Auklet – arithmetic, let

Bluebird – booleans, conditionals

Cardinal – checking for types, call C functions

Dove – declarations for functions

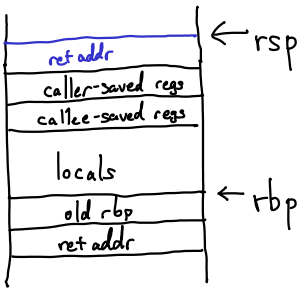
Assume we want to make x64 assembly to call a C function declared as

```
int foo(int x, int y) asm "foo";
```

Presume that you have values of x and y in registers r10 and r11

chronological

1. Caller sets up arguments & calls ← save caller-saved registers* — volatile
2. Callee sets up its stack ← save callee-saved registers* — non-volatile / stable
3. Callee runs
4. Callee tears down its stack ← restore callee-saved registers*
5. Caller tears down arguments ← restore caller-saved registers*



only if I need r10 and r11 values

```
push r10
push r11
mov rdi, r10
mov rsi, r11
call foo   ← label
pop r11
pop r10
```

1

5

push reg ≡ sub rsp, 8
mov [rsp], reg

call lbl ≡ push addr of next instruction
jmp lbl

ret ≡ pop rip

Dove

<program> ::= <declaration_list> <expr>

<declaration_list> ::= ∈
| <declaration> <declaration_list>

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

<param_list> ::= ∈
| <param> , <param_list>
| <param>

<param> ::= <ident>

<expr> ::= (anything from Cardinal)
| <ident> (<expr_list>)

```
def dbl(x)
  x*2
end

dbl(5)
```

⇓

10

```
def triple(n)
  n*3
end

triple(triple(2))
```

⟹ 18

triple(triple(2)) ⟶
triple(2*3) ⟶
triple(6) ⟶
6*3 ⟶ 18

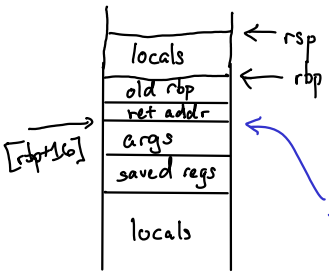
Bird Calling Conventions:

Same as x64 POSIX C conventions except all arguments go onto the stack

```
def inc(n)
  n+1
end
```

```
fn_inc:
  push rbp
  mov rbp, rsp
  sub rsp, 16
  mov rax, [rbp+16]
  mov [rbp-8], rax
  ~~~
  mov esp, rbp
  pop rbp
  ret
```

(2) (3) (4)



8 bytes (1 word) of argument