Union College
ECE352
Assignment 2 Solution

Due Date: Thursday April 17th

Problem 1.

Complete the following using stacks. This program is to be written in assembler.

The stack is a "first in, last out" structure. It is important to initialize the stack pointer so that your stack is large enough to contain all the bytes that you will push onto it. The stack pointer is initialized to 0x07, so the first value pushed onto the stack is stored at 0x08, which is the value of R0 in register bank 1. If more than one register bank is to be used, the stack pointer should be moved at the beginning of the program. Choose a location in the data memory that is not being used. We will use location 0x30 as the first stack data value, so we will initialize the stack to 2Fh.

Write a program that initializes the stack pointer, and then pushes the following 4 memory locations on the stack, and then pops them back out so that the data is re-ordered. The following "before" and "after" example illustrates how it should work.

Before		**After**	
Address:	Data	Address:	Data
70	34	70	89
71	f2	71	66
72	66	72	f2
73	89	73	34

Since you do not know what data is initially in RAM, add some instructions to initialize these memory locations so that you can tell if your program works. (If the locations all held the same data value, then you would not be able to see the changes). Name your program stacks.asm and include a printout of your program with your assignment.

```
;-------------------------------------------------------------------------------
; Program Name: stack.asm
; Author: prof. Hedrick
; Date:
; Purpose: fun with stacks
;-------------------------------------------------------------------------------

$include (c8051F020.inc) ; Include register definition file
;-------------------------------------------------------------------------------
; RESET and INTERRUPT VECTORS
;-------------------------------------------------------------------------------

; Reset Vector
   CSEG AT 00h
   ljmp Main     ; Jump beyond interrupt vector space.

;-------------------------------------------------------------------------------
; CODE SEGMENT
;-------------------------------------------------------------------------------
        cseg at 2000h
Main:    mov WDTCN, #0DEh   ; disable watchdog timer
        mov WDTCN, #0ADh
        mov SP, #0x27       ; initialize stack pointer

        ; store initial values in locations 70h to 73h
        mov 0x70, #34h
        mov 0x71, #0f2h
        mov 0x72, #66h
        mov 0x73, #89h

       ; push the values from locations 70h to 73h on the stack
       push 0x70
       push 0x71
       push 0x72
       push 0x73

       ; pop values on stack back to locartions 70h to 73h and order will be reversed
       pop 0x70
       pop 0x71
       pop 0x72
       pop 0x73
       nop
end
```

Problem 2.

Write a program to move 4 data values from external memory, starting at 2000h, to RAM, starting at address 20h. After they are moved, put bit 4 of location 21 in the carry flag.

```
mov DPTR, #2000h
mov R0, #20h
loop1:
        movx A, @DPTR
        mov @R0, A
        inc R0
        inc DPTR
        cjne R0, #24h, loop1

mov C, 21h.4
```

Problem 3.

Write a program to compute the equation: Z = X/(Y - P). You may assume that X, Y, and P are held in registers R0, R1, and R2, and the Z result should be written to memory at location 60h.

```
mov A, R1
subb A, R2      ; form (Y-P)
mov B, A
mov A, R0
div AB          ; from X/(Y-P)

mov 60h, A      ; quotient
mov 61h, B  ; remainder
```

Problem 4.

In class we wrote a program to blink the light using timer 0 and polling the overflow flag in C. Rewrite this program in assembler.

```
;-------------------------------------------------------------------------
; Program Name: timer1.asm
; Author:Prof. Hedrick
; Date: 03/28/2012
; Purpose: use timer 0 to blink using 16 bit mode and the internal 2 MHz clock
; Use polling to determine when timer overflow occurs
;-------------------------------------------------------------------------
#pragma debug list
$include (c8051F020.inc) ; Include register definition file

;-------------------------------------------------------------------------
; EQUATES
;-------------------------------------------------------------------------
GREEN_LED equ P1.6

;-------------------------------------------------------------------------
; RESET and INTERRUPT VECTORS
;-------------------------------------------------------------------------

; Reset Vector
   CSEG AT 00h
   ljmp Main     ; Jump beyond interrupt vector space.

;-------------------------------------------------------------------------
; CODE SEGMENT
;-------------------------------------------------------------------------
        cseg at 0x100
Main:    mov WDTCN, #0DEh   ; disable watchdog timer
        mov WDTCN, #0ADh
        mov SP, #2Fh       ; initialize stack pointer

; set up port 1 pin 6 as an output
        orl P1MDOUT, #40h
; set up port 3 as inputs
        mov P3MDOUT, #0
             mov P3, #0xFF
; enable crossbar
                              mov XBR2, #40h
; initialize LED to off
        clr GREEN_LED

; configure timer 0
```

```
;    CKCON -> clock configuration
;    TMOD  -> mode of timer 0
;    TCON -> set timer on
;    TH0 and TL0 initialize timer 0  count
     mov CKCON, #0x00 ; set the clock to divide by 12
                              mov TMOD, #0x01 ; set  counter to mode 16b

                              anl TL0, #0x00 ; set initial count to 0
                              anl TH0, #0x00

               orl TCON, #0x10 ; enable timer

                              setb GREEN_LED

Repeat:
     jnb TF0, $
                              cpl GREEN_LED
                              clr TF0
                              sjmp Repeat

                              nop
END
```

Problem 5.

In class we wrote a program to blink the light using time 0 and generating an interrupt from the overflow flag in C. Rewrite this program in assembler.

```
;-------------------------------------------------------------------------
; Program Name: timer3.asm
; Author:Prof. hedrick
; Date: 03/28/2012
; Purpose: use timer 0 to blink using 16 bit mode and the internal 2 MHz clock
; Use an iterrupt to determine when timer overflow occurs
;-------------------------------------------------------------------------
#pragma debug list
$include (c8051F020.inc) ; Include register definition file

;-------------------------------------------------------------------------
; EQUATES
;-------------------------------------------------------------------------
GREEN_LED equ P1.6

;-------------------------------------------------------------------------
; RESET and INTERRUPT VECTORS
```

```
;-------------------------------------------------------------------------------

; Reset Vector
   CSEG AT 00h
   ljmp Main     ; Jump beyond interrupt vector space.

 cseg at 000Bh   ; ISR vector in interupt table
                 ljmp TM0_ISR

;-------------------------------------------------------------------------------
; CODE SEGMENT
;-------------------------------------------------------------------------------
       cseg at 0x100
Main:    mov WDTCN, #0DEh   ; disable watchdog timer
       mov WDTCN, #0ADh
       mov SP, #2Fh       ; initialize stack pointer

; set up port 1 pin 6 as an output
       orl P1MDOUT, #40h
; set up port 3 as inputs
       mov P3MDOUT, #0
                                  mov P3, #0xFF
; enable crossbar
                            mov XBR2, #40h
; initialize LED to off
       clr GREEN_LED

; configure timer 0
;     CKCON -> clock configuration
;     TMOD  -> mode of timer 0
;     TCON -> set timer on
;     TH0 and TL0 initialize timer 0  count
     mov CKCON, #0x00 ; set the clock to divide by 12
                          mov TMOD, #0x01 ; set  counter to mode 16b

                          anl TL0, #0x00 ; set initial count to 0
                          anl TH0, #0x00

             orl TCON, #0x10 ; enable timer

                          setb GREEN_LED
; set up interupt
;      enable global interupts
;      enable Timer 0 interupt
       orl IE, #82h
here: sjmp here     ; loop here waiting for interupts
```

```
;-------------------------------------------------------
; TM0_ISR -> timer 1 Interupt Service Routine
;-------------------------------------------------------


                cseg at 3000h ; put ISR in code memory but leave space for other code
TM0_ISR:
    cpl GREEN_LED
                clr TF0         ; TF0 is cleared by hardware anyway
                reti
;-----------------------------------------------------------------------
; end of timer 0 ISR code
;-----------------------------------------------------------------------

END
```

Problem 6.

Modify the C program we wrote in class to display the numbers 0 to F on the seven segment display to use timer0 and polling the overflow flag instead of nested loops.

```
//-----------------------------------------------------------------------------------
// Program: sevenseg.c
// Author: Prof Hedrick
// Purpose: write program to drive a seven segment display connected to port 0
// and use timer using 16 bit mode and polling for delay.
//-----------------------------------------------------------------------------------
#pragma CODE          //generate assembly code in the LST file
#include <c8051f020.h>                  // SFR declarations

//-----------------------------------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------------------------------

sbit  LED = P1^6;
sbit  SW = P3^7;

unsigned char digits[] = {0x01,0x79, 0x12, 0x06, 0x4C, 0x24,
                0x20,0x0F, 0x00, 0x04, 0x08, 0x60,
                                                0x31, 0x42,0x30,0x38};
//-----------------------------------------------------------------------------------
```

```
// Function PROTOTYPES
//-------------------------------------------------------------------------------

void PORT_Init (void);
void Timer_Init(void);
void delay(void);


//-------------------------------------------------------------------------------
// MAIN Routine
//-------------------------------------------------------------------------------
void main (void) {
int index;

          WDTCN = 0xde;                 // disable watchdog timer
          WDTCN = 0xad;
          PORT_Init ();
          Timer_Init();

    for (; ;) {
       for (index = 0; index < 16; index++) {

                P0 = digits[index];
          delay();
               }
       delay();
           }


}

/*
  generate delay using timer0
  .5 uS * 12 *65536 = 393 ms
*/
void delay(void) {
  // wait for timer to overflow
  while ( ! TF0);
  TF0 = 0;

}

//-------------------------------------------------------------------------------
// PORT_Init
//-------------------------------------------------------------------------------
//
```

```
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
  XBR2    = 0x40;                 // Enable crossbar and weak pull-ups

  P1MDOUT |= 0x40;                // enable P1.6 (LED) as push-pull output
  P0MDOUT = 0xFF;                 // enable port 0
}



void Timer_Init(void){
    /*
          CKCON -> clock configuration
           TMOD -> mode of timer 0
           TCON -> set timer 0 on
           THO and TL0 count registers - initialize to 0 for AMXIMUM COUNT
           */
     CKCON = 0x00;  // SET THE CLOCK TO DIVIDE BY 12
           TMOD = 0x01;  // set mode to 16b

           TL0 = 0x00;  // set initial count to 0
           TH0 = 0x00;

           TCON = TCON | 0x10; // enable timer
     LED = 0;

}
```

Problem 7.

Modify the C program we wrote in class to display the numbers 0 to F on the seven segment display to use timer0 and interrupts instead of nested loops. Write code for the ISR such that the delay between displays is close to 2 seconds.

```
//-----------------------------------------------------------------------------
// Program: sevenseg.c
// Author:Prof Hedrick
// Purpose: write program to drive a seven segment display connected to port 0
//  and use timer using 16 bit mode and an interrupt for delay.
//-----------------------------------------------------------------------------
#pragma CODE          //generate assembly code in the LST file
#include <c8051f020.h>                  // SFR declarations

//-----------------------------------------------------------------------------
// Global CONSTANTS
//-----------------------------------------------------------------------------

sbit  LED = P1^6;
sbit  SW = P3^7;

unsigned char digits[] = {0x01,0x79, 0x12, 0x06, 0x4C, 0x24,
                    0x20,0x0F, 0x00, 0x04, 0x08, 0x60,
                                                0x31, 0x42,0x30,0x38};
//-----------------------------------------------------------------------------
// Function PROTOTYPES
//-----------------------------------------------------------------------------

void PORT_Init (void);
void Timer_Init(void);
void delay(void);


//-----------------------------------------------------------------------------
// MAIN Routine
//-----------------------------------------------------------------------------
void main (void) {


        WDTCN = 0xde;                   // disable watchdog timer
        WDTCN = 0xad;
        PORT_Init ();
        Timer_Init();
```

```
    for (; ;) {

            }


}

/*
  generate an interupt using timer0
  .5 uS * 12 *65536 = 393 ms
  Want a delay of 2 seconds so count interupts.
  2/ .393 = about 5.1 so use 5
*/
void T0ISR(void) interrupt 1 {
static int count;
static int IRQCount;

   if (IRQCount >= 5) {
   /* display digits 0 to F */
      if (count > 15) {
              count = 0;
      }
            P0 = digits[count];
      count++;

            IRQCount = 0;
   } else {
      IRQCount ++;
   }
   TF0 = 0;
}

//-----------------------------------------------------------------------------
// PORT_Init
//-----------------------------------------------------------------------------
//
// Configure the Crossbar and GPIO ports
//
void PORT_Init (void)
{
   XBR2    = 0x40;                  // Enable crossbar and weak pull-ups

   P1MDOUT |= 0x40;                 // enable P1.6 (LED) as push-pull output
   P0MDOUT = 0xFF;                   // enable port 0
}
```

```
void Timer_Init(void){
    /*
          CKCON -> clock configuration
           TMOD -> mode of timer 0
           TCON -> set timer 0 on
           THO and TL0 count registers - initialize to 0 for AMXIMUM COUNT
           */
     CKCON = 0x00;  // SET THE CLOCK TO DIVIDE BY 12
           TMOD = 0x01;  // set mode to 16b

           TL0 = 0x00;  // set initial count to 0
           TH0 = 0x00;

           TCON = TCON | 0x10; // enable timer
     LED = 0;
           IE = 0x82;   // Enable TIMER0 and global interupts.


}
```