

Announcements

- If you didn't (maybe still don't) have power yesterday, take another day to turn in the assignments 9/10.
- The midterm is 1 week from this Friday, 2/10

Adding some light to computing

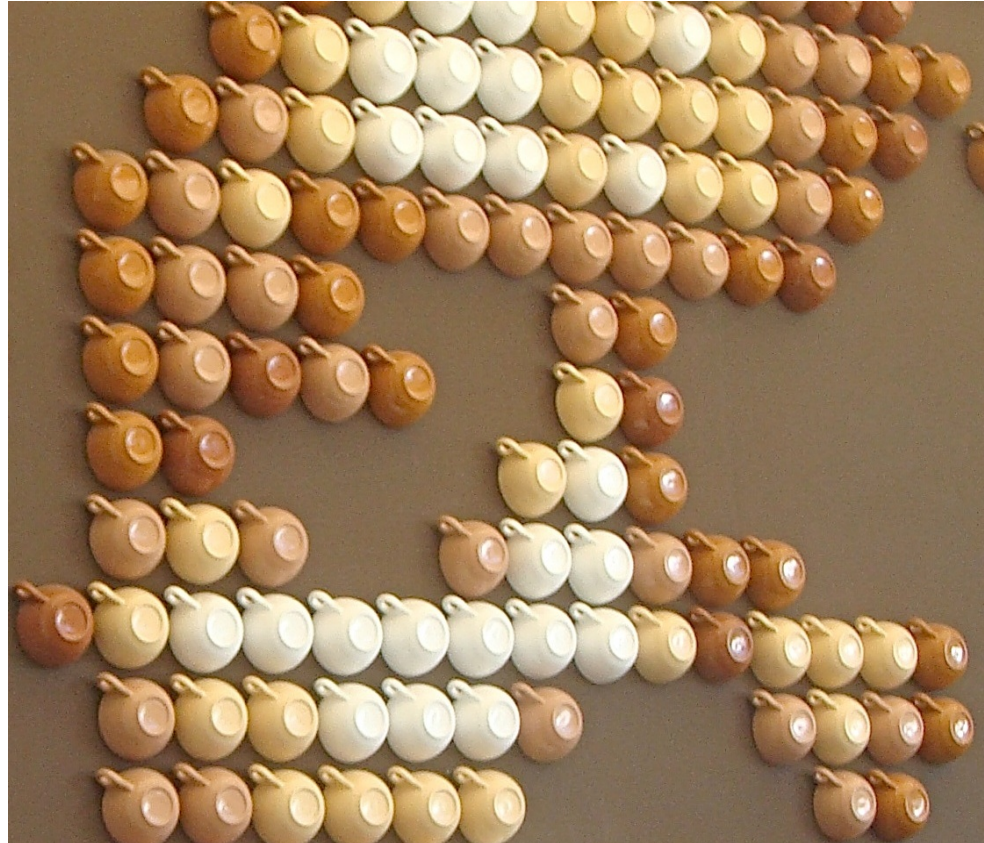
Bits of Color

Lawrence Snyder
University of Washington, Seattle

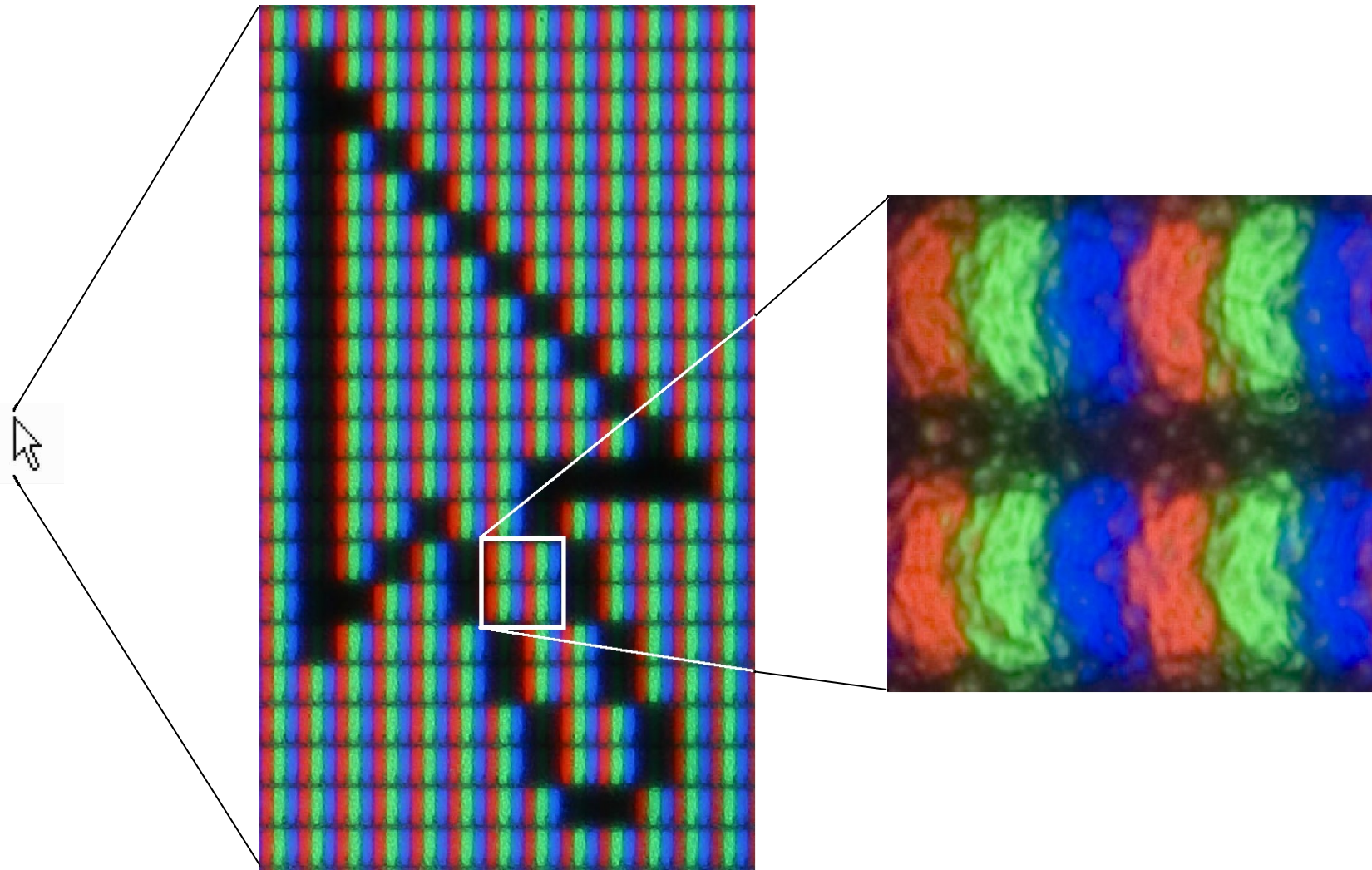
Return To RGB

- Recall that the screen (and other video displays) use red-green-blue lights, arranged in an array of picture elements, or *pixels*

Coffee Cup
Pixels

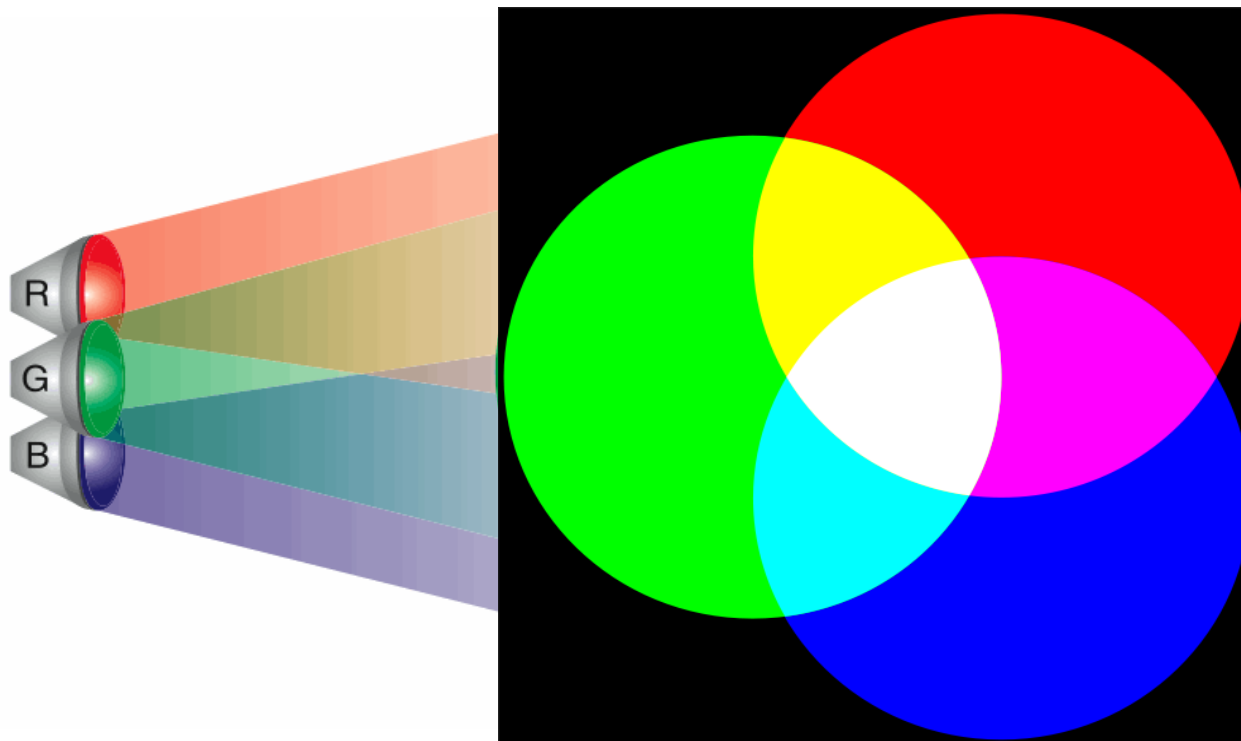


Actual Pixels From TFT LCD Display



Combining Colored Light

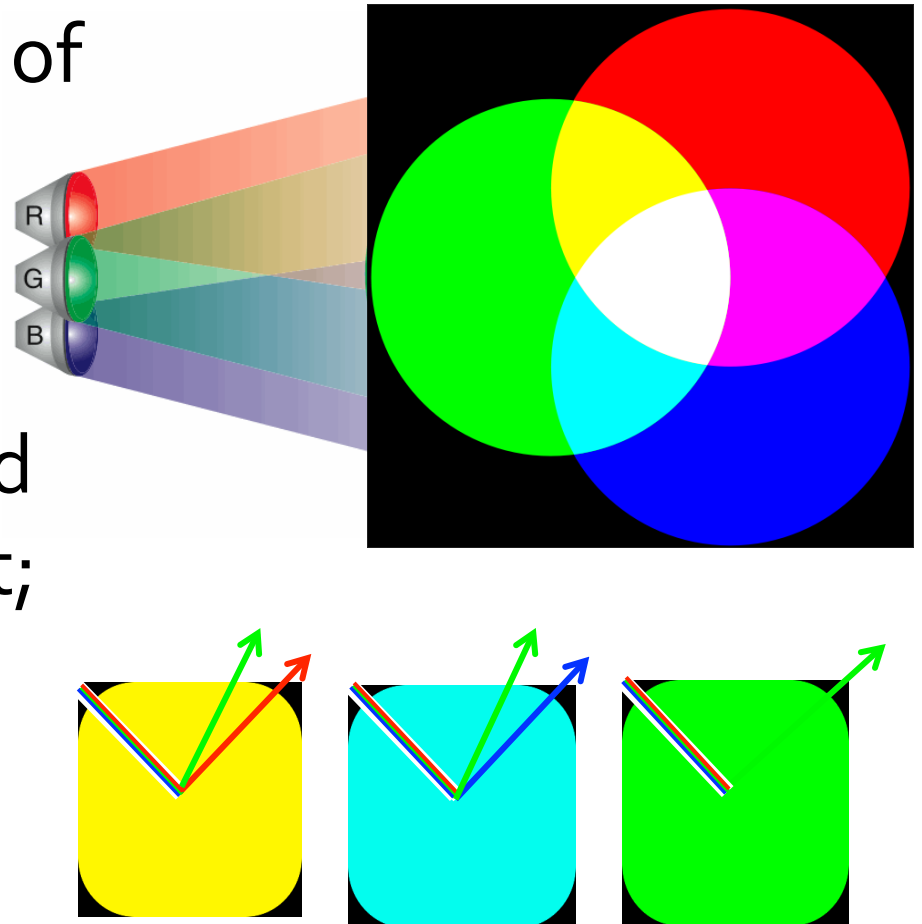
- The Amazing Properties of Colored Light!



- Caution: It doesn't work like pigment


Green + Red = Yellow?


- Colored light seems to violate our grade school rule of green = blue + yellow
What gives?
- In pigment, the color we see is the reflected color from white light; the other colors are absorbed




White, Gray, Black

- You know that gray is just different degrees of white as the “light is turned down” till we get to black

Black = [0, 0, 0] 0000 0000 0000 0000 0000 0000 

Gray = [128,128,128] 1000 0000 1000 0000 1000 0000 

White = [255,255,255] 1111 1111 1111 1111 1111 1111 

White-gray-black all have same values for RGB

Colors

Colors use different combinations of RGB

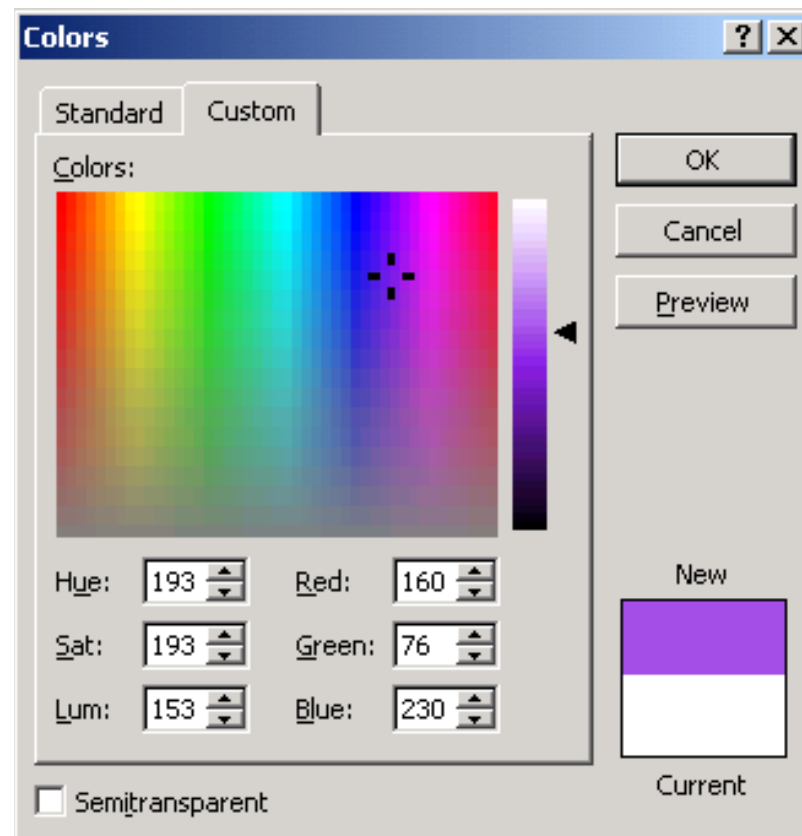
Husky Purple

Red=160

Green=76

Blue=230

The number gives
how intensely the
light is to shine



Positional Notation

- The RGB intensities are binary numbers
- Binary numbers, like decimal numbers, use *place notation*

$$\begin{aligned} 1101 &= 1 \times 1000 + 1 \times 100 + 0 \times 10 + 1 \times 1 \\ &= 1 \times 10^3 + 1 \times 10^2 + 0 \times 10^1 + 1 \times 10^0 \end{aligned}$$

except that the base is 2 not 10

$$\begin{aligned} 1101 &= 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \\ &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \end{aligned}$$

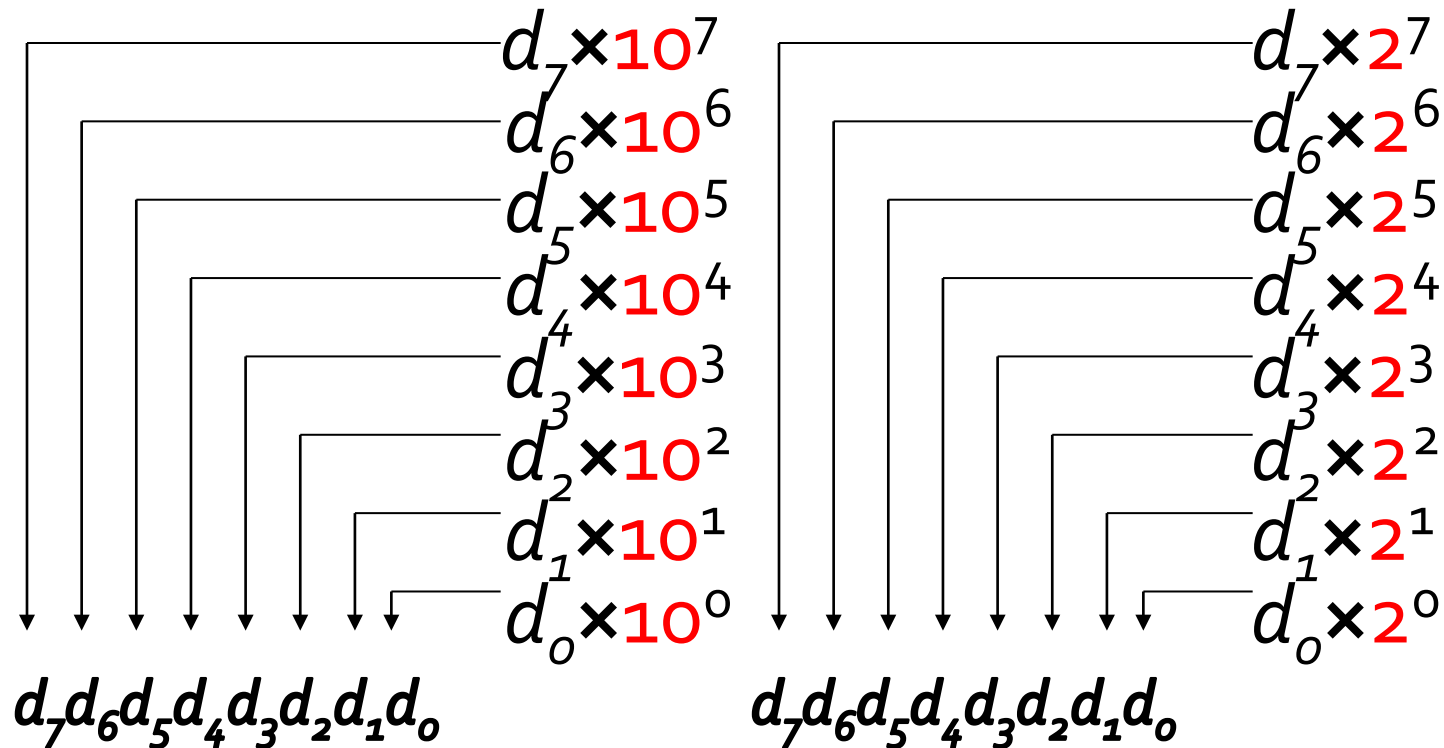
Base or
radix



1101 in binary is 13 in decimal

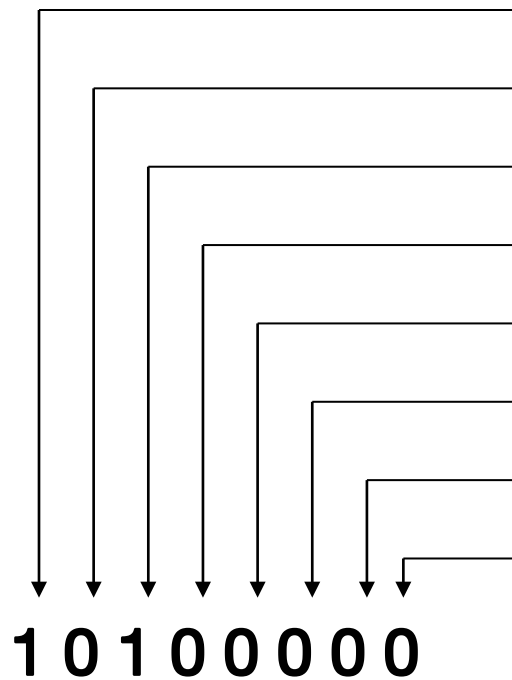
Positional Notation Logic

Recall that the place represents a power of the base value



The Red of HP As A Binary Number

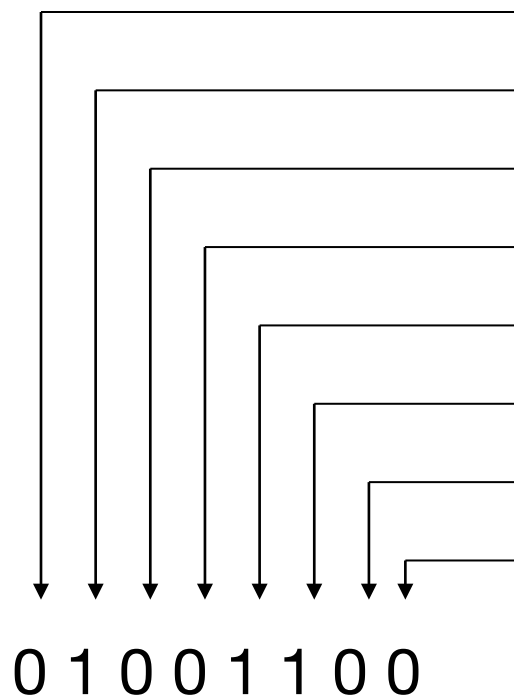
Given a binary number, add up the powers of 2 corresponding to 1s



$$\begin{aligned} 1 \times 2^7 &= 1 \times 128 &= 128 \\ 0 \times 2^6 &= 0 \times 64 &= 0 \\ 1 \times 2^5 &= 1 \times 32 &= 32 \\ 0 \times 2^4 &= 0 \times 16 &= 0 \\ 0 \times 2^3 &= 0 \times 8 &= 0 \\ 0 \times 2^2 &= 0 \times 4 &= 0 \\ 0 \times 2^1 &= 0 \times 2 &= 0 \\ 0 \times 2^0 &= 0 \times 1 = 0 & \\ &&= \underline{160} \end{aligned}$$

Green of HP As A Binary Number

Given a binary number, add up the powers of 2 corresponding to 1s



0×2^7	$= 1 \times 128$	$= 0$
1×2^6	$= 0 \times 64$	$= 64$
0×2^5	$= 1 \times 32$	$= 0$
0×2^4	$= 0 \times 16$	$= 0$
1×2^3	$= 0 \times 8$	$= 8$
1×2^2	$= 0 \times 4$	$= 4$
0×2^1	$= 0 \times 2$	$= 0$
0×2^0	$= 0 \times 1 = 0$	
		<u>$= 76$</u>

Is It Really Husky Purple?

- So Husky purple is (160,76,230) which is

1010 0000 0100 1100 1110 0110

160 76 230

Suppose you decide it's not "red" enough

- Increase the red by 16 = 1 0000

```
1010 0000
+ 1 0000
-----
1011 0000
```

Adding in binary is pretty much like adding in decimal except you have just 2 digits, 0 & 1

A Redder Purple

- Increase by 16 more

$$\begin{array}{r} 00110\ 000 \leftarrow \text{Carries} \\ 1011\ 0000 \\ + \underline{1\ 0000} \\ \hline 1100\ 0000 \\ \uparrow \uparrow \end{array}$$

Original
+16
+16

The rule: When the “place sum” equals the radix or more, subtract radix & carry

Check it out online: searching [binary addition](#) hits 19M times, and all of the p.1 hits are good explanations

Find Binary From Decimal

What is 230 (the Blue of HP)? Fill in the Table:

Num Being Converted	230	230	102	38	6	6	6	2	0
Place Value	256	128	64	32	16	8	4	2	1
Subtract		102	38	6			2	0	
Binary Num	0	1	1	1	0	0	1	1	0

Find Binary From Decimal

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

Num Being Converted	230								
Place Value	256	128	64	32	16	8	4	2	1
Subtract									
Binary Num									

Find Binary From Decimal

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

Num Being Converted	230 → 230								
Place Value	256	128	64	32	16	8	4	2	1
Subtract									
Binary Num	0								

Find Binary From Decimal

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

Num Being Converted	230	→ 230	102						
Place Value	256	128	64	32	16	8	4	2	1
Subtract		102							
Binary Num	0	1							

Find Binary From Decimal

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

Num Being Converted	230	→ 230	102	38					
Place Value	256	128	64	32	16	8	4	2	1
Subtract		102	38						
Binary Num	0	1	1						

Find Binary From Decimal

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

Num Being Converted	230	→ 230	102	38	6				
Place Value	256	128	64	32	16	8	4	2	1
Subtract		102	38	6					
Binary Num	0	1	1	1					

Find Binary From Decimal

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

Num Being Converted	230 → 230	102	38	6 → 6					
Place Value	256	128	64	32	16	8	4	2	1
Subtract		102	38	6					
Binary Num	0	1	1	1	0				

Find Binary From Decimal

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

Num Being Converted	230 → 230	102	38	6 → 6 → 6					
Place Value	256	128	64	32	16	8	4	2	1
Subtract		102	38	6					
Binary Num	0	1	1	1	0	0			

Find Binary From Decimal

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

Num Being Converted	230	→ 230	102	38	6	→ 6	→ 6	2	
Place Value	256	128	64	32	16	8	4	2	1
Subtract		102	38	6			2		
Binary Num	0	1	1	1	0	0	1		

Find Binary From Decimal

Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

Num Being Converted	230	→ 230	102	38	6	→ 6	→ 6	2	0
Place Value	256	128	64	32	16	8	4	2	1
Subtract		102	38	6			2	0	
Binary Num	0	1	1	1	0	0	1	1	

Find Binary From Decimal

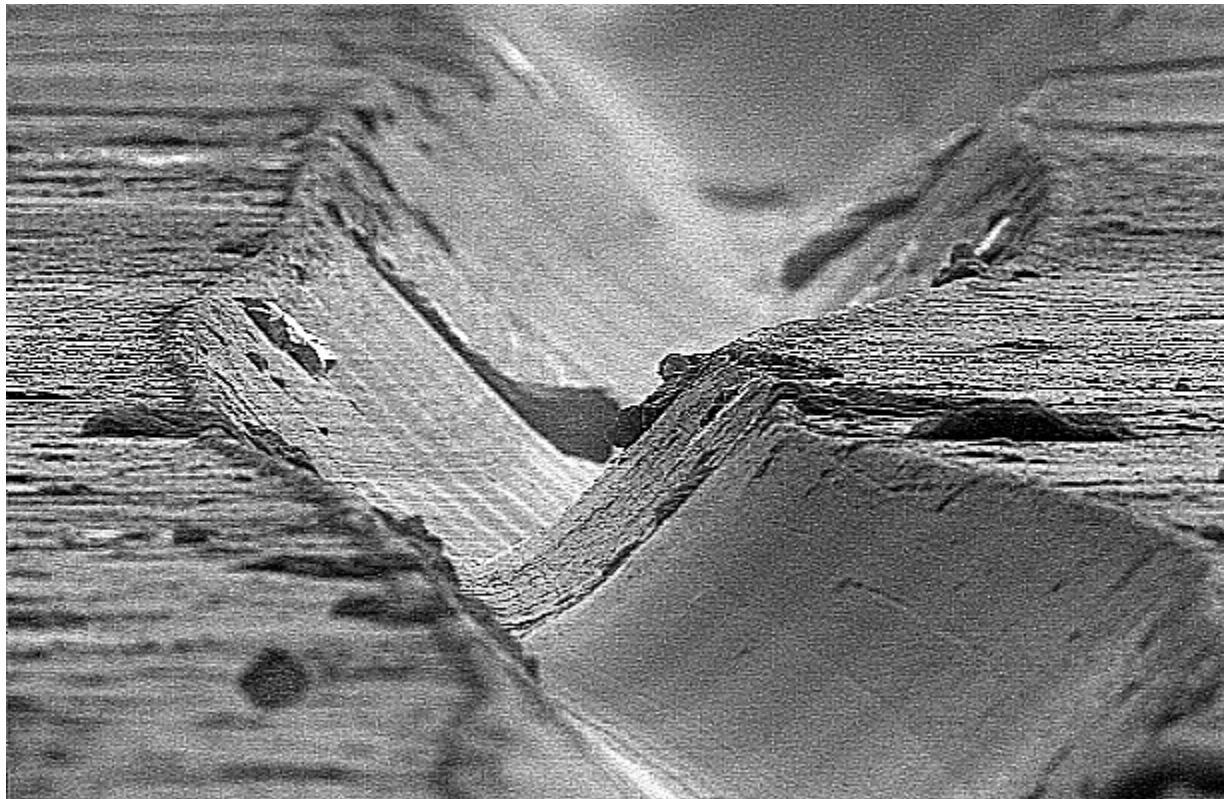
Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

Num Being Converted	230	→ 230	102	38	6	→ 6	→ 6	2	0
Place Value	256	128	64	32	16	8	4	2	1
Subtract		102	38	6			2	0	
Binary Num	0	1	1	1	0	0	1	1	0

Read off the result: 0 1110 0110

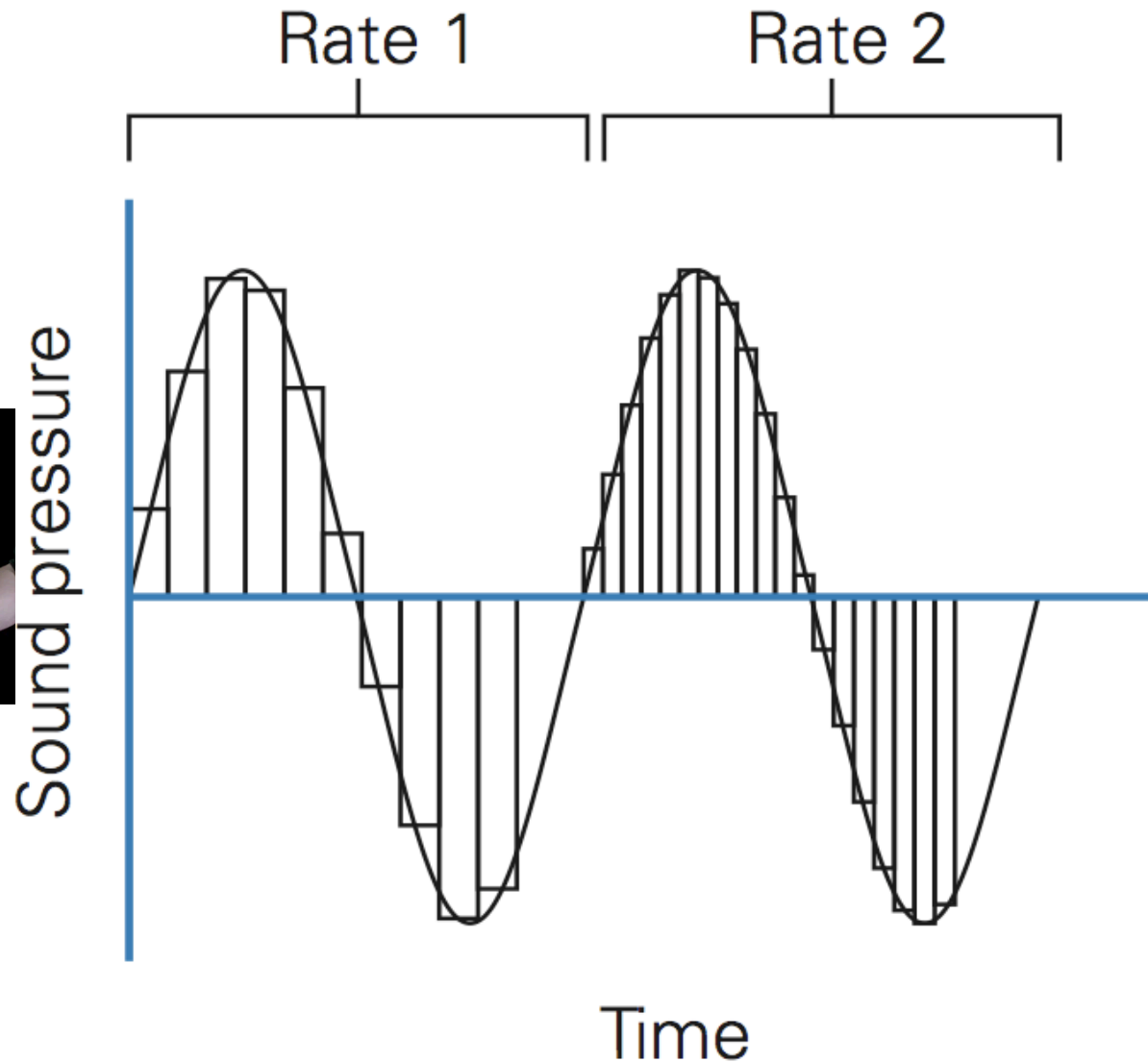
Not All Information Is Discrete

- Analogue information directly applies physical phenomena, e.g. vinyl records

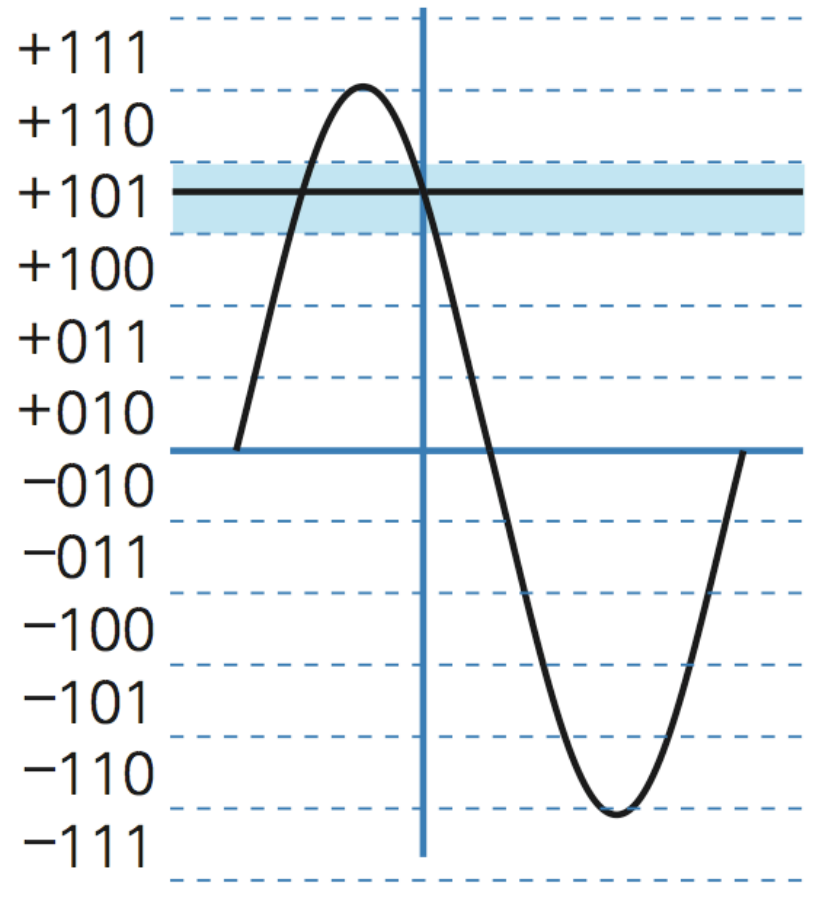
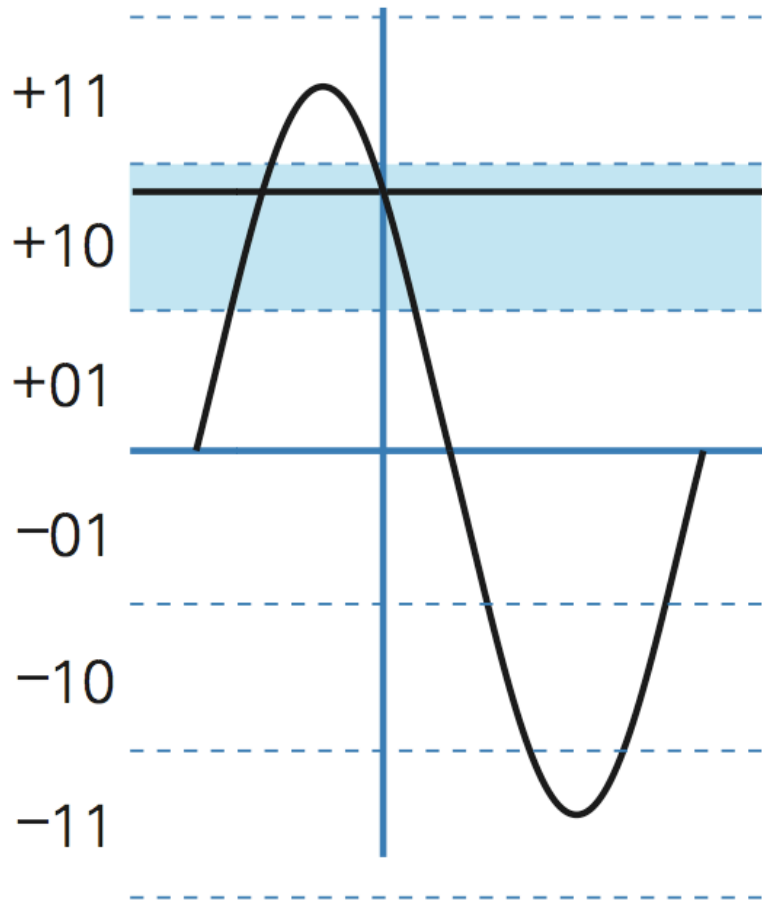


Analog Signals Become Discrete

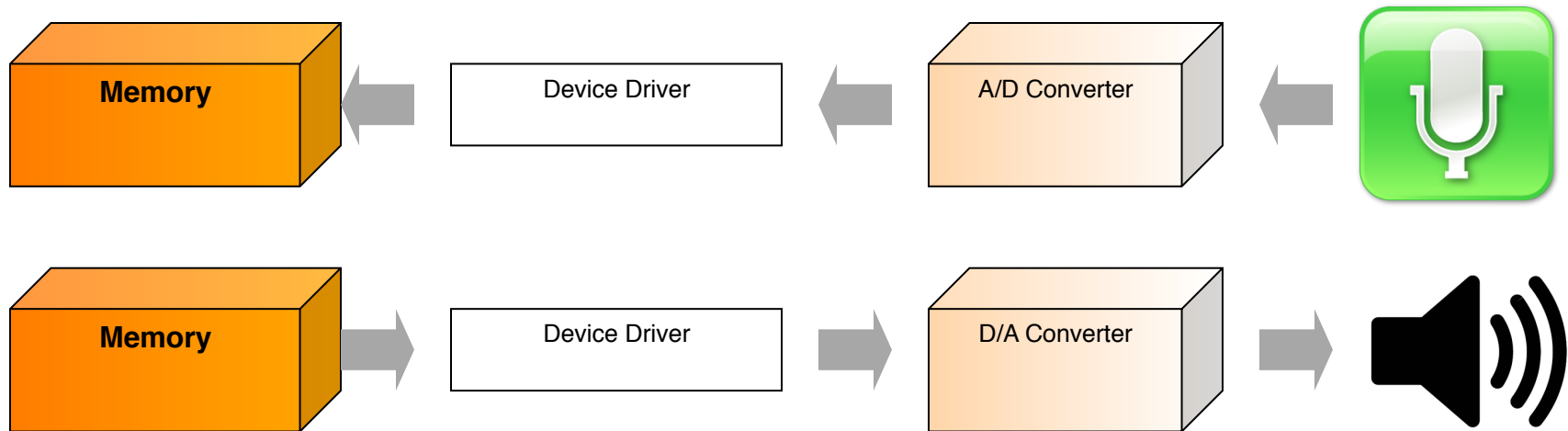
Sampling
the wave ...



Precision of the Sample



The World Is Analog – Go Between



Analog is needed for the “real world”
Digital is best for “information world”

- Can be modified, enhanced, remixed, etc
- Shared, stored permanently, reproduced, ...

Multimedia

- Many different forms of online information with special representations
 - JPG, MP3, MPEG, WAV ...
 - Most forms of multimedia require many, many bits
 - A minute of digital audio:
 - 60 seconds x
 - 44,100 samples per second x
 - 16 bits each
 - x 2 for stereo
 - Is 84,672,000 bits, or 10,584,000
 - 1 hour is 635 MB!

Compress: Change Representation

- Often, most of the bits are not needed – MP3 audio is less than 1MB/min because many sounds can be eliminated – we can't hear them
- Compression ... comes in two forms
 - Lossless – eliminated bits can be recovered
 - Lossy – eliminated bits are gone for good ... MP3

Susanne Vega sings *Tom's Diner*
www.youtube.com/watch?v=nLedFWpF9EA






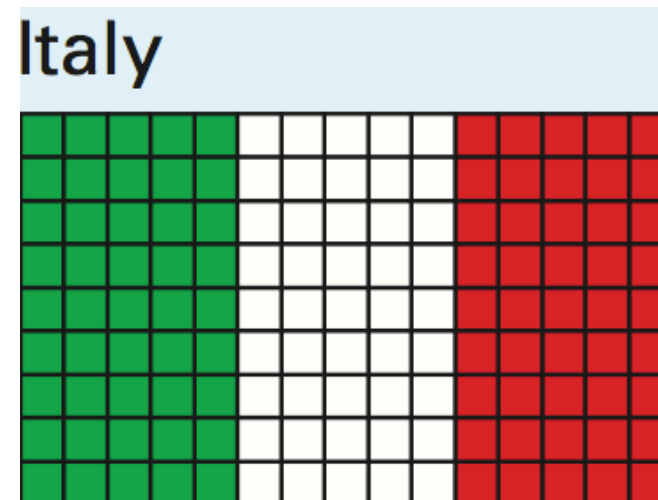
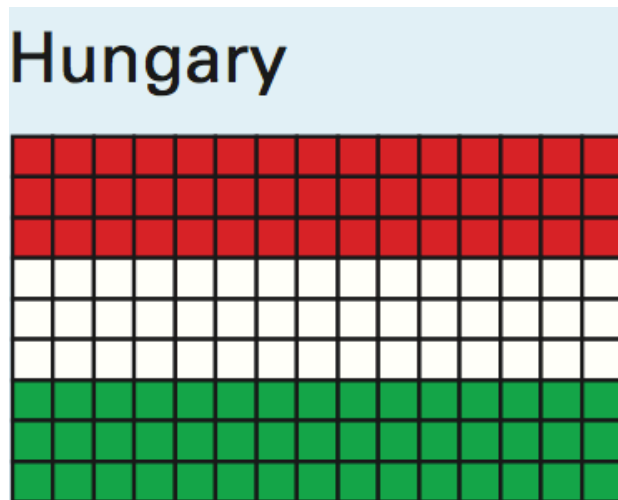
Lossless Compression

- Lossless compression seems strange – it eliminates bits that can be recovered again ... weren't they necessary in the first place???
- Consider a fax –
 - Usually faxes use a scanner that produces rows of 0s and 1s.
 - Compress by counting ... it's *run-length encoding*:
00000000000000000000000000000000111111100000000011
== 22:0,7:1,8:0,2:1

GIF Uses Same Idea

- Graphics Interchange Format (GIF) uses several kinds of compression
 - Color Table
 - Run Length Encoding
 - Lemple/Ziv/Welch Encoding

Color Table		
1	FF 00 00	
2	FF FF FF	
3	00 FF 00	



Compare Images Using Gif

- Compare Hungary Flag and Italian Flag




- huFlag: [15 × 9] 45:1, 45:2, 45:3

- itFlag: [15 × 9]

5:3,5:2,5:1,5:3,5:2,5:1,5:3,5:2,5:1,

5:3,5:2,5:1,5:3,5:2,5:1,5:3,5:2,5:1,

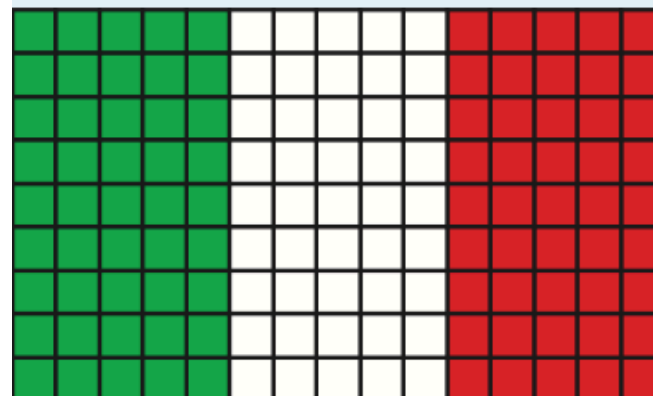
5:3,5:2,5:1,5:3,5:2,5:1,5:3,5:2,5:1

Color Table		
1	FF 00 00	
2	FF FF FF	
3	00 FF 00	

Hungary



Italy



JPG is Lossy

- Areas of similar color are represented by one shade ... it's OK for a while



Review What We Know About Bits

- Facts about physical representation:
 - Information is represented by the presence or absence of a physical phenomenon (Panda)
 - Hole punched in a card; no hole [Hollerith]
 - Dog barks in the night; no barking in the night [Holmes]
 - Wire is electrically charged; wire is neutral
 - ETC
- Abstract all these cases with 0 and 1; it unifies them so we don't have to consider the details

Bits Work For Arithmetic

- Binary is sufficient for number representation (place/value) and arithmetic
 - The number base is 2, instead of 10
 - Binary addition is just like addition in any other base except it has fewer cases ... better for circuits
 - All arithmetic and standard calculations have binary equivalents
- We conclude: bits “work” for quantities

Bytes – 8 bits in a row

- Bytes illustrate that bits can be grouped in sequence to generate unique patterns
 - 2 bits in sequence, $2^2 = 4$ patterns: 00, 01, 10, 11
 - 4 bits in sequence, $2^4 = 16$ patterns: 0000, 0001, ...
 - 8 bits in sequence, $2^8 = 256$ patterns: 0000 0000, ...


- ASCII groups 8 bits in sequence
 - They seem to be assigned intelligently, but they're just patterns

ASCII	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
0000	Nu	sh	sx	Ex	Er	Ed	Ak	Bl	Bs	Hr	Lf	Yr	Ff	Cr	So	Si	
0001	pl	p1	p2	p3	p4	Nk	Sy	Ez	CN	EM	Sb	Ec	Fs	Gs	Rs	Us	
0010	!	"	#	\$	%	&	'	()	*	+	,	-	.	/		
0011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
0100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
0101	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_	
0110	~	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
0111	p	q	r	s	t	u	v	w	x	y	z	{		}	~	pt	
1000	so	s1	s2	s3	ln	nl	ss	es	hs	hs	ys	ps	pv	ri	s2	s3	
1001	pc	p1	p2	se	cc	mm	sp	ep	os	oa	oa	cs	st	os	pm	ap	
1010	so	i	ç	£	♀	¥		\$..	©	σ	«	¬	-	®	™	
1011	o	±	²	³	´	µ	¶	·	,	ˆ	°	»	¼	½	¾	¿	
1100	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï	
1101	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	
1110	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	
1111	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	

Representing Anything

- Compare binary arithmetic to ASCII
 - Binary encodes the positions to make using the information (numbers) easy, like for addition
 - ASCII assigns some pattern to each letter
- Given any finite set of things – colors, computer addresses, English words, etc.
 - We might figure out a smart way to represent them as bits – colors can give light intensity of RGB
 - We can just assign patterns, and manipulate them by pattern matching – red can be 0000 0001, dark red 0000 0010, etc.

Bits Have No Inherent Meaning

- What does this represent:
0000 0000 1111 0001 0000 1000 0010 0000?
- You don't know until you know how it was encoded
 - As a binary number: 15,796,256
 - As a color, RGB(241,8,32) 
 - As a computer instruction: Add 1, 7, 17
 - As ASCII: $n_u^b_s \tilde{n}$ <space>
 - IP Address: 0.241.8.32
 - Hexadecimal number: 00 F1 08 20
 - ... → to infinity and beyond

A Bias-free Universal Medium

- This is the principle:

Bias-free Universal Medium Principle:

Bits can represent all discrete information;
bits have no inherent meaning

- Bits are it!!!
- “Computers encode information with bits, not numbers ... the bits might be numbers, but they might be a lot of other stuff instead”

Assignment 11

