**EXPERIMENT NO. 1**

**AIM:** To implement pass 1 of 2 pass assembler

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

FILE \*ptr;

FILE \*preptr;

char ch;

int lc = 0;

int prev\_lc = 0;

char c;

int n = 0;

int i;

int count=0;

typedef struct{

char mnemonic[10];

// char opcode[10];

int length;

// char format[2];

}mot;

typedef struct{

char pseudo[20];

}pot;

typedef struct{

char symbol[20];

int lc\_val;

// int length;

char relocation;

}st;

typedef struct{

int reg;

int reg\_val;

}bt;

st symbol\_table[10];

mot m\_table[10];

pot p\_table[10];

bt b\_table[1];

void skipLine(){

do{

ch = fgetc(preptr);

i++;

count++;

}while(ch!='\n');

}

int findSymbol(char \*str){

int j=0;

for(j=0;j<5;j++){

if(strcmp(symbol\_table[j].symbol,str) == 0){

return symbol\_table[j].lc\_val;

}

}

}

int potSearch(char\* str){

int j=0;

// printf("%s\n",str);

for(j=0;j<7;j++){

if(strcmp(p\_table[j].pseudo,str) == 0){

if(strcmp(p\_table[j].pseudo,"DC") == 0 || strcmp(p\_table[j].pseudo,"DS") == 0){

// fscanf(ptr,"%s",str);

// while((ch = fgetc(preptr)) != ' '){

// i++;

// count++;

// if(ch != ' '){

// break;

// }

// }

ch = fgetc(preptr);

ch = fgetc(preptr);

i+=2;

count+=2;

// i--;

// // count--;

// printf("\n\*\*\* %c \*\*\*\*",ch);

// if((ch >= 'A' && ch <='Z')){

// if(ch == 'F'){

// lc += 4;

// }

// else{

// lc += 2;

// }

// // printf("\n\*\*\*\*\* %d \*\*\*\*",lc);

// }

if(ch == 'F'){

lc += 4;

// printf("\n\*\*\*\*\* %d \*\*\*\*",lc);

}

else if(ch == 'H'){

lc += 2;

}

else{

char prech = ch;

do{

ch = fgetc(preptr);

i++;

count++;

if(ch != ' '){

break;

}

}while(ch != ' ');

// i--;

// count--;

if(ch == 'F'){

lc += (prech - '0') \* 4;

}

else{

lc += (prech - '0') \* 2;

}

}

}

else if(strcmp(p\_table[j].pseudo,"EQU") == 0){

// fscanf(ptr,"%s",str);

// p = getNext();

// do{

// ch = fgetc(preptr);

// i++;

// count++;

// if(ch == ' '){

// break;

// }

// }while(ch != ' ');

ch = fgetc(preptr);

ch = fgetc(preptr);

i+=2;

count+=2;

// printf("\n\*\*\*\*\*\*\* %c \*\*\*\*\*",ch);

symbol\_table[n-1].lc\_val = ch - '0';

symbol\_table[n-1].relocation = 'A';

}

// printf("\*\*\*\*\*\*\*Found in pot\*\*\*\*\*\*\n");

return 1;

}

}

return 0;

}

int potFind(char \*str){

int j=0;

for(j=0;j<7;j++){

if(strcmp(p\_table[j].pseudo,str) == 0){

if(strcmp(p\_table[j].pseudo,"USING") == 0){

ch = fgetc(preptr);

ch = fgetc(preptr);

i+=2;

count+=2;

if(ch == '\*'){

b\_table[0].reg\_val = lc;

}

ch = fgetc(preptr);

i++;

count++;

char s[2];

fscanf(preptr,"%s",s);

i+=2;

count++;

b\_table[0].reg = atoi(s);

// printf("\*\*\*\*\*REG:%d\*\*\*\*\*\*",atoi(s));

printf("\nBT\n");

printf("BASE REG\tVALUE\n");

printf("%d\t\t%d\n",b\_table[0].reg,b\_table[0].reg\_val);

}

return 1;

}

}

return 0;

}

int motSearch(char str[]){

int j=0;

for(j=0;j<5;j++){

// printf("%s\n",m\_table[i].mnemonic);

if(strcmp(m\_table[j].mnemonic,str) == 0){

// prev\_lc = lc;

lc += m\_table[j].length;

// printf("LC: %s %d \n",str,lc);

// printf("\*\*\*\*\*found in mot\*\*\*\*\*\*\n");

return 1;

}

}

return 0;

}

int motFind(char str[]){

int offset;

int l;

char s[5];

int j=0;

for(j=0;j<5;j++){

// printf("%s\n",m\_table[i].mnemonic);

if(strcmp(m\_table[j].mnemonic,str) == 0){

// prev\_lc = lc;

lc += m\_table[j].length;

// printf("LC: %s %d \n",str,lc);

// printf("\*\*\*\*\*found in mot\*\*\*\*\*\*\n");

printf("\n%s ",str);

l = fscanf(preptr,"%s",s);

// printf("\*\*\*\*\*\*\*%d\*\*\*\*\*",l);

i+=strlen(s);

count++;

if(s[0] >= '0' && s[0] <= '9'){

printf("%s",s);

return 1;

}

offset = findSymbol(s);

printf("%d(0,%d)",offset,b\_table[0].reg);

i++;

ch = fgetc(preptr);

// printf("\n\*\*%c\*\*\n",ch);

if(ch != ' '){

return 1;

}

fscanf(preptr,"%s",s);

i+=strlen(s);

count++;

offset = findSymbol(s);

printf(" %d(0,%d)",offset,b\_table[0].reg);

return 1;

}

}

return 0;

}

void printTables(){

int j;

printf("\nPOT:\n");

printf("PSEUDO\n");

for(j=0;j<7;j++){

printf("%s\n",p\_table[j].pseudo);

}

printf("\nMOT:\n");

printf("MNEMONIC\tLENGTH\n");

for(j=0;j<5;j++){

printf("%s\t\t%d\n",m\_table[j].mnemonic,m\_table[j].length);

}

printf("\nST:\n");

printf("SYMBOL\tLC VAL\tRELOCATION\n");

for(j=0;j<n;j++){

printf("%s\t%d\t%c\n",symbol\_table[j].symbol,symbol\_table[j].lc\_val,symbol\_table[j].relocation);

}

}

void main(){

int length;

char \*buffer;

int val;

printf("PASS1: \n");

strcpy(p\_table[0].pseudo,"START");

strcpy(p\_table[1].pseudo,"BALR");

strcpy(p\_table[2].pseudo,"USING");

strcpy(p\_table[3].pseudo,"EQU");

strcpy(p\_table[4].pseudo,"DC");

strcpy(p\_table[5].pseudo,"DS");

strcpy(p\_table[6].pseudo,"END");

strcpy(m\_table[0].mnemonic,"L");

m\_table[0].length = 4;

strcpy(m\_table[1].mnemonic,"A");

m\_table[1].length = 4;

strcpy(m\_table[2].mnemonic,"ST");

m\_table[2].length = 4;

strcpy(m\_table[3].mnemonic,"BNE");

m\_table[3].length = 4;

strcpy(m\_table[4].mnemonic,"BR");

m\_table[4].length = 4;

ptr = fopen("p.txt","r");

preptr = fopen("p.txt","r");

if(ptr){

fseek(ptr,0,SEEK\_END);

length = ftell(ptr);

fseek(ptr,0,SEEK\_SET);

buffer = malloc(length);

if(buffer){

fread(buffer,1,length,ptr);

// printf("%s\n",buffer);

}

for(i=0;i<length;i++){

if((buffer[i] == ' ' && buffer[i+1] != ' ') || buffer[i] == '\n' || buffer[i] == '\'' || buffer[i] == '\t' || buffer[i] == ',' || buffer[i] == EOF){

char word[i-count+1];

fseek(preptr,count,SEEK\_SET);

if(i-count!=0){

fread(word,1,i-count,preptr);

word[i-count] = '\0';

if(isspace(word[0])== 0){

// printf("%s\n",word);

val = potSearch(word);

if(val == 1){

skipLine();

count--;

i--;

}

else{

val = motSearch(word);

if(val == 1){

skipLine();

count--;

i--;

}

else{

// printf("\*\*\*\*\*SYM\*\*\*\n");

strcpy(symbol\_table[n].symbol,word);

symbol\_table[n].lc\_val = lc;

symbol\_table[n].relocation = 'R';

n++;

}

}

}

count = i+1;

}

}

}

}

printTables();

// printSt();

free(buffer);

fclose(ptr);

fclose(preptr);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\nPASS2: \n");

printTables();

lc = 0;

count = 0;

ptr = fopen("p.txt","r");

preptr = fopen("p.txt","r");

if(ptr){

fseek(ptr,0,SEEK\_END);

length = ftell(ptr);

fseek(ptr,0,SEEK\_SET);

buffer = malloc(length);

if(buffer){

fread(buffer,1,length,ptr);

// printf("%s\n",buffer);

}

for(i=0;i<length;i++){

if((buffer[i] == ' ' && buffer[i+1] != ' ') || buffer[i] == '\n' || buffer[i] == '\'' || buffer[i] == '\t' || buffer[i] == ',' || buffer[i] == EOF){

char word[i-count+1];

fseek(preptr,count,SEEK\_SET);

if(i-count!=0){

fread(word,1,i-count,preptr);

word[i-count] = '\0';

if(isspace(word[0])== 0){

// printf("%s\n",word);

val = potFind(word);

if(val == 1){

skipLine();

count--;

i--;

}

else{

val = motFind(word);

if(val == 1){

// skipLine();

// printInst();

count--;

i--;

}

// else{

// // printf("\*\*\*\*\*SYM\*\*\*\n");

// strcpy(symbol\_table[n].symbol,word);

// symbol\_table[n].lc\_val = lc;

// symbol\_table[n].relocation = 'R';

// n++;

// }

}

}

count = i+1;

}

}

}

}

free(buffer);

fclose(ptr);

fclose(preptr);

printf("\n");

}

**OUTPUT:**

PASS1:

POT:

PSEUDO

START

BALR

USING

EQU

DC

DS

END

MOT:

MNEMONIC LENGTH

L 4

A 4

ST 4

BNE 4

BR 4

ST:

SYMBOL LC VAL RELOCATION

SIMPLE 0 R

LOOP 0 R

R1 1 A

TWO 20 R

FOUR 24 R

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PASS2:

POT:

PSEUDO

START

BALR

USING

EQU

DC

DS

END

MOT:

MNEMONIC LENGTH

L 4

A 4

ST 4

BNE 4

BR 4

ST:

SYMBOL LC VAL RELOCATION

SIMPLE 0 R

LOOP 0 R

R1 1 A

TWO 20 R

FOUR 24 R

BT

BASE REG VALUE

15 0

L 1(0,15) 20(0,15)

A 1(0,15) 20(0,15)

ST 1(0,15) 24(0,15)

BNE 0(0,15)

BR 14