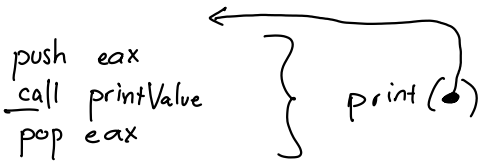
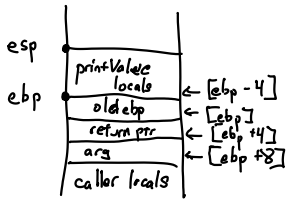

push eax
call printValue
pop eax
print(x)


esp
ebp
printValue locals
oldebp
return ptr
arg
caller locals
← [ebp - 4]
← [ebp]
← [ebp + 4]
← [ebp + 8]

```
x /= 2;
x = x / 2;
```

Dove

```
<prog> ::= <declList> <expr>
<declList> ::= ε | <decl> <declList>
<decl> ::= def <id>(<paramList>) <expr> end
         | def <id>() <expr> end

<paramList> ::= <param>
              | <param> , <paramList>
<param> ::= <id>
<exprList> ::= <expr>
             | <expr> , <exprList>

<expr> ::= ... .
         | <id>(<exprList>)
         | <id>()
```

b ? x : y

```
int f() {
  cout << "a" << endl;
  return 1;
}
```

```
def f(x)
  x + 1
end
f(4)
```

ASTs

```
type expr =
    | ....
    | ECall of string * expr list


type declaration =
    | DFunction of string *
                   string list *
                   expr

type program =
    | Program of decl list * expr
```

Compiling Functions

```
let compile_declaration (d : declaration) : instruction list =
  let label = "function_" ^ name of fn? in
```

Callee things : ebp, esp

Compile_expression param_env body

bird_main:

def f(x)
  x
end

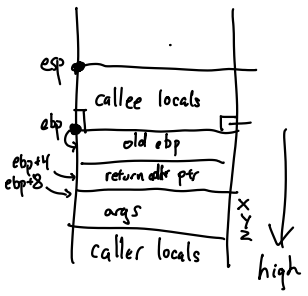
Call foo

push onto stack the next instr addr
jumps to label foo

def else_6(x):
  ....
end

fresh_name "fn"

fn_4 ↦ else_6



let x=5 in
....

def f(x, y, z)
  ⋮
end

$$\left(\{x \mapsto -4\}, -8\right)$$

$$\left(\begin{Bmatrix} x \mapsto +8 \\ y \mapsto +12 \\ z \mapsto +16 \end{Bmatrix}, -4\right)$$

$$\left(\begin{Bmatrix} x \mapsto -8 \\ y \mapsto -12 \\ z \mapsto -16 \end{Bmatrix}, 4\right)$$

```
def f(x, y)
  after(x)
end
```

⟹

```
function_f:
  push ebp
  mov ebp, esp
  sub esp, 0
  mov eax, [ebp+8]
  add eax, 2
  mov esp, ebp   } leave
  pop ebp        }
  ret
```