ELL100: INTRODUCTION TO ELECTRICAL ENGG.

Number Systems

Instructor for this part: Debanjan Bhowmik, Assistant Professor,
Department of Electrical Engineering, IIT Delhi

Contact : debanjan@ee.iitd.ac.in

Textbook: ‘Digital Design’ by Moris Mano (Chapter 1: Digital Systems and Binary Numbers)

Number Systems

- Non-Positional Systems (symbols represent numbers) :
  - Roman Numerals : I (1), V (5), X (10), L (50), C (100), D (500), M (1000)
  - Chinese Numerals : 一(1), 二 (2), 三 (3), 十 (10), 二十 (20), 百 (100), 千 (1000)
- Positional Systems (symbols get value/significance from position):
  - Decimal: 0,1,2,3,4,5,6,7,8,9 [Base 10]
  - Binary: 0,1 [Base 2]
  - Octal: 0,1,2,3,4,5,6,7 [Base 8]
  - Hexadecimal: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F [Base 16]

Numerical Value

- Any number *N* when represented in a base *b* is written as

$$N = (d_r d_{r-1} \cdots d_0)_b$$

- ‘b’ denotes the base

$$N = d_r b^r + d_{r-1} b^{r-1} + \cdots + d_1 b + d_0$$

- The notation can also be extended to non-integers

$$(0.d_{-1} d_{-2} \cdots)_b = d_{-1} b^{-1} + d_{-2} b^{-2} + \cdots$$

Bit/Digit Significance

- For positional number systems,
  - Right most: Least significant digit/bit (LSB)
  - Left most: Most significant digit/bit (MSB)

Decimal: 4153231

Binary: 1010001

Hex: AE143

MSB

LSB

Common Number Systems

- Decimal Notation (Digits: 0-9) [Usage : Common day-to-day]
$$2198_{10} = 2 \times 10^3 + 1 \times 10^2 + 9 \times 10 + 8 \times 10^0$$
- Binary Notation (Digits: 0 & 1) [Usage: Digital Computers]
$$100010010110_2 = 2^{11} + 2^7 + 2^4 + 2^2 + 2^1 = 2198_{10}$$
- Hexadecimal (Digits : 0-9+A-F) [Usage: Assembly Language] $2018_{10} = 7 \times 16^2 + 14 \times 16 + 2 = 7E2_{16}$

Conversion: Base b → Decimal

- Base b → Decimal
  1. Multiply each bit of the Base b number by its corresponding weighting factor
  2. Sum all the products to get the decimal number

  Examples :

- **Bin to Dec:**

$$\begin{aligned} 1011.01_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= 8 + 2 + 1 + 0.25 = 11.25 \end{aligned}$$

- **Hex to Dec**

$$\begin{aligned} B92.A_{16} &= 11 \times 16^2 + 9 \times 16^1 + 2 \times 16^0 + 10 \times 16^{-1} \\ &= 2816 + 144 + 2 + 0.625 = 2962.625 \end{aligned}$$

Conversions

- Decimal → Base b

  Let the integer to be converted be N_{10}. Then $N_{10} = (d_r d_{r-1} \cdots d_0)_b$

  1. Set $q_0 = N_{10}$, $i = 0$
  2. While $q_i > 0$
     1. $q_{i+1} = \left\lfloor \frac{q_i}{b} \right\rfloor$. (Here $\lfloor x \rfloor$ denotes the largest integer less than or equal to x)
     2. $d_i = q_i - b q_{i+1}$
     3. $i = i + 1$
     4. End while

Conversion: Decimal → Base b

- Example: Converting 7530_{10} to Hex.
  - $q_0=7530$
  - $q_1=\lfloor 7530/16 \rfloor = 470$, $d_0 = 7530-16 \times 470 = 10 \rightarrow A$
  - $q_2= \lfloor 470/16 \rfloor = 29$, $d_1 = 470 - 16 \times 29 = 6$
  - $q_3= \lfloor 29/16 \rfloor = 1$, $d_2 = 29-16=13 \rightarrow D$
  - $q_4= \lfloor 1/16 \rfloor=0$, $d_3 = 1$

  - $7530_{10} = 1D6A_{16}$

Conversions (Non-integers)

- Decimal → Base b

  Let the number to be converted be n_{10} with $0<n<1$. Then

  $$n_{10} = (0.d_{-1}d_{-2}\cdots d_{-k})_b$$

  1. Set $r_0 = n_{10}$, $i = 0$
  2. While ($r_i > 0$ & $i > -k$)
     1. $d_{i-1} = \lfloor br_i \rfloor$. (Here $\lfloor x \rfloor$ denotes the largest integer less than or equal to x)
     2. $r_{i-1} = br_i - d_{i-1}$
     3. $i = i - 1$
     4. End while

Conversion: Decimal → Base b

- Example: Converting 0.2_{10} to Binary
  - $r_0 = 0.2$
  - $d_{-1} = \lfloor 2 \times 0.2 \rfloor = 0$, $r_{-1} = 2 \times 0.2 - 0 = 0.4$
  - $d_{-2} = \lfloor 2 \times 0.4 \rfloor = 0$, $r_{-2} = 2 \times 0.4 - 0 = 0.8$
  - $d_{-3} = \lfloor 2 \times 0.8 \rfloor = 1$, $r_{-3} = 2 \times 0.8 - 1 = 0.6$
  - $d_{-4} = \lfloor 2 \times 0.6 \rfloor = 1$, $r_{-4} = 2 \times 0.6 - 1 = 0.2$

  - $0.2_{10} = 0.00110011\ldots_2$

Conversion: Binary → Base 2^n

Doesn't work for decimal since base 10 can't be expressed as 2^n

- Group bits in groups of n-bits starting from the '.'
- Replace the n-bits by its equivalent in Base 2^n

- Example: Convert 1101110.110101_2 into hexadecimal
  - 1101110.110101 → 110 | 1110 . 1101 | 01
  - 110 | 1110 . 1101 | 01 → 0110 | 1110 . 1101 | 0100
  - 0110 | 1110 . 1101 | 0100 → 6 | E . D | 4 → $6E.D4_{16}$

Conversion: Base 2^n → Binary

- Replace each digit by its n-bit binary equivalent.
- Ignore leading zeros before MSB and trailing zeros after decimal point.
- Example : Convert $3C.B6_{16}$ into Binary.
  - 3C.B6 → 0011 | 1100 . 1011 | 0110
  - 0011 1100 . 1011 0110 → 111100.1011011_2

Representation of Negative Numbers

- Easy to do on paper.
- $-(106_{10})=-106_{10}$.
- $-(1101010_2)=-1101010_2$.

- Not trivial to represent '-' in electronic hardware (in binary)
  - ON and OFF Present
  - How to differentiate Positive/Negative numbers

Signed and Unsigned Numbers

- Signed Number: One of the bits (the MSB) is used to represent the sign of the number
  - Eg: Assuming 8 bit (one byte) representation
  - Unsigned 21 : 0001 0101
  - Signed +21 : 0001 0101 MSB=0 → Positive Number
  - Signed -21 : 1001 0101 MSB=1 → Negative Number

  - Range: $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$ (-127 to +127 in case of 8 bit)
  - Includes: +0 and -0 (0000 0000 and 1000 0000)
  - Tedious: Needs keeping note of MSB BEFORE operation.

Signed Number Representation: 1s Complement

- A method to more efficiently represent negative numbers, which is more conducive to arithmetic operations.
  - MSB is still the sign bit.
  - Range: $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$
  - To obtain the 1s complement of a binary number, flip all the bits.

  Under 8-bit representation:

  - Eg: $17_{10} = 0001\ 0001_2 \rightarrow 1110\ 1110_2 = -17_{10}$

    $127_{10} = 0111\ 1111_2 \rightarrow 1000\ 0000_2 = -127_{10}$

    $+0_{10} = 0000\ 0000_2 \rightarrow 1111\ 1111_2 = -0_{10}$
  - Still Contains +0 and -0

- To get decimal equivalent from 1s complement:
  - Use position based power summation
  - BUT, MSB is weighed $-(2^{n-1}-1)$ instead of 2^{n-1}

Signed Number Representation: 2s Complement

- More useful than ones complement.
  - MSB is still the sign bit.
  - Range: $-(2^{n-1})$ to $+(2^{n-1}-1)$
  - To obtain the 2s complement of a binary number, obtain (1s complement)+1

  Under 8-bit representation:

  - Eg : $17_{10} = 0001\ 0001_2 \rightarrow 1_2 + 1110\ 1110_2 = 1110\ 1111_2 = -17_{10}$

    $127_{10} = 0111\ 1111_2 \rightarrow 1_2 + 1000\ 0000_2 = 1000\ 0001_2 = -127_{10}$

    $+0_{10} = 0000\ 0000_2 \rightarrow 1_2 + 1111\ 1111_2 = 0000\ 0000_2$
  - Contains only +0 [Thus increasing range by 1]
- To get decimal equivalent from 2s complement:
  - Use position based power summation
  - BUT, MSB is weighed -2^{n-1} instead of 2^{n-1}

1-s and 2-s complement for 4-bit representation

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

From Moris Mano's 'Digital Design' textbook (Chapter 1)

Binary Arithmetic

- Addition:

+	0	1
0	0	1
1	1	10

- Whenever, operations in an N-bit system produces an N+1 bit result, there is a carry bit

	1011001
+	1101101
C	1111--1-
	11000110

Binary Arithmetic

- Subtraction:
  - X-Y → X+(-Y)
  - Add X and (complement of Y)
- 2's Complement
  - In case of carry bit
    - If MSB = 0, then result is positive number ignore carry bit.
    - If MSB = 1, add carry bit to result. Take 2's complement and place a (-) sign in front.

Example: Subtraction (2s Complement)

- n=1101001, m=101110. Obtain n-m and m-n using 2s complement.

- n-m

	0101110	m
	1010001	1's complement of m
+	1	
	1010010	2's complement of m
+	1101001	n
	10111011	Sum (MSB=0), Ignore carry
	0111011	n-m

m-n

	1101001	n
	0010110	1's complement of n
+	1	
	0010111	2's complement of n
+	0101110	m
	1000101	Sum (MSB=1), no carry
	0111010	1's complement
+	1	
	(-)0111011	m-n