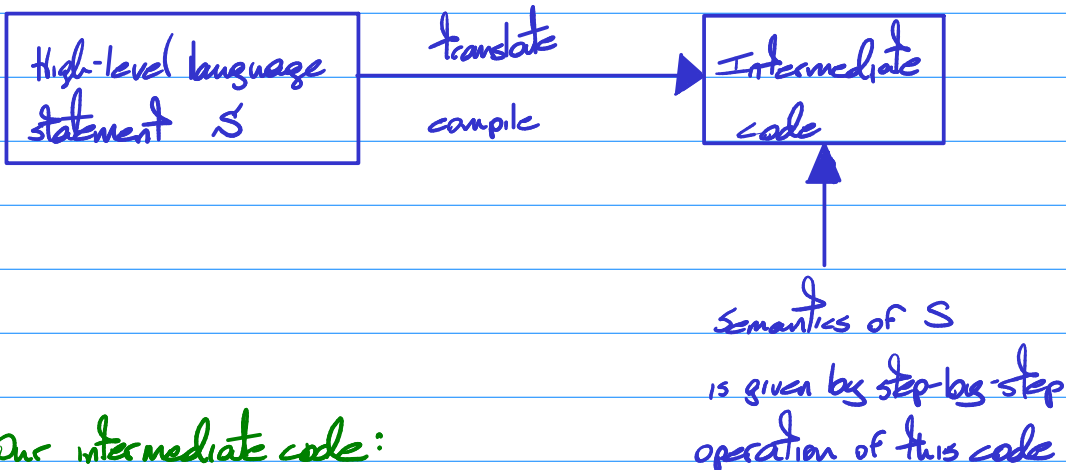


316 10-13-09

Operational Semantics

- operational semantics by intermediate code and virtual machine
design a simple, clear virtual machine and its assembly-like intermediate code. This is presumed to be fully understood



Our intermediate code:

goto <label> - unconditional goto to label

if (<var> = <bool>) goto <label>

<bool> = 0 (false) or 1 (true)

1-branch conditional

IF (B) S

code to evaluate B

b = result of above evaluation

if (b=0) goto Out

code to execute S

Out:

2-branch conditional

IF (B) S₁ else S₂

code to evaluate B

b = result of above evaluation

if (b=0) goto Else

code to execute S₁

goto Out

Else: code to execute S₂

Out:

Alternative:

if (b=1) goto L1

code to execute S₂

goto Out

L1: code to execute S₁

Out:

- Switch statements

Switch (E) { L1: S₁ ... L_n: S_n; default }

③ fall-through semantics (C++, Java)

code to evaluate E

c = result of above evaluation

if ($c = L_1$) goto A_1

if ($c = L_2$) goto A_2

⋮

if ($c = L_n$) goto A_n

goto A_d

A_1 : code to execute S_1 ; break;

A_2 : code to execute S_2 ; break;

A_n : S_n ; break;

A_d : S_d ; break;

② Exclusive-case semantics (Pascal / Ada)

→ Same code

A_1 : code to execute S_1

goto Out

A_2 : code to execute S_2

goto Out

A_n : code to execute S_n

A_d : code to execute S_d

Out:

while (B) S

Loop: code to execute B

b = result of above evaluation

if ($b = 0$) goto Out

code to execute S

goto Loop

Out:

do S while (B)

Loop: code to execute S

code to evaluate B

b = result of above evaluation

if ($b = 1$) goto Loop

for (init-statement; B; incr-statement) S

code to execute init-statement

Loop: code to evaluate B

b = result of above evaluation

if (b=0) goto out

code to execute S

code to execute incr-statement

goto Loop

out:

break

{
: S₁

code to execute S₁

break;

goto out

: S₂
}

code to execute S₂

out:

Let this be the immediately surrounding block

if (B₁) { while (B₂) S₁ } else S₂

Q = provide intermediate-code semantics

A = code to execute B₁

b₁ = result of above evaluation

if (b₁=0) goto Else

Loop: code to evaluate B₂

b₂ = result of above evaluation

if (b₂=0) goto out

code to execute S₁

goto Loop

out: goto outIF

Else: code to execute S₂

outIF:

→ inefficient and redundant

while (B₁) { if (B₂) S₁ else S₂ }

Loop: code to evaluate B₁

b₁ = result of above evaluation

if (b₁ = 0) goto Out

code to evaluate B₂

b₂ = result of above evaluation

if (b₂ = 0) goto Else

code to execute S₁

goto OutIF

Else: code to execute S₂

OutIF: goto Loop

Out:

switch (x)

{ L1: { if (B) S₁; break; }

L2: { S₂; break; }

Ld: Sd;

}

if (x = L₁) goto A₁

if (x = L₂) goto A₂

goto Ad

A₁: code to evaluate B

b = result of above evaluation

if (b = 0) goto Out1

code to execute S₁

Out1: goto Out

A₂: code to execute S₂

goto Out

Ad: code to execute Sd

Out:

Redundant and inefficient