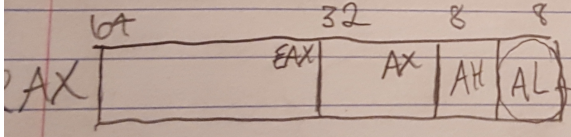
x86₀ → x86₁

arg ::= ...
| (byte-reg register)
cc ::= e | ne | g | ge | l
instr ::= ...
| (xorq arg arg) | (cmpq arg arg)
| (set cc arg) | (movzbq arg arg)
| (jmp label) | (jmp-if cc label)
| (label label)
x86₁ ::= (program info (type type) instr+)

#t ⇒ $1 #f ⇒ $0
(not x) ⇒ xorq x, $1

xor	1	0
0	1	0
1	0	1

(< 3 4)
⇒ cmpq $4 $3 → sets EFLAGS
read the Flags
sete dst
byte-register



How to go from byte-register to a regular register?
movzbq Al Reg

labels:

```
my-label: movq
          addq
          cmpq
          je my-label
          movq
```

(if (cmp a1 a2) thn els) d...

⇒
```
cmpq a2 a1
jcmp thn_lab
els...
jmp end
thn_lab: thn...
end: d...
```

Select-instr

(set! x #t) ⇒ (movq $1 x)
(set! x #f) ⇒ (movq $0, x)

(set! lhs (cmp a1 a2))
⇒
```
(cmpq a2 a1)
(set cmp (byte-reg AL))
(movzbq (byte-reg AL) LHS)
```

si(if c te) ⇒ (if si(c) si(t) si(e))

liveness-analysis 8/3

(liveness (if (cmp a1 a2) (t...) (e...)), LA)

LB_t = (liveness (t...) LA)
LB_e = (liveness (e...) LA)

return $(LB_t \cup LB_e) \cup vars(a_1) \cup vars(a_2)$

New-Pass: Lower conditionals

(if (cmp arg1 arg2) ths els)

⇒

(cmpq arg2 arg1)
(jmp-if cc thenlabel)
els
(jmp endlabel)
(label thenlabel)
ths
(label endlabel)

patch-instr

(cmpq arg2 arg1)

↳ must be a register

cmpq $3 $4

⇓

movq $3, %RAX
cmpq %RAX, $4

(if #t A B) ⇒ A dead code elimination
(if (< 3 8) A B) ⇒ A partial evaluation / constant folding

(if (not x) A B)
⇒ (if x B A)

(if (begin x y) A B)
⇒ x; (if y A B)

(if (begin x #t) A B) x; A
x = new cons (1, 2)

(if (new C $\vec{A}$) T F) ⇒ $\vec{A}$; T

(begin expr... expr)

(begin (+ 3 4) (+ 2 2)) = 4

(if C (let ([x a]) T) F)
⇒ (let ([x a])
(if C T F))

x ∉ FV(C) ∪ FV(F)
a has no side effect