RT-Assembler: Befehlsliste

a b c p q ... Variablen oder Zahlen, *i* ... muss eine Zahl sein, *m* ... Marke, *s t* ... Zeichenketten

mov	*a b*	*a* ← *b* (Transportbefehl)
clr	*a*	*a* ← 0 (Löschbefehl)
inc	*a*	*a* ← *a*+1 (Inkrementbefehl)
dec	*a*	*a* ← *a*−1 (Dekrementbefehl)
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^b
root	*a b*	*a* ← *b*-te Wurzel aus *a*. (Bei Quadratwurzel *b*=2 angeben)
exp	*a*	*a* ← e^a (*e*=2,7182...)
exp10	*a*	*a* ← 10^a
exp2	*a*	*a* ← 2^a
expx	*a b*	*a* ← b^a
log	*a*	*a* ← $\log_e a$ (*e*=2,7182...)
log10	*a*	*a* ← $\log_{10}(a)$
log2	*a*	*a* ← $\log_2(a)$
logx	*a b*	*a* ← $\log_b(a)$
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 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*	Wenn *a*≠0 dann *a* ← 1, sonst *a* ← 0 (Logische Identität)
not	*a*	Wenn *a*=0 dann *a* ← 1, sonst *a* ← 0 (Logische Negation)
or	*a b*	Wenn *a*≠0 oder *b*≠0 dann *a* ← 1, sonst *a* ← 0 (Logisch Oder)
and	*a b*	Wenn *a*≠0 und *b*≠0 dann *a* ← 1, sonst *a* ← 0 (Logisch Und)

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*	wenn *a*<*b* dann *a* ← *b*, wenn *a*>*c* dann *a* ← *c*, sonst *a* ← *a*
cmod	*a b c*	*a* mit Modulofunktion (Sägezahn) in *b* ... *c* clippen („clip and modulo“)
random	*a*	*a*←Zufallszahl zwischen 0 und 1
cmpgt	*a b m*	Wenn *a*>*b* Sprung nach *m* („compare greather than“)
cmpge	*a b m*	Wenn *a*≥*b* Sprung nach *m* („ ... greather equal“)
cmplt	*a b m*	Wenn *a*<*b* Sprung nach *m* („ ... less than“)
cmple	*a b m*	Wenn *a*≤*b* Sprung nach *m* („ ... less equal “)
cmpeq	*a b m*	Wenn *a*=*b* Sprung nach *m* („ ... equal“)
cmpne	*a b m*	Wenn *a*≠*b* Sprung nach *m* („ ... not equal“)
tstgt	*a m*	Wenn *a*≥0 Sprung nach *m* („test greather than“)
tstge	*a m*	Wenn *a*>0 Sprung nach *m* („ ... greather equal“)
tstlt	*a m*	Wenn *a*<0 Sprung nach *m* („ ... less than“)
tstle	*a m*	Wenn *a*≤0 Sprung nach *m* („ ... less equal“)
tsteq	*a m*	Wenn *a*=0 Sprung nach *m* („ ... equal “)
tstne	*a m*	Wenn *a*≠0 Sprung nach *m* („ ... not equal“)
jump	*m*	Unbedingter Sprung nach *m*
input	*a s*	Zahl *a* mit Text *s* anfordern (mit Programmunterbrechung)
output	*a s*	Zahl *a* mit Text *s* anzeigen (mit Programmunterbrechung)
pause	*s*	Text *s* anzeigen (mit Programmunterbrechung)
proof	*a s*	Zahl *a* mit Text *s* anzeigen (ohne Programmunterbrechung)
info	*s*	Text *s* anzeigen (ohne Programmunterbrechung)
cls		Globalen Ausgabetext löschen
printn	*a b c*	Zahl *a* mit *b* Vor- und *c* Nachkommastellen in globalen Aus-
prints	*s*	Text *s* in globalen Ausgabetext ausgeben [gabetext ausgeben
save	*s t*	Globalen Ausgabetext in Datei *s.t* speichen, *s*=Name, *t*=Typ
read	*a b t*	Zahl *a* (*b*=0) oder Feld *a* (*b*=Länge+1) aus Datei *a.t* lesen
write	*a b t*	Zahl *a* (*b*=0) oder Feld *a* (*b*=Länge+1) in Datei *a.t* schreiben
adrof	*p a*	*p* ← Adresse von *a*.
get	*a p q*	Das Symbol mit der Adresse (*p*+*q*) nach *a* lesen
put	*p q a*	*a* auf das Symbol mit der Adresse (*p*+*q*) schreiben
init		Programminitialisierung. Wird automatisch als 1. Befehl er-
nop		Nullbefehl. Nichts machen („no operation“) [zeugt
mode	*a*	Programmmodus setzen. *a*=0:‚Ohne Halt‘/1:‚Fehlerhalt‘/
halt		Halt-Befehl. Hält den Prozessor an [2:‚Schrittweise‘
err	*a m*	*a* ← Fehlercode. 0: Kein Fehler/≠0: Fehler. Bei Fehler Sprung
exit		Programmbeendigung [nach *m*
_name	*s*	Dem Programm den Namen *s* geben
_var	*a*	Variable *a* deklarieren
_dim	*a i*	Feld *a* mit den Feldelementen *a*(0) ... *a*(*i*) definieren
_lab	*m*	Marke *m* definieren. Andere (üblichere) Schreibweise „*m:*“
_config	*i*	Virtuelle Maschine mit Parameter *i* konfigurieren
_end		Programmende. Letzte Zeile im Quellprogramm