The Hardware/Software Interface

CSE 351 Autumn 2024

- **Instructor:**
- Ruth Anderson
- **Teaching Assistants:**
- Alexandra Michael
- Connie Chen
- Chloe Fong
- Chendur Jayavelu
- Joshua Tan
- Nikolas McNamee
- Nahush Shrivatsa
- Naama Amiel
- Neela Kausik
- Renee Ruan
- Rubee Zhao
- Samantha Dreussi
- Sean Siddens
- Waleed Yagoub

http://xkcd.com/676/

Lecture Outline

- **Course Introduction**
- Course Policies
  - https://courses.cs.washington.edu/courses/cse351/24au/syllabus.html
- Binary and Numerical Representation

Introductions: Course Staff

- **Instructor:** Ruth
- **TAs:** Alexandra, Connie, Chloe, Chendur, Josh, Nikolas, Nahush, Naama, Neela, Renee, Rubee, Samantha, Sean, Waleed
- Learn more about us on the course website!

- More than anything, we want you to feel
  - Welcome in this space
  - Able to learn and succeed in this course
  - Comfortable reaching out if you need help or want change

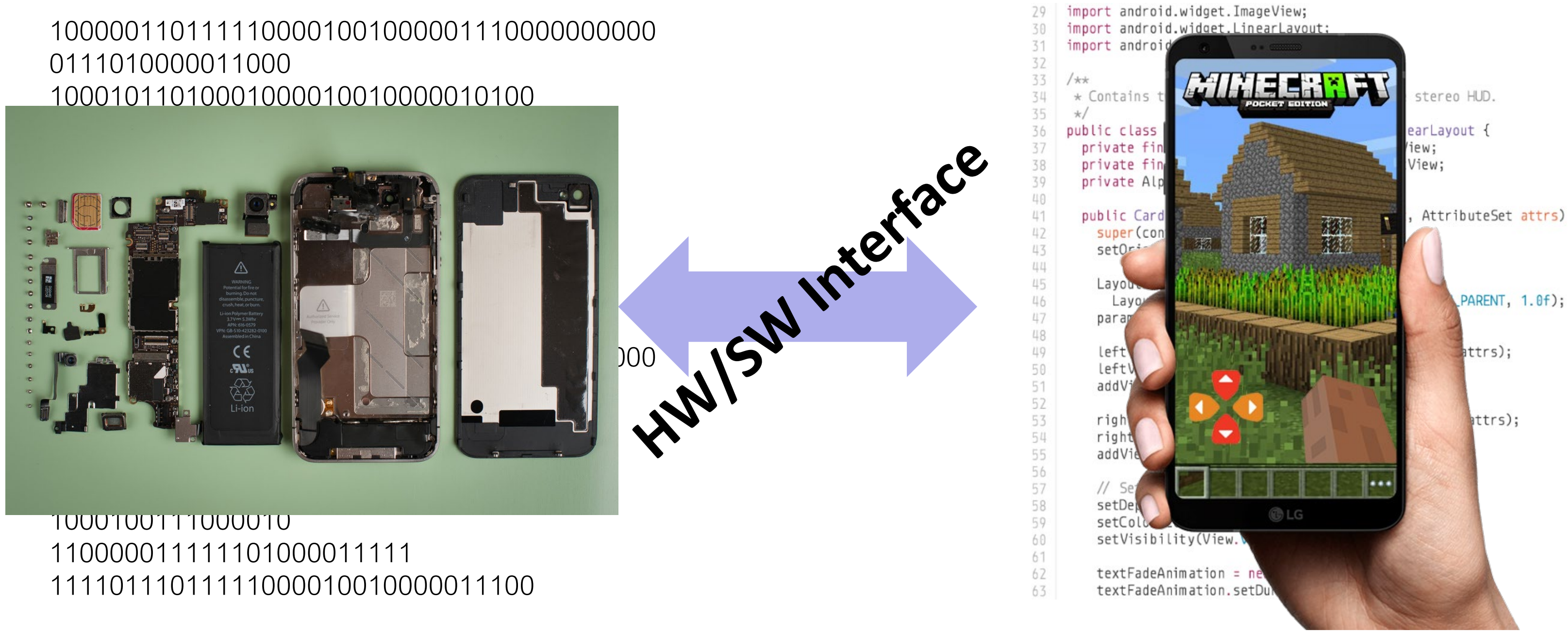
Introductions: You!

- ~300 students registered across 2 lecture sections

- CSE majors, ECE 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!



- 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?

Code in Many Forms

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

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

Compiler

Assembly Language

```
    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:
```

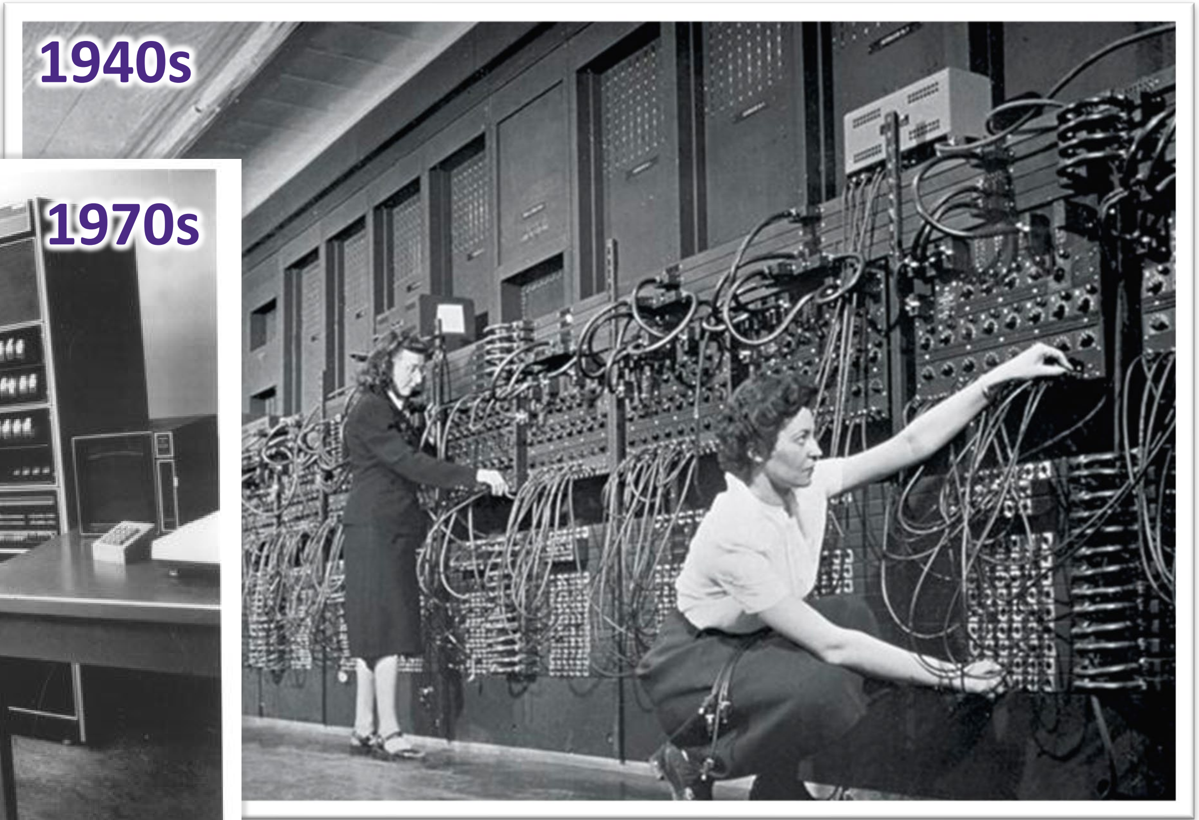
Assembler

Machine Code

```
1000001101111100001001000001110000000000
0111010000011000
10001011010001000010010000010100
10001011010001100010010100010100
100011010000010000000010
1000100111000010
110000011111101000011111
11110111011111000010010000011100
10001001010001000010010000011000
```

HW/SW Interface: Historical Perspective

- Hardware started out quite primitive



Jean Jennings (left), Marlyn Wescoff (center), and Ruth Lichterman program ENIAC at the University of Pennsylvania, circa 1946.
Photo: Corbis
http://fortune.com/2014/09/18/walter-isaacson-the-women-of-eniac/

https://s-media-cache-ak0.pinimg.com/564x/91/37/23/91372375e2e6517f8af128aab655e3b4.jpg

Course Perspective

- CSE351 will make you a better programmer
  - Learn how software really works
    - Understand some of the abstractions that exist between hardware and software, why they exist, and how they build upon each other
  - Why is this important?
    - Better debugging
    - Better basis for evaluating performance

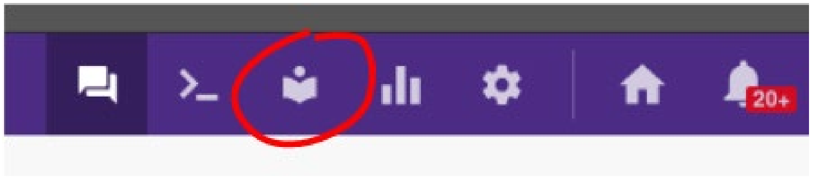
- You might miss Java, but keep an open mind!
  - Take note of anything that piques your interest. There's lots of classes you can take to explore this space further

Lecture Outline

- Course Introduction
- **Course Policies**
  - https://courses.cs.washington.edu/courses/cse351/24au/syllabus.html
- Binary and Numerical Representation

Important Tools

- Website: https://courses.cs.washington.edu/courses/cse351/24au/
- Ed course: https://edstem.org/us/courses/61460/
  - Discussion: announcements, ask and answer questions
  - Lessons: pre-lecture readings, lecture polls, homework

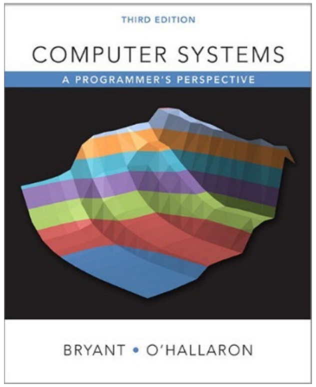


Find Ed Lessons in the top-right corner of Ed.

- Linked from website and Ed
  - Canvas: surveys, grade book, Zoom links
  - Gradescope: lab submissions, take-home exams
  - Panopto: lecture recordings

Reference Material

- The readings on Ed Lessons - constitute a "mini-textbook" for this course, but may not have enough detail for everyone
- *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

- **Pre-Lecture Readings:** ~5% ← Group work Allowed
  - Can reveal solution after one attempt (completion)
- **Homework:** ~20%
  - Unlimited submission attempts (autograded correctness)
- **Labs:** ~40% (optional partner) ← Partner work Allowed
  - Last submission graded (correctness)
- **Exams:** Midterm (~16%) and Final (~16%) ← Individual Work
  - Take-home; individual, but some discussion permitted. More info on these later.
- **Participation :** ~3%

To-Do List

- Admin
  - Explore/read website *thoroughly*, especially the syllabus
  - Check that you can access Ed Discussion & Lessons
  - **Get your machine set up to access the CSE Linux environment (`attu` or `calgary`) *as soon as possible***
  - Optionally, sign up for CSE 391: System and Software Tools
    - Tuesdays 1:30-2:20pm, in CSE2 G20 (1 CR)
- Assignments
  - Pre-Course Survey and HW0 due Friday (9/27) – 11:59pm
  - Lab 0 & HW1 due Monday (9/30) – 11:59pm
  - Pre-lecture Readings due before each lecture – 11am
  - Lecture activities are due before NEXT lecture – 11am

Lecture Outline

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

Decimal Numbering System

Base Ten

- 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)
  - $7061_{10} = (7 \times 10^3) + (0 \times 10^2) + (6 \times 10^1) + (1 \times 10^0)$

1000 100 10

10^3 10^2 10^1 10^0

Octal Numbering System

Base 8

- 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?
  - $7061_8 = (7 \times 8^3) + (0 \times 8^2) + (6 \times 8^1) + (1 \times 8^0) = 3633_{10}$

$8^3 \; 8^2 \; 8^1 \; 8^0$

Binary and Hexadecimal

- Binary is base 2
  - Symbols: 0, 1
  - Convention: $2_{10} = 10_2 = 0b10$
- Example: What is 0b110 in base 10?
  - $0b110 = 110_2 = (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 6_{10}$

- 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: $16_{10} = 10_{16} = 0x10$
- Example: What is 0xA5 in base 10?
  - $0xA5 = A5_{16} = (10 \times 16^1) + (5 \times 16^0) = 165_{10}$

Decimal to Binary

- Convert 13_{10} into binary

$2^3 + 2^2 + 2^1 + 2^0$

1

$8 + 4 + 0 + 1$

13 0b1 1 0 1

$13 - 8 = 5$

$5 - 4 = 1$

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

- Think!
  - No voting for this question

Converting from Decimal to Binary

- Given a decimal number N:
  1. List increasing powers of 2 from *right to left* until $\geq$ N
  2. 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
0	1	1	0	1

0b

13
− 8
5
−4
1

Converting Binary ↔ Hexadecimal

- Hex → 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 → 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

Polling Questions – Answer in Ed Lessons

1. What is the *decimal value* of the numeral 107_8?

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

   $64+0+7$

   71

   A. 71
   B. 87
   C. 107
   D. 568

2. What is the decimal number 108 in hex?

   16^2 16^1 16^0

   256 16 1

   $108=\underbrace{6\times16^1}_{96}+\underbrace{12\times16^0}_{12}=0x6C$

   $108-96=12$

   A. 0x6C = 0x6C
   B. 0xA8
   C. 0x108
   D. 0x612

3. Represent 0b1001101101011101 in hex.

   0x4DAD

4. Represent 0x3C9 in binary.

   0b0011 1100 1001

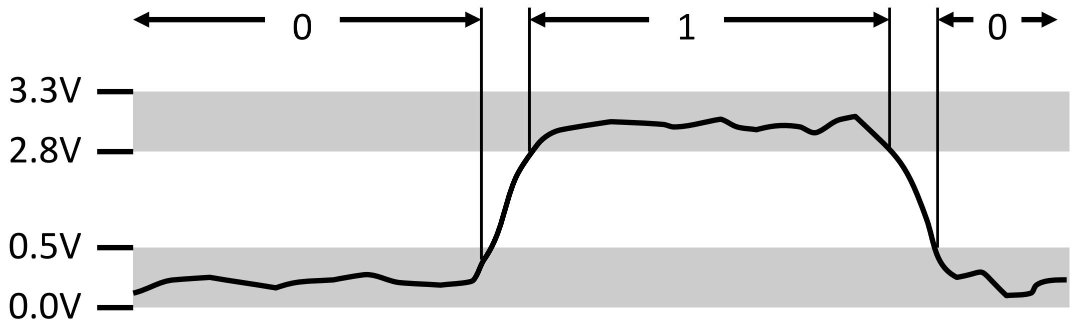
Base Comparison

- 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

Why Base 2?

- Electronic implementation
  - Easy to store with bi-stable elements
  - Reliably transmitted on noisy and inaccurate wires



- Other bases possible, but not yet viable:
  - DNA data storage (base 4: A, C, G, T) is hot @UW
  - Quantum computing

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 – 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

Source: www.LookupTables.com

So What's It Mean?

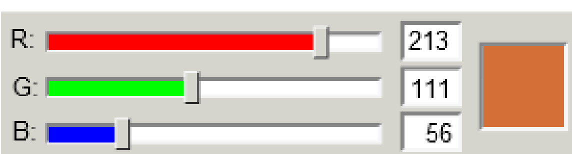
- *A sequence of bits can have many meanings!*

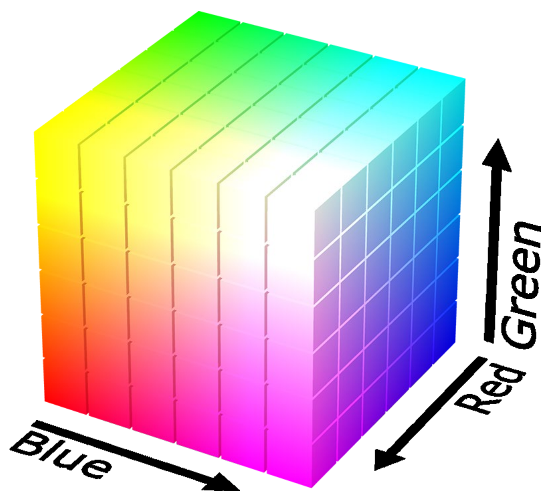
- Consider the hex sequence $\boxed{0x4E6F21}$
  - Common interpretations include:
    - The decimal number 5140257 10
    - The characters "No!"
    - The background color of this slide (sort of an olive green?)
    - The real number 7.203034 × 10^{-39}

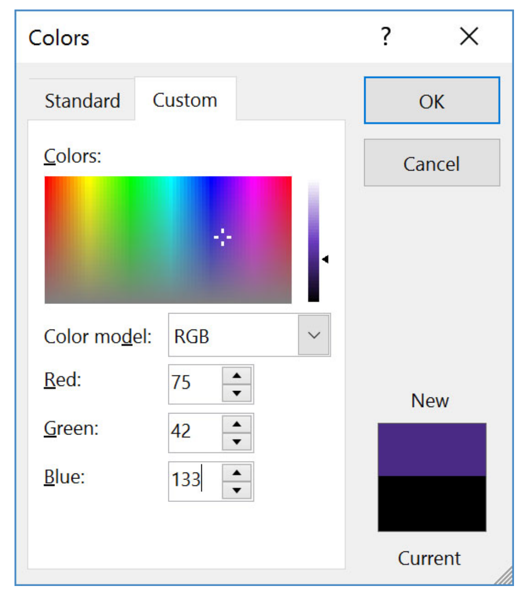
- It is up to the program/programmer to decide how to interpret the sequence of bits

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.)
  - <u>Examples</u>: **Blue**→0x0000FF, **Gold**→0xFFD700, **White**→0xFFFFFF, **Deep Pink**→0xFF1493







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 (try it!)
  - 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

Bonus Question: (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)