<table>
<thead>
<tr>
<th>TITLE: AC Print, Octal Number, Sign, Single Column</th>
<th>LSR # OT 1,1 t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Closed</td>
</tr>
<tr>
<td>No. of Regs. in Subroutine</td>
<td>Temp. Regs. used by Subroutine</td>
</tr>
<tr>
<td>35</td>
<td>1t, 2t</td>
</tr>
</tbody>
</table>

Preset Parameters
- none

Program Parameters
- on entering Subroutine
  - ac: Number to be printed

Results
- on leaving Subroutine
- ac: 0

Description
When control is transferred to this routine, the number to be printed should be in the accumulator. A five digit octal number is printed preceded by either a "0" for a positive or a "1" for a negative number. After the number is printed a carriage return is executed so that continual use of this routine will result in a single column of numbers.

Notes:
1. No point is printed.
2. The qπ orders are set to suppress the punch.

MD, Nov. 9, 1951
JWC III, Nov. 12, 1951

DL-291
Abstract: This subroutine prints a five digit octal number with a "0" or "1" and no point preceding it and then a carriage return. A "0" designates a positive number and a "1" a negative number.

Upon entering the subroutine:
AC: number to be printed

Temporary Registers:
1t: register used to store the remainder of the word.
2t: register used to store the digit counter.

00 ta 19r Set return address
01 ts 1t Store value
02 cp 20r Is word negative?
03 ca 26r No. Print "0"
20→04 QP 144 Printing
05 cs 25r Set digit
06 ts 2t Counter
16→07 ca 1t Value in AC
08 sr*12
09 ad 34r Add start of
10 td 13r number table
11 sl 15 Store
12 ts 1t remainder
13 (ca 0) Put flexo code for digit in AC
14 qP 128 Print digit
15 ao 2t Have all digits been printed? No.
16 cp 7r
"Yes. Cause a carriage return Go back to main program"
17 ca 4r
18 qP 128 carriage return
19 (sp 0) Go back to main program
20 ad 24r Change sign
21 ts 1t Store value
22 ca 27r Print "1"
23 sp 4r
24 0.77777
25 p4 Initial value of counter
26 p45
27 p36
28 p39
29 p3 Number
30 p21 Table
31 p33
32 p43
33 p15
34 p25r

DL-292
DIGITAL COMPUTER LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
WHIRLWIND SUBROUTINE SPECIFICATION

<table>
<thead>
<tr>
<th>TITLE: AC Print and/or Punch, Octal Number (Magnitude), Sign, No Carriage Return</th>
<th>LSR # OT 1.2 t</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Regs.</td>
<td>Temp. Regs.</td>
</tr>
<tr>
<td>33</td>
<td>3</td>
</tr>
</tbody>
</table>

Classification Closed

Preset Parameters

vl p0 (does not need to be inserted) to print, or p64 to punch, or pl28 to punch and print simultaneously

Program Parameters

ac: Number to be recorded

Results

ac: Number has been recorded and is no longer available, C(AC) = pl

Description

When control is transferred to this subroutine, the number to be printed should be in the accumulator. A five digit octal number is printed preceded by either a "+" for a positive number or a "-" for a negative number, i.e. 0.12345→+12345 and 1.12345→-65432.

Notes:
1. No point is printed
2. No carriage return or space is printed
3. The gp instructions are normally set to print and suppress the punch.
**Title:** AC Print and/or Punch, Octal Number (Magnitude), Sign, No Carriage Return

**Abstract:** This subroutine records a five digit octal number with a "+" or a "-" and no point preceding it. An initial "+" designates a positive number and an initial "-" a negative number; i.e., 0.12345→+12345 and 1.12345→-55432 respectively.

**Preset Parameter:**
vl: p0 to print (does not need to be inserted), or p64 to punch, or pl28 to punch and print simultaneously.

Upon entering the subroutine:
AC: number to be recorded

**Temporary Registers:**
d: unused
1t: register used to store the remainder of the word.
2t: register used to store the digit counter.

| Enter-300 | ta 17r | Set return address | (Or)17 | sp (0) | Return to main program |
| 01 | ts 1t | Store number | 2r→18 | ca 1t | Positive magnitude in AC |
| 02 | _cp 18r | Is number positive? | 19 | ts 1t | Store number |
| 03 | ca 23r | Put Flexo code for "+" in AC | 20 | ca 24r | Put Flexo code for "-" in AC |
| 21r→04 | qpl28al | Print sign | 21 | sp 4r | |
| 05 | ar *12 | Set digit | 06 | ts 2t | Counter |
| 16r→07 | ca 1t | Value in AC | 11 | sl 15 | Store |
| 08 | ar *12 | |
| 09 | ad 22r | Add start of number table | 10 | td 13r | |
| 12 | ts 1t | Reminder | 11 | sl 15 | |
| (10r)13 | ca (0) | Put Flexo code for digit in AC | 14 | qpl28al | Print digit |
| 15 | ao 2t | Have all digits | 30 | p3 | Number |
| 16 | _cp 7r_ | been printed? | 31 | p4 | Table |
| | | |

25 | p45 |
26 | p36 |
27 | p39 |
28 | p3 |
29 | p21 |
30 | p33 |
31 | p43 |
32 | pl5 |
**TITLE:** Print C(AC) as Octal Number, Sign Digit and Complement, Point, Single Column Layout.

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Sub.</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>3</td>
<td>8 print</td>
<td>8 print</td>
</tr>
</tbody>
</table>

**Preset Parameters**
- v1 PC (does not need to be inserted) to print, or P64 to punch, or P128 to punch and print simultaneously.
- v2 (desired digit length) - 5

**Program Parameters**
- on entering Subroutine
  - ac: number to be recorded

**Results**
- on leaving Subroutine
  - ac: number has been recorded and is no longer available, C(AC) = qp 144 sl.

**Description**
When control is transferred to this subroutine, the number in the accumulator is printed as a five digit octal number preceded by either a "0" for a positive or a "1" for the complement of a negative number and an octal point. After the last digit is printed a carriage return is executed so that a single column of numbers is tabulated whenever the subroutine is used repeatedly.

**Notes**
1. The qp instructions are normally set to print and suppress the punch. By inserting p64 or p128 in preset parameter v1, one can punch and suppress the printer or punch and print simultaneously.

2. Form of output is
   - 0.12345 appears as 0.12345 and
   - 1.12345 appears as 1.12345.
TITLE: PRINT C(AC) AS OCTAL NUMBER, SIGN DIGIT AND COMPLEMENT, POINT, SINGLE COLUMN LAYOUT

Abstract: This subroutine prints a five digit octal number with a "0" or "1" and a point preceding it and then a carriage return. An initial "0" designates a positive number and an initial "1" the complement of a negative number.

Preset Parameters:

vl: p0 (does not need to be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously.

v2: (desired digit length) - 5.

Upon entering the subroutine:

AC: number to be recorded

Temporary registers:

d: unused

1t: register used to store the remainder of the word.

2t: register used to store the digit counter.

00 ta 20r Set return address
01 ts 1t Store number

02 op 21r Is number positive?
03 ca 27r Put flexo code for "0" and point in AC
2r → 04 qp 134s1 Print[20], shift

05 qp 128s1 Print point
06 os 35r Set digit counter

07 ts 2t

17r → 08 ca 1t Number in AC
09 sr* 12
10 ad 26r Add start of number table
11 td 14r

12 sl 15 Store remainder
13 ts 1t

11r → 14 ca (0) Put flexo code for digit in AC
15 |qp 144s1 Print digit

16 ao 2t Have all digits been printed?
17 op 8r

18 ca 15r } Print a carriage return
19 qp 128s1 } Return to main program
(0r)20 sp 0
2r → 21 ad 25r Change sign

22 ts 1t Store number
23 ca 28 r Put flexo code for "1" and point in AC

24 sp 4r

25 0.77777

26 p27r Start of number table
27 0.07155 "." and "0"
28 0.07144 "." and "1"

29 p39
30 p3
31 p21 Number table

32 p33
33 p43
34 p15

35 p4a2 Value for digit counter
**DIGITAL COMPUTER LABORATORY**

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

**WHIRLWIND SUBROUTINE SPECIFICATION**

<table>
<thead>
<tr>
<th>TITLE:</th>
<th>Print C(AC) as Decimal Fraction, Sign and Magnitude, Point, Single Column Layout.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LSR OT 2.2t</td>
</tr>
<tr>
<td></td>
<td>Tape T664-3</td>
</tr>
<tr>
<td></td>
<td>Classification</td>
</tr>
<tr>
<td></td>
<td>Closed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Sub.</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>d - 2t (3)</td>
<td>8 print</td>
<td>8 print</td>
</tr>
</tbody>
</table>

**Preset Parameters**

- v1 p0 (does not need to be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously
- v2 (Desired digit length) - 5

**Program Parameters**

- on entering Subroutine
  - ac: x, number to be recorded

**Results**

- on leaving Subroutine
  - ac: number has been recorded and is no longer available,
    - c(ac) = +0

**Description:**

When control is transferred to this subroutine the number in the accumulator is printed as a five-digit decimal fraction preceded by a sign and decimal point and followed by a carriage return. The last printed digit does not contain roundoff of additional digits.

**Note:**

1. The qp orders are normally set to print and suppress the punch. By inserting p64 or p128 in preset parameter v1 one can punch and suppress the printer or punch and print simultaneously.

**MD | JWCIII | AS**
---|-------|-----
1/18/52 | 1/21/51 | 1/23/52
TITLE: Print C(AC) as Decimal Fraction, Sign and Magnitude, Point, Single Column Layout.

Abstract: This subroutine prints a five digit decimal fraction preceded by a sign and decimal point and followed by a carriage return; i.e. 0.77776 appears as +.99993 and 1.77776 as -.00006. The last printed digit does not contain roundoff of additional digits.

Preset Parameters:
- v1: p0 to print, or p64 to punch, or p128 to print and punch simultaneously.
- v2: (Desired digit length) - 5.

Upon entering the subroutine:
- AC: x, number to be recorded

Temporary Registers:
- d: unused
- lt: register used to store remainder of the word.
- 2t: register used to store the digit counter.

| 00 | ta 20r | Set return address (Or) 20 sp(0) | Return to main program |
| 01 | ts lt  | Store number 2r→ 21 ca 26r | Put Flexo code for "-"and dec. point in AC |
| 02 | op 21r | Is number positive? | 22 sp 4r | Return to print |
| 03 | ca 25r | Put Flexo code for"+"and dec. point in AC | 23 p 10 | 10 x 2^15 |
| 22r→ | 04 | qp 134 s1 | Print {+}, shift | 24 p 27r | Start of number table. |
| 05 | qp 128 s1 | Print decimal point | 25 0.07143 | +. character |
| 06 | ca 37r | | 26 0.07107 | -. character |
| 07 | ts 2t | Set digit counter | 27 p 45 |

| 17r→ | 08 | om lt | | x | in AC | 28 p 36 |
| 09 | mh 23r | | x; 10 x 2^-15 | 29 p 39 |
| 10 | ad 24r | Add start of number table | 30 p 3 |
| 11 | td 14r | number table | 31 p 21 |
| 12 | s1 15 | | 32 p 33 |
| 13 | ts lt | Store remainder | 33 p 43 |

(11r) | 14 | ca(0) | Flexo code for digit in AC | 34 p 15 |
| | 15 | qp 128 s1 | Print digit | 35 p 13 |
| | 16 | ao 2t | Have all digits | 36 p 49 |
| | 17 | op 8r | been printed? | 37 p 4 a2 | Value for digit counter |
| | 18 | ca 19r | Print a | |
| | 19 | qp 144 s1 | carriage return | |
DIGITAL COMPUTER LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print Magnitude as Decimal Integer from
AC, Initial Zero Suppression, No
Carriage Return

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Sub.</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>4</td>
<td>5 print</td>
<td>5 print</td>
</tr>
</tbody>
</table>

LSR# OT 2.5 t

Classification
Closed

Preset Parameter
v1 p0 (does not need to be inserted) to print, or p64 to punch,
or p128 to punch and print simultaneously

Program Parameters
on entering Subroutine
AC: integer to be printed

Results
on leaving Subroutine
Number printed out on typewriter

Description

This subroutine prints out the magnitude of the number contained in the accumulator as a decimal integer. If \( x \cdot 2^{-15} \) is the binary number in the accumulator, the decimal digits of the equivalent decimal integers \( d_1 \) are obtained as follows:

\[
d_1 = \left[ \frac{x \cdot 2^{-15}}{10^4} \right] \cdot 2^{-15}
\]

where \([ ]\) = "integral part of"

\[
d_2 = \left( \left[ \frac{x}{10^4} \right] \cdot 10^4 \cdot 2^{-15} \right) \left( 2^4 \right) \left( \frac{10}{2^4} \right)
\]

\[
= \left[ \frac{x - \left( \frac{x}{10^4} \right) \cdot 10^3}{10^4} \right] \cdot 2^{-15}
\]

etc.

This form is used so that none of the manipulations will yield an overflow. Actually, because the number \( 2^{11}/10^4 \) is not
exact in the machine, a binary round-off is added at one place in the program to give a correct answer.

Notes:

1. Initial zeroes are suppressed. Zero will be printed as five spaces.
2. The form of output is:
   \[
   \begin{array}{c}
   63 \\
   17352 \\
   1 \\
   \text{(zero)} \\
   27
   \end{array}
   \]
3. The program resets automatically.
4. There is no carriage return.
Abstract: This subroutine prints out the magnitude of the number contained in the AC as a decimal integer. If \( x \cdot 2^{-15} \) is the binary number in the AC, the decimal digits of the equivalent decimal integers \( d_1 \) are obtained as follows:

\[
d_1 = \left[ x \cdot 2^{-15} \right] \cdot 2^{-15}
\]

where \([ \_ ] = \text{integral part of}\)

\[
d_2 = \left[ \frac{x - 2^{-15}}{10} \right] \cdot 2^{-15}
\]

etc.

This form is used so that none of the manipulations will yield an overflow. Actually, because the number \( 2^{11}/10^4 \) is not exact in the machine, a binary round-off is added at one place in the program to give a correct answer.

Preset Parameters

\begin{align*}
\text{vl} & \quad \text{p0 (does not need to be inserted) to print, or p64 to punch, or pl28 to punch and print simultaneously} \\
\end{align*}

Temporary Registers

\begin{align*}
d & \quad \text{unused} \\
1t & \quad \text{temporary storage} \\
2t & \quad \text{digit counter} \\
3t & \quad \text{suppressor}
\end{align*}

\begin{align*}
Enter & \quad 0 \quad \text{Set return address} \quad 06 \quad \text{cm} \quad 21r \quad x \quad 2^{11}/10^4 \\
01 & \quad \text{ts} \quad 21r \quad \text{Store number in AC} \quad 07 \quad \text{mh} \quad 32r \\
02 & \quad \text{cs} \quad 31r \quad \text{Set digit} \quad 27r \quad 08 \quad \text{sr} \quad \ast 11 \quad x \quad 2^{-11} \\
03 & \quad \text{ts} \quad 2t \quad \text{Counter} \quad 09 \quad \text{ts} \quad 21r \quad \text{and store} \\
04 & \quad \text{ca} \quad 0 \quad \text{Set zero} \quad 10 \quad \text{sl} \quad \ast 15 \quad \text{Subtract off} \\
05 & \quad \text{ts} \quad 3t \quad \text{Suppressor} \quad 11 \quad \text{ts} \quad 1t \quad \text{Integral part}
\end{align*}
TITLE: Print Magnitude as Decimal Integer from LSR. OT 2.5 t
AC, Initial Zero Suppression, No Carriage
Return

12 ao 2t \{Advance
17r-28 ca 29r \{Print
13 cp 15r \{Counter
29 qpl36sl \{Space
(0r)14 sp (0) \{Return to
30 sp 25r \{main program
13r-15 ca 21r \{Check for
31 p5 \{Counter
16 su 3t \{Initial
32 0.15067 \approx 2^{11}/10^{4}
17 cp 26r \{Zero
33 ca 35r \{Table entry
18 ca 21r \{No initial zero;
34 0.50000 \{10/16
19 ad 33r \{Add in table.
entry 35 p45
20 ts 21r \{Set
36 p36 \{Table of
(1r)(20r)21 (p0) \{Printer
37 p39 \{Decimal
22 qpl28sl \{Print
38 p3 \{Flexewriter
23 cs 0 \{Eliminate
39 p21 \{Characters
Zero 40 p33
24 ts 3t \{Suppressor
30r-25 ao 1t \{Round-off
41 p43
26 mh 34r \{Multiply by \^{%6
42 p15
27 sp 8r \{Recycle
43 p13
44 p49
**DIGITAL COMPUTER LABORATORY**

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

**WHIRLWIND SUBROUTINE SPECIFICATION**

<table>
<thead>
<tr>
<th>TITLE:</th>
<th>Print $6(A_0)$ as Decimal Integer, Magnitude only, Initial Zero Suppression, Print Final Zero, No Layout</th>
<th>LSR OT 2.5lt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TAPE T-883</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Classification Closed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Sub. 4 (d - 3t)</th>
<th>Average Time (operations) 5 print</th>
<th>Max. Time (operations) 5 print</th>
</tr>
</thead>
</table>

**Preset Parameters**

vl p0 (need not be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously

**Program Parameters**
on entering Subroutine

A0: Integer to be printed

**Results**
on leaving Subroutine

Number printed out on typewriter

**Notes:**

1. Initial zeroes are suppressed. Zero will be printed out as four spaces and a zero.
2. The form of output is:
   
   |   |
   | 63 |
   | 17352 |
   | 1 |
   | 27 |

3. The program resets automatically.
4. There is no carriage return.

<table>
<thead>
<tr>
<th>JWC III</th>
<th>MAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/10/52</td>
<td>2/11/52</td>
</tr>
</tbody>
</table>
TITLE: Print 0(AC) as Decimal Integer, Magnitude only, Initial Zero Suppression, Print Final Zero, No Layout

Abstract: This subroutine prints out the magnitude of the number contained in the accumulator as a decimal integer. If \( x \cdot 2^{-15} \) is the binary number in the accumulator, the decimal digits \( d_i \) of the equivalent decimal integers are obtained as follows:

\[
\begin{align*}
d_1 &= \left( x \cdot 2^{-15} \right) \left( \frac{211}{10^4} \right) 2^{-11} = \left( \frac{x}{10^4} \right) 2^{-15} \\
d_2 &= \left[ \frac{x \cdot 2^{-15}}{10^4} \right] - \left[ \frac{x}{10^4} \right] 10^4 \cdot 2^{-15} \left( \frac{2^4}{10^4} \right) \\
&= \left[ \frac{x - \frac{x}{10^4} \cdot 10^4}{10^4} \right] \cdot 2^{-15}
\end{align*}
\]

where \([ \cdot ] = \text{integral part of} \)

This form is used so that none of the manipulations will yield an overflow. Actually, because the number \( 211/10^4 \) is not exact in the machine, a binary round-off is added at one place in the program to give a correct answer.

Temporary Registers
- d - unused
- 1t - temporary storage
- 2t - digit counter
- 3t - suppressor

Preset Parameters.
- vl p0 (need not be inserted) to print, p64 to punch, or p128 to print and punch

00 ts 19r \ Set return address
01 ts 25r \ Set no. in AC
02 cs 35r \ Set digit counter
03 ts 2t \ Set zero suppressor
04 ca 0 \ (16r) 17 (p0) \ Print
05 ts 3t \ (Or) 19 sp (__) \ Return to main program
06 cm 25r \ x \ \frac{2^{11}}{10^4} \ 13r \rightarrow 20 \ ca 25r \ \text{Not at last digit}
07 mh 36r \ \text{Check for initial zero.}
08 **sr 11 \ x \ \frac{2^{11}}{10^4} \ 13r \rightarrow 20 \ ca 25r \ \text{Not at last digit}
09 ts 25r \ 22 \ cp 32r \ \text{Check for initial zero.}
10 sl 15 \ Subtract off integral part
11 ts 1t \ (24r) 25 (p0) \ Set Printer
12 as 2t \ Advance counter
13 \_op 20r \ Are we at last digit?
14 ca 25r \ Last digit
15 ad 37r \ Add in start of table
16 ts 17r
17 (p0) \ Print
18 qp 128s1
19 sp (__) \ Return to main program
20 ca 25r \ Last digit
21 su 3t
22 \_op 32r
23 ad 37r \ Add in table entry
24 ts 25r
25 (p0) \ Set Printer
26 qp 128s1 \ Print
27 cs 0 \ Eliminate
Print C(AC) as Decimal Integer, Magnitude only, Initial Zero Suppression, Print Final Zero, No Layout

28 ts 3t / zero suppressor 39 p45
34r 29 ao 1t Round off 40 p36
30 mh 38r
31 sp 8r
22r 32 ca 33r 
33 qp 136s1 Print space 43 p21
34 sp 29r 
35 p4
36 0.15067 \frac{211}{4}
37// ca 39r \frac{10}{16}
38 0.50000

Table of Digits

40 p36
41 p39
42 p5
43 p21
44 p33
45 p43
46 p15
47 p13
48 p49
TITLE: Print Out Flexowriter Characters, Previously Stored Three Characters to a Register by LSR OT 3.1at, Using Program Parameter

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Sub.</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>d = 4t (5)</td>
<td>13 + 14n (n=no. of chars.) (Actually limited by number of printouts)</td>
<td>13+20n (n=no. of chars.) (Actually limited by no. of printouts)</td>
</tr>
</tbody>
</table>

Classification Closed

Preset Parameters
vl: p0 (does not need to be inserted) to print, p64 to punch, and p128 to print and punch simultaneously.

Program Parameters on entering Subroutine
ul: p(address of first register in storage in which characters have been previously stored.)

Results on leaving Subroutine
Flexowriter characters printed out from storage.

Description
This subroutine prints out Flexowriter characters previously stored by LSR OT 3.1at, three to a register. Subroutine OT 3.1at stores characters 5-digits long, with a 00000 "tag" character indicating when the final digit is to be a 0 and when a 1. (For further description of LSR OT 3.1at, see the Specification sheet for LSR OT 3.1at). This subroutine reverses the process, unstacking and printing out. It will print out characters, placing a zero in the sixth digit, until a "tag" character is reached. Then it will place 1's in the sixth digit until another "tag" character is reached, etc. This is the exact reverse of LSR OT 3.1at.

When a negative number is reached (previously stored by the read-in routine) the routine stops printing and returns control to the main program.

Note
For notes on how many characters may be stored, etc., see the Specification Sheet for LSR OT 3.1at.
TITLE: Print Out Flexowriter Characters
Previously Stored Three Characters to a Register by LSR OT 3.1t, Using Program Parameter

Upon entering the subroutine:
sp (subroutine)
p (first address to be printed out of)

Preset Parameters:
v1: p0 (does not need to be inserted) to print, or p64 to punch, or p128 to print and punch simultaneously

Temporary Registers:
d: temp. character storage
1t: 3 counter
2t: operational register
3t: find digit
4t: digit to be exchanged

00 ta 1r Plant ul address
(3r)(0r) 01 ca 0 ) Plant first address
02 td 11r ) Set return address
03 ao 1r )
04 td 13r )
05 ca 0 ) Set final digit
06 ts 3t ) Set digit to be exchanged to pl
07 ca 36r )
08 ts 4t )
28r → 09 cs 37r ) Set 3-counter
10 ts 1t )
(2r) 11 ca 0 ) Pick up register to be printed
12 ts 2t )
(4r) 13 op 0 ) Are we finished?
28r → 14 ca 2t ) No. Shift character into position
15 sr* 10 )
16 su 0 ) Is this character a zero?
17 op 29r )
18 ts d ) No
19 sl* 15 ) Shift left and restore reg. contents
20 ts 2t )
21 ca d ) Add in final digit
22 sl* 1 )
23 ad 3t )
24 qp 128s1 ) Print
25 ao 1t ) Are we finished with registers?
26 _op 14r_ )
27 ao 11r ) Advance register no. and recycle
28 _sp 9r_ )
17r → 29 ca 2t ) Restore register contents shifted.
30 sl* 5 )
31 ts 2t )
32 ca 4t ) Exchange final digit
33 ex 3t )
34 ts 4t ) Back to check finish
35 _sp 25r_ )
36 pl ) final digit
37 p2 ) character counter
DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Read in Standard Flexowriter Characters and Store Three Characters to a Register, Using Program Parameter. (For Use with LSR CT 3.1t.)

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Sub.</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>66</td>
<td>d = 6t (7)</td>
<td>40 + 13u, where n is no. of characters (Present limiting case is no. of read-ins on PETR.)</td>
<td>40 + 26n (in the most unusual case) (Present limiting case is no. of read-ins on PETR)</td>
</tr>
</tbody>
</table>

Program Parameters
on entering Subroutine

\[ a_r : u_l \] (automatic)
\[ w_k : p \] (address of first register in which characters are to be stored)

Results
on leaving Subroutine
Flexowriter characters stored, five binary digits to the character, beginning at the register given in ul, three characters to the register. All registers are positive except the one succeeding the final storage register, which is set to no.

Description
This subroutine reads in Flexowriter characters continuously from the photoelectric reader, and stores them in the registers beginning at the address stored in ul. The standard Flexowriter tape to be called in is typed on a Flexowriter having a binary code the same as the code on the Flexowriter connected to the Whirlwind output.

A "stop" character, with a seventh hole, must be typed at the end of the tape as a "tag" character so that the subroutine may stop the storage process.

Notes
1. The Flexowriter characters are shortened to five-digit length, and stored three to a register. A second "tag" character, 00000, is used to indicate whether or not the sixth digit in a character is to be a zero or one. Whenever a series of characters switches from final digit "zero", to final digit "one", and vice versa, such a tag is inserted. This is not necessarily a waste of storage, since all alphabetical characters in the standard Flexowriter code have the sixth hole zero, and thus there is no need for such a tag in most storage of straight language. Similarly, all numbers and most punctuation marks have the sixth hole a "one", and so a series of numbers does not require such a "tag" character until an alphabetical character appears. Mixtures of constantly changing letters and
numbers will, however, be wasteful of storage; and in such rare cases, another pair of storage and print routines should be used. With ordinary typescript, an estimated 25% of storage can be gained, however, over "two-to-a-register" methods of storing Flexowriter standard characters.

2. The number of registers needed to store a given tape will thus be a function of the number of changes from "sixth-hole-zero" to "sixth-hole-a-one". In standard English language, an average formula would seem to be

\[ N = \frac{n}{3} + \frac{n}{20} = \frac{23n}{60} = .38n \]

Where \( N \) is the number of registers needed for storage, and \( n \) the number of characters, \( n/20 \) being considered an average number of such changeovers.

3. This subroutine must always be used in conjunction with Print-Out Routine OT 3.1t. It may be used in the following manner: Read in subroutine OT 3.1t, preferably in the upper or lower ends of storage. When control is transferred to this subroutine it will automatically read in and store the required standard Flexowriter characters in the addresses beginning at the address given by program parameter \( ul \). After the final stop character is reached, it will not store this, but instead make the succeeding register a negative zero. Control is then automatically transferred to the instruction following the \( ul \) register.

Following this, OT 3.1t may be read in by means of the standard read-in conversion program, or else, if it has been previously stored, control may be transferred to it at any desired time. For a fuller explanation of how this program prints-out, see the Specification Sheet for OT 3.1t.

4. The character 000001 cannot be read in by this subroutine, since it will be confused by the machine with the 5-digit "tag" character 00000. However, in the 1951 Flexowriter code, this is "back-space", which should not be needed, and in the 1952 Flexowriter code, this character is not produced by any key on the keyboard.
Upon entering the subroutine:
sp: subroutine
p: first address to be stored

Temporary registers:
d: temporary digit store
lt: last 6th digit representation
t: character store
st: present 6th digit representation
at: temporary store
5t: 3-counter
6t: temporary store

00 ta 1r Plant ul address
(3r)(0-1) 02 ca { } Plant 1st address where
02 td 53r } machine is to store chars.
03 ao 1r } Set return address
04 td 42r } Set return address
05 os 0 } Set last digit reg. in
06 ts 1t } for zero
07 os 64r } Store 3 counters
08 ts 5t } Store 3 counters
09 ca 0 } Set zero in temp
10 ts 4t } Store
29r → 11 qr 0 Read in
12 ca 8
13 sr* 5
14 td d } Store 6 digits
15 ca d } in 2t
16 sr* 5
17 ts 2t
18 su 65r } Check to see if
19 op 30r } char. is a"stop"
30r → 31 su 2t } Is char. a"stop"?
32 op 34r
33 sr* 20r } No. Back to check digits
34 ca 0 } Yes.
35 sp 48r } Store 00000
36 ca 0 } Store 00000
37 sp 48r
38 ca 53r } Plant last register
39 td 41r } address
TITLE: Read in Standard Flexwriter Characters and Store Three Characters to a Register, Using Program Parameter. (For Use with LSR # OT 3.1t.)

40 cs 0 \{ Put negative no. in last reg. \}
39r → 41 ts (0)

(04r) → 42 \( sp (0) \) \{ Back to main Program \}
26r → 43 ca 3t \{ Switch digit representations \}
44 ts 1t

45 ca 0 \{ Store 00000 \}
46 \( sp 48r \)

47 \( sp 27r \) \{ Back to store 5 digits \}
28r, 57r \( 48 \) ta 63r \{ Plant link of internal subroutine \}
35r, 46r

49 sr* 1
50 ts 6t
51 ca 6t
52 ad 4t \{ Shift left and store \}

(62r)(02r) 63 ts (0)
54 sl* 5
55 ts 4t

56 ao 5t \{ Are we at end of register? \}
57 \( cp 63r \)
58 ca 0 \{ Yes. Reset temp. store. \}
59 ts 4t
60 ca 64r
61 ts 5t
62 ao 53r \{ Reset 3-counter \}
57r
63 \( sp (0) \) \{ Advance Storage \}
64 p2 \{ Exit from internal s.r. \}
65 p51 \{ 3-counter reset \}

"stop" character
DIGITAL COMPUTER LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: (30,0,0) MRA Print and/or Punch, Decimal Fraction, Sign, Number of Digits Arbitrary, No Carriage Return, Sign Agreement (Interpreted)

<table>
<thead>
<tr>
<th>No. of_regs. in Subroutine</th>
<th>Temp. Regs. used by Sub.</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>76</td>
<td>4</td>
<td>94+19n, number of digits printed = n</td>
<td>94 + 27n</td>
</tr>
</tbody>
</table>

Preset Parameters

vx: pN, where N is the address in storage of the first register of the interpretive subroutine (in title of main program)
vl: p0 (does not need to be inserted) to print, p64 to punch, or pl28 to print and punch simultaneously
v2: pn, where n is the number of decimal digits to be printed.

Description

This subroutine prints and/or punches the sign and magnitude of the contents of the MRA in the following manner

\[ \pm \overline{d_1 d_2 \ldots d_n} \]

The number, n, of decimal digits to be printed is a preset parameter (v2). The digits, d_i, are obtained by multiplying the magnitude of the contents of the MRA successively by pl0.

This subroutine contains a sign agreement program so that the contents of the MRA need not be a number whose major and minor parts are of like sign.

The sp instruction transferring control to this subroutine must be an interpreted sp (i.e., control must be in the interpretive subroutine). After execution of the subroutine control remains in the interpretive subroutine which then proceeds to interpret the instruction following the sp instruction in storage.

There is no carriage return.

This subroutine can be used with any (30,0,0) interpretive subroutine. The contents of the MRA are left undisturbed during the execution of this subroutine.

<table>
<thead>
<tr>
<th>PCH</th>
<th>MRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/1/52</td>
<td>2/8/52</td>
</tr>
</tbody>
</table>
TITLE: MRA Print and/or Punch in (30,0,0) Interpretive

Abstract: This subroutine prints out a sign and a decimal point followed by the magnitude of the contents of the MRA as a decimal fraction. The decimal digits are obtained by multiplying the contents of the MRA successively by p10. The number of digits to be printed is a preset parameter (v2). There is no carriage return. The subroutine is interpreted and can be used with any (30,0,0) interpretive subroutine.

Preset Parameters

vx: pH, where H is the address in storage of the first register of the interpretive subroutine (in title of main program)
vl: p0 (does not need to be inserted) to print, or p64 to punch, or p128 to punch and print simultaneously
v2: pn, where n is the number of decimal digits to be printed

Temporary Storage

d unused
1t 
2t } Temporary storage
3t
4t Digit counter

```
00 | ta 37r
01 | sp ax  
02 | ca 3ax  
03 | ts 2t 
04 | ca 2ax  
55r → 05 | ts 1t 
54r → 06 | mr 2t  
07 | op 33r 
08 | ca 1t 
56r → 09 | op 13r 
10 | ca 74r  
11 | sp 15r 
9r → 12 | ca 75r 
13 | qp 134sl
14 | qp 128sl
15 | cs 61r  
16 | ts 4t 

.35r → 17 | cm 1t 
18 | mh 62r 
19 | ts 3t 
20 | sl 15 
21 | ts 1t 
22 | cm 2t 
23 | mh 62r 
24 | ts 2t 
25 | sl 15 
26 | ex 2t 
27 | sa 1t 
28 | ts 1t 
29 | ca 3t 
30 | ad 63r 
31 | td 32r 
32 | ca (0) 
33 | qp 128sl
```

Print a single digit
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>ao 4t</td>
<td>Have enough digits</td>
</tr>
<tr>
<td>35</td>
<td>cp 17r</td>
<td>been printed?</td>
</tr>
<tr>
<td>36</td>
<td>sp ax</td>
<td>Return control to</td>
</tr>
<tr>
<td>37</td>
<td>sp 0</td>
<td>int. subroutine</td>
</tr>
<tr>
<td>38</td>
<td>cm 1t</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>su 0</td>
<td>Is C(1t) ( \neq 0 )?</td>
</tr>
<tr>
<td>40</td>
<td>cp 57r</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>cm 2t</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>su 0</td>
<td>Is C(2t) ( \neq 0 )?</td>
</tr>
<tr>
<td>43</td>
<td>cp 51r</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>su 60r</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>ad 59r</td>
<td>Form 1 - (</td>
</tr>
<tr>
<td>46</td>
<td>ts 2t</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>ca 1t</td>
<td>Is C(1t) pos.?</td>
</tr>
<tr>
<td>48</td>
<td>cp 55r</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>su 60r</td>
<td>Form C(1t) - 2^{-15}</td>
</tr>
<tr>
<td>50</td>
<td>ts 1t</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>cs 2t</td>
<td>Complement C(2t)</td>
</tr>
<tr>
<td>52</td>
<td>ts 2t</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>ca 1t</td>
<td>Re-enter sign agree-ment</td>
</tr>
<tr>
<td>54</td>
<td>sp 6r</td>
<td></td>
</tr>
</tbody>
</table>

*Table*

<table>
<thead>
<tr>
<th></th>
<th>ca0</th>
<th>p0</th>
<th>nla2</th>
<th>v2</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>pi</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>p1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>p64r</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>0.07143</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>0.07107</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PROGRAMMED ARITHMETIC

P: 1.2 t (T747-3) is the same as PA 1.1 t except that
register 26r reads pax2.

PA 2.2 t (T723-1) is the same as PA 2.1 t except that
register 197r reads pax2.

Both the above subroutines use the new Flexowriter code
whereas PA 1.1 t and PA 2.1 t both use the old
Flexowriter code.
**DIGITAL COMPUTER LABORATORY**

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

**WHIRLWIND SUBROUTINE SPECIFICATION**

| TITLE: Operations on Real (15,15,0) Floating Point Double Register Numbers (General Subroutine) | LSR PA 1.1 t |
| Classification Interpretive |
| No. of Regs. in Subroutine | Temp. Regs. used by Sub. | Average Time (operations) | Max. Time (operations) |
| 97 | ---- | see description | see description |

Preset Parameters (to be typed in the title of the main program)

v8/ pk: k = separation between two registers of the number

vx/ pN: N = location of first register of subroutine in storage

**Description:**

By means of this subroutine various logical and arithmetic operations can be carried out upon real numbers represented in the (15,15,0) system and stored in two registers, say n and n+k, where k is specified by a preset parameter.

If \( a \) is any number such that
\[
-2^{-15} < |a| < 2^{15}, \quad \text{or} \quad a = 0,
\]
then there exists a signed 15 binary digit number \( x \) and a signed 15 binary digit integer \( y \) such that

\[
\frac{1}{2} \leq |x| < 1 \quad \text{or} \quad x = 0,
\]

\[
0 \leq |y| < 2^{15},
\]

and

\[
\left| \frac{a - x \cdot 2^y}{a} \right| \leq 2^{-16}.
\]

\( x \) and \( y \) are stored respectively in the two registers \( n \) and \( n+k \) for use by this subroutine.

**Example (a)** Let \( a = -300 \) (decimal)
Then \( a = -100101100 \) (binary)
and hence \( x = 1.011010011111111 \)
and \( y = 1001 \)
Register \( n \) contains \( 1.011010011111111 \)
and Register \( n+k \) contains \( 0.000000000001001 \)
Example (b) Let \( a = \frac{1}{10} \) (decimal)

Then \( a = 0.000110011001100110011 \ldots \)
and hence \( x = 0.110011001100110011 \),
and \( y = -11 \)
Register \( n \) contains 0.110011001100110011
and Register \( n+k \) contains 1.111111111111100

The programmer need not carry out this conversion process himself, but instead need only find a five decimal digit \( X \) and \( Y \) such that

\[
\frac{1}{10} \leq |X| < 1, \text{ or } X = 0
\]

\[
0 \leq |Y| \leq 32,767
\]

and \( \left| \frac{a - X \cdot 10^Y}{a} \right| \leq 10^{-s} \)

By giving this information to the proper subroutine in the IP section of the Library of Subroutines, the remainder of the conversion can be done automatically.

Operations upon numbers in this representation are coded using an instruction code similar to but differing somewhat from the standard WW code. Any number of these instructions can be performed in sequence by placing an \textit{sp ax} before the first instruction of the sequence. The instructions are then interpreted successively until a change-of-control instruction is reached at which point another sequence is performed unless the instruction is an \textit{sp ax}. In the latter case, ordinary WW operation is resumed at the register following the \textit{sp ax}.

The subroutine contains the analogy of an accumulator and a program counter. The "program counter" is in register 76r and contains the address of the register containing the instruction being interpreted. The "multiple register accumulator" (mra) is in registers 2r, and 3r and contains the result of the last instruction (except ta, cp, and sp). Register 2r contains \( x \) and 3r contains \( y \).
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ts n</td>
<td>Store the number in the mra in registers n and n+k.</td>
</tr>
<tr>
<td>ta n</td>
<td>Store the address of the register following the last sp or cp instruction in the address section of register n.</td>
</tr>
<tr>
<td>ex n</td>
<td>Exchange the number in the mra with the number in registers n and n+k.</td>
</tr>
<tr>
<td>cp n</td>
<td>If the number in the mra is negative, proceed as in sp, otherwise ignore the instruction.</td>
</tr>
<tr>
<td>sp n</td>
<td>If n ≠ ax, interpret next the instruction in register n. If n = ax, resume ordinary WW operation at the register following the register containing the sp ax.</td>
</tr>
<tr>
<td>ca n</td>
<td>Clear the mra and then add into the mra the number in registers n and n+k.</td>
</tr>
<tr>
<td>cs n</td>
<td>Clear the mra and then add the negative of the number in registers n and n+k into the mra.</td>
</tr>
<tr>
<td>ad n</td>
<td>Add the number in registers n and n+k to the number in the mra and leave the result in the mra.</td>
</tr>
<tr>
<td>su n</td>
<td>Subtract the number in registers n and n+k from the number in the mra and leave the result in the mra.</td>
</tr>
<tr>
<td>cm n</td>
<td>Clear the mra and add the magnitude of the number in registers n and n+k into the mra.</td>
</tr>
<tr>
<td>mr n</td>
<td>Multiply the number in the mra by the number in registers n and n+k and leave the result in the mra.</td>
</tr>
<tr>
<td>dv n</td>
<td>Divide the number in the mra by the number in registers n and n+k and leave the result in the mra.</td>
</tr>
</tbody>
</table>

N.B. In the above the phrase "the number in" should be read to mean "the number represented by the contents of".
NOTES:

1. Entering and Leaving the Subroutine
   Wherever an sp ax is encountered, the machine will enter the subroutine if it is not already in it, or leave it if it is in the subroutine.

2. Preset Parameters
   A general preset parameter, vx, is used to specify the location in storage of the first register of the subroutine. Another general preset parameter, v0, is used to specify the separation k of the two registers containing a number. Both of these parameters must be specified in the title of the program tape.

3. Accuracy of Arithmetic Operations
   The operations multiply and divide give a result with at least 14 1/2 digit accuracy, i.e. the result is rounded off at worst in the 15th place. The relative error in the operations add and subtract is at worst 2^-15, but it may be biased downwards if the magnitude of the sum is sufficiently greater than the magnitude of either of the addends.

4. Alarms
   Arithmetic overflow alarms can occur in registers 19r, 20r, 47r, 71r, or 72r. In all cases this occurs when the exponent of the result being calculated during an ad, su, mr, or dv instruction lies outside the range (-2^15, 2^15). When such an alarm occurs, the address of the instruction that is being interpreted will be in register 76r. Any other alarms in the subroutine will be due to excessive addresses or use of an instruction not listed above.
TITLE: Operations on Real (15,15,0) Floating Point Double \( \text{LSP} \leq 1 \text{,} \text{t} \) \( \text{Register Numbers (General Subroutine)} \)

**Instruction Code and Operation Times:**

- \( \text{ts} \) 23 \( \text{cp} \) 17 (+), 27 (-) \( \text{cs} \) 26 \( \text{cm} \) 25
- \( \text{sa} \) 18 \( \text{sp} \) 24 \( \text{ad} \) 41 \( \text{mr} \) 23
- \( \text{ex} \) 23 \( \text{ca} \) 27 \( \text{st} \) 43 \( \text{dv} \) 26

**reset Parameters:** (to be typed in the title of the main program)

- \( \text{v8/pk: } k = \text{ separation in storage of two registers of numbers} \)
- \( \text{v8/pN: } N = \text{ address in storage of initial register of subroutine} \)

```
00 \text{ta} 76r \{ \text{Store address of 1st instruction to be interpreted} \}
01 \text{sp} 76r \text{25} \text{(p0)} \text{digit storage}
02 \text{sp} 76r \text{26} \text{pa8 separation parameter}
03 \text{p0} \{ \text{multiple register} \}
04 \text{sp} 90r \text{89r} \text{27} \text{sp} 90r \text{"ts"}
05 \text{sp} 90r \text{89r} \text{29} \text{ca} 42r \text{"ta"} \text{store digits in n-register}
06 \text{sp} 90r \text{89r} \text{30} \text{ta} 0 \text{"cs"} \text{store digits in n-register}
07 \text{sp} 90r \text{89r} \text{31} \text{sp} 75r \text{"ex"}
08 \text{sp} 90r \text{89r} \text{33} \text{sp} 22r \text{"cp"}
09 \text{ts} 93r \text{24r,} \text{89r} \text{34} \text{ao 76r \"sp\" set n-register}
10 \text{sp} 90r \text{35} \text{ta} 42r \text{address}
11 \text{pl6} \text{36} \text{ca 88r \"set pick-up instruction\"}
12 \text{mr} 2r \text{37} \text{td 76r \"instruction's address\"}
13 \text{sp} 69r \text{38} \text{su 4r}
14 \text{ex} 2r \text{39} \text{ts 95r \"does address of cp \"}
15 \text{sr} 1 \text{\"Form x1 \times x2\"}
16 \text{dv} 2r \text{40} \text{cm 95r \"sp \"}
17 \text{sf} 25r \text{41} \text{su 0 \"sp instruction\"}
18 \text{ex} 3r \text{42} \text{sp 71r \"cp (0) \" different from ax?\"}
19 \text{su 95r} \text{43} \text{ts 91r \"Complement x2\"}
20 \text{ad 11r \"Form y1 - y2 + 16\"}
21 \text{sp 72r} \text{45} \text{ca 91r \"\"}
22 \text{cs 2r \"Is x1 < 0?\"}
23 \text{sp 75r} \text{47} \text{su 3r \"y2 - y1\"}
24 \text{sp 34r \"Go to sp routine\"}
25 \text{sp 55r \"Is y2 - y1 \> 0?\"}
```
Operations on Real (15,15,C) Floating Point Numbers (General Subroutine)

50  ad 3r  
51  ts 3r  Interchange 
52  ca 2r  \((x_1, y_1)\) and \(23r, 31r, 5, 75\) 
53  ex 95r  \((x_2, y_2)\) 
54  ts 3r  

49r \(\to\) 55  cm 91r  \(\text{Store } |y_2 - y_1|\) 
56  td 2r  
57  su 96r  \(\text{Is } |y_2 - y_1| > 15?\) 
58  op 60r  
59  sp 75r  Addition unnecessary 

58r \(\to\) 60  ca 96r  \(\text{Set } y_2 = 15\) 
61  ex 95r  

156r 62  sr (0)  \(\text{Form } x_2 2^{15} y_2 y_1\) 
63  sa 2r  \(\text{Add}\) \((82r)86\ ca (0)\) 
64  ts 2r  
65  ca 0  \(\text{Add overflow into}\) \((75r)86\ ca (0)\) 
66  ex 2r  \(\text{Pick up and}\) 
67  sr *15  \(\text{2}^{15}\) times the \(10r, 32r \to 90\ ca 2r\) 

12r \(\to\) 69  sf 25r  \(\text{Scale factor}\) 
70  ex 3r  and store result \((9r), (15r)93\) (p0) 
71  au 95r  \(\text{Form } y_2^+ y_1\) 
72  su 25r  \(\text{Subtract scale factor}\) 

94 \(\to\) 73  ex 3r  \(\text{Store result in}\) 
74  ts 2r  \(\text{proper registers}\) 
75  ao 76r  \(\text{Increase address of pick-up instruction}\) 
76  ca (0)  \(\text{Pick-up instruction}\) 
77  ts 91r  
78  td 30r  \(\text{Store instruction}\) 
79  td 85r  \(\text{and digits}\) 
80  ao 26r  \(\text{where}\) 
81  ts 93r  \(\text{necessary}\) 
82  td 86r  
83  sr *11  \(\text{form sp to proper}\) 
84  ao 28r  \(\text{part of subroutine}\) 

85  td 89r  
86  ca (0)  \(\text{Pick up and}\) 
87  ts 95r  \(\text{store } y_2\) 
88  sp (0)  \(\text{Pick up } x_2\) 

89  Go to at of subroutine for particular instruction 

4r \(\to\) ca 14r \(\to\) dv  
5r \(\to\) cs 27r \(\to\) ts  
6r \(\to\) ad 29r \(\to\) ta  
7r \(\to\) su 32r \(\to\) ex  
8r \(\to\) cm 33r \(\to\) cp  
12r \(\to\) mr 34r \(\to\) sp  

Temporary storage 

95 (p0) 
96 p15  

Specifications of WHIRLWIND I LIBRARY SUBROUTINE Number PA 2 2

Title: Extra-Precision and Floating-Point Real Number Arithmetic, using 2-register 24,6,0 Numbers; Basic Instruction Code with Division, INTERPRETIVE

Total Number of Registers Occupied by the Subroutine: 204 storage registers
Temporary Storage Registers Required by the Subroutine: no temporary regs.
Time Required to Perform the Subroutine: average = 50* WNI operations
maximum = 76* WNI operations
* per interpreted operation; see page 4 for details

Preset Parameters (Values to be indicated in tape title line)
\[ x \mid \text{pH}; \quad k = \text{address assigned to the initial register of the subroutine} \]
\[ x_2 \mid \text{pk}; \quad k = \text{separation between registers assigned to each 2-register number} \]

Description

This interpretive subroutine, when called into action, takes instructions (more strictly, program parameters written as instructions) one at a time from consecutive storage registers and performs the designated single-address operations defined by the interpreted-instruction code given on page 4. These operations are primarily arithmetical operations performed on real numbers represented in the 24,6,0 system. Each number is stored in some multiple-register location \( n \) consisting of the pair of registers \( n \) and \( n+k \), where \( n \) is the address of the given location and \( k \) is determined by preset parameter \( x_2 \).

The 24,6,0 number system represents any real number \( N \), provided that either \( N = 0 \) or \( 2^{-65} \leq |N| \leq 2^{-54} \), as a signed 24-binary-digit fraction \( x \) and a signed 6-binary-digit integer \( y \), where \( x \) and \( y \) are chosen in such a way that either \( x = 0 \) or \( 1 \leq |x| \leq 5 \) and that \( |1 - x_2 2^{-7}| < 2^{-24} \). Thus the number pair \( x, y \) represents \( N \) to within \( 0.0000006 \), equivalent to about 7 significant decimal digits. The sign and first 15 digits of \( x \) occupy one register while the sign and 6 digits of \( y \) and the last 9 digits of \( x \) occupy the second register of the pair assigned to contain the number \( N \). Details of this and other number systems are available elsewhere.

A multiple-register accumulator (MRA) is used in place of the AC in many interpreted operations. This MRA is not a special register as is the AC but rather is a group of 3 ordinary storage registers contained within the interpretive subroutine, specifically registers 2r,3r, and 4r. Even though only 2 registers are needed to contain a 24,6,0 number, 3 registers are used for the MRA to avoid the time-consuming operation of packing the last 9 digits of the number and the sign and 6 digits of the exponent together into one register after each interpreted instruction. A further advantage is gained in that any sequence of arithmetic operations is performed using 30 digits for the number and 15 digits for the exponent. This provides in effect a 30,15,0 system. The 24 and 6 limitation is imposed only when necessary, namely on ts and ex operations. Thus greater range and greater precision are available in sequences of arithmetic operations than the 24,6,0 system would normally allow.
Specifications of WHIRLWIND I LIBRARY SUBROUTINE  Number PA 2.2  Page 2

The roundoff error on ad and su is made in the 29th digit of the sum before it is scale-factored. That is, in adding any two 24, 6, 0 numbers, \( v \cdot \hat{2}^w \) to \( x \cdot \hat{2}^y \), assuming 1 > \( |v| > 0.5 \), 1 > \( |x| > 0.5 \), \( w > y \), the sum obtained is 
\[
\left( (y + x) \cdot \hat{2}^{y - w + 29} \right) \cdot \hat{2}^{w - z} = u \cdot \hat{2}^{w - z},
\]
where \( z \) is chosen in such a way that 1 > \( |u| > 0.5 \).

The roundoff in mr is made in the 28th digit.
The roundoff in \( \hat{c}v \) is made in the 27th digit.
The roundoff in \( ts \) and \( ex \) (i.e., in packing the 30, 15, 0 numbers into 24, 6, 0 form) is of course made in the 25th digit. If the exponent \( y \) is less than -63, the value -63 is substituted for it, without changing \( x \) in any way.

Arithmetic alarms, because of the floating point system employed, and because of the extended range allowed within the MRA, will normally not occur in an interpreted program unless the contents of the MRA call it \( x \cdot \hat{2}^w \), prior to a ts or ex operation has an exponent \( y > 63 \), in which case an overflow alarm always occurs at register 203r during the interpretation of the ts or ex operation, even if \( x = 0 \). If during an arithmetic operation the exponent \( y \) exceeds the bounds \( 2^{-15} < y < 2^{15} \), an overflow alarm will occur at register 28r, 85r, 130r, 175r or 176r.

Entry to and exit from the subroutine is accomplished by means of the instruction \texttt{spax}. The first instruction in a program is always performed in the Whirlwind code. When 24, 6, 0 operations are needed, control is transferred to the subroutine by \texttt{spax}, \( x \) being the parameter which specifies the location of the subroutine. Instructions following the first \texttt{spax} are then performed in the interpreted code. When operations on 1-register fixed-point words are desired, control is transferred back to the main program by \texttt{spax}. This \texttt{spax} is given a special interpretation by the subroutine and results in the instructions following it being performed in the Whirlwind code. Use of a sequence of Whirlwind-coded instructions between two interpreted instructions does not affect the contents of the MRA, but use of any interpreted instruction does affect the contents of the AC.

For numerical input at the present time, all decimal numbers to be converted to 24, 6, 0 form must be written as a signed decimal fraction which is less than \( 1 \cdot 0 \) and not less than \( 0.1 \) followed by a single signed decimal digit indicating the actual position of the decimal point. That is, any number \( N \) is written in the form \( N = x \cdot 10^y \), with 1 > \( |x| > 0.1 \) and -9 \( \leq y \leq 9 \), and with \( x \) having at most 8 decimal digits. For example,
- the number 300, which equals \( 0.3 \times 10^3 \), is written as \( +3 \mid +3 \);
- the number 0.01\( \pi \), which equals \( 0.31415927 \times 10^{-1} \) is written as \( +0.31415927 \mid -1 \);
- the number -1/128, which equals \( -0.78125 \times 10^{-2} \) is written as \( -0.78125 \mid -2 \).

Alternatively, any number may be converted to 24, 6, 0 binary form by hand and written as 2 standard single length octal numbers. The procedure for converting by hand is described elsewhere.
**Specifications of WHIRLWIND I LIBRARY SUBROUTINE Number PA 2.2 Page 5**

Allocation of storage locations to the necessary 2-register numbers, (both for the main program and the subroutines), temporary storage, the main program, the subroutines, and the interpretive subroutine PA 2.2 must at present follow a rather inflexible rule because of the input conversion procedures currently in use. The scheme to be followed is shown diagrammatically below, with decimal addresses used throughout. Notice that parameter \( x \) is at present assigned the value 852 in all programs.

<table>
<thead>
<tr>
<th>Numbers designated by programmer</th>
<th>Storage registers</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>address at start of program, usually 32.</td>
<td>main program 2-register numbers, 1st halves</td>
<td>the assignments to consecutive locations of the 2-register constants needed by individual subroutines is handled automatically by the conversion program. The number of locations needed is the sum of the numbers needed by individual subroutines.</td>
</tr>
<tr>
<td>total number of locations = 2k = parameter ( x ).</td>
<td>subroutine 2-register numbers, 1st halves</td>
<td>the number of temporary locations needed is the maximum of the numbers needed by the main program and the subroutines. Note that all locations are 2-register locations. For 1-register temporary storage, both halves of any 2-register location ( n ) may be used by referring to ( n + 1 ) for the first half and to ( n \times 1024 \times 2 ) for the second half.</td>
</tr>
<tr>
<td>address of start of temporary storage = parameter 0.</td>
<td>temporary storage, 1st halves</td>
<td>address of 2nd half of last main program number must be less than 530.</td>
</tr>
<tr>
<td>address of start of main program and of each subroutine and address of first instruction to be performed must be indicated</td>
<td>main program 2-register numbers, 2nd halves</td>
<td></td>
</tr>
<tr>
<td>address of start of interpretive subroutine = 852 = parameter ( x )</td>
<td>temporary storage, 2nd halves</td>
<td></td>
</tr>
<tr>
<td></td>
<td>interpretive subroutine</td>
<td></td>
</tr>
</tbody>
</table>

(address of last word of last subroutine must be less than 704.)

Space available for print subroutine 102.1
The interpreted instruction code of this subroutine is given below. The instructions have the same binary value as the similar Whirlwind instructions. Hence they are written, typed and converted in the same way as Whirlwind instructions and are in fact indistinguishable from them. The term "number in location m" is used to signify the number represented in 24,6,0 form by the 32 binary digits contained in the pair of registers n and n+1. The term "register m" is used to signify the single register m. Figures in parentheses give the number of Whirlwind instructions required to interpret the indicated instructions.

<table>
<thead>
<tr>
<th>Interpreted Instructions</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ca n (38)</td>
<td>Clear the MRA and add into it the number in location n.</td>
</tr>
<tr>
<td>cs n (36)</td>
<td>Clear the MRA and subtract from it the number in location n.</td>
</tr>
<tr>
<td>cm n (37)</td>
<td>Clear the MRA and add into it the magnitude of the number in location n.</td>
</tr>
<tr>
<td>ad n (72)</td>
<td>Add the number in the MRA to the number in location n and leave the sum in the MRA.</td>
</tr>
<tr>
<td>su n (76)</td>
<td>Subtract from the number in the MRA the number in location n and leave the difference in the MRA.</td>
</tr>
<tr>
<td>mr n (49)</td>
<td>Multiply the number in the MRA by the number in location n and leave the product in the MRA.</td>
</tr>
<tr>
<td>dv n (74)</td>
<td>Divide the number in the MRA by the number in location n and leave the quotient in the MRA.</td>
</tr>
<tr>
<td>ts n (48)</td>
<td>Transfer the number in the MRA to location n.</td>
</tr>
<tr>
<td>ex n (48)</td>
<td>Exchange the number in the MRA with the numbers in location n.</td>
</tr>
<tr>
<td>sp m (25)</td>
<td>Interpret next the instruction in register m (unless m = sp, in which case transfer control to the register following the one which contains the sp, so that the instruction following the sp is performed using the Whirlwind code).</td>
</tr>
<tr>
<td>cp m (24)</td>
<td>If the contents of the MRA is a negative number, proceed as in sp n above; if positive, ignore this instruction.</td>
</tr>
<tr>
<td>ta m (22)</td>
<td>Transfer the address p + 1 into the right 11 digit positions of register m, leaving the left 5 digit positions unchanged; p being the address of the most recently interpreted sp or effective cp operation.</td>
</tr>
</tbody>
</table>
**Purpose:** Operations on Real (24,6,0) Floating Point Double Register Numbers (General Subroutine)

**Instruction Code and Operation Times:**

- **ts** 48
- **cp** 21(+) 
- **cs** 36
- **cm** 27
- **ta** 22
- **sp** 25
- **ad** 72
- **mr** 49
- **ex** 48
- **ca** 38
- **su** 76
- **dv** 74

**Preset Parameters (to be typed in program title)**

- **vex2/pk:** k-separation in storage of two registers of number
- **vex/pN:** N-address in storage of initial register of this subroutine

```
Enter-->00  ta 179r: Set address of 1st 196r--->25  ex 198r: "dv"
01  sp 179r: Interpreted 26  ts 97r
02  (p0)  x1: Multiple 27  cs 102r: Form exponent
03  (p0)  x1: Multiple 28  ad 54r: of \(2^{-2/x_2}\)
04  (p0)  y1: register 29  ts 102r
196r--->05  | sr#30  "ca"
06  lca ax 30  cs 97r: Form and
13, 196r--->07  ca 191r  "cs"
08  sp 95r 31  mh 97r: Form and store
196r--->09  sp 129r  "ad"
10  p29 32  ex 198r
196r--->11  ta 97r  "su"
12  sp 126r 33  sr *2
196r--->13  sp 7r  "cm"
(170r)14  (p0) Temporary digits storage
24r--->15  sa 3r: Add two minor products
16  ts 3r
17  ca 0: Store
18  ex 198r: overflow
19  mh 2r: Form major product
20  sp 158r
49, 196r--->21  mr 2r  "mr" Form two
22  ex 3r: minor products
23  mr 198r 46  dv 97r
24  sp 15r 47  sl *15
48  ad 151r: Add two minor parts
49  sp 21r: of reciprocal,
```

\(2^{-2/x_2}\)

\(\left(\frac{2^{-2/x_2}}{x_2^2}\right)\)

(Use Euclid's algorithm)
50 p1
132r=73r sp 73r "cs"
111r=52 ca 2r Complement x
53 sp 164r
54 p2
196r=55 ca 201r "ts" Store digits
(181r)56 td(0) in indicated
57 sp 178r address
119r=58 ao 2r Increase x
59 sp 167r by 2^15
60 sp 35r
196r=61 sp 73r "ex"
62 p63
196r=63 cs 2r "cp" Is x negative?
64 sp 178r (180r)97 (p0)
196r=65 ao 179r "sp" Set return address
66 td 201r for sp ex
67 ca 18r Set pick up order
68 td 79r for ordinary cp & sp
69 su 6r (95r)102 (p0)
sp 199r Test to see whether
71 sp 179r instruction is sp ex
72 0.20000
51x,61r=73 ca 3r Round off x and
74 sr 6 store x x 2^6
75 ts 3r
76 sr *9 Add round-off carry 166r=109 cm 2r
77 sa 2r into x
78 ts 2r
79 ca 0 Is there an overflow?
80 sp 83r
81 su 0
82 sp 86r
80r=83 sl 14 Add overflow
84 ts 2r into x and x
85 ao 4r Increase y
82r=86 cm 4r
87 su 62r |y| <= 0?
196r=93 ca 97r ts n + k
88 op 92r
89 cs 4r y <= 0?
90 op 202r (i.e. y < -63?)
91 cs 62r Set y = -63
92 ts 4r
8r=95 ts 102r or ex n + k
94 ad 197r Store ts,ex,ca,cs, or cm n+k
96 ca 2r
99 sr *9
100 ex 4r
101 sl *9
103 sr *9
104 ex 3r
105 ts 2r
106 sl *15
107 ex 3r
108 sp 177r
110 su 0 x != 0?
111 op 52r
112 cm 3r x != 0?
113 su 0
114 op 122r Form
126  sd 17r  $|x_1^1| = 1$  
127  ts 3r  
128  ca 2r  $x_1 > 0$?  
119  sp_58r  
120  su 50r  Form  (146r) 151  (p0)  
121  ts 2r  $x_1 = 2^{-15}$  
114→122  cs 3r  Complement $x_1^1$  
123  ts 3r  
124  mr 2r  Form $x_1, x_1^1$  
125  sp_166r  
126  cs 198r  Complement  
127  ts 198r  $x_2, x_2^1$  
128  cs 97r  
9→129  ex 102r  Form and store  
130  su 4r  $y_2 = y_1$  
131  ts 97r  
132  sp_141r  $y_2 - y_1 > 0$?  
133  ad 4r  53→164  ts 2r  
134  ts 4r  Interchange  
135  ca 2r  $(x_1, x_1^1, y_1)$  125→166  sp_109r  
136  ex 198r  and $(x_2, x_2^1, y_2)$  59→167  ca 3r  
137  ts 2r  
138  ca 3r  
139  ex 102r  
140  ts 3r  
132→141  cm 97r  
142  su 10r  $y_2 = y_1 \ mod 29 > 0$?  
143  sp_145r  
144  sp_178r  No need for addition  
143→145  ad 5r  Store  
146  ts 151r  $sr^1 + |y_2 - y_1| = 108r \mod 177  ts 4r  

147  ca 50r  $y_2 = 1$  
148  ex 102r  
149  sr *15  
150  ad 198r  

Form and store  

$(x_2^1x_2^2 \cdot 2^{-15}) \cdot 2^{-1} = |y_2 - y_1|$

Form  

$(x_1^1x_1^2 \cdot 2^{-15}) \cdot 2^{-1}$

Store $x_1$

Add $x_1, x_1^1$

and $x_2, x_2^1$

Does sign $x_1 = \text{sign } x_1^1$?

Scale factor and store $x_1, x_1^1$

Form exponent of $x_1, x_1^1$
178  set 179  Increase address (181r)191 ca(0)  \( y_2 \) in reg. 102
179  set 180  Pick up next instruction 192 sr *9  Hold \( x_2 \) in AC.
180  ts 97r  Store instruction 193 ex 102r
181  td 56r  and digits 194 ts 198r
182  td 199r
183  ad 197r  (188r)196 (p0)  Go to part of I,S, for
184  td 191r  particular instruction
185  sr *27  Separation parameter
186  sl *17  Form sp to address 198 (p0)  Temporary storage
187  ad 60r  for particular
188  ts 196r  instruction 200 sp *19  Does address equal ax?
189  ca(0)  (66r)201 (sp)0  Return to register
190  ts 102r  Pick up \( x_2, x_2 \) and \( y_2 \) 202 ca 108r  Produce overflow
        203 ad 108r  alarm
**DIGITAL COMPUTER LABORATORY**

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

**WHIRLWIND SUBROUTINE SPECIFICATION**

<table>
<thead>
<tr>
<th>TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Minimal Routine with Sign Agreement, giving 28 Binary Digit Accuracy in mr)</th>
<th>LSR</th>
<th>PA 3.1 t</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Classification</td>
<td>Interpretable</td>
</tr>
<tr>
<td>No. of Regs. in Subroutine</td>
<td>Temp. Regs. used by Sub.</td>
<td>Average Time (operations)</td>
</tr>
<tr>
<td>96</td>
<td>------</td>
<td>see description</td>
</tr>
</tbody>
</table>

Preset Parameters (to be inserted in main program)
- \( v8/ pk: k \) = separation in storage between the two registers of a double register number
- \( vx/ pN: N \) = address in storage of the first register of the interpretive subroutine

**Description**

By means of this subroutine various logical and arithmetic operations can be performed upon real (30,0,0) double-register numbers. The numbers are stored in two registers whose addresses in storage differ by a preset parameter \( k \), i.e., if the major half of a double register number is stored in register \( n \), then the minor half is stored in register \( n+k \).

The operations are written in the usual WW instruction code, but the meanings of these operations may differ from the usual ones (see description of instruction code). Any number of these instructions may be performed in sequence by placing an \( sp \ ax \) before the first instruction in the sequence. The instructions in the sequence are then interpreted successively until a change-of-control instruction is reached at which point either another sequence of instructions is interpreted, or, if the change-of-control instruction is an \( sp \ ax \) ordinary WW operation is resumed at the register following the instruction \( sp \ ax \).

The multiple register accumulator (mra), in which the results of instructions are left, consists of the storage registers 2r and 3r.
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Function</th>
<th>Av. No. Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ts n</code></td>
<td>Transfer the contents of mra to registers <code>n</code> and <code>n+k</code></td>
<td>22</td>
</tr>
<tr>
<td><code>ta n</code></td>
<td>If <code>n</code> is the address of the last sp instruction executed by the subroutine, transfer <code>(m+1)</code> to the last 11 digits of register <code>n</code></td>
<td>18</td>
</tr>
<tr>
<td><code>ex n</code></td>
<td>Exchange the contents of mra with the contents of <code>n</code> and <code>n+k</code></td>
<td>22</td>
</tr>
<tr>
<td><code>cp n</code></td>
<td>If the contents of mra is negative, proceed as in the sp instruction, if the contents are positive disregard the instruction</td>
<td>22</td>
</tr>
<tr>
<td><code>sp n</code></td>
<td>If <code>sp n = sp ax</code>, take the next instruction to be interpreted from register <code>n</code>, if <code>sp n = sp ax</code>, resume ordinary WW operation at the register following the instruction <code>sp ax</code></td>
<td>22</td>
</tr>
<tr>
<td><code>ca n</code></td>
<td>Clear mra, put the contents of registers <code>n</code> and <code>n+k</code> in mra</td>
<td>23</td>
</tr>
<tr>
<td><code>cs n</code></td>
<td>Clear mra, put the complement of the contents of registers <code>n</code> and <code>n+k</code> in mra</td>
<td>22</td>
</tr>
<tr>
<td><code>ad n</code></td>
<td>Add the contents of mra to the contents of registers <code>n</code> and <code>n+k</code> and store the result in mra.</td>
<td>40</td>
</tr>
<tr>
<td><code>su n</code></td>
<td>Subtract the contents of registers <code>n</code> and <code>n+k</code> from the contents of mra, store the result in mra.</td>
<td>44</td>
</tr>
<tr>
<td><code>cm n</code></td>
<td>Clear mra, put the absolute value of the contents of registers <code>n</code> and <code>n+k</code> in mra</td>
<td>22</td>
</tr>
<tr>
<td><code>mr n</code></td>
<td>Multiply the contents of mra by the contents of registers <code>n</code> and <code>n+k</code>, store the result in mra.</td>
<td>34</td>
</tr>
</tbody>
</table>
Notes:

1. Entering and Leaving Subroutine
   Both entering and leaving the subroutine are accomplished by the instruction sp ax, where vx, a preset parameter, is the address in storage of the first register of the interpretive subroutine. When used to enter the subroutine, the first instruction interpreted is that following the instruction sp ax in storage. When used to leave the subroutine, ordinary WW operation is resumed at the register following the instruction sp ax.

2. Accuracy
   All of the operations executed by the subroutine are carried out with a 30 binary digit accuracy except mr, which is carried out with 28 binary digit accuracy.

3. Sign Agreement
   A sign agreement routine has been incorporated into the interpretive subroutine. This means that the major and minor halves of the number contained in the mra always have the same algebraic sign after an instruction has been executed by the interpretive subroutine.

4. Reasons for Machine Stoppage During the Subroutine
   (a) Arithmetic overflow at 55r - sp n+k 1, i.e. an excessive address is being used for storing the minor half of a double register constant.
   (b) Arithmetic overflow at 71r or 72r - overflow in sum or difference for the interpreted instructions ad or su.
   (c) Various alarms can result from trying to interpret an instruction which was not meant to be interpreted by the subroutine.

5. In any of the operations the address n may be the address of the mra.

6. The mra consists of the registers 2r and 3r.

7. If k = 1, the two parts of the double register number are stored in consecutive storage registers.

8. No subroutine using preset parameter v8 should be used unless it is later reset.
Abstract: This subroutine is a (30,0,0) interpretive subroutine which performs the instructions ta, ta, ex, cp, sp, ca, ca, ad, su, cm and mr. The double register constants dealt with by the subroutine are in registers whose addresses in storage differ by a preset parameter k, i.e., the C(n,n+k) represents a double register number. Exit and entry to the subroutine are accomplished by the instruction sp ax. In leaving the subroutine, if C(m) = sp ax, then ordinary WW operation is resumed at register m+1. In the description given below, the subroutine is assumed to be executing the instruction C(m) = xx n. There are two preset parameters.

Preset Parameters: (to be inserted in main program)

v6 pk: k = the amount by which the halves of a double register number are separated in storage
vx pN: N = the address in storage of the first register of the interpretive subroutine

```
00 ta 51r Enter interpretive
01 sp 51r subroutine
02 (p0) mra
03 (p0)
64r--->04 |sr 19r ca
64r--->05 sp 44r cs
64r--->06 sp 69r ad
64r--->07 sp 65r su
64r--->08 sp 44r cm
09 |p1
10 |p0a8 vl
11 |td ax vx
64r--->12 mr 2r
13 ex 3r
14 mr 28r
15 sa 3r
16 ts 3r
17 |ca 0
18 ex 2r mr
19 mh 28r
20 ts 28r
21 sl 15
22 sp 69r
87r--->23 ao 2r Form C(2r) + 2^-15
24 sp 50r
79r--->25 cs 2r Complement C(2r)
26 sp 49r
64r--->27 sp 44r ts
(62r)28 (p0)
64r--->29 ca 43r ta
13 (53r)30 td (0) Transfer m=1 to digit
14 sp 50r section of register n
15 sp 44r ex
16 sp 93r cp
16 ao 51r sp
17 95r,64r--->34
35 td 43r Store sp(m=1) in 43r
```
III. Operations on Real \((30, C, 0)\) Fixed-Point Double Precision Register Numbers (Minimal Routine with Sign Agreement, Giving 28 Binary Digit Accuracy in \(mr\))

---

36 ca 30r Store ca n in 51r 66 cs 28r
37 td 51r
38 su 11r
39 cp 41r Is xx n = sp ax? 64-
40 sp 61r 22-
39-
41 ad 9r 70 ts 3r
42 cp 51r 71 ca 2r ad, su, mr
55 r=43 sp (o) Leave interpretive sub. 73 ts 2r
27r, 32r c3 94
(56r)45 (po) Perform 75 mr 3r disagree in sign?
46 ts 3r ts, ex, ca, cs, cm 76 _cp 50r
47 ca 2r 77 cm 2r Is C(2r) \(\neq 0\)?
(54r)48 (po) 78 su 0
26r-49 ts 2r 79 _cp 25r
24r, 31r-50 ao 51r 80 cm 3r Is C(3r) \(\neq 0\)?
(56r)94 (Cr) 1r-51 ca (o) 81 su 0
40r, 42r
52 td 61r Store nn 82 _cp 20r
53 td 30r 83 su 9r Form 1 - |C(3r)|
54 ts 48r Store xx n 84 ad 17r
55 ad 10r 85 ts 3r
56 ts 45r Store xx (n+k) 86 ca 2r Is C(2r) positive?
57 td 63r Store (n+k) 87 _cp 23r
58 sr *11 Set up entry 88 su 9r Form C(2r) - 2^{-15}
59 ad 4r into table 89 ts 2r
60 td 64r 82-
52r-61 ca (o) Store C(n) in 28r 90 cs 3r Complement C(3r)
62 ts 28r 91 ts 3r
(57r)63 ca (o) Put C(n+k) in AC 33r-
60r64 sp (o) Enter table 94 _cp 50r Is C(2r) negative?
7r-65 ts 45r 95 _sp 34r Perform sp order.
**DIGITAL COMPUTER LABORATORY**

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

**WHIRLWIND SUBROUTINE SPECIFICATION**

<table>
<thead>
<tr>
<th>TITLE: Operations on Real (30,0,0) Fixed-Point Double-Register Numbers (General Routine with Sign Agreement, No Division) (Interpretive)</th>
<th>LSR PA 3.5t</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAPE T721-2</td>
<td></td>
</tr>
<tr>
<td>Classification Interpretive</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Subroutine</th>
<th>Average Time (operations) see description</th>
<th>Max. Time (operations) see description</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Preset Parameters**

\[ \text{pk: } k = \text{separation in storage between the two registers of a double register number} \]

\[ \text{pN: } N = \text{address in storage of the first register of the interpretive subroutine} \]

**Description**

By means of this subroutine various logical and arithmetic operations can be performed upon real (30,0,0) double-register numbers. The numbers are stored in two registers whose addresses in storage differ by a preset parameter \( k \), i.e., if the major half of a double register number is stored in register \( n \), then the minor half is stored in register \( n+k \).

The operations are written in the usual \( \text{M} \) instruction code, but the meanings of these operations may differ from the usual ones (see description of instruction code). Any number of these instructions may be performed in sequence by placing an \( \text{sp ax} \) before the first instruction in the sequence. The instructions in the sequence are then interpreted successively until a change-of-control instruction is reached at which point either another sequence of instructions is interpreted, or, if the change-of-control instruction is an \( \text{sp ax} \) ordinary \( \text{M} \) operation is resumed at the register following the instruction \( \text{sp ax} \).

The multiple register accumulator (mra), in which the results of instructions are left, consists of the storage registers 2r and 3r.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Function</th>
<th>Avgs. No. Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{ts n} )</td>
<td>Transfer the contents of ( \text{mra} ) to registers ( n ) and ( n+k )</td>
<td>22</td>
</tr>
<tr>
<td>( \text{ta n} )</td>
<td>If ( n ) is the address of the last ( \text{sp} ) instruction executed by the subroutine, transfer ((n+1)) to the last 11 digits of register ( n )</td>
<td>18</td>
</tr>
<tr>
<td>( \text{ex n} )</td>
<td>Exchange the contents of ( \text{mra} ) with the contents of ( n ) and ( n+k )</td>
<td>22</td>
</tr>
<tr>
<td>( \text{sp n} )</td>
<td>If the contents of ( \text{mra} ) is negative, proceed as in the ( \text{sp} ) instruction, if the contents are positive disregard the instruction</td>
<td>22</td>
</tr>
<tr>
<td>Instruction</td>
<td>Function</td>
<td>Avg. No. Operations</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>---------------------</td>
</tr>
<tr>
<td>sp n</td>
<td>If (sp_n = sp_{ax}), take the next instruction to be interpreted from register (n), if (sp_n = sp_{ax}), resume ordinary (WW) operation at the register following the instruction (sp_{ax})</td>
<td>22</td>
</tr>
<tr>
<td>ca n</td>
<td>Clear (mra), put the contents of registers (n) and (n+k) in (mra)</td>
<td>23</td>
</tr>
<tr>
<td>cs n</td>
<td>Clear (mra), put the complement of the contents of registers (n) and (n+k) in (mra)</td>
<td>22</td>
</tr>
<tr>
<td>ad n</td>
<td>Add the contents of (mra) to the contents of registers (n) and (n+k) and store the result in (mra)</td>
<td>40</td>
</tr>
<tr>
<td>su n</td>
<td>Subtract the contents of registers (n) and (n+k) from the contents of (mra), store the result in (mra)</td>
<td>44</td>
</tr>
<tr>
<td>cm n</td>
<td>Clear (mra), put the absolute value of the contents of registers (n) and (n+k) in (mra)</td>
<td>22</td>
</tr>
<tr>
<td>mr n</td>
<td>Multiply the contents of (mra) by the contents of registers (n) and (n+k), store the result in (mra)</td>
<td>34</td>
</tr>
<tr>
<td>s1(800+n)</td>
<td>Multiply (C(mra)) by (2^n) and leave the result in (mra)</td>
<td>29</td>
</tr>
<tr>
<td>sr(800+n)</td>
<td>Multiply (C(mra)) by (2^{-n}), and store the first 30 digits of the result without roundoff in (mra)</td>
<td>28</td>
</tr>
</tbody>
</table>

Notes

1. **Entering and Leaving Subroutine**

   Both entering and leaving the subroutine are accomplished by the instruction \(sp_{ax}\), where \(v_k\), a preset parameter, is the address in storage of the first register of the interpretive subroutine. When used to enter the subroutine, the first instruction interpreted is that following the instruction \(sp_{ax}\) in storage. When used to leave the subroutine, ordinary \(WW\) operation is resumed at the register following the instruction \(sp_{ax}\).

2. **Accuracy**

   All of the operations executed by the subroutine are carried out with 30 binary digit accuracy, affected only by roundoff on the thirty-first digit.

3. **Sign Agreement**

   A sign agreement routine has been incorporated into the interpretive subroutine. This means that the major and minor halves of the number contained in the \(mra\) always have the same algebraic sign after an instruction has been executed by the interpretive subroutine.

4. **Reasons for machine stoppage during the subroutine**

   (a) Arithmetic overflow at \(sp = sp_{n+k}\), an excessive address is being used for storing the minor half of a double register constant.
(b) Arithmetic overflow at 72r or 73r - overflow in sum or difference for the interpreted instructions ad or su.

(c) Various alarms can result from trying to interpret an instruction which was not meant to be interpreted by the subroutine.

5. The mra consists of registers 2ax (at 2r) and 3ax (at 3r).
Abstract: This subroutine is a (30,0,0) interpretive subroutine which performs the instructions ta, ta, ex, cp, ca, cs, ad, su, om, mr, al, and sr. The double register constants dealt with by the subroutine are in registers whose addresses in storage differ by a preset parameter k, i.e. \( C(n,n+k) \) represents a double register number. Exit and entry to the subroutine are accomplished by the instruction sp ax. In leaving the subroutine, if \( C(m) \) = sp ax, then ordinary \( WW \) operation is resumed at register \( m+1 \). In the description given below, the subroutine is assumed to be executing the instruction \( C(m) = xx n \). There are two preset parameters.

Preset Parameters: (to be inserted in main program)

\( v x 2 \): \( p k \): \( k \) = the amount by which the halves of a double register number are separated in storage

\( v x \): \( p n \): \( n \) = the address in storage of the first register of the interpretive subroutine

00 ta 51r \( \{ \) Set next instruction address 01 sp 51r
\( (25r) \rightarrow 02 (p0) x_2 \) mra
\( 03 (p0) x_1 \)
65r \( \rightarrow 04 |sr 19r \) ca
65r \( \rightarrow 05 sp 44r \) cs
65r \( \rightarrow 06 sp 70r \) ad
65r \( \rightarrow 07 sp 66r \) su
65r \( \rightarrow 08 sp 44r \) om
65r \( \rightarrow 09 cs 2r \) Is \( C(2r) \) negative?
10 op 50r
11 sp 34r \( \{ \) Perform sp instruction 65r \( \rightarrow 12 ts 19r \)
13 mr 3r \( \) mr
14 sp 94r
65r \( \rightarrow 15 |ca ax \) sl (\( v x \))
65r \( \rightarrow 16 ca 3r \) sr
17 sr* 15
18 ad 2r \( \{ \) sl, sr \( C(n+k)=x_2 \)
(103r)(55r)(12r) \( \rightarrow 19 (p0) \)
20 ts 2r
21 sl* 15
22 sp 92r
23 ao 2r Form \( C(2r)+2^{-15} \)
24 sp 50r
25 cs 2r Complement \( C(2r) \)
26 sp 49r
27 sp 44r \( ts \)
28 (65r)(65r) \( \rightarrow 29 (p0) \) C(n) = \( x_2 \)
30 td(0) \( \{ \) Transfer \( m+1 \) to digit section of register D
31 sp 50r
32 sp 44r \( ex \)
33 sp 9r \( op \)
34 ao 51r \( sp \)
35 td 43r \{ store sp(m+1) \)
36 ca 62r \( in 43r \)
37 td 51r \{ Is \( xxx = spax \)
38 su 15r \( \}
39 cp 41r
TITLE: Operations on Real (30,0,0) Fixed-Point Double-Register Numbers (General Routine with Sign Agreement, No Division) (Interpretive)

40 sp 51r
39r ⏸ 41 ad 122r
42 op 51r
5r,8r
27r,32r
(35r) 43 sp 0
Leave interpretive subroutine
(97r)(66r)(57r) 45 (po)
(113r)
46 ts 3r
47 ca 2r
Perform ts, ex, ca, cs, and cm.
(54r) 48 (po)
26r ⏸ 49 ts 2r
10r, 24r
31r, 77r
50 ao 51r
Pick up next instruction to be interpreted
(1r,40r,42r,0r)(34r)(37r)(50r)
51 ca(0)
52 td 62r
Store n
53 td 30r
54 ts 48r
Store xxx
55 ts 19r
56 ad 121r
57 ts 45r
Store xx(n+k)
58 td 64r
Store (n+k)
59 sr* 11
Set up entry into table
60 ad 4r
61 td 65r
(52r) 62 ca(0)
Store C(n) in 28r
63 ts 28r
(58r) 64 ca(0)
put C(n+k) = x'_2 in AC 14r
(61r) 65 sp(0)
enter table
7r ⏸ 66 ts 45r
67 cs 29r
78 ts 28r
69 cs 45r

6r ⏸ 70 sa 3r
12r ⏸ 71 ts 3r
72 ca 2r
73 ad 28r
74 ts 2r
75 cs 2r
Do C(2r)and C(3r) agree in sign
76 mr 3r
77 op 50r
78 cm 2r
79 su0
Is C(2r) ≠ 0?
80 op 25r
81 cm 3r
82 su0
Is C(3r) ≠ 0?
83 op 91r
84 su 122r
85 ad 118r
Form - 1 + |C(5r)|
86 ts 3r
87 ca 2r
Is C(2r) positive?
88 op 23r
89 su 122r
Form C(2r) - 2^-15
90 ts 2r
91 cs 3r
Complement C(3r)
92 ts 3r
93 sp 75r
Re-enter sign agreement
94 sr 2
Form 1/4 [x'_1 x'_2] and store in 19r
95 ex 19r
96 mh 2r
Form x'_1 x'_2, store major
97 ts 45r
half in 46r
98 a1*15
99 sr 2
Form 1/4 [x'_1 x'_2]
100 ad 19r \{ Accumulate round-off and store in 2r
101 ex 2r
102 mh 28r
103 ts 19r \{ Form $x_1x_2$, store major
104 sl* 15 \{ half in 19r and minor
105 ex 28r \{ half in 28r
106 mh 3r \{ Form $x'_1x'_2$ and store
107 ts 3r \{ major half in 3r
108 sl* 15 \{ Form $\frac{1}{4}[x'_1x'_2]$
109 sr 2
110 ad 2r \{ Form total round-off and add into major half of $x_1x_2$
112 sa 45r
113 ts 45r
114 ca 19r
115 ex 28r
116 sa 45r \{ Add $x_1x'_2$ and the
117 ts 2r \{ major half of $x_1x'_2$
118 ca0
119 ex 2r
120 sp 70r \{ Enter addition routine
121 |0ax2 v1
122 |p1
DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

<table>
<thead>
<tr>
<th>TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Short, fast routine without sign agreement giving 28 binary digit accuracy in mr) Interpretive</th>
<th>LSR PA 3.10t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tape: TAPE T798</td>
<td></td>
</tr>
<tr>
<td>Classification Interpretive</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Regs.</th>
<th>Temp. Regs.</th>
<th>Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>in Subroutine</td>
<td>used by Sub.</td>
<td>(operations)</td>
</tr>
<tr>
<td>78</td>
<td>0</td>
<td>---</td>
</tr>
<tr>
<td>Max. Time</td>
<td>(operations)</td>
<td>---</td>
</tr>
</tbody>
</table>

Preset Parameters (To be inserted in title of main program)

\[ vx \ pN, \text{ where } N \text{ is the address in storage of the first register of the interpretive subroutine} \]

\[ vx2 \ pk, \text{ where } k \text{ is the separation in storage between the two registers of a double register number} \]

Description

By means of this subroutine various logical and arithmetic operations can be performed upon real (30,0,0) double-register numbers. The numbers are stored in two registers whose addresses in storage differ by a preset parameter \( k \), i.e. if the major half of a double register number is stored in register \( n \), then the minor half is stored in register \( n+k \).

The operations are written in the usual WW instruction code, but the meanings of these operations may differ from the usual ones (see description of instruction code). Any number of these instructions may be performed in sequence by placing an \( sp \ ax \) before the first instruction in the sequence. The instructions in the sequence are then interpreted successively until a change-of-control instruction is reached at which point either another sequence of instructions is interpreted, or, if the change-of-control instruction is an \( sp \ ax \) ordinary WW operation is resumed at the register following the instruction \( sp \ ax \).

The multiple register accumulator (MRA), in which the results of instructions are left, consists of the storage registers 2r and 3r.

This interpretive subroutine does not contain a sign agreement routine and hence the contents of the MRA may have different signs, e.g. in the subtraction

\[ 10\cdot2^{-15} + 5\cdot2^{-30} - [5\cdot2^{-15} + 10\cdot2^{-30}] = 5\cdot2^{-15} + (-10)\cdot2^{-30} \]

the result has major and minor halves of unlike sign. This fact must be remembered by the programmer if computations are carried on outside of the subroutine or if other library subroutines are interpreted by the interpretive subroutine. In the latter case, a library subroutine should not be used unless it is explicitly stated in its specification sheet that it is permissible to use it with interpretive subroutines not containing sign agreement.
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Function</th>
<th>Av. No. Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ts</td>
<td>Transfer the contents of MRA to registers ( n ) and ( n+k )</td>
<td>17</td>
</tr>
<tr>
<td>ta</td>
<td>If ( m ) is the address of the last sp instruction executed by the subroutine, transfer ((m+1)) to the last 11 digits of register ( n )</td>
<td>13</td>
</tr>
<tr>
<td>ex</td>
<td>Exchange the contents of MRA with the contents of ( n ) and ( n+k )</td>
<td>17</td>
</tr>
<tr>
<td>cp</td>
<td>If the contents of MRA is negative, proceed as in the sp instruction, if the contents are positive disregard the instruction</td>
<td>((+)14 ) ((-)25)</td>
</tr>
<tr>
<td>sp</td>
<td>If ( sp \neq sp \text{ ax} ), take the next instruction to be interpreted from register ( n ), if ( sp \neq sp \text{ ax} ), resume ordinary ( \text{MM} ) operation at the register following the instruction ( sp \text{ ax} )</td>
<td>21</td>
</tr>
<tr>
<td>ca</td>
<td>Clear MRA, put the contents of registers ( n ) and ( n+k )</td>
<td>17</td>
</tr>
<tr>
<td>cs</td>
<td>Clear MRA, put the complement of the contents of registers ( n ) and ( n+k )</td>
<td>17</td>
</tr>
<tr>
<td>ad</td>
<td>Add the contents of MRA to the contents of registers ( n ) and ( n+k )</td>
<td>18</td>
</tr>
<tr>
<td>su</td>
<td>Subtract the contents of registers ( n ) and ( n+k ) from the contents of MRA</td>
<td>17</td>
</tr>
<tr>
<td>mr</td>
<td>Multiply the contents of MRA by the contents of registers ( n ) and ( n+k ), store the result in MRA</td>
<td>33</td>
</tr>
</tbody>
</table>
Notes:

1. **Entering and Leaving Subroutine**
   Both entering and leaving the subroutine are accomplished by the instruction `sp ax`, where `vx`, a preset parameter, is the address in storage of the first register of the interpretive subroutine. When used to enter the subroutine, the first instruction interpreted is that following the instruction `sp ax` in storage. When used to leave the subroutine, ordinary `W` operation is resumed at the register following the instruction `sp ax`.

2. **Accuracy**
   All of the operations executed by the subroutine are carried out with a 30 binary digit accuracy except `mr`, which is carried out with 28 binary digit accuracy.

3. **Sign Agreement**
   `PA 3.10t` does not contain sign agreement. Hence the major and minor halves of a number need not have the same sign.

4. **Reasons for Machine Stoppage During the Subroutine**
   (a) Arithmetic overflow at 56r—an excessive address is being used for storing the minor half of a double register number.

   (b) Arithmetic overflow at 49r or 56r—overflow in sum or difference for the interpreted instructions `ad` or `or`.

   (c) Various alarms can result from trying to interpret an instruction which was not meant to be interpreted by the subroutine.

5. The MRA consists of registers 2ax(2r) and 3ax(3r).

---

<table>
<thead>
<tr>
<th>FOR</th>
<th>MAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/5/52</td>
<td>2/8/52</td>
</tr>
</tbody>
</table>
Abstract: This subroutine is a $(30,0,0)$ interpretive subroutine which performs the instructions \(ts, ta, ex, cp, sp, ca, cs, ad, su, cm\) and \(mr\). The double register constants dealt with by the subroutine are in registers whose addresses in storage differ by a preset parameter \(k\), i.e., the \(C(n, n+k)\) represents a double register number. Exit-and entry to the subroutine are accomplished by the instruction \(sp\ ax\). In leaving the subroutine, if \(C(m) = sp\ ax\), then ordinary \(WW\) operation is resumed at register \(m+1\). In the description given below, the subroutine is assumed to be executing the instruction \(C(m) = xx\ n\). There are two preset parameters. The subroutine does not contain sign agreement. Hence the major and minor halves of a double register number can have unlike signs.

Preset Parameters
\(vx\ pN\), where \(N\) is the address in storage of the first register of the interpretive subroutine
\(vx2\ pK\), where \(K\) is the separation in storage between the two registers of a double register number

\[
\begin{align*}
00 & \quad ta\ 53r \quad \text{Enter interpretive subroutine} \\
01 & \quad sp\ 53r \\
(76r) \quad 02 & \quad (p0) \quad mr_a \\
(74r) \quad 03 & \quad (p0) \\
59r & \quad \rightarrow 04 \quad sp\ 44r \quad ca \\
59r & \quad \rightarrow 05 \quad sp\ 44r \quad cs \\
59r & \quad \rightarrow 06 \quad sp\ 7r \quad ad \\
6r, 59r & \quad \rightarrow 07 \quad ad\ 26r \quad su \\
08 & \quad vs\ ur \quad \{ \text{Form} ca(n+k) \text{ or cs} \} \\
77r & \quad \rightarrow 09 \quad (p0) \quad (n+k) & \text{store in 9r} \\
10 & \quad sa\ 3r \quad \{ \text{store in 3r (ad, su)} \} \\
11 & \quad sp\ 48r \quad \{ \text{Form} x_2^1 + x_1^1 \} \\
39r & \quad \rightarrow 12 \quad td\ 68r \quad mr \\
13 & \quad ts\ 72r \quad \{ \text{Set address at 68r to} \} \\
14 & \quad ad\ 25r \quad n & \text{store mr(n+k) in 9r} \\
15 & \quad ts\ 9r \\
16 & \quad ca\ 5lr \quad \{ \text{Store ts2r in 50r} \} \\
17 & \quad ts\ 50r \\
18 & \quad sp\ 67r \\
33r & \quad \rightarrow 19 \quad cm\ 2r \quad \{ \text{Is} O(2r) \neq 0 \}
\end{align*}
\]
TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Short, fast routine without sign agreement giving 28 binary digit accuracy in mr) Interprete

\[
\begin{align*}
40 & \text{ cm 47r} \\
41 & \text{ su 0} \\
(35r) & \text{ 42 cp(0) } \\
43 & \text{ sp 53r} \\
4r,5r & \rightarrow 44 \text{ ad 25r } \\
27r,32r & \rightarrow 45 \text{ ts 47r } \\
\{39r\} & \rightarrow 46 \text{ ca 3r} \\
(45r) & \rightarrow 47 \text{ (p0)} \\
11r & \rightarrow 48 \text{ ts 3r} \\
\{39r\} & 49 \text{ ca 2r} \\
(17r)(54r) & 50 \text{ (p0)} \\
51 & \text{ ts 2r} \\
23r,66r & \rightarrow 52 \text{ ao 53r } \\
(34r)(37r)(0r) & \rightarrow 53 \text{ ca(0)} \\
1r,43r & \rightarrow 54 \text{ ts 50r } \\
(52r) & 55 \text{ sr* 11} \\
56 & \text{ ad 55r } \\
57 & \text{ ts 59r} \\
58 & \text{ ca 50r } \\
\{57r\} & 59 \text{ (p0) Enter table} \\
21r & \rightarrow 60 \text{ cm 3r} \\
61 & \text{ su 0 } \\
62 & \text{ op 34r} \\
63 & \text{ cs 3r} \\
64 & \text{ sp 23r} \\
(29r) & 65 \text{ td (0) Transfer m+1 to digit section } \\
66 & \text{ sp 52r of register n} \\
18r & \rightarrow 67 \text{ ca 2r } \\
68 & \text{ mh(0)n} \\
69 & \text{ ts 47r Store } [x_1 x_2]^{1} \text{ in 47r} \\
70 & \text{ sl 15 } \\
71 & \text{ ex 3r } \\
(13r) & 72 \text{ (p0) Form } [x_1 x_2]^{1} \\
73 & \text{ sa 3r } \\
74 & \text{ ts 3r } \\
75 & \text{ ca 47r Store } [x_1 x_2]^{1} \text{ in 47r} \\
76 & \text{ ex 2r } \\
77 & \text{ sp 9r }
\end{align*}
\]
TITLE: Operations on real (15,0,c) fixed-point single register numbers (Interpretive)

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Sub.</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>None</td>
<td>See description</td>
<td>of order code.</td>
</tr>
</tbody>
</table>

Preset Parameters (To be typed in the title of the Main Program)

vx: pN: N = address in storage of initial register of subroutine
vx3:pc: c = the number of binary digits to the right of the binary point in the (15,0,c) numbers
vx1:pM: M = address in storage of the initial register of the storage block for (15,0,c) numbers

Description

By means of this subroutine various logical and arithmetic operations can be performed on real numbers expressed in the (15,0,c) system, 0 ≤ c ≤ 15. The (15,0,c) constants are stored in the following manner. Let c be fixed and let a be a positive number such that

\[ 2^{-(15-c)} \leq a \leq 2^{c+1} - 1 \]

Then a can be written as the sum of a c digit binary integer and a binary fraction. The binary fraction is then rounded off to 15 - c digits and the result stored with the sign digit zero. If a < 0, repeat this procedure for -a, complement this number and store the result.

For example, let c = 3 and \( a = -\frac{1}{3} \)

Then

\[ -a = 3 + \frac{1}{3} \]

and

\[ 3 = +.011 \]

\[ \frac{1}{3} = +.010101010101010 \]

\[ 3 + \frac{1}{3} = 011.010101010101 \]

We now complement the following number

0.01101010101010101

and store the following result

1.10010101010101010

The programmer need not carry out this conversion process himself but instead need only write the fixed point decimal number. By giving this information to the proper subroutine in the IP section of the library the conversion can be done automatically.
**DIGITAL COMPUTER LABORATORY**  
**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**  
**WHIRLWIND SUBROUTINE SPECIFICATION**

**TITLE:** Operations on real $(15,0,c)$ fixed-point  
single register numbers (Interpretive)  

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Function</th>
<th>Average # Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ts n</td>
<td>Transfer $c(mra)$ to register n.</td>
<td>14</td>
</tr>
<tr>
<td>td n</td>
<td>Transfer the last 11 digits from the mra to the last 11 digits of register n</td>
<td>14</td>
</tr>
<tr>
<td>ta n</td>
<td>If $m$ is the address of the last sp instruction or effective op instruction executed by the subroutine, transfer $(m+1)$ into the last 11 digits of register n</td>
<td>13</td>
</tr>
<tr>
<td>ex n</td>
<td>Exchange $c(mra)$ with $c(n)$</td>
<td>14</td>
</tr>
<tr>
<td>op n</td>
<td>If $c(mra)$ is negative, proceed as in the sp instruction, if $c(mra)$ is positive, disregard the instruction</td>
<td>18</td>
</tr>
<tr>
<td>sp n</td>
<td>If $snp \neq spax$, take the next instruction to be interpreted from register n, if $snp = spax$, resume ordinary Whirlwind operation at the register following the instruction spax</td>
<td>21</td>
</tr>
<tr>
<td>ca n</td>
<td>Clear mra, put $c(n)$ in the mra</td>
<td>14</td>
</tr>
<tr>
<td>cs n</td>
<td>Clear mra, put the complement of $c(n)$ in the mra</td>
<td>14</td>
</tr>
<tr>
<td>ad n</td>
<td>Add $c(mra)$ to $c(n)$ and store the result in the mra</td>
<td>14</td>
</tr>
<tr>
<td>su n</td>
<td>Subtract $c(n)$ from $c(mra)$ and store the result in the mra</td>
<td>14</td>
</tr>
<tr>
<td>on n</td>
<td>Clear the mra, and put the positive magnitude of $c(n)$ in the mra</td>
<td>14</td>
</tr>
<tr>
<td>mr n</td>
<td>Multiply $c(mra)$ by $c(n)$, store the result in the mra</td>
<td>14</td>
</tr>
<tr>
<td>sl n</td>
<td>Multiply $c(mra)$ by $2^n$ and store the result in the mra</td>
<td>14</td>
</tr>
<tr>
<td>sr n</td>
<td>Multiply $c(mra)$ by $2^{-n}$ and store the result in the mra</td>
<td>14</td>
</tr>
</tbody>
</table>
DIGITAL COMPUTER LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Operations on real (15,0,c) fixed-point single register numbers (Interpretive)

Notes:
1. Entering and leaving subroutine
   Both entering and leaving the subroutine are accomplished by the instruction $sp\ ax$, where $vx$, a preset parameter, is the address in storage of the first register of the interpretive subroutine. When used to enter the subroutine, the first instruction interpreted is that following the instruction $sp\ ax$ in storage. When used to leave the subroutine, ordinary $w$ operation is resumed at the register following the instruction $sp\ ax$.

2. Accuracy
   All of the operations executed by the subroutine are carried out with a 15 binary digit accuracy.

3. Calculation with integers
   If $c = 15$, the (15,0,c) numbers dealt with by the subroutine are binary integers.

4. Reasons for machine stoppage during the subroutine
   a) Arithmetic overflow at $35r$ - overflow in a $sr$ or $ar$ difference for the interpreted instructions $ad$ or $su$.
   b) Various alarms can result from trying to interpret an instruction which was not meant to be interpreted by the subroutine.

5. Overflow during $mr$
   It should be noted that overflows can occur during multiplication of (15,0,c) numbers. These will not cause an alarm since the overflow is shifted out of the left hand end of the accumulator, i.e., the product is formed modulo $2^c$.

6. In any of the operations the address $n$ may be the address of the mra.

7. The mra consists of register $2r$ (or $2ax$).

PCH 12/6/51  AS 2/5/52
Operations on real \((15,0,0)\) fixed-point single register numbers (Interpretive)  

Abstract: PA 8.1 is a \((15,0,0)\) interpretive subroutine which performs the instructions \(ts, td, ta, ex, cp, sp, ca, cs, ad, su, cm, mr, sl\) and \(sr\). Exit and entry to the subroutine are accomplished by the instruction \(spax\). In leaving the subroutine, if \(C(m) = spax\), then ordinary \(WW\) operation is resumed at register \(m + 1\).

Preset Parameters:
- \(vx1\): \(M\) address in storage of the initial register of the storage block for \((15,0,0)\) numbers.
- \(vx3\): \(C\) where \(C\) is the number of binary digits to the left of the binary point
- \(vx\): \(N\) where \(N\) is the address in storage of the first register of the interpretive routine

In the description given below the subroutine is assumed to be executing the order \(C(m) = xxx\)

```
00 ta 38r  { Enter interpretive subroutine }
01 sp 33r
02 (p0) mra
46r -> 03 sr 18r ca
46r -> 04 ca ax cs
46r -> 05 sp 34r ad
46r -> 06 sp 34r su
46r -> 07 sp 34r cm
32r -> 08 cs 2r
09 cp 37r cp
10 sp 16r
46r -> 11 mh 2r
12 sl0ax3 mr
13 sp 36r
46r -> 14 sp 34r sl
10r,33r -> 16 ao 38r
17 td 24r
18 ca 45r
19 td 38r
20 su 4r
21 ts 35r
22 cm 35r
23 su0
(17r) 24 cp0
46r -> 25 sp 38r
26 sp 34r ts
46r -> 27 sp 34r td
46r -> 28 ca 24r ta
(41r) 29 td(0) Transfer m+1 to digit section of register n
30 sp 37r
46r -> 31 sp 34r ex
46r -> 32 sp 8r cp
46r -> 33 sp 16r sp
5r,6r,7r
14r,15r
26r,27r,3ly
(21r)(39r)
13r -> 36 ts 2r
1r,30r -> 37 ao 38r
(Or)(16r)
(37r)(19r)
1r, 25r
39 ts 35r Store xxx
40 td 45r Store n
41 td 29r
42 sr* 11
43 ad 3r
44 td 46r
(40r) 45 ca(0) Put C(n) in AC
(44r) 46 sp(0) Enter table
```

Is \(xxx=spax\)?
**DIGITAL COMPUTER LABORATORY**

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

**WHIRLWIND SUBROUTINE SPECIFICATION**

<table>
<thead>
<tr>
<th>TITLE:</th>
<th>Form $\sin \frac{\pi}{2} x$ from $x$ Stored in AC, and Leave Result in AC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LSR TF 0.1 t</td>
</tr>
<tr>
<td></td>
<td>Classification Closed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Sub.</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>$d - 1t$</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

Program Parameters on entering Subroutine
- ac: $x$
- ar: return address

Results on leaving Subroutine
- ac: $\sin \frac{\pi}{2} x$

Description

The subroutine gives $\sin \frac{\pi}{2} x = ax - bx^3 + cx^5 - dx^7$ for $-1 < x < 1$. The subroutine is entered with $x$ in the accumulator and on returning to the main program, $\sin \frac{\pi}{2} x$ is in the AC. The maximum error is approximately $\pm 0.00005$ and the average error is $\pm 0.00003$. 
TITLE: Form $\sin \frac{\pi}{2} x$ from $x$ Stored in AC, and Leave

Result in AC.

Abstract: This subroutine gives $\sin \frac{\pi}{2} x = ax - bx^3 + cx^5 - dx^7$ as a single
length fixed point number in the accumulator where $-1 < x < 1$.

Entering the subroutine:
ac: $x$

Leaving the subroutine:
ac: $\sin \frac{\pi}{2} x$

Temporary Storage:
d unused
lt used to store the value of $x$

```
00 ta 14r Set return address
01 ts lt Store x
02 mh lt $x^2$
03 mh 15r $-dx^2$
04 ad 16r $a = dx^2$
05 mh lt $cx - dx^3$
06 mh lt $cx^2 - dx^4$
07 ad 17r $-b + cx^2 = dx^4$
08 mh lt $-bx + cx^3 - dx^5$
09 mh lt $-bx^2 + cx^4 - dx^6$
10 sr *1 $(-bx^2 + cx^4 - dx^6)_2 = 1$
11 ad 18r $(a - bx^2 + cx^4 - dx^6)_2 = 1$
12 mh lt $(ax - bx^2 + cx^4 - dx^7)_2 = 1$
13 sl 1 $ax - bx^2 + cx^4 - dx^7$
14 sp (0) Return to main program
15 1,77560 =d
16 0,05055 c
17 1,26521 =b
18 0,62210 $a \times 2^{-1}$
```
### DIGITAL COMPUTER LABORATORY

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

**WHIRLWIND SUBROUTINE SPECIFICATION**

<table>
<thead>
<tr>
<th>TITLE: Form cosine $\frac{\pi}{2} y$ from $y$ Stored in AC, and Leave Result in AC (15,0,0)</th>
<th>LSR TF 1.1t</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Regs. in subroutine</td>
<td>Temp. Regs. used by Sub.</td>
</tr>
<tr>
<td>26</td>
<td>d $-1t(2)$</td>
</tr>
<tr>
<td>Average Time (operations)</td>
<td>Max. Time (operations)</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

**Program Parameters**
on entering Subroutine
ac: $y$
ar: Return address

**Results**
on leaving Subroutine
ac: cosine $\frac{\pi}{2} y$

**Description**
Gives cosine $\frac{\pi}{2} y$ by changing $y$ to $x$ so that
$$\cos \frac{\pi}{2} y = \sin \frac{\pi}{2} x = ax - bx^3 + cx^5 - dx^7$$
for $-1 < x < 1$. If $y=0$, there will be an overflow since the cosine of zero is one. The subroutine is entered with $y$ in the accumulator and on returning to the main program cosine $\frac{\pi}{2} y$ is in the accumulator. The maximum error is approximately $\pm 0.00005$ and the average error $\pm 0.00002$.

**Notes**
1. If $y$ is $\pm$zero, an arithmetic overflow will occur in register 5r.

**DMJ+DGA JWC III MAS**

| 1/14/52 | 1/16/52 | 1/22/52 |
TITLE: Form \( \cos \frac{\pi}{2} \) from \( y \) Stored in AC, and Leave Result in AC (15,0,0) 
LSR# TF 1,1t 
Tape 705-1

Abstract: Gives \( \cos \frac{\pi}{2} \) by forming \( x=1-|y| \) so that \( \cos \frac{\pi}{2} = \sin \frac{\pi}{2} \) and evaluating
\[ \sin \frac{\pi}{2} = ax - bx^3 + cx^5 - dx^7, \quad -1 < x < 1. \] If \( y=0 \)
there will be an overflow at register 5r.

Upon entering the subroutine:
\( ac: y \)

Upon leaving the subroutine:
\( ac: \cos \frac{\pi}{2} \)

Temporary registers:
\( d \) - unused
\( lt \) - used to store \( x \)

```
00 ta 19r  Set return address
01 ts 1t   Transfer y to lt
02 cp 4r   y negative? Yes
03 cs 1t   No. -y in ac
2r ----> 04 ad 20r \((1-2^{-15}) \cdot -y\)
05 ad 21r  Add \(2^{-15}\) \(\text{possible overflow}\)
06 ts 1t   \(1-|y|=x\) in lt
07 mh 1t   \(x^2\)
08 mh 22r  \(-dx^2\)
09 ad 23r  \(c=-dx^2\)
10 mh 1t   \(cx - dx^3\)
11 mh 1t   \(cx^2 - dx^4\)
12 ad 24r  \(-b+cx^2 - dx^4\)
13 mh 1t   \(-bx + cx^3 - dx^5\)
14 mh 1t   \(-bx^2 + cx^4 - dx^6\)
15 sr*1   \((a-bx^2 + cx^4 - dx^6)2^{-1}\)
16 ad 25r  \((a-bx^2 + cx^4 - dx^6)2^{-1}\)
17 mh 1t   \((ax-bx^3 + cx^5 - dx^7)2^{-1}\)
18 s1 l    \(ax-bx^3 + cx^5 - dx^7\)
19 sp(0)   Return to main program
```

Return to main program
**DIGITAL COMPUTER LABORATORY**

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

**WHIRLWIND SUBROUTINE SPECIFICATION**

**TITLE:** Form \( \frac{d}{dx} \) from \( x \) Stored in AC, and/or
Form Cosine \( \frac{d}{dx} y \) from \( y \) Stored in AC,
Leave Result in AC, (15,0,0)

<table>
<thead>
<tr>
<th>No. of Regs. in Subroutine</th>
<th>Temp. Regs. used by Sub.</th>
<th>Average Time (operations)</th>
<th>Max. Time (operations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>( d \cdot 1t )</td>
<td>15 - sine</td>
<td>15 - sine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 - cosine</td>
<td>20 - cosine</td>
</tr>
</tbody>
</table>

**Program Parameters**
on entering Subroutine
ac: \( x \) or \( y \)
ar: return address

**Results**
on leaving Subroutine
ac: sine \( \frac{d}{dx} x \) or cosine \( \frac{d}{dx} y \)

**Description**
If this subroutine is entered at register Or, it will
calculate cosine \( \frac{d}{dx} y \) by changing \( y \) to \( x \) so that cosine \( \frac{d}{dx} y = 
\text{sine} \frac{d}{dx} x \) and evaluating sine \( \frac{d}{dx} x = ax - bx^3 + cx^5 - dx^7, -1 < x < 1.\)
If it is entered at register 7r, it will calculate sine \( \frac{d}{dx} x.\)
There will be an overflow at register 5r if \( y = 0.\) The subroutine is entered with either \( x \) or \( y \) in the accumulator, and
on returning to the main program either sine \( \frac{d}{dx} x \) or cosine \( \frac{d}{dx} y \)
is in the accumulator. The maximum error is approximately
\( \pm 0.0005 \) and the average error is \( \pm 0.00002.\)

**Notes:**
1. Enter at Or if cosine \( \frac{d}{dx} y \) is desired
   enter at 7r if sine \( \frac{d}{dx} x \) is desired
2. If \( y = \pm 0, \) an overflow will occur in register 5r.

**Dates:**

<table>
<thead>
<tr>
<th>DNFDGA</th>
<th>JWC III</th>
<th>DLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/14/52</td>
<td>1/16/52</td>
<td>1/18/52</td>
</tr>
</tbody>
</table>
TITLE:  FORM SINE \(\frac{\pi}{2^x}\) FROM \(x\) STORED IN AC, AND/OR FORM COSINE \(\frac{\pi}{2^y}\) FROM \(y\) STORED IN AC, LEAVE RESULT IN AC, (15,0,0)

Abstract: If this subroutine is entered at register 0r, it will calculate cosine \(\frac{\pi}{2^x}\) by changing \(y\) to \(x\) so that cosine \(\frac{\pi}{2^y} = \text{sine} \frac{\pi}{2^x}\) and evaluating sine \(\frac{\pi}{2^x} = ax-bx^3+cx^5-dx^7\), -1<\(x<1\). If it is entered at register 7r, it will calculate sine \(\frac{\pi}{2^x}\).

There will be an overflow at register 5r if \(y=0\).

Upon entering the subroutine:
ac: \(x\) or \(y\)

Upon leaving the subroutine:
ac: sine \(\frac{\pi}{2^x}\) or cosine \(\frac{\pi}{2^y}\).

Temporary registers:
d = unused
1t = used to store \(x\)

Sine --- 07 ta 21r Set return address (0r)(7r) 21 sp (0) Return to main program

08 ts 1t x in 1t
09 mh 1t \(x^2\)
10 mh 24r -dx^2
11 ad 25r c -dx^2
12 mh 1t ox -dx^3
13 mh 1t ox^2 -dx^4
14 ad 26r -b+ox^2 -dx^4
15 mh 1t -bx + ox^3 -dx^5
16 mh 1t -bx^2 + ox^4 -dx^6
17 sre 1 (-bx^2+cx^4 -dx^6)2 -1
18 ad 27r (a-bx^2+cx^4 -dx^6)2 -1
19 mh 1t (ax-bx^3+cx^5 dx^7)2 -1
20 s1 1 ax -bx^3+cx^5 -dx^7
22 0.77777 1 -2^-15
23 0.00001 2^-15
24 1.77560 -d
25 0.05055 c
26 1.26521 -b
27 0.62210 a x 2^-1

Cosine --- 00 ta 21r Set return address
01 ts 1t y in 1t
02 cp 4t Is \(y\) negative? Yes
03 cs 1t No. \(-y\) in ac
2r --- 04 ad 22r (1 -2^-15) -|y| in ac
05 ad 23r 1 -|y| = x in ac Possible overflow
06 sp 8r