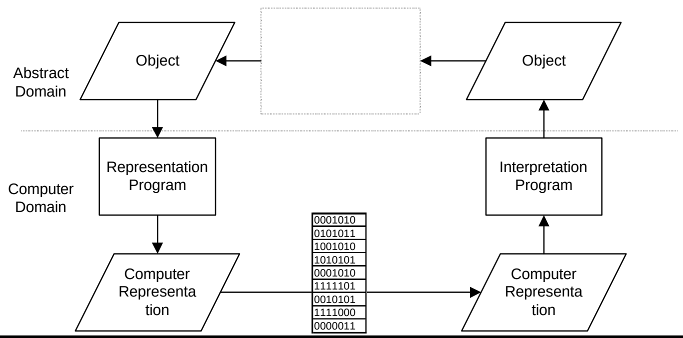
Outline: Representations

- Representation and Abstraction
- Representations in Binary Computers
  - Fractions
  - Characters
  - Character Strings
  - Formatted documents
  - Images and drawings
  - Postscript
- Representation Transformations
  - Compression

1

The Representation Game

- A representation program translates an object into a computer representation (= a set of binary numbers)
- An interpretation program translates a computer representation back into its equivalent object
- Representation and interpretation are inverses
- Representations are also called data structures



2

Why Study Representation Internals?

- To select the best representation for the task at hand
  - Multiple representations possible for each object
  - Each representation is best suited to different tasks
- To understand what is going on when system breaks down
  - That's when the internals show-through

3

Examples of Representations

- Fractions
- Letters
- Text strings
- Formatted text documents
- Images and drawings

4

Fractions: fixed point

- Invisible decimal point
- Bits to left of decimal point interpreted as whole number
- m bits to right of decimal point interpreted as fraction with 2^m as denominator
- Example: 8 bits: m=4; whole number uses 2's complement
  - Represent 3.75 as
  - Interpret 1001 0100 as

5

Fractions: floating point

sign (1 bit)	Exponent (m bits)	Significand (n-m-1 bits)

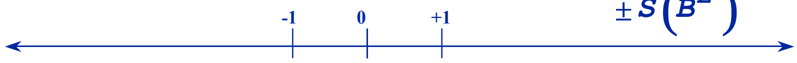
- Interpret as $\pm S\left(B^E\right)$
- Base B is implied, not encoded at all
  - Just as location of decimal point is implied in fixed point representation
- Sign, S, and E encoded in the n-bit word
  - Normally, E encoded in 2's complement
- Examples: suppose n=12; m = 4; n–m–1=7; B=10
  - Interpret 0 0100 0001111
  - Represent –0.16 as

6

Precision vs. Range

- How many numbers can be represented with 32 bits?
- How many different fractions are there?
  - I(R(X)) can’t always be X!
- Precision: distinguish numbers “close” to each other
- Range: allow very small and very large numbers
- Ways to increase precision
  - decrease B
  - devote more bits to significand
- Ways to increase range
  - increase B
  - devote more bits to exponent

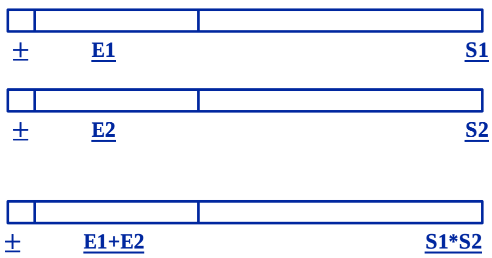
$\pm S\left(B^{E}\right)$



7

Floating Point Multiplication

- Example of implementing an abstract function



x = ± **S1 (10^{E1})**

y = ± **S2 (10^{E2})**

x*y = ± **S1*S2 (10^{E1+E2})**

add two integer exponents

multiply two integer significants

8

Representing Letters

- Play the representation game again
  - Represent each letter by a number (e.g., A = 0; B = 1)
  - We already know how to represent numbers in binary
  - To display a letter, computer translates back from binary word
- ASCII representation
  - space––>32; ‘2’––>50; ‘9’––>57; ‘A’––>65;’ Z’––>90; ‘a’––>97; ‘z’––>_____
  - 256 characters, only some of which actually appear on keyboard or screen
    - for example, the newline character is 10
  - Notation: when we mean a character, enclose in single quotes
    - '2' is a character whose representation is 50
    - 2 is a number; interpret as the control-B character
- Other representations are possible

9

Character strings

- How many bits needed to represent one ASCII letter?
- A collection of adjacent bytes (8-bit groups) can store a sequence of letters

  `'H' 'e' 'l' 'l' o' ' ' 'W' 'o' 'r' 'l' 'd' '\0'`
- Notation: we enclose character sequences in double quotes
  - "Hello world"
- Representation convention: null character defines end of string
  - null is sometimes written as '\0'
  - Its ASCII representation is the number 0
  - Some operations are easy to implement
    - E.g., find length of string
  - Other operations are hard with this representation
    - E.g., divide string in two after the fourth character

10

Formatted Documents

- In addition to text, must contain information about how it appears on paper
  - bold, italic, underlined text
  - different sizes of type
  - page breaks
- “Invisible” formatting characters are embedded in text
  - special “begin formatting” character
  - format specification character (i.e. “bold type”)
  - text string for which formatting applies
  - special “end formatting” character
- Same character codes have different meaning when interpreted as letters and when as format specifications
  - 65 could mean both ‘A’ and ‘bold’ depending on context

11

Formatted Documents (cont’d)

Example:

- This is a <u>nicely</u> formatted line.

Would be stored internally as:

- <BG PAR> ‘T’ ‘h’ ‘i’ ‘s’ ‘ ’ ‘i’ ‘s’ ‘ ’ ‘a’ <BG UNDERLINE> ‘n’ ‘I’ ‘c’ ‘e’ ‘l’ ‘y’ ‘ ’<EN UNDERLINE> <BG COLOR> 1 ‘f’ ‘o’ ‘r’ ‘m’ ‘a’ ‘t’ ‘t’ ‘e’ ‘d’ <EN COLOR> ‘ ’ ‘l’ ‘i’ ‘n’ ‘e’ ‘.’ <CR>

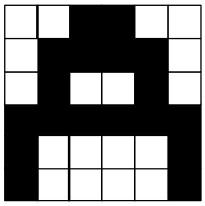
Where:

- <BG PAR>, <BG UNDERLINE>, <EN UNDERLINE>, <BG COLOR>, <EN COLOR>, <CR> are special byte sequences that denote the beginning and end of various formatting features
- Different word processors use different byte sequences, that’s why documents require conversion to be used by a different wp

12

Bitmapped graphics

- Representing a picture
  - Draw a very fine grid on it
    - grid cells are called pixels or dots
  - See what is in each grid cell
    - bitmap: is cell empty or full?
    - grayscale: how dark is the cell?
    - color: what color is the cell?
  - Represent each cell with a prespecified # of bits (how many?)
  - Store the bits for the cells in a prespecified order
    - e.g., all the cells for the top row, then the next row, etc.

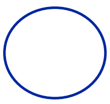


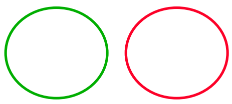
- Interpreting an image representation
  - What is needed to interpret a sequence of bits as an image?

15

Vector graphics

- We can do better than images for computer-generated pictures
  - e.g., MacDraw, SuperPaint
- Keep track of the shapes used
  - Represent the dimensions and position in drawing of each rectangle, line, etc.
    - e.g., circle centered at (79,95) with radius 150 and 1 pixel wide boundary
- Why is this better?
  - More compact representation
    - An image of a large circle encodes every pixel





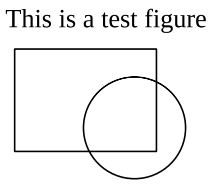
  - Useful operations are easy
    - E.g., moving a circle
  - prints better
    - a screen image is printed at screen's resolution
    - a "draw" representation is printed at printer's resolution, displayed at screen's

16

PostScript/PDF

- The best known Page Description Language
- Describes a page's contents with text

```
3.49121 0. 32 0.34912 0.(This is a test figure)awidthshow
64 gr
60 48 131 146 1 rc
0 gr
60.5 48.5 130.5 145.5 0 rc
64 gr
79 95 150 166 1 ov
0 gr
79.5 95.5 149.5 165.5 0 ov
```



- Why text rather than binary representation?
  - Humans can read text
    - Useful in debugging
  - Can send documents via electronic mail

17

Selecting between representations

- Different representations best suited to different operations
  - storage and transmission
  - processing
  - display
  - transfer to a different kind of computer
- Often it makes sense to translate between two representations

18

Compression

- Prefer shorter representations to longer
  - For storing or transmitting data
- Three basic techniques
  - Encode high probability symbols with fewer bits
    - Shannon-Fano, Huffman, UNIX compact
  - Encode sequences of symbols with location of sequence in a dictionary
    - LZ77, LZ78, QIC, PKZIP, ARC, GIF, UNIX compress, V.42bis
  - Lossy compression
    - JPEG and MPEG
- Further Reading
    - The Data Compression Book, by Mark Nelson, 1992 M&T books

19

Variable Length Bit Codings

- Suppose you know letter ‘A’ will appear 50 times in text, but ‘B’ will appear only 10 times
- ASCII coding assigns 8 bits per character, so total bits for ‘A’ and ‘B’ is 60 * 8 = 480
- If ‘A’ gets a code that’s only 4 bits and ‘B’ gets a code that’s 12 bits, total is 50 * 4 + 10 * 12 = 320
  - Representation game requires that same scheme be used for decoding as coding!
- Why not assign 4-bit codes to all letters?
- Can also employ context-sensitive coding
  - ‘u’ is unlikely
    - normally it’s code should have many bits
  - after a ‘q’ it’s very likely
    - it’s code should have few bits
  - Decompression has to take account of context in same way!

20

Example: Huffman coding

- Assume only four letters: A through D

60%	A	A: 0
30%	B	B: 10
5%	C	C: 110
5%	D	D: 111

Compression efficiency

- Fixed length encoding: 2 bits per character
- Variable length encoding: 0.6*1+0.3*2+0.1*3=1.5 bits/character
- Compressed file has 1.5/2 = 75% the size of original file
- What types of files can be compressed more efficiently using this technique?

Exercise: Huffman coding

- Assume eight letters: A through H

40%	A	`A: 0`
20%	B	`B: 110`
15%	C	`C: 100`
9%	D	`D: 101`
8%	E	`E: 1110`
5%	F	`F: 11110`
2%	G	`G: 111110`
1%	H	`H: 111111`

23

Dictionary Based Codings

- Keep a dictionary of common words and phrases
- Translate symbol string in input to a location in the dictionary
- Example:
- *"Ask not what your country can do for you -- ask what you can do for your country."* (JFK)

24

Compressing JFK...

"Ask not what your country can do for you -- ask what you can do for your country." (JFK)

- "ask" appears two times
- "what" appears two times
- "your" appears two times
- "country" appears two times
- "can" appears two times
- "do" appears two times
- "for" appears two times
- "you" appears two times

25

Compressing JFK...

- Dictionary:

1. ask
2. what
3. your
4. country
5. can
6. do
7. for
8. you

- Compressed sentence:

"1 not 2 3 4 5 6 7 8 -- 1 2 8 5 6 7 3 4"

26

Graphics Compression: GIF

- Lossless compression
- Identifies strings of identical pixels and replaces them with: (count) (pixel value)
- Best for line drawings

27

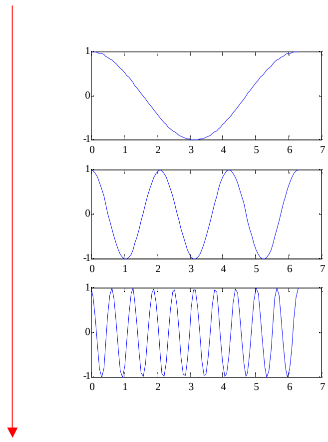
Graphics Compression:JPEG

- JPEG
  - Lossy compression
  - Based on Discrete Cosine Transformation
  - Amount of compression can be chosen by user
  - Best for photographs
- See Webopedia for details!

28

JPEG Overview

- Images contain different frequencies; low frequencies correspond the slowly varying colors, high frequencies correspond to fine detail.
- The low frequencies are much more important than the high frequencies; we can throw away some high frequencies to compress our data!



29

JPEG summary

- We break up the image into 8x8 blocks.
- We calculate the frequencies in each block, this allows us to identify the important and less important data.
- We throw away some less important data.
- We compress the resulting data (using Huffman coding)
- The result: ~ 1:40 compression!

30

Video Compression

- MPEG formats
  - There is a lot of similarity between one frame and the next
  - Only encode the difference between successive frames
  - Can achieve compression ratios of 30 or more
  - Used to encode movies in DVD
- MP3
  - Audio layer compression scheme in MPEG
- See Webopedia for details!

31

Looking Back

- All objects computers use must be represented as sequences of binary integers
- Same abstract object can have multiple representations -- select best for task at hand
- Data Compression techniques play an increasingly important role in our data hungry society

32