Images

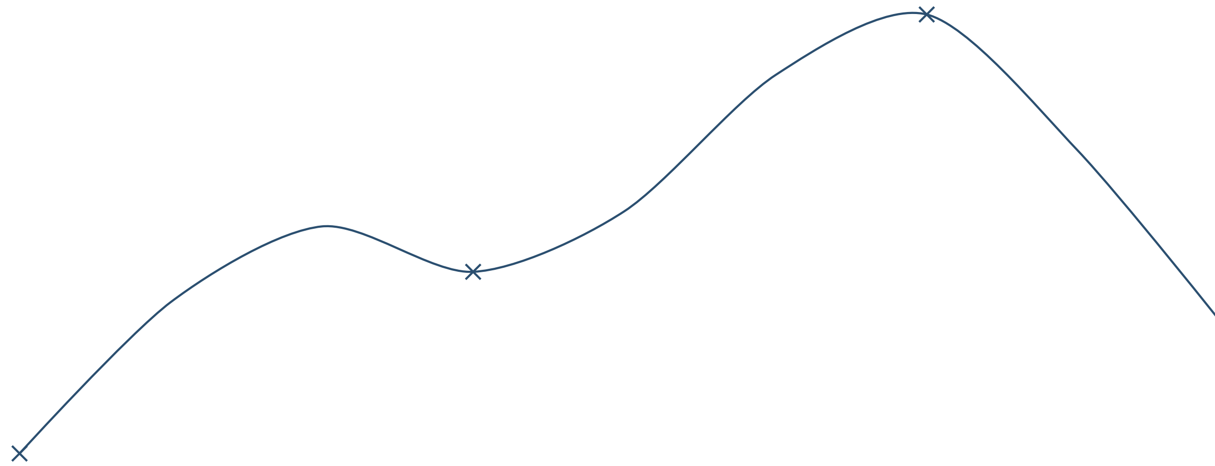
Data Representatation
Data Compression
Error Correction
Computer Architecture

- 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 $\times$ 1024 pixels
      - megapixels (how many millions of pixels)
      - need to compress

Images

- 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

Data Represenatation
Data Compression
Error Correction
Computer Architecture

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

Data Representation
Data Compression
Error Correction
Computer Architecture

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

Data Representation
Data Compression
Error Correction
Computer Architecture

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

Data Representation
Data Compression
Error Correction
Computer Architecture

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

Data Representation
Data Compression
Error Correction
Computer Architecture

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

Data Representation
Data Compression
Error Correction
Computer Architecture

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

Data Representation
Data Compression
Error Correction
Computer Architecture

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

Data Representation
Data Compression
Error Correction
Computer Architecture

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

University of Southern Denmark

Lempel-Ziv-Welch

Data Representation
Data Compression
Error Correction
Computer Architecture

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

Data Representation
Data Compression
Error Correction
Computer Architecture

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

Data Represenatation
Data Compression
Error Correction
Computer Architecture

- 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

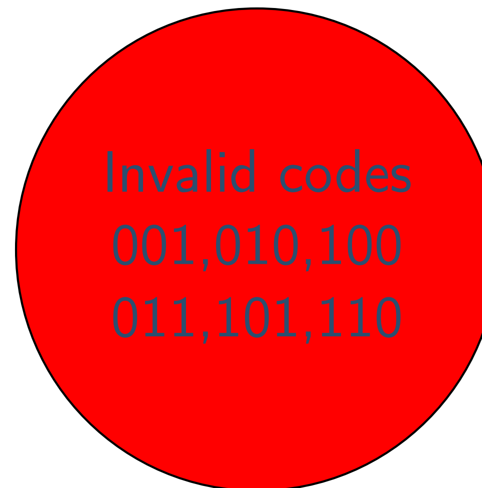
Error detection

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

Error correction

Data Represenatation
Data Compression
Error Correction
Computer Architecture

- 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

Data Represenatation
Data Compression
Error Correction
Computer Architecture

Von Neumann architecture
— architecture where program stored in memory

Computer architecture

Data Represenatation
Data Compression
Error Correction
Computer Architecture

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

Registers:

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

Computer architecture

Data Represenatation
Data Compression
Error Correction
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)

Data Represenatation
Data Compression
Error Correction
Computer Architecture

Example machine language — Appendix C
Instruction:

- 4 bits op-code
- 12 bits operands
  - 4 bits register
  - 8 bits address — 256 words in memory

How many general purpose registers are there?

A. 4
B. 8
C. 12
D. 16
E. 32

University of Southern Denmark

Example machine language

Data Represenatation
Data Compression
Error Correction
Computer Architecture

Instructions:

Op-code	Operands	Meaning
1	RXY	Load reg R from memory cell XY
2	RXY	Load reg R with value XY
3	RXY	Store contents of reg R in cell XY
4	$0RS$	Move contents of reg R to reg S
5	RST	Add two's compl. contents of reg S to reg T; store result in R
6	RST	Foating point add
7	RST	OR
8	RST	AND
9	RST	XOR
A	$R0X$	Rotate reg R X bits to right
B	RXY	Jump to XY if $c(R) = c(0)$
C	000	HALT

Note operands are hexadecimal.

Example machine language

One word (cell) is 1 byte.
One instruction is 16 bits.

Machine cycle:

- fetch — get next instr., increment program counter by 2
- decode
- execute (instr)

Example machine language

Data Representation
Data Compression
Error Correction
Computer Architecture

Example: check if low-order 4 bits of value in reg 1 = 0

2000	load	load zero into reg 0
220F	load	load string 00001111 into reg 2
8312	AND	c(reg 1) AND c(reg 2) —-> reg 3 — masking
B3XY	JMP	jump to address XY if c(reg 3) = c(reg 0)

Example machine language

Data Represenatation
Data Compression
Error Correction
Computer Architecture

How can we complement a byte in reg 1?

A. load 11 in register 2; OR 3,1,2;
B. load FF in register 2; OR 3,1,2;
C. load 00 in register 2; XOR 3,1,2;
D. load 11 in register 2; XOR 3,1,2;
E. load FF in register 2; XOR 3,1,2;

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

Computer architecture

Data Represenatation
Data Compression
Error Correction
Computer Architecture

RISC — reduced instr. set — fast per instr. — cell phones
CISC — complex instruction set — easier to program — PC

Clock

- coordinates activities
- faster clock $\rightarrow$ faster machine cycle
- Hz — one cycle per second
- MHz — mega Hz (1 million Hz)
- GHz — giga Hz (1000 MHz)
- flop — floating point ops / sec
- benchmark — program to run on different machines for comparison