Ex.No.:11 **SINGLE PASS ASSEMBLER**

Date:

AIM:

To implement a single pass assembler in C language.

ALGORITHM:

1. Open and Read the input file
2. If the input line has the opcode “START” do the following
2.1 Find if there is any operand field after “START”, initialize the LC to the operand value
2.2 Otherwise if there is no value in the operand field then LC is set to 0
3. Write the input line to the intermediate file
4. Do the following steps until the opcode is END
4.1 Check the Symbol table, if the symbol is not available then enter that symbol into the SYMTAB, along with the memory address in which it is stored. Otherwise, the error message should be displayed
4.2 If there is a opcode
4.2.1 If opcode is present in the OPTAB, then increment the LC by 3 and Start writing the location counter, opcode and operand fields of the corresponding statement to the output file, along with the object code.
4.2.2 If opcode is “WORD”, then increment LC by 3;
4.2.3 If opcode is “BYTE”, then increment LC by 1;
4.2.4 If opcode is “RESW” then increment LC by the integer equivalent of the operand value * 3;
4.2.5 If opcode is “RESB”, then increment LC by the integer equivalent of the operand value
4.2.6 If there is no symbol/label in the operand field, then the operand address is assigned as zero and it is assembled with the object code of the instruction
4.2.7 Write the processed lines in the intermediate file along with their location counters

5. To find the length of the program, Subtract the starting address of the program from the final value of the LC
Close all the files and exit

\

}

PROGRAM:

```
#include<stdio.h>
#include<string.h>
int findInSYMTAB(char findLabel[]){
FILE *SYMTAB;
char label[30],addr[30];
SYMTAB=fopen("symtab.dat","r");
fscanf(SYMTAB,"%s%s",label,addr);
while(1){
if(feof(SYMTAB)){
fclose(SYMTAB);
break;
}
if(strcmp(findLabel,label)==0){
fclose(SYMTAB);
return atoi(addr);
}
fscanf(SYMTAB,"%s%s",label,addr);
}
}
int getMnemonicCode(char mnemonic[]){
if(strcmp(mnemonic,"LDA")==0)
return 33;
else if(strcmp(mnemonic,"STA")==0)
return 44;
else if(strcmp(mnemonic,"LDCH")==0)
return 53;
else if(strcmp(mnemonic,"STCH")==0)
return 57;
else
return -1;
}
void main(){
FILE  *INPUT,*OUTPUT,*SYMTAB,*INTERMEDIATE,*FINAL,*OBJ;
char  mnemonic[10][10]={"START","LDA","STA","LDCH","STCH","END"};
```

```
int LOCCTR,start=0,j=0,i,length,Tlength,count=0,finalAddress,startAddr;
char label[20],opcode[20],operand[20],address[20];
INPUT=fopen("input.dat","r");
OUTPUT=fopen("output.dat","w");
SYMTAB=fopen("symtab.dat","w");
fscanf(INPUT,"%s%s%s",label,opcode,operand);
if(strcmp(opcode,"START")==0){
start=atoi(operand);
LOCCTR=start;
fprintf(OUTPUT,"t%st%st%sn",label,opcode,operand);
fscanf(INPUT,"%s%s%s",label,opcode,operand);
}else
LOCCTR=0;
while(strcmp(opcode,"END")!=0){
j=0;
fprintf(OUTPUT,"%d",LOCCTR);
if(strcmp(label,"**")!=0)
fprintf(SYMTAB,"t%st%dn",label,LOCCTR);
while(strcmp(mnemonic[j],"END")!=0){
if(strcmp(mnemonic[j],opcode)==0){
LOCCTR+=3;
}
j++;
}
if(strcmp(opcode,"WORD")==0)
LOCCTR+=3;
else if (strcmp(opcode,"RESW")==0){
LOCCTR=LOCCTR+(3* atoi(operand));
count+=(3* atoi(operand));
}
else if (strcmp(opcode,"RESB")==0){
LOCCTR=LOCCTR+atoi(operand);
count+=atoi(operand);
}
else if (strcmp(opcode,"BYTE")==0){
```

```
LOCCTR=LOCCTR+(strlen(operand)-3);
}
else {
printf(" ");
}
fprintf(OUTPUT,"t%st%st%sn",label,opcode,operand);
fscanf(INPUT,"%s%s%s",label,opcode,operand);
}
fprintf(OUTPUT,"%d",LOCCTR);
fprintf(OUTPUT,"t%st%st%sn",label,opcode,operand);
printf("nn THE LENGTH OF THE PROGRAM IS %d",LOCCTR-start);
length=LOCCTR-start;
finalAddress=LOCCTR;
Tlength=length-count;
fcloseall();
//=============================================================
=
INTERMEDIATE=fopen("output.dat","r");
OBJ=fopen("obj.dat","w");
FINAL=fopen("final.dat","w");
fscanf(INTERMEDIATE,"%s%s%s",label,opcode,operand);
startAddr=atoi(operand);
if( strcmp(opcode,"START")==0){
fprintf(FINAL,"%st%st%stn",label,opcode,operand);
fprintf(OBJ,"H^%s^00%s^00%dn",label,operand,length);
fscanf(INTERMEDIATE,"%s%s%s%s",address,label,opcode,operand);
fprintf(OBJ,"T^%06d^%d^",atoi(address),Tlength);
}
while(strcmp(opcode,"END")!=0){
if(strcmp(label,"**")==0 ){
fprintf(OBJ,"%d%d^",getMnemonicCode(opcode),findInSYMTAB(operand));
fprintf(FINAL,"%st%st%st%st%d%dn",address,label,opcode,operand,getMnemonicC
ode(opcode),findInSYMTAB(operand));
fscanf(INTERMEDIATE,"%s%s%s%s",address,label,opcode,operand);
}
```

```
else if(strcmp(opcode,"BYTE")==0){
fprintf(FINAL,"%st%st%st%s",address,label,opcode,operand);
for(i=2;i<(strlen(operand)-1);i++){
fprintf(OBJ,"%x",operand[i]);
fprintf(FINAL,"%x",operand[i]);
}
fprintf(FINAL,"n");
fscanf(INTERMEDIATE,"%s%s%s%s",address,label,opcode,operand);
}
else if(strcmp(opcode,"WORD")==0){
fprintf(OBJ,"%06x^",atoi(operand));
fprintf(FINAL,"%st%st%st%st%06xn",address,label,opcode,operand,atoi(operand));
Name : Hariharan M
Register Number : 913121205028
21CS211-Operating Systems Laboratory Dept of IT
fscanf(INTERMEDIATE,"%s%s%s%s",address,label,opcode,operand);
}
else{
fscanf(INTERMEDIATE,"%s%s%s%s",address,label,opcode,operand);
}
}
fprintf(FINAL,"%st%st%st%sn",address,label,opcode,operand);
fprintf(OBJ,"nE^%06dn",startAddr);
fcloseall();
remove("output.dat");
}
```

Input Files :

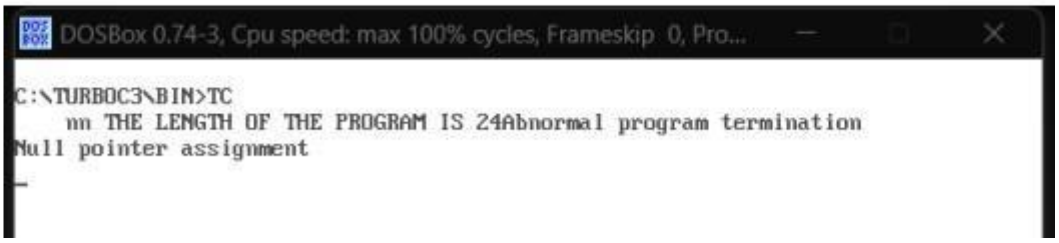
input.dat

```
** START 2000
** LDA FIVE
** STA ALPHA
** LDCH STRING
** STCH C1
ALPHA RESW 1
FIVE WORD 5
STRING BYTE C'HELLO'
C1 RESB 1
** END **
```

Output files

```
final.dat ( output generated )
** START 2000
2000 ** LDA FIVE 332015
2003 ** STA ALPHA 442012
2006 ** LDCH STRING 532018
2009 ** STCH C1 572023
2015 FIVE WORD 5 000005
2018 STRING BYTE C'HELLO'48454c4c4f
2024 ** END **
symtab.dat ( SYMTAB )
ALPHA 2012
FIVE 2015
STRING 2018
C1 2023
obj.dat ( object code generated )
H^**^002000^0024
T^002000^20^332015^442012^532018^572023^000005^48454c4c4f
E^002000
```

Output :



```
C:\TURBOC3\BIN>TC
    nn THE LENGTH OF THE PROGRAM IS 24Abnormal program termination
Null pointer assignment
_
```

Observation	
Record	
Total	
Initial	

RESULT:

Thus single pass assembler is implemented in C language.

Ex.No: 12a **PASS ONE OF A TWO PASS ASSEMBLER**

Date:

AIM:

To write a “c” program to implement pass one of a two pass assembler

ALGORITHM:

1. Open and Read the input file
2. If the input line has the opcode “START” do the following

2.1 Find if there is any operand field after “START”, initialize the LOCCTR to the operand value

2.2 Otherwise if there is no value in the operand field then LOCCTR is set to 0

3. Write the input line to the intermediate file
4. Do the following steps until the opcode is END

4.1 If opcode is “WORD”, then increment LOCCTR by 3;

4.2 If opcode is “BYTE”, then increment LOCCTR by 1;

5. Write the processed lines in the intermediate file along with their location counters
6. To find the length of the program, Subtract the starting address of the program from the final value of the LOCCTR

Close all the files and exit

Program :

```
#include<stdio.h>

#include<string.h>

void main()

{

 FILE *f1,*f2,*f3,*f4;

 int lc,sa,l,op1,o,len;

 char m1[20],la[20],op[20],otp[20];
```

Name : Hariharan M

Register Number : 913121205028

21CS211-Operating Systems Laboratory Dept of IT

```
 f1=fopen("input.txt","r");
```

```
f3=fopen("symtab.txt","w");
fscanf(f1,"%s%s%d",la,m1,&op1);
if(strcmp(m1,"START")==0)
{
sa=op1;
lc=sa;
printf("\t%s\t%s\t%d\n",la,m1,op1);
}
else
{
lc=0;
}
fscanf(f1,"%s%s",la,m1);
while(!feof(f1))
{
fscanf(f1,"%s",op);
printf("\n%d\t%s\t%s\n",lc,la,m1,op);
if(strcmp(la,"-")!=0)
{
fprintf(f3,"\n%d\t%s\n",lc,la);
}
f2=fopen("optab.txt","r");
fscanf(f2,"%s%d",otp,&o);
while(!feof(f2))
{
if(strcmp(m1,otp)==0)
{
lc=lc+3;
```

```
 break;
}
 fscanf(f2,"%s%d",otp,&o);
 }
 fclose(f2);
 if(strcmp(m1,"WORD")==0)
 {
 lc=lc+3;
 }
 else if(strcmp(m1,"RESW")==0)
 {
 op1=atoi(op);
 lc=lc+(3*op1);
 }
 else if(strcmp(m1,"BYTE")==0)
 {
 if(op[0]=='X')
 {
 lc=lc+1;
 }
 else
 {
 len=strlen(op)-2;
 lc=lc+len;
 }
 }
 else if(strcmp(m1,"RESB")==0)
 {
```

```
 op1=atoi(op);
 lc=lc+op1;
 }
fscanf(f1,"%s%s",la,m1);
 }
 if(strcmp(m1,"END")==0)
 {
 printf("program length=\n%d",lc-sa);
 }
 fclose(f1);
 fclose(f3);
}
```

Input Files :

input.txt

```
copy START 1000
 LDA ALPHA
 ADD ONE
 SUB TWO
 STA BETA
ALPHA BYTE C'KLNCE
ONE RESB 2
TWO WORD 5
BETA RESW 1
_ END _
```

optab.txt

```
LDA 00
STA 23
ADD 01
```

SUB 05

symtab.txt

1000 LDA

1000 ONE

1003 STA

1003 BYTE

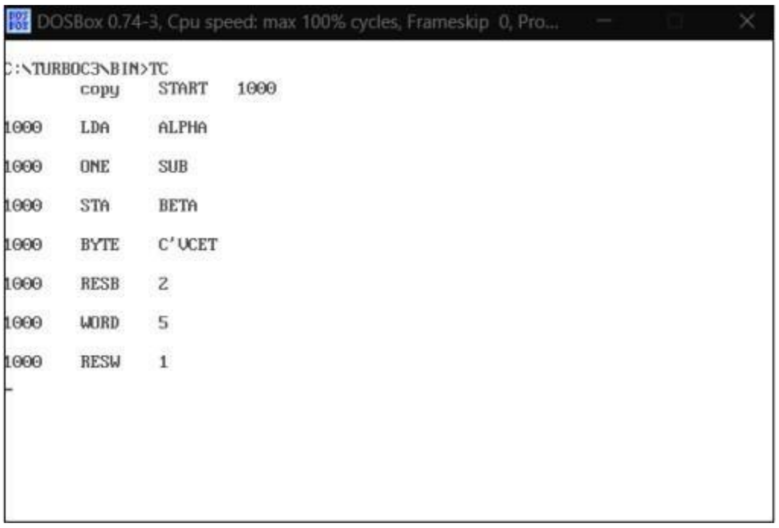
1003 RESB

1003 WORD

1003 RESW

1003 END

OUTPUT:



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

```
C:\TURBOC3\BIN>TC
        copy    START   1000

1000    LDA     ALPHA

1000    ONE     SUB

1000    STA     BETA

1000    BYTE    C'UCET

1000    RESB    2

1000    WORD    5

1000    RESW    1
-
```

RESULT:

Thus the “c” program to implement pass one of a two pass assembler has been written and executed successfully

Ex.No: 12b **PASS TWO OF TWO PASS ASSEMBLER**

Date :

AIM:

To implement pass two of a two pass assembler in C language.

ALGORITHM:

```
1. Open and read the first line from the intermediate file.
2. If the first line contains the opcode "START", then write the label, opcode and operand field
values of the corresponding statement directly to the final output file.
3. Do the following steps, until an "END" statement is reached.
3.1 Start writing the location counter, opcode and operand fields of the corresponding
statement to the output file, along with the object code.
3.2 If there is no symbol/label in the operand field, then the operand address is assigned as
zero and it is assembled with the object code of the instruction
3.3 If the opcode is BYTE, WORD, RESB etc convert the constants to the object code.
4. Close the files and exit
```

Program :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
 char a[10],ad[10],label[10],opcode[10],operand[10],symbol[10],ch; int
st,diff,i,address,add,len,actual_len,finaddr,prevaddr,j=0;
 char mnemonic[15][15]={"LDA","STA","LDCH","STCH"};
 char code[15][15]={"33","44","53","57"};
 FILE *fp1,*fp2,*fp3,*fp4;
 clrscr();
 fp1=fopen("ASSMLIST.DAT","w");
 fp2=fopen("SYMTAB.DAT","r");
 fp3=fopen("INTERMED.DAT","r");
 fp4=fopen("OBJCODE.DAT","w");
 fscanf(fp3,"%s%s%s",label,opcode,operand);
 while(strcmp(opcode,"END")!=0)
 {
 prevaddr=address;
 fscanf(fp3,"%d%s%s%s",&address,label,opcode,operand);
 }
 finaddr=address;
 fclose(fp3);
 fp3=fopen("INTERMED.DAT","r");
 fscanf(fp3,"%s%s%s",label,opcode,operand);
 if(strcmp(opcode,"START")==0)
```

```
{
fprintf(fp1,"\t%s\t%s\t%s\n",label,opcode,operand);
fprintf(fp4,"H^%s^00%s^00%d\n",label,operand,finaddr);
fscanf(fp3,"%d%s%s%s",&address,label,opcode,operand);
st=address;
diff=prevaddr-st;
fprintf(fp4,"T^00%d^%d",address,diff);
}
while(strcmp(opcode,"END")!=0)
{
if(strcmp(opcode,"BYTE")==0)
{
fprintf(fp1,"%d\t%s\t%s\t%s\t",address,label,opcode,operand);
len=strlen(operand);
actual_len=len-3;
fprintf(fp4,"^");
for(i=2;i<(actual_len+2);i++)
{
itoa(operand[i],ad,16);
fprintf(fp1,"%s",ad);
fprintf(fp4,"%s",ad);
}
fprintf(fp1,"\n");
}
else if(strcmp(opcode,"WORD")==0)
{
len=strlen(operand);
itoa(atoi(operand),a,10);
fprintf(fp1,"%d\t%s\t%s\t%s\t00000%s\n",address,label,opcode,operand,a);
fprintf(fp4,"^00000%s",a);
}
else  if((strcmp(opcode,"RESB")==0)||(strcmp(opcode,"RESW")==0))
fprintf(fp1,"%d\t%s\t%s\t%s\n",address,label,opcode,operand);
else
{
while(strcmp(opcode,mnemonic[j])!=0)
j++;
if(strcmp(operand,"COPY")==0)
fprintf(fp1,"%d\t%s\t%s\t%s\t%s0000\n",address,label,opcode,operand,code[j]);
else
{
rewind(fp2);
fscanf(fp2,"%s%d",symbol,&add);
while(strcmp(operand,symbol)!=0)
fscanf(fp2,"%s%d",symbol,&add);
fprintf(fp1,"%d\t%s\t%s\t%s\t%s%d\n",address,label,opcode,operand,code[j],add);
fprintf(fp4,"^%s%d",code[j],add);
}
}
fscanf(fp3,"%d%s%s%s",&address,label,opcode,operand);
}
fprintf(fp1,"%d\t%s\t%s\t%s\n",address,label,opcode,operand);
```

```
 fprintf(fp4,"\nE^00%d",st);
 printf("\n Intermediate file is converted into object code");
 fcloseall();
 printf("\n\nThe contents of Intermediate file:\n\n\t");
 fp3=fopen("INTERMED.DAT","r");
 ch=fgetc(fp3);
 while(ch!=EOF)
 {
 printf("%c",ch);
 ch=fgetc(fp3);
 }
 printf("\n\nThe contents of Symbol Table :\n\n");
 fp2=fopen("SYMTAB.DAT","r");
 ch=fgetc(fp2);
 while(ch!=EOF)

 {
 printf("%c",ch);
 ch=fgetc(fp2);
 }
 printf("\n\nThe contents of Output file :\n\n");
 fp1=fopen("ASSMLIST.DAT","r");
 ch=fgetc(fp1);
 while(ch!=EOF)
 {
 printf("%c",ch);
 ch=fgetc(fp1);
 }
 printf("\n\nThe contents of Object code file :\n\n");
 fp4=fopen("OBJCODE.DAT","r");
 ch=fgetc(fp4);
 while(ch!=EOF)
 {
 printf("%c",ch);
 ch=fgetc(fp4);
 }
 fcloseall();
 getch();
}
```

INPUT FILES:

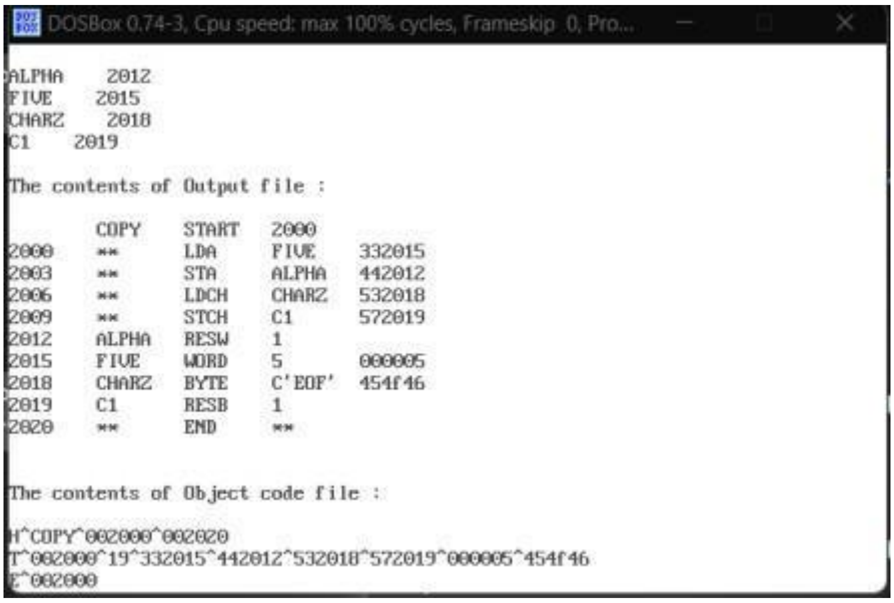
INTERMED.DAT

```
COPY START 2000
2000 ** LDA FIVE
2003 ** STA ALPHA
2006 ** LDCH CHARZ
2009 ** STCH C1
2012 ALPHA RESW 1
2015 FIVE WORD 5
2018 CHARZ BYTE C'EOF'
2019 C1 RESB 1
2020 ** END **
```

SYMTAB.DAT

ALPHA 2012
FIVE 2015
CHARZ 2018
C1 2019

OUTPUT:


DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Pro...

```
ALPHA    2012
FIVE    2015
CHARZ    2018
C1    2019

The contents of Output file :

        COPY    START   2000
2000    **      LDA     FIVE    332015
2003    **      STA     ALPHA   442012
2006    **      LDCH    CHARZ   532018
2009    **      STCH    C1      572019
2012    ALPHA   RESW    1
2015    FIVE    WORD    5       000005
2018    CHARZ   BYTE    C'EOF'  454f46
2019    C1      RESB    1
2020    **      END     **


The contents of Object code file :

H^COPY^002000^002020
T^002000^19^332015^442012^532018^572019^000005^454f46
E^002000
```

Observation	
Record	
Total	
Initial	

RESULT:

Thus pass two of a two pass assembler is implemented in C language.