§1 Number Systems and Conversion

Babylonian number system (3100 B.C.)

Reference: http://www.math.ubc.ca/~cass/courses/m446-03/
Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
  - Negative number representation
    - Signed magnitude
    - 1’s complement
    - 2’s complement
    - Arithmetic under 1’s and 2’s complements
- Binary codes
Given some inputs, a digital system produces some outputs.

- It may or may not have “memory” (i.e., its outputs may or may not depend on the history of the inputs).

\[ x_1 \rightarrow \text{digital system} \rightarrow y_1 \]
\[ x_2 \rightarrow \text{digital system} \rightarrow y_2 \]
\[ \vdots \]
\[ x_m \rightarrow \text{digital system} \rightarrow y_n \]

\( x_i \) and \( y_j \) are often binary, i.e., \( x_i, y_j \in \{0,1\} \)

Given a specification constraining the input and output relation of a digital system, we’ll learn how to design a logic circuit that implements the specification.
A digital system usually contains **datapath** and **control** parts

**Datapath part**
- performs arithmetic operations, data transfer, etc.
  - Unit 1 introduces digital *data representation* and *arithmetic operation*

**Control part**
- controls when and where data should be ready in the datapath
  - Later we'll study *combinational* and *sequential circuits* for control logic

---

**Outline**

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
  - Negative number representation
    - Signed magnitude
    - 1’s complement
    - 2’s complement
    - Arithmetic under 1’s and 2’s complements
- Binary codes
Number Systems and Conversion

- Decimal number (base 10):
  - $953.78_{10}$
  - $= 9 \times 10^2 + 5 \times 10^1 + 3 \times 10^0 + 7 \times 10^{-1} + 8 \times 10^{-2}$

- Binary number (base 2):
  - $1011.11_2$
  - $= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$
  - $= 11.75_{10}$

- Octal number (base 8):
  - $147.3_8$
  - $= 1 \times 8^2 + 4 \times 8^1 + 7 \times 8^0 + 3 \times 8^{-1}$
  - $= 103.375_{10}$

- Hexadecimal number (base 16):
  - $A2F_{16}$
  - $= 10 \times 16^2 + 2 \times 16^1 + 15 \times 16^0$
  - $= 2607_{10}$

The decimal, binary, octal, and hexadecimal points separate positive and negative powers of 10, 2, 8, 16, respectively.

Number Systems and Conversion

Generalized Radix System

- Any integer $R > 1$ can be chosen as the **radix** or **base**
  - Base (radix) $R$: $(0, 1, \cdots, R-1)$, where $R > 1$
    - E.g.,
    - Base 5: $(0, 1, 2, 3, 4)$
    - Base 16: $(0, 1, 2, \ldots, 9, A, B, C, D, E, F)$

- $N_{(10)} = (a_4 \ a_3 \ a_2 \ a_1 \ a_0 \cdot a_{-1} \ a_{-2} \ a_{-3})_R$
  - $= a_4 \times R^4 + a_3 \times R^3 + a_2 \times R^2 + a_1 \times R^1 + a_0 \times R^0 + a_{-1} \times R^{-1} + a_{-2} \times R^{-2}$
  - $+ a_{-3} \times R^{-3}$

  **Note that $R$ is in base 10!**

- E.g.,
  - $A2F_{16}$
    - $= 10 \times 16^2 + 2 \times 16^1 + 15 \times 16^0$
    - $= 2607_{10}$
Number Systems and Conversion

Generalized Radix System

<table>
<thead>
<tr>
<th>Name</th>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>10</td>
<td>2</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Digits</td>
<td>0,1,2,3,4,5,6,7,8,9</td>
<td>0,1</td>
<td>0,1,2,3,4,5,6,7</td>
<td>0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F</td>
</tr>
<tr>
<td>Numbers</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>11</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1000</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>1001</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1010</td>
<td>12</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>1011</td>
<td>13</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>1100</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>1101</td>
<td>15</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>1110</td>
<td>16</td>
<td>E</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>1111</td>
<td>17</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>10000</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>

Number Systems and Conversion

Integer Conversion by Division

- $N = (a_n a_{n-1} \cdots a_2 a_1 a_0)_R$
  
  $= a_n \times R^n + a_{n-1} \times R^{n-1} + \cdots + a_2 \times R^2 + a_1 \times R^1 + a_0 \times R^0$

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

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

- $Q_2/R = a_n \times R^{n-3} + a_{n-1} \times R^{n-4} + \cdots + a_4 \times R^1 + a_3 = Q_3$
  remainder $a_2$
Number Systems and Conversion

Exercise

- Convert $53_{10}$ to binary

\[
\begin{array}{c|c}
2 & 53 \\
2 & 26 \quad \text{... remainder } 1 = a_0 \\
2 & 13 \quad \text{... remainder } 0 = a_1 \\
2 & 6 \quad \text{... remainder } 1 = a_2 \\
2 & 3 \quad \text{... remainder } 0 = a_3 \\
2 & 1 \quad \text{... remainder } 1 = a_4 \\
0 & \quad \text{... remainder } 1 = a_5 \\
\end{array}
\]

So $53_{10} = (a_5 a_4 a_3 a_2 a_1 a_0)_2 = 110101_2$

Number Systems and Conversion

Fraction Conversion by Multiplication

- $F = (a_{-1}a_{-2}a_{-3} \cdots a_m)_R$
  \[
  = a_{-1}R^{-1} + a_{-2}R^{-2} + a_{-3}R^{-3} + \cdots + a_mR^{-m}
  \]

- $F \cdot R = a_{-1} + a_{-2}R^{-1} + a_{-3}R^{-2} + \cdots + a_mR^{-m+1}$
  \[
  = a_{-1} + F_1
  \]

- $F_1 \cdot R = a_{-2} + a_{-3}R^{-1} + \cdots + a_mR^{-m+2}$
  \[
  = a_{-2} + F_2
  \]

- $F_2 \cdot R = a_{-3} + \cdots + a_mR^{-m+3}$
  \[
  = a_{-3} + F_3
  \]
Number Systems and Conversion Exercise

- Convert 0.625\(_{10}\) to binary
  
  \[
  \begin{array}{c|c|c}
  F & F1 & F2 \\
  \hline
  0.625 & 0.250 & 0.500 \\
  \times 2 & \times 2 & \times 2 \\
  1.250 & 0.500 & 1.000 \\
  \hline
  a_{-1} & a_{-2} & a_{-3}
  \end{array}
  \]

  So 0.625\(_{10}\) = (\(a_{-1} a_{-2} a_{-3}\))\(_{2}\) = 0.101\(_{2}\)

Number Systems and Conversion Exercise

- Convert 0.7\(_{10}\) to binary

  \[
  \begin{array}{c|c|c}
  0.7 & \times 2 & \text{repeat} \\
  \hline
  a_{-1} & (1)4 & (0)8 \\
  \times 2 & \times 2 & \times 2 \\
  (1)6 & (1)2 & (0)4 \\
  \times 2 & \times 2 & \times 2 \\
  (0)8 & & \text{Not finite conversion!}
  \end{array}
  \]

  So 0.7\(_{10}\) = (0.1 0110 0110 0110 \ldots)\(_{2}\)
Number Systems and Conversion Exercise

- Convert $231.3_4$ to base 7
  
  $231.3_4 = 2 \times 4^2 + 3 \times 4^1 + 1 \times 4^0 + 3 \times 4^{-1} = 45.75_{10}$

  
  $7 \mid 45$
  $7 \mid 6$ ... remainder 3 = $a_0$
  $0$ ... remainder 6 = $a_1$

  
  So $231.3_4 = (63.5151...)_7$

Decimal system is useful as an intermediate representation for such conversion (though not necessary)

Number Systems and Conversion Exercise

- Conversion from binary to Octal
  
  $101110.0011_2$
  $= 101 \ 110.001 \ 100_2$
  $= 56.14_8$

- Conversion from binary to Hexadecimal
  
  $1001101.010111_2$
  $= 0100 \ 1101.0101 \ 1100_2$
  $= 4D.5C_{16}$
Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
  - Negative number representation
    - Signed magnitude
    - 1’s complement
    - 2’s complement
    - Arithmetic under 1’s and 2’s complements
- Binary codes

Binary Arithmetic

extracting the square root of 1/2 on a binary abacus

Photo: www.binaryabacus.com
Binary Arithmetic

Addition

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 0$ (carry 1 to next column)

\[
\begin{align*}
1111 & \quad \text{carry} \\
13_{10} &= 1101_2 \\
11_{10} &= 1011_2 \\
\hline
11000_2 &= 24_{10} \\
\end{align*}
\]

Binary Arithmetic

Subtraction

- $0 - 0 = 0$
- $1 - 0 = 1$
- $1 - 1 = 0$
- $0 - 1 = 1$ (borrow 1 from next column)

\[
\begin{align*}
1 & \quad \text{borrow} \\
11101_2 & \\
- 10011_2 & \\
\hline
1010_2 &
\end{align*}
\]
Binary Arithmetic

Subtraction

More examples

\[
\begin{align*}
16_{10} &= 10000_2 \\
- 3_{10} &= 11_2 \\
\hline
1101_2 &= 13_{10}
\end{align*}
\]

\[
\begin{align*}
57_{10} &= 111001_2 \\
- 11_{10} &= 1011_2 \\
\hline
101110_2 &= 46_{10}
\end{align*}
\]

Binary Arithmetic

Multiplication

Multiplication

\[
\begin{align*}
0 \times 0 &= 0 \\
0 \times 1 &= 0 \\
1 \times 0 &= 0 \\
1 \times 1 &= 1
\end{align*}
\]

\[
\begin{align*}
13_{10} &= 1101_2 \\
\times \quad 11_{10} &= 1011_2 \\
\hline
1101 \\
1101 \\
0000 \\
1101 \\
\hline
1000111_2 &= 143_{10}
\end{align*}
\]

product
Binary Arithmetic Division

- Division of $145_{10}$ by $11_{10}$ in binary

\[
\begin{array}{c|cccccc}
& 1 & 0 & 0 & 1 & 0 & 0 \\
\hline
1 & 1 & 0 & 1 & \phantom{1} & \phantom{1} & \phantom{1} \\
1 & 0 & 1 & 1 & \phantom{1} & \phantom{1} & \phantom{1} \\
\hline
1 & 1 & 1 & 0 & \phantom{1} & \phantom{1} & \phantom{1} \\
1 & 0 & 1 & 1 & \phantom{1} & \phantom{1} & \phantom{1} \\
\hline
1 & 1 & 0 & 1 & \phantom{1} & \phantom{1} & \phantom{1} \\
1 & 0 & 1 & 1 & \phantom{1} & \phantom{1} & \phantom{1} \\
\hline
& & & & & 1 & 0 \\
\end{array}
\]

quotient: $1101$
remainder: $10$

Negative Number Representation

- The 1st bit (most significant bit, MSB) is reserved for sign indication
  - Assume base-2 system

An 8-bit word example

sign bit: 0 for + (positive)
1 for – (negative)

(Modern computers are of word size 64 or 32 bits.)
Negative Number Representation
Signed Magnitude Numbers

\[ N^{\text{sm}} = (a_{n-1}a_{n-2} \ldots a_1a_0)_2 = (-1)^{a_{n-1}} \cdot (a_{n-2} \times 2^{n-2} + \ldots + a_1 \times 2^1 + a_0 \times 2^0) \]

- 0 for + (positive)
- 1 for – (negative)

E.g., -12 = 11100 in a 5-bit word = 101100 in a 6-bit word

Negative Number Representation
1’s Complement

- For an \( n \)-bit word, the negative number \(-N\) of \( N\) in the 1’s complement is
  \[ N^{(1)} = (2^n - 1) - N \]
  - Both \( N \) and \(-N\) are in the 1’s complement (\( N \) can be positive/negative)
  - \( N^{(1)} \) is the bitwise complement of \( N \)
  - \( N = (2^n - 1) - N^{(1)} \)

E.g., \( n = 6 \) and \( N = 010101 \) (= 21)

\[
\begin{align*}
(2^n - 1) &= 111111 \\
- N &= 010101 \\
N^{(1)} &= 101010 (= -21)
\end{align*}
\]
Negative Number Representation
2’s Complement

- For an n-bit word, the negative number \(-N\) of \(N\) in the 2’s complement is
  \[ N^{(2)} = 2^n - N = N^{(1)} + 1 \]
  - Both \(N\) and \(-N\) are in the 2’s complement (\(N\) can be positive/negative)
    - Mind the marginal case when \(N = 0\) and \(N = -2^{n-1}\)
  - \(N^{(2)}\) is the bitwise complement of \(N\) plus 1
    - Effectively, only complementing every bit of \(N\) starting from the MSB until the last bit with value 1 (not applicable when \(N = 0\))

E.g., \(n = 6, N = 001110\) (14)

\[ N^{(2)} = 110001 + 000001 = 110010\] (-14)

Negative Number Representation
2’s Complement

- The value of \(N^{(2)}\) can be obtained from \(N\)
  E.g., \(n = 6\)

\[ N^{(2)} = 101010\] (\(?) \Rightarrow N = 010110\] (22) \Rightarrow N^{(2)}

represents the negative number \(-22\)

\[ N^{(2)} = 100000\] (-32) \Rightarrow no corresponding \(N\!

\[ N = 000000\] (0) \Rightarrow no corresponding \(N^{(2)}\)!
Negative Number Representation

- The signed magnitude, 1’s complement, and 2’s complement systems all have the same representation for any nonnegative number.
  - For a negative number, we need to specify which system we use (but not for a positive number).

E.g., \( n = 6 \)

\[ N^{(1)}, N^{(2)}, N^{sm} = 011111 \ (31) \]

\[ \Rightarrow N = 011111 \ (31) \]

---

Signed binary integers (word length: \( n = 4 \))

<table>
<thead>
<tr>
<th>+N</th>
<th>Positive integers (same for all systems!)</th>
<th>-N</th>
<th>Negative integers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sign and magnitude</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( N^{sm} )</td>
</tr>
<tr>
<td>+0</td>
<td>0000</td>
<td>-0</td>
<td>1000</td>
</tr>
<tr>
<td>+1</td>
<td>0001</td>
<td>-1</td>
<td>1001</td>
</tr>
<tr>
<td>+2</td>
<td>0010</td>
<td>-2</td>
<td>1010</td>
</tr>
<tr>
<td>+3</td>
<td>0011</td>
<td>-3</td>
<td>1011</td>
</tr>
<tr>
<td>+4</td>
<td>0100</td>
<td>-4</td>
<td>1100</td>
</tr>
<tr>
<td>+5</td>
<td>0101</td>
<td>-5</td>
<td>1101</td>
</tr>
<tr>
<td>+6</td>
<td>0110</td>
<td>-6</td>
<td>1110</td>
</tr>
<tr>
<td>+7</td>
<td>0111</td>
<td>-7</td>
<td>1111</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-8</td>
<td>-----</td>
</tr>
</tbody>
</table>
Negative Number Representation
Arithmetic with 2’s Complement

- Subtraction is replaced with addition
- Addition is computed as if its two operands were positive numbers
  - Carry from the sign bit is ignored (bit-length preserved)
    - E.g., \(-5\) 1011
    - \(+6\) 0110
    - \(+1\) (1)0001 
      - discard the carry
  - Result is correct when no overflow occurs (why?)
  - Overflow occurs when
    1. adding two positive numbers yields a negative number, or
    2. adding two negative numbers yields a positive number.
    - E.g., \(+5\) 0101
      - \(+6\) 0110
      - \(+1\) (1)0101 wrong! (overflow)
    - E.g., \(-5\) 1011
      - \(-6\) 1010
      - \(-1\) (1)0101 wrong! (overflow)

Negative Number Representation
Arithmetic with 2’s Complement

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A+B</th>
<th>overflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>+</td>
<td>no</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
<td>–</td>
<td>yes</td>
</tr>
<tr>
<td>+</td>
<td>–</td>
<td>+/-</td>
<td>no</td>
</tr>
<tr>
<td>–</td>
<td>+</td>
<td>+/-</td>
<td>no</td>
</tr>
<tr>
<td>–</td>
<td>–</td>
<td>+</td>
<td>yes</td>
</tr>
<tr>
<td>–</td>
<td>–</td>
<td>–</td>
<td>no</td>
</tr>
</tbody>
</table>

Checking sign-bits for overflow detection
Negative Number Representation
Arithmetic with 1’s Complement

- Similar to the addition of 2’s complement except for having end-around carry (instead of discarding the last carry)
  - Add the last carry to the sum
  
  E.g.,
  
  \[
  \begin{array}{c}
  -5 \\
  +6 \\
  \hline
  \end{array}
  \]
  
  \[
  \begin{array}{c}
  1010 \\
  0110 \\
  \hline
  \end{array}
  \]
  
  \[
  (1) 0000 \\
  \rightarrow 1 \\
  \hline
  0001
  \]
  
  (end-around carry)

  \[
  \begin{array}{c}
  +5 \\
  +6 \\
  \hline
  \end{array}
  \]
  
  \[
  \begin{array}{c}
  0101 \\
  0110 \\
  \hline
  \end{array}
  \]
  
  \[
  1011 \\
  \]
  
  wrong! (overflow)

  \[
  \begin{array}{c}
  -5 \\
  -6 \\
  \hline
  \end{array}
  \]
  
  \[
  \begin{array}{c}
  1010 \\
  1001 \\
  \hline
  \end{array}
  \]
  
  \[
  (1) 0011 \\
  \rightarrow 1 \\
  \hline
  0100 \\
  \]
  
  wrong! (overflow)

Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
  - Negative number representation
    - Signed magnitude
    - 1’s complement
    - 2’s complement
    - Arithmetic under 1’s and 2’s complements
- Binary codes
Binary Codes

Use binary codes to represent *symbols*
- Not associated with arithmetic
- Different from the *binary number system*

E.g.,

3.1415

0011 . 0001 0100 0001 0101

in Binary Coded Decimal (BCD)

Different from converting the number as a whole into binary!
## Binary Codes

### Common binary codes for decimal digits

<table>
<thead>
<tr>
<th>Decimal digit</th>
<th>8-4-2-1 code (BCD)</th>
<th>6-3-1-1 code</th>
<th>Excess-3 code (BCD+3)</th>
<th>2-out-of-5 code (good for error checking)</th>
<th>Gray code (good for low power and reliability)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0000</td>
<td>0011</td>
<td>00011</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>0001</td>
<td>0100</td>
<td>00101</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>0011</td>
<td>0101</td>
<td>00110</td>
<td>0011</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>0100</td>
<td>0110</td>
<td>01001</td>
<td>0010</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>0101</td>
<td>0111</td>
<td>01010</td>
<td>0110</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>0111</td>
<td>1000</td>
<td>01100</td>
<td>1110</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>1000</td>
<td>1001</td>
<td>10001</td>
<td>1010</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>1001</td>
<td>1010</td>
<td>10010</td>
<td>1011</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1011</td>
<td>1011</td>
<td>10100</td>
<td>1001</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>1100</td>
<td>1100</td>
<td>11000</td>
<td>1000</td>
</tr>
</tbody>
</table>

\[ a_3a_2a_1a_0 \text{ in } w_3w_2w_1w_0 \text{ (4-bit weighted) code represents } N = w_3a_3 + w_2a_2 + w_1a_1 + w_0a_0 \]

E.g., 1011 in 6-3-1-1 code represents \( N = 6 \cdot 1 + 3 \cdot 0 + 1 \cdot 1 + 1 \cdot 1 = 8 \)

### ASCII code

- 7 bits
- Used to represent computer characters
Binary Codes

<table>
<thead>
<tr>
<th>ASCII printable characters (decimal code, character)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
</tr>
<tr>
<td>33</td>
</tr>
<tr>
<td>34</td>
</tr>
<tr>
<td>35</td>
</tr>
<tr>
<td>36</td>
</tr>
<tr>
<td>37</td>
</tr>
<tr>
<td>38</td>
</tr>
<tr>
<td>39</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>41</td>
</tr>
<tr>
<td>42</td>
</tr>
<tr>
<td>43</td>
</tr>
<tr>
<td>44</td>
</tr>
<tr>
<td>45</td>
</tr>
<tr>
<td>46</td>
</tr>
<tr>
<td>47</td>
</tr>
<tr>
<td>48</td>
</tr>
<tr>
<td>49</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>51</td>
</tr>
<tr>
<td>52</td>
</tr>
<tr>
<td>53</td>
</tr>
<tr>
<td>54</td>
</tr>
<tr>
<td>55</td>
</tr>
<tr>
<td>56</td>
</tr>
<tr>
<td>57</td>
</tr>
<tr>
<td>58</td>
</tr>
<tr>
<td>59</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>61</td>
</tr>
<tr>
<td>62</td>
</tr>
<tr>
<td>63</td>
</tr>
</tbody>
</table>

Other Binary Codes

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.
Other Binary Codes