

(Problem 2) Given that Fosc = 1 MHz, calculate the delay generated by the following subroutine code, show your work. (4 / 4 points)

```

SUB1
    MOVLW D'240'
    MOVWF D1
    MOVLW 1
    MOVWF D2
    MOVWF D3

L0
    DECFSZ D1, F
    GOTO L1
    DECF D2, F
    GOTO L2
    DECF D3, F
    GOTO L0
    RETURN
    
```

Handwritten calculations for Problem 2:

2 + (1 + 1 + 1 + 1 + 1) + (1 + 2 + 2 + 2) \* 2 = 2 + 5 + 12 = 19 instructions

Annotations: "call of sub", "sub 1", "D1", "D2", "D3", "return", "D1", "D2", "D3", "return", "D1", "D2", "D3", "return".

2 + 5 + 1743 + 8 = 1758 #

Delay = 1758 \* 4 / 1000000 = 7.032 \* 10^-3 = 7.032 ms

(Problem 3) Given the code below, answer the following questions: (7 points)

```

1) STATUS EQU 3
2) Z EQU 2
3) COUNTER EQU 0x22
4) ORG 0x05
5) MOVLW 0x03
6) MOVWF COUNTER
7) B EQU 3
8) A EQU 0
9) LOOP ADDWF COUNTER, A
    DECF COUNTER, B
    BTFSS STATUS, Z
    GOTO LOOP
    DECF COUNTER, B
    END
    
```

A) What is the size of this program in Bytes? (1 point)

14 \* 14 / 8 = 24.5 Bytes

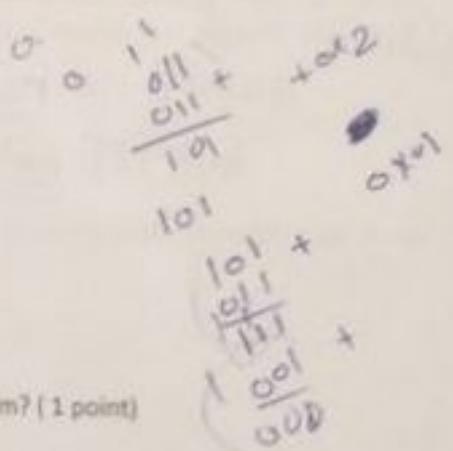
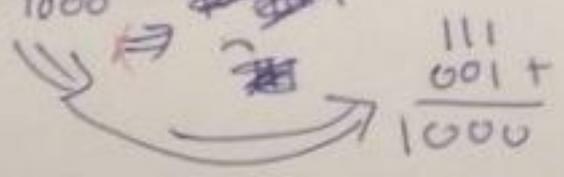
B) How many instruction cycles does it take to execute the program? (1 point)

# 7

C) What are the final values of location 0x22 and W-register? (2 points)

I. [0x22] = 0x00

II. [W] = 1000



Name: [Redacted]

Student ID: [Redacted]

(Problem 1) ( 3 /9)

a) List the configurations required to keep the PIC in reset mode on power up for longest period of time, and without using external circuits. Do not calculate the delay. (2 points)

- \* the pcounter is set to zero
- \* the SFRs are set such that peripherals are in safe and disabled
- \* applying logic zero in master clear input (MCLR)

b) Replace the following code by one instruction which should have the same effect (1 point)

```

MOVLW 0xFF
XORWF 0x32, 0

```

*Handwritten:* XORLW B'11111111', 0x32

c) Replace the following code by one instruction which should have the same effect (2 point)

```

MOVF STATUS, 0
MOVWF 0x23
MOVF 0x23, 0
ADDWF 0x22, 1
BTFSF 0x23, 0
INCF 0x22, 1
MOVF 0x22, 0

```

*Handwritten:* MOVF 0x22, 0

d) Is the following statement true or false and why? (2 point)

False  
 Harvard architecture computers are usually faster and simpler:  
 Harvard is faster because it use pipelining  
 but it's not simpler (it has different sizes of busses and more than 1 bus)

e) Write a piece of code that checks data memory location 0x25 and increments it if it is odd. (2 points)

```

btfsc 0x25, 0
incf 0x25, 1
goto done
end

```



- d) In a PIC16F84A, given that there are 5 LEDs connected to portA such that: the LEDs connected to RA0-RA2 are in current sink mode, while the LEDs connected to RA3-RA4 are connected in current source mode. Write below only two instructions which are required to make the five LEDs to be on. Assume that PORT A is already configured as output. Explain briefly the answer. (2 points)

Problem 3) ( 3 / 7 points)

- a) In the boxes below, write the required code which is required to complement memory location: 0x195 using both direct addressing and indirect addressing modes. (2 points)

Using direct addressing

```

BCF PC1ATH, 4
BCF PC1ATH, 3
ORG 0x195
COMF TEMP, F
    
```

Using indirect addressing

```

BCF PC1ATH, 4
BCF PC1ATH, 3
MOVLW 0x195
MOVWF FSR
COMF INDF, F
    
```

0x195  
= 0001 1001 0101  
byte 0

- b) In the box below, write the required code to make the code execution goes to the instruction in location 0x15AA (i.e. goto 0x15AA). (1 point)

```

BCF PC1ATH, 4
BCF PC1ATH, 3
goto 0x15AA
    
```

0x15AA  
= 0001 0101 1010 1010  
page

- c) Assume microcontroller A wants to send one byte to microcontroller B serially:

Write a piece of code that will make A sends the data in 9 bits such that the ninth bit is 0. (2 points)

```

BCF TRISC, 0
BCF TRISC, 7
MOVLW 0b10010000
MOVWF SPBRG
BSF TXSTA, TXEN
BSF TXSTA, TX9D
    
```

```

MOVLW B'10010000' ; TX9 = 0
MOVWF RCSTA
MOVWF TXREG
BSF TXSTA, TXSTA
wait BTFSS TXSTA, TRMT
go to wait
    
```

Assume B detected OERR, write the code to solve the problem and starts receiving again. (2 points)

```

BCF RCSTA, CREN
BCF RCSTA, OERR
BTFSS PIR1, RCIF
go to wait
go to save
MOVWF RCOREG, W
    
```

```

MOVWF INDF
INCF FSR, F
go to wait
    
```

wait

save

1

30/4/2014

Embedded Systems  
Computer Engineering DepartmentSecond exam  
70 minutes

Name: [REDACTED]

Student ID: [REDACTED]

Section: (ح 5) 5

**Problem 1)** Given that the following code continuously adds the content of memory location 0x0A until an external interrupt is observed on RB0. In this case the result is stored in location 0x10 and the W-register is cleared. (8.5 / 11 points)

```
#INCLUDE P16F84A.INC
ORG 0X0000
GOTO START
ORG 0X0004
GOTO ISR
START BSF STATUS, RPO
      BSF INTCON, INTE
      BSF OPTION_REG, 6
      BSF INTCON, GIE
      BCF STATUS, RPO
ADD   ADDWF 0X0A, 0
      GOTO  ADD
ISR   MOVWF 0X10
      CLRW
      BCF INTCON, INTF
      RETFIE
      END
```

- a) Modify the code such that the interrupt happens when a **falling edge** arrives on RB0. Do the modification beside the same code. (1 point) **BCF OPTION\_REG, 6**
- b) What are the values to be stored in the option register and in timer0 register which are required to make timer0 overflows after 25.6ms given that  $F_{osc} = 1$  MHz. Initialize any non-required bit to zero. Show your work. (3 points)

$$\text{delay} = (256 - \text{TMR0}) \times \frac{4}{F_{osc}} \times \text{Pre-scaler}$$

$$\Rightarrow 25.6 \times 10^{-3} = \frac{100}{4 \times 10^6} \times Y$$

$$\Rightarrow Y = 64 \Rightarrow 256 - 100 = 156 = \text{TMR0}$$

$$\Rightarrow \text{pre-scaler} = 64 = 101$$

$$256 - \text{TMR0} = 100 \Rightarrow \text{TMR0} = 156$$

Timer 0 = D'156'

Option = 0XD5

5 = 101 = 64 scale

- c) In box1 below, modify the code above such that when either timer0 interrupt or external interrupt on RB0 observed, the result is stored in location 0x10 and the working register is cleared. For this part, use the values to be stored in option register and in timer0 register according to what has been calculated in b above. (3 points) **اذا فله الارتفاع**
- d) In box2 below, modify only the ISR from the above code such that every 5 rising edges arrive on RB0, the result is stored in location 0x10 and the working register is cleared. Register COUNTER is initialized to zero if you need it. (4 points)

② Box (1)

\*include P16F84A.INC

org 0x00

goto start

org 0x04

goto ISR

start

bsf status, RPO

bsf INTCN, INTE

bsf ~~INTCON~~<sup>INTCON</sup>, GIE

bcf INTCN, TOIF

bsf INTCN, TOIE

BankSel option\_reg

MOVLW 0xDB

MOVWF option\_reg.

BankSel TRR0

MOVLW D'156'

MOVWF TRR0

Bcf Status, RPO

ADD ADDWF 0x0A,0  
goto ADD

ISR

MOVWF 0x10

CLRWF

Bcf INTCN, INTF

Bcf INTCN, TOIF

~~RETFIE~~ RETFIE

END