X86 conditionals

Monday, 21 February 2022 8:30 AM

not (not #t)
#f

xor

	0	1
0	0	1
1	1	0

xor $1, v

v = 0 → 1

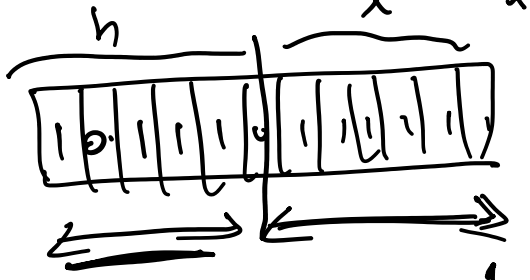
cmpq

EFLAGS register

cmpq arg1 arg2 <

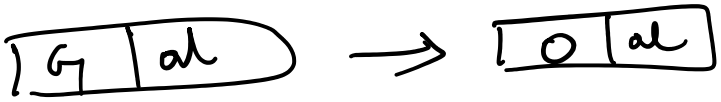
→ setl %al

lgeq



setl %bl

movzbq %al var



var = any regs or stack

-8(%rbp)

if (< arg1 arg2)

goto l1; →

else

goto l2; →

cmpq arg2 arg1

jl l1 → then

jmp l2 → else

(let [x read])
(let [y read])
(if (if (< x 1) (eq? x 0) (eq? x 2))
(+ y 2)
(+ y 10)))

(if (< x 1)) (if (cond ( ) ( )))

cmp $1 x
set l %al
movzbq %al, tmp 0 1

cmpq $1, tmp
je then_branch
jmp else_branch

1) if cond then else
↓
atomic comp
<, eq

(if (if (< x 1) (eq? x 0) (eq? x 2))
(+ y 2)
(+ y 10)))

x = 2

(if (< x 1) → if_1
(if (eq? x 0) → if_2
(+ y 2) → then_2
(+ y 10)) → else_2) → then_1
(if (eq? x 2)
(+ y 2)
(+ y 10)) } 3 → else_1

then : (+ y 2)
el : (+ y 10)

then_1 : { if (eq? x 0)
goto then
else goto el

else_1 : { if (eq? x 2)
goto then
else goto el

(if (< x 1)
goto then_1
else
goto else_1

Dexp-tail
```
(let [[x read]]
  (let [[y read]]
    (if (if (< x 1) (eq? x 0) (eq? x 2))
        (+ y 2)
        (+ y 10))))
```

↳ (1.1) exp-tail (let [[y read]] ....) =>B0

(1.1.1) exp-tail (if (if (> x 1) (eq? x 0) (eq? x 2)) (+ y 2) (+ y 10)))

(1.1.1.1) exp-tail (+ y 2)
↳ return (+ y 2) → B3

(1.1.1.2) exp-tail (+ y 10)
↳ return (+ y 10) → B4

(1.1.1.3) (exp-pred (if (> x 1) (eq? x 0) (eq? x 2)) (goto B3) (goto B4))
e1 e2 e3

(1.1.1.3.1)
(exp-pred (eq? x 0) (goto B3) (goto B4))
if (eq? x 0)
  goto B3
else goto B4
=> B6

(1.1.1.3.2)
(exp-pred (eq? x 2) B3 B4)
if (eq? x 2)
  goto B3
else goto B4
B7

(1.1.1.3.3)
(exp-pred (> x 1) goto B6 goto B7)
if (> x 1)

-> BS

if (> x 1)
goto B6
else
goto B7

} B5

exp_assign (read) y B1

(1.2) exp_assign (read x B0)
[x = read
B0]

x = read
y = read

B5
c > b

(1<2) ≤ a < b

(if (and (cond1) (cond 2)))

(if (let ( )))

exp_pred (let ... (eq ...))

match e

[let x val body] ( )

if ( )

B1

(if (let x out))

B1

if (eq? x y)

v

(eq? x y)

(if and ( ) ( ))

(if (if ( ) ( ) ( ))

Exp-ctrl implementation

Monday, 21 February 2022 9:15 AM

exp_predicate R2_pred × C1 tail × C2 tail

cond if cond then else

↳ C2 tail × var list

exp_pred (## B1 B2)
return B1

if ....
C2
C2

exp_pred (cmp atm1 atm2) B1 B2

return (if (cmp atm1 atm2)
- (goto B1)
else
- (goto B2))

exp_pred ((if e1 e2 e3) B1 B2)

exp_tail (if (if e1 e2 e3) then else)

exp_tail then → B1
exp_tail else → B2

↳ exp_pred (if e1 e2 e3) B1 B2)

exp_pred (e2 goto B1 goto B2) → B3
exp_pred (e3 goto B1 goto B2) → B4

esp-pred (e3 goto B1 goto B2) → B4

esp-pred (e1 goto B3 goto B4) -> B5