

RT-Assembler: Befehlsliste

(a b c p q ... Variablen oder Zahlen, i ... Zahlen, m ... Marken, s ... Zeichenketten)

mov	b a	a nach b schaffen (Transportbefehl)
clr	a	a auf 0 setzen (Löschbefehl)
inc	a	a = a + 1 (Inkrement)
dec	a	a = a - 1 (Dekrement)
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	a = a hoch b
root	a b	a = b-te Wurzel aus a
exp	a	a = e hoch a
exp10	a	a = 10 hoch a
exp2	a	a = 2 hoch a
expx	a b	a = b hoch a
log	a	a = natürlicher Logarithmus aus a (Basis e)
log10	a	a = dekadischer Logarithmus aus a (Basis 10)
log2	a	a = dyadischer Logarithmus aus a (Basis 2)
logx	a b	a = beliebiger Logarithmus aus a (Basis 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	a = cosec(a) = 1 / sin(a)
asin	a	a = arcsin(a)
acos	a	a = arccos(a)
atan	a	a = arctan(a)
acot	a	a = arccot(a)
asec	a	a = arcsec(a)
acsc	a	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	wenn a ≠ 0 dann a = 1, sonst 0 (Log. Identität)
not	a	wenn a = 0 dann a = 1, sonst 0 (Log. Negation)
or	a b	wenn a ≠ 0 oder b ≠ 0 dann a = 1, sonst 0
and	a b	wenn a ≠ 0 und b ≠ 0 dann a = 1, sonst 0

abs	a	Absolutbetrag von a bilden
neg	a	Vorzeichen von a umkehren (Negation)
sgn	a	Vorzeichen von a (+1, 0 oder -1)
round	a	a runden
ceil	a	a aufrunden
floor	a	a abrunden
fix	a	Vorkommastellen von a (a zur 0 hin runden)
frac	a	Nachkommastellen von a
clip	a b c	a = a, wenn a < b dann a = b, wenn a > c, a = c
cmmod	a b c	a mit Modulofunktion in b ... c clippen
random	a	a = Zufallszahl zwischen 0 und 1
cmpgt	a b m	wenn a > b dann Sprung nach m
cmpge	a b m	wenn a ≥ b dann Sprung nach m
cmplt	a b m	wenn a < b dann Sprung nach m
cmple	a b m	wenn a ≤ b dann Sprung nach m
cmpeq	a b m	wenn a = b dann Sprung nach m
cmpne	a b m	wenn a ≠ b dann Sprung nach m
tstgt	a m	wenn a ≥ 0 dann Sprung nach m
tstge	a m	wenn a > 0 dann Sprung nach m
tstlt	a m	wenn a < 0 dann Sprung nach m
tstle	a m	wenn a ≤ 0 dann Sprung nach m
tsteq	a m	wenn a = 0 dann Sprung nach m
tstne	a m	wenn a ≠ 0 dann Sprung nach m
jump	m	Unbedingter Sprung nach m
input	a s	fordert a mit Text s im Dialog an (mit Pause)
output	a s	zeigt a mit Text s im Dialog an (mit Pause)
pause	s	Programmpause mit Dialogtext s (mit Pause)
proof	a s	Testausgabe von a, Mitteilung s (ohne Pause)
info	s	Mitteilung s (ohne Pause)
cls		Löschen des Ausgabertextes
printn	a b c	Ausgabe der Zahl a mit b.c Stellen
prints	s	Ausgabe der Zeichenkette s in den Ausgabertext
save	s	Speichern des Ausgabertextes in Textdatei s
read	a b	Zahl a (b=0)/Feld a (b=Länge) aus Datei lesen
write	a b	Zahl a (b=0)/Feld a (b=Länge) in Datei schr.
adrof	p a	Adresse von a nach p schaffen.
get	a p q	Das Symbol mit der Adresse (p+q) nach a
put	p q a	a auf das Symbol mit der Adresse (p+q)
init		Erste Zeile eines Programms
nop		Nullbefehl. Dieser Befehl macht nichts
mode	a	a=0: 'Ohne Halt', 1: 'Fehlerhalt', 2: 'Step'
halt		Hält das Programm an
errcode	a	Fehlercode nach a. 0: Kein Fehler.
errjump	m	Bei Fehler Sprung nach m
exit		Programmebeendigung
_name	s	Dem Programm den Namen s geben
_var	a	Variable a deklarieren
_dim	a i	Feld a mit a(0) ... a(i) definieren
_lab	m	Marke m definieren (Identisch mit m:)
_config	i	Virtuelle Maschine konfigurieren
_end		Programmende