asm.js

Dave Herman, Luke Wagner, and Alon Zakai

November 10, 2012

1 Abstract syntax

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

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

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

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

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

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

$$\begin{array}{rcl} v & ::= & r \mid n \end{array}$$

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

$$\begin{array}{rcl} \mathit{unop} & ::= & \texttt{+} \mid \texttt{\~{}} \mid \texttt{!} \end{array}$$

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

2 Type rules

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

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

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

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

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

$$
\begin{array}{rcl}
\texttt{constant} & <: & \texttt{signed}, \texttt{unsigned} \\
\texttt{signed}, \texttt{unsigned} & <: & \texttt{int}, \texttt{extern} \\
\texttt{bit}, \texttt{int} & <: & \texttt{boolish} \\
\texttt{double} & <: & \texttt{extern} \\
\texttt{unknown}, \texttt{int} & <: & \texttt{intish}
\end{array}
$$

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

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

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

$$
\begin{array}{rcl}
\Delta & ::= & \{\overline{x : \rho}\} \\
\Gamma & ::= & \{\overline{x : \tau}\}
\end{array}
$$

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

[T-Program]

$$\dfrac{\begin{array}{c}\{\overline{x}\} \cap \{\overline{f}\} = \emptyset \qquad \{\overline{x}\} \cap \{[g],[e],[b]\} = \emptyset \qquad \{\overline{f}\} \cap \{[g],[e],[b]\} = \emptyset \\ \forall y.\Delta(y) = \mathit{type}(v) \\ \forall i.[e];[b];\Delta \vdash \mathit{imp}_x\ \mathbf{ok} \qquad \forall i.\Delta \vdash \mathit{fn}_f\ \mathbf{ok} \qquad \forall i.\Delta \vdash \mathit{exp}\ \mathbf{ok}\end{array}}{\vdash \texttt{function}\ [g]\texttt{(}[e[,b]]\texttt{)}\ \texttt{\{}\ \overline{\mathit{imp}_x}\ \overline{\mathit{fn}_f}\ \overline{\texttt{var}\ y\ \texttt{=}\ v\texttt{;}}\ \mathit{exp}\ \texttt{\}}\ \mathbf{ok}}$$

Import checking $\boxed{[e];[b];\Delta \vdash \mathit{imp}\ \mathbf{ok}}$

[T-ImportStd]

$$\dfrac{\Delta(x) = M(y)}{e;[b];\Delta \vdash \texttt{var}\ x\ \texttt{=}\ e\texttt{.}y\texttt{;}\ \mathbf{ok}}$$

[T-ImportFFI]

$$\dfrac{y \notin \mathit{dom}(M), \mathit{dom}(A) \qquad \Delta(x) = \texttt{function}}{e;[b];\Delta \vdash \texttt{var}\ x\ \texttt{=}\ e\texttt{.}y\texttt{;}\ \mathbf{ok}}$$

[T-NewView]

$$\dfrac{\Delta(x) = \texttt{array}^n_{A(y)}}{e;b;\Delta \vdash \texttt{var}\ x\ \texttt{=}\ \texttt{new}\ e\texttt{.}y(b)\texttt{;}\ \mathbf{ok}}$$

Function checking $\boxed{\Delta \vdash \mathit{fn}\ \mathbf{ok}}$

[T-Function]

$$\dfrac{\begin{array}{c}\{\overline{x}\} \cap \{\overline{y}\} = \emptyset \qquad \Delta(f) = (\overline{\sigma}) \to \tau \qquad \overline{\sigma} = \overline{\mathit{type}(\kappa_x)} \qquad \tau \neq \texttt{void} \\ \Delta; \{\overline{x : \sigma}, \overline{y : \mathit{type}(v)}\}; \emptyset \vdash \mathit{ss} : \tau/\mathsf{return}\end{array}}{\Delta \vdash \texttt{function}\ f\texttt{(}\overline{x}\texttt{)}\ \texttt{\{}\ \overline{x\ \texttt{=}\ \kappa_x\texttt{;}}\ \texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}\ \mathit{ss}\ \texttt{\}}\ \mathbf{ok}}$$

[T-VoidFunction]

$$\dfrac{\begin{array}{c}\{\overline{x}\} \cap \{\overline{y}\} = \emptyset \qquad \Delta(f) = (\overline{\sigma}) \to \texttt{void} \qquad \overline{\sigma} = \overline{\mathit{type}(\kappa_x)} \\ \Delta; \{\overline{x : \sigma}, \overline{y : \mathit{type}(v)}\}; \emptyset \vdash \mathit{ss} : \texttt{void}/\varepsilon\end{array}}{\Delta \vdash \texttt{function}\ f\texttt{(}\overline{x}\texttt{)}\ \texttt{\{}\ \overline{x\ \texttt{=}\ \kappa_x\texttt{;}}\ \texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}\ \mathit{ss}\ \texttt{\}}\ \mathbf{ok}}$$

Export checking $\boxed{\Delta \vdash \mathit{exp}\ \mathbf{ok}}$

[T-Singleton]

$$\dfrac{\Delta(f) = (\overline{\sigma}) \to \tau \qquad \tau <: \texttt{extern}}{\Delta \vdash \texttt{return}\ f\texttt{;}\ \mathbf{ok}}$$

[T-Module]

$$\dfrac{\forall f.\Delta(f) = (\overline{\sigma}) \to \tau \wedge \tau <: \texttt{extern}}{\Delta \vdash \texttt{return}\ \texttt{\{}\ \overline{x\texttt{:}f}\ \texttt{\};}\ \mathbf{ok}}$$

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

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

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

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

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

$$\text{[T-ReturnExpr]}\ \frac{\mathit{type}(e) <: \tau \quad \Delta; \Gamma \vdash e : \tau}{\Delta; \Gamma; L \vdash \texttt{return}\ e\texttt{;} : \tau/\mathsf{return}} \qquad \text{[T-ReturnVoid]}\ \frac{}{\Delta; \Gamma; L \vdash \texttt{return;} : \mathsf{void}/\mathsf{return}}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\text{[T-Constant]}\ \frac{-2^{31} \leq n < 2^{32}}{\Delta; \Gamma \vdash n : \texttt{constant}} \qquad \text{[T-Double]}\ \frac{}{\Delta; \Gamma \vdash r : \texttt{double}}$$

$$\text{[T-VarRef]}\ \frac{(\Delta \cdot \Gamma)(x) = \tau}{\Delta; \Gamma \vdash x : \tau} \qquad \text{[T-Assign]}\ \frac{\Delta; \Gamma \vdash e : \tau \qquad \tau <: (\Delta \cdot \Gamma)(x)}{\Delta; \Gamma \vdash x \texttt{ = } e : \tau}$$

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

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

$$\text{[T-IMul]}\ \frac{(\Delta \cdot \Gamma)(f) = \texttt{imul} \qquad \forall i. \Delta; \Gamma \vdash e_i : \texttt{intish}}{\Delta; \Gamma \vdash f\texttt{(}e_1, e_2\texttt{)} : \texttt{signed}} \qquad \text{[T-FunCall]}\ \frac{(\Delta \cdot \Gamma)(f) = (\overline{\sigma}) \to \tau \qquad \forall i. \Delta; \Gamma \vdash e_i : \sigma_i}{\Delta; \Gamma \vdash f\texttt{(}\overline{e}\texttt{)} : \tau}$$

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

$$\text{[T-Conditional]}\ \frac{\Delta; \Gamma \vdash e_1 : \texttt{boolish} \qquad \Delta; \Gamma \vdash e_2 : \tau \qquad \Delta; \Gamma \vdash e_3 : \tau}{\Delta; \Gamma \vdash e_1 \texttt{ ? } e_2 \texttt{ : } e_3 : \tau} \qquad \text{[T-Paren]}\ \frac{\forall i \leq n. \Delta; \Gamma \vdash e_i : \tau_i}{\Delta; \Gamma \vdash \texttt{(}\overline{e}\texttt{)} : \tau_n}$$

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

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

$$\text{[T-Sub]}\ \frac{\Delta; \Gamma \vdash e : \sigma \qquad \sigma <: \tau}{\Delta; \Gamma \vdash e : \tau} \qquad \text{[T-Cast]}\ \frac{\Delta; \Gamma \vdash e : \texttt{double}}{\Delta; \Gamma \vdash \texttt{\textasciitilde\textasciitilde} e : \texttt{signed}}$$