Lab #7: Simple Data-Entry Program

Overview:

In this lab you will complete the development of a program that implements a simple data-entry of personal information.

Requirements

The program should support the following features:

- allow user to enter personal information of 4 people
- store entered information in memory
- after entering the information of all people, the program should display the full database
- personal information should include full name (max 30 chars), address (max 60 chars), city (max 16 chars), state (max 2 chars) and zip code (max 5 chars)

The following example shows how the screen should look after a program run (user input shown in bold):

Full-Name: John Doe I
Address: 5555 NW Anywhere St.
City: Portland
State: OR
Zip: 97000

------------------------------------
Full-Name: Jane Doe I
Address: 6666 SE Lost St.
City: Hillsboro
State: OR
Zip: 97124

------------------------------------
Full-Name: John Doe II
Address: 7777 NW Anywhere St.
City: Portland
State: OR
Zip: 97000

------------------------------------
Full-Name: Jane Doe II
Address: 8888 SE Lost St.
City: Hillsboro
State: OR
Zip: 97124

------------------------------------
Directions

1. Create an assembly language code file named \texttt{c:\lab7.asm} using Notepad++.

2. Type in (or copy & paste) the following code skeleton:

```assembly
.model small ;specify small memory model

;ASCI codes
CR equ 0dh
LF equ 0ah

;TODO: Step #4 here

;TODO: Step #5 here

.stack 200h ;specify a stack size of 512 bytes

.data
separator_msg db CR,LF,"-------------------------------------",CR,LF,'

crlf_msg db CR,LF,'

fullname_msg db 'Full-Name: ' '

address_msg db 'Address: '

city_msg db 'City: '

state_msg db 'State: '

zip_msg db 'Zip Code: '

;TODO: Step #6 here
```
X86 Assembly Language Programming for the PC (EET-241 / ENGR-275)       Walter Lara

.code

.start:
    mov ax,@data        ;set-up ds to be able to access our data
    mov ds,ax

    ;set-up index & loop counter
    mov si,0
    mov cx,length database

.read:
    ;Use Interrupt 21h, Service 09h to
    ;print full-name prompt message
    mov ah,09h
    lea dx,fullname_msg
    int 21h

    ;Use Interrupt 21h, Service 0ah to
    ;read full-name from user
    mov ah,0ah
    lea dx,database[si].fullname
    int 21h

    ;Use Interrupt 21h, Service 09h to
    ;print CR & LF (goto next line)
    mov ah,09h
    lea dx,crlf_msg
    int 21h

    ;TODO: Step #7 here

    ;Use Interrupt 21h, Service 09h to
    ;print separator line
    mov ah,09h
    lea dx,separator_msg
    int 21h

    ;increment index & loop
    add si,type database
    loop read

    ;TODO: Step #8 here

.exit:
    ;Use DOS interrupt 21h, Service 4ch
    ;to exit program
    mov ax,4c00h
    int 21h

.end start        ;tell assembler to finish

3. Take a close look at the above code and pay attention to the comments. Make sure that you understand what the given pieces of code do before proceeding any further.
4. You will use DOS Interrupt 21h, Service 0ah to read user input. This service allows reading a full string from the keyboard. Using this service requires an input buffer formatted as follows:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Size (bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Input: maximum number of characters to read</td>
</tr>
</tbody>
</table>
| 1      | 1            | Input: Set to 0 for our case  
Output: Actual number of characters read (including carriage-return) |
| 2      | n            | Output: characters read |

A convenient way to use this service is to create a structure to encapsulate the buffer:

```
String struc
    max_len     db ?
    act_len     db 0
    chars       db 80 dup('$')
ends
```

Initializing the chars to ‘$’ is not needed by this service but it is done here for convenience, to later facilitate displaying the string using Interrupt 21, Service 09.

Copy the above structure to the `lab7.asm` file, where indicated by the code comments.

5. Right after the String structure declaration create a new structure named Info to encapsulate the personal information described on the Requirements section. All fields must be of type String. For example, inside the structure the full-name variable must be declared as:

```
fullname String<31>
```

Notice that we initialize the `max_len` variable of `fullname` to 31 instead of 30. The reason is that an extra character space needs to be allocated for the carriage-return (as explained on Step #4).
6. At the `.data` section, create an array variable named `database` of type `Info` and size 4 (use the `dup` directive). This variable will be used to store the user input.

7. Take a close look at how we use interrupt 21h to print the prompt message, read the user input and print the separator line for `fullname`. Implement a similar sequence for `address`, `city`, `state` and `zipcode`.

8. Create a loop with a label named `print`. The loop should use a counter and index similar to the `read` loop on the skeleton.

9. Inside the `print` loop, print the `fullname`, `address`, `city`, `state` and `zipcode`. For example, for `fullname`:

   ```
   mov ah, 09h
   lea dx, database[si].fullname.chars
   int 21h
   lea dx, crlf_msg
   int 21h
   ```

   Notice that we pass the effective address of the `fullname.chars` field as oppose to the effective address of `fullname`.

10. Assemble the code using TASM. At the DOS prompt type:
    ```
    tasm /l /zi lab7.asm
    ```

    The “/l” switch tells the assembler to generate an output listing file (named `lab7.lst`) while the “/zi” tells it to generate symbolic information needed for the linker.

    Note: MASM32 equivalent command is:
    ```
    ml /c /Fl lab7.asm
    ```

11. If you get any error on your code, take a look at `lab7.lst` (you can use notepad). It shows the generated machine code side-by-side with your assembly language. Always pay attention to the first error as subsequent errors may be caused by the first one. Correct your error by editing `lab7.asm` and then go back to step #10.

12. Link the program by using TLINK. At the DOS prompt type:
    ```
    tlink /v lab7.obj
    ```
The “/v” switch tells the linker to generate full symbolic information needed for the debugger.

Note: MASM32 equivalent command is:
link16 lab7.obj,,

13. Run the program. At the DOS prompt type:
lab7

14. Verify that it operates correctly (as described on the Requirements section).

15. If the program doesn’t operate correctly, debug it using the Turbo Debugger tool. At the DOS prompt type:
td lab7.exe

Note: For MASM32 use DEBUG. The invocation command is:
debug lab7.exe

16. If necessary, use the Turbo Debugger help (from the menu) for reference on how to use it (or for DEBUG, use the ? command).

17. If necessary, make corrections to your program by editing lab7.asm and going back to step #10.

Lab Report

To complete this lab you should provide the instructor with the following files:

lab7.asm
lab7.lst
lab7.exe

The files should be archived on a zip file and send via e-mail. The zip file should be named as follows: lab7_<your name>.zip (e.g., lab7_WalterLara.zip).