

# The Binary System

Lecture #2

(4)

The digits 0, 1 in the binary system have a special name. They are called bits.

Bits represent the smallest memory element used in every computer.

## Example

How many bits are needed to represent:

- (a) A decimal digit?
- (b) A binary digit?
- (c) An octal digit?
- (d) A hexadecimal digit?

(a) For a decimal digit, the largest digit is 9.

$$\begin{aligned} \text{In binary: } (9)_{10} &= (8 + 1)_{10} \\ &= (1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0)_{10} \\ &= \underline{\underline{(1 \ 0 \ 0 \ 1)_2}} \end{aligned}$$

requiring a total of 4 bits.

(b) 1 bit.

(c) The octal numbers are: 0, 1, ..., 7.

$$(7)_8 = (1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0)_{10}$$

$$= \underline{\underline{(1 \ 1 \ 1)_2}} \text{ requiring } \underline{\underline{3 \text{ bits}}}$$

d) For hexadecimal numbers, we have:

$$\begin{array}{c}
 \overline{F}_{16} = (15)_{10} = (1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) = (1111)_2 \\
 \begin{array}{ccc}
 \uparrow & & \uparrow \\
 \text{base 16} & & \text{binary} \\
 & \nearrow & \\
 & \text{decimal} & 
 \end{array}
 \end{array}$$

Hexadecimals require 4 bits.

Working with binary, octal and hexadecimal numbers is very easy. We can convert:

- (a) binary to octal, hexadecimal, and
- (b) octal, hexadecimal to binary

Very easily.

beginning from the right (least significant bit)

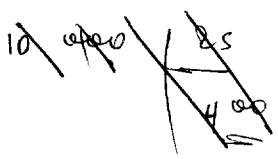
(a) For converting:

I binary to octal: "group" every three bits and convert each group of 3 bits into octa

Example:  $(\underbrace{101}_{3 \text{ bits}} \underbrace{111}_{3 \text{ bits}})_2 = (\underbrace{101}_{2 \text{ bits}})_2 \cdot 8^1 + (\underbrace{111}_{2 \text{ bits}})_2 \cdot 8^0$

first octal digit
zeroth digit

$$\begin{aligned}
 &= (1 \cdot 4 + 1 \cdot 1) \cdot 8^1 + 7 \\
 &= (57)_8
 \end{aligned}$$



In general, for  $n$  bits, the range of numbers we can represent is  $(0)_{10}$  to  $(2^n - 1)_{10}$

e.g. unsigned short int in C uses 16 bits  
 0 to  $2^{16} - 1 \rightarrow 0$  to 65535, 0000 to FFFF hex

" long in " " 32 bits  
 0 to  $2^{32} - 1 \rightarrow$  0000 0000<sub>h</sub> to FFFF FFFF<sub>h</sub>

In computer work literature (memory)

$$2^{10} = 1024 = 1K \text{ (Kilo)}$$

$$2^{20} = 2^{10} \times 2^{10} = 1024 \times 1024 = 1M \text{ (mega)}$$

$$2^{30} = 2^{10} \times 2^{20} = 1024 M = 1G \text{ (giga)}$$

also 1 byte = 8 bits

i.e. 64 Mbytes of memory =  $64 \times 1024 \times 1024 \times 8$  bits

**II** binary to hexadecimal: beginning from the right (least significant bit) (6)  
 group every four bits and convert each group of 4 bits into hexadecimal.

Example:

$$\left( \underbrace{1010}_{4 \text{ bits}} \underbrace{0110}_{4 \text{ bits}} \right)_2 = \left( \begin{array}{cccc} 8 & 4 & 2 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 0 & 1 & 0 \end{array} \right)_2 \cdot 16^1 + \left( \begin{array}{ccc} 8 & 4 & 1 \\ \downarrow & \downarrow & \downarrow \\ 0 & 1 & 1 & 0 \end{array} \right)_2 \cdot 16^0$$

↑ first hexadecimal digit.  
↑ second hexadecimal digit.

$$= (1 \cdot 8 + 1 \cdot 2)_{16} \cdot 16^1 + (1 \cdot 4 + 1 \cdot 2)_{16}$$

$$= (AG)_{16}$$

(b) For converting:

**I** octal to binary: "ungroup" every digit and convert it into 3 bits.

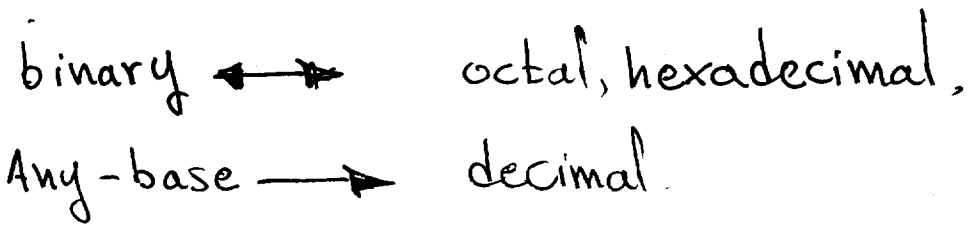
$$\begin{aligned} \text{Example: } (75)_8 &= (7 \cdot 8^1 + 5 \cdot 1)_{10} \\ &= \left( \begin{array}{ccc} 4 & 2 & 1 \\ \downarrow & \downarrow & \downarrow \\ 1 & 1 & 1 \end{array} \right)_2 \cdot 8^1 + \left( \begin{array}{ccc} 4 & 2 & 1 \\ \downarrow & \downarrow & \downarrow \\ 1 & 0 & 1 \end{array} \right)_2 \\ &= \left( \overbrace{111} \quad \overbrace{101} \right)_2 \end{aligned}$$

## II hexadecimal to binary

“Ungroup” every digit and convert it into 4 bits.

Example:  $(FE)_{16} = (F)_{16} \cdot 16^1 + (E)_{16} \cdot 16^0$   
 $= (1111)_2 \cdot 16^1 + (1110)_2 \cdot 1$   
 $= (1111 \ 1110)_2$

So far we covered:



**Convert** How about the general problem:  
 a number from any base  $b_1$  to any other base  $b_2$ ?

To compute this (in-general), construct the table:

Quotient on division by $b_2$	Remainder	Digit
$N^{(1)} = \text{int}(N/b_2)$	$N - N^{(1)} \cdot b_2$	$C_0 = N - N^{(1)} \cdot b_2$
$N^{(2)} = \text{int}(N^{(1)}/b_2)$	$N^{(1)} - N^{(2)} \cdot b_2$	$C_1 = N^{(1)} - N^{(2)} \cdot b_2$
$\vdots$		
$0$	$N^{(n)} - N^{(n+1)} \cdot b_2$	$C_{n-1} = N^{(n)} - N^{(n+1)} \cdot b_2$

Answer:  $(C_{n-1} \dots C_1 C_0)_{b_2}$

these two columns are

Example: Decimal to Binary

$b_2 = 2$ . Consider converting  $23_{10}$

Quotient	Remainder	
$\text{int}(23/2) = 11$	$23 - 11 \times 2 = 1$	(23 is odd)
$\text{int}(11/2) = 5$	$11 - 5 \times 2 = 1$	(1 is odd)
$\text{int}(5/2) = 2$	$5 - 2 \times 2 = 1$	(5 is odd)
$\text{int}(2/2) = 1$	$2 - 1 \times 2 = 0$	(2 is even)
$\text{int}(1/2) = 0$	$1 - 0 \times 2 = 1$	(1 is odd)

Hence:  $23_{10} = (10111)_2$

NB: The order at which we list the digits has been reversed.

Another example:

Decimal to hex.

Eg: In this case, keep dividing by 16.  
Convert  $(957)_{10}$

Quotient	Remainder	Hex
$\text{int}(957/16) = 59$	13	D
$\text{int}(59/16) = 3$	11	B
$\text{int}(3/16) = 0$	3	3

Stop since we reached zero.

Answer:  $(957)_{10} = (3BD)_{16}$

Notice: You can also verify your answers by converting back:

$$\begin{aligned}
 (3BD)_{16} &= (3 \cdot 16^2 + 11 \cdot 16 + 13)_{10} \\
 &\stackrel{?}{=} 957
 \end{aligned}$$

For Decimal to octal, just keep dividing by 8.