Status

Working on Dove — functions, function calls, etc.
Eagle — heap, tuples
Falcon — first-order functions

```
  push rax
≅
  sub rsp, 8
  mov [rsp], rax
```

push rax push rbx	sub rsp, 16 mov [rsp], rbx mov [rsp+8], rax

```
   call foo
≅*
   push rip
   jmp foo
```

Falcon

Goal: make fns into values

Functions can't fit in registers, so we'll keep ptrs to closures

```
def add x y =
    x + y
end

let inc = add 1 in
...
```

```
def inc y =
    1 + y
end
```

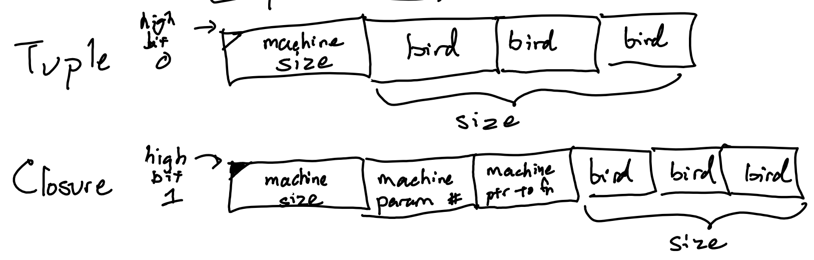
Closure growth (application)

Inductive argument

1. inductive step
2. base case

Pointers : $\overset{\text{Bird}}{0b\cdots\cdots 01}$ $\overset{\text{Machine}}{-1}$

Heap Layouts



Dove:
ECall(name, args)

assumption:
all args are present

Falcon:
EAppl(fn_expr, arg_expr)

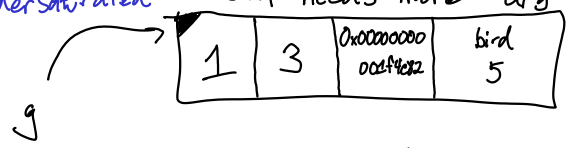
f 1 2

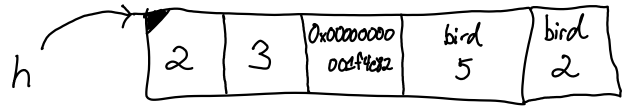
EAppl(EAppl(EVar "f", EInt 1), EInt 2)

Saturated — has enough args to run (#args = #params)

Undersaturated — still needs more args (#args < #params)

```
def f a b c =
    a * b + c
end
let g = f 5 in
let h = g 2 in
h 3
```





When I apply a fn to an arg:

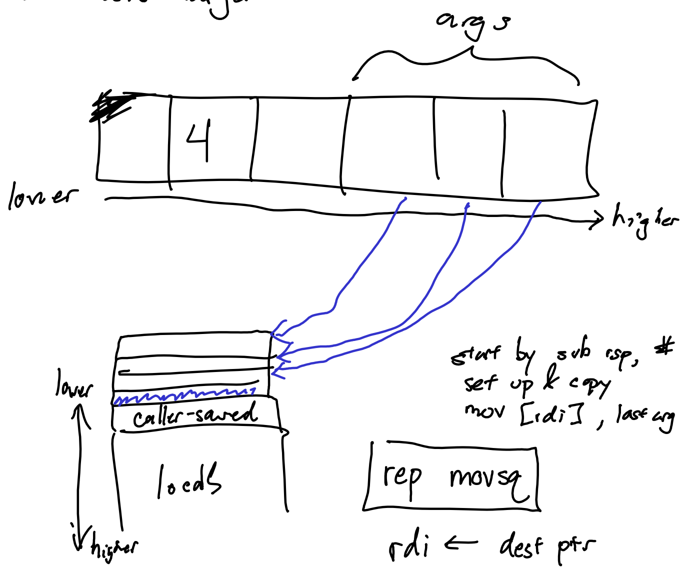
runs in assembly

if the closure + new arg is undersaturated:
- allocate space on heap for closure of size 1 word larger
- copy old closure into that space
- add new arg to end
- update arg count

if the closure + new arg is saturated:

GO TIME

- push caller-saved regs
- put last arg
- copy other args from closure onto stack
- call
- caller teardown



start by sub rsp, #
set up & copy
mov [rdi], last arg

$\boxed{\text{rep movsq}}$

rdi ← dest ptr
rsi ← src ptr
rcx ← count

copy rcx 8-byte words from [rsi] to [rdi]

Dove: "call fun_foo"

Falcon: "call rax"