EE308 HOMEWORK ASSIGNMENT #2

AYBARS YAZICI 25330

0.1. **Question Q1.** We are asked to write a function that displays the content of W0 register in binary on line 1 of the LCD

```
displayBinary:
      mov         #'0', W3
      mov         #LCD_line1, W10
      add         W10, #15, W10 ;As we will start from the LSB and go to MSB
      mov         #0, W11
      loop:
      mov         #0x0001, W2 ;We will and the LSB of W0, with 1
      and         W0, W2, W1  ;Store it in W1
      add         W1, W3, W1  ;Add the ASCII code of 0 to be able to display it
      mov.b       W1, [W10--] ;Move it to the according spot on the LCD
      inc         W11,W11 ;Increment W11 to mark one iteration as done.
      lsr         W0, W0  ;Shift W0 1 bit to the right.
      cp          W11, #16  ;Have we iterated through the loop 16 times?
      bra         NZ,loop ;If we haven't go the start of the loop once again.
      return  ;If we have, then return from the function.
```

Note that if we wanted to preserve the information of W0, and not lose it after shifting to right, we can just copy the information to another register and shift that register without touching the information stored in W0.

0.2. **Question Q2.** Let's take a quick look at the first lines of ee308.s file:

```
.bss
    LCD_line1:     .space 16
    LCD_line2:     .space 16
    LCD_ptr:       .space 2
    LCD_cmd:       .space 2
    LCD_offset:    .space 2
```

From here, and plus from the knowledge that our RAM starts from 0x0800,

(1) LCD_Line1 will be allocated into 0x800 until 0x080F(inclusive)
(2) LCD_Line2 will be allocated into 0x0810 and it will occupy another 16 byes, thus it will end at 0x081F
(3) LCD_ptr at 0x0820 and 0x0821
(4) LCD_cmd at 0x0822 and 0x0823
(5) LCD_offset at 0x0824 and 0x0825

Thus the code below

```
    mov         #__SP_init, W15      ; Initalize the Stack Pointer
```

will actually result in

```
    mov         #0x0826, W15   ; Initalize the Stack Pointer
```

The remaining part of the RAM after the variable memory allocations, will be used for the stack. To avoid the stack overflow, we use the following line of code

```
    mov         #__SPLIM_init,W0
```

Seeing this line the compiler thanks to the Data Space memory map, knows the limit and sets it to 0x47FE.

0.3. **Question Q3.** We are asked to find the faults in the below codes:

```
    mov W0, 0x0801
    add W0, #45, W1
```

(1) For the first line see that we are trying to write into a ODD memory location, which is not possible.
(2) For the second line we are trying an add operation with W0 and the number 45, though this will give an error due to number 45 being too large. Largest number we can fit in that instruction is 31, to get around this what we can do is the following:

```
    mov #45, W2
    add W0,W2,W1
```

0.4. **Question Q4.** Looking at the given SSD and the connections for 5 to be displayed, the LEDs that are connected to Ports:

- ”PB0”
- ”PB2”
- ”PB3”
- ”PB5”
- ”PB6”

should be turned on, and others turned off. Knowing this we need to set and clear the according bits of the PORTB of PIC24, as follows:

```
    mov  #0x0000, PORTB; Clear all the bits
    bset PORTB, #0
    bset PORTB, #2
    bset PORTB, #3
    bset PORTB, #5
    bset PORTB, #6
```

With a similar approach we can deduce that for the SSD to display 7

```
    mov  #0x0000, PORTB; Clear all the bits
    bset PORTB, #0
    bset PORTB, #1
    bset PORTB, #2
```

0.5. **Question Q5.** We are to write a lookup table for the correct SSD displays.

- ”To display 0: P0,P1,P2,P3,P4,P6 — In Binary: 1011111 — In Hexa: 5F”
- ”To display 1: P1,P2 — In Binary: 0000110 — In Hexa: 06”
- ”To display 2: P0,P1,P3,P4,P5 — In Binary: 0111011 — In Hexa: 3B”
- ”To display 3: P0,P1,P2,P3,P5 — In Binary: — In Hexa: 2F”
- ”To display 4: P1,P2,P5,P6 — In Binary: 1100110 — In Hexa: 66”
- ”To display 5: P0,P2,P3,P5,P6 — In Binary: 1101101 — In Hexa: 6D”
- ”To display 6: P0,P2,P3,P4,P5,P6 — In Binary: 1111101 — In Hexa: 7D”
- ”To display 7: P0,P1,P2 — In Binary: 0000111 — In Hexa: 07”
- ”To display 8: P0,P1,P2,P3,P4,P5,P6 — In Binary: 1111111 — In Hexa: 7F”
- ”To display 9: P0,P1,P2,P3,P5,P6 — In Binary: 1101111 — In Hexa: 6F”

Knowing all of these we can construct our look up table and write a simple function as follows:

```
    lookup: .byte 0x5F063B2F666D7D077F6F
    ssdDisplayer:
        mov #0x000F,W11 ;We will use this to get the last 4 bits of W0
        and W0,W1,W12
        mov #psvoffset(lookup), W11 ;W11 points to the start of the table
        mov.b [W11+W12],PORTB ; Get the appropriate value from thetable and move it
    return
```