CSE302: Compiler Design

Instructor: Dr. Liang Cheng
Department of Computer Science and Engineering
P.C. Rossin College of Engineering & Applied Science
Lehigh University

April 19, 2007

Outline

- Recap
  - Intermediate code generation
- Things related to project part II
- Summary

Translation of Boolean Expressions (1)

- S→if({B.true=newlabel(); B.false=S.next;}
  B) S1 {S.code=B.code || label(B.true) || S1.code
  - if B goto L1
  - goto L2
  - L1: S1
  - L2:
- B → B1 || B2
- S → if({B.true=newlabel(); B.false=S.next;
  B1.true=B.true; B1.false=newlabel();
  B2.true=B.true; B2.false=B.false;}
  B1||B2) {B.code=B1.code || label(B1.false) || B2.code}
  S1 {S.code=B.code || label(B.true) || S1.code
  - if B1 goto L1
  - goto L2
  - L2: if B2 goto L1
  - goto L3
  - L1: S1
  - L3:

Translation of Boolean Expressions (2)

- B → B1 && B2
- S → if({ B.true=newlabel(); B.false=S.next;

  B1.true=newlabel(); B1.false=B.false;

  B2.true=B.true; B2.false=B.false;}

  B1&&B2) {B.code=B1.code || label(B1.true) || B2.code}

  S1 {S.code=B.code || label(B.true) || S1.code

```
      if B1 goto L2
      goto L3
L2:   if B2 goto L1
      goto L3
L1:   S1
L3:
```

Translation of Boolean Expressions (3)

- B $\rightarrow$!B1
- S $\rightarrow$ if({B.true=newlabel(); B.false=S.next;
  B1.true=B.false; B1.false=B.true;}
  !B1) {B.code=B1.code}
  S1 {S.code=B.code || label(B.true) || S1.code}

  - if B1 goto L2
  - goto L1
  - L1: S1
  - L2:

Translation of Boolean Expressions (4)

- B → E1 rel E2
- S → if({B.true=newlabel(); B.false=S.next;}

  E1 rel E2) {B.code=E1.code || E2.code ||

  gen('if E1.addr rel.op E2.addr 'goto' B.true ||

  gen('goto' B.false);

  }

  S1 {S.code=B.code || label(B.true) || S1.code}

  - E1.code
  - E2.code
  - if E1 rel.op E2 goto L1
  - goto L2
  - L1: S1
  - L2:

Translation of Boolean Expressions (5)

- B $\rightarrow$ true {B.code=gen('goto' B.true);}
- B $\rightarrow$ false {B.code=gen('goto' B.false);}

Outline

- Recap
- Things related to project part II
- Summary

Support for Arbitrary Value Types

- YYSTYPE may be a union type

```
%union {
    body of union ...
}
```

  - This declares the Yacc value stack, and the external variables yylval and yyval, to have type equal to this union.
  - -d: the union declaration is copied onto the y.tab.h
- The union member names are associated with terminals and nonterminals
  - %token <name> _TOKEN
  - %type <name> _nonterminal

LR parsing with L-attributes

- Section 5.5.4
- Yacc implementation

SDT for While

- SDD of **while** to generate 3-address code
  - $S \rightarrow$ while (C) S
    - while(i<a) i=i*2;
      j=j*i;
  - label L5: if i<a goto L6
                    goto L7
    label L6: i=i*2
                    goto L5
    label L7: j=j*i

- $S \rightarrow$ while (C) S L1=new(); L2=new();
  S1.next=L1;
  C.true=L2; C.false=S.next;

S.code= **label** || L1 || C.code || **label** || L2 || S1.code

Attribute Computation

- For synthesized attribute computations
  - End of the production body
- For computing inherited attributes of a nonterminal
  - Immediately before the nonterminal occurrence

$S \rightarrow$ while ({ L1=new(); L2=new();
C.true=L2; C.false=S.next;
}
C) { S1.next=L1; }
S1 { S.code= **label** || L1 || C.code || **label** || L2 || S1.code; }

Markers

- $M \to \varepsilon$
  - Hold the inherited attributes

Outline

- Recap
  - Intermediate code generation
- Things related to project part II
- Summary