EE306 Introduction to Computing

Lab 5 (accepted between 11/2 to 11/4, 9pm, on Canvas)
Course Instructor: Dr. Nina Telang

- **All Lab assignments must be completed individually.**
- **You are not permitted to seek help or clarification from anyone (this includes your peers, friends, upper class students, former students, former TAs etc.) other than the instructor or the TAs.**
- **WE WILL BE CHECKING YOUR CODE FOR PLAGIARISM/ACADEMIC DISHONESTY.**

Your file should be named exactly after your EID, for example, xy1234.asm. Your program will not be graded if you fail to follow the naming convention of the file name.

Purpose: The purpose of this assignment is to write a program in the **LC-3 assembly language** that creates a user interface to determine if the course number entered by the use of the program matches a list of course numbers. The list of course numbers is organized as a "**LINKED LIST**" data structure.

Your program must:

1. Prompt the user for the Course Number **(refer to course number description in point 5)** by printing on the monitor the string "**Type Course Number and press Enter:** " and wait for the user to input a string followed by <Enter> (ASCII: x0A). **(Assume that there is no case where the user input exceeds the maximum number of characters).**
2. Your program must then search the list of courses to find a match for the entered course number. The list stores the course number for each course being offered. You will find a match only if the course is in the list. It is possible to not find a match in the list.
3. If your program finds a match, then it must print out "**<Course Number> is offered this semester!" (eg., "EE306 is offered this semester!")**
4. If your program does not find a match, then it must print out **"<Course Number> is not offered this semester.". (eg., "EE307E is not offered this semester.")**.
5. **Unique course numbers** are **a maximum of 7 characters long** and contains only uppercase alphabets and numbers**.** For example, EE306. Refer to UT's official course numbering system for more information. The important information related to the specifications of your program is also summarized below.
    a. A course can have a one, two, or three letter abbreviations (e.g., M, EE, PHY, etc)
    b. Each course is identified by a number made up of three digits, which **may** be followed by a letter (e.g., EE307E).
6. Head pointer for the course list is at x4000. In other words, the address of the first node of the list is at **x4000**.

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. If the last node contains x0000 as its next pointer, it implies 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. In our case, the database of courses is implemented as a linked-list with *k+1=2*. Each node consists of two words in the following order:

1. The next-pointer.
2. A pointer to an ASCII string representing the course number (maximum of 7 Characters long+ NULL).

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 is an example database implemented as a linked-list. When you test your program, you can use a database with a similar structure. The test cases that we will use to test your program will be similar to this example.

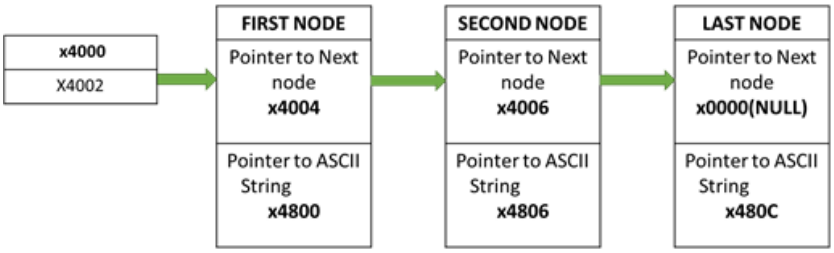

Figure 1: Example Linked List Structure – Course Offering List

Address	M[Address]
X4000	X4002
X4001	…
X4002	X4004 **(Next pointer)**
X4003	X4800 **(Pointer to Course Number)**
X4004	X4006
X4005	X4806
X4006	X0000
X4007	X480C
…	…
X4800	"E"

X4801	"E"
X4802	"3"
X4803	"0"
X4804	"6"
X4805	X0000
X4806	"M"
X4807	"4"
X4808	"0"
X4809	"8"
X480A	"C"
X480B	X0000
X480C	"P"
X480D	"H"
X480E	"Y"
X480F	"1"
X4810	"0"
X4811	"5"
X4812	"M"
X4813	X0000
…	…

Figure 2: Contents of Memory

If the input were:

Type Course Number and press Enter: EE307E

Then the output message would be:

EE307E is not offered this semester.

If the input were:

Type Course Number and press Enter: M308C

Then the output message would be:

M308C is offered this semester!

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 EXACTLY "**Type Course Number and press Enter:** ". Then wait for the user to input a string followed by <Enter>. Note that you will get a 0 on the assignment if you do not

print this string EXACTLY. The user will input a character string from the keyboard, terminating the Course Number with the <Enter> key.

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 linked list lookup.

(i) If the course number entered by the user is found in the course listing, print out "**<Course Number> is offered this semester!**".

(ii) If the course number entered by the user is not found in the course listing, print out "**<Course Number> is not offered this semester.**".

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

A sample of what your program will produce, when supplied with the input from the user trying to add the course:

EXAMPLE 1:

Type Course Number and press Enter: EE307E
EE307E is not offered this semester.

----- Halting the processor -----

EXAMPLE 2:

Type Course Number and press Enter: EE306
EE306 is offered this semester!

----- Halting the processor ----

EXAMPLE 3:

Type Course Number and press Enter: PHY105M
PHY105M is offered this semester!

----- Halting the processor -----

Notes & Hints:

1. Your program must start at location x3000.

2. The linked lists is an input to your program. The list is loaded in memory before your program begins to run. Your program will search the list.

3. The pointer to the first node to the list of courses is stored in memory location x4000 before the program execution begins. Further, assume that all the nodes are stored between memory locations x4002 and xFDFF. 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 may assume that every course in the list have a unique Course Number.

IMPORTANT: The file that you will upload to Canvas for this assignment must be named **youreid.asm. (eg., xy123.asm)**