21MIS1001 01-10-2024

Sabbarish S Tuesday

SWE4001- Lab 6

Lab Question 5 – Direct Linking Loader

Write a C program to Implement pass one of a direct-linking loader. Sample three input files are given.

**PROGA.txt**

H^PROGA^000000^0063

D^LISTA^000040^ENDA^000054^

R^LISTB^ENDB^LISTC^ENDC

T^000020^0A^03201D^77100004^050014

T^000054^0F^000014^FFFFF6^00003F^000014^FFFFC0

M^000024^05^+LISTB

M^000054^06^+LISTC

M^000057^06^+ENDC

M^000057^06^-LISTC

M^00005A^06^+ENDC

M^00005A^06^-LISTC

M^00005A^06^+PROGA

M^00005D^06^-ENDB

M^00005D^06^+LISTB

M^000060^06^+LISTB

M^000060^06^-PROGA

E^000020

**PROGB.txt**

H^PROGB^000000^007F

D^LISTB^000060^ENDB^000070^

R^LISTA^ENDA^LISTC^ENDC

T^000036^0B^03100000^772027^05100000

T^000070^0F^000000^FFFFF6^FFFFFF^FFFFF0^000060

M^000037^05^+LISTA

M^00003E^05^+ENDA

M^00003E^05^-LISTA

M^000070^06^+ENDA

M^000070^06^-LISTA

M^000070^06^+LISTC

M^000073^06^+ENDC

M^000073^06^-LISTC

M^000076^06^+ENDC

M^000076^06^-LISTC

M^000076^06^+LISTA

M^000079^06^+ENDA

M^000079^06^-LISTA

M^00007C^06^+PROGB

M^00007C^06^-LISTA

E

**PROGC.txt**

H^PROGC^000000^0051

D^LISTC^000030^ENDC^000042^

R^LISTA^ENDA^LISTB^ENDB

T^000018^0C^03100000^77100004^05100000

T^000042^0F^000030^000008^000011^000000^000000

M^000019^05^+LISTA

M^00001D^05^+LISTB

M^000021^05^+ENDA

M^000021^05^-LISTA

M^000042^06^-ENDA

M^000042^06^-LISTA

M^000042^06^+PROGC

M^000048^06^+LISTA

M^00004B^06^+ENDA

M^00004B^06^-LISTA

M^00004B^06^-ENDB

M^00004B^06^+LISTB

M^00004E^06^+LISTB

M^00004E^06^-LISTA

E

**CODE**

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <vector>

#include <unordered\_map>

#include <iomanip>

using namespace std;

struct Symbol {

string symbolName;

int address;

};

struct ControlSection {

string sectionName;

int startAddress;

int length;

};

int hexToInt(const string& hexStr) {

int value;

stringstream ss;

ss << hex << hexStr;

ss >> value;

return value;

}

int main() {

unordered\_map<string, Symbol> symbolTable;

vector<ControlSection> controlSections;

int currentAddress = 0x4000;

string inputFiles[] = { "PROGA.txt", "PROGB.txt", "PROGC.txt" };

for (const string& fileName : inputFiles) {

ifstream inputFile(fileName);

string line;

if (!inputFile.is\_open()) {

cerr << "Error opening file: " << fileName << endl;

return 1;

}

string currentSectionName;

int sectionLength = 0;

while (getline(inputFile, line)) {

if (line[0] == 'H') {

stringstream ss(line);

string recordType, sectionName, startAddressStr, lengthStr;

getline(ss, recordType, '^');

getline(ss, sectionName, '^');

getline(ss, startAddressStr, '^');

getline(ss, lengthStr, '^');

currentSectionName = sectionName;

sectionLength = hexToInt(lengthStr);

controlSections.push\_back({ currentSectionName, currentAddress,

sectionLength });

} else if (line[0] == 'D') {

stringstream ss(line);

string recordType, symbolName, addressStr;

getline(ss, recordType, '^');

while (getline(ss, symbolName, '^') && getline(ss, addressStr, '^'))

{

int symbolAddress = hexToInt(addressStr) + currentAddress;

symbolTable[symbolName] = { symbolName, symbolAddress };

}

}

}

currentAddress += sectionLength;

inputFile.close();

}

cout << "Pass one of a Direct-Linking Loader" << endl;

cout << "---------------------------------------------------------------" <<

endl;

cout << "Control Section\tSymbol Name\tAddress\tLength" << endl;

cout << "---------------------------------------------------------------" <<

endl;

for (const auto& section : controlSections) {

cout << section.sectionName << "\t\t" << hex << section.startAddress << "\t"

<< hex << section.length << endl;

for (const auto& symbol : symbolTable) {

if (symbol.second.address >= section.startAddress &&

symbol.second.address < section.startAddress + section.length) {

cout << "\t\t" << symbol.second.symbolName << "\t" << hex <<

symbol.second.address << endl;

}

}

}

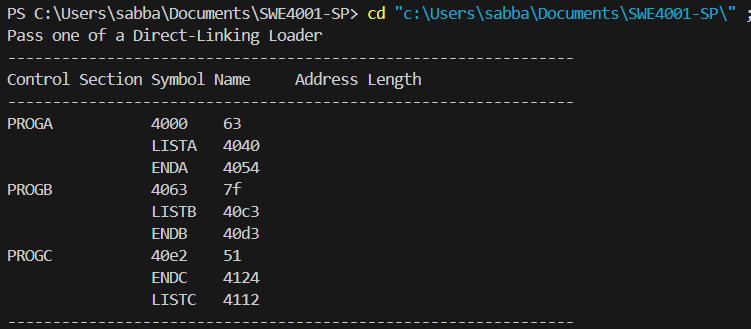
cout << "---------------------------------------------------------------" <<

endl;

return 0;

}

**OUTPUT**

****