UNIVERSITY OF ILLINOIS

DIGITAL COMPUTER

LIBRARY ROUTINE A7 - 244

By Roger Farrell

TITLE	1.7 Precision Floating Binary Arithmetic and Double Precision Fixed Point Arithmetic With Floating (DOI or SADOI) Decimal Conversion.
TYPE	Interpretive routine entered like a closed subroutine.
NUMBER OF WORDS	610
TEMPORARY STORAGE	Locations 0, 1, 2 and 6 locations specified by a preset parameter S3, S3-5S3.
DURATION	See the order code.
ACCURACY	Floating point numbers are rounded to 68 binary places (together with a sign). Fixed point numbers are carried to a full 78 binary places (together with a sign). Print out is rounded.
PARAMETERS	S3; during input of the program location 3 must contain $t \times 2^{-39}$ where t is the location of the first word of temporary storage.

DESCRIPTION This routine was written as a flexible general purpose double precision routine. It is suitable for problems requiring twelve to twenty-three decimal places of accuracy which do not require a large amount of computing time. As an example of problem sizes inversion of a 30x30 matrix takes approximately 40 minutes using an interpretive matrix inversion routine.

The method of interpretation and the interpretive order code are taken almost directly from ILLIAC library routine A1. It is suggested that a potential user of this routine who has not used either A1 or this routine should also read the description for routine A1. Coding methods and examples are discussed there.

Because this routine has some special features it is longer than a similar routine, ILLIAC library routine A4. It is slower or approximately the same speed as routine A4 in its arithmetic operation. The special features referred to are

(1) a closed subroutine for standardizing numbers which is very fast

(2) closed subroutinesfor multiplication, division, sign change, read a fixed point fraction or integer from tape, print an integer. These subroutines can be used by auxiliary subroutines.

(3) Fast input which reads numbers at maximum reader speed.

(4) Ability to read numbers punched in the form sign, integer part, decimal point, fractional part.

(5) Dual fixed point - floating point operations with changes from one mode to the other mode by execution of interpretive instructions.

(6) Interpretive instructions for adding and subtracting B-registers.

(7) 4096 addressable numbers (pairs of memory locations some of which are on the drum).

(8) In floating point division the maximum error is less than .52 in the least significant quotient bit. If the divisor should divide exactly, the quotient is exact.

NUMBER STORAGE, METHODS OF NUMBER REPRESENTATION

Locations S3, 1S3, 2S3 will be called the floating accumulator ; locations 3S3, 4S3, and 5S3 will be called the number register. In preparation for each arithmetic operation an operand is placed into the number register from the memory. The arithmetic result is placed in the floating accumulator. Numbers are not brought from the memory before the execution of non-arithmetic instructions.

During floating point operation a number is stored in the memory in a packed form requiring two consecutive memory locations n, n+1. It has the following standard form

n	$\longleftarrow x_1 \longrightarrow$	n+1	0	$\longleftarrow x_2 \longrightarrow$	E_x+512

and $x = (x_1 + 2^{-39} x_2) \cdot 2^{E_x}$. In the standard form either

$$x_1 = x_2 = 0 \text{ and } E_x = -512$$

or one of $-1 \leq x_1 + 2^{-39} x_2 < -1/2$

$$1/2 \leq x_1 + 2^{-39} x_2 < 1$$

with $-512 < E_x < 512$. Thus, of 80 digital positions, the exponent uses 10, the positive sign of the least significant part $L(x_1 + 2^{-39} x_2)$ uses 1, and the fractional part $x_1 + 2^{-39} x_2$ uses 69 digital position.

During fixed point operation numbers are stored in an unpacked form in locations n, n+1, in the following way.

n	← x_1 →	n+1	0	x_2

where $x = x_1 + 2^{-39} x_2$ and $-1 \leq x < 1$.

When a number is brought from the memory and is put into the number register, it has the following form:

$$N(3S3) = x_1$$

$$N(4S3) = x_2$$

$$N(5S3) = 2^{-39}(E_x + 512) \qquad \text{floating point}$$

$$= 0 \qquad \text{fixed point}$$

When a floating point number is unpacked the ten digital positions of 4S3 which hold the exponent are cleared to zero.

Arithmetic floating point results in the floating accumulator have the form

$$2[N(S3) + 2^{-39} N(1S3)] \cdot 2^{[2^{39} N(2S3) - 512]} = \text{answer}.$$

Results in the floating accumulator are not standardized after arithmetic. Standardization may be accomplished by use of the interpretive N2 F instruction.

Arithmetic fixed point results in the floating accumulator have the form

$$2[N(S3) + 2^{-39} N(1S3)] \, 2^{2^{39} [N(2S3)]} = \text{answer}$$

In this case, $N(2S3) \geq 0$ and $N(2S3) > 0$ results from the automatic right shift of overflowed numbers resulting from addition, subtraction and multiplication. To perform a fixed point division using this routine the programmer must guarantee division overflow will not occur. The division subroutine further requires the divisor y to satisfy

$$|y| \geq 1/2 .$$

NUMBER STORAGE, ADDRESSING OF LOCATIONS

The 1024 locations in the Williams Memory together with additional locations on the drum are directly addressable using a 12 digit address carried by an interpretive instruction together with use, if desired, of 12 digit b-modifiers. A 12 digit address plus a 12 digit modifier can be used to construct any address in the range $0 \leq$ address ≤ 8191. The correspondence between interpretive addresses and ILLIAC addresses are

Interpretive	0 - 1023	Williams Memory 0 - 1023
Interpretive	1024 - 8191	Drum 2560 - 9727

As an example, if the interpretive address (unmodified) is 1023, then the most significant part is at Williams Memory location 1023. The least significant part is at drum location 2560.

INTERPRETIVE INSTRUCTIONS

The interpretive instructions have the same form as the regular ILLIAC instructions. This consists of two function digits (two sexadecimal characters) followed by a single 12-bit address.

The first sexadecimal function digit designates a b register to be used in executing the instruction. The correspondence is as follows:

A first function digit of 0, 1, ..., 7 refers to b-registers 0, 1, ..., 7 respectively.

A first function digit of 9, K, S, ..., L refers to b-registers 1, 2, 3, ..., 7.

A first function digit of 8 does not refer to a b-register.

The second sexadecimal function digit indicates the type of operation to be performed. This is discussed in detail below.

The actual structure of the b-registers will be described in the sequel. However, each b-register contains two parts, an address-register and a count register. The former contains 12 binary stages while the latter contains 20 binary stages.

Let F denote the number in the floating accumulator and F(n) the number in locations n, n+1. Let $n_0, n_1, \ldots, n_7$ denote the addresses and $c_0, c_1, \ldots, c_7$ the counts in b registers 0, 1, ..., 7 respectively. In the sequel numbers $F(n+n_b)$ will be indicated using the following conventions:

$$n_8 = 0 \quad \text{and} \quad n + n_b \text{ is taken mod } 2^{13} .$$

Interpretive instructions may be stored only in the Williams Memory.

b0 n	$0 \leq b \leq 8$. Form F - $F(n + n_b)$
b1 n	$0 \leq b \leq 8$ Replace F by - $F(n + n_b)$
b2 n	$0 \leq b \leq 7$ Replace c_b by $c_b + 1$, n_b by $n_b + 2$ and jump to the interpretive instruction at the right side of location n ($0 \leq n \leq 1023$) if $c_b + 1 \geq 0$. If $c_b + 1 < 0$ take the next interpretive instruction.
82 n	Test the sign of the floating accumulator. If $F \geq 0$ jump to the interpretive instruction at the right side of location n ($0 \leq n \leq 1023$). If $F < 0$ take the next interpretive instruction.
92 n	Jump to the interpretive instruction at the right side of location n. ($0 \leq n \leq 1023$)
K2 0	Change the mode of operation from floating point to fixed point.
S2 0	Change the mode of operation from fixed point to floating point.
N2 0	Standardize the floating accumulator.

Instructions 92, K2, S2, N2, 93, K3, S3, N3 do not effect any change in the b-registers but they do make a reference. These variants might be used before the 8L instruction.

b3 n	$0 \leq b \leq 9$. Same as corresponding b2 n instruction except that the jump is to the left side of the location.
b3 0	b = K, S, N. Same as corresponding b2 0 instructions.
b4 n	$0 \leq b \leq 8$. Form $F + F(n + n_b)$
b5 n	$0 \leq b \leq 8$. Replace F by $F(n + n_b)$
b6 n	$0 \leq b \leq 8$. Form $F / F(n + n_b)$
b7 n	$0 \leq b \leq 8$. Form $F \times F(n + n_b)$
88 0	Read one number from the input tape and put it in the floating accumulator. See the description below of the punching of data tapes. Numbers on input are converted to floating binary numbers and will not be standardized unless they are stored. Floating decimal numbers may have 24 or less decimal digits in the fractional part.
89 n	$1 \leq n \leq 22$. Punch or print F as a sign, one space, n decimal digits, one space, sign of exponent, and the exponent to 3 decimal digits with zero suppression on the first two digits. F is converted before punching or printing to a floating decimal number. The floating accumulator is not standardized. This instruction destroys the contents of the floating accumulator.
	Format may be controlled by use of the 8F n and 9F n instructions which control the punching of CR-LF characters and spaces between groups of decimal digits.

The b0, b1, b4, b5, b6, b7 instructions with b = 9, K, S, ..., L do the same thing as the corresponding instructions with b = 1, 2, 3, ..., 7.

The b8 and b9 instructions with $b \neq 8$ do the same thing as the 88, 89 instructions but in addition make a reference to a b-register.

bK n $0 \leq b \leq 7$. Set $n_b = 0$ and $c_b = -n$.

bK n $b = 8, 9, \ldots, L$.
Used to place an integer in F. If $n < 512$ set $F = n$ in standardized form. If $n \geq 512$ set $F = n-1024$ (standardized).

bS n $0 \leq b \leq 8$ Replace $F(n+n_b)$ by F;
The number F is standardized before being stored at location $n + n_b$. The contents of the floating accumulator are not changed.

When the binary exponent E_x of F becomes too large or too small the following special conventions apply. $E_x \leq -512$. The number stored at location $n + n_b$ has fractional part = 0, exponent = -512. The machine representation of E_x is $E_x + 512$ so that underflow results in two ILLIAC zeros being stored.

$E_x \geq 512$. ILLIAC stops on FF 039. This stop occurs only during the execution of the bS n instruction.

bN n $0 \leq b \leq 8$ Replace F by $|F| - |F(n + n_b)|$

8J n Leave the interpretive mode of operation and jump to the ordinary ILLIAC instruction on the left side of location n. To re-enter the interpretive mode of operation after an 8J n instruction one may
(1) use a normal subroutine entry to A7
(2) jump to the left instruction at 17 L of A7. In this case the next interpretive instruction is the instruction following the 8J n instruction.
(3) use other special purpose entries described below.

bJ n $0 \leq b \leq 7$. Let m be the integer whose digits are the least significant 12 binary digits of the number at Williams Memory location n. Replace n_b by $n_b - m \pmod{2^{12}}$ and leave c_b unchanged. This instruction is especially useful when n is the address of a b-register because this instruction then differences 2 b-registers.

8F n If $n > 0$, punch or print one CR-LF now; one CR-LF and one 2-hole delay before the next 89 instruction; an CR-LF and delay before every nth such 89 instruction thereafter. For example if $n = 2$ a CR-LF and delay are punched before the first, third, fifth, ... 89 n instruction executed.

If n = 0, one CR-LF character is punched or printed but otherwise the instruction acts as if n = 4096.

After execution of each 89 n instruction a 5 hole delay and two spaces are punched or printed unless a CR-LF will precede the next 89 n instruction.

9F n	During the punching or printing of all succeeding numbers, punch or print one space after each n digits of the fractional part. If n = 0, or, if this instruction is never executed, no spacing occurs.
bF n	$0 \leq b \leq 7$. Same as bJ instruction except that $n_b + m \pmod{2^{12}}$ is formed. Does not change c_b.
bL n	$0 \leq b \leq 7$. Replace n_b by $n_b + n \pmod{2^{12}}$. Does not change c_b.
bL n	b = 9, K, ..., L. Replace n_1, n_2, ..., n_7 respectively by $n_1 - n$, $n_2 - n$, ... $n_7 - n \pmod{2^{12}}$. Does not change c_b.
8L n	If b is the number of the last b-register referred to, replace n_b by n. Does not change c_b.

OPERATION TIMES The following table of approximate operating times is offered as a guide to the programmer. The time for each instruction involves three component times, interpretation, memory access, and time of execution. Further, the drum acts as a slow access memory for which only a random access time can be given.

The time for interpretation of the digital combinations in an instruction is so nearly the same for b = 8 vs b ≠ 8 that no distinction is made between the two types of instructions.

Interpretation:	1.6 ms per instruction	
Memory access before instructions:	b0, b1, b4, b5, b6, b7, bN	
	Williams Memory	2.2 ms
	Drum	14.4 ms random access.

Times of execution

b0		3.4 ms	
b1		1.4 ms	
b2,3	$0 \leq b \leq 7$	1.5 ms	
82,3		1.0 ms	
92,3		1.0 ms	
b4		2.5 ms	(generally)
		1.0 ms	for F + 0
		1.8 ms	for $0 + F(n + n_b)$
b5		.9 ms	
b6		7.9 ms	
b7		4.9 ms	
88		3.5 ms per character + 30 ms per number	
89	n	Minimum time 176 + 16n ms. If 9F instruction is used, add 16 ms for each space which is punched. For example, a 20 digit number with spaces after every 3 digits, 176 + 16 x 20 + 16 x 6 = 592 ms.	
bK	$0 \leq b \leq 7$	.7 ms	
8K	n n=0	.7 ms	
8K	n n≠0	3.7 ms	
bS		Williams Memory 5.2 + .0185(3k + 24 [k/8]) ms where k is the number of shifts to standardize and [k/8] the greatest integer $\leq$ k/8.	
		Drum Memory 17.0 + .0185(3k + 24 [k/8]) ms random access.	
bJ	$0 \leq b \leq 7$	1.3 ms	
8J		.2 ms	
bF	$0 \leq b \leq 7$	1.3 ms	
bL	$0 \leq b \leq 7$	1.0 ms	

EXAMPLE OF TIME ESTIMATION

The following is a loop which can be used to compute the inner product of a sequence of numbers stored in the Williams Memory with a sequence of numbers stored on the drum

0.	0K	nF
	8K	F
1.	8S	4F
	05	2000F
2.	07	900F
	84	4F
3.	8S	4F
	02	1L

In this loop 5 instructions are interpreted, requiring 5 x 1.6 = 8.0 ms. Two Williams Memory accesses require 4.4 ms. Execution times are

.9 + 4.9 + 2.5 + 5.2 + 1.5 = 15.0

Total time exclusive of the drum access,

8.0 + 4.4 + 15.0 = 27.4 ms.

Consequently the loop time is the time of 2 drum revolutions and 5 sector times or about 35 ms.

ENTRIES TO A7

The closed subroutine entry to A7 is

p	50	pF
p + 1	26	qF

where q is the location of the first word of A7. The first interpretive instruction must be contained in the right half of location p+1.

After leaving the interpretive mode by use of the 8J instruction, A7 may be re-entered in several ways. A closed subroutine entry may be used. A jump to the left side of 17L (relative to the first word of A7) results in the interpretive instruction following the 8J instruction being interpreted next. A jump to the left side of 2L will cause the 8J instruction to be executed again. A jump to the left side of 90 L causes standardization of the floating accumulator, then proceding to the next interpretive instruction. This assumes the floating accumulator is not overflowed. A jump to the right side of 139 L causes a test and correction of the floating accumulator for overflow. The next interpretive instruction is then taken. To use this entry Q must agree with 1S3. A jump to the left side of 154 L clears the floating accumulator. The next interpretive instruction is taken.

B-REGISTERS The b-registers are stored in eight consecutive locations 38L, 39L, ..., 45L of A7. If we denote a word by a_0, a_1, ... a_{39} then a_0 ... a_{19} constitute the count c_b carried by the b-register. $a_{20}...a_{27}$ are always zero. $a_{28}...a_{39}$ constitute the address n_b carried by the b register.

The programmer should remember that storage of a number requires <u>two</u> locations. Consequently the b2 n and b3 n instructions, the count-jump instructions, form $n_b + 2$.

Instructions of the form b_1F $38 + b_2L$ will cause the addition $n_{b_1} + n_{b_2} = n'_{b_1}$. Instructions of the form b_1J $38 + b_2L$ will cause the subtraction $n_{b_1} - n_{b_2} = n'_{b_1}$.

PUNCHING DATA TAPES FOR INPUT

Numbers may be punched on data tapes as floating decimal numbers or as a sign, integer part, fractional part. In either case, the number when read is converted to a floating binary number which is put in the floating accumulator. It will not in general be standardized. No special instructions or markings are necessary to distinguish the two cases. In the fixed point mode, numbers with zero exponents are input to full double precision accuracy. The routine determines the punching format from the 5-hole characters read: each number on tape must be terminated by a non-space character.

<u>Floating decimal numbers having 12</u> or less decimal digits in the fractional part may be punched as follows

1. Sign of number
2. decimal digits of the absolute value of the fractional part
3. sign of the exponent

4. decimal digits of the absolute value of the exponent. Non significant zeros on the left may be omitted. For example 052 or 52.
5. Any 5-hole character other than a space causes reading of exponent digits to end.

The input routine is sensitive to all 5 hole characters except spaces. Spaces may be used for punching out errors. So long as the number of decimal digits in the fractional part is 12 or less, one 5-hole character other than a space or decimal point may be used anywhere between the 1st decimal digit of the fractional part and the sign of the exponent. Consequently output tapes by library routine A1 may be read by routine A7, but not conversely.

Floating decimal numbers having 13 to 24 decimal digits in the fractional part must be punched as

1. Sign of the fractional part.
2. k_1 digits of the fractional part, $k_1 \leq 12$, a 5 hole character other than a space or decimal point, k_2 digits of the fractional part, $k_2 \leq 12$.

3-5. As for numbers with less than 13 digits in the fractional part.

Spaces may be used, again, for format or error correction. When 12 or more decimal digits occur in the fractional part, there must not be any other 5-hole characters used except as described above.

Numbers in the integer-fraction representation are punched as

1. sign of the number
2. integer part of the absolute value
3. decimal point
4. fractional part of the absolute value
5. any 5-hole character other than a space.

If either the integer or fractional part is zero it need not be punched. Non-significant zeros may be omitted to the left of the decimal point, trailing non-significant zeros to the right of the decimal point may be omitted. Hence

0 = + . = - . = +.0

1 = +1.

1/10 = + .1

The integer part can have 12 or less decimal digits, the fractional part 12 or less decimal digits.

The integer-fraction representation was included as a convenience for the manual preparation of data tapes. The routine has no corresponding output facility. Further, where the greatest possible accuracy is desired for the floating binary number put into the floating accumulator, the floating decimal representation should be used on data tapes. During reading of a integer-fraction, the integer part is read and stored in S3, the fractional part is read and stored at 1S3. The exponent 39 is assigned. If for example the number is +.1, this is clearly less accurate than computing 1/10 by a double precision division, the process used when reading floating decimal numbers.

The output of the routine can, of course, be read by the routine, and hence conforms to the above rules. A punching error on a tape which causes the above rules on format to be violated results in an FF 03K stop.

The following descriptions are primarily intended for those who wish to write fast auxiliary subroutines.

STANDARDIZATION SUBROUTINE

The subroutine is entered using a pair of instructions

p	F5 pF
p + 1	26 284L

where the address of the jump is relative to the 1st word of A7. The most significant part of the double precision number should be in 3S3, and the least significant part in Q. When the routine is left, the standardized number is in AQ. The number of shifts used in standardizing the number is in 5S3. If the number was zero, 5S3 is set to zero. Except for zero the standardized number satisfies the inequalities $1/2 \leq x < 1$ or $-1 \leq x < -1/2$. Temporary storage, 3S3, 4S3, 5S3. Operating time 1.6 + .0185 (3k + 24[k/8]) ms. [k/8] denotes the "greatest integer less than".

MULTIPLICATION SUBROUTINE

This subroutine multiplies the double precision fixed point number in S3, 1S3 by the double precision fixed point number at n, n+1. It is entered by the instructions

p	50	n+1 F		F5	pF
p + 1	26	230L			

When the subroutine is left, the rounded double precision product is in S3, 1S3. The least significant part is also in Q. Temporary storage, location 1.

If the two numbers are $x_1 + 2^{-39} x_2$ and $y_1 + 2^{-39} y_2$ the "product" which is formed is

$$x_1 y_1 + 2^{-39} (x_1 y_2 + x_2 y_1 + 3/4 \times 2^{-39}) .$$

This product is truncated to a double precision number. The rounding factor $3/4 \times 2^{-39}$ is justified on the basis of 1/2 + expected $(x_2 y_2) = 1/2 + 1/4 = 3/4$.

DIVISION SUBROUTINE

This routine divides the double precision fixed point number in S3, 1S3 by the double precision fixed point number in 3S3, Q. It is entered by the instructions

p	F5	pF
p + 1	26	244L

When the routine is left, the double precision quotient is in S3, 1S3. Temporary storage locations used are 0, 1, 3S3, 4S3.

The method of division is as follows: The sign of the divisor is stored at location 0. The absolute value of the divisor is computed and stored in 3S3, 4S3.

Let $x_1 + 2^{-39} x_2$ be the dividend and $y_1 + 2^{-39} y_2$ be the absolute value of the divisor. First a quotient q_1 satisfying

$$x_1 + 2^{-39} x_2 = q_1 y_1 + 2^{-39} r, \quad y_1 > r \geq 0$$

is computed. The quotient q_1 and remainder r are then corrected so that

$$x_1 + 2^{-39} x_2 = q_1^*(y_1 + 2^{-39} y_2) + 2^{-39}(r^* + (q_1 - q_1^*) y_2)$$

with $y_1 > r^* \geq 0$. This correction results in a double precision r^*. A correctly rounded quotient $r^*/y_1 = q_2^*$ is obtained giving a final quotient

$$q_1^* + 2^{-39} q_2^*$$

Remarks. In case the divisor is -1, the subroutine takes the negative of $x_1 + 2^{-39} x_2$ and then jumps out. In all other cases the above method is followed. When $y_2 = 0$ the division method gives an exact quotient when an exact quotient of 79 or fewer binary digits exists. This is particularly useful for integer work. The correction process works best if $y_1 + 2^{-39} y_2$ is a standardized number.

The mathematics of the correction process are:

Given $\quad x_1 + 2^{-39} x_2 = q_1 y_1 + 2^{-39} r \quad$ and $\quad y_1 > r \geq 0,$

determine an integer α such that

$$x_1 + 2^{-39} x_2 = (q_1 + \alpha 2^{-39})(y_1 + 2^{-39} y_2) + 2^{-39}(r^* - 2^{-39} \alpha y_2)$$

with $\quad y_1 > r^* \geq 0$.

This gives the equation $\quad r = \alpha y_1 + q_1 y_2 + r^*$.

Since $\quad 1 > y_1 \geq 1/2$ and $|q_1| \leq 1$ we have $|q_1 y_2| \leq 2y_1$ and

$(-2-\alpha)y_1 < r - q_1 y_2 - \alpha y_1 < (3-\alpha)\, y_1$ or

$(-2-\alpha)\, y_1 < r^* < (3-\alpha)\, y_1$. Since $y_1 > r^* \geq 0$ we must have $-3 < \alpha < 3$, or, since α is an integer α = one of -2, -1, 0, 1, 2.

Then $r^* = q_2^* y_1 + 2^{-39} r^{**}$ with $|r^{**}| \leq 1/2\, y_1$. Consequently

$$x_1 + 2^{-39} x_2 = q_1^* (y_1 + 2^{-39} y_2) + 2^{-39} (q_2^* y_1 + 2^{-39} r^{**} + 2^{-39} \quad - \alpha\, y_2)$$

$$= (q_1^* + 2^{-39} q_2^*)(y_1 + 2^{-39} y_2)$$

$$+ 2^{-78} (-q_2^* y_2 + r^{**} - \alpha\, y_2)$$

we have $\quad |-q_2^* y_2 + r^{**} - \alpha\, y_2 \quad| \leq 6\ 1/2\, y_1 < 7(y_1 + 2^{-39} y_2)$.

Hence the maximum error in the <u>quotient</u> is 7×2^{-78}.

OVERFLOW ANALYSIS

The initial division is y_1 into $x_1 + 2^{-39} x_2$. It is known that $1 > y_1 \geq 1/2$ while $-1/2 \leq 1 + 2^{-39} x_2 < 1/2$ since the dividend lies in the floating accumulator. Consequently the first quotient is less than 1 in absolute value unless $x_1 = -1/2$, $x_2 = 0$ and $y_1 = 1/2$. In this case, the ILLIAC quotient is $-1 + 2^{-39}$; when corrected this yields $q_1 = -1$ and $r = 0$. The correction process yields nothing further since $y_2 = 0$. In all other cases we have

$$y_1 + 2^{-39} y_2 \geq y_1 > |x_1 + 2^{-39} x_2|$$

Therefore q_1 of the equation

$$x_1 + 2^{-39} x_2 = q_1 y_1 + 2^{-39} r \qquad y_1 > r \geq 0$$

satisfies $\quad -1 \leq q_1 < 1$. The correction α satisfies the relation

$$r = \alpha y_1 + q_1 y_2 + r^* \qquad \text{with} \quad y_1 > r^* \geq 0 \quad .$$

Consequently if $q_1 \geq 0$ we must have $\alpha \leq 0$ since $y_1 \geq 0$, $y_2 \geq 0$, $r^* \geq 0$ and $y_1 > r$. If $q_1 < 0$, then, as $0 \leq r^* < y_1$, should we have $\alpha \leq -1$, then $r^* + \alpha y_1 < 0$ and $q_1 y_2 + \alpha y_1 + r^* < 0$. Contradiction as $r \geq 0$. Hence it has been shown that $|q_1^*| = |q_1 + \alpha| \leq |q_1|$ and overflow cannot result.

SIGN CHANGE SUBROUTINE

This little subroutine takes the negative of the double precision fixed point number in S3, 1S3 and puts it back in S3, 1S3. When the subroutine is left, A is cleared to zero. Enter by

p	F5	pF
p + 1	26	279L

Remember that $-(-1) = -1$ in ILLIAC.

FIXED POINT INTEGER-FRACTION READ

Read one unsigned integer or fraction of 12 or less decimal digits from tape, convert the number to its binary equivalent, and store it at a specified address.

For <u>integer</u> input, make the entry

p	J0	nF	50	pF
p + 1	26	327L		

The subroutine reads the decimal digits of a unsigned integer counting the number of digits read. Input is stopped by any character K, S, N, J, F, L or any 5 hole character other than a space. If c is in A on termination and c > 0 then c-10 is in location 0 when the routine is left. If c < 0 then c-15 is in location 0.

If k digits are read, Q contains $5 \times 10^{k-1}$ on exit from the routine. The integer is stored at location n of the Williams Memory.

For unsigned fraction input make the entry

p	50 nF	50 pF
p + 1	26 327L	

The subroutine reads the decimal fraction and converts it to a binary fraction using essentially the techniques of library routine N12. The binary fraction is stored at location n. The input is terminated in the same way as for integer input. The contents of location 0 are, on exit, C-10 or C-15 according as C > 0 or C < 0. See above.

Caution! This subroutine uses a 91 4F instruction to read from tape. The routine is sensitive to all 5 hole characters except the space. The subroutine is programmed to skip spaces.

Other parts of routine A7 use 80 1F instructions. Indiscriminate use of the 80 1F instruction between two 91 4F instructions can cause ILLIAC to read the same 5 hole character twice rather than advance the tape in the reader. The programmer who wishes to use this subroutine must be prepared to program around this difficulty.

PRINT ONE k-DIGIT POSITIVE INTEGER

This subroutine should be entered with the positive integer $n \times 2^{-39}$ in A. There are four entries corresponding to controls of zero suppression and spacing. The spacing is that specified by the last interpretive 9F nF instruction executed. The integer may contain $k \leq 12$ decimal digits.

p	J2 kF	50 pF	Punch or print a k digit integer without spacing or zero suppression on the left.
p + 1	26 349L		
p	J0 kF	50 pF	Punch or print a k digit integer with a space after each n decimal digits but without zero suppression on the left.
p + 1	26 349L		

p	52	kF	50 pF	Punch or print a k digit integer without spacing but with zero suppression on the left.
p + 1	26	349L		
p	50	kF	50 pF	Punch or print a k digit integer with a space after each n digits and with zero suppression on the left.
p + 1	26	349L		

Caution: if k digits are requested but the integer $n \geq 10^k$, this subroutine will not punch or print the extra digits.

METHOD OF EXPONENT CONVERSION

The method of exponent conversion used is based on the observation that the fractions $1/2$, $10/16$, $10^3/2^{10}$, $10^6/2^{20}$, $10^{15}/2^{50}$ can be represented exactly as single precision binary fractions. Since these fractions are nearly equal to 1, the amount of standardization necessary for exponent conversion is small.

The process of obtaining the floating binary equivalent of, for example, 10^{24}, is that of forming a double precision product $10^{15}/2^{50} \times 10^6/2^{20} \times 10^3/2^{10}$ and assigning the binary exponent 80. For a decimal exponent E_x in the range $0 \leq E_x \leq 153$ we have

$$0 \leq E_x = a_1 + 3a_3 + 6a_6 + 15a_{15} \leq 153$$

which gives the inequalities $a_{15} \leq 10$, $a_6 + a_3 \leq 2$, $a_3 \leq 1$ and $a_1 \leq 2$. Thus for exponents in this range the worst case is $E_x = 149$ for which $a_{15} = 9$, $a_6 = 2$, $a_1 = 2$. 23 single precision multiplications is the maximum number required to determine

$$10^{E_x} = f \cdot 2^{E_f} .$$

Let $E_x = 3k_1 + k_2$ with $0 \leq k_2 \leq 2$. Then $k_1 \leq 50$ and $(10^3/2^{10})^{k_1} \geq .3$ and $(10/16)^2 \cdot (10^3/2^{10})^{k_1} \geq .19$. Hence during input the process of converting the decimal power 10^{E_x} to a binary equivalent

$$f \cdot 2^{E_f}$$

can be accomplished with at most two standardizing shifts of f. Further $1/f \cdot 2^{-E_f}$ is a binary equivalent of 10^{-E_x}. The problem of exponent conversion during <u>input</u> is thus reduced to the following steps:

(1) Read fractional part and store in floating accumulator

(2) Read decimal exponent

(3) Compute binary equivalent $f \cdot 2^{E_f}$ of 10^{E_x} and standardize f.

(4) Form a double precision product or quotient of the floating accumulator with $f \cdot 2^{E_f}$

The problem of binary to decimal conversion during output is essentially the same as the above process. However, while in binary 1/2, 10/16, ... represent positive powers of 10, in the reverse process they represent negative powers of 2. That is to say

$$(10^3/2^{10} \times 10^6/2^{20} \times 10^{15}/2^{50}) \times 10^{-24} = 2^{-80}$$

Hence for binary exponents in the range $-512 \leq E_x < 0$ we have

$$-50\, b_{50} - 20\, b_{20} - 10\, b_{10} - 4\, b_4 - b_1 = E_x$$

which gives the inequalities $b_{50} \leq 10$, $b_{20} + b_{10} \leq 2$, $b_{10} \leq 1$, $b_4 + b_1 \leq 3$, $b_1 \leq 3$.

The worst case is $E_x = -493$ requiring 25 multiplications. The worst scaling problem is

$$(10^3/2^{10})^{50}\ (10/16)^2\ (1/2)^3 \geq .02 .$$

Thus after obtaining the decimal exponent equivalent at most one standardization by 10 is required.

Conversion of a positive binary exponent by means of this process requires a division of the fractional part by the adjusting factor computed. Prior to this division the fractional part is multiplied by 1/10 in double precision to eliminate the possibility of division overflow.

During input the coded conversion process will convert any decimal exponent to a binary exponent and place this exponent in the floating accumulator. Converted exponents out of the range $\pm$ 512 cause no difficulty as long as no attempt is made to store the number from the floating accumulator. The exponent range in the accumulator is $\pm 2^{39}$.

During calculation of a series of products binary exponents out of range may be obtained. Such exponents may be converted to decimal equivalents and a floating decimal answer punched. This will cause no difficulty so long as the computed conversion factor is

$$(10^3/2^{10})^{b_{10}} (10/16)^{b_4} (1/2)^{b_1} > .01 ,$$

where $b_4 \leq 2$ and $b_1 \leq 3$. Failure of this scaling rule may result in a division overflow during conversion of positive binary exponents but will not cause a failure of the routine during conversion of negative binary exponents.

The number in the floating accumulator is not standardized before exponent conversion for output. Hence if the fractional part has become less than 1/10, one or more zeros may appear on the left of the fractional part which is punched. This only means that enough arithmetic was performed without standardization to lose digits of accuracy.

FIXED POINT OPERATION

The only essential difference between the fixed point and floating point modes of operation is in the number storage. These differences have been described.

Fixed point addition operates exactly as floating point addition. When a number is brought to the number register 3S3, 4S3, 5S3, location 5S3 is cleared to zero. The number in 2S3 gives the number of previous shifts right of the floating accumulator to correct for overflows. In forming the sum this number is used to position the summands. Thus an arithmetically correct result may be constructed using the count in 2S3.

When operating in the fixed point mode the storage subroutine does not take into account location 2S3. The programmer must write his program in such a way that if overflow occurs it is detected by, for example, testing 2S3 for zero.

The division subroutine requires the divisor to be standardized. In floating point operation the divisor is brought from the memory and hence is standardized. In fixed point operation this need no longer be the case.

Use of the standardize instruction N2 F will standardize the floating accumulator. If k shifts are required then N(2S3)' = N(2S3) - k. This is valid for both modes of operation.

LOCATION OF PARTS OF A7

0L	Interpretation of instructions
17L	Entry for next interpretive instruction
19L	Make the reference to a b-register
22L	Switch for interpretation of 2^{nd} function digit
38L	b-registers
46L	Fetch one number from the memory
230L	Multiplication subroutine
244L	Division subroutine
279L	Take the negative of the floating accumulator
284L	Standardize subroutine
327L	Integer-fraction input subroutine
349L	Print one integer subroutine

HIDDEN CONSTANTS

159L	00 F 00 521F
385L	$2^{35}/10^{11}$ (single precision)
404L	00 F 00 10F
505L	1/10 (double precision)

DATE July 21, 1958 RT: 3/18/59
PROGRAMMED BY Roger W Farrell
APPROVED BY D. B. Gillies

lgr

A 7

LOCATION	ORDER		NOTES PAGE 1
	00 K(A7)		
0	00 59F		Bring next interpretive instruction
	L0 12L		from memory.
1	L0 13L		Closed subroutine entry
	40 2L		
2	00 F		
	00 F		
3	40 2F		
	50 2F		
4	36 19L		
	L4 392L		
5	36 19L		
	01 7F		
6	L4 7L		
	42 8L		
7	42 46L		
	S5 22L		
8	40 9L		
	26 F		Jump to function switch
9	00 F		
	00 F		
10	40 F		Fixed point-floating point switch
	00 F		
11	LL 4095F		
	LL 3072F		Constants
12	LL 4094F		
	4K 4076F		
13	5S F		
	S5 F		
14	5S 1F		
	S5 F		
15	50 F		
	S5 20F		
16	00 1F		
	00 2F		

LOCATION	ORDER		NOTES
17	L5 2L		
	36 1L		Next interpretive instruction
18	L4 14L		after 8J
	22 1L		
19	01 3F		
	L4 51L		Make b - reference
20	42 50L		
	42 181L		
21	01 4F		
	26 6L		
22	26 46L		b0 Function Switch
	00 74L		
23	26 46L		b1
	00 71L		
24	L1 13L		b2
	26 80L		
25	27 80L		b3
	00 F		
26	26 46L		b4
	00 101L		
27	26 46L		b5
	00 125L		
28	26 46L		b6
	00 129L		
29	26 46L		b7
	00 135L		
30	81 4F		b8
	26 414L		
31	L5 219L		b9
	26 469L		
32	41 F		bK
	26 145L		
33	L5 S3		bS
	26 160L		

LOCATION	ORDER		NOTES
34	26 46L		bN
	00 201L		
35	L5 2F		bJ
	26 205L		
36	L5 2F		bF
	26 215L		
37	50 9L		bL
	26 223L		
38	00 F		b-registers
	00 F		
39	00 F		
	00 F		
40	00 F		
	00 F		
41	00 F		
	00 F		
42	00 F		
	00 F		
43	00 F		
	00 F		
44	00 F		
	00 F		
45	00 F		
	00 F		
46	50 9L		Bring a number from the memory
	L5 F		
47	42 61L		
	01 12F		
48	40 F		
	L5 2F		
49	32 50L		
	L0 395L		

LOCATION	ORDER	NOTES	PAGE 4
50	36 51L		
	50 F		
51	J0 396L		
	S5 38L		
52	L4 F		
	42 55L		
53	L4 68L		
	40 62L		
54	L0 69L		
	36 62L		
55	41 5S3		
	L5 F		
56	40 3S3		
	F5 55L		
57	42 58L		
	L0 70L		
58	32 63L		
	50 F		
59	L3 10L		
	32 61L		
60	S5 F		
	J0 11L		
61	42 5S3		
	26 F		
62	85 11F		
	00 F		
63	40 3S3		
	F5 62L		
64	40 65L		
	50 F		
65	85 11F		
	00 F		
66	40 4S3		
	50 4S3		

A 7

LOCATION	ORDER		NOTES PAGE 5
67	41 5S3		
	26 59L		
68	85 11F		
	00 1536F		
69	85 11F		
	00 2560F		
70	41 5S3		
	L5 1024F		
71	F5 27L		
	42 79L		Execute b0, b1 inst.
72	L5 3S3		
	10 1F		
73	40 3S3		
	26 75L		
74	F5 26L		
	22 71L		
75	S1 F		
	40 4S3		
76	50 4S3		
	32 78L		
77	J0 390L		
	F1 3S3		
78	26 79L		
	L1 3S3		
79	40 3S3		
	22 F		
80	50 2F		
	L4 15L		Execute b2, b3 inst.
81	40 F		
	01 3F		
82	L4 83L		Set switch for 8, 9, K, S, N
	42 85L		
83	10 3F		
	S5 86L		

LOCATION	ORDER		NOTES
84	46 F		Jump for b ≤ 7
	32 96L		
85	L5 S3		
	26 F		Switch
86	36 87L		b = 8
	26 17L		
87	L5 F		b = 9
	22 1L		
88	41 10L		b = K
	26 17L		
89	49 10L		b = S
	26 17L		
90	L5 S3		b = N, Execute standardize N2 inst.
	40 3S3		Entry after 8J to standardize FA.
91	50 1S3		
	F5 91L		
92	26 284L		
	10 1F		
93	40 S3		
	S5 F		
94	40 1S3		
	F5 2S3		
95	L0 5S3		
	40 2S3		
96	26 17L		Execute count-jump on b-register
	L5 50L		
97	42 98L		
	42 99L		
98	L5 16L		
	L4 F		
99	50 F		
	40 F		

LOCATION	ORDER		NOTES
100	36 17L		
	26 87L		
101	L5 3S3		Execute b4 instruction
	10 1F		
102	40 3S3		
	L5 5S3		
103	L0 2S3		$E_y - E_x$
	40 F		
104	L3 F		$\|E_y - E_x\| = 0$. Add immediately
	32 116L		
105	L1 F		
	32 111L		
106	L5 5S3		$E_y > E_x$. Interchange
	40 2S3		
107	S5 F		
	50 1S3		
108	40 1S3		
	L5 S3		
109	40 4S3		
	L5 3S3		
110	40 S3		
	L5 4S3		
111	40 3S3		Set for positioning.
	L7 F		
112	42 115L		$\|E_y - E_x\| - 79 \geq 0$ Skip addition
	L0 393L		
113	36 17L		$\|E_y - E_x\| - 64 \geq 0$ Too many shifts
	L4 394L		
114	32 122L		
	26 115L		
115	L5 3S3		Position and add.
	10 F		
116	40 3S3		
	L5 1S3		

LOCATION	ORDER		NOTES PAGE 8
117	S4 F		
	40 1S3		
118	50 1S3		
	32 120L		
119	J0 390L		
	F5 3S3		
120	26 121L		
	L5 3S3		
121	L4 S3		
	40 S3		
122	22 139L		In case ≥ 64 shifts
	L4 389L		
123	42 115L		
	L5 3S3		
124	10 63F		
	22 115L		
125	L5 3S3		Execute b5 instruction
	10 1F		
126	40 S3		
	40 S3		
127	L5 5S3		
	40 2S3		
128	22 139L		
	00 F		
129	L3 10L		Execute b6 instruction
	32 130L		
130	L5 387L		
	L4 2S3		
131	L0 5S3		
	40 2S3		
132	22 132L		
	F5 132L		
133	26 244L		
	50 1S3		

LOCATION	ORDER		NOTES
134	22 139L		
	00 F		
135	L3 10L		Execute b7 instruction
	32 136L		
136	L1 387L		
	L4 2S3		
137	L4 5S3		
	40 2S3		
138	L5 4S3		
	F5 138L		
139	26 230L		Entry to correct FA for overflow
	LJ S3		
140	36 143L		
	F5 2S3		
141	40 2S3		
	L5 S3		
142	10 1F		
	40 S3		
143	S5 F		
	40 1S3		
144	26 17L		
	00 F		
145	L5 2F		Execute bK instruction
	46 F		
146	32 155L		
	L3 F		
147	36 154L		
	L5 F		
148	50 386L		
	00 10F		
149	40 3S3		
	F5 149L		

LOCATION	ORDER		NOTES
150	26 284L		
	10 1F		
151	40 S3		
	41 1S3		
152	L5 159L		
	L0 5S3		
153	40 2S3		
	26 17L		
154	41 S3		Entry to clear floating accumulator
	41 1S3		
155	26 153L		
	L5 50L		
156	42 157L		
	50 386L		
157	L1 F		
	40 F		
158	26 17L		
	00 F		
159	00 F		
	00 521F		
160	40 3S3		Execute bS instruction
	50 1S3		
161	L3 10L		Jump for fixed point
	36 199L		
162	50 1S3		
	F5 162L		
163	26 284L		Standardize
	40 3S3		
164	L3 5S3		test for zero.
	36 174L		
165	L5 387L		Round
	S4 F		
166	40 4S3		
	50 4S3		

LOCATION	ORDER	NOTES
167	32 172L	
	J0 390L	
168	F5 3S3	
	40 3S3	
169	32 172L	
	L1 3S3	
170	32 172L	
	49 3S3	
171	F5 389L	
	L4 2S3	
172	26 173L	Correct exponent for stand.
	F5 2S3	
173	L0 5S3	
	36 175L	
174	41 3S3	Set no. = 0.
	50 386L	
175	J0 11L	Pack exponent and test for overflow
	S4 F	
176	40 4S3	
	S0 F	
177	L0 398L	
	32 195L	
178	50 9L	
	01 12F	Unpack address
179	40 F	
	L5 2F	
180	32 181L	Determine b-modification
	L0 395L	
181	36 182L	
	50 F	
182	J0 396L	B-modify address and set store
	S5 F	instructions.
183	L4 F	
	42 186L	

LOCATION	ORDER	NOTES	
184	L4 196L		
	40 191L		
185	L0 197L		W.M. or drum?
	32 190L		
186	L5 3S3		
	40 F		
187	F5 186L		
	42 189L		
188	L0 198L		
	36 192L		
189	L5 4S3		
	40 F		
190	26 17L		
	L5 3S3		
191	86 11F		
	00 F		
192	F5 191L		
	40 194L		
193	50 F		
	L5 4S3		
194	86 11F		
	00 F		
195	26 17L		FF stop for overflow.
	FF 57F		
196	86 11F		
	00 1536F		
197	86 11F		
	00 2560F		
198	L5 3S3		
	40 1024F		
199	L5 S3		
	00 1F		

LOCATION	ORDER	NOTES
200	40 3S3	
	23 175L	
201	L5 S3	Execute bN
	32 203L	
202	22 202L	
	F5 202L	
203	26 279L	
	L5 3S3	
204	36 74L	
	26 101L	
205	32 206L	
	46 206L	
206	26 F	Execute bJ
	46 207L	
207	L1 F	
	40 F	
208	L5 50L	
	42 210L	
209	42 213L	
	50 396L	
210	J0 F	
	L5 F	
211	S4 F	
	40 F	
212	50 F	
	J0 399L	
213	S5 F	
	40 F	
214	26 17L	
	00 F	
215	32 219L	Execute bF
	L4 392L	
216	32 221L	
	46 218L	

LOCATION	ORDER		NOTES PAGE 14
217	47 219L		
	92 131F		
218	50 F		
	26 17L		
219	00 F		
	46 220L		
220	L5 F		
	22 207L		
221	00 F		
	46 211L		
222	47 221L		
	26 17L		
223	01 12F		Execute bL
	40 F		
224	47 207L		
	L5 2F		
225	36 208L		
	L4 392L		
226	36 207L		
	L5 50L		
227	42 228L		
	42 213L		
228	50 397L		
	J0 F		
229	43 210L		
	22 210L		
230	42 238L		Multiply subroutine
	L4 397L		
231	46 234L		
	46 235L		
232	LJ 388L		
	74 S3		
233	40 1F		
	S5 F		

LOCATION	ORDER	NOTES	
234	50 F		
	74 1S3		
235	50 F		
	74 S3		
236	40 S3		
	L5 1F		
237	S4 F		
	40 1S3		
238	50 1S3		
	32 F		Link
239	J0 390L		
	L5 1F		
240	36 242L		
	F1 386L		
241	L4 S3		
	22 242L		
242	F5 S3		
	40 S3		
243	27 237L		
	00 F		
244	42 276L		
	L5 3S3		Divide subroutine
245	40 F		Save Sign
	32 248L		
246	L5 253L		
	42 79L		
247	26 75L		Absolute value of divisor
	32 248L		$\lvert y_1 + 2^{-39} y_2 \rvert$
248	26 277L		Jump for -1
	S5 F		
249	40 4S3		
	50 1S3		

LOCATION	ORDER	NOTES
250	J0 389L	Save d_{78}
	S5 276L	
251	40 1F	
	50 1S3	
252	L5 S3	1^{st} division
	66 3S3	
253	40 1S3	Store remainder r_1
	S5 247L	
254	40 S3	Store quotient q_1
	36 266L	
255	F1 386L	
	L4 1F	
256	L4 3S3	
	L4 1S3	
257	40 1S3	Obtain correct remainder
	32 260L	
258	L4 3S3	
	40 1S3	
259	L5 S3	
	F0 386L	
260	40 S3	$- q_1 \times y_2$
	50 4S3	
261	71 S3	
	32 267L	
262	L4 1S3	
	32 271L	
263	L4 3S3	Adjust first remainder
	40 1S3	for double length divisor
264	L5 S3	
	F0 386L	
265	40 S3	
	27 262L	
266	L5 1F	
	L0 3S3	

LOCATION	ORDER		NOTES
267	22 256L		
	L4 1S3		
268	32 271L		
	40 1S3		
269	F5 S3		
	40 S3		
270	L5 1S3		
	L0 3S3		
271	26 268L		
	L0 3S3		
272	36 278L		
	80 1F		
273	L4 3S3		2nd division
	50 3S3		
274	66 3S3		
	10 1F		
275	SJ F		
	40 1S3		
276	L5 F		Link
	32 F		
277	L5 250L		Change sign of quotient when
	26 279L		divisor negative
278	L4 3S3		
	22 268L		
279	42 283L		Take negative of FA
	L1 1S3		
280	32 282L		Subroutine
	F0 390L		
281	40 1S3		
	F1 S3		
282	26 283L		
	L1 S3		
283	40 S3		Link
	23 F		

LOCATION	ORDER		NOTES
284	42 322L		Standardize subroutine
	41 5S3		
285	L5 3S3		
	32 286L		
286	26 291L		Form $-\lvert x_1 + 2^{-39} x_2 \rvert$
	S1 8F		
287	36 290L		
	40 4S3		
288	50 4S3		
	J0 390L		
289	F1 3S3		
	26 291L		
290	L1 3S3		
	32 322L		
291	00 1F		"Inner loop"
	36 303L		
292	00 1F		Shift until sign of A changes.
	36 304L		
293	00 1F		
	32 305L		
294	00 1F		
	36 307L		
295	00 1F		
	32 308L		
296	00 1F		
	32 309L		
297	00 1F		
	36 311L		
298	00 1F		
	32 312L		
299	00 1F		
	36 314L		

LOCATION	ORDER		NOTES
300	40 4S3		Count 8 and repeat above
	L5 5S3		
301	L4 286L		
	42 5S3		
302	L5 4S3		
	26 292L		
303	40 4S3		Count 0, 1, ..., 7
	22 316L		
304	40 4S3		According to the jump executed
	L5 389L		
305	22 315L		
	40 4S3		
306	F5 389L		
	22 315L		
307	40 4S3		
	L5 391L		
308	22 315L		
	40 4S3		
309	27 315L		
	40 4S3		
310	L5 389L		
	26 315L		
311	40 4S3		
	F5 389L		
312	26 315L		
	40 4S3		
313	L5 391L		
	26 315L		
314	40 4S3		
	F5 391L		
315	F4 391L		
	L4 5S3		
316	42 5S3		if $x_1 > 0$, take negative again
	L5 3S3		

LOCATION	ORDER		NOTES PAGE 20
317	32 318L		
	L5 4S3		
318	22 321L		
	S1 F		
319	36 323L		
	40 3S3		
320	50 3S3		
	J0 390L		
321	F1 4S3		
	10 1F		
322	F0 390L		Link
	22 F		
323	L1 4S3		
	32 324L		
324	22 321L		
	F1 386L		
325	L4 5S3		
	42 5S3		
326	2S 322L		
	00 F		
327	K5 F		Integer-Fraction input subroutine
	42 348L		
328	46 348L		
	36 330L		
329	L5 327L		
	22 330L		
330	F5 330L		
	42 345L		
331	41 4S3		
	50 405L		
332	26 337L		Read loop
	L0 404L		
333	32 338L		
	10 3F		

LOCATION	ORDER		NOTES
334	F4 4S3		
	00 2F		
335	F4 4S3		
	00 1F		
336	40 4S3		
	00 1F		
337	91 4F		
	32 332L		
338	L0 394L		Test for spaces.
	40 F		
339	L7 F		
	36 341L		
340	00 4F		
	26 337L		
341	S5 F		Decode for 5×10^k
	10 4F		
342	01 4F		
	L4 347L		
343	42 344L		
	42 346L		
344	L5 4S3		Divide for fraction input
	50 F		
345	22 345L		
	26 F		
346	S0 F		
	66 F		
347	10 1F		
	SJ 400L		
348	40 F		
	22 F		Link - Store Number
349	40 F		Integer Print Routine
	41 2F		

LOCATION	ORDER		NOTES
350	K5 374L		Plant link.
	42 375L		
351	46 2F		Store no. of digits
	32 352L		
352	27 353L		
	L5 365L		
353	42 353L		Set zero suppression
	S5 F		
354	00 6F		
	32 356L		
355	L5 350L		Set spacing
	46 370L		
356	22 357L		
	L5 365L		
357	46 370L		Set count on number of digits
	L5 384L		
358	L0 2F		
	40 2F		
359	L5 F		
	50 385L		
360	32 361L		Put rounding constant in A.
	L1 364L		
361	26 362L		
	L5 364L		
362	74 F		
	36 364L		
363	L4 385L		x $2^{35}/10^{11}$
	L4 385L		
364	10 35F		Store one character at 0
	40 F		
365	S5 377L		Store residue at 1
	40 1F		
366	L5 2F		
	36 382L		

LOCATION	ORDER		NOTES PAGE 23
367	L3 F		Test for zero.
	32 370L		
368	43 353L		
	L5 F		
369	00 36F		Print one character.
	82 4F		
370	22 F		Zero suppress?
	L5 353L		
371	10 1F		
	SJ F		
372	32 368L		
	F5 2F		
373	00 20F		Last character to be printed?
	32 368L		
374	26 381L		Count.
	F5 2F		
375	40 2F		Link.
	32 F		
376	50 1F		
	75 404L		
377	22 364L		Space between groups of digits?
	L5 221L		
378	L4 397L		
	46 221L		
379	32 374L		
	L5 211L		
380	L4 397L		
	46 221L		
381	92 963F		Print one space.
	22 374L		
382	F4 397L		
	40 2F		
383	26 376L		
	00 F		

LOCATION	ORDER		NOTES
384	00 11F		Count.
	LL 4084F		
385	2S 4015F		$2^{35}/10^{11}$
	LN 755F		
386	00 F		
	00 F		Constants
387	00 F		
	00 512F		
388	20 F		
	00 F		
389	00 F		
	00 1F		
390	7L 4095F		
	LL 4095F		
391	80 F		
	00 3F		
392	70 F		
	00 F		
393	00 F		
	00 79F		
394	00 F		
	00 15F		
395	10 F		
	00 F		
396	00 F		
	00 4095F		
397	LL 4095F		
	00 F		
398	00 F		
	00 1024F		
399	LL 4095F		
	00 4095F		

LOCATION	ORDER		NOTES
400	74 1701F		5×10^{11} Table of $5 \times 10^{k} \times 2^{-39}$
	28 2048F		
401	0S 2627F		5×10^{10}
	S7 1024F		
402	00 47F		5×10^{7}
	KL 128F		
403	00 4F		5×10^{6}
	N4 2880F		
404	00 F		10×2^{-39}
	00 10F		
405	23 619F		Key word
	J6 1616F		
406	00 476F		5×10^{8}
	J6 1280F		
407	00 F		5×10^{4}
	ON 848F		
408	01 672F		5×10^{9}
	5L 512F		
409	00 F		5×10
	00 50F		
410	00 F		5×10^{5}
	7K 288F		
411	00 F		5×10^{2}
	00 500F		
412	00 F		5
	00 5F		
413	00 F		5×10^{3}
	01 904F		
414	LO 404L		Execute 88 instruction
	40 5S3		Store sign
415	36 416L		
	FF 58F		Format error.
416	JO S3		Read first part of number as integer
	50 416L		

LOCATION	ORDER		NOTES
417	26 327L		
	S5 551F		
418	40 3S3		Store $5 \times 10^{k} \times 2^{-39}$
	L5 F		
419	L4 394L		Test termination for "."
	32 438L		
420	L0 404L		
	40 F		
421	L7 F		Jump for floating decimal format.
	36 440L		
422	50 1S3		
	50 422L		Read fractional part
423	26 327L		insert correct exponent.
	L5 417L		
424	42 2S3		
	50 1S3		
425	L5 S3		
	32 429L		
426	10 1F		Test integer part for overflow
	F0 390L		and correct.
427	40 S3		
	F5 2S3		
428	40 2S3		
	S5 595L		
429	40 1S3		Take negative of ab. value if minus.
	L3 5S3		
430	32 431L		
	F5 430L		
431	26 279L		
	L5 S3		
432	40 3S3		
	50 597L		
433	50 1S3		Standardize
	F5 433L		

LOCATION	ORDER		NOTES
434	26 284L		
	10 1F		
435	40 S3		
	S5 F		
436	40 1S3		
	L5 2S3		
437	L0 5S3		
	42 2S3		
438	26 17L		Set 1S3 = 0 when $\leq$ 12 digits
	41 1S3		
439	22 441L		
	00 F		
440	50 1S3		Read floating decimal numbers.
	50 440L		
441	26 327L		Read 2nd part of number as a fraction
	L5 F		
442	36 443L		Format error if number in 3 parts.
	22 415L		
443	42 455L		Store sign of exponent.
	41 4S3		
444	50 1S3		
	L5 S3		
445	10 1F		
	32 446L		
446	F0 390L		Scale floating accumulator to correct
	10 1F		range.
447	40 S3		
	S5 F		
448	40 1S3		
	50 F		
449	50 4S3		Double length division by $5 \times 10^{k} \times 2^{-39}$
	F5 449L		

LOCATION	ORDER		NOTES
450	26 244L		Take negative for negative numbers
	L3 5S3		
451	32 452L		
	F5 451L		
452	26 279L		
	50 F		
453	J0 F		Read exponent.
	50 453L		
454	26 327L		
	L1 F		
455	40 1F		Obtain signed exponent.
	L5 F		
456	40 F		
	L3 F		
457	36 467L		
	F5 457L		
458	26 564L		Entry to exponent conversion subroutine
	L5 5S3		
459	40 1F		
	F5 459L		
460	26 284L		Standardize conversion factor
	40 3S3		
461	L5 F		
	32 464L		
462	L1 1F		
	L4 5S3		
463	L4 387L		Multiply or divide according to sign of exponent.
	40 2S3		
464	22 132L		
	L5 1F		
465	L0 5S3		
	L4 387L		
466	40 2S3		
	26 138L		

LOCATION	ORDER	NOTES
467	L5 387L	
	40 2S3	
468	50 1S3	
	22 139L	
469	L4 397L	LF-CR? Execute 89 instruction.
	46 219L	
470	36 473L	
	92 131F	
471	92 515F	Print LF-CR and delay.
	L5 218L	
472	L4 397L	Reset spacing control.
	46 219L	
473	L5 211L	
	L4 397L	
474	46 221L	
	L5 S3	
475	36 477L	Print $\pm$ sign
	92 706F	
476	L5 519L	Take absolute value
	26 279L	
477	92 642F	
	92 963F	
478	L5 387L	
	L0 2S3	
479	40 F	$512 \times 2^{-39} - (E_x + 512)\cdot 2^{-39}$
	32 481L	
480	50 506L	Multiply fractional part by 1/10
	F5 480L	for later division
481	26 230L	
	L3 F	
482	36 507L	
	F5 482L	
483	26 566L	Compute exponent conversion factor.
	L5 3S3	

LOCATION	ORDER		NOTES
484	40 F		
	L0 505L		
485	36 488L		
	F5 485L		
486	26 586L		Multiply conversion factor by 10.
	40 3S3		
487	F5 5S3		
	40 5S3		
488	L5 387L		
	L0 2S3		
489	36 502L		Jump when $E_x \leq 0$
	F5 5S3		
490	40 2S3		Divide by correction factor.
	F5 490L		
491	26 244L		
	50 1S3		
492	L5 S3		
	00 1F		
493	40 F		
	40 S3		
494	S5 F		
	40 1S3		
495	L5 F		Multiply by 10 if possible
	F0 505L		
496	36 509L		
	F5 496L		
497	26 586L		Check if x10 produces overflow.
	32 498L		
498	26 509L		
	40 S3		
499	S5 F		
	40 1S3		

LOCATION	ORDER		NOTES
500	L5 2S3		
	FO 386L		
501	40 2S3		
	26 509L		
502	L1 5S3		Multiply by conversion factor if
	40 2S3		$E_x < 0$.
503	L5 4S3		
	F5 503L		
504	26 230L		
	22 491L		
505	ON 3276F		
	NN 3276F		1/10 double precision
506	66 1638F		
	66 1638F		
507	40 2S3		
	LJ S3		
508	32 491L		
	22 547L		
509	41 F		
	L5 2F		
510	46 F		Determine number k of digits
	L5 F		to print.
511	LO 384L		jump if $\geq$ 12 digits.
	36 531L		
512	L5 F		
	46 514L		
513	46 520L		
	50 F		
514	50 F		
	50 514L		
515	26 551L		Multiply by 10^k, rounded.
	LJ 1S3		
516	32 517L		Round product to nearest integer
	F5 S3		Test rounded product for overflow.

LOCATION	ORDER		NOTES
517	40 S3		
	L5 S3		
518	L0 F		
	32 547L		
519	L5 S3		
	50 477L		
520	J0 F		Print k digit integer
	50 520L		
521	26 349L		
	92 963F		Space
522	L5 2S3		
	36 525L		
523	92 706F		Print sign of exponent.
	L1 2S3		
524	40 2S3		
	22 525L		
525	92 642F		Print exponent.
	L5 2S3		
526	52 3F		
	50 526L		
527	26 349L		
	L5 219L		
528	L4 397L		
	32 529L		
529	26 17L		
	92 1F		
530	92 967F		Exit. Print delay and 2 spaces if
	26 17L		LF-CR will not preceed next number.
531	L0 397L		when $\geq$ 12 digits, take
	46 532L		x 10^{k-11}
532	50 F		
	50 532L		
533	26 551L		
	50 401L		

LOCATION	ORDER		NOTES
534	00 1F		and then x 10^{11} for residue
	7J 1S3		
535	40 1S3		
	L0 401L		
536	L0 401L		Test low order part for overflow.
	36 544L		
537	22 545L		
	L5 532L		
538	46 539L		Print higher part
	L5 S3		
539	J0 F		
	50 539L		
540	26 349L		delay
	92 1F		
541	L5 1S3		
	50 F		
542	J0 11F		Print lower part.
	50 542L		
543	26 349L		
	22 521L		
544	41 1S3		Correct high order part and
	F5 S3		test for overflow.
545	40 S3		
	L5 S3		
546	L0 F		
	32 547L		
547	22 537L		
	F5 2S3		When rounded product overflows,
548	40 2S3		replace by 1/10 double precision
	L5 505L		and repeat.
549	40 S3		
	L5 506L		

LOCATION	ORDER		NOTES
550	40 1S3		
	26 509L		
551	41 F		Given k, determine $5 \times 10^{k-1} \times 2^{-39}$
	K5 F		
552	42 563L		
	46 F		
553	L5 F		from table multiply $x_1 + 2^{-39} x_2$
	00 1F		
554	L4 F		
	L4 558L		
555	46 556L		by 10^k
	L5 405L		
556	10 F		
	01 4F		
557	L4 347L		
	42 558L		
558	50 8F		
	L5 F		
559	80 1F		
	40 F		
560	50 1S3		
	7J F		
561	50 S3		
	74 F		
562	40 S3		
	S5 F		
563	40 1S3		
	22 F		
564	42 581L		Convert Exponent subroutine
	F5 428L		
565	42 573L		
	22 567L		
566	42 581L		
	L5 428L		

LOCATION	ORDER	NOTES	
567	42 573L		
	49 3S3		
568	41 4S3		
	L7 F		
569	50 386L		
	00 20F		
570	40 5S3		
	L3 F		
571	32 581L		
	L5 432L		
572	42 583L		
	42 584L		
573	50 4S3		
	L5 F		
574	L4 5S3		
	36 583L		
575	L5 583L		
	L4 391L		
576	42 583L		
	42 584L		
577	L5 573L		
	L4 391L		
578	42 573L		
	L0 582L		
579	32 573L		
	L5 3S3		
580	00 1F		
	40 3S3		
581	22 581L		
	22 F		
582	50 4S3		
	L5 610L		
583	40 5S3		
	7J F		

LOCATION	ORDER		NOTES
584	50 3S3		
	74 F		
585	40 3S3		
	22 573L		
586	42 594L		
	S5 F		Multiply by 10 subroutine
587	40 1F		
	L5 F		
588	00 2F		
	L4 F		
589	40 F		
	S5 F		
590	L4 1F		
	40 1F		
591	50 1F		
	32 593L		
592	J0 390L		
	F5 F		
593	26 594L		
	L5 F		
594	00 1F		
	22 F		
595	LL 4046F		
	00 15F		Table for exponent conversion
596	LL 4081F		
	00 50F		
597	71 2813F		$10^{15}/2^{50}$
	49 2256F		
598	LL 4076F		
	00 6F		
599	LL 4090F		
	00 20F		

LOCATION	ORDER		NOTES
600	7K 288F		$10^6/2^{20}$
	00 F		
601	LL 4086F		
	00 3F		
602	LL 4093F		
	00 10F		
603	7J F		$10^3/2^{10}$
	00 F		
604	LL 4092F		
	00 1F		
605	LL 4095F		
	00 4F		
606	50 F		$10/2^4$
	00 F		
607	LL 4095F		
	00 F		
608	LL 4095F		
	00 1F		
609	40 F		1/2
	00 F		