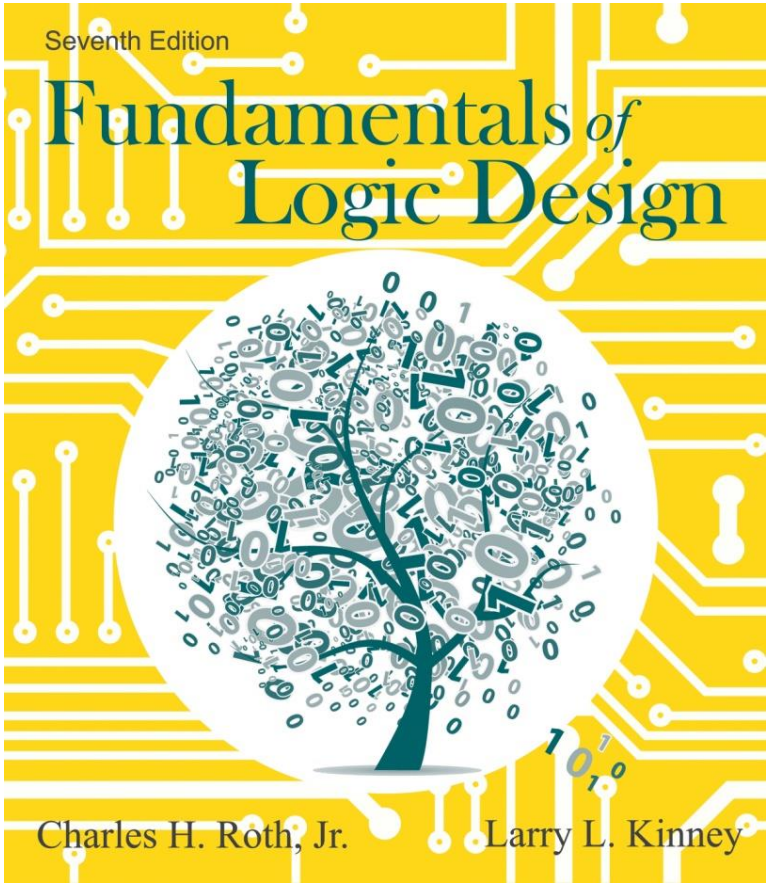


UNIT 1

Introduction Number Systems and Conversion

This chapter in the book includes:

Objectives
Study Guide
1.1 Digital Systems and Switching Circuits
1.2 Number Systems and Conversion
1.3 Binary Arithmetic
1.4 Representation of Negative Numbers
1.5 Binary Codes
Problems

Learning Objectives

Introduction:

- To be able to explain the difference between analog and digital systems, as well as why digital systems are capable of greater accuracy
- To be able to understand the difference between combinational and sequential circuits
- To be able to explain why two-valued signals and binary numbers are commonly used in digital systems

Learning Objectives

Number Systems and Conversion:

To be able to solve the following types of problems:

- Given a positive integer, fraction, or mixed number in any base (2 through 16); convert to any other base. Justify the procedure using a power series expansion for the number.
- Add, subtract, multiply, and divide positive binary numbers. Explain the addition and subtraction process in terms of carries and borrows.
- Write negative binary numbers in sign and magnitude, 1's complement, and 2's complement forms. Add signed binary numbers using 1's complement and 2's complement arithmetic. Justify the methods used and identify overflows.
- Represent a decimal number in binary-coded-decimal (BCD), 6-3-1-1 code, excess-3 code, etc. Given a set of weights, construct a weighted code.

Digital Systems and Switching Circuits

Digital and Analog Systems:

- Digital systems are used extensively in computation and data processing, control systems, communications, and measurement.
- In a **digital system**, physical quantities can only assume discrete values.
- In an **analog system**, physical quantities or signals can vary continuously over a specified range.
- In many cases digital systems can be designed so that for a given input, the output is exactly correct.
- The output of an analog system might have an error ranging from a fraction of one percent to a few percents because these systems do not work with discrete quantities.

Digital Systems and Switching Circuits

Design of digital systems:

- **System design**: breaking the overall system into subsystems and specifying the characteristics of each subsystem.
- **Logic design:** determining how to interconnect basic logic building blocks to perform a specific function.
- **Circuit design:** specifying the interconnection of specific hardware components to form a gate, flip-flop or other logic building block.

Digital Systems and Switching Circuits

Switching circuits:

- A **switching circuit** has one or more inputs and one or more outputs which take on discrete values.
- In **combinational circuits**, output values of circuit only depend on the present values of inputs, not past.
- In **sequential circuits**, output values of circuit depend on both past and present input values. For this reason, sequential circuits are said to have "memory", because they must remember past sequence of inputs.

Digital Systems and Switching Circuits

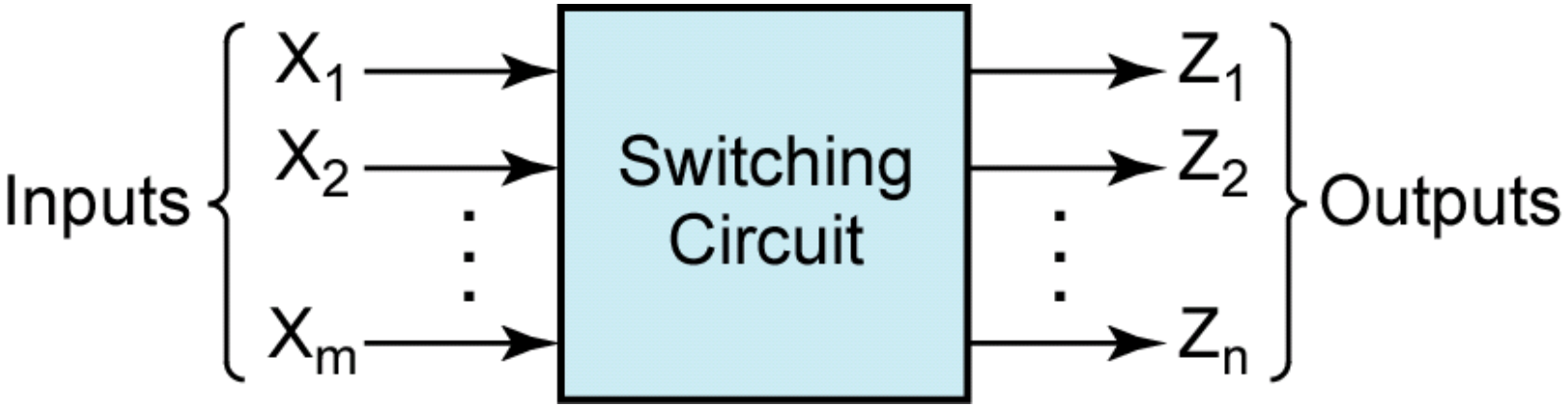


Figure 1-1: Switching circuit

Digital Systems and Switching Circuits

Building Blocks of Switching Circuits:

- Basic building block of combinational circuits: **logic gate**
- Basic building block of sequential circuits: **flip-flop**
- Switching devices used in digital systems are usually two-state devices, meaning the output can only assume one of two discrete values, i.e. relays, diodes and transistors.
- Because the outputs of most switching devices assume only two different values, it is natural to use **binary** numbers internally in digital systems.

Number Systems and Conversion

Explanation of decimal and binary numbers:

- When we write decimal (base 10) numbers, we use a positional notation; each digit is multiplied by an appropriate power of 10 depending on its position in the number.

$$953.78_{10} = 9 \times 10^2 + 5 \times 10^1 + 3 \times 10^0 + 7 \times 10^{-1} + 8 \times 10^{-2}$$

- When we write binary (base 2) numbers, each digit is multiplied by an appropriate power of 2 on its position in the number.

$$1011.11_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$
$$= 8 + 0 + 2 + 1 + \tfrac{1}{2} + \tfrac{1}{4} = 11\tfrac{3}{4} = 11.75_{10}$$

Number Systems and Conversion

Power series expansion in any number system of base R:

Any positive integer R ($R > 1$) can be chosen as the *radix* or *base* of a number system. If the base is R, then R digits $(0, 1, \ldots, R-1)$ are used. For example, if $R = 8$, then the required digits are 0,1,2,3,4,5,6, and 7. A number written in positional notation can be expanded in a power series in R. For example,

$$\begin{aligned} N &= (a_4 a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3})_R \\ &= a_4 \times R^4 + a_3 \times R^3 + a_2 \times R^2 + a_1 \times R^1 + a_0 \times R^0 \\ &\quad + a_{-1} \times R^{-1} + a_{-2} \times R^{-2} + a_{-3} \times R^{-3} \end{aligned}$$

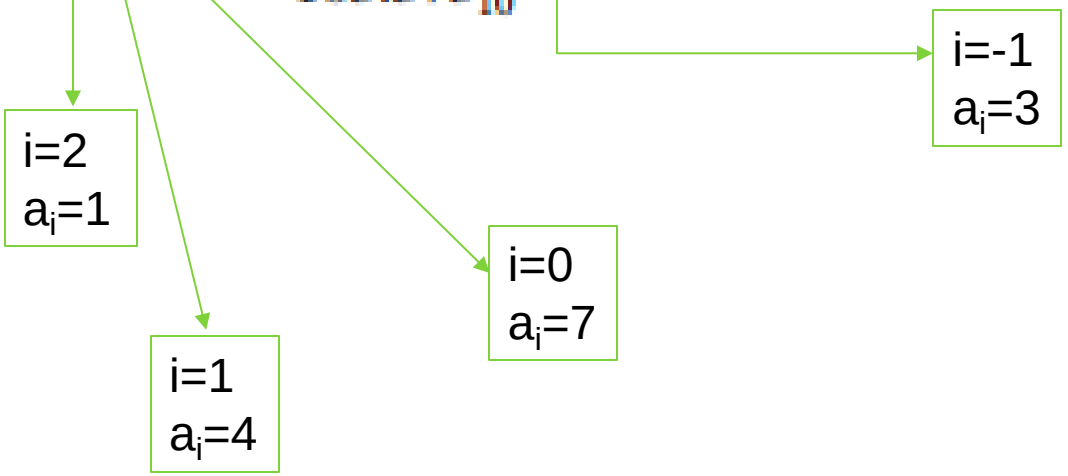
where a_i is the coefficient of R^i and $0 \leq a_i \leq R - 1$.

Number Systems and Conversion

Example:

Using the previous slide, the following conversion can be made from base 8 to base 10.

$$147.3_8 = 1 \times 8^2 + 4 \times 8^1 + 7 \times 8^0 + 3 \times 8^{-1} = 64 + 32 + 7 + \frac{3}{8}$$
$$= 103.375_{10}$$



Number Systems and Conversion

Conversion to other bases:

The power series expansion can be used to convert to any base. For example, converting 147_{10} to base 3 would be written as

$$147_{10} = 1 \times (101)^2 + (11) \times (101)^1 + (21) \times (101)^0$$

where all the numbers on the right-hand side are base 3 numbers. (*Note:* In base 3, 10 is 101, 7 is 21, etc.) To complete the conversion, base 3 arithmetic would be used.

Number Systems and Conversion

Bases greater than 10:

For bases greater than 10, more than 10 symbols are needed to represent the digits.

In this case, letters are usually used to represent digits greater than 9. For example, in hexadecimal (base 16), A represents 10_{10}, B represents 11_{10}, C represents 12_{10}, D represents 13_{10}, E represents 14_{10}, and F represents 15_{10}. Thus,

$$A2F_{16} = 10 \times 16^2 + 2 \times 16^1 + 15 \times 16^0 = 2560 + 32 + 15 = 2607_{10}$$

Number Systems and Conversion

Conversion of a decimal integer to base R using the division method:

The base R equivalent of a decimal integer N is as follows (from previous slides):

$$N = (a_n a_{n-1} \cdots a_2 a_1 a_0)_R = a_n R^n + a_{n-1} R^{n-1} + \cdots + a_2 R^2 + a_1 R^1 + a_0$$

If we divide N by R, the remainder is a_0:

$$\frac{N}{R} = a_n R^{n-1} + a_{n-1} R^{n-2} + \cdots + a_2 R^1 + a_1 = Q_1, \text{ remainder } a_0$$

Then we divide the quotient Q_1 by R:

$$\frac{Q_1}{R} = a_n R^{n-2} + a_{n-1} R^{n-3} + \cdots + a_3 R^1 + a_2 = Q_2, \text{ remainder } a_1$$

Number Systems and Conversion

Conversion of a decimal integer to base R using the division method (continued):

Next we divide Q_2 by R:

$$\frac{Q_2}{R} = a_n R^{n-3} + a_{n-1} R^{n-4} + \cdots + a_3 = Q_3, \text{ remainder } a_2$$

This process is continued until we finally obtain a_n. Note that the remainder obtained at each division step is one of the desired digits and the least significant digit is obtained first.

Number Systems and Conversion

Example 1: Conversion of decimal integer to binary

Example Convert 53_{10} to binary.

$2 \overline{)53}$		
$2 \overline{)26}$	rem. $= 1 = a_0$	
$2 \overline{)13}$	rem. $= 0 = a_1$	
$2 \overline{)6}$	rem. $= 1 = a_2$	$53_{10} = 110101_2$
$2 \overline{)3}$	rem. $= 0 = a_3$	
$2 \overline{)1}$	rem. $= 1 = a_4$	
0	rem. $= 1 = a_5$	

Number Systems and Conversion

Conversion of a decimal fraction F to base R using successive multiplications:

$$F = (.a_{-1}a_{-2}a_{-3}\cdots a_{-m})_R = a_{-1}R^{-1} + a_{-2}R^{-2} + a_{-3}R^{-3} + \cdots + a_{-m}R^{-m}$$

Multiplying by R yields

$$FR = a_{-1} + a_{-2}R^{-1} + a_{-3}R^{-2} + \cdots + a_{-m}R^{-m+1} = a_{-1} + F_1$$

where F_1 represents the fractional part of the result and a_{-1} is the integer part. Multiplying F_1 by R yields

$$F_1R = a_{-2} + a_{-3}R^{-1} + \cdots + a_{-m}R^{-m+2} = a_{-2} + F_2$$

Number Systems and Conversion

Conversion of a decimal fraction F to base R using successive multiplications (continued):

Next, we multiply F_2 by R:

$$F_2R = a_{-3} + \cdots + a_{-m}R^{-m+3} = a_{-3} + F_3$$

This process is continued until we have obtained a sufficient number of digits. Note that the integer part obtained at each step is one of the desired digits and the most significant digit is obtained first.

Number Systems and Conversion

Example 2: Conversion of a decimal fraction F to base R using successive multiplications

Example

Convert 0.625_{10} to binary.

$F = .625$	$F_1 = .250$	$F_2 = .500$	
$\times\ 2$	$\times\ 2$	$\times\ 2$	$.625_{10} = .101_2$
1.250	0.500	1.000	
$(a_{-1} = 1)$	$(a_{-2} = 0)$	$(a_{-3} = 1)$	

Number Systems and Conversion

Example 3: Conversion of a decimal fraction F to base R using successive multiplications

Example Convert 0.7_{10} to binary.

$$
\begin{array}{r}
.7 \\
\underline{\quad 2} \\
(1).4 \\
\underline{\quad 2} \\
(0).8 \\
\underline{\quad 2} \\
(1).6 \\
\underline{\quad 2} \\
(1).2 \\
\underline{\quad 2} \\
(0).4 \quad \longleftarrow \text{ process starts repeating here because 0.4 was previously obtained} \\
\underline{\quad 2} \\
(0).8
\end{array}
$$

$0.7_{10} = 0.1\underline{0110}\,\underline{0110}\,\underline{0110}\ \ldots_2$

Number Systems and Conversion

Example 4: Conversion between two bases other than decimal

Example Convert 231.3_4 to base 7.

$$231.3_4 = 2 \times 16 + 3 \times 4 + 1 + \tfrac{3}{4} = 45.75_{10}$$

$7 \overline{)45}$		.75
$7 \overline{)6}$	rem. 3	7
0	rem. 6	(5).25
		7
		(1).75
		7
		(5).25
		7
		(1).75

$45.75_{10} = 63.5151\ldots_7$

Number Systems and Conversion

Conversion from binary to hexadecimal and binary to octal:

Conversion from binary to hexadecimal (and conversely) can be done by inspection because each hexadecimal digit corresponds to exactly four binary digits (bits).

$$1001101.010111_2 = \underbrace{0100}_{4}\ \underbrace{1101}_{D} . \underbrace{0101}_{5}\ \underbrace{1100}_{C} = 4D.5C_{16}$$

A similar conversion can be done from binary to octal, base 8 (and conversely), except each octal digit corresponds to three binary digits, instead of four.

$100101101011010_2 = 100\ \ 101\ \ 101\ \ 011\ \ 010 = 45532_8$

Binary Arithmetic

Introduction and Binary Addition:

For simplicity, arithmetic in digital systems is performed in binary (base 2).

Addition table for binary:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 0 \quad \text{and carry 1 to the next column}$$

Carrying 1 to a column is equivalent to adding 1 to that column.

Binary Arithmetic

Example 5: Binary addition

Example Add 13_{10} and 11_{10} in binary.

$$\begin{array}{rl} & 1111 \leftarrow \text{carries} \\ 13_{10} = & 1101 \\ 11_{10} = & \underline{1011} \\ & 11000 = 24_{10} \end{array}$$

Binary Arithmetic

Binary Subtraction:

Subtraction table for binary:

The subtraction table for binary numbers is

$$0 - 0 = 0$$
$$0 - 1 = 1 \quad \text{and borrow 1 from the next column}$$
$$1 - 0 = 1$$
$$1 - 1 = 0$$

Borrowing 1 from a column is equivalent to subtracting 1 from that column.

Binary Arithmetic

Example 6: Subtractions in Binary

Examples of Binary Subtraction

(a)
```
  1 <-- (indicates
 11101   a borrrow
-10011   from the
 1010    3rd column)
```

(b)
```
 1111 <-- borrows
 10000
-   11
 1101
```

(c)
```
  111 <-- borrows
 111001
-  1011
 101110
```

Notice how in (b), the borrow propagates from column to column.

Binary Arithmetic

Binary Multiplication:

Multiplication table for binary:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Binary Arithmetic

Example 7: Binary Multiplication

13_{10} by 11_{10} in binary:

$$
\begin{array}{r}
1101 \\
1011 \\
\hline
1101 \\
1101 \\
0000 \\
1101 \\
\hline
10001111
\end{array} = 143_{10}
$$

Binary Arithmetic

When doing binary multiplication, a common way to avoid carries greater than 1 is to add in the partial products one at a time as illustrated by the following example:

```
      1111      multiplicand
      1101      multiplier
      1111      1st partial product
     0000       2nd partial product
    (01111)     sum of first two partial products
     1111         3rd partial product
   (1001011)    sum after adding 3rd partial
product
      1111      4th partial product
  11000011      final product (sum after adding 4th
                                 partial product)
```

Binary Arithmetic

Binary Division:

- Binary division is similar to decimal division, except it is much easier because the only two possible quotient digits are 0 and 1.
- We start division by comparing the divisor with the upper bits of the dividend.
- If we cannot subtract without getting a negative result, we move one place to the right and try again.
- If we can subtract, we place a 1 for the quotient above the number we subtracted from and append the next dividend bit to the end of the difference and repeat this process with this modified difference until we run out of bits in the dividend.

Binary Arithmetic

Example 8: Binary Division

145_{10} by 11_{10} in binary

```
          1101
    ┌──────────
1011│ 10010001
     1011
     ----
      1110
      1011
      ----
      1101
      1011
      ----
        10
```

The quotient is 1101 with a remainder of 10.

Representation of Negative Numbers

Common methods of representing both positive and negative numbers are:

- Sign and magnitude
- 2's complement
- 1's complement

In each of these methods, the leftmost bit of a number is 0 for positive numbers and 1 for negative numbers.

Representation of Negative Numbers

3 Systems for representing negative numbers in binary- Overview:

Sign & Magnitude: Most significant bit is the sign

Ex: $-5_{10} = 1101_2$

2's Complement: $N^* = 2^n - N$

Ex: $-5_{10} = 2^4 - 5 = 16 - 5 = 11_{10} = 1011_2$

1's Complement: $= (2^n - 1) - N$

Ex: $-5_{10} = (2^4 - 1) - 5 = 16 - 1 - 5 = 10_{10} = 1010_2$

Representation of Negative Numbers

Sign & Magnitude:

Most significant bit is the sign

Ex: $-5_{10} = 1101_2$

TABLE 1-1
Signed Binary Integers (word length: $n = 4$)

			Negative Integers		
$+N$	Positive Integers (all systems)	$-N$	Sign and Magnitude	2's Complement N^*	1's Complement $\bar{N}$
+0	0000	−0	1000	——	1111
+1	0001	−1	1001	1111	1110
+2	0010	−2	1010	1110	1101
+3	0011	−3	1011	1101	1100
+4	0100	−4	1100	1100	1011
+5	0101	−5	1101	1011	1010
+6	0110	−6	1110	1010	1001
+7	0111	−7	1111	1001	1000
		−8	——	1000	——

Representation of Negative Numbers

2's compliment:

- Positive number, N, is represented by a 0 followed by the magnitude of N as in the sign and magnitude system
- For negative numbers, -N, 2's compliment is represented by N*, as follows:

$$N^* = 2^n - N$$

Where the word length is n bits.

Representation of Negative Numbers

Example 9: 2's Compliment Addition

1. Addition of two positive numbers, sum $< 2^{n-1}$

+3	0011	
+4	0100	
+7	0111	(correct answer)

2. Addition of two positive numbers, sum $\geq 2^{n-1}$

+5	0101	
+6	0110	
	1011	⟵ wrong answer because of overflow (+11 requires 5 bits including sign)

Representation of Negative Numbers

Example 9: 2's Compliment Addition (continued)

3. Addition of positive and negative numbers (negative number has greater magnitude)

+5	0101	
−6	1010	
−1	1111	(correct answer)

4. Same as case 3 except positive number has greater magnitude

−5	1011	
+6	0110	
+1	(1)0001	⟵ correct answer when the carry from the sign bit is ignored (this is *not* an overflow)

Representation of Negative Numbers

Example 9: 2's compliment Addition (continued)

5. Addition of two negative numbers, $|\text{sum}| \leq 2^{n-1}$

-3	1101	
-4	1100	
-7	(1)1001	⟵ correct answer when the last carry is ignored (this is *not* an overflow)

6. Addition of two negative numbers, $|\text{sum}| > 2^{n-1}$

-5	1011	
-6	1010	
	(1)0101	⟵ wrong answer because of overflow (-11 requires 5 bits including sign)

Representation of Negative Numbers

1's compliment:

$$\overline{N} = (2^n - 1) - N$$

Ex: $-5_{10} = (2^4 - 1) - 5 = 16 - 1 - 5 = 10_{10} = 1010_2$

End around carry: In one's compliment, the last carry is not discarded as it is in 2's compliment, but rather added to the n-bit sum in the position furthest to the right.

Representation of Negative Numbers

Example 10: 1's compliment Addition

3. Addition of positive and negative numbers (negative number with greater magnitude)

+5	0101	
−6	1001	
−1	1110	(correct answer)

4. Same as case 3 except positive number has greater magnitude

−5	1010	
+6	0110	
(1)	0000	
└→	1	(end-around carry)
	0001	(correct answer, *no* overflow)

Representation of Negative Numbers

Example 10: 1's compliment Addition (continued)

5. Addition of two negative numbers, $|\text{sum}| < 2^{n-1}$

-3	1100	
-4	1011	
(1)	0111	
$\hookrightarrow$	1	(end-around carry)
	1000	(correct answer, *no* overflow)

6. Addition of two negative numbers, $|\text{sum}| \geq 2^{n-1}$

-5	1010	
-6	1001	
(1)	0011	
$\hookrightarrow$	1	(end-around carry)
	0100	(wrong answer because of overflow)

Representation of Negative Numbers

Example 10: 1's compliment Addition (continued)

1. Add −11 and −20 in 1's complement.

$+11 = 00001011$ $+20 = 00010100$

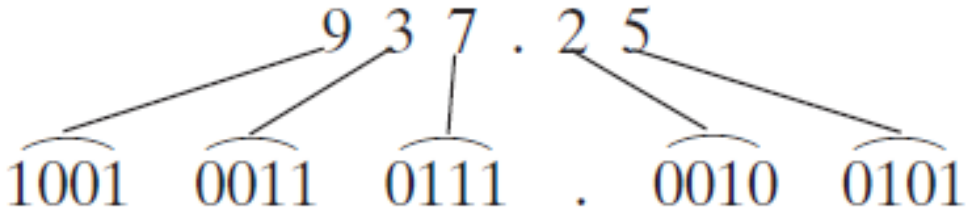
taking the bit-by-bit complement,

−11 is represented by 11110100 and −20 by 11101011

$$\begin{array}{rl} 11110100 & (-11) \\ \underline{11101011} & \underline{+(-20)} \\ (1)\,11011111 & \\ \quad \longrightarrow\ 1 & \text{(end-around carry)} \\ \hline 11100000 = -31 & \end{array}$$

Binary Codes

- Although most large computers work internally with binary numbers, the input-output equipment generally uses decimal numbers.
- Because most logic circuits only accept two-valued signals, the decimal numbers must be coded in terms of binary signals.
- In the simplest form of binary code, each decimal digit is replaced by its binary equivalent. For example, 937.25 is represented by:



Binary Codes

Decimal Digit	8-4-2-1 Code (BCD)	6-3-1-1 Code	Excess-3 Code	2-out-of-5 Code	Gray Code
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011
8	1000	1011	1011	10100	1001
9	1001	1100	1100	11000	1000

Table 1–2. Binary Codes for Decimal Digits

Binary Codes

TABLE 1-3 ASCII Code

Character	ASCII Code A_6	A_5	A_4	A_3	A_2	A_1	A_0	Character	ASCII Code A_6	A_5	A_4	A_3	A_2	A_1	A_0	Character	ASCII Code A_6	A_5	A_4	A_3	A_2	A_1	A_0
space	0	1	0	0	0	0	0	@	1	0	0	0	0	0	0	'	1	1	0	0	0	0	0
!	0	1	0	0	0	0	1	A	1	0	0	0	0	0	1	a	1	1	0	0	0	0	1
"	0	1	0	0	0	1	0	B	1	0	0	0	0	1	0	b	1	1	0	0	0	1	0
#	0	1	0	0	0	1	1	C	1	0	0	0	0	1	1	c	1	1	0	0	0	1	1
$	0	1	0	0	1	0	0	D	1	0	0	0	1	0	0	d	1	1	0	0	1	0	0
%	0	1	0	0	1	0	1	E	1	0	0	0	1	0	1	e	1	1	0	0	1	0	1
&	0	1	0	0	1	1	0	F	1	0	0	0	1	1	0	f	1	1	0	0	1	1	0
'	0	1	0	0	1	1	1	G	1	0	0	0	1	1	1	g	1	1	0	0	1	1	1
(	0	1	0	1	0	0	0	H	1	0	0	1	0	0	0	h	1	1	0	1	0	0	0
)	0	1	0	1	0	0	1	I	1	0	0	1	0	0	1	i	1	1	0	1	0	0	1
*	0	1	0	1	0	1	0	J	1	0	0	1	0	1	0	j	1	1	0	1	0	1	0
+	0	1	0	1	0	1	1	K	1	0	0	1	0	1	1	k	1	1	0	1	0	1	1
,	0	1	0	1	1	0	0	L	1	0	0	1	1	0	0	l	1	1	0	1	1	0	0
-	0	1	0	1	1	0	1	M	1	0	0	1	1	0	1	m	1	1	0	1	1	0	1
.	0	1	0	1	1	1	0	N	1	0	0	1	1	1	0	n	1	1	0	1	1	1	0
/	0	1	0	1	1	1	1	O	1	0	0	1	1	1	1	o	1	1	0	1	1	1	1
0	0	1	1	0	0	0	0	P	1	0	1	0	0	0	0	p	1	1	1	0	0	0	0
1	0	1	1	0	0	0	1	Q	1	0	1	0	0	0	1	q	1	1	1	0	0	0	1
2	0	1	1	0	0	1	0	R	1	0	1	0	0	1	0	r	1	1	1	0	0	1	0
3	0	1	1	0	0	1	1	S	1	0	1	0	0	1	1	s	1	1	1	0	0	1	1
4	0	1	1	0	1	0	0	T	1	0	1	0	1	0	0	t	1	1	1	0	1	0	0
5	0	1	1	0	1	0	1	U	1	0	1	0	1	0	1	u	1	1	1	0	1	0	1
6	0	1	1	0	1	1	0	V	1	0	1	0	1	1	0	v	1	1	1	0	1	1	0
7	0	1	1	0	1	1	1	W	1	0	1	0	1	1	1	w	1	1	1	0	1	1	1
8	0	1	1	1	0	0	0	X	1	0	1	1	0	0	0	x	1	1	1	1	0	0	0
9	0	1	1	1	0	0	1	Y	1	0	1	1	0	0	1	y	1	1	1	1	0	0	1
:	0	1	1	1	0	1	0	Z	1	0	1	1	0	1	0	z	1	1	1	1	0	1	0
;	0	1	1	1	0	1	1	[	1	0	1	1	0	1	1	{	1	1	1	1	0	1	1
<	0	1	1	1	1	0	0	\	1	0	1	1	1	0	0	|	1	1	1	1	1	0	0
=	0	1	1	1	1	0	1	]	1	0	1	1	1	0	1	}	1	1	1	1	1	0	1
>	0	1	1	1	1	1	0	^	1	0	1	1	1	1	0	~	1	1	1	1	1	1	0
?	0	1	1	1	1	1	1	_	1	0	1	1	1	1	1	delete	1	1	1	1	1	1	1