UNIVERSITY OF ILLINOIS

DIGITAL COMPUTER

LIBRARY ROUTINE A 8 - 321

TITLE:	Multiple Precision Integer Subroutines (DOI or SADOI)
TYPE:	Set of Independent Subroutine
TEMPORARY STORAGE:	0, 1, 2
DURATION:	See below
ACCURACY:	Every integer is accurate to the least-significant digit.
DESCRIPTION:	This is a set of subroutines which will enable the user to perform simple computations on integers to full precision, without restricting the range of the integer variables to being less than 2^{39}. The set at present includes routines for addition, multiplication, shifting any number of binary digits left or right, output, small-integer output, and number transfer. Each is described in detail below.

NUMBER REPRESENTATION: The multiple-precision integer

$A = A_0 + A_1(2^{39}) + A_2(2^{39})^2 + \ldots An_A(2^{39})^nA$,

where $A_i < 2^{39}$, is represented in the Williams Memory by a set of numbers in memory locations m_A, m_A+1, m_A+2, ..., $m_A + n_A + 1$, such that

Location	Contains
m_A	Key Word for A
m_A+1	A_0
m_A+2	A_1
.	.
.	.
.	.
$m_A + n_A + 1$	A_{nA}

Thus, n_A+2 is the number of locations required for storing the "digits" A_i of the integer A.

The Key word is formed in such a way that on entry to a subroutine, the Key word has n_A+1 in the left-hand address position, m_A+1 (the location of A_0) in the right-hand address position, and all other digits zero. Results of arithmetic operations will also be stored in this form.

Key A =

0	n_A+1	0	m_A+1
0 . . . 9	10 . . . 19	20 . . . 29	30 . . . 39

EXAMPLES: Suppose the integer 549, 755, 813, 896 = 2^{39} + 8 is to be stored at location 100. Then the orders

002F 00101F	or	003F 00101F
00F 008F		00F 008F
00F 001F		00F 001F
		00F 00F

are both correct representations.

To store the integer 2^{100} at location 200, the order-pair representation is

003F 00201F

00F 00F

00F 00F

004F 00F.

ADDITION SUBROUTINE: 50 words, temporary storage 0, 1, 2.

This subroutine forms the sum of two positive multiple-precision integers, and stores the result. To form the sum A + B = C, this subroutine is entered by the instructions

p-1	L5 m_BF	m_B = location of Key B, now in accumulator
	50 m_AF	m_A = location of Key A
p	50 pF	
	26 xF	x = location of Addition Subroutine
p+1	00 m_CF	m_C = location of Key C, where sum will be stored
	-- --	
p+2		

When the subroutine returns control to the left-hand side of p+2, the sum C will be in locations m_C, m_C+1, m_C+2, ...m_C + n_C+1, Key C will be in the accumulator, and m_C+1 will be in the right-hand address position of the quotient register. The routine may or may not use location m_C+n_C+2.

It is possible to put $m_C = m_A$ or m_B and thus to have the sum stored in the position of one of the original operands.

Let $n_1 = \min(n_A, n_B)$, and $n_2 = \max(n_A, n_B)$. Then, the time required is $(6 + 0.7n_1 + 1.3n_2)$ msec.

MULTIPLICATION SUBROUTINE:

55 words, temporary storage 0, 1, 2.

This subroutine forms the product of two positive multiple-precision integers, and stores the result. To form the product A x B = C, this subroutine is entered by the instructions

p-1	L5 m_AF	m_A = location of Key A, now is accumulator
	50 m_BF	m_B = location of Key B
p	50 pF	
	26 xF	x = location of Multiplication subroutine
p+1	00 m_CF	m_C = location of Key C, where product will be stored
	-- --	
p+2		

When the subroutine returns control to the left-hand side of p+2, the product C will be in locations m_C, m_C+1, m_C+2 ... $m_C + n_C + 1$, Key C will be in the accumulator, and m_C+1 will be in the right-hand address position of the quotient register. The routine may or may not use locations $m_C + n_C + 2$ and $m_C + n_C + 3$. It should be noted that it is <u>not</u> possible to put $m_C = m_A$ or m_B as in the case of the Addition Subroutine.

Time required = $8.3 + 0.55n_A + [1.38 + 2.43(n_A+1)]\ (n_B+1)$ milliseconds.

If one operand is consistently much larger than the other, a slight amount of time may be saved by always letting the larger be A.

SHIFTING SUBROUTINE: 87 words, temporary storage 0, 1, 2.

This subroutine will shift the multiple-precision integer any number of binary digits, that is, will multiply the integer A by 2^S, to form B: $A2^S = B$. The number S may be either positive or negative. If A is shifted left S places, this is equivalent to an unbiased, multiplication by 2^S. If A is shifted right S places, S binary digits are discarded, and the remaining number is not rounded.

To shift the multiple-precision integer A by S binary places, enter this subroutine with S x 2^{-39} in the accumulator, and with the orders

	50 m_AF	
p	50 pF	for a left shift, or with
	26 xF	
p+1	-- --	

	J0 m_AF	
p	50 pF	for a right shift, where
	26 xF	
p+1	-- --	

m_A = location of Key A

x = location of this routine.

If S is zero, this routine returns control to the right-hand side of p+1 immediately; is S is negative, then the shift is in the direction opposite to that specified by the 50 or J0 program parameter.

When the subroutine returns control to the right-hand side of p+1, the number B will be in locations m_A, m_A+1, m_A+2, ... $m_A + n_B + 1$. In the case where the number of right shifts exceeds the number of possible available digits, (i.e. the number A is shifted away) location m_A+1 is set to zero and $n_B = 0$.

Operation times are difficult to estimate accurately.

If S = 39N + r, is used to define N and R then approximate times are

Right Shift: $\begin{cases} T \approx 5.0 + (1.05 + .02R)(n_A - N) \text{ m sec for } R \neq 0 \\ \quad\;\; 5.0 + 0.76(n_A - N) \text{ m sec for } R = 0 \end{cases}$

Left shift: $\begin{cases} T \approx 8.0 + (1.1 + .02R)n_A + 0.76N \text{ m sec for } R \neq 0 \\ \quad\;\; 8.0 + 0.76\, n_A + 0.6N \text{ m sec for } R = 0 \end{cases}$

SMALL-INTEGER PRINT SUBROUTINE:

29 words, temporary storage 0, 1, 2.

This subroutine will punch a positive integer which is less than 10^{11}, with suppression of leading zeros. This routine is used as part of the output routine for a multiple-precision integer, so that it contains controls for spaces and line-feeds. In order to keep correct output format, this routine can be entered in two ways. The first entry is used to reset all counters and the zero-suppression test; subsequent entries retain the information left when the subroutine last returned control by its link. The controls will punch a space after every 10 digits, except that after every 50 digits, there will be a line-feed and two two-hole delay characters. Thus, if a single integer is to be printed, a space will appear between the tenth and eleventh digits if the integer is greater than 10^{10}. This routine does not precede the number by a line-feed character.

To punch the positive integer N, enter this routine with $N \times 2^{-39}$ in the accumulator, and with the orders

	-- -
p	50 pF
p+1	26 xF

whenre x is the location of this routine. Leading zeros of N will be suppressed; thus if N is zero, nothing will be punched.

For subsequent entries which retain the previous information regarding zero suppression, spaces, and line-feeds, enter with $N \times 2^{-39}$ in the accumulator and with the orders

	-- -
p	54 pF
p+1	26 xF

The time required is 16 milliseconds per character punched.

OUTPUT SUBROUTINE:

40 + 29 words, temporary storage 0, 1, 2.

This subroutine will convert to decimal and punch the multiple-precision integer A with Key word at m_A. This routine uses roughly $n_A(1.06)^{n_A} + 2$ locations for converting A, starting at m_A. The number A is destroyed.

This routine does not punch line-feed characters either preceding of following the number to be printed; it punches two one-hole delays immediately following the last digit of the number. Space and line-feed controls are carried in the small-integer print routine described above.

THE SMALL-INTEGER PRINT ROUTINE MUST FOLLOW IMMEDIATELY AFTER THIS ROUTINE IN THE WILLIAMS MEMORY.

To punch the integer with Key Word at m_A, enter this routine with the instructions

	-- m_AF
p	50 pF
p+1	26 xF

where x is the location of this routine.

The user must be sure there are enough spare memory locations following the integer A so as not to overwrite other information.

The time required to output an integer is roughly $1.29(n_A+1)^2 + 1.21(n_A) + 176[n_A(1.06)^{n_A+1} +1]$ msec.

NUMBER TRANSFER ROUTINE:

11 words, no temporary storage.

This subroutine transfers a multiple-precision integer from one part of the Williams Memory to another. The Key Word is corrected so that the new value of m_A+1 is in the right-hand address position.

To move the number with Key Word at m_A to the set of locations beginning at m_B, enter this routine with m_B, the final address, in the right-hand address position of the accumulator, and with the instructions.

	-- m_AF
p	50 pF
	26 xF
p+1	-- --

m_A = initial address of number

x = location of this routine.

The time required is $(2.07 + 0.58n_A)$ milliseconds.

NOTE: Although no symbolic addresses are used in any subroutine, for the user's convenience the library copies of these routines are identified by the following symbolic addresses:

Addition	(ADD)
Multiplication	(MULT)
Shifting	(SHIFT)
Output	(OUT)
Small-Integer Print	(PR)
Number Transfer	(MOVE).

DATE May 17, 1961
PROGRAMMED BY John Ehrman
APPROVED BY J. Snyder

nj

LOCATION	ORDER		NOTES PAGE 1 A 8
	00K(ADD)		
0	40 F		Save Key B
	K5 F		
1	42 9L		
	46 3L		
2	L4 5L		
	42 47L		Plant link
3	L5 [ ]F	by 1'	get Key A
	42 11L		Plant $m_A + 1$
4	10 20F		
	40 1F		
5	L1 1F		
	40 1F		$-(n_A + 1)$ at 1
6	L5 F		
	42 12L		plant $m_B + 1$
7	10 20F		
	40 F		
8	L1 F		
	40 F		$-(n_B + 1)$ at 0
9	41 48L		Clear carry box
	L5 [ ]F	by 1	get m_C
10	42 46L		plant Key C store
	L4 5L		address and to store address
11	42 16L		
	L5 [ ]F	← 24'; by 3', 17'	A_i
12	L4 48L		+ carry
	L4 [ ]F	by 19; 6'	$+ b_i$
13	40 2F		temp store
	32 37L		
14	L4 49L		correct overflowed
	40 2F		sum
15	L5 5L		and plant carry bit
	42 48L	← 37'	
16	L5 2F		
	40 [ ]F	by 11	Store C_i

LOCATION	ORDER		NOTES PAGE 2 A 8
17	F5 11L		
	42 11L		Step A_i address
18	42 28L		
	F5 12L		Step B_i address
19	42 12L		
	F5 16L		Step C_i address
20	42 16L		
	42 32L		Plant store addreses
21	42 42L		
	F5 1F		Count on A
22	40 1F		
	36 25L		
23	F5 F		not done A; count
	40 F		on B
24	36 28L		
	22 11L		
25	F5 F	← 22'	A done;
	32 38L		if ≥ 0, both A and B done
26	40 1F		B not done; save B count
	L5 12L		at 1, get B_i pickup
27	42 28L		address
	41 F		Clear 0
28	L5 48L	← 37	carry + A_i or B_i
	L4 [ ]F		
29	40 2F		
	36 38L		
30	L4 49L		Correct overflowed sum
	40 2F		store temporarily
31	L5 5L		set carry to 1
	42 48L	← 38	
32	L5 2F		
	40 [ ]F		Store C_i
33	F5 28L		
	42 28L		Step pickup address
34	F5 32L		
	42 32L		Step store addresses

LOCATION	ORDER		NOTES PAGE 3 A 8
35	42 42L		
	F5 1F		Count at 1
36	40 1F		
	32 38L		if ≥ 0, done; ∴ 1 is clear
37	26 28L		
	23 15L	← 13'	
38	23 31L	← 30'	
	41 F	← 25', 36'	Clear 0
39	L5 42L		
	42 F		loc 0 = $m_C + n_C + \begin{cases} 2 \text{ if no carry} \\ 1 \text{ if a carry} \end{cases}$
40	F5 46L		
	42 1F		loc 1 = $m_C + 1$
41	L3 48L		
	36 44L		
42	L5 48L		
	40 [ ]F		Store carry as C_n
43	L1 1F		
	22 44L		
44	F1 1F	← 41'	$\begin{Bmatrix} -m_C-1 \\ -m_C-2 \end{Bmatrix} + \begin{Bmatrix} m_C + n_C + 2 \\ m_C + n_C + 3 \end{Bmatrix} = n_C + 1$
	F4 F		
45	50 49L		Clear Q
	00 20F		Form Key C
46	L4 1F		
	40 [ ]F		
47	50 1F		$m_C + 1$ in Q
	26 [ ]F		exit via link
48	00 F		
	00 1F		carry storage
49	80 F		
	00 F		Overflow correction bit

LOCATION	ORDER		NOTES
	00K(MULT)		
0	40 F		Store Key A
	K5 [ ]F		R.H.A. = loc. of lowest C_k
1	42 6L		
	46 3L		Plant pickup addresses
2	L4 11L		
	42 49L		Plant link
3	L5 [ ]F		Get Key B
	46 50L		Plant $n_B + 1$
4	50 53L		Clear Q
	00 20F		position $m_B + 1$
5	46 19L		plant B_0 location
	L4 50L		
6	46 50L		end const = $m_B + n_B + 2$
	L5 [ ]F		get m_C
7	42 L		Plant C lowest,
	42 13L		Initialize clear loop
8	42 48L		Plant Key C store
	L5 F		
9	10 20F		
	L4 53L		$-1 + (n_A + 1) \times 2^{-39}$
10	40 F		Save A count
	F4 11L		
11	40 1F		loc 1 = NO 1F 00($n_A + 3$) F
	00 1F		
12	S5 [ ]F		Save A_0 address at
	46 12L		LHA
13	46 18L		Plant starting address
	41 [ ]F	← 17	Clear loop
14	F5 13L		NO 1F 00(n_A + j)F
	42 13L		-40 1F 00 1F
15	L5 1F		= 80 F 00 (n_A + j - 1)F
	L0 11L		
16	42 1F		
	32 35L		jump to initialize
17	22 13L		loop
	41 54L	← 40	Clear carry box

LOCATION	ORDER		NOTES PAGE 5 A 8
18	50 [ ]F		A_1
	L5 [ ]F		$+(C_k \times 2^{-39})$
19	74 [ ]F		$\times B_j$
	L4 [ ]F		$+ C_{k+1}$
20	L4 54L		+ carry
	32 40L		
21	L4 53L		Correct for overflow
	40 2F		
22	L5 11L		
	42 54L	← 41	Set carry digit
23	L5 2F		
	40 [ ]F		Store current MSP = C_{k+1}
24	S5 F		
	40 [ ]F		Store current LSP = C_k
25	F5 18L		Advance addresses
	42 18L		
26	42 24L		
	42 50L		
27	L4 11L		
	46 18L		
28	42 19L		
	42 23L		
29	42 31L		
	F5 1F		Count on A
30	40 1F		
	36 18L		Loop if not done on A
31	L5 54L		
	40 [ ]F		Store carry appropriately
32	L5 19L		
	L4 11L		Advance $B_j \rightarrow B_{j+1}$
33	46 19L		
	L0 50L		Test for end
34	32 41L		
	L5 12L		reset A_0 address
35	46 18L		
	L1 F	← 16'	Reset A count

LOCATION	ORDER		NOTES PAGE 6 A 8
36	40 1F		
	F5 L		Reset C addresses
37	42 L		
	42 18L		
38	42 24L		
	L4 11L		
39	42 19L		
	42 23L		
40	22 17L		Loop with carry clear
	40 2F	← 20'	Save non-overflowed sum
41	23 22L		jump to clear carry
	41 2F	← 34	Begin setup for Key C
42	41 1F		
	L5 31L		
43	42 2L		Loc 2 = $m_C + n_C + x$
	F5 48L		
44	42 1L		Loc 1 = $m_C + 1$
	L3 54L		Any carry at end?
45	32 50L		
	L5 54L	← 51'	yes; complete Key C
46	L4 2F	← 52'	
	L0 1F		
47	50 53L		Clear Q
	00 20F		
48	L4 1F		
	40 [F]F		Store Key C
49	50 1F		$m_C + 1$ in Q
	26 [F]F		exit via link
50	S4 [F]F		B loop end constant
	L3 [F]F	← 45	Test last MSP = 0?
51	36 52L	→ 52	
	22 45L		
52	F1 54L	← 51	Yes; back up n_C by 1
	26 46L		
53	80 F		
	00 F		Overflow correction bit
54	00 F		
	00 1F		Carry storage

LOCATION	ORDER		NOTES PAGE 7 A 8
	00K(SHIFT)		
0	40 F		Save S
	K5 [ ]F		RHA = lowest - order "digit" location
1	42 37L		
	46 37L		Plant link and key correction
2	46 5L		
	40 1F		Save for left-right test
3	L3 F		
	32 37L		If S = 0, exit immediately
4	L1 F		
	36 79L		If S < 0, reverse S and parameter
5	L5 [ ]F		Get Key word
	42 L		
6	42 22L		
	42 28L		
7	40 2F		
	43 2F		Save count at 2
8	51 F		
	00 1F		
9	66 42L		Compute S = 39N + R
	10 1F		
10	40 F		R at 0, N in Q
	42 21L		
11	L5 1F		Test L - R parameter
	36 43L		
12	L5 L		RIGHT SHIFT
	S4 F		
13	42 20L		
	00 20F		
14	46 28L		
	L4 23L		
15	46 20L		
	L1 2F		
16	10 20F		
	S4 F		Store N - C at 1
17	40 1F		If
	36 33L		N ≥ C, all is lost

LOCATION	ORDER		NOTES PAGE 8 A 8
18	L3 F		If
	36 28L		R = 0, no shift
19	F5 1F	← 27'	Last time thru loop,
	32 20L		put 0 in A
20	L5 [ ]F		
	50 [ ]F		Shift a word
21	22 21L		Waste
	10 [ ]F		
22	S5 F		
	40 [ ]F		
23	F5 1F		
	40 1F		Advance count
24	32 34L		
	F5 22L		Advance addresses
25	42 22L		
	F5 20L		
26	42 20L		
	L4 23L		
27	46 20L		
	26 19L		Loop
28	L5 [ ]F	← 18',32'	Transfer directly if R = 0
	40 [ ]F		
29	F5 1F		
	40 1F		
30	32 33L		
	F5 28L		
31	42 28L		
	L4 29L		
32	46 28L		
	26 28L		
33	23 28L	← 18'	N $\geq$ C, store a 0
	F5 28L	← 30	
34	26 36L		
	S3 [ ]F	← 24	Test last MSP for 0
35	36 38L		
	F5 22L	← 41',40'	

LOCATION	ORDER		NOTES PAGE 9 A 8
36	L0 L	← 34	
	00 20F		
37	46 [ ]F		Correct count in Key word
	22 [ ]F		and exit via link
38	L5 22L	← 35	See if last MSP was
	L0 L		also first MSP:
39	10 10F		whether we started
	S3 F		with only one word
40	32 35L		If ≥ 0, had only 1
	F5 L		had more than 1 word so
41	42 L		back up count by 1
	22 35L		
42	00 F		
	00 39F		39×2^{-39}
43	L5 5L		LEFT SHIFT
	L4 2F		
44	46 64L		
	46 74L		
45	L5 F		
	00 20F		
46	46 59L		
	46 65L		
47	L1 2F		
	10 20F		
48	40 1F		-C at 1
	S1 F		
49	40 2F		-N at 2
	L5 L		Set addresses
50	S4 F		
	L0 1F		
51	42 61L		
	L0 55L		
52	42 66L		
	42 34L		
53	42 74L		
	K0 F		

LOCATION	ORDER		NOTES PAGE 10 A 8
54	42 64L		
	L4 55L		
55	42 58L		
	L4 1F		
56	S4 F		
	42 81L		
57	L3 F		
	36 74L		If R = 0, jump to transfer
58	32 58L		waste
	51 [ ]F		Test for overflow on MSP
59	00 [ ]F		
	F0 86L		
60	36 63L		
	F4 86L		Correct the overflow
61	22 61L		waste
	40 [ ]F		Store properly
62	L5 61L		
	42 34L		Advance MSP address
63	F5 1F	← 60,72	
	32 72L		Test for last time through loop
64	L5 [ ]F	← 74'	
	50 [ ]F		Shift a word
65	00 [ ]F		
	32 66L		
66	L4 86L		Correct overflow
	40 [ ]F		
67	F5 1F		
	40 1F		Count
68	32 80L		
	L5 66L		Back up all addresses
69	L0 67L		By 1
	42 66L		
70	L5 64L		
	LQ 67L		
71	42 64L		
	46 64L		

LOCATION	ORDER		NOTES PAGE 11 A 8
72	26 63L		
	L5 60L	63'	Set to put 0's in Q
73	42 64L		on last time thru shift loop
	26 64L		
74	L5 [ ]F		Transfer Bodily
	40 [ ]F		
75	F5 1F		
	40 1F		Count
76	32 80L		
	L5 74L		Back up addresses
77	L0 75L		
	42 74L		
78	46 74L		
	26 74L		
79	40 F		Reverse S and
	L1 1F		Left-right parameter
80	22 2L		
	L5 2F ←	84' 68,76	Any location to be cleared?
81	36 85L		
	41 [ ]F		Clear a location
82	F5 2F		
	40 2F		Count
83	L5 81L		
	L0 79L		
84	42 81L		
	22 80L		
85	F5 34L		Get MSP address for exit
	26 36L		
86	80 F		
	00 F		Correction bit

LOCATION	ORDER		NOTES PAGE 12 A 8
	00K(OUT)		
0	K5 F		
	42 39L		Plant link
1	10 20F		
	42 2L		
2	42 34L		Plant end constant
	L5 F		get Key
3	10 20F		Construct most significant
	L4 34L		digit address
4	42 5L		
	L5 34L		
5	22 20L		Enter loop
	00 [ ]F		RHA = M.S. digit address
6	41 $[R_k]$F ←	28	
	L5 $[D_j]$F ←	6'	
7	10 1F		
	01 1F		Save last bit
8	40 F		
	50 $[D_j]$F		
9	L5 F		
	66 68L		$\div 10^{11} \times 2^{-39}$
10	L4 F		
	L0 68L		Test for $\geq 10^{11}$
11	36 13L		
	L4 68L		
12	10 1F		
	00 1F		Drop division roundoff bit
13	40 $[R_k]$F		Store partial remainder
	S5 F		
14	L0 58L		Test for quotient = 0
	36 32L		
15	L4 58L		
	40 $[D_{j+1}]$F		Store quotient
16	43 58L		block zero test
	L5 8L		
17	42 15L		
	L0 12L		

LOCATION	ORDER		NOTES PAGE 13 A 8
18	42 6L		
	42 8L		
19	F0 31L		
	32 6L		
20	F5 31L		
	42 31L	← 5	Advance end constant
21	00 20F		
	46 6L		Reset addresses
22	46 9L		
	46 13L		
23	L5 7L		Restore zero test
	42 58L		
24	L5 5L		
	42 6L		
25	42 8L		
	F5 8L		
26	42 5L		Advance M.S.D. address
	42 15L		
27	L0 12L		
	L0 31L		End test
28	36 6L		
	L4 31L		
29	42 35L		Plant address for print
	50 37L		
30	22 35L		
	00 F		
31	2L 4095F		
	50 [ ]F		End constant
32	L5 5L		Have zero quotient;
	L0 7L		back up MSD address
33	42 5L		by one
	26 16L		
34	40 35L		
	L4 [ ]F		Print end constant
35	50 36L		
	L5 [ ]F		get "digit", base 10^{11}

LOCATION	ORDER		NOTES PAGE 14 A 8
36	26 40L		jump to print
	L5 35L		
37	L0 7L		
	42 35L		Step pickup address
38	L0 34L		End test
	36 35L		
39	92 5F		Punch 2 1-hole delays
	22 [ ]F		Exit via link

Print Routine (PR) must be attached immediately after the last word of this routine.

LOCATION	ORDER		NOTES
	00K(PR)		
0	40 1F		
	K5 F		
1	42 12L		Plant link
	00 25F		
2	36 24L		Jump to initialize
	F1 20L	← 27'	
3	40 2F		Set digit count to -11
	L5 1F		
4	50 2L		Roundoff to Q
	66 28L		$\div 10^{11}$
5	75 20L	← 13	$x\ 10\ x\ 2^{-39}$
	L0 18L		Test for zero
6	32 11L		
	L4 18L		Restore bit
7	00 36F		Position digit,
	40 F		Store temporarily
8	43 18L		Block zero test
	F5 19L		
9	40 19L	← 16'	Test for space
	32 13L		
10	L5 F		
	82 4F		Punch digit
11	10 40F		reposition fractional part
	F5 2F		

LOCATION	ORDER		NOTES PAGE 15 A 8
12	40 2F		Count digits
	32 [ ]F	by 1	link
13	26 5L		Loop
	F5 17L	← 9'	count spaces
14	32 20L		if ≥ 0, do LFCR
	40 17L		
15	00 3F		
	93 963F		punch space
16	L1 20L		Reset space count
	26 9L		
17	00 F		
	00 F		Space count for LFCR
18	80 F		
	00 [1]F	by 8,24'	Test for zero digits
19	00 F		
	00 F		Digit count for space
20	00 F		
	00 10F	← 14	10×2^{-39}
21	92 131F		LFCR
	92 519F		2 delays
22	L1 20L		
	10 1F		Reset space count
23	40 17L		
	26 16L		re-enter loop
24	L5 3L	← 2	Initialize:
	42 18L		Plant test bit
25	F1 20L		
	40 19L		Set space count
26	L1 20L		
	10 1F		
27	40 17L		Set LFCR count
	22 2L		
28	17 1159F		
	6F 2048F		$10^{11} \times 2^{-39}$

LOCATION	ORDER		NOTES
	00K(MOVE)		
0	42 5L		
	42 9L		Plant final address n
1	K5 1023F		
	42 10L		Plant link
2	46 3L		Plant Key pickup
	46 5L		initialize work pickup
3	L4 [ ]F	by 2	get Key word
	L0 1L		construct end constant
4	46 10L		
	26 5L		
5	L5 1024[ ]F		
	44 [ ]F		Move a word
6	F5 5L		
	42 5L		Advance store address
7	L0 1L		
	46 5L		Advance pickup address
8	L0 10L		
	36 5L		End test
9	F5 9L		Plant correct address
	42 [ ]F	by 0	in Key word
10	J0 [$n+n_A+2$]F	by 4	end constant
	22 [ ]F	by 1	exit via link