FRANCIS XAVIER ENGINEERING COLLEGE

TIRUNELVELI - 627003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS 2308 –SYSTEM SOFTWARE LAB MANUAL

Prepared By,

Mr. S. Malaiarasan, AP/CSE

Ms. K. Rajasundari, AP/CSE

CS2308 SYSTEM SOFTWARE LAB (Using C)

1. Implement a symbol table with functions to create, insert, modify, search, and display.
2. Implement pass one of a two pass assembler.
3. Implement pass two of a two pass assembler.
4. Implement a single pass assembler.
5. Implement of macro processor
6. Implement an absolute loader.
7. Implement a relocating loader.
8. Implement pass one of a direct-linking loader.
9. Implement pass two of a direct-linking loader.
10. Implement a simple text editor with features like insertion / deletion of a character, word, and sentence.
11. Implement a symbol table with suitable hashing

(For loader exercises, output the snap shot of the main memory as it would be, after the loading has taken place)

EX.NO.1 SYMBOL TABLE CREATION

AIM

To write a “C” program to implement symbol table with functions like create, insert, modify, search and display.

ALGORITHM

- Get the variables as token name, token type and token value.
- To display the contents of the symbol table call the function display () function.\
- To insert the value on to the symbol table get the token name, token type and token value of the variable and it is given by calling insert () function.
- To perform the modifying operation, for example the old token name, token type, token name and token value is modified with new token type, token name and token value. This operation is performed by calling the modify () function.
- The search operation is performed by calling the search () function. If the token name is present in the symbol table, then the “search operation is successful”.
- Otherwise display the result as the “Search operation is unsuccessful”.
- Display the token name, token type and token value in the symbol table after each operation is being performed.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
int i,m,n,flag=1;
struct symbol
{
        int tokenno;
        char tokenname[15];
        char tokentype[15];
        char tokenvalue[15];
}s[50];
void create()
{
        printf("enter the no. of token\n");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("enter the token no,token name,token type,token value\n");
                scanf("%d",&s[i].tokenno);
                scanf("%s%s%s",s[i].tokenname,s[i].tokentype,s[i].tokenvalue);
        }
}
void insert()
{
        int t;
        printf("enter the no. of token to be inserted\n");
        scanf("%d",&t);
        for(i=n;i<n+t;i++)
        {
                printf("enter the token no,token name,token type,token value\n");
                scanf("%d",&s[i].tokenno);
                scanf("%s%s%s",s[i].tokenname,s[i].tokentype,s[i].tokenvalue);
        }
        n=n+t;
}
void modify()
{
        flag=1;
        printf("enter the no. of token to be modified\n");
        scanf("%d",&m);
        for(i=0;i<n;i++)
```

```
        {
                if(s[i].tokenno==m)
                {
                        flag=0;
                        printf("enter the new values to be modified\n");
                        scanf("%d",&s[i].tokenno);
                        scanf("%s%s%s",s[i].tokenname,s[i].tokentype,s[i].tokenvalue);
                        break;
                }
        }
        if(flag!=0)
        printf("\nsymbol table entry not found");
}
void search()
{
        flag=1;
        printf("\nenter the no. of token to be searched\n");
        scanf("%d",&m);
        for(i=0;i<n;i++)
        {
                if(s[i].tokenno==m)
                {
                        flag=0;
                        printf("\nsymbol table entry found\n");
                        printf("token number\ttoken name\ttoken type\ttoken value\n");
                        printf("%d\t\t",s[i].tokenno);
                        printf("%s\t\t%s\t\t%s",s[i].tokenname,s[i].tokentype,s[i].tokenvalue);
                        break;
                }
        }
        if(flag!=0)
        printf("\nsymbol table entry not found");
}
void display()
{
        printf("symbol table\n");
        printf("token number\ttoken name\ttoken type\ttoken value\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t\t",s[i].tokenno);
                printf("%s\t\t%s\t\t",s[i].tokenname,s[i].tokentype);
                printf("%s\t",s[i].tokenvalue);
                printf("\n");
        }
}
void main()
{
        int c,ch;
        clrscr();
```

```
        do
        {
                printf("MENU\n");
                printf("1.create\n2.insert\n3.modify\n4.search\n5.display\n");
                printf("Enter Your Choice :");
                scanf("%d",&c);
                switch(c)
                {
                        case 1:
                                create();
                                break;
                        case 2:
                                insert();
                                break;
                        case 3:
                                modify();
                                break;
                        case 4:
                                search();
                                break;
                        case 5:
                                display();
                                break;
                }
                printf("\n Do you want to continue 0 or 1(0 for NO and 1 for YES)\n");
                scanf("%d",&ch);
        }while(ch==1);
        getch();
}
```

OUTPUT

```
MENU
1.create
2.insert
3.modify
4.search
5.display
enter the choice :1
enter the no. of token
1
enter the token no,token name,token type,token value
1
start
command
5
Do you want to continue 0 or 1(0 for NO and 1 for YES) 1
MENU
1.create
2.insert
3.modify
4.search
5.display
enter the choice :2
enter the no. of token to be inserted
1
enter the token no,token name,token type,token value
2
stop
command
4
Do you want to continue 0 or 1(0 for NO and 1 for YES) 1
1
MENU
1.create
2.insert
3.modify
4.search
5.display
enter the choice :5
symbol table
token number    token name      token type      token value
1               start           command         5
2               stop            command         4
Do you want to continue 0 or 1(0 for NO and 1 for YES) 1
MENU
1.create
2.insert
3.modify
4.search
```

5.display
enter the choice :3
enter the no. of token to be modified
1
enter the new values to be modified
1
begin
command
5
Do you want to continue 0 or 1(0 for NO and 1 for YES) 1
MENU
1.create
2.insert
3.modify
4.search
5.display
enter the choice :4
enter the no. of token to be searched
1
symbol table entry found

token number	token name	token type	token value
1	begin	command	5
2	stop	command	4

Do you want to continue 0 or 1(0 for NO and 1 for YES) 0

RESULT:

Thus the program to implement symbol table with functions like create, insert, modify, search and display was executed and the symbol table was created.

EX.NO.2	PASS 1 OF A 2 PASS ASSEMBLER

AIM

To write a “C” program to implement Pass 1 of a 2 Pass assembler.

ALGORITHM

- Create three structures called inst, syntax and optab
- The file “Symbol.txt” is created and it is opened in the read mode.
- Check opcode = START, if so convert character form and assign it to LOCCTR, if not then it is equal to 0.
- If opcode = WORD, then add 3 to LOCCTR, if opcode = RESW, then add 3* to LOCCTR.
- If opcode is not equal to END, the search SYMTAB for labels. If labels are found then set the error flag. If not, convert it into character form and copy it in the SYMTAB
- If opcode = WORD, then add 3 to LOCCTR if opcode = RESW then add 3* to LOCCTR.
- If opcode = byte, then find the length of the constant in byte and add it to the LOCCTR, if not add 3 to the LOCCTR. Repeat the step 4 and step 5 until END statement is reached. Display the opcode and assembler directives in the OPTAB

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct inst
{
        char label[10];
        char opcode[10];
        char operand[10];
}inst;
struct symtab
{
        char name[10];
        char address[10];
        int length;
}symtab[10];
struct optab
{
        char opcode[10];
        char numericalvalue[10];
}
op[]={{"LDA","OO"},{"STA","0C"},{"ADD","18"}};
void main()
{
        int s,i,j,n,h,ns=0,x,l,k,y,f;
        char a[20];
        FILE *fp;
        FILE *st;
        clrscr();
        fp=fopen("symbol.txt","r");
        printf("\n***SYMBOL TABLE***\n");
        printf("-----------------\n");
        printf("NAME ADDRESS LENGTH\n");
        fscanf(fp,"%s%s%s",inst.label,inst.opcode,inst.operand);
        if(strcmp(inst.opcode,"START")==0)
        {
                n=atoi(inst.operand);
                l=n;
                fscanf(fp,"%s%s%s",inst.label,inst.opcode,inst.operand);
        }
        else
                l=0;
        while(strcmp(inst.opcode,"END")!=0)
        {
                if(strcmp(inst.label,"-")!=0)
                {
                        for(i=0;i<ns;i++)
```

```
                {
                        if(strcmp(symtab[i].name,inst.label)==0)
                        {
                                f=0;
                                break;
                        }
                }
                if(f==0)
                {
                        f++;
                }
                else
                {
                        strcpy(symtab[ns].name,inst.label);
                        itoa(l,a,10);
                        strcpy(symtab[ns].address,a);
                        symtab[ns].length=3;
                        ns++;
                }
        }
        if(strcmp(inst.opcode,"WORD")==0)
        {
                l=3+l;
                symtab[ns-1].length=(3*x);
        }
        else if(strcmp(inst.opcode,"RESW")==0)
        {
                x=atoi(inst.operand);
                l=l+(3*x);
                symtab[ns-1].length=(3*x);
        }
        else if(strcmp(inst.opcode,"BYTE")==0)
        {
                k=(strlen(inst.opcode)-3);
                l=l+k;
                symtab[ns-1].length=k;
        }
        else
                l=l+3;
        fscanf(fp,"%s%s%s\n",inst.label,inst.opcode,inst.operand);
        s=l-n;
    }
        for(i=0;i<ns;i++)
        {
                printf("\n%s\t%s\t%d\n",symtab[i].name,symtab[i].address,symtab[i].length);
        }
printf("_____\n");
printf("\t\nLength of the program is : \n%d\n\n",s);
printf("\n\n Operation code table\n\n");
```

```
printf("\nmnemonic code \t opcode\n");
printf("\n_____");
fclose(fp);
st=fopen("symbol.txt","r");
fscanf(st,"%s%s%s",inst.label,inst.opcode,inst.operand);
while(strcmp(inst.opcode,"END")!=0)
{
        for(j=0;j<=2;j++)
        {
                if(strcmp(inst.opcode,op[j].opcode)==0)
                {
                        printf("\n\t%s\t%s\n",op[j].opcode,op[j].numericalvalue);
                }
        }
        fscanf(fp,"%s%s%s",inst.label,inst.opcode,inst.operand);
        }
        printf("\n_____THE END_____");
        fclose(st);
        getch();
}
```

INPUT & OUTPUT:

INPUT

Symbol.txt

```
-       START       1000
FIRST LDA   ALPHA
-       ADD   BETA
-       STA   GAMMA
ALPHA       RESW 1
BETA  WORD        8
GAMMA       RESB  3
-       END   FIRST
```

OUTPUT:

```
***SYMBOL TABLE***
----------------------
NAME ADDRESS LENGTH

FIRST   1000    3

ALPHA   1009    3

BETA    1012    3

GAMMA   1015    3
_____________________________

Length of the program is :
18

 Operation code table

mnemonic code   opcode
_____________________
      LDA    OO

      ADD    18

      STA    0C

_________THE END_________
```


RESULT

Thus the program to implement Pass 1 of a 2 Pass assembler was executed and output is verified.

EX.NO.3 PASS 2 OF A 2 PASS ASSEMBLER

AIM

To write a “C” program to implement Pass 2 of a 2 Pass assembler

ALGORITHM

- Create the structure “stc”, declare and initialize all the variables
- Open a file “source.txt” in read mode.
- If (location counter) lc%3 = 2 and source code not equal to end the open a file “opcode.txt” in the read mode.
- Compare “source.txt” and “opcode.txt” and do the process until it reaches the end of the file.
- If lc%3 = 0 and source code not equal to end then open “passone.txt” in read mode.
- Increment the location counter (lc) and repeat the above 4 steps until it reaches the end of file.
- Then close all files and display as “PASS 2 COMPLETED”

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct ste
{
        char a[100];
}s;
void main()
{
        char d[30];
        FILE *fp1,*fp2,*fp3;
        int n=5,lc=1,g=5,h=5,s1,flag;
        clrscr();
        printf("\n\t\tASSEMBLER CODE\n");
        printf("\n\tMACHINE CODE");
        printf("\tMNEMONIC OPCODE\n");
        gotoxy(7,6);
        fp1=fopen("source.txt","r");
        do
        {
                s1=0;
                fscanf(fp1,"%s",s.a);
                gotoxy(15,10);
                if((lc%3==2)&&(strcmp(s.a,"END")!=0))
                {
                        fp3=fopen("opcode.txt","r");
                        printf("%s",s.a);
                        do
                        {
                                fscanf(fp3,"%s",d);
                                if(strcmp(d,s.a)==0)
                                {
                                        fscanf(fp3,"%s",d);
                                        gotoxy(10,h++);
                                        printf("%s",d);
                                }
                        }
                while(!feof(fp3));
        }
        else
        {
                if((lc%3==0)&&strcmp(s.a,"END")!=0)
                {
                        gotoxy(30,g++);
                        fp2=fopen("passone.txt","r");
                        printf("%s",s.a);
                        do
```

```
                {
                        fscanf(fp2,"%s",d);
                        if(strcmp(d,s.a)==0)
                        {
                                fscanf(fp2,"%s",d);
                                gotoxy(12,n);
                                if(s1==0)
                                        printf("%s",d);
                                s1=1;
                        }
                }while(!feof(fp2));
                gotoxy(12,n);
                if(s1==0)
                        printf("%s",s.a);
                n++;
        }
    }
        lc++;
    }
    while(!feof(fp1));
    fclose(fp1);
    fclose(fp2);
    fclose(fp3);
    printf("\n\tPASS 2 COMPLETED");
    getch();
}
```

INPUT & OUTPUT

INPUT

source.txt

```
-       START   2000
CLOOP   JSUB    READ
READ    J       CLOOP
-       JET     BUFFER
BUFFER  END     CLOOP
```

opcode.txt

```
START   20
JSUB    28
J       3C
JET     35
END
```

passone.txt

```
CLOOP   2003
READ    2006
BUFFER  200C
```

OUTPUT

ASSEMBLER CODE

```
MACHINE CODE    MNEMONIC OPCODE
 202000         2000
 282006         READ
 3C2003         CLOOP
 35200C         BUFFER
        2003            CLOOP
```

PASS 2 COMPLETED

RESULT:

Thus the program to implement Pass 2 of a 2 Pass assembler was executed and the output was verified

EX.NO.4 SINGLE PASS ASSEMBLER

AIM

To write a “C” program to implement the single pass assembler.

ALGORITHM

- Initialize all the variables.
- Open the file “assemble.txt” in read mode.
- Read the contents of the file one by one.
- Compare the opcode of the file using string compare. If START then write it in a new file “object.txt” which is in write append mode.
- While opcode = END, check for the next opcode if opcode = RESW and opcode = RESB, write it in the file “object.txt”.
- If opcode = END then close the file and display the content “ONE PASS HAS BEEN COMPLETED SUCCESSFULLY”.
- The output of the file will be in the “object.txt” file.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
        FILE *fp,*pf;
        struct instruction
        {
                char label[10];
                char opcode[10];
                char operand[10];
        }il;
        struct symtab
        {
                char name[10];
                char address[10];
                int length;
        }st[5]={{"FIRST","1000",3},{"TWO","1003",3},{"RESULT","1006",3},{"TEMP","1009",3}};
        struct optab
        {
                char opr[10];
                char mc[10];
        }o[5]={{"ADD","18"},{"LDA","OO"},{"STA","0C"}};
        struct object
        {
                int address;
                char objcode[20];
        }obj;
        int locctr=0,i,j,n0,n,length;
        clrscr();
        fp=fopen("assemble.txt","r");
        pf=fopen("object.txt","w+");
        fscanf(fp,"%s%s%s\n",il.label,il.opcode,il.operand);
        if(strcmp(il.opcode,"START")==0)
        {
                locctr=atoi(il.operand);
                fscanf(fp,"%s%s%s\n",il.label,il.opcode,il.operand);
        }
        else
                locctr=0;
        while(strcmp(il.opcode,"END")!=0)
        {
                if(strcmp(il.opcode,"WORD")==0)
                {
                        strcpy(obj.objcode,"0000");
```

```
            strcat(obj.objcode,il.operand);
            length=3;
            locctr+=length;
    }
    else if(strcmp(il.opcode,"RESW")==0)
    {
            strcpy(obj.objcode,"-");
            length=3*atoi(il.operand);
            locctr+=length;
    }
    else if(strcmp(il.opcode,"RESB")==0)
    {
            strcpy(obj.objcode,"");
            length=atoi(il.operand);
            locctr+=length;
    }
    else if(strcmp(il.opcode,"BYTE")==0)
    {
            strcpy(obj.objcode,"454f46");
            length=n-3;
            locctr+=length;
    }
    else
    {
            length=3;
            locctr+=length;
            for(i=0;i<5;i++)
            {
                    if(strcmp(il.opcode,o[i].opr)==0)
                    {
                            n0=0;
                            for(j=0;j<5;j++)
                            {
                                    if(strcmp(il.operand,st[j].name)==0)
                                    {
                                            n0++;
                                            strcpy(obj.objcode,o[i].mc);
                                            strcat(obj.objcode,st[j].address);
                                            break;
                                    }
                            }
                            if(n0==0)
                            {
                                    printf("THERE IS AN UNDEFINED SYMBOL");
                                    exit(1);
                            }
                    }
            }
    }
```

```
            obj.address=locctr-length;
            fprintf(pf,"%d\t%s\n",obj.address,obj.objcode);
            fscanf(fp,"%s%s%s",il.label,il.opcode,il.operand);
        }
        fclose(fp);
        fclose(pf);
        printf("SINGLE PASS HAS BEEN COMPLETED SUCCESSFULLY\n");
        getch();
}
```

INPUT & OUTPUT:

INPUT

Assemble.txt

```
-           START       1000
FIRST       WORD        5
TWO         WORD        8
RESULT      RESW        2
TEMP        BYTE        C"EOF"
-           LDA         FIRST
-           ADD         TWO
-           STA         RESULT
-           END         FIRST
```

OUTPUT

SINGLE PASS HAS BEEN COMPLETED SUCCESSFULLY

Object.txt

```
1000        00005
1003        00008
1006        -
1012        454f46
2833        OO1000
2836        181003
2839        0C1006
```


RESULT:

Thus the program to implement the single pass assembler was executed and output was verified.

EX.NO.5 **MACRO PROCESSOR**

AIM

To write a “C” program to implement a macro processor.

ALGORITHM

- Get the macro program in the file “macro.txt” which is in read mode.
- Store the macro name in the Name table.
- Find the macro invocation statement and store the parameters in the Argument table.
- Read the macro program (macro.txt) line by line until MEND statement is reached and store the same.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
struct macro
{
        char label[10];
        char opcode[10];
        char parameter[10];
}m;
struct table
{
        char name[10];
        char parameter[10];
}t;
void main()
{
        FILE *fp,*pf;
        clrscr();
        fp=fopen("macro.txt","r");
        fscanf(fp,"%s%s%s",m.label,m.opcode,m.parameter);
        if(strcmp(m.label,"MACRO")==0)
        strcpy(t.name,m.opcode);
        printf("\n MACRO NAME: \t%s\n",t.name);
        while(strcmp(m.opcode,"END")!=0)
        {
                fscanf(fp,"%s%s%s",m.label,m.opcode,m.parameter);
                if(strcmp(t.name,m.opcode)==0)
                {
                        strcpy(t.parameter,m.parameter);
                        break;
                }
        }
        printf("\nArgument Table: \t%s\n",t.parameter);
        printf("\nMACRO \tDEFINITION\n");
        fclose(fp);
        pf=fopen("macro.txt","r");
        fscanf(pf,"\n%s%s%s",m.label,m.opcode,m.parameter);
        while(strcmp(m.opcode,"MEND")!=0)
        {
                fscanf(pf,"%s%s%s",m.label,m.opcode,m.parameter);
                printf("%s\t%s\n",m.opcode,m.parameter);
        }
        getch();
}
```

INPUT & OUTPUT:

INPUT

Macro.txt

```
MACRO       INCR  &x,&y
-           LDA   &x
-           ADD   &x
-           STA   &y
-           MEND-
-           INCR  B,J
-           END   -
```

OUTPUT

MACRO NAME: INCR

Argument Table: B,J

MACRO DEFINITION

```
LDA   &x
ADD   &x
STA   &y
MEND  -
```


RESULT :

Thus the program to implement the macro processor was executed and output was verified.

EX.NO.6 ABSOLUTE LOADER

AIM

To write a “C” program to implement an absolute loader.

ALGORITHM

- Declare all the variables
- Open a file name “inp.txt” in the read mode.
- If the record is the HEADER record the display the program name, starting address and length of the program.
- Read the next record.
- If the record is the TEXT record, convert the object code in character form into hexadecimal representation.
- Repeat step 5 and step 6 until the END record is reached.
- When END record is reached, jump to the address specified in the END record and start execution from the beginning.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<ctype.h>
void main()
{
        char *record,*obj,*name;
        int addr,len,i;
        FILE *fp;
        clrscr();
        fp=fopen("inp.txt","r");
        fscanf(fp,"%s%s%d%d",record,name,&addr,&len);
        if(strcmp(record,"H")==0)
        {
                printf("Program Name : %s\n\n",name);
                printf("Starting address : %d\n\n",addr);
                printf("Length of the Program : %d\n",len);
        }
        while(!feof(fp))
        {
                fscanf(fp,"%s",record);
                if(strcmp(record,"E")==0)
                {
                        fscanf(fp,"%x",&addr);
                        printf("\nAddress of the first statement : %x",addr);
                        break;
                }
                else if(strcmp(record,"T")==0)
                {
                        fscanf(fp,"%x%x%s",&addr,&len,obj);
                        for(i=0;i<len;i+=2)
                        {
                                printf("\n0x%x\t",addr++);
                                printf("%c%c",obj[i],obj[i+1]);
                        }
                }
        }
        fclose(fp);
        getch();
}
```

INPUT & OUTPUT:

INPUT

Inp.txt

```
H       COPY 001000        000036
T       001000        12   001203001203001203
T       001013        12   001203001203001203
E       001000
```

OUTPUT

```
Program Name : COPY
Starting address : 1000
Length of the Program : 36
0x1000  00
0x1001  12
0x1002  03
0x1003  00
0x1004  12
0x1005  03
0x1006  00
0x1007  12
0x1008  03
0x1013  00
0x1014  12
0x1015  03
0x1016  00
0x1017  12
0x1018  03
0x1019  00
0x101a  12
0x101b  03
Address of the first statement : 1000
```


RESULT:

Thus the program to implement an absolute loader was executed and the output was verified.

EX.NO.7 RELOCATING LOADER

AIM

To write a “C” program for relocating loader.

ALGORITHM

- Start the program
- Declare all the variables and enter the relocation address.
- Open a file named “inp.txt” in the read mode.
- If the record type is HEADER record then display the program name, starting address and length of the program.
- If the record type is TEXT record read the object code in the text record until the END record is reached.
- Exit the program.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
        char rect[10],opcode[10],name[10];
        int addr,len,i;
        int relo;
        FILE *fp;
        clrscr();
        printf("\n Enter the relocation address:");
        scanf("%d",&relo);
        fp=fopen("inp.txt","r");
        fscanf(fp,"%s%s%d%x",rect,name,&addr,&len);
        if(strcmp(rect,"H")==0)
        {
                printf("\nProgram Name : %s",name);
                printf("\nStarting Address :%d",addr+relo);
        printf("\nLength : %x\n",len);
        }
        while(!feof(fp))
        {
                fscanf(fp,"%s",rect);
                {
                if(strcmp(rect,"E")==0)
                {
                fscanf(fp,"%d",&addr);
                printf("\n\nAddress of executable instance %d",addr+relo);
                break;
        }
        else if(strcmp(rect,"T")==0)
        {
                fscanf(fp,"%d%x%s\n",&addr,&len,&opcode);
                for(i=0;i<len;i+=2)
                {
                        printf("%d\t",addr+++relo);
                        printf("%c%c\n",opcode[i],opcode[i+1]);
                }
        }
        }
        }
        fclose(fp);
        getch();
}
```

INPUT

Inp.txt

```
H       COPY 001000        000036
T       001000        12   001203001203001203
T       001013        12   001203001203001203
E       001000
```

OUTPUT

```
Enter the relocation address:100
Program Name : COPY
Starting Address :1100
Length : 36
1100    00
1101    12
1102    03
1103    00
1104    12
1105    03
1106    00
1107    12
1108    03
1113    00
1114    12
1115    03
1116    00
1117    12
1118    03
1119    00
1120    12
1121    03
ADDRESS OF EXECUTABLE INSTANCE: 1100
```

RESULT:

Thus the program to implement relocating loader was executed and output was verified.

EX.NO.8 PASS 1 OF A DIRECT LINKING LOADER

AIM

To write a "C" program to implement Pass 1 of a Direct Linking loader.

ALGORITHM

- Start the program.
- Initialize all the variables.
- Open a file by name "proga.txt" in read mode. Read the contents of the file.
- In this "proga.txt", we have to make an external symbol table for the symbol LISTA and ENDA.
- Open a file by name "progb.txt" in read mode. Read the contents of the file.
- In this "progb.txt", we have to make an external symbol table for the symbol LISTB and ENDB.
- Open a file by name "progc.txt" in read mode. Read the contents of the file.
- In this "progc.txt", we have to make an external symbol table for the symbol LISTC and ENDC.
- If the symbol is in the external symbol, then set error flag otherwise put the symbol in the symbol table with its corresponding value.
- Find the length of the symbol by using length = enda – temp. Find the address of the symbol where the control section is indicated by using the address = temp + address + length.
- Find the length of the symbol by using length = endb – temp. Find the address of the symbol where the control section is indicated by using the address = temp + address + length.
- Find the length of the symbol by using length = endc – temp. Find the address of the symbol where the control section is indicated by using the address = temp + address + length.
- Display the symbol name, address of length of the symbols in the program.
- Terminate the program.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#include<string.h>
FILE *fp1,*fp2,*fp3;
int d=0,c=0,enda,endb,endc,n1,n2,n3,i,temp,y=0;
struct proga
{
        char loc[10],label[10],opcode[10],operand[30];
}p1[50];
struct progb
{
        char loc[10],label[10],opcode[10],operand[30];
}p2[50];
struct progc
{
        char loc[10],label[10],opcode[10],operand[30];
}p3[50];
struct output
{
        char consec[20],symbol[10];
        int address,length;
}s[10];
void main()
{
        int i=0,n1,n2,n3,enda,endb,endc,y,z;
        long int temp,temp1;
        fp1=fopen("progA.txt","r");
        fp2=fopen("progB.txt","r");
        fp3=fopen("progC.txt","r");
        while(!feof(fp1))
        {
                fscanf(fp1,"\n%s%s%s%s",p1[i].loc,p1[i].label,p1[i].opcode,p1[i].operand);
                if(strcmp(p1[i].opcode,"END")==0)
                {
                        enda=atoi(p1[i].loc);
                }
                i++;
        }
        n1=i;
        i=0;
        while(!feof(fp2))
        {
                fscanf(fp2,"\n%s%s%s%s",p2[i].loc,p2[i].label,p2[i].opcode,p2[i].operand);
                if(strcmp(p2[i].opcode,"END")==0)
                {
                        endb=atoi(p2[i].loc);
```

```
		}
		i++;
	}
		n2=i;
		i=0;
	while(!feof(fp3))
	{
		fscanf(fp3,"\n%s%s%s%s",p3[i].loc,p3[i].label,p3[i].opcode,p3[i].operand);
		if(strcmp(p3[i].opcode,"END")==0)
		{
			endc=atoi(p3[i].loc);
		}
		i++;
	}
	n3=i;
	i=0;
	y=0;
	strcpy(s[y].consec,p1[0].label);
	strcpy(s[y].symbol,"NULL");
	temp=atoi(p1[10].loc);
	s[y].address=temp;
	s[y].length=enda-temp;
	y++;
	for(i=0;i<n1;i++)
	{
		if(strcmp(p1[i].opcode,"EQU")==0)
		{
			strcpy(s[y].consec,"NULL");
			strcpy(s[y].symbol,p1[i].label);
			temp=atoi(p1[i].loc);
			s[y].address=temp;
			s[y].length=0;
			y++;
		}
	}
	strcpy(s[y].consec,p2[0].label);
	strcpy(s[y].symbol,"NULL");
	temp=atoi(p2[0].loc);
	s[y].address=temp+s[0].address+s[0].length;
	s[y].length=endb-temp;
	y++;
	for(i=0;i<n2;i++)
	{
		if(strcmp(p2[i].opcode,"EQU")==0)
		{
			strcpy(s[y].consec,"NULL");
			strcpy(s[y].symbol,p2[i].label);
			temp=atoi(p2[i].loc);
			s[y].address=temp+s[0].address+s[0].length;
```

```
                s[y].length=0;
                y++;
            }
        }
        strcpy(s[y].consec,p3[0].label);
        strcpy(s[y].symbol,"NULL");
        temp=atoi(p3[0].loc);
        s[y].address=temp+s[3].address+s[3].length;
        s[y].length=endc-temp;
        y++;
        for(i=0;i<n3;i++)
        {
            if(strcmp(p3[i].opcode,"EQU")==0)
            {
                strcpy(s[y].consec,"NULL");
                strcpy(s[y].symbol,p3[i].label);
                temp=atoi(p3[i].loc);
                s[y].address=temp+s[3].address+s[3].length;
                s[y].length=0;
                y++;
            }
        }
        clrscr();
        printf("CONTROL SECTION\tSYMBOL NAME ADDRESS\tLENGTH\n");
        printf("\n");
        for(i=0;i<y;i++)
        {
            if(strcmp(s[i].consec,"NULL")==0)
                printf("\n\t");
            else
                printf("\n\t%s\t",s[i].consec);
            if(strcmp(s[i].symbol,"NULL")==0)
                printf("\t\t");
            else
                printf("\t%s\t\t",s[i].symbol);
            printf("%d\t",s[i].address);
            if(s[i].length==0)
                printf("\n");
            else
                printf("%d\n",s[i].length);
        }
        getch();
}
```

INPUT & OUTPUT:

INPUT

proga.txt

```
0000    PROGA           START 0
_ _     EXTDEF      LISTA,ENDA
_ _     EXTREF      LISTB,ENDB,LISTC,ENDC
0040    LISTA       EQU *
0054    ENDA        EQU *
0090    _END        PROGA
```

progb.txt

```
0000    PROGB           START 0
_ _     EXTDEF      LISTB,ENDB
_ _     EXTREF      LISTA,ENDA,LISTC,ENDC
0060    LISTB       EQU *
0070    ENDB        EQU *
0100    _END        _
```

progc.txt

```
0000    PROGC           START 0
_ _     EXTDEF      LISTC,ENDC
_ _     EXTREF      LISTA,ENDA,LISTB,ENDB
0030    LISTC       EQU *
0042    ENDC        EQU *
0150    _END        _
```

OUTPUT

CONTROL SECTION	SYMBOL NAME	ADDRESS	LENGTH
PROGA		0	6475
	LISTA	40	
	ENDA	54	
PROGB		6475	64
	LISTB	6535	
	ENDB	6545	
PROGC		6539	1824
	LISTC	6569	
	ENDC	6581	

RESULT:

Thus the program to implement Pass 1 of a Direct Linking loader was executed and output was verified

EX.NO.9 PASS 2 OF A DIRECT LINKING LOADER

AIM

To write a "C" program to implement the Pass 2 of a Direct Linking Loader.

ALGORITHM

- Start the program.
- Get the external Symbol table as the input.
- Open the "objpgm.txt" in read mode and read the contents in that file.
- Repeat the next two steps until END record is reached.
- If the record type is "REFER RECORD", then get the address of the external symbol from the external symbol table and modify the same in the object program.
- Else if the record type is "MODIFICATION RECORD", get the address of the external symbol table and replace the symbol with the address value in the object program.
- The output of the Direct Linking Loader will be written in the file "result.txt".
- Terminate the program.

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
#include<process.h>
struct estab
{
        char sy[20];
        char addr[10];
}
e[]={{"LISTA","0040"},{"ENDA","0054"},{"LISTB","00C3"},{"ENDB","00D3"},{"LI
STC","0112"},{"ENDC","0124"}};
struct obj
{
        char r[10];
        char se[20];
        char th[20];
        char fo[20];
        char fi[20];
}o;
void main()
{
        int n=0,j=0,i;
        FILE *pf,*st;
        clrscr();
        pf=fopen("objpgm.txt","r");
        st=fopen("result.txt","w+");
        fscanf(pf,"%s%s%s%s%s",o.r,o.se,o.th,o.fo,o.fi);
        for(i=0;i<=2;i++)
        {
                while(strcmp(o.r,"E")!=0)
                {
                if(strcmp(o.r,"R")==0)
                {
                        for(j=0;j<6;j++)
                        {
                                if(strcmp(o.se,e[j].sy)==0)
                                        strcat(o.se,e[j].addr);
                                if(strcmp(o.th,e[j].sy)==0)
                                        strcat(o.th,e[j].addr);
                                if(strcmp(o.fo,e[j].sy)==0)
                                        strcat(o.fo,e[j].addr);
                                if(strcmp(o.fi,e[j].sy)==0)
                                        strcat(o.fi,e[j].addr);
                        }
                }
                if(strcmp(o.r,"M")==0)
                {
```

```
                        for(j=0;j<6;j++)
                        {
                                if(strcmp(o.fi,e[j].sy)==0)
                                {
                                        strcpy(o.fi,e[j].addr);
                                }
                        }
                }
                n++;
                fprintf(st,"\n%s\t%s\t%s\t%s\t%s\t\n",o.r,o.se,o.th,o.fo,o.fi);
                fscanf(pf,"\n%s%s%s%s%s",o.r,o.se,o.th,o.fo,o.fi);
        }
        fprintf(st,"\n%s\t%s\t%s\t%s\t%s\t\n",o.r,o.se,o.th,o.fo,o.fi);
        fprintf(st,"\n\n");
        fscanf(pf,"%s%s%s%s%s",o.r,o.se,o.th,o.fo,o.fi);
    }
    fclose(pf);
    fclose(st);
    printf("\nPASS 2 DIRECT LINKING LOADER RESULT IS WRITTEN IN
    RESULT.TXT FILE");
    getch();
}
```

INPUT & OUTPUT:

INPUT

Objpgm.txt

```
H       PROGA       0000    0063    -
D       LISTA 0040  ENDA 0054
R       LISTB ENDB  LISTC ENDC
M       000024      06      +       LISTB
M       000054      05      +       LISTC
M       000057      05      +       ENDB
M       00005A      06      +       ENDC
E       -       -       -       -

H       PROGB       0000    007F    -
D       LISTB 00C3  ENDB 00D3
R       LISTA ENDA  LISTC ENDC
M       000024      06      +       LISTA
M       000054      06      +       LISTC
M       000057      05      +       ENDA
M       00005A      05      +       ENDC
E       -       -       -       -

H       PROGC       0000    0051    -
D       LISTC 0112  ENDC 0124
R       LISTB ENDB  LISTA ENDA
M       000024      05      +       LISTB
M       000054      06      +       LISTA
M       000057      05      +       ENDB
M       00005A      06      +       ENDA
E       -       -       -       -
```

OUTPUT:

PASS 2 DIRECT LINKING LOADER RESULT IS WRITTEN IN RESULT.TXT FILE

Result.txt

```
H       PROGA       0000    0063    -

D       LISTA 0040  ENDA 0054

R       LISTB00C3   ENDB00D3   LISTC0112    ENDC0124

M       000024      06      +       00C3

M       000054      05      +       0112

M       000057      05      +       00D3

M       00005A      06      +       0124
```

```
E       -       -       -       -
H       PROGB   0000    007F    -
D       LISTB 00C3      ENDB 00D3
R       LISTA0040       ENDA0054        LISTC0112       ENDC0124
M       000024  06      +       0040
M       000054  06      +       0112
M       000057  05      +       0054
M       00005A  05      +       0124
E       -       -       -       -
H       PROGC   0000    0051    -
D       LISTC 0112      ENDC 0124
R       LISTB00C3       ENDB00D3        LISTA0040       ENDA0054
M       000024  05      +       00C3
M       000054  06      +       0040
M       000057  05      +       00D3
M       00005A  06      +       0054
E       -       -       -       -
```


RESULT:

Thus the program to implement the pass2 of direct linking loader was executed and output was verified.

EX.NO.10 **TEXT EDITOR**

AIM

To write a "C" program for implementing simple text editor with the features like insert, search, replace, delete and display.

ALGORITHM

- Start the program.
- Declare all the variables in the text editor.
- Declare all the functions insert (), search (), replace (), del () and display ().
- Create the structure having one variable str[100].
- If we want to enter the new string in the text editor, call the insert () function and enter the string to be inserted and write the content in the text editor.
- If we want to search a particular string in the text editor, call the search () function and enter the string to be searched. The enter string is compared with the available strings in the text editor. If found then type "String Found" otherwise "String Not Found".
- If we want to replace the string in the text editor, call the replace () function and enter the string to be replaced and also the new string. The function will replace the existing string with the new string.
- If we want to delete a particular string in the text editor, call the del () function and enter the string to be deleted. The function will delete the string from the text editor.
- The display () function is called after performing each operation to display the contents in the text editor.
- Terminate the program

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
struct list
{
        char text[100];
}l[100];
int length,i,len=1,x=0,n;
char string[100],r[100],d[100],s[100];
void insert();
void search();
void replace();
void del();
void display();
void main()
{
        int ch;
        clrscr();
        do
        {
                printf("\n1.Insert\n2.Search\n3.Replace\n4.Delete\n5.Display\n6.Exit\n");
                printf("Enter Your Choice:");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                insert();
                                break;
                        case 2:
                                search();
                                break;
                        case 3:
                                replace();
                                break;
                        case 4:
                                del();
                                break;
                        case 5:
                                display();
                                break;
                        case 6:
                                exit(1);
                                break;
                }
        }while(ch<6);
}
```

```
void insert()
{
        printf("\nEnter the string :");
        scanf("%s",l[len].text);
        len=len+1;
}
void search()
{
        printf("\n Enter the string to be searched :");
        scanf("%s",s);
        for(i=1;i<=len;i++)
        {
                x=strcmp(l[i].text,s);
                if(x==0)
                {
                        printf("\n String Found!");
                        break;
                }
        }
        if(x!=0)
        {
                printf("\n String Not Found!");
        }
}
void replace()
{
        printf("\n Enter the string to be replaced :");
        scanf("%s",s);
        for(i=0;i<=len;i++)
        {
                x=strcmp(l[i].text,s);
                if(x==0)
                {
                        printf("\n Enter the new string :");
                        scanf("%s",r);
                        strcpy(l[i].text,r);
                        printf("\n String replaced successfully!");
                        break;
                }
        }
        if(x!=0)
        {
                printf("\n String Not Found to Replace.");
        }
}
void del()
{
        printf("\n Enter the String to be deleted :");
        scanf("%s",d);
```

```
        for(i=1;i<=len;i++)
        {
                x=strcmp(l[i].text,d);
                if(x==0)
                {
                        strcpy(l[i].text,"");
                        printf("\n String is deleted successfully!");
                        break;
                }
        }
        if(x!=0)
                printf("\n String Not Found.");
}
void display()
{
        printf("\nFinal String :\n");
        for(i=0;i<len;i++)
        {
                printf("%s",l[i].text);
        }
        printf("\n");
        getch();
}
```

OUTPUT

```
1.Insert
2.Search
3.Replace
4.Delete
5.Display
6.Exit
Enter Your Choice :1
Enter the string :SYSTEM

1.Insert
2.Search
3.Replace
4.Delete
5.Display
6.Exit
Enter Your Choice :1
Enter the string :SOFTWARE

1.Insert
2.Search
3.Replace
4.Delete
5.Display
6.Exit
Enter Your Choice :5
Final String :SYSTEMSOFTWARE

1.Insert
2.Search
3.Replace
4.Delete
5.Display
6.Exit
Enter Your Choice :2
Enter the string to be searched :SYSTEM
String Found!
1.Insert
2.Search
3.Replace
4.Delete
5.Display
6.Exit
Enter Your Choice :2
Enter the string to be searched :LAB
String Not Found!
```

1.Insert
2.Search
3.Replace
4.Delete
5.Display
6.Exit
Enter Your Choice :3
Enter the string to be replaced :SOFTWARE
Enter the new string :SOFT
String replaced successfully!

1.Insert
2.Search
3.Replace
4.Delete
5.Display
6.Exit
Enter Your Choice :5
Final String :SYSTEMSOFT

1.Insert
2.Search
3.Replace
4.Delete
5.Display
6.Exit
Enter Your Choice :4
Enter the String to be deleted :SOFT
String is deleted successfully!

1.Insert
2.Search
3.Replace
4.Delete
5.Display
6.Exit
Enter Your Choice :5
Final String :SYSTEM

1.Insert
2.Search
3.Replace
4.Delete
5.Display
6.Exit
Enter Your Choice :6

RESULT:

Thus the program for implementing simple text editor with the features like insert, search, replace, delete and display was executed and the output was verified.

EX.NO.11 SYMBOL TABLE USING HASHING

AIM

To write a C program to implement symbol table using hashing.

ALGORITHM

- Start the program.
- Declare all the variables in the HASHING.
- Declare all the functions create (), search (), and display ().
- Create the structure having one variable str[100].
- If we want to enter the new string in the text editor, call the insert () function and enter the string to be inserted and write the content in the text editor.
- If we want to search a particular string in the text editor, call the search () function and enter the string to be searched. The enter string is compared with the available strings in the text editor. If found then type “String Found” otherwise “String Not Found”.
- If we want to replace the string in the text editor, call the replace () function and enter the string to be replaced and also the new string. The function will replace the existing string with the new string.
- If we want to delete a particular string in the text editor, call the del () function and enter the string to be deleted. The function will delete the string from the text editor.
- The display () function is called after performing each operation to display the contents in the text editor.
- Terminate the program

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 11
char l[10];
struct symb
{
int add;
char label[10];
}sy[11];
void search();
void main()
{
int a[MAX],num,key,i,ch;
char ans;
int create(int);
void lprob(int [],int,int);
void display(int []);
clrscr();
for(i=0;i<MAX;i++)
a[i]=0;
do
{
printf("\nenter your choice: 1.create a symbol table 2.search in the symbol table\n");
scanf("%d",&ch);
switch(ch)
{
case 1:
do
{
printf("\nEnter the address:");
```

```
scanf("%d",&num);
key=create(num);
printf("enter The label:");
scanf("%s",l);
lprob(a,key,num);
printf("\nContinue(y/n)?");
ans=getche();
}
while(ans=='y');
display(a);
break;
case 2:
search();
break;
}
}while(ch<=2);
getch();
}

int create(int num)
{
int key;
key=num%11;
return key;
}
void lprob(int a[MAX],int key,int num)
{
int flag,i,count=0;
void display(int a[]);
flag=0;
if(a[key]==0)
{
```

```
a[key]=num;
sy[key].add=num;
strcpy(sy[key].label,l);
}
else
{
i=0;
while(i<MAX)
{
if(a[i]!=0)
count++;
i++;
}
if(count==MAX)
{
printf("\nHash table is full");
display(a);
getch();
exit(1);
}
for(i=key+1;i<MAX;i++)
if(a[i]==0)
{
a[i]=num;
flag=1;
sy[key].add=num;
strcpy(sy[key].label,l);
break;
}
for(i=0;i<key && flag==0;i++)
if(a[i]==0)
{
a[i]=num;
```

```
flag=1;
sy[key].add=num;
strcpy(sy[key].label,l);
break;
}
}
}
void display(int a[MAX])
{
FILE *fp;
int i;
fp=fopen("symbol.txt","w");
printf("\nThe Symbol Table is");
printf("\nhashvalues address label");
for(i=0;i<MAX;i++)
{
printf("\n%d\t %d\t %s",i,sy[i].add,sy[i].label);
fprintf(fp,"\n%d %d %s",i,sy[i].add,sy[i].label);
}
fclose(fp);
}

void search()
{
FILE *fp1;
char la[10];
int set=0,s;
int j,i;
printf("enter the label: ");
scanf("%s",la);
fp1=fopen("symbol.txt","r");
for(i=0;i<MAX;i++)
{
```

```
fscanf(fp1,"%d%d",&j,&sy[i].add);
if(sy[i].add!=0)
fscanf(fp1,"%s",sy[i].label);
}
for(i=0;i<MAX;i++)
{
if(sy[i].add!=0)
{
if(strcmp(sy[i].label,la)==0)
{
set=1;
s=sy[i].add;
}
}
}
if(set==1)
printf("\nThe label --%s-- is present in the symbol table at address:%d\n",la,s);
else
printf("\nThe label is not present in the symbol table\n");
}
```

OUTPUT:

```
enter your choice: 1.create a symbol table 2.search in the symbol table
1

Enter the address:1000
enter The label:sta

Continue(y/n)?y
Enter the address:
2000
enter The label:lda

Continue(y/n)?y
Enter the address:
3000
enter The label:add

Continue(y/n)?n
The Symbol Table is
hashvalues address label
0       0
1       0
2       0
3       0
4       0
5       0
6       0
7       0
8       3000    add
9       2000    lda
10      1000    sta
enter your choice: 1.create a symbol table 2.search in the symbol table

2
enter the label: sta

The label --sta-- is present in the symbol table at address:1000

enter your choice: 1.create a symbol table 2.search in the symbol table
```


RESULT:

Thus the program for implementing hashing with the features like create, search and display was executed and the output was verified.