Lab 7: Subroutines with Result Shown on LCD Display

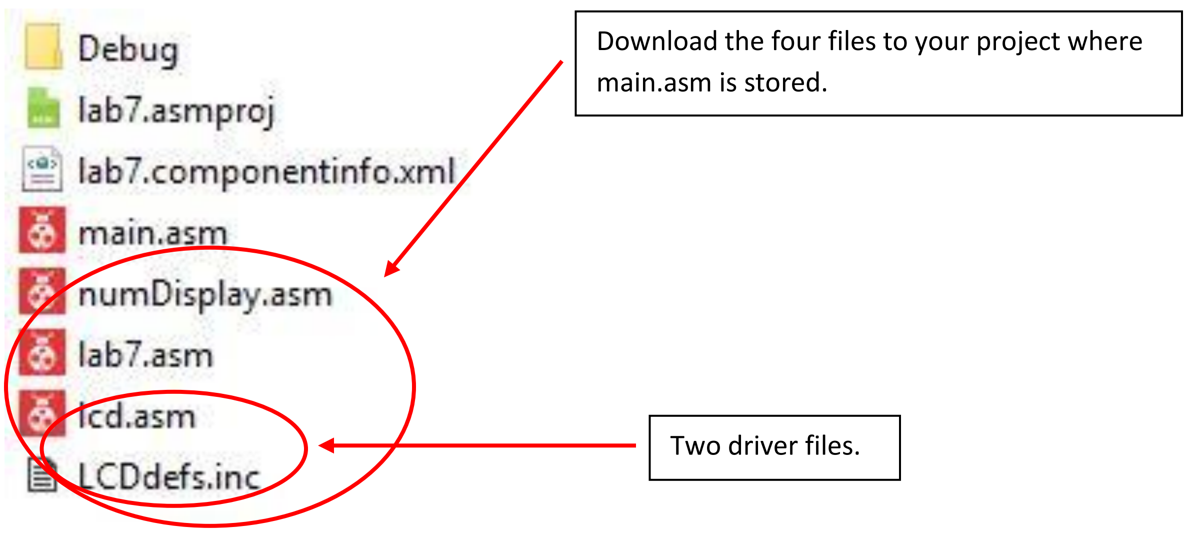
I. LCD Display

In lab6, we implemented the *unsigned int upperCaseCount (String str)* subroutine. How about displaying the result on an LCD display? LCD stands for Liquid Crystal Display. The one that we use has built-in controller and driver chips. What we concern in this lab is the character LCD module. Character modules can display only text and perhaps some special symbols. This LCD display shows 16x2 characters (2 rows and 16 characters per row).

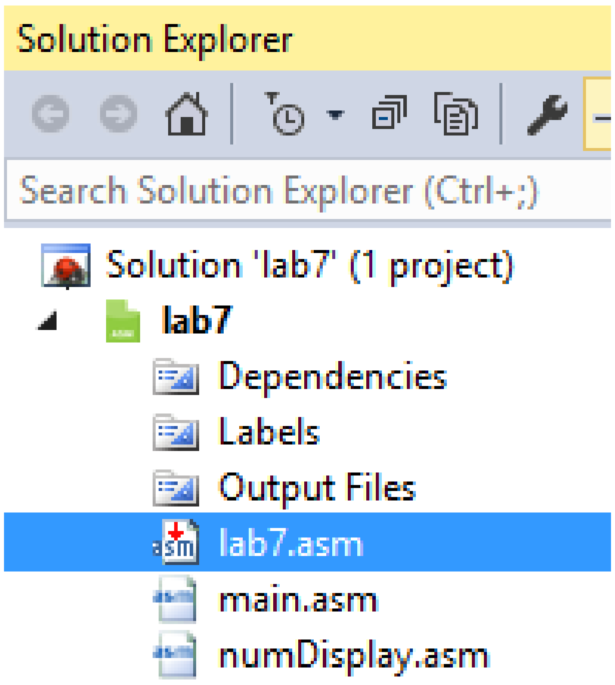
Create a new project named lab7, and then download the two driver files (HD44780 LCD Driver files for ATmega2560) to the same directory containing main.asm file:

LCDdefs.inc (LCD driver)
lcd.asm (LCD driver)

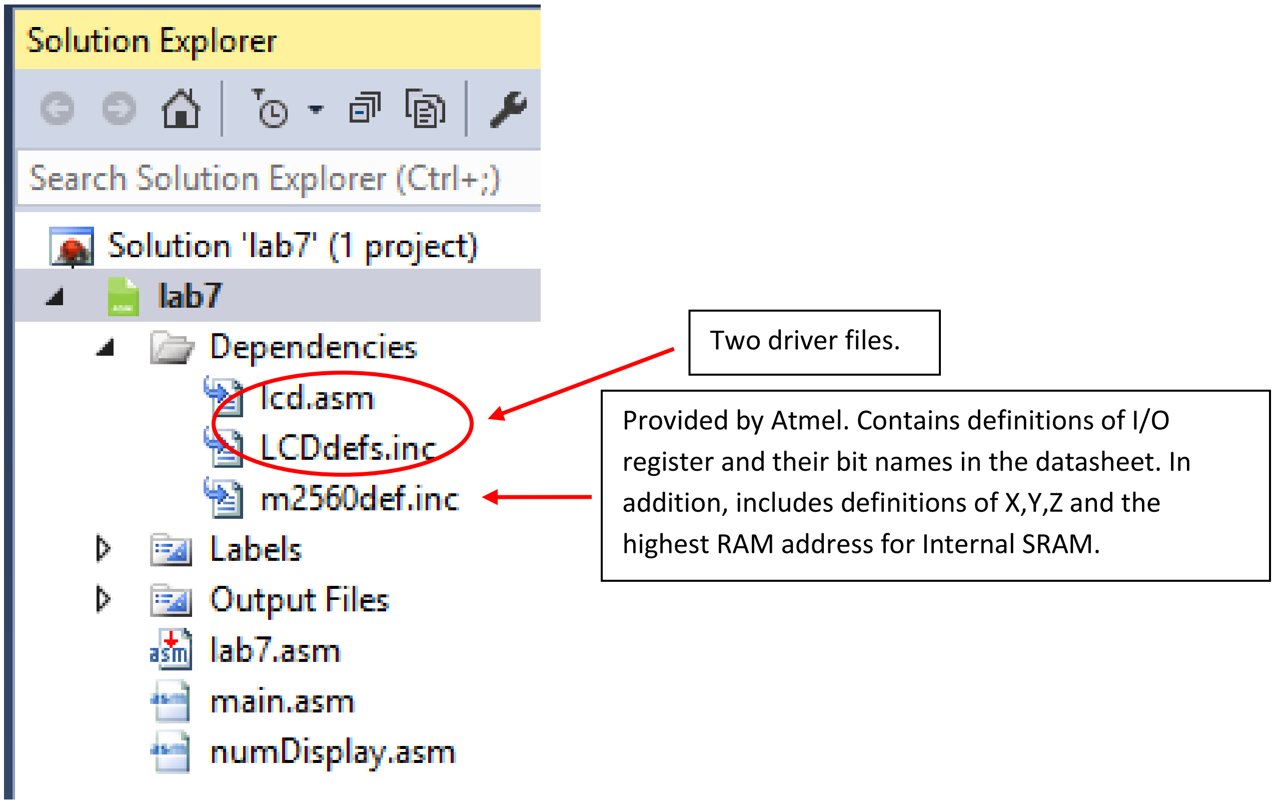
Download *lab7.asm* and *numDisplay.asm* to the same directory above. The directory of your project looks like this:



In *Solution Explorer* of the *Atmel Studio 7.0* project, add *lab7.asm* and *numDisplay.asm* to your project and set *lab7.asm* as entry file. The project Solution Explorer looks like this:



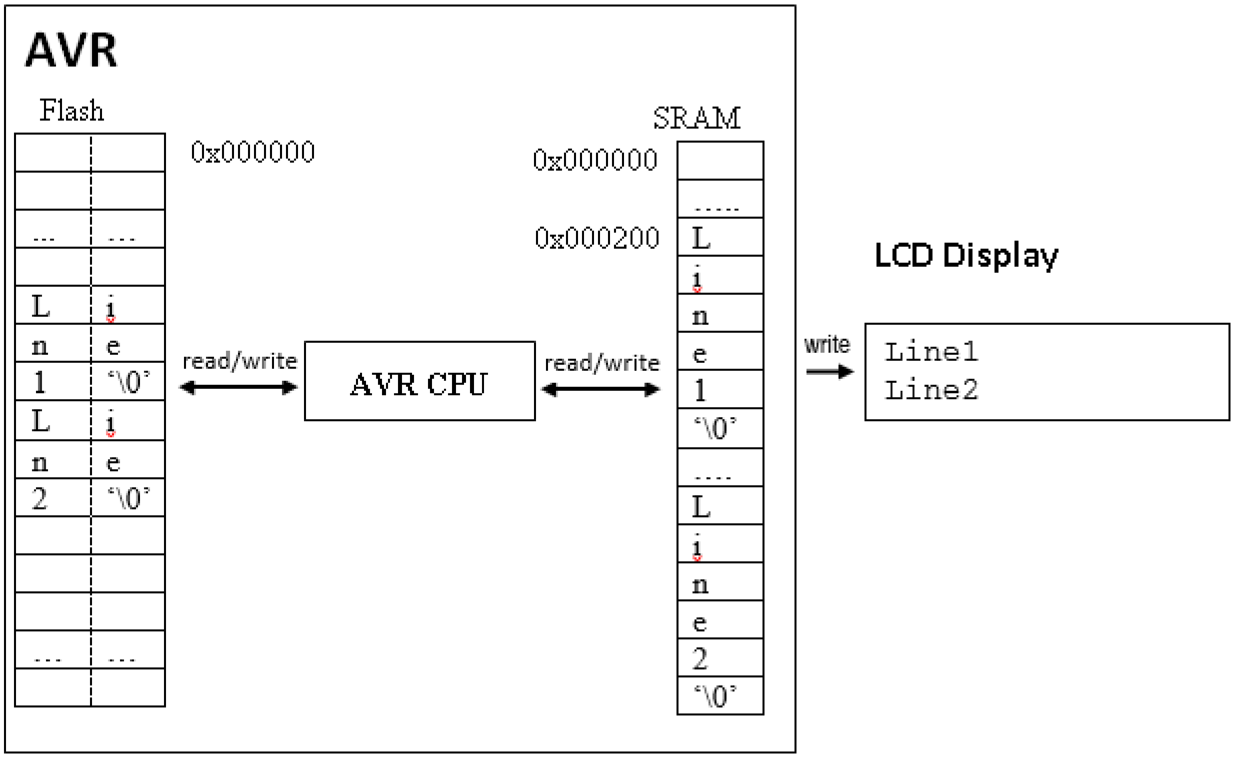
Build the project. In *Solution Explorer* of the *Atmel Studio 7.0* project, expand the *Dependencies* folder. The *Solution Explorer* looks like this:



Two driver files.

Provided by Atmel. Contains definitions of I/O register and their bit names in the datasheet. In addition, includes definitions of X,Y,Z and the highest RAM address for Internal SRAM.

The following diagram demonstrates the relationships between the program (Flash) memory, data (SRAM) memory, the AVR CPU of the AVR board and the LCD display. In lab7.asm, the two strings “Line1” and “Line2” are copied from the program memory to the data memory. Then the two strings are displayed from the data memory to the LCD display. Notice that the program memory is word (2 bytes per word) addressable and the data memory is byte addressable.



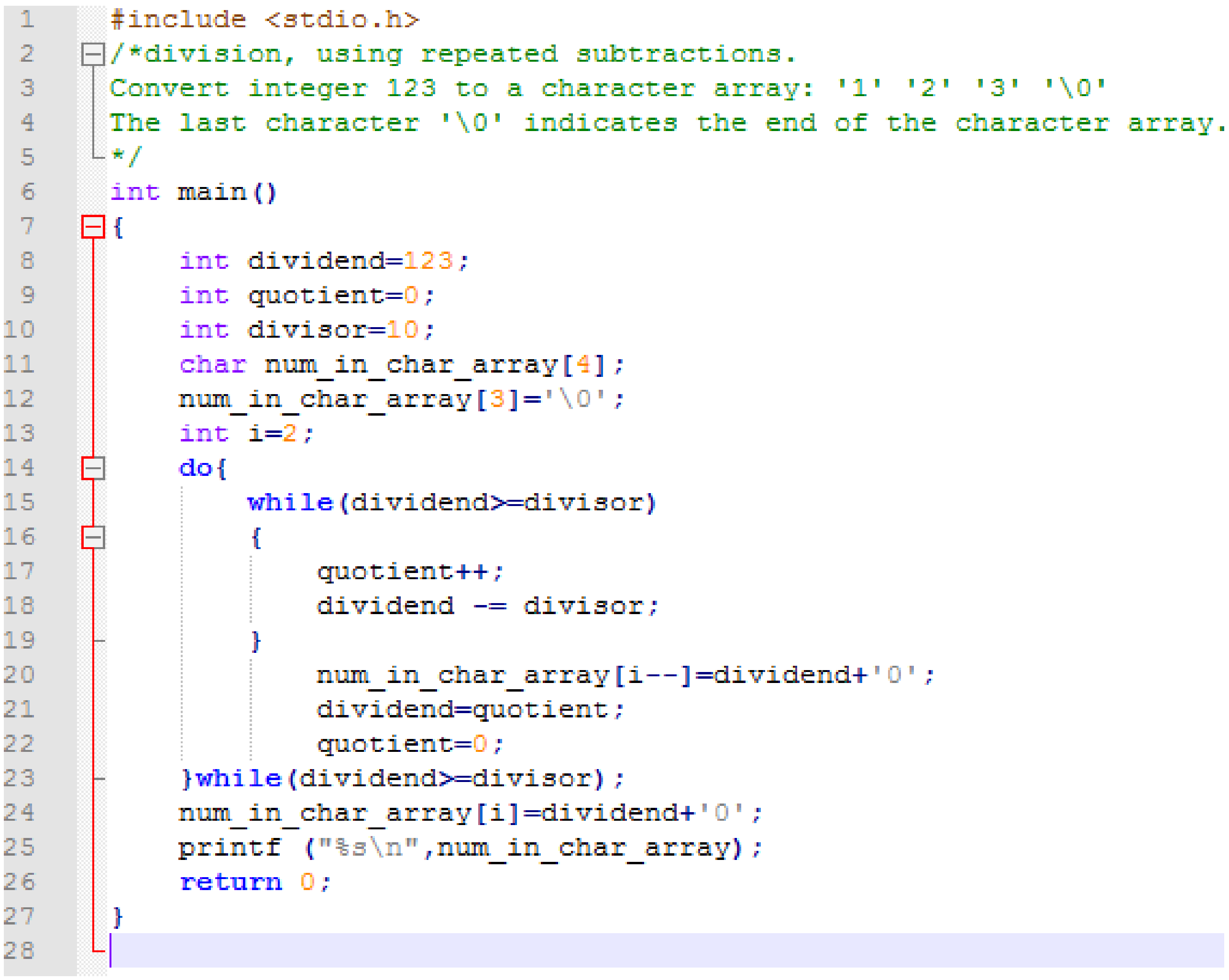
Note that *str_init* is defined in *lcd.asm*. It copies a string from the program memory (flash) to the data memory (SRAM), like what we did in lab 6. “SP_OFFSET” is defined in LCDdefs.inc (line 42 and 43). It is the number of bytes of the return address for a subroutine call. It is 3 bytes in this case.

Open lab7.asm. Understand the code.
In the *display_strings* subroutine, the algorithm is:

```
    Clear the LCD display (lcd_clr)
    Move the cursor to the desired location (row 0, column 0) (2X16 display) (lcd_gotoxy)
    Display “Line1” stored in SDRAM (lcd_puts)
    Move the cursor to the desired location (row 1, column 0) (2X16 display) (lcd_gotoxy)
    Display “Line2” stored in SDRAM (lcd_puts)
```

II. Exercises: Set *numDisplay.asm* as entry file. Finish implementing *void display_num()* subroutine. If time permits, modify the subroutine such that it accepts a parameter – the number to be displayed, e.g. *void display_num (unsigned int num)* – pass the number to be displayed as a parameter.

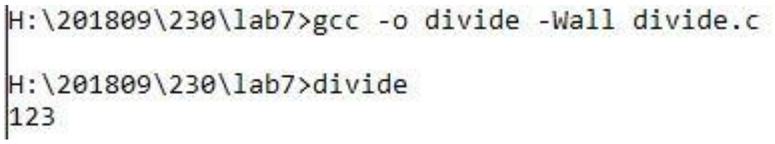
The C equivalent code is:



```c
#include <stdio.h>
/*division, using repeated subtractions.
Convert integer 123 to a character array: '1' '2' '3' '\0'
The last character '\0' indicates the end of the character array.
*/
int main()
{
    int dividend=123;
    int quotient=0;
    int divisor=10;
    char num_in_char_array[4];
    num_in_char_array[3]='\0';
    int i=2;
    do{
        while(dividend>=divisor)
        {
            quotient++;
            dividend -= divisor;
        }
            num_in_char_array[i--]=dividend+'0';
            dividend=quotient;
            quotient=0;
    }while(dividend>=divisor);
    num_in_char_array[i]=dividend+'0';
    printf ("%s\n",num_in_char_array);
    return 0;
}

```

You may download *divide.c*, compile and run the code from the DOS command window:



```
H:\201809\230\lab7>gcc -o divide -Wall divide.c

H:\201809\230\lab7>divide
123
```

To accomplish a similar task in assembly language, we need to convert each digit to a character, e.g. convert integer 1 to character ‘1’ and store integer 123 as a string ‘1’ ‘2’ ‘3’ in SRAM. It is required to manipulate memory addresses, and pass parameters using stack. It is important to draw the stack frame properly.

Written by Victoria Li on October 26, 2018.