asm.js

Dave Herman, Luke Wagner, and Alon Zakai

November 5, 2012

1 Abstract syntax

$$
\begin{aligned}
b, e, f, g, x, y, z &\in & Identifier \\
\texttt{arguments}, \texttt{eval} &\notin & Identifier
\end{aligned}
$$

$$
\begin{aligned}
P &::=& \texttt{function}(b, e) \ \{ \ \overline{imp_x} \ \overline{fn_f} \ exp \ \} \\
imp_x &::=& \texttt{var} \ x \ \texttt{=} \ e.y\texttt{;} \\
&|& \texttt{var} \ x \ \texttt{=} \ \texttt{new} \ e.y(b)\texttt{;} \\
exp &::=& \texttt{return} \ f\texttt{;} \\
&|& \texttt{return} \ \{ \ \overline{x\texttt{:}f} \ \}\texttt{;} \\
fn_f &::=& \texttt{function} \ f(\overline{x}) \ \{ \ \overline{x \ \texttt{=} \ \kappa_x\texttt{;}} \ \texttt{var} \ \overline{y \ \texttt{=} \ v}\texttt{;} \ ss \ \}
\end{aligned}
$$

$$
\begin{aligned}
s &::=& \{ \ ss \ \} \\
&|& e\texttt{;} \\
&|& \texttt{;} \\
&|& \texttt{if} \ (e) \ s \\
&|& \texttt{if} \ (e) \ s \ \texttt{else} \ s \\
&|& \texttt{return} \ e\texttt{;} \\
&|& \texttt{while} \ (e) \ s \\
&|& \texttt{do} \ s \ \texttt{while} \ (e)\texttt{;} \\
&|& \texttt{for} \ (e^?\texttt{;} \ e^?\texttt{;} \ e^?) \ s \\
&|& \texttt{switch} \ (e) \ \{ \ \overline{c} \ \} \\
&|& \texttt{switch} \ (e) \ \{ \ \overline{c} \ d \ \} \\
&|& \texttt{break;} \\
&|& \texttt{break} \ lab\texttt{;} \\
&|& \texttt{continue;} \\
&|& \texttt{continue} \ lab\texttt{;} \\
&|& lab\texttt{:}\, s
\end{aligned}
$$

$$
ss \ ::= \ \overline{s}
$$

$$
\begin{aligned}
c &::=& \texttt{case} \ e\texttt{:}\, ss \\
d &::=& \texttt{default:}\, ss \\
cd &::=& c \mid d
\end{aligned}
$$

$$\kappa_x ::= \texttt{\~{}\~{}}x \mid \texttt{+}x \mid x\texttt{|0} \mid x\texttt{>>>0}$$

$$v \quad ::= \quad r \mid n$$

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

$$unop \quad ::= \quad \texttt{+} \mid \texttt{\~{}} \mid \texttt{!}$$

$$
\begin{aligned}
binop \quad ::= \quad & \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{\%} \\
\mid \quad & \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>} \\
\mid \quad & \texttt{<} \mid \texttt{<=} \mid \texttt{>} \mid \texttt{>=} \mid \texttt{!=} \mid \texttt{==} \\
lval \quad ::= \quad & x \mid x\texttt{[(}e\ \texttt{\&}\ m\texttt{)}\ \texttt{>>}\ n\texttt{]}
\end{aligned}
$$

2 Type rules

$$
\begin{aligned}
\sigma, \tau \quad ::= \quad & \texttt{bit} \mid \texttt{int} \mid \texttt{boolish} \\
\mid \quad & \texttt{signed} \mid \texttt{unsigned} \\
\mid \quad & \texttt{double} \\
\mid \quad & \texttt{array}^n_\tau \mid \texttt{function} \mid \texttt{unknown} \mid \texttt{jsval} \\
\mid \quad & \texttt{intish} \\
\mid \quad & \texttt{imul} \\
\mid \quad & ((\overline{\sigma}) \to \tau) \land \ldots \land ((\overline{\sigma'}) \to \tau') \\
\mid \quad & \texttt{void}
\end{aligned}
$$

$$
\begin{aligned}
\ell \quad &::= \quad lab \mid \epsilon \\
L \quad &::= \quad \{\overline{\ell}\} \\
\varepsilon \quad &::= \quad L \mid \texttt{return}
\end{aligned}
$$

$$
\begin{aligned}
L \ \texttt{;}\ L' \quad &= \quad L \cup L' \\
\emptyset \ \texttt{;}\ \texttt{return} \quad &= \quad \texttt{return} \\
\{\ell, \overline{\ell'}\} \ \texttt{;}\ \texttt{return} \quad &= \quad \{\ell, \overline{\ell'}\} \\
\texttt{return}\ \texttt{;}\ L \quad &= \quad \texttt{return}
\end{aligned}
$$

$$
\begin{aligned}
L \cup \texttt{return} \quad &= \quad L \\
\texttt{return} \cup L \quad &= \quad L \\
\texttt{return} \cup \texttt{return} \quad &= \quad \texttt{return}
\end{aligned}
$$

$$
\begin{aligned}
type(\texttt{\~{}\~{}}X) &= \texttt{int} \\
type(\texttt{+}X) &= \texttt{double} \\
type(n) &= \texttt{int} \\
type(r) &= \texttt{double} \\
type(X\texttt{|0}) &= \texttt{signed} \\
type(X\texttt{>>>0}) &= \texttt{unsigned}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{constant} &<: \texttt{signed}, \texttt{unsigned} \\
\texttt{signed}, \texttt{unsigned} &<: \texttt{int}, \texttt{jsval} \\
\texttt{bit}, \texttt{int} &<: \texttt{boolish} \\
\texttt{void}, \texttt{double}, \texttt{array}^n_\tau, \texttt{function}, \texttt{unknown} &<: \texttt{jsval} \\
\texttt{unknown}, \texttt{int} &<: \texttt{intish} \\
((\overline{\sigma}_1) \to \tau_1) \wedge \ldots \wedge ((\overline{\sigma}_n) \to \tau_n) &<: ((\overline{\sigma}_1) \to \tau_1) \wedge \ldots \wedge ((\overline{\sigma}_{n-1}) \to \tau_{n-1}) \\
\texttt{imul} &<: (\texttt{intish}, \texttt{intish}) \to \texttt{signed}
\end{aligned}
$$

$$
\begin{aligned}
M(\texttt{imul}) &: \texttt{imul} \\
M(\texttt{ceil}), M(\texttt{sin}), M(\texttt{cos}) &: (\texttt{double}) \to \texttt{double}
\end{aligned}
$$

$$
\begin{aligned}
A(\texttt{Uint8Array}), A(\texttt{Int8Array}) &= \texttt{array}^8_{\texttt{int}} \\
A(\texttt{Uint16Array}), A(\texttt{Int16Array}) &= \texttt{array}^{16}_{\texttt{int}} \\
A(\texttt{Uint32Array}), A(\texttt{Int32Array}) &= \texttt{array}^{32}_{\texttt{int}} \\
A(\texttt{Float32Array}) &= \texttt{array}^{32}_{\texttt{double}} \\
A(\texttt{Float64Array}) &= \texttt{array}^{64}_{\texttt{double}}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{+}, \texttt{-} \quad &: \quad (\texttt{double}, \texttt{double}) \to \texttt{double} \\
&\quad \wedge (\texttt{int}, \texttt{int}) \to \texttt{intish} \\
\texttt{*} \quad &: \quad (\texttt{double}, \texttt{double}) \to \texttt{double} \\
\texttt{/}, \texttt{\%} \quad &: \quad (\texttt{double}, \texttt{double}) \to \texttt{double} \\
&\quad \wedge (\texttt{signed}, \texttt{signed}) \to \texttt{intish} \\
&\quad \wedge (\texttt{unsigned}, \texttt{unsigned}) \to \texttt{intish}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{|}, \texttt{\&}, \texttt{\^{}}, \texttt{<<}, \texttt{>>} \quad &: \quad (\texttt{intish}, \texttt{intish}) \to \texttt{signed} \\
\texttt{>>>} \quad &: \quad (\texttt{intish}, \texttt{intish}) \to \texttt{unsigned}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{<}, \texttt{<=}, \texttt{>}, \texttt{>=}, \texttt{==}, \texttt{!=} \quad &: \quad (\texttt{signed}, \texttt{signed}) \to \texttt{bit} \\
&\quad \wedge (\texttt{unsigned}, \texttt{unsigned}) \to \texttt{bit} \\
&\quad \wedge (\texttt{double}, \texttt{double}) \to \texttt{bit}
\end{aligned}
$$

$$
\begin{aligned}
\texttt{+} \quad &: \quad (\texttt{intish}) \to \texttt{double} \\
\texttt{\~{}} \quad &: \quad (\texttt{intish}) \to \texttt{signed} \\
\texttt{!} \quad &: \quad (\texttt{boolish}) \to \texttt{bit}
\end{aligned}
$$

$$\Gamma ::= \{\overline{x : \tau}\} \mid \Gamma, \{\overline{x : \tau}\}$$

Program checking $\boxed{\vdash P \textbf{ ok}}$

[T-Program]
$$\frac{\begin{array}{ccc} \{\overline{x}\} \cap \{\overline{f}\} = \emptyset & \{\overline{x}\} \cap \{b, e\} = \emptyset & \{\overline{f}\} \cap \{b, e\} = \emptyset \\ \forall i.b; e; \Gamma_0 \vdash imp_x \textbf{ ok} & \forall i.\Gamma_0, \Gamma_1 \vdash fn_f \textbf{ ok} & \forall i.\Gamma_0, \Gamma_1 \vdash exp \textbf{ ok} \end{array}}{\vdash \texttt{function}(b, e) \ \{ \ \overline{imp_x} \ \overline{fn_f} \ exp \ \} \textbf{ ok}}$$

Import checking $\boxed{b; e; \Gamma \vdash imp \textbf{ ok}}$

[T-ImportStd]
$$\frac{\Gamma(x) = M(y)}{b; e; \Gamma \vdash \texttt{var} \ x \ \texttt{=} \ e.y\texttt{;} \textbf{ ok}}$$

[T-ImportFFI]
$$\frac{y \notin dom(M) \qquad \Gamma(x) = \texttt{function}}{b; e; \Gamma \vdash \texttt{var} \ x \ \texttt{=} \ e.y\texttt{;} \textbf{ ok}}$$

[T-View]
$$\frac{\Gamma(x) = \texttt{array}^n_{A(y)}}{b; e; \Gamma \vdash \texttt{var} \ x \ \texttt{=} \ e.y(b)\texttt{;} \textbf{ ok}}$$

[T-NewView]
$$\frac{\Gamma(x) = \texttt{array}^n_{A(y)}}{b; e; \Gamma \vdash \texttt{var} \ x \ \texttt{=} \ \texttt{new} \ e.y(b)\texttt{;} \textbf{ ok}}$$

Function checking $\boxed{\Gamma \vdash fn \textbf{ ok}}$

[T-Function]
$$\frac{\begin{array}{cccc} \{\overline{x}\} \cap \{\overline{y}\} = \emptyset & \Gamma(f) = (\overline{\sigma}) \to \tau & \overline{\sigma} = \overline{type(\kappa_x)} & \tau \neq \texttt{void} \end{array} \qquad \Gamma, \{\overline{x : \sigma}, \overline{y : type(v)}\}; \emptyset \vdash ss : \tau/\texttt{return}}{\Gamma \vdash \texttt{function} \ f(\overline{x}) \ \{ \ \overline{x \ \texttt{=} \ \kappa_x\texttt{;}} \ \texttt{var} \ \overline{y \ \texttt{=} \ v}\texttt{;} \ ss \ \} \textbf{ ok}}$$

[T-VoidFunction]
$$\frac{\begin{array}{ccc} \{\overline{x}\} \cap \{\overline{y}\} = \emptyset & \Gamma(f) = (\overline{\sigma}) \to \texttt{void} & \overline{\sigma} = \overline{type(\kappa_x)} \end{array} \qquad \Gamma, \{\overline{x : \sigma}, \overline{y : type(v)}\}; \emptyset \vdash ss : \texttt{void}/\varepsilon}{\Gamma \vdash \texttt{function} \ f(\overline{x}) \ \{ \ \overline{x \ \texttt{=} \ \kappa_x\texttt{;}} \ \texttt{var} \ \overline{y \ \texttt{=} \ v}\texttt{;} \ ss \ \} \textbf{ ok}}$$

Export checking $\boxed{\Gamma \vdash exp \textbf{ ok}}$

[T-Singleton]
$$\frac{\begin{array}{c} \Gamma(f) = (\overline{\sigma}) \to \tau \\ \forall i.\sigma_i <: \texttt{jsval} \qquad \tau <: \texttt{jsval} \end{array}}{\Gamma \vdash \texttt{return} \ f\texttt{;} \textbf{ ok}}$$

[T-Module]
$$\frac{\forall f.(\Gamma(f) = (\overline{\sigma}) \to \tau \wedge \forall i.\sigma_i <: \texttt{jsval} \wedge \tau <: \texttt{jsval})}{\Gamma \vdash \texttt{return} \ \{ \ \overline{x\texttt{:}f} \ \}\texttt{;} \textbf{ ok}}$$

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

[T-Statements]
$$\forall i.\Gamma; L \vdash s_i : \tau/\varepsilon_i$$

[T-NoStatements]
$$\frac{n > 0 \qquad \varepsilon = \varepsilon_1 \; ; \ldots ; \varepsilon_n}{\Gamma; L \vdash \epsilon : \tau/\emptyset} \qquad \frac{}{\Gamma; L \vdash \overline{s} : \tau/\varepsilon}$$

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

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

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

[T-If]
$$\frac{\begin{array}{c} \Gamma \vdash e : \texttt{boolish} \\ \Gamma; \emptyset \vdash s : \tau/\varepsilon \\ \varepsilon' = \varepsilon \cup \emptyset \end{array}}{\Gamma; L \vdash \texttt{if } (e)\ s : \tau/\varepsilon'}$$

[T-IfElse]
$$\frac{\begin{array}{c} \Gamma \vdash e : \texttt{boolish} \\ \Gamma; \emptyset \vdash s_1 : \tau/\varepsilon_1 \qquad \Gamma; \emptyset \vdash s_2 : \tau/\varepsilon_2 \\ \varepsilon = \varepsilon_1 \cup \varepsilon_2 \end{array}}{\Gamma; L \vdash \texttt{if } (e)\ s_1\ \texttt{else } s_2 : \tau/\varepsilon}$$

[T-ReturnExpr]
$$\frac{type(e) <: \tau \qquad \Gamma \vdash e : \tau}{\Gamma; L \vdash \texttt{return } e; : \tau/\texttt{return}}$$

[T-ReturnVoid]
$$\frac{}{\Gamma; L \vdash \texttt{return}; : \texttt{void}/\texttt{return}}$$

[T-While]
$$\frac{\begin{array}{c} \Gamma \vdash e : \texttt{boolish} \\ \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Gamma; L \vdash \texttt{while } (e)\ s : \tau/\varepsilon'}$$

[T-DoWhile]
$$\frac{\begin{array}{c} \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \Gamma \vdash e : \texttt{boolish} \\ \varepsilon' = \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Gamma; L \vdash \texttt{do } s\ \texttt{while } (e); : \tau/\varepsilon'}$$

[T-For]
$$\frac{\begin{array}{c} e_1^? = \varepsilon \vee \Gamma \vdash e_1^? : \sigma_1 \qquad e_2^? = \varepsilon \vee \Gamma \vdash e_2^? : \texttt{boolish} \qquad e_3^? = \varepsilon \vee \Gamma \vdash e_3^? : \sigma_3 \\ \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \qquad \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Gamma; L \vdash \texttt{for } (e_1;\ e_2;\ e_3)\ s : \tau/\varepsilon'}$$

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

[T-Break]

$$\frac{}{\Gamma; L \vdash \mathtt{break;} : \tau/\{\epsilon\}}$$

[T-BreakLabel]

$$\frac{}{\Gamma; L \vdash \mathtt{break}\ lab\mathtt{;} : \tau/\{lab\}}$$

[T-Continue]

$$\frac{}{\Gamma; L \vdash \mathtt{continue;} : \tau/\emptyset}$$

[T-ContinueLabel]

$$\frac{}{\Gamma; L \vdash \mathtt{continue}\ lab\mathtt{;} : \tau/\emptyset}$$

[T-Label]

$$\frac{\Gamma; L \cup \{lab\} \vdash s : \tau/\varepsilon \quad \varepsilon' = \varepsilon - (L \cup \{lab\})}{\Gamma; L \vdash lab\mathtt{:}\ s : \tau/\varepsilon'}$$

[T-Switch]

$$\frac{\begin{array}{c} \Gamma \vdash e : \sigma \\ \forall i.\Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau/\varepsilon_i \\ \Gamma; L \cup \{\epsilon\} \vdash cd : \sigma, \tau/\varepsilon \\ \varepsilon \neq \mathsf{return} \vee \exists i.\varepsilon_i \cup \emptyset \neq \emptyset \\ \varepsilon' = (\varepsilon \cup \bigcup_i \varepsilon_i) - (L \cup \{\epsilon\}) \end{array}}{\Gamma; L \vdash \mathtt{switch}\ (e)\ \{\ \overline{c}\ cd\ \} : \tau/\varepsilon'}$$

[T-SwitchReturn]

$$\frac{\begin{array}{c} \Gamma \vdash e : \sigma \\ \forall i.\Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau/\varepsilon_i \\ \forall i.\varepsilon_i \cup \emptyset = \emptyset \\ \Gamma; L \cup \{\epsilon\} \vdash cd : \sigma, \tau/\mathsf{return} \end{array}}{\Gamma; L \vdash \mathtt{switch}\ (e)\ \{\ \overline{c}\ cd\ \} : \tau/\mathsf{return}}$$

[T-EmptySwitch]

$$\frac{\Gamma \vdash e : \sigma}{\Gamma; L \vdash \mathtt{switch}\ (e)\ \{\ \ \} : \tau/\emptyset}$$

[T-EmptyStatement]

$$\frac{}{\Gamma; L \vdash \mathtt{;} : \tau/\emptyset}$$

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

[T-Case]

$$\frac{\Gamma \vdash e : \sigma \quad \Gamma; L \vdash ss : \tau/\varepsilon}{\Gamma; L \vdash \mathtt{case}\ e\mathtt{:}\ ss : \sigma, \tau/\varepsilon}$$

[T-Default]

$$\frac{\Gamma; L \vdash ss : \tau/\varepsilon}{\Gamma; L \vdash \mathtt{default:}\ ss : \sigma, \tau/\varepsilon}$$

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

[T-Constant]
$$\frac{}{\Gamma \vdash n : \texttt{constant}}$$

[T-Double]
$$\frac{}{\Gamma \vdash r : \texttt{double}}$$

[T-VarRef]
$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}$$

[T-Assign]
$$\frac{\Gamma \vdash e : \Gamma(x)}{\Gamma \vdash x \texttt{ = } e : \tau}$$

[T-Load]
$$\frac{m = 2^k - 1 \qquad \Gamma(x) = \texttt{array}_\tau^n \qquad \Gamma \vdash e : \texttt{int}}{\Gamma \vdash x\texttt{[(}e \texttt{ \& } m\texttt{) >> } n/8\texttt{]} : \tau}$$

[T-Store]
$$\frac{m = 2^k - 1 \qquad \Gamma(x) = \texttt{array}_\tau^n \qquad \Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash x\texttt{[(}e_1 \texttt{ \& } m\texttt{) >> } n/8\texttt{]} \texttt{ = } e_2 : \tau}$$

[T-FunCall]
$$\frac{\Gamma(f) = (\overline{\sigma}) \to \tau \qquad \forall i.\Gamma \vdash e_i : \sigma_i}{\Gamma \vdash f(\overline{e}) : \tau}$$

[T-FFI]
$$\frac{\Gamma(f) = \texttt{function} \qquad \forall i.\Gamma \vdash e_i : \texttt{jsval}}{\Gamma \vdash f(\overline{e}) : \texttt{unknown}}$$

[T-Conditional]
$$\frac{\Gamma \vdash e_1 : \texttt{boolish} \qquad \Gamma \vdash e_2 : \tau \qquad \Gamma \vdash e_3 : \tau}{\Gamma \vdash e_1 \texttt{ ? } e_2 \texttt{ : } e_3 : \tau}$$

[T-Paren]
$$\frac{\forall i \le n.\Gamma \vdash e_i : \tau_i}{\Gamma \vdash (\overline{e}) : \tau_n}$$

[T-Unop]
$$\frac{unop : \_ \wedge (\sigma) \to \tau \wedge \_ \qquad \Gamma \vdash e : \sigma}{\Gamma \vdash unop\ e : \tau}$$

[T-Binop]
$$\frac{binop : \_ \wedge (\sigma_1, \sigma_2) \to \tau \wedge \_ \qquad \Gamma \vdash e_1 : \sigma_1 \qquad \Gamma \vdash e_2 : \sigma_2}{\Gamma \vdash e_1\ binop\ e_2 : \tau}$$

[T-Sub]
$$\frac{\Gamma \vdash e : \sigma \qquad \sigma <: \tau}{\Gamma \vdash e : \tau}$$

[T-Cast]
$$\frac{\Gamma \vdash e : \texttt{double}}{\Gamma \vdash \texttt{\~{}\~{}}e : \texttt{signed}}$$