|  |
| --- |
|  |
| User’s Guide |
| Lab 3-Group BEERZ |
|  |
| **Zach Smith** |
| **Brad Kline, Elliot Schumacher, Evan Todd, and Ryan Powers** |
| **3/11/2011** |

|  |
| --- |
|  |

# Table of Contents

# Introduction

This user’s guide describes the capabilities and requirements of the W11 Loader. This program takes a list of multiple object files, and links them together to create one absolute executable, which can then be run using the W11 Simulator. The input files to the Loader must be of the output format specified in the W11 Assembler user’s guide, and the output executable file will be of the format described as the input to the W11 Simulator. This user’s guide assumes that the reader has knowledge about the W11 Assembler and W11 Simulator and their associated file formats, and the meaning of a Linking Loader.

# Installing the Software

The W11 Loader will run on any platform that supports Java Runtime Environment, version 1.6.0. The Loader and the Common folder together require around 500 KB of disk space, and minimal memory.

Installation of every component can be completed as detailed in the Integration User’s Guide. Installing only the Loader and Common components can be performed by the command:

> javac Common\\*.java Assembler\\*.java

after the program has been unzipped and the directory path is in this location. An equivalent command is:

> build-ld

# Running the Loader

The basic command to run the Loader is:

> java Loader.Main [inputfiles] –o outfile [options]

Or its equivalent:

> ld [inputfiles] –o outfile [options]

This command requires at least one input file, which must be of the format output by the Assembler (refer to the Assembler User’s Guide for a detailed description), and may be absolute or relocatable. If multiple input files are provided, only one may be absolute, and it must be the first file listed. The list of input files must be followed by a “-o”, which signals the end of the list. There must then be an outfile, which will specify the file to write the linked output, which will follow the format specified in the Simulator User’s Guide. The placeholder [options] is where any additional flags would go in the command.

To view the possible flags for the Loader, type any of the following commands:

> java Loader.Main --help

> java Loader.Main

> ld --help

> ld

This will produce the output to the console:

Useage: java Loader.Main [inputfiles] –o outfile [options]

-l Generate listing

-a addr Relocate program to addr <4-digit hex memory address>

Note: if linking an absolute object file, it must come first in the file list.

The output file of the Loader will be the linking of all of the given input object files, in that they will all be continuous in memory following the first input object file. Also, any modifications that need made to a relocatable file will be resolved.

# Loading an Object File

Running the Loader on a single object file will do nothing if the file is absolute. This is because the goal of the Loader is to link multiple programs together and make them contiguous in memory. If the single object file is relocatable however, the Loader will resolve memory locations of the modification records based on the address passed in as a command line argument. For example, running the loader with an input file named test5.out that has the following contents:

HTest5 MMMM0018

T00000004

T00012200M0

T0002e007M0

T0003f022

T0004127f

T00050203M0

T0006400cM0

T00070068

T00080069

T00090021

T000a0020

T000b0000

T000c5020

T000d9000

T000e3013M0

T000fea13M0

T00102c17

T00117141

T0012f025

T00160010

T00170064

E0001

> java Loader.Main test5.out –o test5

Notice –a is not used. This will produce the following output, relocated as if the program was at memory location 0 :

HTest5 00000018

T00000004

T00012200

T0002e007

T0003f022

T0004127f

T00050203

T0006400c

T00070068

T00080069

T00090021

T000a0020

T000b0000

T000c5020

T000d9000

T000e3013

T000fea13

T00102c17

T00117141

T0012f025

T00160010

T00170064

E0001

This output can then be used by the Simulator, which will execute the program this object file outlines. Using –a with any 4-bit hex value will change the address to which the object file is loaded.

# Loading Multiple Programs

This section will describe how to load multiple programs into memory, and will provide examples. It will cover multiple relocatable programs, and one absolute program with multiple relocatable programs.

## Multiple Relocatable Programs

Running the Loader with multiple relocatable programs can be done with a command like:

> java Loader.Main test1a.out test1b.out –o test1

if test1a.out is:

HTest44MMMM0002

I000040Symbol

I0001f0Symbol

T00001060

T00010000

E0000

and test1b.out is:

HTest46MMMM0001

XR0000Symbol

T00001061

E0000

The output of this command will then look like:

HTest4400000003

T00001062

T00010002

T00021061

E0000

Notice that there are no “X” or “I” records present in the output, and that the text records from the two files have been combined into one linked file. Also, the file has been relocated to the address specified, which was x0000, as we provided no –a flag.

## One Absolute Program and Multiple Relocatables

When running the Loader, only one object file may be absolute, and it must be the first one in the list of object files. A command to the Loader with one absolute object file and more than one relocatable ones would look like:

> java Loader.Main test2abs.out test2b.out test2c.out –o test2

where test2abs.out is an absolute object file:

HMain 30000005

XR3000Start

I300080Displ

I300180V

I300280X

I300380Displ

T30004800

T30012200

T30023200

T30034800

T3004f025

E3000

and test2b.out is relocatable:

HMesg MMMM001d

XR000cDispl

XR0008X

T00000056

T00010061

T0002006c

T00030075

T00040065

T0005003d

T00060020

T00070000

T00080006

T000c3009M0

T000d320aM0

T000e3e0bM0

T000f2208M0

T00100819M0

T0011e000M0

T0012f022

T00132008M0

T0014f031

T0015103f

T00163008M0

T00170819M0

T00184011M0

T00192009M0

T001a220aM0

T001b2e0bM0

T001cd000

E000c

and test2c.out is relocatable:

HData MMMM0005

XR0000V

T00000002

T0001f043

T0002f025

T00032204

T00040001

E0002

This produces the following output:

HMain 30000027

T30004811

T30012222

T3002320d

T30034811

T3004f025

T30050056

T30060061

T3007006c

T30080075

T30090065

T300a003d

T300b0020

T300c0000

T300d0006

T3011300e

T3012320f

T30133e10

T3014220d

T3015081e

T3016e005

T3017f022

T3018200d

T3019f031

T301a103f

T301b300d

T301c081e

T301d4016

T301e200e

T301f220f

T30202e10

T3021d000

T30220002

T3023f043

T3024f025

T30252204

T30260001

E3000

This output object file is the absolute input file concatenated with the other relocatable object files, with any modification records resolved.

# Error Messages

Note: the pound sign (#) represents a number, and it may vary in certain situations; any word between brackets [ ] indicates the same.

The following list is sorted alphabetically.

##### "Failed to open file [file] for reading."

**Cause**: The file given as an input does not exist in the directory. If a list of files is given as input, then one of them does not exist.

**Solution**: Ensure that the file(s) inputted to the Loader exist and are in the directory. Note that this error will occur sequentially if the input was a list of files, i.e. it will catch the first file that does not exist and exit, without looking at the rest of the files.

##### "Failed to open file [file] for writing."

**Cause**: The file designated as the output is unable to be opened.

**Solution**: Ensure that the desired file is not one that already exists and is currently open, or one that you do not have permissions to modify.

##### "Object file [file] was expected to be relocatable, but is not."

**Cause**: A file inputted to the Loader was absolute, but the –a flag was used to signal a relocatable output.

**Solution**: Either make the absolute file relocatable (by altering its assembly language code) or remove the –a flag from the call to the Loader.

##### "Resulting program would span multiple memory pages. First address: # Last address: #"

**Cause**: The linked object file spans multiple memory pages.

**Solution**: If the inputted object file(s) are relocatable, try to relocate them to the first address of a new page. If the object file list has an absolute member, try to change the absolute address to which to load the program. Otherwise, reduce the number of text records in the assembly program.

##### “Useage: java Loader.Main [inputfiles] –o outfile [options]

##### -l Generate listing

##### -a addr Relocate program to addr <4-digit hex memory address>

##### Note: if linking an absolute object file, it must come first in the file list. “

**Cause:** The arguments given to the Loader were incorrect.

**Solution**: Check the number, order, spelling, and completeness of the arguments passed to the Loader. The –o is mandatory.

The following errors are redundant in the fact that, if the input object files were assembled correctly, these errors should never be encountered. Regardless, they are listed here because the possibility of them arising does exist.

##### "Execution start address 0x#### outside specified memory segment range."

**Cause**: An input object file had an invalid end record.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "First character of line is invalid; must be 'H', 'T', 'E', 'I', or 'X'."

**Cause**: An input object file had an invalid record.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Length of end record is incorrect. Should be 5 characters."

**Cause**: An input object file had an invalid end record.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Length of export record is incorrect. Should be at least 7 characters."

**Cause**: An input object file had an invalid export record.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Length of header record is incorrect. Should be 15 characters."

**Cause**: The length of the header record in an input object file is longer than 15 characters.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Length of import record is incorrect. Should be at least 6 characters."

**Cause**: An input object file had an invalid import record.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Length of text record is incorrect. Should be 9 or 11 characters."

**Cause**: An input object file had an invalid text record.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Malformed modification record. Must be "M0" or "M1"."

**Cause**: An input object file had an invalid modification record.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Memory segment length is too large for virtual machine."

**Cause**: The segment in memory is larger than the total memory of the machine.

**Solution**: Change the starting address of the first assembly language program in the list of inputs and reassemble it, or make that program shorter.

##### "Object file does not contain a header record."

**Cause**: An input object file does not have a valid header record.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Object file does not contain an end record."

**Cause**: An input object file does not have a valid end record.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Second character of export record should be 'A' for absolute or 'R' for relative."

**Cause**: An input object had an invalid export record.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Symbol name must not contain spaces or commas."

**Cause**: An input object file uses an invalid character in a symbol name.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Symbol name must not start with 'x', 'R', '#', or '='."

**Cause**: An input object file uses an invalid character in a symbol name.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.

##### "Text record address 0x#### exists outside program memory range specified in header record."

**Cause**: An input object file references a memory address that does not exist in the segment.

**Solution**: Reassemble the input object files; one or more of them did not assemble correctly or was modified since assembly.