

asm.js

Dave Herman, Luke Wagner, and Alon Zakai

October 26, 2012

1 Abstract syntax

$$\begin{array}{ll} b, e, f, g, x, y, z & \in \text{Identifier} \\ \text{arguments} & \notin \text{Identifier} \\ \text{eval} & \notin \text{Identifier} \end{array}$$
$$\begin{array}{ll} P & ::= \text{function}(b, e) \{ \overline{imp_x} \overline{fn_f} \exp \} \\ imp_x & ::= \text{var } x = e.y; \\ & \quad | \text{var } x = e.y(b); \\ & \quad | \text{var } x = \text{new } e.y(b); \\ exp & ::= \text{return } f; \\ & \quad | \text{return } \{ \overline{x:f} \}; \\ fn_f & ::= \text{function } f(\overline{x}) \{ \overline{x = \kappa_x}; \text{var } \overline{y = v}; ss \} \end{array}$$

```

s ::= { ss }
    | e;
    | if (e) s
    | if (e) s else s
    | return v;
    | while (e) s
    | do s while (e);
    | for (e; e; e) s
    | switch (e) {  $\bar{c}$  }
    | switch (e) {  $\bar{c}$  d }
    | break;
    | break lab;
    | continue;
    | continue lab;
    | lab: s

```

```

ss ::=  $\bar{s}$ 

```

```

c ::= case e: ss
d ::= default: ss
cd ::= c | d

```

```

 $\kappa_X$  ::= X | 0
    | X >>> 0
    | +X
    | [X[0] >>> 0, X[1] | 0]
    | [X[0] >>> 0, X[1] >>> 0]

```

$$\begin{aligned}
v &::= \kappa_{num} \\
&| [\kappa_{num}, \kappa_{num}] \\
\\
e &::= \kappa_{num} \\
&| \kappa_e \\
&| lval \\
&| lval = e \\
&| f(\bar{e}) \\
&| unop\ e \\
&| e\ aop\ e \\
&| e / e \\
&| e \% e \\
&| e\ bop\ e \\
&| e\ relop\ e \\
&| e\ ?\ e\ :\ e \\
&| (\bar{e}) \\
&| [e, e] \\
&| e[0] \\
&| e[1] \\
\\
unop &::= \sim\ |\ -\ |\ ! \\
\\
aop &::= +\ |\ -\ |\ * \\
bop &::= |\ \&\ |\ ^\ |\ <<\ |\ >>\ |\ >>> \\
\\
relop &::= <\ |\ <=\ |\ >\ |\ >=\ |\ !=\ |\ == \\
\\
lval &::= x \\
&| x[e] \\
&| x[0] \\
&| x[1]
\end{aligned}$$

2 Type rules

$$\begin{aligned}
\sigma, \tau &::= \text{bits1} \mid \text{bits32} \mid \text{bits64} \mid \text{boolish} \\
&| \text{int32} \mid \text{uint32} \\
&| \text{int64} \mid \text{uint64} \\
&| \text{float64} \\
&| \text{array}_\tau \mid \text{function} \mid \text{jsval} \\
&| \text{floor} \\
&| (\bar{\sigma}) \rightarrow \tau \\
&| \text{void}
\end{aligned}$$

$$\begin{aligned}\ell &::= lab \mid \epsilon \\ L &::= \{\bar{\ell}\} \\ \varepsilon &::= L \mid \text{return}\end{aligned}$$

$$\begin{aligned}L ; L' &= L \cup L' \\ \emptyset ; \text{return} &= \text{return} \\ \{\ell, \bar{\ell}'\} ; \text{return} &= \{\ell, \bar{\ell}'\} \\ \text{return} ; L &= \text{return}\end{aligned}$$

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

$$\begin{aligned}\text{type}(X \mid 0) &= \text{int32} \\ \text{type}(X \ggg 0) &= \text{uint32} \\ \text{type}(+X) &= \text{float64} \\ \text{type}([X[0] \ggg 0, X[1] \mid 0]) &= \text{int64} \\ \text{type}([X[0] \ggg 0, X[1] \ggg 0]) &= \text{uint64}\end{aligned}$$

$$\begin{aligned}\text{int32}, \text{uint32} &<: \text{bits32} \\ \text{int64}, \text{uint64} &<: \text{bits64} \\ \text{bits1} &<: \text{boolish} \\ \text{bits32} &<: \text{boolish} \\ \text{bits32} &<: \text{float64} \\ \text{float64} &<: \text{jsval} \\ \text{function} &<: \text{jsval} \\ \text{array}_\tau &<: \text{jsval} \\ \text{void} &<: \text{jsval} \\ \text{floor} &<: (\text{float64}) \rightarrow \text{float64} \\ (\sigma) \rightarrow \tau &<: \text{function}\end{aligned}$$

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

$$\begin{aligned}M(\text{floor}) &= \text{floor} \\ M(\text{ceil}) &= (\text{float64}) \rightarrow \text{float64} \\ M(\text{sin}) &= (\text{float64}) \rightarrow \text{float64} \\ M(\text{cos}) &= (\text{float64}) \rightarrow \text{float64} \\ &\dots\end{aligned}$$

$A(\text{Uint8Array}) = \text{uint32}$
 $A(\text{Uint16Array}) = \text{uint32}$
 $A(\text{Uint32Array}) = \text{uint32}$
 $A(\text{Int8Array}) = \text{int32}$
 $A(\text{Int16Array}) = \text{int32}$
 $A(\text{Int32Array}) = \text{int32}$
 $A(\text{Float32Array}) = \text{float64}$
 $A(\text{Float64Array}) = \text{float64}$

Program checking

$\vdash P \text{ ok}$

$$\begin{array}{c}
\text{[T-PROGRAM]} \\
\frac{\{\bar{x}\} \cap \{\bar{f}\} = \emptyset \quad \{\bar{x}\} \cap \{b, e\} = \emptyset \quad \{\bar{f}\} \cap \{b, e\} = \emptyset \\
\forall i. b; e; \Gamma_0 \vdash \text{imp}_x \text{ ok} \quad \forall i. \Gamma_0, \Gamma_1 \vdash \text{fn}_f \text{ ok} \quad \forall i. \Gamma_0, \Gamma_1 \vdash r \text{ ok}}{\vdash \text{function}(b, e) \{ \overline{\text{imp}_x} \overline{\text{fn}_f} \exp \} \text{ ok}}
\end{array}$$

Import checking

$b; e; \Gamma \vdash \text{imp ok}$

$$\begin{array}{c}
\text{[T-IMPORTSTD]} \quad \text{[T-IMPORTFFI]} \\
\frac{\Gamma(x) = M(y)}{b; e; \Gamma \vdash \text{var } x = e.y; \text{ ok}} \quad \frac{y \notin \text{dom}(M)}{b; e; \Gamma \vdash \text{var } x = e.y; \text{ ok}} \\
\text{[T-VIEW]} \quad \text{[T-NEWVIEW]} \\
\frac{\Gamma(x) = \text{array}_{A(y)}}{b; e; \Gamma \vdash \text{var } x = e.y(b); \text{ ok}} \quad \frac{\Gamma(x) = \text{array}_{A(y)}}{b; e; \Gamma \vdash \text{var } x = \text{new } e.y(b); \text{ ok}}
\end{array}$$

Function checking

$\Gamma \vdash \text{fn ok}$

$$\begin{array}{c}
\text{[T-FUNCTION]} \\
\frac{\{\bar{x}\} \cap \{\bar{y}\} = \emptyset \quad \Gamma(f) = (\bar{\sigma}) \rightarrow \tau \quad \bar{\sigma} = \overline{\text{type}(\kappa_x)} \quad \tau \neq \text{void} \\
\Gamma, \{\bar{x} : \bar{\sigma}, \bar{y} : \text{type}(v)\}; \emptyset \vdash ss : \tau / \text{return}}{\Gamma \vdash \text{function } f(\bar{x}) \{ \bar{x} = \kappa_x; \text{var } \bar{y} = v; ss \} \text{ ok}} \\
\text{[T-VOIDFUNCTION]} \\
\frac{\{\bar{x}\} \cap \{\bar{y}\} = \emptyset \quad \Gamma(f) = (\bar{\sigma}) \rightarrow \text{void} \quad \bar{\sigma} = \overline{\text{type}(\kappa_x)} \\
\Gamma, \{\bar{x} : \bar{\sigma}, \bar{y} : \text{type}(v)\}; \emptyset \vdash ss : \text{void} / \varepsilon}{\Gamma \vdash \text{function } f(\bar{x}) \{ \bar{x} = \kappa_x; \text{var } \bar{y} = v; ss \} \text{ ok}}
\end{array}$$

Export checking

$\Gamma \vdash \text{exp ok}$

$$\begin{array}{c}
\text{[T-SINGLETON]} \quad \text{[T-MODULE]} \\
\frac{\Gamma(f) = (\bar{\sigma}) \rightarrow \tau \quad \tau <: \text{jval}}{\Gamma \vdash \text{return } f; \text{ ok}} \quad \frac{\forall f. \Gamma(f) = (\bar{\sigma}) \rightarrow \tau \wedge \tau <: \text{jval}}{\Gamma \vdash \text{return } \{ \bar{x} : \bar{f} \}; \text{ ok}}
\end{array}$$

Statement list checking

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

$$\frac{[T\text{-NoStatements}] \quad \Gamma; L \vdash \epsilon : \tau/\emptyset}{\Gamma; L \vdash \epsilon : \tau/\emptyset} \quad \frac{[T\text{-Statements}] \quad \begin{array}{c} \forall i. \Gamma; L \vdash s_i : \tau/\varepsilon_i \\ n > 0 \quad \varepsilon = \varepsilon_1 ; \dots ; \varepsilon_n \end{array}}{\Gamma; L \vdash \bar{s} : \tau/\varepsilon}$$

Statement checking

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

$$\begin{array}{c} [T\text{-Block}] \quad \frac{\Gamma; \emptyset \vdash ss : \tau/\varepsilon}{\Gamma; L \vdash \{ ss \} : \tau/\varepsilon} \quad [T\text{-ExprStmt}] \quad \frac{\Gamma \vdash e : \sigma}{\Gamma; L \vdash e ; : \tau/\emptyset} \\ \\ [T\text{-If}] \quad \frac{\begin{array}{c} \Gamma \vdash e : \sigma <: \text{boolish} \\ \Gamma; \emptyset \vdash s : \tau/\varepsilon \\ \varepsilon' = \varepsilon \cup \emptyset \end{array}}{\Gamma; L \vdash \text{if } (e) \ s : \tau/\varepsilon'} \quad [T\text{-IfElse}] \quad \frac{\begin{array}{c} \Gamma \vdash e : \sigma <: \text{boolish} \\ \Gamma; \emptyset \vdash s_1 : \tau/\varepsilon_1 \quad \Gamma; \emptyset \vdash s_2 : \tau/\varepsilon_2 \\ \varepsilon = \varepsilon_1 \cup \varepsilon_2 \end{array}}{\Gamma; L \vdash \text{if } (e) \ s_1 \ \text{else } s_2 : \tau/\varepsilon} \\ \\ [T\text{-ReturnExpr}] \quad \frac{\Gamma \vdash e : \tau}{\Gamma; L \vdash \text{return } e ; : \tau/\text{return}} \quad [T\text{-ReturnVoid}] \quad \frac{}{\Gamma; L \vdash \text{return}; : \text{void}/\text{return}} \\ \\ [T\text{-While}] \quad \frac{\begin{array}{c} \Gamma \vdash e : \sigma <: \text{boolish} \\ \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Gamma; L \vdash \text{while } (e) \ s : \tau/\varepsilon'} \quad [T\text{-DoWhile}] \quad \frac{\begin{array}{c} \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \Gamma \vdash e : \sigma <: \text{boolish} \\ \varepsilon' = \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Gamma; L \vdash \text{do } s \ \text{while } (e) ; : \tau/\varepsilon'} \\ \\ [T\text{-For}] \quad \frac{\begin{array}{c} \forall i. \Gamma \vdash e_i : \sigma_i \\ \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Gamma; L \vdash \text{for } (e_1; e_2; e_3) \ s : \tau/\varepsilon'} \end{array}$$

Statement checking (cont'd)

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

$$\begin{array}{c} \text{[T-BREAK]} \\ \hline \Gamma; L \vdash \mathbf{break}; : \tau/\{\epsilon\} \end{array} \quad \begin{array}{c} \text{[T-BREAKLABEL]} \\ \hline \Gamma; L \vdash \mathbf{break} \text{ } lab; : \tau/\{lab\} \end{array}$$

$$\begin{array}{c} \text{[T-CONTINUE]} \\ \hline \Gamma; L \vdash \mathbf{continue}; : \tau/\emptyset \end{array} \quad \begin{array}{c} \text{[T-CONTINUELABEL]} \\ \hline \Gamma; L \vdash \mathbf{continue} \text{ } lab; : \tau/\emptyset \end{array}$$

$$\begin{array}{c} \text{[T-LABEL]} \\ \Gamma; L \cup \{lab\} \vdash s : \tau/\varepsilon \\ \varepsilon' = \varepsilon - (L \cup \{lab\}) \\ \hline \Gamma; L \vdash lab : s : \tau/\varepsilon' \end{array}$$

[T-SWITCH]

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

[T-SWITCHRETURN]

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

Case checking

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

$$\begin{array}{c} \text{[T-CASE]} \\ \Gamma \vdash e : \sigma \\ \Gamma; L \vdash ss : \tau/\varepsilon \\ \hline \Gamma; L \vdash \mathbf{case} \text{ } e : ss : \sigma, \tau/\varepsilon \end{array} \quad \begin{array}{c} \text{[T-DEFAULT]} \\ \Gamma; L \vdash ss : \tau/\varepsilon \\ \hline \Gamma; L \vdash \mathbf{default} : ss : \sigma, \tau/\varepsilon \end{array}$$

Expression checking

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

$$\begin{array}{c}
\text{[T-Cast]} \\
\frac{\kappa_e = (e_1 / e_2) \mid 0 \Rightarrow \exists i. \Gamma \not\vdash e_i : \mathbf{int32} \quad e = e_1 \text{ aop } e_2 \Rightarrow \exists i. \Gamma \not\vdash e_i : \tau <: \mathbf{bits32}}{\Gamma \vdash e : \tau} \\
\text{[T-Number]} \\
\frac{}{\Gamma \vdash \kappa_{num} : type(\kappa_{num})} \quad \frac{}{\Gamma \vdash \kappa_e : type(\kappa_e)}
\\
\text{[T-VarRef]} \quad \text{[T-Assign]} \\
\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \quad \frac{\Gamma(x) = \tau \quad \Gamma \vdash e : \tau}{\Gamma \vdash x = e : \tau}
\\
\text{[T-Load]} \quad \text{[T-Store]} \\
\frac{\Gamma(x) = \mathbf{array}_\tau \quad \Gamma \vdash e : \mathbf{uint32}}{\Gamma \vdash x[e] : \tau} \quad \frac{\Gamma \vdash e_1 : \mathbf{uint32} \quad \Gamma \vdash e_2 : \Gamma(x)}{\Gamma \vdash x[e_1] = e_2 : \tau}
\\
\text{[T-LowUint]} \quad \text{[T-LowInt]} \quad \text{[T-HighUint]} \\
\frac{\Gamma \vdash e : \mathbf{uint64}}{\Gamma \vdash e[0] : \mathbf{uint32}} \quad \frac{\Gamma \vdash e : \mathbf{int64}}{\Gamma \vdash e[0] : \mathbf{int32}} \quad \frac{\Gamma \vdash e : \tau <: \mathbf{bits64}}{\Gamma \vdash e[1] : \mathbf{uint32}}
\\
\text{[T-FunCall]} \quad \text{[T-FFI]} \\
\frac{\Gamma(f) = (\bar{\sigma}) \rightarrow \tau \quad \forall i. \Gamma \vdash e_i : \sigma_i}{\Gamma \vdash f(\bar{e}) : \tau} \quad \frac{\Gamma(f) = \mathbf{function} \quad \forall i. \Gamma \vdash e_i : \sigma_i <: \mathbf{jsval}}{\Gamma \vdash f(\bar{e}) : \mathbf{jsval}}
\\
\text{[T-Conditional]} \quad \text{[T-Paren]} \\
\frac{\Gamma \vdash e_1 : \sigma <: \mathbf{boolish} \quad \forall i. \Gamma \vdash e_i : \tau}{\Gamma \vdash e_1 ? e_2 : e_3 : \tau} \quad \frac{\forall i \leq n. \Gamma \vdash e_i : \tau_i}{\Gamma \vdash (\bar{e}) : \tau_n}
\\
\text{[T-IArith]} \quad \text{[T-UArith]} \\
\frac{\forall i. \Gamma \vdash e_i : \tau <: \mathbf{bits32}}{\Gamma \vdash (e_1 \text{ aop } e_2) \mid 0 : \mathbf{int32}} \quad \frac{\forall i. \Gamma \vdash e_i : \tau <: \mathbf{bits32}}{\Gamma \vdash (e_1 \text{ aop } e_2) >>> 0 : \mathbf{uint32}}
\\
\text{[T-FArith]} \\
\frac{\forall i. \Gamma \vdash e_i : \tau_i <: \mathbf{float64}}{\Gamma \vdash e_1 \text{ aop } e_2 : \mathbf{float64}}
\end{array}$$

Expression checking (cont'd)

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

$$\begin{array}{c}
\begin{array}{c} \text{[T-IDiv]} \\ \frac{\forall i. \Gamma \vdash e_i : \text{int32}}{\Gamma \vdash (e_1 / e_2) \mid 0 : \text{int32}} \end{array}
\quad
\begin{array}{c} \text{[T-UDiv]} \\ \frac{\Gamma(f) = \text{floor} \quad \forall i. \Gamma \vdash e_i : \text{uint32}}{\Gamma \vdash f(e_1 / e_2) : \text{uint32}} \end{array}
\quad
\begin{array}{c} \text{[T-FDiv]} \\ \frac{\forall i. \Gamma \vdash e_i : \tau_i <: \text{float64}}{\Gamma \vdash e_1 / e_2 : \text{float64}} \end{array} \\
\\
\begin{array}{c} \text{[T-IMod]} \\ \frac{\forall i. \Gamma \vdash e_i : \text{int32}}{\Gamma \vdash e_1 \% e_2 : \text{int32}} \end{array}
\quad
\begin{array}{c} \text{[T-UMod]} \\ \frac{\forall i. \Gamma \vdash e_i : \text{uint32}}{\Gamma \vdash e_1 \% e_2 : \text{uint32}} \end{array}
\quad
\begin{array}{c} \text{[T-FMod]} \\ \frac{\forall i. \Gamma \vdash e_i : \tau_i <: \text{float64}}{\Gamma \vdash e_1 \% e_2 : \text{float64}} \end{array} \\
\\
\begin{array}{c} \text{[T-REL]} \\ \frac{\forall i. \Gamma \vdash e_i : \tau <: \text{float64}}{\Gamma \vdash e_1 \text{ relop } e_2 : \text{bits1}} \end{array}
\quad
\begin{array}{c} \text{[T-BITWISE]} \\ \frac{\forall i. \Gamma \vdash e_i : \tau_i <: \text{bits32}}{\Gamma \vdash e_1 \text{ bop } e_2 : \text{int32}} \end{array} \\
\\
\begin{array}{c} \text{[T-BITWISENOT]} \\ \frac{\Gamma \vdash e : \tau <: \text{bits32}}{\Gamma \vdash \sim e : \text{int32}} \end{array}
\quad
\begin{array}{c} \text{[T-NOT]} \\ \frac{\Gamma \vdash e : \tau <: \text{boolish}}{\Gamma \vdash !e : \text{bits1}} \end{array}
\quad
\begin{array}{c} \text{[T-NEGATE]} \\ \frac{\Gamma \vdash e : \tau <: \text{bits32}}{\Gamma \vdash -e : \text{int32}} \end{array}
\end{array}$$