`asm.js`

Dave Herman

October 19, 2012

1 Abstract syntax

$$
\begin{array}{rcl}
\rho & ::= & \textit{runtime lexical environment} \\
\Delta & ::= & \{\overline{f : (\overline{\sigma}) \to \tau}\} \\
\Gamma & ::= & \{\overline{x : \tau}\}
\end{array}
$$

$$
\begin{array}{rcl}
\textit{program} & ::= & \overline{\textit{fn}}\ \texttt{return}\ f\texttt{;} \\
& | & \overline{\textit{fn}}\ \texttt{return \{}\ \overline{x\texttt{:}f}\ \texttt{\};} \\
\textit{fn} & ::= & \texttt{function}\ f\texttt{(}\overline{x}\texttt{)}\ \texttt{\{}\ \overline{x\ \texttt{=}\ \kappa_x\texttt{;}}\ \texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}\ ss\ \texttt{\}}
\end{array}
$$

$$
\begin{array}{rcl}
s & ::= & \texttt{\{}\ ss\ \texttt{\}} \\
& | & e\texttt{;} \\
& | & \texttt{if (}e\texttt{)}\ s \\
& | & \texttt{if (}e\texttt{)}\ s\ \texttt{else}\ s \\
& | & \texttt{return}\ v\texttt{;} \\
& | & \texttt{while (}e\texttt{)}\ s \\
& | & \texttt{do}\ s\ \texttt{while (}e\texttt{);} \\
& | & \texttt{for (}e\texttt{;}\ e\texttt{;}\ e\texttt{)}\ s \\
& | & \texttt{switch (}e\texttt{) \{}\ \overline{c}\ \texttt{\}} \\
& | & \texttt{switch (}e\texttt{) \{}\ \overline{c}\ d\ \texttt{\}} \\
& | & \texttt{break;} \\
& | & \texttt{break}\ \textit{lab}\texttt{;} \\
& | & \texttt{continue;} \\
& | & \texttt{continue}\ \textit{lab}\texttt{;} \\
& | & \textit{lab}\texttt{:}\ s \\
\\
ss & ::= & \overline{s} \\
\\
c & ::= & \texttt{case}\ e\texttt{:}\ ss \\
d & ::= & \texttt{default:}\ ss \\
cd & ::= & c \mid d
\end{array}
$$

$$
\begin{array}{rcl}
\kappa_X & ::= & X\ \texttt{|}\ \texttt{0} \\
 & \mid & X\ \texttt{\&}\ \texttt{0xff} \\
 & \mid & X\ \texttt{\&}\ \texttt{0xffff} \\
 & \mid & X\ \texttt{>>>}\ \texttt{0} \\
 & \mid & \texttt{+}X \\
 & \mid & X\ \texttt{+}\ \texttt{''} \\
 & \mid & X\ \texttt{+}\ \texttt{""} \\
 & \mid & \texttt{!!}X
\end{array}
$$

$$
\begin{array}{rcl}
v & ::= & \kappa_{num} \\
 & \mid & bool \\
 & \mid & str \\
 & \mid & \texttt{null}
\end{array}
$$

$$
\begin{array}{rcl}
e & ::= & v \\
 & \mid & lval \\
 & \mid & lval\ \texttt{=}\ e \\
 & \mid & f\texttt{(}\overline{e}\texttt{)} \\
 & \mid & unop\ e \\
 & \mid & e\ binop\ e \\
 & \mid & e\ \texttt{?}\ e\ \texttt{:}\ e \\
 & \mid & \texttt{(}\overline{e}\texttt{)}
\end{array}
$$

$$
\begin{array}{rcl}
unop & ::= & \texttt{void} \mid \texttt{\textasciitilde} \mid \texttt{-} \mid \texttt{!}
\end{array}
$$

$$
\begin{array}{rcl}
binop & ::= & \texttt{<} \mid \texttt{<=} \mid \texttt{>} \mid \texttt{>=} \mid \texttt{==} \\
 & \mid & \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{\%} \\
 & \mid & \texttt{|} \mid \texttt{\&} \mid \texttt{\textasciicircum} \\
 & \mid & \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>} \\
 & \mid & \texttt{||} \mid \texttt{\&\&}
\end{array}
$$

$$
\begin{array}{rcl}
lval & ::= & x \\
 & \mid & x\texttt{[}e\texttt{]}
\end{array}
$$

2 Type rules

$$
\begin{array}{rcl}
\sigma, \tau & ::= & \texttt{boolean} \\
 & \mid & \texttt{uint8} \mid \texttt{uint16} \mid \texttt{uint32} \\
 & \mid & \texttt{int8} \mid \texttt{int16} \mid \texttt{int32} \\
 & \mid & \texttt{float32} \mid \texttt{float64} \\
 & \mid & \texttt{array}_\tau \mid \texttt{Function} \\
 & \mid & \texttt{any} \mid \texttt{undefined} \mid \texttt{null} \mid \texttt{string} \mid \texttt{number} \mid \texttt{object}
\end{array}
$$

$$
\begin{array}{rcl}
\ell & ::= & lab \mid \epsilon \\
L & ::= & \{\overline{\ell}\} \\
\varepsilon & ::= & L \mid \textsf{return}
\end{array}
$$

$$
\begin{array}{rcl}
L \mathbin{;} L' & = & L \cup L' \\
\emptyset \mathbin{;} \textsf{return} & = & \textsf{return} \\
\{\ell, \overline{\ell'}\} \mathbin{;} \textsf{return} & = & \{\ell, \overline{\ell'}\} \\
\textsf{return} \mathbin{;} L & = & \textsf{return}
\end{array}
$$

$$
\begin{array}{rcl}
L \cup \textsf{return} & = & L \\
\textsf{return} \cup L & = & L \\
\textsf{return} \cup \textsf{return} & = & \textsf{return}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{type}(\mathit{bool}) & = & \texttt{boolean} \\
\mathit{type}(\mathit{str}) & = & \texttt{string} \\
\mathit{type}(\mathit{null}) & = & \texttt{null} \\
\mathit{type}(X\ \texttt{\&}\ \texttt{0xff}) & = & \texttt{int8} \\
\mathit{type}(X\ \texttt{\&}\ \texttt{0xffff}) & = & \texttt{int16} \\
\mathit{type}(X\ \texttt{|}\ \texttt{0}) & = & \texttt{int32} \\
\mathit{type}(X\ \texttt{>>>}\ \texttt{0}) & = & \texttt{uint32} \\
\mathit{type}(\texttt{+}X) & = & \texttt{float64} \\
\mathit{type}(X\ \texttt{+}\ \texttt{''}) & = & \texttt{string} \\
\mathit{type}(X\ \texttt{+}\ \texttt{""}) & = & \texttt{string} \\
\mathit{type}(\texttt{!!}X) & = & \texttt{boolean}
\end{array}
$$

$$
\begin{array}{rcl}
\texttt{uint8}, \texttt{uint16}, \texttt{uint32} & <: & \texttt{int} \\
\texttt{int8}, \texttt{int16}, \texttt{int32} & <: & \texttt{int}
\end{array}
$$

Function checking $\boxed{\rho; \Delta \vdash \mathit{fn}\ \mathbf{ok}}$

[T-Function]
$$\frac{\rho; \Delta; \{\overline{x : \sigma}, \overline{y : \mathit{type}(v)}\}; \emptyset \vdash \mathit{ss} : \Delta(f)/\mathsf{return} \quad \overline{\forall i.\mathit{type}(\kappa_{x_i}) = \sigma_i}}{\rho; \Delta \vdash \texttt{function}\ f\texttt{(}\overline{x}\texttt{)}\ \texttt{\{}\ \overline{x\ \texttt{=}\ \kappa_x\texttt{;}}\ \texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}\ \mathit{ss}\ \texttt{\}}\ \mathbf{ok}}$$

Statement list checking $\boxed{\rho; \Delta; \Gamma; L \vdash \mathit{ss} : \tau/\varepsilon}$

[T-NoStatements]
$$\frac{}{\rho; \Delta; \Gamma; L \vdash \epsilon : \tau/\emptyset}$$

[T-Statements]
$$\frac{\forall i.\rho; \Delta; \Gamma; L \vdash s_i : \tau/\varepsilon_i \quad n > 0 \quad \varepsilon = \varepsilon_1\ ;\ \ldots\ ;\ \varepsilon_n}{\rho; \Delta; \Gamma; L \vdash \overline{s} : \tau/\varepsilon}$$

Statement checking $\boxed{\rho; \Delta; \Gamma; L \vdash s : \tau/\varepsilon}$

[T-Block]
$$\frac{\rho; \Delta; \Gamma; \emptyset \vdash \mathit{ss} : \tau/\varepsilon}{\rho; \Delta; \Gamma; L \vdash \texttt{\{}\ \mathit{ss}\ \texttt{\}} : \tau/\varepsilon}$$

[T-ExprStmt]
$$\frac{\rho; \Delta; \Gamma \vdash e : \sigma}{\rho; \Delta; \Gamma; L \vdash e\texttt{;} : \tau/\emptyset}$$

[T-If]
$$\frac{\rho; \Delta; \Gamma \vdash e : \texttt{boolean} \quad \rho; \Delta; \Gamma; \emptyset \vdash s : \tau/\varepsilon \quad \varepsilon' = \varepsilon \cup \emptyset}{\rho; \Delta; \Gamma; L \vdash \texttt{if (}e\texttt{)}\ s : \tau/\varepsilon'}$$

[T-IfElse]
$$\frac{\rho; \Delta; \Gamma \vdash e : \texttt{boolean} \quad \rho; \Delta; \Gamma; \emptyset \vdash s_1 : \tau/\varepsilon_1 \quad \rho; \Delta; \Gamma; \emptyset \vdash s_2 : \tau/\varepsilon_2 \quad \varepsilon = \varepsilon_1 \cup \varepsilon_2}{\rho; \Delta; \Gamma; L \vdash \texttt{if (}e\texttt{)}\ s_1\ \texttt{else}\ s_2 : \tau/\varepsilon}$$

[T-Return]
$$\frac{}{\rho; \Delta; \Gamma; L \vdash \texttt{return}\ v\texttt{;} : \mathit{type}(v)/\mathsf{return}}$$

[T-While]
$$\frac{\rho; \Delta; \Gamma \vdash e : \texttt{boolean} \quad \rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \quad \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\})}{\rho; \Delta; \Gamma; L \vdash \texttt{while (}e\texttt{)}\ s : \tau/\varepsilon'}$$

[T-DoWhile]
$$\frac{\rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \quad \rho; \Delta; \Gamma \vdash e : \texttt{boolean} \quad \varepsilon' = \varepsilon - (L \cup \{\epsilon\})}{\rho; \Delta; \Gamma; L \vdash \texttt{do}\ s\ \texttt{while (}e\texttt{);} : \tau/\varepsilon'}$$

[T-For]
$$\frac{\forall i \in \{1,2,3\}.\rho; \Delta; \Gamma \vdash e_i : \sigma_i \quad \rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \quad \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\})}{\rho; \Delta; \Gamma; L \vdash \texttt{for (}e_1\texttt{;}\ e_2\texttt{;}\ e_3\texttt{)}\ s : \tau/\varepsilon'}$$

Statement checking (cont'd) $\boxed{\rho; \Delta; \Gamma; L \vdash s : \tau/\varepsilon}$

[T-Break]
$$\frac{\varepsilon = \{\epsilon\}}{\rho; \Delta; \Gamma; L \vdash \texttt{break;} : \tau/\varepsilon}$$

[T-BreakLabel]
$$\frac{\varepsilon = \{lab\}}{\rho; \Delta; \Gamma; L \vdash \texttt{break}\ lab\texttt{;} : \tau/\varepsilon}$$

[T-Continue]
$$\frac{}{\rho; \Delta; \Gamma; L \vdash \texttt{continue;} : \tau/\emptyset}$$

[T-ContinueLabel]
$$\frac{}{\rho; \Delta; \Gamma; L \vdash \texttt{continue}\ lab\texttt{;} : \tau/\emptyset}$$

[T-Label]
$$\frac{\begin{array}{c}\rho; \Delta; \Gamma; L \cup \{lab\} \vdash s : \tau/\varepsilon \\ \varepsilon' = \varepsilon - (L \cup \{lab\})\end{array}}{\rho; \Delta; \Gamma; L \vdash lab\texttt{:}\ s : \tau/\varepsilon'}$$

[T-Switch]
$$\frac{\begin{array}{c}\rho; \Delta; \Gamma \vdash e : \sigma \\ \forall i.\rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau/\varepsilon_i \\ \rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash cd : \sigma, \tau/\varepsilon \\ \varepsilon \neq \mathsf{return} \vee \exists i.\varepsilon_i \cup \emptyset \neq \emptyset \\ \varepsilon' = (\varepsilon \cup \bigcup_i \varepsilon_i) - (L \cup \{\epsilon\})\end{array}}{\rho; \Delta; \Gamma; L \vdash \texttt{switch (}e\texttt{)}\ \{\ \overline{c}\ cd\ \} : \tau/\varepsilon'}$$

[T-SwitchReturn]
$$\frac{\begin{array}{c}\rho; \Delta; \Gamma \vdash e : \sigma \\ \forall i.\rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau/\varepsilon_i \\ \forall i.\varepsilon_i \cup \emptyset = \emptyset \\ \rho; \Delta; \Gamma; L \cup \{\epsilon\} \vdash cd : \sigma, \tau/\mathsf{return}\end{array}}{\rho; \Delta; \Gamma; L \vdash \texttt{switch (}e\texttt{)}\ \{\ \overline{c}\ cd\ \} : \tau/\mathsf{return}}$$

Case checking $\boxed{\rho; \Delta; \Gamma; L \vdash cd : \sigma, \tau/\varepsilon}$

[T-Case]
$$\frac{\begin{array}{c}\rho; \Delta; \Gamma \vdash e : \sigma \\ \rho; \Delta; \Gamma; L \vdash ss : \tau/\varepsilon\end{array}}{\rho; \Delta; \Gamma; L \vdash \texttt{case}\ e\texttt{:}\ ss : \sigma, \tau/\varepsilon}$$

[T-Default]
$$\frac{\rho; \Delta; \Gamma; L \vdash ss : \tau/\varepsilon}{\rho; \Delta; \Gamma; L \vdash \texttt{default:}\ ss : \sigma, \tau/\varepsilon}$$

Expression checking $\boxed{\rho; \Delta; \Gamma \vdash e : \tau}$

$$\text{[T-VALUE]}\quad \frac{}{\rho; \Delta; \Gamma \vdash v : \mathit{type}(v)}$$

$$\text{[T-VARREF]}\quad \frac{\Gamma(x) = \tau}{\rho; \Delta; \Gamma \vdash x : \tau}$$

$$\text{[T-ASSIGN]}\quad \frac{\Gamma(x) = \tau \qquad \rho; \Delta; \Gamma \vdash e : \tau}{\rho; \Delta; \Gamma \vdash x \texttt{ = } e : \tau}$$

$$\text{[T-LOAD]}\quad \frac{\begin{array}{c}\mathit{type}(\rho(x)) = \texttt{array}_\tau \\ \rho; \Delta; \Gamma \vdash e : \sigma \qquad \sigma <: \texttt{int}\end{array}}{\rho; \Delta; \Gamma \vdash x\texttt{[}e\texttt{]} : \tau}$$

$$\text{[T-STORE]}\quad \frac{\begin{array}{c}\mathit{type}(\rho(x)) = \texttt{array}_\tau \\ \rho; \Delta; \Gamma \vdash e_1 : \sigma \qquad \sigma <: \texttt{int} \\ \rho; \Delta; \Gamma \vdash e_2 : \tau\end{array}}{\rho; \Delta; \Gamma \vdash x\texttt{[}e_1\texttt{] = } e_2 : \tau}$$

$$\text{[T-FUNCALL]}\quad \frac{\begin{array}{c}\Delta(f) = (\overline{\sigma}) \to \tau \\ \forall i . \rho; \Delta; \Gamma \vdash e_i : \sigma_i\end{array}}{\rho; \Delta; \Gamma \vdash f\texttt{(}\overline{e}\texttt{)} : \tau}$$

$$\text{[T-FFI]}\quad \frac{\begin{array}{c}f \notin \mathit{dom}(\Delta) \qquad f \notin \mathit{dom}(\Gamma) \\ \mathit{type}(\rho(f)) = \texttt{Function} \\ \forall i . \rho; \Delta; \Gamma \vdash e_i : \sigma_i\end{array}}{\rho; \Delta; \Gamma \vdash f\texttt{(}\overline{e}\texttt{)} : \texttt{any}}$$

$$\text{[T-UNARYOP]}\quad \frac{\mathit{type}(\mathit{unop}) = \sigma \to \tau \qquad \rho; \Delta; \Gamma \vdash e : \sigma}{\rho; \Delta; \Gamma \vdash \mathit{unop}\ e : \tau}$$

$$\text{[T-BINARYOP]}\quad \frac{\forall i \in \{1, 2\} . \rho; \Delta; \Gamma \vdash e_i : \sigma_i \qquad \mathit{type}(\mathit{binop}) = \sigma_1 \times \sigma_2 \to \tau}{\rho; \Delta; \Gamma \vdash e_1\ \mathit{binop}\ e_2 : \tau}$$

$$\text{[T-CONDITIONAL]}\quad \frac{\begin{array}{c}\rho; \Delta; \Gamma \vdash e_1 : \texttt{boolean} \\ \forall i \in \{2, 3\} . \rho; \Delta; \Gamma \vdash e_i : \tau\end{array}}{\rho; \Delta; \Gamma \vdash e_1 \texttt{ ? } e_2 \texttt{ : } e_3 : \tau}$$

$$\text{[T-PAREN]}\quad \frac{\forall i \leq n . \rho; \Delta; \Gamma \vdash e_i : \tau_i}{\rho; \Delta; \Gamma \vdash \texttt{(}\overline{e}\texttt{)} : \tau_n}$$