Internal Report No.: VA7
Author: Allan Giese
Ed.: February 1967.

# BINARY ARITHMETIC

ABSTRACT,

The purpose of this paper is to explain and to verify the algorithms used by digital computers for performing binary arithmetic. The algorithms discussed are, addition, subtraction, multiplication, and division. The paper also includes a brief introduction to the 2's complement notation.

A/S Regnecentralen,
1, Falkoneralle,
Copenhagen, -

Internal Report No.: VA7
Author: Allan Giese
Ed.: February 1967.

# BINARY ARITHMETIC

A/S Regnecentralen,
1, Falkoneralle,
Copenhagen, F.

CONTENTS

## 2. REPRESENTATION OF INTEGER NUMBERS IN THE 2's COMPLEMENT NOTATION.

Before stating the rules for the 2's complement notation, some general remarks are required.

1. N denotes the number of binary digits in the bit string

$$a_0 a_1 \cdots a_k \cdots a_{N-1}.$$

2. a equals the value of the bit string, which is found by interpreting the binary digits as coefficients in a polynomial of 2, i.e.

$$a = a_0 2^{N-1} + a_1 2^{N-2} + \cdots + a_k 2^{N-1-k} + \cdots + a_{N-1} 2^0$$

$$= \sum_{k=0}^{N-1} a_k 2^{N-1-k}.$$

We will often make use of the notation $a = a_0 a_1 \cdots a_k \cdots a_{N-1}$, meaning the summation.

3. A is the signed integer represented by the above bit string.

4. ::= denotes >>represented by<<, hence A::= a.

The bit string is interpreted as a signed integer A, in accordance with the following rules:

Definition

$$A = \begin{cases} a & \text{for } a_0 = 0 \\ -(2^N - a) & \text{for } a_0 = 1. \end{cases} \qquad (2.1)$$

Conversely, the bit string representation of the signed integer A is easily obtained by solving the equations (2.1) with respect to a.

$$a = \begin{cases} A = |A| & \text{for } A \geq 0 \\ 2^N + A = 2^N - |A| & \text{for } A < 0. \end{cases} \qquad (2.2)$$

Examples.

$N = 4, \quad -8 \leq A \leq 7$

$6::= 0110 \qquad -6::= 2^4 - 6 = 1010.$

From the above-mentioned it should be observed that each of the $2^N$ N-bit integer numbers, x, in the interval

$$-2^{N-1} \leq x \leq 2^{N-1} - 1,$$

which forms an Abelian group X with respect to modulo N addition will have an image y in the 2's complement notation.

$$y = f(x) = \begin{cases} |x| & \text{for } x \geq 0 \text{ modulo } 2^N \\ 2^N - |x| & \text{for } x < 0 \text{ modulo } 2^N. \end{cases}$$

The mapping establishes in fact a one-to-one correspondence between the two intervals $-2^{N-1} \leq x \leq 2^{N-1} - 1$ and $0 \leq y \leq 2^N - 1$.

We will now develop three useful rules for finding the bit string, representing -A, when the bit string of A is known.

Theorem 2.1. If $A::= a$ then $-A::= 2^N - a$.

Proof. 1) $A \geq 0$: $a = |A|$, $-A::= 2^N - |A| = 2^N - a$

2) $A < 0$: $a = 2^N - |A|$, $-A::= |A| = 2^N - a$. Q.E.D.

Theorem 2.2. If $A::= a_0 a_1 \cdots a_{N-1}$ then $-A::= \bar{a}_0 \bar{a}_1 \cdots \bar{a}_{N-1} + 1$

Proof. We obtain from Theorem 2.1 that

$$-A::= 2^N - a_0 a_1 \cdots a_{N-1}$$

$$= 2^N - 1 - a_0 a_1 \cdots a_{N-1} + 1$$

$$= 11 \cdots 1 - a_0 a_1 \cdots a_{N-1} + 1$$

$$= \bar{a}_0 \bar{a}_1 \cdots \bar{a}_{N-1} + 1.$$ Q.E.D.

Theorem 2.3. If $A::= a_0 \cdots a_{m-1}\ a_m\ a_{m+1} \cdots a_{N-1}$ then $-A::= \bar{a}_0 \cdots \bar{a}_{m-1}\ a_m\ a_{m+1} \cdots a_{N-1}$, where $a_m$ is the least significant bit having the value 1.

Proof. Theorem 2.2 implies that

$-A::= \bar{a}_0 \cdots \bar{a}_{m-1}\ \bar{a}_m\ \bar{a}_{m+1} \cdots \bar{a}_{N-1} + 1.$

As $a_{N-1} = 0$ then $\bar{a}_{N-1} = 1$ and $\bar{a}_{N-1} + 1 = 0$ plus a carry.

$a_{N-2} = 0$ then $\bar{a}_{N-2} = 1$ and $\bar{a}_{N-2} + 1 = 0$ plus a carry, and so on until we encounter $a_m$. Because,

$a_m = 1$ then $\bar{a}_m = 0$ and $\bar{a}_m + 1 = 1$ and no carry. This implies that the bits $\bar{a}_0\bar{a}_1 \cdots \bar{a}_{m-1}$ are undisturbed. Q.E.D.

The table below shows some characteristic numbers.

| | | |
|---|---|---|
| Largest number | is | 011 ... 1 |
| | stands for | $2^{N-1} - 1$ |
| Smallest number | is | 100 ... 0 |
| | stands for | $-2^{N-1}$ |
| Zero | | 000 ... 0 |

## 3. SHIFTING.

To implement the multiply and the divide instructions, it is necessary to have a shift register at one's disposal. The mathematical behaviour of shift operations is explained in this section.

An arithmetical left shift is performed by shifting the number one bit position to the left and inserting a zero in the least significant bit.

$$A ::= a_0 a_1 a_2 \cdots a_{N-1} \quad \underline{ashl}A ::= a_1 a_2 \cdots a_{N-1} 0$$

Theorem 3.1. The value of A shifted k places to the left equals $2^k A$, provided that no overflow has occured.

Proof.

$$A \geq 0\text{: } \underline{kashl}\ A = 2^k a = 2^k |A|$$

$$A < 0\text{: } \underline{kashl}\ A = 2^k a = 2^k (2^N - |A|) = 2^{k+N} - 2^k |A|$$

$$= 2^N - 2^k |A| \quad \text{modulus } 2^N. \qquad \text{Q.E.D.}$$

An arithmetical right shift is performed by shifting the number one bit position to the right and the left most bit remains unchanged. The right most bit is either discarded or a rounding-off procedure may take place.

$$A ::= a_0 a_1 \cdots a_{N-2} a_{N-1} \quad \underline{ashr}A ::= a_0 a_0 a_1 \cdots a_{N-2}\ (a_{N-1})$$

Theorem 3.2. The value of A shifted k places to the right equals $2^{-k} A$, provided that no truncation has taken place.

Proof.

$$A \geq 0\text{: } \underline{kashr}\ A = 2^{-k} a = 2^{-k} |A|$$

$$A < 0\text{: } \underline{kashr}\ A = 2^{N-1} + 2^{N-2} + \cdots + 2^{N-k} + 2^{-k}(2^N - |A|)$$

$$= 2^N - 1 - (2^{N-k} - 1) + 2^{N-k} - 2^{-k} |A|$$

$$= 2^N - 2^{-k} |A| \qquad \text{Q.E.D.}$$

## 4. ROUNDING-OFF.

Either the shifting or division process may produce more digits in the result than are desired. The positive number

$$01011.x_N x_{N+1}$$

would be recorded as the integer 01100 or 01011 when $x_N$ equals 1 or 0, respectively. The most straightforward way to obtain this rounded number is to add a 1 in the highest order which is to be dropped, and a carry will propagate if $x_N = 1$. Unfortunately this method does not work for negative numbers. For example 10010.1 (-13.5) would be rounded to 10011 (-13.0) instead of 10010 (-14.0). From these two examples the following rounding procedure is deducted.

Theorem 4.1. If $A \geq 0$: add 1 to $a_{N-1}$ if $a_N a_{N+1} \geq 10$

If $A < 0$: add 1 to $a_{N-1}$ if $a_N a_{N+1} > 10$

Example.

| | | | | | |
|---|---|---|---|---|---|
| 01110 | 00 | 14.00 | 10010 | 00 | -14.00 |
| 01101 | 11 | 13.75 | 10010 | 01 | -13.75 |
| 01101 | 10 | 13.50 | 10010 | 10 | -13.50 |
| 01101 | 01 | 13.25 | 10010 | 11 | -13.25 |
| 01101 | 00 | 13.00 | 10011 | 00 | -13.00 |
| $a_{N-1}$ | | | $a_{N-1}$ | | |

The amount of equipment required to produce a rounding procedure as above-mentioned may be deemed excessive for some applications, and therefore the straightforward method is often used.

## 5. ADDITION.

With pencil and paper arithmetic it is customary to add more than two numbers simultaneously at a time. This is never done in computer arithmetic, because the sum of only two numbers may be a (N+1)-bit number, and hence exceeds the word-capacity, - this is named overflow. By using one more bit, the link bit $a_{-1}$, in the adder circuitry a simple overflow detection is possible, and it is then left to the programmer to remedy the situation.

Theorem 5.1. Let A::= a and B::= b then the sum S::= s = a+b.
Overflow occurs if and only if $s_{-1} \neq s_0$. $s_{-1}$ always gives the correct sign.

Proof.

1. $A \geq 0$, $B \geq 0$

$$\begin{array}{ll} 0\ 0\ a_1 a_2 \cdots a_{N-1} & A::= a = A \\ 0\ 0\ b_1 b_2 \cdots b_{N-1} & B::= b = B \\ \hline 0\ s_0 s_1 s_2 \cdots s_{N-1} & \end{array}$$

S::= s = a+b = A+B

$s_0$ equals 1 if and only if $A+B \geq 2^{N-1}$, consequently overflow occurs if $s_{-1} \neq s_0$. The link always gives the correct information about the sign of the sum.

2. $A < 0$, $B < 0$

$$\begin{array}{ll} 1\ 1\ a_1 a_2 \cdots a_{N-1} & A::= a = 2^{N+1} - |A| \\ 1\ 1\ b_1 b_2 \cdots b_{N-1} & B::= b = 2^{N+1} - |B| \\ \hline 1\ s_0 s_1 s_2 \cdots s_{N-1} & \end{array}$$

$$S::= s = a+b = 2^{N+1} - |A| + 2^{N+1} - |B| = 2^{N+2} - (|A|+|B|) = 2^{N+1} - (|A|+|B|)$$

When the sum of the two negative numbers satisfies the not-overflow condition i.e. $-2^{N-1} \leq S < 0$, the following inequality $3 \times 2^{N-1} \leq 2^{N+1} - (|A|+|B|) < 2^{N+1}$ is correct, which implies $s_0 = 1$. The overflow test is therefore equivalent to a test of the two bits $s_{-1}$ and $s_0$. The contents of $s_{-1}$ determines the sign.

3. $A \geq 0$, $B < 0$

$$\begin{array}{rll} & 00a_1a_2 \ldots a_{N-1} & A::= a = |A| \\ & \underline{11b_1b_2 \ldots b_{N-1}} & B::= b = 2^{N+1} - |B| \\ & 11s_1s_2 \ldots s_{N-1} & \\ \text{or} & 00s_1s_2 \ldots s_{N-1} & \end{array}$$

$$S::= s = a+b = |A|+2^{N+1}- |B| = \begin{cases} |A| - |B| & \text{for } 0 \leq |A|-|B| < 2^{N-1} \\ 2^{N+1} + |A|-|B| & \text{for } -2^{N-1} \leq |A|-|B| < 0 \end{cases}$$

Overflow is naturally impossible in this case and $s_{-1} = s_0$. Again the correct sign is equivalent to $s_{-1}$.

4. $A < 0$, $B \geq 0$

This case is similar to 3.

Q.E.D.

6. SUBTRACTION.

As A - B = A + (-B) subtraction is done as follows.

Theorem 6.1. Let A::= a and B::= b then the difference $D::= d = a_0a_1 \ldots a_{N-1} + \bar{b}_0\bar{b}_1 \ldots \bar{b}_{N-1} + 1$. Overflow occurs if and only if $d_{-1} \neq d_0$. $d_{-1}$ always gives the correct sign.

Proof. The verification of this rule is obtained by combining the Theorems 2.2 and 5.1. Q.E.D.

Example.

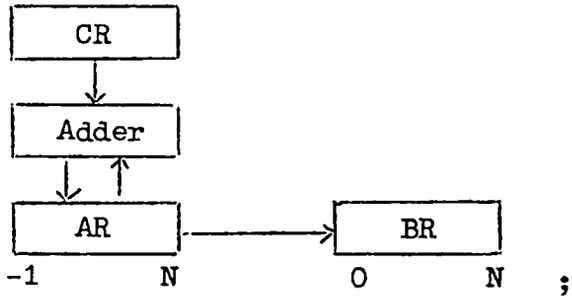
$$\begin{array}{ll} 5::= 00101 & 00101 \\ 7::= 00111 & 11000 \\ & \quad\;\; 1 \\ & \overline{11110} \end{array}$$

## 7. MULTIPLICATION.

Multiplication is carried out by means of additions, subtractions, and shifts alone. The number of operations and their order is determined by the multiplier digits. Overflow is never a problem in integer multiplication, because the maximum obtainable result $(-2^{N-1})\times(-2^{N-1}) = 2^{N-2}$ can be represented within 2N bits.

The algorithm is described in terms of a HARGOL procedure followed by a mathematical verification. Before the reader proceeds, he might find it convenient to consult the examples on page 13.

```
procedure MULTIPLY;
comment A::= a is the multiplicand and B::= b is the multiplier. The concatenated
register AR(0:N-1)conBR(0:N-1) contains, after termination of the procedure, the
result. AR and CR are both supplied with one extra bit to avoid overflow. The re-
gister configuration is as follows,
```




;

```
begin integer k; register AR(-1:N-1), BR(0:N-1), CR(-1:N-1);
      AR:= 0; BR:= b; CR:= a;
      for k:= N-1 step -1 until 1 do
          begin if BR(N-1) = 1 then AR:= AR + CR;
                ashr AR con BR
          end;
      if BR(N-1) = 1 then AR:= AR - CR;
      ashr AR con BR
end MULTIPLY;
```

Proof.

The numerical value produced by the algorithm equals

$$(N\underline{ashr}A)\times b_{N-1} + \ldots + (k\underline{ashr}A)\times b_{k-1} + \ldots + (2\underline{ashr}A)\times b_1 - (\underline{ashr}A)\times b_0.$$

$A \geq 0$: $A\times B := 2^{-N}Ab_{N-1} + \ldots + 2^{-k}Ab_{k-1} + \ldots + 2^{-2}Ab_1 + (2^N - 2^{-1}A)b_0$

$$= A2^{-N}(b_{N-1} + \ldots + 2^{N-k}b_{k-1} + \ldots + 2^{N-2}b_1 - 2^{N-1}b_0) + 2^N b_0$$

$A < 0$: $A\times B := (2^N - 2^{-N}|A|)b_{N-1} + \ldots + (2^N - 2^{-k}|A|)b_{k-1} + \ldots + (2^N - 2^{-2}|A|)b_1 + 2^{-1}|A|b_0$

$$= -|A|2^{-N}(b_{N-1} + \ldots + 2^{N-k}b_{k-1} + \ldots + 2^{N-2}b_1 - 2^{N-1}b_0) + 2^N \sum_{k=N-1}^{1} b_k.$$

1. $A \geq 0$, $B \geq 0$ ($b_0 = 0$).

$A\times B := A2^{-N}b = AB2^{-N}$.

The factor $2^{-N}$ designates the fact that the binary point must be moved N places to the right.

2. $A \geq 0$, $B < 0$ ($b_0 = 1$).

$$A\times B := A2^{-N}(b - 2\times 2^{N-1}b_0) + 2^N b_0 = A2^{-N}(2^N - |B| - 2^N) + 2^N$$

$$= 2^N - A|B|2^{-N}.$$

This is the expected result.

3. $A < 0$, $B \geq 0$ ($b_0 = 0$).

$$A\times B := -|A|2^{-N}b + 2^N \sum_{k=N-1}^{1} b_k$$

$$= 2^N \sum_{k=N-1}^{1} b_k - |A|B2^{-N} = \begin{cases} 2^N - |A|B2^{-N} & \text{for } B \neq 0 \\ |A|B2^{-N} = 0 & \text{for } B = 0. \end{cases}$$

4. $A < 0$, $B < 0$ ($b_0 = 1$).

$$A\times B := -|A|2^{-N}(b - 2\times 2^{N-1}b_0) + 2^N \sum_{k=N-1}^{1} b_k$$

$$= -|A|2^{-N}(2^N - |B| - 2^N) + 2^N \sum_{k=N-1}^{1} b_k = |A||B|2^{-N}.$$

Q.E.D.

Examples, N = 3.

2×3 = 6

```
        0010 × 011
        0000
      + 0010 |011
        0010
ashr    0001 0|01
      + 0010
        0011
ashr    0001 10|0
ashr    0000 110
```

2×(-3) = -6

```
        0010 × 101
        0000
      + 0010 |101
        0010
ashr    0001 0|10
ashr    0000 10|1
      - 0010
        1110
ashr    1111 010
```

2×(-4) = -8

```
        0010 × 100
        0000
ashr    0000 0|10
ashr    0000 00|1
      - 0010
        1110
ashr    1111 000
```

(-4)×(-4) = 16

```
        1100 × 100
        0000
ashr    0000 0|10
ashr    0000 00|1
      - 1100
        0100
ashr    0010 000
```

## 8. DIVISION.

The non-restoring division method has been chosen in preference to the restoring method, because the latter requires extra cycles for restoration, which of course is time consuming. Speed may be gained by using more elaborate methods, but they are unfortunately very expensive to implement. As it is of paramount importance that the remainder and the dividend have the same sign much of the following is devoted to this topic.

The symbols used are: $X_0$ dividend, $X_k$ partial remainder, Q quotient, D divisor, R remainder, and N is the register capacity; thus

$$X_0 = QD + R.$$

### 8.1. Quotient Determination.

By the non-restoring method, one binary quotient bit is determined in each iteration of a recursive process. The recursive equation for the k'th iteration generates a new partial remainder $X_{k+1}$, as a function of the present partial remainder $X_k$ and the divisor D. The dividend $X_0$ is the remainder before the first iteration. The recursive equation includes either a subtraction or an addition depending on the relative signs of $X_k$ and D.

The relationships are:

$$\text{sgn}X_k = \text{sgn } D \Rightarrow q_{k-1} = 1, \quad X_{k+1} = 2X_k - D2^N$$

$$\text{sgn}X_k \neq \text{sgn } D \Rightarrow q_{k-1} = 0, \quad X_{k+1} = 2X_k + D2^N$$

$$k=0,1,\ldots,N-1.$$

A single recursive equation combining the above conditions is:

$$X_{k+1} = 2X_k - (2q_{k-1} - 1)D2^N. \tag{8.1}$$

The process, from which the definitions of $q_{k-1}$ have been derived, generates an infinite sequence of quotient digits. This is necessary since the dividend and divisor are, in general, incommensurable. However, the truncation of the recursive process after the determination of N+1 digits requires some form of rounding of the quotient. This implies that the least significant digit $q_{N-1}$ cannot be defined as in (8.1). In order to determine the digit values in a quotient of N+1 digits, the ratio $X_0/D$ is derived by combining the first N iterations of (8.1).

$X_0$

$X_1 = 2X_0 - (2_{q-1} - 1)D2^N$

$X_2 = 2X_1 - (2q_0 - 1)D2^N = 2[2X_0 - (2q_{-1} - 1)D2^N] - (2q_0 - 1)D2^N$

$= 2^2X_0 - 2(2q_{-1} - 1)D2^N - (2q_0 - 1)D2^N$

and the final remainder $X_N$ is easily seen to be

$$X_N = 2^NX_0 - D2^N[2^{N-1}(2q_{-1} - 1) + 2^{N-2}(2q_0 - 1) + \ldots + (2q_{N-2} - 1)]$$

$$X_N = 2^NX_0 - D2^N \sum_{k=-1}^{N-2} 2^{N-k-2} (2q_k - 1)$$

$$X_0/D = (X_N/D)2^{-N} + \sum_{k=-1}^{N-2} 2^{N-k-2} (2q_k - 1)$$

$$= (X_N/D)2^{-N} + \sum_{k=-1}^{N-2} 2^{N-k-1}q_k - 2^N \sum_{k=-1}^{N-2} 2^{-k-2}$$

$$= (X_N/D)2^{-N} + 2^Nq_{-1} + \sum_{k=0}^{N-2} 2^{N-k-1}q_k - 2^N(2^{-1} + 2^{-2} + \ldots + 2^{-N})$$

$$= (X_N/D)2^{-N} + 2^Nq_{-1} + \sum_{k=0}^{N-2} 2^{N-k-1} q_k + 1 - 2^N$$

$$X_0 = [(q_{-1} - 1)2^N + \sum_{k=0}^{N-2} 2^{N-k-1} q_k + 1]D + X_N2^{-N} \qquad (8.2)$$

By comparing the above equation with $X_0 = QD + R$, the quotient is seen to be included in the parentheses and $q_{N-1} = 1$. The final remainder is $X_N 2^{-N}$.

1. If $\text{sgn}X_0 = \text{sgn}D$ then $q_{-1} = 1$ and

$$Q = 0 \times 2^N + \sum_{k=0}^{N-2} 2^{N-k-1} q_k + 1 \qquad 0 < Q < 2^N$$

2. If $\text{sgn}X_0 \neq \text{sgn}D$ then $q_{-1} = 0$ and

$$Q = -2^N + \sum_{k=0}^{N-2} 2^{N-k-1} q_k + 1 \qquad -2^N < Q < 0$$

When Q is negative, the representation of Q in the 2's complement form is

$$Q::= 2^{N+1} - |Q| = 2^{N+1} - (2^N - \sum_{k=0}^{N-2} 2^{N-k-1} q_k - 1)$$

$$Q::= 2^N + \sum_{k=0}^{N-2} 2^{N-k-1} q_k + 1$$

Now it should be clear that the various digits of Q can be defined as follows:

$$Q::= \bar{q}_{-1} q_0 q_1 \cdots q_{N-2} 1. \qquad (8.3)$$

The example below serves as an illustration of how division is implemented by using three N-bit registers plus one additional bit.

$X_0 = QD + R,\ 27 = 6 \times 4 + 3,\ N=4.$

D = 0.100

| | | |
|---|---|---|
| $X_0 = 00.001$ | 1 0 1 1 | $sgnX_0 = sgnD$ |
| $2X_0 = 00.011$ | 0 1 1 \| 1 | $q_{-1} = 1$ |
| 11.100 | | sub |
| $X_1 = 11.111$ | 0 1 1 \| 1 | $sgnX_1 \neq sgnD$ |
| $2X_1 = 11.110$ | 1 1 \| 1 0 | $q_0 = 0$ |
| 00.100 | | add |
| $X_2 = 00.010$ | 1 1 \| 1 0 | $sgnX_2 = sgnD$ |
| $2X_2 = 00.101$ | 1 \| 1 0 1 | $q_1 = 1$ |
| 11.100 | | sub |
| $X_3 = 00.001$ | 1 \| 1 0 1 | $sgnX_3 = sgnD$ |
| $2X_3 = 00.011$ | \| 1 0 1 1 | $q_2 = 1$ |
| 11.100 | | sub |
| $X_4 = 11.111$ | | $q_3 = 1$ |

Q::= 00111 = 7 and R::= 11111 = -1.

Note, the partial remainder $X_1$ lies in the interval $-2^{2N} \leq X_1 \leq 2^{2N} - 2$, which implies that it is always correct represented in the (2N + 1)-bit register.

## 8.2. Remainder and Overflow Determination.

In all non-restoring division methods, the remainder and the dividend sometimes end with different signs. As we wish that the signs are equal - except when the remainder is zero - a correction is required. Another problem arises when the quotient exceeds N bits, and a simple check must be provided to detect this overflow situation. Let us consider the following four cases.

| Case | $X_0$ | D |
|---|---|---|
| 1 | $\geq 0$ | $\geq 0$ |
| 2 | $\geq 0$ | $< 0$ |
| 3 | $< 0$ | $\geq 0$ |
| 4 | $< 0$ | $< 0$ |

$X_0 = QD + R$

$X_0 / D = Q + R/D$

$-1 < R/D < 1$

The restrictions on $X_0$, D, Q, and R are for N = 24:

$$-2^{47} \leq X_0 \leq 2^{47} - 1, \qquad -2^{23} \leq D \leq 2^{23} - 1,$$

$$-2^{23} \leq Q \leq 2^{23} - 1, \qquad -2^{23} < R < 2^{23} - 1.$$

Throughout this section q denotes the summation of the bit pattern $\bar{q}_{-1}\, q_0\, q_1 \cdots q_{23}$; a notation which only differs from the one introduced in Section 2 with respect to $q_{-1}$. In the final 24-bit quotient $q_{-1}$ is dropped, but as we shall see later on it plays an important role in the overflow test.

1. $X_0 \geq 0$, $D \geq 0$; $0 \leq R < D$, $0 \leq Q \leq 2^{23} - 1$.

Overflow Condition.

$$Q = X_0/D - R/D \geq 0 \quad \Rightarrow X_0/D \geq R/D \quad \Rightarrow X_0 \geq 0 \quad +);$$

this condition is always fulfilled.

$$Q = X_0/D - R/D \leq 2^{23} - 1 \quad \Rightarrow X_0/D \leq 2^{23} - 1 + R/D < 2^{23} \quad \Rightarrow$$

$$X_0 - 2^{23}D < 0;$$

i.e. correct quotient if $X_1 = 2X_0 - 2^{24}D < 0$. +) (8.4)

Remainder.

Now condition (8.4) implies that $X_2 = 2X_1 + 2^{24}D$ from which we can deduce $X_2 \geq -2 \times 2^{24}D + 2^{24}D = -2^{24}D$, $X_2 < 2 \times 0 + 2^{24}D = 2^{24}D$, or

$$-2^{24}D \leq X_2 < 2^{24}D.$$

Continuous iterations confirm the following result

$$-2^{24}D \leq X_{24} < 2^{24}D.$$

---

+) This statement is also correct for D = 0.

The final modified remainder $r = 2^{24}R$ and Q are

$r = X_{24}$ $\quad Q::= q \quad$ for $\quad 0 \leq X_{24} < 2^{24}D$

$r = X_{24} + 2^{24}D \quad Q::= q-1 \quad$ for $-2^{24}D \leq X_{24} < 0.$

Alternative Overflow Condition.

As shown later on, an overflow test based on $X_1$ requires that the inequalities $X_1 < 0$, $X_1 < 0$, $X_1 \geq 0$, and $X > 0$ are satisfied in the four cases 1, 2, 3, and 4 respectively.

From a technical point of view, this is not a suitable solution for two reasons; firstly, the inequality depends on $X_0$ and D, and secondly, the hardware inplementation of $X_1 \geq 0$ requires a costly decoding. Our alternative solution is a simple comparison of $q_{-1}$ and $q_0$, independent of $X_0$ and D. The proof of this test is one of the major aims of this section.

1Ta) $X_1 < 0$.
As $sgnX_0 = sgnD$ and $sgnX_1 \neq sgnD$ then $\bar{q}_{-1}q_0 = 00$.
The remainder correction has no influence on $q_{-1}$ and $q_0$, consequently
$X_1 < 0 \Rightarrow \bar{q}_{-1} = q_0$.
1Tb) $X_1 \geq 0$.
As $sgn\, X_0 = sgnD$ and $sgnX_1 = sgnD$ then $\bar{q}_{-1}q_0 = 01$, which implies
$X_1 \geq 0 \Rightarrow q_{-1} \neq q_0$.
The alternative overflow test is found by combining 1Ta and 1Tb.
No overflow $\Leftrightarrow X_1 < 0 \Leftrightarrow \bar{q}_{-1} = q_0$; before or after remainder correction.

2. $X_0 \geq 0$, $D < 0$; $0 \leq R < -D$, $-2^{23} \leq Q \leq 0$.

Overflow Condition.

$Q = X_0/D - R/D \leq 0 \Rightarrow X_0/D \leq R/D \Rightarrow X_0 \geq 0$;
this condition is always fulfilled.

$$Q = X_0/D - R/D \geq -2^{23} \Rightarrow X_0/D \geq -2^{23} + R/D > -2^{23} - 1 \Rightarrow X_0 < -2^{23}D - D;$$

i.e. correct quotient if $X_1 = 2X_0 + 2^{24}D < -2D$. (8.5)

Remainder.

Let us divide the investigation of (8.5) into two parts.

2Ra) $X_1 < 0$ ; hence $\mathrm{sgn}X_1 = \mathrm{sgn}D$ and $X_2 = 2X_1 - 2^{24}D$.

We obtain as in 1, $2^{24}D \leq X_{24} < -2^{24}D$, from which r and Q are deduced.

$r = X_{24}$ $\quad Q := q$ $\quad$ for $\quad 0 \leq X_{24} < -2^{24}D$

$r = X_{24} - 2^{24}D$ $\quad Q := q + 1$ $\quad$ for $2^{24}D \leq X_{24} < 0$.

2Rb) $0 \leq X_1 < -2D$; hence $\mathrm{sgn}X_1 \neq \mathrm{sgn}D$ and $X_2 = 2X_1 + 2^{24}D$.

$X_2 \geq 2^{24}D$ and $X_2 < -2^2D + 2^{24}D$ or $2^{24}D \leq X_2 < -2^2D + 2^{24}D$.

$X_3 = 2X_2 - 2^{24}D$ is now calculated to satisfy the inequalities

$X_3 \geq 2^{24}D$, $X_3 < -2^3D + 2^{24}D$, or $2^{24}D \leq X_3 < -2^3D + 2^{24}D$

Continuous iterations result in $2^{24}D \leq X_{24} < -2^{24}D + 2^{24}D = 0$,

that is a correction of the remainder is always necessary.

$r = X_{24} - 2^{24}D$ $\quad Q ::= q + 1$ $\quad$ for $2^{24}D \leq X_{24} < 0$.

Note, $q_0 = q_1 = \ldots = q_{22} = 1$.

Alternative Overflow Condition.

2Ta) $X_1 < 0$.

As $\mathrm{sgn}X_0 \neq \mathrm{sgn}D$ and $\mathrm{sgn}X_1 = \mathrm{sgn}D$ then $\bar{q}_{-1}q_0 = 11$.
The bit pattern $\bar{q}_{-1}q_0$ is only influenced by the remainder correction if $q = 11 \ldots 1$ and $X_{24} \leq 0$, in which case $\bar{q}_{-1}q_0$ becomes 00.
Hence $X_1 < 0 \Rightarrow \bar{q}_{-1} = q_0$.

2Tb) $0 \leq X_1 < -2D$.
As sgn $X_0 \neq$ sgnD and sgn$X_1 \neq$ sgnD then $\bar{q}_{-1}q_0 = 10$.
The necessary correction, however, cancels the discrepancy between the two overflow tests, because Q::= 1011 ... 1 + 1 = 1100 ... 0 (= $-2^{23}$).
Therefore, so far, the alternative test is valid after remainder correction.

2Tc) $X_1 \leq -2D$.
As sgn$X_0 \neq$ sgnD and sgn$X_1 \neq$ sgnD then $\bar{q}_{-1}q_0 = 10$.
This shows that the $\bar{q}_{-1}q_0$ test is correct before correction, but as seen from the example below this is not generally true when a correction is involved.

Example. $X_0 = QD + R$. $127 = Q(-1) + R$. $N = 4$.

| | | | | |
|---|---|---|---|---|
| 1111 | $X_0$ | 00111 | 1 1 1 1 | |
| | | 01111 | 1 1 1 \| 0 | $q_{-1} = 0$ |
| | | 11111 | | |
| | $X_1$ | 01110 | 1 1 1 \| 0 | |
| | | 11101 | 1 1 \| 0 0 | $q_0 = 0$ |
| | | 11111 | | |
| | $X_2$ | 11100 | 1 1 \| 0 0 | |
| | | 11001 | 1 \| 0 0 1 | $q_1 = 1$ |
| | | 00001 | | |
| | $X_3$ | 11010 | 1 \| 0 0 1 | |
| | | 10101 | \| 0 0 1 1 | $q_2 = 1$ |
| | | 00001 | | |
| | $X_4$ | 10110 | | |

$q + 1 = 10111 + 1 = 11000$ and Q::= 1000.

The consequence of this example is that the $q_{-1}q_0$ test is only valid before any correction, and therefore the otherwise correct quotients obtained in 2Rb are considered to exceed the N-bit capacity. We would like to emphasize that the test $X_1 < -2D$ is normally too complex to be used in a microprogram, so anyway we would have restricted ourselves to the interval $X_1 < 0$.

$X_1 = 2X_0 + 2^{24}D < 0$ implies $X_0/D = Q + R/D > -2^{23}$, hence the simple overflow test becomes:

No overflow and $Q + R/D > -2^{23} \iff X_1 < 0 \iff \bar{q}_{-1} = q$ before remainder correction.

3. $X_0 < 0$, $D \geq 0$; $-D < R \leq 0$, $-2^{23} \leq Q \leq 0$.

Overflow condition.

$Q = X_0/D - R/D \leq 0 \Rightarrow X_0/D \leq R/D \Rightarrow X_0 < 0$ +);
this condition is always fulfilled.

$Q = X_0/D - R/D \geq -2^{23} \Rightarrow X_0/D \geq -2^{23} + R/D > -2^{23} - 1 \Rightarrow X_0 > -2^{23}D - D$;

i.e. correct quotient if $X_1 = 2X_0 + 2^{24}D > -2D$. +) (8.6)

Remainder.

To investigate this, we divide the range of $X_1$ into two parts.

3Ra) $X_1 \geq 0$; hence $\mathrm{sgn}X_1 = \mathrm{sgn}D$ and $X_2 = 2X_1 - 2^{24}D$.

We obtain as in 1, $-2^{24}D \leq X_{24} < 2^{24}D$, from which r and Q are deduced.

| | | |
|---|---|---|
| $r = X_{24} - 2^{24}D$ | $Q::= q + 1$ | for $0 < X_{24} < 2^{24}D$ |
| $r = X_{24}$ | $Q::= q$ | for $-2^{24}D < X_{24} \leq 0$ |
| $r = X_{24} + 2^{24}D$ | $Q::= q - 1$ | for $X_{24} = -2^{24}D$. |

+) This statement is also correct for $D = 0$.

3Rb) $-2D < X_1 < 0$; hence $sgnX_1 \neq sgnD$ and $X_2 = 2X_1 + 2^{24}D$.

Continuous iterations, as in 2Rb, confirm the following inequality

$$0 = -2^{24}D + 2^{24}D < X_{24} < 2^{24}D,$$

that is a correction of the remainder is always necessary.

$$r = X_{24} - 2^{24}D \qquad Q::= q + 1 \qquad \text{for } 0 < X_{24} < 2^{24}D.$$

Note, $q_0 = q_{-1} = \ldots = q_{22} = 1$.

Alternative Overflow Condition.

3Ta) $X_1 \geq 0$.
As $sgnX_0 \neq sgnD$ and $sgnX_1 = sgnD$ then $\bar{q}_{-1}q_0 = 11$.
The bit pattern $\bar{q}_{-1}q_0$ is only influenced by the remainder correction if $q = 11 \ldots 1$ and $X_{24} > 0$, in which case $\bar{q}_{-1}q_0$ becomes 00.
Hence $X_1 \geq 0 \Rightarrow \bar{q}_{-1} = q_0$.

3Tb) $-2D < X_1 < 0$.
As $sgnX_0 \neq sgnD$ and $sgnX_1 \neq sgnD$ then $\bar{q}_{-1}q_0 = 10$.
Although $\bar{q}_{-1}q_0 = 11$ after the appropriate remainder modification, the quotient is considered to exceed the N-bit capacity. This case is similar to 2Tb.

3Tc) $X_1 \leq -2D$.
As $sgnX_0 \neq sgnD$ and $sgnX_1 \neq sgnD$ the $\bar{q}_{-1}q_0 = 10$.
$X_1 = 2X_0 + 2^{24}D \geq 0$ implies $X_0/D = Q + R/D \geq -2^{23}$ and this inequality in conjunction with 3Ta, 3Tb, and 3Tc determines the alternative overflow test.

No overflow and $Q + R/D \geq -2^{23} \Leftrightarrow X_1 \geq 0 \Leftrightarrow \bar{q}_{-1} = q_0$ before remainder correction.

## 4. $X_0 < 0$, $D < 0$; $D < R \leq 0$, $0 \leq Q \leq 2^{23} - 1$.

Overflow condition.

$Q = X_0/D - R/D \geq 0 \Rightarrow X_0/D \geq R/D \Rightarrow X_0 < 0$;
this condition is always fulfilled

$Q = X_0/D - R/D \leq 2^{23} - 1 \Rightarrow X_0/D \leq 2^{23} - 1 + R/D < 2^{23} \Rightarrow X_0 > 2^{23}D$;

i.e. correct quotient if $X_1 = 2X_0 - 2^{24}D > 0$. (8.7)

Remainder.

Now condition (8.7) implies that $X_2 = 2X_1 + 2^{24}D$ and we obtain similar to 1 $-2^{24}D \leq X_{24} < 2^{24}D$.
The rules for r and Q are therefore as follows.

| | | |
|---|---|---|
| $r = X_{24} + 2^{24}D$ | $Q::= q - 1$ | for $0 < X_{24} < -2^{24}D$ |
| $r = X_{24}$ | $Q::= q$ | for $2^{24}D < X_{24} \leq 0$ |
| $r = X_{24} - 2^{24}D = 0$ | $Q:= q + 1$ | for $X_{24} = 2^{24}D$ . |

Alternative Overflow Condition.

4Ta) $X_1 > 0$.
As $sgnX_0 = sgnD$ and $sgnX_1 \neq sgnD$ then $\bar{q}_{-1}q_0 = 00$.
The remainder correction has only influence on $q_{-1}q_0$ if the two conditions $q_1 = q_2 = \ldots = q_{22} = 1$ and $X_{24} = 2^{24}D$ are satisfied. It is easily proved that these two conditions are contradictory.

| | |
|---|---|
| $q_0 = 0$ | $\Rightarrow X_2 = 2X_1 + 2^{24}D > 2^{24}D$ |
| $q_1 = 1 \Rightarrow X_2 < 0$ | $\Rightarrow X_3 = 2X_2 - 2^{24}D > 2^{24}D$ |
| $q_2 = 1 \Rightarrow X_3 < 0$ | $\Rightarrow X_4 = 2X_3 - 2^{24}D > 2^{24}D$ |
| ..... | |
| $q_{22} = 1 \Rightarrow X_{23} < 0$ | $\Rightarrow X_{24} = 2X_{22} - 2^{24}D > 2^{24}D$. |

Hence $X_1 > 0 \Rightarrow \bar{q}_{-1} = q_0$.

4Tb) $X_1 = 0$.
As $sgnX_0 = sgnD$ and $sgnX_1 \neq sgnD$ then $\bar{q}_{-1}q_0 = 00$, which is in contradistinction to (8.7). But from the above formulae it follows that $X_2 = X_3 = \ldots = X_{24} = 2^{24}D$ and $q_1 = q_2 = \ldots = q_{23} = 1$; therefore $Q::= q + 1$ and $\bar{q}_{-1}q_0 = 01$ after correction.

4Tc) $X_1 < 0$.
As $sgnX_0 = sgnD$ and $sgnX_1 = sgnD$ then $\bar{q}_{-1}q_0 = 01$, which implies $X_1 < 0 \Rightarrow q_{-1} \neq q_0$

The overflow test is found by combining 4Ta, 4Tb, and 4Tc.

No overflow $\langle\Rightarrow X_1 > 0 \langle\Rightarrow \bar{q}_{-1} = q_0$ both before and after remainder correction.

This completes the proof of the alternative overflow condition, which we can summarize as follows:
The quotient is correct represented within N bits if

a) $\bar{q}_{-1} = q_0$ before remainder correction and

b) $\bar{q}_{-1} = q_0$ after remainder correction,

the latter being a consequence of the former except for $X_0 < 0$, $D < 0$, and $X_N = 2^N D$.

## 8.3. A Survey.

| Case | $X_0$ | D | $\bar{q}_{-1}q_0$ | $X_{24}$ | r | Q | | |
|---|---|---|---|---|---|---|---|---|
| 1a | $\geq 0$ | $\geq 0$ | 00 | $\geq 0$ | $X_{24}$ | q | $00q_1 \ldots 1$ | $Q + R/D < 2^{23}$; $R \geq 0$ |
| 1a | $\geq 0$ | $\geq 0$ | 00 | $< 0$ | $X_{24}+2^{24}D$ | q-1 | $00q_1 \ldots 0$ | |
| 1b | $\geq 0$ | $\geq 0$ | 01 | | | | | capacity exceeded. |
| 2a | $\geq 0$ | $< 0$ | 11 | $\geq 0$ | $X_{24}$ | q | $11q_1 \ldots 1$ | $Q + R/D > -2^{23}$, $R \geq 0$ |
| 2a | $\geq 0$ | $< 0$ | 11 | $< 0$ | $X_{24}-2^{24}D$ | q+1 | $11\dot{q}_1 \ldots 0$ or $000 \ldots 0$ | |
| 2b,2c | $\geq 0$ | $< 0$ | 10 | | | | | capacity exceeded. |
| 3a | $< 0$ | $\geq 0$ | 11 | $> 0$ | $X_{24}-2^{24}D$ | q+1 | $11\dot{q}_1 \ldots 0$ or $000 \ldots 0$ | $Q + R/D \geq -2^{23}$, $R \leq 0$ |
| 3a | $< 0$ | $\geq 0$ | 11 | $> -2^{24}D$, $\leq 0$ | $X_{24}$ | q | $11q_1 \ldots 1$ | |
| 3a | $< 0$ | $\geq 0$ | 11 | $= -2^{24}D$ | $X_{24}+2^{24}D=0$ | q-1 | $11q_1 \ldots 0$ | |
| 3b,3c | $< 0$ | $\geq 0$ | 10 | | | | | capacity exceeded. |
| 4a | $< 0$ | $< 0$ | 00 | $> 0$ | $X_{24}+2^{24}D$ | q-1 | $00q_1 \ldots 0$ | $Q + R/D < 2^{23}$, $R \leq 0$ |
| 4a | $< 0$ | $< 0$ | 00 | $> 2^{24}D$, $\leq 0$ | $X_{24}$ | q | $00q_1 \ldots 1$ | |
| 4a | $< 0$ | $< 0$ | 00 | $= 2^{24}D$ | $X_{24}-2^{24}D=0$ | q+1 | $00\dot{q}_1 \ldots 0$ | |
| 4a | $< 0$ | $< 0$ | 00 | $= 2^{24}D$ | $X_{24}-2^{24}D=0$ | q+1 | $010 \ldots 0$ | capacity exceeded. |
| 4b,4c | $< 0$ | $< 0$ | 01 | | | | | |

$\dot{q} = q$ or $\bar{q}$