# فصل اول

# سیستم‌های دودویی

## ۱ ـ مبنای نمایش اعداد

**اعداد دهدهی یا دسیمال (Decimal):** پایه یا مبنای ۱۰ : استفاده از ۱۰ رقم ۰ تا ۹

$$a_5a_4a_3a_2a_1a_0.\ a_{-1}a_{-2}a_{-3}$$

معادل است با

$$10^5a_5 + 10^4a_4 + 10^3a_3 + 10^2a_2 + 10^1a_1 + 10^0a_0 + 10^{-1}a_{-1} + 10^{-2}a_{-2} + 10^{-3}a_{-3}$$

**اعداد دودویی یا باینری (Binary):** مبنای ۲: استفاده از ۲ رقم ۰ و ۱. به هر رقم در این حالت یک «**بیت**» گفته می‌شود.

11010.11 is 26.75

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 26.75$$

**اعداد هشت‌هشتی یا اکتال (Octal):** استفاده از ۸ رقم ۰ تا ۷

$$(127.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = (87.5)_{10}$$

**اعداد شانزده‌شانزدهی یا هگز یا هگزادسیمال (Hexadecimal):** استفاده از ارقام ۰ تا ۹ و حروف A تا F

$$(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (46{,}687)_{10}$$

**اعداد در مبنای دلخواه r:** از r رقم شامل ۰ تا r-1 استفاده می‌کند.

$$(4021.2)_5 = 4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1} = (511.4)_{10}$$

**تبدیل بین مبناهایی که توان صحیحی از یکدیگر هستند:** *از مبنای کوچکتر به مبنای بزرگتر*: با دسته‌بندی ارقام و جایگزینی رقم معادل (مربوط به مبنای بزرگتر). *از مبنای بزرگتر به مبنای کوچکتر*: جایگزینی معادل هر رقم مبنای بزرگتر با نمایش معادل مربوط به مبنای کوچکتر.

$$\begin{array}{ccccccccccc} (10 & 110 & 001 & 101 & 011 & \cdot & 111 & 100 & 000 & 110)_2 & = (26153.7406)_8 \\ 2 & 6 & 1 & 5 & 3 & & 7 & 4 & 0 & 6 & \end{array}$$

$$\begin{array}{cccccccc} (10 & 1100 & 0110 & 1011 & \cdot & 1111 & 0010)_2 & = (2C6B.F2)_{16} \\ 2 & C & 6 & B & & F & 2 & \end{array}$$

$$\begin{array}{cccccccc} (673.124)_8 = & (110 & 111 & 011 & \cdot & 001 & 010 & 100)_2 \\ & 6 & 7 & 3 & & 1 & 2 & 4 \end{array}$$

$$\begin{array}{cccccc} (306.D)_{16} = & (0011 & 0000 & 0110 & \cdot & 1101)_2 \\ & 3 & 0 & 6 & & D \end{array}$$

## ۲ـ تبدیل از مبنای دلخواه به مبنای ۱۰

از روش بسط عدد مشابه با مثالهای فوق استفاده می‌کنیم.

## ۳ـ تبدیل از مبنای ۱۰ به مبنای دلخواه r

قسمت صحیح و قسمت اعشاری را جداگانه تبدیل کرده و در پایان دو نمایش به دست آمده را با نقطه ممیز از هم جدا می‌کنیم. قسمت صحیح با تقسیمات متوالی بر r (و انتخاب باقیمانده‌ها) و قسمت اعشاری با ضربهای متوالی در r (و انتخاب قسمتهای صحیح) تبدیل می‌شوند.

## EXAMPLE 1.1

Convert decimal 41 to binary. First, 41 is divided by 2 to give an integer quotient of 20 and a remainder of $\frac{1}{2}$. Then the quotient is again divided by 2 to give a new quotient and remainder. The process is continued until the integer quotient becomes 0. The *coefficients* of the desired binary number are obtained from the *remainders* as follows:

| | Integer Quotient | | Remainder | Coefficient |
|---|---|---|---|---|
| $41/2 =$ | 20 | + | $\frac{1}{2}$ | $a_0 = 1$ |
| $20/2 =$ | 10 | + | 0 | $a_1 = 0$ |
| $10/2 =$ | 5 | + | 0 | $a_2 = 0$ |
| $5/2 =$ | 2 | + | $\frac{1}{2}$ | $a_3 = 1$ |
| $2/2 =$ | 1 | + | 0 | $a_4 = 0$ |
| $1/2 =$ | 0 | + | $\frac{1}{2}$ | $a_5 = 1$ |

Therefore, the answer is $(41)_{10} = (a_5a_4a_3a_2a_1a_0)_2 = (101001)_2$.

## EXAMPLE 1.2

Convert decimal 153 to octal. The required base $r$ is 8. First, 153 is divided by 8 to give an integer quotient of 19 and a remainder of 1. Then 19 is divided by 8 to give an integer quotient of 2 and a remainder of 3. Finally, 2 is divided by 8 to give a quotient of 0 and a remainder of 2. This process can be conveniently manipulated as follows:

| | |
|---|---|
| 153 | |
| 19 | 1 |
| 2 | 3 |
| 0 | $2 = (231)_8$ |

The conversion of a decimal *fraction* to binary is accomplished by a method similar to that used for integers. However, multiplication is used instead of division, and integers instead of remainders are accumulated. Again, the method is best explained by example.

**EXAMPLE 1.3**

Convert $(0.6875)_{10}$ to binary. First, 0.6875 is multiplied by 2 to give an integer and a fraction. Then the new fraction is multiplied by 2 to give a new integer and a new fraction. The process is continued until the fraction becomes 0 or until the number of digits has sufficient accuracy. The coefficients of the binary number are obtained from the integers as follows:

| | Integer | | Fraction | Coefficient |
|---|---|---|---|---|
| $0.6875 \times 2 =$ | 1 | + | 0.3750 | $a_{-1} = 1$ |
| $0.3750 \times 2 =$ | 0 | + | 0.7500 | $a_{-2} = 0$ |
| $0.7500 \times 2 =$ | 1 | + | 0.5000 | $a_{-3} = 1$ |
| $0.5000 \times 2 =$ | 1 | + | 0.0000 | $a_{-4} = 1$ |

Therefore, the answer is $(0.6875)_{10} = (0.\, a_{-1}\, a_{-2}\, a_{-3}\, a_{-4})_2 = (0.1011)_2$.

To convert a decimal fraction to a number expressed in base *r*, a similar procedure is used. However, multiplication is by *r* instead of 2, and the coefficients found from the integers may range in value from 0 to $r - 1$ instead of 0 and 1.

**EXAMPLE 1.4**

Convert $(0.513)_{10}$ to octal.

$$0.513 \times 8 = 4.104$$
$$0.104 \times 8 = 0.832$$
$$0.832 \times 8 = 6.656$$
$$0.656 \times 8 = 5.248$$
$$0.248 \times 8 = 1.984$$
$$0.984 \times 8 = 7.872$$

The answer, to seven significant figures, is obtained from the integer part of the products:

$$(0.513)_{10} = (0.406517\ldots)_8$$

The conversion of decimal numbers with both integer and fraction parts is done by converting the integer and the fraction separately and then combining the two answers. Using the results of Examples 1.1 and 1.3, we obtain

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

From Examples 1.2 and 1.4, we have

$$(153.513)_{10} = (231.406517)_8$$

## ۴ـ متمم‌ها

ـ متمم‌ها جهت **ساده‌سازی برخی عملیات** مانند تفریق استفاده می‌شوند.

ـ در هر مبنای دلخواه r دو نوع متمم وجود دارد: **متمم r** (یا متمم مبنا) و **متمم r-1** (یا متمم کاهش یافته، Diminished)
**برای مثال** در حالت r=1 دو نوع متمم ۲ و متمم ۱ داریم (Two's complement و One's complement)

ـ اگر عدد دلخواه N را n رقمی فرض کنیم، متمم r-1 آن به صورت $(r^n-1)-N$ و متمم r به صورت $r^n-N$ تعریف می‌شود.

**متمم r-1:**

برای محاسبه کافی است هر رقم عدد N را از r-1 کم کنیم. **چرا؟**

**مثال: در حالت r=10**

The 9's complement of 546700 is 999999 − 546700 = 453299.

The 9's complement of 012398 is 999999 − 012398 = 987601.

در حالت ساده‌ی r=2 (باینری)، کافی است هر بیت را معکوس کنیم. **چرا؟**

**مثال: در حالت r=2**

The 1's complement of 1011000 is 0100111.

The 1's complement of 0101101 is 1010010.

**متمم r:**

ـ یک راه برای محاسبه‌ی متمم r عدد n رقمی N این است که ابتدا متمم r-1 را محاسبه کرده و سپس آن را با عدد ۱ جمع کنیم. **چرا؟**

ـ راه دیگر این است که از سمت راست شروع کرده و تمام صفرهای متوالی را (در صورت وجود) نوشته، اولین رقم غیر صفر را از r و مابقی را از r-1 کم کنیم. **چرا؟**

the 10's complement of 012398 is 987602

the 10's complement of 246700 is 753300

ـ در حالت r=2 کافی است تمام صفرهای متوالی (از سمت راست) و اولین رقم ۱ را عیناً نوشته و مابقی را معکوس کنیم. **چرا؟**

the 2's complement of 1101100 is 0010100

the 2's complement of 0110111 is 1001001

## ۵ـ تفریق اعداد بدون علامت

برای تفریق دو عدد بدون علامت (یعنی اولا اعداد همواره و به صورت پیش‌فرض مثبت هستند؛ دوماً هیچ رقم یا بیتی برای نمایش علامت مصرف نمی‌کنیم) M و N که هر دو n رقمی فرض می‌شوند، مراحل زیر را طی می‌کنیم:

۱ ـ مفروق منه (یعنی عدد M) را با متمم r مفروق (یعنی N) جمع می‌کنیم:

$$M + (r^n - N) = M - N + r^n$$

۲ ـ اگر M≥N باشد این عمل جمع یک رقم نقلی تولید می‌کند که باید آن را صرفنظر کنیم؛ آن چه باقی می‌ماند همان نتیجه‌ی M-N است.

۳ ـ اگر M<N (یعنی حاصل تفریقِ M-N عددی منفی باشد) باشد، هیچ رقم نقلی تولید نمی‌شود و نتیجه برابر است با $r^n - (N - M)$؛ یعنی نتیجه برابر است با متمم r عدد N-M (که عدد N-M عددی ذاتاً مثبت و منطبق بر فرض بدون علامت بودن اعداد است). ملاحظه می‌کنید که چون اعداد بدون علامت هستند و ما برای علامت هیچ رقم و علامتی در نظر نمی‌گیریم و از طرفی نتیجه در اینجا باید منفی شود، این علامت منفی در این سیستم، به صورت متمم r ِ نتیجه نمایش داده شده است. **در این جا** برای یافتن جواب معمول، از حاصلجمع (یا نتیجه) متمم r گرفته و یک علامت منفی جلوی آن می‌گذاریم: ====> *<u>از هر عدد دو بار متمم r گرفته شود دو باره به خود آن عدد می‌رسیم</u>*. **چرا؟**

**EXAMPLE 1.5**

Using 10's complement, subtract 72532 − 3250.

$$\begin{aligned}
M &= \phantom{+\ }72532\\
\text{10's complement of } N &= +\ 96750\\
\text{Sum} &= \phantom{+\ }169282\\
\text{Discard end carry } 10^5 &= -\ \underline{100000}\\
\textit{Answer} &= \phantom{+\ }69282
\end{aligned}$$

Note that $M$ has five digits and $N$ has only four digits. Both numbers must have the same number of digits, so we write $N$ as 03250. Taking the 10's complement of $N$ produces a 9 in the most significant position. The occurrence of the end carry signifies that $M \geq N$ and that the result is therefore positive.

## EXAMPLE 1.6

Using 10's complement, subtract 3250 − 72532.

$$
\begin{aligned}
M &= \phantom{+\ }03250 \\
\text{10's complement of } N &= +\ \underline{27468} \\
\text{Sum} &= \phantom{+\ }30718
\end{aligned}
$$

There is no end carry. Therefore, the answer is −(10's complement of 30718) = −69282.

Note that since 3250 < 72532, the result is negative. Because we are dealing with unsigned numbers, there is really no way to get an unsigned result for this case. When subtracting with complements, we recognize the negative answer from the absence of the end carry and the complemented result. When working with paper and pencil, we can change the answer to a signed negative number in order to put it in a familiar form.

Subtraction with complements is done with binary numbers in a similar manner, using the procedure outlined previously.

## EXAMPLE 1.7

Given the two binary numbers $X = 1010100$ and $Y = 1000011$, perform the subtraction **(a)** $X - Y$ and **(b)** $Y - X$ by using 2's complements.

(a)

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

(b)

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

There is no end carry. Therefore, the answer is $Y - X = -$(2's complement of 1101111) = −0010001.

تفریق فوق‌الذکر را به کمک متمم r-1 نیز می‌توان انجام داد؛ این متمم یکی کمتر از متمم r است لذا همان عملیات قبلی را این بار به کمک متمم r-1 انجام می‌دهیم و پس از حذف رقم نقلی، نتیجه را با رقم ۱ جمع می‌کنیم. به این کار، رقم نقلی چرخشی (end-around carry) می‌نامند.

**EXAMPLE 1.8**

Repeat Example 1.7, but this time using 1's complement.

**(a)** $X - Y = 1010100 - 1000011$

$$\begin{aligned} X &= \phantom{+}\ 1010100 \\ \text{1's complement of } Y &= +\ \underline{0111100} \\ \text{Sum} &= \phantom{+}\ 10010000 \\ \text{End-around carry} &= +\ \underline{\phantom{000000}1} \\ \textit{Answer}: X - Y &= \phantom{+}\ 0010001 \end{aligned}$$

**(b)** $Y - X = 1000011 - 1010100$

$$\begin{aligned} Y &= \phantom{+}\ 1000011 \\ \text{1's complement of } X &= +\ 0101011 \\ \text{Sum} &= \phantom{+}\ 1101110 \end{aligned}$$

There is no end carry. Therefore, the answer is $Y - X = -(\text{1's complement of } 1101110) = -0010001$.

## ۶ـ اعداد دودویی علامت‌دار

ما انسانها علامت اعداد را روی کاغذ با علامت + یا ـ می‌توانیم نمایش دهیم اما برای نمایش مثبت یا منفی بودن عدد باینری در کامپیوتر نیاز به مکانیزم و قرارداد خاصی داریم و برای نمایش علامت باید از همین بیتهای ۰ و ۱ استفاده کنیم. این مکانیزم باید به گونه‌ای باشد که (۱) قابل ارائه و ذخیره در کامپیوتر باشد، (۲) در کامپیوتر، محاسبات علامت‌دار به درستی انجام شود، و (۳) ما انسانها بتوانیم نتایج عددی ارائه شده از سوی کامپیوتر را به درستی تفسیر و (به اعداد علامتدار دهدهی قابل فهم برای خودمان) تعبیر کنیم.

سه روش شناخته شده برای نمایش اعداد علامتدار:

(۱) روش مقدارـعلامت‌دار (Signed Magnitude): در این روش ابتدا بیت علامت و سپس اندازه‌ی عدد استفاده می‌شود.

(۲) روش متممـعلامت‌دار (Signed Complement): در این روش اعداد منفی با متمم خود نمایش داده می‌شوند. با توجه به این که دو نوع متمم ۱ و متمم ۲ در اختیار داریم، پس روش متممـعلامت‌دار خود به دو روش «متمم۱ـعلامت‌دار» و «متمم۲ـعلامت‌دار» دسته‌بندی می‌شود.

هر سه روش فوق نمایش یکسان و مشترکی برای اعداد مثبت ارائه می‌دهند و تفاوت آنها در نمایش اعداد منفی است. نمایش عدد 9+ در هر سه روش و به کمک ۸ بیت به صورت 00001001 می‌باشد. اما نمایش عدد 9- به صورت زیر خواهد شد:

signed-magnitude representation: 10001001
signed-1's-complement representation: 11110110
signed-2's-complement representation: 11110111

در روش مقدارــ علامت‌دار برای تغییر علامت عدد (از مثبت به منفی و یا از منفی به مثبت) کافی است تنها بیت علامت را معکوس (یا همان متمم به معنای عام خود) کنیم. در روش متمم۱ ــ علامت‌دار برای این کار باید نمایش عدد را متمم ۱ کنیم. در روش متمم۲ ــ علامت‌دار نیز باید نمایش عدد را متمم ۲ کنیم. به جدول زیر دقت کنید.

**Table 1.3**
*Signed Binary Numbers*

| Decimal | Signed-2's Complement | Signed-1's Complement | Signed Magnitude |
|---|---|---|---|
| +7 | 0111 | 0111 | 0111 |
| +6 | 0110 | 0110 | 0110 |
| +5 | 0101 | 0101 | 0101 |
| +4 | 0100 | 0100 | 0100 |
| +3 | 0011 | 0011 | 0011 |
| +2 | 0010 | 0010 | 0010 |
| +1 | 0001 | 0001 | 0001 |
| +0 | 0000 | 0000 | 0000 |
| −0 | — | 1111 | 1000 |
| −1 | 1111 | 1110 | 1001 |
| −2 | 1110 | 1101 | 1010 |
| −3 | 1101 | 1100 | 1011 |
| −4 | 1100 | 1011 | 1100 |
| −5 | 1011 | 1010 | 1101 |
| −6 | 1010 | 1001 | 1110 |
| −7 | 1001 | 1000 | 1111 |
| −8 | 1000 | — | — |

در سیستم‌های کامپیوتری از روش مقدارــ علامت‌دار استفاده نمی‌شود زیرا مستلزم این است که در پردازها و محاسبات، بیت علامت و بیتهای مقدار را جداگانه مدیریت و دستکاری کنیم. از بین دو روش متمم۱ ــ علامت‌دار و متمم۲ ــ علامت‌دار نیز روش متمم۲ ــ علامتدار ترجیح داده می‌شود زیرا اولاً یک نمایش منحصربفرد برای تمام اعداد دسیمال فراهم می‌کند (به نمایش 0+ و 0- در متمم۱ ــ علامت‌دار دقت کنید) دوم این که تعداد اعداد دسیمال مختلفی که می‌توان به کمک این روش نمایش داد بیشتر از روش متمم۱ ــ علامتدار است. بنابراین نتیجه این که:

**ما از این به بعد از محاسبات و در مدارات، اگر لازم باشد، از سیستم متمم۲ ــ علامت‌دار استفاده خواهیم کرد.**

نکته مهم بعدی این است که با توجه به این که از هر عدد دو بار متمم ۲ (و البته متمم ۱ هم همین ویژگی را دارد) گرفته شود، دو باره به خود آن عدد می‌رسیم، پس با گرفتن متمم ۲ از عدد علامت آن عدد تغییر می‌کند. برای مثال، نمایش عدد

5+ (با ۸ بیت) به صورت 00000110 است. اگر متمم ۲ این عدد را محاسبه کنیم، می‌شود 11111010 که معادل با عدد -6 است (پس نمایش عدد 6- در سیستم متمم۲ــ علامت‌دار برابر با 11111010 می‌شود). اگر از نمایش عدد 6- مجدداً متمم ۲ بگیریم، به نمایش اولیه‌ی 00000110 می‌رسیم که معادل با عدد علامت‌دارِ 6+ است.

نکته‌ی مهم بعدی این است که در روش نمایش متمم۲ــ علامت‌دار قرارداد می‌کنیم که اعداد مثبت همیشه با بیت صفر شروع شوند و این قرارداد الزامی است زیرا برای مثال نمایش عدد 6 با سه بیت برابر 110 می‌شود اما نمایش عدد 6+ با سه بیت امکان‌پذیر نیست زیرا تمام سه بیت موجود صرف نمایش اندازه‌ی این عدد شده و دیگری بیتی برای نمایش علامت عدد باقی نمانده است. پس حتماً نیاز به یک بیت چهارم جهت نمایش علامت عدد 6+ داریم که بر طبق قرارداد، این بیت چهارم برای اعداد مثبت برابر صفر در نظر گرفته می‌شود (بنابراین نمایش عدد 6+ با کمترین تعداد بیتهای ممکن برابر 0110 خواهد بود و کمتر از این تعداد امکان‌پذیر نیست). حال، مطابق روال قبل، برای نمایش عدد 6- باید از نمایش بیتیِ 0110 متمم ۲ بگیریم که دیدیم برابر با 1010 می‌شود؛ این نتیجه همان نمایش عدد 6- است که ملاحظه می‌کنید مقدار بیت واقع در سمت چپ‌ترین محل، برابر ۱ است. این مطلب و قرارداد همیشه صحیح است و باید آن را همیشه رعایت کنیم؛ یعنی «در روش متمم۲ــ علامت‌دار همیشه اعداد مثبت با صفر و اعداد منفی با ۱ شروع می‌شوند». برای رعایت همین قرارداد است که در جدول اخیر، نمایشی برای عدد 0- نشان داده نشده است (نمایش عدد 0- همان 0000 است که معادل با 0+ است اما چون قرارداد اخیر را نقض کرده است، در جدول نشان داده نشده است).

## ۶ــ جمع حسابی

برای جمع حسابی دو عدد باینری علامت‌دار نمایش داده شده به روش متمم۲ــ علامت‌دار کافی است به سادگی آن دو عدد را با هم جمع و از رقم نقلی تولید شده در محل بیت علامت صرفنظر کنیم. در این عملیات، حاصل جمع همیشه به قالب و نمایش متمم ۲ خواهد بود.

مثال:

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

یک نکته مهم که در عملیات جمع باید به آن توجه کنیم امکان وقوع سرریز (Overflow) است. اگر تعداد بیتهای در نظر گرفته شده (مثلاً n بیت) برای نمایش حاصل جمع کافی نباشد، سرریز رخ داده و بنابراین، نتیجه‌ی جمع که ناچاراً در n بیت ذخیره شده است، نامعتبر خواهد بود.

## ۷ـ تفریق حسابی (Arithmetic Subtraction)

برای انجام تفریق A-B (که A و B دو عدد علامت‌دار در سیستم متمم۲ ـ علامت‌دار هستند،) کافی است A را با متمم ۲ عدد باینری B جمع کنیم. بنابراین، تفریق نیز به کمک عملیات جمع قابل پیاده‌سازی است.

$$(\pm A) - (+B) = (\pm A) + (-B);$$
$$(\pm A) - (-B) = (\pm A) + (+B).$$

### کدهای دودویی

کد دودویی یعنی یک نوع نمایش خاص برای نشان دادن یک مفهوم ثابت و مشترک. بنابراین، کدهای دودویی مختلفی برای نمایش یک مفهوم یا مقدار عددی ثابت و مشخص وجود دارند مانند کدهای BCD، کد باینری، کد گری، کد one-hot، کد two-hot و غیره.

### کد BCD

در کد BCD یا «دهدهی کد شده به دودویی[1]» هر رقم دسیمال با چهار بیت نمایش داده می‌شود.

Table 1.4
*Binary-Coded Decimal (BCD)*

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

یک عدد دهدهی (یا دسیمال) سه رقمی با ۱۲=۴×۳ بیت نمایش داده می‌شود. مثلاً:

$$(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2$$

ملاحظه می‌کنید که نمایش باینری تنها به ۸ بیت اما نمایش BCD به ۱۲ بیت برای نمایش یک مفهوم یا مقدار مشخص و ثابت نیاز دارد.

یک نکته‌ی مهم این است که کدهای BCD علیرغم این که در ساختارشان از بیت های ۰ و ۱ استفاده شده است، اما این کدها دهدهی هستند نه باینری!!!. تنها تفاوت بین کد BCD و دهدهی این است که در عددنویسیِ دهدهی از ارقام 0 و 1

[1] Binary Coded Decimal

و ... و 9 استفاده می‌شود اما در کد BCD از (به ترتیب) 0000، 0001، و ... و 1001. مقدار دهدهی آنها یکی است. برای مثال نمایش عددهای دهدهی 10 و 15 به ترتیب برابر با 0000 0001 و 0101 0001 (یعنی هر کدام با ۸ بیت قابل انجام) است در حالی که نمایش باینری این اعداد به ترتیب 1010 و 1111 یعنی تنها با چهار بیت قابل انجام است.

مشابه با سیستم باینری که از روش متمم ۲ برای نمایش اعداد علامت‌دار استفاده می‌شد، در این جا از روش متمم ۱۰ استفاده می‌شود. بنابراین، برای مثال، برای انجام جمع

$$(+375) + (-240) = +135$$

در سیستم BCD ابتدا هر یک از این اعداد را به قالب متمم ۱۰ نمایش داده و سپس با هم جمع کرده و از رقم نقلی احتمالی صرفنظر می کنیم. مشابه با روش متمم ۲ که بیت صفر نشان دهنده‌ی علامت مثبت و بیت ۱ نشان دهنده‌ی علامت منفی بود، در این جا رقم دهدهیِ 0 (معادل با نمایش BCDِ 0000) نشان دهنده‌ی علامت مثبت و رقم دهدهی 9 (معادل با نمایش BCDِ 1001) نشان دهنده‌ی علامت منفی است.

```
0 375
9 760
-------
0 135
```

اگر بخواهیم عملیات فوق را به صورت BCD انجام دهیم،

```
0000   0011   0111   0101
1001   0111   0110   0000
------------------------
1001   1010   1101   0101
```

ملاحظه می کنیم که برخی گروههای چهار بیتی نامعتبر هستند زیرا یک رقم BCD باید بین 0000 تا 1001 (یعنی بین 0 تا 9) باشد اما در این جا رقم اول از سمت راست برابر 5 (معتبر)، رقم دوم برابر 13 (نامعتبر) و رقم سوم برابر 10 (نامعتبر) است؛ یعنی دو رقم نامعتبر وجود دارد که باید تصحیح شوند؛ در این عملیات تصحیح باید رقم 10 تا 15 تبدیل به 0 تا 5 شوند که برای انجام این تبدیل کافی است عدد 6 را به حاصل جمع اضافه کنیم. بنابراین:

```
   1      1
0000   0011   0111   0101
1001   0111   0110   0000
------------------------
1010   1011   1101   0101
0110   0110   0110   ----
------------------------
0000   0001   0011  0101    =    +135
```

ملاحظه می‌کنید که نتیجه‌ای معتبر به دست آمده است.

## سایر کدهای دهدهی

**Table 1.5**
*Four Different Binary Codes for the Decimal Digits*

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

**کدهای وزن‌دار (Weighted):** کدهایی هستند که هر بیت دارای ارزش یا وزن ویژه‌ای است. مثلاً کد BCD دارای وزن‌های ۸ و ۴ و ۲ و ۱ است (جدول فوق).

**کد بدون‌وزن (Unweighted):** کدی است که بیتهای آن دارای ارزش یا وزن خاصی نیستند. مثلاً کد مازاد‌ـ۳ (Excess-3) بدون وزن است.

کدهای 2421 و مازاد‌ـ۳ نمونه‌هایی از کدهای خود‌ـ‌متمم (Self-Complementing) هستند؛ در این کدها برای به دست آوردن متمم ۹ یک عدد دهدهی (دسیمال) کافی است بیتهای موجود در نمایش آن عدد را معکوس (یا همان متمم) کنیم. برای مثال نمایش عدد دهدهی 395 به روش کد مازاد‌ـ۳ برابر با 1000 1100 0110 است. از طرفی متمم ۹ این عدد دهدهی برابر است با 604 که نمایش آن در کد مازاد‌ـ۳ برابر با 0111 0011 1001 است و این نمایش به راحتی با متمم کردن بیتهای نمایش قبلی به دست می‌آید.

## کد گری (Gray)

کد گری کدی است که با رفتن از یک کد به کد بعدی یا قبلی، تنها یک بیت تغییر کرده و بقیه‌ی بیتها ثابت می‌مانند. یکی از موارد استفاده‌ی این کد در کد کردن محل یک چرخ دوّار خاص (برای مثال کد کردن مکان یک شفت موتور) است.

**Table 1.6**
*Gray Code*

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

## کد تشخیص خطا

برای تشخیص وقوع خطا در هنگام ارسال داده‌های بیتی، یک روش ساده استفاده از بیت توازن (parity bit) است. دو نوع توازن زوج و فرد وجود دارد.

## منطق دودویی

منطق دودویی با متغیرهایی (مانند x و y و z) که دو ارزش یا مقدار مختلف و با عملیاتی (مانند AND و OR و NOT) که مفهوم منطقی دارند، سر و کار دارد. ما این دو مقدار مختلف را معمولاً با ۰ و ۱ نمایش می‌دهیم. این منطق معادل با جبری به نام جبر بول در ریاضیات است که در فصل بعد به آن پرداخته خواهد شد.

**Table 1.8**
*Truth Tables of Logical Operations*

| AND | | | OR | | | NOT | |
|---|---|---|---|---|---|---|---|
| $x$ | $y$ | $x \cdot y$ | $x$ | $y$ | $x + y$ | $x$ | $x'$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | | |
| 1 | 1 | 1 | 1 | 1 | 1 | | |

## گیت‌های منطقی

گیت‌های منطقی نوع خاصی از مدارات الکترونیکی هستند که روی یک یا چند سیگنال ورودی عمل می‌کنند تا یک یا چند سیگنال خروجی تولید کنند. گیت‌های منطقی با ترکیب با هم می‌توانند یک سیستم دیجیتال تشکیل دهند. در یک سیستم دیجیتال، ولتاژها و جریانها دارای دو مقدار (و به عبارت بهتر و دقیق‌تر دو «سطح») مجزا و متمایز از هم هستند. مدارات موجود در یک سیستم دیجیتال، به این سطوح ولتاژی خاص حساس بوده و واکنش نشان می‌دهند. مقدار این سطوح ولتاژی در حالت کلی چندان مهم نبوده و ممکن است از یک سیستم دیجیتال به سیستم دیگر متغیر باشد. یک نمونه از این سطوح در شکل زیر نشان داده شده است.

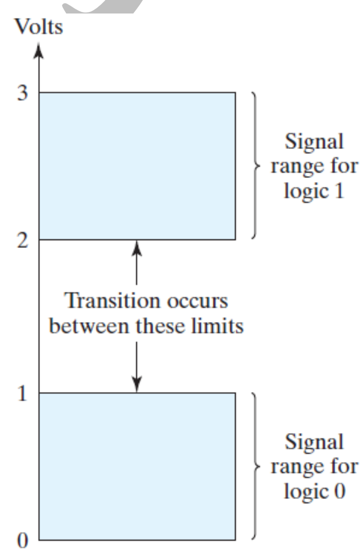
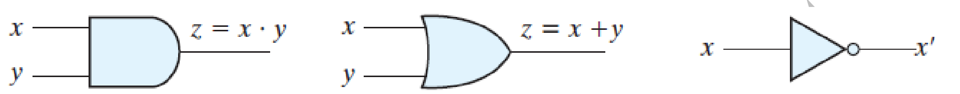



**FIGURE 1.3**
Signal levels for binary logic values

در شکل فوق، ولتاژ مدار در حالت پایدار در یکی از سطوح/حالتهای مشخص شده قرار داشته و تنها زمانی که بخواهد از یک حالت/سطح به حالت/سطح دیگر برود، ناحیه‌ی میانی را قطع می‌کند.

پایانه‌های[1] (یا پایه‌ها) ورودی مدارات دیجیتال، سیگنالهای دودویی را در محدوده‌ی مجازی پذیرفته و پایانه‌های خروجی در محدوده‌ی مجازی پاسخ می‌دهند.

سمبل‌ها یا نمادهای گرافیکی مورد استفاده برای سه نوع گیت منطقی که قبلاً معرفی شدند (یعنی AND و OR و NOT) در شکل زیر نمایش داده شده‌اند.




(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter

**FIGURE 1.4**
Symbols for digital logic circuits

نمونه‌هایی از خروجی این گیتها و عملکرد آنها در شکل زیر نشان داده شده است.

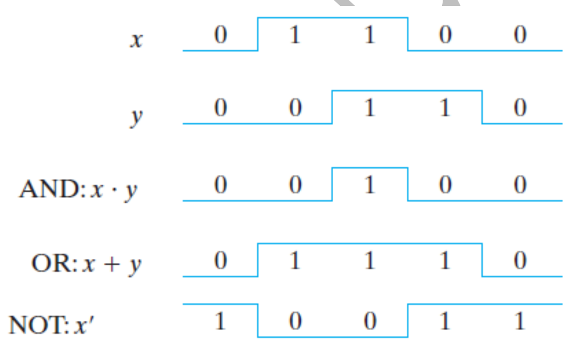
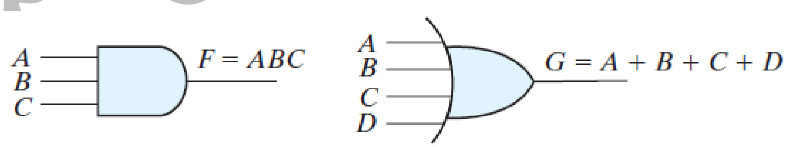



**FIGURE 1.5**
Input–output signals for gates

گیتهای فوق می‌توانند دارای بیش از یک یا دو ورودی باشند (شکل زیر).




(a) Three-input AND gate (b) Four-input OR gate

**FIGURE 1.6**
Gates with multiple inputs

[1] Terminal

# براي سلامتي رهبر انقلاب و تعجيل در ظهور حضرت ولي عصر (عج) صلوات