Ruide Xie

Lab report 1

1.) In our bit-shifting algorithm, what is the purpose of the line of line of code: **`LATB |= 0x8000;`** that occurs immediately after the shift? What would happen if this line was not included?

In task 2, this line of code plays the role of turning the LED attached to RB15 pin off after RB14 was shifted to turn on, while having no effect to all other bits. The reason is that 0x8000 only has 1 on the most significant bit and when ORed with LATB only RB15 will be switched to 1 and other bits stay the same. If there is no such line of code, LEDs RB15 and RB14 will be on at the same time.

2.) One important bitwise operation not discussed in the lab is bitwise negation (inversion). This can be performed using the "exclusive-or" operation (implemented with "^" in C). Suppose I want to invert the most significant bit in a byte of data but leave the other seven bits unchanged. Write a C function that takes a byte of data as an argument and uses a bitmask, and an exclusive-or to invert the most significant bit of that data.

Suppose a variable data is a byte of data that has 16 bits, the following function can be implemented.

```
int shift01(int data){
     data ^= (1<<15);
     return data;
}
```

3.) What was the error that caused the delay function to hang? List two possible solutions to this problem.

The error occurred because the int curCount was set out of bound which should be 32767 as maximum. One solution is to reduce the looping condition from 40000 to an integer that is less than or equal to 32767. Another solution is to make curCount a larger data type, for instance long.

4.) Explain why the LED turns on when the value of the pin it is connected to is set LOW and off when it is set HIGH. Why isn't it the other way around?

LEDs are diodes which means current only flows when it is forward biasing from + to -. In the breadboard design, the LEDs are connected high to VDD on one end and that determines that the other end should be connected to the opposite of VDD.

Task 4 code:

```
int main(void) {
    setup();
    LATB = LATB >> 1;
    delay();
    while(1) {
        // Execute repeatedly forever and ever and ever and ever ...
        LATB = (LATB >> 1) | 0x8000;
        delay();
        LATB = (LATB >> 1) | 0x8000;
        delay();
        LATB = (LATB >> 1) | 0x8000;
        delay();
        LATB = (LATB << 1) | 0x1001;
        delay();
        LATB = (LATB << 1) | 0x1003;
        delay();
        LATB = (LATB << 1) | 0x1007;
        delay();
    }
    return 0;
}
```

First shift LATB(0xffff) to the right 1 bit making the MSB 0 0x7fff others unchanged, Then shift LATB(0x7ffff)to the right 1 bit and ored with 0x8000, making RB14 0 and RB15 1 and others unchanged. Then shift LATB(0xbfff)to the right 1 bit and ored with 0x8000, making RB13 0 and RB14 1 and others unchanged. Then shift LATB(0xdfff) to the right 1 bit and ored with 0x8000, making RB12 0 and RB 13 1 and others unchanged. Then shift LATB(0xefff) 1 bit to the left and ored with 0x1001, making RB12 and RB0 1 and RB13 0 and others 1. Then shift LATB(0xdfff) 1 bit to the left ored with 0x1003, making RB13 1 and RB14 0 and others unchanged. Then shift LATB(0xbfff) 1 bit to the left ored with 0x1007, making RB14 1 and RB15 0 and others 1. This whole process runs in an infinite loop which makes only one LED lights up at a time and it goes one after another.