASCII
- Non-text requires encoding

Question



Can you recognise encoded information?