CSE12 Lab 4: Simple CSV File Analysis

Due at 11:59 PM on the marked due date

Objective

The objective of this lab is to learn about function calling, RISC-V protocols for the use of registers.

This lab takes as input a CSV (comma separated values) file (.csv extension) that is used for generating tabular data in spreadsheets. ***Specifically, for this assignment, you will NEED to assume this CSV file was generated under Windows*** (the reason will be explained shortly). Consider the *data.csv* file below as it appears when you open it in Excel as an exa

This file shows ntains the stock name and the "B" col ***ock return in all our CSV files***).

You will run the ut CSV file. Doing so will yield the fol ll be submitting):

1. Find the to
2. List the do
3. Provide the
4. Provide the
5. Calculate t

When you run vi *ou*, you will get the output console a

```
==========================
Kruger Industrial Smoothing,365
Kramerica,0
Vandelay Industries,500
Pendant Publishing,100
J. Peterman Catalog,42
==========================
Size of file data (in bytes): 119
income records: 365 0 500 100 42
Total income garnered from all stocks: $1007
Stock name with maximum income:Vandelay Industries
Stock name with minimum income:Kramerica

-- program is finished running (0) --
```

Clear

Figure 2 After running the lab4_testbench_rev##.asm file with the Lab4 assignment fully completed

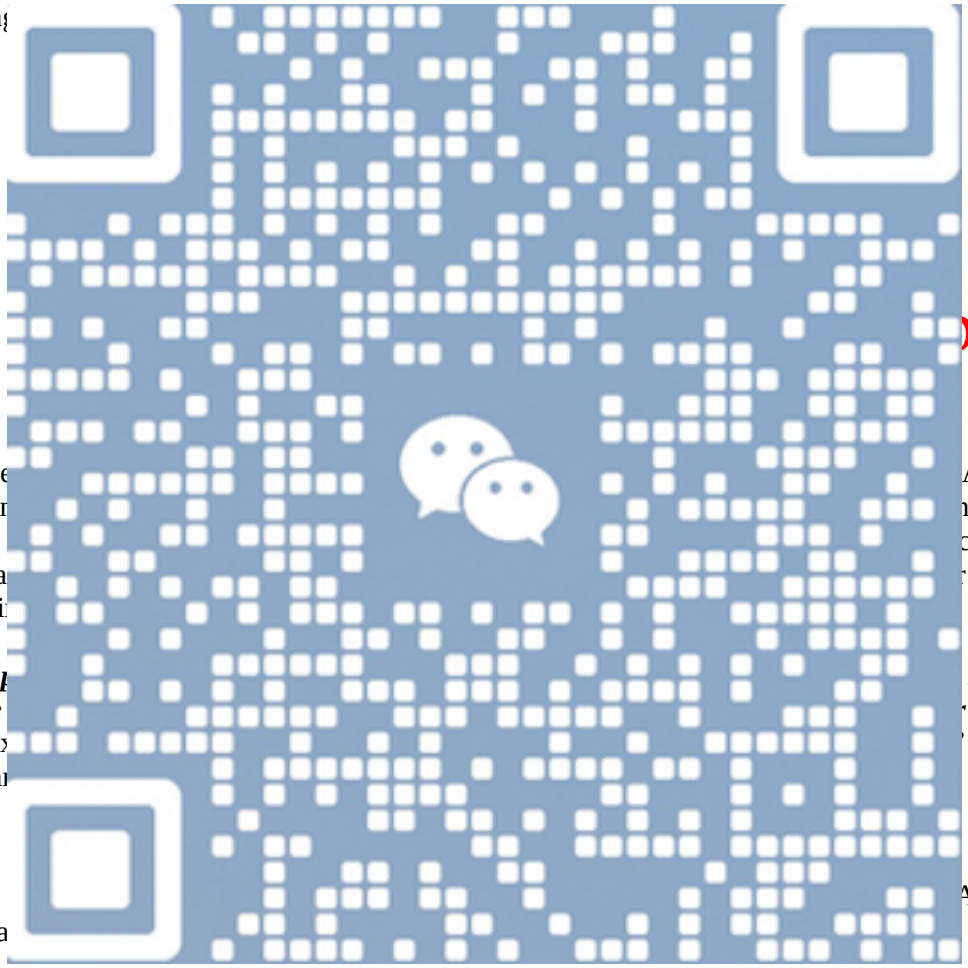
About the Windows CSV file format

To distinguish between each entry/row/record in the spreadsheet format of the CSV file, the following convention is adopted depending on the OS :

Windows - Lines end with a <CR> and a <LF> character
Linux - Lines end with only a <LF> character
Macintosh (Mac OSX) - Lines end with only a <LF> character
Macintosh (old) - Lines end with only a <CR> character

where <CR> is the carriage return ('\r') character and <LF> is the line feed/newline ('\n') character.

If you open the provided *data.csv* file in Notepad++ on Windows with "Show all Characters" enabled, then you should see the followin



So, for example ARS, I would write: "Kramerica,0\r\n not open correctly due to the encoding characters, make sure that the "\r\n" a the data.csv that we include in the di

Another assump name of the stock is followed by the in base 10. So, with record 2 as an ex where the 2 red dots indicate two bla

Resources

Much like how ARS based RISC-V programs have a

In the Lab4 folder in the course you will see 9 assembly files. They are meant to be read *(and understood)* **in sequence** and they will provide you with lotsof hints as to how to build your program:

1. *add_function.asm* – This program accepts two integers as user inputs and prints their addition result. The actual addition is done through calling a function, sum. Sum accepts two arguments in the a0, a1 registers and returns a0+a1 in a0 register

2. *multiply_function.asm* – This program accepts two integers as user inputs and prints their multiplication result. The actual multiplication is done through calling a function, multiply. Multiply accepts two arguments in the a0, a1 registers and returns a0*a1 in a0 register. This function in turn calls the function sum, described previously, to do a particular addition. Thus, multiply function is an example of a ***nested function call***, a function which itself calls another function, sum in our case.

3. *The lecture slides provide a lot of information about register calling and saving conventions.* Studying the comments in add_function.asm, multiply_function.as alongside with the notes should be sufficient to create to allow

you to complete this assignment.

4. ***lab4_testbench_rev##.asm*** - This is the main testbench program you will run upon completion of all coding in Lab4 to ensure your Lab4 assignment works as expected. This file is initially provided such that if you run it as it is (with the other .asm files in the same directory), you will still get partially correctly generated output. This testbench will also run the the function *allocate_file_record_pointers* from *the allocate_file_record_pointers.asm* which will aide you in writing your program. **DO NOT MODIFY THIS FILE.**

5. ***allocate_file_record_pointers.asm*** - This .asm file contains a function that **creates an internal reference table** in memory to pointer pairs. These pointer pairs indicate 1) the location of the start of a string corresponding to the stock name and 2) the start of a location containing the stock price for each and every record/entry coming from the data.csv file. This function has been fully written out for you. **DO NOT MODIFY THIS FILE.**

6. macros_rev#. *a# registers, so be careful. Become*

6. ***income_fron*** the string data from the income of a e string "1234" into the actual intege

7. **income_from** om_record.asm.

7. ***length_of_file*** unt of data bytes in the csv file. Refe

8. ***maxIncome.a*** me of the stock that has the maximu

9. ***minIncome.a*** of the stock that has the minimum in ed later)

10. ***totalIncome*** tock incomes in the csv file. Refer to

Please downloa rwise the comments and other impor learning assembly language intuitiv

These files hav RISC-V assembly programming fo ng.

Beyond these th ***mselves. The slides are very self-explanatory and it is encouraged you start reading them even if the instructor hasn't started discussing them in lecture.***

For the usage of macros (which are utilized heavily in this lab to generate ecalls), please also refer to the RARS documentation on macros and ecalls as well.

Please read the provided files carefully. You can learn a lot about assembler from reading these files. Note that using macros resembles calling functions. The macros have been written to make use of the a# registers, so don't assume that any values in your a# registers remain valid after calling a macro.

Memory arrangement as defined in Lab4

The memory of RISC V is used as per the given requirements.

File Data Buffer

The **data.csv** file is treated as the default input file. It's contents are store in the file buffer location 0xffff0000 (referred to as MMIO).

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0xffff0000	g u r K	I r e	s u d n	a i r t	m S l	h t o o	, g n i	\r 5 6 3
0xffff0020	a r K \n	i r e m	0 , a c	a V \n \r	l e d n	I y a	s u d n	e i r t
0xffff0040	0 5 , s	P \n \r 0	a d n e	P t n	i l b u	n i h s	0 1 , g	J \n \r 0
0xffff0060	e P .	m r e t	C n a	l a t a	4 , g o	\0 \n \r 2	\0 \0 \0 \0	\0 \0 \0 \0
0xffff0080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0xffff00a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

Figure 4 data segment window for MMIO after running completed Lab4 assignment.

The input is achi…ecuted. Your code *does not need to*

For reference, …"**Kruger Industrial Smoothing,365**…r 64 in base 10) **is at 0xffff0000**. The …as byte 0x33 or 51 in base 10) **is at 0x**…n" with 'K' being the 0th character, th…

File Recor…

We need a syste…from the file buffer at 0xffff0000. Rem…characters. Thus, **we set aside the me…ses) of the first character appe…**for the number) of each record in th…

Address	Valu…	…ue (+18)	Value (+1c)
0x10040000		0xffff0047	0xffff005a
0x10040020		0x00000000	0x00000000

This is ach…provided function ***allocate_file_rec…***ytes (119 in our given example from *d*…mple from *data.csv*), respectively. T…*ord_pointers.asm* file separately throu…

For reference, **f**…tart of **income name (0xffff0000**) and **income value(0xffff001c)** are stored as words **in consecutive heap memory locations 0x10040000** and **0x10040004** respectively. Likewise, the **second record**'s (i.e. "**Kramerica,0\r\n**") location of start of **income name (0xffff0021)** and **income value(0xffff002b)** are stored as words in consecutive heap memory locations 0x10040008 and 0x1004000c respectively. And so on and so forth. ***It is left as a HIGHLY recommended exercise that the student verifies this pattern for the remaining records in the CSV file and how they are allocated in the memory locations as shown in Figure 5.*** As we see in Figure 5, the 10 non zero heap memory locations from 0x10040000 to 0x10040024 indicate there were originally 10/2 = 5 records in our *data.csv* file. This value (no. of records) is returned in a0.

Student coded functions

All student coded functions are to be written in the .asm files listed in the *lab4_testbench_rev##.asm* file using the .include statement (excluding *allocate_file_record_pointers.asm*). ***The functions written MUST abide by the register saving conventions of RISC-V.***

1. **length_of_file(a1)** - This function is to be written in *length_of_file.asm.* It accepts as argument in a1 register the buffer address holding file data and returns in a0 the length of the file data in bytes.
From our example involving *data.csv*, length_of_file(0xffff0000)=119

2. **income_from_record(a0)** - This function is to be written in *income_from_record.asm.* It accepts as argument in a0 register the poir lue in a0.

From our examp me from the stock of Kruger Industria stock of Kramerica.

You may use the on.

3. **totalIncome(** record pointer array location (0x100 ntains the number of records in the C)

From our initial

4. **maxIncome(** record pointer array location (0x100 ntains the number of records in the C ord stock name in the file buffer.

From our exam m Figure 5 that this address value po he maximum value of $500, and provi g as ever.

5. **minIncome(a** record pointer array location (0x100 ntains the number of records in the C ord stock name in the file buffer.

From our exam m Figure 5 that this address value points to the location of the stock name "Kramerica", which is valued at the minimum value of $0, proving once and for all, a sales pitch about a "coffee table book about coffee tables" is an absolutely terrible idea.

To keep the code simple for both maxIncome and minIncome functions, you may assume that no two entries in the CSV file will have the same income field value.

Test Cases

Your Lab4 Google Drive folder called TestCases contains more test data files: *data1.csv, data2.csv* and *data3.csv, and data0.csv*. To test with these you have to copy any one of them to data.csv in your working directory. The initial data.csv you use by default is the same as TestCases/data0.csv. It corresponds to the file described in this write-up.