ECE 2049: Homework #2

Due on Monday, February 01, 2015

Ted Meyer

Problem 1

Label	Address	Little Endian	Big Endian
ss	02421	C0	00
ss	02420	AC	00
ss	0241F	00	AC
ss	0241E	00	C0
class[7]	0241D	39	39
class[6]	0241C	34	34
class[5]	0241B	30	30
class[4]	0241A	32	32
class[3]	02419	2D	2D
class[2]	02418	45	45
class[1]	02417	43	43
class[0]	02416	45	45
ser_num	02415	9B	F6
ser_num	02414	15	75
ser_num	02413	19	00
ser_num	02412	AC	12
ser_num	02411	12	AC
ser_num	02410	00	19
ser_num	0240F	75	15
ser_num	0240E	F6	9B
trp	0240D	00	04
trp	0240C	00	00
trp	0240B	00	01
trp	0240A	00	00
trp	02409	00	00
trp	02408	01	00
trp	02407	00	00
trp	02406	04	00
a[1]	02405	FF	FF
a[1]	02404	FF	FF
a[0]	02403	00	7F
a[0]	02402	7F	00
ii	02401	01	00
ii	02400	00	01

The MSP430 is a little endian RISC architecture.

Problem 2

A) the msp430 has both RAM and FLASH because both have separate functions. FLASH (non-volatile) memory is cheaper and retains it's state when power is lost, but has slower write times than RAM. Random access memory (RAM) has fast write times in addition to its fast read times, but it does not reatain information when the power is lost. Using just RAM would be expensive and would require a reprogramming step every time the chip is re-powered, and using only FLASH would cause poor performance.

Problem 2 continued on next page. . .

B) In total, the MSP430f5529 has 10 KB of RAM, 2 of which is for the USB, but can be re-purposed provided the USB is not in use. There are 4 sectors of ram, each containing 2KB, the first of which starts at address: 0x002BFF

C) The MSP430 has 128 KB of FLASH memory. There are two types of FLASH memory, main memory and information memory. The information memory is divided into 4 sedments labeled A through D of 128 bytes each, and the main memory is divided into 255 segments of 512 bytes each. The address `0x00B04A` is in the 87th bank of main flash memory.

D) The code is stored at address `0x4400`

E) A 20 bit adress bus allows it to communicate at a true 1 Megabit range, instead of the normal 1 Mebibit range.

F)

Device	I/O Ports & Pins	Package Pins
2 push buttons	P1.7, P2.1-2	28, 31
LEDs 1-3	P1.0, P8.1-2	21, 15, 16
Touch Pad LEDs	P1.1-5	22-26
Scroll Wheel	P8.0	

Problem 3

A)

```
#include "msp430.h"

int main(void)
{   // give the size of these variable
    int j,k,indx=1;         // declare three 16 bit integers, and set indx to 1
    long unsigned int sqSum; // declare a lon unsigned int (32 bits)

    WDTCTL = WDTPW + WDTHOLD;   // Stop watchdog timer

    j = 1;      // set j to 1
    k = 84;   // set k to 84
    while(indx < 64)    // do the following while index is less than 64
    {
        k=k/j;  // divide k by j and put the result back into k
        j*=2;   // multiply j by 2 (in place)
        sqSum += indx*indx; // add index squared to sqSum
        indx = indx<<1; // multiply index by 2
    }
}
```

This program would take 80 bits of stack space to declare the variables, and would then be fairly quick in terms of operations, due to the fact that the loop is only run 6 times, in which there are 8 assembly operations per iteration. This is fairly quick.

Problem 3 continued on next page. . .

B)

```
#include "msp430.h"
#include <math.h>

void main(void)
{   // give the size of these variables
    float d, e; // declare d and e (both 32 bits)
    int lp=1; // declare lp and set its value to 1 (16 bits)
    long unsigned int sqSum; // declare sqSum (32 bits)

    WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer

    d = 0.75; // set d to 0.75
    e = 84.5; // set e to 84.5
    while (lp < 64) // do loop while lp is less than 64
    {
        sqSum += pow(lp,2); // add lpˆ2 to sqSum
        e=e/d; // divide e by d
        d*=2; // multiply d by 2
        lp=(lp<<1); // multiply lp by 2
    }
}
```

This program would take 112 bits of stack space to store its variables, and it's main loop would also only run 6 times. However, because there is a call to pow() in the loop, there are 6 instances of the stack being extended, populated, and then discarded, in addition to the operations that take place inside the pow function. Because the MSP430 does not have a FPU, each floating point operation takes multiple assembly instructions, and is significantly slower than the previous example. Unrelated, but the main method also has the wrong prototype.

Problem 4

Pin 1 is being used as output, and pin 3 is being used as input. If the value of P3IN is `0x66`, then P1OUT is `0x66 << 1`, or `0xCC`.

Problem 5

A) The 7 segment display is an output

```
void config7seg() {
    P2SEL &= 0x00;
    p2DIR &= 0xFF;
}
```

B)

```
void sevenSegIO(char inVal, char DP) {
    char WRITE = DP ? BIT0 : 0x00;
    switch(inVal) {
        case '0': P2OUT = (WRITE | BIT1 | BIT2 | BIT3 | BIT4 | BIT5 | BIT6);
            break;
        case '1': P2OUT = (WRITE | BIT1 | BIT2);
            break;
        case '2': P2OUT = (WRITE | BIT1 | BIT2 | BIT4 | BIT5 | BIT7);
            break;
        case '3': P2OUT = (WRITE | BIT1 | BIT2 | BIT3 | BIT4 | BIT7);
            break;
        case '4': P2OUT = (WRITE | BIT2 | BIT3 | BIT6 | BIT7);
            break;
        case '5': P2OUT = (WRITE | BIT1 | BIT3 | BIT4 | BIT6 | BIT7);
            break;
        case '6': P2OUT = (WRITE | BIT1 | BIT3 | BIT4 | BIT5 | BIT6 | BIT7);
            break;
        case '7': P2OUT = (WRITE | BIT1 | BIT2 | BIT3);
            break;
        case '8': P2OUT = (WRITE | BIT1 | BIT2 | BIT3 | BIT4 | BIT5 | BIT6 | BIT7);
            break;
        case '9': P2OUT = (WRITE | BIT1 | BIT2 | BIT3 | BIT4 | BIT6 | BIT7);
            break;
        case 'A': P2OUT = (WRITE | BIT1 | BIT2 | BIT3 | BIT5 | BIT6 | BIT7);
            break;
        case 'B': P2OUT = (WRITE | BIT3 | BIT4 | BIT5 | BIT6 | BIT7);
            break;
        case 'C': P2OUT = (WRITE | BIT1 | BIT4 | BIT5 | BIT6);
            break;
        case 'D': P2OUT = (WRITE | BIT2 | BIT3 | BIT4 | BIT5 | BIT7);
            break;
        case 'E': P2OUT = (WRITE | BIT1 | BIT4 | BIT5 | BIT6 | BIT7);
            break;
        case 'F': P2OUT = (WRITE | BIT1 | BIT5 | BIT6 | BIT7);
            break;
    }
}
```

Problem 6

A)

```
void setupP4() {
        P4SEL &= 0x83;
        P4DIR &= 0x83;
}
```

B)

```
void setupP3() {
        P3SEL &= 0xE1;
        P3DIR |= 0x1E;
}
```

C)

```
void InOut() {
        char val = (P4IN >> 3) & 0x0F;
        if (val >= 8) {
                P3OUT |= (~(val<<1) & 0x1E);
        } else {
                P3OUT |= ((val<<1) & 0x1E);
        }
}
```

D)

```
#include "msp430.h"

int main() {
        setupP4();
        setupP5();

        while(1) {
                InOut();
        }
}
```