DIGITAL LOGIC DESIGN I
CS1026

Paula.Roberts@scss.tcd.ie

NUMBER SYSTEMS

The position of each digit in a weighted number system is assigned a weight based on the **base** or **radix** of the system. The radix of decimal numbers is ten, because only ten symbols (0 through 9) are used to represent any number.

The column weights of decimal numbers are powers of ten that increase from right to left beginning with $10^0 = 1$:…$10^5\ 10^4\ 10^3\ 10^2\ 10^1\ 10^0$**.** For fractional decimal numbers, the column weights are negative powers of ten that decrease from left to right:

$$10^2\ 10^1\ 10^0.\ 10^{-1}\ 10^{-2}\ 10^{-3}\ 10^{-4}\ \ldots$$

Decimal Numbers

Decimal numbers can be expressed as the sum of the products of each digit times the column value for that digit. Thus, the number 9240 can be expressed as

$$(9 \times 10^3) + (2 \times 10^2) + (4 \times 10^1) + (0 \times 10^0)$$

or

$$9 \times 1{,}000 + 2 \times 100 + 4 \times 10 + 0 \times 1$$

Example

Express the number 480.52 as the sum of values of each digit.

Solution

$$480.52 = (4 \times 10^2) + (8 \times 10^1) + (0 \times 10^0) + (5 \times 10^{-1}) + (2 \times 10^{-2})$$

For digital systems, the binary number system is used. Binary has a radix of two an d uses the digits 0 and 1 to represent quantities.

The column weights of binary numbers are powers of two that increase from right to left beginning with 2^0 =1:

$$\ldots 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0.$$

For fractional binary numbers, the column weights are negative powers of two that decrease from left to right:

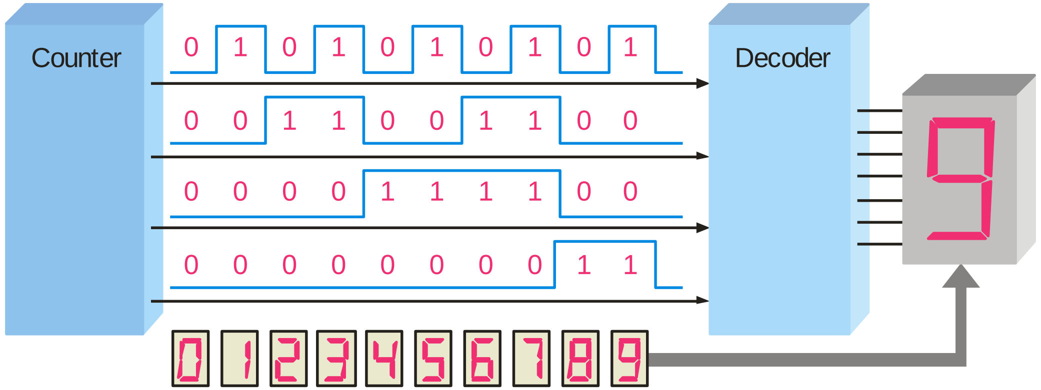
$$2^2\ 2^1\ 2^0.\ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4} \ldots$$

Binary Numbers

A binary counting sequence for numbers from zero to fifteen is shown.

Notice the pattern of zeros and ones in each column.

Digital counters frequently have this same pattern of digits:



Decimal Number	Binary Number
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Binary Conversions

The decimal equivalent of a binary number can be determined by adding the column values of all of the bits that are 1 and discarding all of the bits that are 0.

Example

Convert the binary number 100101.01 to decimal.

Solution

Start by writing the column weights; then add the weights that correspond to each 1 in the number.

2^5	2^4	2^3	2^2	2^1	2^0.	2^{-1}	2^{-2}
32	16	8	4	2	1 .	½	¼
1	0	0	1	0	1.	0	1
32			+4		+1		+¼ = 37¼

Binary Conversions

You can convert a decimal whole number to binary by reversing the procedure. Write the decimal weight of each column and place 1's in the columns that sum to the decimal number.

Example Convert the decimal number 49 to binary.

Solution The column weights double in each position to the right. Write down column weights until the last number is larger than the one you want to convert.

2^6	2^5	2^4	2^3	2^2	2^1	2^0.
64	32	16	8	4	2	1.
0	1	1	0	0	0	1.

Binary Conversions

You can convert a decimal fraction to binary by repeatedly multiplying the fractional results of successive multiplications by 2. The carries form the binary number.

Example

Convert the decimal fraction 0.188 to binary by repeatedly multiplying the fractional results by 2.

Solution

0.188 x 2 = 0.376	carry = 0	MSB
0.376 x 2 = 0.752	carry = 0	
0.752 x 2 = 1.504	carry = 1	
0.504 x 2 = 1.008	carry = 1	
0.008 x 2 = 0.016	carry = 0	

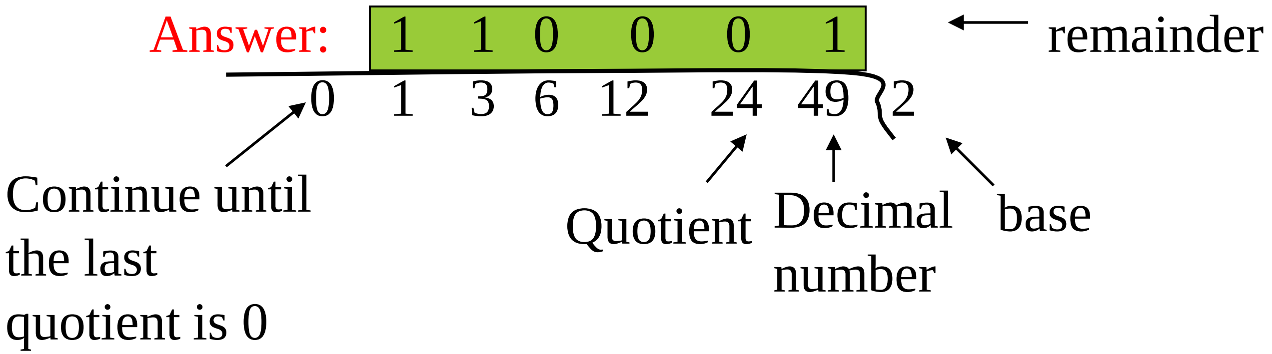
Answer = .00110 (for five significant digits)

Binary Conversions

You can convert decimal to any other base by repeatedly dividing by the base. For binary, repeatedly divide by 2:

Example Convert the decimal number 49 to binary by repeatedly dividing by 2.

Solution You can do this by "reverse division" and the answer will read from left to right. Put quotients to the left and remainders on top.



Binary Addition

The rules for binary addition are

0 + 0 = 0	Sum = 0, carry = 0
0 + 1 = 1	Sum = 1, carry = 0
1 + 0 = 1	Sum = 1, carry = 0
1 + 1 = 10	Sum = 0, carry = 1

When an input carry = 1 due to a previous result, the rules are

1 + 0 + 0 = 01	Sum = 1, carry = 0
1 + 0 + 1 = 10	Sum = 0, carry = 1
1 + 1 + 0 = 10	Sum = 0, carry = 1
1 + 1 + 1 = 11	Sum = 1, carry = 1

Binary Addition

Example Add the binary numbers 00111 and 10101 and show the equivalent decimal addition.

Solution

```
 0111
 00111      7
 10101     21
 -----    ---
 11100  =  28
```

Binary Subtraction

The rules for binary subtraction are

$0 - 0 = 0$
$1 - 1 = 0$
$1 - 0 = 1$
$10 - 1 = 1$ with a borrow of 1

Example Subtract the binary number 00111 from 10101 and show the equivalent decimal subtraction.

Solution

$$\begin{array}{r} {\scriptstyle 1\,1\,1} \\ \cancel{1}\cancel{0}1\cancel{0}1 \\ 00111 \\ \hline 01110 \end{array} \quad = \quad \begin{array}{r} 21 \\ 7 \\ \hline 14 \end{array}$$

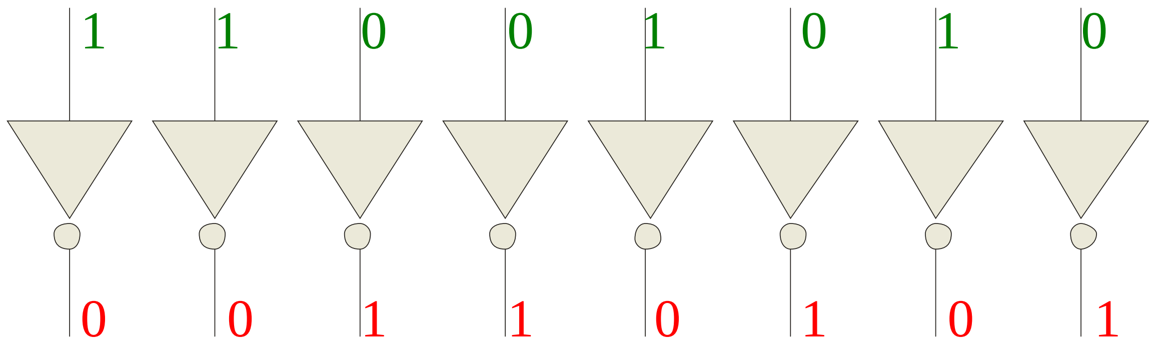
1's Complement

The 1's complement of a binary number is just the inverse of the digits. To form the 1's complement, change all 0's to 1's and all 1's to 0's.

For example, the 1's complement of 11001010 is

00110101

In digital circuits, the 1's complement is formed by using inverters:



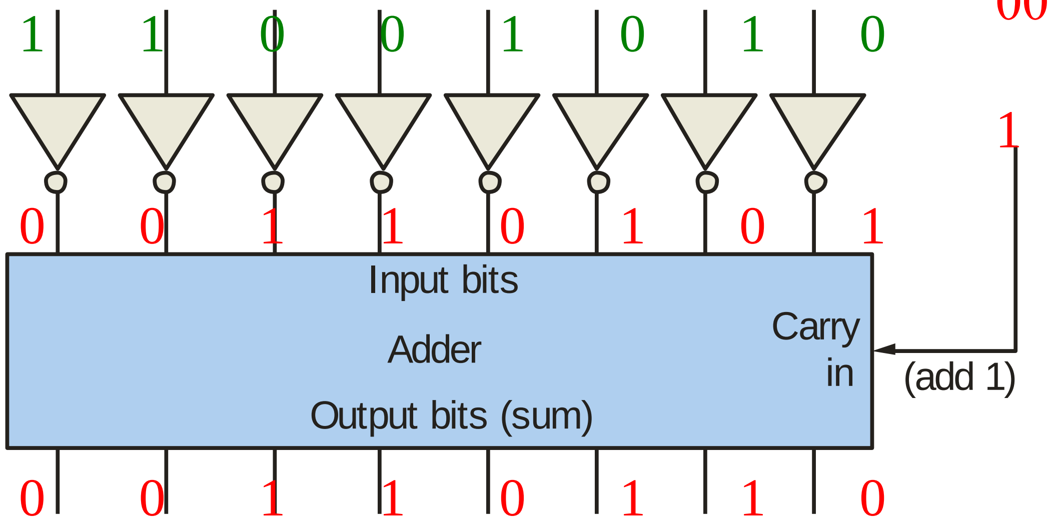
2’s Complement

The 2’s complement of a binary number is found by adding 1 to the LSB of the 1’s complement.

Recall that the 1’s complement of 11001010 is

00110101 (1’s complement)
+1
00110110 (2’s complement)

To form the 2’s complement, add 1:



Signed Binary Numbers

There are several ways to represent signed binary numbers. In all cases, the MSB in a signed number is the sign bit, that tells you if the number is positive or negative.

Computers use a modified 2's complement for signed numbers. Positive numbers are stored in *true* form (with a 0 for the sign bit) and negative numbers are stored in *complement* form (with a 1 for the sign bit).

For example, the positive number 58 is written using 8-bits as 00111010 (true form).

Sign bit

Magnitude bits

Negative numbers are written as the 2's complement of the corresponding positive number.

The negative number –58 is written as:

–58 = 11000110 (complement form)

Sign bit Magnitude bits

An easy way to read a signed number that uses this notation is to assign the sign bit a column weight of –128 (for an 8-bit number). Then add the column weights for the 1's.

Example

Assuming that the sign bit = –128, show that 11000110 = –58 as a 2's complement signed number:

Solution

Column weights:	–128	64	32	16	8	4	2	1.	
	1	1	0	0	0	1	1	0	
	–128	+64				+4	+2		= –58

Floating Point Binary Numbers

Floating point notation is capable of representing very large or small numbers by using a form of scientific notation. A 32-bit single precision number is illustrated below.

The sign bit represents the sign of the number

The Mantissa represents the magnitude of the number and is between 0 and 1.

The Exponent represents the number of paces that the (decimal or binary) point must be moved,.

S	E (8 bits)	F (23 bits)
Sign bit	Biased exponent (+127)	Magnitude with MSB dropped

The binary point is understood to be left of the 23 bit mantissa.

Effectively there are 24 bits in the mantissa as the MSB is always a 1. This 1 is understood to be there but does not occupy an actual bit position.

The biased exponent is obtained by adding 127 to the actual exponent. The bias allows you to express very small or big number without requiring a sign bit for the exponent. Actual exponent values are from -126 to +128

Example Express 1011010010001 in floating point format

Solution

Move binary point 12 places to the left and multiply by appropriate power of 2

$1011010010001 = 1.011010010001 \times 2^{12}$

S=0 since positive

Exponent =12 => biased exponent E =12+127 = 139 = 10001011

Mantissa = .011010010001 (note we ignore the 1) and pad to 23 bits with zeros

S	E	F
0	10001011	011010010001 00000000000

Single Precision Floating Point Binary Numbers

Example

Express the speed of light, *c*, in single precision floating point notation. ($c = 0.2998 \times 10^9$)

Solution

Convert c to binary

In binary, c =10001 1101 1110 1001 0101 1100 0000

In scientific notation, $c = 1.0001\ 1101\ 1110\ 1001\ 0101\ 1100\ 0000 \times 2^{28}$.

S = 0 because the number is positive. E = 28 + 127 = 155_{10} = $1001\ 1011_2$.

F is the next 23 bits after the first 1 is dropped.

In floating point notation, c =

0	10011011	0001 1101 1110 1001 0101 110

Arithmetic Operations with Signed Numbers

Using the signed number notation with negative numbers in 2's complement form simplifies addition and subtraction of signed numbers. Rules for **addition**: Add the two signed numbers. Discard any final carries. The result is in signed form.

Examples:

$$\begin{array}{r l}
00011110 & = +30 \\
00001111 & = +15 \\
\hline
00101101 & = +45
\end{array}
\qquad
\begin{array}{r l}
00001110 & = +14 \\
11101111 & = -17 \\
\hline
11111101 & = -3
\end{array}
\qquad
\begin{array}{r l}
11111111 & = -1 \\
11111000 & = -8 \\
\hline
\cancel{1}\,11110111 & = -9
\end{array}$$

Discard carry

Arithmetic Operations with Signed Numbers

Note that if the number of bits required for the answer is exceeded, overflow will occur. This occurs only if both numbers have the same sign. The overflow will be indicated by an incorrect sign bit.

Two examples are:

$$\begin{array}{l} 01000000 = +64 \\ 01000001 = +65 \\ \hline 10000001 = -1 \end{array}$$

$$\begin{array}{l} 10000001 = -127 \\ 10000001 = -127 \\ \hline \cancel{1}00000010 = \cancel{+2} \end{array}$$

Discard carry

Wrong! The answer is incorrect and the sign bit has changed.

Arithmetic Operations with Signed Numbers

Rules for **subtraction**: 2’s complement the subtrahend and add the numbers. Discard any final carries. The result is in signed form. Repeat the examples done previously, but subtract:

00011110 (+30)	00001110 (+14)	11111111 (–1)
– 00001111 –(+15)	– 11101111 –(–17)	– 11111000 –(–8)

2’s complement subtrahend and add:

00011110 = +30	00001110 = +14	11111111 = –1
11110001 = –15	00010001 = +17	00001000 = +8
~~1~~00001111 = +15	00011111 = +31	~~1~~00000111 = +7
Discard carry		Discard carry

Hexadecimal Numbers

Hexadecimal uses sixteen characters to represent numbers: the numbers 0 through 9 and the alphabetic characters A through F.

Large binary number can easily be converted to hexadecimal by grouping bits 4 at a time and writing the equivalent hexadecimal character.

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Example Express $1001\ 0110\ 0000\ 1110_2$ in hexadecimal:

Solution Group the binary number by 4-bits starting from the right. Thus, 960E

Hexadecimal Numbers

Hexadecimal is a weighted number system. The column weights are powers of 16, which increase from right to left.

Column weights $\begin{cases} 16^3 & 16^2 & 16^1 & 16^0. \\ 4096 & 256 & 16 & 1\ . \end{cases}$

Example Express $1A2F_{16}$ in decimal.

Solution Start by writing the column weights:

4096 256 16 1

1 A 2 F_{16}

$1(4096) + 10(256) + 2(16) + 15(1) = 6703_{10}$

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

BCD

Binary coded decimal (BCD) is a weighted code that is commonly used in digital systems when it is *necessary* to show decimal numbers such as in clock displays.

The table illustrates the difference between straight binary and BCD. BCD represents each decimal digit with a 4-bit code. Notice that the codes 1010 through 1111 are not used in BCD.

Decimal	Binary	BCD
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	00010000
11	1011	00010001
12	1100	00010010
13	1101	00010011
14	1110	00010100
15	1111	00010101

BCD

You can think of BCD in terms of column weights in groups of four bits. For an 8-bit BCD number, the column weights are: 80 40 20 10 8 4 2 1.

Question: What are the column weights for the BCD number 1000 0011 0101 1001?

Answer:

8000 4000 2000 1000 800 400 200 100 80 40 20 10 8 4 2 1

Note that you could add the column weights where there is a 1 to obtain the decimal number. For this case:

$8000 + 200 + 100 + 40 + 10 + 8 + 1 = 8359_{10}$

ASCII

ASCII is a code for alphanumeric characters and control characters. In its original form, ASCII encoded 128 characters and symbols using 7-bits. The first 32 characters are control characters, that are based on obsolete teletype requirements, so these characters are generally assigned to other functions in modern usage. In 1981, IBM introduced extended ASCII, which is an 8-bit code and increased the character set to 256. Other extended sets (such as Unicode) have been introduced to handle characters in languages other than English.

Parity Method

The parity method is a method of error detection for simple transmission errors involving one bit (or an odd number of bits). A parity bit is an “extra” bit attached to a group of bits to force the number of 1’s to be either even (even parity) or odd (odd parity).

Example The ASCII character for “a” is 1100001 and for “A” is 1000001. What is the correct bit to append to make both of these have odd parity?

Solution The ASCII “a” has an odd number of bits that are equal to 1; therefore the parity bit is 0. The ASCII “A” has an even number of bits that are equal to 1; therefore the parity bit is 1.

Problems/ solutions

Convert the following binary numbers to decimal

(a) $11 = 1 \times 2^1 + 1 \times 2^0 = 2 + 1 = 3$
(b) $100 = 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 4$
(c) $111 = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 4 + 2 + 1 = 7$
(d) $1000 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 8$
(e) $1001 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 1 = 9$
(f) $1100 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 8 + 4 = 12$
(g) $1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 2 + 1 = 11$
(h) $1111 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 4 + 2 + 1 = 15$

Convert the following binary numbers to decimal

(a) $110011.11 = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$
$= 32 + 16 + 2 + 1 + 0.5 + 0.25 = 51.75$
(b) $101010.01 = 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^{-2} = 32 + 8 + 2 + 0.25$
$= 42.25$
(c) $1000001.111 = 1 \times 2^6 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$
$= 64 + 1 + 0.5 + 0.25 + 0.125 = 65.875$

Problems/ solutions

- Convert the following to binary using the sum of weights method

(a) $10 = 8 + 2 = 2^3 + 2^1 = 1010$
(b) $17 = 16 + 1 = 2^4 + 2^0 = 10001$
(c) $24 = 16 + 8 = 2^4 + 2^3 = 11000$
(d) $48 = 32 + 16 = 2^5 + 2^4 = 110000$
(e) $61 = 32 + 16 + 8 + 4 + 1 = 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = 111101$
(f) $93 = 64 + 16 + 8 + 4 + 1 = 2^6 + 2^4 + 2^3 + 2^2 + 2^0 = 1011101$
(g) $125 = 64 + 32 + 16 + 8 + 4 + 1 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = 1111101$
(h) $186 = 128 + 32 + 16 + 8 + 2 = 2^7 + 2^5 + 2^4 + 2^3 + 2^1 = 10111010$

- Convert the following to binary by repeated division by 2

15, 21, 28

$$\frac{15}{2} = 7,\ R = 1\text{(LSB)}$$
$$\frac{7}{2} = 3,\ R = 1$$
$$\frac{3}{2} = 1,\ R = 1$$
$$\frac{1}{2} = 0,\ R = 1\ \text{(MSB)}$$

(b)
$$\frac{21}{2} = 10,\ R = 1\ \text{(LSB)}$$
$$\frac{10}{2} = 5,\quad R = 0$$
$$\frac{5}{2} = 2,\quad R = 1$$
$$\frac{2}{2} = 1,\quad R = 0$$
$$\frac{1}{2} = 0,\quad R = 1\ \text{(MSB)}$$

(c)
$$\frac{28}{2} = 14,\quad R = 0\ \text{(LSB)}$$
$$\frac{14}{2} = 7,\quad R = 0$$
$$\frac{7}{2} = 3,\quad R = 1$$
$$\frac{3}{2} = 1,\quad R = 1$$
$$\frac{1}{2} = 0,\quad R = 1\ \text{(MSB)}$$

Problems/ solutions

Add the following binary numbers

(a) $\begin{array}{r} 11 \\ +\,01 \\ \hline 100 \end{array}$ (b) $\begin{array}{r} 10 \\ +\,10 \\ \hline 100 \end{array}$ (c) $\begin{array}{r} 101 \\ +\,011 \\ \hline 1000 \end{array}$

(d) $\begin{array}{r} 111 \\ +\,110 \\ \hline 1101 \end{array}$ (e) $\begin{array}{r} 1001 \\ +\,0101 \\ \hline 1110 \end{array}$ (f) $\begin{array}{r} 1101 \\ +\,1011 \\ \hline 11000 \end{array}$

Subtract the following binary numbers

(a) $\begin{array}{r} 11 \\ -\,01 \\ \hline 10 \end{array}$ (b) $\begin{array}{r} 101 \\ -\,100 \\ \hline 001 \end{array}$ (c) $\begin{array}{r} 110 \\ -\,101 \\ \hline 001 \end{array}$

(d) $\begin{array}{r} 1110 \\ -\,0011 \\ \hline 1011 \end{array}$ (e) $\begin{array}{r} 1100 \\ -\,1001 \\ \hline 0011 \end{array}$ (f) $\begin{array}{r} 11010 \\ -\,10111 \\ \hline 00011 \end{array}$

Problems/ solutions

Determine the 2's complement of the following
10, 111, 1001, 1101, 11100, 10011, 10110000, 00111101

(a)	01 + 1 = 10	(b)	000 + 1 = 001
(c)	0110 + 1 = 0111	(d)	0010 + 1 = 0011
(e)	00011 + 1 = 00100	(f)	01100 + 1 = 01101
(g)	01001111 + 1 = 01010000	(h)	11000010 + 1 = 11000011

Convert each pair of decimal numbers to binary and add using 2's complement form

(a)
$$\begin{aligned} 33 &= 00100001 \\ 15 &= 00001111 \end{aligned} \qquad \begin{array}{r} 00100001 \\ +\ 00001111 \\ \hline 00110000 \end{array}$$

(b)
$$\begin{aligned} 56 &= 00111000 \\ 27 &= 00011011 \\ -27 &= 11100101 \end{aligned} \qquad \begin{array}{r} 00111000 \\ +\ 11100101 \\ \hline 00011101 \end{array}$$

(c)
$$\begin{aligned} 46 &= 00101110 \\ -46 &= 11010010 \\ 25 &= 00011001 \end{aligned} \qquad \begin{array}{r} 11010010 \\ +\ 00011001 \\ \hline 11101011 \end{array}$$

(d)
$$\begin{aligned} 110_{10} &= 01101110 \\ -110_{10} &= 10010010 \\ 84 &= 01010100 \\ -84 &= 10101100 \end{aligned} \qquad \begin{array}{r} 10010010 \\ +\ 10101100 \\ \hline 100111110 \end{array}$$

Problems/ solutions

Convert these hex numbers to binary

(a) 38_{16} = 0011 1000
(b) 59_{16} = 0101 1001
(c) $A14_{16}$ = 1010 0001 0100
(d) $5C8_{16}$ = 0101 1100 1000
(e) 4100_{16} = 0100 0001 0000 0000
(f) $FB17_{16}$ = 1111 1011 0001 0111
(g) $8A9D_{16}$ = 1000 1010 1001 1101

Convert to BCD

(a) 10 = 0001 0000
(b) 13 = 0001 0011
(c) 18 = 0001 1000
(d) 21 = 0010 0001
(e) 25 = 0010 0101
(f) 36 = 0011 0110
(g) 44 = 0100 0100
(h) 57 = 0101 0111
(i) 69 = 0110 1001
(j) 98 = 1001 1000
(k) 125 = 0001 0010 0101
(l) 156 = 0001 0101 0110

Problems/ solutions

Convert these BCD numbers to Decimal

(a) 1000 0000 = 80
(b) 0010 0011 0111 = 237
(c) 0011 0100 0110 = 346
(d) 0100 0010 0001 = 421
(e) 0111 0101 0100 = 754
(f) 1000 0000 0000 = 800
(g) 1001 0111 1000 = 978
(h) 0001 0110 1000 0011 = 1683
(i) 1001 0000 0001 1000 = 9018
(j) 0110 0110 0110 0111 = 6667

Problems/ solutions

Attach an even parity bit to the following bytes

(a) 1 10100100 (b) 0 00001001 (c) 1 11111110