

CST8134: In-lab Exercise 6

.lcf files for multiple program files

You will use the completed function `ReadLine` in a test program you will name `ReadTest` to read lines of ASCII text from the keyboard. The address and the length of the buffer will be passed to `ReadLine` in registers `a0` and `d0`. Once a line has been read, print the length of the line read (returned in `d0`) and the contents of your buffer. Continue until `ReadLine` indicates that end-of-file has been reached by returning `-1` instead of the length.

Assemble the two source files, `ReadTest.a68` and `ReadLine.a68`, separately. Correct any errors that may have occurred and re-assemble until your test program assembles cleanly. Do not link the files yet. Get help from your instructor if `ReadLine.a68` doesn't assemble properly at once.

A normal `.lcf` file looks more or less like this:

```
ENTRY    (ReadTest)

OUTPUT   (ReadTest.hex)

OUTPUT_FORMAT (srec)

SECTIONS {
    .text 0x1000 :
    {
        ReadTest.o(.text)
        ..\lib\bdblib.o(.text)
    }

    .data 0x2000 :
    {
        ReadTest.o(.data)
        ..\lib\bdblib.o(.data)
    }
}
```

This particular file will link only your `ReadTest.o` object file (`ReadTest.o`) to the BDB library routines (`bdblib.o`), so it's clearly inadequate. Modify it by inserting this line:

```
    ReadLine.o(.text)
```

immediately after the similar line in the text section for `ReadTest.o` (you do not need a `ReadLine` entry for the data section – why not?) Now link both object files with the BDB library to create a single executable, `ReadTest.hex`.

From now on in this course, you will write each function that you need as a separate source file and link them together in a manner like this. This method will work for any (reasonable) number of object files.

Use the debugger to make sure your program operates correctly, fixing any bugs and re-assembling and re-linking as necessary, and then demonstrate it to your lab instructor well before the end of your lab period.