****

**JOMO KENYATTA UNIVERSITY**

**OF AGRICULTURE AND TECHNOLOGY**

**UNIT: MICROPROCESSORS I**

**UNIT CODE: EEE2406**

**TITLE: LCD DISPLAY STRING**

**LAB I**

**MEMBERS**

**Davies Momanyi ENE211-0233/2016**

**Allan Mwirigi ENE211-0232/2016**

**Vitalis Ayacko ENE211-0025/2016**

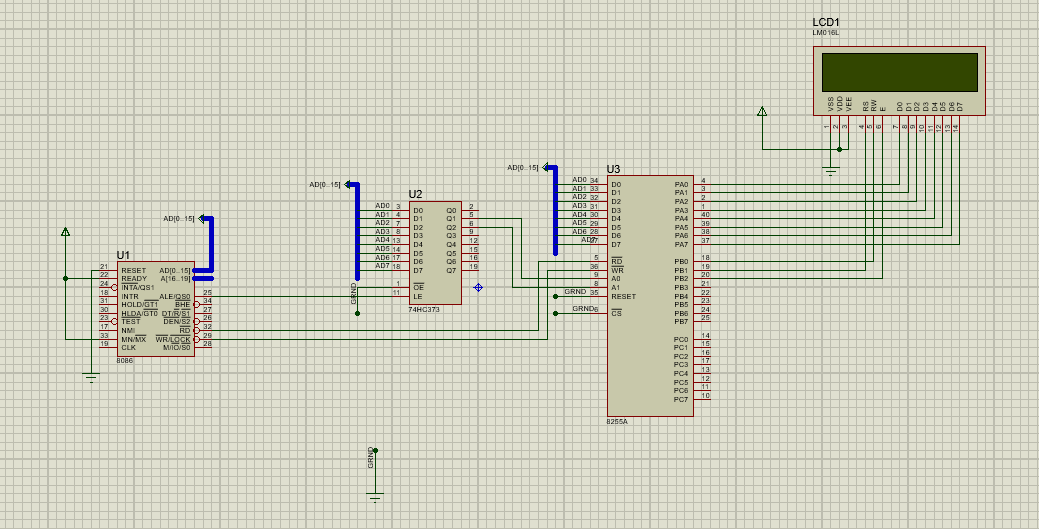
**Thoenes Mbui ENE211-0016/2016**

**Allan Kipkosgei ENE211-0249/2016**

**INSTRUCTIONS**

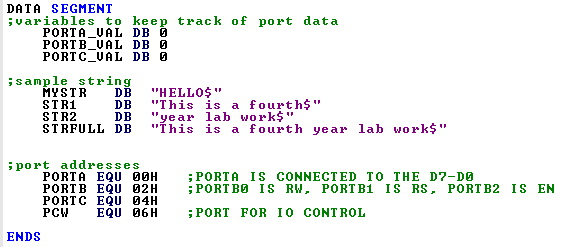
Develop an Intel 8086 assembly language program that displays the string “This is a fourth year lab work” to an LCD via an Intel 8255 interface”. The string is shifted right at two (2) seconds interval and to reappear from the left of the screen. Simulate the system using Proteus software.

**CIRCUIT DIAGRAM**

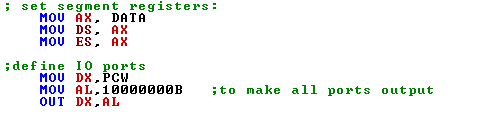
****

In the above circuit, the 8086 microprocessor is connected to the 8255 PPI through a latch. The latch is required in order to reconcile the difference in operation speed between the microprocessor and the PPI. The 8086 is much faster than the 8255 therefore when it writes to the data bus, the data is only retained for very short period of time before it continues with other processes. The data needs to be held by the latch which contains flip flops. The data can then be transferred to the LCD through the PPI at its normal operation speed.

**IMPLEMENTATION**

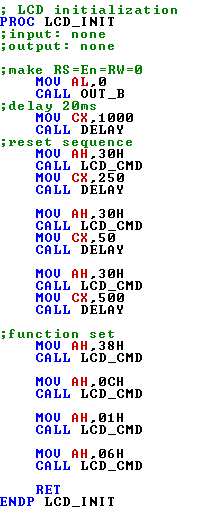
****

Variables are defined in the data segment. The addresses of PortA, PortB, PortC and the ControlPort are defined as well as PORT\_VAL variables to temporarily hold data to be read or written to the ports. Variables are also defined for the strings to be displayed.



The data segment is initialized then the control word **10000000B** is sent to the ControlPort (**PCW)** to set all the ports for output.

The **LCD\_INIT** procedure is then called, which initializes the LCD before it is used. This is a necessary step specified by the device manufacturer whereby a series of commands are sent to the LCD. This procedure is defined as follows.



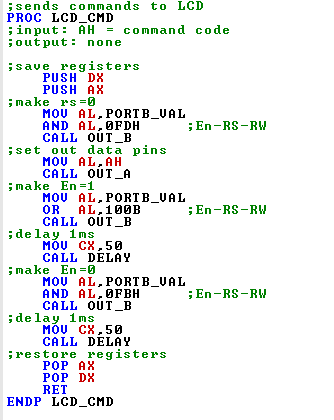
The initialization consists of the following steps:

* Send command 0x30
* Delay 20ms
* Send command 0x30
* Delay 20ms
* Send command 0x30
* Delay 20ms
* Send Function set
* Display Clear command
* Set entry mode command

Each LCD command has a corresponding hex code that can be sent to the LCD. For instance:

|  |  |
| --- | --- |
| **Hex Code** | **Command to LCD instruction Register** |
| 01 | Clear display screen |
| 02 | Return home |
| 04 | Decrement cursor (shift cursor to left) |
| 06 | Increment cursor (shift cursor to right) |
| 05 | Shift display right |
| 07 | Shift display left |

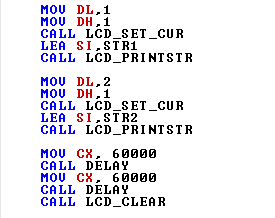
These commands are used to control the operation of the LCD. To send these commands to the LCD, the **LCD\_CMD** procedure is called. Its implementation is as follows.



The command to be output is placed in the AL register and a series of required steps carried out for successful sending of the command. These steps are based on manufacturer specifications. The **OUT\_A** and **OUT\_B** procedures are called in order to send the command using the **OUT** instruction on the appropriate port.

**String Display**

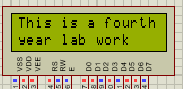
This implementation of the task at hand involves first displaying the entire string and then scrolling on the screen.



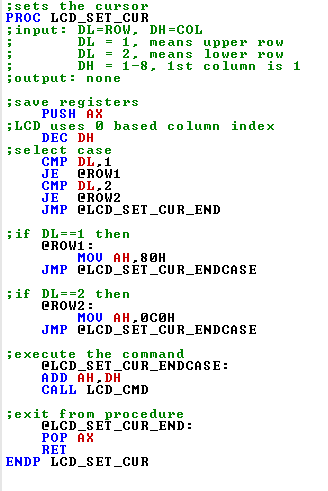
Since a 16 by 2 LCD is in use, the entire text “This is a fourth year lab work” does not fit on the screen as it is greater than 16 characters. It is therefore broken into two parts, STR1 consisting of the first 16 characters and STR2 with the rest of the characters. STR1 is displayed on the first row and STR2 on the second row. To display the strings on the appropriate rows, the **LCD\_SET\_CUR** procedure is called. It uses values of register DL for the row and DH for the column. A value of 1 for DL corresponds to the first row while 2 corresponds to the second row.

**LCD\_SET\_CUR** procedure makes use of two commands to set the cursor position. **080H** moves the cursor to the beginning of the first line while **C0H** moves the cursor to the start of the second line.

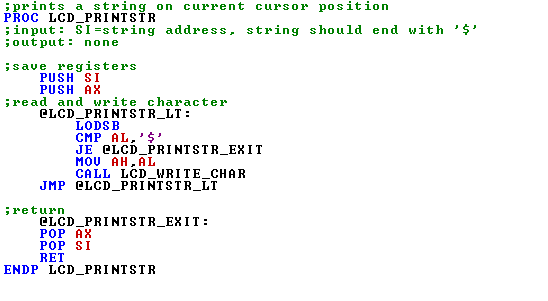
Below is what is displayed on the string.



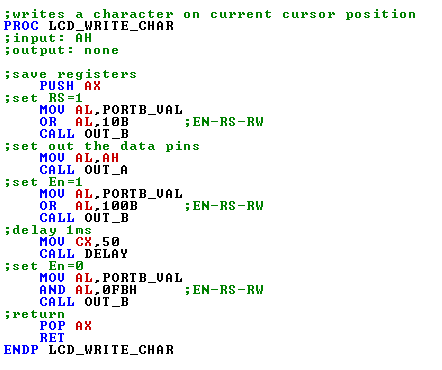
After both portions of the string are displayed, there is a delay then the screen is cleared in order to go to the next step of scrolling the displayed string.



To display the string once the cursor has been set, the effective address of the string is first loaded into register SI and then the **LCD\_PRINTSTR** procedure is called.

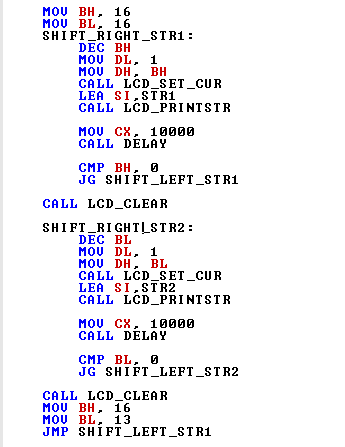


This procedure uses the LODSB instruction to load a byte from SI to the register AL. SI is the source operand and points to a sequence of bytes corresponding to the string. The byte position to be loaded is automatically incremented on each iteration of the LODSB instruction such that each character is placed into AL and displayed. AL is compared to ‘$’ on each iteration to check if the string has been fully covered then the loop ends. To display each character the **LCD\_WRITE\_CHAR** procedure is called with the character placed in AH register.



**STRING SHIFTING**

The following code is used to shift the text right at two second intervals.



This code uses a loop to shift the cursor right when displaying the string such that the string appears to be scrolling right. Since there are two portions of the complete string, two loops are used to shift each portion i.e. STR1 and STR2. These loops run 16 times based on the values of BL and BH respectively. On each iteration of the first loop, BH is decremented and its value used to set the cursor position on the row by assigning it to DH and calling the **LCD\_PRINTSTR** procedure. The string is then displayed by loading its effective address to SI and calling the **LCD\_PRINTSTR** procedure. For instance, if BH is 15 and **LCD\_PRINTSTR** is called, only the first character of the string is printed. As BH is decremented, more characters become visible hence a shifting action is observed until the entire screen is covered. A conditional jump is used to proceed with the loop as long the value of BH is greater than 0. There is a delay between each iteration of the loop to give a two second interval before the next character is shifted.

When the first loop ends, the screen is cleared then the second loop begins to shift the second portion of the string. The process above occurs for the second loop as well. When it is completed, the screen is cleared, the values of BH and BL reset, then the code jumps back to the first loop. Hence a continuous shifting of the string.