EE 215 Homework 5 Solution

1. Bit banging (6 pts) Write one C program line to complete each task below. Explain the bit mask and show your work for bit wise operations.

a. Clear bit 3 of the register P2OUT, but keep other bits unchanged

the idea is to 'AND with a ZERO' `BIT3 = 0000 1000`

```
        P2OUT = XXXX XXXX
        ~BIT3 = 1111 0111
      ---------------------
P2OUT & ~BIT3 = XXXX 0XXX
```

The C code shorthand is `P2OUT &= ~BIT3;`

b. Set bit 5 of register TA0CTL, but keep other bits unchanged

the idea is to 'OR with a ONE'

```
        TAOCTL = XXXX XXXX
        BIT5   = 0010 0000
      ---------------------
TAOCTL | BIT5  = XX1X XXXX
```

The C code shorthand is `TAOCTL |= BIT5;`

c. Toggle bit 4 of register P1OUT, but keep other bits unchanged

the XOR function will toggle a bit

```
          P1OUT = 0101 0101          P1OUT = 0110 0101
          BIT4  = 0001 0000          BIT4  = 0001 0000
        ---------------------      ---------------------
P1OUT ^ BIT4    = 0100 0101                = 0111 0101
```

The C code shorthand is `P1OUT ^= BIT4;`

2. Timer A (14 pts) Explain the following words. If it is a register, explain all bits. If it is a substitute for bits (macro), write out the binary and hex equivalent and explain pin/clock connections. [x4xx Family User's Guide, x5xx Family User's Guide]

TA0CTL a register of 16 bits
bit 15-10 reserved
bit 9-8 TASSEL, timer A source select one of 4 clock sources
bit 7-6 input divider for the clock source
bit 5-4 mode control: stop, up, continuous, up/down
bit 3 reserved
bit 2 TACLR resets the TAxR timer value to zero
bit 1 TAIE flag to enable interrupt request in bit 0
bit 0 Timer A interrupt flag TAIFG

TAIFG a hex equivalent, 0b0000000000000001, 0x0001 it refers to BIT0 of TAOCTL
it is the timer A interrupt flag
is set when in UP mode and timer reaches value in TA0CCR0 register and resets to zero in TAR

TA0CCR0 a register of 16 bits, it holds a limit value for the timer A

MC_1 a hex equivalent, 0b0000000000010000 , 0x0010, selects UP mode on a timer

ID_2 a hex equivalent, 0b0000000010000000, 0x0080, divides the input clock by 4

TASSEL_1 a hex equivalent, 0b0000000100000000, 0x0100, selects the Aux Clock

TAIE a hex equivalent, 0b0000000000000010, 0x0002, enables the interrupt flag for Timer A

INCLK a hex equivalent, 0b0000001100000000, 0x0300, selects the inverted external clock.
ACLK a hex equivalent, 0b0000000100000000, 0x0100, selects the Aux Clock
SMCLK a hex equivalent, 0b0000001000000000, 0x0200, selects the Sub Main Clock

P2IN a register of 8 bits, each bit controls a pin on Port 2, senses if input is high (1) or low (0), and responds dynamically.

P2OUT a register of 8 bits, each bit controls a pin on Port 2, set high (1) to provide Vcc to corresponding output pin.

P2DIR a register of 8 bits, each bit controls a pin on Port 2, determines whether the pin is output (1) or input (0)

P2REN a register of 8 bits, each bit controls a pin on Port 2 enables the pullup/pulldown resistor

BIT0 a hex equivalent, 0b00000001, 0x01, used to set/clear Bit 0 in a register w/ bit wise logic

BIT3 a hex equivalent, 0b00001000, 0x08, used to set/clear Bit 3 in a register w/ bit wise logic

BIT7 a hex equivalent, 0b10000000, 0x80, used to set/clear Bit 7 in a register w/ bit wise logic

3. Clocks (6 pts) Explain the difference between the three clocks of the MSP430, and list their frequency of operation.

[Book p. 35]

The Master Clock, MCLK, is used by the CPU and a few peripherals. Typically 0.8 to 1.1 MHz from the DCO. The 5529 has a 25 MHz master clock. (DCO = digitally controlled oscillator)

The Subsystem Master Clock, SMCLK, is distributed to peripherals. Typically 0.8 to 1.1 MHz from the DCO.

The Auxiliary Clock, ACLK, is also distributed to peripherals. Its source is a low frequency crystal oscillator, typically 32.768 kHz.

4. C program, Timer A (14 pts) Write a C program for the MSP430. LED1 (P1.0) should initially be off. If S1 (P2.6) is pushed, blink the LED at ½ second intervals, using Timer A. If S1 is pushed again the cycle of 'off' or 'blinking' should repeat.

```
#include <msp430.h>
// LED off to start, blinking the LED if one press of S1
// press again, turn off.  Repeat.
unsigned int state, i;
int main(void)

{
      WDTCTL = WDTPW | WDTHOLD;     // stop watchdog timer
// set up LED
      P1OUT &=~ BIT0;              // LED off
      P1DIR |= BIT0;              // Set LED as output
// set up switch
      P2DIR &=~ BIT1; // input direction
      P2REN |= BIT1;  // input resistor enable
      P2OUT |= BIT1;  // pullup resistor

      // set up timer, but Mode Control 0 is "off"
      TA0CTL |=TASSEL_1|MC_0|TACLR|ID_0; // ACLK, count up to CCR,
divide clock by 1
      TA0CCR0 = 0x8000;  // counter limit 0x8000 = 32768 = 1 second

      state=0;  // LED off is state=0, LED blinking is state =1

      for(;;){
          if (((P2IN & BIT1)==0) && (state==0)){  // switch press and
state LED off
              // turn on blinking
              TA0CTL |= MC_1;  // turn on TimerA0
              state=1;        // LED blinking
              for (i=2000;i>0;i--);  // delay for button press

          }
          if (((P2IN & BIT1)==0) && (state==1)) {  // switch press and
LED on (then stop blink)
              TA0CTL &=~ BIT4; // turn off TimerA0
              P1OUT &=~ BIT0; // turn off LED
              state=0;        // LED state is not blinking
              for (i=2000;i>0;i--);  // delay for button press
          }
          if(TA0CTL & TAIFG){     //If flag is set from counting up
              P1OUT ^= BIT0;      //Toggle LED
              TA0CTL &= ~BIT0;    //Reset Flag
          }
      } // end for

}  // end main
```