**EXP NO:1 DESIGN OF TWO PASS ASSEMBLER**

**AIM:**

To write a C program to implement the design of two pass assembler.

**ALGORITHM:**

**FIRST PASS OF TWO PASS ASSEMBLER**:

1.loc\_cntr : =0; (default value)

Pooltab\_ptr : =1; POOLTAB[1] : =1;

Littab\_ptr : =1;

2.while next statement is not an END statement

a) if label is present then

this\_label : =symbol in label field;

enter(this label,loc\_cntr)in symtab

b)if a START or ORIGIN statement then loc\_cntr : =value specified in operand field;

c)if a declaration then

i)code : =code of the declaration statement;

ii)size : =size of memory area required by DC/DS

iii)loc\_cntr : =loc\_cntr+size;

iv)Gerenate ‘IC (DL,code)…..’

d)If an imperative statement then

i)code : =machine opcode from OPTAB

ii)loc\_cntr : =loc\_cntr+instruction length from OPTAB

iii)if(i.e, operand is a symbol)

this entry : =SYMTAB entry no of operand.

Generate lc”(Is,code)(s,this\_entry);

3. processing of END statement

a)perform step 2 (b)

b)Generate ‘IC(AD,02)’

c)Goto pass two

**TWO PASS OF TWO PASS ASSEMBLER :**

1.code\_area\_address : =address of code\_area;

Pooltab\_ptr : =1;

Loc\_cntr : =0;

2.while next statement is not an END statement

a) clear machine\_code\_buffer;

b) If a START or ORIGIN statement then

i) loc\_cntr : =value specified in operand field;

ii)size : =0;

c)if a declaration statement

i)If a DC statement then

Assemble the constant in machine\_code\_buffer.

ii)size : =size of memory area required by DC/DS;

d)If an imperative statement

i)Get operand address from SYMTAB

ii)Assemble instruction in machine\_code\_buffer

iii)size : =size of instruction;

e)If size!=0 then

i)Move contents of machine-code\_buffer to the address code\_area\_address+loc\_cntr;

3.(processing of END statement)

a) write code\_area into output file

**PROGRAM:**

**First pass of two pass assembler:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

struct menumonic\_table

{

char op\_code[7];

char class1[3];

char info[3];

}nem[20];

int checkreg(char var1[])

{

if(strcmp(var1,"AREG")==0)

return 1;

if(strcmp(var1,"BREG")==0)

return 2;

if(strcmp(var1,"CREG")==0)

return 3;

else

return 0;

}

char symb[6][10];

int add[6];

int checksym(char var[])

{

static int i=1;

int j;

for(j=1;j<=i;j++)

{

//printf("%s",sym[j]);

if(strcmp(symb[j],var)==0)

return j;

}

strcpy(symb[i],var);

i++;

return 0;

}

int search(char var2[])

{

int i;

for(i=0;i<19;i++)

{

if(strcmp(var2,nem[i].op\_code)==0)

return i;

}

return 20;

}

main()

{

int i;

strcpy(nem[0].op\_code,"ADD");

strcpy(nem[0].class1,"IS");

strcpy(nem[0].info,"01");

strcpy(nem[1].op\_code,"SUB");

strcpy(nem[1].class1,"IS");

strcpy(nem[1].info,"02");

strcpy(nem[2].op\_code,"MULT");

strcpy(nem[2].class1,"IS");

strcpy(nem[2].info,"03");

strcpy(nem[3].op\_code,"MOVER");

strcpy(nem[3].class1,"IS");

strcpy(nem[3].info,"04");

strcpy(nem[4].op\_code,"MOVEM");

strcpy(nem[4].class1,"IS");

strcpy(nem[4].info,"05");

strcpy(nem[5].op\_code,"COMP");

strcpy(nem[5].class1,"IS");

strcpy(nem[5].info,"06");

strcpy(nem[6].op\_code,"STOP");

strcpy(nem[6].class1,"IS");

strcpy(nem[6].info,"00");

strcpy(nem[7].op\_code,"BC");

strcpy(nem[7].class1,"IS");

strcpy(nem[7].info,"07");

strcpy(nem[8].op\_code,"DIV");

strcpy(nem[8].class1,"IS");

strcpy(nem[8].info,"08");

strcpy(nem[9].op\_code,"READ");

strcpy(nem[9].class1,"IS");

strcpy(nem[9].info,"09");

strcpy(nem[10].op\_code,"PRINT");

strcpy(nem[10].class1,"IS");

strcpy(nem[10].info,"10");

strcpy(nem[11].op\_code,"START");

strcpy(nem[11].class1,"AD");

strcpy(nem[11].info,"01");

strcpy(nem[12].op\_code,"END");

strcpy(nem[12].class1,"AD");

strcpy(nem[12].info,"02");

strcpy(nem[13].op\_code,"LTROG");

strcpy(nem[13].class1,"AD");

strcpy(nem[13].info,"03");

strcpy(nem[14].op\_code,"EQU");

strcpy(nem[14].class1,"AD");

strcpy(nem[14].info,"04");

strcpy(nem[15].op\_code,"ORIGIN");

strcpy(nem[15].class1,"AD");

strcpy(nem[15].info,"05");

strcpy(nem[16].op\_code,"DS");

strcpy(nem[16].class1,"DL");

strcpy(nem[16].info,"02");

strcpy(nem[17].op\_code,"DC");

strcpy(nem[17].class1,"DL");

strcpy(nem[17].info,"01");

strcpy(nem[18].op\_code,"LE");

strcpy(nem[18].class1,"CS");

strcpy(nem[18].info,"02");

FILE \*f1,\*f2,\*f3;

int j=0,k=0,lc,sy,no,sym=0,reg,num;

char var[256],var1[4][10];

char delim[] = " ";

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

f2=fopen("output.txt","w");

f3=fopen("symtab.txt","w");

while(k<18)

{

fgets(var,sizeof(var),f1);

i=0;

char \*ptr = strtok(var, delim);

while(ptr != NULL)

{

strcpy(var1[i],ptr);

// printf("%s\n", var1[i]);

ptr = strtok(NULL, delim);

i++;

}

j=i;

strtok(var1+0,"\n");

//for(i=0;i<j;i++)

// printf("%s\n",var1+i);

no=search(var1+0);

//printf("%d\n",no);

if(strcmp(var1+0,"START")==0)

{

lc=atoi(var1+1);

printf("(%s,%s)(c,%d)\n",nem[no].class1,nem[no].info,lc);

fprintf(f2,"%s %s c %d\n",nem[no].class1,nem[no].info,lc);

}

if(strcmp(nem[no].info,"09")==0)

{

lc+=1;

sym+=1;

checksym(var1+1);

printf("(%s,%s) (0) (s,%d)\n",nem[no].class1,nem[no].info,sym);

fprintf(f2,"%s %s 0 s %d\n",nem[no].class1,nem[no].info,sym);

}

if(strcmp(nem[no].info,"04")==0||strcmp(nem[no].info,"05")==0||strcmp(nem[no].info,"06")==0)

{

lc+=1;

sy=checksym(var1+2);

if(sy==0)

{

sym+=1;

sy=sym;

}

reg=checkreg(var1+1);

printf("(%s,%s) (%d) (s,%d)\n",nem[no].class1,nem[no].info,reg,sy);

fprintf(f2,"%s %s %d s %d\n",nem[no].class1,nem[no].info,reg,sy);

}

if(no==20)

{

int ad;

lc+=1;

no=search(var1+1);

strcat(var1+0,"\n");

sy=checksym(var1+0);

if(sy==0)

{

sym+=1;

sy=sym;

}

sy=checksym(var1+3);

if(sy==0)

{

sym+=1;

sy=sym;

}

ad=checksym(var1+0);

add[ad]=lc-1;

reg=checkreg(var1+2);

if(strcmp(nem[no].class1,"DL")==0)

{

sy=atoi(var1+2);

ad=checksym(var1+0);

add[ad]=lc-1;

printf("(%s,%s) (c,%d) \n",nem[no].class1,nem[no].info,sy);

fprintf(f2,"%s %s c %d\n",nem[no].class1,nem[no].info,sy);

}

else

{

printf("(%s,%s) (%d) (s,%d)\n",nem[no].class1,nem[no].info,reg,sy);

fprintf(f2,"%s %s %d s %d\n",nem[no].class1,nem[no].info,reg,sy);

}

}

if(strcmp(var1+0,"STOP")==0)

{

lc+=1;

printf("(%s,%s) \n",nem[no].class1,nem[no].info);

fprintf(f2,"%s %s\n",nem[no].class1,nem[no].info);

}

if(strcmp(nem[no].info,"01")==0&&strcmp(nem[no].class1,"IS")==0)

{

lc+=1;

sy=checksym(var1+2);

if(sy==0)

{

sym+=1;

sy=sym;

}

reg=checkreg(var1+1);

printf("(%s,%s) (%d) (s,%d)\n",nem[no].class1,nem[no].info,reg,sy);

fprintf(f2,"%s %s %d s %d\n",nem[no].class1,nem[no].info,reg,sy);

}

if(strcmp(nem[no].info,"07")==0)

{

lc+=1;

sy=checksym(var1+2);

if(sy==0)

{

sym+=1;

sy=sym;

}

num=search(var1+1);

printf("(%s,%s) (%s) (s,%d)\n",nem[no].class1,nem[no].info,nem[num].info,sy);

fprintf(f2,"%s %s %s s %d\n",nem[no].class1,nem[no].info,nem[num].info,sy);

}

if(strcmp(nem[no].info,"10")==0)

{

lc+=1;

sy=checksym(var1+1);

if(sy==0)

{

sym+=1;

sy=sym;

}

printf("(%s,%s) (0) (s,%d)\n",nem[no].class1,nem[no].info,sy);

fprintf(f2,"%s %s 0 s %d\n",nem[no].class1,nem[no].info,sy);

}

if(strcmp(var1+0,"END")==0)

{

printf("(%s,%s) \n",nem[no].class1,nem[no].info);

fprintf(f2,"%s %s\n",nem[no].class1,nem[no].info);

}

k++;

}

for(i=1;i<6;i++)

{

strtok(symb[i],"\n");

fprintf(f3,"%s\t %d\n",symb[i],add[i]);

}

}

**Second pass of two pass assembler :**

#include<stdio.h>

#include<conio.h>

#include<string.h>

main()

{

int i,l;

FILE \*f1,\*f2,\*f3;

int j=0,k=0,lc,sy,no,sym=0,reg,num,vari;

char var[256],var1[4][10],symb[10][10],add[10][10];

char delim[] = " ";

f1=fopen("output.txt","r");

f3=fopen("symtab.txt","r");

for(l=1;l<6;l++)

{

fscanf(f3,"%s%s",symb[l],add[l]);

}

while(k<18)

{

fgets(var,sizeof(var),f1);

i=0;

char \*ptr = strtok(var, delim);

while(ptr != NULL)

{

strcpy(var1[i],ptr);

// printf("%s\n", var1[i]);

ptr = strtok(NULL, delim);

i++;

}

j=i;

strtok(var1+0,"\n");

//for(i=0;i<j;i++)

// printf("%s\n",var1+i);

if(strcmp(var1+0,"AD")==0&&strcmp(var1+1,"01")==0)

{

lc=atoi(var1+3);

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

}

if(strcmp(var1+0,"IS")==0)

{

vari=atoi(var1+4);

strtok(var1+1,"\n");

if(strcmp(var1+1,"00")==0)

{

printf("%d 00 0 000 \n",lc);

}

else

printf("%d %s %s %s\n",lc,var1+1,var1+2,add[vari]);

lc+=1;

}

if(strcmp(var1+0,"DL")==0)

{

if(strcmp(var1+1,"02")==0)

{

lc+=1;

printf("%d Mermory allocation\n",lc);

}

if(strcmp(var1+1,"01")==0)

{

printf("%d 00 0 001",lc);

lc+=1;

}

}

k++;

}

}

**INPUT:**

**input.txt:**

START 101

READ N

MOVER BREG ONE

MOVEM BREG TERM

AGAIN MULT BREG TERM

MOVER CREG TERM

ADD CREG ONE

MOVEM CREG TERM

COMP CREG N

BC LE AGAIN

MOVEM BREG RESULT

PRINT RESULT

STOP

N DS 1

RESULT DS 1

ONE DC 1

TERM DS 1

END

**OUTPUT:**

**output.txt:**

AD 01 C 101

IS 04 0 S 2

IS 04 2 S 3

IS 05 2 S 4

IS 03 2 S 4

IS 04 3 S 3

IS 01 3 S 3

IS 05 3 S 4

IS 06 3 S 2

IS 07 2 S 5

IS 05 2 S 6

**symtab.txt:**

N 113

ONE 115

TERM 116

AGAIN 104

RESULT 114

**Final output:**

101 9 0 113

102 4 2 115

103 5 2 116

104 3 2 116

105 4 3 116

106 1 3 115

107 5 3 116

108 6 3 113

109 7 2 104

110 5 2 114

111 10 0 114

112 0 0 000

113 Memory allocation

114 Memory allocation

115 0 0 001

116 Memory allocation

**RESULT:**

Thus,design of two pass assembler is executed and output is verified and displayed successfully