

# Chapter 2

## Data Representation

### 2.1 Positional Number Systems

Common representations:

- Decimal  $(0, 1, 2, \dots, 9)$
- Binary  $(0, 1)$
- Hexadecimal  $(0, 1, 2, 3, \dots, 15)$
- Octal  $(0, 1, 2, \dots, 7)$

A radix- $r$  number is encoded as a digital vector of  $n + k$  digits. Each digit is weighted in terms of its position in the vector and a radix or base number  $r$ . Let  $A$  be vector defined as follows:

$$A = a_{n-1}a_{n-2}\dots a_1a_0.a_{-1}a_{-2}\dots a_{-k}$$

where each component  $a_i$  for  $-k \leq i \leq n - 1$  is referred to as the  $i$ th digit of the vector  $A$ , and  $r$  denotes the radix. The weight assigned to the  $i$ th element of  $A$  is  $r^i$ . The first  $n$  digits form the integer portion of the number  $A$  and the remaining  $k$  digits form the fraction portion of the  $A$ . The overall value  $V$  represented is obtained as follows:

$$\begin{aligned}
 V &= a_{n-1}r^{n-1} + \dots + a_1r^1 + a_0r^0.a_{-1}r^{-1} \dots a_{-k}r^{-k} \\
 (2.1) \qquad &= \sum_{i=-k}^{n-1} a_i r^i
 \end{aligned}$$

Note that  $V$  is in turn the decimal ( $r = 10$ ) representation of  $A$ .

Examples:

1. For  $r = 10$  and  $A = 95243.25$  then

$$V = 9 \times 10^4 + 5 \times 10^3 + 2 \times 10^2 + 4 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$$

2. For  $r = 2$  and  $A = 1011010.101$  then

$$\begin{aligned}
 V &= 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 \\
 &\quad + 1 \times 2^3 + 1 \times 2^1 + 0 \times 2^0 \\
 (2.2) \qquad &\quad + 1 \times 2^{-1} + 1 \times 2^{-3} \\
 &= 64 + 16 + 8 + 2 + .5 + .125 = 90.625
 \end{aligned}$$

3. For  $r = 16$  and  $A = 5A7$

$$\begin{aligned}
 V &= 5 \times 16^2 + 10 \times 16^1 + 7 \times 16^0 \\
 &= 1280 + 160 + 7 = 1447
 \end{aligned}$$

4. For  $r = 8$  and  $A = 2647$  then:

$$\begin{aligned}
 V &= 2 \times 8^3 + 6 \times 8^2 + 4 \times 8^1 + 7 \times 8^0 \\
 &= 1024 + 384 + 32 + 7 = 1447
 \end{aligned}$$

Interpretation of notation:

$$\begin{aligned}
 (1011010)_2 &= (90)_{10} \\
 (5A7)_{16} &= (2647)_8 \\
 &= (1447)_{10}
 \end{aligned}$$

In the context of assembly language note that:

$$\begin{aligned}(5A7)_{16} &= 5A7h \\ (1011010)_2 &= 1011010b\end{aligned}$$

Note also that a hex number that starts with a letter from A to F must be preceded by a 0. Example:  $AFC Dh = 0AFC Dh$ . Otherwise, the assembler will process it as a string of characters.

## 2.2 Conversions

### 2.2.1 X-to-decimal (shown in previous examples)

### 2.2.2 decimal-to-X

*Rule:* divide by the radix  $r$  and keep track of remainder

- Decimal-to-binary

Example:  $(90)_{10} = (?)_2$

$$\begin{array}{rcl} 90: & 45 & 0 \\ & 22 & 1 \\ & 11 & 0 \\ & 5 & 1 \\ & 2 & 1 \\ & 1 & 0 \\ & 0 & 1 \end{array}$$

and  $(90)_{10} = (1011010)_2$

*Rule:* take remainders in reverse order

- Decimal-to-hex

Example:  $(6841)_{10} = (?)_{16}$

$$\begin{array}{rcl} 6841: & 427 & 9 \\ & 26 & 11 \text{ (B)} \\ & 1 & 10 \text{ (A)} \\ & 0 & 1 \end{array}$$

and  $(6841)_{10} = (1AB9)_{16}$

- Decimal-to-octal

Example:  $(1447)_{10} = (?)_8$

$$\begin{array}{rcl} 1447: & 180 & 7 \\ & 22 & 4 \\ & 2 & 6 \\ & 0 & 2 \end{array}$$

and  $(1447)_{10} = (2647)_8$

## 2.3 Other conversions

- Hex-to-binary: convert each digit in the hex representation to its binary representation. Example:

$$(0EAC)_{16=2^4} = (0000\ 1110\ 1010\ 1100)_2$$

- Binary-to-hex: from right to left divide binary digits in groups of four. Example:

$$(101\ 1010)_2 = (5A)_{16}$$

- Octal-to-binary: convert each digit in the octal representation to its binary representation. Example:

$$(2647)_{8=2^3} = (010\ 110\ 100\ 111)_2$$

- Binary-to-octal: from right to left divide binary digits in groups of three. Example:

$$(010\ 110\ 100\ 111)_2 = (2647)_{8=2^3}$$

- Octal-to-hex: Do octal-to-binary-to-hex. Example:

$$(2647)_8 = (010\ 110\ 100\ 111)_2 = (5A7)_{16}$$

## 2.4 ASCII (American Standard Code for Information Interchange)

ASCII characters are represented by a unique numeric value (code)

- 7-bit codes — up to  $2^7 = 128$  characters
- 0 — 31 (00h — 1Fh) — are control (non-printing) characters
- 32 (20h) — blank character
- 33 — 126 (21h — 7Eh) — printing characters
- 127 (7Fh) — delete character
- 48 — 57 (30h — 39h) — digits (0 to 9)

Note that: "0" = 48 thus, the ascii code for a digit "n" is given as follows: "n" = 48 + n

Extended IBM ascii characters

- 8-bit codes — up to  $2^8 = 256$  characters
- includes foreign characters
- graphics symbols
- math symbols

## 2.5 The Unicode Standard

Ref: <http://www.unicode.org/standard/principles.html>

The Unicode Standard is the universal character encoding standard used for representation of text for computer processing. The design of Unicode is based on the simplicity and consistency of ASCII, but goes far beyond ASCII's limited ability to encode only the Latin alphabet. The Unicode Standard provides the capacity to encode all of the characters used for the written languages of the world. To keep character coding simple and efficient, the Unicode Standard assigns each character a unique numeric value and name. The Unicode Standard defines codes for characters used in all the major languages written today. Scripts include the

European alphabetic scripts, Middle Eastern right-to-left scripts, and many scripts of Asia.

The Unicode Standard further includes punctuation marks, diacritics, mathematical symbols, technical symbols, arrows, dingbats, etc. It provides codes for diacritics, which are modifying character marks such as the tilde ( ), that are used in conjunction with base characters to represent accented letters (, for example). In all, the Unicode Standard, Version 3.2 provides codes for 95,221 characters from the world's alphabets, ideograph sets, and symbol collections. The original goal was to use a single 16-bit encoding that provides code points for more than 65,000 characters. While 65,000 characters are sufficient for encoding most of the many thousands of characters used in major languages of the world, the Unicode standard now supports three encoding forms that use a common repertoire of characters but allow for encoding as many as a million more characters. This is sufficient for all known character encoding requirements, including full coverage of all historic scripts of the world, as well as common notational systems.

## 2.6 Signed Radix Numbers

In general a signed radix number can be represented (using the vector  $A$ ) as follows:

$$A = (a_{n-1}a_{n-2} \dots a_1a_0)_r$$

where  $a_{n-1}$  is used to represent the sign and takes the value:

$$a_{n-1} = \begin{cases} 0 & \text{if } V \geq 0; \\ r-1 & \text{if } V < 0, \end{cases}$$

Thus, for  $r = 2$ ,  $a_{n-1}$  takes the value 0 to represent positive values and 1 for negative values. The remaining digits in  $A$ , represent the true magnitude of  $A$  or the magnitude in a complemented form. In a positional number system, there are three conventional ways to represent positive and negative numbers: *sign magnitude*, *diminished-radix complement* and *radix complement*.

### 2.6.1 Signed magnitude

Positive integers are represented as follows:

$$A = (0a_{n-2}a_{n-3} \dots a_1a_0)_r$$

with values  $V$  in the range  $0 \leq V \leq r^{n-1} - 1$

Negative numbers are represented as follows:

$$-A = [(r-1)a_{n-2} \dots a_1 a_0]_r$$

with values  $V$  in the range:  $-(r^{n-1} - 1) \leq V \leq 0$

Note that, since the most significant digit is used for the sign, then the magnitude of  $A$  takes values in the range  $0 \leq |A| \leq r^{n-1} - 1$ .

Thus, positive and negative representations differ only in the sign.

Problems:

1. the value of 0 is represented as +0 and -0.
2. Signs must be compared during "+" and "-" operations.

### 2.6.2 Diminished-radix Complement

Positive integer representation:

$$A = (0a_{n-2} \dots a_1 a_0)_r$$

with values  $V$  represented in the range  $0 \leq V \leq r^{n-1} - 1$

Given the positive representation of a number, the same absolute value can be represented as negative number by:

$$\bar{A} = [(r-1)\bar{a}_{n-2} \dots \bar{a}_1 \bar{a}_0]_r$$

where  $\bar{a}_i = (r-1) - a_i$

The values represented are also in the range  $-(r^{n-1} - 1) \leq V \leq 0$

To obtain the representation of negative numbers suffices to complement the absolute value of the given number.

Problem: observe that there is still a dual representation of zero

### 2.6.3 Radix Complement

Representation of positive integers:

$$A = (0a_{n-2} \dots a_1 a_0)_r$$

with values  $V$  in the range  $0 \leq V \leq r^{n-1} - 1$

Given  $A = (0a_{n-2} \dots a_1 a_0)_r$  a representation of negative integers is obtained as:

$$(\bar{A})_{+1} = \{[(r-1)\bar{a}_{n-2} \dots \bar{a}_1 \bar{a}_0] + 1\}_r$$

with values in the range  $-(r^{n-1}) \leq V < 0$

The procedure to obtain a radix-complement representation of negative integers can be divided into three steps:

1. Obtain the absolute value of the number given,
2. Diminished radix complement each digit, i.e.,  $a_i = (r-1) - a_i$ ,
3. Add one to the least significant digit.

Example: -018135 ( $n = 6, r = 10$ )

1. 018135 (absolute value)
2. 981864 (9's complement)
3. 981865 (add 1)

Simplified rule:

1. From right to left obtain the  $r$ 's-complement of first non-zero digit,
2.  $(r-1)$ -complement the remaining digits

Example 1 ( $r=2$ ):

$$\begin{aligned} 50 &= 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \\ -50 &= 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \end{aligned}$$

Example 2 ( $r=10$ ):

$$\begin{aligned} &0 \ 1 \ 8 \ 9 \ 3 \ 0 \ 0 \\ &9 \ 8 \ 1 \ 0 \ 7 \ 0 \ 0 \end{aligned}$$



### 2.6.4 Two's complement

Advantages:

1. Only one representation of zero
2. Binary ( $r=2$ ) arithmetic operations are easier
3. Hardware implementation is also easier

Example:

$$\begin{aligned}
 (51)_{10} &= 00110011 \\
 +(-51)_{10} &= 11001101(2's \text{ complement}) \\
 &= 00000000
 \end{aligned}$$

Disadvantages: the most negative value  $-(r^{n-1})$  has no positive counterpart. For example, for  $r = 2$  and  $n = 8$ , the range of values:

$$-128 (1000\ 0000) \leq V \leq 127 (0111\ 1111)$$

for  $n = 16$

$$-32768 \leq V \leq 32767$$

Note that with  $r=2$ , negative values can be obtained directly from the binary representation considering the negative contribution of the sign bit. For example using equation 2.1, the value of -51 is obtained from its binary representation as follows:

$$V = \sum_{i=0} -2^7 + 2^6 + 2^3 + 2^2 + 2^0 = -128 + 64 + 8 + 4 + 1 = -51$$

#### Exercises

1. Obtain the decimal representation of the following unsigned binary integers:

(a) 1000 1111 0101 1110

(b) 0010 1110

(c) 1100 1011 0101 1011

2. What is the minimum number of binary bits required to represent each of the following unsigned integers?
  - (a) 3098
  - (b) 512
  - (c) 65534
3. Obtain the binary representation of the decimal values in the previous exercise with a number of digits  $n = 16$
4. Obtain the hexadecimal representation of the following binary numbers:
  - (a) 0110 1110 1100 0010
  - (b) 1111 1010 0011 1101
  - (c) 1111 1110 1011 1110
5. Obtain the binary representation of the following numbers:
  - (a)  $(B687A3C1)_{16}$
  - (b)  $(3112301223231331)_4$
  - (c)  $(5645231671)_8$
6. Obtain a 16-bit hexadecimal representation of each of the following signed decimal integers:
  - (a) -455
  - (b) -62
  - (c) +138
7. The following 16-bit hexadecimal values represent signed integers. Convert to decimal:
  - (a) F789h
  - (b) 7123h
  - (c) 83BAh
8. Obtain the 16-bit binary 2's complement representation of the following signed numbers:
  - (a) -127

- (b) -48
  - (c) +12345
  - (d) -5
9. What is the largest value you can represent using a 256-bit unsigned integer?
10. What is the smallest and largest number you can represent unsigned a 256-bit number in 2's complement.