Lab 04

University Directory

Overview: You will design the user-interface of a university directory that stores the major of every student at the university. Your program will prompt the user for a student's last name and print the major code of that student (only if it exists in the directory). For example, if John Student's major is ECE, if the user types 'student' at the prompt, your program will print 'ECE'.

Your job: Write a program in LC-3 assembly language that will

（1） prompt the user to type a student's last name

（2） search the directory to find the student's major

（3） print the major of the student on the screen

（4） halt the machine

We now discuss each step in turn.

1. Prompting for last name: Your program will print on the screen the string "`Type a last name and press Enter:`" and wait for the user to input a string (followed by an <Enter>). Once the user has pressed the <Enter> key, your program will search the directory for the entered last name.
2. Searching the directory: The directory stores for each student his or her major code. You will find a match for the last name if and only if the last name exists in the directory. Note that it is possible to not find a match in the directory.

   The directory is organized as a data-structure called a linked-list. The working of the linked-list is described below. The address of the first node of the directory is stored in memory location x3300 before the program is run.

3. Printing the major: If your program does not find a match for a last name in the directory, your program must EXACTLY print the string

"No Entry" on the screen. If your program finds a match in the directory, your program will print the major.

4. Halting the machine: The machine halts after printing the major or "No Entry".

The Linked-List

A linked-list is a set of nodes connected to each other via pointers. Each node in the linked-list contains a pointer to (the address of) the next node in the linked-list. This pointer is commonly known as the next-pointer. The last node contains x0000 as its next pointer, indicating that there is no "next node." That is, this is the last node. We call x0000 in this context a NULL pointer.

Each node in a linked-list is comprised of k+1 words: one word containing the next-pointer (the pointer to the next node) and k words of data which are being stored by the node.

The directory of student majors is implemented as a linked-list. Each node consists of three words in the following order:

1. The next-pointer.
2. A pointer to an ASCII string representing the student's last name (in lower case).
3. A pointer to an ASCII string representing the major of the student.

Recall that a string consists of ASCII codes stored in consecutive memory locations, one ASCII code per location. The string is null-terminated, i.e., the end of a string is signified by the NULL character which is ASCII code 0.

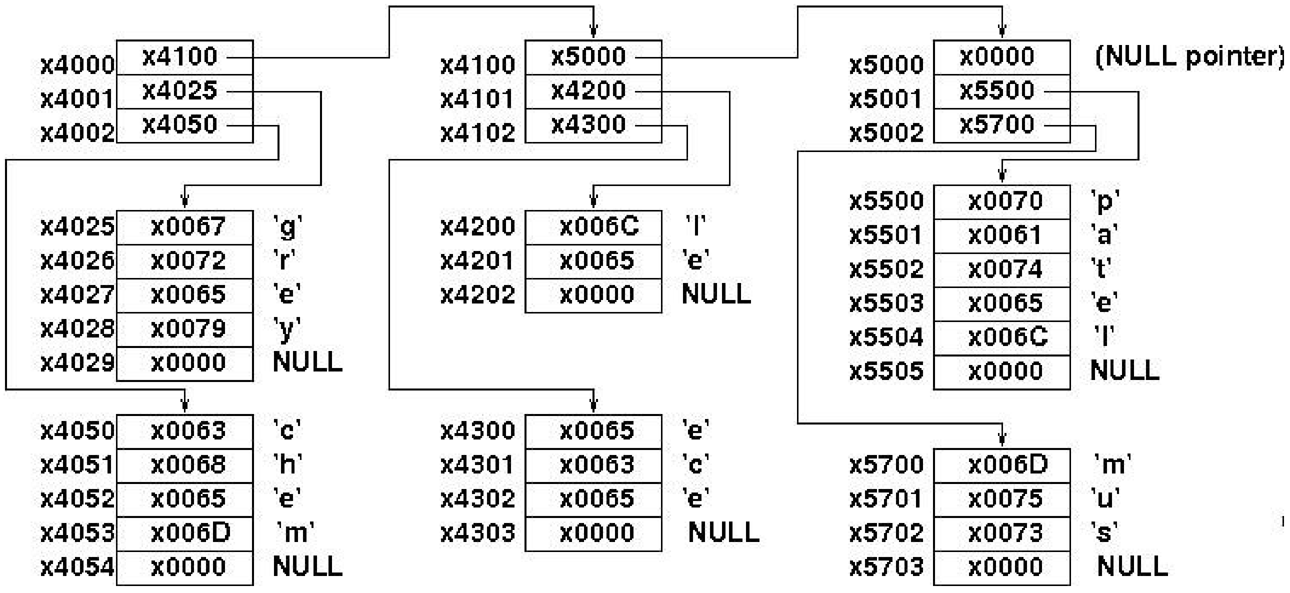
Below we **show an example directory implemented as a linked-list**. This directory contains three nodes: the majors of students grey (chem), le (ece), and patel (mus).

Address	Contents	Description
x3300	x4000	Pointer to the first node
...	...	
x4000	x4100	Next-pointer
x4001	x4025	Pointer to the first ASCII code of the last name
x4002	x4050	Pointer to the first ASCII code of the major

...	...	
x4025	x0067	ASCII code for ″g″
x4026	x0072	ASCII code for ″r″
x4027	x0065	ASCII code for ″e″
x4028	x0079	ASCII code for ″y″
x4029	x0000	NULL character which signifies the end of the last name
...	...	
x4050	x0063	ASCII code for ″c″
x4051	x0068	ASCII code for ″h″
x4052	x0065	ASCII code for ″e″
x4053	x006D	ASCII code for ″m″
x4054	x0000	NULL character which signifies the end of the major
...	...	
x4100	x5000	Next-pointer
x4101	x4200	Pointer to the first ASCII code of the last name
x4102	x4300	Pointer to the first ASCII code of the major
...	...	
x4200	x006C	ASCII code for ″l″
x4201	x0065	ASCII code for ″e″
x4202	x0000	NULL character which signifies the end of the last name
...	...	
x4300	x0065	ASCII code for ″e″
x4301	x0063	ASCII code for ″c″
x4302	x0065	ASCII code for ″e″
x4303	x0000	NULL character which signifies the end of the major
...	...	
x5000	x0000	Next-pointer, x0000 means there is no next node
x5001	x5500	Pointer to the first ASCII code of the last name
x5002	x5700	Pointer to the first ASCII code of the major
...	...	
x5500	x0070	ASCII code for ″p″
x5501	x0061	ASCII code for ″a″
x5502	x0074	ASCII code for ″t″

x5503	x0065	ASCII code for "e"
x5504	x006C	ASCII code for "l"
x5505	x0000	NULL character which signifies the end of the last name
...	...	
x5700	x006D	ASCII code for "m"
x5701	x0075	ASCII code for "u"
x5702	x0073	ASCII code for "s"
x5703	x0000	NULL character which signifies the end of the major
...	...	

Below is a graphical representation of the linked-list.



To search the linked-list, your program will visit each node of the list and compare the last name stored at that node with the last name entered by the user. If a match is found, it will print the student's major and terminate the search. If a match is not found, it will visit the next node and repeat the above process. It will continue to visit nodes until either a match is found or it reaches the last node and no match is found (i.e., a node whose next pointer is 0). Thus, if the program reaches the last node and the name does not match, this implies that a match was not found.

Input/Output Requirements

Described below are detailed requirements about the Inputs and Outputs of your program. You should adhere to these guidelines to receive full credit for this assignment

Input

Your program should prompt the user for the last name from the keyboard, as follows:

- Print a string "`Type a last name and press Enter:`". **Note that you will get a 0 on the assignment if you do not print the string "Type a last name and press Enter:" EXACTLY.**
- The user will input a character string from the keyboard, terminating the last name with the <Enter> key. The <Enter> key is not part of the message, but you will need to print it back out to the console.
- You can assume that only lower-case letters will be entered.
- You may assume that the last name is typed correctly without using backspace and delete.
- ~~The length of the entered string is between 1 and 15 characters.~~ You may assume that the user will type in a last name consisting of between 1 and 15 lower-case alphabetic characters.

Hint: To continually read from the keyboard without first printing a prompt on the screen, use TRAP x20 (assembler name GETC). That is, for each key you wish to read, the LC-3 operating system must execute the TRAP x20 service routine. If you follow TRAP x20 with the instruction TRAP x21 (assembler name OUT), the character the user types will be displayed on the screen.

Output

Your program should output one of two strings depending on the outcome of the directory lookup.

- When the last name entered by the user is found in the linked list, output to the screen the major as recorded in the directory.
- When the last name entered by the user is not found in the linked list, output to the screen: "No Entry".

Hint: To output a string to the console display, use TRAP x22 (assembler name PUTS). What needs to go into R0 in order to use this TRAP instruction?

A sample of what your program will produce, when supplied with the input from users trying to look up a student's major, is shown below:

```
Type a last name and press Enter:grey
chem
----- Halting the processor -----
Type a last name and press Enter:guvenilir
No Entry
----- Halting the processor -----
Type a last name and press Enter:le
ece
----- Halting the processor -----
```

Notes

1. Your program must start at location `x3000`.
2. The linked-list representing the directory is an input to your program. The directory is loaded in memory before your program begins to run. Your program will only search the directory, not modify it. You will lose points if you modify the directory.
3. The pointer to the first node is stored in memory location x3300 before the program execution begins. Further, assume that all the nodes are stored between memory locations x4000 and xEFFF. Make no other assumptions about the location of the nodes. **Note that the pointer to the first node may be set to NULL (i.e., 0), indicating that there are no nodes in the list.**
4. Your program should NOT make any assumptions about the number of nodes in the list.
5. You can assume that the linked list will always contain a last node.
6. The names are stored in the directory using all lower-case letters.
7. You may assume that everyone in the directory does have a legitimate last name. For example, the content of x4200 in this example is the ASCII code for 'g', the first letter of a student's last name. The contents of x4200 may not be x0000, the null character, signifying no last name.
8. You may assume that everyone in the directory has a major.
9. The <Enter> key is the line feed character which is the ASCII code `x0A`.
10. You can assume that the directory never contains two nodes with the same last name.

Attention:

1. Your zip file should contain at least two files:
   .asm file and **report in pdf format**.
   As subject name , please refer to the notice on course web page.