Compile (ast : expr) : instruction list =

{
{
{

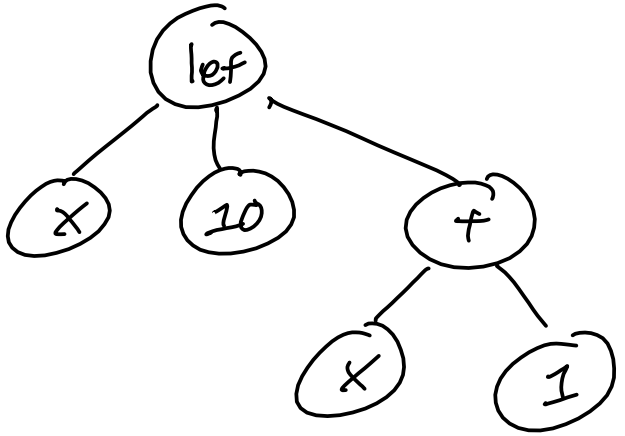
;;

<expr> ::= <integer>
| inc (<expr>)
| dec (<expr>)
| <identifier>
| let <identifier> = <expr> in <expr>

inc(5) ⟶

```
mov eax, 5
add eax, 1
```

let x = 10 in x+1



```
let x =
    let y = 10
    in y + 1
in x + 2
```

```
let y =
    let x = 4 in
    inc(x)
in
dec(x)
```

```
let y = 5 in x
```

```
let x = e1 in e2
```

```
x
```

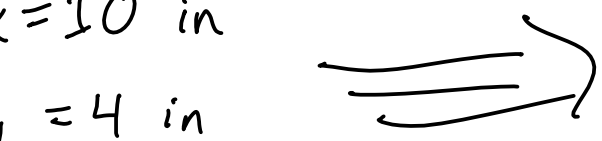
---

let x = 10 in x

"mov eax, 10"

*(esp - 4)

let x = 10 in
let y = 4 in
x

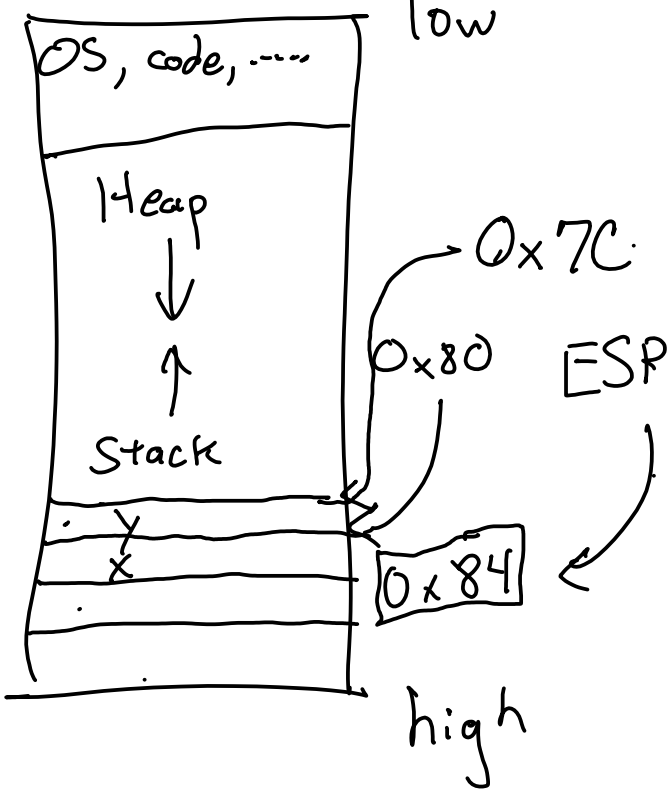


```
mov [esp-4], 10
add esp, -4
mov [esp-4], 4
add esp, -4
```

esp + 4

esp + 8

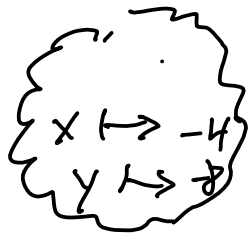
```
mov eax, [esp+8]
```



lowest memory address currently in use on stack

```
let x = 10 in
let y = 11 in
x
```

```
mov [esp-4], 10
mov [esp-8], 4
mov eax, [esp-4]
```



environment.ml

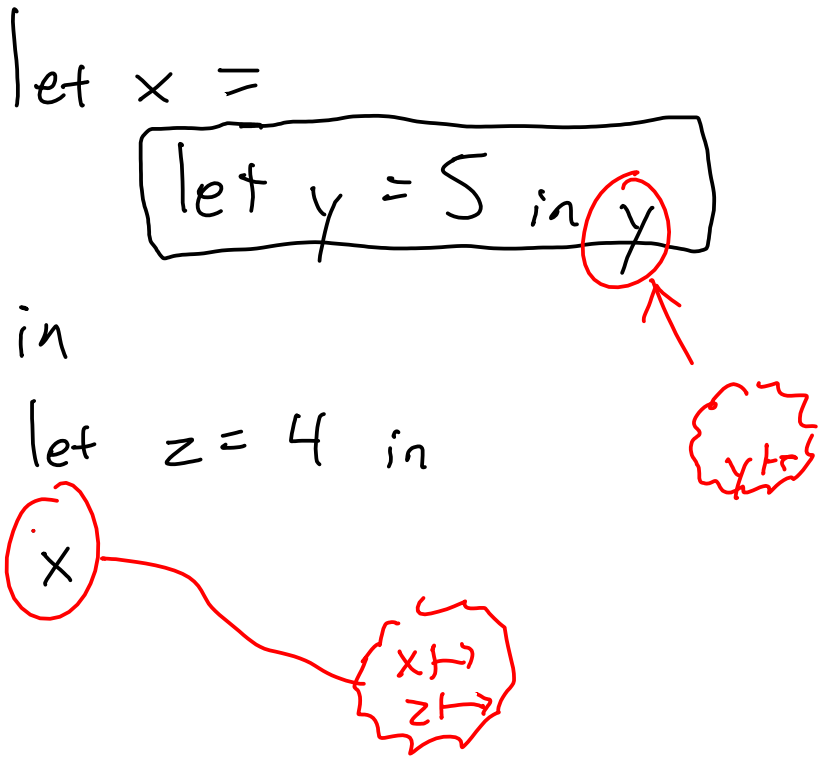
String Map

StringMap.insert "x" (-4) old

```
let rec compile (ast : expr) (env : environment) : instruction list =
  match ast with
  | EInt n -> [XMov(XRegister EAX, XConstant n)]
  | ... ->
  | ELet (x, e1, e2) ->
      let instrs1 = compile e1 env in
      let new_env = env_add x env in
      let offs = env_get new_env x in
      let store_instr = XMov(XRegisterOffset(ESP, offs), XRegister EAX)
      let instrs2 = compile e2 new_env in
      instrs1 @ [store_instr] @ instrs2
```

mov eax, result of e1
mov [esp -4], eax

let x = x in x


let x =
let y = 5 in y
in
let z = 4 in
x
y ↦
x ↦
z ↦