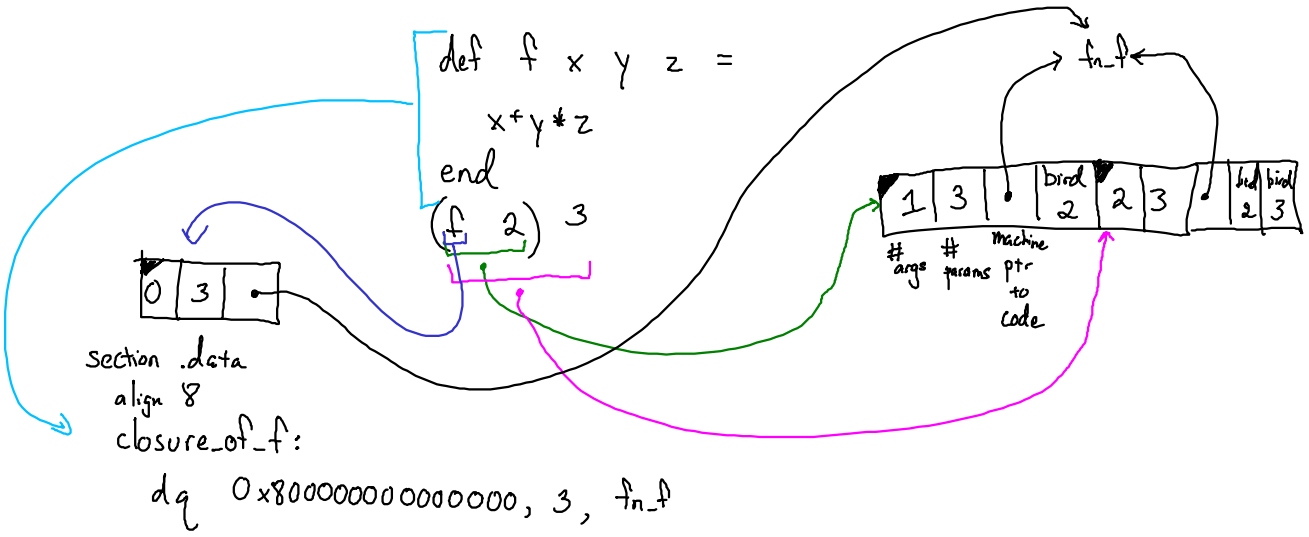
Functional Programming

- First-class functions
- Partial application

```
def f x y z =
  x+y*z
end
f 2 3 4
```

$\Rightarrow 14$

(if ... then f else g) 4



f $\Rightarrow$ mov rax, closure_of_f +1

Finch

Falcon + anonymous functions (fun x -> fun y -> x+y)

<expr> ::= .....
| fun <ident> -> <expr>

Finch

```
def twice f n =
  f (f n)
end
twice (fun a -> a*5) 4
```

⟹ 100

Falcon

```
def $0 a =
  a*5
end
def twice f n
  f (f n)
end
twice $0 4
```

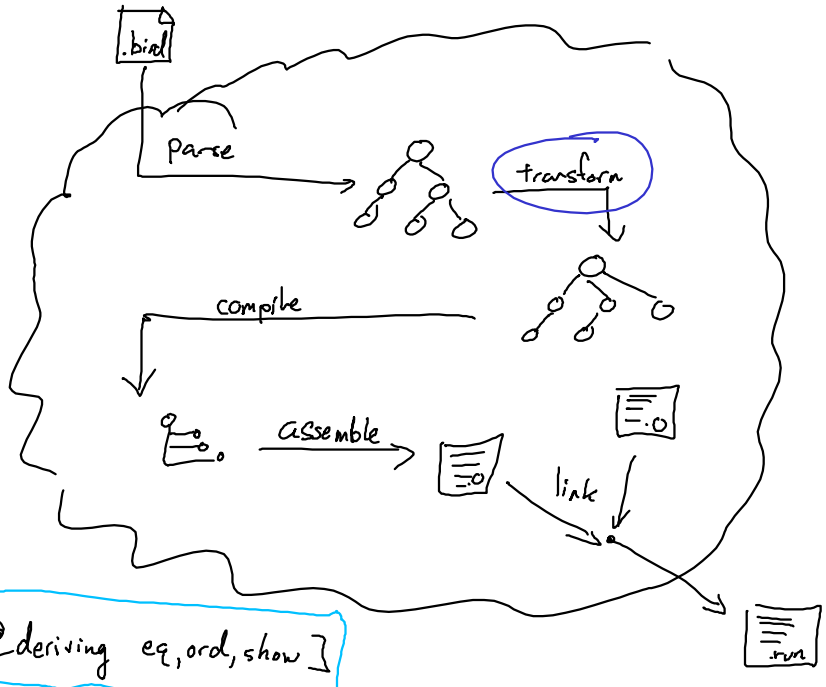
Finch

```
def twice f n
  f (f n)
end
let w=2 in
twice (fun n -> n*w) 4
```

⟹ 16

Falcon

```
def $0 w n =
  n*w
end
def twice f n=
  f (f n)
end
let w=2 in
twice ($0 w) 4
```



Closure Conversion

[@@deriving eq, ord, show]