Attendance checking

When you swipe your student ID card, it says how many times you have attended. You can keep track of this yourself.

Install party

If you are interested in installing the software necessary/useful for your course of studies such as

- Python, Java, and programming environment
- R, Maple, Matlab, and other mathematical software
- Linux (Ubuntu, Debian, ...) and other operating systems
- anything else relevant (LaTeX)

or you want to HELP OTHERS to do this, sign up (at http://goo.gl/HIiOkH, by Friday at 12:00) for the install party on Tuesday, September 15, 14-17 in the Friday bar "Nedenunder".

There will be a number of volunteers (teaching assistants and study group supervisors), but the idea is also to help each other and exchange experience. So come also if you are already set up. There should be pizza and soft drinks.

Required video.
http://www.sdu.dk/Om_SDU/Beredskab_paa_SDU/Informationsmateriale/Beredskabsfilm

Excess notation

with 4 bits, bias 8 (p.62)

1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	-1
0110	-2
0101	-3
0100	-4
0011	-5
0010	-6
0001	-7
0000	-8

How do you get the value?

Excess notation

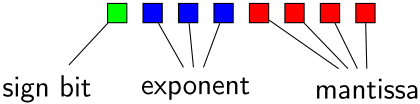
with 4 bits, bias 8 (p.62)

1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	-1
0110	-2
0101	-3
0100	-4
0011	-5
0010	-6
0001	-7
0000	-8

subtract bias to get value

Floating point

Textbook doesn't use implicit leading bit (you should)



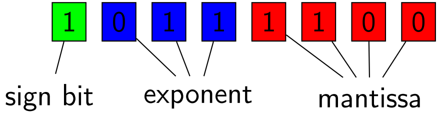
exponent — excess notation, bias 4

111	3
110	2
101	1
100	0
011	-1
010	-2
001	-3
000	-4

p. 63

Floating point

Textbook doesn't use implicit leading bit (you should)



mantissa — implicit leading bit

It is really 5 bits, with the first bit 1. 1100 $\rightarrow$ 1.1100

sign — negative

exponent — 011 $\rightarrow$ -1

$-(1.11 \cdot 2^{-1}) = -\frac{7}{8}$

Floating point

$$1\frac{1}{8}$$

mantissa — 1.001 $\rightarrow$ 0010
exponent — 0 $\rightarrow$ 100
result — 01000010

How do we represent $2\frac{5}{8}$? (Note: can't in book.)

A. 01010101
B. 00101010
C. 01011101
D. 00111010

Vote at m.socrative.com. Room number 415439.

Floating point

$$1\frac{1}{8}$$

mantissa — 1.001 $\rightarrow$ 0010
exponent — 0 $\rightarrow$ 100
result — 01000010

How do we represent $2\frac{5}{8}$? (Note: can't in book.)

[A.] 01010101

Floating point

$$
\begin{aligned}
4\tfrac{5}{8} &= 100.101 \\
\text{exponent} = 2 &\rightarrow 110
\end{aligned}
$$

result — 01100010

Last bit is truncated. $4\frac{5}{8} = 4\frac{1}{2}$?
$(4\frac{1}{2} + \frac{1}{8}) + \frac{1}{8} = 4\frac{1}{2}$?
$4\frac{1}{2} + (\frac{1}{8} + \frac{1}{8}) = 4\frac{3}{4}$?

Truncation errors and reducing them
— numerical analysis

Floating point

What about $\frac{1}{3}$ and $\frac{1}{10}$.

A. $\frac{1}{3}$ and $\frac{1}{10}$ both require truncation.
B. $\frac{1}{3}$ requires truncation, but not $\frac{1}{10}$
C. $\frac{1}{10}$ requires truncation, but not $\frac{1}{3}$.
D. Neither $\frac{1}{3}$ nor $\frac{1}{10}$ require truncation.

Vote at m.socrative.com. Room number 415439.

Floating point

What about $\frac{1}{3}$ and $\frac{1}{10}$.

[A.] $\frac{1}{3}$ and $\frac{1}{10}$ both require truncation.

Images

Bit map — scanner, video camera, etc.

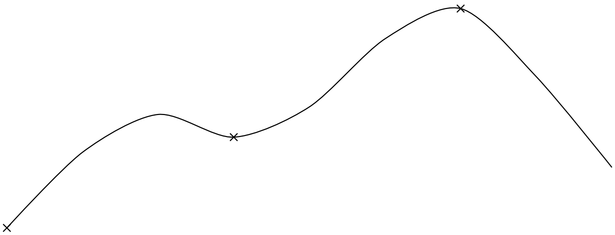
- image consists of dots — pixels
- 0 — white; 1 — black
- colors — use more bits —
  - red, green, blue components
  - 3 bytes per pixel
  - example: 1024×1024 pixels
  - megapixels (how many millions of pixels)
  - need to compress

Images

Vector techniques — fonts for printers

- scalable to arbitrary sizes
- image = lines and curves
- poorer photographic quality

Sound



Sounds waves

- sample amplitude at regular intervals — 16 bits
  -8000/sec — long distance telephone
  -more for music
- Musical Instrument Digital Interface — MIDI
  -musical synthesizers, keyboards, etc.
  -records directions for producing sounds (instead of sounds)
      -what instrument, how long

Data compression

Many lossless techniques:

- run-length encoding: represent 253 ones, 118 zeros, 87 ones
- relative encoding/ differential encoding: record differences (film)
- frequency-dependent encoding: variable length codes, depending on frequencies
  - Huffman codes
- Dictionary encoding: (can be lossy)
  - Lempel-Ziv methods: most popular for lossless — adaptive dictionary encoding
  - Lempel-Ziv-Welch (LZW): used a lot - GIF

Lempel-Ziv-Welch

Create a dictionary, as reading data.
Refer to data already seen in the dictionary.

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Lempel-Ziv-Welch

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Input: $ACAGAATAGAGA$
Dictionary: 8-bit ASCII alphabet
Output:

Lempel-Ziv-Welch

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Input: $ACAGAATAGAGA$
Dictionary: ASCII alphabet, $AC : 256$
Output: 65

Lempel-Ziv-Welch

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Input: $ACAGAATAGAGA$
Dictionary: ASCII alphabet, AC : 256, CA : 257
Output: 65, 67

Lempel-Ziv-Welch

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Input: $ACAGAATAGAGA$
Dictionary: ASCII alphabet, AC : 256, CA : 257, AG : 258
Output: 65,67,65

Lempel-Ziv-Welch

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Input: $ACAGAATAGAGA$
Dictionary: ASCII alphabet, AC : 256, CA : 257, AG : 258, GA : 259
Output: 65,67,65,71

Lempel-Ziv-Welch

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Input: $ACAGAATAGAGA$
Dictionary: ASCII alphabet, AC : 256, CA : 257, AG : 258, GA : 259, AA : 260
Output: 65,67,65,71,65

Lempel-Ziv-Welch

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Input: $ACAGAATAGAGA$
Dictionary: ASCII alphabet, AC : 256, CA : 257, AG : 258, GA : 259, AA : 260, AT : 261
Output: 65,67,65,71,65,65

Lempel-Ziv-Welch

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Input: $ACAGAATAGAGA$
Dictionary: ASCII alphabet, AC : 256, CA : 257, AG : 258, GA : 259, AA : 260, AT : 261, TA : 262
Output: 65,67,65,71,65,65,84

Lempel-Ziv-Welch

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Input: $ACAGAATAGAGA$
Dictionary: ASCII alphabet, AC : 256, CA : 257, AG : 258, GA : 259, AA : 260, AT : 261, TA : 262, AGA : 263
Output: 65,67,65,71,65,65,84,258

Lempel-Ziv-Welch

1. Initialize the dictionary to contain all strings of length one.
2. Find the longest string W in the dictionary that matches the current input.
3. Write dictionary index for W to output and remove W from the input.
4. Add W followed by the next symbol in the input to the dictionary.
5. Go to Step 2.

Input: $ACAGAATAGAGA$
Dictionary: ASCII alphabet, AC : 256, CA : 257, AG : 258, GA : 259, AA : 260, AT : 261, TA : 262, AGA : 263
Output: 65,67,65,71,65,65,84,258,263

Images

- GIF — Graphic Interchange Format
  - allows only 256 colors — lossy?
  - table specifying colors — palette
  - LZW applied
- PNG — Portable Network Graphic
  - successor to GIF
  - palette, plus 24 or 48 bit truecolor
  - LZ method compression (better, avoided patent problem)

Images

- JPEG — photographs
  - lossless and lossy modes
  - different qualities
- TIFF — has LZW option — patent has expired

Audio and video

MPEG — Motion Picture Experts Group

MP3/MP4 most common for audio

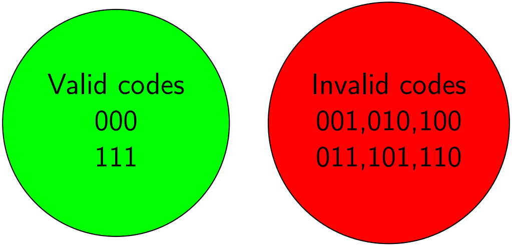
For audio/video — use properties of human hearing and sight

Error detection

- detecting that 1 bit has flipped — parity bit
  - odd
  - even
- can have more to increase probability of detection
- checksums (hashing or parity)

Error correction

- Hamming distance – number of different bits
  - 01010101 and 11010100
  - Hamming distance 2
- error correcting codes — Hamming distance $2d + 1$
  — correct d errors
  — detects more errors than it can fix



Computer architecture

Von Neumann architecture

— architecture where program stored in memory

Computer architecture

Von Neumann architecture —
(bottleneck — memory slower than processor)

Registers:

- general purpose
- special purpose
  - program counter
  - instruction register
  - others...

Computer architecture

Adding 2 values from memory:

1. Get first value in a register
2. Get second value in a register
3. Add results in ALU — result in a register
4. Store result in memory (or a register)