EE 2004
2020-21: Semester B
Assignment 2
Due: Mar. 29, 2021

Instructions:

You will submit the following 5 files: (1) `ConvertHexArrayToASCIIArray.asm`, (2) `ConvertHexArrayToASCIIArray.lst`, (3) `InterruptProgram.asm`, (4) `InterruptProgram.lst` and (5) The Word file showing your calculation in Question 2(a). All files should be zipped and submitted as a single zip file.

Students will submit the assignment through Canvas. Click on the item "Assignment" on the left panel. You should see a row with title "Assignment 2". Click on the "Assignment 2" label and find the "Submit Assignment" label on the right panel. Click on it and upload the requested zip file.

Question 1 (50 marks)

(a) (10 marks) Write a subroutine `ConvertHexToASCII` that implements a lookup table converting a hexadecimal digit ranging from 0 to F to its corresponding ASCII representation. Please use the `tblrd` mechanism (instead of the computed goto method). Store the output in WREG. The lookup table is provided below. (Hint: Use the program `LookupTable.asm` I provided as a blueprint and make appropriate modifications. Also recall what you did in the lab when you implemented the `SVN_SEG` subroutine.)

Hexadecimal	ASCII
0	0x30
1	0x31
2	0x32
3	0x33
4	0x34
5	0x35
6	0x36
7	0x37
8	0x38
9	0x39
A	0x41
B	0x42
C	0x43
D	0x44
E	0x45
F	0x46

(b) (40 marks) Given an array of 4 bytes stored in data memory location 0x040 through 0x043 as shown in Fig. 1, write a **loop** that writes the ASCII codes corresponding to the most and least significant nibbles stored in the data memory address 040 + *i* to data memory addresses 050 + 2*i* and 050 + 2*i* + 1, respectively, where *i* = 0, 1, 2, 3 (See Fig. 2). Use `ConvertHexToASCII` you wrote in (a) to convert a hexadecimal digit to its corresponding ASCII representation. You can choose to use the program template shown in Fig. 3. Your program **must** involve a loop and you must use FSRs to point to the arrays starting from data memory with address 0x040 and 0x050. **For this question, you will submit a single `.asm` file, named `ConvertHexArrayToASCIIArray.asm`, containing the subroutine `ConvertHexToASCII` you implemented in (a) and the program you write in (b). Please also submit the `.lst` file generated when you assemble your `.asm` file.**

Data Memory Address	Value
0x040	0xF1
0x041	0x54
0x042	0xAC
0x043	0x3F

Figure 1 Array storing hexadecimal digits

Data Memory Address	Value
0x050	0x46
0x051	0x31
0x052	0x35
0x053	0x34
…….	……

Figure 2 Expected result after your program completes execution

```
                ORG        0x000000
Main:           movlw 0xF1; Initialize array starting from
0x040
                movwf 0x40, A
                movlw 0x54
                movwf 0x41, A
                movlw 0xAC
                movwf 0x42, A
                movlw 0x3F
                movwf 0x43, A

                ; Other initializations

MainLoop:

                bra $

ConvertHexToASCII: ;Subroutine you wrote in Part (a)

                END
```

Figure 3 Program template for Question 1

```
; EE 2004
; Assignment 2 Q1
; ----------------------------------------------------------------------
    LIST   P=18F4520     ;directive to define processor
    #include <P18F4520.INC>    ;processor specific
;variable definitions
; ----------------------------------------------------------------------

    CBLOCK 0x000
    Count
    InputConvertHexToASCII
    CountSub
    ENDC

    ORG 0x000000
Main:                   movlw 0xF1; Initialize array starting from 0x040
    movwf 0x40, A
    movlw 0x54
    movwf 0x41, A
    movlw 0xAC
    movwf 0x42, A
    movlw 0x3F
    movwf 0x43, A
; Other initializations
    movlw 0x04;totally 4 iterations needed
    movwf Count, A
    lfsr 0, 0x040
    lfsr 1, 0x050


MainLoop:        movlw 0xF0
    andwf INDF0, W; Extract the most significant nibble
    swapf WREG, W
    movwf InputConvertHexToASCII
    call ConvertHexToASCII
    movwf POSTINC1

    movlw 0x0F
    andwf POSTINC0, W; Extract the least significant nibble
    movwf InputConvertHexToASCII
    call ConvertHexToASCII
    movwf POSTINC1

    decfsz Count, F, A
    bra MainLoop
    bra $


;Subroutine required in Part (a)
ConvertHexToASCII:             movlw  upper ASCIIList
           movwf  TBLPTRU
           movlw  high ASCIIList
    movwf  TBLPTRH
    movlw  low ASCIIList
    movwf  TBLPTRL
    movf   InputConvertHexToASCII, W, A
    incf   WREG, F; Read InputConvertHexToASCII + 1 number of times to get our result
    movwf   CountSub, A
Loop:                          tblrd*+
                               decfsz CountSub, F, A
    bra Loop
    movf TABLAT, W   ;Result in WREG
                               return

    org 0x000200
ASCIIList db 0x30, 0x31, 0x32, 0x33, 0x34, 0x35,0x36, 0x37, 0x38, 0x39, 0x41, 0x42, 0x43, 0x44,
0x45, 0x46

    END
```

Question 2 (50 marks)

(a) (5 marks) Determine the initial value of TMR0 to generate a 0.5s time delay using a prescaler = 8. The clock frequency of the PIC18 microcontroller is 4MHz. **Submit a Word file showing your calculation.**

Time delay generated = (65536 – x) * Prescaler * 1us = 0.5 s, where x is the initial value to be loaded to TMR0.

(65536 – x) * 8 = 500,000
x = d'3036'= 0x0BDC

(b) (45 marks) Suppose Port C is connected to the 8-button keypad, Port D is driving 8 LEDs, two switches are connected to INT0 (RB0) and INT1 (RB1), two LEDs are connected to RB6 and RB7. Write an interrupt-based program to:

- Send the voltage levels in Port C continuously to the file register with address 0x001.
- Set up INT0 as a high-priority interrupt to toggle the LED connected to RB6 every time when the switch connected to INT0 is turned on.
- Set up INT1 as a high-priority interrupt to toggle the LED connected to RB7 every time when the switch connected to INT1 is turned on.
- Set up Timer 0 interrupt as a low-priority interrupt to increment Port D every 0.5s.

You can make use of the following template. **You will submit a `.asm` file, named `InterruptProgram.asm` and the `.lst` file generated when you assemble your `.asm` file.**

```
           ORG 0x000000
           bra Main
           ORG 0x000008
           ; Check which interrupt flag is raised, branch to the
           ; appropriate ISR.

           ORG 0x000018
           bra T0_ISR

           ORG 0x000100
Main:

Over:
           bra Over


T0_ISR:

           retfie


INT0_ISR:


           retfie


INT1_ISR:
```

```
            retfie

            END
```

```
; EE 2004
; Assignment 2 Q2
; ------------------------------------------------------------------------------------
    LIST  P=18F4520                 ;directive to define processor
    #include <P18F4520.INC>         ;processor specific variable definitions
; ------------------------------------------------------------------------------------


    MyReg equ 0x001

    cblock 0x7D
    w_temp, status_temp, bsr_temp
    endc

    ORG 0x000000
    bra Main

    ORG 0x000008
    btfsc INTCON, INT0IF; check INT0 interrupt flag
    bra INT0_ISR
    btfsc INTCON3, INT1IF; check INT1 interrupt flag
    bra INT1_ISR
    retfie 1
; Check which interrupt flag is raised, branch to the
; appropriate ISR.
    ORG 0x000018
    bra T0_ISR
    ORG 0x000100
Main:
    ;Configure input/output
    setf TRISC
    clrf TRISD
    clrf PORTD
    bsf TRISB, RB0
    bsf TRISB, RB1
    bcf TRISB, RB6
    bcf TRISB, RB7

    ;Configure Timer0
    movlw 0x02; Timer0, 16-bit, prescale value = 8, internal clock
    movwf T0CON
    movlw 0x0B; refer to Part 9a)
    movwf TMR0H
    movlw 0xDC
    movwf TMR0L

    ; Interrupt bits
    bsf RCON, IPEN; enable priority interrupt
    bcf INTCON, INT0IF; clear INT0 interrupt flag
    bcf INTCON3, INT1IF; clear INT1 interrupt flag
    bcf INTCON, TMR0IF; clear Timer0 interrupt flag
    bsf INTCON, INT0IE; enable INT0 interrupt
    bsf INTCON3, INT1IE; enable INT1 interrupt
    bsf INTCON, TMR0IE; enable Tim0er0 interrupt
    bsf INTCON3, INT1IP; set INT1 as high-priority interrupt
    bcf INTCON2, TMR0IP; set Timer0 as low-priority interrupt
    bsf INTCON, GIEH; enable global high-priority interrupt
    bsf INTCON, GIEL; enable global low-priority interrupt

    ; Optional because these are the default on reset.
    bsf INTCON2, INTEDG0; INT0 interrupt on rising edge
    bsf INTCON2, INTEDG1; INT1 interrupt on rising edge

    bsf T0CON, TMR0ON; start Timer0
Over:                 movff PORTC, MyReg
    bra Over

T0_ISR:   btfss INTCON, TMR0IF
    bra EXIT_T0
    movwf w_temp ; preserve context
    movff STATUS, status_temp
    movff BSR, bsr_temp
    bcf INTCON, TMR0IF
    incf PORTD
    movlw 0x0B
    movwf TMR0H
    movlw 0xDC
    movwf TMR0L
    movff bsr_temp, BSR ; restore context
    movf w_temp, w
    movff status_temp, STATUS
EXIT_T0:  retfie

INT0_ISR: bcf INTCON, INT0IF
    btg PORTB, RB6
    retfie 1

INT1_ISR: bcf INTCON3, INT1IF
    btg PORTB, RB7
    retfie 1
    END
```