asm.js

Dave Herman, Luke Wagner, and Alon Zakai

October 26, 2012

1 Abstract syntax

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

$$
\begin{array}{rcl}
P & ::= & \texttt{function}(b, e)\ \{\ \overline{\mathit{imp}_x}\ \overline{\mathit{fn}_f}\ \mathit{exp}\ \} \\
\mathit{imp}_x & ::= & \texttt{var}\ x\ \texttt{=}\ e\texttt{.}y\texttt{;} \\
 & \mid & \texttt{var}\ x\ \texttt{=}\ e\texttt{.}y(b)\texttt{;} \\
 & \mid & \texttt{var}\ x\ \texttt{=}\ \texttt{new}\ e\texttt{.}y(b)\texttt{;} \\
\mathit{exp} & ::= & \texttt{return}\ f\texttt{;} \\
 & \mid & \texttt{return}\ \{\ \overline{x\texttt{:}f}\ \}\texttt{;} \\
\mathit{fn}_f & ::= & \texttt{function}\ f\texttt{(}\overline{x}\texttt{)}\ \{\ \overline{x\ \texttt{=}\ \kappa_x\texttt{;}}\ \texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}\ \mathit{ss}\ \}
\end{array}
$$

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

$$
\begin{array}{rcl}
\kappa_X & ::= & X\ \mathtt{|}\ \mathtt{0} \\
 & | & X\ \mathtt{>>>}\ \mathtt{0} \\
 & | & \mathtt{+}X \\
 & | & \mathtt{[}X\mathtt{[0]}\ \mathtt{>>>}\ \mathtt{0}, X\mathtt{[1]}\ \mathtt{|}\ \mathtt{0]} \\
 & | & \mathtt{[}X\mathtt{[0]}\ \mathtt{>>>}\ \mathtt{0}, X\mathtt{[1]}\ \mathtt{>>>}\ \mathtt{0]}
\end{array}
$$

$$
\begin{array}{rcl}
v & ::= & \kappa_{num} \\
& | & \texttt{[}\kappa_{num}, \kappa_{num}\texttt{]} \\
\\
e & ::= & \kappa_{num} \\
& | & \kappa_e \\
& | & \mathit{lval} \\
& | & \mathit{lval}\ \texttt{=}\ e \\
& | & f\texttt{(}\overline{e}\texttt{)} \\
& | & \mathit{unop}\ e \\
& | & e\ \mathit{aop}\ e \\
& | & e\ \texttt{/}\ e \\
& | & e\ \texttt{\%}\ e \\
& | & e\ \mathit{bop}\ e \\
& | & e\ \mathit{relop}\ e \\
& | & e\ \texttt{?}\ e\ \texttt{:}\ e \\
& | & \texttt{(}\overline{e}\texttt{)} \\
& | & \texttt{[}e, e\texttt{]} \\
& | & e\texttt{[0]} \\
& | & e\texttt{[1]} \\
\\
\mathit{unop} & ::= & \texttt{\~{}} \mid \texttt{-} \mid \texttt{!} \\
\\
\mathit{aop} & ::= & \texttt{+} \mid \texttt{-} \mid \texttt{*} \\
\mathit{bop} & ::= & \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>} \\
\\
\mathit{relop} & ::= & \texttt{<} \mid \texttt{<=} \mid \texttt{>} \mid \texttt{>=} \mid \texttt{!=} \mid \texttt{==} \\
\\
\mathit{lval} & ::= & x \\
& | & x\texttt{[}e\texttt{]}
\end{array}
$$

2 Type rules

$$
\begin{array}{rcl}
\sigma, \tau & ::= & \texttt{bits1} \mid \texttt{bits32} \mid \texttt{boolish} \\
& | & \texttt{int32} \mid \texttt{uint32} \\
& | & \texttt{int64} \mid \texttt{uint64} \\
& | & \texttt{float64} \\
& | & \texttt{array}_\tau \mid \texttt{function} \mid \texttt{jsval} \\
& | & \texttt{floor} \\
& | & (\overline{\sigma}) \to \tau
\end{array}
$$

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

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

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

$$
\begin{array}{rcl}
\mathit{type}(X\ \texttt{|}\ \texttt{0}) & = & \texttt{int32} \\
\mathit{type}(X\ \texttt{>>>}\ \texttt{0}) & = & \texttt{uint32} \\
\mathit{type}(\texttt{+}X) & = & \texttt{float64} \\
\mathit{type}(\texttt{[}X\texttt{[0]}\ \texttt{>>>}\ \texttt{0}, X\texttt{[1]}\ \texttt{|}\ \texttt{0]}) & = & \texttt{int64} \\
\mathit{type}(\texttt{[}X\texttt{[0]}\ \texttt{>>>}\ \texttt{0}, X\texttt{[1]}\ \texttt{>>>}\ \texttt{0]}) & = & \texttt{uint64}
\end{array}
$$

$$
\begin{array}{rcl}
\texttt{int32}, \texttt{uint32} & <: & \texttt{bits32} \\
\texttt{bits1} & <: & \texttt{boolish} \\
\texttt{bits32} & <: & \texttt{boolish} \\
\texttt{bits32} & <: & \texttt{float64} \\
\texttt{float64} & <: & \texttt{jsval} \\
\texttt{function} & <: & \texttt{jsval} \\
\texttt{array}_\tau & <: & \texttt{jsval} \\
\texttt{floor} & <: & (\texttt{float64}) \to \texttt{float64} \\
(\sigma) \to \tau & <: & \texttt{function}
\end{array}
$$

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

$$
\begin{array}{rcl}
M(\texttt{floor}) & = & \texttt{floor} \\
M(\texttt{ceil}) & = & (\texttt{float64}) \to \texttt{float64} \\
M(\texttt{sin}) & = & (\texttt{float64}) \to \texttt{float64} \\
M(\texttt{cos}) & = & (\texttt{float64}) \to \texttt{float64} \\
 & \cdots &
\end{array}
$$

$$
\begin{array}{rcl}
A(\texttt{Uint8Array}) & = & \texttt{uint32} \\
A(\texttt{Uint16Array}) & = & \texttt{uint32} \\
A(\texttt{Uint32Array}) & = & \texttt{uint32} \\
A(\texttt{Int8Array}) & = & \texttt{int32} \\
A(\texttt{Int16Array}) & = & \texttt{int32} \\
A(\texttt{Int32Array}) & = & \texttt{int32} \\
A(\texttt{Float32Array}) & = & \texttt{float64} \\
A(\texttt{Float64Array}) & = & \texttt{float64}
\end{array}
$$

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

[T-Program]

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

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

[T-ImportStd]

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

[T-ImportFFI]

$$\frac{y \notin dom(M)}{b; e; \Gamma \vdash \texttt{var}\ x\ \texttt{=}\ e\texttt{.}y\texttt{;}\ \mathbf{ok}}$$

[T-View]

$$\frac{\Gamma(x) = \texttt{array}_{A(y)}}{b; e; \Gamma \vdash \texttt{var}\ x\ \texttt{=}\ e\texttt{.}y(b)\texttt{;}\ \mathbf{ok}}$$

[T-NewView]

$$\frac{\Gamma(x) = \texttt{array}_{A(y)}}{b; e; \Gamma \vdash \texttt{var}\ x\ \texttt{=}\ \texttt{new}\ e\texttt{.}y(b)\texttt{;}\ \mathbf{ok}}$$

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

[T-Function]

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

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

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

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

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

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

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

$$\text{[T-Return]}\quad \frac{\Gamma \vdash e : \tau}{\Gamma; L \vdash \texttt{return}\ e\texttt{;} : \tau/\mathsf{return}}$$

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

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

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

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

$$\frac{\varepsilon = \{\epsilon\}}{\Gamma; L \vdash \texttt{break;} : \tau/\varepsilon}\;[\textsc{T-Break}] \qquad \frac{\varepsilon = \{lab\}}{\Gamma; L \vdash \texttt{break}\ \ lab\texttt{;} : \tau/\varepsilon}\;[\textsc{T-BreakLabel}]$$

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

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

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

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

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

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

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

[T-Number]
$$\dfrac{}{\Gamma \vdash \kappa_{num} : \mathit{type}(\kappa_{num})}$$

[T-Cast]
$$\dfrac{\Gamma \vdash e : \tau}{\Gamma \vdash \kappa_e : \mathit{type}(\kappa_e)}$$

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

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

[T-Load]
$$\dfrac{\Gamma(x) = \texttt{array}_\tau \qquad \Gamma \vdash e : \texttt{uint32}}{\Gamma \vdash x\texttt{[}e\texttt{]} : \tau}$$

[T-Store]
$$\dfrac{\Gamma \vdash e_1 : \texttt{uint32} \qquad \Gamma \vdash e_2 : \tau \qquad \tau <: \Gamma(x)}{\Gamma \vdash x\texttt{[}e_1\texttt{]} \texttt{ = } e_2 : \tau}$$

[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]
$$\dfrac{\Gamma(f) = \texttt{function} \qquad \forall i. \Gamma \vdash e_i : \sigma_i \wedge \sigma_i <: \texttt{jsval}}{\Gamma \vdash f\texttt{(}\overline{e}\texttt{)} : \texttt{jsval}}$$

[T-Conditional]
$$\dfrac{\Gamma \vdash e_1 : \sigma \qquad \sigma <: \texttt{boolish} \qquad \forall i \in \{2,3\}. \Gamma \vdash e_i : \tau}{\Gamma \vdash e_1 \texttt{ ? } e_2 \texttt{ : } e_3 : \tau}$$

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

[T-IArith]
$$\dfrac{\forall i \in \{1,2\}. \Gamma \vdash e_i : \tau \qquad \tau <: \texttt{bits32}}{\Gamma \vdash (e_1 \ \mathit{aop} \ e_2) \texttt{ | 0} : \texttt{int32}}$$

[T-UArith]
$$\dfrac{\forall i \in \{1,2\}. \Gamma \vdash e_i : \tau \qquad \tau <: \texttt{bits32}}{\Gamma \vdash (e_1 \ \mathit{aop} \ e_2) \texttt{ >>> 0} : \texttt{uint32}}$$

[T-FArith]
$$\dfrac{\forall i \in \{1,2\}. \Gamma \vdash e_i : \tau_i \qquad \forall i. \tau_i <: \texttt{float64}}{\Gamma \vdash e_1 \ \mathit{aop} \ e_2 : \texttt{float64}}$$

Expression checking (cont'd) $\boxed{\Gamma \vdash e : \tau}$

[T-IDiv]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \texttt{int32}}{\Gamma \vdash (e_1 \texttt{ / } e_2) \texttt{ | 0} : \texttt{int32}}$$

[T-UDiv]
$$\frac{\Gamma(f) = \texttt{floor} \quad \forall i \in \{1,2\}.\Gamma \vdash e_i : \texttt{uint32}}{\Gamma \vdash f(e_1 \texttt{ / } e_2) : \texttt{uint32}}$$

[T-FDiv]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau_i \quad \forall i.\tau_i <: \texttt{float64}}{\Gamma \vdash e_1 \texttt{ / } e_2 : \texttt{float64}}$$

[T-IMod]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \texttt{int32}}{\Gamma \vdash e_1 \texttt{ \% } e_2 : \texttt{int32}}$$

[T-UMod]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \texttt{uint32}}{\Gamma \vdash e_1 \texttt{ \% } e_2 : \texttt{uint32}}$$

[T-FMod]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau_i \quad \forall i.\tau_i <: \texttt{float64}}{\Gamma \vdash e_1 \texttt{ \% } e_2 : \texttt{float64}}$$

[T-Rel]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau \quad \tau <: \texttt{float64}}{\Gamma \vdash e_1 \ \textit{relop}\ e_2 : \texttt{bits1}}$$

[T-Bitwise]
$$\frac{\forall i \in \{1,2\}.\Gamma \vdash e_i : \tau_i \quad \forall i.\tau_i <: \texttt{bits32}}{\Gamma \vdash e_1 \ \textit{bop}\ e_2 : \texttt{int32}}$$

[T-BitwiseNot]
$$\frac{\Gamma \vdash e : \tau \quad \tau <: \texttt{bits32}}{\Gamma \vdash \texttt{\~{}}e : \texttt{int32}}$$

[T-Not]
$$\frac{\Gamma \vdash e : \tau \quad \tau <: \texttt{boolish}}{\Gamma \vdash \texttt{!}e : \texttt{bits1}}$$

[T-Negate]
$$\frac{\Gamma \vdash e : \tau \quad \tau <: \texttt{bits32}}{\Gamma \vdash \texttt{-}e : \texttt{int32}}$$