

# Lesson Outline

## ❖ 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

$107_8?$   $1 \times 8^2 + 0 \times 8^1 + 7 \times 8^0$   
 $2 \ 0 \ 0$   $64 + 0 + 7$   
 $= 71$

A. 71

B. 87

C. 107

D. 568

- ❖ Convert  $0b100110110101101$  in hex.

- ❖ What is the decimal number 108 in hex?

A. 0x6C

B. 0xA8

C. 0x108

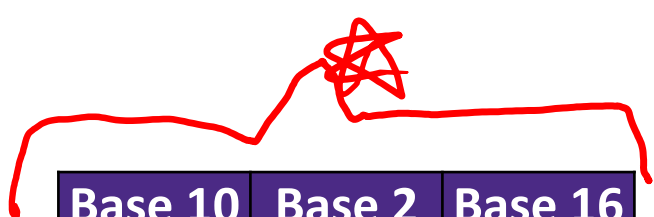
D. 0x612

- ❖ Convert 0x3C9 to binary.

$0b001111001001$   
 $\overline{3216}$   
 $1 \times 2^8 + 1 \times 2^2 = 8 + 4$  3

# 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

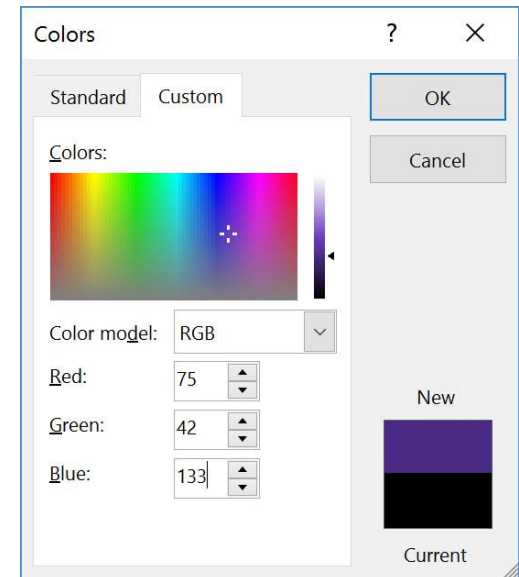
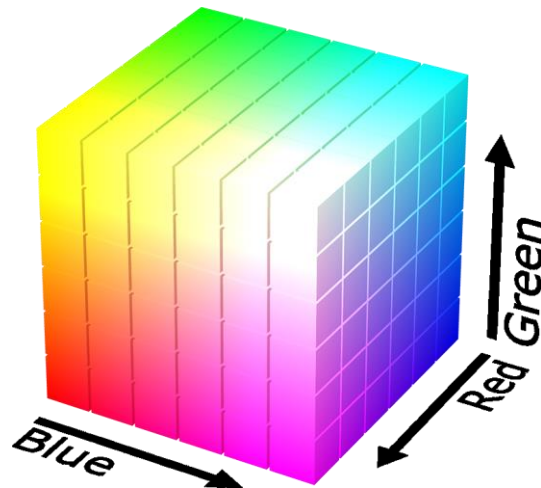
# So What's It Mean?

- ❖ *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  $7.203034 \times 10^{-39}$
- ❖ 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](http://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	<b>NUL</b> (null)	32	20	040	&#32;	<b>Space</b>	64	40	100	&#64;	<b>@</b>	96	60	140	&#96;	<b>`</b>
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	<b>!</b>	65	41	101	&#65;	<b>A</b>	97	61	141	&#97;	<b>a</b>
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	<b>"</b>	66	42	102	&#66;	<b>B</b>	98	62	142	&#98;	<b>b</b>
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	<b>#</b>	67	43	103	&#67;	<b>C</b>	99	63	143	&#99;	<b>c</b>
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	<b>\$</b>	68	44	104	&#68;	<b>D</b>	100	64	144	&#100;	<b>d</b>
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	<b>%</b>	69	45	105	&#69;	<b>E</b>	101	65	145	&#101;	<b>e</b>
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	<b>&amp;</b>	70	46	106	&#70;	<b>F</b>	102	66	146	&#102;	<b>f</b>
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	<b>'</b>	71	47	107	&#71;	<b>G</b>	103	67	147	&#103;	<b>g</b>
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	<b>(</b>	72	48	110	&#72;	<b>H</b>	104	68	150	&#104;	<b>h</b>
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	<b>)</b>	73	49	111	&#73;	<b>I</b>	105	69	151	&#105;	<b>i</b>
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	<b>*</b>	74	4A	112	&#74;	<b>J</b>	106	6A	152	&#106;	<b>j</b>
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	<b>+</b>	75	4B	113	&#75;	<b>K</b>	107	6B	153	&#107;	<b>k</b>
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	<b>,</b>	76	4C	114	&#76;	<b>L</b>	108	6C	154	&#108;	<b>l</b>
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	<b>-</b>	77	4D	115	&#77;	<b>M</b>	109	6D	155	&#109;	<b>m</b>
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	<b>.</b>	78	4E	116	&#78;	<b>N</b>	110	6E	156	&#110;	<b>n</b>
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	<b>/</b>	79	4F	117	&#79;	<b>O</b>	111	6F	157	&#111;	<b>o</b>
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	<b>0</b>	80	50	120	&#80;	<b>P</b>	112	70	160	&#112;	<b>p</b>
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	<b>1</b>	81	51	121	&#81;	<b>Q</b>	113	71	161	&#113;	<b>q</b>
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	<b>2</b>	82	52	122	&#82;	<b>R</b>	114	72	162	&#114;	<b>r</b>
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	<b>3</b>	83	53	123	&#83;	<b>S</b>	115	73	163	&#115;	<b>s</b>
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	<b>4</b>	84	54	124	&#84;	<b>T</b>	116	74	164	&#116;	<b>t</b>
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	<b>5</b>	85	55	125	&#85;	<b>U</b>	117	75	165	&#117;	<b>u</b>
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	<b>6</b>	86	56	126	&#86;	<b>V</b>	118	76	166	&#118;	<b>v</b>
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	<b>7</b>	87	57	127	&#87;	<b>W</b>	119	77	167	&#119;	<b>w</b>
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	<b>8</b>	88	58	130	&#88;	<b>X</b>	120	78	170	&#120;	<b>x</b>
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	<b>9</b>	89	59	131	&#89;	<b>Y</b>	121	79	171	&#121;	<b>y</b>
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	<b>:</b>	90	5A	132	&#90;	<b>Z</b>	122	7A	172	&#122;	<b>z</b>
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	<b>;</b>	91	5B	133	&#91;	<b>[</b>	123	7B	173	&#123;	<b>{</b>
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<b>&lt;</b>	92	5C	134	&#92;	<b>\</b>	124	7C	174	&#124;	<b> </b>
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	<b>=</b>	93	5D	135	&#93;	<b>]</b>	125	7D	175	&#125;	<b>}</b>
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	<b>&gt;</b>	94	5E	136	&#94;	<b>^</b>	126	7E	176	&#126;	<b>~</b>
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	<b>?</b>	95	5F	137	&#95;	<b>_</b>	127	7F	177	&#127;	<b>DEL</b>

# 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