CMPT 150: Introduction to Computer Design

A Course Overview

Jeffrey Leung
Simon Fraser University

Summer 2015

Contents

1 Encoding

1.1 Introduction

- *Alphabet:* Finite set of symbols
  - E.g. $\{0, 1\}, \{T, F\}, \{0, 1, 2, ..., 9\}, \{A, B, C, ..., Z\}$
- *Message:* Meaningful sequence of symbols from an alphabet
  - E.g. "CMPT 150", "XYZ-AB2", 0110101 (a binary sequence)
- *Encoding:* Representation of the symbols of one alphabet by sequences of symbols from a second alphabet
  - *Codeword:* Meaningful sequence of symbols which translates into a sequence of symbols of another alphabet
    * *Uniquely decipherable:* Each codeword has only one meaning
    * *Fixed-length:* Each codeword has the same length
      · Number of different codewords of length k where there are j elements in the alphabet is j^k (e.g. 4-bit binary has $2^4 = 16$ different possibilities)

1.2 Binary, Hexadecimal, and BCD

- Codewords: See Table 1
- Binary:
  - Computers interpret voltage levels of cells as 0 or 1
  - *Little Endian notation:* The rightmost bit is numbered as the least significant bit (0) and the leftmost bit is numbered as the most significant bit (n-1 where n = length of sequence)
  - Integer field: Digits to the left of the decimal point
  - Fractional field: Digits to the right of the decimal point
- Hexadecimal:
  - Add additional leading 0s if necessary for conversion
  - E.g. Base 2 to base 16:

$$001111100101011_2 = x_{16}$$

0001	1111	0010	1011
1	F	2	B

$$001111100101011_2 = \text{1F2B}_{16}$$

Table 1: Binary/Decimal/Hexadecimal encoding scheme

Binary:	Decimal:	Hexadecimal:
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

- *Binary Coded Decimal (BCD):* Encoding scheme where each decimal digit is encoded as 4 binary bits

Table 2: BCD encoding scheme

Decimal:	BCD:
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

- E.g. BCD encoding:

$$169_{10} = x_{BCD}$$

$$\begin{array}{c|c|c} 1 & 6 & 9 \\ 0001 & 0110 & 1001 \end{array}$$

$$169_{10} = 0001\ 0110\ 1001_{BCD}$$

- E.g. BCD decoding:

$$1010\ 0001\ 0110_{BCD} = x_{10}$$

$$\begin{array}{c|c|c} 1010 & 0001 & 0110 \\ ? & 1 & 6 \end{array}$$

$$1010\ 0001\ 0110_{BCD} \text{ is meaningless.}$$

1.3 Positional Number System Conversions

1.3.1 Base 10 to Base X

- Integer: Divide the base 10 number by x and write the remainder to the right. The number in base x is the sequence of remainders from bottom to top.
  - E.g. Base 10 to base 2:

$$13_{10} = x_2$$

$$\begin{array}{r|rr} 2 & 13 & \\ \hline 2 & 6 & 1 \\ \hline 2 & 3 & 0 \\ \hline 2 & 1 & 1 \\ \hline & 0 & 1 \end{array}$$

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

  - E.g. Base 10 to base 16:

$$38_{10} = x_{16}$$

$$\begin{array}{r|rr} 16 & 38 & \\ \hline 16 & 2 & 6 \\ \hline & 0 & 2 \end{array}$$

$$38_{10} = 26_{16}$$

- Fractional: Draw a line down from the decimal point. While the right side is greater than 0, multiply the right side by 2 and write the result below. The fraction in binary is the sequence of 0s and 1s on the left side from top to bottom.
  - E.g. Base 10 to base 2:

$$0.625_{10} = x_2$$

$$\begin{array}{r|l} . & 625 \\ 1 & 25 \\ 0 & 5 \\ 1 & 0 \end{array}$$

$$0.625_{10} = 0.101_2$$

- E.g. Base 10 to base 2:

$$18.375_{10} = x_2$$

$$\begin{array}{r|rr} 2 & 18 & \\ \hline 2 & 9 & 0 \\ \hline 2 & 4 & 1 \\ \hline 2 & 2 & 0 \\ \hline 2 & 1 & 0 \\ \hline & 0 & 2 \end{array} \qquad\qquad \begin{array}{r|l} . & 375 \\ 0 & 75 \\ 1 & 5 \\ 1 & 0 \end{array}$$

$$18.375_{10} = 1\ 0010.011_2$$

1.3.2 Base X to Base 10

- Write the position values underneath each digit, then add the position values of all digits with 1s.
- E.g. Base 2 to base 10:

$$10\ 1010\ 0111_2 = x_{10}$$

$$\begin{array}{c|c|c|c|c|c|c|c|c|c} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 2^9 & 2^8 & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

$$2^9 + 2^7 + 2^5 + 2^2 + 2^1 + 2^0 = 679$$

$$10\ 1010\ 0111_2 = 679_{10}$$

- E.g. Base 16 to base 10:

$$26_{16} = x_{10}$$

$$\begin{array}{c|c} 2 & 6 \\ 16^1 & 16^0 \end{array}$$

$$(2 \times 16^1) + (6 \times 16^0) = 32 + 6 = 38$$

$$26_{16} = 38_{10}$$

- E.g. Base 2 to base 10:

$$0.101_2 = x_{10}$$

$$\begin{array}{ccc|c|c} 0 & . & 1 & 0 & 1 \\ 2^0 & & 2^{-1} & 2^{-2} & 2^{-3} \end{array}$$

$$2^{-1} + 2^{-3} = 0.5 + 0.125 + 0.625$$

$$0.101_2 = 0.625_{10}$$

1.4 Signed Arithmetic

- *Signed arithmetic:* Binary encoding which represents both positive and negative numbers
  - Codewords: See Table 3
  - Rules:
    - 0 is always represented
    - For any positive number which is represented, its corresponding negative number must also be represented
  - $2^k - 1$ codewords where k is the number of bits
    - Greatest number which can be represented: $\frac{2^k-1}{2} = 2^{k-1} - 1$
    - Least number which can be represented: $-2^{k-1} - 1$
    - E.g. 4 bits can represent:
      - In signed magnitude encoding: $\{-7, -6, \ldots, -1, 0, 1, \ldots, 6, 7\} = 15$ numbers
      - In 2's complement encoding: $\{-8, -7, -6, \ldots, -1, 0, 1, \ldots, 6, 7\} = 16$ numbers

- *Signed magnitude:* Binary encoding where the most significant bit represents whether the number is positive/negative (0 for positive, 1 for negative) and the other bits represent the value
  - Conversion: Interpret the sign and value separately, then combine them
    - E.g. $-13_{10} = x_{2 \text{ (signed magnitude)}}$

      Sign = - = 1
      Magnitude = $13_{10} = 1101_2$

      $\therefore -13_{10} = 1\ 1101_{2 \text{ (signed magnitude)}}$

    - E.g. $100\ 1001_{2 \text{ (signed magnitude)}} = x_{10}$

      Sign = 1 = -
      Magnitude = $1001_2 = 9_{10}$

      $\therefore 100\ 1001_{2 \text{ (signed magnitude)}} = -9_{10}$

    - E.g. $1000_{2 \text{ (signed magnitude)}} = x_{10}$

      Sign = 1 = -
      Magnitude = 0

Table 3: Binary representations

Codeword	Sign-magnitude decoding	2's complement decoding
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	-0	-8
1001	-1	-7
1010	-2	-6
1011	-3	-5
1100	-4	-4
1101	-5	-3
1110	-6	-2
1111	-7	-1

$\therefore 1000_{2\text{ (signed magnitude)}} = -0_{10}$

2 Digital Systems

- *Digital system:* Electronic circuit which processes discrete signals representing logic values
- Computer design:
  - Instruction set architecture involves the selection, design, and representation of a set of instructions and of some basic data types
  - Construction of a circuit that can:
    * Interpret binary sequences representing instructions
    * Perform computations on binary sequences as directed by instructions
    * Express results as binary encoded data
- *Register:* Component which stores one binary sequence
  - *Memory:* A 1-dimensional array of registers
- *Bus:* Component which transmits one binary sequence
  - *Signal line:* A bus of size 1
- *Registry Transfer Notation:* Convention for naming registers
  - Register names begin with an uppercase letter
Bus names begin with a lowercase letter
  - Bit positions are specified by a number in parenthesis after the name
    * E.g. RNG(MSB), abc(12)
  - *Field:* Sequence of bits within a binary sequence written as NAME(start:end) where the bit positions include the start and end positions
    * E.g. For INST = 0110 1101 0100 0000,
INST(15:8) = 0110 1101
    * E.g. For $R = 10010.011_2$,
R(INT) = R(8:3) and R(FRAC) = R(2:0)

3 Assembly Programming

- x

4 Boolean Algebra

4.1 Introduction

- A specific input creates a specific output

- Values can be 0 (false) or 1 (true)
- Boolean expressions:
  - Evaluate to 0 or 1
  - Function table: Representation of all possible inputs and their corresponding outputs displayed in a table

Table 4: Example of a function table

I_2	I_1	Sum
0	0	0
0	1	1
0	2	2
$\vdots$	$\vdots$	$\vdots$
9	8	17
9	9	18