

DIGITAL COMPUTER LABORATORY
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY
 WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: AC Print, Octal Number, Sign, Single Column		LSR # OT 1.1 t	
		Classification Closed	
No. of Regs. in Subroutine	Temp. Regs. used by Subroutine	Average Time (operations)	Max. Time (operations)
35	1t, 2t	61.5	63.0
Preset Parameters			
none			
Program Parameters on entering Subroutine			
ac: Number to be printed			
Results on leaving Subroutine			
ac: 0			
Description			
<p>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.</p> <p>Notes:</p> <ol style="list-style-type: none"> 1. No point is printed. 2. The qp orders are set to suppress the punch. 			
MD, Nov.9,1951 JWC III, Nov.13,1951			

DL-291

TITLE: AC Print, Octal Number, Sign, Single Column LSR # OT 1.1 t

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	17	ca 4r	Yes. Cause a
01	ts 1t	Store value	18	qp 128	carriage return
02	<u>cp 20r</u>	Is word negative?	19	<u>(sp 0)</u>	Go back to main
03	ca 26r	No. Print "0"	20	ad 24r	program
23r → 04	qp 144	Printing	→ 20	ad 24r	Change sign
05	cs 25r	} Set digit	21	ts 1t	Store value
06	ts 2t	} counter	22	ca 27r	Print "1"
16r → 07	ca 1t	Value in AC	23	<u>sp 4r</u>	
08	sr*12		24	0.77777	
09	ad 34r	} Add start of	25	p4	Initial value of counter
10	td 13r	} number table	26	p45	
11	s1 15	} Store	27	p36	
12	ts 1t	} remainder	28	p39	
13	(ca 0)	} Put flexo code for	29	p3	Number
		digit in AC	30	p21	Table
14	qp 128	Print digit	31	p33	
15	ao 2t	} Have all digits been	32	p43	
16	<u>cp 7r</u>	} printed? No.	33	p15	
			34	p26r	

DIGITAL COMPUTER LABORATORY
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY
 WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: AC Print and/or Punch, Octal Number (Magnitude), Sign, No Carriage Return			LSR # OT 1.2 t
			Classification Closed
No. of Regs.	Temp. Regs.	Average Time (operations) 7 print	Max. Time (operations) 7 print
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			
<p>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.</p>			
Notes:			
<ol style="list-style-type: none"> 1. No point is printed 2. No carriage return or space is printed 3. The <u>op</u> instructions are normally set to print and suppress the punch. 			

TITLE: AC PRINT AND/OR PUNCH, OCTAL NUMBER (MAGNITUDE), LSR# OT 1.2 t
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 → -65432 respectively.

Preset Parameter:

v1: p0 to print (does not need to be inserted), or p64 to punch, or p128 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→00	ta 17r	Set return address	(0r)17	sp (0)	Return to main program
01	ts 1t	Store number	2r→18	cm 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
21r→04	qpl28s1	Print sign	21	sp 4r	"-" in AC
05	sr *12	} Set digit	22	p25r	Start of number table
06	ts 2t	} Counter	23	1.30043	"+" character
16r→07	ca 1t	Value in AC	24	1.30007	"-" character
08	sr *12		25	p45	
09	ad 22r	} Add start of	26	p36	
10	td 13r	} number table	27	p39	Number
11	sl 15	} Store	28	p3	
12	ts 1t	} Remainder	29	p21	Table
(10r)13	ca (0)	} Put Flexo code for digit in AC	30	p33	
14	qpl28s1	Print digit	31	p43	
15	ao 2t	} Have all digits	32	p15	
16	_cp 7r_	} been printed?			

DIGITAL COMPUTER LABORATORY
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY
 WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print C(AC) as Octal Number, Sign Digit and Complement, Point, Single Column Layout.			LSR OT 1.3t
			TAPE T542-3
			Classification Closed
No. of Regs. in Subroutine 36	Temp. Regs. used by Sub. 3	Average Time (operations) 8 print	Max. Time (operations) 8 print
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: 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.			

MSD 1/23/52	AS 2/5/52
----------------	--------------

TITLE: PRINT C(AC) AS OCTAL NUMBER, SIGN DIGIT AND
COMPLEMENT, POINT, SINGLE COLUMN LAYOUT

LSR: OT 1.3t
TAPE T 542-3

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:

v1: p0 (does not need to be inserted) to print, or p64 to punch,
or pl28 to punch and print simultaneously.
v2: (desired digit length) - 5.

Upon entering the subroutine:

AC: number to be recorded

Temporary registers:

d: unused

lt: register used to store the remainder of the word.

2t: register used to store the digit counter.

00	ta 20r	Set return address	18	ca 15r	Print a carriage return
01	ts lt	Store number	19	qp 128sl	
02	<u>op 21r</u>	Is number positive?	(Or) 20	<u>sp 0</u>	Return to main program
03	ca 27r	Put Flexo code for "0" and point in AC	2r → 21	ad 25r	Change sign
24r → 04	qp 134sl	Print {"0"}, shift	22	ts lt	Store number
05	qp 128sl	Print point	23	ca 28 r	Put flexo code for "1" and point in AC
06	cs 35r	Set digit counter	24	<u>sp 4r</u>	
07	ts 2t		25	0.77777	
17r → 08	ca lt	Number in AC	26	p27r	Start of number table
09	sr* 12		27	0.07155	". and "0"
10	ad 26r	Add start of number table	28	0.07144	". and "1"
11	td 14r		29	p39	
12	sl 15	Store remainder	30	p3	
13	ts lt		31	p21	Number table
(11r) → 14	ca (0)	Put Flexo code for digit in AC	32	p33	
15	qp 144sl	Print digit	33	p43	
16	ao 2t	Have all digits been printed?	34	p15	
17	<u>op 8r</u>		35	p4a2	Value for digit counter

DIGITAL COMPUTER LABORATORY
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY
 WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print C(AC) as Decimal Fraction, Sign and Magnitude, Point, Single Column Layout.			LSR OT 2.2t Tape T664-3
			Classification Closed
No. of Regs. in Subroutine 38	Temp. Regs. used by Sub. d - 2t (3)	Average Time (operations) 8 print	Max. Time (operations) 8 print
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 1/18/52	JWCIII 1/21/51	AS 1/23/52	

TITLE: Print C(AC) as Decimal Fraction,
Sign and Magnitude, Point, Single
Column Layout.

LSR# OT 2.2t
Tape T664-3

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 pl28 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	<u>op 21r</u>	Is number positive?	22	sp 4r	Return to print
03	ca 25r	{ Put Flexo code for "+" and dec. point in AC	23	p 10	10×2^{-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	cs 37r	} Set digit counter	26	0.07107	-. character
07	ts 2t	}	27	p 45	
17r →	08 cm lt	x in AC	28	p 36	
09	mh 23r	$ x \cdot 10 \times 2^{-15}$	29	p 39	
10	ad 24r	} Add start of	30	p 3	
11	td 14r	} number table	31	p 21	Number table
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	ac 2t	} Have all digits	36	p 49	
17	<u>cp 8r</u>	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		LSR# OT 2.5 t	
		Classification Closed	
No. of Regs. in Subroutine 45	Temp. Regs. used by Sub. 4	Average Time (operations) 5 print	Max. Time (operations) 5 print
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: $d_1 = \left[\frac{x \cdot 2^{-15}}{10^4} \right] = \left[\frac{x}{10^4} \right] \cdot 2^{-15}$ where $\left[\quad \right]$ = "integral part of" $d_2 = \left(\frac{x \cdot 2^{-15}}{10^4} - \left[\frac{x}{10^4} \right] 10^4 \cdot 2^{-15} \right) \left(2^4 \right) \left(\frac{10}{2^4} \right)$ $= \left[\frac{x - \left[\frac{x}{10^4} \right] \cdot 10^4}{10^3} \right] \times 2^{-15}$ 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			

TITLE: Print Magnitude as Decimal Integer from AC, Initial Zero Suppression, No Carriage Return	LSR# OT 2.5 t
---	---------------

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 :

63
17352
1
(zero)
27

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

TITLE: Print Magnitude as Decimal Integer from LSR# OT 2.5 t
 AC, Initial Zero Suppression, 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_i are obtained as follows:

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

where $\left[\quad \right]$ = "integral part of"

$$d_2 = \left[\frac{\frac{x \cdot 2^{-15}}{10^4} - \left[\frac{x}{10^4} \right] \cdot 10^4 \cdot 2^{-15}}{2^4} \right] = \left[\frac{x - \left[\frac{x}{10^4} \right] \cdot 10^4}{10^8} \right] \cdot 2^{-15}$$

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

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

Temporary Registers

d unused
 lt temporary storage
 2t digit counter
 3t suppressor

Enter→00	ta 14r	Set return address	06	cm 21r	$x \cdot 2^{11}/10^4$
01	ts 21r	Store number in AC	07	mh 32r	
02	cs 31r	Set digit	27r→08	sr *11	$x \cdot 2^{-11}$
03	ts 2t	Counter	09	ts 21r	and store
04	ca 0	Set zero	10	sl *15	Subtract off
05	ts 3t	Suppressor	11	ts lt	Integral part

TITLE: Print Magnitude as Decimal Integer from LSR# OT 2.5 t
 AC, Initial Zero Suppression, No Carriage
 Return

12	<u>ao 2t</u>	Advance	17r-28	<u>ca 29r</u>	Print Space
13	<u>cp 15r</u>	Counter	29	<u>qpl36s1</u>	
(0r)14	<u>sp (0)</u>	Return to main program	30	<u>sp 25r</u>	
13r-15	<u>ca 21r</u>	Check for	31	p5	Counter
16	<u>su 3t</u>	Initial	32	0.15067	$\approx 2^{11}/10^4$
17	<u>ep 28r</u>	Zero	33	ca 35r	Table entry
18	<u>ca 21r</u>	No initial zero; Add in table.	34	0.50000	10/16
19	<u>ad 33r</u>	entry	35	p45	
20	<u>ts 21r</u>	Set	36	p36	Table of
(lr)(20r)21	<u>(p0)</u>	Printer	37	p39	Decimal
22	<u>qpl28s1</u>	Print	38	p3	Flexowriter
23	<u>cs 0</u>	Eliminate	39	p21	
24	<u>ts 3t</u>	Zero Suppressor	40	p33	Characters
30r--25	<u>ao 1t</u>	Round-off	41	p43	
26	<u>mh 34r</u>	Multiply by $\frac{1}{6}$	42	p15	
27	<u>sp 8r</u>	Recycle	43	p13	
			44	p49	

DIGITAL COMPUTER LABORATORY
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY
 WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print C(AC) as Decimal Integer, Magnitude only, Initial Zero Suppression, Print Final Zero, No Layout			LSR OT 2.51t
			TAPE T-883
			Classification Closed
No. of Regs. in Subroutine 49	Temp. Regs. used by Sub. 4 (d - 3t)	Average Time (operations) 5 print	Max. Time (operations) 5 print
Preset Parameters vl p0 (need not be inserted) to print, or p64 to punch, or pl28 to punch and print simultaneously			
Program Parameters on entering Subroutine AC: 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 →→→0 (zero) 27 3. The program resets automatically. 4. There is no carriage return.			
JWC III 2/10/52	MAS 2/11/52		

TITLE: Print C(AC) as Decimal Integer, Magnitude only,
 Initial Zero Suppression, Print Final Zero, No
 Layout

LSR OT 2.51t
 TAPE T-883

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 of the equivalent decimal integers d_i are obtained as follows:

$$d_1 = \left[\left(x \cdot 2^{-15} \right) \left(\frac{2^{11}}{10^4} \right) \left(2^{-11} \right) \right] = \left[\frac{x}{10^4} \right] \cdot 2^{-15}$$

where $\left[\quad \right] = \text{"integral part of"}$

$$\begin{aligned} d_2 &= \left[\frac{x \cdot 2^{-15}}{10^4} - \left[\frac{x}{10^4} \right] 10^4 \cdot 2^{-15} \right] \left(\frac{2^4}{2^4} \right) \left(\frac{10}{2^4} \right) \\ &= \left[\frac{x - \left[\frac{x}{10^4} \right] \cdot 10^4}{10^3} \right] \cdot x \cdot 2^{-15} \end{aligned}$$

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.

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 pl28 to print and punch

00	ta 19r	Set return address	14	ca 25r	Last digit
01	ts 25r	Set no. in AC	15	ad 37r	Add in start of table
02	cs 35r	} Set digit counter	16	ts 17r	
03	ts 2t	}	(16r)	17 (p0)	} Print
04	ca 0	} Set zero suppressor	18	qp 128s1	
05	ts 3t	}	(Or)	19 sp ()	Return to main program
06	cm 25r	$x \cdot \frac{2^{11}}{10^4}$	13r → 20	ca 25r	} Not at last digit
07	mh 36r	}	21	su 3t	} Check for initial zero.
31r → 08	sr* 11	$x \cdot 2^{-11}$	22	cp 32r	
09	ts 25r	}	23	ad 37r	} Add in table entry
10	sl* 15	} Subtract off integral	24	ts 25r	
11	ts 1t	part	(24r)	25 (p0)	Set Printer
12	ao 2t	} Advance counter	26	qp 128s1	Print
13	cp 20r	} Are we at last digit?	27	cs 0	} Eliminate

TITLE: Print C(AC) as Decimal Integer, Magnitude only,
Initial Zero Suppression, Print Final Zero, No
Layout

LSR OT 2.51t
TAPE T-883

28	ts 3t	zero suppressor	39	p45	}	
34r	→ 29	ao 1t	Round off	40		p36
	30	mh 38r		41		p39
	31	<u>sp 8r</u>		42		p3
22r	→ 32	ca 33r		43		p21
	33	qp 136s1	Print space	44		p33
	34	<u>sp 29r</u>		45		p43
	35	p4		46		p15
	36	0.15067	<u>11</u>	47		p13
	37	// ca 39r	<u>2/4</u>	48	p49	
	38	0.50000	<u>10</u>			
					}	
					Table of Digits	

DIGITAL COMPUTER LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Print Out Flexowriter Characters, Previously Stored Three Characters to a Register by LSR OT 3.lat, Using Program Parameter		LSR OT 3.lt	
		TAPE T782-2	
		Classification Closed	
No. of Regs. in Subroutine 38	Temp. Regs. used by Sub. $d = 4t \quad (5)$	Average Time (operations) $13 + 14n \quad (n=\text{no. of chars.})$ (Actually limited by number of printouts)	Max. Time (operations) $13+20n \quad (n=\text{no. of chars.})$ (Actually limited by no. of printouts)
Preset Parameters			
vl: p0 (does not need to be inserted) to print, p64 to punch, and pl28 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.lat, three to a register. Subroutine OT 3.lat 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.lat, see the Specification sheet for LSR OT 3.lat). 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.lat.			
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.lat.			
JWC III 2/1/52	AS 2/5/52		

TITLE: Print Out Flexowriter Characters,
Previously Stored Three Characters
to a Register by LSR OT 3.lat, Using
Program Parameter

LSR# OT 3.lt
TAPE# T782-2

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 pl28 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 lr	Plant ul address	19	sl* 15	}	Shift off left and restore reg. contents
(3r)(Or)	01 ca (0)	Plant first address	20	ts 2t	}	
02	td 1lr		21	ca d		
03	ao lr	Set return address	22	sl* 1	}	Add in final digit
04	td 13r		23	ad 3t		
05	ca 0	Set final digit	24	qp 128sl		Print
06	ts 3t		35r →	25	ao 1t	Are we finished with registers?
07	ca 36r	Set digit to be exchanged to pl	26	<u>op 14r</u>		
08	ts 4t		27	ao 1lr		Advance register no. and recycle
28r →	09 cs 37r	Set 3-counter	28	<u>sp 9r</u>		
10	ts 1t		17r →	29	ca 2t	
(2r)	11 ca (0)	Pick up register to be printed	30	sl* 5		Restore register contents shifted.
12	ts 2t		31	ts 2t		
(4r)	13 <u>op (0)</u>	Are we finished?	32	ca 4t		
26r →	14 ca 2t	No. Shift character into position	33	ex 3t		Exchange final digit
15	sr* 10		34	ts 4t		
16	su 0	Is this character a zero?	35	<u>sp 25r</u>		Back to check finish final digit
17	<u>op 29r</u>		36	pl		
18	ts d	No	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 OT 3.lt.)			LSR OT 3.lat
			TAPE T781-2
			Classification Closed
No. of Regs. in Subroutine 66	Temp. Regs. used by Sub. d - 6t (7)	Average Time (operations) $40 + 13n$, where n is no. of characters (Present limiting case is no. of read-ins on PETR.)	Max. Time (operations) $40 + 26n$ (in the most unusual case) (Present limiting case is no. of read-ins on PETR.)
<p>Program Parameters on entering Subroutine ar: u_1 (automatic)</p> <p>utx p (address of first register in which characters are to be stored)</p>			
<p>Results: on leaving Subroutine</p> <p>Flexowriter characters stored, five binary digits to the character, beginning at the register given in u_1, three characters to the register. All registers are positive except the one succeeding the final storage register, which is set to n0.</p>			
<p>Description</p> <p>This subroutine reads in Flexowriter characters continuously from the photoelectric reader, and stores them in the registers beginning at the address stored in u_1. 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.</p> <p>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.</p>			
<p>Notes</p> <p>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</p>			

TITLE: Read in Standard Flexowriter Characters and Store Three Characters to a Register, Using Program Parameter. (For Use with LSR OT 3.lt.) LSR OT 3.lt
TAPE T781-2

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.lt. It may be used in the following manner: Read in subroutine OT 3.lt, 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.lt 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.lt.

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.

JWC III 1/24/52	AS 2/5/52
--------------------	--------------

TITLE: Read in Standard Flexowriter Characters
and Store Three Characters to a Register,
Using Program Parameter. (For Use with
LSR # OT 3.lt.)

LSR# OT 3.1at
TAPE# T781-2

Upon entering the subroutine:

sp: subroutine

p: first address to be stored

Temporary registers:

d: temporary digit store

lt: last 6th digit representation

2t: character store

3t: present 6th digit representation

4t: temporary store

5t: 3-counter

6t: temporary store

00	ta lr	Plant ul address	33r → 20	ca 2t	
(3r)(0r)01	ca ()	Plant 1st address where	21	sl* 14	
02	td 53r	machine is to store chars.	22	sr* 14	Store present 6th digit representation
03	ao lr	} Set return address	23	su0	
04	td 42r		24	ts 3t	
05	cs0	} Set last digit reg. in	25	mh lt	Are past and present
06	ts 1t	for zero	26	op 43r	6th digits the same?
07	os 64r	} Store 3 counters	47r → 27	ca 2t	Store 5 digits of
08	ts 5t		28	sp 48r	character
09	ca0	} Set zero in temp	29	sp 11r	Recycle
10	ts 4t	store	19r → 30	ca 65r	
29r → 11	qr0	Read in	31	su 2t	Is char. a "stop"?
12	ca 8		32	op 34r	
13	sr* 5		33	sp 20r	No. Back to check digits
14	td d	} Store 6 digits in 2t	32r → 34	ca 0	Yes.
15	ca d		35	sp 48r	Store 00000
16	sr* 5		36	ca 0	Store 00000
17	ts 2t		37	sp 48r	
18	su 65r	} Check to see if	38	ca 53r	Plant last register
19	op 30r	char. is a "stop"	39	td 41r	address

TITLE: Read in Standard Flexowriter Characters
and Store Three Characters to a Register,
Using Program Parameter. (For Use with
LSR # OT 3.lt.)

LSR# OT 3.lat
TAPE# T781-2

40 cs 0 } Put negative no.	56 ao 5t } Are we at end
39r → 41 ts (0) } in last reg.	57 cp 63r } of register?
(04r) → 42 sp (0) Back to main Program	58 ca 0 } Yes. Reset temp.
26r → 43 ca 3t } Switch digit	59 ts 4t } store.
44 ts 1t } representations	60 es 64r } Reset 3-counter
45 ca 0 } Store 00000	61 ts 5t } Advance Storage
46 sp 48r } 57r	62 ao 53r } Exit from internal s.r.
47 sp 27r } Back to store 5 digits (43r)	63 sp (0) } 3-counter reset
28r, 37r } 48 ta 63r Plant link of internal	64 p2 } "stop" character
35r, 46r } 49 sr* 1 subroutine	65 p51 }
50 ts 6t } Shift left and store	
51 ca 6t } 52 ad 4t }	
(62r)(02r) 53 ts (0) }	
54 sl* 5 }	
55 ts 4t }	

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)			LSR OT 103.10t
			TAPE T799-1
			Classification Closed, Interpreted
No. of Regs. in Subroutine 76	Temp. Regs. used by Sub. 4	Average Time (operations) $94 + 19n$, number of digits printed = n	Max. Time (operations) $94 + 27n$
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			
$\begin{matrix} + \\ - \end{matrix} . d_1 d_2 \dots d_n$			
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 p10.			
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.			
FCH 2/1/52	MRS 2/8/52		

TITLE: MRA Print and/or Punch in (30,0,0) Interpretive LSR OT 103.10t
 Subroutine without Sign Agreement, Decimal Number, TAPE T799-1
 Sign, Number of Digits Arbitrary, No Carriage
 Return (Interpreted)

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: pN, where N is the address in storage of the first register of the interpretive subroutine (in title of main program)
 v1: 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	.35r → 17 cm 1t	
01	<u>sp ax</u>	resume ordinary mn operation	18 mh 62r
02	ca 3ax		19 ts 3t
03	ts 2t	Store C(mra) in 1t and 2t	20 s1 15
04	ca 2ax		21 ts 1t
58r → 05	ts 1t		22 cm 2t
54r → 06	mr 2t	Do 1t and 2t agree in sign?	23 mh 62r
07	<u>cp 38r</u>		24 ts 2t
08	ca 1t		25 s1 15
56r → 09	<u>cp 12r</u>	Sense and print algebraic	26 ex 2t
10	ca 74r	sign of C(mra) followed	27 sa 1t
11	<u>sp 13r</u>	by a decimal point.	28 ts 1t
9r → 12	ca 75r		29 ca 3t
1r → 13	qp 134s1		30 ad 63r
14	qp 128s1	Set up entry into table	31 td 32r
15	cs 61r		(31r) 32 ca (0)
16	ts 4t	Set up digit counter	33 qp 128s1
			Print a single digit

TITLE: MRA Print and/or Punch in (30,0,0) Interpretive
 Subroutine without Sign Agreement, Decimal Number,
 Sign, Number of Digits Arbitrary, No Carriage
 Return (Interpreted)

LSR OT 103.10t
 TAPE T799-1

34	ao 4t	Have enough digits been printed?	48r	→	55	ao lt	
35	<u>cp 17r</u>		56	<u>sp 9r</u>			
36	<u>sp ax</u>	Return control to (0r) 37 <u>sp (0)</u>	40r	→	57	cs lt	
			58	<u>sp 5r</u>			
7r	→	38 cm 1t	59	ea0			
		Is C(1t) ≠ 0 ?	60	p1			
			61	nla2	v2		
		40 <u>cp 57r</u>	62	p10			
		41 cm 2t	63	p64r			
		Is C(2t) ≠ 0 ?	64	p45	0	Table	
			65	p36	1		
		42 su 0	66	p39	2		
		43 <u>cp 51r</u>	67	p3	3		
			68	p21	4		
		44 su 60r	69	p33	5		
		45 ad 59r	70	p43	6		
			71	p15	7		
		46 ts 2t	72	p13	8		
		47 ca 1t	73	p49	9		
			74	0.07143	+		
		48 <u>cp 55r</u>	75	0.07107	-		
		49 su 60r					
		50 ts 1t					
43r	→	51 cs 2t					
		52 ts 2t					
		53 ca 1t					
		54 sp 6r					

PROGRAMMED ARITHMETIC

PA 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 97	Temp. Regs. used by Sub. -----	Average Time (operations) see description	Max. Time (operations) 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 <u>n</u> and <u>n+k</u> , where <u>k</u> is specified by a preset parameter. If <u>a</u> is any number such that $\frac{1}{2} \leq x < 1 \text{ or } x = 0,$ then there exists a signed 15 binary digit number <u>x</u> and a signed 15 binary digit integer <u>y</u> such that $\frac{1}{2} \leq x < 1 \text{ or } x = 0,$ $0 \leq y < 2^{15},$ and $\left \frac{a - x \cdot 2^y}{a} \right \leq 2^{-16}.$ <u>x</u> and <u>y</u> are stored respectively in the two registers <u>n</u> and <u>n+k</u> for use by this subroutine. Example (a) Let <u>a</u> = -300 (decimal) Then <u>a</u> = -100101100 (binary) and hence <u>x</u> = 1.011010011111111 and <u>y</u> = 1001 Register <u>n</u> contains 1.011010011111111 and Register <u>n+k</u> contains 0.000000000001001			

TITLE: Operations on Real (15,15,0) Floating Point Double Register Numbers (General Subroutine)	LSR	PA 1.1 t
---	-----	----------

Example (b) Let $a = \frac{1}{10}$ (decimal)

Then $a = 0.000110011001100110011\dots$
and hence $x = 0.110011001100110$,
and $y = -11$
Register n contains 0.110011001100110
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

$$\begin{aligned} & \frac{1}{10} \leq x < 1, \text{ or } x = 0 \\ & 0 \leq |y| \leq 32,767 \\ \text{and } & \left| \frac{a - x \cdot 10^y}{a} \right| \leq 10^{-5} \end{aligned}$$

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 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 sp ax. In the latter case, ordinary WW operation is resumed at the register following the 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.

TITLE: Operations on Real (15,15,0) Floating Point Double Register Numbers (General Subroutine)		LSR PA 1.1 t
<u>Instruction</u>		<u>Function</u>
ts n		Store the number in the mra in registers <u>n</u> and <u>n+k</u> .
ta n		Store the address of the register following the last sp or cp instruction in the address section of register <u>n</u> .
ex n		Exchange the number in the mra with the number in registers <u>n</u> and <u>n+k</u> .
cp n		If the number in the mra is negative, proceed as in sp, otherwise ignore the instruction.
sp n		If <u>n</u> ≠ ax, interpret next the instruction in register <u>n</u> . If <u>n</u> = ax, resume ordinary WW operation at the register following the register containing the <u>sp ax</u> .
ca n		Clear the mra and then add into the mra the number in registers <u>n</u> and <u>n+k</u> .
cs n		Clear the mra and then add the negative of the number in registers <u>n</u> and <u>n+k</u> into the mra.
ad n		Add the number in registers <u>n</u> and <u>n+k</u> to the number in the mra and leave the result in the mra.
su n		Subtract the number in registers <u>n</u> and <u>n+k</u> from the number in the mra and leave the result in the mra.
cm n		Clear the mra and add the magnitude of the number in registers <u>n</u> and <u>n+k</u> into the mra.
mr n		Multiply the number in the mra by the number in registers <u>n</u> and <u>n+k</u> and leave the result in the mra.
dv n		Divide the number in the mra by the number in registers <u>n</u> and <u>n+k</u> and leave the result in the mra.
N.B. In the above the phrase "the number in" should be read to mean "the number represented by the contents of".		

TITLE: Operations on Real (15,15,0) Floating Point Double Register Numbers (General Subroutine)	LSR PA 1.1 t
---	--------------

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, v8, 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 y 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 LS#7 PA 1.1 t
 Register Numbers (General Subroutine)

Instruction Code and Operation Times:

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

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

v8/ pk: k = separation in storage of two registers of numbers
 vx/ pN: N = address in storage of initial register of subroutine

00 ta 76r	Store address of 1st instruction to be interpreted	25 (p0)	digit storage
01 sp 76r		26 pa8	separation parameter
02 (p0)	x ₁ multiple register	89r-27 sp 90r	"ts"
03 (p0)	y ₁ accumulator	28 sp 19r	
89r-24	ca ax "ca"	89r-29 ca 42r "ta"	Store digits in a-register
89r-25	sp 8r "cs"	(78r)30 ta (0)	
89r-26	sp 46r "ad"	31 sp 75r	
89r-27	sp 44r "su"	89r-23 sp 90r "ex"	
5r, 89r-28	ca 86r "cm" Pick up and store cs ca n+k	89r-233 sp 22r "cp"	
09 ts 93r		24r, 89r-234 ao 76r "sp"	Set A-register
10 sp 90r	cm	35 ta 42r	address
11 pl6		36 ca 88r	Set pick-up
89r-212	mr 2r "mr" Form x ₁ • x ₂	37 td 76r	instruction's address
13 sp 69r		38 su 4r	
89r-214	ex 2r "dv" Form and store	39 ts 95r	
15 sr 1		40 cm 95r	Does address of cp
16 dv 2r	x ₁ • 2 ⁻¹	41 su 0	or sp instruction
17 sf 25r	x ₂ • 2 ^{s.f.}	(35r) 42 cp (0)	differs from ax?
18 ex 3r		43 sp 76r	
19 su 95r		7r-244 ts 91r	Complement x ₂
20 ad 11r	Form y ₁ = y ₂ + 16	45 cs 91r	
21 sp 72r		6r-246 ex 95r	Form and store
33r-22	cs 2r Is x ₁ < 0?	47 su 3r	y ₂ - y ₁ .
23 cp 75r		48 ts 91r	
24 sp 34r	Go to sp routine	49 cp 55r	Is y ₂ - y ₁ > 0?

PTIME: Operations on Real (15,15,0) Floating Point Double LSR# PA 1.1 t
 Register Numbers (General subroutine)

50 ad 3r		94r->73 ex 3r	Store result in
51 ts 3r	Interchange	74 ts 2r	proper registers
52 ca 2r	(x_1, y_1) and	23r,31r,5,r->75 ao 76r	Increase address of pick-up
53 ex 95r	(x_2, y_2)	(37r)(0r)1r,43r->76 ca (0)	instruction
54 ts 2r		77 ts 91r	Pick-up instruction
49r->55 cm 91r	Store $ y_2 - y_1 $	78 td 30r	store instruction
56 td 62r		79 td 88r	and digits
57 su 96r	Is $ y_2 - y_1 > 15$?	80 sa 26r	where
58 cp 60r		81 ts 93r	necessary
59 sp 75r	Addition unnecessary	82 td 86r	
58r->60 ca 96r	Set $y_2 = 15$	83 sr *11	form <u>sp</u> to proper
61 ex 95r		84 ad 28r	part of subroutine
(56r)62 sr (0)	Form $x_2 = y_2 - y_1$	85 td 89r	
63 sa 2r	Add	(82r)86 ca (0)	Pick up and
64 ts 2r		87 ts 95r	store y_2
65 ca 0		(74r)88 ca (0)	Pick up x_2
66 ex 2r	Add overflow into	(85r)89 sp (0)	Go to <u>art</u> of subroutine for
67 sr *15	2^{-15} times the	10r,32r->90 ca 2r	particular instruction
68 ad 2r	sum	(77r)91 (p0)	
13r->69 sf 25r	Scale factor	92 ex 3r	
70 ex 3r	and store result	(9r),(Elr)93 (p0)	
71 ac 95r	Form $y_2 + y_1$	94 sp 73r	
21r->72 su 25r	Subtract scale factor	95 (p0)	Temporary storage
		96 p15	

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

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

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* WWI operations
maximum = 76* WWI operations

*per interpreted operation; see page 4 for details

Preset Parameters (Values to be indicated in tape title line)

x | pH: N = address assigned to the initial register of the subroutine
x2 | pk: k = 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 x2.

The 24,6,0 number system represents any real number N, provided that either $N=0$ or $2^{-63} < |N| \leq 2^{-64}$, 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 $1 > |x| \geq .5$ and that $|1 - x^2 N^{-1}| < 2^{-24}$. Thus the number pair x,y represents N to within $\pm 0.000006\%$, 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, $v \cdot 2^w$ to $x \cdot 2^y$, assuming $1 > |v| \geq .5$, $1 > |x| \geq .5$, $w \geq y$, the sum obtained is $\lfloor (v + x \cdot 2^{y-w} \pm 2^{-29}) \cdot 2^z \rfloor \cdot 2^{w-z} = u \cdot 2^{w-z}$, where z is chosen in such a way that $1 > |u| \geq .5$.

The roundoff in mr is made in the 28th digit.

The roundoff in dv 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 $x \cdot 2^y$, 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 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 spax, x being the parameter which specifies the location of the subroutine. Instructions following the first 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 spax. This 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 1.0 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 $1 > |x| \geq .1$ and $-9 \leq Y \leq 9$, and with X having at most 8 decimal digits. For example,

the number 300, which equals $.3 \times 10^3$, is written as $+.3 | +3$;
 the number .01π, which equals $.31415927 \times 10^{-1}$ is written as $+.31415927 | -1$
 the number -1/128, which equals $-.78125 \times 10^{-2}$ is written as $-.78125 | -2$

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 3

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.

<u>Numbers designated by programmer</u>	<u>Storage registers</u>	<u>Comments</u>
address at start of program, usually 32.	main program 2-register numbers, 1st halves	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.
total number of locations = k = parameter x_2 .	subroutine 2-register numbers, 1st halves	
address of start of temporary storage = parameter 0.	temporary storage, 1st halves	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 for the first half and to $n_{max}2$ for the second half.
address of start of main program and of each subroutine and address of first instruction to be performed must be indicated	main program 2-register numbers, 2nd halves	address of 2nd half of last main program number must be less than 530.
	subroutine 2-register numbers, 2nd halves	
	temporary storage, 2nd halves	
	main program	
	subroutines	{ address of last word of last subroutine must be less than 704.
		space available for print subroutine OF 102.1
address of start of interpretive subroutine = 852 = parameter x	interpretive subroutine	

Specifications of WHIRLWIND I LIBRARY SUBROUTINE Number PA 2.2 Page 4

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 n" is used to signify the number represented in 24,6,0 form by the 52 binary digits contained in the pair of registers n and ntk. 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.

<u>Interpreted Instructions</u>	<u>Function</u>
ca n (38)	Clear the MRA and add into it the number in location n.
cs n (36)	Clear the MRA and subtract from it the number in location n.
cm n (37)	Clear the MRA and add into it the magnitude of the number in location n.
ad n (72)	Add the number in the MRA to the number in location n and leave the sum in the MRA.
su n (76)	Subtract from the number in the MRA the number in location n and leave the difference in the MRA.
mr n (49)	Multiply the number in the MRA by the number in location n and leave the product in the MRA.
dv n (74)	Divide the number in the MRA by the number in location n and leave the quotient in the MRA.
ts n (48)	Transfer the number in the MRA to location n.
ex n (48)	Exchange the number in the MRA with the numbers in location n.
sp m (25)	Interpret next the instruction in register m (unless m = <u>ss</u> , in which case transfer control to the register following the one which contains the <u>spax</u> so that the instruction following the <u>spax</u> is performed using the Whirlwind code).
cp m (24)	If the contents of the MRA is a negative number, proceed as in <u>sp n</u> above; if positive, ignore this instruction.
ta m (22)	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 <u>sp</u> or effective <u>cp</u> operation.

CHAPTER: Operations on Real (24,6,0) Floating Point
Double Register Numbers (General Subroutine)

LSR# PA 2.2 %

Instruction Code and Operation Times:

ts 48	cp 21(+), 27(-)	cs 36	cm 37
ta 22	sp 25	ad 72	mr 49
ex 48	ca 38	su 76	dv 74

Preset Parameters (to be typed in program title)

vx2/pk: k=separation in storage of two registers of number
vx/pN: N=address in storage of initial register of this subroutine

Enter-->00	ta 179r	Set address of 1 st instruction to be interpreted	196r->25	ex 198r "dv"
01	<u>sp 179r</u>		26	ts 97r
02	(p0)	x_1 } Multiple	27	cs 102r } Form exponent
03	(p0)	x_1 } register	28	ad 54r } of $2^{-2}/x_2$
04	(p0)	y_1 } accumulator	29	ts 102r }
196r-->05	sr*30	"ca"	30	cs 97r }
06	ca ax		31	mh 97r } Form and store
13r, 196r->07	ca 191r	"cs"	32	ex 198r }
08	<u>sp 95r</u>		33	sr *2 }
196r->09	<u>sp 129r</u>	"ad"	34	dv 198r } $\frac{2^{-2}}{x_2}$
10	p29		35	sl *15 }
196r->11	ts 97r	"su"	36	ts 15lr }
12	<u>sp 126r</u>		37	ca 72r } Form and store
196r->13	<u>sp 7r</u>	"cm"	38	dv 97r } $\frac{2^{-2}}{x_2}$
(170r)14	(p0)	Temporary digits storage	39	sl *15 }
24r->15	sa 3r	} Add two minor	40	te 198r }
16	ts 3r	} products	41	mh 97r } Form
17	ca 0	} Store	42	su 72r } $\left(\frac{2^{-2}}{x_2}\right)$
18	ex 198r	} overflow	43	sl *15 }
19	mh 2r	Form major product	44	su 17r } (Use Euclid's algorithm)
20	<u>sp 158r</u>		45	ad 50r }
49r, 196r->21	mr 2r	"mr" Form two	46	dv 97r }
22	ex 3r	minor products	47	sl *15 }
23	mr 198r		48	ad 15lr } Add two minor parts
24	<u>sp 21r</u>		49	of reciprocal,

50 p1	80r=83 sl 14 } Add overflow
120r=51 <u>sp 73r</u> "ts"	84 ts 2r } into x_1 and x_1'
111r=52 ca 2r } Complement x_1	85 ao 4r } Increase y_1 .
53 <u>sp 164r</u>	82r=86 cm 4r }
54 p2	87 su 62r } $ y_1 > -63$?
196r=55 ca 201r "ta" } Store digits	88 <u>cp_93r</u>
(181r)56 td(0) } in indicated	89 cs 4r } $y_1 \leq 0$?
57 <u>sp 178r</u> } address	90 <u>cp_202r</u> } (i.e. $y_1 < -63$?)
119r=58 ao 2r } Increase x_1	91 cs 62r } Set $y_1 = -63$
59 <u>sp 167r</u> } by 2^{-15}	92 ts 4r }
60 sp 35r	88r=93 ca 97r } ts n + k
196r=61 <u>sp 73r</u> "ex"	94 ad 197r } or ex n + k
62 p63	8r=95 ts 102r } Store ts, ex, ca, cs, or cm n+k
196r=63 cs 2r "cp" } Is x_1 negative?	96 ca 2r }
64 <u>cp_178r</u>	(180r)97 (p0) }
196r=65 ao 179r "sp" } Set return address	98 ex 3r } Perform
66 td 201r } for <u>sp ax</u> .	99 sr *9 } ts, ex, ca, cs, or cm
67 ca 180r } Set pick up order	100 ex 4r }
68 td 179r } for ordinary cp & sp	101 sl *9 }
69 su 6r	(95r)102 (p0) }
~ <u>cp_199r</u> } Test to see whether	103 sr *9 }
71 <u>sp 179r</u> } instruction is <u>sp ax</u>	104 ex 3r }
72 0,20000	105 ts 2r }
51r,61r=73 ca 3r } Round off x_1' and	106 sl *15 }
74 sr 6 } store $x_1' \times 2^{-6}$	107 ex 3r }
75 ts 3r }	108 <u>sp 177r</u> }
76 sr *9 } Add round-off carry	166r=109 cm 2r }
77 sa 2r } into x_1	110 su 0 } $x_1' \neq 0$?
78 ts 2r }	111 <u>cp 52r</u> }
79 ca 0 } Is there an overflow?	112 cm 3r }
80 <u>cp_83r</u>	113 su 0 } $x_1' \neq 0$?
81 su 0 }	114 <u>cp_122r</u> }
82 <u>cp 86r</u>	115 su 50r } Form

NAME: Operations on Real (24,6,0) Floating Point
Double Register Numbers (General Subroutine)

LSR # PA 2.2 t

116 ad 17r	$ x_1' > 1$	147 ca 50r	Set $y_2 = 1$
117 ts 3r		148 ex 102r	
118 ca 2r	$x_1' > 0?$	149 sr *15	Form and store
119 cp_58r		150 ad 198r	$(x_2 + x_2') \cdot 2^{-15} \cdot 2^{-1} y_2 - y_1 $
120 su 50r	Form	(146r) 151 (p0)	
121 ts 2r	$x_1' = 2^{-15}$	152 ts 198r	
114r-122	cs 3r	153 sl *15	
	Complement x_1'	154 ex 3r	
123 ts 3r		155 sr *15	Form
124 mr 2r	Form $x_1 \cdot x_1'$	156 ad 2r	$(x_1 + x_1') \cdot 2^{-15} \cdot 2^{-1}$
125 sp 166r		157 sr *1	
12r-126	cs 198r	20r-158 ts 2r	Store x_1
	Complement	159 sl *15	
127 ts 198r	x_2, x_2'	160 sa 3r	
128 cs 97r		161 ts 3r	Add x_1, x_1'
9r-129	ex 102r	162 ca 198r	and x_2, x_2'
	Form and store	163 ad 2r	
130 su 4r	$y_2 = y_1$	53r-164 ts 2r	
131 ts 97r		165 mr 3r	Does sign $x_1 =$ sign $x_1'?$
132 cp_141r	$y_2 - y_1 > 0?$	125r-166 sp_109r	
133 ad 4r		59r-167 ca 3r	Scale factor and
134 ts 4r	Interchange	168 sr *15	store x_1, x_1'
135 ca 2r	(x_1, x_1', y_1)	169 ad 2r	
136 ex 198r	and (x_2, x_2', y_2)	170 sf 14r	
137 ts 2r		171 ts 2r	
138 ca 3r		172 sl *15	
139 ex 102r		173 ts 3r	
140 ts 3r		174 cs 14r	Form exponent
132r-141	cm 97r	175 ad 102r	of x_1, x_1'
	$ y_2 - y_1 > 29 > 0?$	176 ad 4r	
142 su 10r		108r-177 ts 4r	
143 cp_145r			
144 sp 178r	No need for addition		
143r-145	ad 5r		
	Store		
146 ts 151r	$sr *1 + y_2 - y_1 $		

Operations on Real (24,6,0) Floating Point
Double Register Numbers (General Subroutine)

LSR P 2.2 t

(171r, 172r, 173r) 178	cc 179r	Increase address	(181r) 191	ca(0)	y_2 in reg. 102
171r, 200r, 179	ca(0)	Pick up next instruction	192	sr *9	Hold x_2^* in AG.
180 ts 97r		Store instruction	193	ex 102r	
181 td 56r		and digits	194	ts 198r	
182 td 189r			195	sl *15	
183 ad 197r			(188r) 196	(p0)	Go to part of I.S. for
184 td 191r			197	pex2	particular instruction
185 sr *27		Form sp to address	198	(p0)	Separation parameter
186 sl *17		for particular	70r -> 199	ad 50r	Temporary storage
187 ad 60r		instruction	200	sp 179	SP 60 Does address equal ax?
188 ts 196r			(66r) 201	(sp 0)	Return to register
(182r) 189	ca(0)	Pick up x_2 , x_2^* and y_2 .	202	ca 108r	following sp_ax. Produce overflow
190 ts 102r		Store x_2 in reg. 198	203	ad 108r	alarm

600 | sp 179 arf

ca 201 arf

su 179 (p0)

SP total separation parameter

sp 201 arf

601 | sp

9 p

DIGITAL COMPUTER LABORATORY
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY
 WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Minimal Routine with Sign Agreement, giving 28 Binary Digit Accuracy in mr)			LSR PA 3.1 t
			Classification Interpretive
No. of Regs. in Subroutine 96	Temp. Regs. used by Sub. -----	Average Time (operations) see description	Max. Time (operations) see description
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 <u>n</u> , then the minor half is stored in register <u>n+k</u> . 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 <u>sp ax</u> 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 <u>sp ax</u> ordinary WW operation is resumed at the register following the instruction <u>sp ax</u> . The multiple register accumulator (mra), in which the results of instructions are left, consists of the storage registers 2r and 3r.			

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Minimal Routine with Sign Agreement, giving 28 Binary Digit Accuracy in mra)		LSR	PA 3.1 t
Instruction	Function	Av. No. Operations	
ts n	Transfer the contents of mra to registers <u>n</u> and <u>n+k</u>	22	
ta n	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	18	
ex n	Exchange the contents of mra with the contents of <u>n</u> and <u>n+k</u>	22	
cp n	If the contents of mra is negative, proceed as in the sp instruction, if the contents are positive disregard the instruction	22	
sp n	If <u>sp n</u> = <u>sp ax</u> , take the next instruction to be interpreted from register n, if <u>sp n</u> = <u>sp ax</u> , resume ordinary WW operation at the register following the instruction <u>sp ax</u>	22	
ca n	Clear mra, put the contents of registers <u>n</u> and <u>n+k</u> in mra	23	
cs n	Clear mra, put the complement of the contents of registers <u>n</u> and <u>n+k</u> in mra	22	
ad n	Add the contents of mra to the contents of registers <u>n</u> and <u>n+k</u> and store the result in mra.	40	
su n	Subtract the contents of registers <u>n</u> and <u>n+k</u> from the contents of mra, store the result in mra.	44	
cm n	Clear mra, put the absolute value of the contents of registers <u>n</u> and <u>n+k</u> in mra	22	
mr n	Multiply the contents of mra by the contents of registers <u>n</u> and <u>n+k</u> , store the result in mra.	34	

TITLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Minimal Routine with Sign Agreement, giving 28 Binary Digit Accuracy in mra)

LSR PA 3.1 t

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.

TITLE: Operations on Real (30,0,0) Fixed-Point Double
 Register Numbers (Minimal Routine with Sign
 Agreement, giving 28 Binary Digit Accuracy in mr)

ISR# PA 3.1 t

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.

Preset Parameters: (to be inserted in main program)

v8 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 5lr	Enter interpretive subroutine	18	ex 2r	mr
01	<u>sp 5lr</u>		19	mh 28r	
02	(p0)		20	ts 28r	
03	(p0)		21	sl 15	
64r-->04	sr 19r	ca	22	<u>sp 69r</u>	
64r-->05	<u>sp 44r</u>	cs	87r-->23	ao 2r	Form C(2r) + 2 ⁻¹⁵
64r-->06	<u>sp 69r</u>	ad	24	<u>sp 50r</u>	
64r-->07	<u>sp 65r</u>	su	79r-->25	cs 2r	Complement C(2r)
64r-->08	<u>sp 44r</u>	cm	26	<u>sp 49r</u>	
09	p1		64r-->27	<u>sp 44r</u>	ts
10	p0a8	vl	(62r)28	(p0)	
11	td ax	vx	64r-->29	ca 43r	ta
64r-->12	mr 2r		(53r)30	td (0)	Transfer m+1 to digit
13	ex 3r		31	<u>sp 50r</u>	section of register n
14	mr 28r		64r-->32	<u>sp 44r</u>	ex
15	sa 3r		64r-->33	<u>sp 93r</u>	cp
16	ts 3r		95r, 64r-->34	ao 5lr	sp
17	ca 0		35	td 43r	Store sp(m+1) in 43r

EXAMPLE: Operations on Real (30,0,0) Fixed-Point Double Register Numbers (Minimal Routine with Sign Agreement, giving 28 Binary Digit Accuracy in mr)

36	ca 30r	Store ca n in 5lr	66	cs 28r	
37	td 5lr		67	ts 28r	su
38	su 11r		68	cs 45r	
39	cp 41r	Is xx n = sp ax?	69	sa 3r	
40	sp 5lr		70	ts 3r	
39r-41	ad 9r		71	ca 2r	ad,su,mr
42	cp 5lr		72	ad 28r	
(35r)43	sp (0)	Leave interpretive sub.	73	ts 2r	
5r,8r, 27r,32r	ca 3r		92r-->74	cs 2r	Do C(2r) and C(3r)
(56r)45	(p0)	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) ≠ 0?
(54r)48	(p0)		78	su 0	
26r-->49	ts 2r		79	cp 25r	
24r,31r 76r,94r	ao 5lr		80	cm 3r	Is C(3r) ≠ 0?
(Or),1r	-->51	ca (0)	81	su 0	
40r,42r	52	td 61r	82	cp 90r	
	53	td 30r	83	su 9r	Form 1 - C(3r)
	54	ts 48r	84	ad 17r	
	55	ad 10r	85	ts 3r	
	56	ts 45r	86	ca 2r	Is C(2r) positive?
	57	td 63r	87	cp 23r	
	58	sr *11	88	su 9r	Form C(2r) - 2 ⁻¹⁵
	59	ad 4r	89	ts 2r	
	60	td 64r	82r-->90	cs 3r	Complement C(3r)
(52r)61	ca (0)	Store C(n) in 28r	91	ts 3r	
	62	ts 28r	92	sp 74r	Re-enter sign agreement
(57r)63	ca (0)	Put C(n+k) in AC	33r-->93	cs 2r	cp
(60r)64	sp (0)	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

TITLE: Operations on Real (30,0,0) Fixed-Point Double-Register Numbers (General Routine with Sign Agreement, No Division) (Interpretive)		LSR PA 3.5t	
		TAPE T721-2	
		Classification Interpretive	
No. of Regs. in Subroutine 123	Temp. Regs. used by Subroutine -----	Average Time (operations) see description	Max. Time (operations) see description
<p>Preset Parameters</p> <p>vx2/ pk: k = separation in storage between the two registers of a double register number</p> <p>vx/ pn: N = address in storage of the first register of the interpretive subroutine</p>			
<p>Description</p> <p>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 <u>n+k</u>.</p> <p>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 <u>sp ax</u> 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 <u>sp ax</u> ordinary WW operation is resumed at the register following the instruction <u>sp ax</u>.</p> <p>The multiple register accumulator (mra), in which the results of instructions are left, consists of the storage registers 2r and 3r.</p>			
Instruction	Function	Av. No.	Operations
ts n	Transfer the contents of mra to registers <u>n</u> and <u>n+k</u>	22	
ta n	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	18	
ex n	Exchange the contents of mra with the contents of <u>n</u> and <u>n+k</u>	22	
cpm	If the contents of mra is negative, proceed as in the sp instruction, if the contents are positive disregard the instruction	22	

TITLE: Operations on Real (30,0,0) Fixed-Point Double-
 Register Numbers (General Routine with Sign
 Agreement, No Division) (Interpretive) LSR PA 3.5t

<u>Instruction</u>	<u>Function</u>	<u>Av. No. Operations</u>
sp n	If <u>sp n = sp ax</u> , take the next instruction to be interpreted from register n, if <u>sp n = sp ax</u> , resume ordinary WW operation at the register following the instruction <u>sp ax</u>	22
ca n	Clear mra, put the contents of registers <u>n</u> and <u>n+k</u> in mra	23
cs n	Clear mra, put the complement of the contents of registers <u>n</u> and <u>n+k</u> in mra	22
ad n	Add the contents of mra to the contents of registers <u>n</u> and <u>n+k</u> and store the result in mra	40
su n	Subtract the contents of registers <u>n</u> and <u>n+k</u> from the contents of mra, store the result in mra	44
cm n	Clear mra, put the absolute value of the contents of registers <u>n</u> and <u>n+k</u> in mra	22
mr n	Multiply the contents of mra by the contents of registers <u>n</u> and <u>n+k</u> , store the result in mra.	34
sl(800+n)	Multiply C(mra) by 2^n and leave the result in the mra	29
sr(800+n)	Multiply C(mra) by 2^{-n} , and store the first 30 digits of the result without roundoff in the mra.	28

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 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 $56r - sp n+k$, an excessive address is being used for storing the minor half of a double register constant.

TITLE: Operations on Real (30,0,0) Fixed-Point Double- LSR PA 3.5t
Register Numbers (General Routine with Sign
Agreement, No Division) (Interpretive)

(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).

FCH	AS
12/5/51	2/5/52

TITLE: Operations on Real (30,0,0) Fixed-Point
 Double-Register Numbers (General Routine
 with Sign Agreement, No Division)
 (Interpretive)

LSR# PA 3.5t
 TAPE# T721-2

Abstract: This subroutine is a (30,0,0) interpretive subroutine which performs the instructions ts, ta, ex, cp, ca, cs, ad, su, cm, mr, sl, 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. 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)

vx_2 : 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 5lr	{ Set next instruction address	20	ts 2r	
01	<u>sp 5lr</u>		21	sl* 15	
(23r) → 02	(p0)	x_2 } mra	22	<u>sp 92r</u>	
03	(p0)	x_1 }	88r → 23	ao 2r	Form $C(2r)+2^{-15}$
65r → 04	sr 19r	ca	24	<u>sp 50r</u>	
65r → 05	<u>sp 44r</u>	cs	80r → 25	cs 2r	Complement $C(2r)$
65r → 06	<u>sp 70r</u>	ad	26	<u>sp 49r</u>	
65r → 07	<u>sp 66r</u>	su	65r → 27	<u>sp 44r</u>	ts
65r → 08	<u>sp 44r</u>	cm	(68r)(63r) 28	(p0)	$C(n) = x_2$
33r → 09	cs 2r	{ Is $C(2r)$ negative?	65r → 29	ca 43r	ta
10	<u>cp 50r</u>		(53r) 30	td(0)	{ Transfer $m+1$ to digit section of register D
11	<u>sp 34r</u>	Perform sp instruc-	31	<u>sp 50r</u>	
65r → 12	ts 19r	tion	65r → 32	<u>sp 44r</u>	ex
13	mr 3r	}	65r → 33	<u>sp 9r</u>	op
14	<u>sp 94r</u>	mr	11r, 65r → 34	ao 5lr	sp
65r → 15	ca ax	sl (vx)	35	td 43r	{ store $sp(m+1)$ in 43r
65r → 16	ca 3r	sr	36	ca 62r	
17	sr* 15	{	37	td 5lr	{ Is $xxn = spax$
18	ad 2r	sl, sr	38	su 15r	
(103r)(55r)(12r) → 19	(p0)	$C(n+k)=x_2^1$	39	<u>cp 41r</u>	

TITLE: Operations on Real (30,0,0) Fixed-Point
 Double-Register Numbers (General Routine
 with Sign Agreement, No Division) (Interpretive)

LSR# PA 3.5t
 TAPE# T721-2

40 <u>sp 51r</u>	<u>6r</u>	70 <u>sa 3r</u>	ad, su, mr
39r → 41 <u>ad 122r</u>	120r → 71 <u>ts 3r</u>		
42 <u>op 51r</u>	72 <u>ca 2r</u>		
(35r) 43 <u>sp 0</u>	73 <u>ad 28r</u>		
5r, 8r → 44 <u>ca 3r</u>	74 <u>ts 2r</u>		
27r, 32r → 45 <u>(p0)</u>	93r → 75 <u>cs 2r</u>	Do C(2r) and C(3r) agree in sign	
(97r)(66r)(57r) (113r) 46 <u>ts 3r</u>	76 <u>mr 3r</u>		
47 <u>ca 2r</u>	77 <u>op 50r</u>		
(54r) 48 <u>(p0)</u>	78 <u>cm 2r</u>		
26r → 49 <u>ts 2r</u>	79 <u>su0</u>	Is C(2r) ≠ 0?	
10r, 24r → 50 <u>ac 51r</u>	80 <u>op 25r</u>		
31r, 77r → 51 <u>ca(0)</u>	81 <u>cm 3r</u>	Is C(3r) ≠ 0?	
1r, 40r, 42r, (0r) (34r)(37r)(50r) 52 <u>td 62r</u>	82 <u>su0</u>		
53 <u>td 30r</u>	83 <u>op 91r</u>		
54 <u>ts 48r</u>	84 <u>su 122r</u>		
55 <u>ts 19r</u>	85 <u>ad 118r</u>	Form - 1 + C(3r)	
56 <u>ad 121r</u>	86 <u>ts 3r</u>		
57 <u>ts 45r</u>	87 <u>ca 2r</u>	Is C(2r) positive?	
58 <u>td 64r</u>	88 <u>op 23r</u>		
59 <u>sr* 11</u>	89 <u>su 122r</u>	Form C(2r) - 2 ⁻¹⁵	
60 <u>ad 4r</u>	90 <u>ts 2r</u>		
61 <u>td 65r</u>	83r → 91 <u>cs 3r</u>	Complement C(3r)	
(52r) 62 <u>ca(0)</u>	22r → 92 <u>ts 3r</u>		
63 <u>ts 28r</u>	93 <u>sp 75r</u>	Re-enter sign agreement	
(58r) 64 <u>ca(0)</u>	put C(n+k) = x'_2 in AC 14r → 94 <u>sr 2</u>	Form $\frac{1}{4} [x'_1 \ x'_2]$ and store	
(61r) 65 <u>sp(0)</u>	enter table	in 19r	
7r → 66 <u>ts 45r</u>	95 <u>ex 19r</u>		
67 <u>cs 28r</u>	96 <u>mh 2r</u>		
68 <u>ts 28r</u>	97 <u>ts 45r</u>		
69 <u>cs 45r</u>	98 <u>sl*15</u>		
	99 <u>sr 2</u>	Form $\frac{1}{4} [x'_1 \ x'_2]$	

TITLE: Operations on Real (30,0,0) Fixed-Point LSR# PA 3.5t
 Double-Register Numbers (General Routine TAPE # T721-2
 with Sign Agreement, No Division)(Interpretive)

100 ad 19r	}	Accumulate round-off and store in 2r	112 sa 45r	}
101 ex 2r			113 ts 45r	
102 mh 28r	}		114 ca 19r	
103 ts 19r			115 ex 28r	}
104 sl* 15	116 sa 45r	Add $x_1'x_2'$ and the major half of $x_1'x_2'$		
105 ex 28r	117 ts 2r			
106 mh 3r	}	Form $x_1'x_2'$ and store major half in 3r	118 ca0	
107 ts 3r			119 ex 2r	}
108 sl* 15	120 sp 70r	Enter addition routine		
109 sr 2	121 p0ax2	vl		
110 ad 2r	}	Form total round-off and add into major half of $x_1'x_2'$	122 pl	
111 sr 13				

DIGITAL COMPUTER LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
WHIRLWIND SUBROUTINE SPECIFICATION

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		LSR PA 3.10t
		TAPE T798
		Classification Interpretive
No. of Regs. in Subroutine	Temp. Regs. used by Sub.	Average Time (operations)
78	0	---
Max. Time (operations)		
Preset Parameters (To be inserted in title of main program) 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		
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 <u>n</u> , then the minor half is stored in register <u>n+k</u> .		
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 <u>sp ax</u> 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 <u>sp ax</u> ordinary WW operation is resumed at the register following the instruction <u>sp ax</u> .		
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		
$10 \cdot 2^{-15} + 5 \cdot 2^{-30} - [5 \cdot 2^{-15} + 10 \cdot 2^{-30}] = 5 \cdot 2^{-15} + (-10) \cdot 2^{-30}$		
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.		

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 LSR PA 3.10t
 TAPE T798

<u>Instruction</u>	<u>Function</u>	<u>Av. No.</u> <u>Operations</u>
ts	Transfer the contents of <u>MRA</u> to registers <u>n</u> and <u>n+k</u>	17
ta	If <u>m</u> is the address of the last sp instruction executed by the subroutine, transfer (<u>m+1</u>) to the last 11 digits of register <u>n</u>	13
ex	Exchange the contents of <u>MRA</u> with the contents of <u>n</u> and <u>n+k</u>	17
cp	If the contents of <u>MRA</u> is negative, proceed as in the sp instruction, if the contents are positive disregard the instruction	(+)14 (-)25
sp	If <u>sp n ≠ sp ax</u> , take the next instruction to be interpreted from register <u>n</u> , if <u>sp n = sp ax</u> , resume ordinary WW operation at the register following the instruction sp ax	18
ca	Clear <u>MRA</u> , put the contents of registers <u>n</u> and <u>n+k</u> in <u>MRA</u>	17
cs	Clear <u>MRA</u> , put the complement of the contents of registers <u>n</u> and <u>n+k</u> in <u>MRA</u>	17
ad	Add the contents of <u>MRA</u> to the contents of registers <u>n</u> and <u>n+k</u>	18
su	Subtract the contents of registers <u>n</u> and <u>n+k</u> from the contents of <u>MRA</u>	17
mr	Multiply the contents of <u>MRA</u> by the contents of registers <u>n</u> and <u>n+k</u> , store the result in <u>MRA</u>	33

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

LSR PA 3.10t
TAPE T798

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

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 59r--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(2r) and 3ax(3r).

FCH	MAS
2/5/52	2/8/52

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

LSR PA 3.10t

TAPE T-798

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

00	ta 53r	Enter interpretive	20	su 0	
01	<u>sp 53r</u>	subroutine	21	<u>ep 60r</u>	
(76r)02	(p0)	} mra	22	cs 2r	} Is C(2r) or C(3r) pos.?
(74r)03	(p0)		23	<u>ep 52r</u>	
59r → 04	<u>sp 44r</u>	ca	24	<u>sp 34r</u>	Perform interpreted sp
59r → 05	<u>sp 44r</u>	cs	25	pa x2	
59r → 06	<u>sp 7r</u>	ad	26	1.67777ax2	
6r, 59r → 07	ad 26r	su	50r → 27	<u>sp 44r</u> ts	
08	ts 9r	Form ca(n+k) or cs (n+k) & store in 9r	28	sp ax	
77r → (8r)(15r)09	(p0)	} Form $x_2^1 + x_1^1$ and store in 3r(ad, su)	59r → 29	td 65r ta	Transfer n to digit section of reg. 65r
10	sa 3r		30	ca 42r	
11	<u>sp 48r</u>	{ Form $x_1^1 x_2^1 + x_1^1 x_2^1$ + $x_0^1 x_2^1$ (mr)	31	<u>sp 65r</u>	
39r → 12	td 68r	} mr	59r → 32	<u>sp 44r</u> ex	
13	ts 72r	Set address at 68r to n & store mrn in 72r	62r → 33	<u>sp 19r</u> cp	
14	ad 25r	} Store mr(n+k) in 9r	24r → 34	ao 53r sp	
15	ts 9r		59r	td 42r	Store sp (m+1) in 42r
16	ca 51r	} Store ts2r in 50r	35	ca 50r	Store ca n in 53r
17	ts 50r		36	td 53r	
18	<u>sp 67r</u>		37	su 28r	
35r → 19	cm 2r	} Is C(2r) ≠ 0	38	ts 47r	Is xxn = sp ax?

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

LSR PA 3.10t
 TAPE T798

40	cm 47r	
41	su 0	
(35r)	42 <u>cp(0)</u>	
	43 <u>sp 53r</u>	
$4r, 5r$	$27r, 32r \rightarrow 44$	ad 25r } Store $xx(n+k)$ in 47r
	45	ts 47r }
(39r)	46 ca 3r	
(45r)	(69r) 47 (p0)	
11r	$\rightarrow 48$	ts 3r } Perform ts, ex, ca, cs, ad, su, mr
	49	ca 2r }
(17r)(54r) 50 (p0)		
	51	ts 2r }
$23r, 66r \rightarrow$	52	ao 53r } Pick up instruction
(34r)(37r)(0r)	53	ca(0) } to be interpreted
$1r, 43r \rightarrow$	54	ts 50r Store xxn
(52r)	55	$sr * 11$ }
	56	ad 33r } Set up entry into
	57	ts 59r table }
	58	ca 50r Pick up xxn

(57r)	59 <u>(p0)</u>	Enter table
21r	$\rightarrow 60$	cm 3r }
	61	su 0 } Is $C(3r) \neq 0$
	62	<u>cp 34r</u>
	63	cs 3r
	64	<u>sp 23r</u>
(29r)	$31r \rightarrow 65$	td (0) } Transfer $m+1$ to digit section
	66	<u>sp 52r</u> of register n }
18r	$\rightarrow 67$	ca 2r Form $x_1 x_2$
	68	mh(0)n
	69	ts 47r Store $[x_1 x_2]$ in 47r
	70	s1 15 } Store $[x_1 x_2]^1$ in 3r
	71	ex 3r }
(13r)	72 (p0)	Form $[x_1^1 x_2]$
	73	sa 3r } Form $[x_1 x_2]^1 + [x_1^1 x_2]$
	74	ts 3r } and store in 3r
	75	ca 47r } Store $[x_1 x_2]$ in 47r
	76	ex 2r }
	77	<u>sp 9r</u>

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Operations on real (15,0,c) fixed-point single register numbers (Interpretive)		LSR PA 8.1t
		TAPE T724-2
		Classification Interpretive
No. of Regs. in Subroutine 47	Temp. Regs. used by Sub. None	Average Time (operations) See description of order code.
<p>Preset Parameters (To be typed in the title of the Main Program)</p> <p>vx:pN: N = address in storage of initial register of subroutine</p> <p>vx3:pc: c = the number of binary digits to the right of the binary point in the (15,0,c) numbers</p> <p>vx1:pM: M = address in storage of the initial register of the storage block for (15,0,c) numbers</p>		
<p>Description</p> <p>By means of this subroutine various logical and arithmetic operations can be performed on real numbers expressed in the (15,0,c) system, $0 \leq c \leq 15$. The (15,0,c) constants are stored in the following manner. Let c be fixed and let a be a positive number such that</p> $2^{-(15-c)} \leq a \leq 2^{c+1} - 1$ <p>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.</p> <p>For example, let $c = 3$ and $a = -3\frac{1}{3}$</p> <p>Then</p> $-a = 3 + \frac{1}{3}$ <p>and</p> $3 = +.011$ $\frac{1}{3} = +.010101010101_010$ <p style="text-align: center;"><small>15-3=12 digits</small></p> $3 + \frac{1}{3} = 011.010101010101$ <p>We now complement the following number</p> 0.011010101010101 <p>and store the following result</p> 1.100101010101010 <p>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.</p>		

DIGITAL COMPUTER LABORATORY
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY
 WHIRLWIND SUBROUTINE SPECIFICATION

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

LSR PA 8.1t

TAPE T724-2

Description (continued)

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

The multiple register accumulator, in which the results of interpreted instructions are left, consists of the storage register 2xx.

<u>Inst.</u>	<u>Function</u>	<u>Average # Operations</u>
ts n	Transfer C(mra) to register n.	14
td n	Transfer the last 11 digits from the mra to the last 11 digits of register n	14
ta n	If m is the address of the last sp instruction or effective cp instruction executed by the subroutine, transfer (m+1) into the last 11 digits of register n	13
ex n	Exchange C(mra) with C(n)	14
cp n	If C(mra) is negative, proceed as in the sp instruction, if C(mra) is positive, disregard the instruction	18
sp n	If spn ≠ spax, take the next instruction to be interpreted from register n, if spn = spax, resume ordinary Whirlwind operation at the register following the instruction spax	21
ca n	Clear mra, put C(n) in the mra	14
cs n	Clear mra, put the complement of C(n) in the mra	14
ad n	Add C(mra) to C(n) and store the result in the mra	14
su n	Subtract C(n) from C(mra) and store the result in the mra	14
cm n	Clear the mra, and put the positive magnitude of C(n) in the mra	14
mr n	Multiply C(mra) by C(n), store the result in the mra	14
sl n	Multiply C(mra) by 2^n and store the result in the mra	14
sr n	Multiply C(mra) by 2^{-n} and store the result in the mra	14

DIGITAL COMPUTER LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
WHIRLWIND SUBROUTINE SPECIFICATION

<p>TITLE: Operations on real (15,0,c) fixed-point single register numbers (Interpretive)</p>	<p>LSR PA 8.elt</p>
	<p>TAPE T724-2</p>
Notes:	
1. <u>Entering and leaving subroutine</u> Both entering and leaving the subroutine are accomplished by the instruction <u>sp ax</u> , 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 <u>sp ax</u> in storage. When used to leave the subroutine, ordinary WW operation is resumed at the register following the instruction <u>sp ax</u> .	
2. <u>Accuracy</u> All of the operations executed by the subroutine are carried out with a 15 binary digit accuracy.	
3. <u>Calculation with integers</u> If c = 15, the (15,0,c) numbers dealt with by the subroutine are binary integers.	
4. <u>Reasons for machine stoppage during the subroutine</u> a) Arithmetic overflow at 35r - overflow in a smaller 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. <u>Overflow during mr</u> 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).	

FCH	AS
12/6/51	2/5/52

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

LSR# PA 8.1
TAPE# T 724-2

Abstract: PA 8.1 is a (15,0,c) interpretive subroutine which performs the instructions ts, td, ta, ex, cp, sp, ca, cs, ad, su, em, 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,C) 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) = xxn

00	ta 38r	Enter interpretive subroutine	46r → 25	<u>sp 38r</u>	
01	<u>sp 38r</u>		26	<u>sp 34r</u>	ts
02	(p0)	mra	46r → 27	<u>sp 34r</u>	td
46r → 03	sr 18r	ca	46r → 28	ca 24r	ta
46r → 04	ca ax	cs	(41r) 29	td(0)	Transfer m+1 to digit section of register n
46r → 05	<u>sp 34r</u>	ad	30	<u>sp 37r</u>	
46r → 06	<u>sp 34r</u>	su	46r → 31	<u>sp 34r</u>	ex
46r → 07	<u>sp 34r</u>	em	46r → 32	<u>sp 8r</u>	cp
32r → 08	cs 2r		46r → 33	<u>sp 16r</u>	sp
09	<u>cp 37r</u>	cp	5r, 6r, 7r		
10	<u>sp 16r</u>		14r, 15r	→ 34	ca 2r
46r → 11	mh 2r		26r, 27r, 31r	→ 35	(p0)
12	s10ax3	mr	(21r)(39r)	13r → 36	ts 2r
13	<u>sp 36r</u>		9r, 30r → 37	ao 38r	
46r → 14	<u>sp 34r</u>	sl	(Or)(16r)	38	ca(0)
46r → 15	<u>sp 34r</u>	sr	1r, 25r	39	ts 35r Store xxn
10r, 33r → 16	ao 38r	store cp(m+1) in	(37r)(19r)	40	td 45r } Store n
17	td 24r	24r		41	td 29r }
18	ca 45r	store ca n in 38r	42	sr * 11	
19	td 38r		43	ad 3r	Set up entry to table
20	su 4r		44	td 46r	
21	ts 35r		(40r) 45	ca(0)	Put C(n) in AC
22	cm 35r	Is xxn=spax?	(44r) 46	<u>sp(0)</u>	Enter table.
23	su0				
(17r) 24	<u>ep0</u>				

DIGITAL COMPUTER LABORATORY
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY
 WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Form $\sin \frac{\pi}{2} x$ from x Stored in AC, and Leave Result in AC			LSR TF 0.1 t
			Classification Closed
No. of Regs. in Subroutine 19	Temp. Regs. used by Sub. d - lt	Average Time (operations) 15	Max. Time (operations) 15
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 <u>+0.00005</u> and the average error is <u>+0.00003</u> .			

TITLE: Form $\sin \frac{\pi}{2} x$ from x Stored in AC, and Leave

LSR# TF 0,1 t

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	10	sr *1	$(-bx^2 + cx^4 - dx^6)2^{-1}$
01	ts lt Store x	11	ad 18r	$(a - bx^2 + cx^4 - dx^6)2^{-1}$
02	mh lt x^2	12	mh lt	$(ax - bx^3 + cx^5 - dx^7)2^{-1}$
03	mh 15r $-dx^2$	13	sl 1	$ax - bx^3 + cx^5 - dx^7$
04	ad 16r $c - dx^2$	(Or) 14	<u>sp (0)</u>	Return to main program
05	mh lt $cx - dx^3$	15	1.77560	-d
06	mh lt $cx^2 - dx^4$	16	0.05055	c
07	ad 17r $-b + cx^2 - dx^4$	17	1.26521	-b
08	mh lt $-bx + cx^3 - dx^5$	18	0.62210	$a x 2^{-1}$
09	mh lt $-bx^2 + cx^4 - dx^6$			

DIGITAL COMPUTER LABORATORY
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY
 WHIRLWIND SUBROUTINE SPECIFICATION

TITLE: Form $\cosine_{\frac{\pi}{2}}y$ from y Stored in AC, and Leave Result in AC (15,0,0)			LSR TF 1.1t
			Tape 705-1
			Classification Closed
No. of Regs. in subroutine 26	Temp. Regs. used by Sub. d -1t(2)	Average Time (operations) 20	Max. Time (operations) 20
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 $\cos_{\frac{\pi}{2}}y = \sin_{\frac{\pi}{2}}x = ax^3 - bx^5 + cx^7 - dx^9$, for $-1 < x < 1$. If $y=0$, there will be an overflow since the cosine of zero is one. The sub- routine 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 ± 0.00005 and the average error ± 0.00002 .			
Notes 1. If y is <u>zero</u> , an arithmetic overflow will occur in register 5r.			
DMN+DGA 1/14/52	JWC III 1/16/52	MAS 1/22/52	

TITLE: Form cosine $\frac{\pi}{2}y$ from y Stored in AC, and LSR# TF 1.lt
 Leave Result in AC (15,0,0) Tape 705-1

Abstract: Gives cosine $\frac{\pi}{2}y$ by forming $x=1-|y|$ so that

$\cosine \frac{\pi}{2}y = \sin \frac{\pi}{2}x$ and evaluating

$\sin \frac{\pi}{2}x = ax - bx^3 + cx^5 - dx^7$, $-1 < x < 1$. If $y = \pm 0$

there will be an overflow at register 5r.

Upon entering the subroutine:

ac: y

Upon leaving the subroutine:

ac: cosine $\frac{\pi}{2}y$

Temporary registers:

d - unused

lt - used to store x

00	ta 19r	Set return address	13	mh lt	$-bx + cx^3 - dx^5$
01	ts lt	Transfer y to lt	14	mh lt	$-bx^2 + cx^4 - dx^6$
02	cp 4r	y negative? Yes	15	sr*1	$(-bx^2 + cx^4 - dx^6)2^{-1}$
03	os lt	No. -y in ac	16	ad 25r	$(a - bx^2 + cx^4 - dx^6)2^{-1}$
2r → 04	ad 20r	$(1 - 2^{-15}) - y$	17	mh lt	$(ax - bx^3 + cx^5 - dx^7)2^{-1}$
05	ad 21r	Add 2^{-15} } (Possible overflow)	18	sl 1	$ax - bx^3 + cx^5 - dx^7$
06	ts lt	$1 - y = x$ in it	(or) 19	sp(0)	Return to main program
07	mh lt	x^2	20	0.77777	$1 - 2^{-15}$
08	mh 22r	$-dx^2$	21	0.00001	2^{-15}
09	ad 23r	$c - dx^2$	22	1.77560	-d
10	mh lt	$cx - dx^3$	23	0.05055	c
11	mh lt	$cx^2 - dx^4$	24	1.26521	-b
12	ad 24r	$-b + cx^2 - dx^4$	25	0.62210	$a x 2^{-1}$

DIGITAL COMPUTER LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

WHIRLWIND SUBROUTINE SPECIFICATION

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)		LSR TF 7.1 t Tape T 750-1
		Classification Closed
No. of Regs. in Subroutine 28	Temp. Regs. used by Sub. d - 1t	Average Time (operations) 15 - sine 20 - cosine
Program Parameters on entering Subroutine ac: x or y ar: return address		
Results on leaving Subroutine ac: sine $\frac{\pi}{2}x$ or cosine $\frac{\pi}{2}y$		
Description If this subroutine is entered at register Or, it will calculate cosine $\frac{\pi}{2}y$ by changing y to x so that cosine $\frac{\pi}{2}y$ = 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. The subroutine is entered with either x or y in the accumulator, and on returning to the main program either sine $\frac{\pi}{2}x$ or cosine $\frac{\pi}{2}y$ is in the accumulator. The maximum error is approximately ± 0.00005 and the average error is ± 0.00002 .		
Notes: 1. Enter at Or if cosine $\frac{\pi}{2}y$ is desired enter at 7r if sine $\frac{\pi}{2}x$ is desired 2. If y = ± 0 , an overflow will occur in register 5r.		
DMN/FDGA 1/14/52	JWC III 1/16/52	DEL 1/18/52

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)

LSR# TF 7.1t

Tape T 750-1

Abstract: If this subroutine is entered at register 0r, it will calculate cosine $\frac{\pi}{2}y$ by changing y to x so that cosine $\frac{\pi}{2}y = \sin \frac{\pi}{2}x$ 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

lt - used to store x

Cosine	→ 00	ta 21r	Set return address	14	ad 26r	$-b + cx^2 - dx^4$
	01	ts lt	y in lt	15	mh lt	$-bx + cx^3 - dx^5$
	02	cp 4r	Is y negative? Yes	16	mh lt	$-bx^2 + cx^4 - dx^6$
	03	cs lt	No. -y in ac	17	sr* 1	$(-bx^2 + cx^4 - dx^6)2^{-1}$
2r	→ 04	ad 22r	$(1 - 2^{-15}) - y $ in ac	18	ad 27r	$(a - bx^2 + cx^4 - dx^6)2^{-1}$
	05	ad 23r	$1 - y = x$ in ac Possible overflow	19	mh lt	$(ax - bx^3 + cx^5 - dx^7)2^{-1}$
	06	sp 8r		20	s1 1	$ax - bx^3 + cx^5 - dx^7$
Sine	→ 07	ta 21r	Set return address (0r)(7r)	21	sp (0)	Return to main program
6r	→ 08	ts lt	x in lt	22	0.77777	$1 - 2^{-15}$
	09	mh lt	x^2	23	0.00001	2^{-15}
	10	mh 24r	$-dx^2$	24	1.77560	-d
	11	ad 25r	$c - dx^2$	25	0.05055	c
	12	mh lt	$cx - dx^3$	26	1.26521	-b
	13	mh lt	$cx^2 - dx^4$	27	0.62210	$a x 2^{-1}$