Chapter 2

2.1 Data Types

The data types found in the registers of digital computers may be classified as being one of the following categories: (1) numbers used in arithmetic computations, (2) letters of the alphabet used in data processing, and (3) other discrete symbols used for specific purposes. All types of data, except binary numbers, are represented in computer registers in binary-coded form. This is because registers are made up of flip-flops and flip-flops are two-state devices that can store only 1's and 0's. The binary number system is the most natural system to use in a digital computer. But sometimes it is convenient to employ different number systems, especially the decimal number system, since it is used by people to perform arithmetic computations.

2.2 Radix:

A number system of *base*, or *radix*, r is a system that uses distinct symbols for r digits.

Decimal:

the decimal number system in everyday use employs the radix 10 system. The 10 symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. The string of digits 724.5 is interpreted to represent the quantity: $7 \times 10^2 + 2 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1}$

Binary:

The *binary* number system uses the radix 2. The two digit symbols used are 0 and 1. The string of digits 101101 is interpreted to represent the quantity

$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 45$$

To distinguish between different radix numbers, the digits will be enclosed in parentheses and the radix of the number inserted as a subscript. For example,

$$(101101)_2 = (45)_{10}.$$

Octal and Hexadecimal:

Besides the decimal and binary number systems, the *octal* (radix 8) and *hexadecimal* (radix 16) are important in digital computer work. The eight symbols of the octal system are 0, 1, 2, 3, 4, 5, 6, and 7. The 16 symbols of the hexadecimal system are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

$$(736.4)_8 = 7 \times 8^2 + 3 \times 8^1 + 6 \times 8^0 + 4 \times 8^{-1}$$
$$= 7 \times 64 + 3 \times 8 + 6 \times 1 + 4/8 = (478.5)_{10}$$

The equivalent decimal number of hexadecimal F3 is

$$(F3)_{16} = F \times 16 + 3 = 15 \times 16 + 3 = (243)_{10}$$

2.3 Conversions:

Conversion of decimal 41.6875 into binary.

Integer = 41

41	
20	1
10	0
5	0
2	1
1	0
0	1

Fraction = 0.6875

0.6875
2
1.3750
x 2
0.7500
x 2
1.5000
x 2
1.0000

$(41)_{10} = (101001)_2$ $(0.6875)_{10} = (0.1011)_2$

$(41.6875)_{10} = (101001.1011)_2$

As the decimal can be converted to binary similarly the other decimal can be converted to octal and hexadecimal.

1	2	7	5	4	3	
1 0 1 0	1 1 1 1	0 1 1 0	0 0 1 1			Binary
A	F	6	3			Hexadecimal

Binary, octal, and hexadecimal conversion.

2.4 Other Codes:

Binary-Coded Octal Numbers

Octal number	Binary-coded octal	Decimal equivalent	
0	000	0	↑
1	001	1	
2	010	2	Code
3	011	3	for one
4	100	4	octal
5	101	5	digit
6	110	6	
7	111	7	↓
10	001 000	8	
11	001 001	9	
12	001 010	10	
24	010 100	20	
62	110 010	50	
143	001 100 011	99	
370	011 111 000	248	

Binary-Coded Hexadecimal Numbers

Hexadecimal number	Binary-coded hexadecimal	Decimal equivalent	
0	0000	0	↑
1	0001	1	
2	0010	2	
3	0011	3	
4	0100	4	
5	0101	5	
6	0110	6	Code
7	0111	7	for one
8	1000	8	hexadecimal
9	1001	9	digit
A	1010	10	
B	1011	11	
C	1100	12	
D	1101	13	
E	1110	14	
F	1111	15	↓
14	0001 0100	20	
32	0011 0010	50	
63	0110 0011	99	
F8	1111 1000	248	

Binary-Coded Decimal (BCD) Numbers

Decimal number	Binary-coded decimal (BCD) number	
0	0000	↑
1	0001	
2	0010	
3	0011	Code
4	0100	for one
5	0101	decimal
6	0110	digit
7	0111	
8	1000	
9	1001	↓
10	0001 0000	
20	0010 0000	
50	0101 0000	
99	1001 1001	
248	0010 0100 1000	

2.5 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 1's complement for binary numbers and the 10's and 9's complement for decimal numbers.

$(r-1)$'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 $99999-12389=87610$.

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 1's. For example, with $n=4$, we have $2^4=(10000)_2$ and $2^4-1=(1111)_2$. Thus the 1'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 1's complement of a binary number is formed by changing 1's into 0's and 0's into 1's. For example, the 1's complement of 1011001 is 0100110 and the 1'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 $r^n - N = [(r^n - 1) - N] + 1$. 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 1'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 complement can be formed by leaving all least significant 0's and the first 1 unchanged, and then replacing 1's by 0's and 0's by 1'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 1's by 0's and 0's by 1's in the other four most significant bits.

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.

2.6 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.

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

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

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

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

There is no end carry

Answer is negative 59282 = 10's complement of 40718

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:

$$
\begin{array}{rr}
X = & 1010100 \\
\text{2's complement of } Y = & +0111101 \\
\text{Sum} = & 10010001 \\
\text{Discard end carry } 2^7 = & -10000000 \\
\text{Answer: } X - Y = & 0010001
\end{array}
$$

$$
\begin{array}{rr}
Y = & 1000011 \\
\text{2's complement of } X = & +0101100 \\
\text{Sum} = & 1101111
\end{array}
$$

There is no end carry
Answer is negative 0010001 = 2's complement of 1101111

2.7 Fixed Point Representation:

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

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.

Integer Representation:

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

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. 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 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 signed binary arithmetic deals exclusively with the signed-2's complement representation of negative numbers.

Arithmetic Addition:

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.

+6	00000110	−6	11111010
+13	00001101	+13	00001101
+19	00010011	+7	00000111
+6	00000110	−6	11111010
−13	11110011	−13	11110011
−7	11111001	−19	11101101

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.

Arithmetic Subtraction:

Subtraction of two signed binary numbers when negative numbers are in 2's complement form is very simple and can be stated as follows: Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including the sign bit). A carry out of the sign bit position is discarded.

But changing a positive number to a negative number is easily done by taking its 2's complement. The reverse is also true because the complement of a negative number in complement form produces the equivalent positive number. Consider the subtraction of (−6) − (−13) = +7. In binary with eight bits this is written as 11111010 − 11110011. The subtraction is changed to addition by taking the 2's complement of the subtrahend (−13) to give (+13). In binary this is 11111010 + 00001101 = 100000111. Removing the end carry, we obtain the correct answer 00000111 (+7).

Overflow:

When two unsigned numbers are added, an overflow is detected from the end carry out of the most significant position. In the case of signed numbers, the leftmost bit always represents the sign, and negative numbers are in 2's complement form. When two signed numbers are added, the sign bit is treated as part of the number and the end carry does not indicate an overflow.

An overflow may occur if the two numbers added are both positive or both negative. range of numbers that each register can accommodate is from binary +127 to binary −128. Since the sum of the two numbers is +150, it exceeds the capacity of the 8-bit register. This is true if the numbers are both positive or both negative.

carries:	0	1
+70	0	1000110
+80	0	1010000
+150	1	0010110

carries:	1	0
−70	1	0111010
−80	1	0110000
−150	0	1101010

An overflow condition can be detected by observing the carry into the sign bit position and the carry out of the sign bit position. If these two carries are not equal, an overflow condition is produced. This is indicated in the examples where the two carries are explicitly shown. If the two carries are applied to an exclusive-OR gate, an overflow will be detected when the output of the gate is equal to 1.

2.8 Floating Point Representation:

The floating-point representation of a number has two parts. The first part represents a signed, fixed-point number called the mantissa. The second part designates the position of the decimal (or binary) point and is called the exponent. The fixed-point mantissa may be a fraction or an integer. For example, the decimal number +6132.789 is represented in floating-point with a fraction and an exponent as follows:

Fraction	*Exponent*
+0.6132789	+04

The value of the exponent indicates that the actual position of the decimal point is four positions to the right of the indicated decimal point in the fraction. This representation is equivalent to the scientific notation $+0.6132789 \times 10^{+4}$.

Floating-point is always interpreted to represent a number in the following form: $m \times r^e$ radix r

Only the mantissa m and the exponent e are physically represented in the register (including their signs). the binary number +1001.11 is represented with an 8-bit fraction and 6-bit exponent as follows:

Fraction	*Exponent*
01001110	000100

$$m \times 2^e = +(.1001110)_2 \times 2^{+4}$$

2.9 Other Binary Codes:

Gray Code:

Gray code changes by only one bit as it sequences from one number to the next. In other words, the change from any number to the next in sequence is recognized by a change of only one bit from 0 to 1 or from 1 to 0.

4-Bit Gray Code

Binary code	Decimal equivalent	Binary code	Decimal equivalent
0000	0	1100	8
0001	1	1101	9
0011	2	1111	10
0010	3	1110	11
0110	4	1010	12
0111	5	1011	13
0101	6	1001	14
0100	7	1000	15

Four Different Binary Codes for the Decimal Digit

Decimal digit	BCD 8421	2421	Excess-3	Excess-3 gray
0	0000	0000	0011	0010
1	0001	0001	0100	0110
2	0010	0010	0101	0111
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1100
6	0110	1100	1001	1101
7	0111	1101	1010	1111
8	1000	1110	1011	1110
9	1001	1111	1100	1010
Unused bit combi-nations	1010	0101	0000	0000
	1011	0110	0001	0001
	1100	0111	0010	0011
	1101	1000	1101	1000
	1110	1001	1110	1001
	1111	1010	1111	1011

Others

2.10 Error Detection Codes:

Binary information transmitted through some form of communication medium is subject to external noise that could change bits from 1 to 0, and vice versa. An error detection code is a binary code that detects digital errors during transmission.

The most common error detection code used is the *parity bit*. A parity bit is an extra bit included with a binary message to make the total number of 1's either odd or even The *P*(odd) bit is chosen in such a way as to make the sum of 1's (in all four bits) odd. The *P*(even) bit is chosen to make the sum of all 1's even.

Parity Bit Generation

Message *xyz*	*P*(odd)	*P*(even)
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1

At the sending end, the message is applied to a *parity generator*, The message, including the parity bit, is transmitted to its destination. At the receiving end, all the incoming bits (in this case, four) are applied to a *parity checker* An error is detected if the checked parity does not conform The parity method detects the presence of one, three, or any odd number of errors.