TABLE 3-4 American Standard Code for information Interchange (ASCII)

Character	Binary code	Character	Binary code
A	100 0001	0	011 0000
B	100 0010	1	011 0001
C	100 0011	2	011 0010
D	100 0100	3	011 0011
E	100 0101	4	011 0100
F	100 0110	5	011 0101
G	100 0111	6	011 0110
H	100 1000	7	011 0111
I	100 1001	8	011 1000
J	100 1010	9	011 1001
K	100 1011		
L	100 1100		
M	100 1101	space	010 0000
N	100 1110	.	010 1110
O	100 1111	(	010 1000
P	101 0000	+	010 1011
Q	101 0001	$	010 0100
R	101 0010	*	010 1010
S	101 0011	)	010 1001
T	101 0100	−	010 1101
U	101 0101	/	010 1111
V	101 0110	,	010 1100
W	101 0111	=	011 1101
X	101 1000		
Y	101 1001		
Z	101 1010		

bits stored in registers so that operations are performed on operands of the same type. In inspecting the bits of a computer register at random, one is likely to find that it represents some type of coded information rather than a binary number.

Binary codes can be formulated for any set of discrete elements such as the musical notes and chess pieces and their positions on the chessboard. Binary codes are also used to formulate instructions that specify control information for the computer. This chapter is concerned with *data* representation. Instruction codes are discussed in Chap. 5.

3-2 Complements

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation. There are two types of complements for each base r system: the r's complement and the $(r - 1)$'s complement.

When the value of the base r is substituted in the name, the two types are referred to as the 2's and l's complement for binary numbers and the 10's and 9's complement for decimal numbers.

$(r - 1)$'s Complement

9's complement

Given a number N in base r having n digits, the $(r - 1)$'s complement of N is defined as $(r^n - 1) - N$. For decimal numbers $r = 10$ and $r - 1 = 9$, so the 9's complement of N is $(10^n - 1) - N$. Now, 10^n represents a number that consists of a single 1 followed by n 0's. $10^n - 1$ is a number represented by n 9's. For example, with $n = 4$ we have $10^4 = 10000$ and $10^4 - 1 = 9999$. It follows that the 9's complement of a decimal number is obtained by subtracting each digit from 9. For example, the 9's complement of 546700 is $999999 - 546700 = 453299$ and the 9's complement of 12389 is $99999 - 12389 = 87610$.

1's complement

For binary numbers, $r = 2$ and $r - 1 = 1$, so the 1's complement of N is $(2^n - 1) - N$. Again, 2^n is represented by a binary number that consists of a 1 followed by n 0's. $2^n - 1$ is a binary number represented by n l's. For example, with $n = 4$, we have $2^4 = (10000)_2$ and $2^4 - 1 = (1111)_2$. Thus the l's complement of a binary number is obtained by subtracting each digit from 1. However, the subtraction of a binary digit from 1 causes the bit to change from 0 to 1 or from 1 to 0. Therefore, the l's complement of a binary number is formed by changing l's into 0's and 0's into l's. For example, the l's complement of 1011001 is 0100110 and the l's complement of 0001111 is 1110000.

The $(r - 1)$'s complement of octal or hexadecimal numbers are obtained by subtracting each digit from 7 or F (decimal 15) respectively.

$(r$'s$)$ Complement

The r's complement of an n-digit number N in base r is defined as $r^n - N$ for $N \neq 0$ and 0 for $N = 0$. Comparing with the $(r - 1)$'s complement, we note that the r's complement is obtained by adding 1 to the $(r - 1)$'s complement since

10's complement

$r^n - N = [(r^n - 1) - N] + 1$. Thus the 10's complement of the decimal 2389 is $7610 + 1 = 7611$ and is obtained by adding 1 to the 9's complement value. The 2's complement of binary 101100 is $010011 + 1 = 010100$ and is obtained by adding 1 to the l's complement value.

Since 10^n is a number represented by a 1 followed by n 0's, then $10^n - N$, which is the 10's complement of N, can be formed also be leaving all least significant 0's unchanged, subtracting the first nonzero least significant digit from 10, and then subtracting all higher significant digits from 9. The 10's complement of 246700 is 753300 and is obtained by leaving the two zeros unchanged, subtracting 7 from 10, and subtracting the other three digits from 9. Similarly, the 2's

2's complement

complement can be formed by leaving all least significant 0's and the first 1 unchanged, and then replacing l's by 0's and 0's by l's in all other higher, significant bits. The 2's complement of 1101100 is 0010100 and is obtained by leaving the two low-order 0's and the first 1 unchanged, and then replacing l's by 0's and 0's by l's in the other four most significant bits.

In the definitions above it was assumed that the numbers do not have a radix point. If the original number N contains a radix point, it should be removed temporarily to form the r's or $(r-1)$'s complement. The radix point is then restored to the complemented number in the same relative position. It is also worth mentioning that the complement of the complement restores the number to its original value. The r's complement of N is $r^n - N$. The complement of the complement is $r^n - (r^n - N) = N$ giving back the original number.

Subtraction of Unsigned Numbers

The direct method of subtraction taught in elementary schools uses the borrow concept. In this method we borrow a 1 from a higher significant position when the minuend digit is smaller than the corresponding subtrahend digit. This seems to be easiest when people perform subtraction with paper and pencil. When subtraction is implemented with digital hardware, this method is found to be less efficient than the method that uses complements.

subtraction

The subtraction of two n-digit unsigned numbers $M - N (N \neq 0)$ in base r can be done as follows:

1. Add the minuend M to the r's complement of the subtrahend N. This performs $M + (r^n - N) = M - N + r^n$.
2. If $M \geq N$, the sum will produce an end carry r^n which is discarded, and what is left is the result $M - N$.
3. If $M < N$, the sum does not produce an end carry and is equal to $r^n - (N - M)$, which is the r's complement of $(N - M)$. To obtain the answer in a familiar form, take the r's complement of the sum and place a negative sign in front.

end carry

Consider, for example, the subtraction $72532 - 13250 = 59282$. The 10's complement of 13250 is 86750. Therefore:

$$
\begin{aligned}
M &= 72532 \\
\text{10's complement of } N &= +\underline{86750} \\
\text{Sum} &= 159282 \\
\text{Discard end carry } 10^5 &= -\underline{100000} \\
\text{Answer} &= 59282
\end{aligned}
$$

Now consider an example with $M < N$. The subtraction $13250 - 72532$ produces negative 59282. Using the procedure with complements, we have

$$
\begin{aligned}
M &= 13250 \\
\text{10's complement of } N &= +\underline{27468} \\
\text{Sum} &= 40718
\end{aligned}
$$

There is no end carry

Answer is negative 59282 = 10's complement of 40718

Since we are dealing with unsigned numbers, there is really no way to get an unsigned result for the second example. When working with paper and pencil, we recognize that the answer must be changed to a signed negative number. When subtracting with complements, the negative answer is recognized by the absence of the end carry and the complemented result.

Subtraction with complements is done with binary numbers in a similar manner using the same procedure outlined above. Using the two binary numbers $X = 1010100$ and $Y = 1000011$, we perform the subtraction $X - Y$ and $Y - X$ using 2's complements:

$$\begin{aligned}
X &= \quad 1010100 \\
\text{2's complement of } Y &= \underline{+01111}01 \\
\text{Sum} &= \quad 10010001 \\
\text{Discard end carry } 2^7 &= -10000000 \\
\text{Answer: } X - Y &= \quad 0010001
\end{aligned}$$

$$\begin{aligned}
Y &= \underline{+0000}11 \\
\text{2's complement of } X &= +0101100 \\
\text{Sum} &= \quad 1101111
\end{aligned}$$

There is no end carry

Answer is negative 0010001 = 2's complement of 1101111

3-3 Fixed-Point Representation

Positive integers, including zero, can be represented as unsigned numbers. However, to represent negative integers, we need a notation for negative values. In ordinary arithmetic, a negative number is indicated by a minus sign and a positive number by a plus sign. Because of hardware limitations, computers must represent everything with 1's and 0's, including the sign of a number. As a consequence, it is customary to represent the sign with a bit placed in the leftmost position of the number. The convention is to make the sign bit equal to 0 for positive and to 1 for negative.

binary point

In addition to the sign, a number may have a binary (or decimal) point. The position of the binary point is needed to represent fractions, integers, or mixed integer-fraction numbers. The representation of the binary point in a register is complicated by the fact that it is characterized by a position in the register. There are two ways of specifying the position of the binary point in a register: by giving it a fixed position or by employing a floating-point representation. The fixed-point method assumes that the binary point is always fixed in one position. The two positions most widely used are (1) a binary point in the extreme left of the register to make the stored number a fraction, and (2) a binary point in the extreme right of the register to make the stored number an integer. In either case, the binary point is not actually present, but its presence is assumed from the fact that the number stored in the register is treated as a fraction or as an integer. The floating-point representation uses a second register to store a number that designates the position of the decimal point in the first register. Floating-point representation is discussed further in the next section.

Integer Representation

signed numbers

When an integer binary number is positive, the sign is represented by 0 and the magnitude by a positive binary number. When the number is negative, the sign is represented by 1 but the rest of the number may be represented in one of three possible ways:

1. Signed-magnitude representation
2. Signed-1's complement representation
3. Signed 2's complement representation

The signed-magnitude representation of a negative number consists of the magnitude and a negative sign. In the other two representations, the negative number is represented in either the l's or 2's complement of its positive value. As an example, consider the signed number 14 stored in an 8-bit register. +14 is represented by a sign bit of 0 in the leftmost position followed by the binary equivalent of 14: 00001110. Note that each of the eight bits of the register must have a value and therefore 0's must be inserted in the most significant positions following the sign bit. Although there is only one way to represent +14, there are three different ways to represent −14 with eight bits.

In signed-magnitude representation	1 0001110
In signed-1's complement representation	1 1110001
In signed-2's complement representation	1 1110010

The signed-magnitude representation of −14 is obtained from +14 by complementing only the sign bit. The signed-1's complement representation of −14 is obtained by complementing all the bits of +14, including the sign bit. The signed-2's complement representation is obtained by taking the 2's complement of the positive number, including its sign bit.

The signed-magnitude system is used in ordinary arithmetic but is awkward when employed in computer arithmetic. Therefore, the signed-complement is normally used. The l's complement imposes difficulties because it has two representations of 0 (+0 and −0). It is seldom used for arithmetic operations except in some older computers. The 1's complement is useful as a logical operation since the change of 1 to 0 or 0 to 1 is equivalent to a logical complement operation. The following discussion of signed binary arithmetic deals exclusively with the signed-2's complement representation of negative numbers.

Arithmetic Addition

2's complement addition

The addition of two numbers in the signed-magnitude system follows the rules of ordinary arithmetic. If the signs are the same, we add the two magnitudes and give the sum the common sign. If the signs are different, we subtract the smaller magnitude from the larger and give the result the sign of the larger magnitude. For example, $(+25) + (-37) = -(37 - 25) = -12$ and is done by subtracting the smaller magnitude 25 from the larger magnitude 37 and using the sign of 37 for the sign of the result. This is a process that requires the comparison of the signs and the magnitudes and then performing either addition or subtraction. (The procedure for adding binary numbers in signed-magnitude representation is described in Sec. 10-2.) By contrast, the rule for adding numbers in the signed-2's complement system does not require a comparison or subtraction, only addition and complementation. The procedure is very simple and can be stated as follows: Add the two numbers, including their sign bits, and discard any carry out of the sign (leftmost) bit position. Numerical examples for addition are shown below. Note that negative numbers must initially be in 2's complement and that if the sum obtained after the addition is negative, it is in 2's complement form.

$$\begin{array}{rr}
+6 & 00000110 \\
\underline{+13} & \underline{00001101} \\
+19 & 00010011
\end{array}
\qquad\qquad
\begin{array}{rr}
-6 & 11111010 \\
\underline{+13} & \underline{00001101} \\
+7 & 00000111
\end{array}$$

$$\begin{array}{rr}
+6 & 00000110 \\
\underline{-13} & \underline{11110011} \\
-7 & 11111001
\end{array}
\qquad\qquad
\begin{array}{rr}
-6 & 11111010 \\
\underline{-13} & \underline{11110011} \\
-19 & 11101101
\end{array}$$

In each of the four cases, the operation performed is always addition, including the sign bits. Any carry out of the sign bit position is discarded, and negative results are automatically in 2's complement form.

The complement form of representing negative numbers is unfamiliar to people used to the signed-magnitude system. To determine the value of a negative number when in signed-2's complement, it is necessary to convert it to a positive number to place it in a more familiar form. For example, the signed binary number 11111001 is negative because the leftmost bit is 1. Its 2's complement is 00000111, which is the binary equivalent of +7. We therefore recognize the original negative number to be equal to −7.