asm.js

Dave Herman, Luke Wagner, and Alon Zakai

November 5, 2012

1 Abstract syntax

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

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

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

$$
\begin{array}{rcl}
ss & ::= & \overline{s}
\end{array}
$$

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

$$\kappa_X ::= \texttt{~~}X \mid \texttt{+}X$$

$$\begin{array}{rcl} v & ::= & r \mid n \\ \\ e & ::= & v \\ & \mid & \mathit{lval} \\ & \mid & \mathit{lval}\ \texttt{=}\ e \\ & \mid & f\texttt{(}\overline{e}\texttt{)} \\ & \mid & \mathit{unop}\ e \\ & \mid & e\ \mathit{binop}\ e \\ & \mid & e\ \texttt{?}\ e\ \texttt{:}\ e \\ & \mid & \texttt{(}\overline{e}\texttt{)} \\ \\ \mathit{unop} & ::= & \texttt{+} \mid \texttt{~} \mid \texttt{!} \\ \\ \mathit{binop} & ::= & \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{\%} \\ & \mid & \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>} \\ & \mid & \texttt{<} \mid \texttt{<=} \mid \texttt{>} \mid \texttt{>=} \mid \texttt{!=} \mid \texttt{==} \\ \mathit{lval} & ::= & x \mid x\texttt{[(}e\ \texttt{\&}\ m\texttt{)}\ \texttt{>>}\ n\texttt{]} \end{array}$$

2 Type rules

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

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

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

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

$$
\begin{array}{rcl}
type(\texttt{\~{}\~{}}X) & = & \texttt{int} \\
type(\texttt{+}X) & = & \texttt{double} \\
type(n) & = & \texttt{int} \\
type(r) & = & \texttt{double}
\end{array}
$$

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

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

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

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

$$
\begin{array}{rcl}
\texttt{+}, \texttt{-} & : & (\texttt{double}, \texttt{double}) \to \texttt{double} \\
 & & \wedge\ (\texttt{int}, \texttt{int}) \to \texttt{intish} \\
\texttt{*} & : & (\texttt{double}, \texttt{double}) \to \texttt{double} \\
\texttt{/}, \texttt{\%} & : & (\texttt{double}, \texttt{double}) \to \texttt{double} \\
 & & \wedge\ (\texttt{signed}, \texttt{signed}) \to \texttt{intish} \\
 & & \wedge\ (\texttt{unsigned}, \texttt{unsigned}) \to \texttt{intish} \\
 & & \\
\texttt{|}, \texttt{\&}, \texttt{\^{}}, \texttt{<<}, \texttt{>>} & : & (\texttt{intish}, \texttt{intish}) \to \texttt{signed} \\
\texttt{>>>} & : & (\texttt{intish}, \texttt{intish}) \to \texttt{unsigned} \\
 & & \\
\texttt{<}, \texttt{<=}, \texttt{>}, \texttt{>=}, \texttt{==}, \texttt{!=} & : & (\texttt{signed}, \texttt{signed}) \to \texttt{bit} \\
 & & \wedge\ (\texttt{unsigned}, \texttt{unsigned}) \to \texttt{bit} \\
 & & \wedge\ (\texttt{double}, \texttt{double}) \to \texttt{bit} \\
 & & \\
\texttt{+} & : & (\texttt{intish}) \to \texttt{double} \\
\texttt{\~{}} & : & (\texttt{intish}) \to \texttt{signed} \\
\texttt{!} & : & (\texttt{boolish}) \to \texttt{bit}
\end{array}
$$

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

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

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

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

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

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

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

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

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

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

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

[T-Singleton]
$$\frac{\Gamma(f) = (\overline{\sigma}) \to \tau \qquad \tau <: \texttt{jsval}}{\Gamma \vdash \texttt{return } f\texttt{;} \ \mathbf{ok}}$$

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

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

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

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

$$\frac{\Gamma; \emptyset \vdash ss : \tau/\varepsilon}{\Gamma; L \vdash \{\ ss\ \} : \tau/\varepsilon}\ [\text{T-Block}] \qquad \frac{\Gamma \vdash e : \sigma}{\Gamma; L \vdash e\texttt{;} : \tau/\emptyset}\ [\text{T-ExprStmt}]$$

$$\frac{\Gamma \vdash e : \texttt{boolish} \quad \Gamma; \emptyset \vdash s : \tau/\varepsilon \quad \varepsilon' = \varepsilon \cup \emptyset}{\Gamma; L \vdash \texttt{if (}e\texttt{)}\ s : \tau/\varepsilon'}\ [\text{T-If}] \qquad \frac{\Gamma \vdash e : \texttt{boolish} \quad \Gamma; \emptyset \vdash s_1 : \tau/\varepsilon_1 \quad \Gamma; \emptyset \vdash s_2 : \tau/\varepsilon_2 \quad \varepsilon = \varepsilon_1 \cup \varepsilon_2}{\Gamma; L \vdash \texttt{if (}e\texttt{)}\ s_1\ \texttt{else}\ s_2 : \tau/\varepsilon}\ [\text{T-IfElse}]$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma; L \vdash \texttt{return}\ e\texttt{;} : \tau/\mathsf{return}}\ [\text{T-ReturnExpr}] \qquad \frac{}{\Gamma; L \vdash \texttt{return;} : \texttt{void}/\mathsf{return}}\ [\text{T-ReturnVoid}]$$

$$\frac{\Gamma \vdash e : \texttt{boolish} \quad \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \quad \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\})}{\Gamma; L \vdash \texttt{while (}e\texttt{)}\ s : \tau/\varepsilon'}\ [\text{T-While}] \qquad \frac{\Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \quad \Gamma \vdash e : \texttt{boolish} \quad \varepsilon' = \varepsilon - (L \cup \{\epsilon\})}{\Gamma; L \vdash \texttt{do}\ s\ \texttt{while (}e\texttt{);} : \tau/\varepsilon'}\ [\text{T-DoWhile}]$$

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

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

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

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

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

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

[T-Label]
$$\frac{\Gamma; L \cup \{lab\} \vdash s : \tau/\varepsilon \quad \varepsilon' = \varepsilon - (L \cup \{lab\})}{\Gamma; L \vdash lab\texttt{:}\ s : \tau/\varepsilon'}$$

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

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

[T-EmptySwitch]
$$\frac{\Gamma \vdash e : \sigma}{\Gamma; L \vdash \texttt{switch (}e\texttt{) \{ \}} : \tau/\emptyset}$$

[T-EmptyStatement]
$$\frac{}{\Gamma; L \vdash \texttt{;} : \tau/\emptyset}$$

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

[T-Case]
$$\frac{\Gamma \vdash e : \sigma \quad \Gamma; L \vdash ss : \tau/\varepsilon}{\Gamma; L \vdash \texttt{case}\ e\texttt{:}\ ss : \sigma, \tau/\varepsilon}$$

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

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

[T-Signed] $\dfrac{}{\Gamma \vdash n : \texttt{signed}}$ [T-Unsigned] $\dfrac{}{\Gamma \vdash n : \texttt{unsigned}}$ [T-Double] $\dfrac{}{\Gamma \vdash r : \texttt{double}}$

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

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

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

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

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

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

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

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

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

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

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