ECE380 Digital Logic

Number Representation and Arithmetic Circuits:

Other Number Representations

Other number representations

- Previously, we dealt with binary integers (signed or unsigned) in a positional number representation
- Other number representations are also commonly used :
  - Fixed-point: allows for fractional representation
  - Floating-point: allows for high precision, very large and/or very small numbers
  - Binary-coded decimal (BCD): another form for integer representation

Fixed-point numbers

- A ***fixed-point*** number consists of integer and fraction parts
- In positional notation, it is written as
  $B=b_{n-1}b_{n-2}...b_1b_0.b_{-1}b_{-2}...b_{-k}$
- With a corresponding value of

$$V(B)=\sum_{i=-k}^{n-1} b_i \times 2^i$$

- The position of the radix point is assumed to be fixed

Fixed-point numbers

- For example,
  $B=(01001010.10101)_2$
  $B=1x2^6+1x2^3+1x2^1+1x2^{-1}+1x2^{-3}+1x2^{-5}$
  $B=64+8+2+.5+.125+.03125$
  $B=(74.65625)_{10}$
  $B=(8A.A8)_{16}$

- Logic circuits that deal with fixed-point numbers are essentially the same as those used for integers

Floating-point numbers

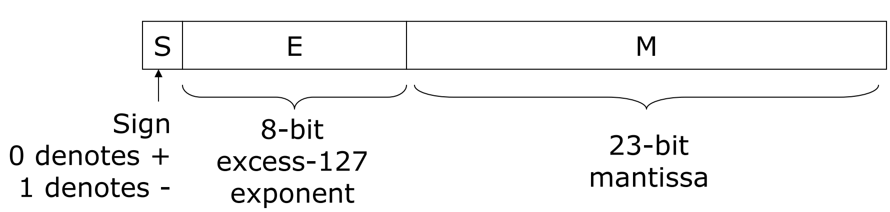
- Fixed-point numbers have a range that is limited by the significant digits used to represent the number
- For some applications, it is often necessary to deal with numbers that are very large (or very small)
- For these cases, it is better to use a ***floating-point*** representation in which numbers are represented by a ***mantissa*** comprising the significant digits and an ***exponent*** of the radix *R*
- The format is

  $Mantissa \times R^{Exponent}$

- The numbers are usually ***normalized*** such that the radix point is placed to the right of the first non-zero digit (for example, 5.234×10^{43} or 3.75×10^{-35})

IEEE single precision format

- The IEEE defines a 32-bit (single precision) format for floating point values
  - Sign bit (S): most significant bit
  - 8-bit exponent field (E): excess-127 exponent
    - True exponent = E-127
    - E=0 -> 32-bit value=0
    - E=255 -> 32-bit value=∞
  - 23-bit mantissa (M)



IEEE single precision format

- The IEEE standard calls for a normalized mantissa, which means that the most-significant bit is always set to 1.
- It is not necessary to include this bit explicitly in the mantissa field
  - If M is the value in the 23-bit mantissa field, the true (24-bit) mantissa is actually 1.M
- The value of the floating point number is then
  - Value = $(-1)^S.M \times 2^{E-127}$

Floating-point example

- For example,

01000000011000000000000000000000

$=+(1.11)_2 x2^{(128-127)}$
$=+(1.11)_2 x2^1$
$=+(11.1)_2$
$=+(1x2^1+1x2^0+1x2^{-1})=(3.5)_{10}$

What is the following?

00111111011000000000000000000000

Binary-coded-decimal numbers

- It is possible to represent decimal numbers simply by encoding each decimal digit in binary form
  - Called ***binary-coded-decimal*** (BCD)
- Because there are 10 digits to represent, it is necessary to use four bits per digit
  - From 0=0000 to 9=1001
  - $(01111000)_{BCD}=(78)_{10}$
- BCD representation was used in some early computers and many handheld calculators
  - Provides a format that is convenient when numerical information is to be displayed on a simple digit-oriented display

ASCII character code

- The most popular code for representing information in computers is used for both numbers and letters and some control codes
- It is the ***American Standard Code for Information Interchange*** (ASCII) code
- ASCII code uses seven-bit patterns to represent 128 different symbols including
  - Digits (0-9)
  - Lowercase (a-z) and uppercase (A-Z) characters
  - Punctuation marks and other commonly used symbols
  - Control codes
- The 8-bit extended ASCII code is used to represent all of the above and another 128 graphics characters

Example ASCII character codes

- $(1000001)_{ASCII}$=(41H)=‘A’
- $(1000010)_{ASCII}$=(42H)=‘B’

- $(1100001)_{ASCII}$=(61H)=‘a’
- $(1100010)_{ASCII}$=(62H)=‘b’

- $(0110000)_{ASCII}$=(30H)=‘0’
- $(0111001)_{ASCII}$=(39H)=‘9’

- ASCII table given in the textbook