

RT Assembler Instructions

(a b c p q ... variables or numbers, i ... numbers only, m ... labels, s ... strings)

mov	a b	a ← b (move)	abs	a	positive value of a
clr	a	a ← 0 (clear)	neg	a	invert the sign of a
inc	a	a ← a+1 (increment)	sgn	a	signum of a (+1, 0 oder -1)
dec	a	a ← a-1 (decrement)	round	a	round
			ceil	a	gives the smallest integer ≥ a
add	a b	a ← a+b	floor	a	gives the largest integer ≤ a
sub	a b	a ← a-b	fix	a	floor if a<0, ceil if a>0, zero if a=0
mul	a b	a ← a*b	frac	a	fraction of a
div	a b	a ← a/b	clip	a b c	clip a into b ... c
			cmod	a b c	clip a into b ... c by modulo function (sawtooth)
power	a b	gives a to the b-th power	random	a	random number 0 ... 1
root	a b	gives the b-th root of a			
exp	a	natural exponential function of a (base e)	cmpgt	a b m	if a > b then goto m ("compare greater than")
exp10	a	exponential function of a to base 10	cmpge	a b m	if a ≥ b then goto m ("compare greater equal")
exp2	a	exponential function of a to base 2	cmplt	a b m	if a < b then goto m ("compare less than")
expx	a b	exponential function of a to base b	cmple	a b m	if a ≤ b then goto m ("compare less equal")
log	a	natural logarithm of a (base e)	cmpeq	a b m	if a = b then goto m ("compare equal")
log10	a	common logarithm of a (base 10)	cmpne	a b m	if a ≠ b then goto m ("compare not equal")
log2	a	logarithm of a to base 2	tstgt	a m	if a > 0 then goto m ("test greater than")
logx	a b	logarithm of a to base b	tstge	a m	if a ≥ 0 then goto m ("test greater equal")
			tstlt	a m	if a < 0 then goto m ("test less than")
sin	a	a ← sin(a)	tstle	a m	if a ≤ 0 then goto m ("test less equal")
cos	a	a ← cos(a)	tsteq	a m	if a = 0 then goto m ("test equal")
tan	a	a ← tan(a)	tstne	a m	if a ≠ 0 then goto m ("test not equal")
cot	a	a ← cot(a)	jump	m	jump to m
sec	a	a ← sec(a) = 1 / cos(a)	input	a s	input of a number a by dialog, dialog text s
csc	a b	a ← cosec(a) = 1 / sin(a)	output	a s	output of a number a by dialog, dialog text s
asin	a b	a ← arcsin(a)	pause	s	program pause, message text s
acos	a b	a ← arccos(a)	proof	a s	message s and number a output (without break)
atan	a b	a ← arctan(a)	info	s	message s output (without break)
acot	a b	a ← arccot(a)	cls		clears the global output text
asec	a b	a ← arcsec(a)	printn	a b c	print number a into output text. Format b.c digits
acsc	a b	a ← arccosec(a)	prints	s	print string s into output text
			save	s	saves output text into textfile s.txt
sinh	a	a ← sinus hyperbolicus (a)	read	a b	reads a number (b=0) or an array (b>0) a from file
cosh	a	a ← cosinus hyperbolicus (a)	write	a b	writes a number (b=0) or an array (b>0) a in a file
tanh	a	a ← tangens hyperbolicus (a)	adrof	p a	gives the address of symbol a to a pointer symbol p
coth	a	a ← cotangens hyperbolicus (a)	get	a p q	moves a value from address (p+q) to a
sech	a	a ← secans hyperbolicus (a)	put	p q a	moves a value from a to address (p+q)
csch	a	a ← cosecans hyperbolicus (a)	init		program initialization
asinh	a	a ← area sinus hyperbolicus (a)	nop		no operation
acosh	a	a ← area cosinus hyperbolicus (a)	mode	a	set mode a. 0='Run', 1='Error Stop', 2='Stepwise'
atanh	a	a ← area tangens hyperbolicus (a)	halt		program break
acoth	a	a ← area cotangens hyperbolicus (a)	err	a m	gives an error code to a, on error jump to label m
asech	a	a ← area secans hyperbolicus (a)	exit		exit
acsch	a	a ← area cosecans hyperbolicus (a)	_name	s	gives the program the name s
			_var	a	declares a as an variable (not necessary)
bin	a	if a≠0 then a ← 1, else 0	_dim	a i	declares a as an array with the length i
not	a	if a=0 then a ← 1, else 0	_lab	m	defines the label m
and	a b	if a≠0 and b≠0 then a ← 1, else 0	_config	i	configuration of virtual machine
or	a b	if a≠0 or b≠0 then a ← 1, else 0	_end		program end