EE215 Final Practice

1. Show the contents of ***R6*** after each of the following instructions.
   (evaluate each instruction individually/separately.)

Memory	
address	data
0x2400	77
0x 2401	6B
0x 2402	09
0x 2403	73
0x2404	C3

Registers	
R4	0x2400
R5	0xA022
R6	0x2300
R7	2
sp	2406

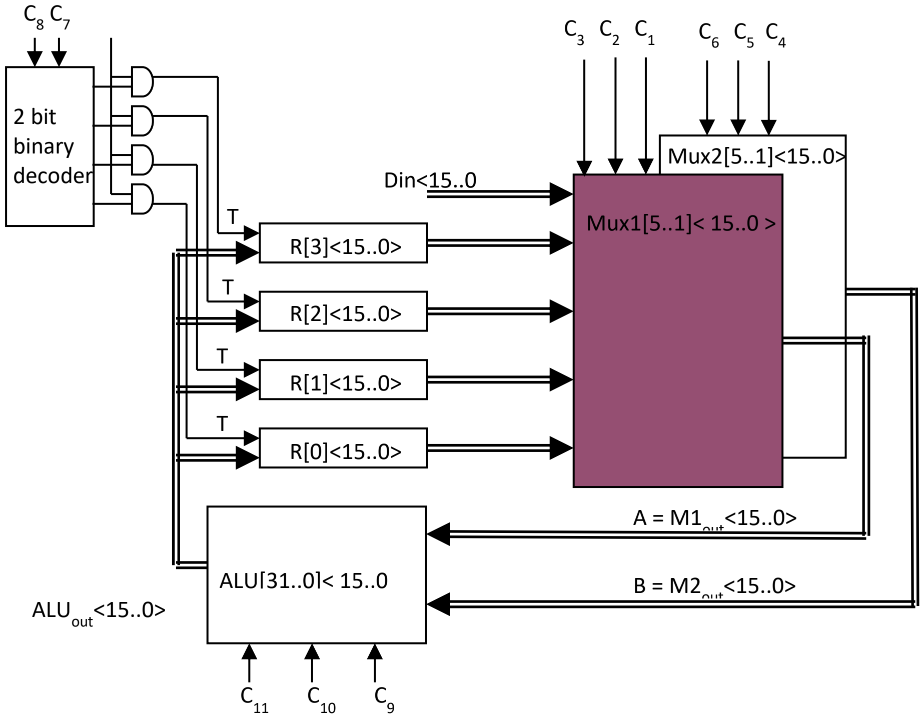
```
(a) bis.b  R7,R6

(b) inv.w  R6

(c) rra.w  R6

(d) xor.b  5(R4),R6
```

2. For the following control signals, C_2C_1 and C_5C_4 are used to select source register R[0]-R[3], C_3 and C_6 is used for D_{in} selection. If $C_3 = 1$, MUX1 output is D_{in}. If $C_6 = 1$, MUX2 output is D_{in}. C_8, C_7 are used for the 2 bit binary decoder to select destination register. $C_{11}C_{10}C_9$ are used for ALU selection, and the details are listed in the following table.

Control signal	000	001	010	011	100	101	110	111
Operation	A + B	A-B	A or B	A and B	A xor B	~A	A*B	A/B

(a) If $C_{11}C_{10}C_9 \;\; C_8C_7 \;\; C_6C_5C_4 \;\; C_3C_2C_1 = 011 \;\; 10 \;\; 011 \;\; 101$, describe the corresponding register transfer and ALU operation.

(b) To accomplish $R[2] \leftarrow R[1] + R[2] + R[3]$, how many steps do we need? What are the control signals for each step?



3. (a) Fill in the final value of the affected registers and the memory at the end of execution.
(b) What is the overall effect of this program?

```
         .data
First: .byte  9, 3, 0, 12,
Last:  .byte  7

             .text
             mov #First, R10
             mov #Last,  R11
             call #Sub
             jmp $

Sub:
L1:          mov.b @R10, R5
             cmp.b #10, R5
             jn Clear
             inc R10
             cmp R10, R11
             jge L1
             jmp End
Clear: mov.b  #0, 0(R10)
             inc R10
             cmp R10, R11
             jge L1

End:          ret
```

(a)

address of memory	contents of memory (in **hexadecimal**)
0x 2400	0x
0x 2401	0x
0x 2402	0x
0x 2403	0x
0x 2404	0x
0x 2405	0x
0x 2406	0x
0x 2407	0x

register	contents of register (in **hexadecimal**)
R5	0x
R10	0x
R11	0x

(b)

4. Write a code to turn on LED1 and toggle (flash) LED2.

```
#include <msp430.h>
void main(void)
{
    WDTCTL = WDTPW | WDTHOLD;

   _________________________________

   _________________________________

   _________________________________

   _________________________________

   _________________________________

   _________________________________

   _________________________________

   _________________________________

   _________________________________

   _________________________________

   _________________________________

   _________________________________

}
```

5. Use Timer A0 to generate interrupt.
   Please configure TA0CTL register, TA0EX0 register, and TA0CCR0 register to
   - choose ACLK (32768 Hz) as the clock source, use **up/down** mode, and clear the TAR register initially, and
   - use Timer A0 to generate interrupt with a period of 32 seconds with a total (/16) divider for the input clock.

6. For the following program, P3.6 is LED, P2.2 is Switch1, P1.4 is Switch 2.
   (a) comment the code followed by //________________ . Write your comment on the underline.
   (b) describe the overall effect.

```
#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;    //__________________________________

    P1DIR |=BIT0;   //  __________________________________

    P4DIR |=BIT7;   //  __________________________________

    P1OUT &=~BIT0;  //  __________________________________

    P4OUT &=~BIT7;  //  __________________________________

    P2DIR &=~BIT1;  //  __________________________________

    P1DIR &=~BIT1;  //  __________________________________

     P2REN |=BIT1; //    __________________________________

     P2OUT |=BIT1;   // __________________________________

     P1REN |=BIT1;

     P1OUT |=BIT1;

    P2IE |=BIT1; //   __________________________________

    P2IES|=BIT1; //   __________________________________

    P2IFG&=~BIT1; //  __________________________________

    P1IE |=BIT1;
    P1IES|=BIT1;
     P1IFG&=~BIT1;

     __enable_interrupt();  //__________________________________
  }


#pragma vector = PORT2_VECTOR   //  __________________________________
__interrupt void PORT2_ISR(void)
{
    while((P2IN&BIT1)==0){

                 P4OUT|=BIT7;     //__________________________________
            }

        P4OUT&=~BIT7;   // __________________________________

        P2IFG &=~BIT1;  // __________________________________
}

#pragma vector = PORT1_VECTOR
__interrupt void PORT1_ISR(void)
{
    while((P1IN&BIT1)==0){

             P4OUT|=BIT7;
        }
    P4OUT&=~BIT7;
    P1IFG &=~BIT1;
}
```