ECE2049 E22: Homework 4 Solutions

Problem 1

a. RAM and Flash memory are used for different purposes. RAM is used for working memory during program execution; it is faster than flash memory, but is volatile, meaning that its data is lost when power is removed. Flash memory is non-volatile, and is thus required for long term storage (such as for code).

b. When code is downloaded from CCS to the MSP430, it is stored in flash is non-volatile so the program will remain on the board even if it is powered off.

c. False. The CPU interacts with peripherals in the same way it interacts with memory (ie. by performing read and write operations on particular memory addresses). No special CPU instructions are required to use peripherals.

Problem 2

a. The circuit in the diagram includes pull-up resistors already connected externally. Thus, the switches always have a defined state and thus there is no need to configure internal pull-up or pull-down resistors.

b. The LED is on when P2.0 is set to 1, which occurs when s is less than or equal to 1.
Thus, we can find the LED state by computing `s`:

`val`	`s`	LED state
0x7A	3	OFF
0x1C	1	ON

Problem 3

a. These switches would be digital inputs. Left would connect to V_{CC}, providing a logic 1 reading. Right would connect to GND, providing a logic 0 reading. Note that a switch works differently from a button because it can easily remain in either position, logic 0 or 1, without constant user input. A button defaults to one logic state and requires a user to hold it down to activate the other logic state.

We can write the configuration and `read_switches` as follows:

```
void config_switches(void)
{
    // Select I/O function
    P7SEL &= ~(BIT2);
    P4SEL &= ~(BIT6| BIT5 | BIT4);

    // Set as inputs
    P7DIR &= ~(BIT2);
    P4DIR &= ~(BIT6 | BIT5 | BIT4);

    // Since a switch is always connected to a logic input (ie, it has
    // no undefined states), we do not need to configure pull-up or
    // pull-down resistors in this case.
}

// Return a value between 0-Fh corresponding to the value of the
// switches, with the values of each switch in the following bit
// positions:
// MSB                              LSB
// Bits 7-4   Bit 3     2     1     0
//        0    P7.2  P4.6  P4.5  P4.4
char read_switches(void)
{
    char ret_val = 0;

    // Read switch values, store into variables containing
    // either 1 or 0 indicating value
    char s3 = (P7IN & BIT2) >> 2;
    char s2 = (P4IN & BIT6) >> 6;
    char s1 = (P4IN & BIT5) >> 5;
    char s0 = (P4IN & BIT4) >> 4;

    // Combine values as specified for function
    ret_val = (s3 << 3) | (s2 << 2) | (s1 << 1) | s0;

    return ret_val;
}
```

Part b

b. Using the values for `P4IN` and `P7IN`, `read_switches()` will return 0xC (1100b). The values for `P4OUT` and `P7OUT` are unneeded, as the output registers are not changed by input pins.