Summary

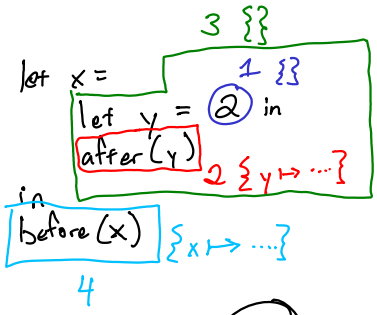
Auklet — numbers, unary operators, binary operators, let

Prepare for Bluebird

Environment

Environment is a dictionary mapping variables to addresses/locations

0. make
1. hatch — env lives
2. ./a bird (.run) — env talks about



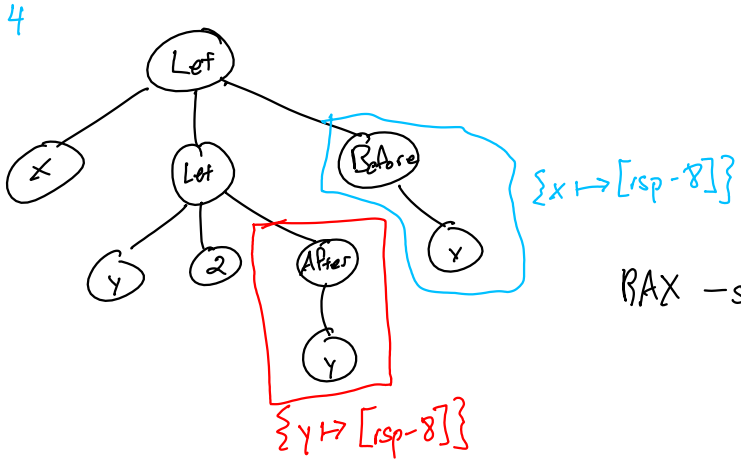
Execution:

1. set y to 2
2. after(y) ⇒ 3
3. set x to 3
4. before(x) ⇒ 2

8 bytes for y; 8 bytes for x

C scope example

```
int x=0;
if (x==0) {
    int y=1;
}
cout<<y<<endl;
```



RAX — stores most recent computation

Binary Operators

↓ add x

(-16, {x ↦ [rsp-8]})

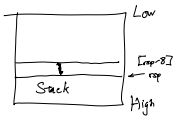
↓ add w

(-24, {w ↦ [rsp-16], x ↦ [rsp-8]})

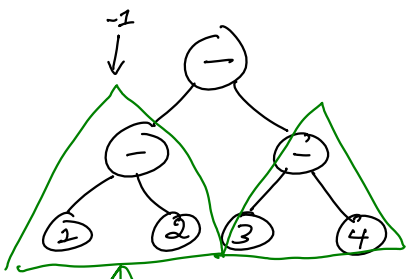
↓ add temporary

(-32, {w ↦ [rsp-16], x ↦ [rsp-8]})

$(1-2)-(3-4)$



Compile:




```
mov rax,1
mov [rsp-8], rax
mov rax, 2
mov [rsp-16], rax
mov rax, [rsp-8]
sub rax, [rsp-16]

mov rax, [rsp-8]
```

Execution:

1. Put 1 in RAX
2. Store RAX on stack
3. Put 2 in RAX
4. Store RAX on stack
5. Load 1 from stack
6. Subtract 2 from RAX

⋮

```
    mov rax, 1
label:
    jmp label        <-- jmp = unconditional jump
```

```
cmp rax, 0      <-- cmp = compare
je label        <-- jump if equal
jg label        <-- jump if greater
jl label        <-- jump if less
```

jne, jge, jle

Exercise:

ifnz x then 4 else 2

Generate assembly for ↑

$\{x \mapsto [rsp-8]\}$

Summary:

1. comparison
2. je to if
3. else code
4. jmp to end
5. if code
6. end:

```
    mov rax, [rsp-8]
    cmp rax, 0
    jne if
    mov rax, 2
    jmp end
if:
    mov rax, 4
end:
```

```
    mov rax, [rsp-8]
    cmp rax, 0
    je else
    mov rax, 4
    jmp end
else:
    mov, rax 2
end:
```