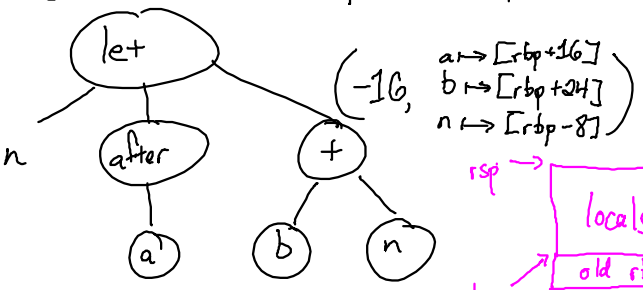
```
let n = after(a) in
b + n
```

offset of next free stack position

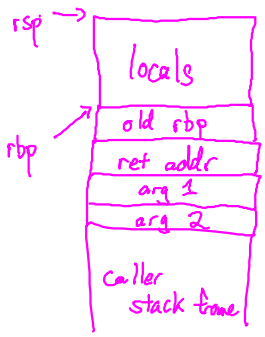
binding & storage location

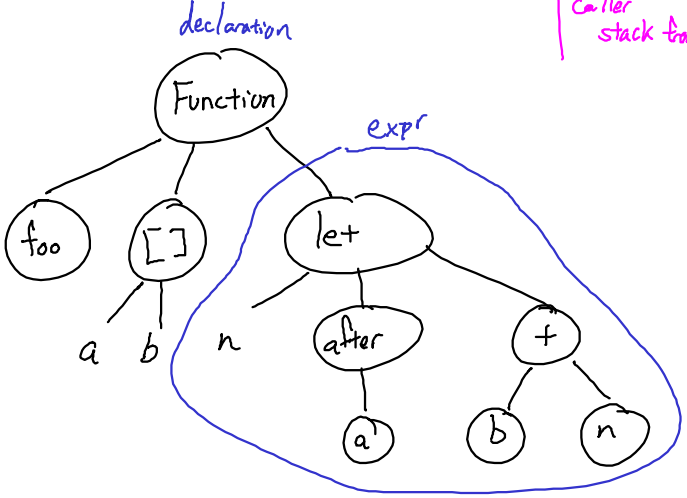
(# , ↦)

(-8, { a ↦ [rbp+16] , b ↦ [rbp+24] })



(-16, a ↦ [rbp+16], b ↦ [rbp+24], n ↦ [rbp-8])






```
fn_foo:
(2)  push rbp
     mov  rbp, rsp
     sub  rsp, 24
(3)  mov  rax, [rbp+16]   ; a
     add  rax, 2          ; after
     mov  [rbp-8], rax    ; let n
     mov  rax, [rbp+24]   ; b
     mov  [rbp-16], rax   ; (temp)
     mov  rax, [rbp-8]    ; n
     mov  [rbp-24], rax   ; (temp)
     mov  rax, [rbp-16]   ; (retrieve temp)
     add  rax, [rbp-24]
(4)  mov  rsp, rbp
     pop  rbp
     ret
```

```
def foo(a,b)
    let n = after a in
    b + n
end
```

<declaration> ::= def <identifier> (<param_list>) <expr> end

def foo(n) n end foo(2) ⟹ 2

def f(k) k+1 end f() ⟹ invalid # of args

def f(k) k+1 end f(f(2)) ⟹ 4

def f(k) k+1 end f(f) ⟹ unbound variable f

g() ⟹ undefined function g

def f() 4 end def f() 5 end f() ⟹ duplicate function name f

def f(a,a,a) a end f(2,3,4) ⟹ duplicate param decl a