Generation of Intermediate Code

66.648 Compiler Design Lecture (03/30//98)

Computer Science

Rensselaer Polytechnic

Lecture Outline

- Intermediate Codes for if
- Backpatching
- Examples
- Administration

Short-circuit code approach

Idea: use conditional jump instruction after evaluating partial expression.

Example:

a and b and c

if a were false, then we need not evaluate the rest of the expressions. So, we insert labels E.true and E.false in the appropriate places.

Short Circuit - Contd

If a goto E.true

goto E.false

E.true: if b goto E1.true

goto E.false

E1.true: if c goto E2.true

goto E.false

E2.true : exp =1

E.false: exp =0

Syntax Directed Translation

The translation uses the following attributes: true, false, and code. True and false are inherited attributes whereas code is a synthesized attribute. The inherited attribute true and false specify the true and false branch destinations for the short circuit code generated by sub-expressions.

Caution: The true and false inherited attributes cannot be evaluated on the fly using bottom-up parsing. One needs to build a parse tree and the dependency graph for evaluating the attributes.

Syntax-Directed Translation

E1 --> E2 or T { E2.true =E1.true; T.true=E1.true; E2.false=newlabel(); T.false=E1.false; E1.code || gen("label"E2.false,":") || T.code }

E1--> T { T.true=E1.true;T.false=E.false;E.code=T.code}

T1--> T2 and F { T2.true=newlabel();F.true=T1.true; T2.false=T1.false;F.false=T1.false;T1.code=T2.code||gen("label",T2.true,".")||F.code}

T-->F {F.true=T.true;F.false=T.false;T.code=F.code}

Syntax-Directed Translation

F--> NOT F1 {F1.false=F.true;F1.true=F.false; F.code=F1.code}

F--> (E) { E.true=F.true;E.false=F.false; F.code=E.code}

F--> ID1 RELOP ID2 {F.code=gen("if"ID1.place RELOP ID2.place "goto " F.true|| gen("goto" F.false}

F--> True { F.code=gen("goto"F.true)}

F--> false {F.code=gen("goto"F.false)}

Generating Code for Switch Stmts

Switch/Case statement in source language form:

switch(expr) {

case_value1: Stmt1

..

Case_valuen-1:Stmt(n-1)

default: Stmtn

}

Translation into 3 A C

Translation of source switch/case statement into intermediate language 3 A C.

Switchstmt.code=

code to evaluate expression into temporary t

goto test

L1: code for Stmt1

goto next

L2: code for Stmt2

Switch Stmt

Goto next

L(n-1): code for Stmt(n-1)

go to next

L(n): code for Stmt(n)

goto next

test: if t=V1 goto L1

if (t=V2) goto L2

… if (t=V(n-1)) goto L(n-1) else goto L(n)

Switch Statement

It becomes the responsibility of the code generation phase to generate an efficient implementation of a n-way branch.

Common ideas: use a hash table, balance tree or a combination of the above.

Back patching

Back patching is a technique to solve the problem of replacing symbolic names in goto statements by the actual target addresses.

This problem comes up because of some languages do not allow symbolic names in the branches.

Idea: Maintain a list of branches that have the same target label and replace them once they are defined.

Example

Source:

if a or b then

if c then

x= y+1

Translation:

if a go to L1

if b go to L1

go to L3

Example-Contd

L1: if c goto L2

goto L3

L2: x= y+1

L3:

After Backpatching:

100: if a goto 103

101: if b goto 103

goto 106

Back patching

List operations:

merge(p1,p2) - merges two lists pointed by p1 and p2

back patch(p,j) inserts the target label j for each list pointed by p.

Attributes are: addr (address of the marker symbol), nextlist (list of jumps whose addresses have to be filled by address that follows the current stmt), truelist and falselist

Back patching- Syntax Directed Translation

We assume that the function next_address() returns the address to be occupied by the next intermediate language statement that is generated.

Stmts--> stmt { stmts.nextlist=stmt.nextlist;}

stmts1-->stmts2 ‘;’ marker stmt { backpatch (stmts2. extlist,marker.addr); stmts1.nextlist = stmt.nextlist}

marker --> epsilon {marker.address = nextaddress() }

Back patching- Syntax Directed Translation

Stmt1 --> IF expr THEN marker stmt2 { backpatch(expr.truelist,marker.addr); stmt1.nextlist=merge(stmt2.nextlist,expr.falselist)}

Stmt1 --> While marker expr DO marker1 stmt2 { backpatch(stmt2.nextlist,marker.addr);backpatch(expr.truelist,marker1.addr); stmt1.nextlist=expr.falselist}

stmt: ID = expr { stmt.nextlist=nil}

Examples

If a or b then

if c then

x=y+1

100: if a goto ____

101: if b goto ____

102: goto ____

103: if c goto ____

104 goto ____

3 AC Statements

Comments and Feedback

Projects 3 and 4 are out. Please start working. PLEASE do not wait for the due date to come.

We are in chapter 8. Please read that chapter and read the relevant portion of Java. Please keep studying this material and work exercises.