EE 215 Homework 5

(5 problems, 76 pts total)

1. MSP430 hardware (12 pts)
   Answer the following questions:
   a. What are the operating modes of the MSP430, how many are there? Why does the MSP430 have operating modes?
   b. What is an interrupt? Why is it used?
   c. What is an interrupt vector?
   d. Why does the MSP430 have a watchdog timer? How is the watchdog timer turned off?

   Answers:

   a. The MSP430 has 8 operating modes,LPM0 (Low Power Mode 0), LPM2, LPM3, and LPM4. The operating modes are for ultra-low-power applications and allow the MSP430 to balance power consumption and performance based on the application's requirements. For example, it can enter low-power modes when the CPU is not actively processing, conserving energy.

   b. An interrupt is a mechanism that allows the MSP430 to pause its current task and execute a separate, predefined routine in response to a specific event. Interrupts are used to handle time-critical tasks or events without constantly checking for them in a loop, improving efficiency.

   c. An interrupt vector is a memory address that points to the location of the interrupt service routine (ISR). located in the address range 0FFFFh to 0FF80h. When an interrupt occurs, the MSP430 uses the corresponding vector to find and execute the ISR, ensuring a rapid and predefined response to the interrupt.

   d. The MSP430 has a watchdog timer to monitor and recover from system failures or lock-ups. It's a timer that needs periodic resets to prevent it from triggering a system reset. If the MSP430 gets stuck or behaves unexpectedly, the watchdog timer can reset the system. To turn off the watchdog timer, a specific sequence of instructions needs to be executed, preventing unintentional resets.
2. Bit operations (14 pts)
   If a = 10010111, b = 10001111, find the answers after the logic operation:
   a. a | b
   b. a & b
   c. ~a
   d. a ^ b
   e. a ^= b
   f. a |= b
   g. a &= b

Answers:

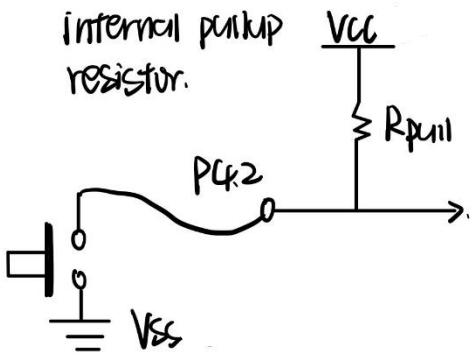
a. a | b = 1001 1111
b. a & b = 1000 0111
c. ~a = 0110 1000
d. a ^ b = 0001 1000
e. a ^= b (This updates the value of 'a' to the result of a ^ b, so 'a' becomes 0001 1000)
f. a |= b (This updates the value of 'a' to the result of a | b, so 'a' becomes 1001 1111)
g. a &= b (This updates the value of 'a' to the result of a & b, so 'a' becomes 1000 0111)

3. Switch setup (15 pts)
   For a switch connected to P4.2 with an internal pullup resistor:
   a. Draw the circuit.

b. Write the C program lines to set up the bits in P4DIR, P4REN, and P4OUT, and comment these lines.

Answers:

a.



b.

```
#include <msp430.h>
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // Stop the watchdog timer

    P4DIR &= ~BIT2;// Set P4.2 as input (clear bit in P4DIR)

    P4REN |= BIT2;// Enable internal pullup resistor for P4.2 (set bit in
P4REN)

    P4OUT |= BIT2;// Set P4.2 pullup resistor as active (set bit in P4OUT)

    return 0;
}
```

4. C with Switch and LED (15 pts)

Write a C program to turn on the LED (P1.0) only when the S1 (P2.1) is pushed. Configure the pullup resistor for the switch.

Answers:

```
#include <msp430.h>
#define CPU_F ((double)1000000) // CPU frequency for delay calculation
#define delay_ms(x) __delay_cycles((long)(CPU_F * (double)x / 1000.0)) //
Delay function in milliseconds

int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    P1DIR |= BIT0; // Set P1.0 as output (LED)

    P2REN |= BIT1; // Enable pull-up resistor on P2.1
    P2OUT |= BIT1; // Set P2.1 pull-up resistor as active

    while (1) {
        if (P2IN & BIT1) {
            delay_ms(20); // Debounce delay
            if (P2IN & BIT1)
                P1OUT &= ~BIT0; // Turn off LED when the button is pressed
        } else {
            delay_ms(20); // Debounce delay
            if ((P2IN & BIT1) == 0)
                P1OUT |= BIT0; // Turn on LED when the button is not pressed
        }
    }
    return 0;
}
```

5. C with Switch and LED (20 pts)
   Write a C program for the MSP 430, using switch 1 (S1) to control LED1, switch 2 (S2) to control LED2 separately. Use pullup resistors for the switch. LED1 is P1.0, LED2 is P1.1, S1 is P1.6, S2 is P1.7 .
   a. if press S1, LED1 is on
   b. if press S2, LED2 is on;
   c. if press both, both LEDs are on;
   d. Make comments on each line of code to explain your program

Answers:

```
#include <msp430.h> // Include MSP430 header file for register definitions
#define CPU_F ((double)1000000)
#define delay_ms(x) __delay_cycles((long)(CPU_F*(double)x/1000.0))

int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    P1DIR |= BIT0 | BIT1;       // Set P1.0 and P1.1 as outputs for LED1 and
LED2
```

```
    P1REN |= BIT6 | BIT7;       // Enable pull-up resistors on P1.6 (S1) and
P1.7 (S2)
    P1OUT |= BIT6 | BIT7;       // Set pull-up resistors as active for S1 and
S2

    while (1) {
        if ((P1IN & BIT6) == 0 && (P1IN & BIT7) == 0) // Both switches are
pressed, turn on both LEDs
        {
            delay_ms(20);                              // Debounce delay
            if ((P1IN & BIT6) == 0 && (P1IN & BIT7) == 0)
            P1OUT |= BIT0 | BIT1;
        }
        else if ((P1IN & BIT6) == 0)                  // S1 is pressed, turn
on LED1
        {
            delay_ms(20);
            if ((P1IN & BIT6) == 0)
            {
                P1OUT |= BIT0;
                P1OUT &= ~BIT1;                       // Turn off LED2
            }
        }
        else if ((P1IN & BIT7) == 0)                // S2 is pressed, turn on
LED2
        {
            delay_ms(20);
            if ((P1IN & BIT7) == 0)
            {
                P1OUT |= BIT1;
                P1OUT &= ~BIT0;                     // Turn off LED1
            }
        }
        else                                        // No switches pressed,
turn off both LEDs
        {
             P1OUT &= ~(BIT0 | BIT1);
        }
    }
    return 0;
}
```