

The Hardware/Software Interface

CSE 351 Autumn 2019

Instructor:

Justin Hsia

Teaching Assistants:

Andrew Hu

Antonio Castelli

Cosmo Wang

Diya Joy

Ivy Yu

Kaelin Laundry

Maurice Montag

Melissa Birchfield

Millicent Li

Suraj Jagadeesh





Introductions: Course Staff

- ❖ Instructor: just call me Justin
 - CSE Lecturer – 5th time teaching 351
 - Learn more about me and the staff on the course website!

- ❖ TAs:



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

Introductions: You!

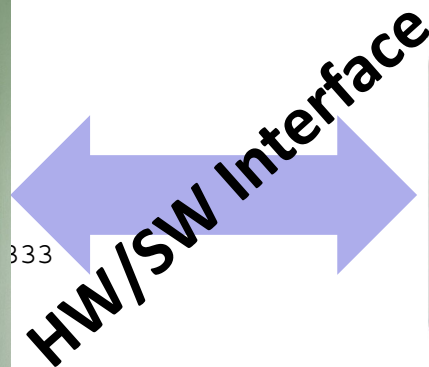
- ❖ ~250 students registered, split across two lectures
- ❖ CSE majors, EE majors, and more
 - Most of you will find almost everything in the course new
- ❖ Get to know each other and help each other out!
 - Learning is much more fun with friends
 - Working well with others is a valuable life skill
 - Diversity of perspectives expands your horizons

Welcome to CSE351!

43333344344444333334334333334443333333333
 3444343333344333
 43334344343334333343343333343433

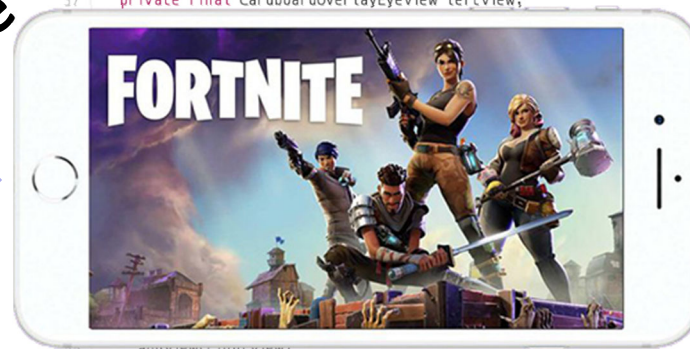


4333433444333343
 4433333444444434333344444
 44443444344444333343343333344433



```

29 import android.widget.ImageView;
30 import android.widget.LinearLayout;
31 import android.widget.TextView;
32
33 /**
34  * Contains two sub-views to provide a simple stereo HUD.
35  */
36 public class CardboardOverlayView extends LinearLayout {
37     private final CardboardOverlayEyeView leftView;
    
```



```

55
56
57 // Set some reasonable defaults.
58 setDepthOffset(0.01f);
59 setColor(Color.rgb(150, 255, 180));
60 setVisibility(View.VISIBLE);
61
62 textFadeAnimation = new AlphaAnimation(1.0f, 0.0f);
63 textFadeAnimation.setDuration(5000);
    
```

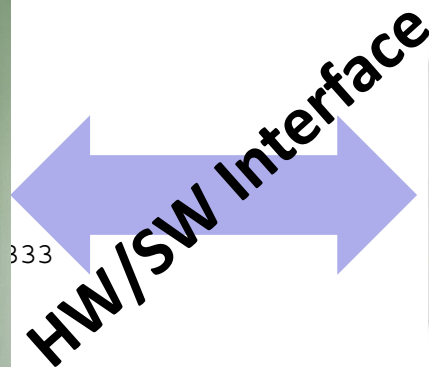
- ❖ Our goal is to teach you the key abstractions “under the hood”
 - How does your source code become something that your computer understands?
 - What happens as your computer is executing one or more processes?

Welcome to CSE351!

43333344344444333334334333334443333333333
 3444343333344333
 43334344343334333343343333343433

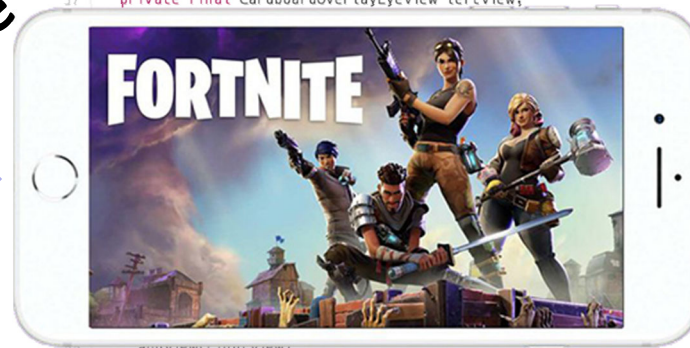


4333433444333343
 4433333444444434333344444
 44443444344444333343343333344433



```

29 import android.widget.ImageView;
30 import android.widget.LinearLayout;
31 import android.widget.TextView;
32
33 /**
34  * Contains two sub-views to provide a simple stereo HUD.
35  */
36 public class CardboardOver layView extends LinearLayout {
37     private final CardboardOver layEyeView leftView;
    
```



```

55
56
57 // Set some reasonable defaults.
58 setDepthOffset(0.01f);
59 setColor(Color.rgb(150, 255, 180));
60 setVisibility(View.VISIBLE);
61
62 textFadeAnimation = new AlphaAnimation(1.0f, 0.0f);
63 textFadeAnimation.setDuration(5000);
    
```

- ❖ 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

```
1000001101111100001001000001110000000000
0111010000011000
10001011010001000010010000010100
10001011010001100010010100010100
100011010000010000000010
1000100111000010
110000011111101000011111
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();
```

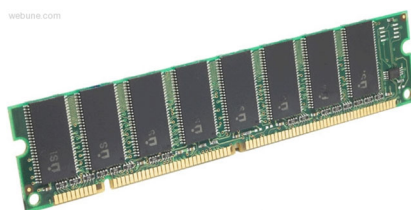
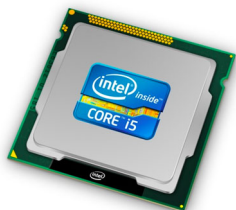
Assembly language:

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

Machine code:

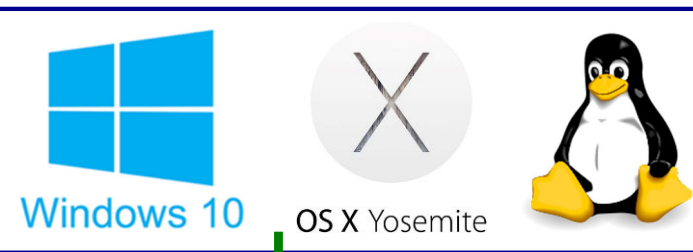
```
0111010000011000
100011010000010000000010
1000100111000010
110000011111101000011111
```

Computer system:



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

OS:



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();
```

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

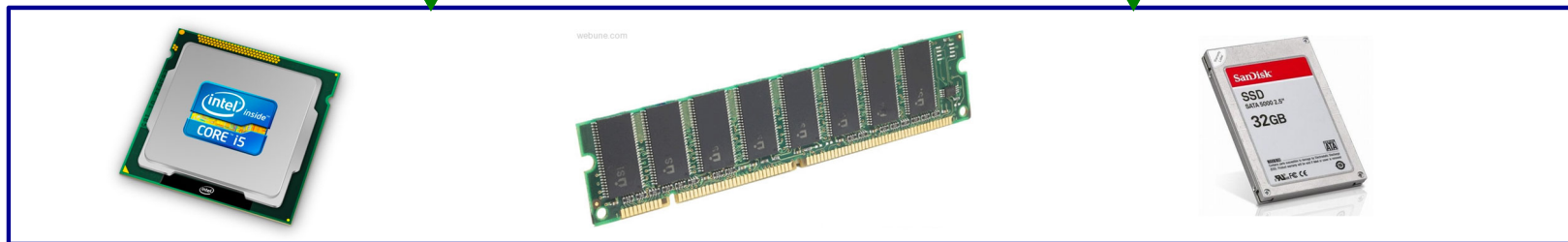
Assembly language:

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

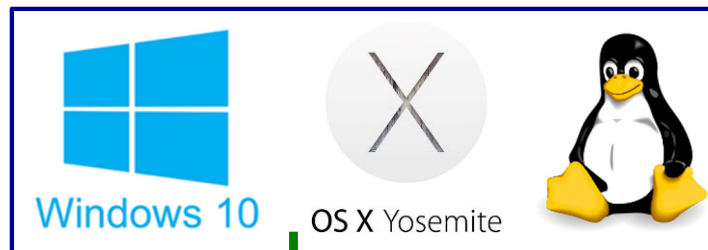
Machine code:

```
0111010000011000
100011010000010000000010
1000100111000010
110000011111101000011111
```

Computer system:



OS:



Lecture Outline

- ❖ Course Introduction
- ❖ **Course Policies**
 - <https://courses.cs.washington.edu/courses/cse351/19au/syllabus/>
- ❖ Binary

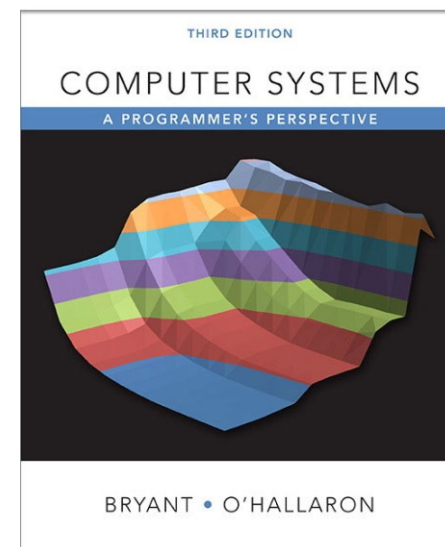
Bookmarks

- ❖ Course Website: <http://cs.uw.edu/351>
 - Schedule, policies, materials, videos, assignments, etc.
- ❖ Discussion: <http://piazza.com/washington/fall2019/cse351>
 - Announcements made here
 - Ask and answer questions – staff will monitor and contribute
- ❖ Gradescope: <https://www.gradescope.com/courses/55133>
 - Assignment submissions
- ❖ Canvas: <https://canvas.uw.edu/courses/1319076/>
 - Gradebook
- ❖ Poll Everywhere: <http://PollEv.com/justinh>
 - In-lecture voting

Textbooks

❖ *Computer Systems: A Programmer's Perspective*

- Randal E. Bryant and David R. O'Hallaron
- Website: <http://csapp.cs.cmu.edu>
- Must be (North American) **3rd edition**
 - <http://csapp.cs.cmu.edu/3e/changes3e.html>
 - <http://csapp.cs.cmu.edu/3e/errata.html>
- This book really matters for the course!
 - Lecture readings
 - Practice problems and homework



❖ A good C book – any will do

- *The C Programming Language* (Kernighan and Ritchie)
- *C: A Reference Manual* (Harbison and Steele)

Grading

- ❖ **Homework:** 20% total
 - Autograded; unlimited submission attempts
 - *Group work encouraged*
- ❖ **Labs:** 30% total
 - Graded by TAs; last submission graded
 - *Individual work only*
- ❖ **Exams:** Midterm (15%) and Final (30%)
 - Many old exams on course website
- ❖ **EPA:** Effort, Participation, and Altruism (5%)


Lab Collaboration and Academic Integrity

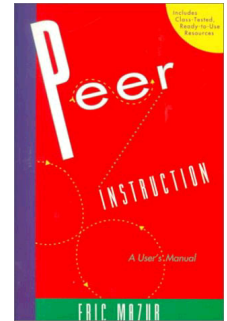
- ❖ All submissions are expected to be yours and yours alone
- ❖ You are encouraged to discuss your assignments with other students (*ideas*), but we expect that what you turn in is yours
- ❖ It is NOT acceptable to copy solutions from other students or to copy (or start your) solutions from the Web (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 Piazza activity
- ❖ Participation
 - Making the class more interactive by asking questions in lecture, section, office hours, and on Piazza
 - Peer instruction voting
- ❖ Altruism
 - Helping others in section, office hours, and on Piazza

Lecture Polling

- ❖ Increase real-time learning in lecture, test your understanding, increase student interactions
 - Lots of research supports its effectiveness
- ❖ Multiple choice question during lecture
 - 1 minute to decide on your own
 - 2-4 minutes in pairs to reach consensus
 - Learn through discussion & teaching
- ❖ Vote using  **Poll Everywhere**
 - Use website (<https://www.polleverywhere.com>) or app
 - Linked to your UWNetID



Some fun topics that we will touch on

- ❖ Which of the following seems the most interesting to you? (vote at <http://PollEv.com/justinh>)
 - 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, # and size of cache, supported memory types)

Tips for Success in 351

- ❖ Attend all lectures and sections
 - Avoid devices during lecture except for Poll Everywhere
- ❖ Do the textbook readings ahead of time
- ❖ **Learn by doing**
 - Can answer many questions by writing small programs
- ❖ Visit Piazza often
 - Ask questions and try to answer fellow students' questions
- ❖ Go to office hours
 - Even if you don't have specific questions in mind
- ❖ Find a study and homework group
- ❖ Start assignments early
- ❖ **Don't be afraid to ask questions**

To-Do List

❖ Admin

- Explore/read website *thoroughly*: <http://cs.uw.edu/351>
- Check that you are enrolled in Piazza; read posts
- Log in to Poll Everywhere
- **Get your machine set up for this class (VM or attu) *as soon as possible***

❖ Assignments

- Pre-Course Survey and hw0 due Friday (9/27)
- hw1 and Lab 0 due Monday (9/30)

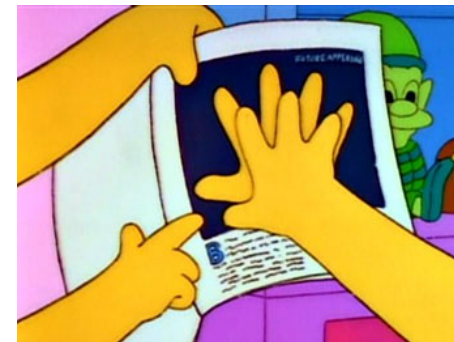
Lecture Outline

- ❖ Course Introduction
- ❖ Course Policies
- ❖ **Binary**
 - **Decimal, Binary, and Hexadecimal**
 - **Base Conversion**
 - **Binary Encoding**

Decimal Numbering System

- ❖ Ten **symbols**: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- ❖ Represent larger numbers as a sequence of **digits**
 - Each digit is one of the available symbols
- ❖ Example: 7061 in decimal (base 10)
 - $7061_{10} = (7 \times 10^3) + (0 \times 10^2) + (6 \times 10^1) + (1 \times 10^0)$

Octal Numbering System



- ❖ Eight symbols: 0, 1, 2, 3, 4, 5, 6, 7
 - Notice that we no longer use 8 or 9
- ❖ Base comparison:
 - Base 10: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12...
 - Base 8: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14...
- ❖ Example: What is 7061_8 in base 10?
 - $7061_8 = (7 \times 8^3) + (0 \times 8^2) + (6 \times 8^1) + (1 \times 8^0) = 3633_{10}$

Warmup Question

- ❖ What is 34_8 in base 10?
 - No voting for this question

A. 32_{10}

B. 34_{10}

C. 7_{10}

D. 28_{10}

E. 35_{10}

Binary and Hexadecimal

- ❖ Binary is base 2
 - Symbols: 0, 1
 - Convention: $2_{10} = 10_2 = \text{0b}10$
- ❖ Example: What is 0b110 in base 10?
 - $0\text{b}110 = 110_2 = (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 6_{10}$
- ❖ Hexadecimal (**hex**, for short) is base 16
 - Symbols? 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, **A, B, C, D, E, F**
 - Convention: $16_{10} = 10_{16} = \text{0x}10$
- ❖ Example: What is 0xA5 in base 10?
 - $0\text{x}A5 = A5_{16} = (10 \times 16^1) + (5 \times 16^0) = 165_{10}$

Polling Question

- ❖ Which of the following orderings is correct?
 - A. $0xC < 0b1010 < 11$
 - B. $0xC < 11 < 0b1010$
 - C. $11 < 0b1010 < 0xC$
 - D. $0b1010 < 11 < 0xC$
 - E. $0b1010 < 0xC < 11$

- ❖ Think on your own for a minute, then discuss with your neighbor(s)
 - Vote at <http://PollEv.com/justinh>

Converting to Base 10

- ❖ Can convert from any base *to* base 10
 - $0b110 = 110_2 = (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 6_{10}$
 - $0xA5 = A5_{16} = (10 \times 16^1) + (5 \times 16^0) = 165_{10}$
- ❖ We learned to think in base 10, so this is fairly natural for us
- ❖ **Challenge:** Convert into other bases (*e.g.* 2, 16)

Challenge Question

- ❖ Convert 13_{10} into binary

- ❖ Hints:
 - $2^3 = 8$
 - $2^2 = 4$
 - $2^1 = 2$
 - $2^0 = 1$

- ❖ Discuss with your neighbor(s)
 - No voting for this question

Converting from Decimal to Binary

- ❖ Given a decimal number N :
 - List increasing powers of 2 from *right to left* until $\geq N$
 - Then from *left to right*, ask is that (power of 2) $\leq N$?
 - If **YES**, put a 1 below and subtract that power from N
 - If **NO**, put a 0 below and keep going

❖ Example: 13 to binary

$2^4=16$	$2^3=8$	$2^2=4$	$2^1=2$	$2^0=1$

Converting from Decimal to Base B

- ❖ Given a decimal number N :
 - List increasing powers of B from *right to left* until $\geq N$
 - Then from *left to right*, ask is that (power of B) $\leq N$?
 - If **YES**, put *how many of that power go into N* and subtract from N
 - If **NO**, put a 0 below and keep going

- ❖ Example: 165 to hex

$16^2=256$	$16^1=16$	$16^0=1$

Converting Binary \leftrightarrow Hexadecimal

❖ Hex \rightarrow Binary

- Substitute hex digits, then drop any **leading zeros**
- Example: 0x2D to binary
 - 0x2 is 0b0010, 0xD is 0b1101
 - Drop two leading zeros, answer is 0b101101

❖ Binary \rightarrow Hex

- Pad with **leading zeros** until multiple of 4, then substitute each group of 4
- Example: 0b101101
 - Pad to 0b 0010 1101
 - Substitute to get 0x2D

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

Binary → Hex Practice

- ❖ Convert 0b100110110101101
 - How many digits?
 - Pad:
 - Substitute:

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

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!

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: $0 \rightarrow 0b0$, $1 \rightarrow 0b1$, $2 \rightarrow 0b10$, etc.
- English Letters: $CSE \rightarrow 0x435345$, $yay \rightarrow 0x796179$
- Emoticons: 😊 0x0, 😞 0x1, 😎 0x2, 😇 0x3, 😈 0x4, 🙋 0x5

Binary Encoding

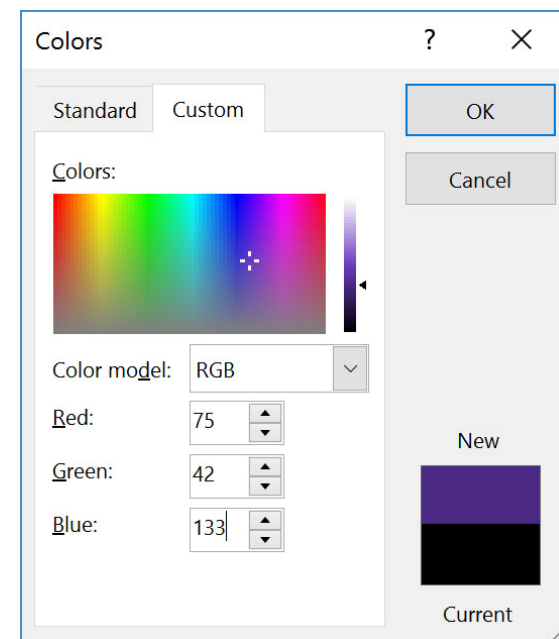
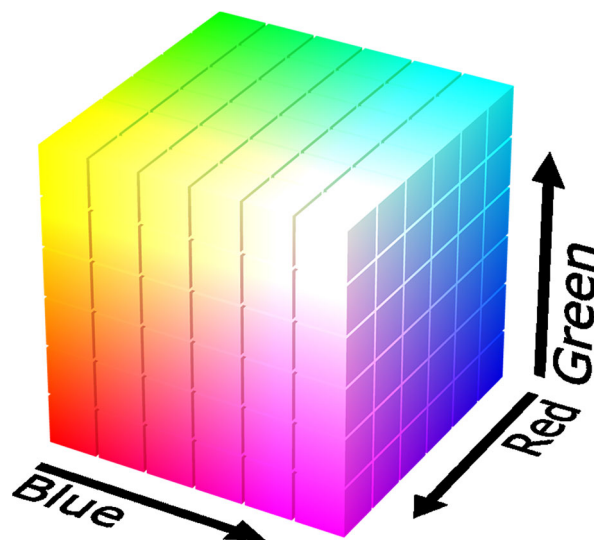
- ❖ With N binary digits, how many “things” can you represent?
 - Need N binary digits to represent n things, where $2^N \geq n$
 - Example: 5 binary digits for alphabet because $2^5 = 32 > 26$
- ❖ A binary digit is known as a **bit**
- ❖ A group of 4 bits (1 hex digit) is called a **nibble**
- ❖ A group of 8 bits (2 hex digits) is called a **byte**
 - 1 bit \rightarrow 2 things, 1 nibble \rightarrow 16 things, 1 byte \rightarrow 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		DEL

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