ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
XVI, 1 (1977)

JANINA BAZAN (Wrocław)

ALGORITHM 54

AN ALGORITHM FOR BINARY-TO-DECIMAL CONVERSION OF REAL NUMBERS

1. Procedure declaration. Assume the following representation of numbers. Let

$$(A_j \ldots A_0)_B = \big(\ldots (A_j \cdot B + A_{j-1}) \cdot B + \ldots + A_1\big) \cdot B + A_0,$$

where $A_i \in \langle 0, B-1 \rangle$ for $i < j$, and $A_j \in \langle 1, B-1 \rangle$;

$$(0.A_{-1}A_{-2} \ldots A_{-j})_B = \big(\ldots (A_{-j}/B + A_{-j+1})/B + \ldots + A_{-1}\big)/B,$$

where $A_{-1} \neq 0$, and $A_{-i} \in \langle 0, B-1 \rangle$ for $i = 1, \ldots, j$;

(1) $$(0.A_{-1} \ldots A_{-j})'_B = \big((\ldots A_{-j}/B + \ldots + A_{-2})/B + A_{-1}\big)/10^k,$$

where $10^{k-1} \leqslant A_{-1} < 10^k \leqslant B$, and $A_{-i} \in \langle 0, B-1 \rangle$ for $i = 2, \ldots, j$.

The procedure $CONV$ transforms the binary floating-point number

(2) $$x = m \cdot 2^c$$

(where $0.5 \leqslant m \leqslant 1 - 2^{-p+1}$ for $x > 0$ or $-1 \leqslant m \leqslant -0.5 - 2^{-p+1}$ for $x < 0$, p is the number of bits in the mantissa of the computer word) to the form

(3) $$(-1)^i \cdot (0.A_{-1} \ldots A_{-j})'_B \cdot 10^{E10},$$

where

$$i = \begin{cases} 0 & \text{if } x \geqslant 0, \\ 1 & \text{if } x < 0, \end{cases}$$

and B is the highest power of 10 smaller than the computer word.

Data:

x — real number of type (2);
n — dimension of the integer array A (see Results);
N — length of the computer word (in bits);
j — if the binary exponent of the number x is smaller than p, then j is the number of digits in representation (6), otherwise its value is inessential;

MULT — procedure with the following procedure head:
procedure *MULT*(*a1*, *a2*, *e*, *c*);
value *e*;
integer *a1*, *a2*, *e*, *c*;
realizing the function:
$a2 := (a1 * 2 \uparrow e + c) \pmod{B}$,
$c := (a1 * 2 \uparrow e + c) \div B$;

DIV — procedure with the following procedure head:
procedure *DIV*(*a1*, *a2*, *e*, *c*);
value *e*;
integer *a1*, *a2*, *e*, *c*;
realizing the function:
$a2 := (c * B + a1) \div 2 \uparrow e$,
$c := (c * B + a1) \pmod{2 \uparrow e}$;

D — procedure with the following procedure head:
procedure *D*(*a*, *A*, *w*, *j*);
value *a*;
integer *a*, *w*, *j*;
integer array *A*;
This procedure presents the integer number a in the B radix system in the form $(U_{j-1} \ldots U_0)_B$, where $U_i \in \langle 0, B-1 \rangle$ for $i < j-1$ and $U_{j-1} \in \langle 1, B-1 \rangle$. The values of U_i ($i = j-1, \ldots, 0$) are placed in consecutive elements of A beginning from the element $A[w-j+1]$.

BDIV — procedure with the following head:
procedure *BDIV*(*A*, *c*, *i*, *e*);
value *e*;
integer *c*, *i*, *e*;
integer array *A*;
After execution of the procedure body, the value of the variable i is such that

$$(\ldots A[1] * B + \ldots) * B + A[i] \geqslant 2 \uparrow e$$

and

$$c := (\ldots A[1] * B + \ldots) * B + A[i-1].$$

Results:

$A[1:j]$ — array containing consecutive numbers A_i from (3) ($i = -1, \ldots, -j$);

j — number of calculated digits of the number x in the B radix system;

k — number of significant decimal digits in A_{-1} (see (1));

E10 — decimal exponent of the number x.

Non-local identifiers:

const — integer number such that $10\uparrow const = B$;

mant — procedure with the following procedure head:

procedure *mant*(x, A, b, i);
value b;
real x;
integer i;
integer array A;
Boolean b;
This procedure presents the mantissa m' of the number x in the B radix system in the form $(U_{i-1} \ldots U_0)_B$, where $U_j \in \langle 0, B-1\rangle$ for $j < i-1$, and $U_{i-1} \in \langle 1, B-1\rangle$. The values of U_j are placed in consecutive elements of A beginning from the element $A[1]$ if $b \equiv$ **true** and beginning from the element $A[n-i+1]$ if $b \equiv$ **false.**

cecha — procedure with the following procedure head:

procedure *cecha*(x, e);
value x;
real x;
integer e;
giving the value of the parameter e equal to e' (see (4)).

2. **Method used.** Let the number x have the binary representation $m \cdot 2^c$, where the mantissa m belongs to the interval $\langle -1, -0.5-2^{-p}\rangle$ if $x < 0$ or to $\langle 0.5, 1-2^{-p}\rangle$ if $x > 0$. To obtain the representation of x in the B radix system in the form (3) we proceed as follows:

(i) Transform the mantissa m and the exponent e according to the formulas

$$(4) \qquad m' = m \cdot 2^p \quad \text{and} \quad e' = e-p.$$

(ii) Present m' in the form $(A_n^0 \ldots A_0^0)_B$ using the method described in [1]. This method consists in repeated division of the number by B; its consecutive digits are calculated by the formulas

$$A_0^0 = m' \pmod{B}, \quad A_1^0 = \text{entier}(m'/B) \pmod{B}$$

until

$$\text{entier}\big(\ldots \text{entier}(m'/B)/B \ldots /B\big) = 0.$$

This method is realized in the body of the procedure D.

(iii) Later, either multiply e' (for $e' > 0$) or divide $(A_n^0 \ldots A_0^0)_B$ (for $e' < 0$) by $2^{|e'|}$. This method is opposite to that proposed by Knuth in [1].

The algorithm used in the procedure *CONV* distinguishes two cases: (i) — for $e' > 0$, and (ii) — for $e' < 0$.

```
procedure CONV(x,A,n,N,j,k,E10,MULT,DIV,D,BDIV);
value x,n,N;
integer n,N,j,k,E10;
real x;
integer array A;
procedure MULT,DIV,D,BDIV;
begin
 integer i,e,s,N1,l,c;
 Boolean b;
 N1:=N-1;
 if x=.0
  then
  begin
   E10:=0;
   k:=1;
   go to ET2;
  end;
 cecha(x,e);
 b:=e≤0;
 s:=sign(x);
 x:=abs(x);
 mant(x,A,b,E10);
 if b
  then
  begin
   if e=0
    then go to ET;
   for N1:=N1 while e>N1,e do
    begin
     e:=e-N1;
```

```
      BDIV(A,c,i,N1);
      E10:=E10-i+1;
      for l:=i step 1 until j do
       DIV(A[l],A[l-i+1],N1,c);
      j:=j-i+1
     end;
   end
   else
   begin
    j:=n-E10+1;
    for N1:=N1 while e>N1,e do
     begin
      e:=e-N1;
      c:=0;
      for i:=n step -1 until j do
       MULT(A[i],A[i],N1,c);
      if c≠0
       then D(c,A,j-1,i);
      j:=j-i
     end;
    E10:=n-j+1;
    copy(E10,A[j],A[1]);
   end;
ET:
  c:=A[1];
  for k:=1 step 1 until const do
   if c<10↑k
    then go to ET1;
ET1:
  E10:=E10×const-const+k;
ET2:
 end
```

Case (i) ($e' > 0$). Multiply the mantissa $(A^0_{n_0} \dots A^0_0)_B$ by $2^{e'}$. This multiplication is executed according to the schema

$$\underbrace{(\dots (A^0_{n_0} \dots A^0_0)\cdot 2^{N-1}\cdot \dots \cdot 2^{N-1})}_{m \text{ times}}\cdot 2^{e'(\mathrm{mod}(N-1))}, \quad \text{where } m = e' \div (N-1).$$

Let the j-th product be of the form $(A^j_{n_j} \dots A^j_0)_B$. The consecutive digits of the product are calculated by the formulas

$$\text{(5)} \qquad \begin{aligned} & c := 0, \\ & \left.\begin{aligned} & a := A^{j-1}_i \cdot 2^e + c \\ & A^j_i := a \ (\mathrm{mod}\ B) \\ & c := a \div B, \end{aligned}\right\} \quad (i = 0, \dots, n_{j-1}), \\ & e = \begin{cases} N-1 & \text{if } j \neq m+1, \\ e' \left(\mathrm{mod}\ (N-1)\right) & \text{if } j = m+1, \end{cases} \end{aligned}$$

and $A^j_{n_j}, \dots, A_{n_{j-1}+1}$ are consecutive digits of the representation of the value of the variable m' in the B radix system.

For fixed j and i the calculations of (5) are realized by the procedure *MULT*. In the procedure *MULT*, multiplications by powers of 2 are calculated as shifts of the computer word. To multiply the number A^{j-1}_i by 2^e, this number must be located in the register A, the register W must be cleared and the contents of AW is shifted to the right by $N-e$ bits. If $e = N-1$, it suffices to shift the contents of AW to the right by one bit only.

Having x represented in the form $(A^{m+1}_{n_{m+1}} \dots A^{m+1}_0)_B$, we receive the form (3) in a simple way.

Case (ii) ($e' < 0$). m' is represented in the form

$$\text{(6)} \qquad (0.A^0_{-1} \dots A^0_{-n_0-1} \dots A^0_{-j})_B \cdot B^{E_0},$$

where

$$A^0_{-1} = A^0_n,\ A^0_{-2} = A^0_{n-1}, \dots, A^0_{-n_0-1} = A^0_0,\ A^0_{-n_0-2} = \dots = A^0_{-j} = 0$$

and

$$E_0 = n_0 + 1.$$

Then $(0.A^0_{-1} \dots A^0_{-j})_B \cdot B^{E_0}$ is to be divided by $2^{|e'|}$ according to the schema

$$\underbrace{(A^0_{-1} \dots A^0_{-j})\cdot B^{E_0}/2^{N-1}/ \dots /2^{N-1}}_{m \text{ times}}/2^{e'(\mathrm{mod}(N-1))}, \quad \text{where } m = e' \div (N-1).$$

Let $(0.A^l_{-1} \dots A^l_{-j_l})\cdot B^E$, where $A^l_{-1} \neq 0$, be the l-th quotient. The digits of the quotients are calculated by the formulas

$$a := 0,\ k := 0,$$

1. $a := a \cdot B + A^{l-1}_{-1-k}$,

2. if $a < 2^e$, then $k := k+1$ and go to 1 else,

$$A^l_{-1} := a \div 2^e, \tag{7}$$

$$E_l := E_{l-1} - k,$$

$$j_l := j_{l-1} - k,$$

3. $a := \big(a \pmod{2^e}\big) \cdot B + A^{l-1}_{-i-k}$ for $i = 2, \ldots, j_l$,

$$A^l_{-i} := a \div 2^e \quad \text{for } i = 2, \ldots, j_l, \tag{8}$$

where $e = N-1$ if $l \neq m+1$, and $e = |e'| \pmod{(N-1)}$ if $l = m+1$.

Finally,

$$x = (0.A^{m+1}_{-1} \ldots A^{m+1}_{-j_{m+1}})_B \cdot B^{E_{m+1}}.$$

From this formula it is easy to obtain formula (3).

The integer division from (7) and (8) is done in the procedure *DIV*. The procedure *DIV* calculates consecutive digits of the quotient as follows:

1. locate the number a in AW,
2. shift the contents of AW to the left by $N-e$ bits.

The register A contains the result of the integer division of a by 2^e. If $e = N-1$, the contents of AW is shifted only by one bit to the left.

3. Certification. The procedure *CONV* has been written also in the machine language of the ODRA 1204 computer using only fixed-point instructions. It has been used to printing the numbers. The comparison of the printouts of numbers in the assembler AS 1204 and using the procedure *CONV* are given in Table 1.

TABLE 1. Printouts of numbers

Number	*AS*-language	Procedure *CONV*
1	$99.99999999996_{10}-002$	$10.00000000000_{10}-001$
10^{50}	$10.00000000686_{10}+049$	$99.99999999810_{10}+048$
-10^{100}	$-10.00000001470_{10}+099$	$-10.00000000053_{10}+099$
28072624	28072624.000719	28072624.000000
32942252	32942251.999861	32942252.000000
909847776	909847776.00897	909847776.00000
64318876	64318876.000868	64318876.000000
7604325.5	7604325.5001363	7604325.5000000
923745504	923745504.00927	923745504.00000
642559200	642559199.99777	642559200.00000

Reference

[1] D. E. Knuth, *The art of computer programming*, Vol. 2, Reading 1972.

INSTITUTE OF INFORMATICS
UNIVERSITY OF WROCŁAW
50-384 WROCŁAW

Received on 2. 11. 1976

ALGORYTM 54

JANINA BAZAN (Wrocław)

ALGORYTM ZAMIANY LICZB RZECZYWISTYCH Z SYSTEMU DWÓJKOWEGO NA DZIESIĘTNY

STRESZCZENIE

Procedura *CONV* przedstawia liczbę zmiennopozycyjną $x = m\cdot 2^c$ w postac

$$(-1)^i\cdot(0.A_{-1}\ldots A_{-j})'_B\cdot 10^{E10},$$

gdzie $i = 0$, jeżeli $x \geqslant 0$, albo $i = 1$, jeżeli $x < 0$, a B jest najwyższą potęgą 10 mniejszą od słowa maszynowego.

Dane:

x — liczba rzeczywista;

n — wymiar tablicy A;

N — długość słowa maszynowego (w bitach);

j — jeżeli cecha dwójkowa liczby x jest mniejsza niż p, to j jest liczbą cyfr w (6), w przeciwnym razie wartość j nie ma znaczenia;

MULT — procedura o nagłówku **procedure** *MULT(a1, a2, e, c)*; **value** *e*; **integer** *a1, a2, e, c*; wykonująca czynności: $a2 := (a1*2\uparrow e+c) \pmod B$, $c := (a1*2\uparrow e+c)\div B$;

DIV — procedura o nagłówku: **procedure** *DIV(a1, a2, e, c)*; **value** *e*; **integer** *a1, a2, e, c*; wykonująca czynności: $a2 := (c*B+a1)\div 2\uparrow e$, $c := (c*B+a1) \pmod{2\uparrow e}$;

D — procedura o nagłówku: **procedure** *D(a, A, w, j)*; **value** *a*; **integer** *a, w, j*; **integer array** *A*. Procedura ta przedstawia liczbę *a* w systemie o podstawie B w postaci $(U_{j-1}, \ldots, U_0)_B$, przy czym $A[w-j+1]$ zawiera cyfrę U_{j-1}, $A[w-j+2]$ cyfrę U_{j-2} itd.

BDIV — procedura o nagłówku: **procedure** *BDIV(A, c, i, e)*; **value** *e*; **integer** *c, i, e*; **integer array** *A*. Po wykonaniu treści procedury wartość zmiennej i jest taka, że

$$(\ldots A[1]*B+\ldots)*B+A[i] \geqslant 2\uparrow e$$

oraz

$$c := (\ldots A[1]*B+\ldots)*B+A[i-1].$$

Wyniki:

$A[1:j]$ — tablica zawierająca kolejne cyfry liczby x w systemie o podstawie B;

j — liczba wyznaczonych cyfr;

k — liczba znaczących cyfr dziesiętnych w A_{-1};

$E10$ — cecha dziesiętna liczby x.

Nazwy nielokalne:

const — liczba całkowita spełniająca równość $10 \uparrow const = B$;

mant — procedura o nagłówku **procedure** *mant*(x, A, b, i); **value** b; **real** x; **integer array** A; **Boolean** b; **integer** i; procedura wyznacza mantysę m' liczby x (patrz p. 2); kolejne cyfry mantysy w systemie o podstawie B umieszcza się w początkowych (gdy $b \equiv$ **true**) lub końcowych (gdy $b \equiv$ **false**) elementach tablicy A;

cecha — procedura o nagłówku **procedure** *cecha*(x, e); **value** x; **real** x; **integer** e; oblicza cechę e' (patrz p. 2) liczby x.

Algorytm polega na k-krotnym dzieleniu (gdy cecha dwójkowa liczby jest mniejsza od p) lub mnożeniu (w przeciwnym wypadku) mantysy m' przez $2\uparrow(N-1)$, gdzie $k = e' \div (N-1)$, oraz przez $2\uparrow e' \pmod{(N-1)}$.