start 100

mover areg, ='1'

movem breg, a

add areg, b

mover areg, ='4'

ltorg

add creg, ='3'

mover areg, ='1'

b dc 2

a ds 3

end

MNEMONIC OPCODE TABLE:..........

--------------------------------------------------------

Assembler Instruction Instruction Opcode Length

stop 0 1

add 1 1

sub 2 1

mult 3 1

mover 4 1

movem 5 1

comp 6 1

bc 7 1

div 8 1

read 9 1

print 10 1

PSEUDO OPCODE TABLE:..........

--------------------------------------------------------

Assembler Instruction Instruction Opcode

start 1

end 2

origin 3

equ 4

ltorg 5

dc 6

ds 7

REGISTER TABLE:..........

--------------------------------------------------------

Register Name Register Opcode

areg, 1

breg, 2

creg, 3

dreg, 4

0

-----------------------------------------------

LITAL TAB

LIT NO LIT Name LIT Address

1 ='1' 103

2 ='3' 109

------------------------------------------------

SYM TAB

SYM NO SYM Name SYM Address

1 a 106

2 b 105

#include<stdio.h>

#include<conio.h>

#include<string.h>

int lcount=0,scount=0,sf=0;

int noltorg=0;

struct pooltab

{

int poolno;

int mdtindex;

}pt[10];

int lc;

struct mot

{

char opcode[6];

int opno,oplen;

};

struct pot

{

char opcode[7];

int opno;

};

struct regt

{

char regname[6];

int regno;

};

struct lit

{

char litname[10];

int litno;

int litadd;

};

struct sym

{

char name[10];

int no;

int add;

};

void main()

{

int i,j,lc,k,f=0;

char buff[10],blit[2],bcol[2];

FILE \*fr,\*fw;

clrscr();

struct mot m[11]={{"stop",0,1},{"add",1,1},{"sub",2,1},{"mult",3,1},{"mover",4,1},{"movem",5,1},{"comp",6,1},{"bc",7,1},{"div",8,1},{"read",9,1},{"print",10,1}};

struct pot p[7]={{"start",1},{"end",2},{"origin",3},{"equ",4},{"ltorg",5},{"dc",6},{"ds",7}};

struct regt r[5]={{"areg,",1},{"breg,",2},{"creg,",3},{"dreg,",4}};

struct sym s[10];

struct lit l[10];

printf("\n MNEMONIC OPCODE TABLE:..........\n");

printf("\n--------------------------------------------------------");

printf("\n Assembler Instruction \t Instruction Opcode \t Length");

for(i=0;i<=10;i++)

{

printf("\n\t %s \t\t %d \t\t %d",m[i].opcode,m[i].opno,m[i].oplen);

}

printf("\n");

printf("\n PSEUDO OPCODE TABLE:..........\n");

printf("\n--------------------------------------------------------");

printf("\n Assembler Instruction \t Instruction Opcode ");

for(i=0;i<=6;i++)

{

printf("\n \t %s \t\t %d",p[i].opcode,p[i].opno);

}

printf("\n");

printf("\n REGISTER TABLE:..........\n");

printf("\n--------------------------------------------------------");

printf("\n Register Name \t Register Opcode ");

for(i=0;i<=4;i++)

{

printf("\n\t %s \t %d",r[i].regname,r[i].regno);

}

fr=fopen("s.txt","r");

if(fr<0)

{

printf("\n error...............");

}

fw=fopen("intmed.dat","w");

if(fw<0)

{

printf("\n error...............");

}

pt[0].poolno=1;

pt[0].mdtindex=1;

fscanf(fr,"%s",buff);

// if(strcmp("start",buff)!=0)

// goto end;

fprintf(fw,"%s","(ad,0");

fprintf(fw,"%d",1);

fprintf(fw,"%c",')');

fscanf(fr,"%d",&lc);

fprintf(fw,"%s","(c,");

fprintf(fw,"%d",lc);

fprintf(fw,"%c",')');

readinst:

fscanf(fr,"%s",buff);

sf=0;

for(i=0;i<=10;i++)

{

if(strcmp(buff,m[i].opcode)==0)

{ sf=1;

fprintf(fw,"\n%s","(IS,0");

fprintf(fw,"%d",m[i].opno);

fprintf(fw,"%c",')');

fscanf(fr,"%s",buff);

for(j=0;j<=3;j++)

if(strcmp(buff,r[j].regname)==0)

{

fprintf(fw,"%d",r[j].regno);

}

fscanf(fr,"%s",buff);

strcpy(bcol,"'");

strcpy(blit,"=");

if(buff[0]==bcol[0])

{

fprintf(fw,"%s","(C,");

fprintf(fw,"%s",buff);

fprintf(fw,"%c\n",')');

}

else if(buff[0]==blit[0])

{

f=0;

for(k=pt[noltorg-1].mdtindex;k<=lcount;k++)

{

if(strcmp(l[k].litname,buff)==0)

{

f=1;

fprintf(fw,"%s","(L,");

fprintf(fw,"%d",l[k].litno);

fprintf(fw,"%c\n",')');

break;

}

}

if(f==0)

{

lcount++;

l[lcount].litno=lcount;

strcpy(l[lcount].litname,buff);

fprintf(fw,"%s","(L,");

fprintf(fw,"%d",l[lcount].litno);

fprintf(fw,"%c\n",')');

}

}

else

{

int f=0;

for(k=1;k<=scount;k++)

{

if(strcmp(s[k].name,buff)==0)

{

f=1;

fprintf(fw,"%s","(S,");

fprintf(fw,"%d",s[k].no);

fprintf(fw,"%c\n",')');

break;

}

}

if(f==0)

{

scount++;

s[scount].no=scount;

strcpy(s[scount].name,buff);

fprintf(fw,"%s","(S,");

fprintf(fw,"%d",s[scount].no);

fprintf(fw,"%c\n",')');

}

}

break;

}

}

//symbol

int inc=0;

if(strcmp(buff,"ltorg")==0)

{

sf=1;

inc=1;

noltorg++;

pt[noltorg].poolno=noltorg+1;

pt[noltorg].mdtindex=lcount+1;

fprintf(fw,"%s","\n(AD,05) ");

for(int h=pt[noltorg-1].mdtindex;h<=lcount;h++)

{

fprintf(fw,"%s\n",l[h].litname);

}

for(h=pt[noltorg-1].mdtindex;h<=lcount;h++)

{

l[h].litadd=lc++;

}

}

if(sf==0)

for(int h=1;h<=scount;h++)

{

if(strcmp(s[h].name,buff)==0)

{

s[h].add=lc;

fscanf(fr,"%s",buff);

if(strcmp(buff,"ds")==0)

{

fprintf(fw,"\n%s","(AD,07) ");

int no;

fscanf(fr,"%d",&no);

lc=lc+no-1

}

if(strcmp(buff,"dc")==0)

{

fprintf(fw,"\n%s","(AD,06) ");

fscanf(fr,"%s",buff);

fprintf(fw,"%s","(C, ");

fprintf(fw,"%s",buff);

fprintf(fw,"%c",')');

}

}

if(strcmp(buff,"end")==0)

{

inc=1;

for(int h=pt[noltorg].mdtindex;h<=lcount;h++)

{

l[h].litadd=lc++;

}

fprintf(fw,"%s","\n(AD,02) ");

for(h=pt[noltorg].mdtindex;h<=lcount;h++)

{

fprintf(fw,"%s\n",l[h].litname);

}

goto end;

}

if(inc==0)

lc++;

goto readinst;

end:

printf("\n LITAL TAB ");

printf("\n LIT NO\t LIT Name\t LIT Address ");

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

{

printf("\n%d \t %s \t %d",l[i].litno,l[i].litname,l[i].litadd);

}

printf("\n SYM TAB ");

printf("\n SYM NO\t SYM Name\t SYM Address ");

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

{

printf("\n%d \t %s \t %d",s[i].no,s[i].name,s[i].add);

}

getch();

}

(ad,01)(c,100)

(IS,04)1(L,1)

(IS,05)2(S,1)

(IS,01)1(S,2)

(AD,05) ='1'

(IS,01)3(L,2)

(AD,06) (C, 2)

(AD,07)

(AD,02) ='3'