System Software programs…

To implement pass one of a two pass assembler.

```
# include <stdio.h>
# include <string.h>
void main()
{
char opcode[10],mnemonic[3],operand[10],label[10],code[10];
int locctr,start,length;
FILE *fp1,*fp2,*fp3,*fp4;
fp1=fopen("input.dat","r");
fp2=fopen("symtab.dat","w");
fp3=fopen("out.dat","w");
fp4=fopen("optab.dat","r");
fscanf(fp1,"%s%s%s",label,opcode,operand);
if(strcmp(opcode,"START")==0){
start=atoi(operand);
locctr=start;
fprintf(fp3,"\t%s\t%s\t%s\n",label,opcode,operand);
fscanf(fp1,"%s%s%s",label,opcode,operand);
}
else
locctr=0;
while(strcmp(opcode,"END")!=0)
{
fprintf(fp3,"%d\t",locctr);
if(strcmp(label,"**")!=0)
fprintf(fp2,"%s\t%d\n",label,locctr);
rewind(fp4);
fscanf(fp4,"%s",code);
while(strcmp(code,"END")!=0)
{
if(strcmp(opcode,code)==0)
{
locctr+=3;
break;
}
fscanf(fp4,"%s",code);
}
if(strcmp(opcode,"WORD")==0)
locctr+=3;
else if(strcmp(opcode,"RESW")==0)
locctr+=(3*(atoi(operand)));
else if(strcmp(opcode,"RESB")==0)
locctr+=(atoi(operand));
else if(strcmp(opcode,"BYTE")==0)
++locctr;
fprintf(fp3,"%s\t%s\t%s\n",label,opcode,operand);
fscanf(fp1,"%s%s%s",label,opcode,operand);
}
fprintf(fp3,"%d\t%s\t%s\t\%s\n",locctr,label,opcode,operand);
length=locctr-start;
printf("The length of the program is %d",length);
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
}
```

INPUT FILES---INPUT.DAT

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

OPTAB.DAT

```
START
LDA
STA
LDCH
STCH
END
```

OUTPUT FILES

OUT.DAT

```
** 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’Z’
2019 C1 RESB 1
2020 ** END **
```

SYMTAB.DAT

```
ALPHA 2012
FIVE 2015
CHARZ 2018
C1 2018
```

Implement pass two of a two pass assembler.

AIM :

To implement pass two of a two pass assembler.

Source code program in c pass two of pass two assembler

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
char a[10],ad[10],label[10],opcode[10],operand[10],mnemonic[10],symbol[10];
int i,address,code,add,len,actual_len;
FILE *fp1,*fp2,*fp3,*fp4;
fp1=fopen("assmlist.dat","w");fp2=fopen("symtab.dat","r");
fp3=fopen("intermediate.dat","r");
fp4=fopen("optab.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);
fscanf(fp3,"%d%s%s%s",&address,label,opcode,operand);
}
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;
for(i=2;i<(actual_len+2);i++)
{
//itoa(operand[i],ad,16);
sprintf(ad,"%d",operand[i]);
fprintf(fp1,"%s",ad);
}
fprintf(fp1,"\n");
}
else if(strcmp(opcode,"WORD")==0)
{
len=strlen(operand);
//itoa(atoi(operand),a,10);
sprintf(a,"%d",atoi(operand));
fprintf(fp1,"%d\t%s\t%s\t%s\t00000%s\n",address,label,opcode,operand,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
{
rewind(fp4);
fscanf(fp4,"%s%d",mnemonic,&code);
while(strcmp(opcode,mnemonic)!=0)
fscanf(fp4,"%s%d",mnemonic,&code);
if(strcmp(operand,"**")==0)
{
fprintf(fp1,"%d\t%s\t%s\t%s\t%d0000\n",address,label,opcode,operand,code);
}
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%d%d\n",address,label,opcode,operand,code,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);
printf("Finished");
fclose(fp1);
fclose(fp2);
fclose(fp3);
fclose(fp4);
}
```

INPUT FILES

INTERMEDIATE.DAT

```
** 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 **
```

OPTAB.DAT

```
LDA 33
STA 44
LDCH 53
STCH 57
END *
```

SYMTAB.DAT

```
ALPHA 2012
FIVE 2015
CHARZ 2018
C1 2019
```

OUTPUT FILE :
ASSMLIST.DAT

```
** 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
END **
```

To implement an absolute loader.

```
# include <stdio.h>
# include <string.h>
void main()
{
char input[10];
int start,length,address;
FILE *fp1,*fp2;
fp1=fopen("input.dat","r");
fp2=fopen("output.dat","w");
fscanf(fp1,"%s",input);
while(strcmp(input,"E")!=0)
{
if(strcmp(input,"H")==0)
{
fscanf(fp1,"%d",&start);
fscanf(fp1,"%d",&length);
fscanf(fp1,"%s",input);
}
else if(strcmp(input,"T")==0)
{
fscanf(fp1,"%d",&address);
fscanf(fp1,"%s",input);
fprintf(fp2,"%d\t%c%c\n",address,input[0],input[1]);
fprintf(fp2,"%d\t%c%c\n",(address+1),input[2],input[3]);
fprintf(fp2,"%d\t%c%c\n",(address+2),input[4],input[5]);
address+=3;
fscanf(fp1,"%s",input);
}
else
{
fprintf(fp2,"%d\t%c%c\n",address,input[0],input[1]);
fprintf(fp2,"%d\t%c%c\n",(address+1),input[2],input[3]);
fprintf(fp2,"%d\t%c%c\n",(address+2),input[4],input[5]);
address+=3;
fscanf(fp1,"%s",input);
}
}
fclose(fp1);
fclose(fp2);
printf("FINISHED");
}
```

INPUT FILE

INPUT.DAT

```
H 1000 232
T 1000 142033 483039 102036
T 2000 298300 230000 282030 302015
E
```

OUTPUT FILE

OUTPUT.DAT

```
1000 14
1001 20
1002 33
```

```
1003 48
1004 30
1005 39
1006 10
1007 20
1008 36
2000 29
2001 83
2002 00
2003 23
2004 00
2005 00
2006 28
2007 20
2008 30
2009 30
2010 20
2011 15
```

To implement a relocating loader.
Source code program in c for Relocation Loader

```
# include <stdio.h>
# include <string.h>
# include <stdlib.h>
void main()
{
char add[6],length[10],input[10],binary[12],bitmask[12],relocbit;
int start,inp,len,i,address,opcode,addr,actualadd;
FILE *fp1,*fp2;
printf("Enter the actual starting address : ");
scanf("%d",&start);
fp1=fopen("relinput.dat","r");
fp2=fopen("reloutput.dat","w");
fscanf(fp1,"%s",input);
while(strcmp(input,"E")!=0)
{
if(strcmp(input,"H")==0)
{
fscanf(fp1,"%s",add);
fscanf(fp1,"%s",length);
fscanf(fp1,"%s",input);
}
if(strcmp(input,"T")==0)
{
fscanf(fp1,"%d",&address);
fscanf(fp1,"%s",bitmask);
address+=start;
len=strlen(bitmask);
for(i=0;i<len;i++)
{
fscanf(fp1,"%d",&opcode);
fscanf(fp1,"%d",&addr);
relocbit=bitmask[i];
if(relocbit=='0')
actualadd=addr;
else
actualadd=addr+start;
fprintf(fp2,"%d\t%d%d\n",address,opcode,actualadd);
address+=3;
}
fscanf(fp1,"%s",input);
}
}
fclose(fp1);
fclose(fp2);
printf("FINISHED");
}
```

INPUT : RELINPUT.DAT

```
H 1000 200
T 1000 11001 14 1033 48 1039 90 1776 92 1765 57 1765T 2011 11110 23 1838 43 1979 89
1060 66 1849 99 1477
```

```
E 1000
```

OUTPUT :

```
Enter the actual starting address :4000
RELOUTPUT.DAT
4000 144033
4003 484039
4006 901776
4009 921765
4012 574765
5011 234838
5014 434979
5017 894060
5020 664849
5023 991477
```

To implement a single pass macro processor.

ALGORITHM

Source code in c program for perform Simple macro processor

```
# include <stdio.h>
# include <string.h>
# include <stdlib.h>
struct deftab
{char lab[10];
char opc[10];
char oper[10];
}d[10];
void main()
{
char label[10],opcode[10],operand[10],newlabel[10],newoperand[10];
char macroname[10];
int i,lines;
FILE *f1,*f2,*f3;
f1=fopen("macin.dat","r");
f2=fopen("macout.dat","w");
f3=fopen("deftab.dat","w");
fscanf(f1,"%s%s%s",label,opcode,operand);
while(strcmp(opcode,"END")!=0)
{
if(strcmp(opcode,"MACRO")==0)
{
strcpy(macroname,label);
fscanf(f1,"%s%s%s",label,opcode,operand);
lines=0;
while(strcmp(opcode,"MEND")!=0)
{
fprintf(f3,"%s\t%s\t%s\n",label,opcode,operand);
strcpy(d[lines].lab,label);
strcpy(d[lines].opc,opcode);
strcpy(d[lines].oper,operand);
fscanf(f1,"%s%s%s",label,opcode,operand);
lines++;
}
}
else if(strcmp(opcode,macroname)==0)
{
printf("Lines = %d\n",lines);
for(i=0;i<lines;i++)
{
fprintf(f2,"%s\t%s\t%s\n",d[i].lab,d[i].opc,d[i].oper);
printf("DLAB = %s\nDOPC = %s\nDOPER = %s\n",d[i].lab,d[i].opc,d[i].oper);
}
}
else
fprintf(f2,"%s\t%s\t%s\n",label,opcode,operand);
fscanf(f1,"%s%s%s",label,opcode,operand);
}
fprintf(f2,"%s\t%s\t%s\n",label,opcode,operand);fclose(f1);
fclose(f2);
fclose(f3);
printf("FINISHED");
}
```

INPUT FILE :

MACIN.DAT

```
CALC START 1000
SUM MACRO **
** LDA #5
** ADD #10
** STA 2000
** MEND **
** LDA LENGTH
** COMP ZERO
** JEQ LOOP
** SUM **
LENGTH WORD 5
ZERO WORD 0
LOOP SUM **
** END **
```

OUTPUT FILES :

MACOUT.DAT

```
CALC START 1000
** LDA LENGTH
** COMP ZERO
** JEQ LOOP
** LDA #5
** ADD #10
** STA 2000
LENGTH WORD 5
ZERO WORD 0
** LDA #5
** ADD #10
** STA 2000
** END **
```

DEFTAB.DAT

```
** LDA #5
** ADD #10
** STA 2000
```

Part B…..
Lex Part/…….

Develop a LEX Program to count the number of characters, words, spaces and lines in a given input

```
%{
#include<stdio.h>
int lines=0, words=0, characters=0, num=0,spaces=0, spl_char=0;
%}
%%
\n { lines++; words++;}
[' '] {words++; spaces++; }
[A-Za-z0-9] characters++;
. spl_char++;
%%
main(void)
{
yyin= fopen("myfile.txt","r");
yylex();
printf(" This File contains ...");
printf("\n\t%d lines", lines);
printf("\n\t%d words",words);
printf("\n\t%d characters", characters);
printf("\n\t%d spaces", spaces);
printf("\n\t%d special characters\n",spl_char);
}
int yywrap()
{
return(1);
}
```

MYFILE.TXT

```
This is my 1st lex program....
Cheers!!!!! Itss woks
```

```
output:
shravan@ubuntu:~/ss_final$ gedit myfile.txt
shravan@ubuntu:~/ss_final$ lex lexp1.l
shravan@ubuntu:~/ss_final$ ./a.out
 This File contains ...
        2 lines
        9 words
        35 characters
        7 spaces
        9 special characters
```

Program to count the numbers of comment lines in a given C program. Also eliminate them and copy the resulting program into separate file.

```
%{
#include<stdio.h>
int com=0;
%}
%%
"/*"[^\n]+"*/" {com++;fprintf(yyout, " ");}
%%
int main()
{
printf("Write a C program\n");
yyout=fopen("output", "w");
yylex();
printf("Comment=%d\n",com);
return 0;
}
```

Ouput:->

```
shravan@ubuntu:~/ss_final/lex$ gedit lexp2.l
shravan@ubuntu:~/ss_final/lex$ cc lex.yy.c -ll
shravan@ubuntu:~/ss_final/lex$ ./a.out
Write a C program
#include<stdio.h>
```

```
int main()
{
int a,b;
/*float c;*/
printf("haaiiiii");
/*hjdhs*/
}
------------[press ctrl+d]-----------------------
Comment=2
shravan@ubuntu:~/ss_final/lex$ cat output
#include<stdio.h>
int main()
{
int a,b;

printf("haaiiiii");

}
```

Program to recognize a valid arithmetic expression and to recognize the identifiers and operators present

```
%{
#include<stdio.h>
int a=0,s=0,m=0,d=0,ob=0,cb=0;
int flaga=0, flags=0, flagm=0, flagd=0;
%}
id [a-zA-Z]+
%%
{id} {printf("\n %s is an identifier\n",yytext);}
[+] {a++;flaga=1;}
[-] {s++;flags=1;}
[*] {m++;flagm=1;}
[/] {d++;flagd=1;}
[(] {ob++;}
[)] {cb++;}
%%
int main()
{
```

```
printf("Enter the expression\n");
yylex();
if(ob-cb==0)
{
printf("Valid expression\n");
}
else
{
printf("Invalid expression");
}
printf("\nAdd=%d\nSub=%d\nMul=%d\nDiv=%d\n",a,s,m,d);
printf("Operators are: \n");
if(flaga)
printf("+\n");
if(flags)
printf("-\n");
if(flagm)
printf("*\n");
if(flagd)
printf("/\n");
return 0;
}
```

Output:->

```
$lex p2a.l
$cc lex.yy.c –ll
$./a.out
Enter the expression
(a+b*c)
a is an identifier
b is an identifier
c is an identifier
–----------------------[press Ctrl-d]-------------------------
Valid expression
Add=1
Sub=0
Mul=1
Div=0
Operators are:
+
*
```

Develop a lex program to recognize and count the number of identifiers in a given input files.

```
%{
#include<stdio.h>
int id=0;
%}

%%
[a-zA-Z][a-zA-Z0-9_]* { id++ ; ECHO; printf("\n");}
.+ { ;}
\n { ;}
%%

int yywrap()
{
return 1;
}
main (int argc, char *argv[])
{
if(argc!=2)
{
printf("Usage: <./a.out> <sourcefile>\n");
exit(0);
}
yyin=fopen(argv[1],"r");
printf("Valid identifires are\n");
yylex();
printf("No of identifiers = %d\n",id);
}
```

INPUT.TXT

```
int
float
78f
90gh
a
d
are
default
printf
```

output:

```
shravan@ubuntu:~/ss_final/lex$ gedit lexp4.l
shravan@ubuntu:~/ss_final/lex$ gedit input.txt
shravan@ubuntu:~/ss_final/lex$ lex lexp4.l
shravan@ubuntu:~/ss_final/lex$ cc lex.yy.c -ll
shravan@ubuntu:~/ss_final/lex$ ./a.out input.txt
Valid identifires are
int
float
a
d
are
default
printf
No of identifiers = 7
```

Part C: Yacc

Develop a yacc program to recognize a valid arithmetic expression that uses operators +,-,* and /.

<u>Lex File:</u>

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
%%
[a-zA-Z0-9]+  {         return ID;              }
.             {         return yytext[0];       }
%%
int yywrap(void)
{
return 1;
}
```

<u>Yacc File:</u>

```
%{
#include<stdio.h>
%}
%token ID
%left '+' '-' '*' '/'
%%
E:E'+'E
|E '-' E
|E '*' E
|E '/' E
|ID ;
%%

main()
{
printf("Enter an arithmetic expression\n");
yyparse();
printf("Valid expression\n");
}
yyerror()
{
printf("Invalid expression\n");
exit(0);
}
```

<u>Execution and Output:</u>

```
shravan@ubuntu:~/ss_final/yacc$ gedit lex1.l
shravan@ubuntu:~/ss_final/yacc$ gedit yacc1.y
shravan@ubuntu:~/ss_final/yacc$ lex lex1.l
shravan@ubuntu:~/ss_final/yacc$ yacc -d yacc1.y
shravan@ubuntu:~/ss_final/yacc$ cc lex.yy.c y.tab.c -ll
shravan@ubuntu:~/ss_final/yacc$ ./a.out
Enter an arithmetic expression
a+b-c

Valid expression
```

Program to recognize a valid variable, which starts with a letter, followed by any number of letters or digits.

<u>Lex File:</u>

```
%{
#include "y.tab.h"
%}
%%
[a-zA-Z]+       {       return LETTER;          }
[0-9]*          {       return DIGIT;           }
.               {       return yytext[0];       }
%%
int yywrap(void)
{
return 1;
}
```

<u>Yacc File:</u>

```
%{
#include<stdio.h>
%}
%token LETTER DIGIT
%%
S:LETTER
| S DIGIT
| S LETTER;
%%

main()
{
printf("Enter a variable\n");
yyparse();
printf("Valid Variable\n");
}
yyerror()
{
```

```
printf("Invalid variable\n");
exit(0);
}
```

Execution and Output:

```
$ lex prog4b.l
$ yacc –d prog4b.y
$ cc lex.yy.c y.tab.c –ll
$ ./a.out

shravan@ubuntu:~/ss_final/yacc$ ./a.out
Enter a variable
dhgtvhjhg675

Valid Variable
shravan@ubuntu:~/ss_final/yacc$ ./a.out
Enter a variable
5767gfvhh
Invalid variable
```

Develop a yacc program to evaluate an arithmetic expression involving operations +,-,*,/.

lex:

```
%{
#include<stdio.h>
#include"y.tab.h"
extern int yylval;
%}
%%
[0-9]+ {
yylval=atoi(yytext);
return NUM;
}
[\t] ;
```

```
\n return 0;
. return yytext[0];
%%
int yywrap(void)
{
return 1;
}
```

yacc:

```
%{
#include<stdio.h>
%}
%token NUM
%left '+' '-'
%left '*' '/'
%left '(' ')'
%%
expr:e{
printf("result:%d\n",$$);
return 0;
}
e:e'+'e {$$=$1+$3;}
|e'-'e {$$=$1-$3;}
|e'*'e {$$=$1*$3;}
|e'/'e {$$=$1/$3;}
|'('e')' {$$=$2;}
| NUM {$$=$1;}
;
%%

main()
{
printf("\n enter the arithematic expression:\n");
yyparse();
printf("\nvalid expression\n");
}
yyerror()
{
printf("\n invalid expression\n");
exit(0);
}
```

output:

```
shravan@ubuntu:~/ss_final/yacc$ gedit yacc3.y
shravan@ubuntu:~/ss_final/yacc$ gedit lex3.l
shravan@ubuntu:~/ss_final/yacc$ lex lex3.l
shravan@ubuntu:~/ss_final/yacc$ yacc -d yacc3.y
shravan@ubuntu:~/ss_final/yacc$ cc lex.yy.c y.tab.c -ll
shravan@ubuntu:~/ss_final/yacc$ ./a.out

 enter the arithematic expression:
5+6
result:11

valid expression
```

Develop a yacc program to recognize the strings of the form $a^m b^n$, where m>=0,n>=0.

Lex:

```
%{
#include "y.tab.h"
%}
%%
a return A;
b return B;
.|\n return yytext[0];
%%
int yywrap(void)
{
return 1;
}
```

----------yacc------------

```
%{
#include<stdio.h>
int valid=1;
%}
%token A B
%%
str:S'\n' {return 0;}
S:A S B
|
;
%%
main()
{
printf("Enter the string:\n");
yyparse();
if(valid==1)
printf("\nvalid string");
}
int yyerror()
{
printf("Invalid String\n");
exit(0);
}
```

output:

```
shravan@ubuntu:~/ss_final/yacc$ gedit yacc4.y
shravan@ubuntu:~/ss_final/yacc$ lex lex4.l
shravan@ubuntu:~/ss_final/yacc$ yacc -d yacc4.y
shravan@ubuntu:~/ss_final/yacc$ cc lex.yy.c y.tab.c -ll
shravan@ubuntu:~/ss_final/yacc$ ./a.out
Enter the string:
aaaaabb
Invalid String
shravan@ubuntu:~/ss_final/yacc$ ./a.out
Enter the string:
aaabbb
valid string
```

Extraaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

1 . PASS 1

```
#include<stdio.h>
#include<string.h>
void main()
{
char label[10],operand[10],opcode[10],code[10];
int start,locctr,length;
FILE *fp1,*fp2,*fp3,*fp4;
fp1 = fopen("input.dat","r");
fp2 = fopen("symtab.dat","w");
fp3 = fopen("out.dat","w");
fp4 = fopen("optab.dat","r");
fscanf(fp1,"%s%s%s",label,opcode,operand);
if(strcmp(opcode,"START")==0){
 start = atoi(operand);
 locctr = start;
 fprintf(fp3,"%s\t%s\t%s\n",label,opcode,operand);
 fscanf(fp1,"%s%s%s",label,opcode,operand);
                         }
else
 locctr = 0;
 while(strcmp(opcode,"END")!=0){
  fprintf(fp3,"%d\t",locctr);
  if(strcmp(label,"**")!=0)
   fprintf(fp2,"%s\t%d\n",label,locctr);
   rewind(fp4);
   fscanf(fp4,"%s",code);
   while(strcmp(code,"END")!=0){
     if(strcmp(opcode,code)==0){
      locctr = locctr + 3;
      break;
                               }
     fscanf(fp4,"%s",code);
                                                                                                                                    }
    if(strcmp(opcode,"BYTE")==0)
      locctr = locctr + 1;
    else if(strcmp(opcode,"WORD")==0)
      locctr = locctr + 3;
    else if(strcmp(opcode,"RESW")==0)
```

```
      locctr = locctr + 3*atoi(operand);
    else if(strcmp(opcode,"RESB")==0)
      locctr = locctr + atoi(operand);
    fprintf(fp3,"%s\t%s\t%s\n",label,opcode,operand);
    fscanf(fp1,"%s%s%s",label,opcode,operand);
                               }
    fprintf(fp3,"%d\t%s\t%s\t%s\n",locctr,label,opcode,operand);
    printf("%d\n",locctr-start);
    fclose(fp4);
    fclose(fp3);
    fclose(fp2);
    fclose(fp1);
}
```

2. PASS 2

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void main()
  {
    char a[10],ad[10],label[10],opcode[10],operand[10],mnemonic[10],symbol[10],c,d,temp[10];
    int i,address,code,add,len,actual_len,u,v,start,length=0;
    char obj[100][100];
    FILE *fp1,*fp2,*fp3,*fp4,*fp5;
    fp1=fopen("assmlist.dat","w");
    fp2=fopen("symtab.dat","r");
    fp3=fopen("intermediate.dat","r");
    fp4=fopen("optab.dat","r");
    fp5=fopen("objectcode.dat","w");
    fscanf(fp3,"%s%s%s",label,opcode,operand);
    if(strcmp(opcode,"START")==0)
    {
     fprintf(fp1,"%s\t%s\t%s\n",label,opcode,operand);
     strcpy(obj[0],label);
     start = atoi(operand);
     strcpy(obj[1],operand);
     fscanf(fp3,"%x%s%s%s",&address,label,opcode,operand);
    }
    else
    {
     strcpy(obj[0],"**");
     strcpy(obj[1],"000000");
    }
    strcpy(obj[2],"null");
    int p=3;
```

```
while(strcmp(opcode,"END")!=0)
{
if(strcmp(opcode,"BYTE")==0)
{
  fprintf(fp1,"%x\t%s\t%s\t%s\t",address,label,opcode,operand);
len = strlen(operand);
for(i=2;i<len-1;i++)
{
fprintf(fp1,"%x",operand[i]);
sprintf(temp,"%x",operand[i]);
strcat(obj[p],temp);
length++;
}
fprintf(fp1,"\n");
p++;
}
else if(strcmp(opcode,"WORD")==0)
{
fprintf(fp1,"%x\t%s\t%s\t%s\t00000%s\n",address,label,opcode,operand,operand);
strcpy(obj[p],"00000");
strcat(obj[p],operand);
p++;
length=length+3;
}
else if((strcmp(opcode,"RESB")==0) || (strcmp(opcode,"RESW")==0))
{
fprintf(fp1,"%x\t%s\t%s\t%s\n",address,label,opcode,operand);
}
else
{
rewind(fp4);
fscanf(fp4,"%s%x",mnemonic,&u);
while(strcmp(opcode,mnemonic)!=0)
fscanf(fp4,"%s%x",mnemonic,&u);
rewind(fp2);
fscanf(fp2,"%s%x",mnemonic,&v);
while(strcmp(operand,mnemonic)!=0)
fscanf(fp2,"%s%x",mnemonic,&v);
fprintf(fp1,"%x\t%s\t%s\t%s\t%x%x\n",address,label,opcode,operand,u,v);
 sprintf(obj[p],"%x%x",u,v);
p++;
length=length+3;
}
fscanf(fp3,"%x%s%s%s",&address,label,opcode,operand);
}
fprintf(fp1,"%x\t%s\t%s\t%s\n",address,label,opcode,operand);
```

```
   sprintf(obj[2],"%x",address);
   start = atoi(obj[2]) - start;
   sprintf(obj[2],"%d",start);
   //print header record
   fprintf(fp5,"H %s %s %s\n",obj[0],obj[1],obj[2]);
   //text record
   fprintf(fp5,"T %s %x ",obj[1],length);
   for(i=3;i<p;i++)
     fprintf(fp5,"%s ",obj[i]);
   fprintf(fp5,"\nE %s",obj[1]);
   fcloseall();
  }
```

3. ABSOLUTE LOADER

```
#define _GNU_SOURCE
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

void main()
{
 char ch,label[10],input[10];
 int start,length,address,l;
 FILE *fp1,*fp2;

 fp1=fopen("input.dat","r");
 fp2=fopen("out.dat","w");

 fscanf(fp1,"%s",input);

 while(strcmp(input,"E")!=0)
 {
        if(strcmp(input,"H")==0)
        {
                fscanf(fp1,"%s%x%x%s",label,&start,&length,input);
        }
        else if(strcmp(input,"T")==0)
        {
                fscanf(fp1,"%x%x%s",&address,&length,input);
                fprintf(fp2,"%x\t%c%c\n",address,input[0],input[1]);
                fprintf(fp2,"%x\t%c%c\n",(address+1),input[2],input[3]);
                fprintf(fp2,"%x\t%c%c\n",(address+2),input[4],input[5]);
                address+=3;
                fscanf(fp1,"%s",input);
```

```
		}
		else
		{
			l=strlen(input);
			if(l==6)
			{
			fprintf(fp2,"%x\t%c%c\n",address,input[0],input[1]);
			fprintf(fp2,"%x\t%c%c\n",(address+1),input[2],input[3]);
			fprintf(fp2,"%x\t%c%c\n",(address+2),input[4],input[5]);
			address+=3;
			}
			else if(l==4)
			{
			fprintf(fp2,"%x\t%c%c\n",address,input[0],input[1]);
			fprintf(fp2,"%x\t%c%c\n",(address+1),input[2],input[3]);
			address+=2;
			}
			else if(l==2)
			{
			fprintf(fp2,"%x\t%c%c\n",address,input[0],input[1]);
			//fprintf(fp2,"%x\t%c%c\n",(address+1),input[2],input[3]);
			address+=1;
			}
			fscanf(fp1,"%s",input);
		}
 }

fcloseall();

fp2=fopen("out.dat","r");
ch=fgetc(fp2);
while(ch!=EOF)
{
		printf("%c",ch);
		ch=fgetc(fp2);
}
}
```

4 . RELOCATING LOADER

```
#include<stdio.h>
   #include<string.h>
```

```
#include<stdlib.h>
void convert(char h[12]);
char bitmask[12];
char bit[12]={0};
void main()
{char pn[5],add[6],length[10],input[10],relocbit,ch;
int start,i,address,tlen,len,opcode,addr,actualadd;
FILE *fp1,*fp2;

printf("\n\n Enter the actual starting address : ");
scanf("%x",&start);
fp1=fopen("RLIN.txt","r");
fp2=fopen("RLOUT.txt","w");
fscanf(fp1,"%s",input);
fprintf(fp2," ADDRESS\tCONTENT\n");

while(strcmp(input,"E")!=0)
{
if(strcmp(input,"H")==0)
{
fscanf(fp1,"%s%x%x%s",pn,add,length,input);
}

if(strcmp(input,"T")==0)
{
                        fscanf(fp1,"%x%x%s",&address,&tlen,bitmask);
                        address+=start;
                        convert(bitmask);
                        len=strlen(bit);
                        if(len>=11)
                        len=10;

for(i=0;i<len;i++)
{
fscanf(fp1,"%x%x",&opcode,&addr);
relocbit=bit[i];
if(relocbit=='0')
actualadd=addr;
else
actualadd=addr+start;
fprintf(fp2,"\n  %x\t\t%x%x\n",address,opcode,actualadd);
address+=3;
}
fscanf(fp1,"%s",input);
}
}
```

```
 fcloseall();

 printf("\n\n The contents of INPUT file(RLIN.TXT)\n\n");
 fp1=fopen("RLIN.txt","r");
 ch=fgetc(fp1);
 while(ch!=EOF)
 {
 printf("%c",ch);
 ch=fgetc(fp1);
 }
 fclose(fp1);


 printf("\n\n The contents of output file(RLOUT.TXT)\n\n");
 fp2=fopen("RLOUT.txt","r");
 ch=fgetc(fp2);
 while(ch!=EOF)
 {
 printf("%c",ch);
 ch=fgetc(fp2);
 }
 fclose(fp2);
 }


void convert(char h[12])
 {
 int i,l;
 strcpy(bit,"");
 l=strlen(h);
 for(i=0;i<l;i++)
 {
 switch(h[i])
 {
 case '0':
     strcat(bit,"0");
 break;
 case '1':
     strcat(bit,"1");
 break;
 case '2':
     strcat(bit,"10");
 break;
 case '3':
     strcat(bit,"11");
 break;
case '4':
    strcat(bit,"100");
break;
case '5':
  strcat(bit,"101");
break;
case '6':
   strcat(bit,"110");
break;
case '7':
    strcat(bit,"111");
break;
case '8':
    strcat(bit,"1000");
break;
case '9':
    strcat(bit,"1001");
break;
case 'A':
    strcat(bit,"1010");
break;
case 'B':
   strcat(bit,"1011");
break;
case 'C':
  strcat(bit,"1100");
break;
case 'D':
    strcat(bit,"1101");
break;
case 'E':
   strcat(bit,"1110");
break;
case 'F':
    strcat(bit,"1111");
break;
}
}
}
```

OR

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
 char input[10],bitmask[10],opcode[10];
 int start,length,address,actual,ad,i,a;
 printf("enter the actual starting aadress\n");
 scanf("%d",&actual);
 FILE *f1, *f2;
 f1 = fopen("relinput.dat","r");
 f2 = fopen("output.dat","w");
 fscanf(f1,"%s",input);
 while(strcmp(input,"E")!=0)
 {
		if(strcmp(input,"H")==0)
		{
			fscanf(f1,"%d",&start);
			fscanf(f1,"%d",&length);
			fscanf(f1,"%s",input);
		}
		else if(strcmp(input,"T")==0)
		{
			fscanf(f1,"%d",&ad);
			fscanf(f1,"%s",bitmask);
			ad = ad + actual-start;
			for(i=0;i<strlen(bitmask);i++)
			{
				fscanf(f1,"%s",opcode);
				fscanf(f1,"%d",&a);
				if(bitmask[i]=='1')
				   a = a + (actual-start);
				fprintf(f2,"%d\t%s%d\n",ad,opcode,a);
				ad = ad+3;
			}
			fscanf(f1,"%s",input);
		}
 }
 fclose(f1);
 fclose(f2);
}
```

5 .PASS 1 OF MACROS

```
 # include <stdio.h>
```

```
# include <string.h>
# include <stdlib.h>
void main()
{
char label[10],opcode[10],operand[10],newlabel[10],newoperand[10];
char macroname[10];
int i,lines;
FILE *f1,*f2,*f3;
f1=fopen("macin.dat","r");
f2=fopen("macout.dat","w");
f3=fopen("deftab.dat","w+");
fscanf(f1,"%s%s%s",label,opcode,operand);
while(strcmp(opcode,"END")!=0)
{
if(strcmp(opcode,"MACRO")==0)
{
strcpy(macroname,label);
fscanf(f1,"%s%s%s",label,opcode,operand);
            while(strcmp(opcode,"MEND")!=0)
            {
            fprintf(f3,"%s\t%s\t%s\n",label,opcode,operand);
            fscanf(f1,"%s%s%s",label,opcode,operand);
            }
            fprintf(f3,"%s\t%s\t%s\t",label,label,label);
}
else if(strcmp(opcode,macroname)==0)
{
 if(strcmp(label,"**")!=0)
   fprintf(f2,"%s\t",label);
 rewind(f3);
 fscanf(f3,"%s%s%s",label,opcode,operand);
 while(strcmp(opcode,"**")!=0)
 {
 fprintf(f2,"%s\t%s\t%s\n",label,opcode,operand);
 fscanf(f3,"%s%s%s",label,opcode,operand);
 }
}
else
fprintf(f2,"%s\t%s\t%s\n",label,opcode,operand);

fscanf(f1,"%s%s%s",label,opcode,operand);
}
fprintf(f2,"%s\t%s\t%s\n",label,opcode,operand);
fcloseall();
}
```