Linker/Loader Lab CSE 560

Table of Contents

Introduction

This programmer's guide is divided into four sections. The first, entitled "Overview of System and Data Flow" describes the design adopted for this lab, i.e. how many classes were used and what the coding conventions were. The second section, "Data Structure Descriptions" gives a detailed account of all data structures shared between the three classes. "Module Descriptions" is the third section; this describes each method used in each of the three classes. It lists the input and output parameters, descriptions of what the procedure/method does, and states the *requires*, *modifies* and *ensures* clauses. The last section, "Code," provides all the code written for the three classes.

Java was chosen as the language for this project. Coding was done using the Eclipse IDE on the operating system Windows Vista.

Section 1: Overview of System and Data Flow

A. Directory structure

Introduction

The project was done by interlocking three classes:

1) Lab3Pass1
2) Lab3Pass2
3) Ordered_Four

Lab3Pass1 is the main class or the driver class. The primary objective of Lab3Pass1 is to go through the input object file given by the user and to ensure that this file follows the syntactic conventions in force. Lab3Pass1 also converts each line into an Ordered_Four, which is a four tuple that compactly represents all information. It creates the intermediate table and the external symbol table and calculates the length of the program. If it finds any errors with the syntax, it outputs an error message and brief explanation.

Lab3Pass2 takes as input the external symbol table and intermediate table from Lab3Pass1 and creates the machine language instruction corresponding to the information in the four tuple, Ordered_Four. It creates and outputs the object file. It prints out errors when symbols are undefined, values are out of range or input files are incorrectly structured.

Ordered_Four is a class made up of four tuples that stores Strings representing record type (such as header, text or end), address (address or the first memory address), value (contents of address or size of memory range) and relocatability.

I. Lab3Pass1

The Lab3Pass1 class contains the following methods shown in Table 1 below.

Table 1: Lab3Pass1 Reference Guide

Lab3Pass1 Class (pp 20-35)			
	Procedure	**Line number**	**Module Description**
1.	checkName	21	pg 12
2.	checkLength	80	pg 12
3.	checkHex	110	pg 12
4.	checkDecimal	139	pg 13
5.	checkLine	168	pg 13
6.	main	201	pg 13

II. Lab3Pass2

The Lab3Pass2 class contains three methods that are recorded in Table 2.

Table 2: Lab3Pass2 Reference Guide

Lab3Pass2 Class (pp 36-40)			
	Method/Procedure	**Line number**	**Module Description**
1.	checkLength	21	pg 14
2.	main	53	pg 14
3.	Sort	98	pg 15

III. Ordered_Four

The Ordered_Four class contains several methods that have been referenced in the following Table 3.

Table 3: Ordered_Four Reference Guide

Ordered_Four Class (pp 41-45)			
	Procedure	**Line number**	**Module Description**
1.	Ordered_Four (String, String)	24	pg 15
2.	Ordered_Four (String, String, String)	49	pg 16
3.	Ordered_Four (String, String, String, String)	74	pg 16
4.	getType()	97	pg 17
5.	getAddress()	116	pg 17
6.	getValue()	136	pg 17
7.	getRelocatable()	155	pg 18
8.	setType()	174	pg 18
9.	setAddress()	193	pg 18
10.	setValue()	214	pg 19
11.	setRelocatable()	233	pg 19

B. Design Conventions

I. Naming conventions

The naming conventions for variables and constants are the short form notation of what was needed in the code. For instance **IPLA** represents initial program loading address, **addr** stands for address in Integer representation and **lineNo** is used to denote line number. **temp** stands for temporary.

The naming conventions for the functions are of the programmers choosing. For example **main** is used to indicate the point at which the program starts running. The name of method **Sort** was chosen to imply sorting records of various types and printing them out. The subroutines **checkHex** and **checkDecimal** were used to check the hex and decimal symbols respectively and therefore were named in this manner.

The naming conventions for the classes and files are as suggested in the assignment of this lab but since the Assembler Lab also possesses classes called Pass1 and Pass2, for this Lab the names of **Lab3Pass1** and **Lab3Pass2** have been used instead.

II. Parameter passing conventions

The parameter passing conventions are that parameters must exist. All operations in Lab3Pass1 class like **lineNo** (line number) need to be initialized. **lineNo** was initialized to 1. The rest were initialized to the values given in the input file, like type, address, value and relocatable. For example the parameter passing convention for String is that it can have any length, and could be an empty String as well.

III. Memory-management conventions

There is no memory management involved in this Linker/Loader Lab 3 therefore no discussion on this topic is included.

IV. Error-catching and Error-message generation conventions

Try catch exceptions were used as one form of error-catching to check the hexadecimal and decimal characters in the **checkDecimal** and **checkHex** methods. The other form of error-catching involved if, else if and else statements to detect and signal for errors. Error-messages were numbered so that common errors would be given in the same

number for easy referencing. All possible situations where an error could occur were taken into account and an error message was generated to indicate its detection.

C. Calls Graphs

I. General Overview

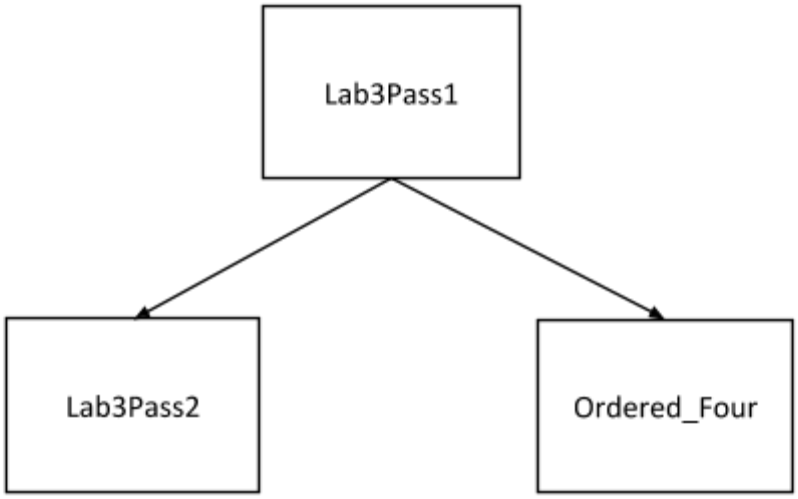

Figure SEQ Figure * ARABIC 1: Lab3Pass1 calls Lab3Pass2 & Ordered_Four.

II. Lab3Pass1

Calls made by the method main inside the Lab3Pass1.

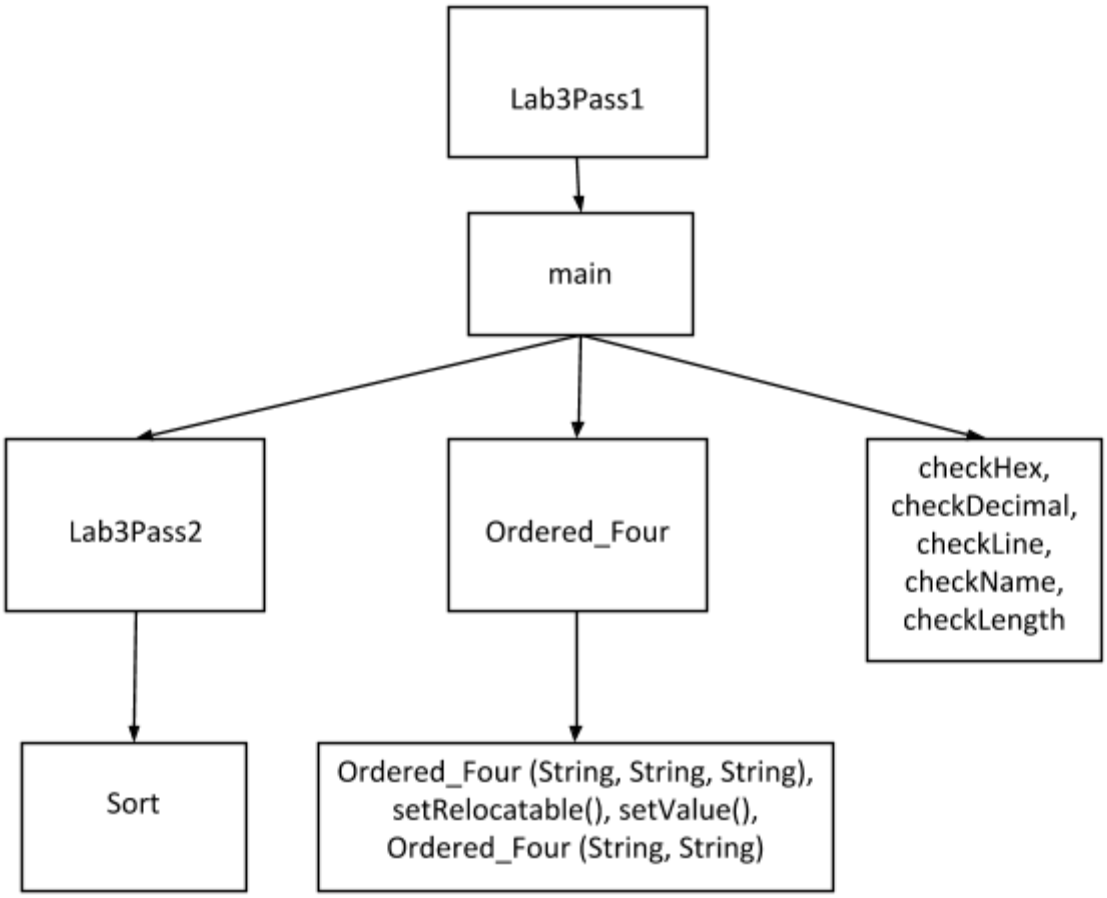

Figure 2: The Lab3Pass1's method *main* calls all the following methods inside Lab3Pass2, Ordered_Four and Lab3Pass1.

Figure 3: No calls made by these methods to any class.

III. Lab3Pass2

Calls made by the main method inside Lab3Pass2.

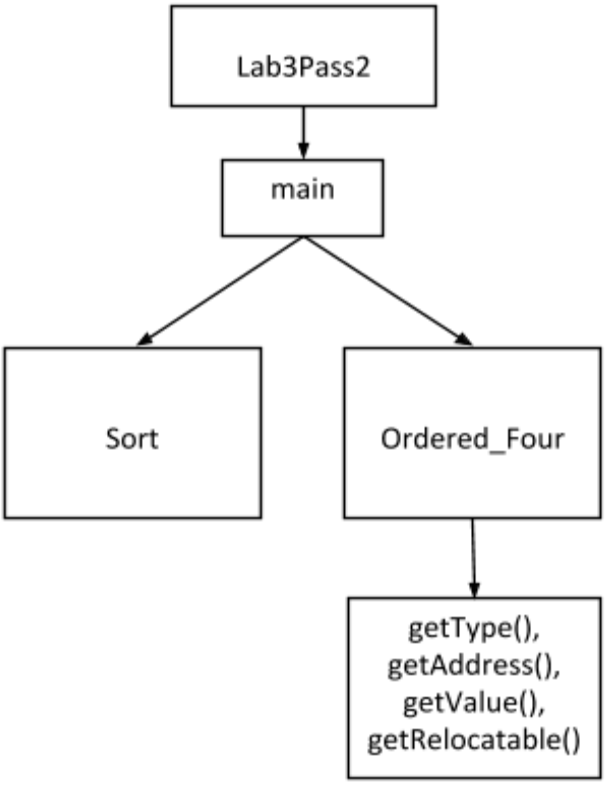

Figure 4: The Lab3Pass2's method *sort* calls all the above methods inside Lab3Pass2 and Ordered_Four.

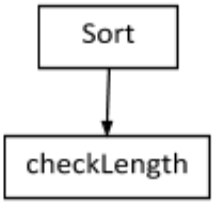

Figure 5: Sort Calls 1 method.

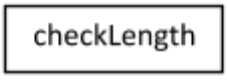

Figure 6: No calls made by this method.

IV. No Calls

No calls are made by the following class:

1. Ordered_Four

Section 2: Data Structure Descriptions

A. Shared Data Structures

The descriptions of the shared data structures are given below and Table 7 summarizes their characteristics.

The **Intermediate Table (IT)** is an object of generic type HashMap which maps Integers onto Ordered_Four. Ordered_Four is a four tuple made of all String parameters which represent the record type, address, value, and relocatable symbol. It is made by Lab3Pass1 to give to Lab3Pass2 so that Lab3Pass2 does not need to read the input file again.

The **External Symbol Table (XST)** is an object of generic type HashMap which maps Strings onto Strings. The String Name, which is the name of the external symbol, has a value that is stored as a Hex String.

I. Table 4: Shared Data Structure Dictionary

Structure Name	Type	Declared In	Used By	Purpose
IT	HashMap<Integer, Ordered_Four>	Lab3Pass1	Lab3Pass1/ Lab3Pass2	It stores every instruction into chunks for Lab3Pass2 and also stores the ultimate address after adding the PLA (program loading address).
XST	HashMap<String, String>	Lab3Pass1	Lab3Pass1/ Lab3Pass2	It stores the external symbol's name in String and value in Hex String. It also stores the program name with its respective PLA as its value. Lab3Pass2 fills this external symbol table up and passes it to Lab3Pass2.

Section 3: Module Descriptions

A.

Class: Lab3Pass1

Method: checkName
Description: Checks if the program name is in proper syntactic form and gives an error if it is not.
Calling Sequence
Input parameters: String (Name, line)
Output parameters: String (Name)
Requires: A name (in String) and the line (in Integer) the program is working on.
Modifies: Nothing.
Ensures: The Name is checked to conform to machine syntax and if Name is incorrect an error message is generated.
Modification Log: No changes needed.

Method: checkLength
Description: Checks if the length of the string is equal to Integer n. If it is less than n then it pads it with zeros. If it is more then it truncates the String (reg).
Calling Sequence
Input parameters: String (reg), Integer (n)
Output parameters: String (reg)
Requires: The String to be in Hex representation or binary representation.
Modifies: The length of the String reg.
Ensures: The length of the String reg is equal to the Integer n.
Modification Log: No changes needed.

Method: checkHex
Description: Checks if the Hex characters given by the user (in the input file) are of proper Hex syntax.
Calling Sequence
Input parameters: String (value, line)
Output parameters: Returns the Integer (v) that is the Integer representation of the Hex string.
Requires: The Hex number in String with the preceding prefix of x.
Modifies: The hex string into its integer representation.
Ensures: The String value is checked thoroughly and if it is incorrect an error message is generated.
Modification Log: No changes needed.

Method: checkDecimal
Description: Checks if the decimal characters given by the user (in the input file) are of proper decimal syntax.
Calling Sequence
Input parameters: String (value, line)
Output parameters: Returns the Integer (v) that is the Integer representation of the decimal string.
Requires: The decimal number in String with the preceding prefix of #.
Modifies: The decimal string into its integer representation.
Ensures: The String value is checked thoroughly and if it is incorrect an error message is generated.
Modification Log: No changes needed.

Method: checkLine
Description: It skips all comments and newlines while input file is being read and gives error if program terminated unexpectedly.
Calling Sequence
Input parameters: String (line), BufferedReader (in)
Output parameters: String (line)
Requires: String and BufferedReader.
Modifies: Changes the value of the String line into the next appropriate value from buffered reader.
Ensures: Get the next proper line in String.
Modification Log: No changes needed.

Method: main
Description: Takes in several linked relocatable files or one absolute file and checks their syntax and converts it into an intermediate table. For the cases of the relocatable files, it takes their external symbols and puts them in the External Symbol Table.
Calling Sequence
Input parameters: Nothing, just needs valid input files.
Output parameters: Nothing.
Requires: Valid input file.
Modifies: Nothing
Ensures: Passes Intermediate Table and External Symbol Table to Lab3Pass2.
Modification Log: No changes needed.

B.

Class: Lab3Pass2

Method: checkLength
Description: Checks if the length of the string is equal to Integer n. If it is less than n then it pads it with zeros. If it is more then it truncates the String (reg).
Calling Sequence
Input parameters: String (reg), Integer (n)
Output parameters: String (reg)
Requires: The String to be in Hex representation or binary representation.
Modifies: The length of the String reg.
Ensures: The length of the String reg is equal to the Integer n.
Modification Log: No changes needed.

Method: main
Description: Creates a file called object.txt which will store the object file.
It takes out the line number, record type, address, value and relocatable symbol from each row of the Intermediate Table and passes it to the Sort subroutine.
Calling Sequence
Input Parameters: HashMap<Integer, Ordered_Four> (IT), HashSet<String, String> (XST)
Output Parameters: Nothing
Requires: A HashMap of Integer and Ordered_Four, which represents the Intermediate Table.
Modifies: Nothing
Ensures: String representation of type, address, value and relocatable are taken out from the Intermediate File and are passed to Sort subroutine The integer lineNo is also given to the Sort subroutine.
Procedure Creation date: August 15th, 2008
Modification Log: No changes needed.

Method: Sort
Description: If the record type (Type) of the input file is the main program Header record then it prints out the Type (which includes the program name), Address (first memory address) and the Value (size of memory range needed).
If the record type (Type) is an End record then it prints out Type and the Address (starting location of where the program starts to read).
If it is a Text Record it prints out Type,Address and Value.
Calling Sequence
Input Parameters: String (Type, Address, Value, Relocatable), HashSet<String, String> (XST), Integer (lineNo, PLA), PrintWriter (output)
Output Parameters: Integer (PLA)
Requires: String Type, i.e. a Header, Text or End record, String Address, String Value, String Relocatable and Integer (PLA).
Modifies: Integer PLA is modified if Type is Header Record
Ensures: If it is the main program Header record then Type (which includes program name), Address and size of memory range is printed out and PLA is updated. If End record then Type

and starting location of where the program is supposed to read from is printed out. If Text record then Type, Address, is printed out. Value is changed and printed when Text record relocatable symbol.
Modification Log: Added Error Messages which detect those symbol names which are not defined in the XST.

C.

Class: Ordered_Four

Method: Ordered_Four (String, String)
Description: It is a constructor which creates a four tuple that consists of a
String tp (type such as Header, Text or End record), String addr (address or Location Counter), String value (contents of address in Text record or the size of the memory range in Header Record) is an empty String, and relocatable (relocatable symbol) is an empty string.
Calling Sequence
Input Parameters: String (tp, addr)
Output Parameters: New Ordered_Four in which String parameters are equal to tp,addr,empty string and empty string.
Requires: Strings as input parameters.
Modifies: Creates a new Ordered_Four as the output with the input String tp, addr,empty string and empty string.
Ensures: A brand new Ordered_Four is made with tp, addr,empty string and empty string as its String parameters.
Modification Log: No changes needed.

Method: Ordered_Four (String, String, String)
Description: It is a constructor which creates a four tuple that consists of a
String tp (type such as Header Text or End record), String addr (address or Location Counter), String v (value) is either the contents of address when reading a Text record or
the size of the memory range while reading the header record, and relocatable (relocatable symbol) is an empty string.
Calling Sequence
Input Parameters: String (tp, addr, v)
Output Parameters: New Ordered_Four in which String parameters are equal to tp, addr, v and empty string.
Requires: Strings as input parameters.
Modifies: Creates a new Ordered_Four as the output with the input String tp, addr, v and empty string.
Ensures: A brand new Ordered_Four is made with tp, addr, v and empty string as its String parameters.
Modification Log: No changes needed.

Method: Ordered_Four (String, String, String, String)
Description: It is a constructor which creates a four tuple that consists of a String tp (type such as Header Text or End record), String addr (address or Location Counter), String v (value) is either the contents of address when reading a Text record, or the size of memory the range when reading the header record, and relocatable contains the relocatable symbols like X0/X1 followed by external symbol name, or M0/M1 followed by no symbol.
Calling Sequence
Input Parameters: String (tp, addr, v, r)
Output Parameters: New Ordered_Four in which String parameters are equal to tp, addr, v & r.
Requires: Strings as input parameters.
Modifies: Creates a new Ordered_Four as the output with the input String tp, addr, v & r.
Ensures: A brand new Ordered_Four is made with tp, addr, v & r as its String parameters.
Modification Log: No changes needed.

Method: getType
Description: Returns type of the Ordered_Four, which is in String and represents either a Header record, Text record or End record. If Header record then it also includes program name which is of maximum length 6.
Calling Sequence
Input Parameters: none
Output Parameters: String (type)
Requires: Valid Ordered_Four.
Modifies: Nothing
Ensures: Returns type (H + program name (of maximum length six), T or E) which is a String.
Modification Log: No changes needed.

Method: getAddress
Description: Returns address (Location Counter) of the Ordered_Four, which is in String and if type is an End record then it represents the starting location of where the program has to start reading.
Calling Sequence
Input Parameters: none
Output Parameters: String (address)
Requires: Valid Ordered_Four.
Modifies: Nothing
Ensures: Returns address (Location Counter) which is in Hex String.
Modification Log: No changes needed.

Method: getValue

Description: Returns value of the Ordered_Four, which is in String and if type is a text record then represents contents of the address. Otherwise if type is Header it represents the size of the memory range.
Calling Sequence
Input Parameters: none
Output Parameters: String (value)
Requires: Valid Ordered_Four.
Modifies: Nothing
Ensures: Returns a value (contents of address or the size of the memory range) which is in Hex String.
Modification Log: No changes needed.

Method: getRelocatable
Description: Returns relocatable of the Ordered_Four, which is in String and represents the relocatable symbols like X0/X1 followed by external symbol, or M0/M1 followed by no symbol.
Calling Sequence
Input Parameters: none
Output Parameters: String (relocatable)
Requires: Valid Ordered_Four.
Modifies: Nothing
Ensures: Returns relocatable which is a String.
Modification Log: No changes needed.

Method: setType
Description: Alters the record type (type) of the Ordered_Four, which is contained in the Intermediate Table (IT), and is in String representation.
Calling Sequence
Input Parameters: String (tp)
Output Parameters: Nothing
Requires: Valid Ordered_Four.
Modifies: Alters the String type (record type) of Ordered_Four.
Ensures: Sets the record type (type) to tp.
Modification Log: No changes needed.

Method: setAddress
Description: Alters the address (Location counter) of the Ordered_Four, which is in String representation.
Calling Sequence
Input Parameters: String (addr)
Output Parameters: Nothing
Requires: Valid Ordered_Four.
Modifies: Alters the String address (Location Counter) of Ordered_Four.

Ensures: Sets the address/Location Counter (address) to addr.
Modification Log: No changes needed.

Method: setValue
Description: Alters the value (contents of address in text record or
memory size range in Header record) of the Ordered_Four, which is in String representation.
Calling Sequence
Input Parameters: String (v)
Output Parameters: Nothing
Requires: Valid Ordered_Four.
Modifies: Alters the String value (contents of address in text record or
the size of the memory range in Header record)
of Ordered_Four.
Ensures: Sets the contents of address or memory size (value) to v.
Modification Log: No changes needed.

Method: setRelocatable
Description: Alters the relocatable symbol (relocatable) of the Ordered_Four,
which is in String representation.
Calling Sequence
Input Parameters: String (r)
Output Parameters: Nothing
Requires: Valid Ordered_Four.
Modifies: Alters the String relocatable (relocatable symbol) of Ordered_Four.
Ensures: Sets the relocatable symbol (relocatable) to r.
Modification Log: No changes needed.