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	pN:	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 $2^{63} > |N| \geq 2^{-64}$, 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 $|1 - x2^{y}N^{-1}| < 2^{-24}$. Thus the number pair x,y represents N to within ±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.

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, $v.2^w$ to $x.2^y$, assuming $1>|v|\geq .5$, $1>|x|\geq .5$, $w\geq y$, the sum obtained is $\left[(v + x.2^{y-w}\pm 2^{-29})\ 2^z\right]\ 2^{w-z} = u.2^{w-z}$, 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 $x.2^y$, 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 $N = X.10^Y$, 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 $.01\pi$,	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.

<u>Allocation of storage locations</u> 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.

Numbers designated by programmer	Storage registers	Comments
address at start of program, usually 32.	main program 2-register numbers, 1st halves	
total number of locations = k = parameter x2.	subroutine 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.
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 nt for the first half and to ntax2 for the second half.
	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	
address of start of main program and of each subroutine and address of first instruction to be performed must be indicated	main program	
	subroutines	address of last word of last subroutine must be less than 704.
		spase available for print subroutine OT 102.1
address of start of interpretive subroutine = 852 = parameter x	interpretive subroutine	

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 32 binary digits contained in the pair of registers n and n+1. The term "register m" is used to signify the single register m. Figures in parentheses give the number of Whirlwind instructions required to interpret the indicated instructions.

Interpreted Instructions	Function
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 = ax, in which case transfer control to the register following the one which contains the spax so that the instruction following the spax is performed using the Whirlwind code).
cp m (24)	If the contents of the MRA is a negative number, proceed as in sp n 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 sp or effective cp operation.

TITLE: Operations on Real (24,6,0) Floating Point Double Register Numbers (General Subroutine) ISR# PA 2.2 3

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 1st instruction to be interpreted	196r→	25	ex 198r	"dv"
	01	sp 179r			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
13r, 196r→	07	ca 191r	"cs"		32	ex 198r	store
	08	sp 95r			33	sr *2	
196r→	09	sp 129r	"ad"		34	dv 198r	$-\frac{2^{-2}x_2'}{x_2^2}$
	10	p29			35	sl *15	
196r→	11	ts 97r	"su"		36	ts 151r	
	12	sp 126r			37	ca 72r	Form and store
196r→	13	sp 7r	"cm"		38	dv 97r	$\frac{2^{-2}}{x_2^2}$
(170r)	14	(p0)	Temporary digits storage		39	sl *15	
24r→	15	sa 3r	Add two minor		40	ts 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
	20	sp 158r			45	ad 50r	algorithm)
49r, 196r→	21	mr 2r	"mr" Form two		46	dv 97r	
	22	ex 3r	minor products		47	sl *15	
	23	mr 198r			48	ad 151r	Add two minor parts
	24	sp 15r			49	sp 21r	of reciprocal.

50	p1		
196r→51	sp 73r	"ts"	
111r→52	cs 2r		Complement x_1
53	sp 164r		
54	p2		
196r→55	ca 201r	"ta"	Store digits
(181r)56	td(0)		in indicated
57	sp 178r		address
119r→58	ao 2r		Increase x_1
59	sp 167r		by 2^{-15}
60	||sp 35r		
196r→61	sp 73r	"ex"	
62	p63		
196r→63	cs 2r	"cp"	Is x_1 negative?
64	cp 178r		
196r→65	ao 179r	"sp"	Set return address
66	td 201r		for sp ax.
67	ca 18[illegible]		Set pick up order
68	td 179r		for ordinary cp & sp
69	su 6r		
70	cp 199r		Test to see whether
71	sp 179r		instruction is sp ax
72	0.20000		
51r,61r→73	ca 3r		Round off x_1' and
74	sr 6		store $x_1' \times 2^{-6}$
75	ts 3r		
76	sr *9		Add round-off carry
77	sa 2r		into x_1
78	ts 2r		
79	ca 0		Is there an overflow?
80	cp 83r		
81	su 0		
82	cp 86r		
80r→83	sl 14		Add overflow
84	ts 2r		into x_1 and x_1'
85	ao 4r		Increase y_1.
82r→86	cm 4r		
87	su 62r		$\lvert y_1 \rvert - 63 > 0$?
88	cp 93r		
89	cs 4r		$y_1 \leq 0$?
90	cp 202r		(i.e. $y_1 < -63$?)
91	cs 62r		Set $y_1 = -63$
92	ts 4r		
88r→93	ca 97r		ts n + k
94	ad 197r		or ex n + k
8r→95	ts 102r		Store ts, ex, ca, cs, or cm n+k
96	ca 2r		
(180r)97	(p0)		
98	ex 3r		Perform
99	sr *9		ts, ex, ca, cs, or cm
100	ex 4r		
101	sl *9		
(95r)102	(p0)		
103	sr *9		
104	ex 3r		
105	ts 2r		
106	sl *15		
107	ex 3r		
108	sp 177r		
166r→109	cm 2r		
110	su0		$x_1 \neq 0$?
111	cp 52r		
112	cm 3r		
113	su 0		$x_1' \neq 0$?
114	cp 122r		
115	su 50r		Form

	116	ad 17r	$\lvert x_1' \rvert - 1$
	117	ts 3r	
	118	ca 2r	$x_1 > 0$?
	119	cp 58r	
	120	su 50r	Form
	121	ts 2r	$x_1 - 2^{-15}$
114r→	122	cs 3r	Complement x_1'
	123	ts 3r	
	124	mr 2r	Form $x_1 . x_1'$
	125	sp 166r	
12r→	126	cs 198r	Complement
	127	ts 198r	x_2, x_2'.
	128	cs 97r	
9r→	129	ex 102r	Form and store
	130	su 4r	$y_2 - y_1$
	131	ts 97r	
	132	cp 141r	$y_2 - y_1 > 0$?
	133	ad 4r	
	134	ts 4r	Interchange
	135	ca 2r	(x_1, x_1', y_1)
	136	ex 198r	and (x_2, x_2', y_2)
	137	ts 2r	
	138	ca 3r	
	139	ex 102r	
	140	ts 3r	
132r→	141	cm 97r	
	142	su 10r	$\lvert y_2 - y_1 \rvert - 29 > 0$?
	143	cp 145r	
	144	sp 178r	No need for addition
143r→	145	ad 5r	Store
	146	ts 151r	sr *1 + $\lvert y_2 - y_1 \rvert$
	147	ca 50r	Set $y_2 = 1$
	148	ex 102r	
	149	sr *15	Form and store
	150	ad 198r	$(x_2 + x_2' \cdot 2^{-15}) \cdot 2^{-1} - \lvert y_2 - y_1 \rvert$
(146r)	151	(p0)	
	152	ts 198r	
	153	sl *15	
	154	ex 3r	
	155	sr *15	Form
	156	ad 2r	$(x_1 + x_1' \cdot 2^{-15}) \cdot 2^{-1}$
	157	sr *1	
20r→	158	ts 2r	Store x_1
	159	sl *15	
	160	sa 3r	
	161	ts 3r	Add x_1, x_1'
	162	ca 198r	and x_2, x_2'
	163	ad 2r	
53r→	164	ts 2r	
	165	mr 3r	Does sign x_1 = sign x_1'?
125r→	166	cp 109r	
59r→	167	ca 3r	Scale factor and
	168	sr *15	store x_1, x_1'
	169	ad 2r	
	170	sf 14r	
	171	ts 2r	
	172	sl *15	
	173	ts 3r	
	174	cs 14r	Form exponent
	175	ad 102r	of x_1, x_1'
	176	ad 4r	
108r→	177	ts 4r	

34r, 144r→178	ao 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 AC.
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 particular instruction
184	td 191r		197	pe x2	Separation parameter
185	sr *27	Form sp to address	198	(p0)	Temporary storage
186	sl *17	for particular	70r→199	ad 50r	sp 600
187	ad 60r	instruction	200	~~sp 179r~~	Does address equal ax?
188	ts 196r		(66r) 201	(sp 0)	Return to register following sp ax.
(182r) 189	ca(0)	Pick up x_2, x_2' and y_2.	90r→202	ca 108r	Produce overflow
190	ts 102r	Store x_2 in reg. 198	203	ad 108r	alarm

[illegible] in interpretive routine: start of 500

600 | sp 179 [illegible]
ca 201 [illegible]
su [illegible]
sp to [illegible] instruction
sp 201 [illegible]

601 | ca
2 | qp