Number Systems

Number System

- Everything in the Computer is represented as Binary Numbers
- Registers contain either data or control information – possible data types are
  - Numbers used in computations
  - Letters of the alphabet used in data processing
  - Other discrete symbols used for specific purposes
- A number system of ***base***, or ***radix***, ***r*** is a system that uses distinct symbols for r digits
- For decimal number system r = 10 – for the natural numbers, we start counting at 0

Number System

- Other digits in the decimal numerals are 0, 1, ..., 9
- For binary numerals, r = 2 – the digits are 0, and 1
- Other number systems frequently used in computer systems are octal (r = 8) with digits 0, 1, ..., 6, 7 and hexadecimal (r = 16) with digits 0, 1, ..., 8, 9, A, B, C, D, E, and F
  - Often we use subscripts to indicate the base, e.g., $(1804)_{10}$, or $(1011)_2$, or $(615)_8$
- Normally we use base 10 when we count – we can become so used to it we don't think about it

Number System

- Although computers are very sophisticated from the outside, with all kinds of flashy buttons, screens, and so forth, the basic works are essentially many rows of on/off switches
  - A single on/off switch has only 2 possible settings or states, but a row of 2 such switches has 4 possible states

On	On	On	Off	Off	On	Off	Off

  - Computers generally work with groups of 32 switches (also called 32 bits, where a bit is the official name for position that can either be on or off) and sometimes now with groups of 64

Number System

- For example, take 9 and write that in binary

  $(9)_{10} = (1001)_2$

  and that only has four digits
  - We could seemingly manage by using only 4 bits (where we make them on, off, off, on) and it seems a waste to use 32 bits – however, it is generally simpler to decide to use 32 bits from the start – for this number 9 we can pad it out by putting zeroes in front

    $(9)_{10} = (1001)_2 = (00...001001)_2$
  - One practical aspect of this system is that it places a limit on the maximum size of the integers we can store

Number System

- So in a 32-bit computer, the decimal number 9 which is equivalent to 1001 in binary will stored in the computer memory as follows:

  0 0 0 ... 0 0 1 0 0 1

  1 2 3 ...27 28 29 30 31 32 – bit position

  - Bit position 1 is reserved for the sign of the integer – 0 for a positive integer, and 1 for a negative integer
    - That means that we have space to store integers from about -2^{31} to 2^{31}
    - To be precise, that would be $2 \times 2^{31} + 1 = 2^{32} + 1$ numbers if we include zero

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

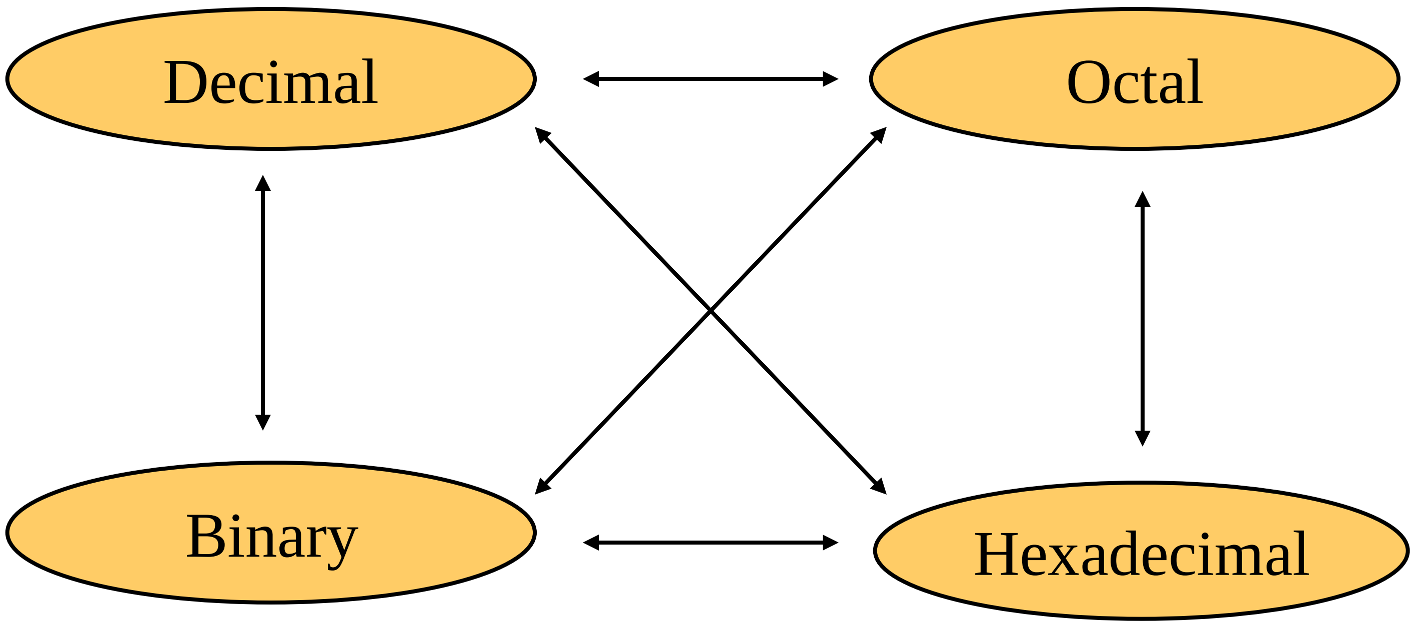
Quantities/Counting (3 of 3)

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:

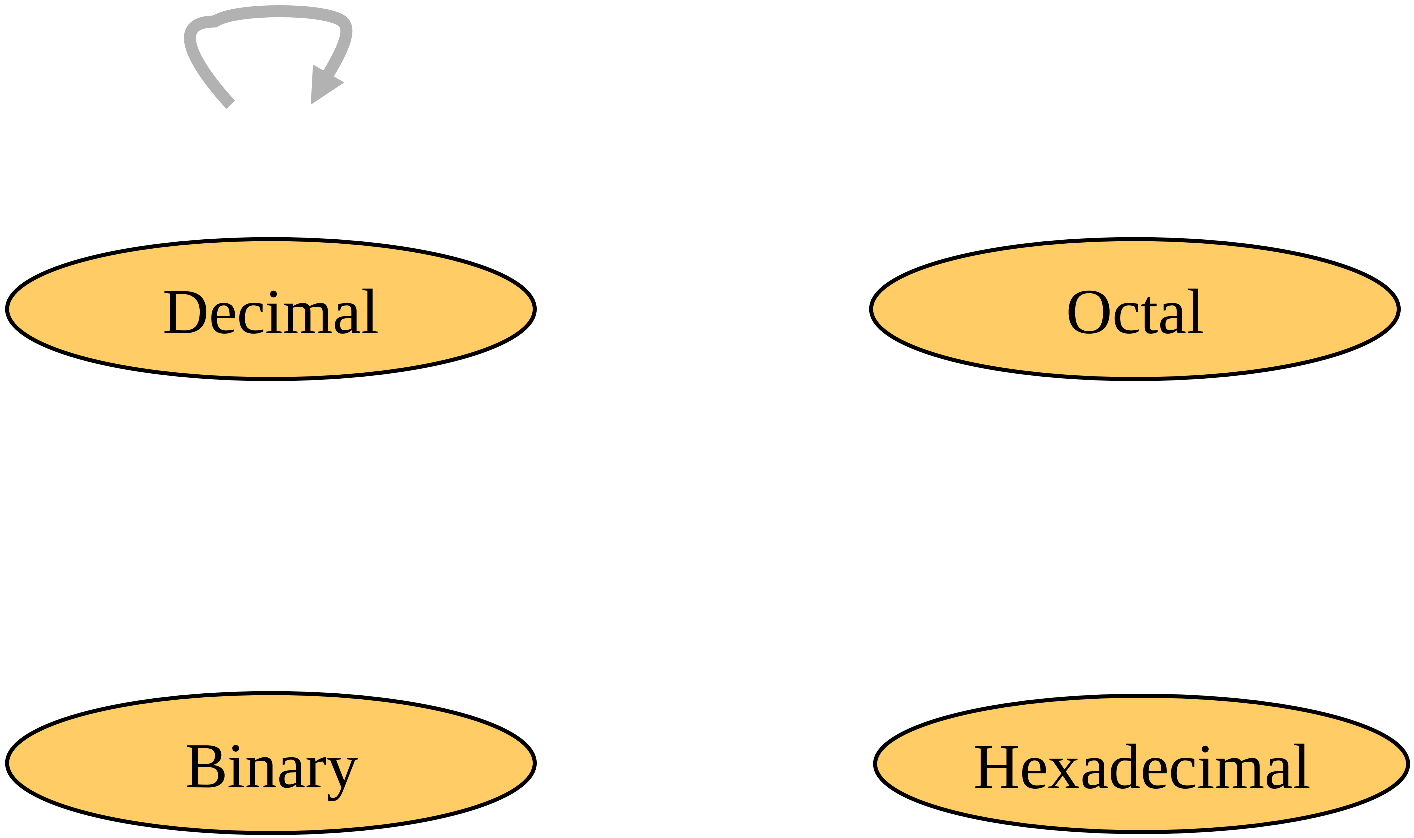


Quick Example

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

Base

Decimal to Decimal (just for fun)

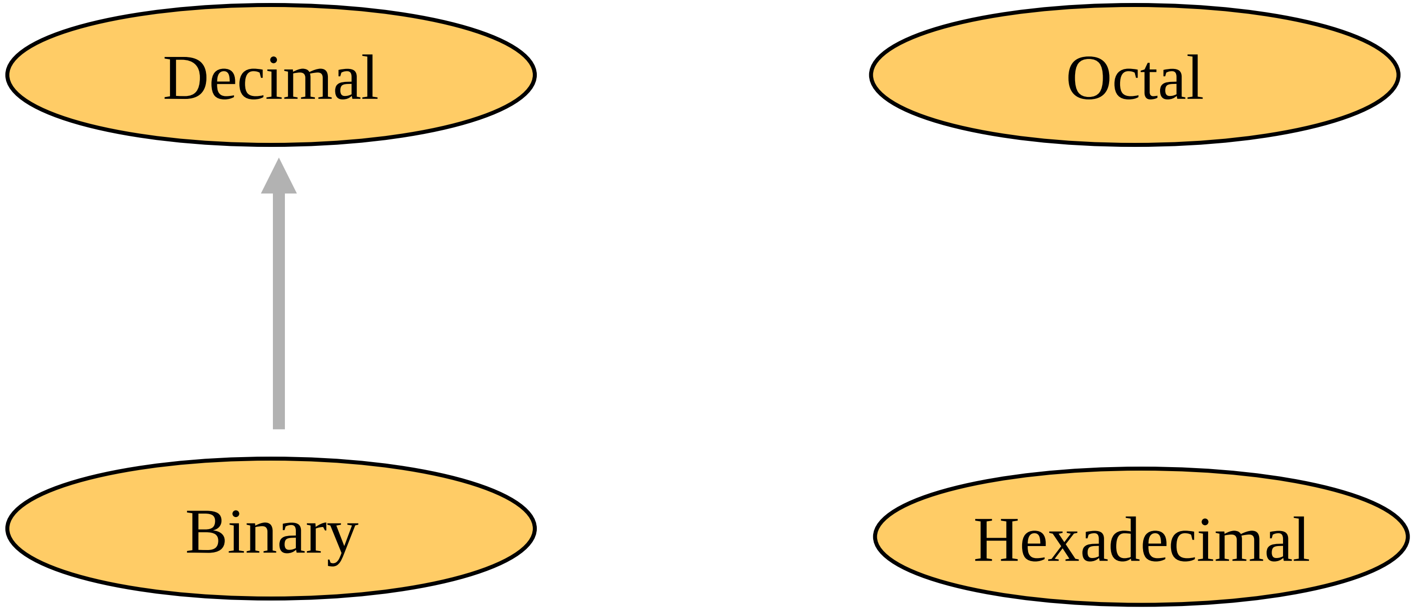


Weight

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

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

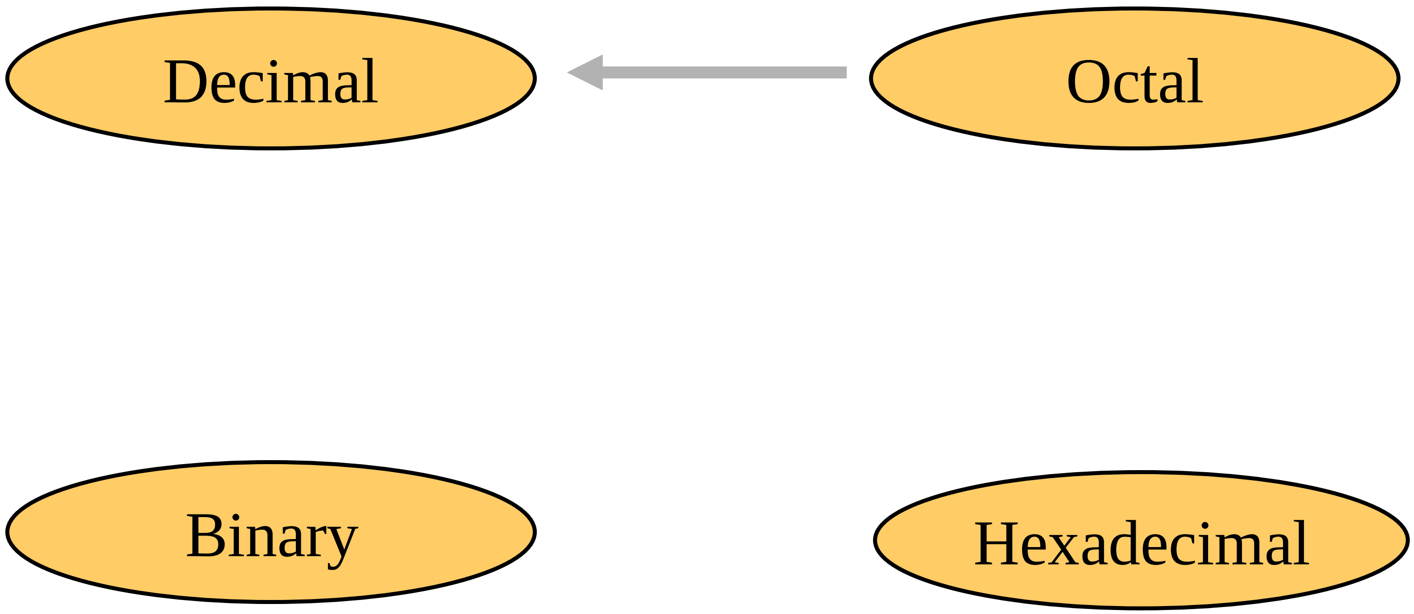
Example

Bit “0”

$101011_2 \Rightarrow$

$$
\begin{array}{lr}
1 \times 2^0 = & 1 \\
1 \times 2^1 = & 2 \\
0 \times 2^2 = & 0 \\
1 \times 2^3 = & 8 \\
0 \times 2^4 = & 0 \\
1 \times 2^5 = & 32 \\
\hline
 & 43_{10}
\end{array}
$$

Octal to Decimal



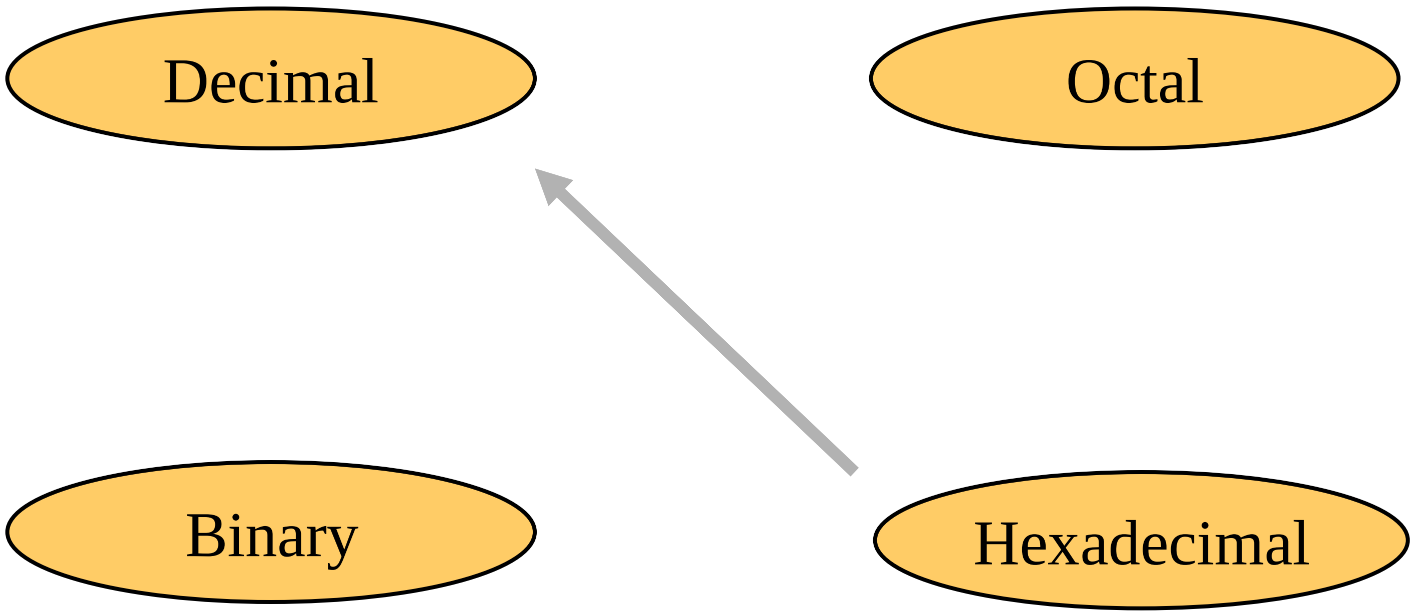
Octal to Decimal

- Technique
  - Multiply each bit by $\underline{8}^n$, 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

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

Hexadecimal to Decimal



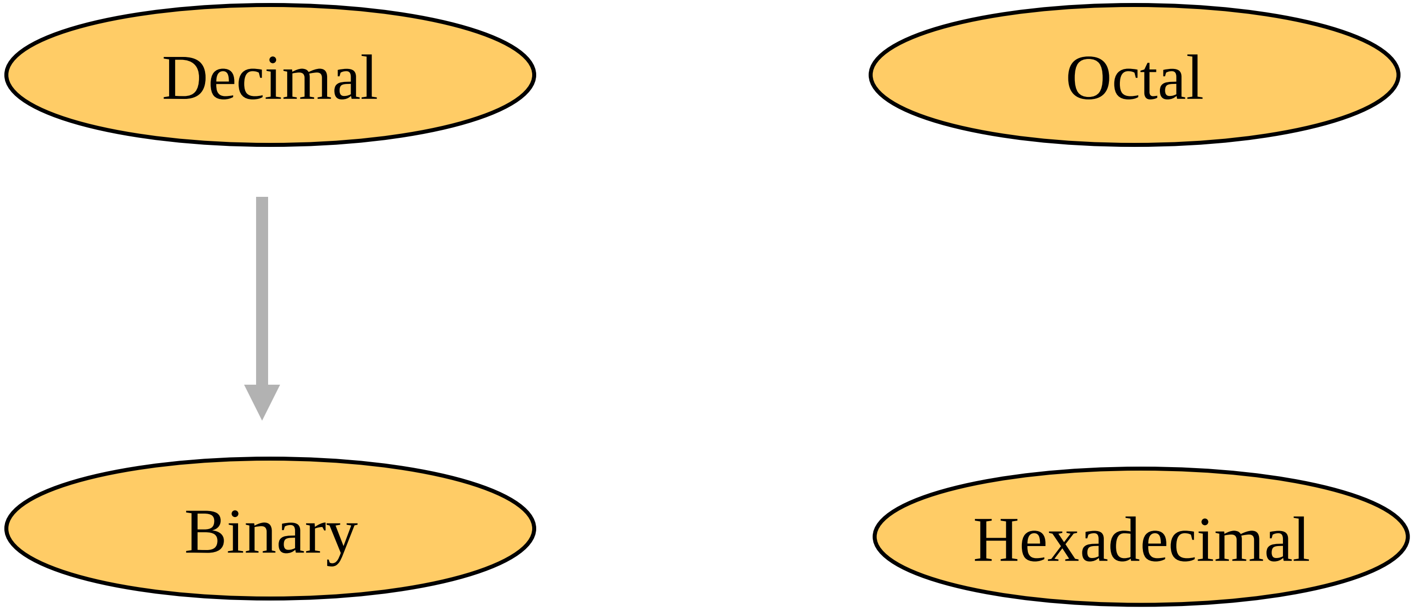
Hexadecimal to Decimal

- 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

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

Decimal to Binary



Decimal to Binary

- Technique
  - Divide by two, keep track of the remainder
  - First remainder is bit 0
  - Second remainder is bit 1
  - Etc.

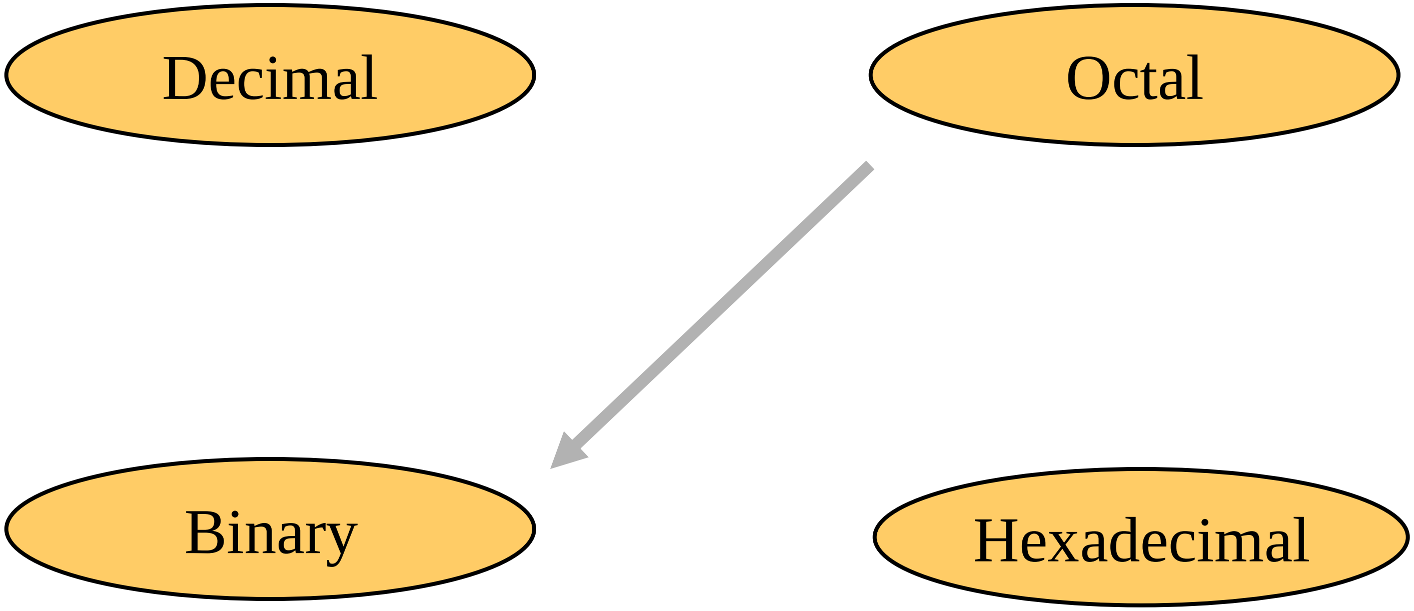
Example

$125_{10} = ?_2$

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

$125_{10} = 1111101_2$

Octal to Binary



Octal to Binary

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

Example

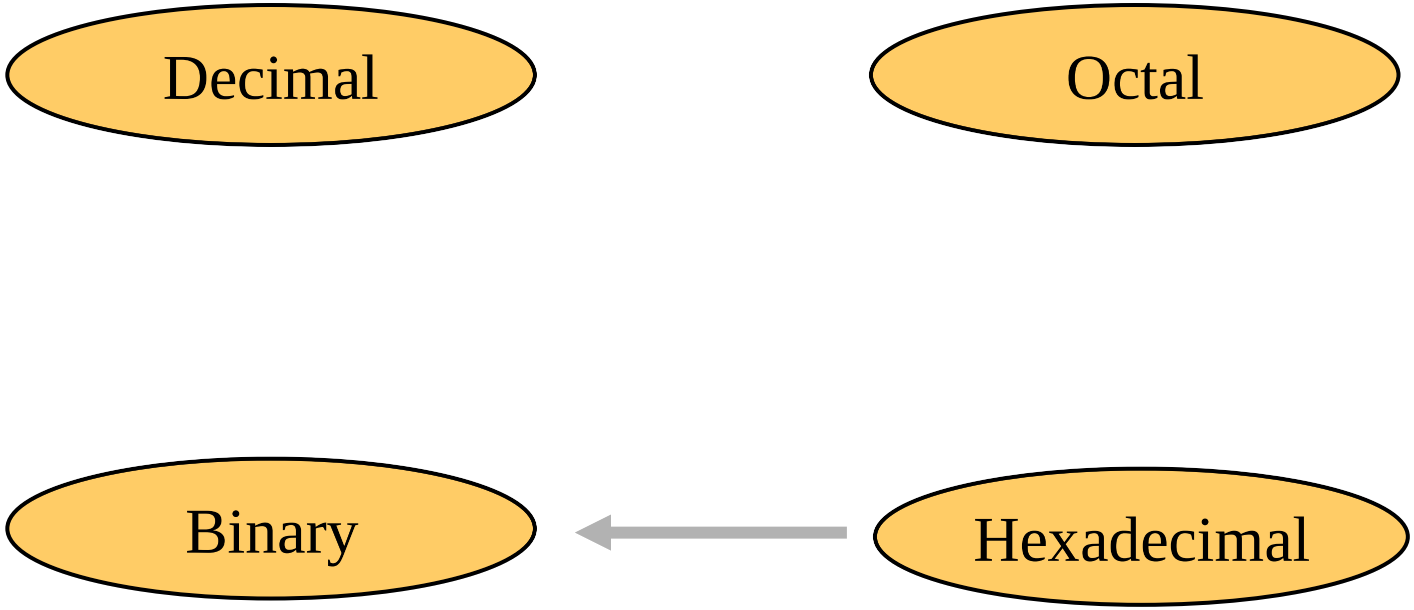
$705_8 = ?_2$

7 0 5

111 000 101

$705_8 = 111000101_2$

Hexadecimal to Binary



Hexadecimal to Binary

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

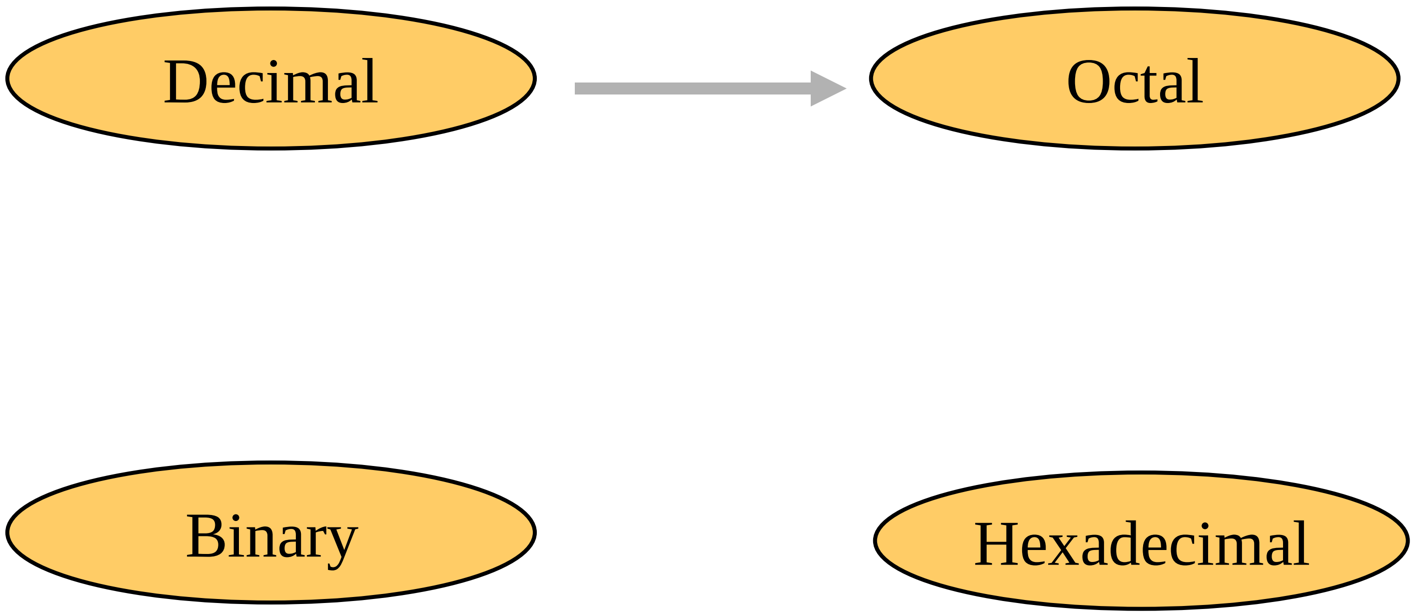
Example

$10AF_{16} = ?_2$

1	0	A	F
0001	0000	1010	1111

$10AF_{16} = 0001000010101111_2$

Decimal to Octal



Decimal to Octal

- Technique
  - Divide by 8
  - Keep track of the remainder

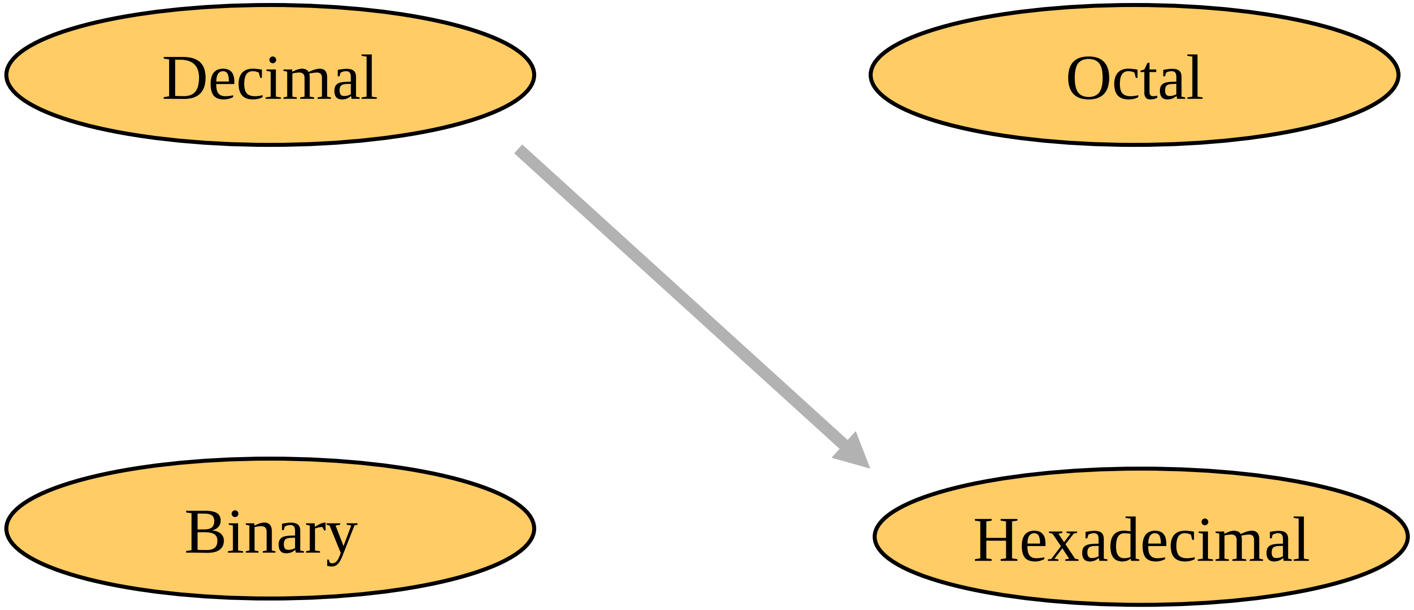
Example

$1234_{10} = ?_8$

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

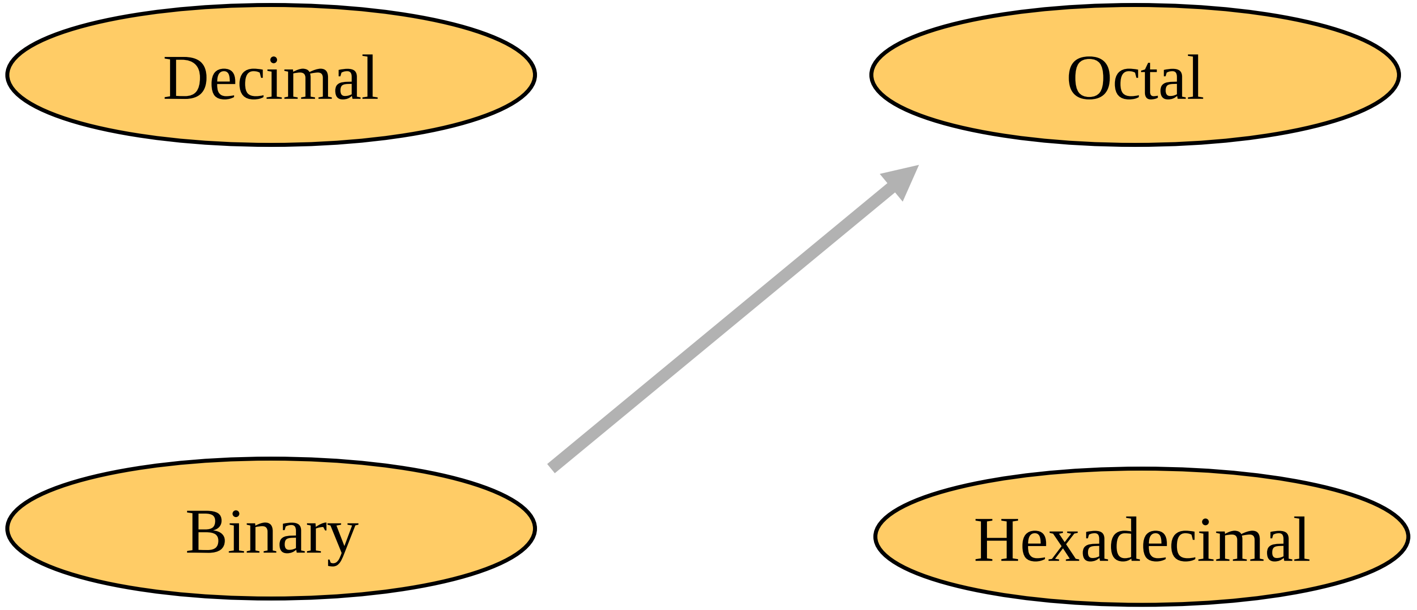
Example

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

$$
\begin{array}{r|l l}
16 & 1234 & \\ \hline
16 & 77 & 2 \\ \hline
16 & 4 & 13 = D \\ \hline
 & 0 & 4
\end{array}
$$

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

Binary to Octal



Binary to Octal

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

Example

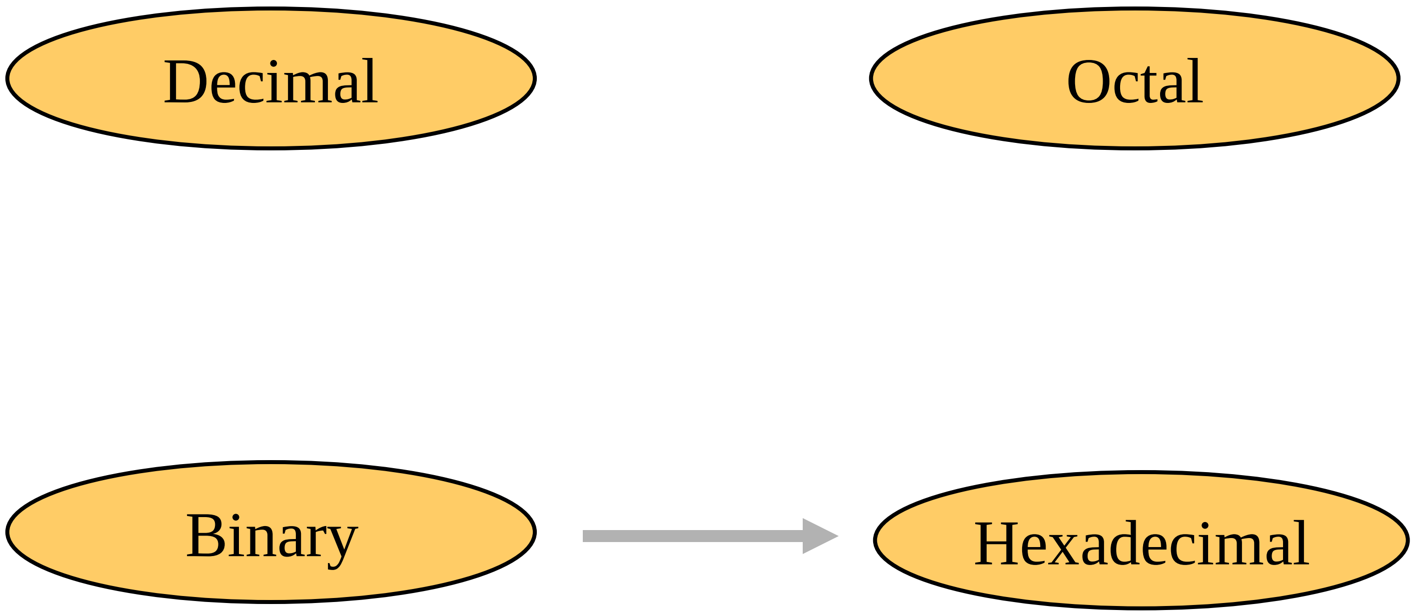
$1011010111_2 = ?_8$

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

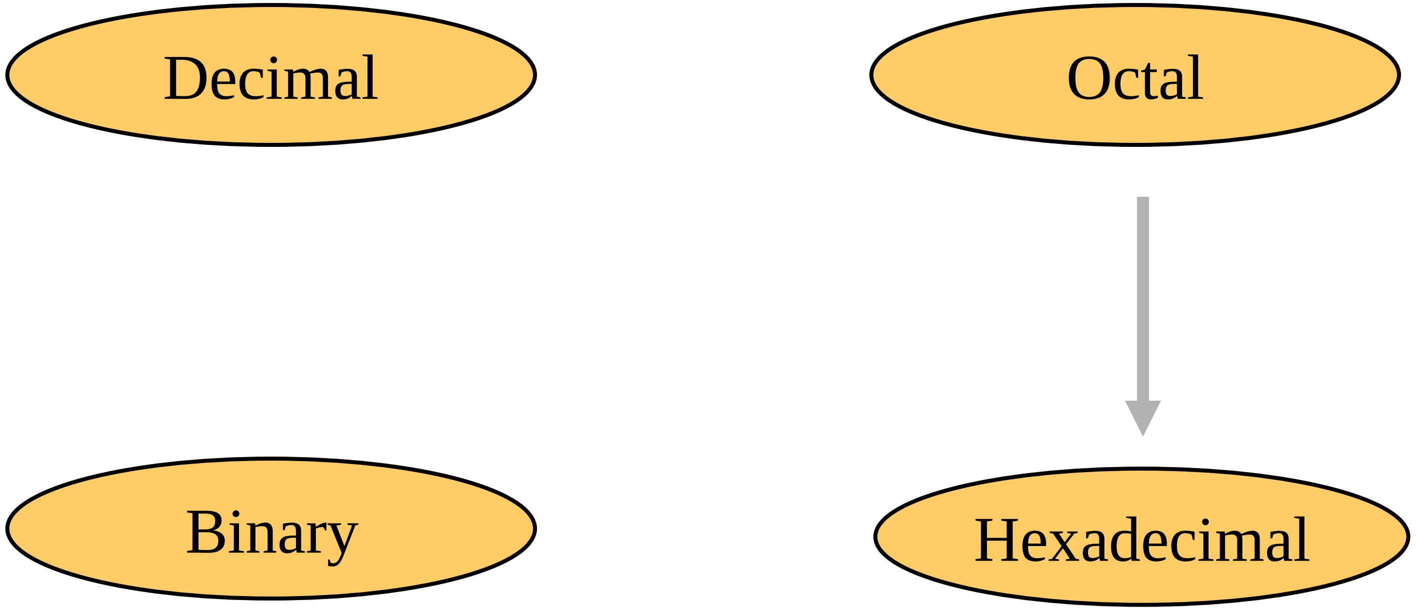
$1010111011_2 = ?_{16}$

10 1011 1011

2 B B

$1010111011_2 = 2BB_{16}$

Octal to Hexadecimal



Octal to Hexadecimal

- Technique
  - Use binary as an intermediary

Example

$1076_8 = ?_{16}$

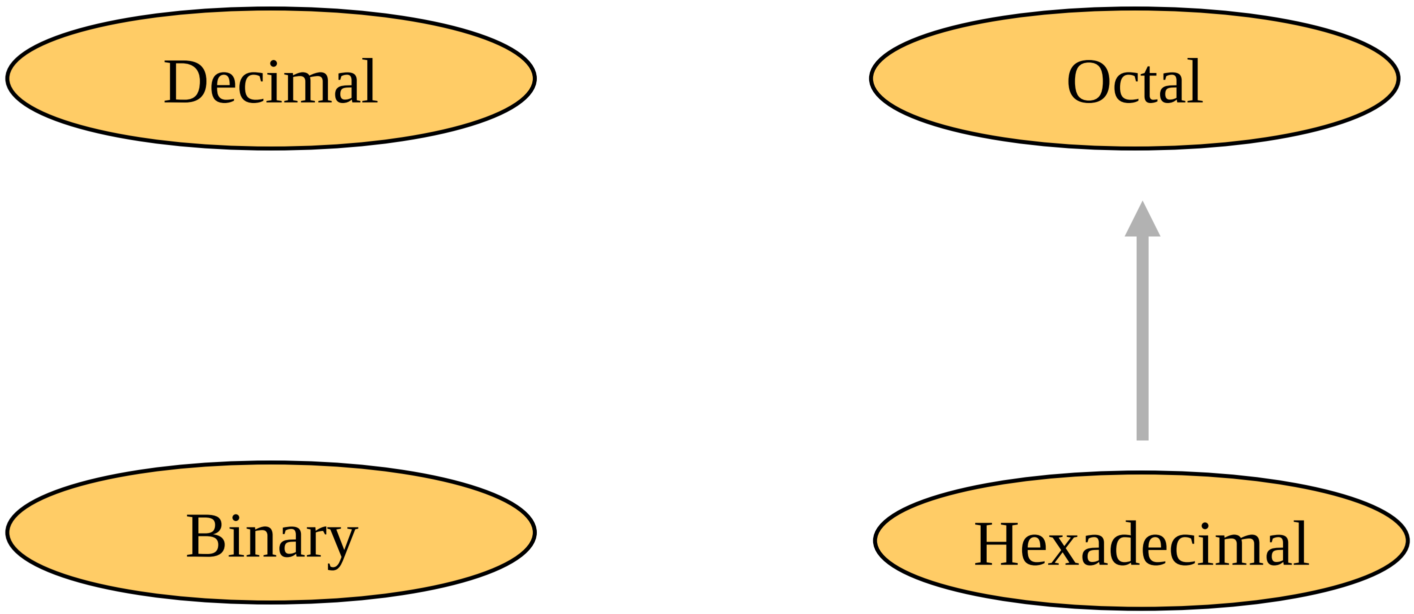
1 0 7 6

001 000 111 110

2 3 E

$1076_8 = 23E_{16}$

Hexadecimal to Octal

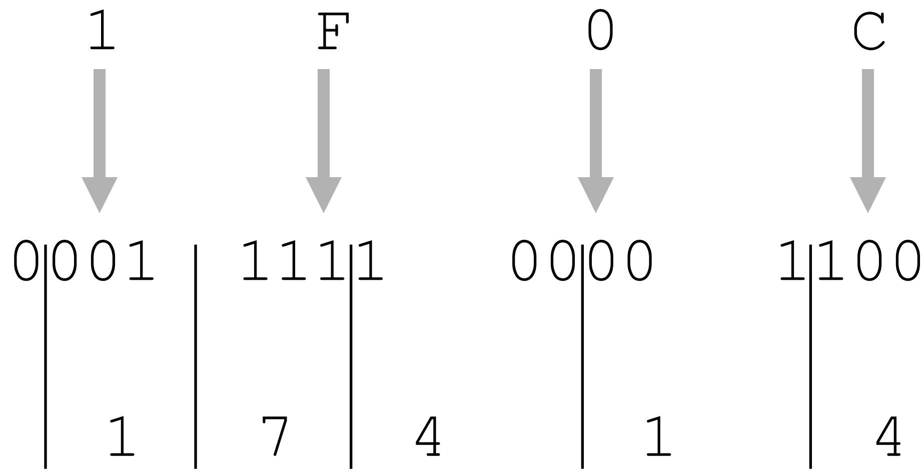


Hexadecimal to Octal

- Technique
  - Use binary as an intermediary

Example

$1F0C_{16} = ?_8$



$1F0C_{16} = 17414_8$

Exercise – Convert ...

Decimal	Binary	Octal	Hexa-decimal
33			
	1110101		
		703	
			1AF

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

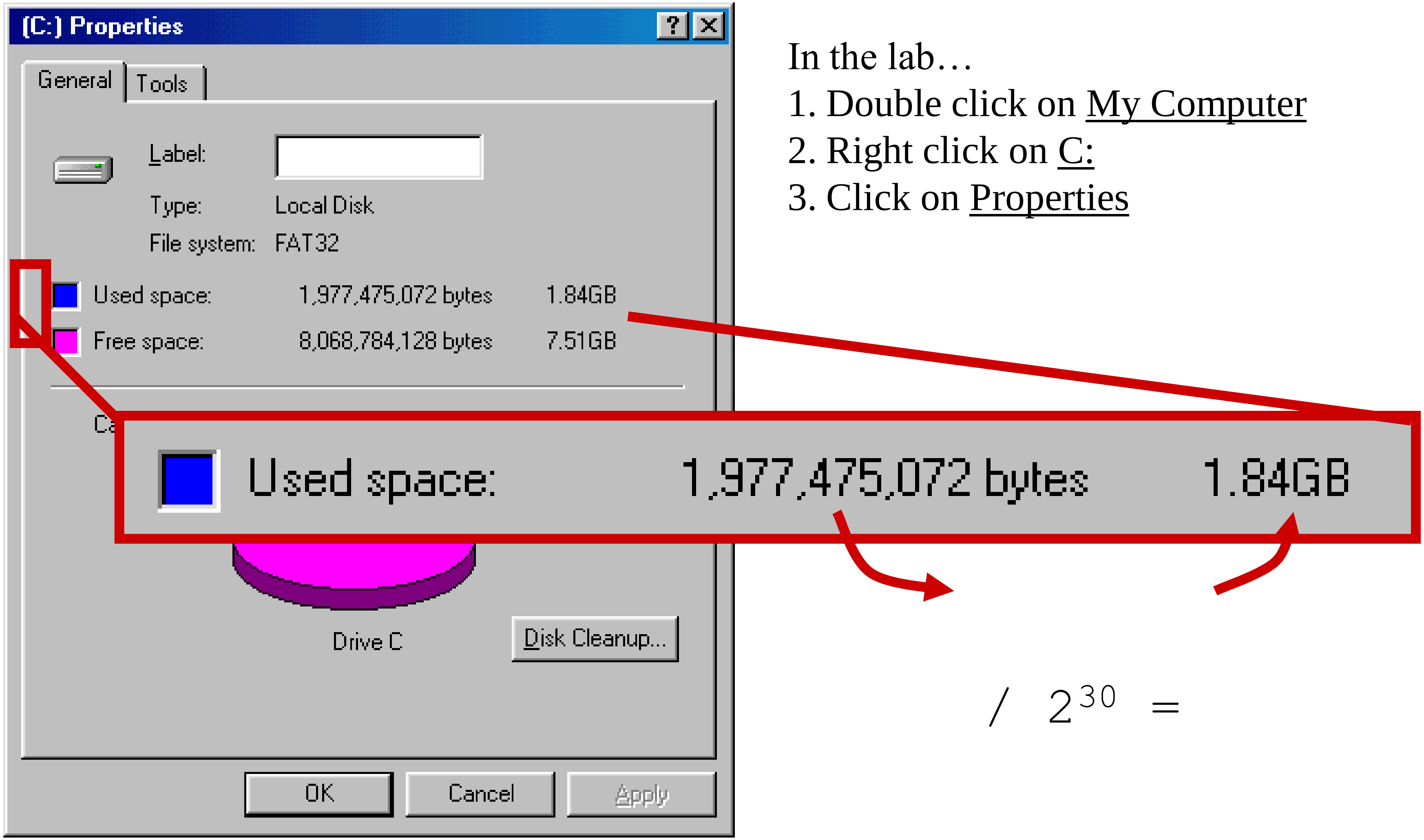
Common Powers (2 of 2)

- 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 My Computer
2. Right click on C:
3. Click on Properties

$/ \quad 2^{30} \quad =$

Exercise – Free Space

- Determine the "free space" on all drives on a machine in the lab

Drive	Free space	
	Bytes	GB
A:		
C:		
D:		
E:		
etc.		

Fractions

- Decimal to decimal (just for fun)

$$
\begin{array}{llrl}
3.14 \Rightarrow & 4 \times 10^{-2} & = & 0.04 \\
& 1 \times 10^{-1} & = & 0.1 \\
& 3 \times 10^{0} & = & 3 \\
\hline
& & & 3.14
\end{array}
$$

Fractions

- Binary to decimal

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

Fractions

- Decimal to binary

3.14579

```
 .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.
```

11.001001...

Exercise – Convert ...

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

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

Number System

- Sometimes for big numbers, we use scientific notation (engineering). The usual scientific notation is like this
  - $54321.67 = 5.432167 \times 10^4$
  - and we call 5.432167, the *mantissa*, and the power (in this case 4), the *exponent*
  - In binary number system too, the numbers can be expressed in scientific notations

Number System

- Like the decimal system, multiplying or dividing by powers of simply moves the 'binary point'
  - $(1101.11)_2 = (1.10111)_2 \times 2^3$
- What do we do with these numbers in scientific notation?
  - Within the 32-bits they have to store the mantissa and the exponent
    - Computers usually allocate 24 bits for storing the mantissa (including its possible sign) and the remaining 8 bits for the exponent
    - In our example, 24 bits is plenty for the mantissa and we would need make it longer to fill up the 24 bits: $(1.10111000\ ...)_2$ will be same as $(1.10111)_2$

Number System

- If there are numbers that need more than 24 binary digits in the mantissa, they are generally rounded off
- If we now fill out the 32 bits for the number $(1101.11)_2$
- Remember that the exponent was $(3)_{10}$ or $(11)_2$, so
- 0 1 1 0 1 1 1 0 ... 0 || 0 ... 0 1 1
- 1 2 3 4 5 6 7 8 ...24 ||25...30 31 32 – bit position
- Bit 1 is kept for the possible sign on the mantissa, in particular, the value of bit 1 is 0 for positive numbers and 1 for negative numbers

THANKS