Ex. 8.4  7-4-2-1 code

Codeconverter 7-4-2-1-code to BCD-code.

When encoding the digits 0 ... 9 sometimes in the past a code having weights 7-4-2-1 instead of the binary code weights 8-4-2-1 was used.

In the cases where a digit's code word can be expressed in various ways the code word that contains the least number of ones is selected

(A variation of the 7-4-2-1 code is used today to store the bar code)
Ex. 8.4  7-4-2-1 code

Codeconverter 7-4-2-1-code to BCD-code.

When encoding the digits 0 ... 9 sometimes in the past a code having weights 7-4-2-1 instead of the binary code weights 8-4-2-1 was used.

In the cases where a digit's code word can be expressed in various ways the code word that contains the least number of ones is selected

(A variation of the 7-4-2-1 code is used today to store the bar code)
Ex. 8.4  7-4-2-1 code

Codeconverter 7-4-2-1-code to BCD-code.

When encoding the digits 0 ... 9 sometimes in the past a code having weights 7-4-2-1 instead of the binary code weights 8-4-2-1 was used.

In the cases where a digit's code word can be expressed in various ways the code word that contains the least number of ones is selected

(A variation of the 7-4-2-1 code is used today to store the bar code)

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₀</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x₁</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x₂</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x₃</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x₄</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x₅</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x₆</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>x₇</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>x₈</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x₉</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

William Sandqvist william@kth.se
### Code converter

| \( x_7 \ x_4 \ x_2 \ x_1 \) | \( y_8 \ y_4 \ y_2 \ y_1 \) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |

#### Table 8.4

<table>
<thead>
<tr>
<th>( x_7 )</th>
<th>( y_8 )</th>
<th>( y_4 )</th>
<th>( y_2 )</th>
<th>( y_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>001</td>
<td>001</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>01</td>
<td>045</td>
<td>045</td>
<td>045</td>
<td>045</td>
</tr>
<tr>
<td>10</td>
<td>112</td>
<td>112</td>
<td>112</td>
<td>112</td>
</tr>
<tr>
<td>11</td>
<td>118</td>
<td>118</td>
<td>118</td>
<td>118</td>
</tr>
</tbody>
</table>

---

William Sandqvist william@kth.se
<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th></th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(2)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(3)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(4)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(5)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(6)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(7)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(8)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(9)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

8.4  
Code converter

<table>
<thead>
<tr>
<th>7-4-2-1</th>
<th>8-4-2-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_7$</td>
<td>$x_4$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y_8$</th>
<th>$y_4$</th>
<th>$y_2$</th>
<th>$y_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

William Sandqvist william@kth.se
8.4 Code converter

\[ y_8 = x_7 x_2 + x_7 x_1 \]
8.4  Code converter

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
x_7 & x_4 & x_2 & x_1 & y_8 & y_4 & y_2 & y_1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 2 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 3 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 4 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 5 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 6 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 7 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 8 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 9 & 1 & 0 & 1 \\
\hline
\end{array}
\]

\[
y_8 = x_7 x_2 + x_7 x_1 \\
y_4 = x_4 + x_7 x_2 x_1
\]
### 8.4 Code converter

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th></th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_7</td>
<td>x_4</td>
<td>x_2</td>
<td>x_1</td>
<td></td>
<td>y_8</td>
<td>y_4</td>
<td>y_2</td>
<td>y_1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

#### y_8 = x_7 x_2 + x_7 x_1

#### y_4 = x_4 + x_7 x_2 x_1

#### y_2 = x_7 x_2 + x_7 x_2 x_1

#### y_1 = x_7 x_1 + x_7 x_2 + x_7 x_2 x_1
Common groupings can provide for shared gates!

\[ y_8 = x_7x_2 + x_7x_1 \]
\[ y_4 = x_4 + x_7x_2x_1 \]
\[ y_2 = x_7x_2 + x_7x_2x_1 \]
\[ y_1 = x_7x_1 + x_7x_2 + x_7x_2x_1 \]
8.4

PLA circuits containing programmable AND and OR gates. (This turned out to be unnecessarily complex, so the common chips became PAL circuits with only the AND network programmable).

The gates have many programmable input connections. The many inputs are usually drawn in a "simplified" way.
\[ y_8 = \overline{x_7 x_2} + x_7 x_1 \]
\[ y_4 = x_4 + x_7 x_2 x_1 \]
\[ y_2 = \overline{x_7 x_2} + \overline{x_7 x_2 x_1} \]
\[ y_1 = \overline{x_7 x_1} + \overline{x_7 x_2} + \overline{x_7 x_2 x_1} \]

8.4

Shared-gates!

William Sandqvist william@kth.se
8.4

Shared-gates!

\[ y_8 = x_7 x_2 + x_7 x_1 \]
\[ y_4 = x_4 + x_7 x_2 x_1 \]
\[ y_2 = x_7 x_2 + x_7 x_2 x_1 \]
\[ y_1 = x_7 x_1 + x_7 x_2 + x_7 x_2 x_1 \]
8.4

Shared-gates!

\[
\begin{align*}
y_8 &= x_7 \overline{x}_2 + x_7 x_1 \\
y_4 &= x_4 + x_7 x_2 \overline{x}_1 \\
y_2 &= \overline{x}_7 x_2 + x_7 x_2 x_1 \\
y_1 &= \overline{x}_7 x_1 + x_7 x_2 + x_7 x_2 x_1
\end{align*}
\]
Real numbers

Decimal point ""," and Binary point "."

$10.3125_{10} = 1010.0101_2$

<table>
<thead>
<tr>
<th>Bin</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010.0101</td>
<td>10.3125</td>
</tr>
</tbody>
</table>

$2^3 2^2 2^1 2^0 \cdot 2^{-1} 2^{-2} 2^{-3} 2^{-4}$

| 8 | 4 | 2 | 1 | 0.5 | 0.25 | 0.125 | 0.0625 |

$8 + 0 + 2 + 0 + 0 + 0.25 + 0 + 0.0625 = 10.3125$
Ex. 1.2b

\[ 110100.010_2 = \]
Ex. 1.2b

\[
110100.010_2 = \\
= (2^5 + 2^4 + 2^2 + 2^{-2} = 32 + 16 + 4 + 0.25) = \\
= 52.25_{10}
\]
Calculation with complement

Subtraction with an adding machine = counting with the complement

\[
\begin{array}{c}
\begin{array}{c}
63 \\
-17 \\
\hline
46
\end{array}
\end{array}
\]

63 - 17 = 46

The number -17 is entered with red digits 17 and gets 82. When the – key is pressed 1 is added. The result is: 63+82+1 = 146. If only two digits are shown: 46
2-complement

The binary number 3, 0011, gets negative -3 if one inverts the digits and adds one, 1101.
Register arithmetic

- Computer registers are "rings"

A four bit register could contains $2^4 = 16$ numbers.

Either 8 positive (+0…+7) and 8 negative (-1…-8) "signed integers", or 16 (0…F) "unsigned integers".

If the register is full +1 makes the register to the "turn around".
Register width

• 4 bit is called a Nibble. The register contains $2^4 = 16$ numbers. 0…15, -8…+7

• 8 bit is called a Byte. The register contains $2^8 = 256$ numbers 0…255, -128…+127

• 16 bit is a Word. $2^{16} = 65536$ numbers. 0…65535, -32768…+32767

Today, general sizes are now 32 bits (Double Word) and 64 bits (Quad Word).
Ex. 1.8

Write the following signed numbers with two's complement notation, $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.

a) $-23$

b) $-1$

c) $+38$

d) $-64$
Ex. 1.8

Write the following signed numbers with two's complement notation, $x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0)$.

a) $-23 = (+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2 + 1_2 ) = 1101001_2$
   $= 105_{10}$

b) $-1 =$

c) $+38 =$

d) $-64 =$
Ex. 1.8

Write the following signed numbers with two's complement notation, \( x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0) \).

a) \(-23 = (+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2 + 1_2) = 1101001_2 = 105_{10}\)

b) \(-1 = (+1_{10} = 0000001_2 \rightarrow -1_{10} = 1111110_2 + 1_2) = 1111111_2 = 127_{10}\)

c) \(+38 = \)

d) \(-64 = \)
Ex. 1.8

Write the following signed numbers with two's complement notation, \( x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0) \).

a) \(-23 = (+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2 + 1_2) = 1101001_2\)
   \(= 105_{10}\)

b) \(-1 = (+1_{10} = 0000001_2 \rightarrow -1_{10} = 1111110_2 + 1_2) = 1111111_2 = 127_{10}\)

c) \(+38 = (32_{10} + 4_{10} + 2_{10}) = 0100110_2 = 38_{10}\)

d) \(-64 = \)
Ex. 1.8

Write the following signed numbers with two's complement notation, \( x = (x_6, x_5, x_4, x_3, x_2, x_1, x_0) \).

a) \(-23 = (+23_{10} = 0010111_2 \rightarrow -23_{10} = 1101000_2 + 1_2) = 1101001_2 = 105_{10}\)

b) \(-1 = (+1_{10} = 0000001_2 \rightarrow -1_{10} = 1111110_2 + 1_2) = 1111111_2 = 127_{10}\)

c) \(+38 = (32_{10} + 4_{10} + 2_{10}) = 0100110_2 = 38_{10}\)

d) \(-64 = (+64_{10} = 1000000_2 \text{ is a too big positive number (for 7 bits)!} \rightarrow -64_{10} = 0111111_2 + 1_2) = 1000000_2 = 64_{10}\)
Ex. 2.1

a) $110 + 010$

b) $1110 + 1001$

c) $110011.01 + 111.1$

d) $0.1101 + 0.1110$

\[\begin{array}{c}
a) \quad 110 \\
\quad + 010 \\
\hline
\quad 10000 \\

b) \quad 1110 \\
\quad + 1001 \\
\hline
\quad 10111 \\

c) \quad 110011.01 \\
\quad + 111.1 \\
\hline
\quad 111010.11 \\

d) \quad 0.1101 \\
\quad + 0.1110 \\
\hline
\quad 0.11011
\end{array}\]
Full adder
A logic circuit that makes a binary addition on any bit position with two binary numbers is called a full adder.
4-bit adder

An addition circuit for binary four bit numbers thus consists of four fulladder circuits.
Subtraction?

Subtracting the binary numbers can be done with the two-complement. Negative numbers are represented as the true complement, which means that all bits are inverted and a one is added. The adder is then used also for subtraction.

The inversion of the bits could be done with XOR-gates, and a one could then be added to the number by letting $C_{IN} = 1$. 

$$f = a \oplus b$$
Figure 5.13. Adder/subtractor unit.
In order to easily produce 2's complement of a binary number, you can use the following procedure:

- Start from right
- Copy all bits from all zeroes to the first 1.
- Invert all the rest of the bits

Example: 2-complement of 0110 is 1010
Ex. 2.2

Add or subtract (add with the corresponding negative number) the numbers below. The numbers are represented as binary 2-complement 4-bit numbers (nibble).

a) $1 + 2$  
b) $4 - 1$  
c) $7 - 8$  
d) $-3 - 5$

The negative number that are used in the examples:

$-1_{10} = (+1_{10} = 0001_2 \rightarrow -1_{10} = 1110_2 +1_2 ) = 1111_2$

$-8_{10} = (+8_{10} = 1000_2 \rightarrow -8_{10} = 0111_2 +1_2 ) = 1000_2$

$-3_{10} = (+3_{10} = 0011_2 \rightarrow -3_{10} = 1100_2 +1_2 ) = 1101_2$

$-5_{10} = (+5_{10} = 0101_2 \rightarrow -5_{10} = 1010_2 +1_2 ) = 1011_2$
2.2

-1₁₀ = 1111₂
-8₁₀ = 1000₂

1+2=3

\[
\begin{array}{c}
a) \quad \begin{array}{c}
0 & 0 & 0 & 1 \\
+ & 0 & 0 & 1 & 0 \\
\hline 
0 & 0 & 1 & 1 \\
\end{array} = 3 \\
\end{array}
\]

4-1=3

\[
\begin{array}{c}
b) \quad \begin{array}{c}
0 & 1 & 0 & 0 \\
+ & 1 & 1 & 1 & 1 \\
\hline 
0 & 0 & 1 & 1 \\
\end{array} = -1 \\
\end{array}
\]

7-8=-1

\[
\begin{array}{c}
c) \quad \begin{array}{c}
0 & 1 & 1 & 1 \\
+ & 1 & 0 & 0 & 0 \\
\hline 
1 & 1 & 1 & 1 \\
\end{array} = -1 \\
\end{array}
\]

-3-5=-8

\[
\begin{array}{c}
d) \quad \begin{array}{c}
1 & 1 & 1 & 1 \\
+ & 1 & 0 & 1 & 1 \\
\hline 
1 & 0 & 0 & 0 \\
\end{array} = -8 \\
\end{array}
\]

-3₁₀ = 1101₂
-5₁₀ = 1011₂
Ex. 2.3 a,b

Multiplicate by hand the following pairs of unsigned binary numbers.

a) 110·010  
   \[
   \begin{array}{c}
   \phantom{1} 1 1 0 \\
   \times \phantom{1} 0 1 0 \\
   \hline
   \phantom{1} 0 0 0 \\
   \phantom{1} 1 1 0 \\
   \hline
   \phantom{1} 0 1 1 0 0
   \end{array}
   \]
   \[=12\]
   

b) 1110·1001  
   \[
   \begin{array}{c}
   1 1 1 0 \\
   \times \phantom{1} 1 0 0 1 \\
   \hline
   \phantom{1} 1 1 1 0 \\
   \phantom{1} 0 0 0 0 \\
   \phantom{1} 0 0 0 0 \\
   \hline
   \phantom{1} 1 1 1 0 0 0 0
   \end{array}
   \]
   \[=126\]
Ex. 2.3 c,d

Multiplicate by hand the following pairs of unsigned binary numbers.

c) \[110011.01 \cdot 111.1 = 110000000.011\]

\[
\begin{array}{c}
110011.01 \\
\times \quad 111.1
\end{array}
\]
\[
\begin{array}{c}
11001101 \\
11001101 \\
11001101 \\
+ 11001101
\hline
1100000000.011
\end{array}
\]

=110000000.011

\[0.1101 \cdot 0.1110 = 0.10110110\]

\[
\begin{array}{c}
0.1101
\times
\quad 0.1110
\end{array}
\]
\[
\begin{array}{c}
1101 \\
\quad 0000 \\
1101 \\
1101 \\
+ 1101
\hline
10110110
\end{array}
\]

=0.10110110

(51.25 \cdot 7.5 = 384.376)

(0.8125 \cdot 0.875 = 0.7109375)

Fixpoint multiplication is an "integer multiplication", the binarypoint is inserted in the result.
Ex. 2.4

Divide by hand the following pairs of unsigned binary numbers.

*Method the Stairs:*

\[ 110 / 010 = (6 / 2 = 3) = 011 \]

a) \[
\begin{array}{c}
10 \\
110 \\
- 10 \\
10 \\
- 10 \\
0
\end{array}
\]
Ex. 2.4

Divide by hand the following pairs of unsigned binary numbers.

Method the Stairs:

110/010 = (6/2 = 3) = 011

a) \[ \begin{array}{c|c c c} & 1 & 1 & 0 \\ \hline 1 & 0 & \overline{1} & 1 & 0 \\ 1 & 0 & - & \hline 1 & 0 & \overline{1} & 0 \\ 1 & 0 & - & \hline 0 & \end{array} \]

1110/1001 = (14/9) = 1.10 ...

b) \[ \begin{array}{c|c c c} & 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & \overline{1} & 1 & 0 & 1 & 0 & - & \hline 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & - & \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \end{array} \]

If integer division the answer will be 1.

William Sandqvist william@kth.se
Ex 2.4

Divide by hand the following pairs of unsigned binary numbers.

Method Short division:

a) $110/010=(6/2=3)=011$

\[
\begin{array}{c}
110 \\
\underline{010} \\
\hline
110 \\
\underline{110} \\
\hline
110 \\
\underline{110} \\
\hline
11
\end{array}
\]
Ex 2.4

Divide by hand the following pairs of unsigned binary numbers.

*Method Short division:*

b) 1110/1001 = (14/9 = 1,55...) = 1.10...

\[
\begin{align*}
\frac{1110}{1001} &= 101 \\
\frac{1110}{1001} &= 1 \\
\frac{1110.}{1001} &= 1. \\
\frac{1110.}{1001} &= 1.1 \\
\end{align*}
\]

If integer division the answer will be 1.

William Sandqvist william@kth.se
IEEE – 32 bit float

The exponent is written exess-127. It is then possible to sort float by size with ordinary integer arithmetic!

Dec → IEEE-754

William Sandqvist william@kth.se
2.5 Float format

IEEE 32 bit float

s  eeeeeee e ffffffffffffffffffffffffffffffffffffffffff
31 30    23 22 0
2.5 Float format

IEEE 32 bit float

s  eeeeeeeee  fffffffffffffffffffffffffffffffffffffffffffffffffffffffff
31 30  23 22  0

What is:

4  0  C  8  0  0  0  0  0
0100000011001000000000000000000000000000
2.5 Float format

IEEE 32 bit float

s  eeeeee  ffffffffffffffff
31 30    23 22                    0

What is:

0 10000001 11001000000000000000000000000000

0 10000001 10010000000000000000000000000000

+ 129-127    1 + 0.5+0.0625
2.5 Float format

IEEE 32 bit float

\[ s \quad eeeeeee \quad fffffffffffffffffffffffffffffffffffffff \]

31 30 23 22 0

What is:

\[ 01000011100100000000000000000000 \]

\[ 0 \quad 10000001 \quad 10010000000000000000000000000000 \]

+ 129–127 1 + 0.5+0.0625

\[ +1,5625 \cdot 2^2 = +6,25 \]

William Sandqvist william@kth.se
IEEE-754 Floating-Point Conversion

From 32-bit Hexadecimal Representation
To Decimal Floating-Point
Along with the Equivalent 64-bit Hexadecimal and Binary Patterns

Enter the 32-bit hexadecimal representation of a floating-point number here, then click the Compute button.

Hexadecimal Representation: 40C80000

Results:

Decimal Value Entered: 6.25

Single precision (32 bits):

Binary:

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bits 30 - 23</th>
<th>Bits 22 - 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign Bit</td>
<td>Exponent Field</td>
<td>Significand</td>
</tr>
<tr>
<td>0: +</td>
<td>Exponent Field = 1000001</td>
<td>Significand = 1.10010000000000000000000000000000</td>
</tr>
<tr>
<td>1: -</td>
<td>Decimal value of exponent field and exponent = 123 - 127 = 2</td>
<td>Decimal value of the significand = 1.5625000</td>
</tr>
</tbody>
</table>

Dec → IEEE-754

William Sandqvist william@kth.se
Figure 5.34. IEEE Standard floating-point formats.

(a) Single precision
- Sign (S)
- Exponent (E): 8-bit excess-127
- Mantissa (M): 23 bits

(b) Double precision
- Sign (S)
- Exponent (E): 11-bit excess-1023
- Mantissa (M): 52 bits
When using signed numbers the sum of two positive numbers could be incorrectly negative (eg. "+4" + "+5" = "-7"), in the same way the sum of two negative numbers could incorrectly be positive (eg. "-6" + "-7" = "+3").

This is called Overflow.
Logic to detect overflow

For 4-bit-numbers

Overflow if $c_3$ and $c_4$ are different
Otherwise it’s not overflow

$$\text{Overflow} = c_3 \bar{c}_4 + \bar{c}_3 c_4 = c_3 \oplus c_4$$

XOR detects "not equal"

For $n$-bit-numbers

$$\text{Overflow} = c_{n-1} \oplus c_n$$
Figure 5.42. A comparator circuit.
Flags, Comparator. Two four-bit signed numbers, $X = x_3x_2x_1x_0$ and $Y = y_3y_2y_1y_0$, can be compared by using a subtractor circuit, which performs the operation $X - Y$. The three Flag-outputs denote the following:

- $Z = 1$ if the result is 0; otherwise $Z = 0$
- $N = 1$ if the result is negative; otherwise $N = 0$
- $V = 1$ if arithmetic overflow occurs; otherwise $V = 0$

Show how $Z$, $N$, and $V$ can be used to determine the cases

- $X = Y$
- $X < Y$
- $X > Y$

Subtractor circuit used as comparator.
BV ex 5.10

\[ X - Y \]
\[ V = c_4 \oplus c_3 \quad N = s_3 \]
\[ Z = (s_3 + s_2 + s_1 + s_0) \]

William Sandqvist william@kth.se
BV ex 5.10

\[ X - Y \]

\[ V = c_4 \oplus c_3 \quad N = s_3 \]

\[ Z = (s_3 + s_2 + s_1 + s_0) \]

\[ X = Y \quad \Rightarrow \quad Z = 1 \]
Some test numbers:

<table>
<thead>
<tr>
<th>$X &lt; Y$</th>
<th>$X - Y$</th>
<th>$V$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 4</td>
<td>3 - 4 = -1</td>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td>-4 -3</td>
<td>-4 - (-3) = -1</td>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td>-3 4</td>
<td>-3 - 4 = -7</td>
<td>0 1</td>
<td>0 1</td>
</tr>
<tr>
<td>-5 4</td>
<td>-5 - 4 = 7</td>
<td>1 0</td>
<td>0 1</td>
</tr>
</tbody>
</table>

$X - Y$

$V = c_4 \oplus c_3 \quad N = s_3$

$Z = (s_3 + s_2 + s_1 + s_0)$
If $X$ and $Y$ has the same sign $X - Y$ will always be correct and the flag $V = 0$. $X$, $Y$ positive eg. $3 - 4$ $N = 1$. $X$, $Y$ negative eg. $-4 - (-3)$ $N = 1$.

If $X$ neg and $Y$ pos and $X - Y$ has the correct sign, $V = 0$ and $N = 1$.

Tex. $-3 - 4$.

If $X$ neg and $Y$ but $X - Y$ gets the wrong sign, $V = 1$.

Then $N = 0$. Ex. $-5 - 4$.

- Summary: when $X < Y$ the flags $V$ and $N$ is always different. This could be indicated by a XOR gate.
BV ex 5.10

\[ X - Y \]
\[ V = c_4 \oplus c_3 \quad N = s_3 \]
\[ Z = (s_3 + s_2 + s_1 + s_0) \]

\( X \leq Y \) ?

If \( X \) and \( Y \) has the same sign \( X - Y \) will always be correct and the flag \( V = 0 \). \( X \), \( Y \) positive eg. \( 3 - 4 \) \( N = 1 \). \( X \), \( Y \) negative eg. \( -4 - (-3) \) \( N = 1 \).

If \( X \) neg and \( Y \) pos and \( X - Y \) has the correct sign, \( V = 0 \) and \( N = 1 \). Tex. \( -3 - 4 \).

If \( X \) neg and \( Y \) but \( X - Y \) gets the wrong sign, \( V = 1 \). Then \( N = 0 \). Ex. \( -5 - 4 \).

- Summary: when \( X < Y \) the flags \( V \) and \( N \) is always different. This could be indicated by a XOR gate.

\[ X < Y \quad \Rightarrow \quad N \oplus V \]

William Sandqvist william@kth.se
BV ex 5.10

\[ X - Y \]

\[ V = c_4 \oplus c_3 \quad N = s_3 \]

\[ Z = (s_3 + s_2 + s_1 + s_0) \]

\[ X = Y \quad \Rightarrow \quad Z = 1 \]

\[ X < Y \quad \Rightarrow \quad N \oplus V \]

\[ X \leq Y \quad \Rightarrow \quad \]

\[ X > Y \quad \Rightarrow \quad \]

\[ X \geq Y \quad \Rightarrow \quad \]
BV ex 5.10

\[ X - Y \]
\[ V = c_4 \oplus c_3 \quad N = s_3 \]
\[ Z = (s_3 + s_2 + s_1 + s_0) \]

\[ X = Y \quad \Rightarrow \quad Z = 1 \]
\[ X < Y \quad \Rightarrow \quad N \oplus V \]
\[ X \leq Y \quad \Rightarrow \quad Z + N \oplus V \]
\[ X > Y \quad \Rightarrow \quad Z + N \oplus V = \bar{Z} \cdot (N \oplus V) \]
\[ X \geq Y \quad \Rightarrow \quad N \oplus V \]
This is how a computer can perform the most common comparisons:

\[
\begin{align*}
X = Y & \implies Z = 1 \\
X < Y & \implies N \oplus V \\
X \leq Y & \implies Z + N \oplus V \\
X > Y & \implies Z + N \oplus V = \overline{Z} \cdot (N \oplus V) \\
X \geq Y & \implies N \oplus V
\end{align*}
\]

\[X - Y\]
\[V = c_4 \oplus c_3 \quad N = s_3\]
\[Z = (s_3 + s_2 + s_1 + s_0)\]
A four bit unsigned integer $x$ ($x_3x_2x_1x_0$) is connected to an 4-bit adder as in the figure. The result is a 5-bit number $y$ ($y_4y_3y_2y_1y_0$). Draw the figure to the right how the same results can be obtained without using the adder. There are also bits with the values 0 and 1 if needed.

William Sandqvist william@kth.se
(Ex 8.12) Adder circuit

\[ x + x = 2 \cdot x \]
Ex 8.11 Multiply with 6?

\[ s = 6 \times x = 2 \times (2 \times x + 1 \times x) \]
Ex 8.11 Multiply with 6!
Ex 8.11 Multiply with 6!
Ex 8.11 Multiply with 6!

2 \cdot (x \cdot 2 + x \cdot 1)
Ex 8.11 Multiply with 6!

\[ 15 \cdot 6 = 90 \]

\[ 1111 = 15 \]

\[ x_3 x_2 x_1 x_0 \]

\[ x \cdot 2 \]

\[ x \cdot 1 \]

\[ \begin{array}{ccccc}
1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 \\
\end{array} \]

\[ 15 \times 2 \]

\[ 15 \times 1 \]

\[ \times 2 \]

\[ 0 \]

\[ 2 \cdot (x \cdot 2 + x \cdot 1) \]

\[ 1011010 = 90 \]