CS 105 - Stanford

* Binary (0/1) : Switches
* Binary addition

  1 + 1 = 10

  11 + 1 = 100

1 1 0 1

↳ $2^0 = 1$

↳ $2^2 = 4$

↳ $2^3 = 8$

13

2^7 combinations

16 bits [0 - 65535]

↳ overflow (65535) : large negative number!

Overflow:

1 1 11 1111 ⟶ 255.

1

(1) 0 0 0 0 0 0 0 0 0 ⟶ 0

↓

Discarded!

Sign bit : MSB

Binary floating point numbers:

* $1/10 = 0.1_{10}$ ⟶ 0.0001100110011_2

  ⟶ $0.0001\overline{1}_2$

*

ASCII:

A → 65_{10}

NULL → 0

Start of heading : 1

Start of text : 2

end of text : 3

bell : 7

Backspace : 8

Tab : 9

Return : 13

Why binary / text

* Plain binary: no encoding
* TXT : encodes ⟶ Send!

ISO - 8859 (more symbols)

↓

use fonts supporting symbols.

Digital Images:

* pixel: unit (Resolution: how many pixels)

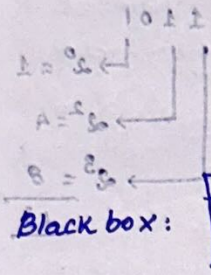
VGA: 640×480

Television: 640×480

Ultrawide: 3440×1440

HDTV/1080P: 1920×1080.

Aspect ratio: $\frac{\text{width}}{\text{height}}$



Pixel:

* R: Red, G: Green, B: Blue → mix (generate colors) → 3bytes
  16.7 million combinations

* RGB Alpha
  ↳ transparency!

2160P:

* P - progressive (update every other line: instead of every single line)
* 30 times a second → odd (1st $\frac{1}{30}$th second), even (2nd $\frac{1}{30}$th second) → Interlaced!

* progressive → update every line at once!
  ↳ All new televisions are progressive.

4K × 2160P

UHDTV

* Ultra high definition TV (Lot of pixels wide, height)
* HDR: High Dynamic Range!
  ↳ higher pixels per inch
  ↳ Better contrast (diff b/w dark & light)

10 bits → R, G, B

$2^{10} \times 2^{10} \times 2^{10}$ = 1 billion colors!

Printer & Color:

* Subtractive color

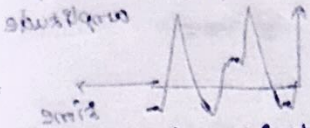
Color

Screen (Additive) → 0 → 255 (white) → RGB

Paint (Subtractive) → CMYK

↳ white light absorbed by paints (what remains: seen by us)

Absorb lights (Subtract)



Bitmaps (vs) objects:

* bitmap: As grid (0,0) → (255, 100, 200) → 9216 bytes
* object/vector: centre (12,12), radius 8px, stroke 1px, color rgb(255,0,0) → 24 bytes

'Store in 2 ways' → Bitmap

↳ objects / graphics (can't manipulate indiv. pixels)

↳ only entire geometry!

Bitmap: can't represent with math formula!

Right format for the right job:

* Bitmap → JPEG, PNG, GIF (RAW, HEIC)
  - ↳ Apple uses
  - ↳ higher end cameras
* object formats → SVG.

1) (0,0): on
   (1,1): on
   (2,2): on
   ↓
   Large space!

2) Row 0 all on
   Row 1 (1,1) except
   Row 2 (2,2) except
   ⋮

JPEG: (Joint Photographics Experts Group)

* Store photographs (16.7 million colors): Lossy compression (10 - 12) ↓ RAW

PNG: Portable Network Graphics:

* Built on top of GIF (diagrams, graphics): hand/computer graphics.
* PNG not efficient for photographs (not lossy)

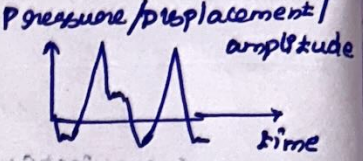
* Can make individual pixels transparent!

GIF: (animations: replaced by animated PNG)

* 256 Colors only (pixelated)

Sound:

* Stereo: 2 microphones (each microphone 44100 times: Sound waves)
* Sampling rate: how many measurements?
* Bit depth:
* Dynamic Range / bit depth: min & max amplitude.
* greater the sampling, bit depth: better quality, high space.



Digital music:

* CD audio: large space (50MB)

(5min x 60s x 44.1K x 2 channels x 2bytes)

44100 Samples: 44.1KHz
16 bits per Sample: 2 bytes
Stereo: left, right channel

* MP3 files (Apple: AAC, WMA: Windows)

Create MP3: (lossy, quality can be set)

* Fast fourier transform (Individual sound frequency of the wave)
* psychoacoustics: eliminate certain frequencies (outside human range)
* Huffmann coding: store high frequency codes! (encode)

Standard ≈ 4.5 MB/5min

5 x 60 x 128 Kb/s = 38,400 Kb

Music (Analog vs digital):

* Analog: continuous (real world)
* Digital: discrete quantities (stored as bits)

Alternative formats:

* FLAC: Free Lossless Audio Codec
  - ALAC (Apple)
  - WMAL (windows): Microsoft windows media Audio lossless

5min Song

35.67 MB : FLAC
4.5 MB : MP3
50.47 MB : CD

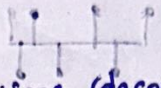
MIDI file:

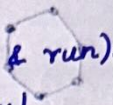
* Individual notes not waveforms (velocity, vibration)
* Less space (manipulate notes : useful for Composers)

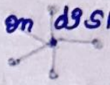
Hardware:

* CPU, main memory, RAM, ROM, I/O
  - CPU → processing
  - RAM, ROM → storage
  - I/O → I/O

Programs / Software:



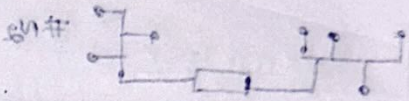




* CPU instructions (decompress & run), store on disk
* move to RAM, run using CPU!

CPU: (bottleneck)

* Core i3, i5, i7, i9 : Intel
* AMD: Ryzen, Athlon



multiCore CPU:

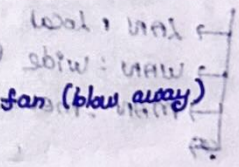
* Can't make faster, add more!
* Take advantage of multi cores (photo processing), weather simulation
* Super computers: milions of cores

GPU:

* Special unit: create 3D graphics, 3D games, editing, AI, neural network Bitcoin mining!

Anatomy of a laptop:

- motherboard (PCB : printed circuit board)
- Hard drive/disk.
  - SSD (no moving parts)
  - heat sink, fan
    - metal conducts heat to fan (blow away)
  - CPU, graphics chip
  - memory module.



Virtual memory:

* Allow additional programs than it can run (virtual memory)
* Abuse: poor performance.

* old: 1 program at a time
* current:
  * can't load many programs (I'm not gonna run them all)!
  * So even loaded, won't run everything at once.

1GB

Browser: 400K
media : 300K
XLS : 100K
word : 100K

RAM SSD HD Magnetic tape

Fast ⟷ Slow

expensive ⟷ cheap!

Solution:

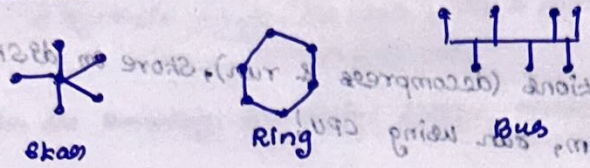
* Present (Say HHD/SSD): main memory (slower) ⟷ Virtual memory
* active apps (running), Inactive (virtual memory)

↑ ⟷ ↑
Swap (based on activity)

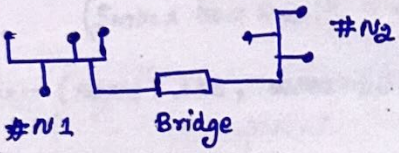
*Thrashing: Spend time moving back & forth (RAm & Virtual memory)

↳ Buy memory! (RAm)

Computer networks: (Topology)



* Internet: network of networks.
* Bridge: Connect 2 different/same networks with the same type.



* Router: Connect two different types of networks.

Factors:

* Bandwidth: Speed of the network (frequency) → band of frequency [wider, ↑Speed]
* Narrowband: Telephone line (low BW)
* Broadband: broader BW (just term, used by Companies)
* Network
  - → LAN: Local
  - → WAN: Wide (city, continents)
  - → MAN: metropolitan (with in city)
  - ...
* Hop: how many networks to pass through!
* more hops, more lag
* lag & latency
  - ↳ Not reached in time: Lag
  - ↳ Amount of delay: latency
* Lot of BW, still face delay (many hops) → Earth is mass
* webservers: use CDN for faster delivery! (Download assets)!

Star topology:

* Central hub (manage transport)

MAC address/Physical address

* Media Access Control : Base 16 (unique)
* 00-00-42-AF-38-05.
  - manufacturer prefix.
* multiple MAC address → wifi, → bluetooth, → ethernet
* Using Social Security number for mailing: won't work! (How to find?)

IP address (Internet protocol):

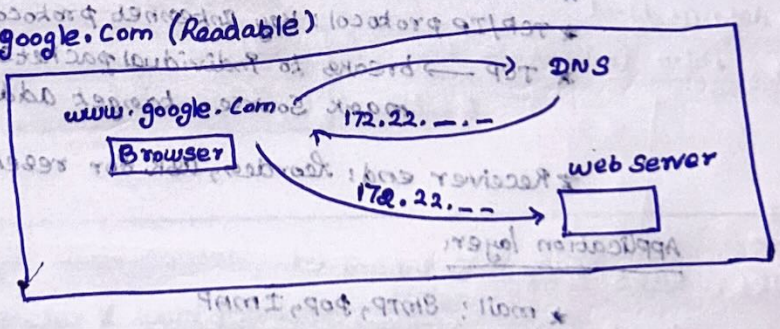
* 124.12.0.152 (4 bytes): problem (4 billion) → 7.8 billion population
* 2002:800C:5311:08fe:800C:5311:6a35:2139
  - ↳ 16 bits each, 8 sets! → 128 bits
  - 16 bytes
* 2^{32} (old) → 2^{128} (new system) → IPV6

MAC → manufacturer assigns
IP → By ISP

181.12.83.0 → local
↳ Know place!

Not easy to remember (Domain – DNS) / host name:

* www.google.com (Readable)

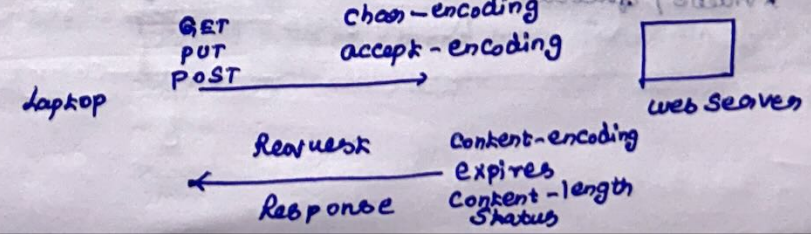


Port number: (firewall turns on/off the ports)

* Identify programs running on a computer
* browser (80), email reader (143)
* Firewall: Controls in/out traffic in a network!
* Browser uses Choosed ephemeral port assigned by System!

Protocols:

* why?
  - * Standard: followed by all (no conflicts)
  - * HTTP: Hyper Text Transfer protocol



* Agreement (how they plan to carry out a task)
  - Actions
  - Type, Format: Info
  - Error Conditions.

Protocol (vs) program:

* program (browsers) understands protocol & communicates with webservers.

Protocols @ the Internet:

* Fedex (delivery)

App layer
Transport
Network
Physical

Physical:

* 0 → 1070Hz tone, 1 → 2025 Hz tone
* new network → new physical layer

Network layer:

* Internet protocol (packets: max 64 KB packet)
  - IP @ sender, recipient, Checksum, Data
* Best effort (No delivery guaranteed)
* each new network: New IP/Network layer.

Transport layer:

* TCP (Transmission Control protocol)
* TCP/IP protocol: Key Internet protocols
* TCP → breaks to individual packets
  - needs Source, target address, number!
* Receiver end: reorder, ask for resend (missing ones).

Application layer:

* mail: SMTP, POP, IMAP
* web: HTTP
* file: SFTP, FTP

'every program'

Packet switching (vs) Circuit switching

* dedicated line (Some one must connect the line): circuit switching
* packet switching: efficient (Everybody's packet): sent in a single line

Shared lines

* videos / facecall: won't wait for every bit (Just play with glitch)

HTML: Origins of the web:

* Internet + Hypertext ⟶ WWW (world wide web)

Book
* pages, chapters
* encyclopedia: ordered!

* **CERN:**
  * physics research consortium (23 members)
  * Headquarters: Geneva
  * popular: collider (27 km ring)
* **Tim Berners Lee**
  * 1980: System to share info (papers)
  * Combined Internet & hypertext!

Computer
* Any order (any node)
* Hypermedia: link text nodes (speech, video, audio, image)

HTML:

* HTTP (how 1 computer can ask for a file: not mentioning formats)
* Hypertext: has grammar like human languages (syntax, semantics)
* Markup: where should appear (how: bold, size)

  → containers (p, bold)

* Stanford → attribute: value pair.
* HTML ignores white space!

<!DOCTYPE html> ⟶ version

head: info about webpage (encoding)

body: Actual content

Create html page:

* Text editor & word processor: different!

validator.w3.org → validate file for any errors!

Text level tags: , <i>

Block level tags: <h1>,

  Hello

  Hello   ⟶ wrong (text level: can't have block tags)

Angled quotes: won't work for HTML syntax

CSS

* HTML: Semantic info, markup
* CSS (Cascading Style Sheet): present info (formatting)
  - separate language (own rules, semantics)
  - style property: value (classes)
* font-properties (family, style, weight, size)

Inline
External → Best!
Style tag (Internal)

Linking webpages:

* <a href = " "> tag → Absolute reference.
  - http://www.google.com/html.html
* <a href = "another.html"> → Relative reference.

Relative: portable, shorter
Absolute: needed to access other servers

index.html (default): mostly!

* home.html
* default.html
* .htm (microsoft only 3 letters)!

5 before dot, 3 after it!

URL (vs) URI:

* URL: https://www.stanford.edu/another.html
* URI: https://www.stanford.edu/purpose (with http method)

→ Send me an email
→ ftp, sftp, facetime, skype, steam

URL: Don't use (specific location: It mentions)
Not portable!

* class → write once, use inline!

pseudo class → class = "red-color" : link, :visited.
real class → .red-color {

Internal Class/pseudo

* link
* visited

Pseudo class:

a : link
a : visited → Colons in style sheet!
a : active

```
a:link {
}
a:visited {
}
```

:first-child
:last-child
:nth-child
:nth-child(3)
:nth-child(odd)
:nth-child(even)

Add elements:

Colors:

* 24 bit (R,G,B) ⟶ #8C1515
* websafe (216 colors)
* 148 named colors
* 16.7 M colors + 256 levels of alpha (opaque / transparent)
  - → named color (CSS)
  - → Name alpha (CSS)

3 hexa: 2^{12} (4 bits per R, G & B)
: 4096 colors

rgb (100%, 0%, 50%)
rgb (255, 0, 128)
rgba (_____)
hsl - hue, Saturation, lightness

Decimal : 10
Octal : 2^3
Hexa : 2^4
→ Every 4 digits
every 3 digits
in binary!

1000 1100 0001 0101 0001 0101

140 (R) → 8C
21 (G) → 15
21 (B)

#8C1515

Other places

* Network physical address
* Computer memory!

0x → hexa
0 → octa

Captions:

* Type selector → apply to all heading (eg)
* class selector
* Id selector

* Text: no natural width → expand as much as available

microformats

→ div with class (use number → print stars)!

↳ use class to pick information, filters!

Host:

* hosting service
* Creat & keep files
* File transfer software (SFTP) → Secure Fx (windows), Fetch (mac)

* HEIC → Not web compatible (apple)
* WebP, WebM : (Google) → half of web browsers don't
* Use JPEGs, PNGs, GIF, SVGs

Layout:

* Table → hack!
* Float-based
* Flexbox → most popular
* grid

Flexbox (97% Support)
Grid (95% Support)

Table based:

* Locked, can't change without extensive HTML reword
* Nested tables: expensive computation.

Float based:

* Fairly tied to HTML, messy issues (not straight forward)

Flexbox:

* Skill fairly tied (change direction, switch to desktop mode: easy)
* elements tied to parent

Grid:

* Like table but defined in css, placement in css

---

* Forms

Advanced Image techniques:

* Imagemap (specific part, specific redirect / image)

```
<map name = 'Computer-map'>
    <area shape = 'rect' coords = '50, 20, 217, 157' href = '___'
          alt = 'monitor' />
    :
</map>
```

* use:

```
<img src = 'comp.jpg' usemap = '#Computer-map' alt = 'photo of comp' />
```

Image Sprites:

* Lot of images: Downloads, slower webpage

[From a combined .gif]

Get (vs) Post (method)

* Info to be sent (Control format)
* Side effect on bookmarking, emailing webpage URLs
* Determine state of server.
  - get (URL) → params
  - post (body) → params
  - Get: no change, Idempotent
    Cache, reuse.

Control over user device:

* UI attractiveness, utility: vary with device size
* Tooltip, button size: mouse / pointer (vs) fingers: Not Supported across devices.
* Solution:
  * minimum users: design for
  * user-agent → mobile websites
  * Responsive web design → Better! [CSS]

```
@media (max-width: 480px) {
    nav { display: none; }
}
```

media queries
- hover (none, hover)
- pointer (none, fine, coarse)
- width, height, orientation (landscape, portrait)
- aspect ratio, color (no. of bits)
- resolution (dots per inch)

Human-Computer Interaction:

* Windows, Icon, mouse, pointer (GUI)
* Future: VR, AR, speech, gesture, Brain Interface
* Psychology: understand human cognition (perceive color, limitations of human memory)
* Graphic design, ergonomics, sociology/Ethnography (usable?)
  - ↳ workplace study

UI design:

* observe workplace, process, potential users
* Task, roles → Prototype, Testing, Iteration
  - Low fidelity (paper sketch)
  - high (wireframe, mockup, prototype)

Usecase

Bookcase:

* visitors (who)?
* why they are here?
* Expertise level
* Devices they have
* Internet connections
* Tasks
  - book
  - topic
  - Random (bored)
* potential Cx
  - Young professional
    - Cellphone, moderate speed
  - Old, low tech expertise
    - good Internet (windows laptop)
  - Junior high student
    - moderate tech skills
    - Low end laptop
    - Good Internet connection

Personas

* figure out (NIKKI)?
* while doing Commute able to access?

Prototype, ASK testers

website design: (unique, Identifiable)

* logo, Font choice
  - Sans-serif → pregen!
  - Serif
  - monospace
* color scheme
* layout, Navigation scheme:

Identification on each page
Successful: user's sharing deep links

Sans-Serif

* Elegant, Contemporary

Serif

* Formal, authorative
* old fashioned, cloudy

Monospace:

* All Character: Same width (program/computer o/p)

Color:

* HUE, Saturation, Brightness (HSB) → Spectrum colors
* RGB: Computers displays colors

Convert HSB → RGB (math formula)

Color wheel
- → Complementary Color Scheme
- → Split Complementary Color scheme
- → Analogous Color Scheme

Layout & navigation:

* How people are going to navigate! (Consistent)

Python

Method (vs) function:

* method → object(method)
* function → directly access (no need for a method)

"doesn't really matter !"

Network Security Concerns:

* Confidentiality
* Authentication (Cryptography)
* Integrity (checksum) hashes)
* Non Reputation

Cryptography (Study: send secure messages)

Encryption, Decryption
- → Caesar cipher
- → Symmetric, asymmetric algo.

Asymmetric:

* Same key both enc & decryption: Symmetric
: 2 keys : Asymmetric

2 Keys → public, private — mathematically related.

* Not easy to generate one from another!

Asymmetric encryption: → Diffie hellman (Stanford EE department, Turning award)
→ British Intelligence Govt Comm. HQ (1973)

Amazon S encryption:

* Get Shared Symmetric Key (very long) → How to get the Key?
* Asymmetric: posts the Key (Amazon has private Key)!

↳ Asymmetric: Share the exchange (Slow)
Symmetric: further (computationally cheap)!

CA → website contacts CA (generate private key)
visit the page (CA is downloaded by the browser) encrypt (Success! — authenticated),

* CA has public Key (initiate the communication)!
*

Https: All the network is encrypted!

---

malware: (virus, worms, Trojan horse, zero-day exploits)

* malicious Software
* Spyware:
  * Spy (access to accounts, copy files, Control microphone, Camera, Keyboard log, track movement)
* Ransomware:
  * Ask for payment to decrypt!
* Adware:
  * Display ad (Software supported by ads, malicious display ads (without providing a service)
  * Trojan horse (Sometimes)
* Ad-related issues:
  * mine Cryptocurrency (mining)!
* Zombie computer:
  * Remot user Control → Spam
    → DoS
    → Fraud.
* worm/virus
* logic bomb (by original programmer)

Virus:

* Attach to other programs (execute) : Zip files can include exe. Code
* word : can include script!

Worms:

* propagate copies itself on a network

Trojan horse:

* claim something, do harm

Zero day exploits:

* exploit unknown weakness.

Attack vectors:

* Social engineer (Ignore tech aspects, go for weak spots)
  - ↳ gain more info
* phising: Send email (get data) → Send (fake)
  (Reproduce brand) → Reply-to (manipulate)
  → SMTP (anyone) Can use!
* Spear phishing: Individual specific target
  - ↳ Learn (Job, target co-workers)
  - ↳ hack coworkers, do harm
* Drop USB → plugged, hacked! (non attached networks: systems)
* Sql injection (encode!) → Sanitize
* Cross site scripting (webpage injected)
* clickjacking (Amazon: block iframe (headers))
* Sandbox security (client program: only allowed within)
* Drive by download (plugin/program)
* Man in the middle (Fake wifi, certificate/encryption issue)
* Dev dependencies (inherit vulnerabilities)
* Vulnerabilities in IP protocol.

Defensive techniques:

* Firewall: Allow only specific app, IP, scan for keywords
* proxy servers: (VPN) → like VPN
* VPN (Connect 2 offices), hide identity
* Air gap: Seperate from common internet
* Secure email (Confidentiality, Authentication, Anonymity (No IP record).
* HTTPS
* Disable JS. (use NoScript / SafeScript)
* MFA, passphrases
* password manager.

Practical measures:

* Update Softwares
* password recommendations (long, random, mixed)
* MFA
* Watch out for phising
* Anti malware Software
* Block JS

Privacy:

* Cell phones can track our movement (towers, wifi)
* closed circuit camera: Track where we go.
* Digital Consumption (news, magazines)
* Smart cars, Smart homes

Computer analysis:

* Sufficient processing power (license plate reading, face recognition)
* Big data

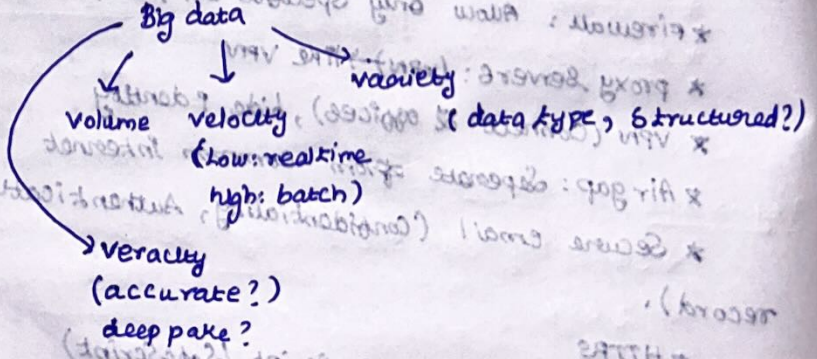
US Law
- → practical obscurity (Individual right)
- → Turn down FBI rap sheet

EU Law
- → right to be forgotten (news paper, search engines (couldn't link))
- → with in half year: look deletion requests (google)
- → 2018: GDPR (breach notification, right to access, forgotten)
- → privacy by design
- → Data protection officers (Report to high level management)

penalty upto 4% of worldwide income

Product / customer:

* Not paid → Somehow direct / target ads
* Google selling ads (where's the line)
* Email tracking (beacon), 3rd party cookies
* TOR: Anonymous routing!



IAAS → Infra (AWS, VPC; setup OS), high skilled IT

PAAS → Computing platform (Heroku, Amazon elastic bean) run Software

Serverless → Amazon lambda (scaling, setup)..

SAS → eg: Google docs.

Edge/Fog/Mesh → Computing.