BOA

let x = —

add1(e)

sub1(e)

"adder"

$e_1 + e_2$

$e_1 * e_2$

"boa"

```
if cond:
    e1
else:
    e2
```

let $x = 10$ in

add1(x)

if cond:

$\boxed{e1}$

else :

e2

eax ← eval cond

"eax ≠ 0"

if_true: $\boxed{e_1}$

if_false: $\boxed{e_2}$

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 [ let a = 10 in
     add1(a) ]:
    sub1(a)
else:
    1072
```

$33 - \boxed{10}$

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

$n_1 \; + \; n_2 \longrightarrow$

```
mov eax, n1
add eax, n2
```

$-$ sub

$*$ mul

$v + n_2$

let $x^{1} = 12$ in

$x' + 10$

$10 + x'$

```
mov eax, 12
mov [esp-4x1], eax
```

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

```
mov eax, 10
add eax, [esp-4]
```

$e_1 + e_2$

e_1: number, variable

e_2: number, variable

IMMEDIATE

```
mov eax, <e1>
add eax, <e2>
```

$(10 + 20) * 30$

```
mov eax, 10
add eax, 20
mul eax, 30
```

$(1+2) * (3+4) * (6+7)$

eax

```
mov eax, 1
add eax, 2
```

```
mov ebx, 3
add ebx, 4
mul eax, ebx
```

0. use another reg.
1. MOV eax onto stack

$i_1 + i_2 + i_3 + i_4 + \ldots$

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

$\vdots$

$$\boxed{\underbrace{(1+2)} * \underbrace{(3+4)}}$$

DONT KNOW

$$\boxed{\begin{array}{l} \text{let } t_1 = 1+2 \\ \quad\;\; t_2 = 3+4 \\ \text{in} \\ \quad\;\; t_1 * t_2 \end{array}}$$

DO KNOW

ANF

ADMINISTRATIVE NORMAL FORM

SOURCE → AST $\xrightarrow{2.}$ ANF $\xrightarrow{1.}$ ASM

compileImm env (Number n)

= [Imov eax n]

compileImm env (Var x)

= [Imov eax, RegOffset ESP i]

(i = lookup x env

compile env (Prim2 Plus i_1 i_2 _)

= [IMOV (Reg EAX) (immArg env i_1)

, IADD (Reg EAX) (immArg env i_2)]

$$\underbrace{((1+2) * \underbrace{(3+4)})} * ((5+6) * (7+8))$$