

x	$::=$	<code>ALPHANUMERIC</code> <code>★ALPHANUMERIC</code>	<i>variables</i>
ℓ^*	$::=$	$\ell \mid *$	<i>general labels</i>
ℓ			<i>labels</i>
T	$::=$	$[t, \dots]$	<i>stack</i>
t	$::=$	$\langle \eta, \ell^*, S \rangle$	<i>stack frames</i>
S	$::=$	$[s, \dots]$	<i>programs</i>
s	$::=$	$\ell : \ell^* : d$	<i>clauses</i>
d	$::=$	<code>$x = e$</code> <code>return x</code> <code>goto ℓ</code> <code>goto ℓ if not x</code> <code>raise x</code> <code>catch x</code> <code>pass</code>	<i>directives</i>
B	$::=$	$\{x \mapsto m, \dots\}$	<i>bindings</i>
H	$::=$	$\{m \mapsto v, \dots\}$	<i>heap</i>
v	$::=$	$\mathbb{Z} \mid [m, \dots] \mid (m, \dots) \mid B \mid F \mid M \mid *$	<i>values</i>
e	$::=$	$v \mid x \mid \text{def } x(x, \dots) = \{S\} \mid x(x, \dots) \mid x.x \mid [x, \dots] \mid (x, \dots)$	<i>expressions</i>
Y	$::=$	$[y, \dots]$	<i>microcode stack</i>
Z	$::=$	$[z, \dots]$	<i>microcode literal stack</i>
y	$::=$	<code>STORE</code> <code>WRAP</code> <code>BIND</code> <code>LOOKUP</code> <code>LIST n</code> <code>TUPLE n</code> <code>ADVANCE</code> <code>POP</code> <code>PUSH S</code> <code>RAISE</code> <code>GOTO ℓ</code> <code>GOTOIFN ℓ</code> <code>IGNORE</code> <code>CALL n</code> <code>ALLOCNAMEERROR</code>	<i>microcode commands</i>
z	$::=$	$x \mid m \mid v$	<i>microcode literals</i>
P	$::=$	$m \mapsto m$	<i>parental map</i>
\hat{m}	$::=$	$m \mid \eta \mid *$	<i>general memory locations</i>
η, m	$::=$	<code><ADDRESS></code>	<i>memory locations</i>
F	$::=$	$\langle \eta, \text{def } (x) \rightarrow S \rangle \mid \mathfrak{F}$	<i>general functions</i>
M	$::=$	$\langle \eta, \eta, \text{def } (x) \rightarrow S \rangle \mid \langle \eta, \mathfrak{M} \rangle$	<i>general methods</i>
\mathfrak{F}			<i>magic functions</i>
\mathfrak{M}			<i>magic methods</i>
n			<i>integers</i>

Figure 1: Expression Grammar

$$\begin{array}{c}
\text{STORE } v \\
\frac{m \notin H \quad H' = H[m \mapsto v]}{P, Z \parallel [v, \text{STORE}] \parallel Y, T, H \longrightarrow^1 P, Z \parallel [m] \parallel Y, T, H'} \\
\\
\text{WRAP } m \\
\frac{v = \text{GETOBJ}(H, m)}{P, Z \parallel [m, \text{WRAP}] \parallel Y, T, H \longrightarrow^1 P, Z \parallel [v] \parallel Y, T, H} \\
\\
\text{BIND } m \text{ TO } x \\
\frac{B = H[\eta] \quad B' = B[x \mapsto m] \quad H' = H[\eta \mapsto B']}{P, Z \parallel [m, x, \text{BIND}] \parallel Y, T, H \longrightarrow^1 P, Z \parallel Y, T, H'} \\
\\
\text{ADVANCE} \\
\frac{S(\ell) = \ell : \ell^{\star'} : d \quad \ell \stackrel{s}{\blacktriangleleft} \ell^{\star''}}{P, Z \parallel [\text{ADVANCE}] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Z \parallel Y, [\langle \eta, \ell^{\star''}, S \rangle] \parallel T, H} \\
\\
\text{POP} \\
\frac{}{P, Z \parallel [\text{POP}] \parallel Y, t \parallel T, H \longrightarrow^1 P, Z \parallel Y, T, H} \\
\\
\text{LOOK UP } x \text{ (BOUND)} \\
\frac{\text{LOOKUP}(P, H, t, x) = m}{P, Z \parallel [x, \text{LOOKUP}] \parallel Y, T, H \longrightarrow^1 P, Z \parallel [m] \parallel Y, T, H} \\
\\
\text{LOOK UP } x \text{ (NAMEERROR)} \\
\frac{\text{LOOKUP}(P, H, t, x) = *}{P, Z \parallel [x, \text{LOOKUP}] \parallel Y, T, H \longrightarrow^1 P, [\text{ALLOCNAMEERROR}, \text{RAISE}], T, H} \\
\\
\text{MAKE LIST} \\
\frac{v = [m_1, \dots, m_n]}{P, Z \parallel [m_1, \dots, m_n, \text{LIST } n] \parallel Y, T, H \longrightarrow^1 P, Z \parallel [v] \parallel Y, T, H} \\
\\
\text{MAKE TUPLE} \\
\frac{v = (m_1, \dots, m_n)}{P, Z \parallel [m_1, \dots, m_n, \text{TUPLE } n] \parallel Y, T, H \longrightarrow^1 P, Z \parallel [v] \parallel Y, T, H} \\
\\
\text{RAISE (NO EXCEPTION LABEL)} \\
\frac{S(\ell) = \ell : * : d}{P, Z \parallel [\text{RAISE}] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Z \parallel [\text{POP}, \text{RAISE}] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H}
\end{array}$$

Figure 2: Microcommands

$$\begin{array}{c}
\text{RAISE (CAUGHT)} \\
\frac{S(\ell) = \ell : \ell_0 : d \quad S(\ell_0) = \ell_0 : \ell_1 : \text{catch } x \quad Y' = [x, \text{BIND}, \text{ADVANCE}]}{P, Z \parallel [\text{RAISE}] \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Z \parallel Y' \parallel Y, [\langle \eta, \ell_0, S \rangle] \parallel T, H} \\
\\
\text{GOTO } \ell \\
\frac{S(\ell) = \ell : \ell' : d}{P, Z \parallel [\text{GOTO } \ell] \parallel Y, [\langle \eta, \ell', S \rangle] \parallel T, H \longrightarrow^1 P, Z \parallel Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{GOTOIFN } \ell \text{ (SUCCESS)} \\
\frac{H[m] = \text{FALSE} \quad S(\ell) = \ell : \ell' : d}{P, Z \parallel [m, \text{GOTOIFN } \ell] \parallel Y, T, H \longrightarrow^1 P, Z \parallel [\text{GOTO}] \parallel Y, T, H} \\
\\
\text{GOTOIFN } \ell \text{ (FAILURE)} \\
\frac{H[m] = \text{TRUE}}{P, Z \parallel [m, \text{GOTOIFN } \ell] \parallel Y, T, H \longrightarrow^1 P, Z \parallel [\text{ADVANCE}] \parallel Y, T, H} \\
\\
\text{IGNORE } Z \\
\frac{}{P, Z \parallel [m, \text{IGNORE}] \parallel Y, T, H \longrightarrow^1 P, Z \parallel Y, T, H} \\
\\
\text{CALL FUNCTION} \\
\frac{H[m_0] = \langle \eta, \text{def } (x_1, \dots, x_2) \rightarrow S \rangle \quad Y' = [\text{PUSH } S, \text{omem}_1, x_1, \text{BIND}, \dots, \text{omem}_1, x_1, \text{BIND}]}{P, Z \parallel [m_0, \dots, m_n, \text{CALL } n] \parallel Y, T, H \longrightarrow^1 P, Z \parallel Y' \parallel Y, T, H}
\end{array}$$

Figure 3: Microcommands (cont.)

LITERAL ASSIGNMENT

$$\frac{S(\ell) = \ell : \ell^{\star'} : x = \mathbb{Z} \quad Y = [v, \text{STORE}, \text{WRAP}, \text{STORE}, x, \text{BIND}, \text{ADVANCE}]}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

(TODO: make literal category (ints, str, bool, None) – TC)

NAME ASSIGNMENT

$$\frac{S(\ell) = \ell : \ell^{\star'} : x_1 = x_2 \quad Y = [x_2, \text{LOOKUP}, x_1, \text{BIND}, \text{ADVANCE}]}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

LIST ASSIGNMENT

$$\frac{S(\ell) = \ell : \ell^{\star'} : x = [x_1, \dots, x_n] \quad Y = [(x_1, \text{LOOKUP}), \dots, (x_n, \text{LOOKUP}), \text{LIST } n, \text{STORE}, \text{WRAP}, \text{STORE}, x, \text{BIND}, \text{ADVANCE}]}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

(Parentheses in Y group instructions together for convenience of reading. – TC)

TUPLE ASSIGNMENT

$$\frac{S(\ell) = \ell : \ell^{\star'} : x = [x_1, \dots, x_n] \quad Y = [(x_1, \text{LOOKUP}), \dots, (x_n, \text{LOOKUP}), \text{TUPLE } n, \text{STORE}, \text{WRAP}, \text{STORE}, x, \text{BIND}, \text{ADVANCE}]}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

FUNCTIONDEF ASSIGNMENT

$$\frac{S(\ell) = \ell : \ell' : x = \text{def } (x_1, \dots, x_n) = \{S'\} \quad v = \langle \eta, \text{def } (x_1, \dots, x_n) \rightarrow S' \rangle \quad Y = [v, \text{STORE}, \text{WRAP}, \text{STORE}, x, \text{BIND}, \text{ADVANCE}]}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

Figure 4: Operational Semantics: Assignment

FUNCTION CALL

$$\frac{S(\ell) = \ell : \ell' : x_1 = x_2(x_3, \dots, x_n) \quad B = \{ \} \quad v = \langle \eta, \text{def } (x_1, \dots, x_n) \rightarrow S' \rangle \quad Y = [v, \text{STORE}, \text{WRAP}, \text{STORE}, x, \text{BIND}, \text{ADVANCE}]}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H}$$

Figure 5: Operational Semantics: Call

$$\begin{array}{c}
\text{PASS} \\
\frac{S(\ell) = \ell : \ell^{\star'} : \text{pass} \quad Y = [\text{ADVANCE}]}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{RETURN} \\
\frac{S(\ell) = \ell : \ell^{\star'} : \text{return } x \quad T = [\langle \eta', \ell'', S' \rangle] \parallel T' \quad S(\ell'') = \ell'' : \ell^{\star'''} : x' = e \quad Y = [x, \text{LOOKUP}, \text{POP}, x', \text{BIND}, \text{ADVANCE}]}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{GOTO} \\
\frac{S(\ell) = \ell : \ell^{\star'} : \text{goto } \ell'' \quad Y = [\text{GOTO } \ell'']}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{GOTOIFNOT} \\
\frac{S(\ell) = \ell : \ell' : \text{goto } \ell'' \text{ if not } x \quad Y = [x, \text{LOOKUP}, \text{GOTOIFN } \ell'']}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{NAME STATEMENT} \\
\frac{S(\ell) = \ell : \ell' : x \quad Y = [x_1, \text{LOOKUP}, \text{IGNORE}, \text{ADVANCE}]}{P, [], [\langle \eta, \ell, S \rangle] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, \ell, S \rangle] \parallel T, H} \\
\\
\text{END OF FUNCTION} \\
\frac{t = \langle \ell, S' \rangle \quad S(\ell) = \ell : \ell^{\star'} : x = e \quad Y = [m_{\text{None}}, x, \text{BIND}, \text{POP}, \text{ADVANCE}]}{P, [], [\langle \eta, *, S \rangle, t] \parallel T, H \longrightarrow^1 P, Y, [\langle \eta, *, S \rangle, t] \parallel T, H} \\
\\
\text{(} m_{\text{None}} \text{ is a memory location reserved for None. - TC)} \\
\\
\text{END OF PROGRAM} \\
\frac{}{P, [], [\langle \eta, *, S \rangle], H \longrightarrow^1 P, [], [], H}
\end{array}$$

Figure 6: Operational Semantics: Flow