Hexadecimal Notation

To shorten bit strings for humans:

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

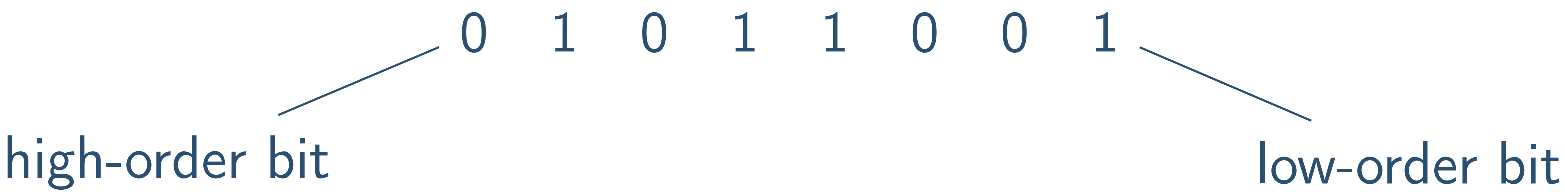
Storage technology

capacitors on chips??? — changes!!!

dynamic memory — need to refresh data, it dissipates
non-volatile memory — doesn't lose data if power lost

Memory:

byte — 8 bits



most significant bit

least significant bit

Storage technology

Main memory

- words = cells — fixed size
  8, 16, 24, 32, 64 bits
- words have addresses - count from 0
- can use consecutive words if need more bits for value
- can access words in any order random access memory (RAM)
- get value of word — read or load
- place value of word — write or store

Storage technology

Main memory

- size — power of 2 — addresses fixed length (usually)
  - $2^{10} = 1024$ bytes = 1 kilobyte — 1 KB
  - 4096 bytes = 4 KB
  - $2^{20} = 1,048,576$ bytes = 1 megabyte — 1MB
  - $2^{30} = 1,073,741,824$ bytes = 1 gigabyte — 1GB
  - $2^{40} = 1,099,511,627,776$ bytes = 1 terabyte — 1TB
- Some people use these terms for powers of 10.

Storage technology

Mass (secondary) storage

- disk, CD's, magnetic tapes, flash memory
- CD → DVD → Blu-ray
  similar technologies — more capacity
- on-line vs. off-line — human intervention
- mechanical, slower (except flash memory)
- disk
  - often several in layers — space for heads
  - read/write heads above tracks
  - cylinder — tracks on top of each other

Storage technology

Mass (secondary) storage

- disk
  - sector — arc of a track
    - files stored as physical records = sectors vs. logical records (fields, keys)
    - each contains same number of bits (512 or 1024 bits, for example)
    - with a group of tracks, each contains same number of sectors — having different groups, with fewer tracks toward middle is zoned-bit recording
    - locations of tracks and sectors marked magnetically during formatting

Storage technology

Secondary storage

- flash memory
  - cameras, cell phones, etc.
  - not mechanical
  - not dynamic
  - hard to erase or rewrite a few locations often
  - intensive writing reduces lifespan

Text

Text — characters (symbols) — standards

- ASCII — appendix A
- EBCDIC
- BCD
- Unicode — implemented by different character encodings
  - UTF–8 — one byte for ASCII, up to 4 bytes
  - UCS–2 — older, 16 bit codes
  - UTF–16 — extends UCS–2, two 16-bit code units

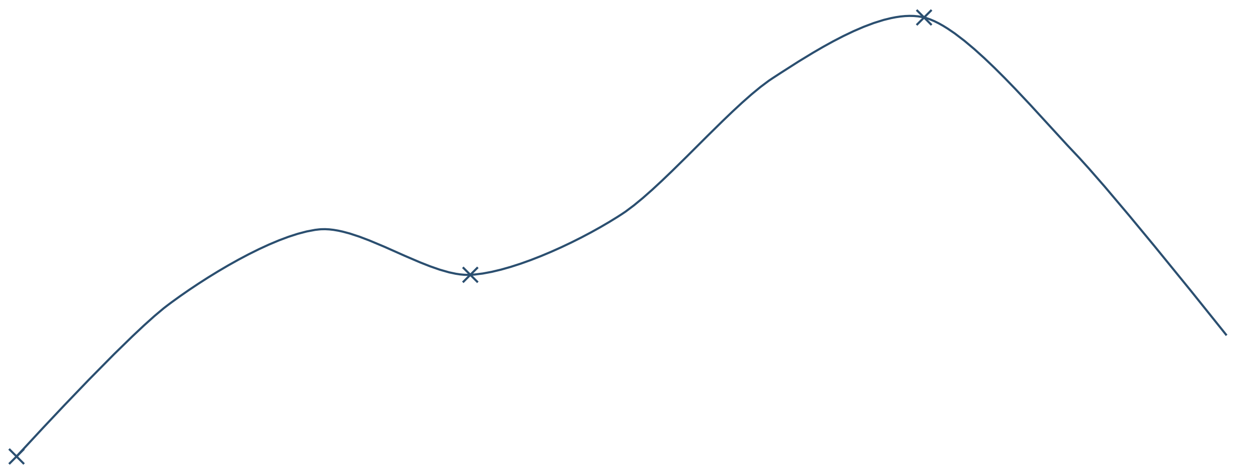
Images

- 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

- 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

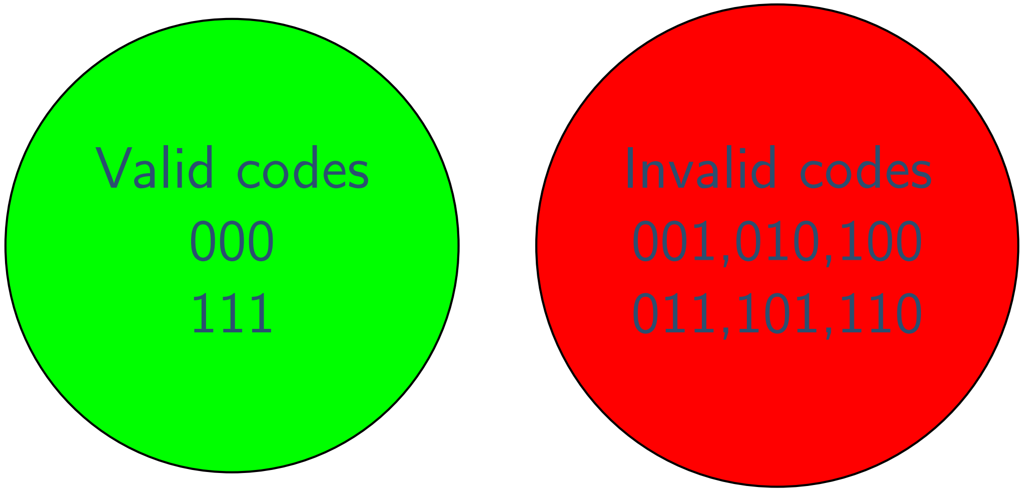
For audio/video — use properties of human hearing and sight

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


Valid codes
000
111
Invalid codes
001,010,100
011,101,110