Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №8**

З дисципліни «СП»

Виконав: Перевірив:

Студент групи ІО-21 доц. Пустоваров В.І.

Коноз А.О.

Дата здачі\_\_\_\_\_\_\_\_\_\_\_\_\_

Захищено з балом\_\_\_\_\_

Київ 2014

**Завдання**

double b, a[4]={2,3,4,5};

unsigned n=2,d=5;

b=d!=0;

b=n\*a[n];

**Код програми:**

// spLb9.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "..\spLb3\token.h"

#include "..\spLb3\visgrp.h"

#include "..\spLb4\tables.h"

#include "..\spLb4\lexan.h"

#include "..\spLb5\syntaxP.h"

#include "..\spLb4\langio.h"

#include "..\spLb7\seman.h"

//#include "..\spLb8\interp.h"

#include "codgen.h"

#include <stdio.h>

#include <string.h>

extern struct recrdKWD tablKWD[];

extern struct lxNode nodes[MAX\_NODES]; // масив приймач вузлів дерева

extern enum ltrType ltClsC[256];

extern enum ltrType ltClsP[256];

extern enum tokType dlCdsC[256];

extern enum tokType dlCdsP[256];

extern enum ltrType ltClsC[256];

extern enum ltrType ltClsP[256];

enum ltrType \*ltCls=ltClsC;

enum tokType \*dlCds=dlCdsC;

char file\_name[20];

int main(int argc, char\* argv[])

{int nn=-1, nr=0, nc=1; //np,

if (argc>1)

{strcpy(file\_name,argv[1]);

printf("Processing file -- %s\n",file\_name);}

else

{printf("Please enter file Name: ");

scanf("%s",file\_name);

strcat(file\_name,".h");

}

opFls(file\_name);

LxAnInit('C');

// srtBin(tablKWD, 67);

do{//np=nn;

nn=LxAnlzr(); // лексичний аналіз

}while(nodes[nn].ndOp!=\_EOF);

prLaTxt(nodes,nn); // реконструкція лексичного аналізу

printf("\n");

SxAnInit('C'); // синтаксичний аналіз

nr=0; nc=1; nodes[0].prnNd=-1;

do nr=nxtProd(nodes,nr,nc);

while(++nc<nn);

// конверсія до семантичної обробки

prLxTxt(nodes+nr);

printf("\n");

nr=prCmpr(nodes,nn,nr);// компресія для скорочення графа

SmAnlzr(nodes+nr,nr);

prLxTxt(nodes+nr);

printf("\n");

// int n=0, ai[5]={7,6,5,4,3};

// ai[++n]=ai[n+=2];

PrCdGen(nodes+nr,"..\\spLb9c\\spLb9c.cpp");//file\_name);

SmCdGen(nodes+nr,1);

EpCdGen(nodes+nr);

return 0;

}

#include "stdafx.h"

#include "..\spLb3\token.h"

#include "..\spLb4\automat.h"

#include "..\spLb5\syntaxP.h"

#include "lexcalc.h"

#include <string.h>

enum ltrTypeS ltrClsV[256] = // Початок таблиці класифікаторів для Verilog HDL

{ncS,ncS,ncS,ncS,ncS,ncS,ncS,ncS, ncS,dlmauxS,dlmauxS,ncS,ncS,dlmauxS,ncS,ncS, //16

ncS,ncS,ncS,ncS,ncS,ncS,ncS,ncS, ncS,ncS,ncS,ncS,ncS,ncS,ncS,ncS, //32

dlmauxS,dlmgrop1,ltrstrlmS,dlmunopS, ltrnmelmS,dlmunopS,dlmgrop2,ltrcnslmS,

dlmunopS,dlmunopS,dlmgrop3,ltrsignS, dlmunopS,ltrsignm,dldotS,ltrstrlmS, //48

d0,d1,d2,d3,d4,d5,d6,d7,

d8,d9,dlmunopS/\*dlmgrop\*/,dlmunopS/\*dlmbrlst\*/,dlmgrop4,dlmgrop5,dlmgrop6,dlmunopS,// 64

dlmunopS,xA,xB,xC, xD,ltrexpltS,xF,ltrnmelmS,

cH,ltrnmelmS,ltrnmelmS,ltrnmelmS, ltrnmelmS,ltrnmelmS,ltrnmelmS,cO,//80

ltrnmelmS,ltrnmelmS,ltrnmelmS,ltrnmelmS, ltrnmelmS,ltrnmelmS,ltrnmelmS,ltrnmelmS,

ltrhxdgtS/\*ltrnmelm\*/,ltrnmelmS,xZ/\*ltrnmelm\*/,dlmunopS, ltrtrnfmS,dlmunopS,dlmgrop7,dlmund,//96

dlmunopS,xA,xB,xC, xD,ltrexpltS,xF,ltrnmelmS,

cH,ltrnmelmS,ltrnmelmS,ltrnmelmS, ltrnmelmS,ltrnmelmS,ltrnmelmS,cO,//112

ltrnmelmS,ltrnmelmS,ltrnmelmS,ltrnmelmS, ltrnmelmS,ltrnmelmS,ltrnmelmS,ltrnmelmS,

ltrhxdgtS/\*ltrnmelm\*/,ltrnmelmS,xZ/\*ltrnmelm\*/,dlmunopS, dlmgrop8,dlmunopS,dlmgrop9,ncS//,128

//...

};extern struct imElem kwTab[LENGTHKW+1]; // для Verilog HDL

unsigned bcnst32\_buf[MAX\_UCNST];//={0,0,-1,-1,0,1,0};для Verilog HDL

// \*bcnst32\_beg, \*bcnst32\_cur;

//extern struct imElem dlmTb23V[10][4];

char bcnst8\_buf[MAX\_CCNST];//={0,1,2,3};//, // buffer of optimized bit constant

int //x=0, y=0, f=0, // координати розміщення тексту в файлі

nImBg=0, nImCr=0,// Початковий та поточний номери образів в буфері

nInBg=7, nInCr=7,// Початковий та поточний номери 32-бітових образів в буфері

nIcBg=4, nIcCr=4,// Початковий та поточний номери 32-бітових образів в буфері

fPrdfCnstr=0, // Номер напередвизначеної конструкції

fLnkCnstr=0, //

brkCntr=0, brkStck[BRK\_LIM];

extern char imgBuf/\*imageBuf\*/[MAX\_SCNST]; // буфер вхідних образів

//=============================================================

union ui64\_t value, valuea, value1, value1a, value0;//, \*valptr;

//long // можлива занадто велика точність

double fvalue64, fvalueR;//, \*fvlptr;

/\* у Visual C/C++ 6.0 без сервіс-паків не працює

struct up80\_t{unsigned \_\_int64 mant;

unsigned short pow;} c2p32={0x8000000000000000l,0x401f};

union ui80\_t{long double lfvalue;

struct up80\_t up80;} c2pn32, fvalue;\*/

unsigned \_\_int64 c2p32=0x41f0000000000000l;

union ui80\_t{double lfvalue;

\_\_int64 up80;} c2pn32, fvalue;

unsigned char aTb[4]={2,10,8,16};

int nPwr=0, n32=0;

unsigned FMode=0;//nBit=0,

int /\*LexNode::\*/ convNum(struct lxNode \*nd //, вказівник на початок масиву вузлів

// unsigned n, int disp, enum ltrTypeS ltrCls[256]

)

{unsigned char aSS, sSS, c=0xff;

int i, dispP, disp=(char\*)(nd->prvNd)-imgBuf;

nPwr=0; n32=0; FMode=0;

// ndOp = n;;

nd->pstNd=(struct lxNode \*)(bcnst32\_buf+nInBg);

if(nd->resLength!=S2s)disp=convInt(disp, 10, 10, 0xFFFFFFFF);

switch(nd->resLength)

{case S1c: nd->resLength=32; break;

case S1p: case S2c: nd->resLength=64;

// if(n32<(((ln)/32)<<1))n32=((ln)/32)<<1;

if((FMode&FPWR)!=0||ltrClsV[imgBuf[disp]]==ltrexplt)

{value.s64.lo32=bcnst32\_buf[nInBg];

// if(n32==0)

value.s64.hi32=0;

// else value.s64.hi32=bcnst32\_buf[nInBg+2];

fvalueR/\*.lfvalue\*/=(\_\_int64)value.ui64;

c2pn32.up80 = c2p32;

fvalue64=1.0;

// fvalue.up80.mant=value.ui64;

// fvalue.up80.pow=0x403f;

for(i=0;i<n32;i+=2)

{fvalue64\*=c2pn32.lfvalue;

value.s64.lo32=bcnst32\_buf[nInBg+i+2];

fvalueR+=((\_\_int64)value.ui64)\*fvalue64;

}

fvalue.lfvalue=fvalueR;

if(ltrClsV[imgBuf[disp]]==ltrexpltS){FMode&=~FPWR;

dispP=disp+1; if((ltrClsV[imgBuf[dispP]]&0xf0)==ltrsignS)disp=disp+1;

disp=convInt(disp+1, 10, 10, 0);

if(value.s64.lo32){

if(imgBuf[dispP]!='-')nPwr+=value.s64.lo32;

else nPwr-=value.s64.lo32;}}

if(FMode&FMIN)fvalue.lfvalue=-fvalue.lfvalue;

FMode&=~FMIN;

if(nPwr>0)while(nPwr){nPwr--; fvalue.lfvalue\*=1e1l;/\*/=1e-1l\*/}

else while(nPwr){nPwr++; fvalue.lfvalue/=1e1l;/\*=1e-1l\*/}

value.fi64=fvalue.lfvalue;

bcnst32\_buf[nInBg+0]=value.s64.lo32;

bcnst32\_buf[nInBg+1]=value.s64.hi32;

n32=0;

}

break;

case S2s: strcpy(bcnst8\_buf+nIcBg,imgBuf+nImBg+1);

nd->pstNd=(struct lxNode \*)(bcnst8\_buf+nIcBg);

nIcBg=nIcCr=nIcBg+strlen(imgBuf+nImBg+1);

\*(bcnst8\_buf+nIcCr-1)=0; // вилучити повторення

return 0;

break;

case S3c:

nd->resLength=value.s64.lo32;

if(nd->resLength==0xFFFFFFFF)nd->resLength=32;

sSS=(ltrClsV[imgBuf[disp+1]]&14)>>1; if(sSS>4)sSS-=4;

aSS=aTb[sSS-1];

convInt(disp+2, aSS, sSS, nd->resLength);

}

nd->stkLength=(n32+2)<<4; nInCr=nInBg-n32-2;

do{if(nInCr<0)break; c=0==0;

for(i=n32+1;i>=0;i--)

if(!(c&=bcnst32\_buf[nInBg+i]==bcnst32\_buf[nInCr+i]))break;

nInCr--;} while(c==0);

if(c==0){nInCr=nInBg+=n32+2;

nImBg=nImCr;}

else {nd->pstNd=(struct lxNode \*)(bcnst32\_buf+nInCr+1); nInCr=nInBg;

nImBg=nImCr;}// це треба б мінімізувати, якщо вхідні коди співпадають

return n32;

}

//------------------------------------------------------------------------

/\* converting source program to lexeme string "wstr" \*/

int convInt(unsigned nc, unsigned char aSS, unsigned char sSS, unsigned nb)

{//try to skip space after constant base like 'b, 'd, 'h

unsigned char cwrk; int i;

unsigned difV, difVM, nBit=0;//, difV, difVM;

value.ui64=value1.ui64=0; //nPwr=0;

if(nb==0xFFFFFFFF)value.s64.lo32=nb;

n32=0; bcnst32\_buf[nInBg+1]=bcnst32\_buf[nInBg]=0;

/\* if(lwstr<=2||wstr[lwstr].code!=div\_diez

||(wstr[lwstr-1].code!=div\_equ&&wstr[lwstr-1].code!=kws\_leeq))

while (c==32||c==9) Read\_symbol();

if (c=='`')

{scan\_identifier();buffer[0]=0;}\*/

while(((cwrk=ltrClsV[imgBuf/\*imageBuf\*/[nc]])&0xcf)<aSS||imgBuf[nc]=='.'

// ||c=='\_'||c=='.'||c=='?'

// ||c=='X'||c=='x'||c=='Z'||c=='z'// x=0, z=-1

)

{if(imgBuf[nc]!='.')

{if(aSS==10)

{difV=(cwrk&0xf);difVM=0;

if((cwrk&0x20)!=0)difVM = 15&(aSS-1);

if(FMode&FPWR)nPwr--;

for(i=0;i<=n32;i+=2)

{value.s64.lo32=bcnst32\_buf[i+nInBg]; value.s64.hi32=0;

value1.s64.lo32=bcnst32\_buf[i+nInBg+1]; value1.s64.hi32=0;

value.ui64 = value.ui64 \* aSS + difV;

value1.ui64 = (value1.ui64 \* sSS) | difVM;

bcnst32\_buf[i+nInBg]=value.s64.lo32;

bcnst32\_buf[i+nInBg+1]=value1.s64.lo32;

difV=value.s64.hi32;

difVM=value1.s64.hi32;

}

if(difV|difVM){n32++; n32++;

bcnst32\_buf[i+nInBg]=difV;bcnst32\_buf[i+nInBg+1]=difVM;}

}else

{difV = cwrk&(aSS-1); if(cwrk==xZ/\*ltrhxdgt\*/)difV = aSS-1;

difVM=0; nBit+=sSS;

if((cwrk&0xfe/\*20\*/)==ltrhxdgt/\*0\*/)difVM = 15&(aSS-1);

for(i=0;i<=n32;i+=2)

{value.s64.lo32=bcnst32\_buf[i+nInBg]; value.s64.hi32=0;

value1.s64.lo32=bcnst32\_buf[i+nInBg+1]; value1.s64.hi32=0;

value.ui64 = (value.ui64 << sSS) + difV;

value1.ui64 = (value1.ui64 << sSS) | difVM;

bcnst32\_buf[i+nInBg]=value.s64.lo32;

bcnst32\_buf[i+nInBg+1]=value1.s64.lo32;

difV=value.s64.hi32;

difVM=value1.s64.hi32;

}

if(difV|difVM){n32++; n32++;

bcnst32\_buf[i+nInBg]=difV; bcnst32\_buf[i+nInBg+1]=difVM;}

}}else if(imgBuf[nc]=='.')

{FMode|=FPWR;} // треба захиститись вiд повторної крапки

// if (strlen(imageBuf+nc)<=1025)strncat(imageBuf+nc,nc/\*&c\*/,1);

//Read\_symbol();

nc++;}

if(nb/\*ulength.ui64\*/!=0xFFFFFFFF){//ulength.s64.lo32=32;

valuea.ui64=1; valuea.ui64<<=((nBit-1)&63);

if (aSS!=10&&nBit</\*ulength.s64.lo32\*/nb&&(valuea.ui64&value1.ui64)!=0)

{value1a.ui64=1; value1a.ui64<<=(nb/\*ulength.s64.lo32\*/);//&63

value1a.ui64=value1a.ui64-valuea.ui64;

bcnst32\_buf[i+nInBg-1]|=value1a.ui64;

if(value1a.s64.hi32){n32++; n32++; bcnst32\_buf[i+nInBg+1]=value1a.s64.hi32;

bcnst32\_buf[i+nInBg]=0;

if((valuea.ui64&value.ui64)!=0)bcnst32\_buf[i+nInBg]|=value1a.s64.hi32;}

if((valuea.ui64&value.ui64)!=0)bcnst32\_buf[i+nInBg-2]|=value1a.ui64;

}else if(nBit>nb/\*ulength.s64.lo32\*/&&aSS!=10)

{bcnst32\_buf[i+nInBg-1]&=valuea.s64.lo32-1;

bcnst32\_buf[i+nInBg-2]&=valuea.s64.lo32-1;

}else

if(i=((nb/\*ulength.s64.lo32\*/+31)>>5)-1-n32/2)

while(i-->0)

{n32+=2; bcnst32\_buf[n32+nInBg]=0;

bcnst32\_buf[n32+nInBg+1]=0;}//?????

}

return nc;}

**Приклад виконання програми:**

// spLb9c.cpp: Визначає вхідну точку для консольної прикладної програми.

//

#include "stdafx.h"

int main(int argc, char\* argv[])

{double b,a[4]={2,3,4,5};

unsigned n=2,d=5;

\_asm{

mov eax,d

cmp eax,0

setne al

movsx eax,al

push eax

fstp b

add esp,4

fild n

mov esi,n

fld a[esi\*8]

fmul

fstp b}

return 0;

}

**Висновок**

В ході виконання лабораторної роботи були отримані навички побудови програм автоматичної генерації програмних модулів об’єктних кодів для компіляторів з використанням угод базової мови програмування.