

BoA

let $x = _$

add 1(e)

sub 1(e)

└──────────┘
"adder"

|

$e_1 + e_2$

$e_1 * e_2$

└──────────┘
"boa"

if cond:

e_1

else:

e_2

let $x = 10$ in
 $\text{add1}(x)$

if cond:

e_1

else:

e_2

$\text{eax} \leftarrow \text{eval cond}$

" $\text{eax} \neq 0$ "

if-true:

e_1

if-false:

e_2

$\text{cmp } a_1, a_2$

- je lab

jne lab

jmp lab

```

if 10:
    22
else:
    33

```

```

mov eax, 10
cmp eax, 0
je if-false

mov eax, 22
jmp if-exit

if-false:
    mov eax, 33

if-exit:

```

```

if cond: i
    e1
else:
    e2

```

```

compile env cond
+ [cmp eax 0
  , je if-false-i]
    ↓
+ compile env e1
+ [jmp if-exit-i]
+ [Label if-false]
+ compile env e2
+ [Label if-exit-i]

```

if $\left[\text{let } \underline{a} = 10 \text{ in} \right]:$

$\text{sub1}(\underline{a})$

else:

1072

$$\underline{33} - \boxed{10}$$

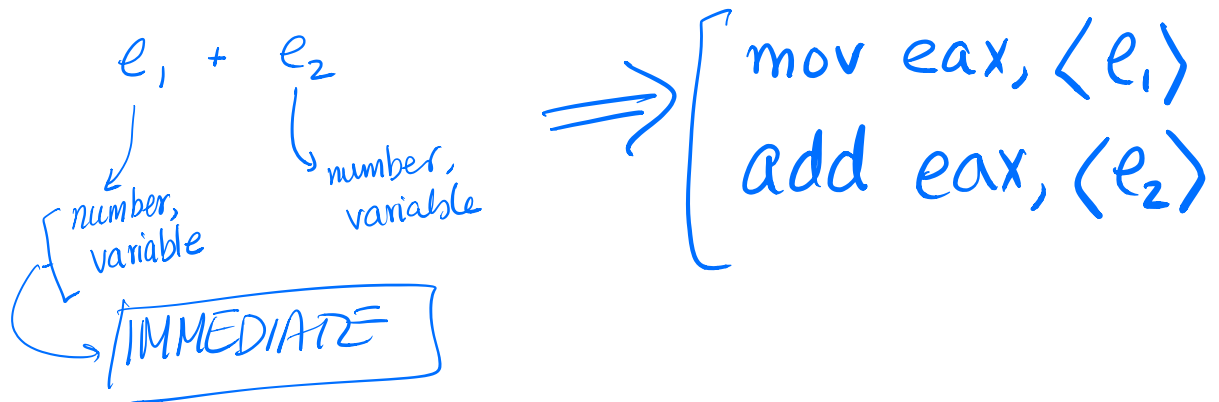
```
mov eax, 33
sub eax, 10
```

$$\begin{array}{c} \textcircled{n_1} + n_2 \\ \hline * \\ \sqrt{} + n_2 \end{array} \longrightarrow \begin{array}{l} [\text{mov eax, } n_1 \\ \text{add eax, } n_2 \\ \text{sub} \\ \text{mul}] \end{array}$$

let $x^{\textcircled{1}} = 12$ in

$$\begin{array}{c} x' + 10 \\ \hline \hline 10 + x' \end{array}$$

```
mov eax, 12
mov [esp-4*1], eax
[ mov eax, [esp-4*1]
  add eax, 10 ]
[ mov eax, 10
  add eax, [esp-4] ]
```



$(10 + 20) * 30$
 (underlined)

$\begin{cases} \text{mov eax, 10} \\ \text{add eax, 20} \\ \text{mul eax, 30} \end{cases}$

$(1+2) * (3+4) * (6+7)$
 (boxed) (underlined) (underlined)
 eax

0. use another reg. $\begin{cases} \text{mov eax, 1} \\ \text{add eax, 2} \end{cases}$
 1. MOV eax onto stack $\begin{cases} \text{mov ebx, 3} \\ \text{add ebx, 4} \\ \text{mul eax, ebx} \end{cases}$

$$i_1 + i_2 + i_3 + i_4 + \dots$$

```
mov  eax, <i1>
add  eax, <i2>
add  eax, <i3>
```

```
⋮
```

$$(1+2) * (3+4)$$

DONT KNOW


$$\begin{aligned} \text{let } t_1 &= 1+2 \\ t_2 &= 3+4 \\ \text{in } t_1 * t_2 \end{aligned}$$

ANF

DO KNOW

ADMINISTRATIVE NORMAL FORM

SOURCE \rightarrow AST $\xrightarrow{(2.)}$ ANF $\xrightarrow{(1.)}$ ASM



compileImm env (Number n)

= [immov eax n]

compileImm env (Var x)

= [immov eax, Reg ESP i]

(i = lookup x env

compile env (Prim2 Plus i₁ i₂ -)

= [IMOV (Reg EAX) (imm Arg env i₁)
 , ADD (Reg EAX) (imm Arg env i₂)]

$$\left((1+2) * (3+4) \right) * \left((5+6) * (7+8) \right)$$