INSTRUCCIONES DE LA MÁQUINA-P

Instr.	Meaning	Cond.	Result
add *N*	*STORE*[*SP*−1] := *STORE*[*SP*−1] $+_N$ *STORE*[*SP*]; *SP* := *SP*−1	(*N*, *N*)	(*N*)
sub *N*	*STORE*[*SP*−1] := *STORE*[*SP*−1] $-_N$ *STORE*[*SP*]; *SP* := *SP*−1	(*N*, *N*)	(*N*)
mul *N*	*STORE*[*SP*−1] := *STORE*[*SP*−1] $*_N$ *STORE*[*SP*]; *SP* := *SP*−1	(*N*, *N*)	(*N*)
div *N*	*STORE*[*SP*−1] := *STORE*[*SP*−1] $/_N$ *STORE*[*SP*]; *SP* := *SP*−1	(*N*, *N*)	(*N*)
neg *N*	*STORE*[*SP*] := $-_N$ *STORE*[*SP*]	(*N*)	(*N*)
and	*STORE*[*SP*−1] := *STORE*[*SP*−1] *and* *STORE*[*SP*]; *SP* := *SP*−1	(b, b)	(b)
or	*STORE*[*SP*−1] := *STORE*[*SP*−1] *or* *STORE*[*SP*]; *SP* := *SP*−1	(b, b)	(b)
not	*STORE*[*SP*] := *not* *STORE*[*SP*]	(b)	(b)
equ *T*	*STORE*[*SP*−1] := *STORE*[*SP*−1] $=_T$ *STORE*[*SP*]; *SP* := *SP*−1	(*T*, *T*)	(b)
geq *T*	*STORE*[*SP*−1] := *STORE*[*SP*−1] $\geq_T$ *STORE*[*SP*]; *SP* := *SP*−1	(*T*, *T*)	(b)
leq *T*	*STORE*[*SP*−1] := *STORE*[*SP*−1] $\leq_T$ *STORE*[*SP*]; *SP* := *SP*−1	(*T*, *T*)	(b)
les *T*	*STORE*[*SP*−1] := *STORE*[*SP*−1] $<_T$ *STORE*[*SP*]; *SP* := *SP*−1	(*T*, *T*)	(b)
grt *T*	*STORE*[*SP*−1] := *STORE*[*SP*−1] $>_T$ *STORE*[*SP*]; *SP* := *SP*−1	(*T*, *T*)	(b)
neq *T*	*STORE*[*SP*−1] := *STORE*[*SP*−1] $\neq_T$ *STORE*[*SP*]; *SP* := *SP*−1	(*T*, *T*)	(b)

Instr.	Meaning	Cond.	Result
ldo *T q*	*SP* := *SP*+1; *STORE*[*SP*] := *STORE*[*q*]	*q* ∈ [0,*maxstr*]	(*T*)
ldc *T q*	*SP* := *SP*+1; *STORE*[*SP*] := *q*	Type(*q*) = *T*	(*T*)
ind *T*	*STORE*[*SP*] := *STORE*[*STORE*[*SP*]]	(a)	(*T*)
sro *T q*	*STORE*[*q*] := *STORE*[*SP*]; *SP* := *SP*−1	(*T*) *q* ∈ [0,*maxstr*]	
sto *T*	*STORE*[*STORE*[*SP*−1]] := *STORE*[*SP*]; *SP* := *SP*−2	(a, *T*)	

Instr.	Meaning	Comments	Cond.	Result
ujp q	$PC := q$	Unconditional branch	$q \in [0,codemax]$	
fjp q	**if** $STORE[SP] = false$	Conditional branch	(b)	
	then $PC := q$		$q \in [0,codemax]$	
	fi			
	$SP := SP - 1$			

Instr.	Meaning	Cond.	Result
ixj q	$PC := STORE[SP] + q$;	(i)	
	$SP := SP - 1$		

Instr.	Meaning	Cond.	Results
ixa q	$STORE[SP-1] := STORE[SP-1] +$ $STORE[SP] * q$	(a,i)	(a)
	$SP := SP - 1$		

Instr.	Meaning	Cond.	Result
inc T q	$STORE[SP] := STORE[SP] + q$	(T) and type (q) = i	(T)
dec T q	$STORE[SP] := STORE[SP] - q$	(T) and type (q) = i	(T)

Instr.	Meaning	Cond.	Result
chk p q	**if** ($STORE[SP] < p$) **or** ($STORE[SP] > q$)	(i,i)	(i)
	then *error*('value out of range')		
	fi		

Instr.	Meaning	Cond.	Result
dpl T	$SP := SP + 1$;	(T)	(T, T)
	$STORE[SP] := STORE[SP - 1]$		
ldd q	$SP := SP + 1$;	(a, T_1, T_2)	(a, T_1, T_2, i)
	$STORE[SP] := STORE[STORE[SP - 3] + q]$		
sli T_2	$STORE[SP - 1] := STORE[SP]$;	(T_1, T_2)	(T_2)
	$SP := SP - 1$		

Instr.	Meaning	Cond.	Result
new	**if** $NP - STORE[SP] \leq EP$	(a,i)	
	then *error*('store overflow')		
	else $NP := NP - STORE[SP]$;		
	$STORE[STORE[SP - 1]] := NP$;		
	$SP := SP - 2$		
	fi;		

Instr.	Meaning
lod $T\ p\ q$	$SP := SP + 1$;
	$STORE[SP] := STORE[base(p, MP) + q]$
lda $p\ q$	$SP := SP + 1$;
	$STORE[SP] := base(p, MP) + q$
str $T\ p\ q$	$STORE[base(p, MP) + q] := STORE[SP]$;
	$SP := SP - 1$

Instr.	Meaning	Comments
mst p	$STORE[SP + 2] := base(p,MP)$;	Static link
	$STORE[SP + 3] := MP$;	Dynamic link
	$STORE[SP + 4] := EP$;	Save EP
	$SP := SP + 5$	The parameters can now be evaluated starting from $STORE[SP + 1]$
cup $p\ q$	$MP := SP - (p + 4)$;	p is the storage requirement for the parameters
	$STORE[MP + 4] := PC$;	Save return address
	$PC := q$	Branch to procedure start address q
ssp p	$SP := MP + p - 1$	p size of static part of data area
sep p	$EP := SP + p$;	p max. depth of local stack
	if $EP \geq NP$ **then** *error*('store overflow') **fi**	Check for collision of stack and heap

Instr.	Meaning	Comments
retf	$SP := MP$;	Function result in the local stack
	$PC := STORE[MP + 4]$;	Return branch
	$EP := STORE[MP + 3]$;	Restore EP
	if $EP \geq NP$	
	then *error*('*store overflow*')	
	fi	
	$MP := STORE[MP + 2]$	Dynamic link
retp	$SP := MP - 1$;	Proper procedure with no results
	$PC := STORE[MP + 4]$;	Return branch
	$EP := STORE[MP + 3]$;	Restore EP
	if $EP \geq NP$	
	then *error*('*store overflow*')	
	fi	
	$MP := STORE[MP + 2]$	Dynamic link

Instr.	Meaning	Cond.	Result
movs q	**for** $i := q - 1$ **down to** 0 **do**	(a)	
	$STORE[SP + i] := STORE[STORE[SP] + i]$		
	od;		
	$SP := SP + q - 1$		
movd q	**for** $i = 1$ **to** $STORE[MP + q + 1]$ **do**		
	$STORE[SP + i] :=$		
	$STORE[STORE[MP + q]$		
	$+ STORE[MP + q + 2] + i - 1]$		
	od;		
	$STORE[MP + q] := SP + 1 - STORE[MP + q + 2]$		
	$SP := SP + STORE[MP + q + 1]$		