EECE.3170: Microprocessor Systems Design I

Spring 2016

Exam 3
May 7, 2016

Name: _____

Section (circle 1): **201** *(MWF 9-9:50)* **202** *(MWF 10-10:50)*

For this exam, you may use a calculator and one 8.5" x 11" double-sided page of notes. All other electronic devices (e.g., cellular phones, laptops, PDAs) are prohibited. If you have a cellular phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 4 questions for a total of 100 points. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

Please note that Question 4 has three parts, but you are only required to complete two of the three parts. You may complete all three parts for up to 10 points of extra credit. If you do so, **please clearly indicate which part is the extra one—I will assume it is part (c) if you mark none of them.**

You will be provided with two pages (1 double-sided sheet) of reference material for the exam that contain the PIC16F1829 instruction set. You do not have to submit this sheet when you turn in your exam.

You will have three hours to complete this exam.

Q1: Multiple choice	/ 20
Q2: General microcontroller programming	/ 15
Q3: PIC C programming	/ 15
Q4: PIC assembly programming	/ 50
TOTAL SCORE	/ 100
EXTRA CREDIT	/ 10

1. (20 points, 5 points per part) ***Multiple choice***

For each of the multiple choice questions below, clearly indicate your response by circling or underlining the single choice you think best answers the question.

a. You are running a PIC16F1829 program that uses two I/O ports, Port A and Port C. If TRISA = 0x0F and TRISC = 0x19, how many I/O pins, in total, are configured as inputs?

i. 4

ii. 7

iii. 9

iv. 15

v. 40

b. Under what conditions will the following code jump to the label L1?

```
   movf    x, W
   subwf   y, W
   btfsc   STATUS, Z
   btfss   STATUS, C
   goto    L1
END:
```

i. $x \neq y$

ii. $x = y$ or $C = 1$

iii. $x = y$ or $C = 0$

iv. $x \neq y$ or $C = 1$

v. $x \neq y$ or $C = 0$

1 (continued)

c. You are given the following short PIC16F1829 assembly function:

```
F: movf   PORTC, W
   andlw  B'00000001'
   addwf  PCL, F
   retlw  B'11110000'
   retlw  B'00111100'
   retlw  B'00001111'
   retlw  B'11111111'
```

Which of the following PORTC values will cause this function to return `B'00001111'`?

i. `PORTC = 0xF0`

ii. `PORTC = 0x19`

iii. `PORTC = 0x86`

iv. `PORTC = 0x3F`

v. None of the above

d. Circle one (or more) of the choices below that you feel best "answers" this "question."

i. "Thanks for the free points."

ii. "I don't REALLY have to answer the last three questions, do I?"

iii. "This exam is the best final I've taken all day."

iv. None of the above.

2. (15 points) ***General microcontroller programming***

a. (4 points) You are given the following delay loop:

```
Ten_1
   decfsz    COUNTL,F       ; Inner loop
   goto      Ten_1
   decfsz    COUNTH,F       ; Outer loop
   goto      Ten_1
   return
```

If COUNTL is initially 100, COUNTH is initially 10, the clock frequency is 500 kHz, and each instruction takes 4 clock cycles, how long does the whole delay loop take? **Show your work for full credit.**

b. (3 points) In the blinking LED state machine program we covered in class, the following instructions isolate the lowest three bits of PORTD and copy them into W:

```
movf   PORTD, W
andlw  B'00000111'
```

Why might it be necessary to isolate the lowest three bits and mask out the upper five bits of PORTD?

2 (continued)

c. (4 points) Say your program uses a 10-bit analog-to-digital converter (ADC) with positive/negative reference voltages of 5 V and 0 V, respectively. If the ADC output is 256, approximately what voltage was input to the converter? **Show your work for full credit.**

d. (4 points) Say you are writing a program using two interrupts. One interrupt is triggered when a timer overflows, while the other is triggered when a switch is pressed. Describe a situation in which you would want to prioritize the timer interrupt (and therefore handle the timer interrupt first if both types of interrupt occur simultaneously), and describe a situation in which you would want to prioritize the switch interrupt.

3. (15 points) ***PIC C programming***

Complete the short function below by writing the appropriate line(s) of C code into each of the blank spaces. The purpose of each line is described in a comment to the right of the blank.

This interrupt service routine detects both switch and timer interrupts and works with the analog-to-digital converter (ADC). The ISR does the following:

- On a timer interrupt, start an analog-to-digital conversion.
- On a switch interrupt, check if the ADC is done. If the ADC is done, assign the lowest four bits of the result to the LEDs, making sure to only change the lowest four bits of Port C while leaving the upper four bits the same.

Assume the LEDs are wired to the lowest four bits of Port C (as on the board used in HW 8) the ADC result is right-justified, and that "SWITCH" and "DOWN" are appropriately defined.

```
void interrupt ISR(void) {

   if (_____________________________) {     // SW1 was pressed
      IOCAF = 0;                            // Clear flag in software
      __delay_ms(5);                        // Delay for debouncing
      if (SWITCH == DOWN) {                 // If switch still pressed

         if (_____________________) {       //   check if ADC is done
                                            // If done, assign lowest 4
                                            //   result bits to LEDs,
                                            //   making sure to only
                                            //   change correct bits




         }
      }
   }
   if (INTCONbits.T0IF) {                   // Timer 0 interrupt

      _________________________________     // Clear flag in software

      _________________________________     // Start ADC
   }
}
```

4. (50 points, 25 points per part) ***PIC assembly programming***

For each of the following complex operations, write a sequence of PIC 16F1829 instructions—**not C code**—that performs an equivalent operation. **CHOOSE ANY TWO OF THE THREE PARTS** and fill in the space provided with appropriate code. **You may complete all three parts for up to 10 points of extra credit, but must clearly indicate which part is the extra one—I will assume it is part (c) if you mark none of them.**

Assume that 8-bit variables "TEMP" and "COUNT" have been defined for cases where you may need extra variables.

Finally, please note that you are not required to write comments describing each instruction. You may certainly do so if you feel comments will make your solution clearer to the instructor.

a. You are given two 8-bit variables, P and Q. Write a sequence of instructions that takes the bit sequence in P and stores its reverse in Q. For example:

   - If P = 0xF0 = 11110000_2, Q = 00001111_2 = 0x0F
   - If P = 0x15 = 00010101_2, Q = 10101000_2 = 0xA8

Your solution should not change P.

4 (continued)

Remember, you can assume that 8-bit variables "TEMP" and "COUNT" have been defined for cases where you may need extra variables.

b. You have a 16-bit variable, X, and an 8-bit variable, P. You can access individual bytes within X—the low byte, XL, holds bit positions 0 to 7, and the high byte, XH, holds bit positions 8 to 15.

   Write a sequence of instructions that sets P equal to the highest bit position within X that contains a 1. (This operation is similar to the x86 BSR instruction). If X = 0, P should be unchanged. For example:

   - If X = 0x61AB = 0<u>1</u>10 0001 1010 1011_2, P = 14 (underlined bit is "highest" 1)
   - If X = 0x0082 = 0000 0000 <u>1</u>000 0010_2, P = 7

   Your solution should not change XL or XH.

4 (continued)

Remember, you can assume that 8-bit variables "TEMP" and "COUNT" have been defined for cases where you may need extra variables.

c. You are given two unsigned 16-bit values, X and Y. You can access individual bytes within each value—"X" contains bytes XH and XL (XL is the least-significant byte) and "Y" contains bytes YH and YL.

   Write a sequence of instructions that jumps to location "L1" if X is greater than or equal to Y. Your solution should not change any of the bytes of X or Y. Again, assume X and Y are unsigned (i.e., non-negative).