COMP 2655 Assignment 5: Arrays of Structures

Given:	Wednesday, November 8, 2017
Due:	Sunday, November 19, 2017 before 11:59pm
Hard Copy Due:	Monday, November 20, 2017 before 1:59pm
Weight:	6%

Objectives:

- To gain more experience with assembly language and the 68000.
- To continue to practice translation of high-level algorithms to assembly language.
- To learn more about how arrays, structures and arrays of structures are represented at the low-level.
- To understand how to use the "register indirect with offset" and "indexed register indirect with offset" addressing modes.
- To work with simple subroutines using the correct method of parameter passing
- To continue to gain experience with testing, debugging and documenting low-level code.

Overview

This assignment is to write a 68000 assembly language program which is a *prototype*, a small demonstration program, for an interactive movie database. You are **not** required to implement a version in C, however, it is recommended that you still do this. **It is always easier to translate an algorithm or a program in a high level language to assembly language.**

Since you have no experience with binary files an initial program, `a5_start.c`, has been provided in the class directory on INS. This program is an updated version of `first.c` that declares a movie structure, reads the binary data file into an array of structures and then outputs the array contents using printf. This code should be modified to do a quick version of this assignment in C using `write_char`. Don't worry about saving the file in C.

The Problem

The movie database is simply an array of records that can be manipulated in a number of ways. In a full program there would be many operations, but since this is a prototype only a couple of operations will be implemented. Being interactive means that the user will enter requests from the keyboard and results will be displayed on the screen. The directory is to be read in from a file and stored in an array of records. The array should hold a maximum of 25 records. Records will be stored in alphabetic order and this order must be maintained during program use. The records within the address file will have the following format:

FIELD NAME	TYPE
Movie Title	char[50+1]
Year released	uWord
Running Time	uByte
Rating	char[5+1]

The data in the file is stored in fixed format, i.e. each field is filled completely. If a field contains data that is shorter than the maximum length it is padded with NULLs, ex. the Title field for the movie Up!

would use 3 characters for the title and the remaining 48 characters would be NULLs. Any holes required in the assembly implementation will be present in the file.

A data file called `movies_db.dat` that meets the given specification is available. The program will generate an output file, discussed later. It is recommended that you name the output file `movies.out` and that these file names are hardcoded into your program.

I/O Information

A GST linkable module called `A5LIB.BIN` is provided that contains all of the I/O routines required for the assignment. These consist of the common `exit` subroutine and the two basic console I/O subroutines, `read_char` and `write_char`, that have been revised to correctly pass parameters. Two additional file I/O routines, `read_file` and `write_file`, are supplied. The former reads binary data from a file into an array and the latter writes the binary contents of an array to a file. Complete descriptions of these routines can be found in the file `A5LIB.TXT`. All of the given subroutines follow the C convention for parameter passing, i.e. parameters are pushed on the stack from right to left.

The default drive for both file operations is `D:\`, however, including the drive and path in the file name will override the default.

Operations

The program MUST provide the following options. Each option MUST be coded as a proper subroutine, meaning that correct parameter passing as discussed in class MUST be used.

1) LIST

The operation will print all entries in the database to the screen. All fields for an entry are to be output with each field being prefaced with a descriptive label. The output should be formatted so that each entry is easily readable and separated from other entries. Leading each entry is to be a record number (i.e. its position in the array) however; in order to accommodate users, these values will start from 1.

2) LIST by YEAR

For this operation the user must supply a desired year and all movies whose release year matches the desired year are to be output in the same format as specified in the LIST operation.

3) DELETE

For this operation the user will supply the record number to be removed from the database. If invalid, an error message should be output; otherwise the entry should be deleted. Since the write_file subroutine only writes consecutive array entries all entries following the removed entry must be shifted in the array.

4) QUIT

Saves the current database to a file and exits.

The interface design is left up to you, since this is a prototype making it extremely fancy is not required, however, it must be easily readable. The Atari screen accepts VT52 screen commands which can be found on page 301 of the Devpac manual or in the file ST_CALL.DOC.

Subroutines

The use of basic subroutines are allowed for this assignment, however, they must follow the standards for subroutines as discussed in class. The use of common registers or global variables for parameter passing is completely unacceptable and will result in a substantial mark deduction.

The creation of the following subroutines is recommended:

`read_number`	- to read and return a word size number from the keyboard
`write_number`	- to write a word sized number to the screen
`write_string`	- to write a null-terminated string to the screen
`print_entry`	- to print a single database entry to the screen

Coding Style and Documentation

You must follow good coding style and documentation practices as discussed in class and documented in the 2655 coding standards.

Submission

Using a program like FileZilla, copy **ONLY** your assignment 5 ".S" file to INS. Do not copy A5LIB.BIN, MOVIES.TXT, any ".BIN" files, or the executable.

Log in to INS, and then use the submit program to submit your ".S" file.

Use the lpr command to print your ".S" file on a wide-carriage printer. Then deposit the hard copy in your instructor's drop box outside of B170.

For marking your instructor will assemble, link (to his own copy of the library routines), and run your program. If your solution is not complete make sure this is documented in your status section.