Functions

2019

New Expressions and Statements

- Extended expressions

Call $f(e_1, \ldots, e_k)$

$$\mathscr{E} \mathrel{+}= \mathscr{X}\,\mathscr{E}^*$$

- Extended statements

$$\mathscr{S} \mathrel{+}= \textbf{\texttt{return}}\ \mathscr{E}^?$$

- Extended configuration

Optional result

$$\mathscr{C} = \Sigma \times \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{Z}^?$$

State $\mathscr{X} \to \mathbb{Z}$

Input

Output

Big-Step Semantics for Expressions

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{n} \langle \sigma, i, o, n \rangle \qquad [\mathsf{Const}^{\mathscr{E}}_{bs}]$$

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{x} \langle \sigma, i, o, \sigma\, x \rangle \qquad [\mathsf{Var}^{\mathscr{E}}_{bs}]$$

$$\frac{\Phi \vdash c \xRightarrow{A} c' = \langle \_, \_, \_, a \rangle \qquad \Phi \vdash c' \xRightarrow{B} \langle \sigma'', i'', o'', b \rangle}{\Phi \vdash c \xRightarrow{A \otimes B} \langle \sigma'', i'', o'', a \oplus b \rangle} \qquad [\mathsf{Binop}^{\mathscr{E}}_{bs}]$$

$$\frac{\begin{array}{c}\text{for } j \in [1..k] \;.\; \Phi \vdash c_{j-1} \xRightarrow{e_j} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\, f = \mathbf{fun}\; f\; (\overline{a})\; \mathbf{local}\; \overline{l}\; \{s\} \\ \mathtt{skip}, \Phi \vdash \left\langle \mathbf{enter}\, \sigma_k\; (\overline{a} @ \overline{l})\; [\overline{a_j \leftarrow v_j}], i_k, o_k, - \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle \end{array}}{\Phi \vdash c_0 = \langle \sigma_0, \_, \_, \_ \rangle \xRightarrow{f(\overline{e_k})} \langle \mathbf{leave}\, \sigma'\; \sigma_0, i', o', n \rangle}$$

$[\mathsf{Call}^{\mathscr{E}}_{bs}]$

New Semantics for Statements

- return

Suppose we fill it

$$\frac{[\![e]\!] \Downarrow \quad ???}{c \xrightarrow{\textbf{return}\ e} c'}$$

- **return** e ; S

Always executes S!

$$\frac{c \xrightarrow{\textbf{return}\ e} c'' \qquad c'' \xrightarrow{S} c'}{c \xrightarrow{\textbf{return}\ e\ ;\ S} c'}$$

CPS Semantics for Statements

- New component in the environment

$$K, \Phi \vdash c \xRightarrow{s} c'$$

Lack of locality

- New meta-operator $\diamond$

$$\begin{array}{rcrcl} S & \diamond & \texttt{\textbf{skip}} & = & S \\ S_1 & \diamond & S_2 & = & S_1; S_2 \end{array}$$

CPS Rules — Basic Stmts

$$\texttt{skip}, \Phi \vdash c \xRightarrow{\texttt{skip}} c \qquad [\text{SkipSkip}]$$

$$\frac{\texttt{skip}, \Phi \vdash c \xRightarrow{K} c' \qquad K \neq \texttt{skip}}{K, \Phi \vdash c \xRightarrow{\texttt{skip}} c'} \qquad [\text{Skip}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad \texttt{skip}, \Phi \vdash \langle \sigma[x \leftarrow n], i, o, \text{—} \rangle \xRightarrow{K} c'}{K, \Phi \vdash c \xRightarrow{x \texttt{:=} e} c'} \qquad [\text{Assign}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad \texttt{skip}, \Phi \vdash \langle \sigma, i, o@[n], \text{—} \rangle \xRightarrow{K} c'}{K, \Phi \vdash c \xRightarrow{\texttt{write}\ (e)} c'} \qquad [\text{Write}]$$

$$\frac{\texttt{skip}, \Phi \vdash \langle \sigma[x \leftarrow z], i, o, \text{—} \rangle \xRightarrow{K} c'}{K, \Phi \vdash \langle \sigma, z :: i, o, \text{—} \rangle \xRightarrow{\texttt{read}\ (x)} c'} \qquad [\text{Read}]$$

CPS Rules — Seq and If

$$\frac{s_2 \diamond K, \Phi \vdash c \xRightarrow{s_1} c'}{K, \Phi \vdash c \xRightarrow{s_1;\, s_2} c'} \qquad [\text{Seq}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n \neq 0 \qquad K, \Phi \vdash \langle \sigma, i, o, \text{—} \rangle \xRightarrow{s_1} c'}{K, \Phi \vdash c \xRightarrow{\mathtt{\textbf{if}}\ e\ \mathtt{\textbf{then}}\ s_1\ \mathtt{\textbf{else}}\ s_2} c'} \qquad [\text{IfTrue}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n \neq 0 \qquad K, \Phi \vdash \langle \sigma, i, o, \text{—} \rangle \xRightarrow{s_2} c'}{K, \Phi \vdash c \xRightarrow{\mathtt{\textbf{if}}\ e\ \mathtt{\textbf{then}}\ s_1\ \mathtt{\textbf{else}}\ s_2} c'} \qquad [\text{IfFalse}]$$

CPS Rules — While

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n \neq 0 \qquad (\texttt{while}\ e\ \texttt{do}\ s) \diamond K, \Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{s} c'}{K, \Phi \vdash c \xRightarrow{\texttt{while}\ e\ \texttt{do}\ s} c'} \quad [\text{WhileTrue}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n = 0 \qquad \texttt{skip}, \Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{K} c'}{K, \Phi \vdash c \xRightarrow{\texttt{while}\ e\ \texttt{do}\ s} c'} \quad [\text{WhileFalse}]$$

CPS Rules — Call and Return

$$\frac{\begin{array}{c}\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xrightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\ f = \mathbf{fun}\ f\ \ (\overline{a})\ \mathbf{local}\ \overline{l}\ \ \{s\} \\ \mathbf{skip}???, \Phi \vdash \left\langle \mathbf{enter}\ \sigma_k\ (\overline{a}@\overline{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, \text{—} \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle \\ \mathbf{skip}, \Phi \vdash ???\langle \mathbf{leave}\ \sigma'\ \sigma_0, i', o', n \rangle \xRightarrow{K} c'' \end{array}}{K, \Phi \vdash c_0 = \langle \sigma_0, \_, \_, \_ \rangle \xRightarrow{f(\overline{e_k})} c''}$$

[Call]

$$K, \Phi \vdash c \xRightarrow{\mathbf{return}} c$$

[ReturnEmpty]

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} c'}{K, \Phi \vdash c \xRightarrow{\mathbf{return}\ e} c'}$$

[Return]

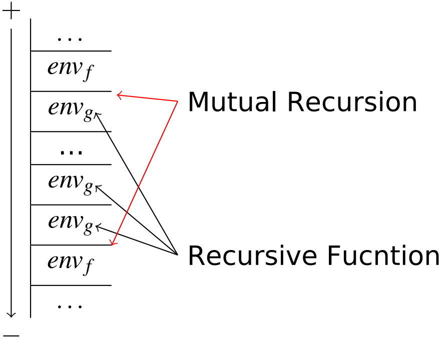
Functions X86-32

X86-32

- Standard caller code:

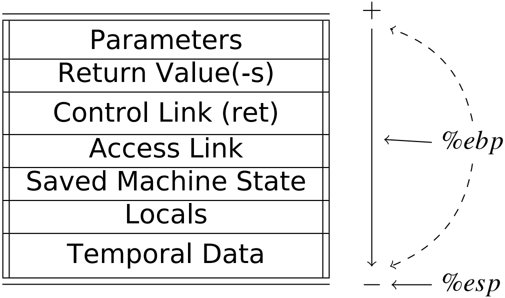
```
push    $arg_n$
push    $arg_{n-1}$
...
push    $arg_1$
call    $< callee\ name >$
addl    $n*4,\ \%esp$
```

- Result → $\%eax$



Frames

Each activation has an ***activation record*** (*frame* or *memory display*) on the call stack



Application Binary Interface

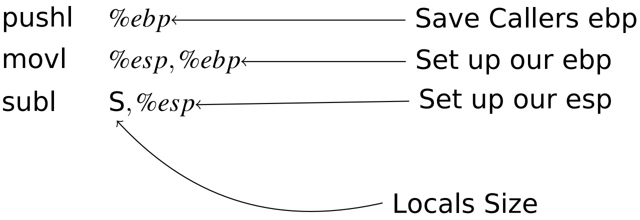
- ABI

- EABI

- Calling convention

Prologue

Standard prologue
X86-32:



```
pushl   %ebp
movl    %esp,%ebp
subl    S,%esp
```

Save Callers ebp

Set up our ebp

Set up our esp

Locals Size

Epilogue

Standard Epilogue X86-32:

```
movl   %ebp, %esp
popl   %ebp
ret
```

Registers

- EAX, EDX, ECX — caller-saved registers
- EBX, EDI, ESI (, and EBP) — callee-saved registers
- EIP, ESP (, and EBP) — special purpose registers