Calling Conventions

- Specify behavior needed to call a function
- Specify behavior needed to be called

POSIX C calling conventions for x86-64

1. Caller sets up the call
2. Callee sets up a stack frame
3. Callee does work
4. Callee tears down stack frame and leaves the answer in RAX
5. Caller tears down the call

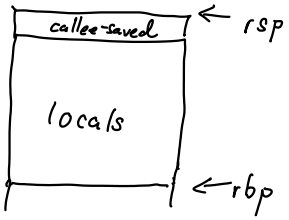
push reg ≡
```
sub rsp, 8
mov [rsp], reg
```

call lbl ≡
```
push (next instr addr)
jmp lbl
```

Suppose C function called "foo" takes 1 argument.

Concrete Assembly to call foo(5)

```
push r10        ; save caller-saved
mov rdi, 5      ; store arguments
call foo        ; do call
pop r10         ; restore caller-saved
```



Dove

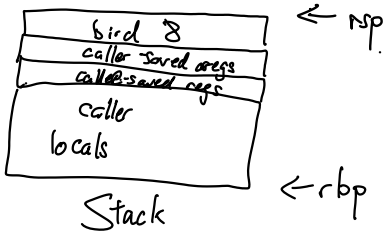
```
<program> ::= <decl_list> <expr>
<decl_list> ::= ∈
              | <decl> <decl_list>
<decl> ::= def <ident> (<param_list>) <expr> end
<paran_list> ::= ∈
               | <param> , <param_list>
               | <param>
<paran> ::= <ident>
<expr> ::= .... (Cardinal)
         | <ident> (<expr_list>)
```

```
def dbl(n)
  n * 2
end
dbl(dbl(8))
```

Calling Conventions

* POSIX C x86-64
* OCaml
* Bird
  * difference in volatile/stable registers
  * put all args on stack

```
def dbl(n)
  n * 2
end
dbl(dbl(8))
```



Concrete Assembly for dbl?

```
push rbp
mov rbp, rsp
sub rsp, 16
mov rax, [rbp+16]
mov [rbp-8], rax
mov rax, 4
mov [rbp-16], rax
mov rax, [rbp-8]
sar rax, 1
imul rax, [rbp-16]
mov rsp, rbp
pop rbp
ret
```