COMPILER DESIGN LAB

MASEERA ALI
13 BCS 0032
BTECH COMPUTER ENGG.
2013 – 2017

INDEX

5. Program to find leaders and basic blocks of a TAC

```
#include<stdio.h>
#include<stdlib.h>
int main(){
int m,i,j,p,q,num=0,temp,no=1;
int led[25],count=0;
FILE *fp;
char arr[100][25],t;
fp=fopen("file.txt","r");
if(fp=='\0'){
printf("opening error:\n");
}
i=0;j=0;
led[no++]=1;
while(fscanf(fp,"%c",&t)!=EOF){
if(t=='\n'){
arr[i][j]='\0';
j=0;
i++; }
else {
arr[i][j]=t;
j++; }
}//end of while
arr[i][j]='\0';
for(p=0;p<i;p++) {
for(q=0;arr[p][q]!='\0';q++) {
printf("%c",arr[p][q]);
```

```
}
printf("\n");
}
for(p=0;p<i;p++){
for(q=0;arr[p][q]!='\0';q++){
if(arr[p][q]=='g' && arr[p][q+1]=='o' && arr[p][q+2]=='t' && arr[p][q+3]=='o'){
q=q+5;
while(arr[p][q]!='\0'){
temp=arr[p][q]-48;
num=num+temp;
if(arr[p][q+1]=='\0')
break;
else{
num=num*10;
q++;
}}
led[no++]=num;
led[no++]=p+2;
num=0;
}}}
for(p=1;p<no-1;p++){
for(q=p+1;q<no;q++){
if(led[p]>led[q]){
temp=led[p];
led[p]=led[q];
led[q]=temp;
}}}
```

```
for(p=1;p<no-1;p++){
for(q=p+1;q<no;q++){
if(led[p]==led[q]){
for(m=q;m<no-1;m++)
led[m]=led[m+1];
no--;
}}}
printf("LEADER\n");
for(p=1;p<no;p++)
printf("\t%d",led[p]);
q=1;
for(p=0;p<i;p++){
printf("\n\t: BLOCK%d:\n",count);
for(m=(led[q]-1);m<=(led[q+1]-2);m++){
printf("\n\t\t\t%s",arr[m]);
p++; }
q=q+1;
count++;
} }
```

6. Program to find natural loops in a TAC

```
#include<iostream>
#include<string.h>
#include<fstream>
using namespace std;
int stack[20],top=-1,llen=0,loop[20],cfg[20][20],size=0;
void read(){
    ifstream fin;
```

```
    fin.open("nlooptxt.txt",ios::in);
    char ch;
    int i,j;
    size=-1;
    while(fin.get(ch)) {
        if(ch=='\n'){
            if(size==-1)
                size=j;
            i++;
            j=0;}
        else
        if(ch!=' ')
         cfg[i][j++]=ch-'0';
    }cout<<size<<"\n";
}
void push(int m){
    stack[++top]=m;}
int pop(){
    int t=stack[top--];
    return t;}
void insert(int m){
    int i,flag=0;
    for(i=0;i<llen;i++) {
        if(m==loop[i]){
            flag=1;
            break;
        } }
```

```
    if(!flag) {
        loop[llen++]=m;
        push(m);
    }}
void nloop(int n,int d){
    int i,j;
    top=-1;
    llen=0;
    loop[llen++]=d;
    insert(n);
    while(top!=-1){
        int item=pop();
        //cout<<item<<"\t";
        for(j=0;j<size;j++){
            if(cfg[j][item]==1 && j>d){
                //cout<<j<<"\t";
                insert(j);
            } }}
    cout<<"\nNatural loop for "<<n+1<<" - "<<d+1<<": ";
    for(j=0;j<llen;j++){
        cout<<loop[j]+1<<"\t";
    }}
void naloop(){
    int i,j;
    for(i=0;i<size;i++) {
        for(j=0;j<size;j++){
            if(cfg[i][j]==1 && i>j)
```

```
                nloop(i,j);
        }}}
int main(){
    read();
    naloop();
}
```

7.Progarm to find IN and OUT value for each block in TAC

```
#include<iostream>
#include<fstream>
using namespace std;
int line[30],llen=0,len,leaders[20],blocks[20][2],gen[20][50],kill[20][50],cfg[20][20],in[20][50],out[20][50];
char tac[50][50];
void read(){
    ifstream fin;
    fin.open("inouttxt.txt",ios::in);
    char ch;
    int num=0;
    llen=0;
    fin.get(ch);
    int i=0,j=0,flag=0;
    while(ch!='\n'){
        if(ch==' '){
            leaders[llen++]=num;
            num=0;   }
        else
            num=num*10+(ch-'0');
```

```
        fin.get(ch);    }
leaders[llen++]=num;
i=0;
j=0;
fin.get(ch);
while(1){
        if(ch=='\n'){
                i++;
                j=0;            }
        else
        if(ch!=' ')
                cfg[i][j++]=ch-'0';
        if(i==llen)
                break;
        fin.get(ch);    }
cout<<llen;
i=j=0;
num=0;
cout<<ch;
fflush(stdin);
flag=0;
while(fin.get(ch))  {
        cout<<ch;
        if(ch=='\n'){
                flag=0;
                tac[i][j]='\0';
                j=0;
```

```
            i++;   }
        else
        if(ch==':') {
            flag=1;
            j=0;
            line[i]=num;
            num=0;   }
        else
        if(!flag) {
            num=num*10+(ch-'0');   }
        else{
            tac[i][j++]=ch;      }
    }
    cout<<line[0];
    tac[i][j]='\0';
    len=i+1; }
int findindex(int num){
    for(int i=0;i<len;i++){
        if(num==line[i])
            return i;}
    return -1; }
void mkblocks(){
    int i,j;
    cout<<line[0]<<"\t1";
    for(i=0;i<llen;i++){
        blocks[i][0]=findindex(leaders[i]);
        if((i+1)!=llen)
```

```
            blocks[i][1]=findindex(leaders[i+1]-1);
        else
            blocks[i][1]=len-1;   }
    for(i=0;i<llen;i++)
        cout<<"\nBlock: "<<blocks[i][0]<<"- "<<blocks[i][1];
}
int find(char *str){
    int i=0;
    while(*(str+i)!='\0'){
        if(*(str+i)=='=')
            return 1;
        i++; }
    return 0; }
void gen_gen(){
    int i,j;
    for(i=0;i<llen;i++){
        for(j=0;j<50;j++)
            gen[i][j]=0;}
    for(i=0;i<llen;i++){
        for(j=0;j<=len;j++){
            if(j>=blocks[i][0] && j<=blocks[i][1]){
                int flag=find(tac[j]);
                if(flag)
                    gen[i][j]=1;   } } }
    cout<<"\nGen Matrix is: \n";
    for(i=0;i<llen;i++){
        cout<<"\n";
```

```
        for(j=0;j<len;j++)
            cout<<gen[i][j]<<"\t";
    } }
void gen_kill(){
    int i,j,k,m,flag=0;
    for(i=0;i<llen;i++){
        for(j=0;j<50;j++)
            kill[i][j]=0; }
    for(i=0;i<llen;i++){
        for(k=blocks[i][0];k<=blocks[i][1];k++){
            for(j=0;j<len;j++){
                if(j<blocks[i][0] || j>blocks[i][1]){
                    int flag1=find(tac[j]);
                    int flag2=find(tac[k]);
                    if(flag1 && flag2){
                        flag=1;
                    for(m=0;tac[j][m]!='=' && tac[k][m]!='=';m++){
                            if(tac[j][m]!=tac[k][m])          {
                                flag=0;
                                break; }    }
                    if(tac[j][m]=='=' && tac[k][m]=='=' && flag)
                            kill[i][j]=1;
     } } } } }
    cout<<"\nKill: \n";
    for(i=0;i<llen;i++){
        cout<<"\n";
        for(j=0;j<len;j++)
```

```
            cout<<kill[i][j]<<"\t";
    }}
void dounion(int i,int j){
    int m,n;
    for(m=0;m<len;m++){
        in[i][m]=in[i][m]|out[j][m];
    } }
void calcout(int index){
    int i,j;
    int temp[50];
    for(j=0;j<len;j++){
        if(in[index][j])
            temp[j]=in[index][j]-kill[index][j];
        else
            temp[j]=0;
    }
    for(j=0;j<len;j++)
        out[index][j]=gen[index][j]|temp[j];
}
void inout(){
    int i,j,k;
    for(i=0;i<llen;i++){
        for(j=0;j<len;j++){
            in[i][j]=0;
            out[i][j]=gen[i][j];
        } }
    int flag=1;
```

```
    while(flag) {
        flag=0;
        for(i=0;i<llen;i++){
            for(j=0;j<llen;j++) {
                if(cfg[j][i]==1){
                    //cout<<i<<" "<<j<<"\t";
                    dounion(i,j);
 } }
            int oldout[50];
            for(j=0;j<len;j++)
                oldout[j]=out[i][j];
            calcout(i);
            for(j=0;j<len;j++){
                if(oldout[j]!=out[i][j])
                    flag=1;
            }
}}
    cout<<"\nINOUT\n\n";
    for(i=0;i<llen;i++){
        cout<<"\n";
        for(j=0;j<len;j++)
            cout<<in[i][j]<<"\t";}
    cout<<"\nOut\n";
    for(i=0;i<llen;i++){
        cout<<"\n";
        for(j=0;j<len;j++)
            cout<<out[i][j]<<"\t";}
```

```
        cout<<"\n\n";
        for(i=0;i<llen;i++){
            cout<<"\n";
            for(j=0;j<llen;j++)
                cout<<cfg[i][j]<<"\t";}}
int main(){
        read();
        mkblocks();
        gen_gen();
        gen_kill();
        inout();
}
```