21MIS1001
Sabbarish S

02-08-2024
Monday

SWE4001
System Programming

LAB-3

Pass-2 Assembler

CODE:

```
#include<bits/stdc++.h>
using namespace std;
class Pass2{
string programName;
int startAddress;
int lenPro;
int loctr;
vector<map<string,string>> instructOp;
map<string, string>opTab = {
{"ADD", "1A"},
{"AND", "40"},
{"COMP", "28"},
{"DIV", "24"},
{"J", "3C"},
{"JEQ", "30"},
{"JGT", "34"},
{"JLT", "38"},
{"JSUB", "48"},
{"LDA", "00"},
{"LDCH", "50"},
{"LDL", "08"},
{"LDX", "04"},
{"MUL", "20"},
{"OR", "44"},
{"RD", "D8"},
{"RSUB", "4C"},
{"STA", "0C"},
{"STCH", "54"},
{"STL", "14"},
{"STSW", "E8"},
{"STX", "10"},
{"SUB", "1C"},
{"TD", "E0"},
{"TIX", "2C"},
{"WD", "DC"}
};
public:
map<string, string> sumTabdata;
void writeProCode(){
fstream pgm("objpgm.txt",ios::app);
if(pgm.is_open()){
```

```
pgm << "H" << "\t" << setw(6) << left << programName
<<"\t"<< setw(6) << setfill('0') << right << hex << startAddress<<"\t"
<< setw(6) << lenPro << "\n";
while (!instructOp.empty()) {
string textRec = "T";
int d = startAddress;
int recordStart =d;
string objectCode;
while (!instructOp.empty() && objectCode.length() < 60) {
auto instruction = instructOp.front();
instructOp.erase(instructOp.begin());
objectCode += instruction["opcode"];
d+= 3;
}
pgm << textRec<<"\t" << setw(6) << setfill('0') << hex << recordStart <<"\t"
<< setw(2) << setfill('0') << (objectCode.length() / 2) <<"\t"
<< objectCode << "\n";
}
pgm << "E" <<"\t"<< setw(6) << setfill('0') << startAddress << "\n";
}
}
void writeObjCode(vector<string> readLine){
fstream obj("objcode.txt",ios::app);
if(obj.is_open()){
if(readLine.size()==4){
for(auto d: readLine){
obj<<d<<"\t";
}
loctr = stoi(readLine[0],nullptr, 16);
}else if(readLine[1]=="START"){
startAddress = stoi(readLine[2],nullptr, 16);
programName = readLine[0];
obj<<"\t\t\t";
for(auto d: readLine){
obj<<d<<"\t";
}
}else if(readLine[0]=="END"){
lenPro = loctr - startAddress;
obj<<"\t\t";
obj<<readLine[0]<<"\t\t\t\t"<<readLine[1];
}
if(opTab.find(readLine[1]) !=opTab.end()){
map<string, string> instruction;
instruction["address"] = to_string(loctr);
instruction["opcode"] = opTab[readLine[1]] + sumTabdata[readLine[2]];
instructOp.push_back(instruction);
loctr = stoi(readLine[0],nullptr, 16);
obj<<readLine[0]<<"\t\t"<<readLine[1]<<"\t"<<readLine[2]<<"\t\t"<<opTab[readLi
ne[1]]+sumTabdata[readLine[2]];
}
obj<<"\n";
}
```

```
}
void readSymTab(){
for (auto itr = sumTabdata.begin(); itr != sumTabdata.end(); ++itr) {
cout << itr->first
<< '\t' << itr->second << '\n';
}
}
void loadSymtab(){
fstream symTabF("symtab.txt",ios::in);
if(symTabF.is_open()){
string symBuffer;
while(getline(symTabF,symBuffer)){
stringstream ss(symBuffer);
string buffer;
bool isFirst = true;
string var;
string add;
while(ss>>buffer){
if(isFirst){
add = buffer;
isFirst = false;
}else{
var = buffer;
isFirst = true;
}
}
sumTabdata.insert({var, add});
}
cout<<"Symtab loaded"<<endl;
}
}
void splitandstore(string line){
stringstream ss(line);
string buffer;
int index = 0;
vector <string> lineBuffer;
while(ss>>buffer){
lineBuffer.push_back(buffer);
}
writeObjCode(lineBuffer);
}
void readIntermediateFile(){
fstream inputF("intermediate.txt", ios::in);
if(inputF.is_open()){
cout<<"LOG: file open success"<<endl;
string buffer;
while(getline(inputF,buffer)){
splitandstore(buffer);
}
cout<<endl;
}
}
```

```
};
int main(){
Pass2 p2;
p2.loadSymtab();
p2.readIntermediateFile();
p2.readSymTab();
p2.writeProCode();
}
```

Output:

```
PS C:\Users\sabba\Documents\SWE4001-SP\LAB> cd "c:\Users\sabba\Documents\SWE4001-SP\LAB\" ; if ($?
) { g++ Pass2.cpp -o Pass2 } ; if ($?) { .\Pass2 }
Symtab loaded
LOG: file open success

ALPHA   1018
BETA    101b
DELTA   1024
GAMMA   1021
INCR    1027
ONE     1015
```

Objcode.txt

objcode.txt

```
1                SAMPLE  START   1000
2    1000        LDA ALPHA       001018
3    1003        ADD INCR        1A1027
4    1006        SUB ONE         1C1015
5    1009        STA BETA        0C101b
6    100c        LDA GAMMA       001021
7    100f        ADD INCR        1A1027
8    1012        STA DELTA       0C1024
9    1015    ONE     WORD    1
10   1018    ALPHA   RESW    1
11   101b    BETA    RESW    2
12   1021    GAMMA   RESW    1
13   1024    DELTA   RESW    1
14   1027    INCR    RESW    1
15           END             SAMPLE
16
```

Objpgm

objpgm.txt

```
1   H   SAMPLE  001000  000027
2   T   001000  15  0010181A10271C10150C101b0010211A10270C1024
3   E   001000
4
```