Week 1010
Number Systems

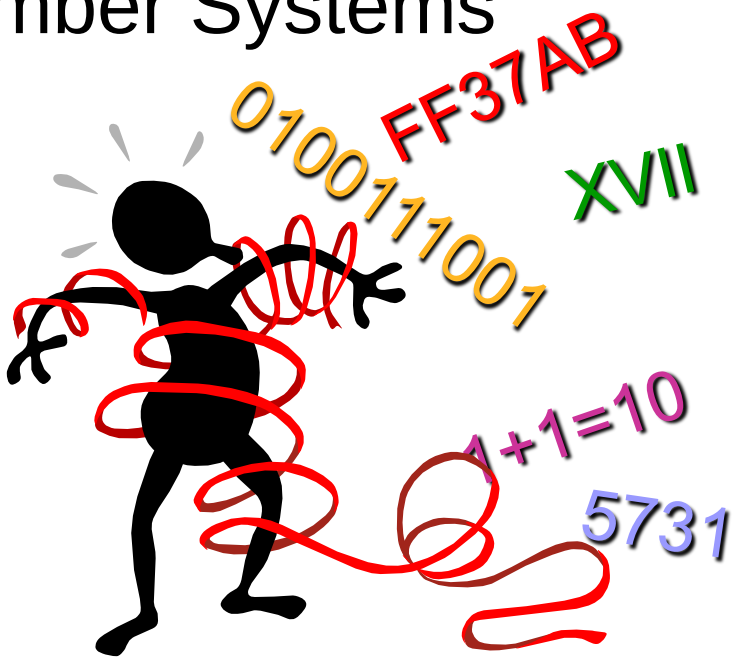
CCS1310

Computer Systems Architecture

Dr. Kostas Dimopoulos

Lecture Outline

- Data Representation and Number Systems
- Data Formats
- The Decimal System
- The Binary System
- The Hexadecimal System
- The Octal System
- Conversions between Systems



Data Formats

- Data must be converted into appropriate representation so it can be processed, stored and used
- Data can be in different types: character, graphic, audio etc.
- Computer systems, programs, input/output devices are also of different types
- Need of **standard** data representations recognized by a wide variety of hardware and software

Data Formats (cont'd)

Type of Data	Standards
Alphanumeric	ASCII
Image (bit map)	GIF, BMP
Image (object)	Postscript
Outline graphics and fonts	True Type
Sound	WAV, midi, mp3
Page description	pdf, html
Video	mpeg, mp4, avi

A data format belonging to a single vendor becomes a de facto standard due to the popularity of the product

Alphanumeric Character Data

- Characters, number digits and punctuation
- Each alphanumeric data must be translated to a corresponding binary code
- **ASCII**
  - **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
  - Uses 8 bits for each character
- **Unicode**
  - Uses 16 bits for each character

Image Data

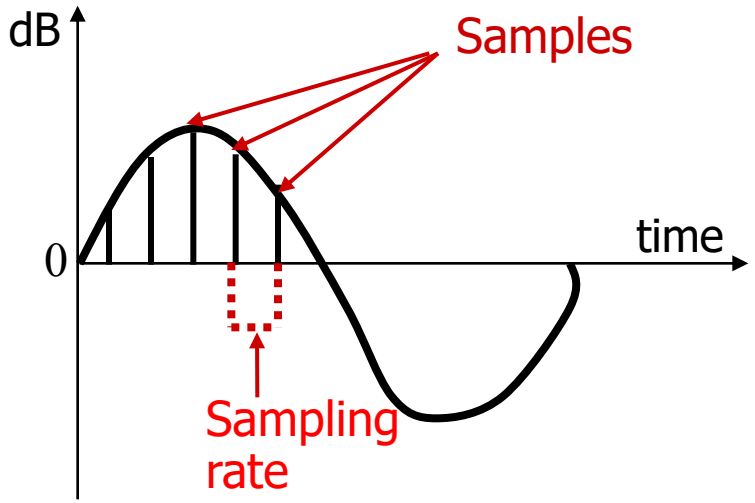
- Different shapes, sizes textures, colors
- Different processing requirements require different forms for image data
- Difficult to define a single universal format that can be used for images (in the way alphanumeric standards are used for text)
- Images will be formatted according to display, processing, application, storage requirements

Image Data Categories

- **Bitmap images**
  - Photographs, paintings that are characterized by continuous, variations in shading, color, shape and texture
  - Produced with image scanner, or using a paint program
  - Need to represent and store each individual point within the image
- **Object images**
  - Made up of graphical shapes such as lines and curves that can be defined geometrically
  - Sufficient to store geometrical information about each object and the relative position of each object in the image
  - Can be produced using drawing or design package

Audio Data

- Original sound wave is analog in nature
- Need to convert it to digital
- The analog waveform is sampled electronically at regular time intervals
- Each time a sample is taken, the amplitude of the sample is measured by an electronic circuit (A-to-D converter) that converts the analog value to a binary equivalent



Storing Data

- We can use bit patterns of zeroes and ones to represent:
  - Numeric values
  - Alphanumeric text
  - Pictures
  - Sounds
  - Instructions

- **WHO** determines if a stored bit pattern represents a **numeric value**, or **alphanumeric text**, or part of a **sound** or part of a **picture**, or part of an **instruction?**
  - **SOFTWARE** DOES!
  - “All bits look the same to the CPU”

Storing Data (cont'd)

- Computers are used to store data
- Typically numbers and characters or composition of these
- The key is to use an efficient representation
- Question: How do computers store data?

  In a program:

  ```
  int x = 12;
  float y = 1.24;
  ```

  What do the variables x and y hold?

Number Systems

- A number is a mathematical concept e.g. 10
- Many ways to represent a number
  - e.g. 10, ten, X, IIIIIIIIII, 2x5, 100/10
- Symbols are used to create a representation
- Which representation is best?

Decimal Number System

- What you use every day when you count
- Latin word Decem which means ten
- Uses ten digits 0-9 (base 10)
- Digits 0-9 are composed for larger numbers

  e.g. **213** = **2**x10^{2}+**1**x10^{1}+**3**x10^{0}

- Easily understood
- Does not produce excessively clumsy written representations of large values

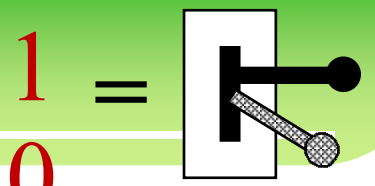
Binary Number System

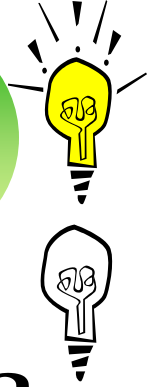
- Latin word Bi which means two
- Bit: short for Binary Digit
- Uses two digits 0-1 (base 2)
  - e.g. **101** = **1**x$2^{\mathbf{2}}$+**0**x$2^{\mathbf{1}}$+**1**x$2^{\mathbf{0}}$

- Produces long written representations

  e.g. $1001110001000_2 = 5000_{10}$

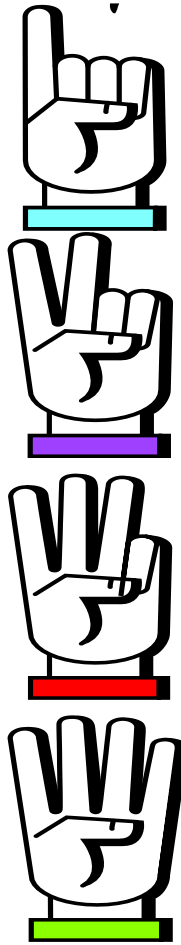
Use of the Binary Number System





- Computers are built from transistors
- Transistor has two states: **ON** or **OFF**
  - Need to represent data using only:

    OFF=0 and ON=1
- Not easily understood but:
  - Direct correspondence to electrical states (on/off) and basic decisions (yes/no)
  - Easier to implement with current technology



Octal and Hexadecimal Number Systems

- Inconvenience of binary system for expressing large values
- Need a number system that allows simplified expression of large values but also to have some commonality with the binary system
  - A number system that is also a power of two

    e.g. base-8, base-16
- Octal system uses eight symbols (0-7)
- Hex system uses sixteen symbols represented by 0-9 and A-F

THE TABLE

Decimal	Binary	Hexadecimal	Octal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7

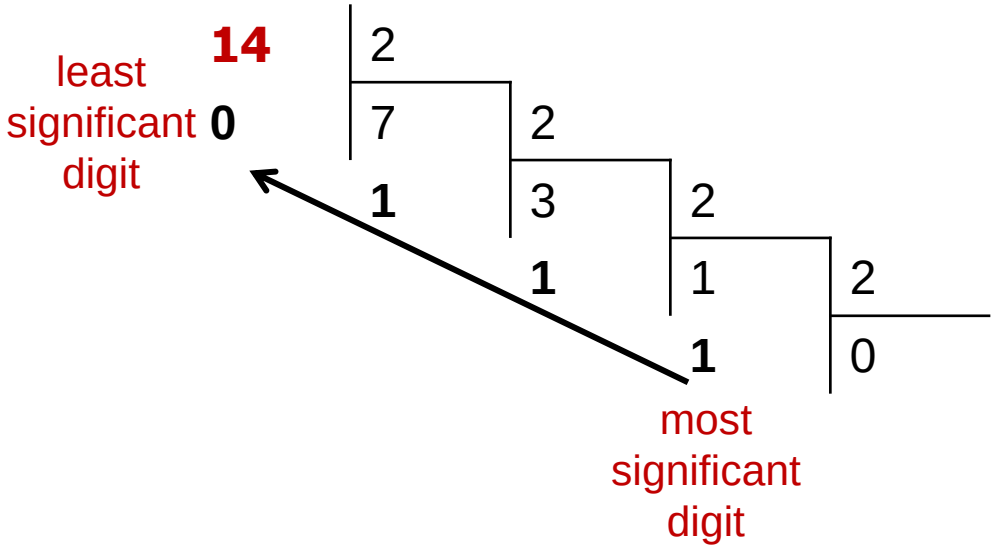
Decimal	Binary	Hexadecimal	Octal
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

Conversion: Decimal to Binary

- Repeatedly divide by the base until there is nothing left to divide
- Each successive remainder represents the value of a digit in the new base reading the value from right to left

Example: $14_{10} = 1110_2$

```
14/2 = 7 Remainder 0
7/2  = 3 Remainder 1
3/2  = 1 Remainder 1
1/2  = 0 Remainder 1
```



Conversion: Binary to Decimal

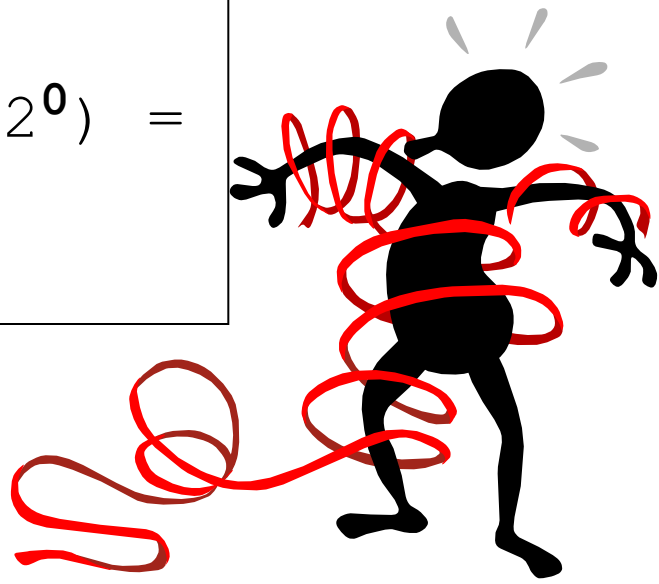
- For n digits, the sum from i=0 to i=n-1:
  $\Sigma(digit*2^i)$
  - Starting from the least significant digit

Example: $1101_2 = 13_{10}$

$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) =$
$8 + 4 + 0 + 1 =$
13

Your turn:

$100110_2 = ?_{10}$



Conversion: Binary to Hex

- Group into 4 bits starting at least significant digit
- Add zero's at most significant end, if necessary
- Convert each group to each corresponding hex number

Example: $110101110_2 = 1AE_{16}$

```
0001 1010 1110 =
 1    A    E
```

Your turn:

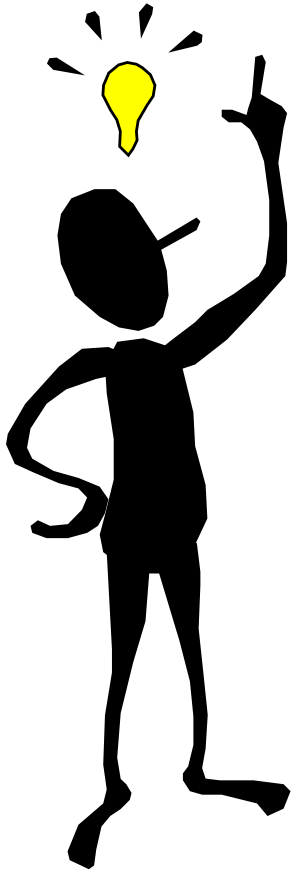
$11101101011101_2 = ?_{16}$

Conversion: Hex to Binary

- Simply find the 4-bit binary code for each hexadecimal digit and replace

Example: $39C8_{16} = 0011100111001000_2$

```
 3    9    C    8   =
0011 1001 1100 1000₂
```



Your turn:

$62AF5_{16} = ?_2$

$D7G3B4_{16} = ?_2$

Conversion: Decimal to Hex

- Just like Decimal to Binary but dividing with base 16

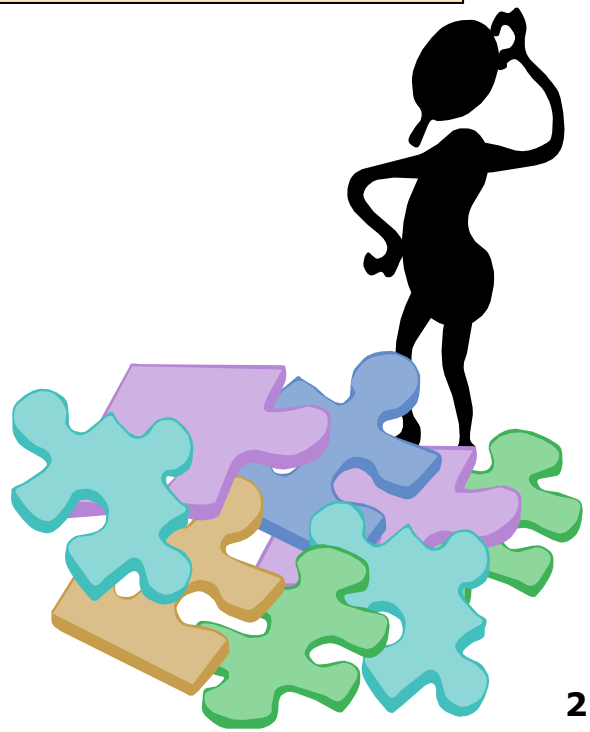
Example: $28_{10} = 1C_{16}$

```
28/16 = 1 Remainder 12 → C
1/16  = 0 Remainder 1
```

Your turn:

$7842_{10} = ?_{16}$

$15112_{10} = ?_{16}$



- Or do it in two steps

  Step 1: Decimal to Binary

  Step 2: Binary to Hexadecimal

Conversion: Hex to Decimal

- Just like Binary to Decimal but with base 16

Example: $2BF_{16} = 703_{10}$

$(2 \times 16^2) + (B \times 16^1) + (F \times 16^0) =$

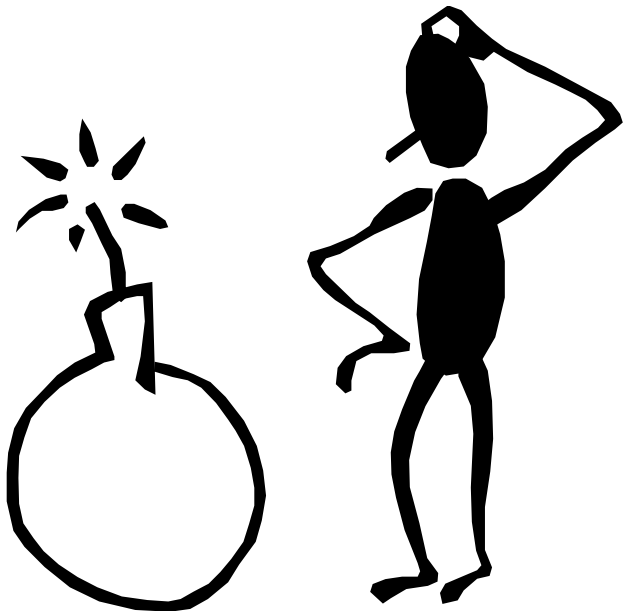
$(2 \times 256) + (11 \times 16) + (15 \times 1) =$

$512 + 176 + 15 =$

703

Your turn:

$D3A_{16} = ?_{10}$

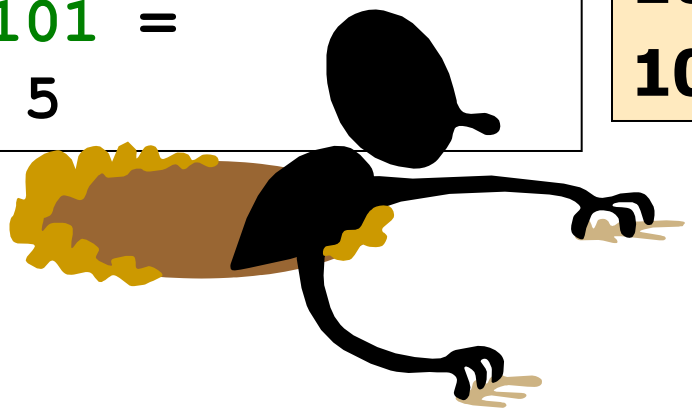


Conversion: Binary to Octal

- Group into 3 bits starting at least significant digit
- Add zeros at most significant end, if necessary
- Convert each triad to each corresponding octal number

Example: $1011101_2 = 135_8$

001 011 101 =
1 3 5



Your turn:

$10010011101_2 = ?_8$

$100101110111_2 = ?_8$

Conversion: Octal to Binary

- Simply find the 3-bit binary code for each octal digit and replace

Example: $3547_8 = 011101100111_2$

3	5	4	7	=
011	101	100	111_2	

Your turn:

$2613_8 = ?_2$

$1482_8 = ?_2$

Conversion:
Hex to Octal and Octal to Hex

- Do the conversion in two simple steps

  Step 1: Hexadecimal to Binary

  Step 2: Binary to Octal

- In a similar manner we convert octal to hexadecimal

Fractions

Converting 55.67 to binary : 110111.101010111

```
55/2 = 27   Remainder = 1 LSD
27/2 = 13   Remainder = 1
13/2 = 6    Remainder = 1
6/2  = 3    Remainder = 0
3/2  = 1    Remainder = 1
1/2  = 0    Remainder = 1 MSD
```

Integer part: 110111

Some sequences may never end, so you must pick a stopping point based on desired accuracy and precision

```
0.67 * 2 = 1.34 Integer = 1 MSD
0.34 * 2 = 0.68 Integer = 0
0.68 * 2 = 1.36 Integer = 1
0.36 * 2 = 0.72 Integer = 0
0.72 * 2 = 1.44 Integer = 1
0.44 * 2 = 0.88 Integer = 0
0.88 * 2 = 1.76 Integer = 1
0.76 * 2 = 1.52 Integer = 1
0.52 * 2 = 1.04 Integer = 1 LSD
```

Fraction part = .101010111

Fractions (cont'd)

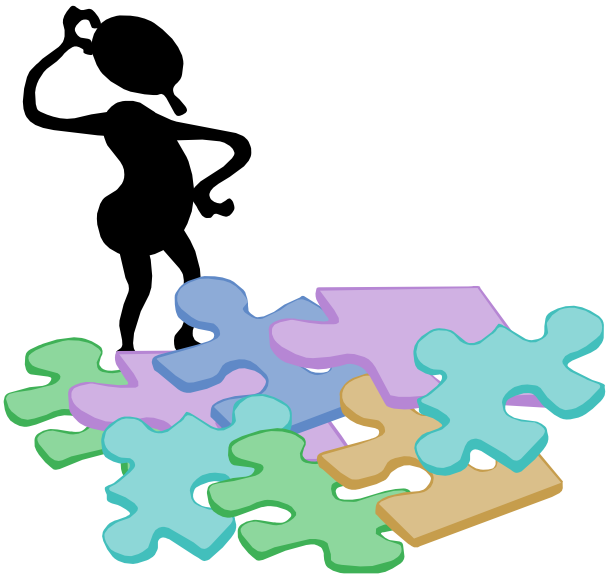
What is 10101.101_2 in Decimal?

- Do the integer part normally - 10101 is:
  $(1x2^4)+(1x2^2)+(1x2^0)=16+4+1=21$
- For the fractional part 101 is:
  $(\mathbf{1}x2^{\mathbf{-1}})+(\mathbf{0}x2^{\mathbf{-2}})+(\mathbf{1}x2^{\mathbf{-3}})$
- 0.5 + 0 + 0.125 = 0.625

  Result: 21.625

Your turn:

$110100.0101_2=?_{10}$



Binary Addition

- The Basic Computer Operation is ADDITION

  0 + 0 = 0

  0 + 1 = 1

  1 + 0 = 1

  1 + 1 = 1 0

How about subtraction?
How about multiplication?
How about division?

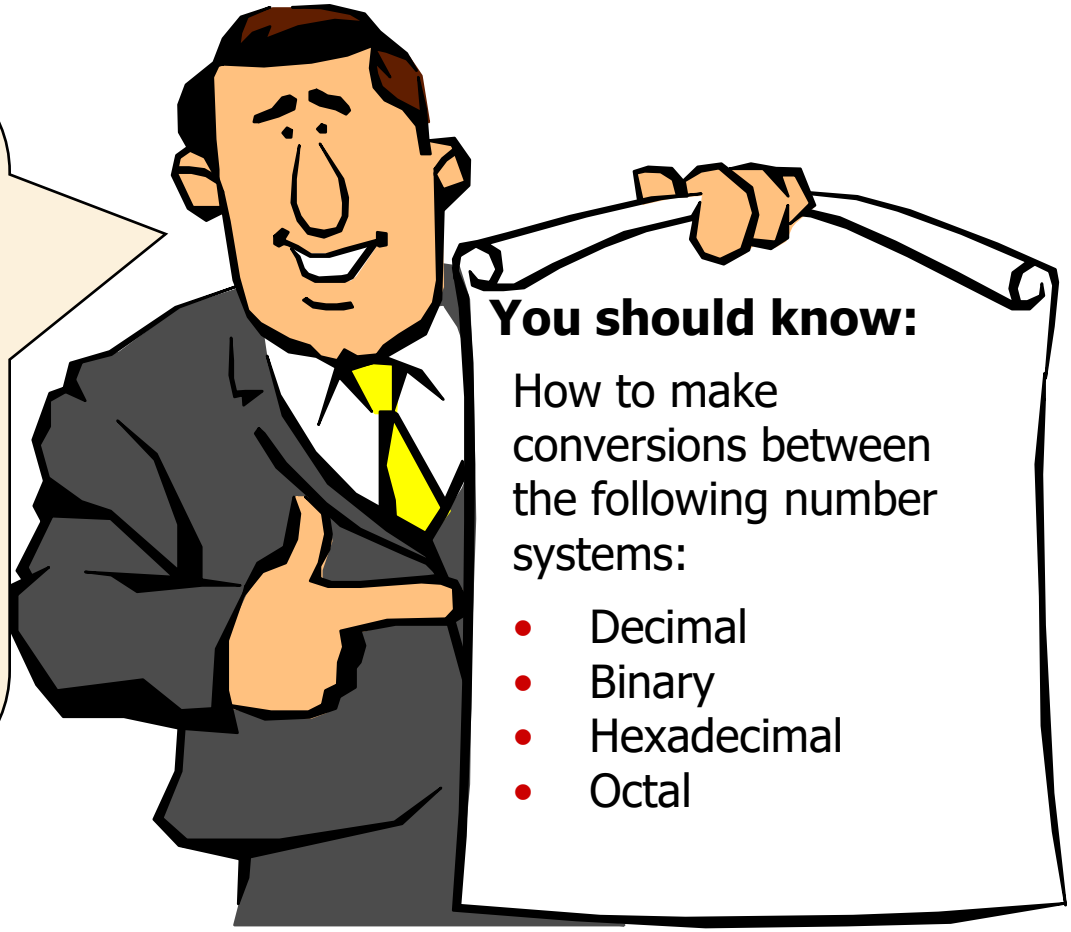
Summary

Bit patterns of 0's and 1's are used to represent every data format in the computer.

The binary system is preferred due to the direct correspondence with electrical states.

Except the binary, the octal and hexadecimal systems are also used in computer science.

Since humans work with the decimal system...



You should know:

How to make conversions between the following number systems:

- Decimal
- Binary
- Hexadecimal
- Octal