

# asm.js

Dave Herman, Luke Wagner, and Alon Zakai

November 10, 2012

## 1 Abstract syntax

$b, e, f, g, x, y, z \in \text{Identifier}$   
 $\text{arguments}, \text{eval} \notin \text{Identifier}$

$P ::= \text{function } [g]([e[, b]]) \{ \text{"use asm"}; \overline{\text{imp}_x} \overline{\text{fn}_f} \overline{\text{var } y = v}; \text{exp} \}$   
 $\text{imp}_x ::= \text{var } x = e.y;$   
           $\quad \text{var } x = \text{new } e.y(b);$   
 $\text{exp} ::= \text{return } f;$   
           $\quad \text{return } \{ \overline{x:f} \};$   
 $\text{fn}_f ::= \text{function } f(\overline{x}) \{ \overline{x = \kappa_x}; \text{var } \overline{y = v}; ss \}$

$s ::= \{ ss \}$   
           $e;$   
           $;$   
           $\text{if } (e) s$   
           $\text{if } (e) s \text{ else } s$   
           $\text{return } e;$   
           $\text{while } (e) s$   
           $\text{do } s \text{ while } (e);$   
           $\text{for } ([e]; [e]; [e]) s$   
           $\text{switch } (e) \{ \overline{c} \}$   
           $\text{switch } (e) \{ \overline{c} d \}$   
           $\text{break};$   
           $\text{break } lab;$   
           $\text{continue};$   
           $\text{continue } lab;$   
           $lab: s$

$ss ::= \overline{s}$

$c ::= \text{case } e: ss$   
 $d ::= \text{default}: ss$   
 $cd ::= c \mid d$

$$\kappa_x ::= \sim \sim x \mid +x \mid x \mid 0 \mid x >> 0$$

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

$$e ::= \begin{array}{l} v \\ lval \\ lval = e \\ f(\bar{e}) \\ unop\ e \\ e\ binop\ e \\ e\ ?\ e : e \\ (\bar{e}) \end{array}$$

$$unop ::= + \mid \sim \mid !$$

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

## 2 Type rules

$$\begin{array}{l} \sigma, \tau ::= \text{bit} \mid \text{double} \mid \text{int} \mid \text{signed} \mid \text{unsigned} \mid \text{boolish} \mid \text{intish} \mid \text{void} \mid \text{unknown} \\ \rho ::= \tau \mid \text{array}_\tau^n \mid \text{imul} \mid \text{function} \mid (\bar{\sigma}) \rightarrow \tau \\ \omega ::= ((\bar{\sigma}) \rightarrow \tau) \wedge \dots \wedge ((\bar{\sigma}') \rightarrow \tau') \end{array}$$

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

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

$$\begin{array}{l} L \cup \text{return} = L \\ \text{return} \cup L = L \\ \text{return} \cup \text{return} = \text{return} \end{array}$$

```

type( $\sim\sim X$ ) = int
type( $+X$ ) = double
type( $n$ ) = int
type( $r$ ) = double
type( $X|0$ ) = signed
type( $X>>>0$ ) = unsigned

```

```

constant <: signed, unsigned
signed, unsigned <: int, extern
bit, int <: boolish
double <: extern
unknown, int <: intish

```

```

M(imul) : imul
M(ceil), M(sin), M(cos) : (double) → double

```

```

A(UInt8Array), A(Int8Array) = arrayint8
A(UInt16Array), A(Int16Array) = arrayint16
A(UInt32Array), A(Int32Array) = arrayint32
A(Float32Array) = arraydouble32
A(Float64Array) = arraydouble64

```

```

+, - : (double, double) → double
      ^ (int, int) → intish
* : (double, double) → double
/, % : (double, double) → double
      ^ (signed, signed) → intish
      ^ (unsigned, unsigned) → intish

```

```

|, &, ^, <<, >> : (intish, intish) → signed
>>> : (intish, intish) → unsigned

```

```

<, <=, >, >=, ==, != : (signed, signed) → bit
                      ^ (unsigned, unsigned) → bit
                      ^ (double, double) → bit

```

```

+ : (intish) → double
~ : (intish) → signed
! : (boolish) → bit

```

```

 $\Delta ::= \{x:\rho\}$ 
 $\Gamma ::= \{x:\tau\}$ 

```

Program checking

$\boxed{\vdash P \text{ ok}}$

$$\frac{\begin{array}{c} \text{[T-PROGRAM]} \\ \{\bar{x}\} \cap \{\bar{f}\} = \emptyset \quad \{\bar{x}\} \cap \{[g], [e], [b]\} = \emptyset \quad \{\bar{f}\} \cap \{[g], [e], [b]\} = \emptyset \\ \forall y. \Delta(y) = \text{type}(v) \\ \forall i. [e]; [b]; \Delta \vdash \text{imp}_x \text{ ok} \quad \forall i. \Delta \vdash \text{fn}_f \text{ ok} \quad \forall i. \Delta \vdash \text{exp} \text{ ok} \end{array}}{\vdash \text{function } [g]([e, b]) \{ \text{"use asm"; } \overline{\text{imp}_x \text{ fn}_f} \text{ var } y = v; \text{exp} \} \text{ ok}}$$

Import checking

$\boxed{[e]; [b]; \Delta \vdash \text{imp} \text{ ok}}$

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

$$\frac{\begin{array}{c} \text{[T-NEWVIEW]} \\ \Delta(x) = \text{array}_{A(y)}^n \end{array}}{e; b; \Delta \vdash \text{var } x = \text{new } e.y(b); \text{ok}}$$

Function checking

$\boxed{\Delta \vdash \text{fn} \text{ ok}}$

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

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

Export checking

$\boxed{\Delta \vdash \text{exp} \text{ ok}}$

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

Statement list checking

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

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

Statement checking

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

$$\begin{array}{c} \frac{[\text{T-Block}] \quad \Delta; \Gamma; \emptyset \vdash ss : \tau/\varepsilon}{\Delta; \Gamma; L \vdash \{ ss \} : \tau/\varepsilon} \quad \frac{[\text{T-ExprStmt}] \quad \Delta; \Gamma \vdash e : \sigma}{\Delta; \Gamma; L \vdash e ; : \tau/\emptyset} \\ \\ \frac{[\text{T-If}] \quad \begin{array}{l} \Delta; \Gamma \vdash e : \text{boolish} \\ \Delta; \Gamma; \emptyset \vdash s : \tau/\varepsilon \\ \varepsilon' = \varepsilon \cup \emptyset \end{array}}{\Delta; \Gamma; L \vdash \text{if } (e) \ s : \tau/\varepsilon'} \quad \frac{[\text{T-IfElse}] \quad \begin{array}{l} \Delta; \Gamma \vdash e : \text{boolish} \\ \Delta; \Gamma; \emptyset \vdash s_1 : \tau/\varepsilon_1 \quad \Delta; \Gamma; \emptyset \vdash s_2 : \tau/\varepsilon_2 \\ \varepsilon = \varepsilon_1 \cup \varepsilon_2 \end{array}}{\Delta; \Gamma; L \vdash \text{if } (e) \ s_1 \ \text{else} \ s_2 : \tau/\varepsilon} \\ \\ \frac{[\text{T-ReturnExpr}] \quad \text{type}(e) <: \tau \quad \Delta; \Gamma \vdash e : \tau}{\Delta; \Gamma; L \vdash \text{return } e ; : \tau/\text{return}} \quad \frac{[\text{T-ReturnVoid}]}{\Delta; \Gamma; L \vdash \text{return}; : \text{void}/\text{return}} \\ \\ \frac{[\text{T-While}] \quad \begin{array}{l} \Delta; \Gamma \vdash e : \text{boolish} \\ \Delta; \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Delta; \Gamma; L \vdash \text{while } (e) \ s : \tau/\varepsilon'} \quad \frac{[\text{T-DoWhile}] \quad \begin{array}{l} \Delta; \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \\ \Delta; \Gamma \vdash e : \text{boolish} \\ \varepsilon' = \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Delta; \Gamma; L \vdash \text{do } s \ \text{while } (e); : \tau/\varepsilon'} \\ \\ \frac{[\text{T-For}] \quad \begin{array}{l} [\Delta; \Gamma \vdash e_1 : \sigma_1] \quad [\Delta; \Gamma \vdash e_2 : \text{boolish}] \quad [\Delta; \Gamma \vdash e_3 : \sigma_3] \\ \Delta; \Gamma; L \cup \{\epsilon\} \vdash s : \tau/\varepsilon \quad \varepsilon' = \emptyset \cup \varepsilon - (L \cup \{\epsilon\}) \end{array}}{\Delta; \Gamma; L \vdash \text{for } ([e_1]; [e_2]; [e_3]) \ s : \tau/\varepsilon'}$$

Statement checking (cont'd)

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

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

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

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

[T-SWITCH]

$$\begin{array}{c} \Delta; \Gamma \vdash e : \sigma \\ \forall i. \Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau/\varepsilon_i \\ \Gamma; L \cup \{\epsilon\} \vdash cd : \sigma, \tau/\varepsilon \\ \varepsilon \neq \mathbf{return} \vee \exists i. \varepsilon_i \cup \emptyset \neq \emptyset \\ \varepsilon' = (\varepsilon \cup \bigcup_i \varepsilon_i) - (L \cup \{\epsilon\}) \\ \hline \Delta; \Gamma; L \vdash \mathbf{switch} \text{ } (e) \{ \bar{c} \text{ } cd \} : \tau/\varepsilon' \end{array} \quad \begin{array}{c} \text{[T-SWITCHRETURN]} \\ \Delta; \Gamma \vdash e : \sigma \\ \forall i. \Gamma; L \cup \{\epsilon\} \vdash c_i : \sigma, \tau/\varepsilon_i \\ \forall i. \varepsilon_i \cup \emptyset = \emptyset \\ \Gamma; L \cup \{\epsilon\} \vdash cd : \sigma, \tau/\mathbf{return} \\ \hline \Delta; \Gamma; L \vdash \mathbf{switch} \text{ } (e) \{ \bar{c} \text{ } cd \} : \tau/\mathbf{return} \end{array}$$

[T-EMPTYSWITCH]

$$\begin{array}{c} \Delta; \Gamma \vdash e : \sigma \\ \hline \Delta; \Gamma; L \vdash \mathbf{switch} \text{ } (e) \{ \quad \} : \tau/\emptyset \end{array}$$

[T-EMPTYSTATEMENT]

$$\begin{array}{c} \hline \Delta; \Gamma; L \vdash ; : \tau/\emptyset \end{array}$$

Case checking

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

[T-CASE]

$$\begin{array}{c} \Delta; \Gamma \vdash e : \sigma \\ \Delta; \Gamma; L \vdash ss : \tau/\varepsilon \\ \hline \Gamma; L \vdash \mathbf{case} \text{ } e : ss : \sigma, \tau/\varepsilon \end{array}$$

[T-DEFAULT]

$$\begin{array}{c} \Delta; \Gamma; L \vdash ss : \tau/\varepsilon \\ \hline \Gamma; L \vdash \mathbf{default} : ss : \sigma, \tau/\varepsilon \end{array}$$

$$(\Delta \cdot \Gamma)(x) = \begin{cases} \Gamma(x) & \text{if } x \in \text{dom}(\Gamma) \\ \Delta(x) & \text{otherwise} \end{cases}$$

Expression checking

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

$$\begin{array}{c}
\begin{array}{c}
\text{[T-CONSTANT]} \\
\frac{-2^{31} \leq n < 2^{32}}{\Delta; \Gamma \vdash n : \text{constant}}
\end{array}
\qquad
\begin{array}{c}
\text{[T-DOUBLE]} \\
\frac{}{\Delta; \Gamma \vdash r : \text{double}}
\end{array}
\\[10pt]
\begin{array}{c}
\text{[T-VARREF]} \\
\frac{(\Delta \cdot \Gamma)(x) = \tau}{\Delta; \Gamma \vdash x : \tau}
\end{array}
\qquad
\begin{array}{c}
\text{[T-ASSIGN]} \\
\frac{\Delta; \Gamma \vdash e : \tau \quad \tau <: (\Delta \cdot \Gamma)(x)}{\Delta; \Gamma \vdash x = e : \tau}
\end{array}
\\[10pt]
\begin{array}{c}
\text{[T-LOAD]} \\
\frac{m = 2^k - 1 \quad (\Delta \cdot \Gamma)(x) = \text{array}_\tau^n \quad \Delta; \Gamma \vdash e : \text{intish}}{\Delta; \Gamma \vdash x[(e \ \& \ m) \gg n/8] : \tau}
\end{array}
\qquad
\begin{array}{c}
\text{[T-STORE]} \\
\frac{m = 2^k - 1 \quad (\Delta \cdot \Gamma)(x) = \text{array}_\tau^n \quad \Delta; \Gamma \vdash e_1 : \text{intish} \quad \Delta; \Gamma \vdash e_2 : \tau}{\Delta; \Gamma \vdash x[(e_1 \ \& \ m) \gg n/8] = e_2 : \tau}
\end{array}
\\[10pt]
\begin{array}{c}
\text{[T-IMUL]} \\
\frac{(\Delta \cdot \Gamma)(f) = \text{imul} \quad \forall i. \Delta; \Gamma \vdash e_i : \text{intish}}{\Delta; \Gamma \vdash f(e_1, e_2) : \text{signed}}
\end{array}
\qquad
\begin{array}{c}
\text{[T-FUNCALL]} \\
\frac{(\Delta \cdot \Gamma)(f) = (\bar{\sigma}) \rightarrow \tau \quad \forall i. \Delta; \Gamma \vdash e_i : \sigma_i}{\Delta; \Gamma \vdash f(\bar{e}) : \tau}
\end{array}
\qquad
\begin{array}{c}
\text{[T-FFI]} \\
\frac{(\Delta \cdot \Gamma)(f) = \text{function} \quad \forall i. \Delta; \Gamma \vdash e_i : \text{extern}}{\Delta; \Gamma \vdash f(\bar{e}) : \text{unknown}}
\end{array}
\\[10pt]
\begin{array}{c}
\text{[T-CONDITIONAL]} \\
\frac{\Delta; \Gamma \vdash e_1 : \text{boolish} \quad \Delta; \Gamma \vdash e_2 : \tau \quad \Delta; \Gamma \vdash e_3 : \tau}{\Delta; \Gamma \vdash e_1 ? e_2 : e_3 : \tau}
\end{array}
\qquad
\begin{array}{c}
\text{[T-PAREN]} \\
\frac{\forall i \leq n. \Delta; \Gamma \vdash e_i : \tau_i}{\Delta; \Gamma \vdash (\bar{e}) : \tau_n}
\end{array}
\\[10pt]
\begin{array}{c}
\text{[T-UNOP]} \\
\frac{\text{unop} : \_ \wedge (\sigma) \rightarrow \tau \wedge \_ \quad \Delta; \Gamma \vdash e : \sigma}{\Delta; \Gamma \vdash \text{unop } e : \tau}
\end{array}
\qquad
\begin{array}{c}
\text{[T-BINOP]} \\
\frac{\text{binop} : \_ \wedge (\sigma_1, \sigma_2) \rightarrow \tau \wedge \_ \quad \Delta; \Gamma \vdash e_1 : \sigma_1 \quad \Delta; \Gamma \vdash e_2 : \sigma_2}{\Delta; \Gamma \vdash e_1 \text{ binop } e_2 : \tau}
\end{array}
\\[10pt]
\begin{array}{c}
\text{[T-SUB]} \\
\frac{\Delta; \Gamma \vdash e : \sigma \quad \sigma <: \tau}{\Delta; \Gamma \vdash e : \tau}
\end{array}
\qquad
\begin{array}{c}
\text{[T-CAST]} \\
\frac{\Delta; \Gamma \vdash e : \text{double}}{\Delta; \Gamma \vdash \sim e : \text{signed}}
\end{array}
\end{array}$$