- Environment
- Binary operators
- Conditionals

} Auklet

} Bluebird

Auklet

- Integer constants
- Unary ops (after, before)
- Binary ops (+, -, *)
- Let
- Vars

"environment": during compilation, stores information about where we are in program, and what variables are in scope

$$\left(\{ a \mapsto [rsp-8],\ b \mapsto [rsp-16] \},\ -24 \right)$$

where I intend to store data

next free stack location

"machine word" is the size of data a processor is designed to handle

the instructions returned, when executed, produce the same value as the expression would if it were run and stores that value in RAX

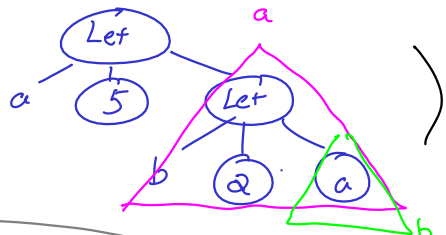
```
let rec compile_expression (env : environment) (e : expr) : instruction list =
```

compile (env, 5) = mov rax, 5

compile (env, after / 5) = compile (env, 5) @ add rax, 1 = mov rax, 5 / add rax, 1

```
let a = 5 in
let b = 2 in
a
```

let $x = e_1$ in e_2



0. make to compile hatch
1. run hatch to compile Auklet
2. run Auklet prog

compile (({}, -8), ...) =

compile (({}, -8), 5) @ mov [rsp-8], rax @

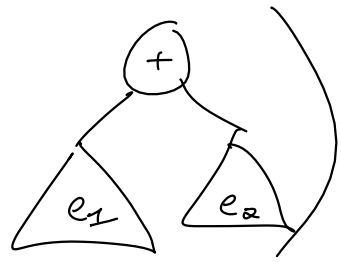
Compile (({ a ↦ [rsp-8] }, -16), let b = 2 in a)

let a = ({}, -8)

let b = 2 in b ({ b ↦ [rsp-8] }, -16)

in

let c = a in c ({ a ↦ [rsp-8] }, -16)

compile({}, -8), 

=

compile(({}, -8), 

)

②

{ mov [rsp-8], rax }

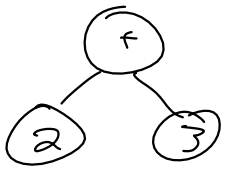
@

compile(({}, -16), 

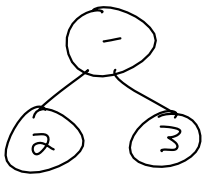
)

@

{ add rax, [rsp-8] }



[rsp-8] = 2
rax = 3
add rax, [rsp-8] ⇒ 5



[rsp-8] = 2
rax = 3
sub rax, [rsp-8] ⇒ 1

Conditionals

C-style: ifnz <expr> then <expr> else <expr>

ifnz 3 then 2 else 4 ⟹ 2
ifnz 0 then 5 else 8+1 ⟹ 9

Some assembly required:

label is an identifier followed by ":"

```
mov rax, 17
fish:
  add rax, 1
  jmp fish
```

```
jmp label
cmp arg, arg
je/jz label
jl   label
js   label
```