INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

CSN-101 (Introduction to Computer Science and Engineering)

Lecture 15: Number Systems

Dr. Sudip Roy

Assistant Professor

Department of Computer Science and Engineering

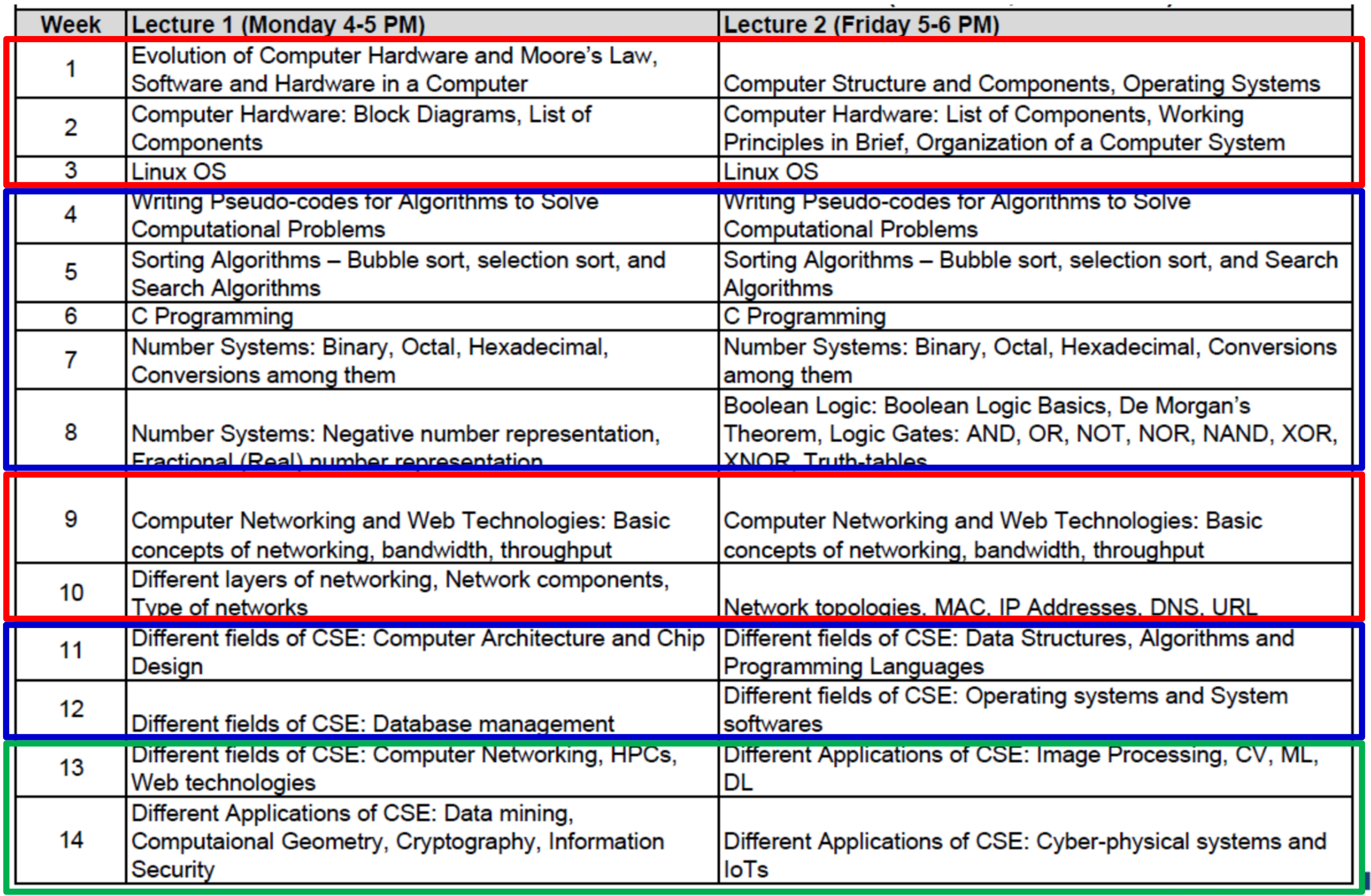
Piazza Class Room: https://piazza.com/iitr.ac.in/fall2019/csn101

[Access Code: csn101@2019]

Moodle Submission Site: https://moodle.iitr.ac.in/course/view.php?id=45

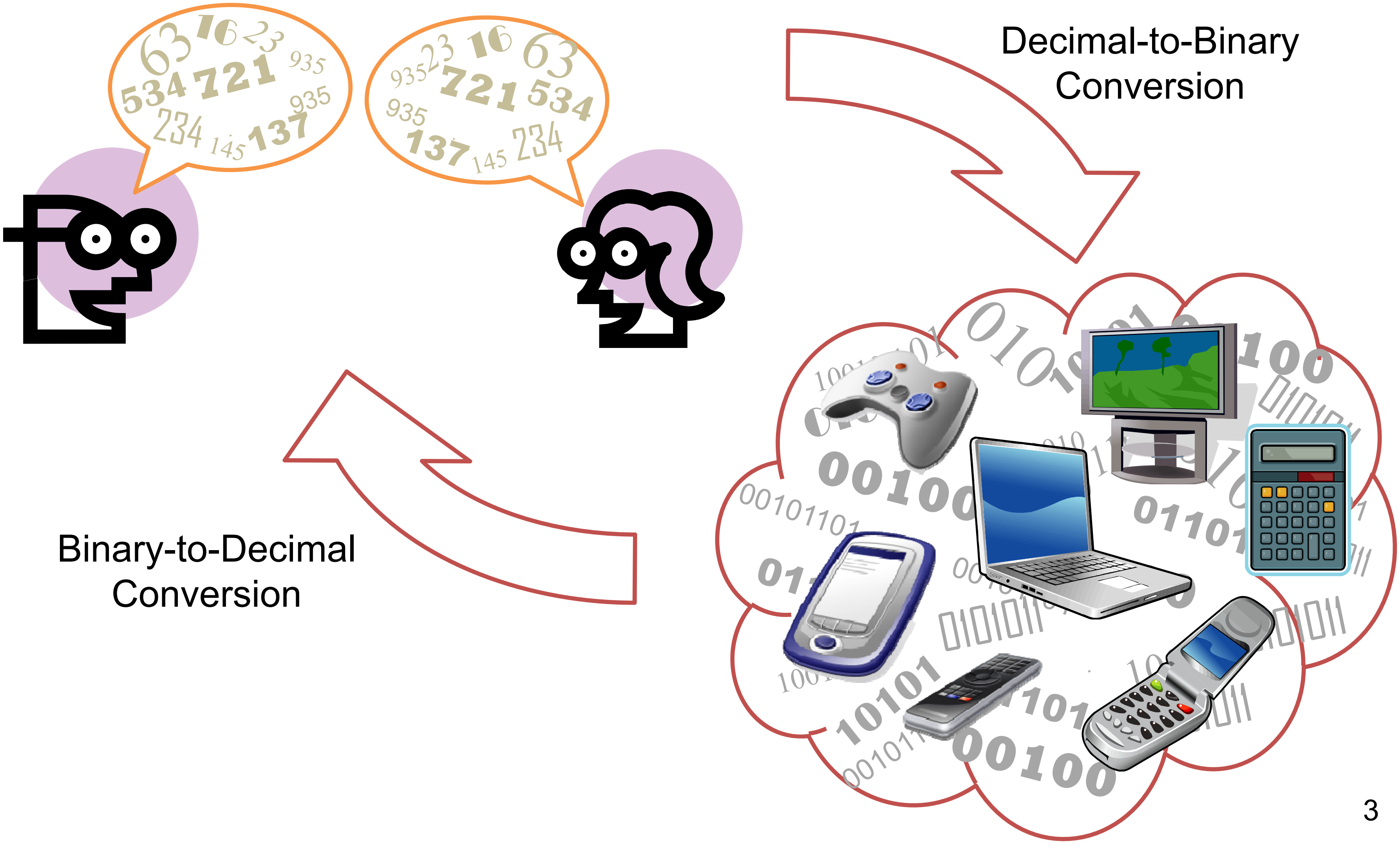
[Enrollment Key: csn101@2019]

Plan for Lecture Classes in CSN-101 (Autumn, 2019-2020)



Week	Lecture 1 (Monday 4-5 PM)	Lecture 2 (Friday 5-6 PM)	
1	Evolution of Computer Hardware and Moore's Law, Software and Hardware in a Computer	Computer Structure and Components, Operating Systems	MTE
2	Computer Hardware: Block Diagrams, List of Components	Computer Hardware: List of Components, Working Principles in Brief, Organization of a Computer System	
3	Linux OS	Linux OS	
4	Writing Pseudo-codes for Algorithms to Solve Computational Problems	Writing Pseudo-codes for Algorithms to Solve Computational Problems	ETE
5	Sorting Algorithms – Bubble sort, selection sort, and Search Algorithms	Sorting Algorithms – Bubble sort, selection sort, and Search Algorithms	
6	C Programming	C Programming	
7	Number Systems: Binary, Octal, Hexadecimal, Conversions among them	Number Systems: Binary, Octal, Hexadecimal, Conversions among them	
8	Number Systems: Negative number representation, Fractional (Real) number representation	Boolean Logic: Boolean Logic Basics, De Morgan's Theorem, Logic Gates: AND, OR, NOT, NOR, NAND, XOR, XNOR, Truth-tables	
9	Computer Networking and Web Technologies: Basic concepts of networking, bandwidth, throughput	Computer Networking and Web Technologies: Basic concepts of networking, bandwidth, throughput	MTE
10	Different layers of networking, Network components, Type of networks	Network topologies, MAC, IP Addresses, DNS, URL	
11	Different fields of CSE: Computer Architecture and Chip Design	Different fields of CSE: Data Structures, Algorithms and Programming Languages	ETE
12	Different fields of CSE: Database management	Different fields of CSE: Operating systems and System softwares	
13	Different fields of CSE: Computer Networking, HPCs, Web technologies	Different Applications of CSE: Image Processing, CV, ML, DL	Term Project
14	Different Applications of CSE: Data mining, Computaional Geometry, Cryptography, Information Security	Different Applications of CSE: Cyber-physical systems and IoTs	

Bridging the Digital Divide



Decimal-to-Binary Conversion

Binary-to-Decimal Conversion

Decimal –to– Binary Conversion

The Process : *Successive Division*

a) Divide the *Decimal Number* by 2; the remainder is the LSB of *Binary Number* .

b) If the quotation is zero, the conversion is complete; else repeat step (a) using the quotation as the Decimal Number. The new remainder is the next most significant bit of the *Binary Number.*

Example:

Convert the decimal number 6_{10} into its binary equivalent.

$2\overset{3}{\overline{)\,6}}$ $r = 0 \leftarrow$ Least Significant Bit

$2\overset{1}{\overline{)\,3}}$ $r = 1$

$2\overset{0}{\overline{)\,1}}$ $r = 1 \leftarrow$ Most Significant Bit

$\therefore\ 6_{10} = 110_2$

Dec → Binary : Example #1

Example:

Convert the decimal number 26_{10} into its binary equivalent.

Dec → Binary : Example #1

Example:

Convert the decimal number 26_{10} into its binary equivalent.

Solution:

$$\begin{array}{r} 13 \\ 2\overline{)\,26} \end{array} \quad r = 0 \leftarrow \text{LSB}$$

$$\begin{array}{r} 6 \\ 2\overline{)\,13} \end{array} \quad r = 1$$

$$\begin{array}{r} 3 \\ 2\overline{)\,6} \end{array} \quad r = 0$$

$$\begin{array}{r} 1 \\ 2\overline{)\,3} \end{array} \quad r = 1$$

$$\begin{array}{r} 0 \\ 2\overline{)\,1} \end{array} \quad r = 1 \leftarrow \text{MSB}$$

$\therefore\ 26_{10} = 11010_2$

Dec → Binary : Example #2

Example:

Convert the decimal number 41_{10} into its binary equivalent.

Dec → Binary : Example #2

Example:

Convert the decimal number 41_{10} into its binary equivalent.

Solution:

$$\begin{array}{r}20\\2\,\overline{)\,41}\end{array}\quad r=1\leftarrow \text{LSB}$$

$$\begin{array}{r}10\\2\,\overline{)\,20}\end{array}\quad r=0$$

$$\begin{array}{r}5\\2\,\overline{)\,10}\end{array}\quad r=0$$

$$\begin{array}{r}2\\2\,\overline{)\,5}\end{array}\quad r=1$$

$$\begin{array}{r}1\\2\,\overline{)\,2}\end{array}\quad r=0$$

$$\begin{array}{r}0\\2\,\overline{)\,1}\end{array}\quad r=1\leftarrow \text{MSB}$$

$$\boxed{\therefore\ 41_{10} = 101001_2}$$

Dec → Binary : More Examples

a) 13_{10} = ?

b) 22_{10} = ?

c) 43_{10} = ?

d) 158_{10} = ?

Dec → Binary : More Examples

a) 13_{10} = ? $1\,1\,0\,1_{2}$

b) 22_{10} = ? $1\,0\,1\,1\,0_{2}$

c) 43_{10} = ? $1\,0\,1\,0\,1\,1_{2}$

d) 158_{10} = ? $1\,0\,0\,1\,1\,1\,1\,0_{2}$

Binary –to– Decimal Process

The Process : *Weighted Multiplication*

a) Multiply each bit of the *Binary Number* by it corresponding bit-weighting factor (i.e. Bit-0→2^0=1; Bit-1→2^1=2; Bit-2→2^2=4; etc).

b) Sum up all the products in step (a) to get the *Decimal Number*.

Example:

Convert the decimal number 0110_2 into its decimal equivalent.

0	1	1	0	
2^3	2^2	2^1	2^0	Bit-Weighting Factors
8	4	2	1	

$$0 + 4 + 2 + 0 = 6_{10}$$

$$\boxed{\therefore\ 0110_2 = 6_{10}}$$

Binary → Dec : Example #1

Example:

Convert the binary number 10010_2 into its decimal equivalent.

Binary → Dec : Example #1

Example:

Convert the binary number 10010_2 into its decimal equivalent.

Solution:

1	0	0	1	0
2^4	2^3	2^2	2^1	2^0
16	8	4	2	1

$$16 + 0 + 0 + 2 + 0 = 18_{10}$$

$$\boxed{\therefore 10010_2 = 18_{10}}$$

Binary → Dec : Example #2

Example:

Convert the binary number 0110101_2 into its decimal equivalent.

Binary → Dec : Example #2

Example:

Convert the binary number 0110101_2 into its decimal equivalent.

Solution:

0	1	1	0	1	0	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1

$$0 + 32 + 16 + 0 + 4 + 0 + 1 = 53_{10}$$

$$\boxed{\therefore 0110101_2 = 53_{10}}$$

Binary → Dec : More Examples

a) 0110_2 = ?

b) 11010_2 = ?

c) 0110101_2 = ?

d) 11010011_2 = ?

Binary → Dec : More Examples

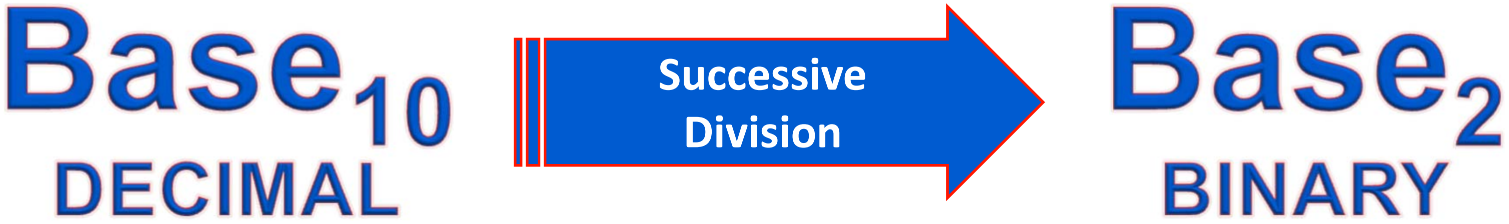
a) 0110_2 = ? 6_{10}

b) 11010_2 = ? 26_{10}

c) 0110101_2 = ? 53_{10}

d) 11010011_2 = ? 211_{10}

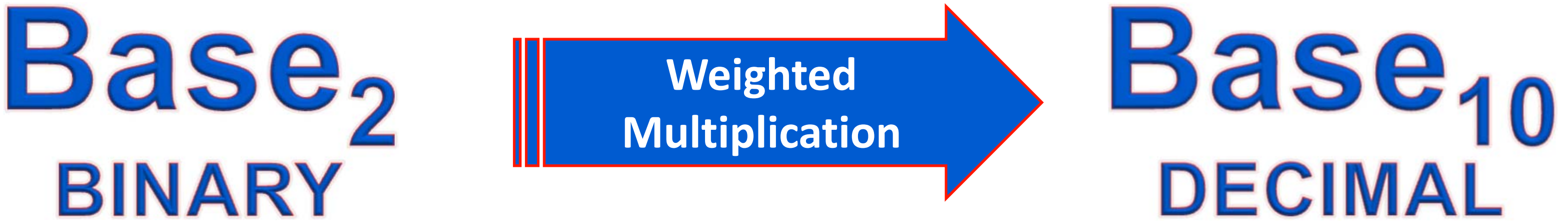
Summary & Review



Base_{10} DECIMAL → Successive Division → Base_{2} BINARY

a) Divide the *Decimal Number* by 2; the remainder is the LSB of *Binary Number* .

b) If the Quotient Zero, the conversion is complete; else repeat step (a) using the Quotient as the Decimal Number. The new remainder is the next most significant bit of the *Binary Number.*



Base_{2} BINARY → Weighted Multiplication → Base_{10} DECIMAL

a) Multiply each bit of the *Binary Number* by it corresponding bit-weighting factor (i.e. Bit-0→2^0=1; Bit-1→2^1=2; Bit-2→2^2=4; etc).

b) Sum up all the products in step (a) to get the *Decimal Number*.

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 (2 of 3):

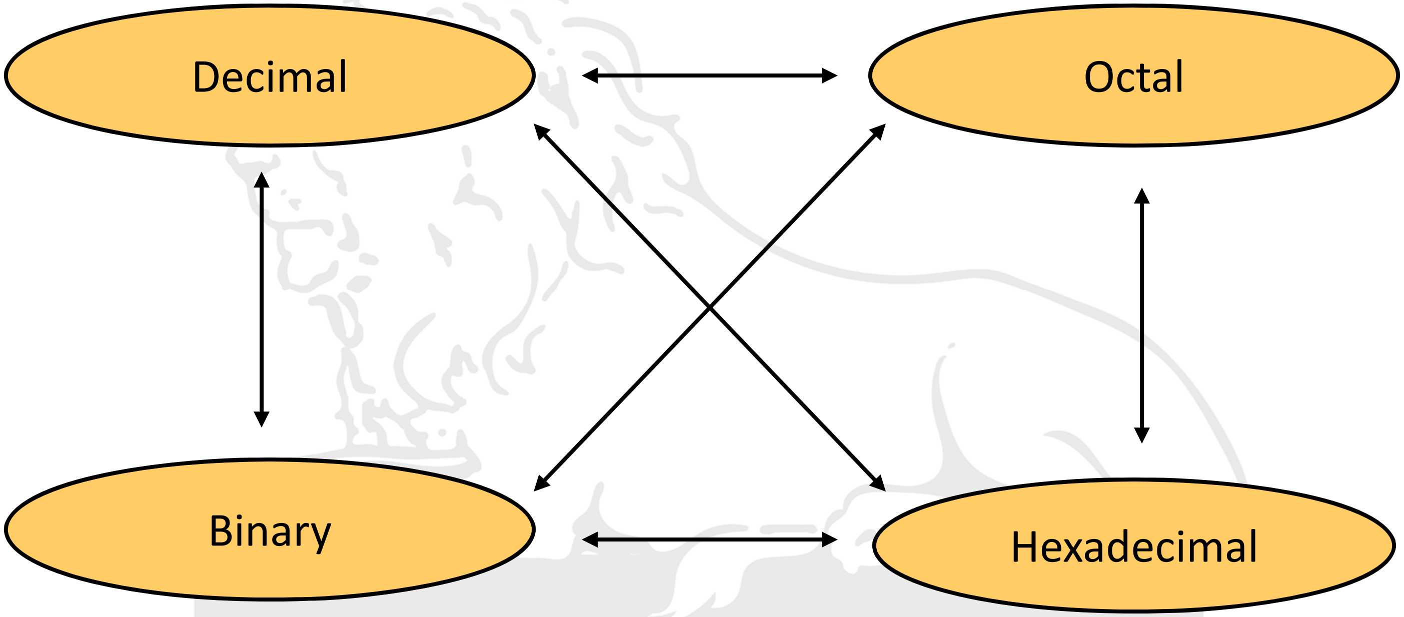
Decimal	Binary	Octal	Hexa-decimal
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

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

Conversion Among Bases:

- The possibilities:

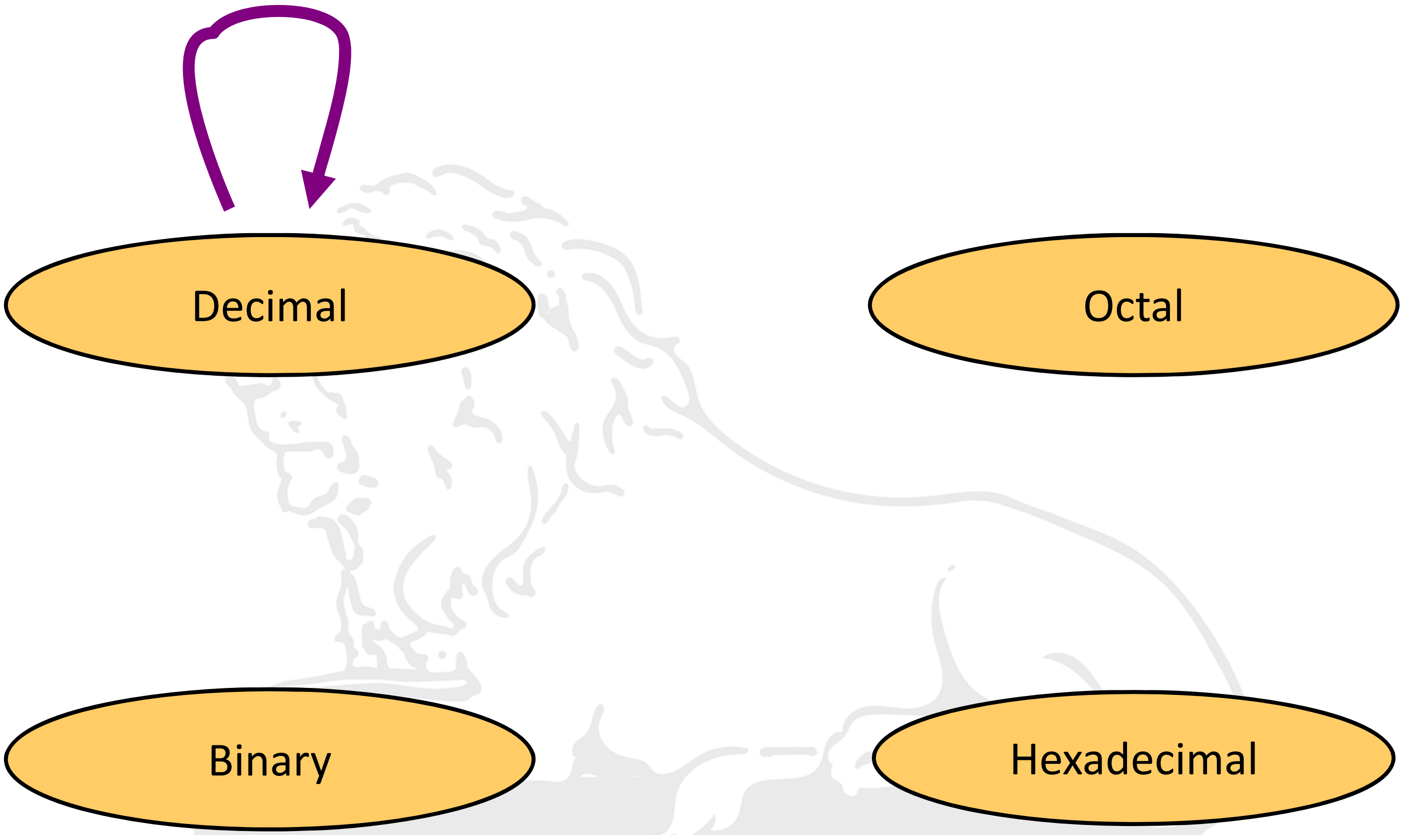


Quick Example:

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

Base

Decimal to Decimal (just for fun):



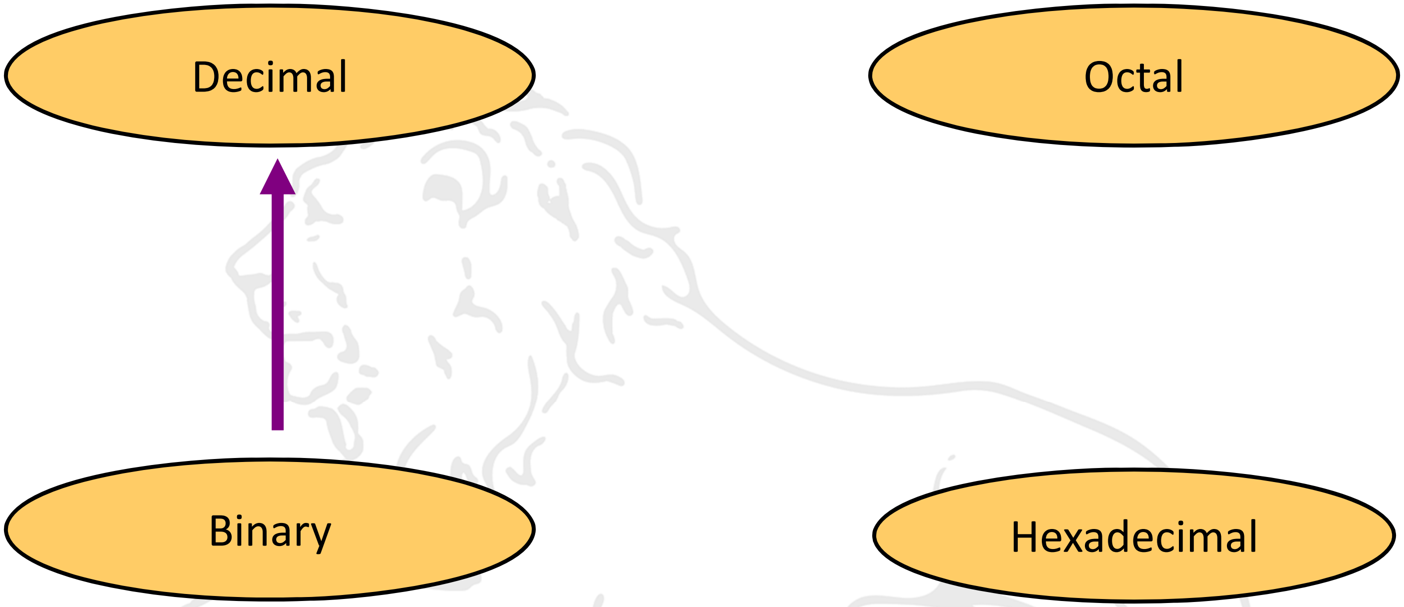
Weight

$125_{10} \Rightarrow$

$$
\begin{aligned}
5 \times 10^0 &= 5 \\
2 \times 10^1 &= 20 \\
1 \times 10^2 &= 100 \\
\hline
& 125
\end{aligned}
$$

Base

Binary to Decimal:



Decimal

Octal

Binary

Hexadecimal

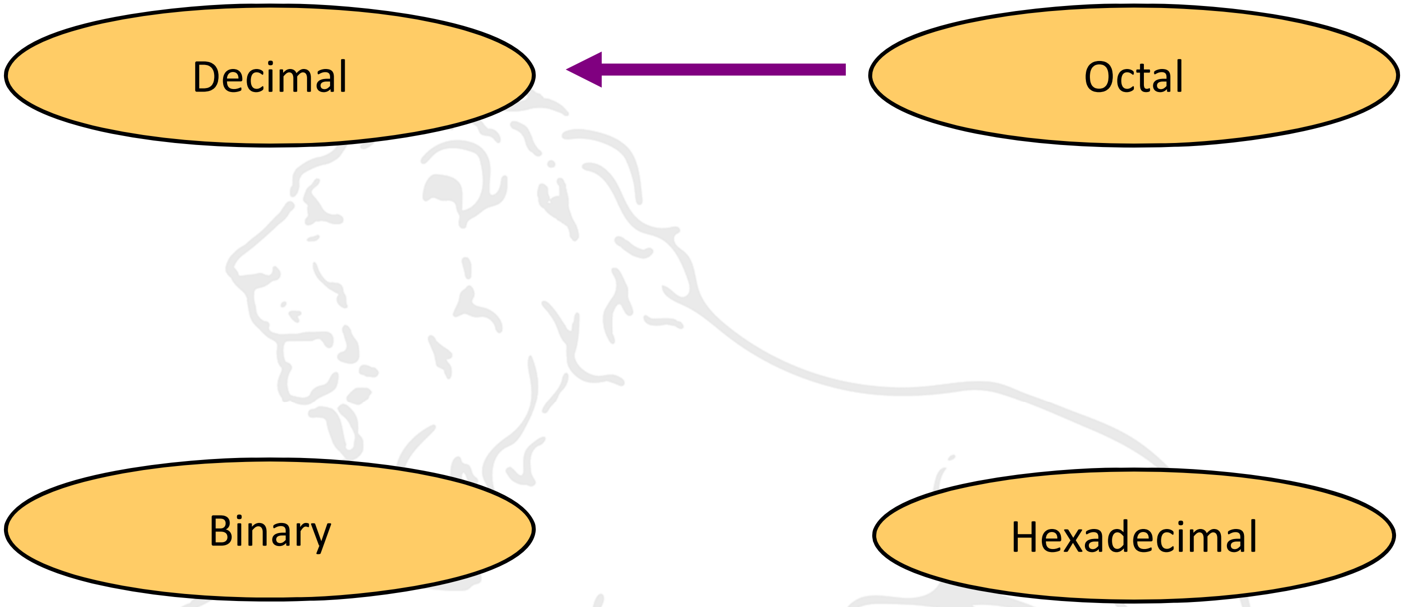
Binary to Decimal:

- 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

Bit “0”

$$
\begin{array}{lccccr}
101011_2 \ => & 1 & \text{x} & 2^0 & = & 1 \\
 & 1 & \text{x} & 2^1 & = & 2 \\
 & 0 & \text{x} & 2^2 & = & 0 \\
 & 1 & \text{x} & 2^3 & = & 8 \\
 & 0 & \text{x} & 2^4 & = & 0 \\
 & 1 & \text{x} & 2^5 & = & \underline{32} \\
 & & & & & 43_{10}
\end{array}
$$

Octal to Decimal:

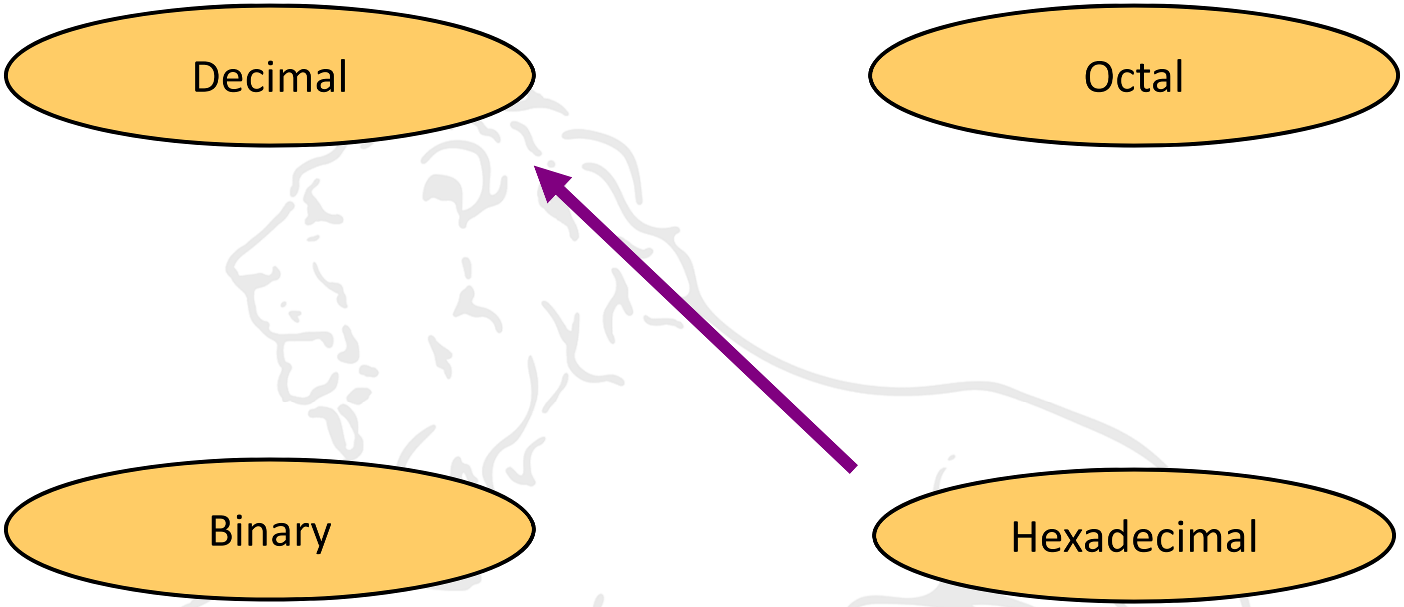


Octal to Decimal:

- 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

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

Hexadecimal to Decimal:

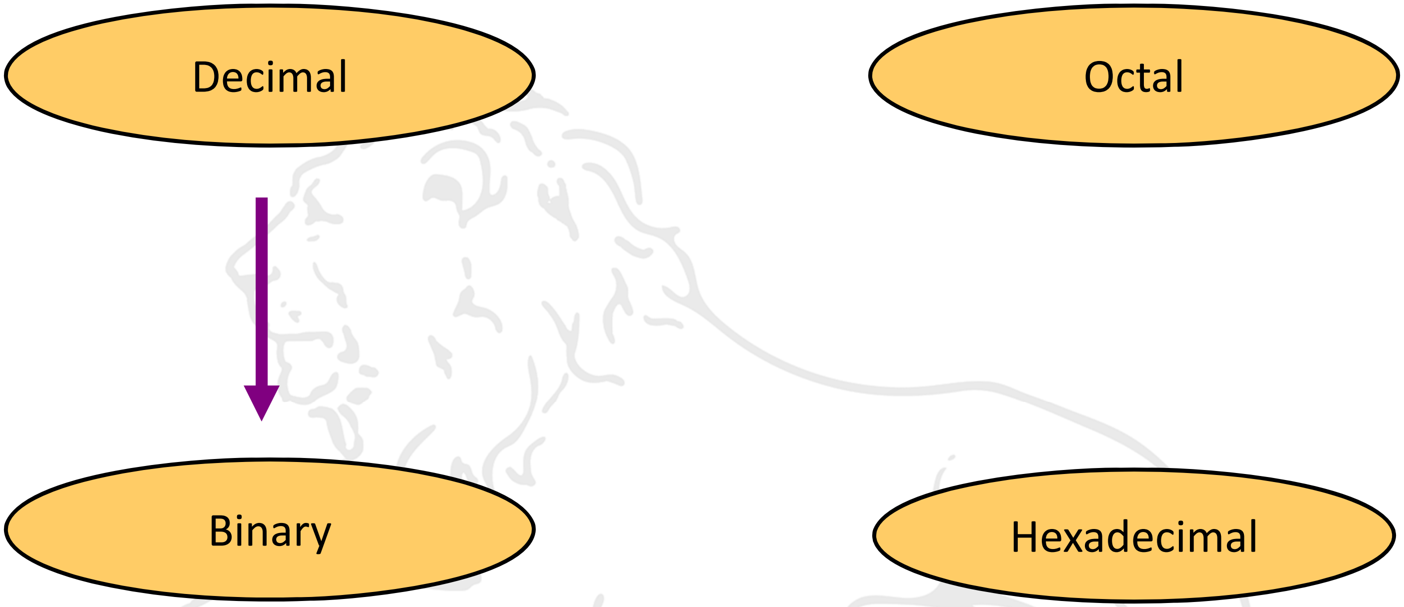


Hexadecimal to Decimal:

- 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

$$
\begin{array}{lllr}
ABC_{16} \Rightarrow & C \times 16^0 = 12 \times 1 & = & 12 \\
 & B \times 16^1 = 11 \times 16 & = & 176 \\
 & A \times 16^2 = 10 \times 256 & = & 2560 \\
\cline{3-4}
 & & & 2748_{10}
\end{array}
$$

Decimal to Binary:



Decimal

Octal

Binary

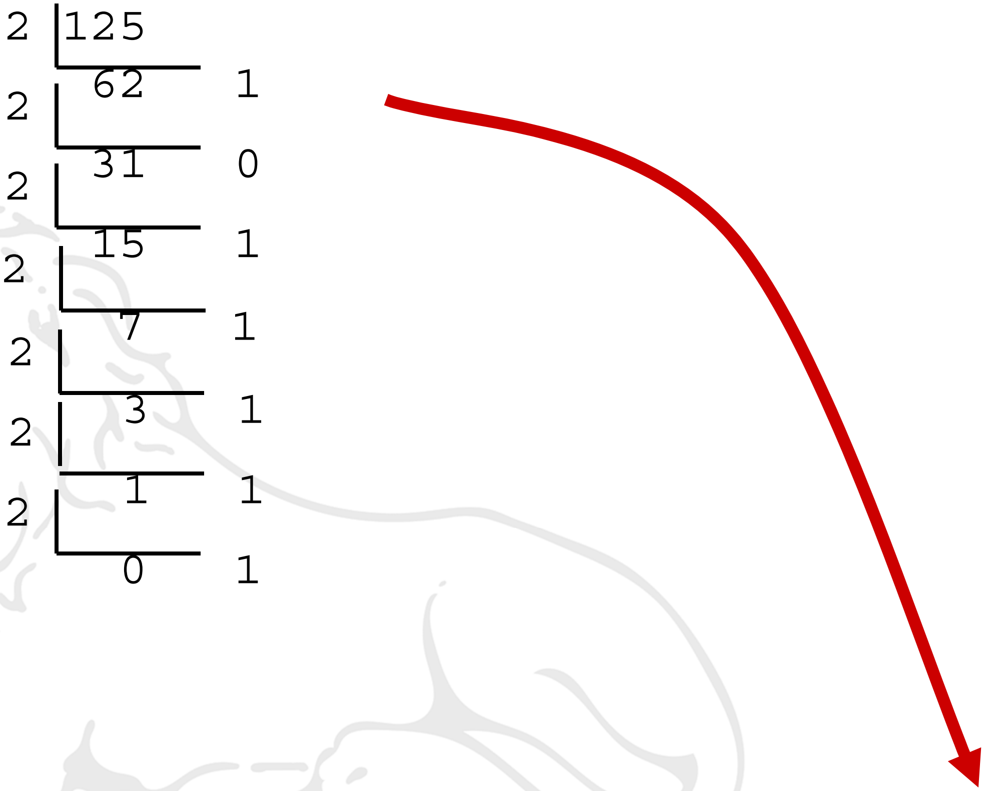
Hexadecimal

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:

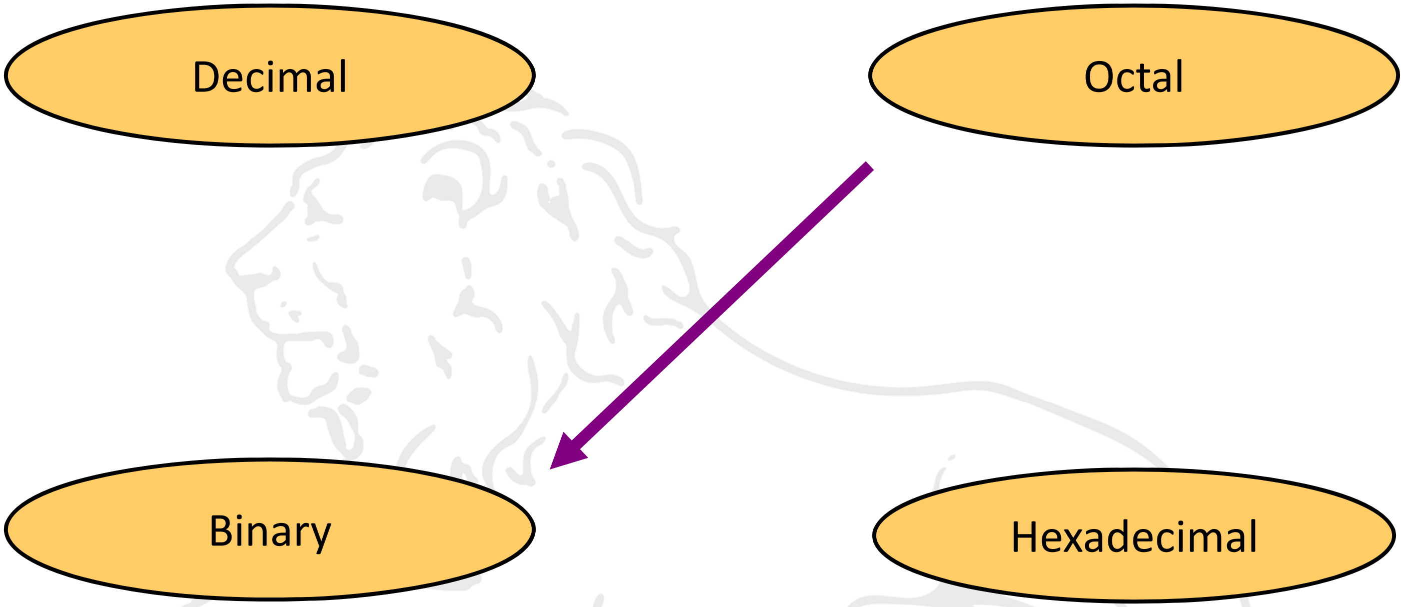
$125_{10} = ?_2$



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

$125_{10} = 1111101_2$

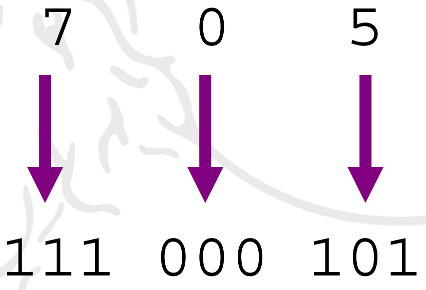
Octal to Binary:



Octal to Binary:

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

$705_8 = ?_2$

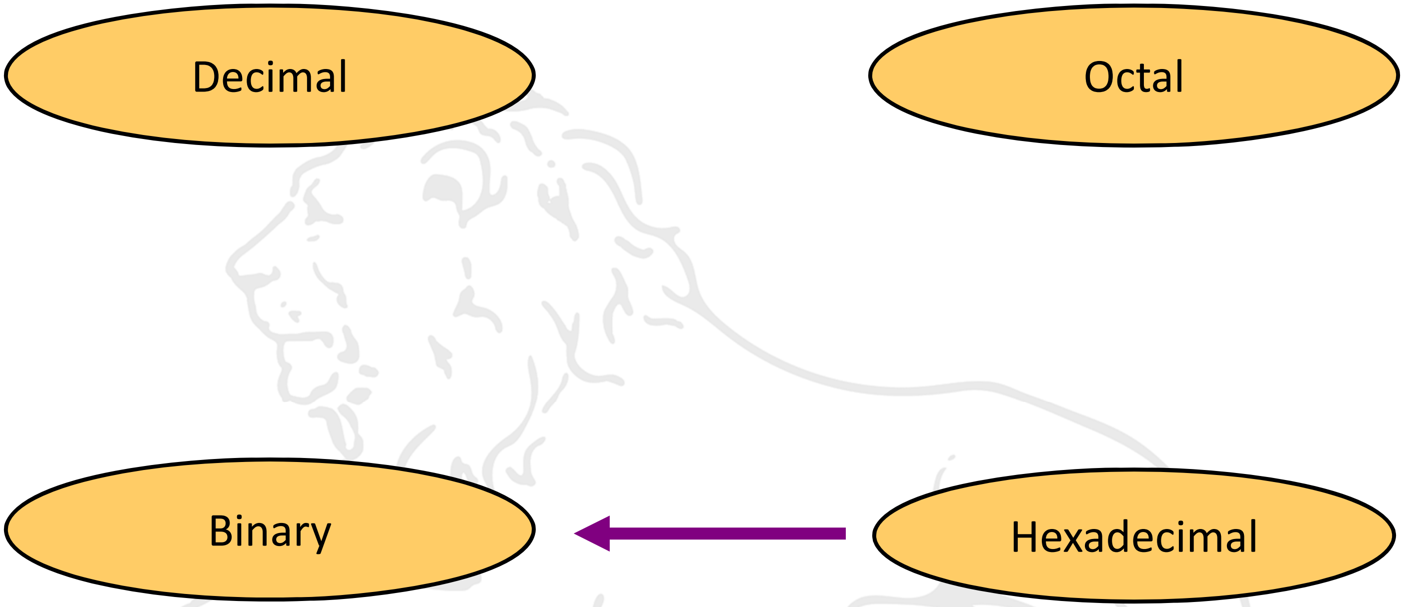


7 0 5

111 000 101

$705_8 = 111000101_2$

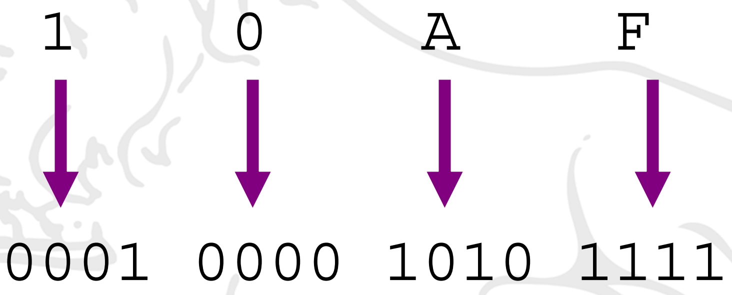
Hexadecimal to Binary:



Hexadecimal to Binary:

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

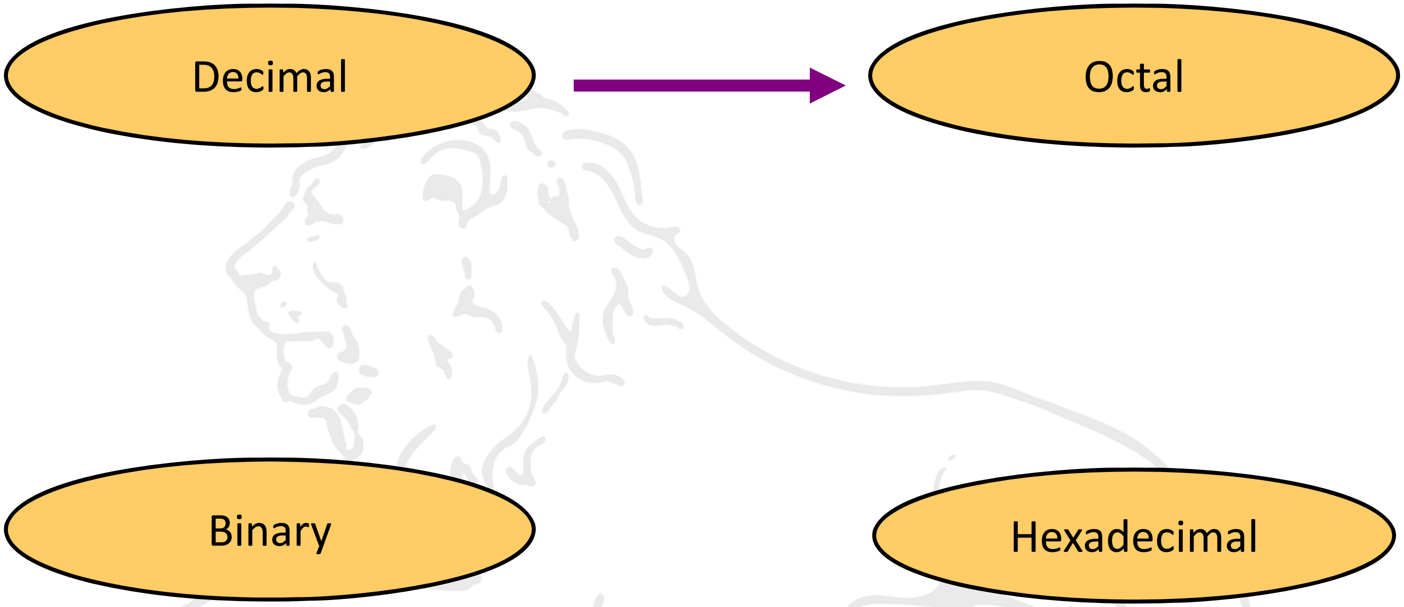
$10AF_{16} = ?_2$


1 0 A F

0001 0000 1010 1111

$10AF_{16} = 0001000010101111_2$

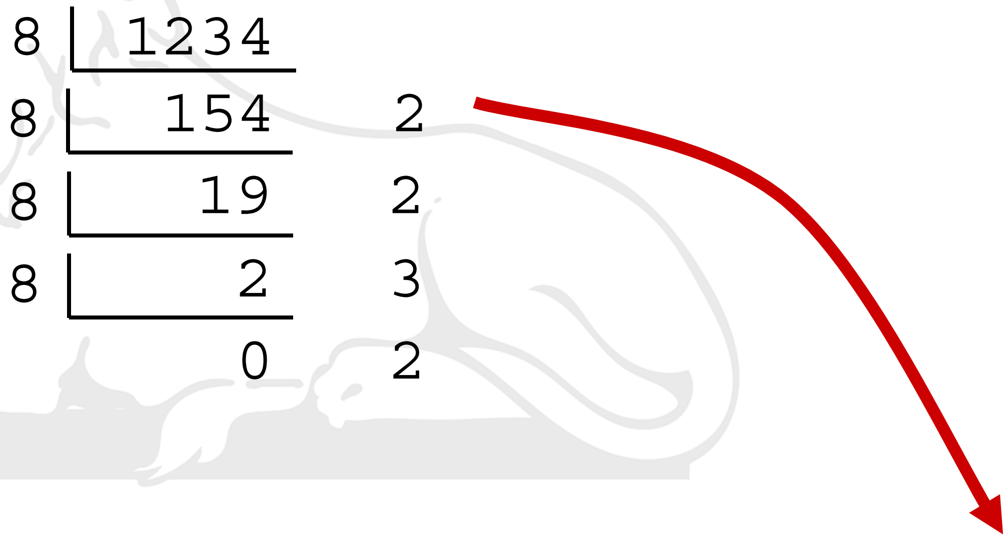
Decimal to Octal:



Decimal to Octal:

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

$1234_{10} = ?_8$



8 | 1234

8 | 154 2

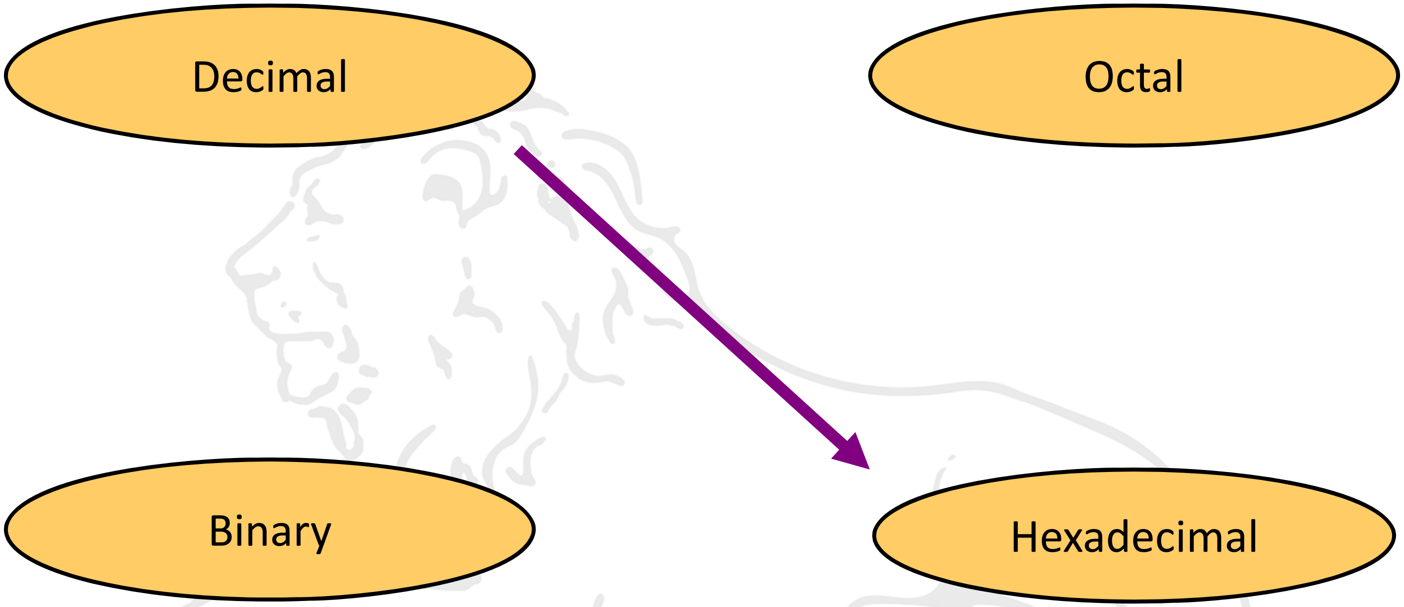
8 | 19 2

8 | 2 3

0 2

$1234_{10} = 2322_8$

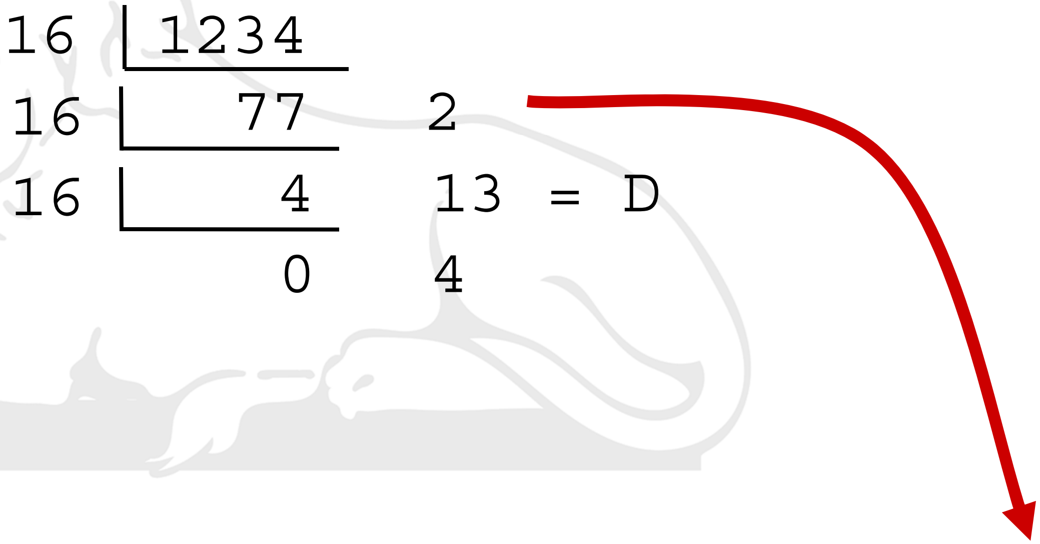
Decimal to Hexadecimal:



Decimal to Hexadecimal:

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

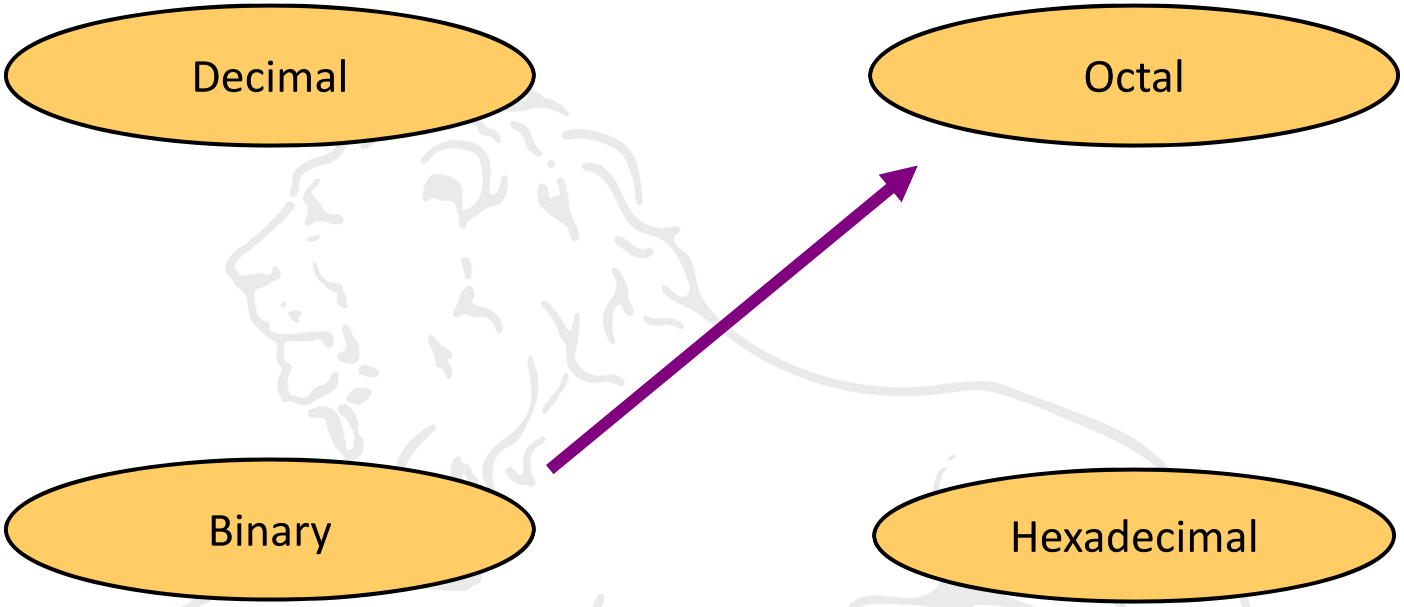
$1234_{10} = ?_{16}$



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

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

Binary to Octal:



Decimal

Octal

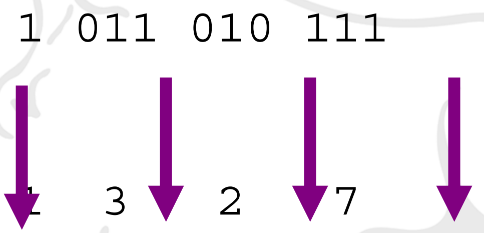
Binary

Hexadecimal

Binary to Octal:

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

$1011010111_2 = ?_8$

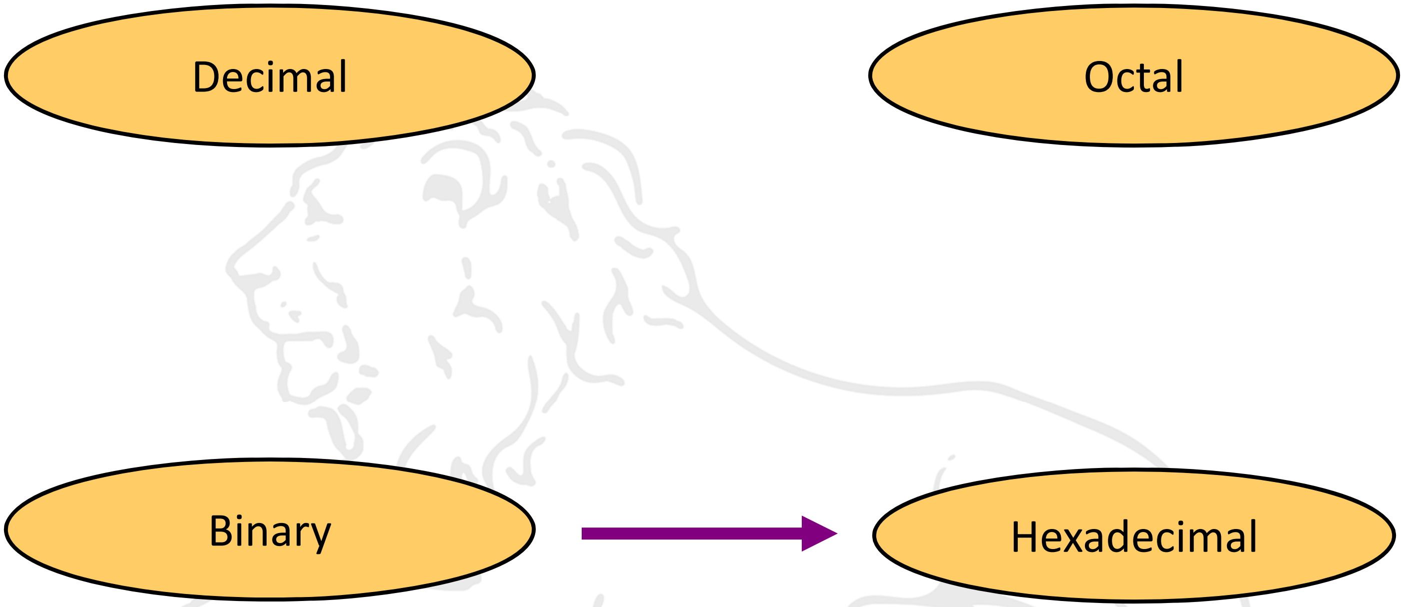


1 011 010 111

1 3 2 7

$1011010111_2 = 1327_8$

Binary to Hexadecimal:



Decimal

Octal

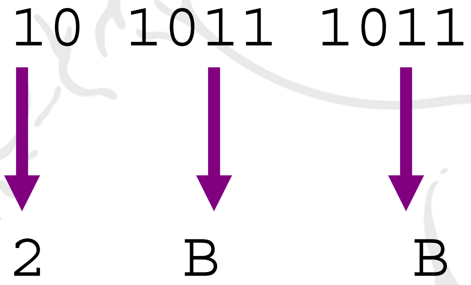
Binary

Hexadecimal

Binary to Hexadecimal:

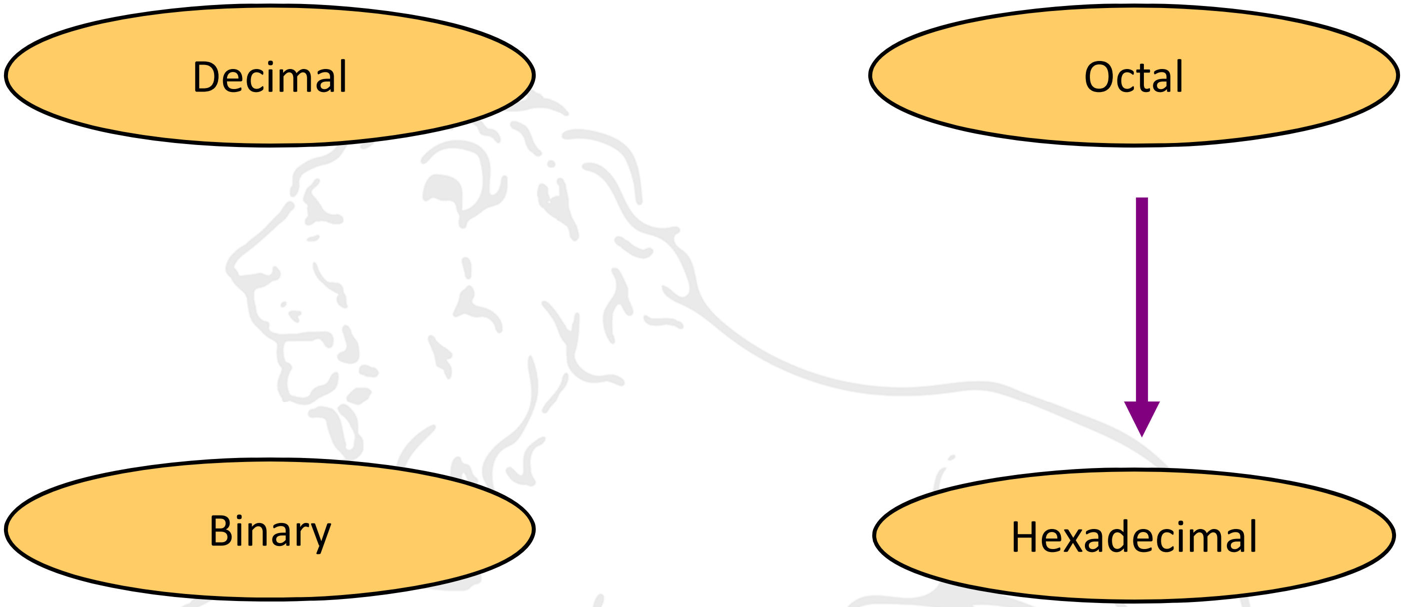
- Technique
  - Group bits in fours, starting on right
  - Convert to hexadecimal digits

$1010111011_2 = ?_{16}$



$1010111011_2 = 2BB_{16}$

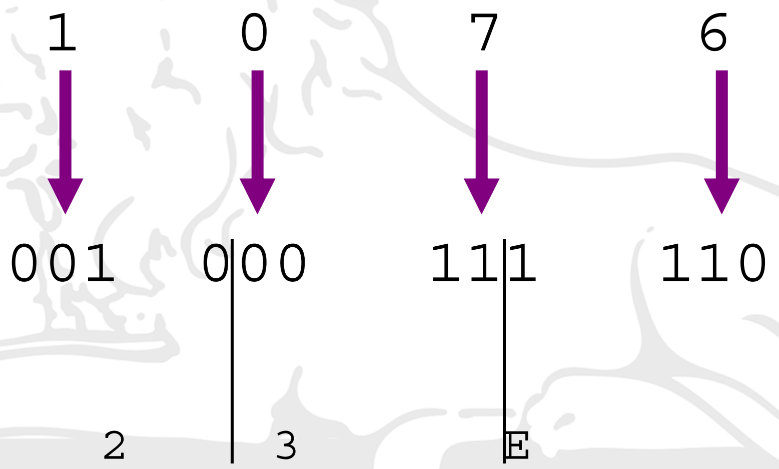
Octal to Hexadecimal:



Octal to Hexadecimal:

- Technique
  - Use binary as an intermediary

$1076_8 = ?_{16}$



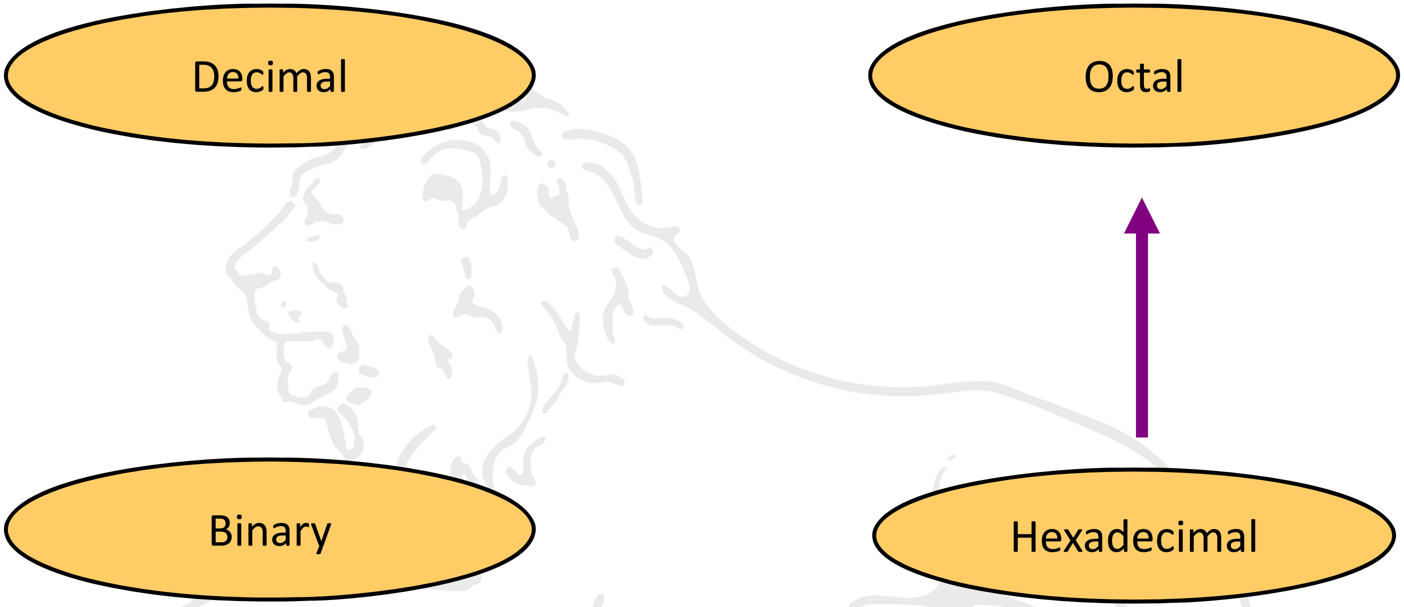
1 0 7 6

001 000 111 110

2 3 E

$1076_8 = 23E_{16}$

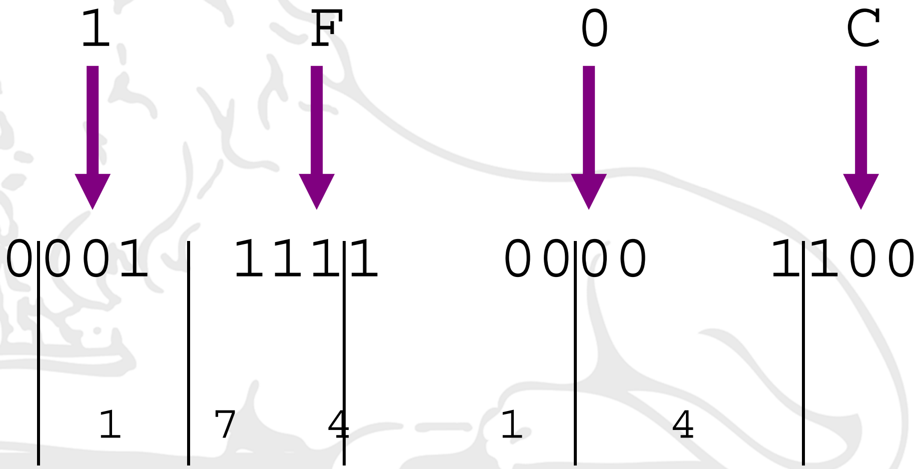
Hexadecimal to Octal:



Hexadecimal to Octal:

- Technique
  - Use binary as an intermediary

1F0C$_{16}$ = ?$_{8}$



1 F 0 C

0001 1111 0000 1100

1 7 4 1 4

1F0C$_{16}$ = 17414$_{8}$

Exercise – Convert …

Decimal	Binary	Octal	Hexadecimal
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

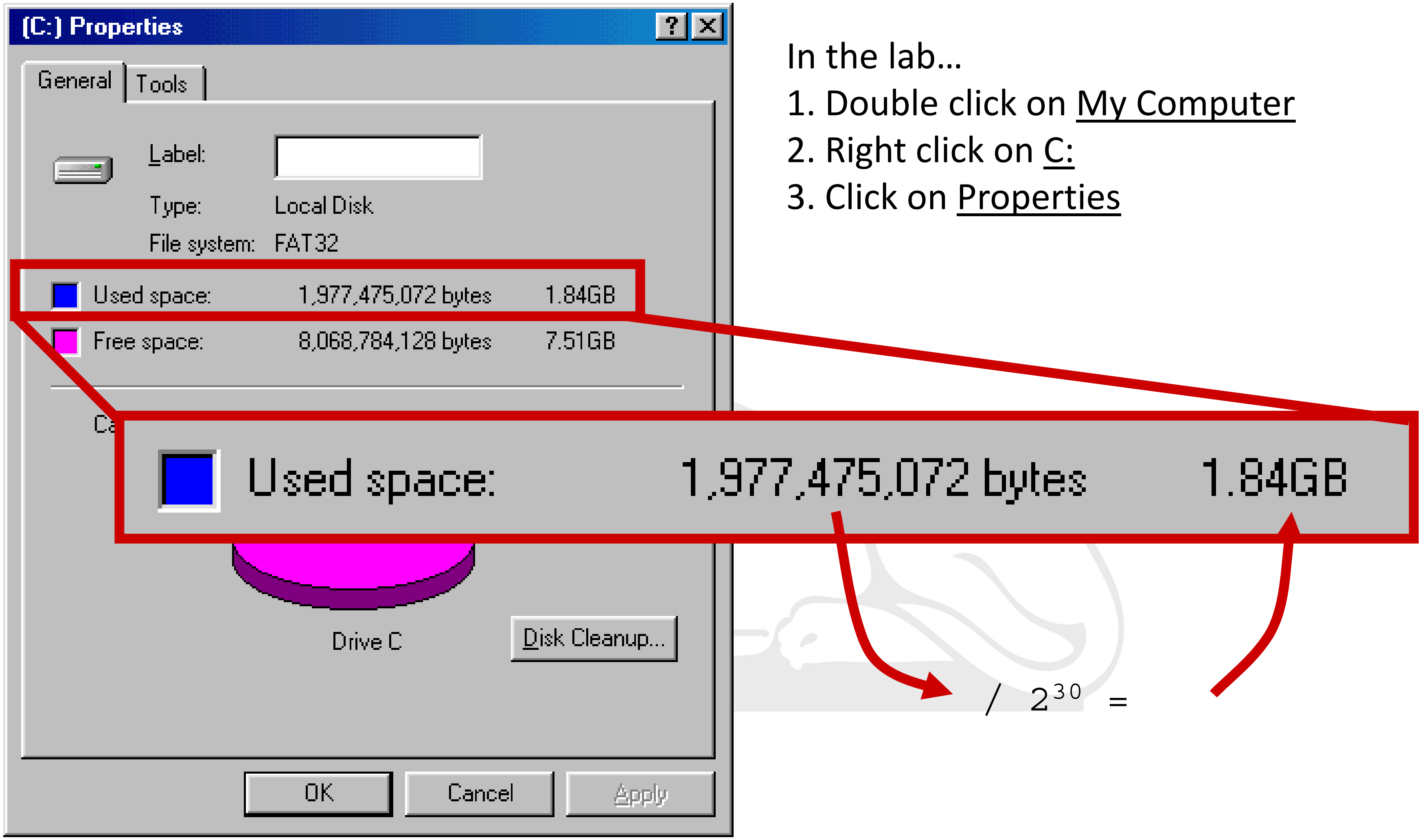
Common Powers (2 of 2):

- 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:



(C:) Properties

Label:

Type: Local Disk

File system: FAT32

🟦 Used space: 1,977,475,072 bytes 1.84GB

🟪 Free space: 8,068,784,128 bytes 7.51GB

🟦 Used space: 1,977,475,072 bytes 1.84GB

$/ \ 2^{30} \ =$

In the lab…

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

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.		

Review – multiplying powers

- For common bases, add powers

$$a^{b} \times a^{c} = a^{b+c}$$

$$2^{6} \times 2^{10} = 2^{16} = 65,536$$

or…

$$2^{6} \times 2^{10} = 64 \times 2^{10} = 64k$$

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”

pp. 36-38

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
```

pp. 39

Multiplication (2 of 3)

- Binary, two 1-bit values

A	B	A × 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)

$$\begin{array}{lrl} 3.14 \Rightarrow & 4 \times 10^{-2} & = 0.04 \\ & 1 \times 10^{-1} & = 0.1 \\ & 3 \times 10^{0} & = 3 \\ & & \quad 3.\underline{14\qquad} \end{array}$$

pp. 46-50

Fractions

- Binary to decimal

10.1011 => 1 x 2^{-4} = 0.0625

1 x 2^{-3} = 0.125

0 x 2^{-2} = 0.0

1 x 2^{-1} = 0.5

0 x 2^{0} = 0.0

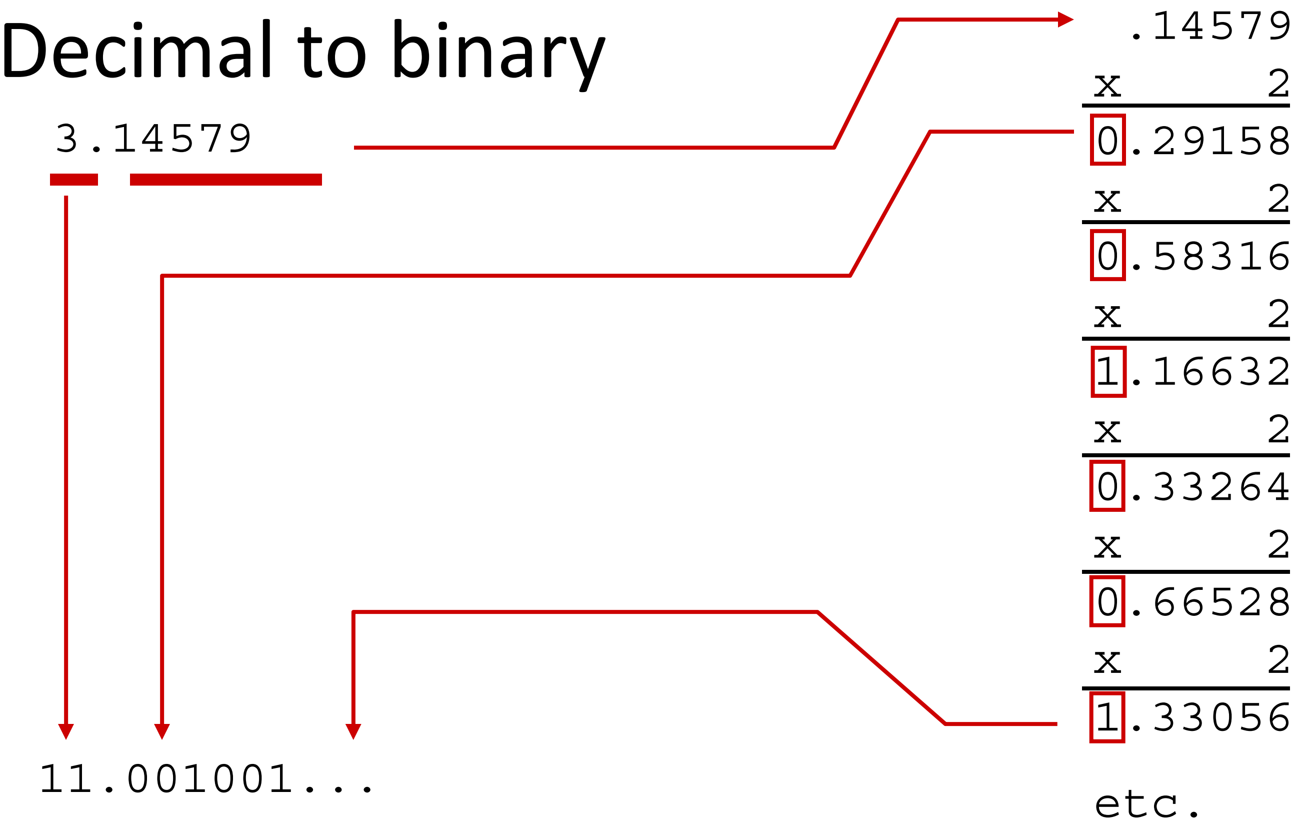
1 x 2^{1} = 2.0

2.6875

pp. 46-50

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...
```

p. 50

Exercise – Convert ...

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

Don’t use a calculator!

Skip answer

Answer

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

Test:

- Decimal: 111

- Hex: 6F
- Oct: 157
- Bin: 0110 1111

How a computer stores information

Binary Numbers are at the heart of how a computer stores all information

- Computers Store ALL information using Binary Numbers
- Computers use binary numbers in different ways to store different types of information.
- Common types of information that are stored by computers are :
  - Whole numbers (i.e. Integers).
    Examples: 8 97 -732 0 -5 etc
  - Numbers with decimal points.
    Examples: 3.5 -1.234 0.765 999.001 etc
  - Textual information (including letters, symbols and digits)
- Keep reading …

Integers

- Integers (e.g., 87)

  A computer stores integer numbers (i.e. “whole” numbers) simply as the equivalent binary value for that number.

Numbers with Decimal Points

- Numbers with decimal points (e.g., 87.123)
  - Internally, a computer stores a number with a decimal point as two different integer numbers (each stored using binary). To get the actual value, the computer performs a mathematical calculation using the two integers to derive the number.
  - We will NOT discuss here the **actual** mathematical calculation nor how the computer breaks a number with a decimal point into two integers.

    [NOTE: The two integers are NOT the whole number part and fractional part.]

Letters and symbols

- Letters and symbols
  - To store letters and symbols, the computer assigns every character on the keyboard a numerical value.
  - Computers remember letters and other symbols by storing the binary number for the symbol.
  - For this system to work a standard numbering system needs to be defined and consistently used for all symbols that the computer needs to process.
  - See the following slide …

ASCII (Americal Standard Code for Information Interchange)

- ASCII (Americal Standard Code for Information Interchange) is the standard numbering given to all characters on a standard keyboard.

- “ASCII values” range in number from 1 to 128. Some “ASCII values” and their associated symbols are listed to the right.

- Note that EVERY symbol on a standard keyboard has an ASCII value. Even the digits 0,1,2,…9 have ASCII values. (see next slide)

Some ASCII values (values 1-31 and 128 are not shown)

- 32 = Space
- 33 = !
- 34 = “
- 35 = #
- 36 = $
- 37 = %
- 38 = &
- 39 = `
- 40 = (
- 41 =)
- 42 = *
- 43 = +
- 44 = ,
- 45 = -
- 46 = .
- 47 = /
- 48 = 0
- 49 = 1
- 50 = 2
- 51 = 3
- 52 = 4
- 53 = 5
- 54 = 6
- 55 = 7
- 56 = 8
- 57 = 9
- 58 = :
- 59 = ;
- 60 = <
- 61 = =
- 62 = >
- 63 = ?
- 64 = @
- 65 = A
- 66 = B
- 67 = C
- 68 = D
- 69 = E
- 70 = F
- 71 = G
- 72 = H
- 73 = I
- 74 = J
- 75 = K
- 76 = L
- 77 = M
- 78 = N
- 79 = O
- 80 = P
- 81 = Q
- 82 = R
- 83 = S
- 84 = T
- 85 = U
- 86 = V
- 87 = W
- 88 = X
- 89 = Y
- 90 = Z
- 91 = [
- 92 = \
- 93 =]
- 94 = ^
- 95 = _
- 96 = `
- 97 = a
- 98 = b
- 99 = c
- 100 = d
- 101 = e
- 102 = f
- 103 = g
- 104 = h
- 105 = i
- 106 = j
- 107 = k
- 108 = l
- 109 = m
- 110 = n
- 111 = o
- 112 = p
- 113 = q
- 114 = r
- 115 = s
- 116 = t
- 117 = u
- 118 = v
- 119 = w
- 120 = x
- 121 = y
- 122 = z
- 123 = {
- 124 = |
- 125 = }
- 126 = ~

Why do 0 through 9 have ASCII values?

- Numbers that are used in mathematical calculations
  - If a computer needs to do math with a number it will store that number using the appropriate binary representation of the number.
  - This makes it easier for the computer to perform mathematical calculations with the number.
  - Example: 5 would be stored as

    00000101

- Numbers that are NOT used in mathematical calculations
  - If the computer does NOT need to do math with the number (e.g. a zip code) then it will generally store the number using the ASCII values of the digits.
  - In this case using the ASCII value is more efficient (for reasons we will not explain here).
  - Example 5 would be stored using its ASCII value of 53 which is represented in binary as

    00110101

Other numbering systems (Unicode and EBCDIC)

- ASCII
  - ASCII was the standard numbering system for many years and is still used widely today.

- EBCDIC
  - Is a different numbering system used by IBM Mainframe computers.
  - It is very similar to ASCII but uses different numbers to represent the symbols.
  - EBCDIC stands for “Extended Binary Coded Decimal Interchange Code”

- Unicode
  - ASCII and EBCDIC are limited to just the basic English letters and common symbols.
  - Today computers use many different symbols including letters from languages that don’t use English letters (e.g. Hebrew, Chinese, etc.) and international symbols (e.g. the English pound sign)
  - Unicode defines a unique number for every symbol in all known languages (e.g. Hebrew, Chinese, etc.) and commonly used non-letter symbols (e.g. English pound sign, copyright symbol, etc).
  - Modern programs are moving towards using Unicode to store letters and symbols.
  - It should be noted that Unicode numbers 1-128 correspond to the EXACT SAME symbols as ASCII 1-128

How a computer stores information (i.e., decimal vs. analog)

How a computer stores info.

- A computer stores all information as binary numbers.
- Computer memory simply remembers ones and zeros
- Computer storage remembers ones and zeros
- Data is passed inside the computer from one portion of the computer to another (e.g. memory to CPU to graphics card, etc) by ones and zeros

Terms (bit, byte, etc.)

- BIT
  - definition: a single Binary digIT (i.e. BIT)
- BYTE
  - definition: 8 bits
- NYBLE
  - definition: 4 bits

Prefixes

- Prefixes
  - Kilo: one thousand
  - Mega: one millioin
  - Giga: one billion
  - Tera: one trillion
  - Peta: one quadrillion
  - Exa: one quintillion
  - … etc.

Data sizes

- Data sizes
  - Kilobyte (KB)
    - "about" one thousand bytes
    - exactly 2^{10} or 1024 bytes
  - Megabyte (MB)
    - "about" one million bytes
    - exactly 2^{20} or 1,048,576 bytes
  - Gigabyte (GB)
    - "about" one billion bytes
    - exactly 2^{30} or 1,073,741,824 bytes
  - Terabyte (TB)
    - "about" one trillion bytes
    - exactly 2^{40} or 1,099,511,627,776 bytes

Data Sizes – bytes vs bits

- MB = one Mega Byte
- Mb = one Mega Bit

Speeds

- MBPS = one MegaByte per second
- MbPS = one Mega Bit per second

How data is stored using binary

- Integers are stored as a binary number
- A character is stored as the ASCII value (i.e. an integer) for that character
- A decimal number is stored using two different integer values - the mantissa and the exponent

- 1 bit
- 1 byte = 8 bits
- 1 kb = 2^{10} bytes = 1024 bytes !=1000
- 1 Mb = 1 k k bytes = $2^{10} * 2^{10}$ bytes
- 1 G b = $2^{10} * 2^{10} * 2^{10}$ bytes
- 1 Terab = $2^{10} * 2^{10} * 2^{10} * 2^{10}$ bytes

Even larger capacity

- 1 petabyte = $2^{10} * 2^{10} * 2^{10} * 2^{10} * 2^{10}$ bytes (2 to the 50th power)
- 1 exabyte= 2^{60}
- 1 zettabyte = 2^{70}
- 1 yottabyte = 2^{80}

Some interesting facts about what these various-sized bytes can store:

- 1 bit: a binary decision
- 1 byte: a character
- 5 Megabytes: The complete works of Shakespeare
- 2 Gigabytes: 20 meters of shelved books
- 10 Terabytes: The printed collection of the US Library of Congress
- 200 Petabytes: All printed material in the whole word.
- 5 Exabytes: All words ever spoken by human beings