The Hardware/Software Interface

CSE 351 Autumn 2020

Instructor:

Justin Hsia

Teaching Assistants:

Aman Mohammed
Ami Oka
Callum Walker
Cosmo Wang
Hang Do
Jim Limprasert
Joy Dang
Julia Wang
Kaelin Laundry
Kyrie Dowling
Mariam Mayanja
Shawn Stanley
Yan Zhe Ong

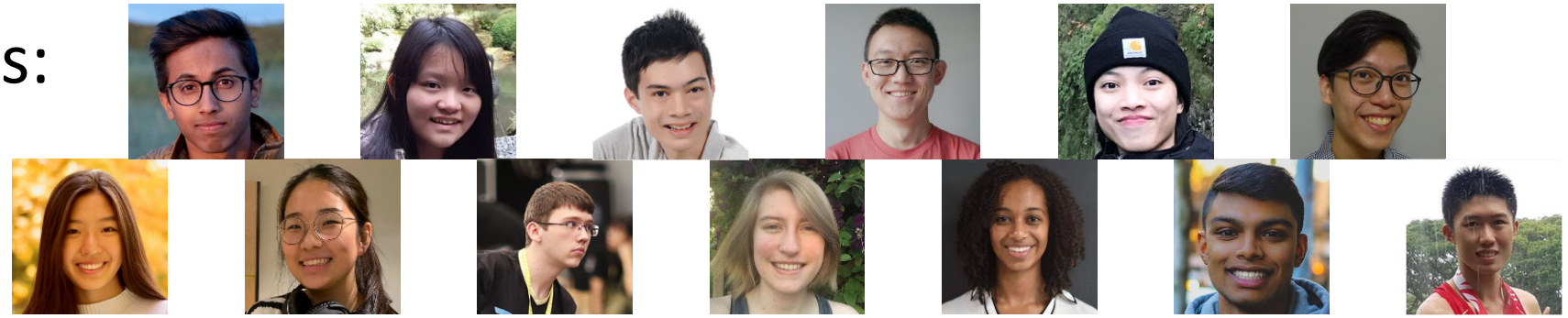
http://xkcd.com/676/

Lecture Outline

- **Course Introduction**
- Course Policies
  - Remote Instruction
  - https://courses.cs.washington.edu/courses/cse351/20au/syllabus
- Binary and Numerical Representation

Introductions: Course Staff

- Instructor: just call me Justin
  - CSE Assistant Teaching Professor – 6th time teaching 351
  - Learn more about me and the staff on the course website!
- TAs:



  - Available in section, office hours, and on Ed Discussion
  - An invaluable source of information and help
- Get to know us
  - We are here to help you succeed!

Introductions: You!

- ~280 students registered, split across two lectures
- CSE majors, EE majors, and more
  - Most of you will find almost everything in the course new
  - Many of you are new to CSE and/or UW!
- Get to know each other and help each other out!
  - Learning is much more fun with friends
  - Working well with others is a valuable life skill
  - Diversity of perspectives expands your horizons
  - A lot of work this work quarter is expected to be done in groups, including a portion of each exam

Welcome to CSE351!

433333443444443333433433333344433333333333
3444343333344333
43333434433334333334333333343433
333
4333433444333343
44333334444443433344444
444434443444443333433433333344433

HW/SW Interface

- Our goal is to teach you the key abstractions "under the hood"
  - How does your source code become something that your computer understands?
  - What happens as your computer is executing one or more processes?

Welcome to CSE351!

433333443444443333433433333344433333333333
3444343333344333
433343443433343333433433333343433


333
4333433444333343
443333344444434333344444
444434443444443333433433333344433

HW/SW Interface

- This is an *introduction* that will:
  - Profoundly change/augment your view of computers and programs
  - Leave you impressed that computers ever work

Code in Many Forms

```
if (x != 0) y = (y+z)/x;
```

Compiler

```
    cmpl   $0, -4(%ebp)
    je     .L2
    movl   -12(%ebp), %eax
    movl   -8(%ebp), %edx
    leal   (%edx,%eax), %eax
    movl   %eax, %edx
    sarl   $31, %edx
    idivl  -4(%ebp)
    movl   %eax, -8(%ebp)
.L2:
```

Assembler

```
10000011011111000010010000011100000000000
0111010000011000
10001011010001000010010000010100
10001011010001100010010100010100
10001101000001000000010
1000100111000010
11000001111111010000111111
11110111011111000010010000011100
10001001010001000010010000011000
```

High Level Language (*e.g.*, C, Java)

Assembly Language

Machine Code

Roadmap

How does your source code become something that your computer understands?

C:

```
car *c = malloc(sizeof(car));
c->miles = 100;
c->gals = 17;
float mpg = get_mpg(c);
free(c);
```

Java:

```
Car c = new Car();
c.setMiles(100);
c.setGals(17);
float mpg =
    c.getMPG();
```

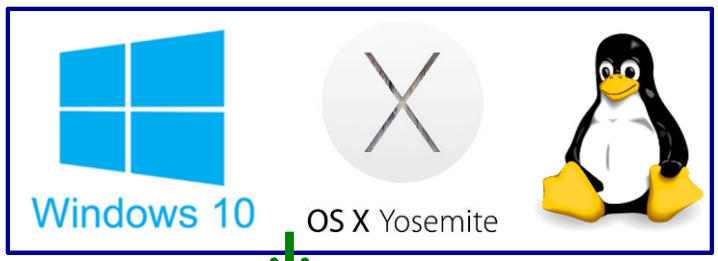
Assembly language:

```
get_mpg:
    pushq   %rbp
    movq    %rsp, %rbp
    ...
    popq    %rbp
    ret
```

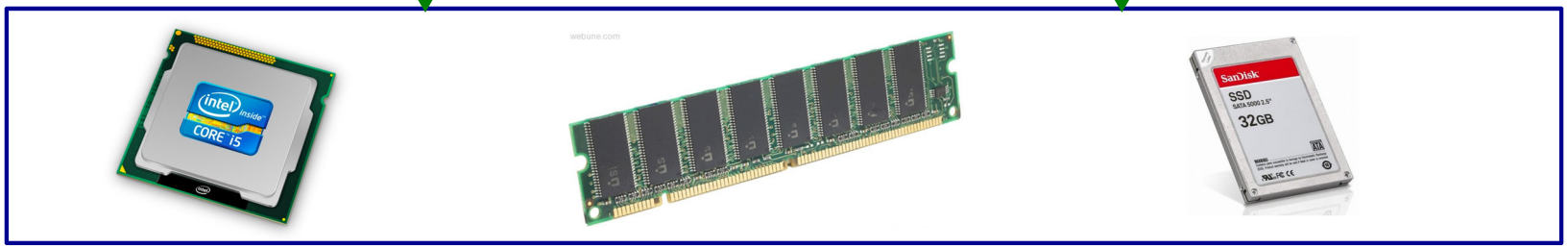
Machine code:

```
0111010000011000
100011010000010000000010
100010011000010
11000001111110100011111
```

OS:



Computer system:



- Memory & data
- Integers & floats
- x86 assembly
- Procedures & stacks
- Executables
- Arrays & structs
- Memory & caches
- Processes
- Virtual memory
- Memory allocation
- Java vs. C

Roadmap

What happens as your computer is executing one or more processes?

C:

```
car *c = malloc(sizeof(car));
c->miles = 100;
c->gals = 17;
float mpg = get_mpg(c);
free(c);
```

Java:

```
Car c = new Car();
c.setMiles(100);
c.setGals(17);
float mpg =
    c.getMPG();
```

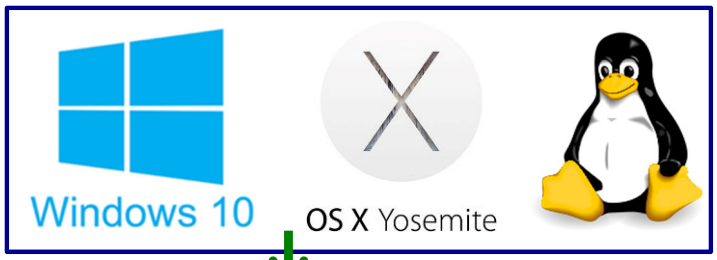
Assembly language:

```
get_mpg:
    pushq   %rbp
    movq    %rsp, %rbp
    ...
    popq    %rbp
    ret
```

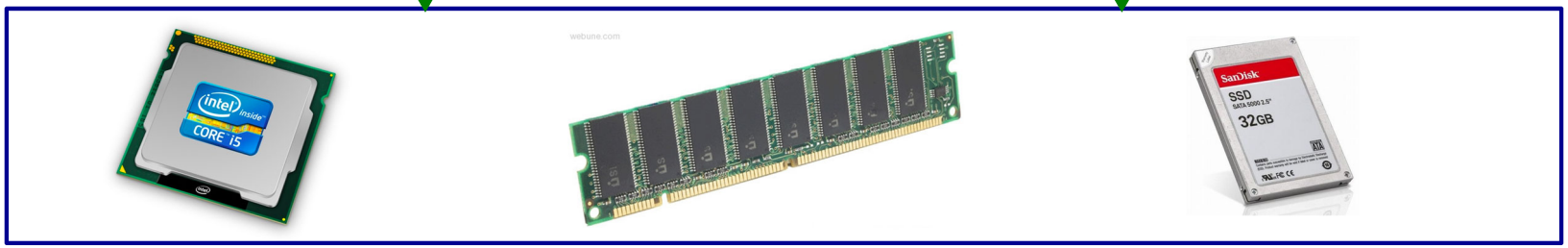
Machine code:

```
0111010000011000
100011010000010000000010
100010011000010
11000001111110100011111
```

OS:



Computer system:



- Memory & data
- Integers & floats
- x86 assembly
- Procedures & stacks
- Executables
- Arrays & structs
- Memory & caches
- Processes
- Virtual memory
- Memory allocation
- Java vs. C

Lecture Outline

- Course Introduction
- **Course Policies**
  - Remote Instruction
  - https://courses.cs.washington.edu/courses/cse351/20au/syllabus
- Binary and Numerical Representation

Bookmarks

- Website: https://courses.cs.washington.edu/courses/cse351/20au/
  - Schedule, policies, materials, videos, assignment specs, etc.
- Discussion: https://us.edstem.org/courses/2402/discussion/
  - Announcements made here
  - Ask and answer questions – staff will monitor and contribute

- Lessons: https://us.edstem.org/courses/2402/lessons/
  - Readings, lecture questions, homework
- Gradescope: https://www.gradescope.com/courses/159941
  - Lab submissions
- Canvas: https://canvas.uw.edu/courses/1396727/
  - Calendar, groups, grade book

351 Remote Instruction

- All meetings (lecture, section, office hours) via **Zoom**
  - Sign in to washington.zoom.us with "Login with SSO"
  - Find Zoom URLs via Canvas Calendar
  - Lectures and section presentations will be recorded
  - If your connection supports it, we would love to see your face (video on), but not required – highly recommend turning on camera during small group work and office hours
  - Watch your audio – background noise can be disruptive
- Practice task: turn on your mic  and say "hello!"
- Practice task: turn on your camera  and wave!
  - You may turn off your camera afterwards

351 Remote Instruction

- This class is highly *conceptual* and meant to be digested gradually
  - **Readings** (and quizzes) need to be done before lecture
  - **Lectures** will include practice problems and homework time
  - **Homework** following most lectures, due two lectures later
- Questions
  - Before and during lecture, there will be a monitored lecture discussion post
  - During group work, questions should be asked verbally or via chat (TAs will circulate through groups)
- Practice task: open the Group Chat  and type your favorite ice cream flavor

351 Remote Instruction

- Group work will be emphasized in this class
  - Lecture and section will use breakout rooms – you will get the most out of it if you actively participate!
  - Many assignments allow collaboration – talking to classmates will help you synthesize concepts and terminology
  - Self-select into small groups on Canvas (https://canvas.uw.edu/courses/1396727/groups#tab-105670)
    - If you are not part of a group on Canvas, then I will randomly assign you to one each lecture
    - You can freely change groups throughout the quarter
  - Responsibility for learning falls on *you*

Some fun topics that we will touch on

- Which of the following seems the most interesting to you? (vote in Ed Lessons)

a) What is a GFLOP and why is it used in computer benchmarks?

b) How and why does running many programs for a long time eat into your memory (RAM)?

c) What is stack overflow and how does it happen?

d) Why does your computer slow down when you run out of *disk* space?

e) What was the flaw behind the original Internet worm, the Heartbleed bug, and the Cloudbleed bug?

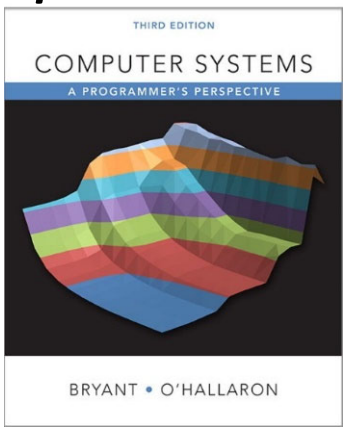
f) What is the meaning behind the different CPU specifications? (*e.g.*, # of cores, size of cache)

351 Remote Instruction

- Extenuating circumstances
  - Students (and staff) are in an extremely varied set of circumstances currently and we are cognizant of that
  - For formal accommodations, go through Disability Resources for Students (DRS)
  - We will try to be accommodating otherwise, but the earlier you reach out, the better
- Don't suffer in silence – talk to a staff member!

Reference Material

- The readings on Ed Lessons constitute a mini-textbook for this course
- *Computer Systems: A Programmer's Perspective*
  - Randal E. Bryant and David R. O'Hallaron
  - Website: http://csapp.cs.cmu.edu
  - North American 3rd edition
  - Optional, additional readings



- C reference (physical or online)
  - *The C Programming Language* (Kernighan and Ritchie)
  - *C: A Reference Manual* (Harbison and Steele)
  - http://www.cplusplus.com

Grading

groupwork allowed

- **Readings:** 5%

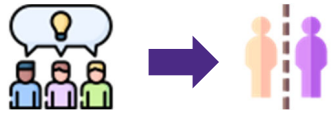
  - One attempt per question (completion)
- **Homework:** 20% total

  - Unlimited submission attempts (autograded correctness)

individual work

- **Labs:** 40% total

  - Last submission graded (correctness)
- **Exams:** Midterm (15%) and Final (15%)



  - Take-home; group test, individual test, optional retake
- **EPA:** Effort, Participation, and Altruism (5%)

Lab Collaboration and Academic Integrity

- All submissions are expected to be yours and yours alone
- You are encouraged to discuss your assignments with other students (*ideas*), but we expect that what you turn in is yours
- It is NOT acceptable to copy solutions from other students or to copy (or start your) solutions from the Web (including Github)
- Our goal is that *YOU* learn the material so you will be prepared for exams, interviews, and the future

EPA

- Encourage class-wide learning!
- Effort
  - Attending office hours, completing all assignments
  - Keeping up with Ed Discussion activity
- Participation
  - Making the class more interactive by asking questions in lecture, section, office hours, and on Ed Discussion
  - Lecture question voting
- Altruism
  - Helping others in section, office hours, and on Ed Discussion

To-Do List

- Admin
  - Explore/read the course website *thoroughly*
  - Check that you can access Ed Discussion & Lessons
  - **Get your machine set up to access the CSE Linux environment (CSE VM or attu) *as soon as possible***
  - Optionally, sign up for CSE 391: System and Software Tools
- Assignments
  - Pre-Course Survey and hw0 due Friday (10/2)
  - hw1 and Lab 0 due Monday (10/5)

Lecture Outline

- Course Introduction
- Course Policies
  - Remote instruction
  - https://courses.cs.washington.edu/courses/cse351/20au/syllabus
- **Binary and Numerical Representation**

Reading Review

- Terminology:
  - numeral, digit, base, symbol, digit position, leading zeros
  - binary, bit, nibble, byte, hexadecimal
  - numerical representation, encoding scheme

- Questions from the Reading?

Review Questions

- What is the *decimal value* of the numeral 107_8?

$1\times8^2 + 0\times8^1 + 7\times8^0$

position: 2 1 0

$64 + 0 + 7$

$= 71$

A. 71

B. 87

C. 107

D. 568

- What is the decimal number 108 in hex?

(base 16) $16^0 = 1$, $16^1 = 16$, $16^2 = 256$

A. 0x6C

B. 0xA8

C. 0x108

D. 0x612

$108 = 96 + 12$

$= 6\times16^1 + 12\times16^0$

$= 0x6C$

- Convert 0b0100110110101101 in hex.

$16 = 2^4$

1 hex digit ↔ 4 bits

0x4DAD

- Convert 0x3C9 to binary.

0b0011 1100 1001

could drop leading zeros

Base Comparison

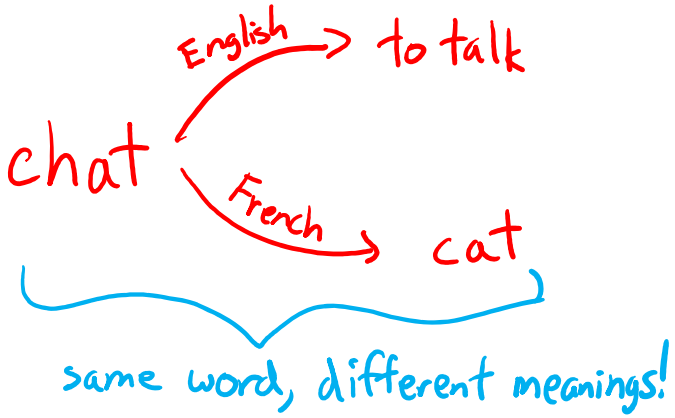
- Why does all of this matter?
  - *Humans* think about numbers in **base 10**, but *computers* "think" about numbers in **base 2**
  - Binary encoding is what allows computers to do all of the amazing things that they do!
- You should have this table memorized by the end of the class
  - Might as well start now!

MEMORIZE ME!!!

Base 10	Base 2	Base 16
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Numerical Encoding

- **AMAZING FACT: You can represent *anything* countable using numbers!**
  - Need to agree on an encoding
  - Kind of like learning a new language



- Examples:
  - Decimal Integers: 0→0b0, 1→0b1, 2→0b10, etc.
  - English Letters: CSE→0x435345, yay→0x796179
  - Emoticons: 😃 0x0, 😟 0x1, 😎 0x2, 😇 0x3, 😈 0x4, 🙋 0x5

Binary Encoding

1 bit:
0b0 → thing 1
0b1 → thing 2

2 bits:
0b00 → thing 1
0b01 → thing 2
0b10 → thing 3
0b11 → thing 4

- With n binary digits, how many "things" can you represent? 2^n things
  - Need n binary digits to represent N things, where $2^n \geq N$
  - Example: 5 binary digits for alphabet because $2^5 = 32 > 26$

- A binary digit is known as a bit
- A group of 4 bits (1 hex digit) is called a nibble
- A group of 8 bits (2 hex digits) is called a byte
  - 1 bit → 2 things, 1 nibble → 16 things, 1 byte → 256 things

So What's It Mean?

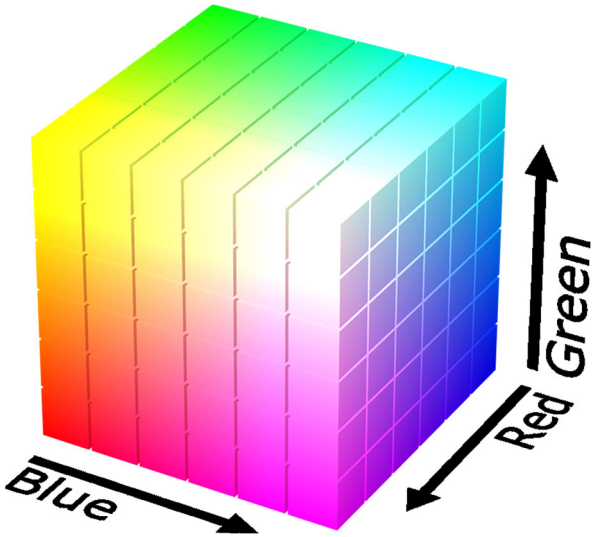
- *A sequence of bits can have many meanings!*

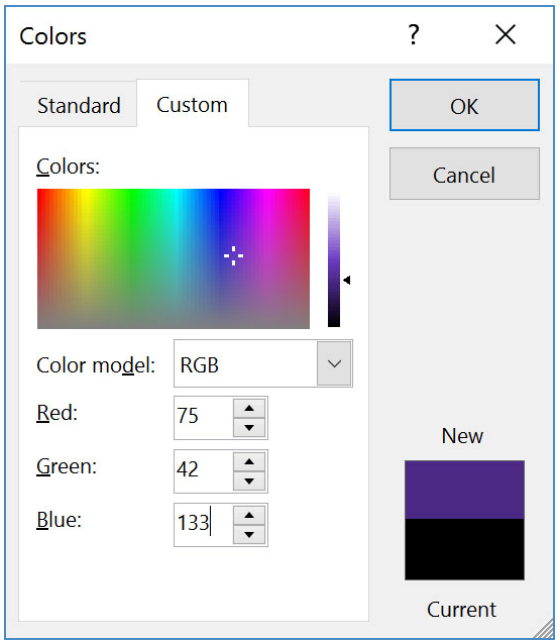
- Consider the hex sequence 0x4E6F21
  - Common interpretations include:
    - The decimal number 5140257
    - The characters "No!"
    - The background color of this slide
    - The real number 7.203034×10^{-39}

- It is up to the program/programmer to decide how to interpret the sequence of bits

Binary Encoding – Colors

- RGB – Red, Green, Blue
  - Additive color model (light): byte (8 bits) for each color
  - Commonly seen in hex (in HTML, photo editing, etc.)
  - Examples: **Blue**→0x0000FF, **Gold**→0xFFD700, White→0xFFFFFF, **Deep Pink**→0xFF1493





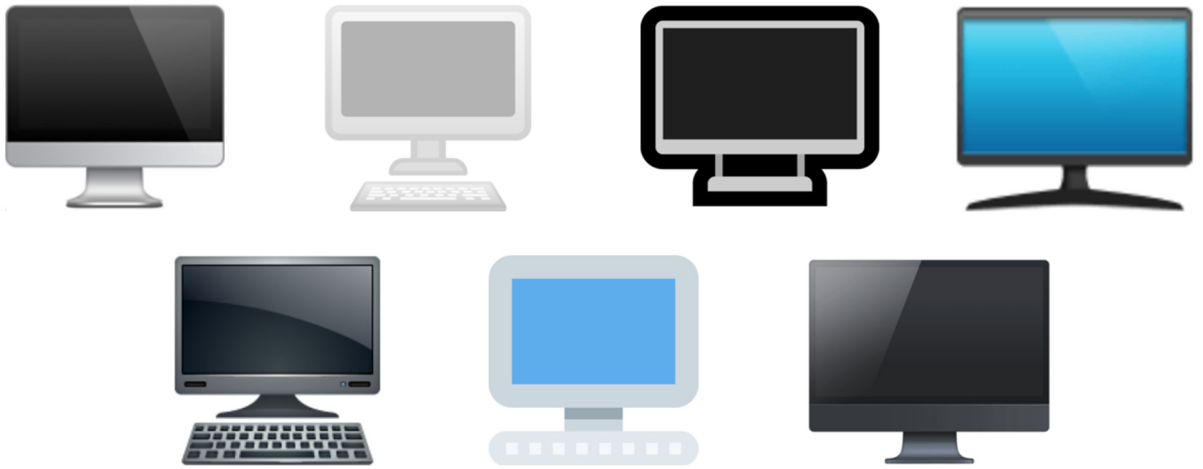
Binary Encoding – Characters/Text

- ASCII Encoding (www.asciitable.com)
  - American Standard Code for Information Interchange

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

Binary Encoding – Emoji

- Handled as part of the Unicode Standard and managed by the Unicode Consortium
  - Adds new emojis every year, though adoption often lags: 🤷
- Emojipedia demo: http://www.emojipedia.com
  - Desktop Computer: 🖥
  - Code points: U+1F5A5, U+FE0F
  - Display:



Binary Encoding – Files and Programs

- At the lowest level, all digital data is stored as bits!

- Layers of abstraction keep everything comprehensible
  - Data/files are groups of bits interpreted by program
  - Program is actually groups of bits being interpreted by your CPU

- Computer Memory Demo (if time)
  - From vim: `%!xxd`
  - From emacs: `M-x hexl-mode`

Summary

- Humans think about numbers in decimal; computers think about numbers in binary
  - Base conversion to go between them
  - Hexadecimal is more human-readable than binary
- All information on a computer is binary
- Binary encoding can represent *anything*!
  - Computer/program needs to know how to interpret the bits