Data Representation

Sabah Sayed

Data Representation

- makes it possible to convert letters, sounds, and images into a form computers can use for processing
- Bit Patterns are used to represent all types of data
  - Numbers
  - Text characters
  - Images
  - Sound

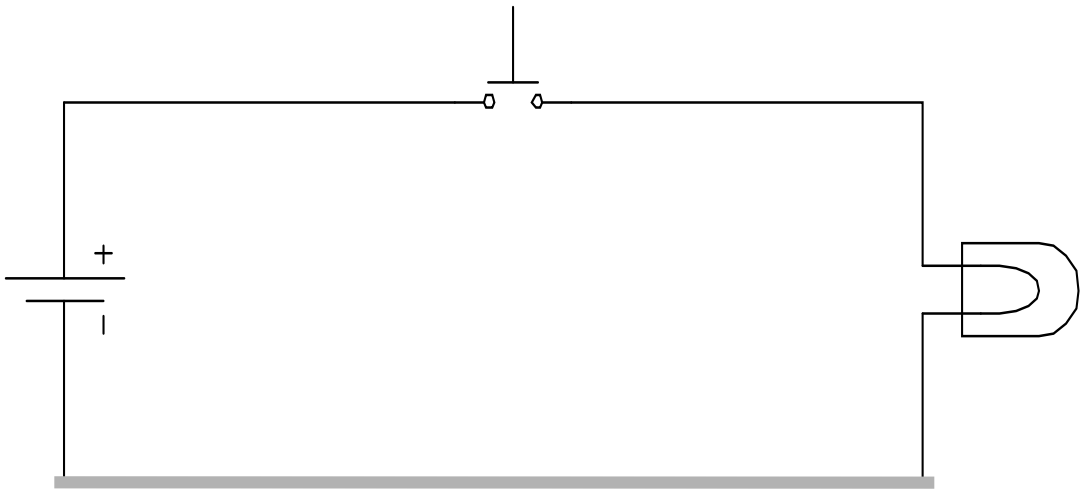
Data Representation

- In computer (digital device) :
  - Everything is represented by series of 0's and 1's – called **Binary Digit** (**bit**, or **b**)
  - Everything is measured in **Byte** (**B**).

Byte	8 bits
Kilobyte	1024 or 2^{10} bytes
Megabytes	1,048,576 or 2^{20} bytes
Gigabytes	2^{30} bytes
TeraBytes	2^{40} bytes
PetaBytes	2^{50} bytes
ExaBytes	2^{60} bytes

Where is the bit stored ?

- Switch open = input logic state 0
- Switch closed = input logic state1
- Lamp off = output logic state 0
- Lamp on = output logic state1



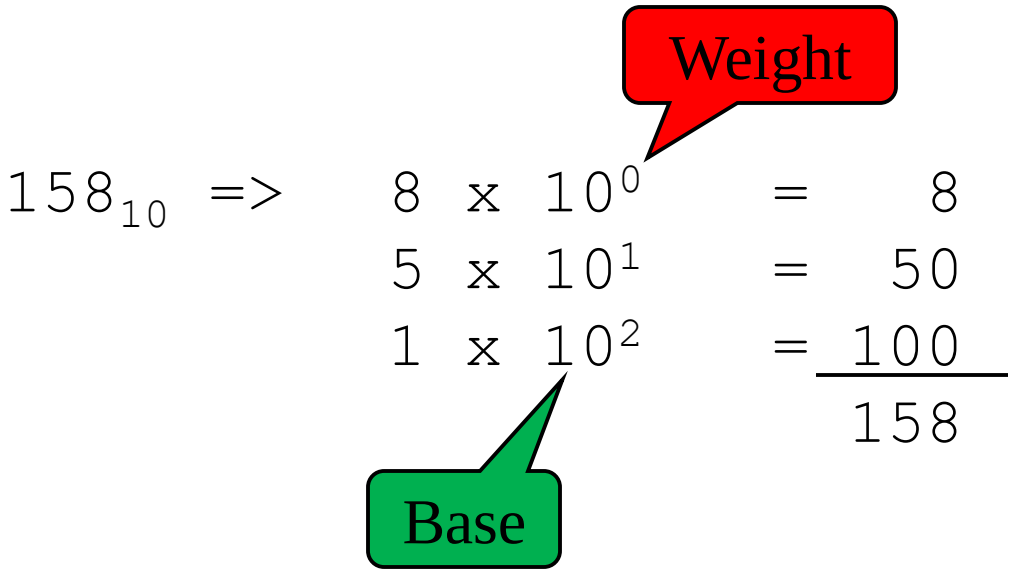
Why Binary System?

- It is easier to implement hardware to deal with binary values than to deal with 10 different values.
  - It can be represented by a transistor being off (0) or on (1).
  - It can be a magnetic stripe magnetized with North in one direction (0) or the opposite (1).

Base 10 (Decimal numbers)

- What does 269 mean?
- $269 = 2 \times 100 + 6 \times 10 + 9 \times 1$

  $= 2 \times 10^2 + 6 \times 10^1 + 9 \times 10^0$



Base 10 vs Base 2

Base 10

$269 = 2 \times 100 + 6 \times 10 + 9 \times 1$

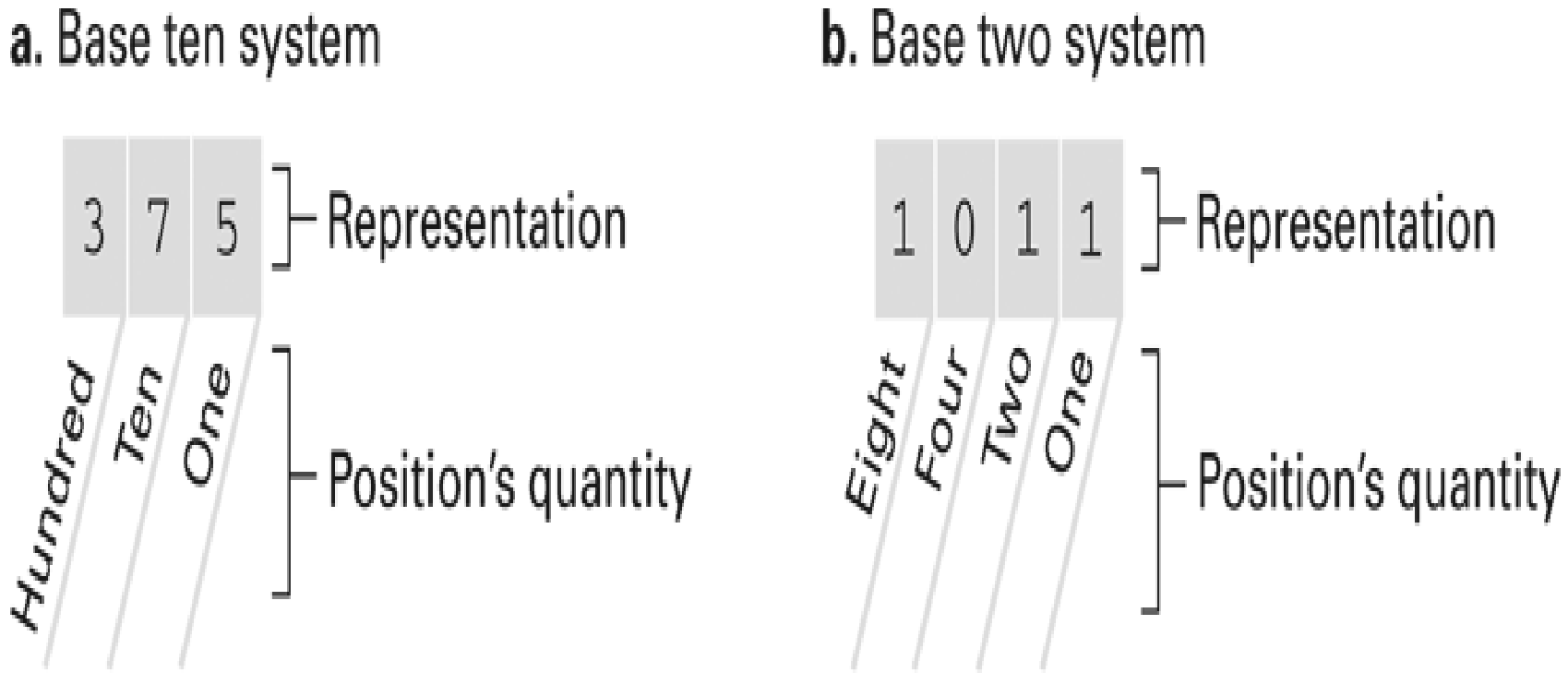
$= 2 \times 10^2 + 6 \times 10^1 + 9 \times 10^0$

Base 2

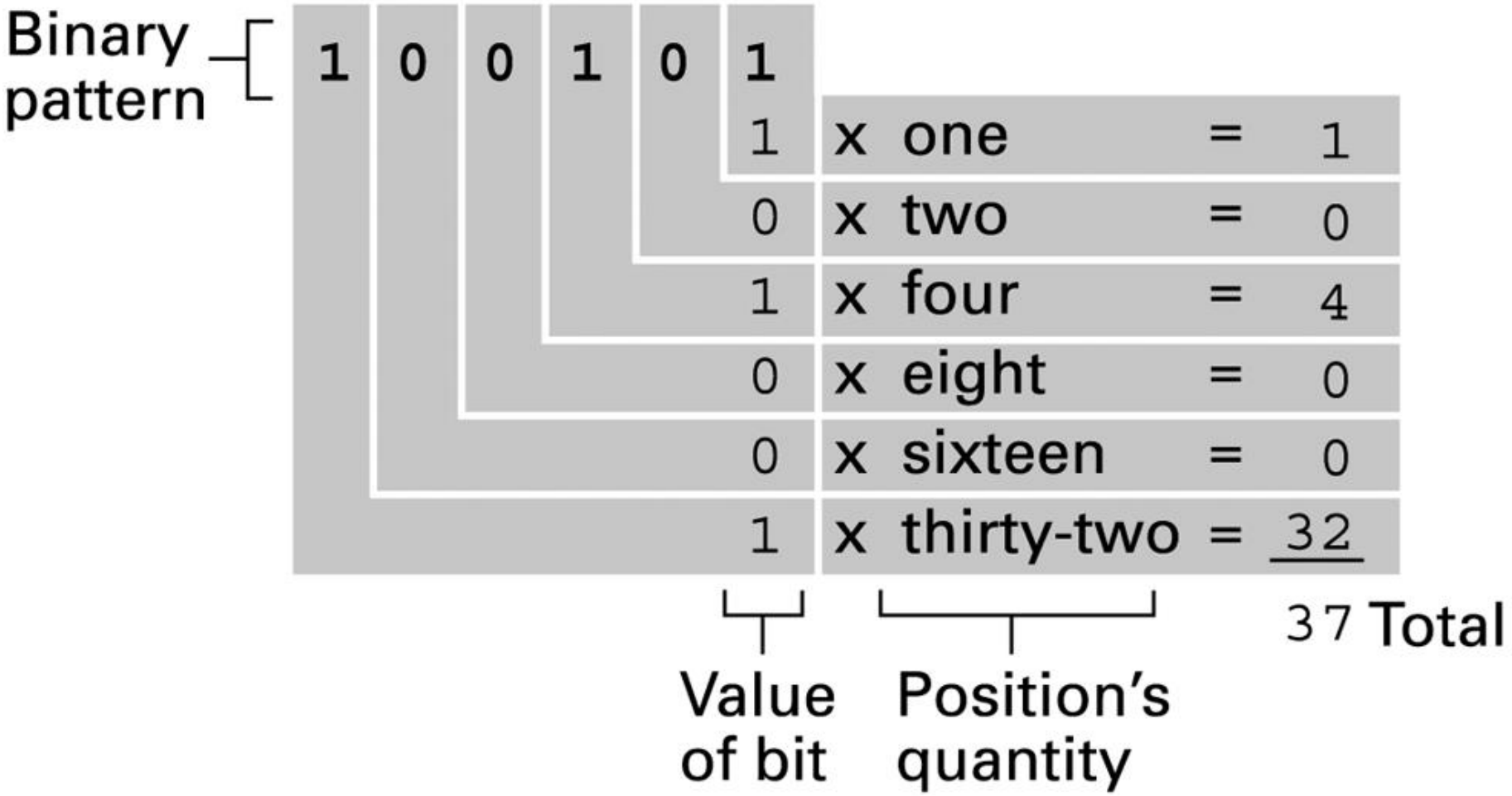
$1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$1011 = 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$

Base 10 vs Base 2



Decoding A Binary Number

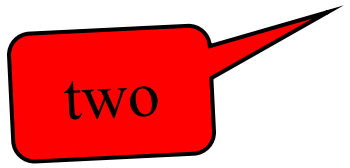


Binary mathematics

- Binary addition
- Binary subtraction
- Binary multiplication

Binary Addition

```
  0     1     0     1
+ 0   + 0   + 1   + 1
---   ---   ---   ---
  0     1     1    10
```



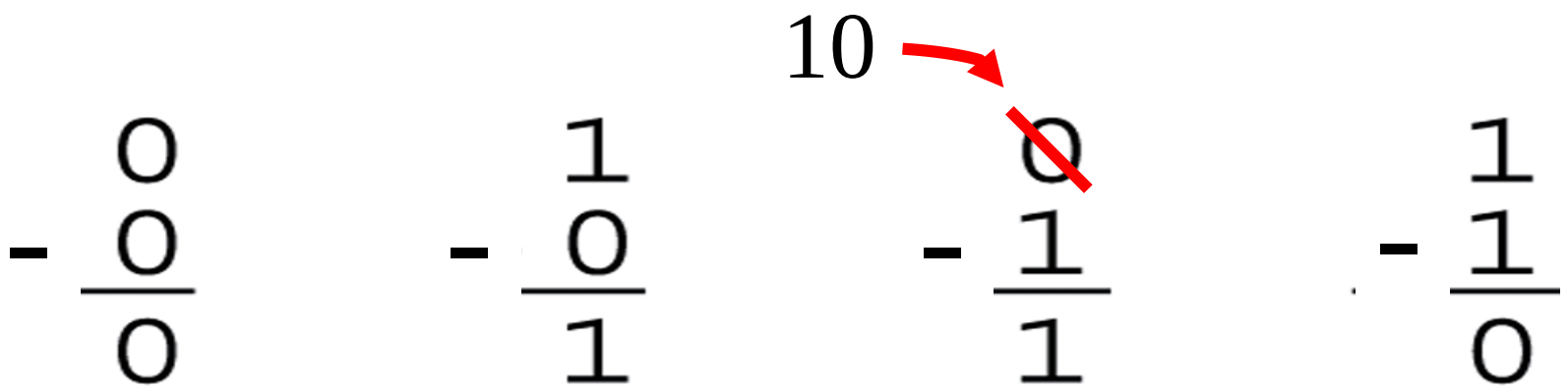
Two *n*-bit numbers

- Add individual bits
- Propagate carries

```
  10101
+ 11001
-------
 101110
```

```
  21
+ 25
----
  46
```

Binary Subtraction



$$\begin{array}{rrrrrrrr} & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ - & & & 1 & 0 & 0 & 1 & 0 \\ \hline & & & & & & & \\ \hline \end{array}$$

Binary Subtraction

			1	1	1	1			
		0	~~1~~0	~~1~~0	~~1~~0	~~1~~0	10		
	193		1	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	1
-	34	-	0	0	1	0	0	0	1	0
	159_{10}		1	0	0	1	1	1	1	1_2

Binary Multiplication

- Decimal

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

Binary Multiplication

- Binary, two 1-bit values

A	B	A × B
0	0	0
0	1	0
1	0	0
1	1	1

Binary Multiplication

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

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

```
    14
  x 11
  ----
   154
```

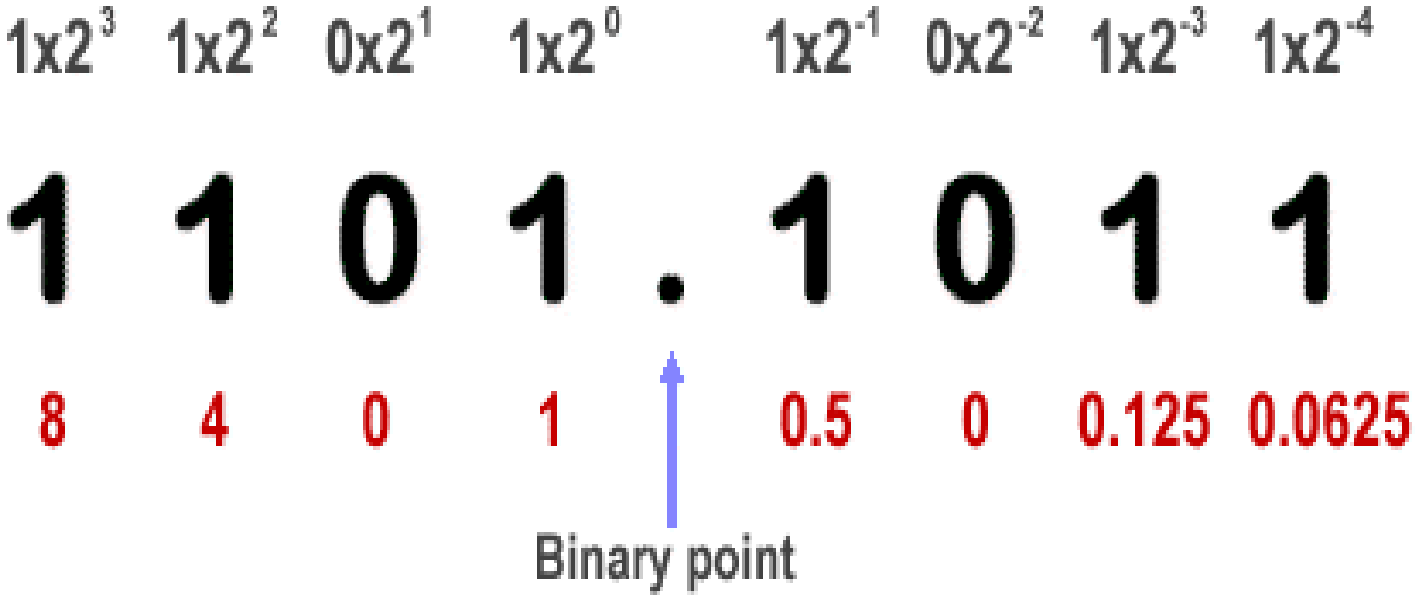
$154_{10} = 10011010_2$

Binary Fractions

- Decimal

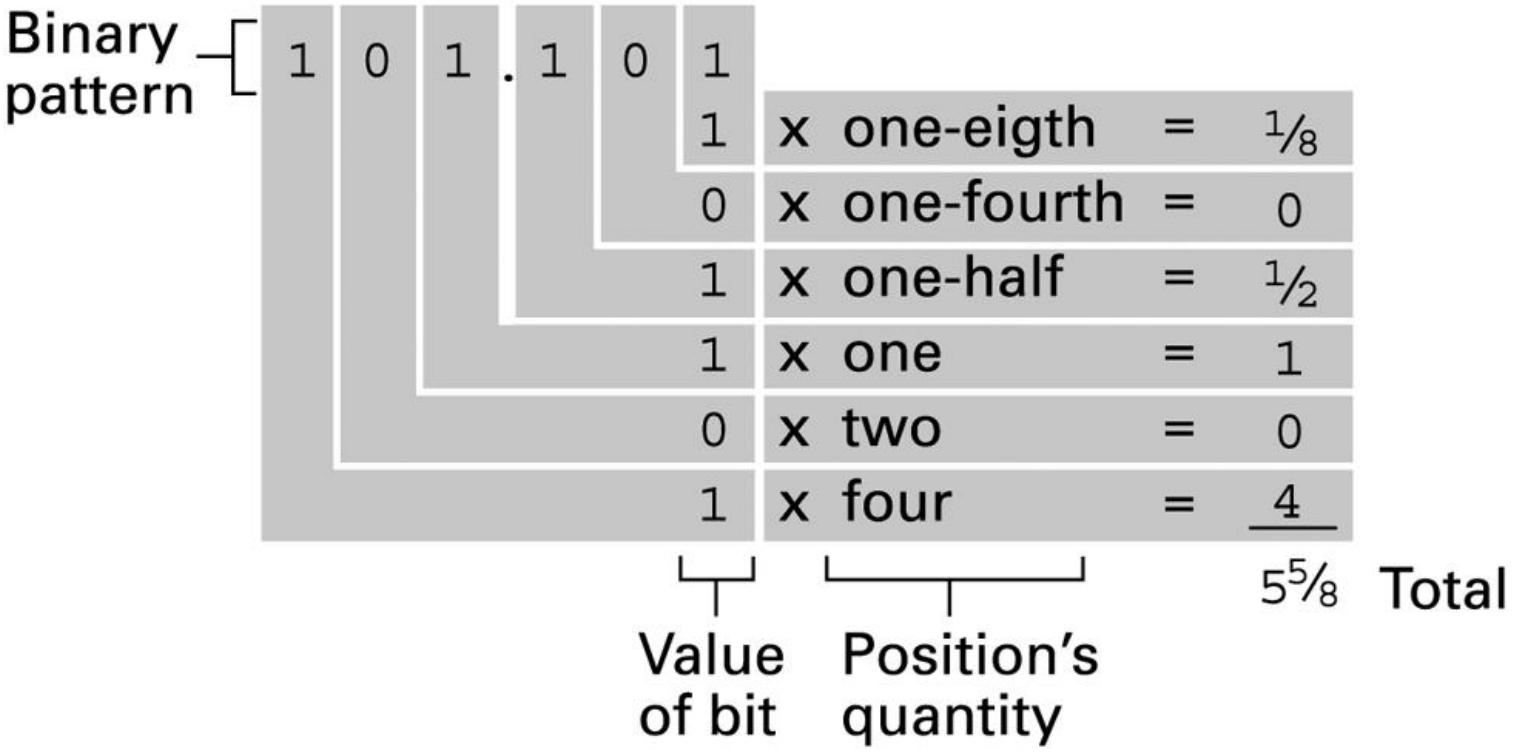
$$
\begin{aligned}
3.14 \Rightarrow \quad & 4 \times 10^{-2} = 0.04 \\
& 1 \times 10^{-1} = 0.1 \\
& 3 \times 10^{0} = 3 \\
\hline
& 3.14
\end{aligned}
$$

Binary Fractions



Binary Fractions

- **Decoding the binary representation of 101.101**



Hexadecimal

- Binary code is too long in representation. Hex is much shorter.
- Base 16

  0 … 9

  A-10 B-11 C-12 D-13 E-14 F-15
- How to do hexadecimal with hexadecimal values:
  - Addition
  - Subtraction
  - Multiplication

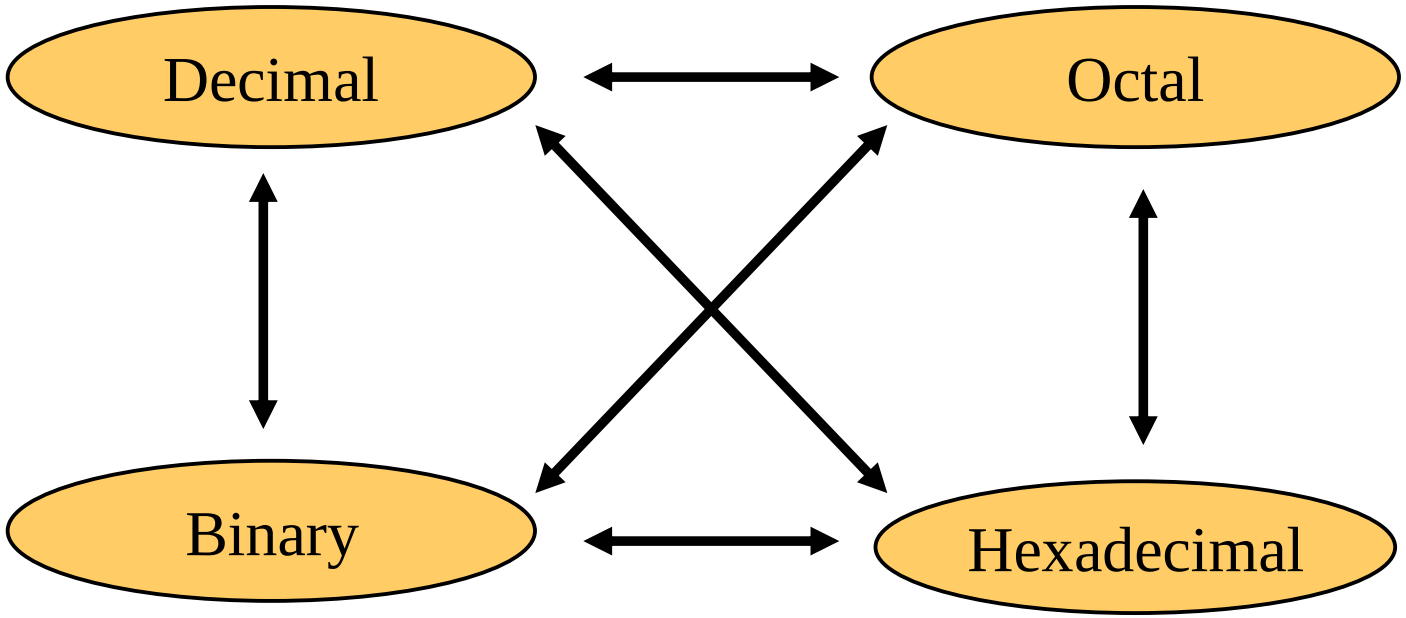
Octal

- Base 8

  0 … 7
- How to do octal with octal values:
  - Addition
  - Subtraction
  - Multiplication

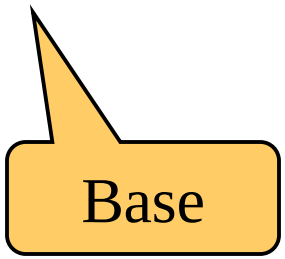
Conversion Among Bases

- The possibilities:

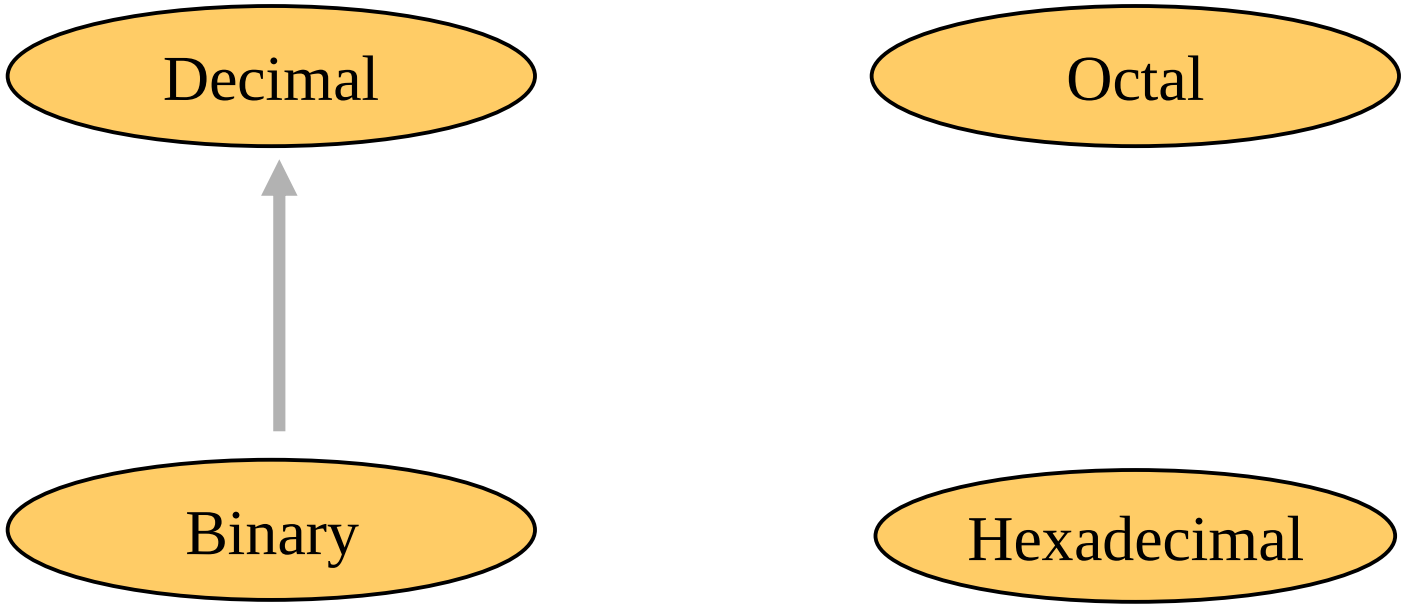


Quick Example

$$25_{10} = 11001_{2} = 31_{8} = 19_{16}$$



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

$101011_2 = ?_{10}$

1	**0**	**1**	**0**	**1**	**1**
x	**x**	**x**	**x**	**x**	**x**
$\mathbf{2^5}$	$\mathbf{2^4}$	$\mathbf{2^3}$	$\mathbf{2^2}$	$\mathbf{2^1}$	$\mathbf{2^0}$

Example

$101011_2 = ??_{10}$

1	0	1	0	1	1
x	x	x	x	x	x
2^5	2^4	2^3	2^2	2^1	2^0

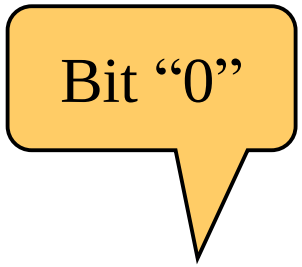
Example

$101011_2 = ??_{10}$

1	0	1	0	1	1
x	x	x	x	x	x
2^5	2^4	2^3	2^2	2^1	2^0
32	0	8	0	2	1

\+ + + + + = 43_{10}

Example



101011_2 =>

$1 \times 2^0 = 1$

$1 \times 2^1 = 2$

$0 \times 2^2 = 0$

$1 \times 2^3 = 8$

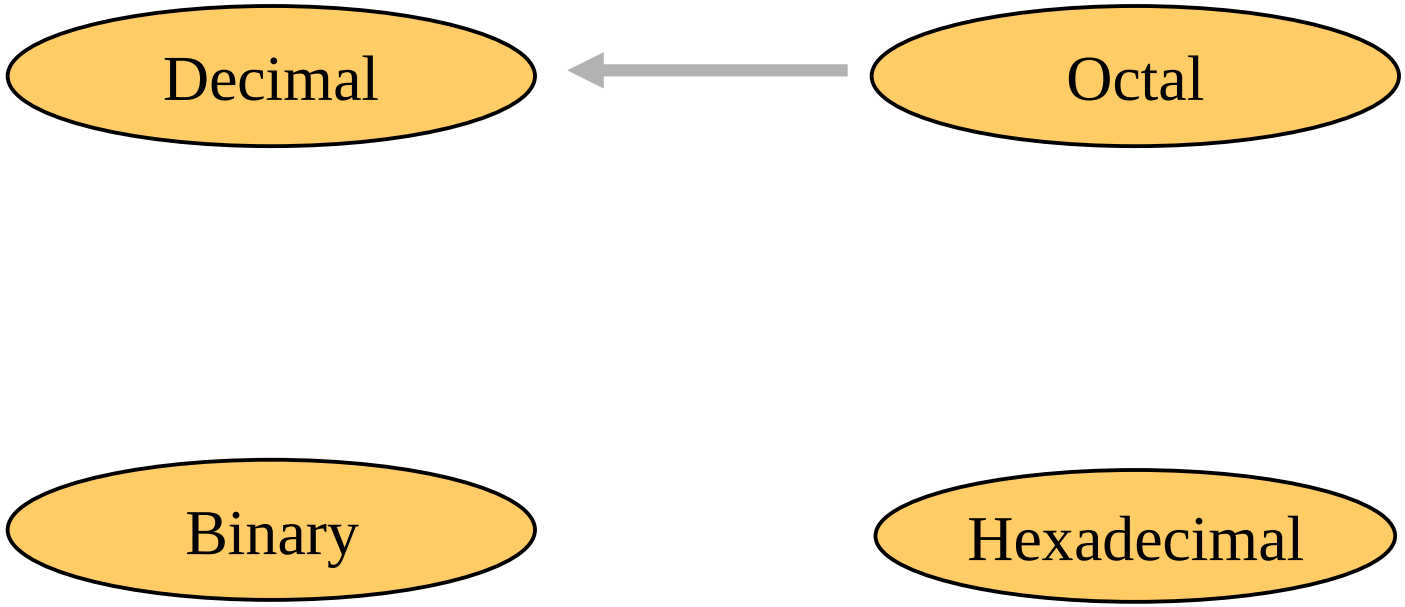
$0 \times 2^4 = 0$

$1 \times 2^5 = 32$

43_{10}

$$N_B = \sum_{i=0}^{n-1} d_i \cdot B^i = d_{n-1}B^{n-1} + d_{n-2}B^{n-2} + \ldots + d_1B^1 + d_0B^0$$

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 =>

$4 \times 8^0 = 4$

$2 \times 8^1 = 16$

$7 \times 8^2 = 448$

468_{10}

$$N_B = \sum_{i=0}^{n-1} d_i \cdot B^i = d_{n-1}B^{n-1} + d_{n-2}B^{n-2} + \ldots + d_1B^1 + d_0B^0$$

Example

724_8 =>

$4 \times 8^0 = 4$

$2 \times 8^1 = 16$

$7 \times 8^2 = 448$

468_{10}

$$N_B = \sum_{i=0}^{n-1} d_i \cdot B^i = d_{n-1}B^{n-1} + d_{n-2}B^{n-2} + \ldots + d_1B^1 + d_0B^0$$

Example

724_8 =>

$4 \times 8^0 = 4$

$2 \times 8^1 = 16$

$7 \times 8^2 = 448$

468_{10}

$$N_B = \sum_{i=0}^{n-1} d_i \cdot B^i = d_{n-1}B^{n-1} + d_{n-2}B^{n-2} + \ldots + d_1B^1 + d_0B^0$$

Example

724_8 =>

$4 \times 8^0 = 4$

$2 \times 8^1 = 16$

$7 \times 8^2 = 448$

$\sum$ 468_{10}

$$N_B = \sum_{i=0}^{n-1} d_i \cdot B^i = d_{n-1}B^{n-1} + d_{n-2}B^{n-2} + \ldots + d_1B^1 + d_0B^0$$

Example

724_8 =>

$4 \times 8^0 = 4$

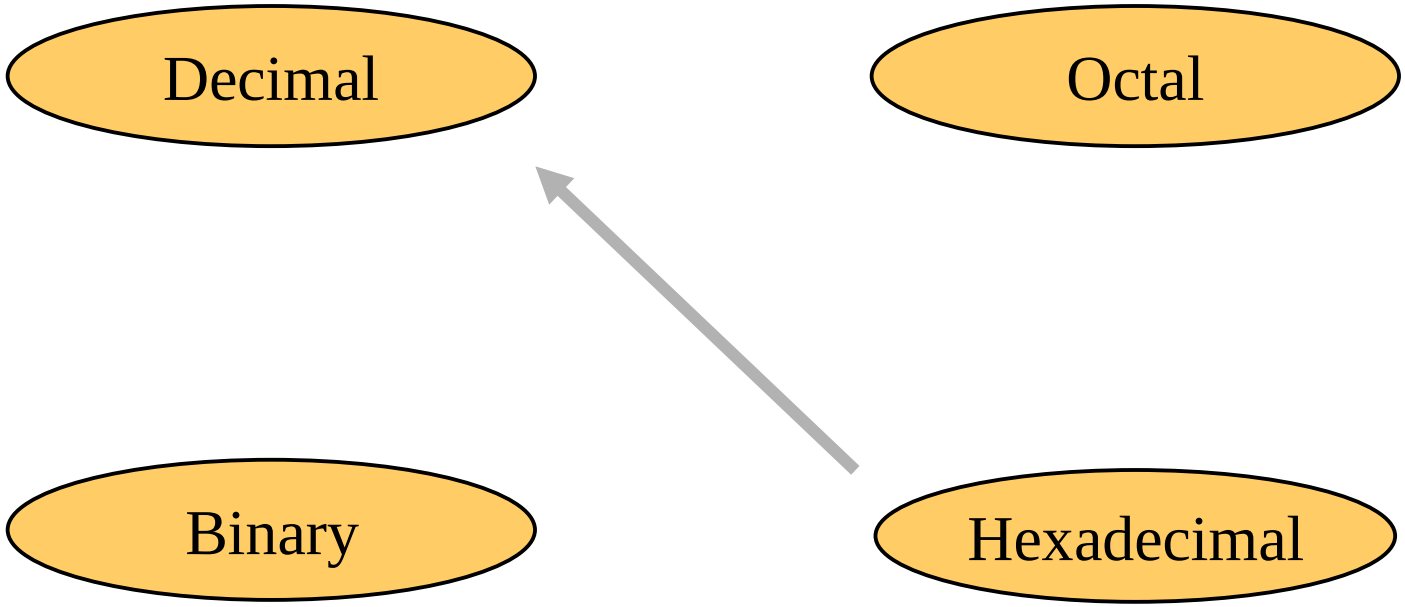
$2 \times 8^1 = 16$

$7 \times 8^2 = 448$

$\overline{\mathbf{468_{10}}}$

$$N_B = \sum_{i=0}^{n-1} d_i \cdot B^i = d_{n-1}B^{n-1} + d_{n-2}B^{n-2} + \ldots + d_1B^1 + d_0B^0$$

Hexadecimal to Decimal



Hexadecimal to Decimal

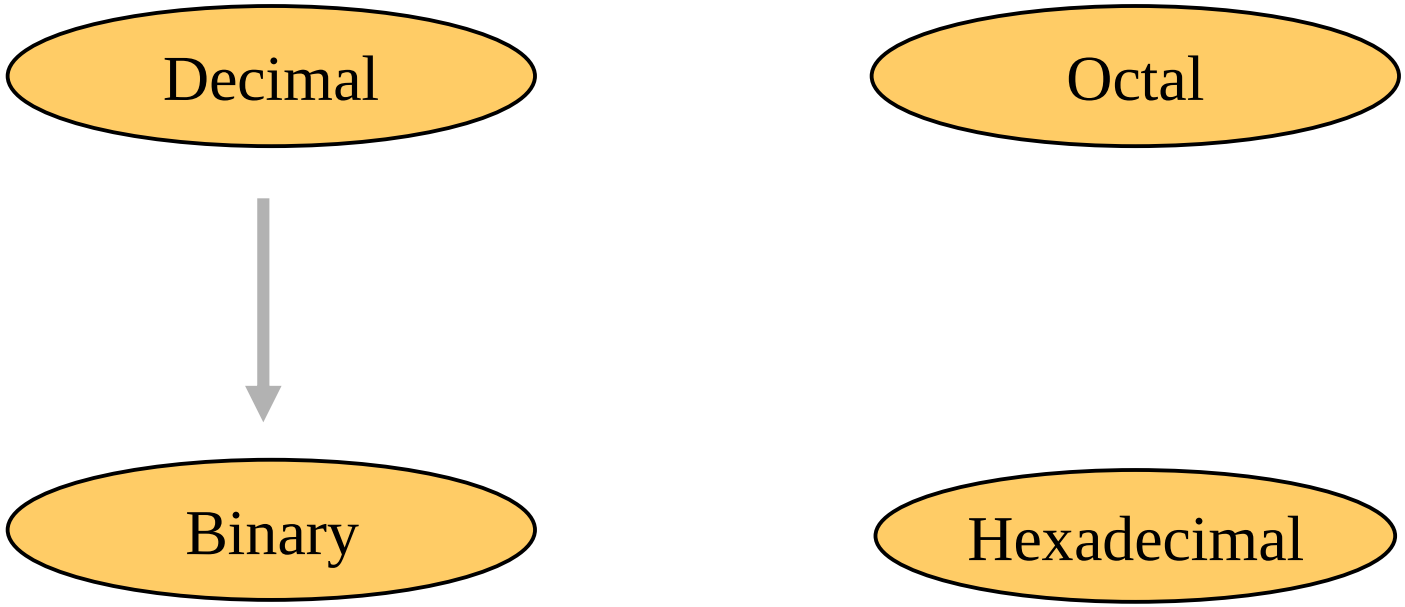
- Technique
  - Multiply each bit by 16^n, where n is the "weight" of the bit
  - The weight is the position of the bit, starting from 0 on the right
  - Add the results

Example

$$4D2_{16} \Rightarrow \begin{array}{rcrcr} 2 & x & 16^0 & = & 2 \\ 13 & x & 16^1 & = & 208 \\ 4 & x & 16^2 & = & 1024 \\ \hline & & & & 1234_{10} \end{array}$$

$$N_B = \sum_{i=0}^{n-1} d_i \cdot B^i = d_{n-1}B^{n-1} + d_{n-2}B^{n-2} + \ldots + d_1 B^1 + d_0 B^0$$

Decimal to Binary

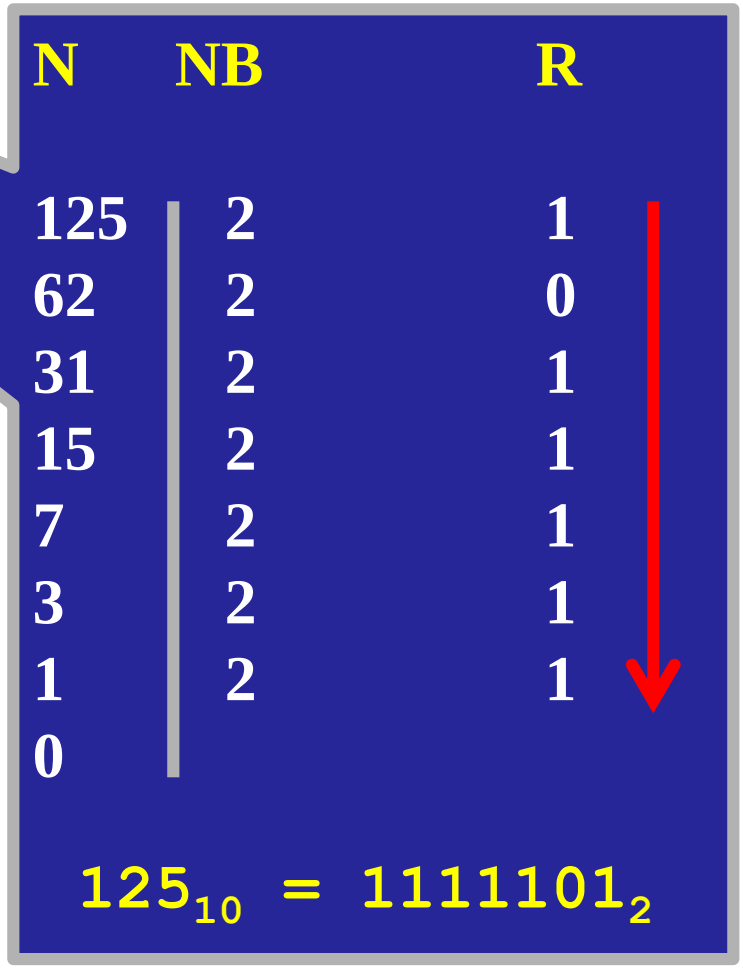


Decimal to Binary

- Technique
  - Divide by two, keep track of the remainder
  - First remainder is bit 0 (LSB, least-significant bit)
  - Second remainder is bit 1
  - Etc.

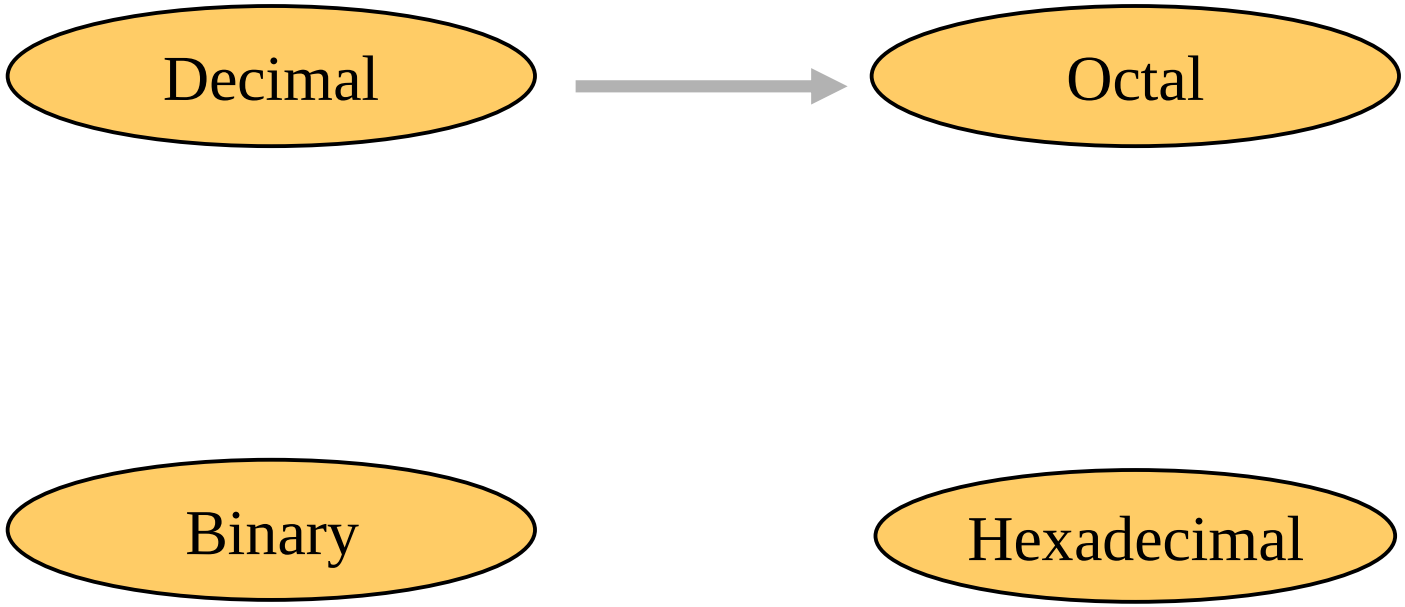
Example

$125_{10} = ?_2$



$\mathbf{125_{10} = 1111101_2}$

Decimal to Octal

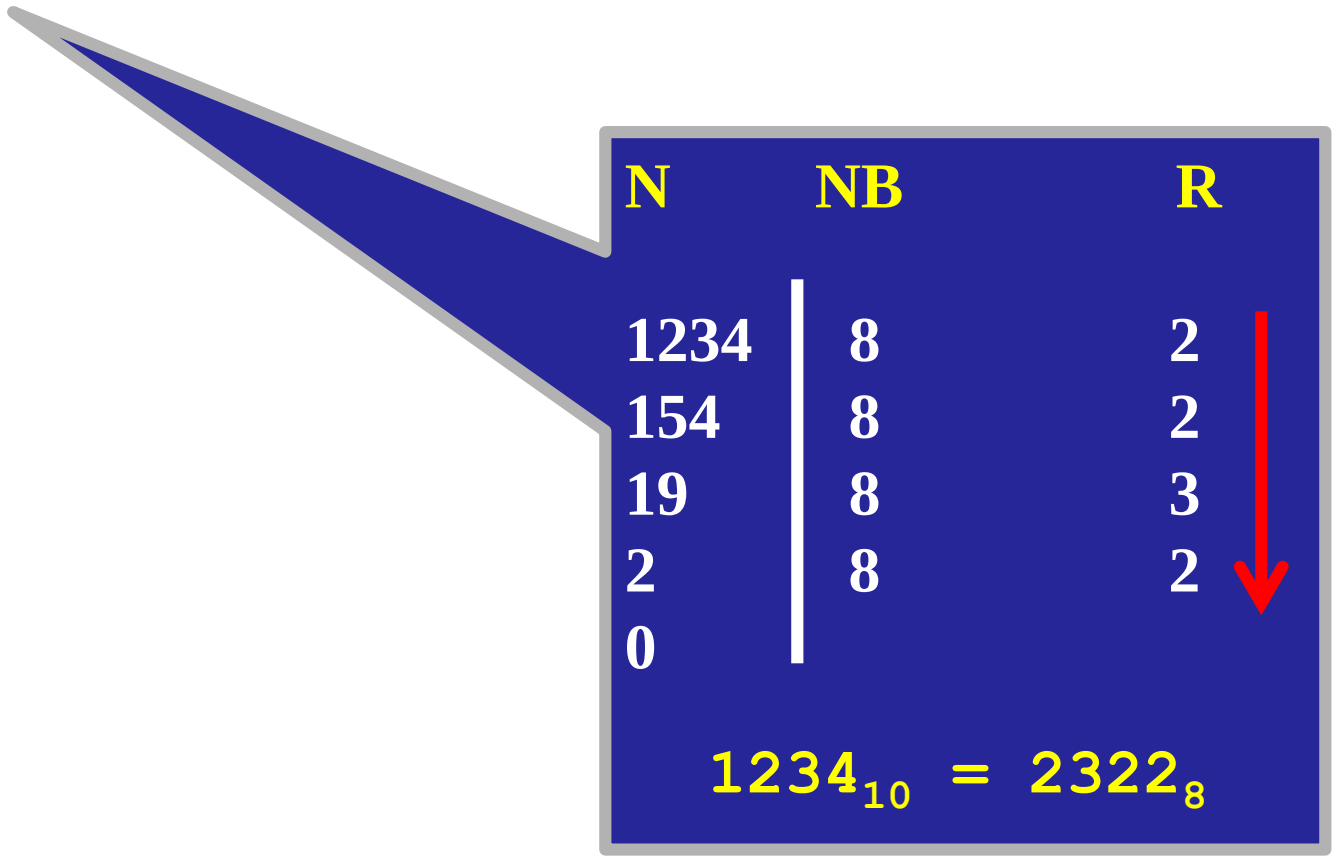


Decimal to Octal

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

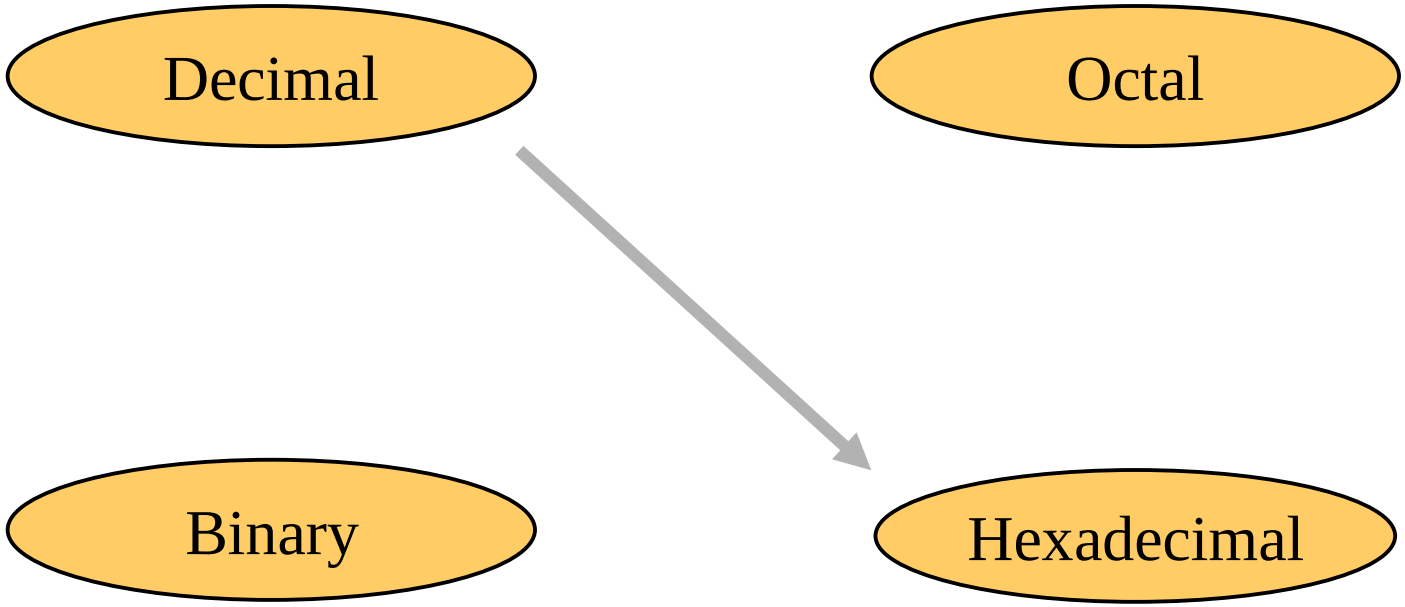
Example

$1234_{10} = ?_{8}$



$1234_{10} = 2322_{8}$

Decimal to Hexadecimal

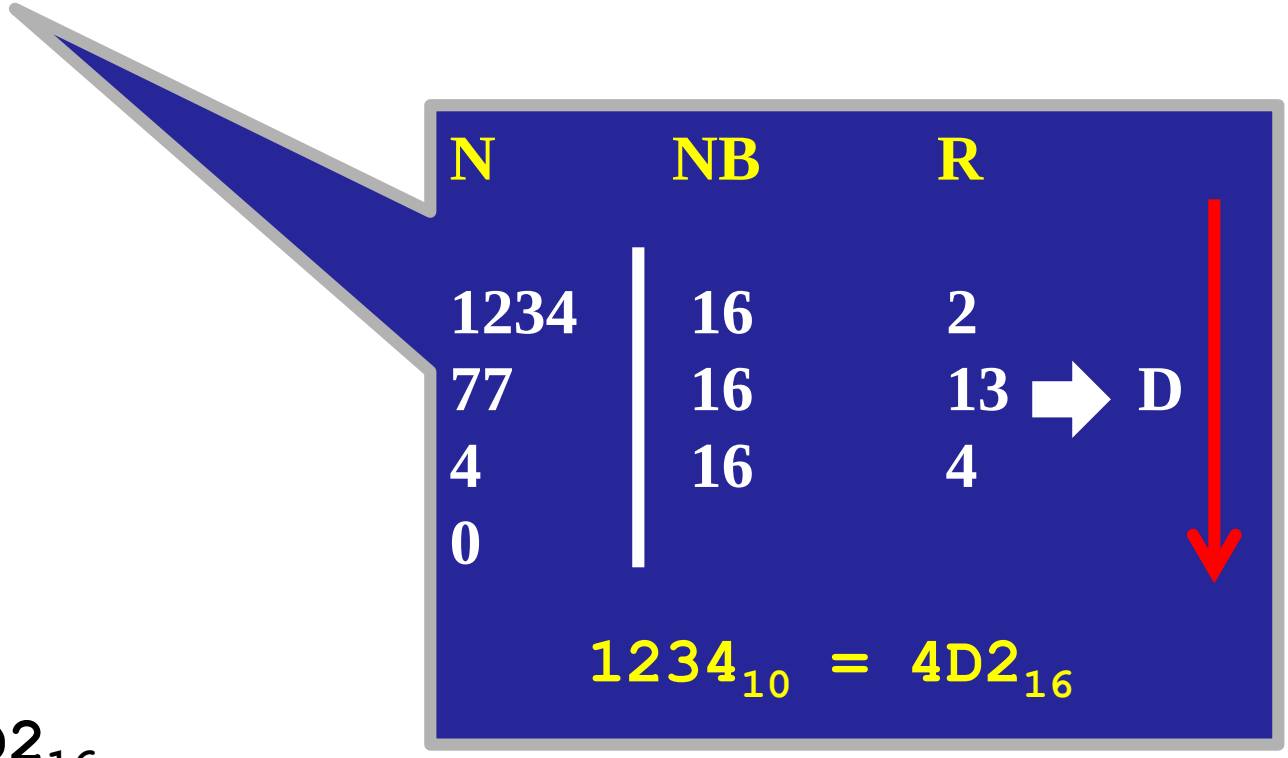


Decimal to Hexadecimal

- Technique
  - Divide by 16
  - Keep track of the remainder

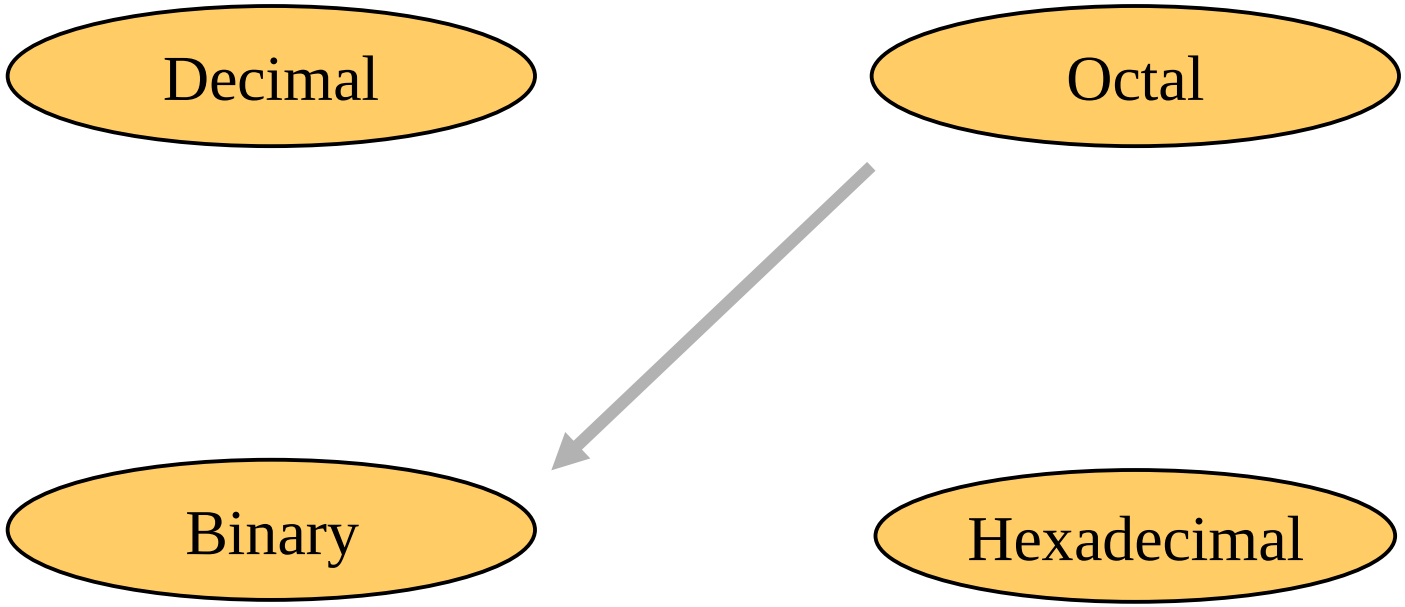
Example

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



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

Octal to Binary



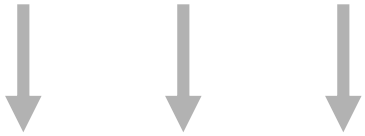
Octal to Binary

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

Example

$703_8 = ?_2$

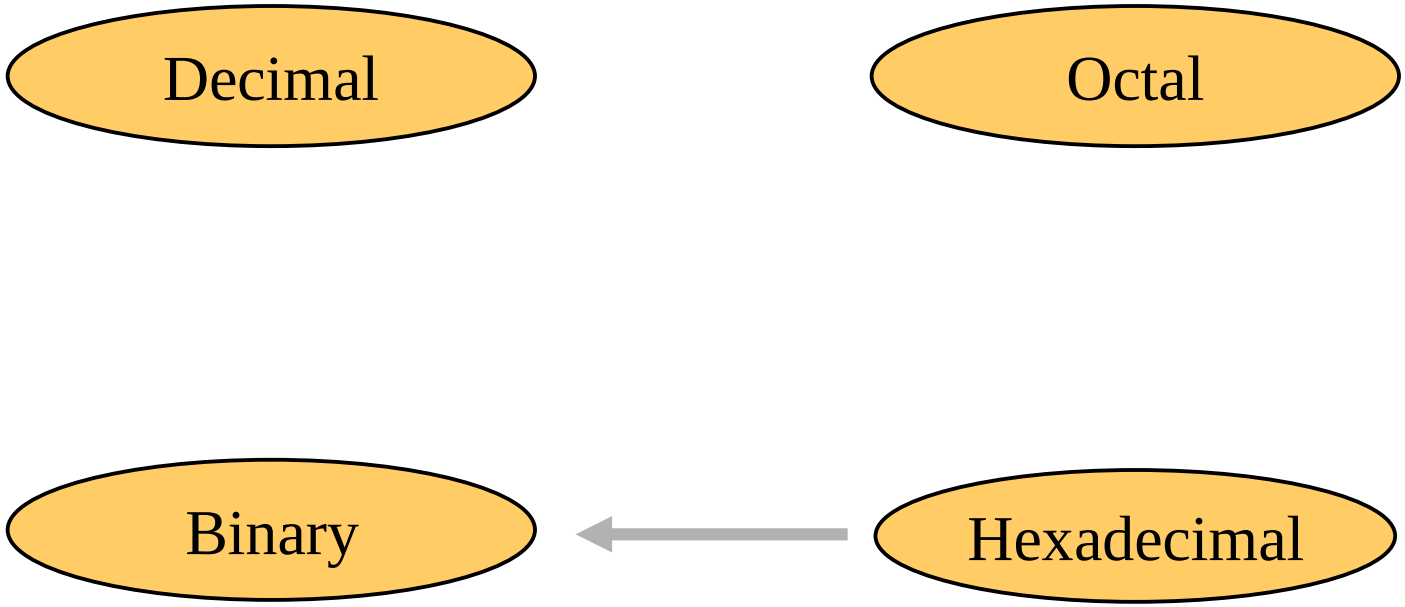
7 0 3



111 000 011

$703_8 = 111000011_2$

Hexadecimal to Binary

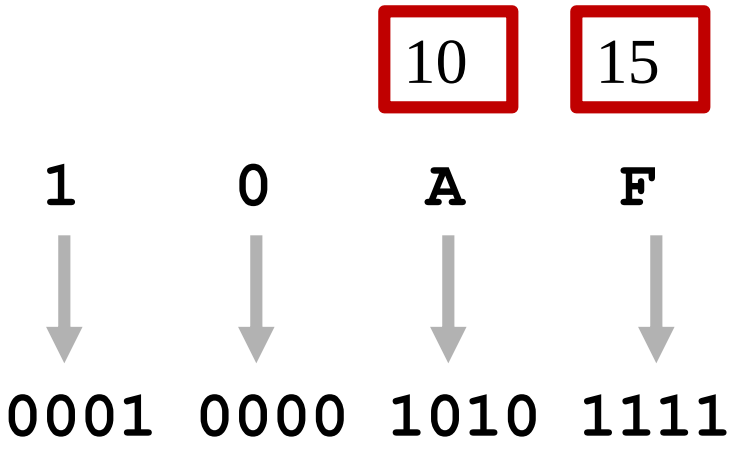


Hexadecimal to Binary

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

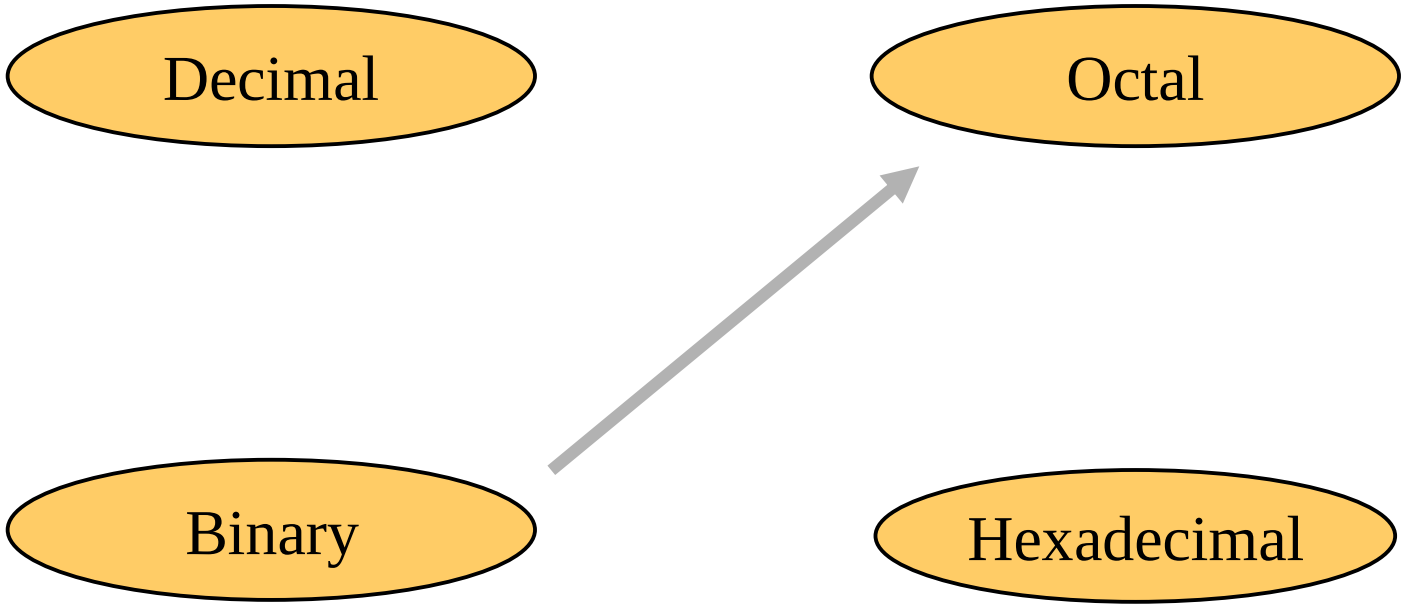
Example

$10AF_{16} = ?_2$



$10AF_{16} = 0001000010101111_2$

Binary to Octal



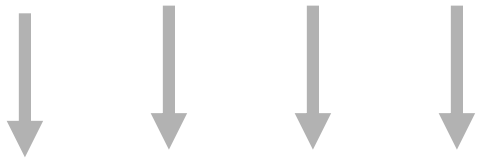
Binary to Octal

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

Example

$1011010111_2 = ?_8$

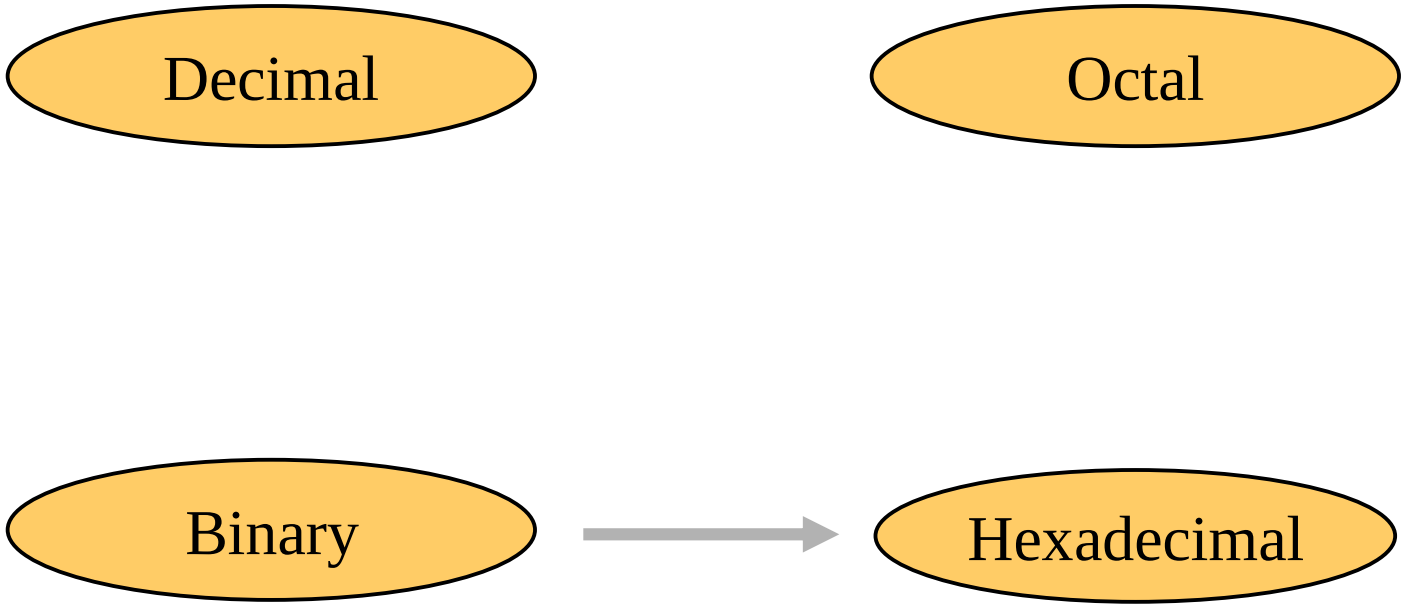
1 011 010 111



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

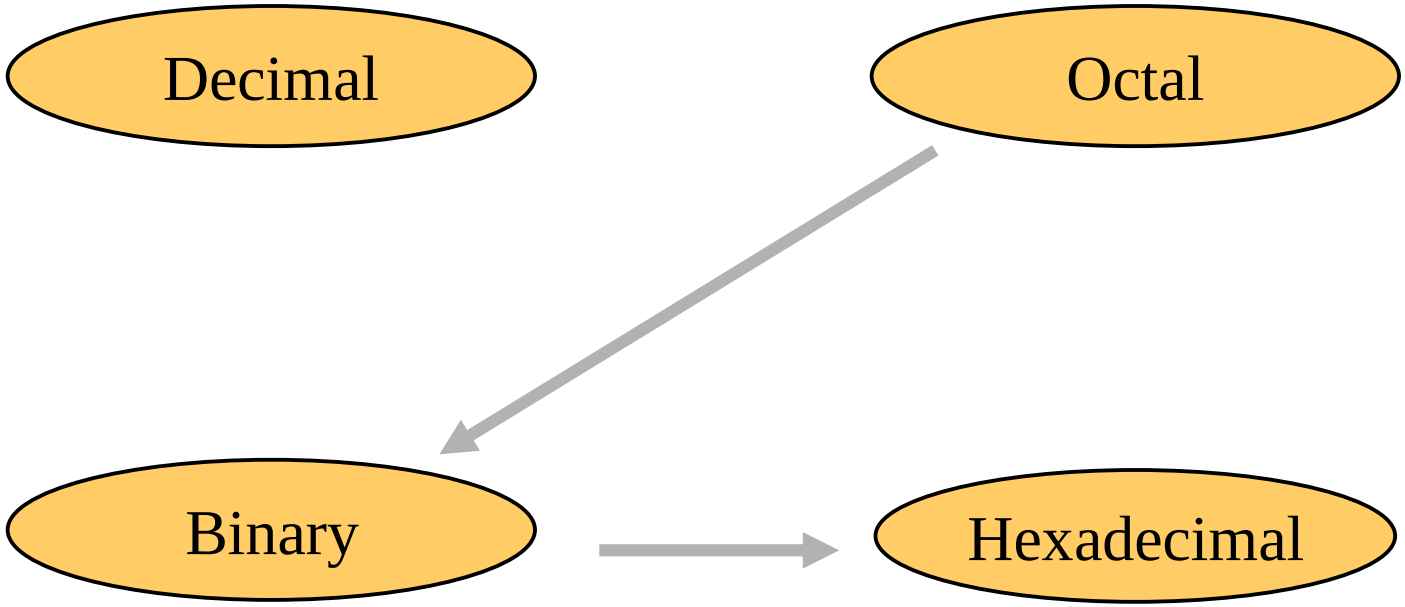
$1010111011_2 = ?_{16}$

10 1011 1011

2 B B

$1010111011_2 = 2BB_{16}$

Octal to Hexadecimal

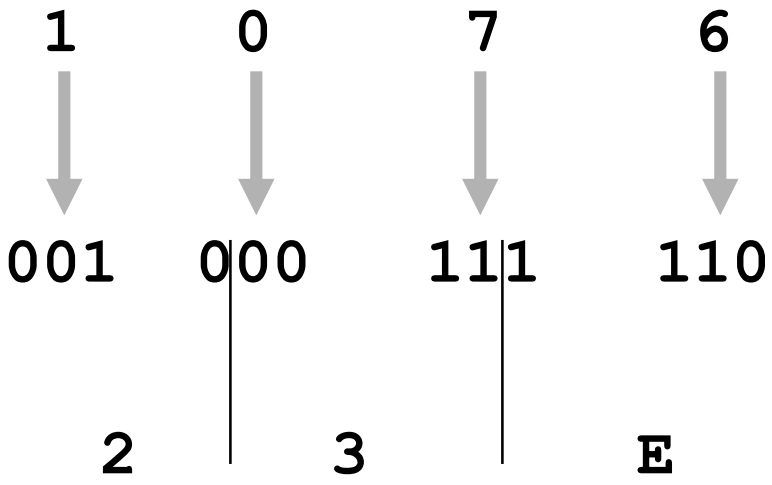


Octal to Hexadecimal

- Technique
  - Use binary as an intermediary

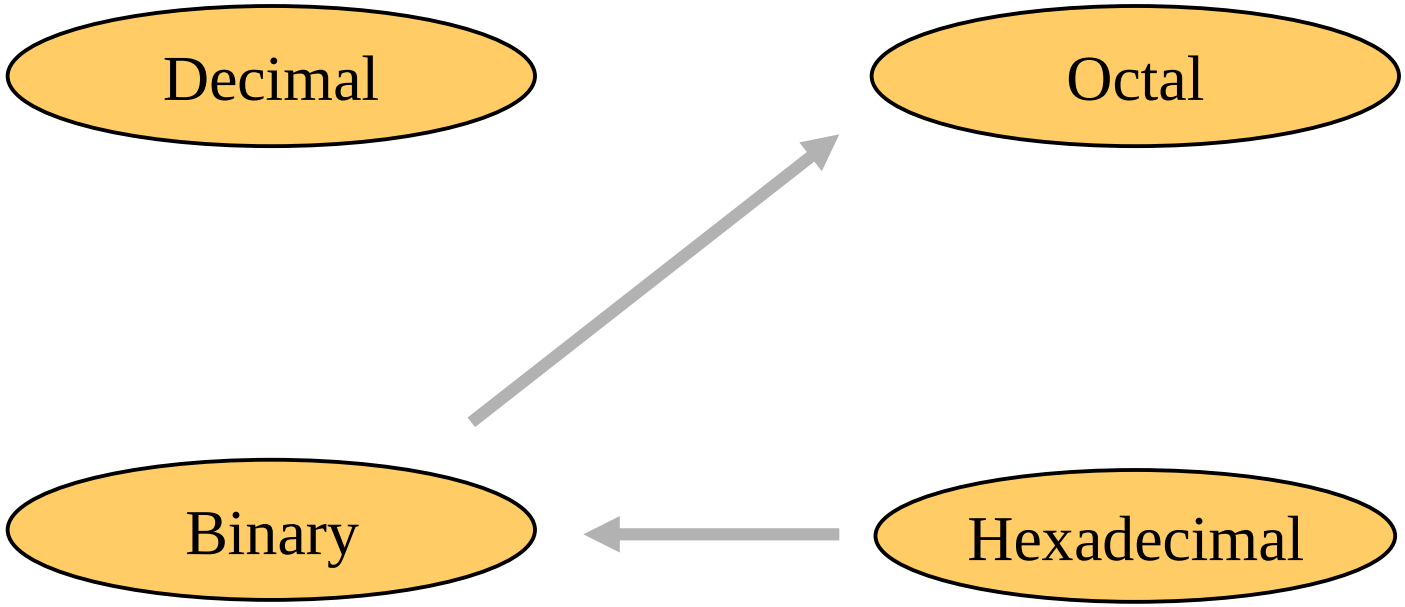
Example

$1076_8 = ?_{16}$



$1076_8 = 23E_{16}$

Hexadecimal to Octal

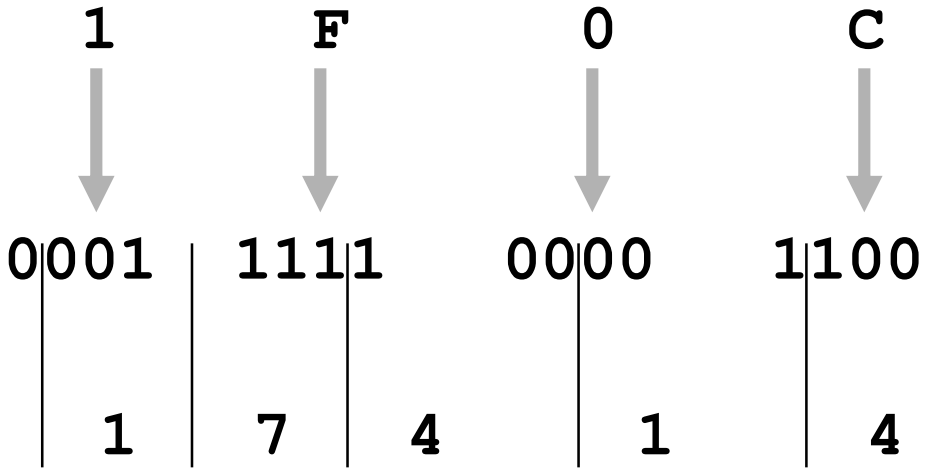


Hexadecimal to Octal

- Technique
  - Use binary as an intermediary

Example

$1F0C_{16} = ?_{8}$



$1F0C_{16} = 17414_{8}$