Number Systems

Mirza Mohammad Lutfe Elahi

Common Number Systems

System	Base	Symbols	Used by humans?	Used in computers?
Decimal	10	0, 1, … 9	Yes	No
Binary	2	0, 1	No	Yes
Octal	8	0, 1, … 7	No	No
Hexa-decimal	16	0, 1, … 9, A, B, … F	No	No

Quantities/Counting (1 of 3)

Decimal	Binary	Octal	Hexa-decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

Quantities/Counting (2 of 3)

Decimal	Binary	Octal	Hexa-decimal
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

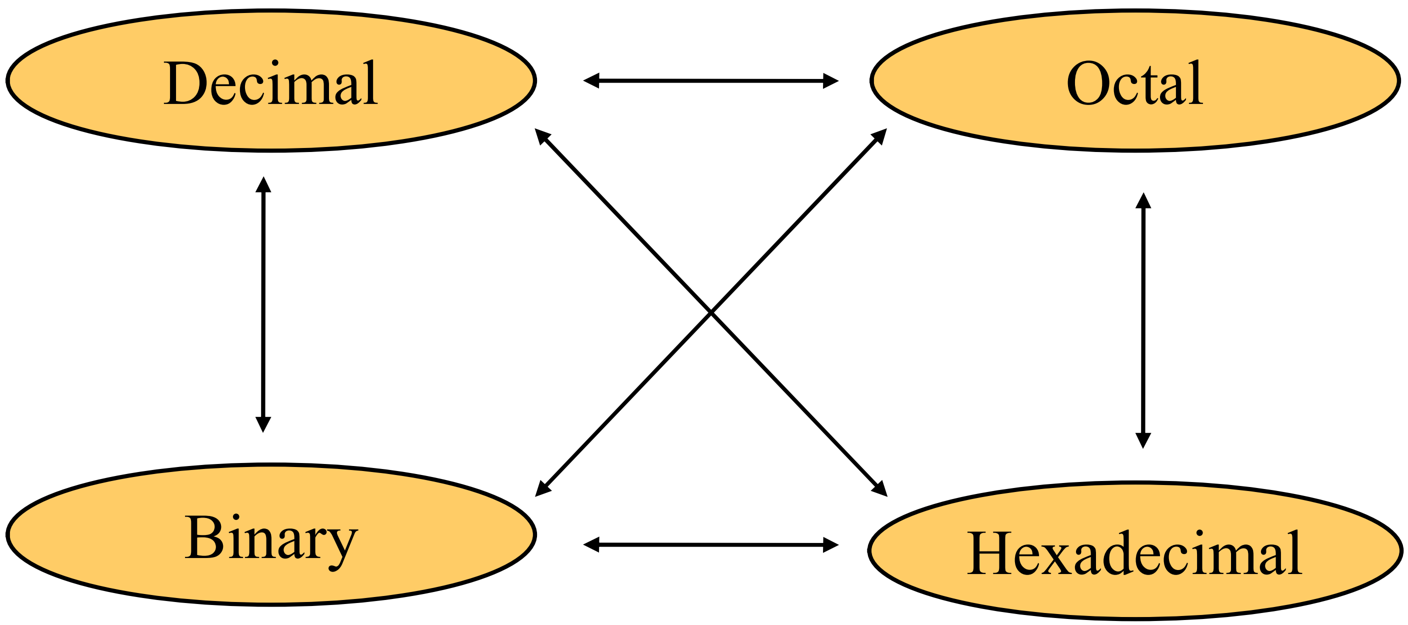
Quantities/Counting (3 of 3)

Decimal	Binary	Octal	Hexa-decimal
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17

Etc.

Conversion Among Bases

- The possibilities:

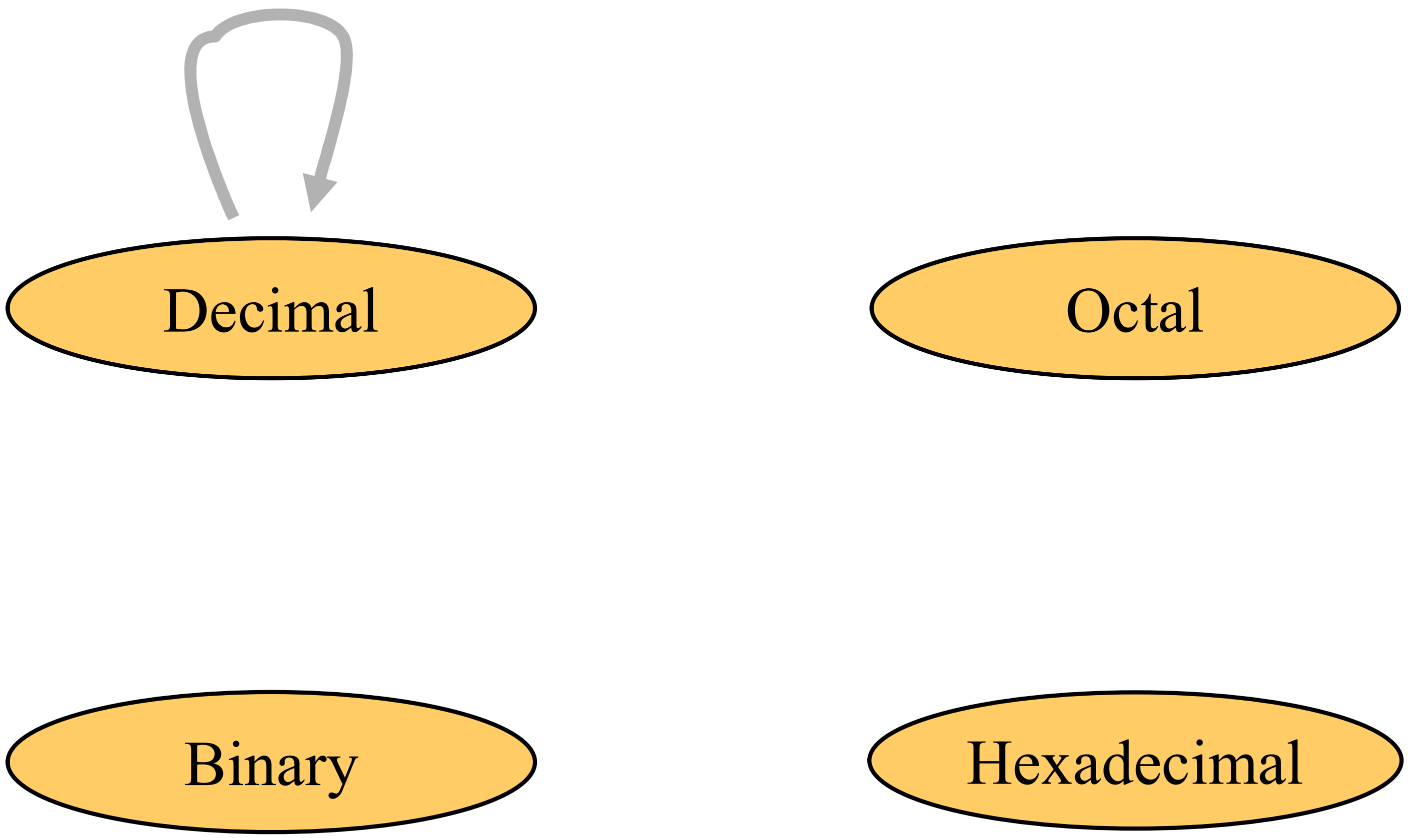


Quick Example

$$25_{10} = 11001_2 = 31_8 = 19_{16}$$

Base

Decimal to Decimal (just for fun)



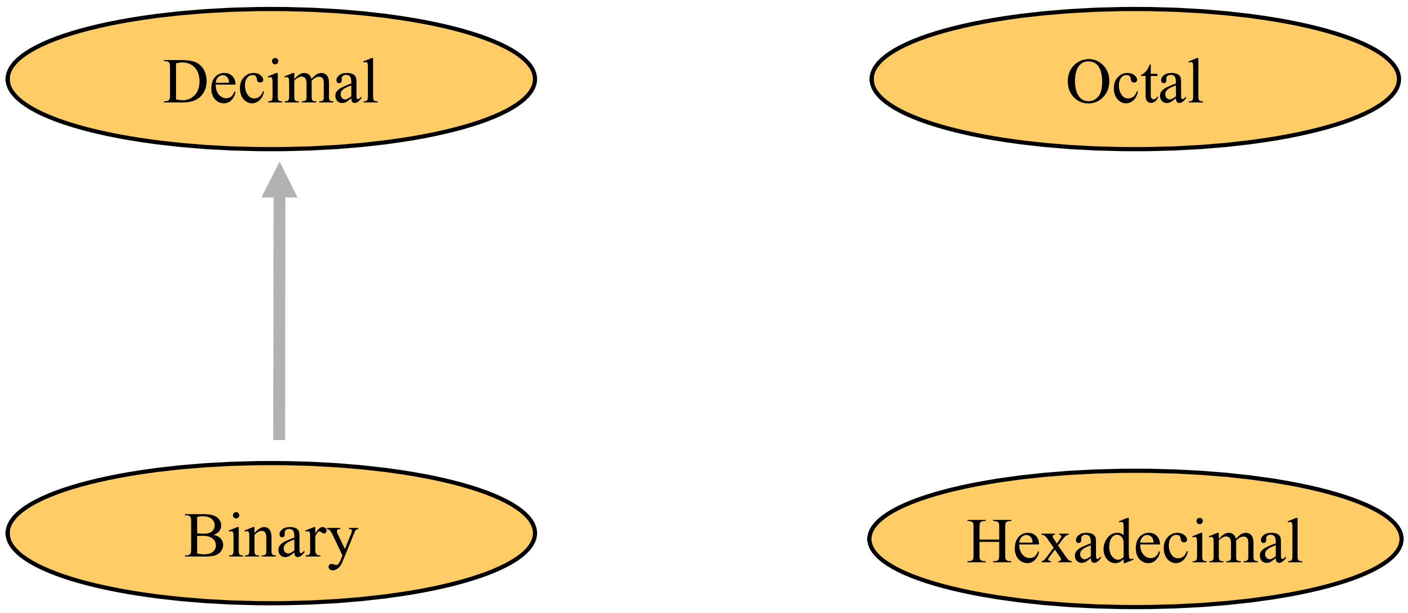
Next slide…

Weight

$$
\begin{array}{llcr}
125_{10} \Rightarrow & 5 \times 10^{0} & = & 5 \\
 & 2 \times 10^{1} & = & 20 \\
 & 1 \times 10^{2} & = & \underline{100} \\
 & & & 125
\end{array}
$$

Base

Binary to Decimal



Binary to Decimal

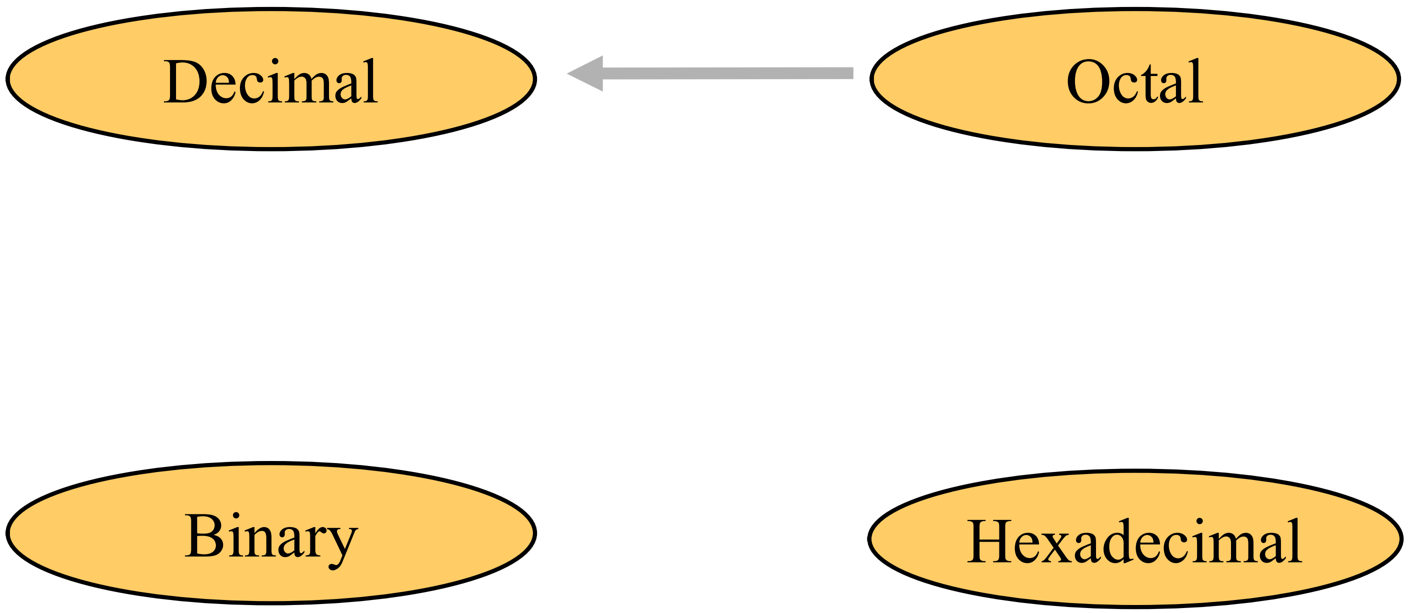
- Technique
  - Multiply each bit by 2^n, where *n* is the "weight" of the bit
  - The weight is the position of the bit, starting from 0 on the right
  - Add the results

Example

Bit “0”

$$
\begin{array}{llcl}
101011_2 \Rightarrow & 1 \times 2^0 & = & 1 \\
 & 1 \times 2^1 & = & 2 \\
 & 0 \times 2^2 & = & 0 \\
 & 1 \times 2^3 & = & 8 \\
 & 0 \times 2^4 & = & 0 \\
 & 1 \times 2^5 & = & \underline{32} \\
 & & & 43_{10}
\end{array}
$$

Octal to Decimal



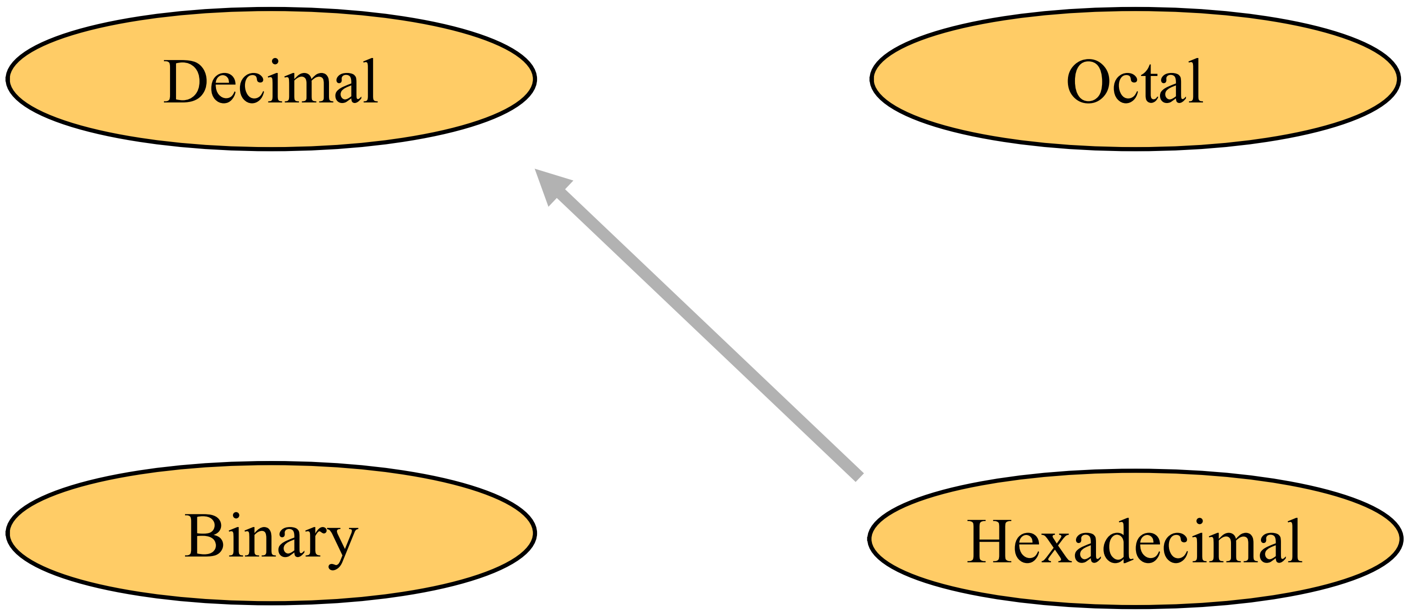
Octal to Decimal

- Technique
  - Multiply each bit by 8^n, where n is the “weight” of the bit
  - The weight is the position of the bit, starting from 0 on the right
  - Add the results

Example

$$724_8 \Rightarrow \begin{array}{lr} 4 \times 8^0 = & 4 \\ 2 \times 8^1 = & 16 \\ 7 \times 8^2 = & 448 \\ \hline & 468_{10} \end{array}$$

Hexadecimal to Decimal



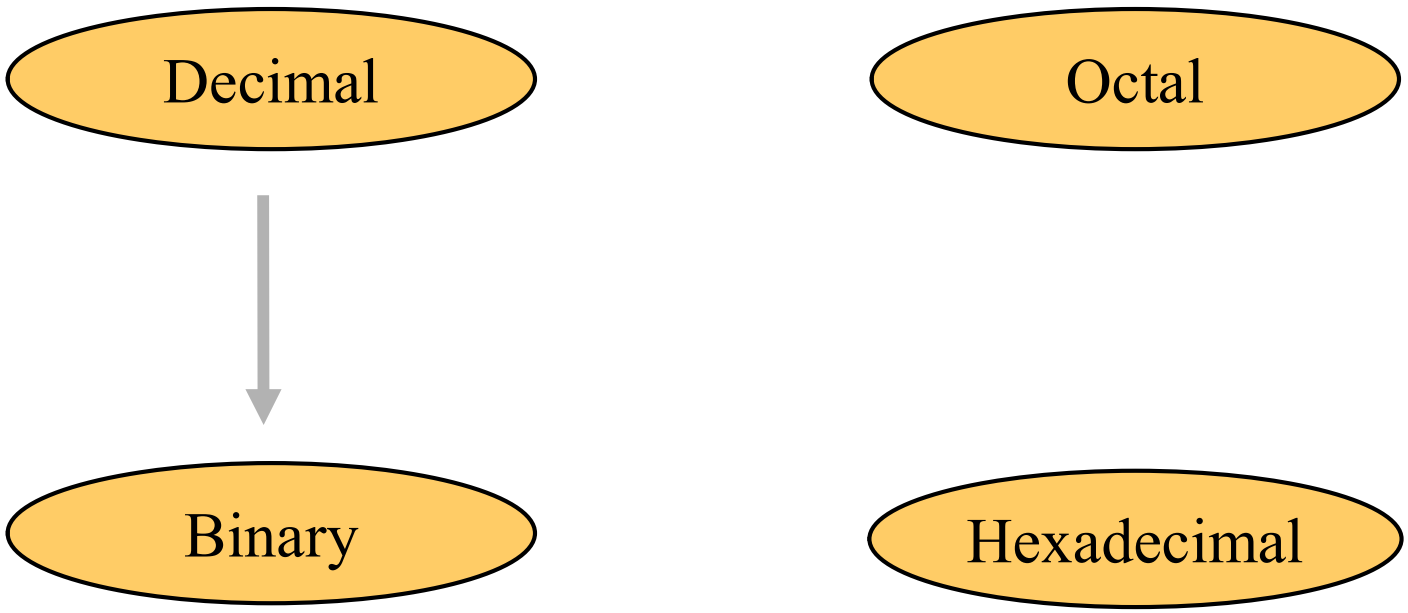
Hexadecimal to Decimal

- Technique
  - Multiply each bit by 16^n, where n is the "weight" of the bit
  - The weight is the position of the bit, starting from 0 on the right
  - Add the results

Example

$$
\begin{array}{llcrcr}
ABC_{16} \Rightarrow & C \times 16^0 & = & 12 \times 1 & = & 12 \\
 & B \times 16^1 & = & 11 \times 16 & = & 176 \\
 & A \times 16^2 & = & 10 \times 256 & = & \underline{2560} \\
 & & & & & 2748_{10}
\end{array}
$$

Decimal to Binary



Decimal to Binary

- Technique
  - Divide by two
  - Keep track of the remainder
  - Possible remainders 0 or 1

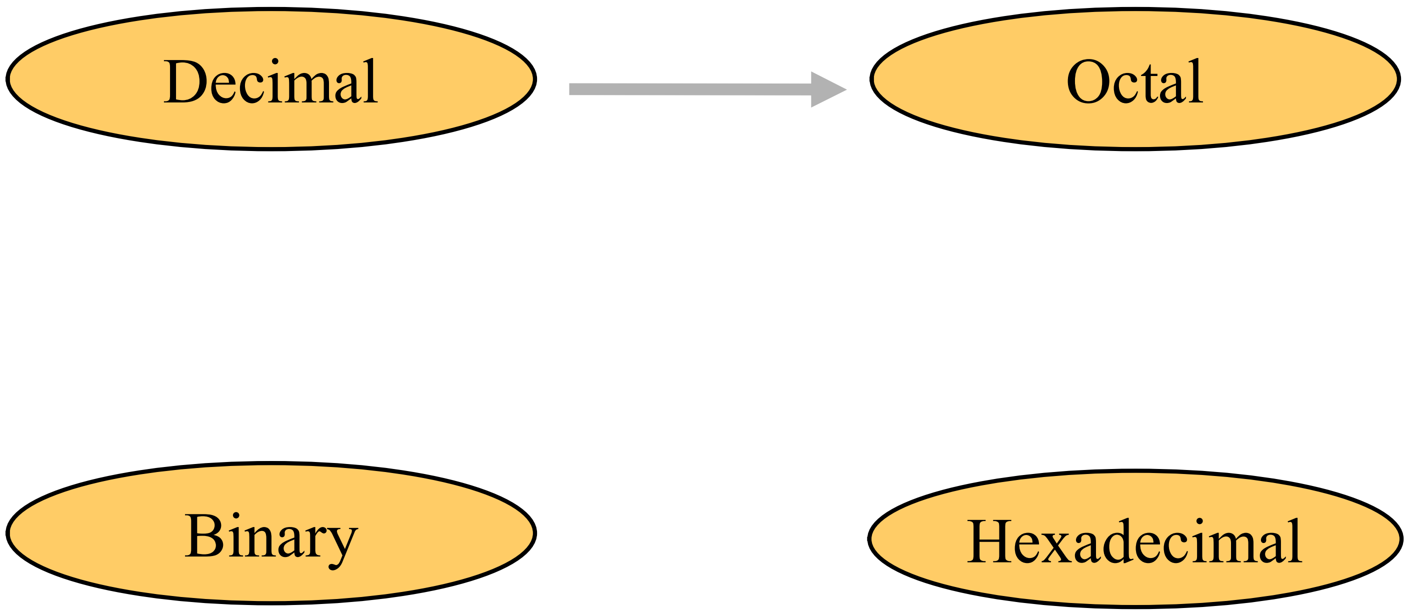
Example

$125_{10} = ?_2$

2	125	
2	62	1
2	31	0
2	15	1
2	7	1
2	3	1
2	1	1
	0	1

$125_{10} = 1111101_2$

Decimal to Octal



Decimal to Octal

- Technique
  - Divide by 8
  - Keep track of the remainder
  - Possible remainders 0 to 7

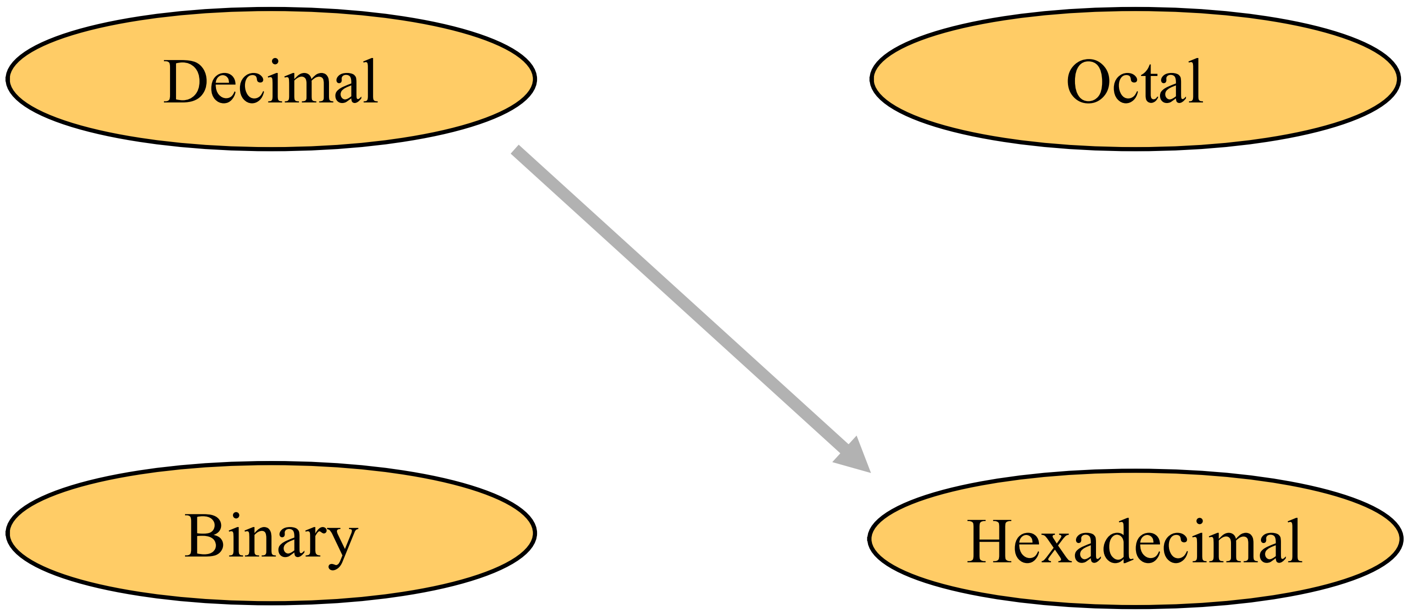
Example

$1234_{10} = ?_8$

8	1234	
8	154	2
8	19	2
8	2	3
	0	2

$1234_{10} = 2322_8$

Decimal to Hexadecimal



Decimal to Hexadecimal

- Technique
  - Divide by 16
  - Keep track of the remainder
  - Possible remainder 0 to 15

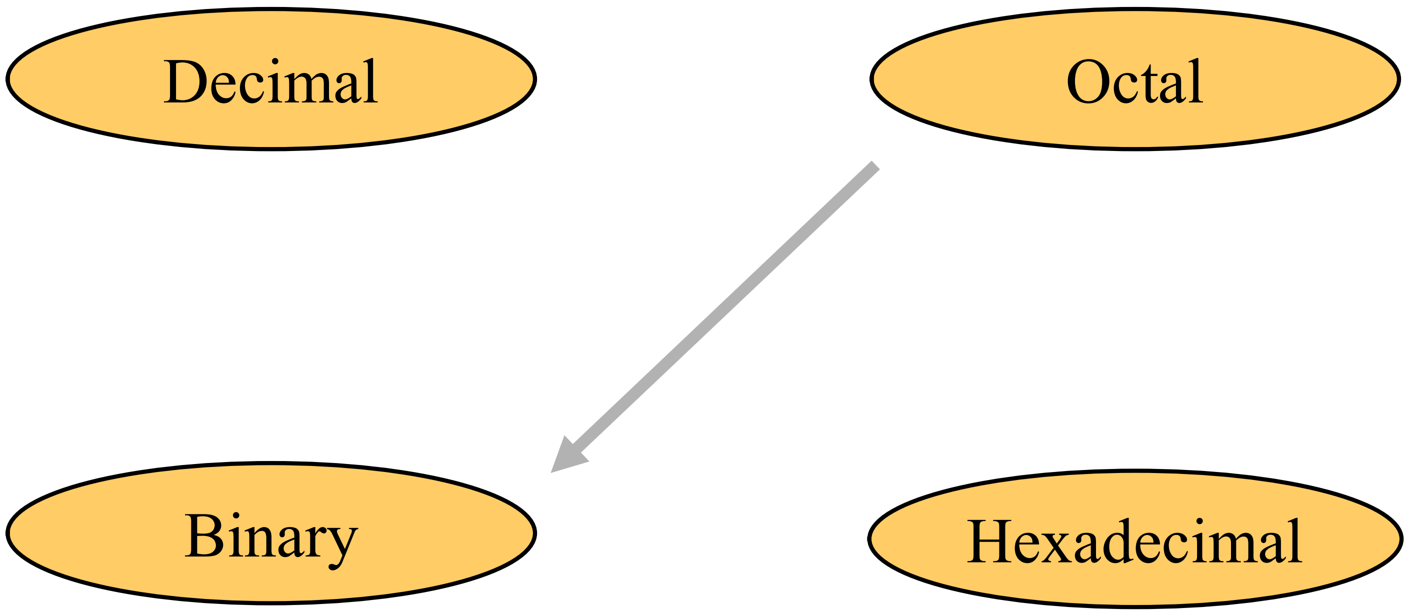
Example

$1234_{10} = ?_{16}$

```
16 | 1234
16 |   77     2
16 |    4     13 = D
        0     4
```

$1234_{10} = 4D2_{16}$

Octal to Binary



Octal to Binary

- Technique
  - Convert each octal digit to a 3-bit equivalent binary representation

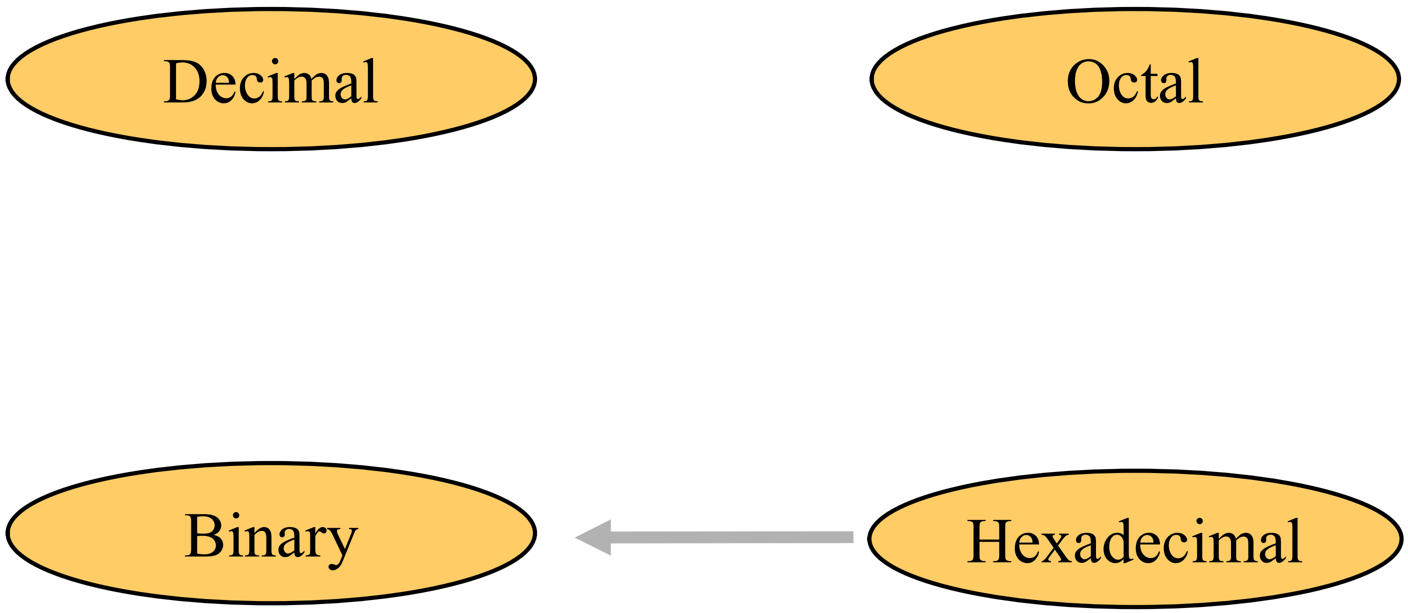
Example

$705_8 = ?_2$

$$\begin{array}{ccc} 7 & 0 & 5 \\ \downarrow & \downarrow & \downarrow \\ 111 & 000 & 101 \end{array}$$

$705_8 = 111000101_2$

Hexadecimal to Binary



Hexadecimal to Binary

- Technique
  - Convert each hexadecimal digit to a 4-bit equivalent binary representation

Example

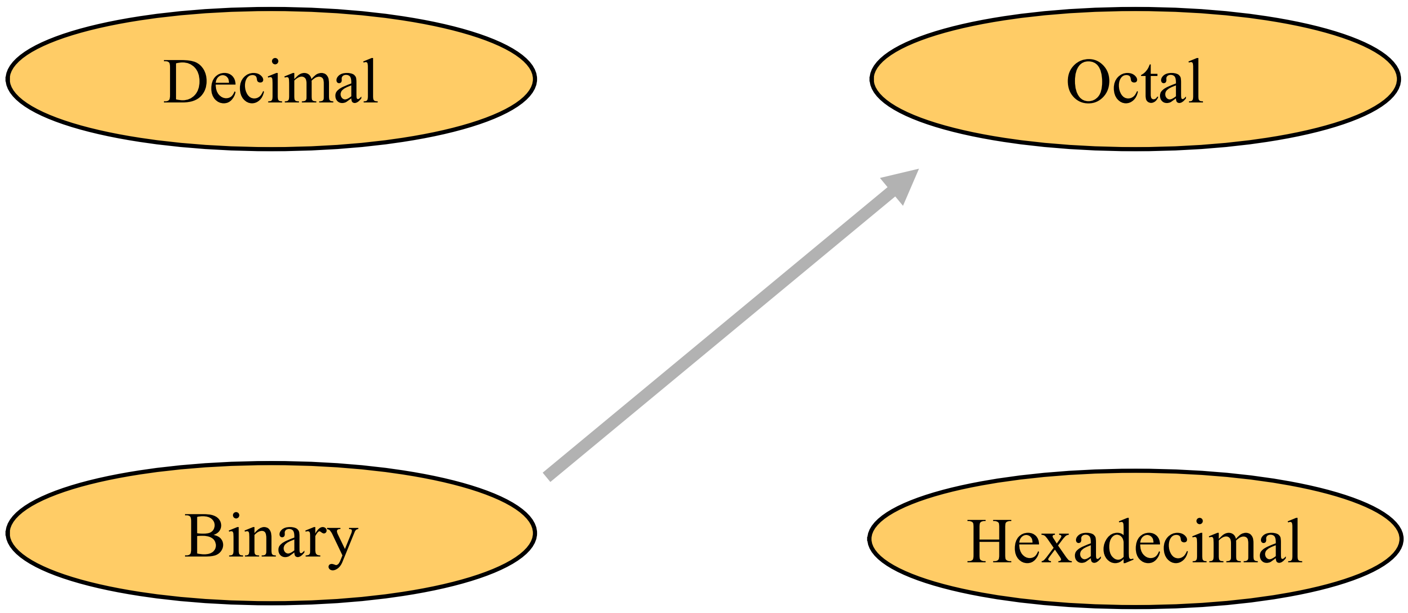
$10AF_{16} = ?_2$

1 0 A F

0001 0000 1010 1111

$10AF_{16} = 0001000010101111_2$

Binary to Octal



Binary to Octal

- Technique
  - Group bits in threes, starting on right
  - Convert to octal digits

Example

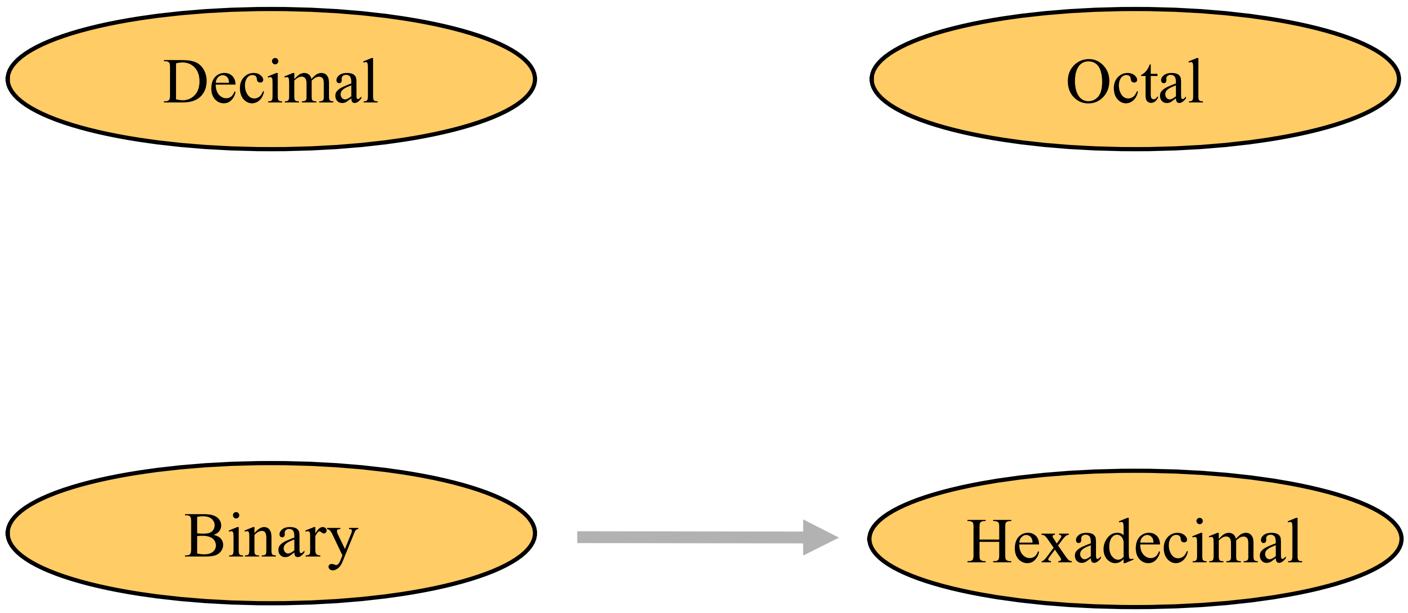
$1011010111_2 = ?_8$

1 011 010 111

1 3 2 7

$1011010111_2 = 1327_8$

Binary to Hexadecimal



Binary to Hexadecimal

- Technique
  - Group bits in fours, starting on right
  - Convert to hexadecimal digits

Example

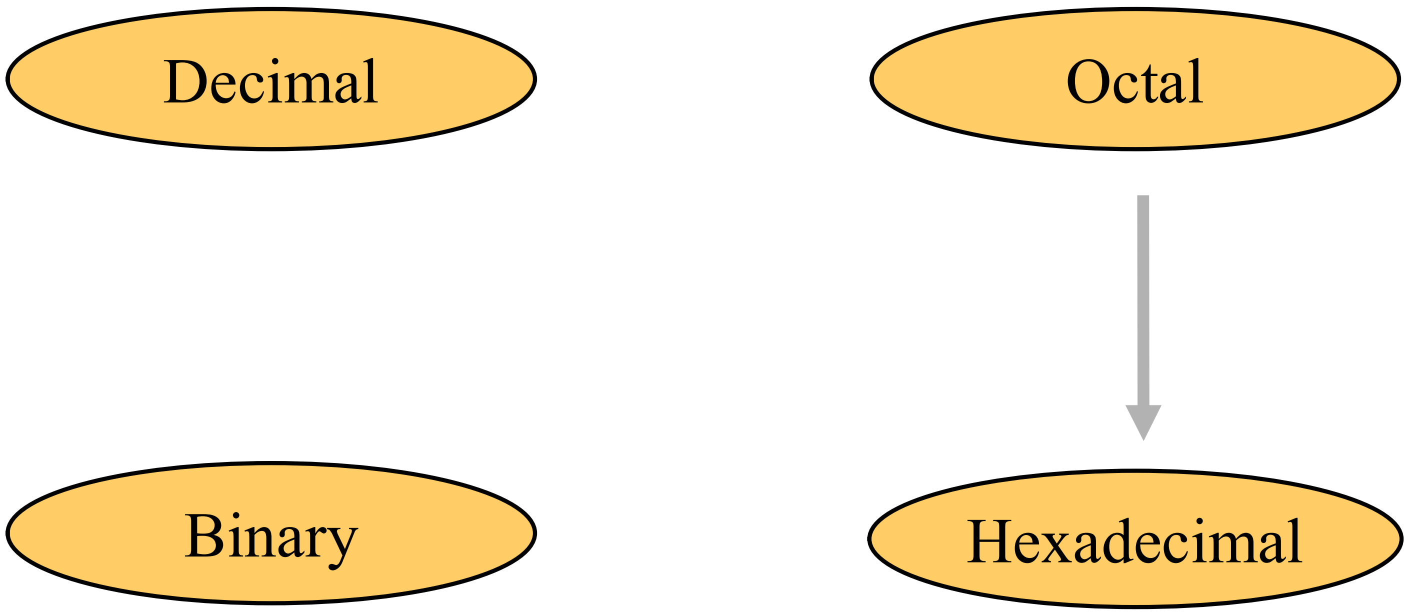
$1010111011_2 = ?_{16}$

0010 1011 1011

2 B B

$1010111011_2 = 2BB_{16}$

Octal to Hexadecimal



Octal to Hexadecimal

- Technique
  - Use binary as an intermediary

Example

$1076_8 = ?_{16}$

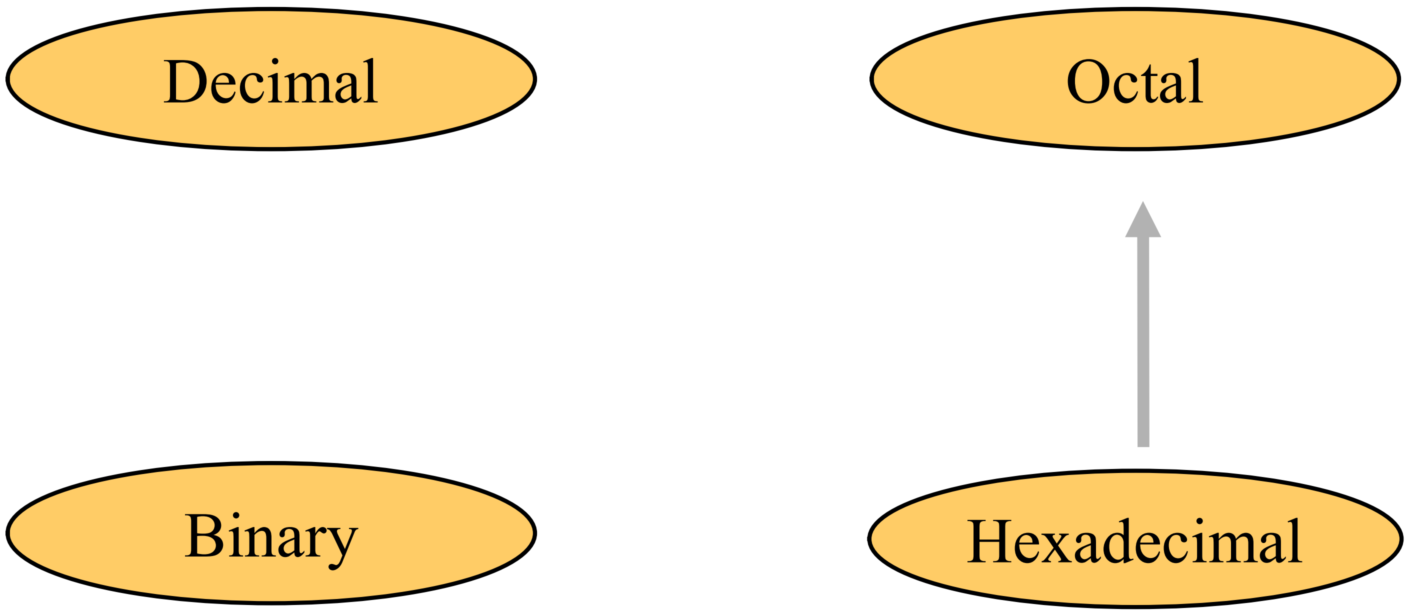
1 0 7 6

001 0|00 11|1 110

2 | 3 | E

$1076_8 = 23E_{16}$

Hexadecimal to Octal



Hexadecimal to Octal

- Technique
  - Use binary as an intermediary

Example

$1F0C_{16} = ?_8$

1 F 0 C

0|001 | 111|1 00|00 1|100

| 1 | 7 | 4 | 1 | 4

$1F0C_{16} = 17414_8$

Exercise – Convert ...

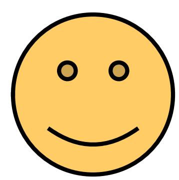
Decimal	Binary	Octal	Hexa-decimal
33			
	1110101		
		703	
			1AF

Don't use a calculator!

Exercise – Convert …

Answer

Decimal	Binary	Octal	Hexa-decimal
33	100001	41	21
117	1110101	165	75
451	111000011	703	1C3
431	110101111	657	1AF



Common Powers (1 of 2)

- Base 10

Power	Preface	Symbol	Value
10^{-12}	pico	p	.000000000001
10^{-9}	nano	n	.000000001
10^{-6}	micro	µ	.000001
10^{-3}	milli	m	.001
10^{3}	kilo	k	1000
10^{6}	mega	M	1000000
10^{9}	giga	G	1000000000
10^{12}	tera	T	1000000000000

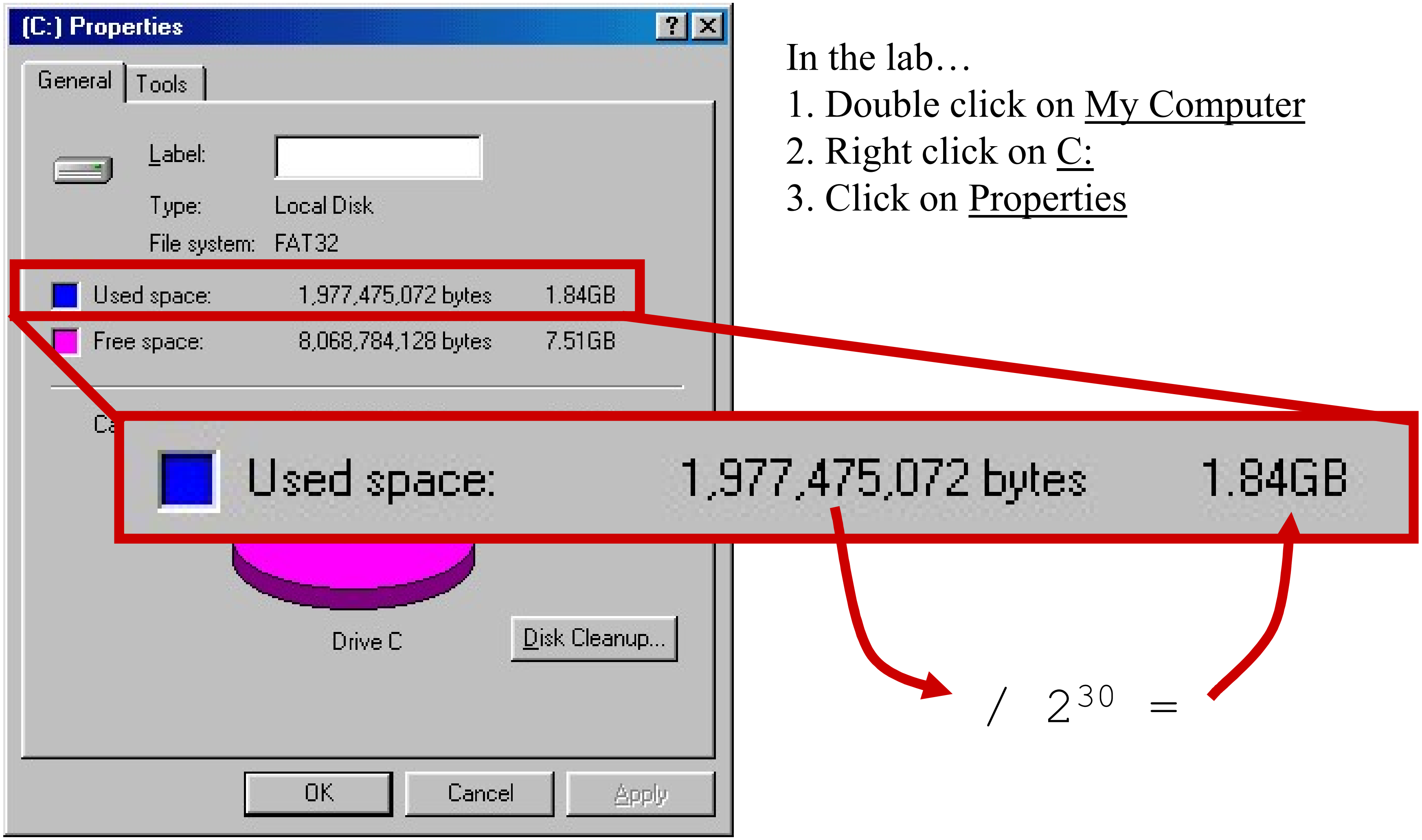
Common Powers (2 of 2)

- Base 2

Power	Preface	Symbol	Value
2^{10}	kilo	k	1024
2^{20}	mega	M	1048576
2^{30}	Giga	G	1073741824

- What is the value of “k”, “M”, and “G”?
- In computing, particularly w.r.t. memory, the base-2 interpretation generally applies

Example



In the lab…

1. Double click on <u>My Computer</u>
2. Right click on <u>C:</u>
3. Click on <u>Properties</u>

Binary Addition (1 of 2)

- Two 1-bit values

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	10

"two"

Binary Addition (2 of 2)

- Two *n*-bit values
  - Add individual bits
  - Propagate carries
  - E.g.,

```
1    1
 10101       21
+ 11001     + 25
101110       46
```

Multiplication (1 of 3)

- Decimal (just for fun)

```
     35
  x 105
  -----
    175
   000
  35
  -----
   3675
```

Multiplication (2 of 3)

- Binary, two 1-bit values

A	B	$A \times B$
0	0	0
0	1	0
1	0	0
1	1	1

Multiplication (3 of 3)

- Binary, two *n*-bit values
  - As with decimal values
  - E.g.,

```
      1110
    x 1011
  --------
      1110
     1110
    0000
   1110
  --------
  10011010
```

Fractions

- Decimal to decimal (just for fun)

```
3.14 =>    4 x 10^-2 = 0.04
           1 x 10^-1 = 0.1
           3 x 10^0  = 3
                      -----
                      3.14
```

$$
\begin{aligned}
3.14 \Rightarrow \quad & 4 \times 10^{-2} = 0.04 \\
& 1 \times 10^{-1} = 0.1 \\
& 3 \times 10^{0} = \underline{3} \\
& \phantom{3 \times 10^{0} =} 3.14
\end{aligned}
$$

Fractions

- Binary to decimal

$$
\begin{array}{llrl}
10.1011 \Rightarrow & 1 \times 2^{-4} & = & 0.0625 \\
 & 1 \times 2^{-3} & = & 0.125 \\
 & 0 \times 2^{-2} & = & 0.0 \\
 & 1 \times 2^{-1} & = & 0.5 \\
 & 0 \times 2^{0} & = & 0.0 \\
 & 1 \times 2^{1} & = & \underline{2.0} \\
 & & & 2.6875
\end{array}
$$

Fractions

- Decimal to binary

```
3.14579

11.001001...
```

```
   .14579
x       2
 0.29158
x       2
 0.58316
x       2
 1.16632
x       2
 0.33264
x       2
 0.66528
x       2
 1.33056
etc.
```

Exercise – Convert ...

Decimal	Binary	Octal	Hexa-decimal
29.8			
	101.1101		
		3.07	
			C.82

Don't use a calculator!

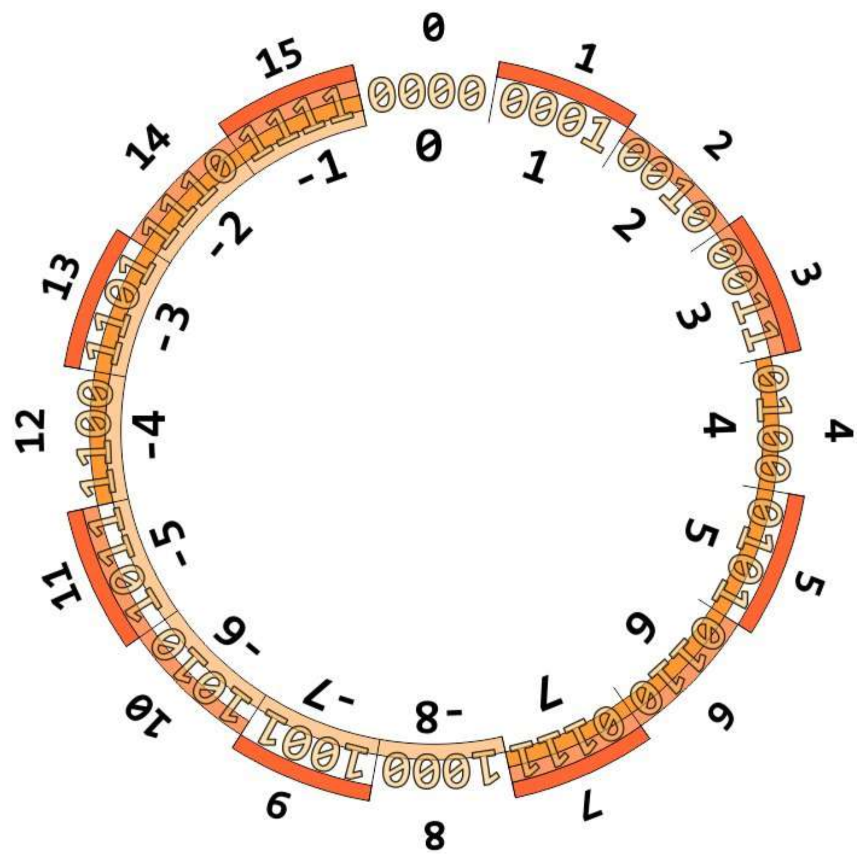
Exercise – Convert …

Answer

Decimal	Binary	Octal	Hexa-decimal
29.8	11101.110011…	35.63…	1D.CC…
5.8125	101.1101	5.64	5.D
3.109375	11.000111	3.07	3.1C
12.5078125	1100.10000010	14.404	C.82

Negative Binary Numbers
Two's Complement

- Negative number: Bitwise complement plus one
  - $0011 \equiv 3_{10}$
  - $1101 \equiv -3_{10}$
- Number wheel
- Only one zero!
- MSB is the sign digit
  - $0 \equiv$ positive
  - $1 \equiv$ negative



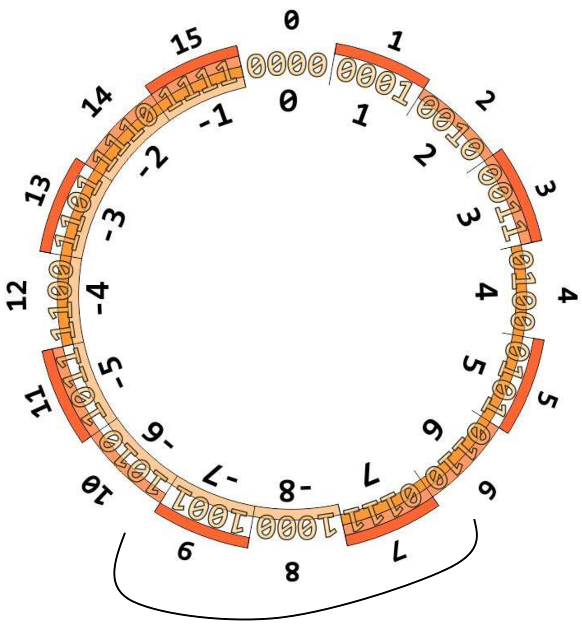
Two's Complement

- Complementing a complement ≡ the original number
- Arithmetic is easy
  - Subtraction = negation and addition
    - Easy to implement in hardware

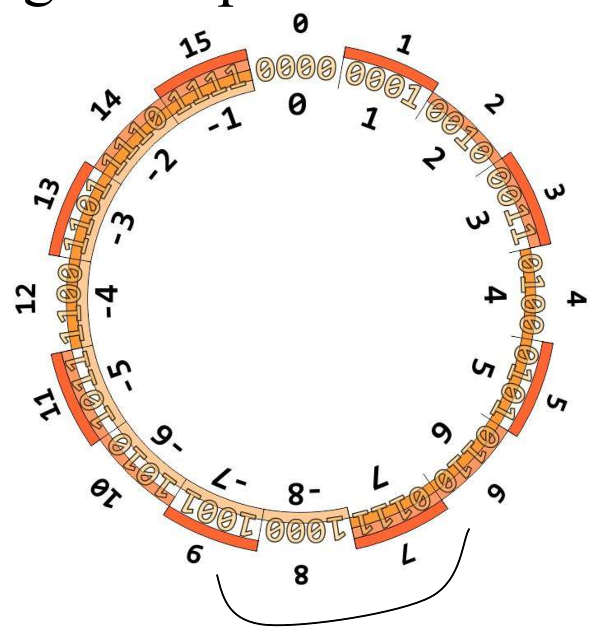
Add		Invert and add		Invert and add	
4	0100	4	0100	– 4	1100
+ 3	+ 0011	– 3	+ 1101	+ 3	+ 0011
= 7	= 0111	= 1	1 0001	– 1	1111
		drop carry	= 0001		

Two's Complement Overflow

- Summing two positive numbers gives a negative result
- Summing two negative numbers gives a positive result



6 + 4 ⇒ –6



–7 – 3 ⇒ +6

- Make sure to have enough bits to handle overflow