RT Assembler Instructions

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

mov	*a b*	*a* ← *b* (move)
clr	a	*a* ← 0 (clear)
inc	a	*a* ← *a*+1 (increment)
dec	a	*a* ← *a*-1 (decrement)
add	*a b*	*a* ← *a*+*b*
sub	*a b*	*a* ← *a*-*b*
mul	a b	*a* ← *a*×*b*
div	*a b*	*a* ← *a*/*b*
power	*a b*	Gives *a* to the *b*-th power
root	*a b*	Gives the *b*-th root of *a*
exp	*a*	Natural exponential function of *a* (base e=2,71..)
exp10	*a*	Exponential function of *a* to base 10
exp2	*a*	Exponential function of *a* to base 2
expx	*a b*	Exponential function of *a* to base b
log	*a*	Natural logarithm of *a* (base e=2.7182...)
log10	*a*	Common logarithm of *a* (base 10)
log2	*a*	Logarithm of *a* to base 2
logx	*a b*	Logarithm of *a* to base *b*
sin	*a*	*a* ← sin(*a*)
cos	*a*	*a* ← cos(*a*)
tan	*a*	*a* ← tan(*a*)
cot	*a*	*a* ← cot(*a*)
sec	*a*	*a* ← sec(*a*) (=1/cos(*a*))
csc	*a b*	*a* ← cosec(*a*) (=1/sin(*a*))
asin	*a b*	*a* ← arcsin(*a*) (b ... full angle inversion sign)
acos	*a b*	*a* ← arccos(*a*)
atan	*a b*	*a* ← arctan(*a*)
acot	*a b*	*a* ← arccot(*a*)
asec	*a b*	*a* ← arcsec(*a*)
acsc	*a b*	*a* ← arccosec(*a*)
sinh	*a*	*a* ← sinus hyperbolicus (*a*)
cosh	*a*	*a* ← cosinus hyperbolicus (*a*)
tanh	*a*	*a* ← tangens hyperbolicus (*a*)
coth	*a*	*a* ← cotangens hyperbolicus (*a*)
sech	*a*	*a* ← secans hyperbolicus (*a*)
csch	*a*	*a* ← cosecans hyperbolicus (*a*)
asinh	a	*a* ← area sinus hyperbolicus *(a)*
acosh	a	*a* ← area cosinus hyperbolicus (*a*)
atanh	*a*	*a* ← area tangens hyperbolicus (*a*)
acoth	*a*	*a* ← area cotangens hyperbolicus (*a*)
asech	*a*	*a* ← area secans hyperbolicus (*a*)
acsch	*a*	*a* ← area cosecans hyperbolicus (*a*)
bin	*a*	if *a*≠0 then *a* ← 1, else 0
not	*a*	if *a*=0 then *a* ← 1, else 0
and	*a b*	if *a*≠0 and *b*≠0 then *a* ← 1, else 0
or	*a b*	if *a*≠0 or *b*≠0 then *a* ← 1, else 0

abs	*a*	Absolute (positive) value of *a*
neg	*a*	Invertion of *a* sign
sgn	*a*	Signum of *a* (+1, 0 oder -1)
round	*a*	Round
ceil	*a*	Gives the smallest integer ≥*a*
floor	*a*	Gives the largest integer ≤*a*
fix	*a*	Floor if *a*<0, ceil if *a*>0, zero if *a*=0
frac	*a*	Fraction of *a*
clip	*a b c*	Clip *a* into *b* ... *c* range
cmod	*a b c*	Clip *a* into b ... c range by modulo function (sawtooth)
random	*a*	Random number 0 ... 1
cmpgt	*a b* ***m***	If *a*>*b* then goto ***m*** ("compare greather than")
cmpge	*a b* ***m***	If *a*≥*b* then goto ***m*** ("compare greather equal")
cmplt	*a b* ***m***	If *a*<*b* then goto ***m*** ("compare less than")
cmple	*a b* ***m***	If *a*≤*b* then goto ***m*** ("compare less equal")
cmpeq	*a b* ***m***	If *a*=*b* then goto ***m*** ("compare equal")
cmpne	*a b* ***m***	If *a*≠*b* then goto ***m*** ("compare not equal")
tstgt	*a* ***m***	If *a*>0 then goto ***m*** ("test greather than")
tstge	*a* ***m***	If *a*≥0 then goto ***m*** ("test greather equal")
tstlt	*a* ***m***	If *a*<0 then goto ***m*** ("test less than")
tstle	*a* ***m***	If *a*≤0 then goto ***m*** ("text less equal")
tsteq	*a* ***m***	If a=0 then goto ***m*** ("test equal")
tstne	*a* ***m***	If *a*≠0 then goto ***m*** ("test not equal")
jump	***m***	Jump to ***m***

input	*a* ***s***	Input of a number *a* by dialog, dialog text ***s***
output	*a* ***s***	Output of a number *a* by dialog, dialog text ***s***
pause	***s***	Program break, message text ***s***
proof	*a* ***s***	Message ***s*** and number *a* output (without break)
info	***s***	Message ***s*** output (without break)
cls		Clear the global output text
printn	*a b c*	Print number *a* into global output text, format *b.c* digits
prints	***s***	Print string s into global output text
save	***s t***	Save the output text into file ***s.t***, ***s***=filename, ***t***=type
read	*a b* ***t***	Read a number (*b*=0)/an array (*b*>0) a from file ***a.t***
write	*a b* ***t***	Write a number (*b*=0)/an array (*b*>0) a in file ***a.t***

adrof	*p a*	Gives the address of symbol *a* to pointer *p*
get	*a p q*	Moves value from address (*p*+*q*) to *a*
put	*p q a*	Moves value from *a* to address (*p*+*q*)

init		Program initialization
nop		No operation
mode	*a*	Set mode *a*. 0='Run', 1='Error Stop', 2='Stepwise'
halt		Processor Halt
err	*a* ***m***	Error code to *a*, on error jump to label ***m***
exit		Program exit

_name	***s***	Names program ***s***
_var	*a*	Variable *a* declaration (recommendend but not necessary)
_dim	*a* ***i***	Array *a* declaration, length ***i***
_lab	***m***	Label ***m*** defintion. Other (common) notation: "***m:***"
_config	***i***	Virtual machine configuration
_end		Program end