Announcements

Project 1B due Today at 11:00 PM

Midterm Friday, in class

More Digital Representation

Discrete information is represented in binary (PandA), and "continuous" information is made discrete

Return To RGB

Images are constructed from picture elements (pixels); color uses RGB light

The RGB color intensities are specified by 3 numbers in the range [0, 255], ie 1 byte each

■	**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

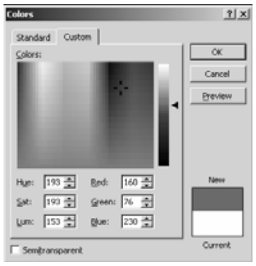
Colors use different combinations of RGB

- **Husky Purple**

Red=160

Green=76

Blue=230



Hue: 193 Red: 160

Sat: 193 Green: 76

Lum: 153 Blue: 230

Positional Notation

The RGB intensities are binary numbers

Binary numbers, like decimal numbers, use *place notation*

1101 = 1x1000 + 1x100 + 0x10 + 1x1

= $1x10^3$ + $1x10^2$ + $0x10^1$ + $1x10^0$

except that the base is 2 not 10

Base or radix

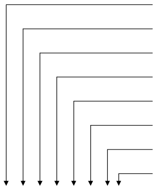
1101 = 1x8 + 1x4 + 0x2 + 1x1

= $1x2^3$ + $1x2^2$ + $0x2^1$ + $1x2^0$

1101 in binary is 13 in decimal

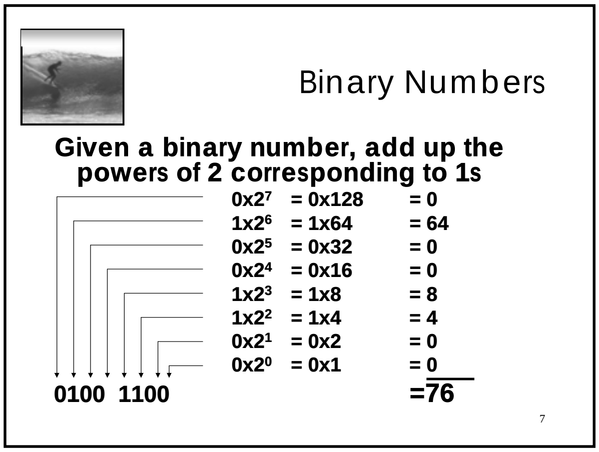
Binary Numbers

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



$1x2^7$	**= 1x128**	**= 128**
$0x2^6$	**= 0x64**	**= 0**
$1x2^5$	**= 1x32**	**= 32**
$0x2^4$	**= 0x16**	**= 0**
$0x2^3$	**= 0x8**	**= 0**
$0x2^2$	**= 0x4**	**= 0**
$0x2^1$	**= 0x2**	**= 0**
$0x2^0$	**= 0x1**	**= 0**

1010 0000 **=160**

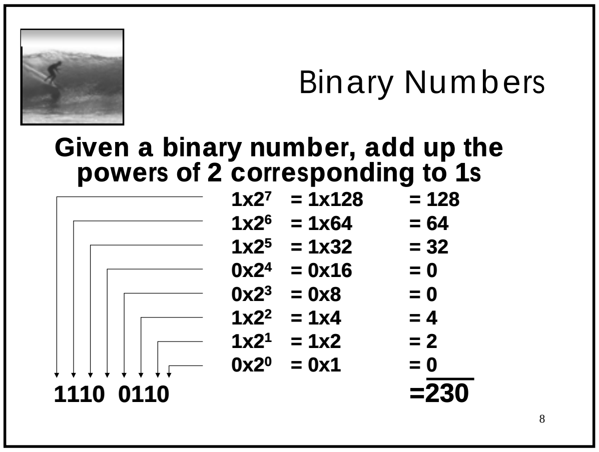


Binary Numbers

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

$0x2^7$	**= 0x128**	**= 0**
$1x2^6$	**= 1x64**	**= 64**
$0x2^5$	**= 0x32**	**= 0**
$0x2^4$	**= 0x16**	**= 0**
$1x2^3$	**= 1x8**	**= 8**
$1x2^2$	**= 1x4**	**= 4**
$0x2^1$	**= 0x2**	**= 0**
$0x2^0$	**= 0x1**	**= 0**

0100 1100 **=76**

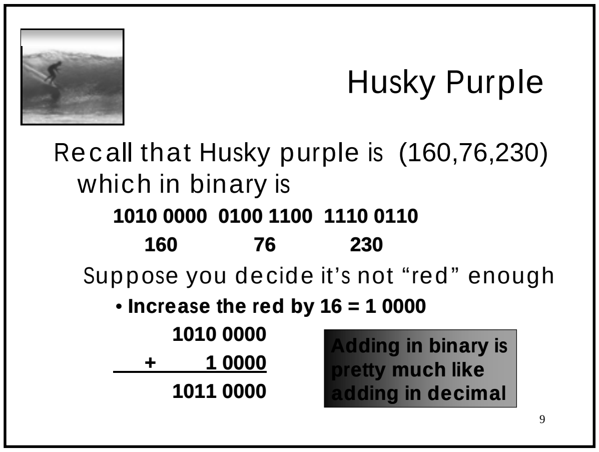


Binary Numbers

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

$1x2^7$	**= 1x128**	**= 128**
$1x2^6$	**= 1x64**	**= 64**
$1x2^5$	**= 1x32**	**= 32**
$0x2^4$	**= 0x16**	**= 0**
$0x2^3$	**= 0x8**	**= 0**
$1x2^2$	**= 1x4**	**= 4**
$1x2^1$	**= 1x2**	**= 2**
$0x2^0$	**= 0x1**	**= 0**

1110 0110 **=230**



Husky Purple

Recall that Husky purple is (160,76,230) which in binary is

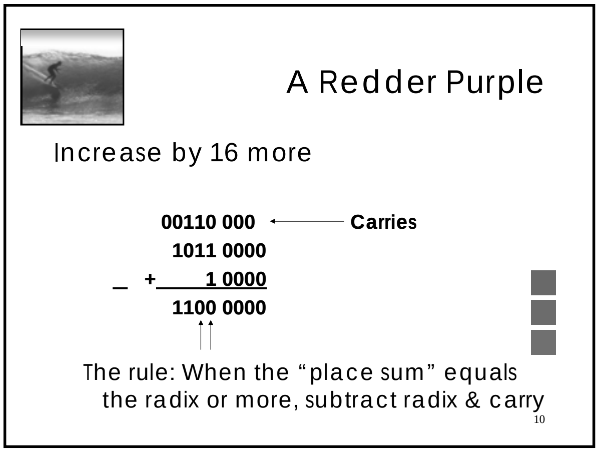
1010 0000 0100 1100 1110 0110
160 76 230

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

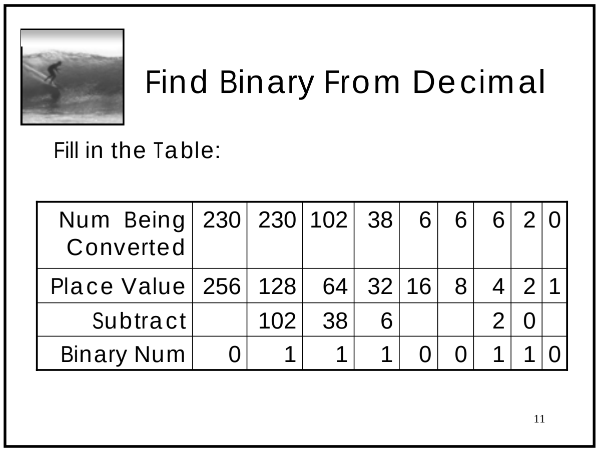


A Redder Purple

Increase by 16 more

```
     00110 000  ←—— Carries
      1011 0000
_  +     1 0000
      1100 0000
```

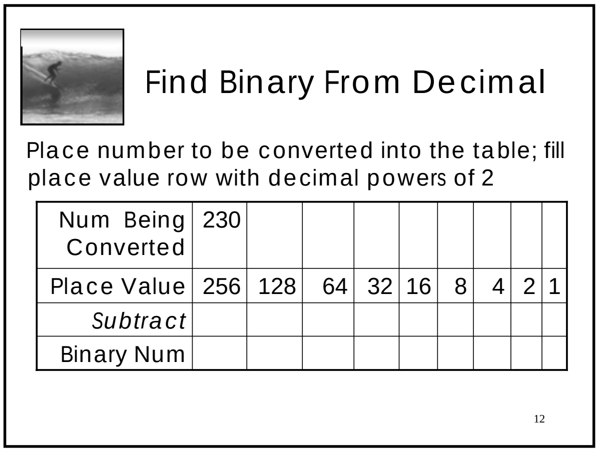
The rule: When the "place sum" equals the radix or more, subtract radix & carry



Find Binary From Decimal

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

Place number to be converted into the table; fill place value row with decimal powers of 2

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

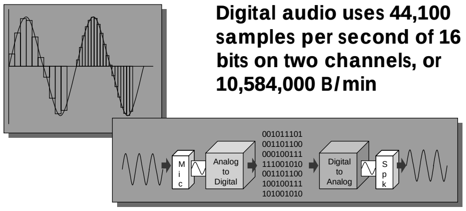
Rule: Subtract PV from the number; a positive result gives new number and “1”; otherwise, “0”

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

Digitizing

“Continuous” information like light and sound must be made “discrete”

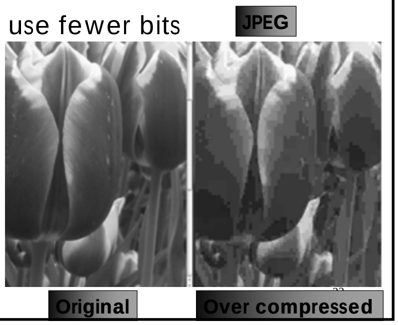


Digital audio uses 44,100 samples per second of 16 bits on two channels, or 10,584,000 B/min

Compression

Compression: use fewer bits

- Lossless – Recover the data
- Lossy– Lose the original data



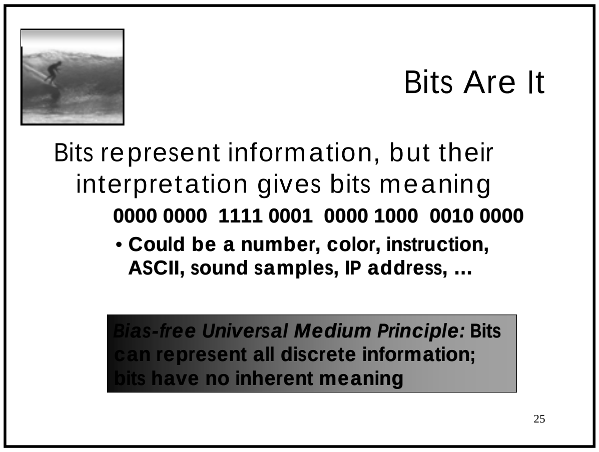
Run-Length Compression

Give number of 1s, number of 0s, etc.

```
→ 1111111111…  → (270 1s)
  1111111111…    (270 1s)
  1100000000…    (2 1s)(266 0s)(2 1s)
  1100000000…    (2 1s)(266 0s)(2 1s)
     …                …
```

Forget row encoding … alternate

[Size: 270x200](542)(266)(4)(266)(4)(266)(4)(266) …



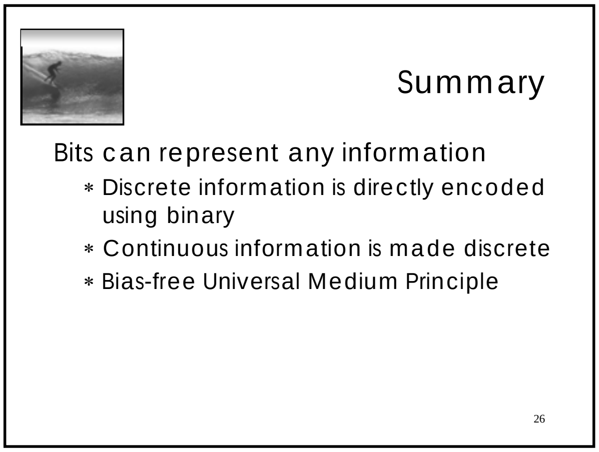
Bits Are It

Bits represent information, but their interpretation gives bits meaning

0000 0000 1111 0001 0000 1000 0010 0000

- **Could be a number, color, instruction, ASCII, sound samples, IP address, …**

Bias-free Universal Medium Principle: **Bits can represent all discrete information; bits have no inherent meaning**



Summary

Bits can represent any information

- Discrete information is directly encoded using binary
- Continuous information is made discrete
- Bias-free Universal Medium Principle