16.317: Microprocessor Systems Design I

Spring 2015

Exam 3

May 4, 2015

Name: ____

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	/ 12
Q3: PIC C programming	/ 18
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. We discussed the following delay loop in class:

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

How many times will the first instruction in this loop (`decfsz COUNTL,F`) execute if COUNTH is initially equal to 0x02 and COUNTL is initially equal to 0x01? (All answers given below are in base 10, unless otherwise noted.)

i. 1

ii. 101

iii. 201

iv. 257 (which is 0x101 in hexadecimal)

v. 258

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

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

i. $x \neq y$

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

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

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

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

1 (continued)

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

```
F: movf    PORTC, W
   andlw   B'00000010'
   addwf   PCL, F
   retlw   B'11110000'
   retlw   B'00111100'
   retlw   B'00001111'
   retlw   B'11111111'
```

If `PORTC = 0xC3`, what value is in the working register when this function returns?

i. `B'00000001'`

ii. `B'00001111'`

iii. `B'00111100'`

iv. `B'11110000'`

v. `B'11111111'`

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. (12 points) ***General microcontroller programming***

a. (3 points) Explain how an interrupt service routine can prioritize one interrupt over another.

b. (3 points) What are the different factors that determine the total amount of delay in an instruction count-based delay loop?

c. (3 points) Given a 10-bit result from an analog to digital converter, as in the PIC 16F1829, explain a case in which you might prefer a left-justified result (upper 8 bits) over a right-justified result (lower 8 bits).

d. (3 points) Explain why button presses cannot accurately be detected by simply checking if the switch produces a low voltage.

3. (18 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 works with the analog-to-digital converter, as well as a global variable, `var`, which determines what is written to the LEDs. The ISR does the following:

- On a switch interrupt, start an analog-to-digital conversion.
- On a timer interrupt, evaluate the value in ADRESH and change the LEDs as follows:
    - If the value in ADRESH is in the upper half of possible values (which you can test by checking if any of the top 4 bits are set), increment the value on the LEDs. You must account for the rollover case when the LEDs go from 1111 to 0000.
    - Otherwise, decrement the value on the LEDs, accounting for the 0000 to 1111 case.

Assume the LEDs are wired to the lowest four bits of Port C, as on the development board used in HW 6, and that "SWITCH" and "DOWN" are appropriately defined.

```
void interrupt ISR(void) {
    if (IOCAF) {                          // SW1 was pressed

        ___________________________       // Clear flag in software
        __delay_ms(5);                    // Delay for debouncing
        if (SWITCH == DOWN) {             // If switch still pressed

          ___________________________     //   start ADC

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

        ___________________________       // Clear flag in software

        if (___________________) {        // If A/D result is in
                                          //  upper half of possible
                                          //  results (any of upper
                                          //  four bits are set)
                                          //  increment LED value
                                          //  Must account for
                                          //    1111 → 0000 case
        }
        else {                            // Otherwise, decrement
                                          //  LED value. Must
                                          //  account for
                                          //     0000 → 1111 case


        }
    }
}
```

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 are certainly welcome to do so if you feel it will make your solution clearer to the instructor.

a. 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 less than 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).

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. Given two 8-bit variables, X and N, flip the lowest N bits of X (change 0 → 1, 1 → 0) while leaving the other bits unchanged. For example:

- If X = 0x00 and N = 0x02, flip the lowest two bits—bits 0 and 1.
  - X = 0x00 = $0000\ 0000_2$ originally → X will change to $0000\ 00\underline{11}_2$ = 0x03
- If X = 0x0C and N = 0x06, flip the lowest six bits—bits 0 through 5.
  - X = 0x0C = $00\underline{00\ 1100}_2$ originally → X will change to $00\underline{11\ 0011}_2$ = 0x33

Note that:

- Since X and N are not constants, you cannot use both values together in any PIC instruction (for example, `btfsc X, N` is $\underline{\text{not}}$ a valid instruction).
- Your code should not modify N.

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 16-bit values, X and Y, and an 8 bit value, ZL. 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.

Perform a 16-bit left rotate without carry: X = Y rotated by ZL. (Note that, because the rotate amount is not greater than 15, a single byte is sufficient to hold that value.) Do not change Y or ZL when performing this operation.

At each step, note that the bit shifted out of the most significant bit should be shifted into the least significant bit. For example, if Y = 0xFF00 = $1111\ 1111\ 0000\ 0000_2$ and ZL = 1, then the final result should have X = $1111\ 1110\ 0000\ 0001_2$.