**Program Files**

**pass1.cpp**

#include "utility.cpp"

#include "tables.cpp"

using namespace std;

int SIZE = 0, STADR = 0;

int pass1()

{

load\_tables();

ifstream fin;

ofstream fout;

fin.open("in.asm", ios::in);

fout.open("inter.asm");

vector<string> row;

string line;

int START=0, LOCCTR=0, SIZE=0;

cout<<endl<<"Pass 1: Intermediate File"<<endl<<endl;

while (getline(fin, line))

{

row = tokenize(line);

if(row[1]=="START")

{

LOCCTR = START = str2int(row[2]);

cout<<hex<<LOCCTR<<"\t"<<line<<endl;

fout<<LOCCTR<<"\t"<<line<<endl;

continue;

}

else if(row[1]=="EQU")

{

if(row[0] != "\0")

{

if(SYMTAB.find(row[0])==SYMTAB.end())

{

if(row[2] == "\0")

{ cout<<"\nInvalid Usage"<<endl; break; }

SYMTAB[row[0]].address = str2int(row[2]);

SYMTAB[row[0]].name = row[0];

SYMTAB[row[0]].exists = 'y';

SYMTAB[row[0]].is\_equ = true;

cout<<hex<<LOCCTR<<"\t"<<line<<endl;

fout<<LOCCTR<<"\t"<<line<<endl;

}

else

{ cout<<"\nDuplicate Symbol"<<endl; break; }

}

else

{ cout<<"\nInvalid Usage"<<endl; break; }

continue;

}

else if(row[1]=="END")

{

SIZE=LOCCTR-START;

cout<<hex<<LOCCTR<<"\t"<<line<<endl;

fout<<LOCCTR<<"\t"<<line<<endl;

break;

}

if(row[0] != "\0")

{

if(SYMTAB.find(row[0])==SYMTAB.end())

{

SYMTAB[row[0]].address = LOCCTR;

SYMTAB[row[0]].name = row[0];

SYMTAB[row[0]].exists = 'y';

}

else

{ cout<<"\nDuplicate Symbol"<<endl; break; }

}

int len = 0;

if(OPTAB[row[1]].exists=='y')

OPTAB[row[1]].format>3?len=2:len=1;

else if(row[1] == "DB")

len = 1;

else

{ cout<<"Error: Invalid Opcode "<<row[1]<<endl; break; }

fout<<LOCCTR<<"\t"<<line<<endl;

cout<<hex<<LOCCTR<<"\t";

for(int i=0; i<row.size(); i++)

cout<<row[i]<<"\t";

cout<<endl;

LOCCTR += len;

}

cout<<endl<<"SYMTAB"<<endl;

for(map<string, struct\_label>::iterator i=SYMTAB.begin(); i!=SYMTAB.end(); i++)

{

cout<<hex<<i->first<<"\t"<<i->second.address<<endl;

}

return SIZE;

}

/\*

int main()

{

int pgm\_size = pass1();

return 0;

}\*/

**pass2.cpp**

#include "pass1.cpp"

using namespace std;

void pass2(int SIZE)

{

cout<<endl<<"Pass 2:"<<endl;

ifstream fin;

ofstream fout;

fin.open("inter.asm", ios::in);

fout.open("out.asm");

vector<string> row;

string line;

int STADR=0, LOCCTR=0;

stringstream INSTR;

int inst\_len = 0;

INSTR<<"T^";

while (getline(fin, line))

{

//cout<<endl<<line<<"\t";

int len = 0;

row = tokenize(line);

string instr ="";

if(row[2]=="START")

{

STADR = str2int(row[3]);

INSTR<<setw(6)<<hex<<STADR<<"^";

fout<<"H^"<<setw(6)<<row[1]<<"^"<<setw(6)<<setfill('0')<<hex<<STADR<<"^"<<setw(6)<<setfill('0')<<hex<<SIZE<<endl;

cout<<"H^"<<setw(6)<<row[1]<<"^"<<setw(6)<<setfill('0')<<hex<<STADR<<"^"<<setw(6)<<setfill('0')<<hex<<SIZE<<endl;

continue;

}

else if(row[2]=="END")

{

fout<<"E^"<<setw(6)<<setfill('0')<<hex<<SYMTAB[row[3]].address<<endl;

cout<<"E^"<<setw(6)<<setfill('0')<<hex<<SYMTAB[row[3]].address<<endl;

continue;

}

else if(row[3]=="EQU")

continue;

int OPCODE, DISPL = 0;

int fmt=0;

if(OPTAB[row[2]].exists=='y')

{

OPCODE = str2int(OPTAB[row[2]].opcode);

fmt = OPTAB[row[2]].format;

if(fmt==1)

{

int op = OPCODE \* 32;

cout<<setw(1)<<setfill('0')<<hex<<op;

fout<<setw(1)<<setfill('0')<<hex<<op<<endl;

len = 2;

}

else if(fmt == 2)

{

int op = OPCODE\*32;

int reg = REGTAB[row[3]].num \* 8;

op += reg;

cout<<setw(1)<<setfill('0')<<hex<<op;

fout<<setw(1)<<setfill('0')<<hex<<op<<endl;

len = 2;

}

else if(fmt == 3)

{

int op = OPCODE\*32;

vector<string> regs = stcut(row[3], ',');

int reg1 = REGTAB[regs[0]].num \* 8;

int reg2 = REGTAB[regs[1]].num \* 2;

op += reg1; op+= reg2;

cout<<setw(1)<<setfill('0')<<hex<<op;

fout<<setw(1)<<setfill('0')<<hex<<op<<endl;

len = 2;

}

else if(fmt == 4)

{

int op = OPCODE\*32;

vector<string> ops = stcut(row[3], ',');

int reg1 = REGTAB[ops[0]].num \* 8;

int reg2 = REGTAB[ops[1]].num \* 2;

op += reg1; op+= reg2;

op \*= 256;

if(SYMTAB[ops[2]].is\_equ)

DISPL = SYMTAB[ops[2]].address;

else

{

int f\_len; OPTAB[row[2]].format>3?f\_len=2:f\_len=1;

LOCCTR = str2int(row[0]) + f\_len;

DISPL = SYMTAB[ops[2]].address - LOCCTR;

if(DISPL<0)

DISPL -= 0xffffff00;

}

op += DISPL;

cout<<setw(1)<<setfill('0')<<hex<<op;

fout<<setw(1)<<setfill('0')<<hex<<op<<endl;

len = 4;

}

else if(fmt == 5)

{

int op = OPCODE\*32\*256;

if(SYMTAB[row[3]].is\_equ)

DISPL = SYMTAB[row[3]].address;

else

{

int f\_len; OPTAB[row[2]].format>3?f\_len=2:f\_len=1;

LOCCTR = str2int(row[0]) + f\_len;

DISPL = SYMTAB[row[3]].address - LOCCTR;

if(DISPL<0)

DISPL -= 0xffffff00;

}

op += DISPL;

cout<<setw(1)<<setfill('0')<<hex<<op;

fout<<setw(1)<<setfill('0')<<hex<<op<<endl;

len = 4;

}

}

cout<<"\t\t"<<line<<endl;

}

}

int main()

{

int SIZE;

SIZE = pass1();

pass2(SIZE);

}

**tables.cpp**

#include<iostream>

#include<map>

#include<string>

using namespace std;

struct struct\_opcode{

string opcode;

int format;

char exists;

struct\_opcode(){

opcode="undefined";

format=0;

exists='n';

}

};

struct struct\_label{

int address;

string name;

bool is\_equ;;

char exists;

struct\_label(){

name="undefined";

address=0;

exists='n';

is\_equ = false;

}

};

struct struct\_register{

int num;

char exists;

struct\_register(){

num = 'F';

exists='n';

}

};

typedef map<string,struct\_label> SYMBOL\_TABLE\_TYPE;

typedef map<string,struct\_opcode> OPCODE\_TABLE\_TYPE;

typedef map<string,struct\_register> REG\_TABLE\_TYPE;

SYMBOL\_TABLE\_TYPE SYMTAB;

OPCODE\_TABLE\_TYPE OPTAB;

REG\_TABLE\_TYPE REGTAB;

void load\_REGTAB(){

REGTAB["R0"].num=0;

REGTAB["R0"].exists='y';

REGTAB["R1"].num=1;

REGTAB["R1"].exists='y';

REGTAB["R2"].num=2;

REGTAB["R2"].exists='y';

REGTAB["R3"].num=3;

REGTAB["R3"].exists='y';

}

void load\_OPTAB(){

OPTAB["ADD"].opcode="0";

OPTAB["ADD"].format=3;

OPTAB["ADD"].exists='y';

OPTAB["NEG"].opcode="1";

OPTAB["NEG"].format=2;

OPTAB["NEG"].exists='y';

OPTAB["LOAD"].opcode="2";

OPTAB["LOAD"].format=4;

OPTAB["LOAD"].exists='y';

OPTAB["STORE"].opcode="3";

OPTAB["STORE"].format=4;

OPTAB["STORE"].exists='y';

OPTAB["JE"].opcode="4";

OPTAB["JE"].format=5;

OPTAB["JE"].exists='y';

OPTAB["JL"].opcode="5";

OPTAB["JL"].format=5;

OPTAB["JL"].exists='y';

OPTAB["JG"].opcode="6";

OPTAB["JG"].format=5;

OPTAB["JG"].exists='y';

OPTAB["HLT"].opcode="7";

OPTAB["HLT"].format=1;

OPTAB["HLT"].exists='y';

}

void load\_tables(){

load\_OPTAB();

load\_REGTAB();

}

**utility.cpp**

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include <iomanip>

#include <map>

#include <algorithm>

#include <cstring>

#include <vector>

#include <cmath>

using namespace std;

vector<string> tokenize(string temp)

{

string line, word;

vector<string> row;

row.clear();

stringstream s(temp);

while (getline(s, word, '\t'))

row.push\_back(word);

return row;

}

int str2int(string s)

{

char cstr[s.size() + 1];

s.copy(cstr, s.size() + 1);

cstr[s.size()] = '\0';

return atoi(cstr);

}

int stlen(string s)

{

char cstr[s.size() + 1];

s.copy(cstr, s.size() + 1);

cstr[s.size()] = '\0';

return strlen(cstr);

}

vector<string> stcut(string temp, char c)

{

string line, word;

vector<string> row;

row.clear();

stringstream s(temp);

while (getline(s, word, c))

row.push\_back(word);

return row;

}

int strhex2int(string s)

{

char cstr[s.size() + 1];

s.copy(cstr, s.size() + 1);

cstr[s.size()] = '\0';

return strtoul(cstr,NULL,16);

}

bool str2strcmp(string s1, string s2)

{

char cstr1[s1.size() + 1];

s1.copy(cstr1, s1.size() + 1);

cstr1[s1.size()] = '\0';

char cstr2[s2.size() + 1];

s2.copy(cstr2, s2.size() + 1);

cstr2[s2.size()] = '\0';

if(strcmp(cstr1,cstr2))

return false;

else

return true;

}

**Input File**

**in.asm**

HYP START 0

A DB 0

B DB 3

C EQU 100

NEG R2

LOOP ADD R2,R1

STORE R1,R2,C

NEG R3

JE HALT

LOAD R0,R1,A

STORE R1,R2,B

JL LOOP

JG LOOP

HALT HLT

END LOOP

**Intermediate file**

**inter.asm**

0 HYP START 0

0 A DB 0

1 B DB 3

2 C EQU 100

2 NEG R2

3 LOOP ADD R2,R1

4 STORE R1,R2,C

6 NEG R3

7 JE HALT

9 LOAD R0,R1,A

11 STORE R1,R2,B

13 JL LOOP

15 JG LOOP

17 HALT HLT

18 END LOOP

Output File

out.asm

H^ HYP^000000^000012

30

12

6c64

38

8008

42f5

6cf4

a0f4

c0f2

e0

E^000003

**Terminal Output**

**Pass 1: Intermediate File**

0 HYP START 0

0 A DB 0

1 B DB 3

2 C EQU 100

2 NEG R2

3 LOOP ADD R2,R1

4 STORE R1,R2,C

6 NEG R3

7 JE HALT

9 LOAD R0,R1,A

b STORE R1,R2,B

d JL LOOP

f JG LOOP

11 HALT HLT

12 END LOOP

SYMTAB

A 0

B 1

C 64

HALT 11

LOOP 3

Pass 2:

H^ HYP^000000^000012

0 A DB 0

1 B DB 3

2 C EQU 100

30 2 NEG R2

12 3 LOOP ADD R2,R1

6c64 4 STORE R1,R2,C

38 6 NEG R3

8008 7 JE HALT

42f5 9 LOAD R0,R1,A

6cf4 11 STORE R1,R2,B

a0f4 13 JL LOOP

c0f2 15 JG LOOP

e0 17 HALT HLT

E^000003