DIGITAL ARITHMETIC
Miloš D. Ercegovac and Tomás Lang
Morgan Kaufmann Publishers, an imprint of Elsevier, ©2004

Chapter 1: Solutions to Exercises

Exercise 1.1

(a)
1. 9 bits since $2^8 \leq 297 \leq 2^9$
2. 3 radix-8 digits since $8^2 \leq 297 \leq 8^3$
3. 3 radix-17 digits since $17^2 \leq 297 \leq 17^3$
4. The weights are 120, 24, 6, 2, and 1. To represent 297, 5 mixed-radix digits are needed: $2 \times 120 + 2 \times 24 + 1 \times 6 + 1 \times 2 + 1 \times 1 = 297$

(b)
1. $x_{max} = 2^9 - 1 = 511$
2. $x_{max} = 8^3 - 1 = 511$
3. $x_{max} = 17^3 - 1 = 4912$
4. $x_{max} = 5 \times 120 + 4 \times 24 + 3 \times 6 + 2 \times 2 + 1 \times 1 = 719$

(c)
1. Binary representation uses 9 bits; $E = 1$
2. Radix-8 digits represented in binary with 3 bits per digit. Digit-vector: $3 \times 3 = 9$ bits; $E = 9/(3 \times 3) = 1$
3. Radix-17 digits represented in binary with 5 bits. Digit-vector: $3 \times 5 = 15$ bits; $E = 9/(3 \times 5) = 0.6$
4. The digit sets for the mixed-radix representation and their lengths in binary representation of digits are

d_0	1,0	1
d_1	2,1,0	2
d_2	3,2,1,0	2
d_3	4,3,2,1,0	3
d_4	5,4,3,2,1,0	3

Digit-vector: 3+3+2+2+1 = 11; $E = 9/11 = 0.82$

Exercise 1.2

X_{RNS} - digit-vector in RNS representation;
$X_{RNS-bin}$ - bit-vector of X_{RNS};

x	X_{RNS}	$X_{RNS-bin}$
0	(0 0 0 0)	(000 000 00 0)
13	(6 3 1 1)	(110 011 01 1)
15	(1 0 0 1)	(001 000 00 1)
19	(5 4 1 1)	(101 100 01 1)
22	(1 2 1 0)	(001 010 01 0)
127	(1 2 1 1)	(001 010 01 1)

To compute the efficiency need to determine the number of bits for the binary representation. This number depends on the range of integers represented; we consider two situations:

i) The largest integer is 127. In such a case, the number of bits is 7 and the efficiency is

$$E = n_{r2}/n_{RNS-bin} = 7/9$$

ii) The largest integer is the maximum allowed by the moduli of the RNS representation. This value is 7x5x3x2-1= 209. Consequently, 8 bits are needed for the radix-2 representation, resulting in

$$E = 8/9$$

Exercise 1.3

If the moduli are not relatively prime, different values may have the same representation. For example, if P = (4,2), $x = 3$ and $x = 7$ have the same RNS digit-vector (3,1).

Exercise 1.4

1. $1 \leq x \leq 2^{8+8} - 1$, $E = 1$
2. $1 \leq x \leq 10^4 - 1$, $E = (10^4 - 1)/(2^{16} - 1) = 0.152$
3. $1 \leq x \leq 16^4 - 1 = 2^{16} - 1$, $E = 1$

Exercise 1.5

(a) Representation values

r	x_R
2	43
8	$8^5 + 8^3 + 8 + 1 = 33289$
10	$10^5 + 10^3 + 10 + 1 = 101,011$
16	$16^5 + 16^3 + 16 + 1 = 1,052,689$

(b) Largest values for $n = 6$

r	x_{Rmax}
2	63
10	$10^6 - 1$
16	$16^6 - 1$

Exercise 1.6

x	$C = 16$	$C = 15$	$C = 19$	$C = 127$
6	0110	0110	00110	0000110
5	0101	0101	00101	0000101
4	0100	0100	00100	0000100
3	0011	0011	00011	0000011
2	0010	0010	00010	0000010
1	0001	0001	00001	0000001
0	0000	0000	00000	0000000
-0	-	1111	10011	1111111
-1	1111	1110	10010	1111110
-2	1110	1101	10001	1111101
-3	1101	1100	10000	1111100
-4	1100	1011	01111	1111011
-5	1011	1010	01110	1111010
-6	1010	1001	01101	1111001

Exercise 1.7

(a) For $r = 2$, $x_R = 11_{10}$. For $r = 7$, $x_R = 351_{10}$. For $r = 16$, $x_R = 4113_{10}$.

(b) For $r = 2$, $x_R = 11$; for 2's complement, $C = 16$; since $x_R > C/2$ we have $x < 0$ and $x = 11 - 16 = -5$.

For $r = 4$, $x_R = 69$; for 1s' complement, $C = 4^4 - 1 = 255$; since $x_R < C/2$, we have $x > 0$ and $x = 69$.

For $r = 8$, $x_R = 521$; for 1s' complement, $C = 8^4 - 1 = 4095$, since $x_R < C/2$, we have $x > 0$ and $x = 521$.

Exercise 1.8

	Value x	Value x_R	Digit vector X
(a)	-39_{10}	4057_{10}	333121_4
(b)	-41_{10}	215_{10}	11010111
(c)	-3_{10}	29_{10}	11101

Exercise 1.9

Number system	Radix r	No. of Digits n	Value x	Value x_R	Digit-vector X
SM	10	4	-837	-837	1837
2's compl.	2	6	-10	54	110110
RC	3	4	-37	44	1122_3
RC	8	3	-149	363	551_8
1s' compl.	2	8	-83	172	10101100
2's compl.	2	7	-19/64	1+45/64	1.101101
DC	8	4	-681	3415	6527_8
1s' compl.	2	7	-19/64	1+44/64	1.101100

Exercise 1.10

NRS	x_{max}	X_{max}	x_{min}	X_{min}
SM	3+15/16	011.1111	-(3+15/16)	111.1111
2's	3+15/16	011.1111	-4	100.0000
1s'	3+15/16	011.1111	-(3+15/16)	100.0000

Exercise 1.11

NRS	integer	fraction
SM	-5	-5/16
2's	-11	-11/16
1s'	-10	-10/16

Exercise 1.12

(a) In the integer case, 2's complement, $x = -5$. Extending to $n = 6$ produces $X_{int-2} = (1, 1, 1, 0, 1, 1)$.

In the 1s' complement system, $x = -4$, and the 6-bit vector is $X_{int-1} = (1, 1, 1, 0, 1, 1)$.

Note that in the case of integers, the extended bit-vectors are the same for 2's complement and for 1s' complement.

(b) We suppose that ”Do not change the position of the radix point” means that the extended value should also be a fraction (having only the ”sign bit” as integer bit).

In the two’s complement fraction case $x = -5/8$. Extending to $n = 6$ produces $X_{frac-2} = (1, 0, 1, 1, 0, 0)$.

In the 1s’ complement fraction case $x = -4/8$ and the extended bit-vector is $X_{frac-1} = (1, 0, 1, 1, 1, 1)$.

Note that in the fraction case the extended bit-vectors are different.

Exercise 1.13
Sign-and-magnitude

- $x + y$.

  Since $x < 0$, we complement x (2’s complement) and add

```
101110
001001
     1
------
111000
```

  The result is negative (sgn=1). We complement to obtain magnitude 00111+1=01000.

- $y - x$.

  Change sign of x and add. Both operands of addition are positive. Sign of result sgn=0.

```
001001
010001
-----
011010
```

- $x - y$.

  Change sign of y and add. Both operands of addition are negative. Consequently, add magnitudes and sign of result is sgn=1.

```
010001
001001
------
011010
```

- $-x - y$. This is $-(x + y)$. So, perform $(x + y)$ and change sign. Result is sgn=0 and magnitude 01000.
- $|x - y|$. Perform $x - y$ and make sgn=0. The magnitude is 11010.

2's complement and 1s' complement
Consider the following table:

Operation	2's Complement	1s' Complement
x	101111	101110
y	001001	001001
	111000	110111
c_{in}/e-a-c	0	0
$x+y$	111000	110111
y	001001	001001
$\bar{x}$	010000	010001
c_{in}/e-a-c	1	0
$y-x$	011010	011010
x	101111	101110
$\bar{y}$	110110	110110
c_{in}/e-a-c	1	1
$x-y$	100110	100101
$\bar{x}$	010000	010001
$\bar{y}$	110110	110110
c_{in}/e-a-c	1	1
	000111	001000
	1	-
$-x-y$	001000	001000
x	101111	101110
$\bar{y}$	110110	110110
c_{in}/e-a-c	1	1
$x-y$	100110	100101
$\overline{x-y}$	011001	011010
c_{in}/e-a-c	1	-
$\lvert x-y\rvert$	011010	011010

Exercise 1.14

The effective operation to compute $z = |x|-|y|$ in the 2's complement system as a function of the signs of the operands is shown in the following table:

x	y	$\lvert x\rvert - \lvert y\rvert$
+	+	$x-y$
+	-	$x+y$
-	+	$-(x+y)$
-	-	$-x+y$

The algorithm is

case of (sign(x), sign(y)):

(0,0): $z = ADD(x, \bar{y}, 1)$;

(0,1): $z = ADD(x, y, 0)$;

(1,0): $z = ADD(\underline{0}, \overline{(ADD(x, y, 0))}, 1)$;

(1,1): $z = ADD(\bar{x}, y, 1)$;

Exercise 1.15

As discussed in this chapter, the change of sign operation in the 2's complement system is performed as

$$z_R = (2^n - 1 - x_R) + 1$$

which corresponds to inverting each bit and adding 1. Let

$$X_b = (X_k, X_{k-1}, \ldots, X_0) = (1, 0, \ldots, 0)$$

and

$$X_a = (X_{n-1}, \ldots, X_{k+1})$$

.

1. After bit-inverting X_b and X_a we get

$$\begin{aligned} \overline{X_b} &= (0, 1, \ldots, 1) \\ \overline{X_a} &= (X'_{n-1}, \ldots, X'_{k+1}) \end{aligned}$$

2. After adding 1, $\overline{X_b}$ is reverted to X_b, while $\overline{X_a}$ remains unaffected.

Since the algorithm produces X_b and $\overline{X_a}$, it performs the change of sign operation.

Exercise 1.16

(a) We show two proofs: in the first we consider all possible cases and in the second we manipulate the expressions.

First proof:

x_{n-1}	y_{n-1}	s_{n-1}	c_{n-1}	c_n	overflow?
0	0	0	0	0	n
0	0	1	1	0	y
0	1	0	1	1	n
0	1	1	0	0	n
1	1	0	0	1	y
1	1	1	1	1	n

Second proof:

The overflow in addition may only happen if the operands are of the same sign, i.e., $x_{n-1} \oplus y_{n-1} = 0$ and, consequently, in this situation

$$s_{n-1} = x_{n-1} \oplus y_{n-1} \oplus c_{n-1} = c_{n-1}$$

On the other hand,

$$\begin{aligned} c_n \oplus c_{n-1} &= (x_{n-1}y_{n-1} + x_{n-1}c_{n-1} + y_{n-1}c_{n-1}) \oplus c_{n-1} \\ &= x_{n-1}y_{n-1}c'_{n-1} + x'_{n-1}y'_{n-1}c_{n-1} \\ &= x_{n-1}y_{n-1}s'_{n-1} + x'_{n-1}y'_{n-1}s_{n-1} \end{aligned}$$

which is the expression for overflow.

(b) The overflow detection using c_n and c_{n-1} does not work in the 1s' complement system since $(-0) + (-2^{n-1} + 1)$ produces $c_n = 1$ and $c_{n-1} = 0$ indicating an overflow which does not exist. For example,

$x = -3 = 100$, $y = -0 = 111$

x	100	
y	111	
	1011	$c_n = 1$, $c_{n-1} = 0$, $c_n \oplus c_{n-1} = 1$ Overflow
s	100	No overflow

Exercise 1.17

(a) 1. Signed integers

NRS	Range
SM	$[-(2^{15} - 1), 2^{15} - 1]$
2's	$[-2^{15}, 2^{15} - 1]$
1s'	$[-(2^{15} - 1), 2^{15} - 1]$

2. Unsigned integers: $[0, 2^{16} - 1]$

(b) 1. With 2's complement adder and flags:

Case	Adder	Z	SGN	$C0$	OVF
unsigned add	yes	yes	no	yes	no
unsigned sub	yes	yes	no	yes	no

2. With 1s' complement adder and flags:

Case	Adder	Z	SGN	$C0$	OVF
unsigned add	no	no	no	yes	no
unsigned sub	no	no	no	yes	no

(c) We consider here only the case for 2's complement representation for signed integers. The case for the other two representations can be determined in a similar manner.

For the comparison of A and B we perform $A - B$ and set the flags. The three conditions are determined as follows:

- For signed integers in 2's complement representation:
  Equal $Z = 1$
  SMALLER ($OVF = 0$ AND $NEG = 1$) OR ($OVF = 1$ AND $NEG = 0$) (no overflow and negative or overflow and not negative)
  GREATER ($OVF = 0$ AND $NEG = 0$) OR ($OVF = 1$ AND $NEG = 1$) AND $Z = 0$ (not smaller and not zero)
- For unsigned:
  Equal $Z = 1$
  For the other cases we need to consider the effect of converting the second operand to 2's complement and adding. So the operation $A - B$ is performed as

$$D = A + (2^{16} - B) = 2^{16} + (A - B)$$

  Consequently, the flag CO is set when $A - B \geq 0$. So,
  GREATER ($CO = 1$ AND $Z = 0$)
  SMALLER $CO = 0$

From these expression we see that only the branch on equal can be the same for both signed and unsigned integers.

Exercise 1.18

(a) Integers a and b represented by A and B:

C	a	b
10^4	-2638	3216
$10^4 - 1$	-2637	3216

(b) Extended to six digits:

$$A = (9, 9, 7, 3, 6, 2), \quad B = (0, 0, 3, 2, 1, 6)$$

(c) $d = 10a$, $e = a/10$ (integer), with seven digits

$$D = (9, 9, 7, 3, 6, 2, 0), \quad E = (9, 9, 9, 9, 7, 3, 6)$$

Exercise 1.19

For $x \geq 0$ we have that $z_R = x_R$. Consequently, since $X_{n-1} = 0$, the algorithm is correct. For $x < 0$, $z_R = C_z - |x|$ and $x_R = C_x - |x|$ where C_z and C_x are the corresponding complementation constants. Consequently,

$$z_R = C_z - C_x + x_R \tag{1}$$

Since for both the 2's and 1s' complement systems

$$C_z - C_x = 2^m - 2^n \tag{2}$$

we obtain

$$z_R = 2^m - 2^n + x_R \tag{3}$$

But $2^m - 2^n$ is represented by the vector

$$(1, 1, ..., 1, 0, 0, ..., 0)$$

Consequently,

$$Z = (1, 1, ..., 1, X_{n-1}, \ldots, X_0) \tag{4}$$

which corresponds to the given algorithm.

Exercise 1.20

Left shift. By definition $z = 2x$. i) If $x \geq 0$ the representation is the same as in the sign-and-magnitude system and, therefore, the same algorithm holds.

ii) If $x \leq 0$ then $x = x_R - C$ and $z = z_R - C$. Therefore, $z_R - C = 2(x_R - C)$ and $z_R = 2x_R - C$. Moreover, since $x \leq 0$ we have $X_{n-1} = 1$ and

$$2x_R = 2 \cdot 1 \cdot 2^{n-1} + 2X_{n-2}2^{n-2} + \ldots + 2X_0$$

In the 2's complement system, since $C = 2^n$ we obtain

$$\begin{aligned} z_R &= 2x_R - 2^n \\ &= 2 \cdot 1 \cdot 2^{n-1} + 2X_{n-2}2^{n-2} + \ldots + 2X_0 - 2^n \\ &= X_{n-2}2^{n-1} + X_{n-3}2^{n-2} + \ldots + X_0 2 + 0 \cdot 2^0 \end{aligned}$$

From the last expression we infer the corresponding left-shift algorithm for the 2's complement system. Note that overflow occurs when $X_{n-2} \neq X_{n-1}$.

In the 1s' complement system $C = 2^n - 1$ so that

$$\begin{aligned} z_R &= 2x_R - (2^n - 1) \\ &= 2x_R - 2^n + 1 \end{aligned}$$

Using the expression for $2x_R$ developed in the previous proof,

$$z_R = X_{n-2}2^{n-1} + X_{n-3}2^{n-2} + \ldots + X_0 2 + 1 \tag{5}$$

This corresponds to the indicated algorithm.

Right shift. By definition $z = 2^{-1}x - \epsilon$. If $x \geq 0$, the same algorithm as in the sign-and-magnitude case holds.

If $x \leq 0$ then $z_R - C = 2^{-1}(x_R - C) - \epsilon$ and $z_R = 2^{-1}(x_R - C) + C - \epsilon$. For the 2's complement system $C = 2^n$, so

$$2^{-1}(x_R - C) = -2^{n-1} + X_{n-1}2^{n-2} + \ldots + X_1 2^0 + X_0 2^{-1} \tag{6}$$

and

$$\begin{aligned} z_R &= 2^n - 2^{n-1} + X_{n-1}2^{n-2} + \ldots + X_1 + X_0 2^{-1} - \epsilon \\ &= 2^{n-1} + X_{n-1}2^{n-2} + \ldots + X_1 + X_0 2^{-1} - \epsilon \end{aligned} \tag{7}$$

Assuming $\epsilon = X_0 2^{-1}$ (this satisfies $|\epsilon| < 1$), we obtain the corresponding algorithm.

In the 1s' complement system $C = 2^n - 1$, so that

$$2^{-1}(x_R - C) = -2^{n-1} + X_{n-1}2^{n-2} + \ldots + X_1 2^0 + (X_0 + 1)2^{-1} \quad (8)$$

and

$$z_R = 2^n - 2^{n-1} + X_{n-1}2^{n-2} + \ldots + X_1 2^0 + (X_0 + 1)2^{-1} - 1 - \epsilon \quad (9)$$

Assuming now $\epsilon = 1 - (X_0 + 1)2^{-1}$ the same algorithm is obtained.

Exercise 1.21

2's complement:

X	00101101	45
$SL(X)$	01011010	90
$SR(X)$	00010110	22
Y	11010110	-42
$SL(Y)$	10101100	-84
$SR(Y)$	11101011	-21

1s' complement:

X	00101101	45
$SL(X)$	01011010	90
$SR(X)$	00010110	22
Y	11010110	-41
$SL(Y)$	10101101	-82
$SR(Y)$	11101011	-20

Exercise 1.22

Overflow happens in the arithmetic shift-left if

$$X_{n-2} \neq X_{n-1}$$

This is because in this case the sign would change by the shift.

Exercise 1.23

Given

$$\begin{array}{ll} A = 1101 & (a = -3) \\ B = 110 & (b = -2) \\ C = 0101 & (c = 5) \\ D = 10101 & (d = -21) \end{array}$$

compute $z = -3 + (-2) + 8 * 5 - 2 * (-21) = -7$.

$$\begin{array}{cc} A & 1111101 \\ B & 1111110 \\ 8C & 0101000 \\ 2D & 1101010 \\ \hline z & 1111001 \end{array}$$

Exercise 1.24

The multiplication is shown in Figure E1.24.

$n = 5 \quad x = 21\ (X = 10101) \quad y = 14\ (Y = 01110)$

$p[0]$	00000	
$2^5 x Y_0$	00000	
	00000	
$p[1]$	00000	0
$2^5 x Y_1$	10101	
	10101	0
$p[2]$	01010	10
$2^5 x Y_2$	10101	
	11111	10
$p[3]$	01111	110
$2^5 x Y_3$	10101	
	100100	110
$p[4]$	010010	0110
$2^5 x Y_4$	00000	
$p[5]$	10010	0110 = 294

Figure E1.24

Exercise 1.25

(a) The multiplication for 2's complement representation is given in Fig. E1.25a.

(b) The multiplication for 1s' complement representation is in Fig. E1.25b. Note that we complement the multiplier and then complement the result.

Exercise 1.26

The execution time of the basic multiplication scheme for n-bit non-negative integers is

$$T_{basic} = (t_{vd} + t_{add} + t_{reg}) \times n$$

The execution time can be reduced by using the multiplier as a radix-4 digit-vector to about $T_{basic}/2$ as follows:

- Precompute $3X = 2X + X$ and store it in a register.
- In each iteration consider two bits of the multiplier as a radix-4 digit $z_j \in \{0, 1, 2, 3\}$. Select $0 \times X$, $1 \times X$, $2 \times X$ (left shifted X produced by wiring - no extra delay), or $3 \times X$ (precomputed using shift and add) depending on the value of z_j using a multiplexer.
- Perform $n/2$ iterations.

Since $n/2$ iterations are performed and one additional cycle is required for the precomputation of $3x$, the reduced execution time is

$n = 6 \quad x = 21\ (X = 010101) \quad y = -17\ (Y = 101111)$

$p[0]$	0 000000	
$2^5 x Y_0$	0 010101	
	0 010101	
$p[1]$	0 001010	1
$2^5 x Y_1$	0 010101	
	0 011111	1
$p[2]$	0 001111	11
$2^5 x Y_2$	0 010101	
	0 100100	11
$p[3]$	0 010010	011
$2^5 x Y_3$	0 010101	
	0 100111	011
$p[4]$	0 010011	1011
$2^5 x Y_4$	0 000000	
	0 010011	1011
$p[5]$	0 001001	11011
$-2^5 x Y_5$	1 101011	
$p[6]$	1 110100	11011 = xy = -357

Figure E1.25a 2's complement multiplication.

$n = 6 \quad x = 21\ (X = 010101) \quad y = -17\ (Y = 101110)$
$-y = 17\ (010001)$

$p[0]$	0 000000	
$2^5 x Y_0$	0 010101	
	0 010101	
$p[1]$	0 001010	1
$2^5 x Y_1$	0 000000	
	0 001010	1
$p[2]$	0 000101	01
$2^5 x Y_2$	0 000000	
	0 000101	01
$p[3]$	0 000010	101
$2^5 x Y_3$	0 000000	
	0 000010	101
$p[4]$	0 000001	0101
$2^5 x Y_4$	0 010101	
	0 010110	0101
$p[5]$	0 001011	00101
complement		
$p[6]$	1 110100	11010 = xy = -357

Figure E1.25b 1s' complement multiplication.

$$T_{reduced} = (t_{MUX} + t_{add} + t_{reg}) \times (n/2 + 1)$$

Exercise 1.27

The recurrence for the left-to-right multiplication of non-negative integers is

$$\begin{aligned} &p[0] = 0 \\ &p[j+1] = rp[j] + xY_{n-1-j} \quad j = 0, 1, \ldots, n-1 \\ &p = p[n] \end{aligned} \tag{10}$$

It can be shown by substitution that

$$p[j+1] = r^{j+1}p[0] + x \sum_{k=n-1-j}^{n-1} Y_k r^{k-(n-1-j)}$$

so that

$$p[n] = r^n p[0] + xy$$

The adder has $2n-1$ digits. The relative position of the operands in the left-to-right recurrence is shown in Figure E1.27

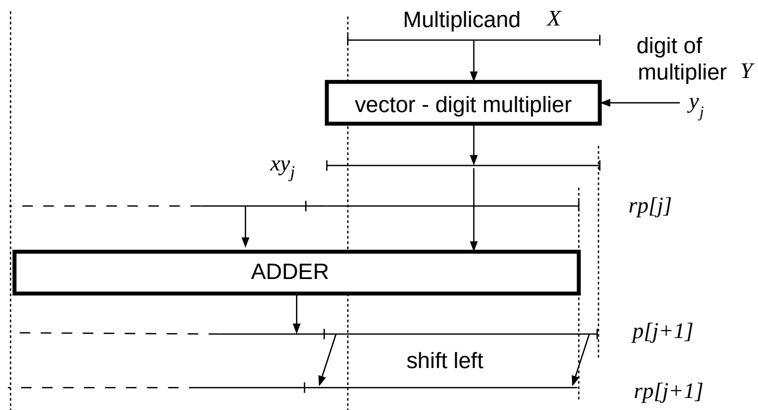

Figure E1.27: Relative position of operands in left-to-right multiplication.

Since the adder is twice as wide as in the right-to-left (basic) multiplication, the execution time is significantly increased.

Exercise 1.28

From Algorithm NRD for integer division of $2n$-bit dividend x and n-bit divisor d we have:

$d^* = d2^n \qquad w[0] = x$

For $j = 0$, $w[1] = 2w[0] - q_{n-1}d^*$
For $j = 1$, $w[2] = 2w[1] - q_{n-2}d^* = 2^2w[0] - (2q_{n-1} + q_{n-2})d^*$
For $j = n-1$, $w[n] = 2^n w[0] - (2^{n-1}q_{n-1} + 2^{n-2}q_{n-2} + \ldots + 2q_1 + q_0)d^*$

The last scaled remainder (corrected if negative) is

$$2^{-n}w[n] = w[0] - (\sum_{j=0}^{n-1} q_j 2^j)d^* 2^{-n} = x - q \cdot d$$

since $w[0] = x$ and $q = \sum_{j=0}^{n-1} q_j 2^j$. Therefore,

$$x = q \cdot d + w$$

Since the quotient-digit selection function guarantees bounded residuals $|w[j]| < d^*$, the algorithm is correct.

Exercise 1.29

Perform non-restoring integer division for the following operands.

Dividend $x = 14_{10} = (00001110)_2$, divisor $d = 3 = (0011)_2$

$$
\begin{array}{rlll}
w[0] = & 0\ 0000 & 1110 & \\
2w[0] = & 0\ 0001 & 1100 & \\
-d^* = & 1\ 1101 & & \\
\hline
w[1] = & 1\ 1110 & 1100 & q_3 = 0 \\
2w[1] = & 1\ 1101 & 1000 & \\
+d^* = & 0\ 0011 & & \\
\hline
w[2] = & 0\ 0000 & 1000 & q_2 = 1 \\
2w[2] = & 0\ 0001 & 0000 & \\
-d^* = & 1\ 1101 & & \\
\hline
w[3] = & 1\ 1110 & 0000 & q_1 = 0 \\
2w[3] = & 1\ 1100 & 0000 & \\
+d^* = & 0\ 0011 & & \\
\hline
w[4] = & 1\ 1111 & 0000 & q_0 = 0 \\
\hline
w[4] = & 0\ 0010 & & (\mathit{corrected})
\end{array}
$$

Quotient $q = (0100)_2 = 4$, remainder $w = (0010)_2 = 2$. Check: $14 = 3 \times 4 + 2$.

Exercise 1.30

We consider the alternative with quotient-digit set $\{-1, +1\}$. If the divisor is signed, the quotient-digit selection depends on the sign of the divisor. To have a bounded residual, the selection function is

$$q_{n-j} = \begin{cases} 1 & \text{if } sign(w[j]) = sign(d) \\ -1 & \text{if } sign(w[j]) \neq sign(d) \end{cases}$$

We also want the quotient to be in 2's complement representation. This is accomplished by making the quotient

$$q = P + N$$

where P is the weighted sum of all digits having value 1 and N is the weighted sum of all digits with value -1. Consequently, the 2's complement representation is obtained by adding P and N (2's complement addition). For this, N (which is negative) should be represented in 2's complement.

It is also possible to do the conversion considering only P as follows. Since all bits of q are either 1 or -1 we get

$$P - N = 2^n - 1$$

and

$$(P + N) + (P - N) = 2P = q + 2^n - 1$$

so that

$$q = -2^n + 2P + 1$$

Morover, since the maximum absolute value of the quotient is $2^{n-1} - 1$ (remember the the $n - th$ bit is the "sign bit"), The two most-significant signed digits of q cannot be of the same sign. Consequently, in P the most-significant two bits are either 10 (positive quotient) or 01 (negative quotient). Therefore, when subtraction 2^n from $2P$, we get a 2's complement representation, as follows:

P=10... then $2P - 2^n = 0...$ (that is, bit $n-1$ is 0 and the result is positive)
P=01... then $2P - 2^n = 1....$(that is, bit $n-1$ is 1 and the result is negative)
This can be implemented during the iterations by

- Replacing -1's with 0's
- Shifting the resulting vector one position to the left
- Inverting the quotient bit in position $n - 1$ and inserting 1 in the least-significant position. If quotient correction is needed, 0 is inserted in its least-significant position.