Lab 7: Subroutines with Result Shown on LCD Display

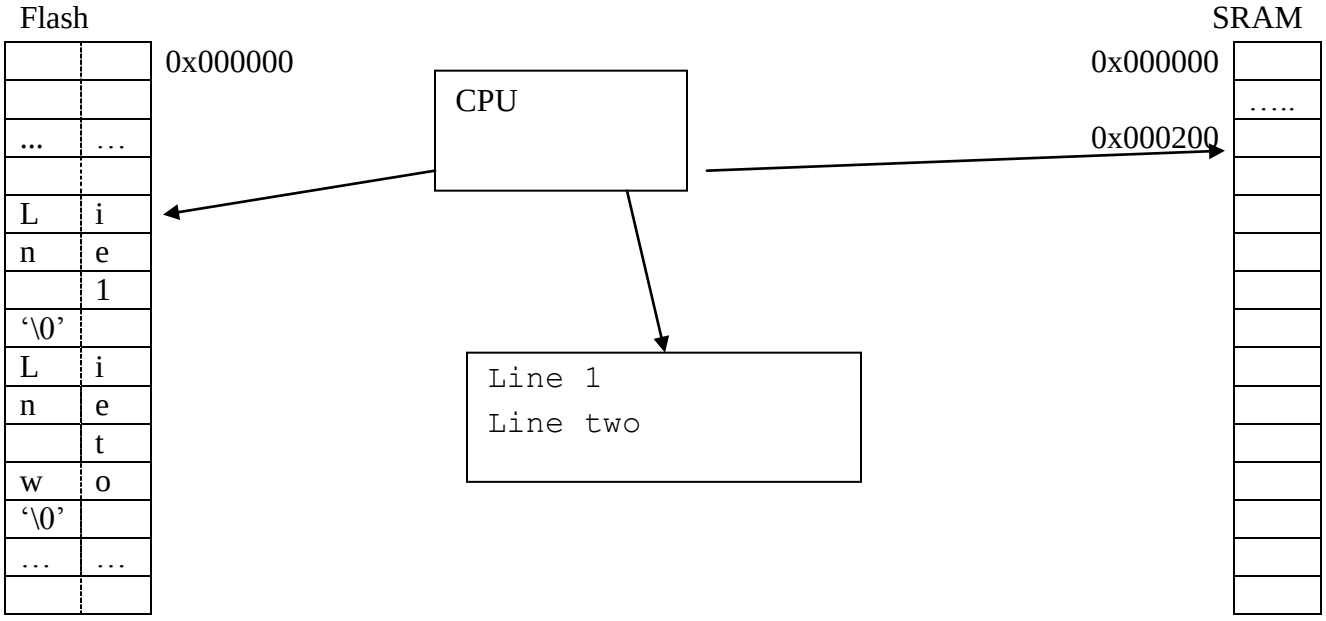
Submit numDisplay.asm at the end of your lab.

I. Subroutine continues

We wrote a subroutine (strlength) in the previous lab. To be more accurate, we should modify the diagram in the previous lab (see the red characters) as the following:

Address	content	details	notes
0x0000 ~ 0x01FF		General Purpose Registers and I/O Registers	
0x0200			.DSEG
	……..		
		<- Z and SP	
	r1	saved register	Programmer pushes those registers onto the stack in the subroutine
	r0	saved register	
	r31	saved register	
	r30	saved register	
	ret	return address	Assembler pushes the return address onto the stack automatically when "call do_something" is executed.
	ret	return address	
	ret	return address	
	0xCC	parameter (Z + 8)	Programmer pushes the values onto the stack in the **location where a subroutine is called, it could be in the main, or another subroutine.**
0x21FF	0xEE	parameter (Z + 9)	

Download LCDdefs.inc, lcd.asm, lab7.asm. HD44780 LCD Driver for ATmega2560 is written in LDCdefs.inc and lcd.asm. "lab7.asm"(by Mr. Jason Corless): how to display strings on LCD.



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 3 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 "Line 1" stored in SDRAM (lcd_puts)
Move the cursor to the desired location (row 1, column 0) (2X16 display) (lcd_gotoxy)
Display "Line two" stored in SDRAM (lcd_puts)

II. Exercises: download numDisplay.asm, Finish implementing display_num subroutine.
The C equivalent code is:

```c
/*division, using repeated subtractions*/
int main()
{
    int dividend=123;
    int quotient=0;
    int divisor=10;
    char num[4];
    num[3]='\0';
    int i=2;
    do{
        while(dividend>=divisor)
        {
            quotient++;
            dividend -= divisor;
        }
            num[i--]=dividend+'0';
            dividend=quotient;
            quotient=0;
    }while(dividend>=divisor);
    num[i]=dividend+'0';
    printf ("%s\n",num);
    return 0;
}
```

To accomplish a similar task in assembly language, we need to convert each digit to a character, e.g. int 1 to character '1' and store integer 123 as a string '1' '2' '3' in SRAM. It is required to manipulate memory addresses, and be responsible for parameter passing using stack frame.

III. Lab 7 Part II - Quiz
Do Lab 7 – Part 1: "Tests & Quizzes" -> "Lab 7 – Part 1".
Submit numDisplay.asm at the end of your lab.