LAB #3: Linker/Loader Lab

DUE: Friday March 13.

Part 1 (Modifying Assembler)

Modify the assembler you built in Lab 2 so that it is capable of dealing with external symbols.

Specifically, your assembler should be capable of processing the following two pseudo-ops, in addition to the ones described in Lab 2.

Mnemonic	Meaning
.ENT	[**ENTry name**] The operand field of this pseudo-op is a list of symbols that must be defined in the current segment, and are permitted to be referenced by other segments.
.EXT	[**EXTernal name**] The operand field is a list of symbols that may be referenced legitimately by this segment, but are not defined in this segment. The symbols must be defined in some other segment.

For both the ENT and EXT pseudo ops, there must not be a label, and there must be an operand. You may, at your option, require the following:

- that the list of symbols be bounded by some small number, such as four
- that these pseudo-ops, if present, precede other pseudo-ops and executable instructions (except, of course, .ORIG)
- that all external symbols be relative

Appropriate error messages should be written if there are violations of the syntax rules for the ENT and EXT pseudo-ops.

The object file produced by your modified assembler is up to you, but it must contain all the essential information (e.g., text, relocation information, length, start address for execution) as well as appropriate information about external symbols, so the loader can perform any linking needed. Describe the format of this file thoroughly in your documentation.

Your modified programmer's guide, user's guide, and test plan should highlight the parts that have changed as a result of the modification. (Either change bars in the margin, or a separate page listing sections that have changed from the Lab 2 submission are sufficient.)

Part 2 (Linking Loader)

Write a (relocating) linking loader for our abstract machine. Your system must be capable of handling multiple source files, and linking them together in order to form the executable file that is input to the simulator. If multiple files are linked, they are all required to be relocatable.

The format of the input file is the same as the output file produced by your modified assembler. The output of the loader should be an absolute executable file, ready to be passed to the 560 machine simulator which will interpret the instructions. *The format of the output file should be exactly that described in Lab 1 as input.*

Note that your loader will need an initial program load address, which normally is provided by the operating system. Since we do not have a "560-machine operating system", your loader will have to insert some appropriate value (probably not zero). This should be inserted by the user from the command line or from a prompt.

Have your loader be capable of detecting "invalid relocation information" errors, "undefined external symbol" errors, errors in any link records used, as well as the other errors associated with the loader specified in the Lab 1 handout.

Part 3 (Integration)

Integrate the assembler (produced in part 1 of Lab 3), the loader (produced in part 2 of Lab 3), and the 560-machine simulator that you wrote in Lab 1, to produce a "system" capable of processing a (correct) source program written in the 560-machine's assembly language from translation through linking, loading, and execution.

The documentation you turn in for lab 3 should include a brief appendix which gives instructions for using the three components in tandem. Include here any appropriate references to the user's guides for the emulator, assembler and loader for more detailed information about these components. You should also produce an integration test plan for the assembler, loader, and simulator working in tandem, and the results of this testing. It may be somewhat more brief than the test plans for the individual labs since it is easier to demonstrate that the three components work together given that they work separately.

Approximate breakdown of Lab 3 grade

Modified Assembler Documentation, Testing, Code: 15%
Linking Loader User's guide: 10%
Linking Loader Programmer's Guide: 10%
Linking Loader Test Plan: 5%
Integration User's Guide and Testing and Meeting Minutes: 20%
Grader's evaluation of "smoothness of integration": 5%
Grader's Testing and evaluation: 25%
Individual grade: 10%