Programming Lab 2: Historical Timelines

EE 306: Introduction to Computing
Professor: Dr. Nina Telang
TAs: Vignesh Radhakrishnan, Jefferson Lint, Suhas Raja, Jerry Yang

Due: October 19, 2019

1 Overview

This lab is intended to expose you to searching and sorting. By the end of this lab, you should be able to:

- implement a search algorithm in LC3 assembly.
- implement a sort algorithm in LC3 assembly.
- utilize loops and branches in a program.

Note: You may NOT talk to or show anyone other than the TAs or the professor your code. **Any violation of this rule constitutes academic dishonesty.**

2 Background

UT Austin and the Cockrell School of Engineering have a rich history that has built them into respected academic instiitutions. In fact, UT Austin was established and chartered as a public school in the 1876 Texas constitution. The school officially opened its doors on September 15, 1883. The Cockrell School of Engineering was established in 1894, and the electrical engineering department was established in 1903.

In this lab, we will explore the histories of UT Austin, Cockrell, and ECE by constructing various timelines from a given database of UT-related events and their corresponding dates (stored in the given `Lab2data.asm` file). Throughout this lab, we encourage you to explore events listed in the database, as they are interesting milestones for the school outside the context of this lab.

3 Lab Specifications

The input dataset for your lab will have the form of a null-terminated list of historical events and dates, stored at x4000 (see Table 1). The events in the list will come directly from the provided database (`Lab2data.asm`). The events are stored as pointers to null-terminated strings located elsewhere in memory, not necessarily right after the list. The dates are stored as four-digit numbers with each digit as a nibble (for example, year 1970 is represented as x1970 in memory). Table 1 gives an example of an input list.

Address	Value
x4000	x4007
x4001	x1970
x4002	x400B
x4003	x2001
x4004	x0000
x4005	–
x4006	–
x4007	‘E’
x4008	‘C’
x4009	‘E’
x400A	x0000
x400B	‘C’
x400C	‘I’
x400D	‘V’
x400E	‘I’
x400F	‘L’
x4010	x0000

Table 1: Example list

3.1 Part A: Sort

Write a program that creates a timeline of historical events by date from most recent to least recent by sorting the given input list. If two events happen in the same year, sort them alphabetically. In this case, “alphabetically” is defined as the following order: “ABC...XYZabc...xyz”. So “Thirty people” comes before “thirty people”.

3.2 Part B: Search

Write a program that creates a frequency table of historical events by half-century and stores it at x6000. A frequency table lists the number of items in the list that fall within a given interval, inclusive. The intervals you will use are: before 1900, between 1901 and 1950, between 1951 and 2000, and 2001 to today. If an event occurs in 1900, it should be placed in the “before 1900” interval, and similarly, if an event occurs in 1901, it should be placed in the “1901-1950” interval.

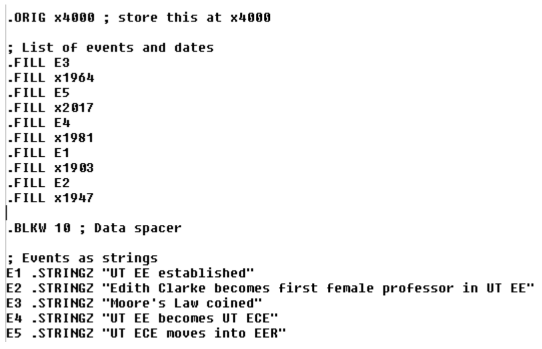
The frequency table should be stored as a null-terminated list with the beginning year of the interval and the number of events that fall into the interval. For example, if you have 2 events before 1900, 3 events between 1901 and 1950, 2 events between 1951 and 2000, and 1 event since 2001, your frequency table should look like Table 2.

3.3 Testcase

This is an example testcase, taken from the given database. The testcase is provided to you along with this lab document. To use the testcase, assemble the .txt file in the LC3 editor, then load the generated .obj file into LC3 before loading your programs. The testcase is displayed as a text file (Figure 1a) and after being loaded into LC3 (Figure 1b). After Part A is run, your memory should look like Figure 2a. After Part B is run, your memory should look like Figure 2b.

Address	Value
x6000	x1900
x6001	x0002
x6002	x1901
x6003	x0003
x6004	x1951
x6005	x0002
x6006	x2001
x6007	x0001
x6008	x0000

Table 2: Frequency table



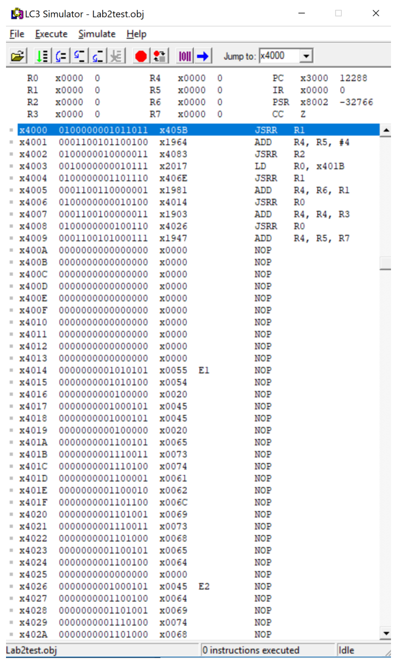
```
.ORIG x4000 ; store this at x4000

; List of events and dates
.FILL E3
.FILL x1964
.FILL E5
.FILL x2017
.FILL E4
.FILL x1981
.FILL E1
.FILL x1903
.FILL E2
.FILL x1947

.BLKW 10 ; Data spacer

; Events as strings
E1 .STRINGZ "UT EE established"
E2 .STRINGZ "Edith Clarke becomes first female professor in UT EE"
E3 .STRINGZ "Moore's Law coined"
E4 .STRINGZ "UT EE becomes UT ECE"
E5 .STRINGZ "UT ECE moves into EER"
```

(a) Testcase code



```
R0  x0000  0        R4  x0000  0        PC   x3000  12288
R1  x0000  0        R5  x0000  0        IR   x0000  0
R2  x0000  0        R6  x0000  0        PSR  x8002  -32766
R3  x0000  0        R7  x0000  0        CC   Z

x4000  0100000001011011  x405B       JSRR  R1
x4001  0001100101100100  x1964       ADD   R4, R5, #4
x4002  0100000010000011  x4083       JSRR  R2
x4003  0010000000010111  x2017       LD    R0, x401B
x4004  0100000001101110  x406E       JSRR  R1
x4005  0001100110000001  x1981       ADD   R4, R6, R1
x4006  0100000000010100  x4014       JSRR  R0
x4007  0001100100000011  x1903       ADD   R4, R4, R3
x4008  0100000000100110  x4026       JSRR  R0
x4009  0001100101000111  x1947       ADD   R4, R5, R7
x400A  0000000000000000  x0000       NOP
x400B  0000000000000000  x0000       NOP
x400C  0000000000000000  x0000       NOP
x400D  0000000000000000  x0000       NOP
x400E  0000000000000000  x0000       NOP
x400F  0000000000000000  x0000       NOP
x4010  0000000000000000  x0000       NOP
x4011  0000000000000000  x0000       NOP
x4012  0000000000000000  x0000       NOP
x4013  0000000000000000  x0000       NOP
x4014  0000000001010101  x0055  E1   NOP
x4015  0000000001010100  x0054       NOP
x4016  0000000000100000  x0020       NOP
x4017  0000000001000101  x0045       NOP
x4018  0000000001000101  x0045       NOP
x4019  0000000000100000  x0020       NOP
x401A  0000000001100101  x0065       NOP
x401B  0000000001110011  x0073       NOP
x401C  0000000001110100  x0074       NOP
x401D  0000000001100001  x0061       NOP
x401E  0000000001100010  x0062       NOP
x401F  0000000001101100  x006C       NOP
x4020  0000000001101001  x0069       NOP
x4021  0000000001110011  x0073       NOP
x4022  0000000001101000  x0068       NOP
x4023  0000000001100101  x0065       NOP
x4024  0000000001100100  x0064       NOP
x4025  0000000000000000  x0000       NOP
x4026  0000000001000101  x0045  E2   NOP
x4027  0000000001100100  x0064       NOP
x4028  0000000001101001  x0069       NOP
x4029  0000000001110100  x0074       NOP
x402A  0000000001101000  x0068       NOP

Lab2test.obj    0 instructions executed    Idle
```

(b) LC3 memory with testcase

Figure 1: Testcase before execution

3.4 Notes

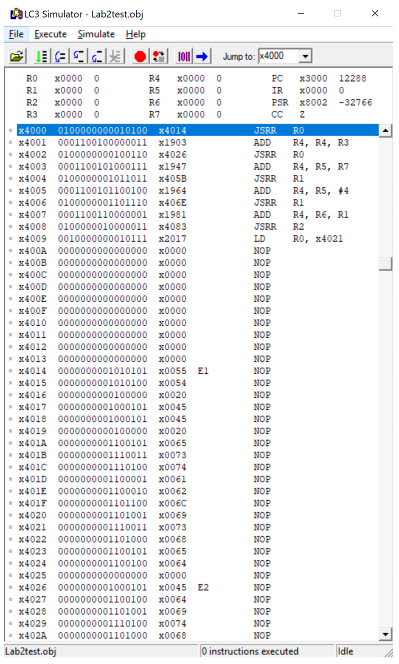
1. For your sort program, name your file [EID]_sort.asm (e.g. jay32_sort.asm). For your search program, name your file [EID]_search.asm (e.g. jay32_search.asm).

2. Start your programs at x3000. (The first line of your program should be .ORIG x3000.)

3. Your programs should not depend on each other (i.e. your solution to part B should not assume that the events are sorted before the program is run).

4. You can edit the provided testcase to create your own testcases that will have the same format.

5. You may assume that the events in the testcases will be printed exactly the same as the ones in the database.



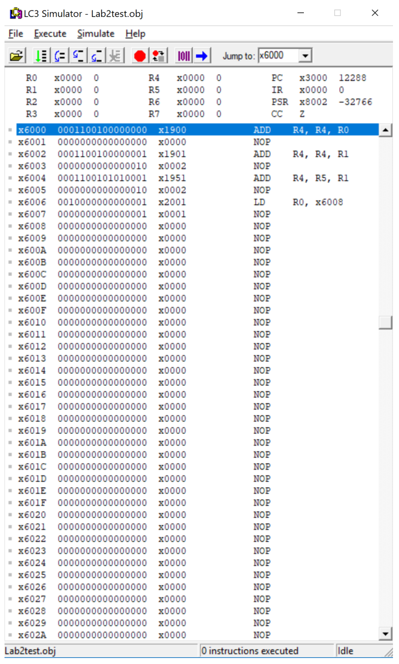
```
LC3 Simulator - Lab2test.obj
Jump to: x4000

  R0   x0000  0        R4   x0000  0        PC   x3000  12288
  R1   x0000  0        R5   x0000  0        IR   x0000  0
  R2   x0000  0        R6   x0000  0        PSR  x8002  -32766
  R3   x0000  0        R7   x0000  0        CC   Z

x4000  0100000000010100  x4014       JSRR  R0
x4001  0001100100000011  x1903       ADD   R4, R4, R3
x4002  0100000000100110  x4026       JSRR  R0
x4003  0001100101000111  x1947       ADD   R4, R5, R7
x4004  0100000001011011  x405B       JSRR  R1
x4005  0001100101100100  x1964       ADD   R4, R5, #4
x4006  0100000001101110  x406E       JSRR  R1
x4007  0001100110000001  x1981       ADD   R4, R6, R1
x4008  0100000010000011  x4083       JSRR  R2
x4009  0010000000010111  x2017       LD    R0, x4021
x400A  0000000000000000  x0000       NOP
x400B  0000000000000000  x0000       NOP
x400C  0000000000000000  x0000       NOP
x400D  0000000000000000  x0000       NOP
x400E  0000000000000000  x0000       NOP
x400F  0000000000000000  x0000       NOP
x4010  0000000000000000  x0000       NOP
x4011  0000000000000000  x0000       NOP
x4012  0000000000000000  x0000       NOP
x4013  0000000000000000  x0000       NOP
x4014  0000000001010101  x0055  E1   NOP
x4015  0000000001010100  x0054       NOP
x4016  0000000000100000  x0020       NOP
x4017  0000000001000101  x0045       NOP
x4018  0000000001000101  x0045       NOP
x4019  0000000000100000  x0020       NOP
x401A  0000000001100101  x0065       NOP
x401B  0000000001110011  x0073       NOP
x401C  0000000001110100  x0074       NOP
x401D  0000000001100001  x0061       NOP
x401E  0000000001100010  x0062       NOP
x401F  0000000001101100  x006C       NOP
x4020  0000000001101001  x0069       NOP
x4021  0000000001110011  x0073       NOP
x4022  0000000001101000  x0068       NOP
x4023  0000000001100101  x0065       NOP
x4024  0000000001100100  x0064       NOP
x4025  0000000000000000  x0000       NOP
x4026  0000000001000101  x0045  E2   NOP
x4027  0000000001100100  x0064       NOP
x4028  0000000001101001  x0069       NOP
x4029  0000000001110100  x0074       NOP
x402A  0000000001101000  x0068       NOP

Lab2test.obj              0 instructions executed    Idle
```

(a) After part A



```
LC3 Simulator - Lab2test.obj
Jump to: x6000

  R0   x0000  0        R4   x0000  0        PC   x3000  12288
  R1   x0000  0        R5   x0000  0        IR   x0000  0
  R2   x0000  0        R6   x0000  0        PSR  x8002  -32766
  R3   x0000  0        R7   x0000  0        CC   Z

x6000  0001100100000000  x1900       ADD   R4, R4, R0
x6001  0000000000000000  x0000       NOP
x6002  0001100100000001  x1901       ADD   R4, R4, R1
x6003  0000000000000010  x0002       NOP
x6004  0001100101010001  x1951       ADD   R4, R5, R1
x6005  0000000000000010  x0002       NOP
x6006  0010000000000001  x2001       LD    R0, x6008
x6007  0000000000000001  x0001       NOP
x6008  0000000000000000  x0000       NOP
x6009  0000000000000000  x0000       NOP
x600A  0000000000000000  x0000       NOP
x600B  0000000000000000  x0000       NOP
x600C  0000000000000000  x0000       NOP
x600D  0000000000000000  x0000       NOP
x600E  0000000000000000  x0000       NOP
x600F  0000000000000000  x0000       NOP
x6010  0000000000000000  x0000       NOP
x6011  0000000000000000  x0000       NOP
x6012  0000000000000000  x0000       NOP
x6013  0000000000000000  x0000       NOP
x6014  0000000000000000  x0000       NOP
x6015  0000000000000000  x0000       NOP
x6016  0000000000000000  x0000       NOP
x6017  0000000000000000  x0000       NOP
x6018  0000000000000000  x0000       NOP
x6019  0000000000000000  x0000       NOP
x601A  0000000000000000  x0000       NOP
x601B  0000000000000000  x0000       NOP
x601C  0000000000000000  x0000       NOP
x601D  0000000000000000  x0000       NOP
x601E  0000000000000000  x0000       NOP
x601F  0000000000000000  x0000       NOP
x6020  0000000000000000  x0000       NOP
x6021  0000000000000000  x0000       NOP
x6022  0000000000000000  x0000       NOP
x6023  0000000000000000  x0000       NOP
x6024  0000000000000000  x0000       NOP
x6025  0000000000000000  x0000       NOP
x6026  0000000000000000  x0000       NOP
x6027  0000000000000000  x0000       NOP
x6028  0000000000000000  x0000       NOP
x6029  0000000000000000  x0000       NOP
x602A  0000000000000000  x0000       NOP

Lab2test.obj              0 instructions executed    Idle
```

(b) After part B

Figure 2: Testcase after execution

4 Hints and Tips

1. Read the entire lab document CAREFULLY before starting.
2. Attend office hours! We are here to help.
3. Develop an algorithm and have it checked in-person before starting to code. We won't tell you how to code, but we can point you in the right direction.
4. Make good comments in your code to save time and frustration when debugging. Use semicolons to make comments in the LC3 editor.
5. Debug as much as possible! The most frustrating thing is when you code fails all of the test cases because of a single mistyped number. If that happens, we won't give credit back, unless the autograder fails as a result.
6. Since it is impossible to try all 2^{36} testcases that can be created from the database, try to think of "unique" or "special" edge cases. For example, what if the list is empty? What if it's at its max length?
7. Manage your time wisely! It will take 2-5x longer to debug your code than it will take to write your code.

5 Submission

Create a Github repository for your code using the link provided in the Canvas assignment, and push your final code to that repository. For your sort program, name your file [EID]_sort.asm (e.g. jay32_sort.asm). For your search program, name your file [EID]_search.asm (e.g. jay32_search.asm).

6 Further Reading

1. History of Cockrell: http://www.engr.utexas.edu/about/history
2. History of ECE: http://www.ece.utexas.edu/about/history