First-class functions: functions that can be used any way any other value can be used

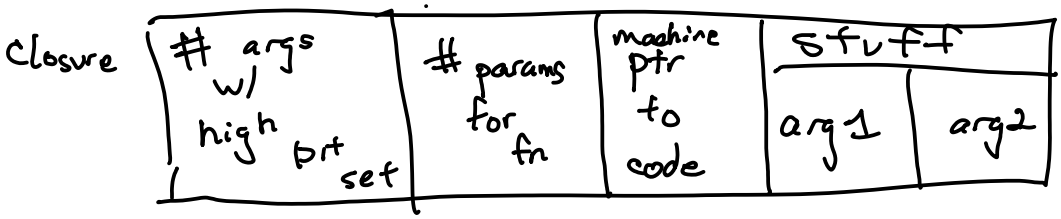
Anonymous functions: function w/o a name (fun x → x)

Partial application: applying a fn to fewer args than it has params to get a fn waiting for the rest

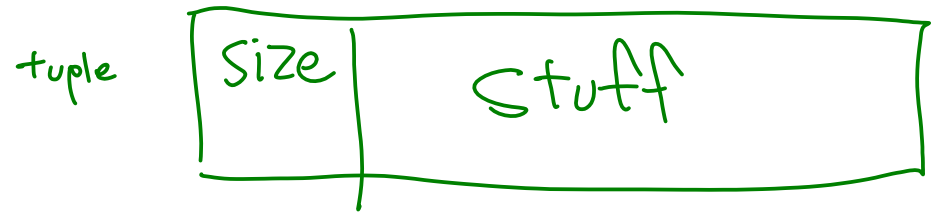
---

Foxsnake

closure ~ a construction in memory; remembers args



0x80000002

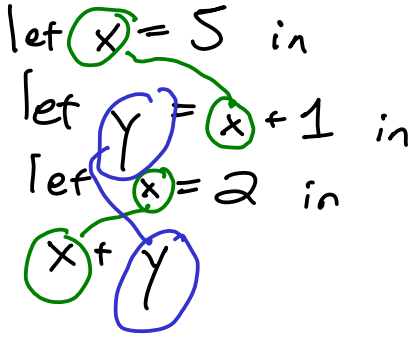


Error handling

- Distinguish tuples and closures
  - `+[1]` "not a tuple"?
- Eliminate compile-time check: wrong # args
- Syntactically: no zero-param fns
- Runtime check for application
  - one side: closure
  - other side: anything

- Unknown fn: eliminate

```
let f x y = x + y in
let g = f 1 in
let twice h x = h (h x) in
twice g
```




```
def f(x)
  x
end

let x=1 in
f(x)
```

DB

```
def f(x)
  x
end

let f=2 in
f(1)
```

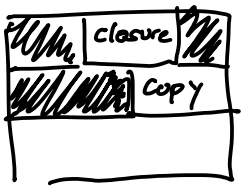
FS

```
def f x =
  x
end

let f=2 in
f 1
```

Function Calls

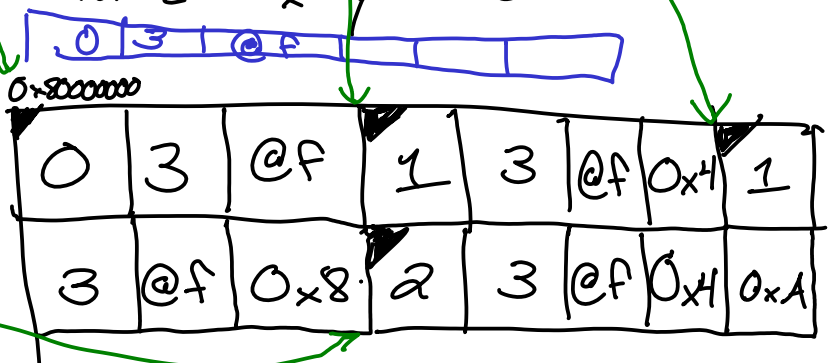
EAppl — "f x"




```
let g = f 1 in
let x = g 4 in
let y = g 5 in
...
```

- Check f is closure
- If closure f has args < # params − 1:
  - Need to make a new closure
  - Make a copy of closure (except reserving 1 more word)
    - Increase copy's arg count
    - Copy in new arg
- Else
  - Copy all args from closure and the new arg onto stack
  - Call fn! Copy fn ptr from closure into reg, then call reg "call eax"

```
let f = fun x -> fun y -> fun z -> x+y+z in
let g1 = f 2 in
let g2 = f 4 in
let h11 = g1 5 in
let h21 = g2 5 in
```



Rep Movsd

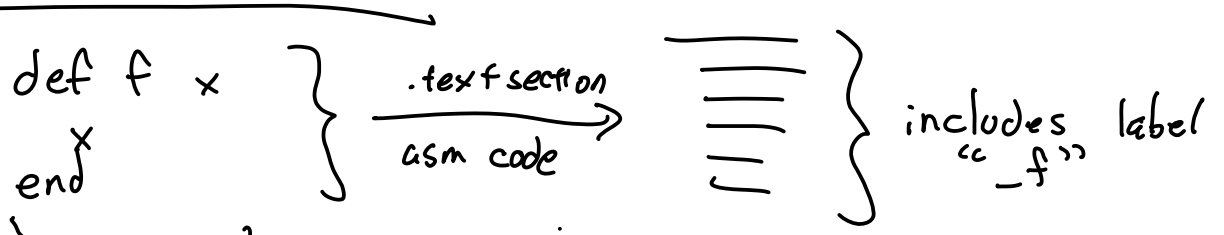
ecx — # of iterations

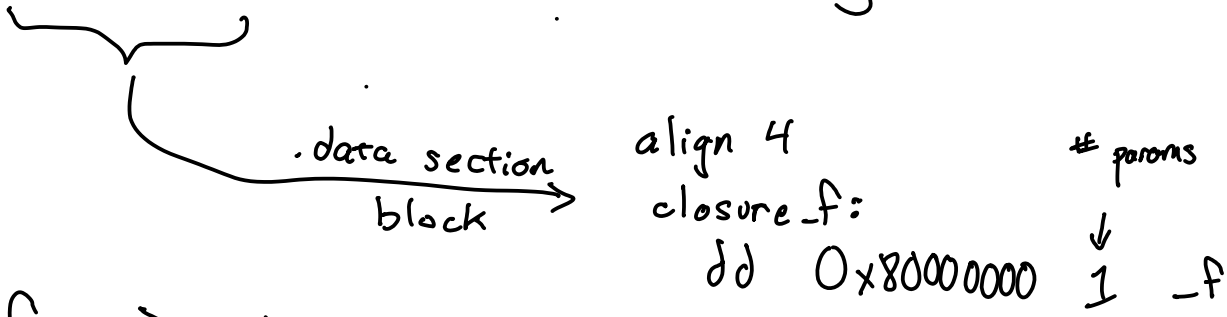
callee-saved:
- esi — Source memory
- edi — target memory

rep movsd : ecx times:

copy into [edi] value at [esi]
add 4 to edi
add 4 to esi

Initial Closures





f => closure_f + 1

```
let x =
    if b then
        f
    else
        0
in
....
```