基于C0文法的编译器V7.0

10011603 2016302731 冯宇

**一.主要功能**

添加逻辑操作“与”和“或”，用于对if,for，while中的条件语句进行判断

<条件>::=<条件子项>|'('＜条件子项＞')'[<&& ||>'('＜条件子项＞')']

**二.代码实现**

首先在main.h中的symbol结构体中添加andsy,orsy类型

在词法分析器中lexical.cpp添加以下代码识别于字符&&和或字符||

else if (ch == '|') {

nextch();

if (ch == '|') {

k++;

sy = orsy;

nextch();

}

}

else if (ch == '&') {

nextch();

if (ch == '&') {

k++;

sy = andsy;

nextch();

}

}

在语法分析phraser.cpp中，将原本的<条件>函数修改变量名为条件子项，添加新的条件函数

//<条件>::=<条件子项>|'('＜条件子项＞')'[<&& ||>'('＜条件子项＞')']

void judgement(string &infixString) {

string logicalLeft = "";

string logicalRight = "";

symbol logicalSy = andsy;

if (sy == lparent) {

insymbol();

judgement\_child(logicalLeft);

if (sy == rparent) {

if (skipFlag) {

skipFlag = false;

}

insymbol();

}

else if (!skipFlag) {

error(RightParentLost); // Right parenthesis lost

symbol nexts[] = { semicolon, rbrace };

int length = sizeof(nexts) / sizeof(symbol);

skipUntil(nexts, length);

}

if (sy == andsy || sy == orsy) {

logicalSy = sy;

insymbol();

if (sy == lparent) {

insymbol();

judgement\_child(logicalRight);

if (sy == rparent) {

if (skipFlag) {

skipFlag = false;

}

insymbol();

}

else if (!skipFlag) {

error(RightParentLost); // Right parenthesis lost

symbol nexts[] = { semicolon, rbrace };

int length = sizeof(nexts) / sizeof(symbol);

skipUntil(nexts, length);

}

}

infixString = createTempVar();

insertTable(vars, ints, infixString.c\_str(), 0, level, 0); // 类型为数字，1表示判断成立，0表示判断不成立

switch (logicalSy) {

case andsy:

insertInfix("AND", logicalLeft, logicalRight, infixString);

break;

case orsy:

insertInfix("OR", logicalLeft, logicalRight, infixString);

break;

}

}

}

else {

judgement\_child(infixString);

}

}

四元式部分infixNotation.cpp文件中

void insertInfix(string ioperator, string operand1, string operand2, string operand3)

添加处理语句

else if (ioperator == "AND") {

outputBuff = outputBuff + operand3 + " = " + operand1 + " && " + operand2 + "\n";

}

else if (ioperator == "OR") {

outputBuff = outputBuff + operand3 + " = " + operand1 + " || " + operand2 + "\n";

}

转化为MIPS代码时，添加新的MIPS语句

else if (notation.ioperator == "AND") {

operandsToRegister(notation);

insertCode("and $t0, $t1, $t2");

storeThirdOperand(notation.operand3, "$t0");

}

else if (notation.ioperator == "OR") {

operandsToRegister(notation);

insertCode("or $t0, $t1, $t2");

storeThirdOperand(notation.operand3, "$t0");

}

**三.测试结果**

|  |  |  |
| --- | --- | --- |
| Input code | 四元式 | MIPS代码 |
| int main(){  int ax,bx;  int i;  bx=3;  i=0;  while((i==0)&&(bx>2)){  ax=bx+i;  }  ax=ax\*2;  return;  } | int main()  int ax  int bx  int i  bx = 3  i = 0  LABEL1  #t0 = i == 0  #t1 = bx > 2  #t2 = #t0 && #t1  goto LABEL0, if #t2 != 0  goto LABEL2, if #t2 == 0  LABEL0  #t3 = bx + i  ax = #t3  jmp LABEL1  LABEL2  #t4 = ax \* 2  ax = #t4  return | .data  .text  j main  main:  sw $fp, ($sp)  move $fp, $sp  sub $sp, $sp, 8  sub $sp, $sp, 28  li $t0, 0  sw $t0, -16($fp)  li $t0, 3  sw $t0, -12($fp)  LABEL1:  lw $t1, -16($fp)  li $t2, 0  seq $t0, $t1, $t2  sw $t0, -20($fp)  lw $t1, -12($fp)  li $t2, 2  sgt $t0, $t1, $t2  sw $t0, -24($fp)  lw $t1, -20($fp)  lw $t2, -24($fp)  and $t0, $t1, $t2  sw $t0, -28($fp)  lw $t1, -28($fp)  li $t2, 0  bne $t1, $t2, LABEL0  lw $t1, -28($fp)  li $t2, 0  beq $t1, $t2, LABEL2  LABEL0:  lw $t1, -12($fp)  lw $t2, -16($fp)  add $t0, $t1, $t2  sw $t0, -8($fp)  j LABEL1  LABEL2:  lw $t0, -8($fp)  sw $t0, -32($fp)  lw $t1, -32($fp)  sll $t0,$t1,1  sw $t0, -8($fp)  move $v1, $t0  j mainTail  mainTail: |
| Input code | 四元式 | MIPS代码 |
| int main(){  int ax,bx;  int i;  bx=3;  ax=1;  i=0;  while((i==0)&&(bx>2)){  if((bx==3)||(ax>0))ax=bx+i;  else i=i+5;  }  ax=ax\*2;  return;  } | int main()  int ax  int bx  int i  i = 0  ax = 1  bx = 3  LABEL1  #t0 = i == 0  #t1 = bx > 2  #t2 = #t0 && #t1  goto LABEL0, if #t2 != 0  goto LABEL2, if #t2 == 0  LABEL0  #t3 = bx == 3  #t4 = ax > 0  #t5 = #t3 || #t4  goto LABEL3, if #t5 == 0  ax = bx + i  jmp LABEL4  LABEL3  i0 = i  i = i0 + 5  LABEL4  jmp LABEL1  LABEL2  ax0 = ax  ax = ax0 \* 2  return | .data  .text  j main  main:  sw $fp, ($sp)  move $fp, $sp  sub $sp, $sp, 8  sub $sp, $sp, 44  li $t0, 0  sw $t0, -16($fp)  li $t0, 1  sw $t0, -8($fp)  li $t0, 3  sw $t0, -12($fp)  LABEL1:  lw $t1, -16($fp)  li $t2, 0  seq $t0, $t1, $t2  sw $t0, -20($fp)  lw $t1, -12($fp)  li $t2, 2  sgt $t0, $t1, $t2  sw $t0, -24($fp)  lw $t1, -20($fp)  lw $t2, -24($fp)  and $t0, $t1, $t2  sw $t0, -28($fp)  lw $t1, -28($fp)  li $t2, 0  bne $t1, $t2, LABEL0  lw $t1, -28($fp)  li $t2, 0  beq $t1, $t2, LABEL2  LABEL0:  lw $t1, -12($fp)  li $t2, 3  seq $t0, $t1, $t2  sw $t0, -32($fp)  lw $t1, -8($fp)  li $t2, 0  sgt $t0, $t1, $t2  sw $t0, -36($fp)  lw $t1, -32($fp)  lw $t2, -36($fp)  or $t0, $t1, $t2  sw $t0, -40($fp)  lw $t1, -40($fp)  li $t2, 0  beq $t1, $t2, LABEL3  lw $t1, -12($fp)  lw $t2, -16($fp)  add $t0, $t1, $t2  sw $t0, -8($fp)  j LABEL4  LABEL3:  lw $t0, -16($fp)  sw $t0, -44($fp)  lw $t1, -44($fp)  addi $t0,$zero,5  add $t0,$t0,$t1  sw $t0, -16($fp)  LABEL4:  j LABEL1  LABEL2:  lw $t0, -8($fp)  sw $t0, -48($fp)  lw $t1, -48($fp)  sll $t0,$t1,1  sw $t0, -8($fp)  move $v1, $t0  j mainTail  mainTail: |

**四. Debug以及C++知识点**

**1.文本指针的及时关闭**

最开始修改之后运行程序，代码修改之后，出现mipsCode.asm中无值，而OptMIPSCode.asm中有值的情况，怀疑是fstream文本指针没有及时关闭，添加mipsFile\_old.close();及时关闭 文本指针后两个文件生成结果正常

**2.修改不完善**

改之后程序运行正确但是优化出错，发现是因为在infixOPt中的insertNewInfix函数没添加AND和OR语句，导致没有输出相关的MIPS代码

