Computational Practices

J.D., Michael, Hannah

- Casey Reas: https://www.youtube.com/watch?v=_8DMEHxOLQE
- Creative Coding: http://www.youtube.com/watch?v=eBV14-3LT-g

Introductions

- We
- You
  - Your origin
  - Your major
  - Your work
  - What do you want from this class? Why are you here?

Goals, nominal

- Computational Systems

- Programming

Goals, underlying

- Learn how to Google!
- Know the steps to take when you get stuck:
    - Not sure how to do something specific?
    - Doesn’t work how you want it to?
- Learn **how to learn** technical things

Specifics

- Bring a computer to every class
- **Ask questions!**
- Don’t be late
- Do your homework, make sure I can read it
- Make use of the coaches in the LRC & Hybrid Lab
- Let us know when you need extra help

The Obvious

- Do your own work. (Don’t plagiarize!)

- Don’t skip class. (Unless you’re sick! Email us.)

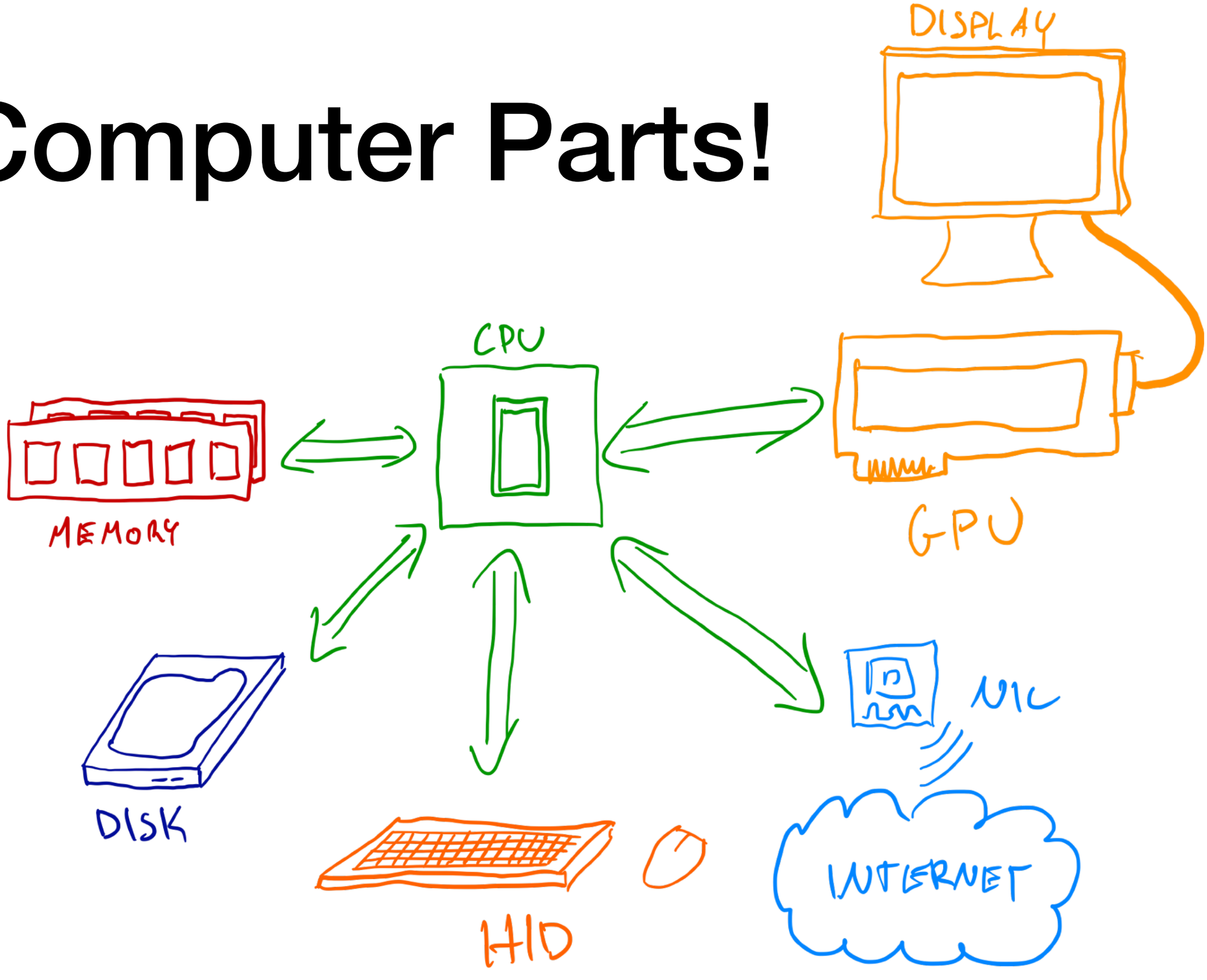
Structure

- First, an intense few weeks
- Then we'll recap, and have a midterm
- Then we'll apply those skills to your own projects

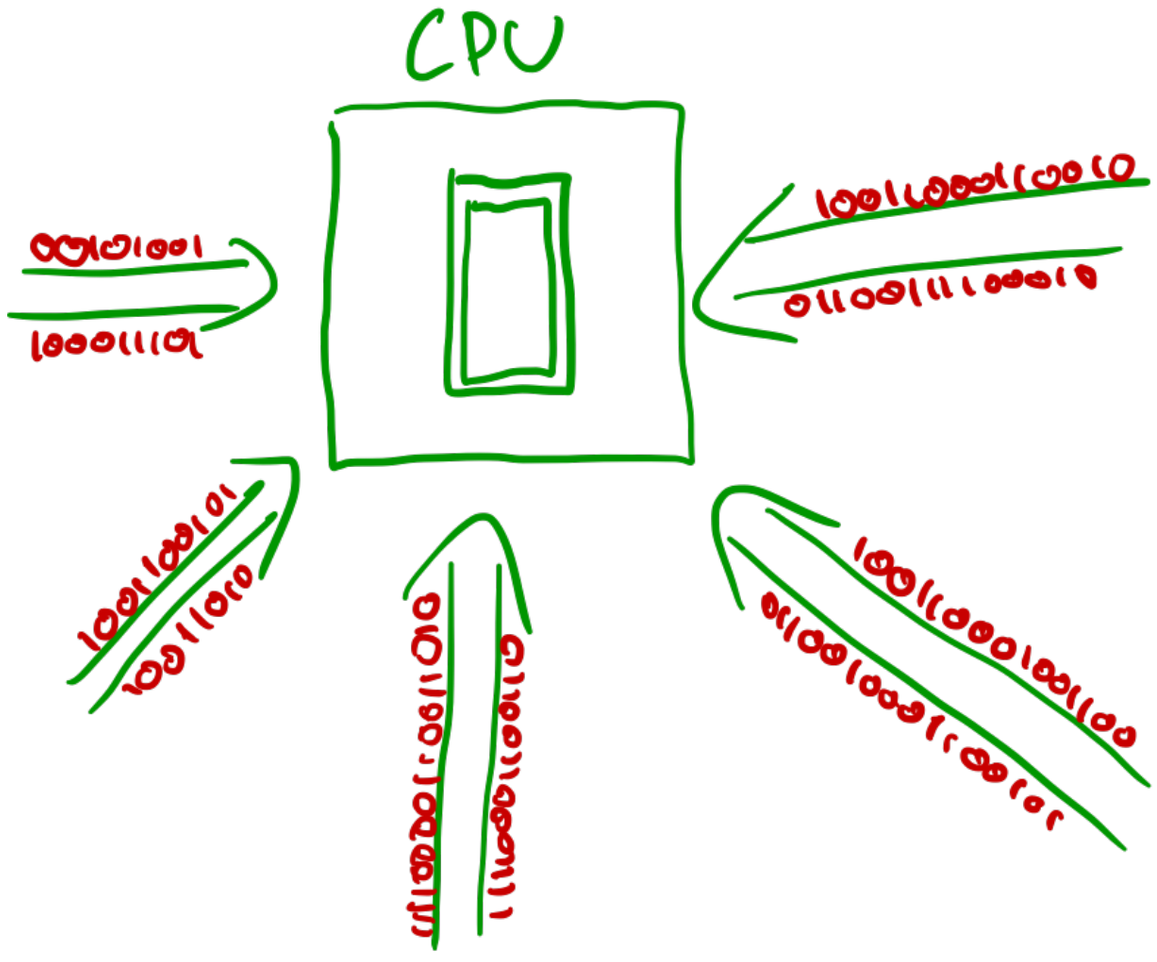
Grading

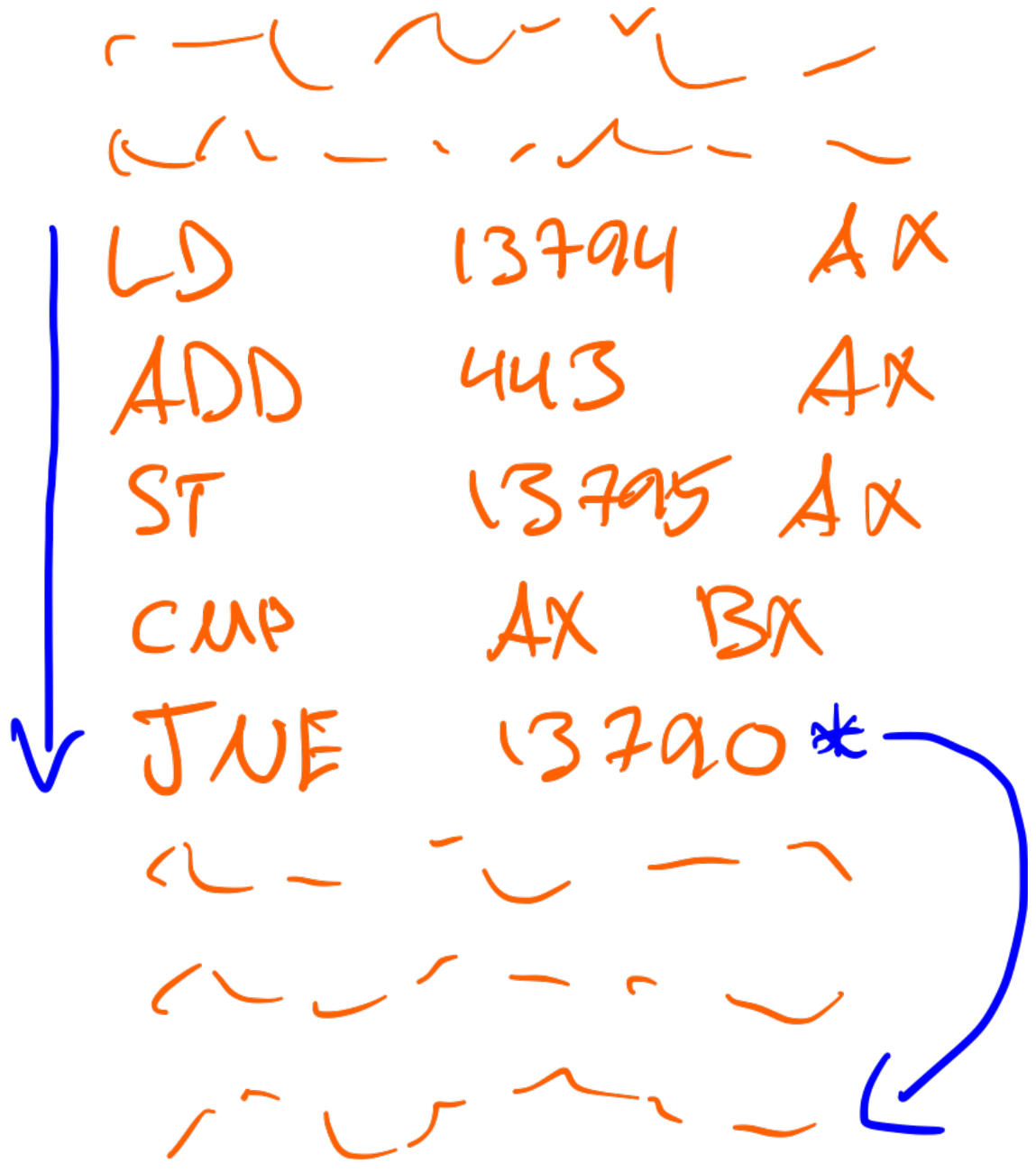
- 40% Homework & Assignments
- 25% Midterm
- 25% In-Class Lab & Project Work
- 10% Attendance & Participation

Computer Parts!



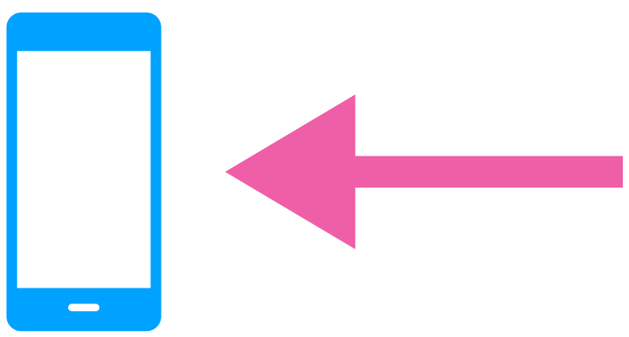
CPU





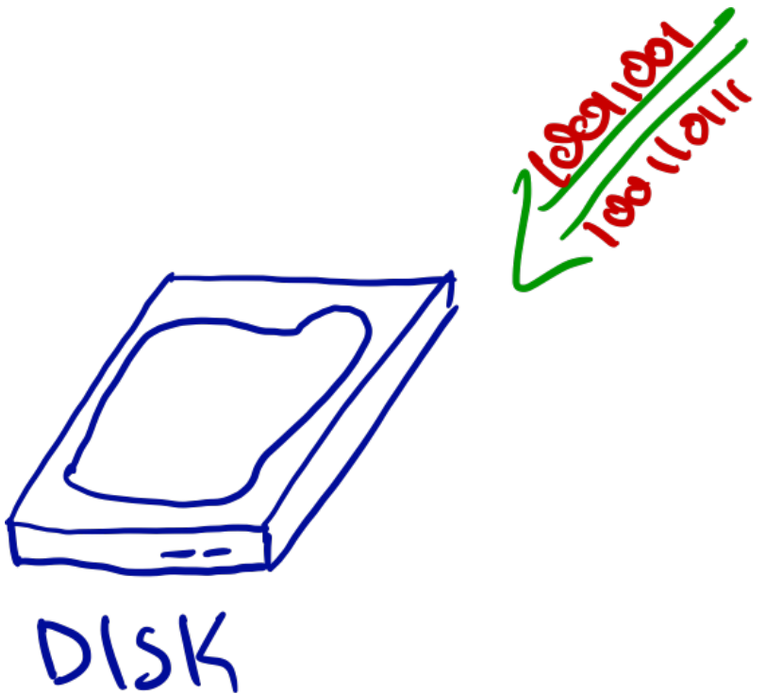
Short-term Memory

ADDR	DATA
13790	443
13794	77
13798	13794



1-2 Gigabytes: *billions* of numbers

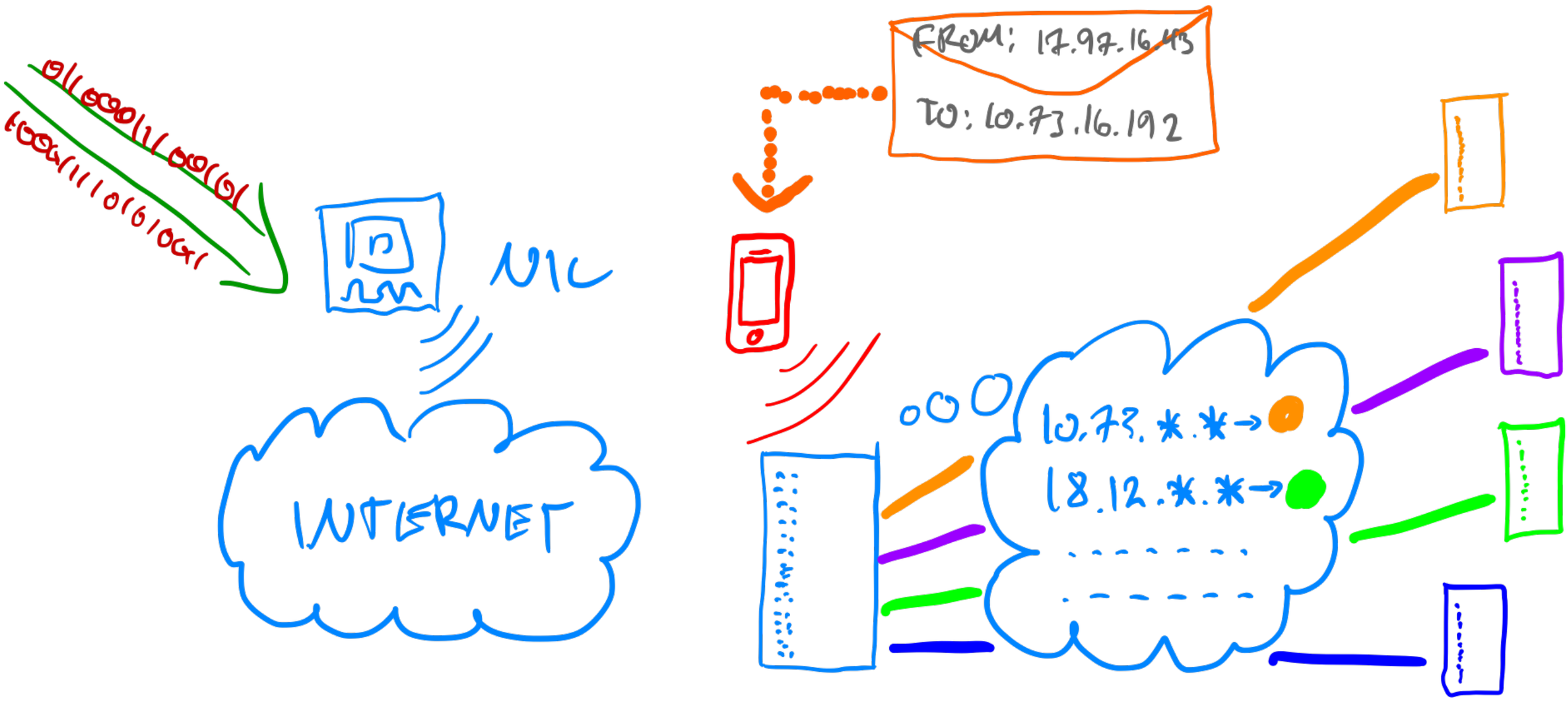
“Disk”



SECTOR	DATA
13790	4431f79361374945378
13791	7713954686333719478
13792	13794443798407160

Filename	Sectors
/home/jd/paper.txt	13790, 13791, 2452, 94314, …
/home/jd/photo.jpg	13792, 7894, 7895, 7896, 7897, …
/home/jd/	31337

Internet



What does **FAST** mean?

Latency Comparison Numbers (~2012)

L1 cache reference	0.5	ns			
Branch mispredict	5	ns			
L2 cache reference	7	ns			14x L1 cache
Mutex lock/unlock	25	ns			
Main memory reference	100	ns			20x L2 cache, 200x L1 cache
Compress 1K bytes with Zippy	3,000	ns	3 us		
Send 1K bytes over 1 Gbps network	10,000	ns	10 us		
Read 4K randomly from SSD	150,000	ns	150 us		~1GB/sec SSD
Read 1 MB sequentially from memory	250,000	ns	250 us		
Round trip within same datacenter	500,000	ns	500 us		
Read 1 MB sequentially from SSD	1,000,000	ns	1,000 us	1 ms	~1GB/sec SSD, 4X memory
Disk seek	10,000,000	ns	10,000 us	10 ms	20x datacenter roundtrip
Read 1 MB sequentially from disk	20,000,000	ns	20,000 us	20 ms	80x memory, 20X SSD
Send packet CA->Netherlands->CA	150,000,000	ns	150,000 us	150 ms	

Thanks to Jeff Dean & Peter Norvig

FAST, in human terms

Latency Comparison Numbers, Humanized

L1 cache reference	0.5 s	One heart beat (0.5 s)
Branch mispredict	5 s	Yawn
L2 cache reference	7 s	Long yawn
Mutex lock/unlock	25 s	Making a coffee
Main memory reference	100 s	Brushing your teeth
Compress 1K bytes with Zippy	50 min	One episode of a TV show, including ads
Send 1K bytes over 1 Gbps network	2.7 hr	Length of this class, minus breaks
Read 4K randomly from SSD	1.7 days	A normal weekend
Read 1 MB sequentially from memory	2.9 days	A long weekend
Round trip within same datacenter	5.8 days	A medium vacation
Read 1 MB sequentially from SSD	11.6 days	Waiting for almost 2 weeks for a delivery
Disk seek	16.5 weeks	A semester in college
Read 1 MB sequentially from disk	7.8 months	Almost producing a new human being
Send packet CA->Netherlands->CA	4.8 years	Bachelor’s degree average time to completion

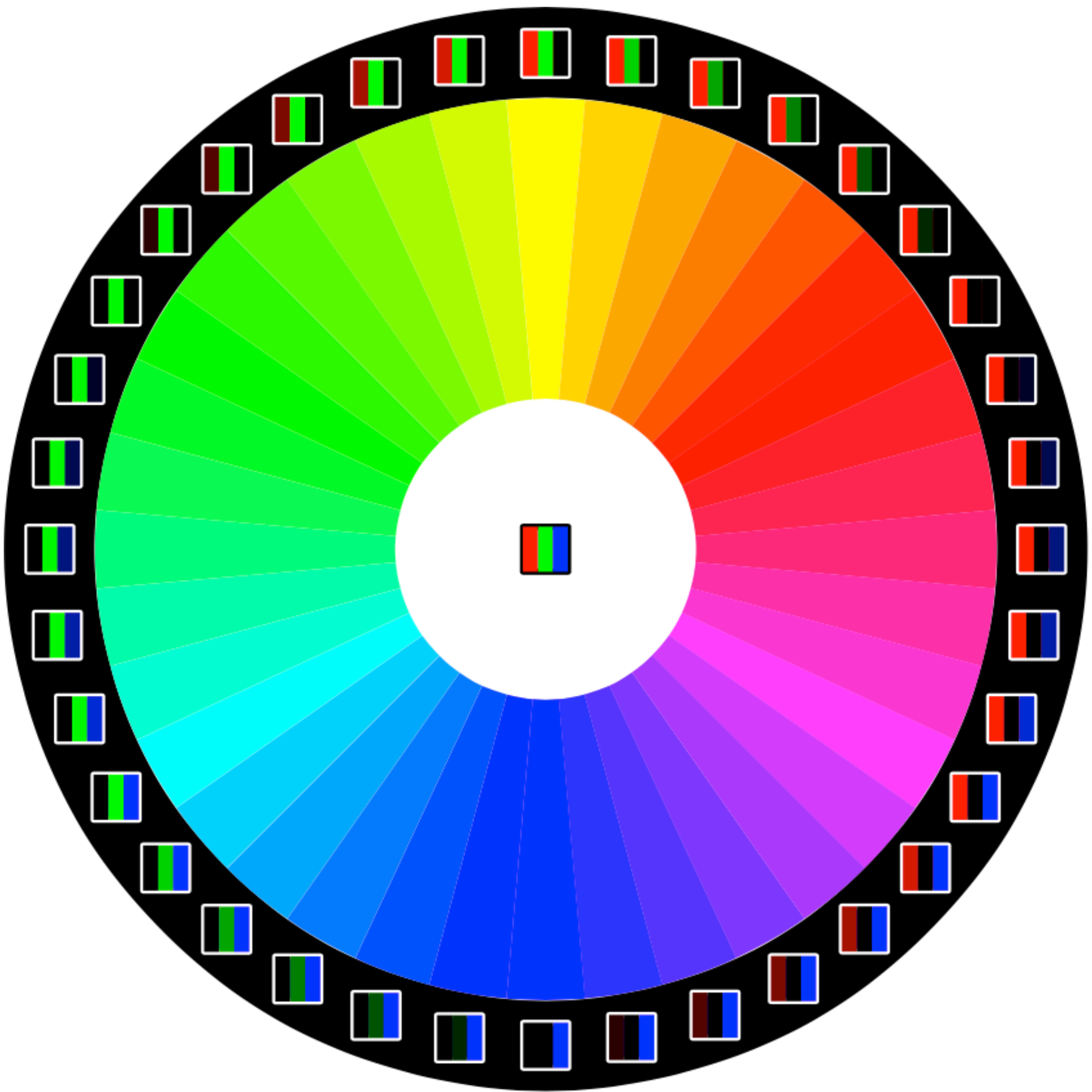
Thanks to P Stark

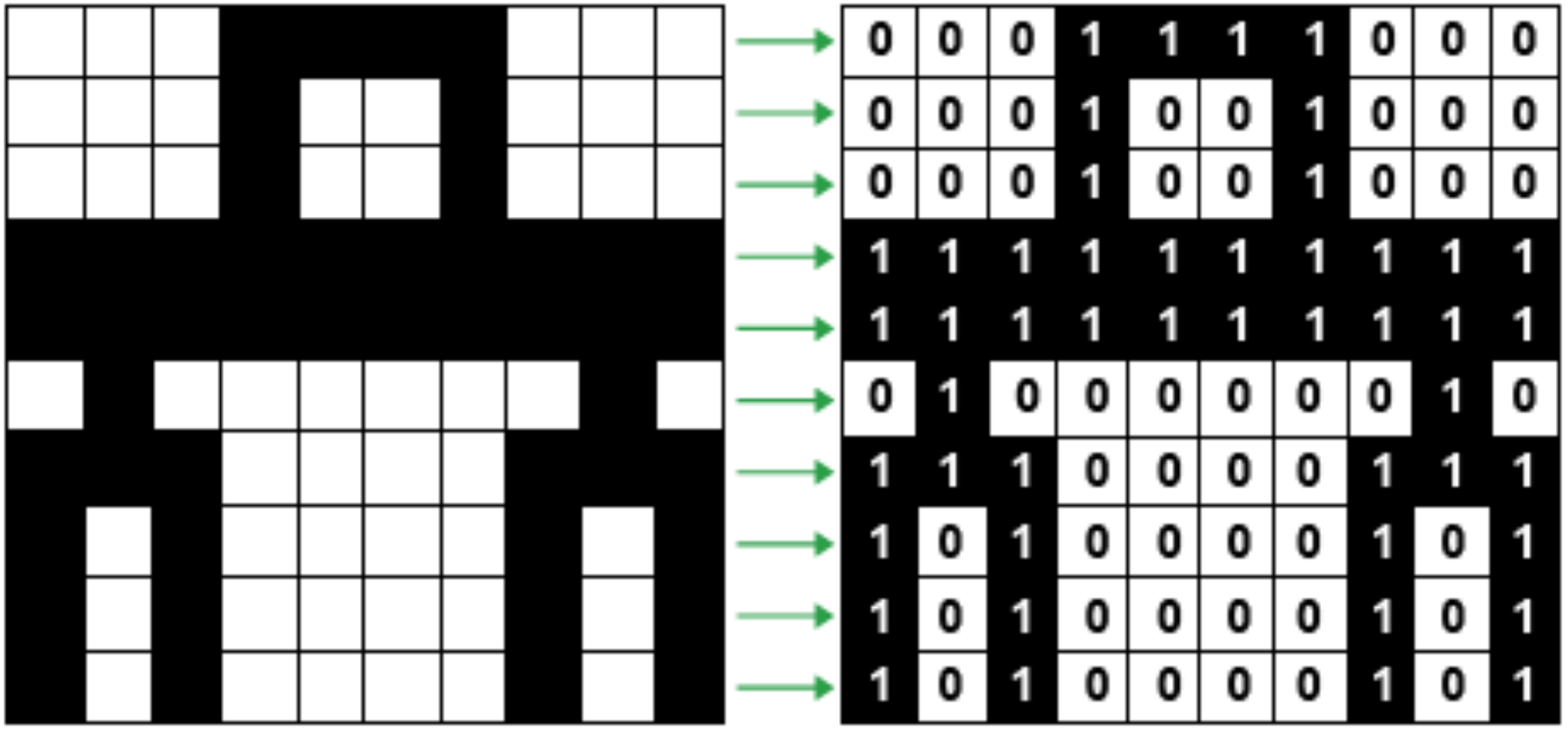
But, numbers?

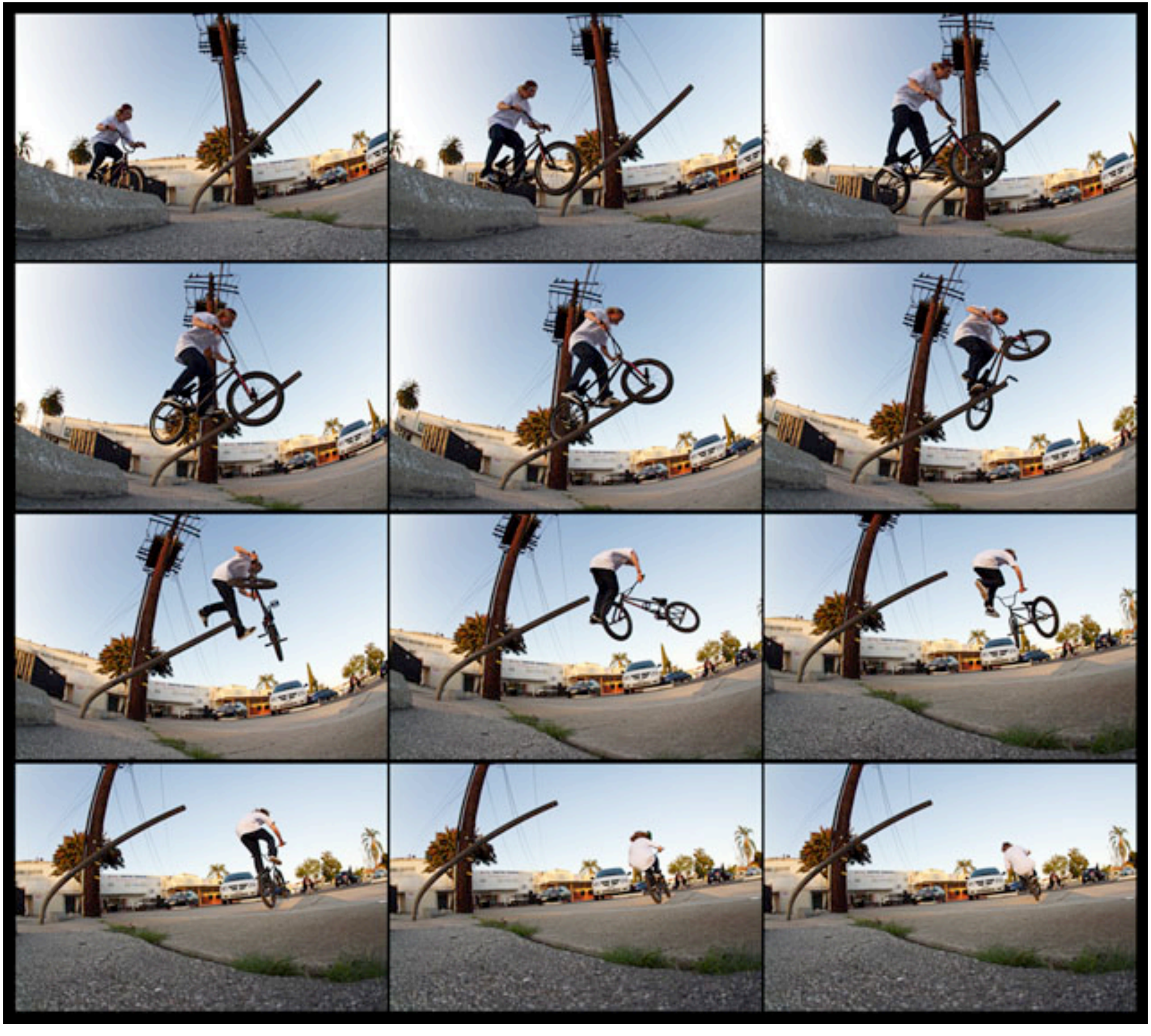
How do you represent things that aren’t numbers

…using numbers?

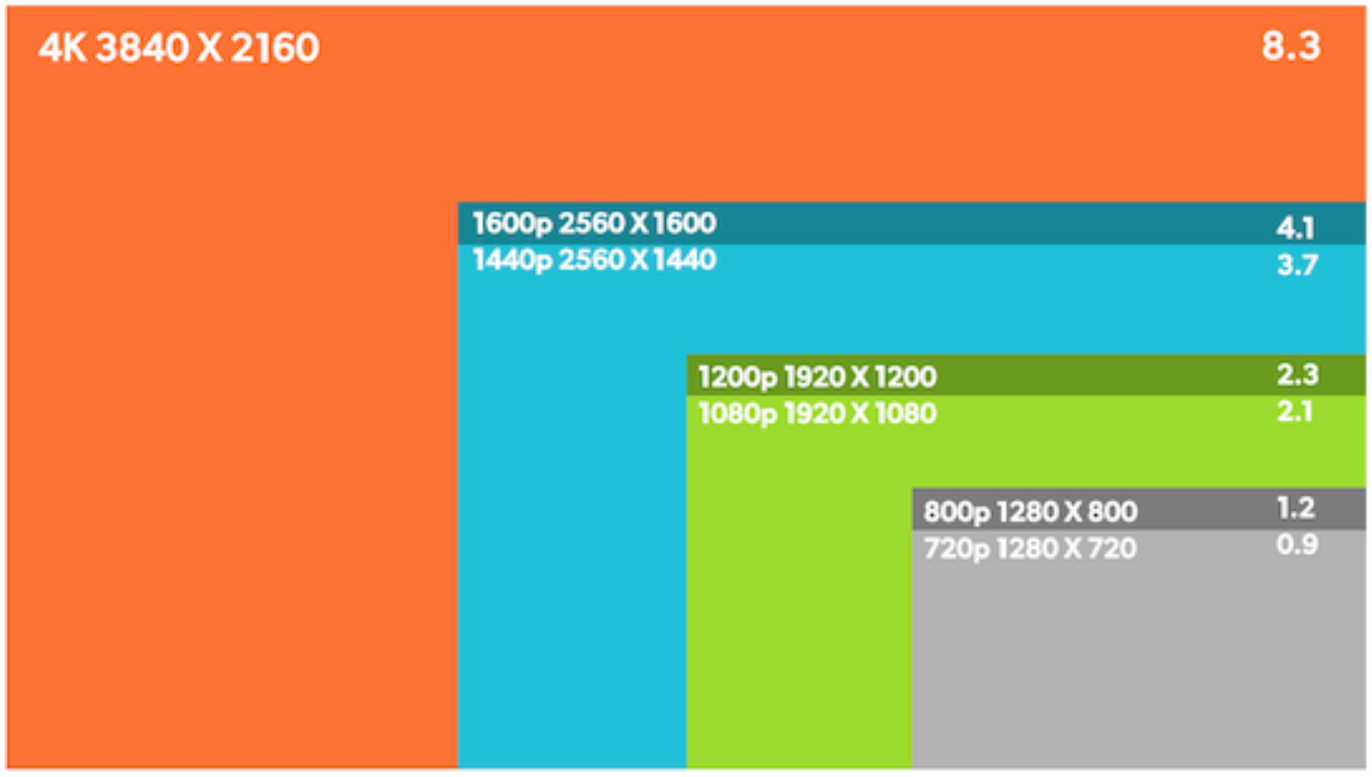
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL




0 0 0 1 1 1 1 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 1 0 0 1 0 0 0
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
0 1 0 0 0 0 0 0 1 0
1 1 1 0 0 0 0 1 1 1
1 0 1 0 0 0 0 1 0 1
1 0 1 0 0 0 0 1 0 1
1 0 1 0 0 0 0 1 0 1



Maybe not so fast?



3840 x 2160 x 3 x 60 is about 1.5 billion numbers each second.

Your CPU runs 3 billion operations each second.

Abstraction

- At the very lowest level, electrons flowing through really, **really** small circuits.
- But we don’t work with electrons when we write code.
- **Abstractions** give you a better/stronger/easier **interface**.

Abstraction

High-Level Language

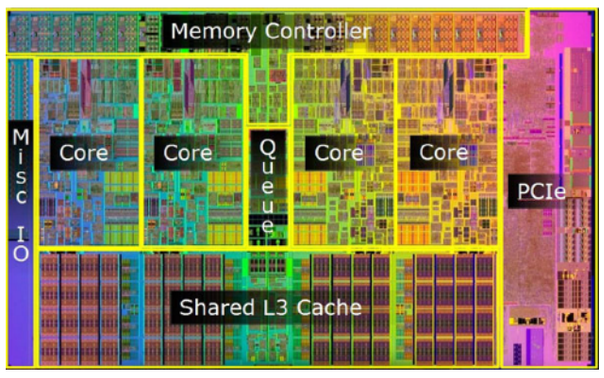
```
function draw() {
  background(255);

  for (var i = 0; i < bubbles.length; i++) {
    var bubble = bubbles[i];

    if (dist(mouseX, mouseY, bubble.x, bubble.y) < bubble.radius) {
      if (mouseIsPressed) {
        bubbles.splice(i, 1); // remove this bubble!
      }
      fill(255, 200, 200, 200);
    } else {
```

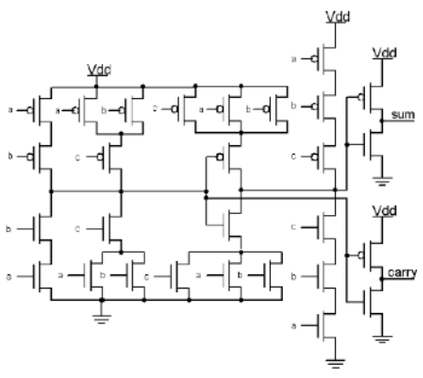
Machine Code & Assembly Language

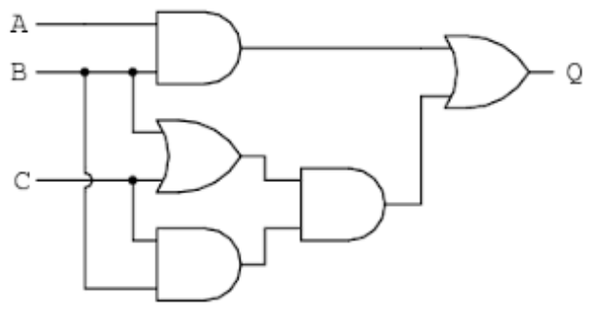
```
FE04-   CA
FE05-   D0 16
FE07-   C9 BA
FE09-   D0 BB
FE0B-   85 31
FE0D-   A5 3E
FE0F-   91 40
FE11-   E6 40
```

```
    DEX
    BNE   $FE1D
    CMP   #$BA
    BNE   $FDC6
    STA   $31
    LDA   $3E
    STA   ($40),Y
    INC   $40
```

Physical Hardware





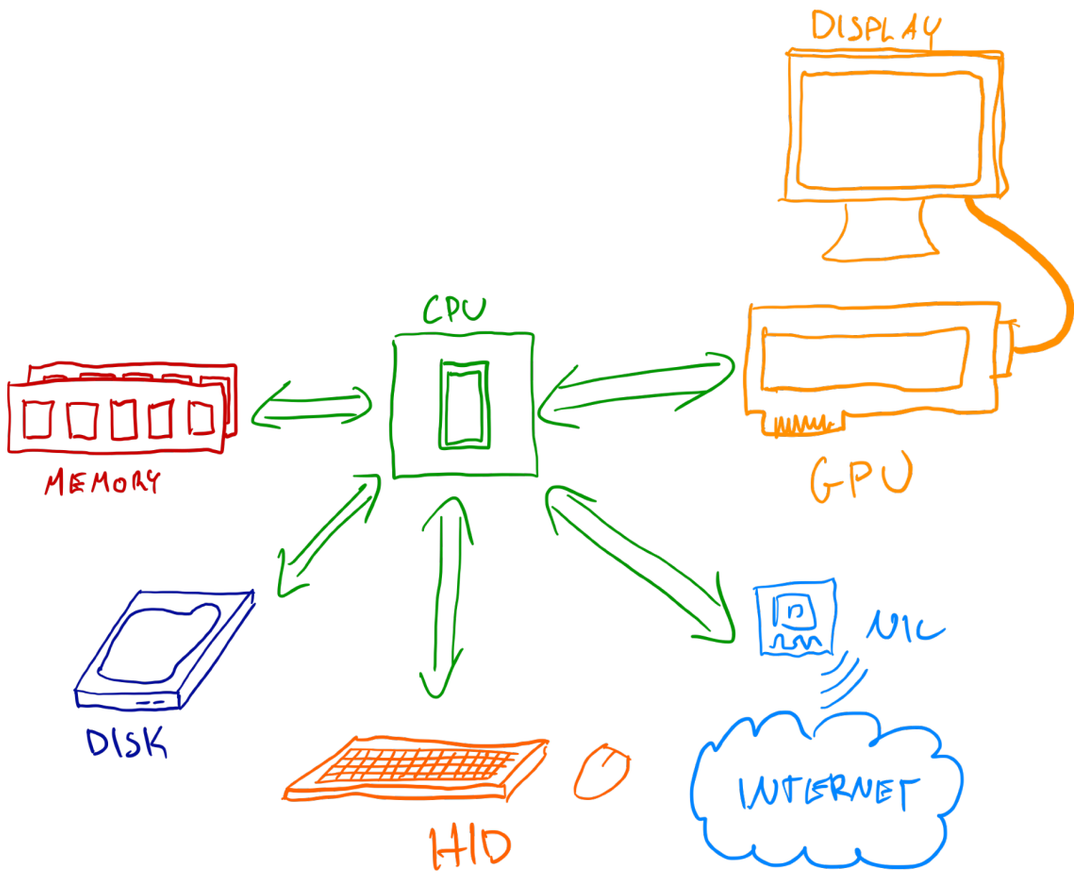


Libraries

- All code uses other code: **nothing stands alone**.

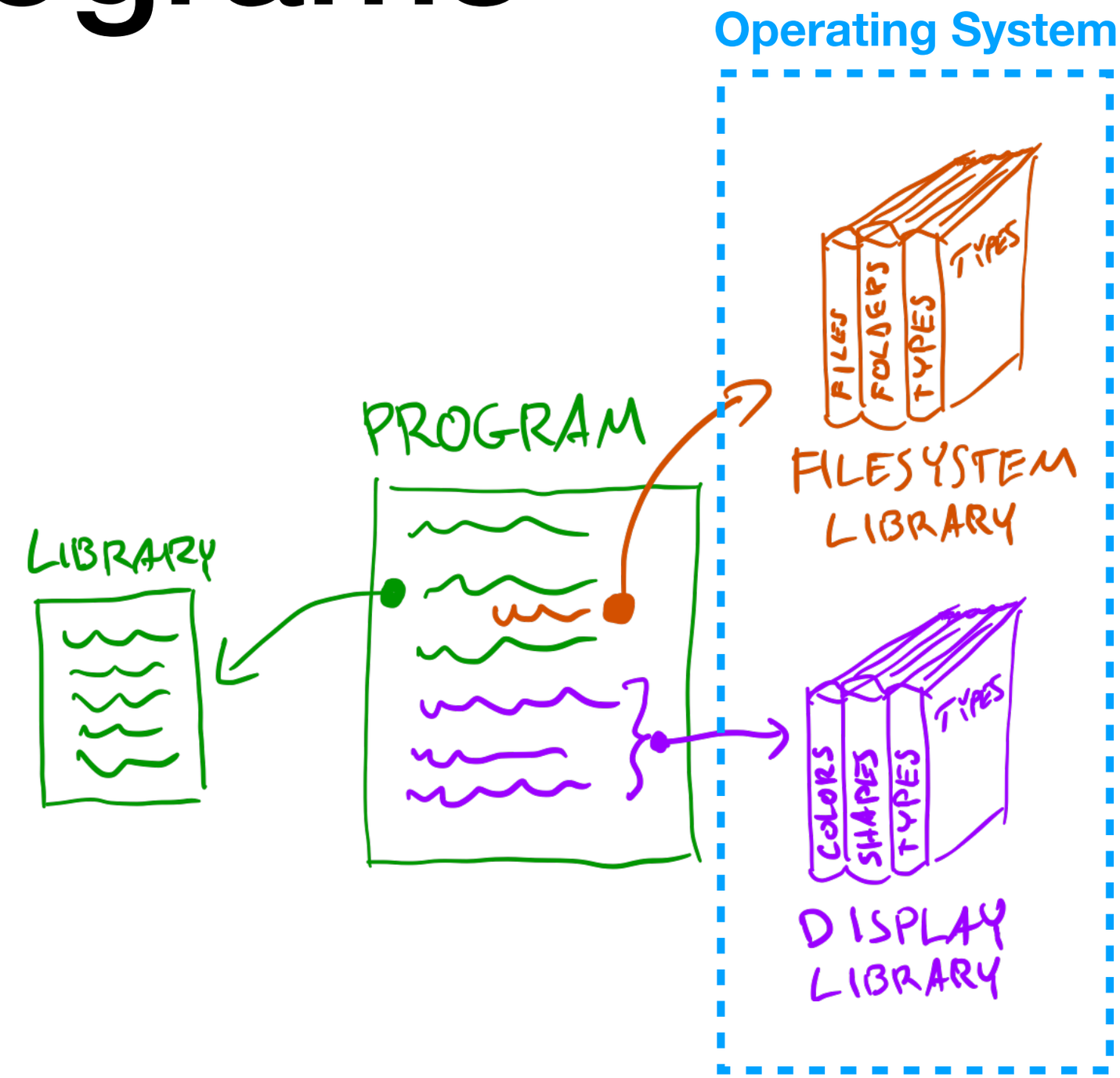
- Your computer & operating system come with some built-in code you can use, the **standard library:**
  - loading and writing files to **disk**
  - drawing pixels to a **display**
  - input from **human interface devices**

- You can use “third-party” libraries too, and you will. **A lot.**



Programs

- The computer runs your program literally.
- The only things it knows how to do are other functions.
- Your program **mixes, matches, and composes those other functions**.
- The language you use must be **specific** and **unambiguous**.
    - *Human languages are not those things.*



Programming

- Wikipedia says:

- **Computer programming** (often shortened to **programming**) is a process that leads from an original formulation of a computing problem to executable computer programs.

True but USELESS

Programming

- A way of describing a **process**, making it permanent.

- Lets you make new systems & new tools.

Looking at Code

- Bubbles!

How does this work?

- What happens when you open `index.html`?
- Browser loads your HTML/CSS/JavaScript from a file on disk
- Converts that JavaScript into machine code
- Machine code runs on your CPU

Programming “Environment”

- Text Editor: Atom? Sublime Text 3? (TextMate?)

- editor.p5js.org

Challenges

- Syntax
    - Symbolic structure
    - Logic
- Libraries
    - Libraries echo, but each is unique