CS224
Section No.: 2
Spring 2019
Lab No.7
Zeynep Cankara/21703381

Answer for Question 1:

```
# Zeynep Cankara -- 15/05/2019
# Course: CS224
# Section: 2
# Lab: 7
# Version: 1.0
# lab7.asm -- Description: Introduction to PIC32 Programming

          .text
     .globl    __start

__start:
     # // Program No 1
     # char junk;
     # SPI2ACONbits.ON = 0;
     initspi:
          # // load the address of SPI2ACON -> $t0
          # // SPI2ACON has virtual address 0xBF80
          # // Access the SPI2ACONbits.ON -> 16th bit -> 5A00
          lui $t0, 0xBF80
          ori $t0, 0x5A00
          # // Get the value of SPI2ACON -> $t1
          lw $t1, 0($t0)
          # // AND mask to clear bit 15
          andi $t1, $t1, 0x7FFF
          # // Store the value back in SPI2ACON
          sw $t1, 0($t0)
          # junk = SPI2ABUF;
          # // Read the value in SPI2ABUF (5A20) -> $t1
          lw $t1, 0x20($t0)
          # SPI2ABRG = 7;
          # // Store value 7 in SPI2ABRG (5A30)
          li $t2, 7
          sw $t2, 0x30($t0)
          # SPI2CONbits.MSTEN = 1;
          # // Enable bit 5 (MSTEN) of SPI2ACON
          lw $t2, 0($t0)
          # // OR mask to enable bit 5 (6th bit)
          ori $t2, $t2, 0x0020
          # // Store the value back in SPI2ACON
          sw $t2, 0($t0)
          # SPI2CONbits.CKE = 1;
          lw $t2, 0($t0)
          # // Apply OR mask to enable bit 8 (CKE) of SPI2ACON
          ori $t2, $t2, 0x0100
```

```
          # // Store the value back in SPI2ACON
          sw $t2, 0($t0)
          # while(!SPI2CONbits.SSEN);
     while:
          # // load the value SPI2ACON -> $t2
          lw $t2, 0($t0)
          # // Mask all bits except bit 7 (SSEN) of SPI2ACON
          # // Masked value -> $t3
          andi $t3, $t2, 0x0080
          # // Spin in the loop while loop untill SSEN = 1
          beq $t3, $0, while
          # SPI2CONbits.ON = 1;
          # // Value of SPI2CON -> $t3
          lw $t3, 0($t0)
          # // Apply OR mask to enable bit 15 (ON) of SPI2ACON
          ori $t3, $t3, 0x8000
          # // Save value back in SPI2ACON
          sw $t3, 0($t0)
     # exit the Programme 1
     li $v0, 10
     syscall
```

Answer for Question 2:

```
# Zeynep Cankara -- 15/05/2019
# Course: CS224
# Section: 2
# Lab: 7
# Version: 1.0
# lab7q2.asm -- Description: Introduction to PIC32 Programming

# ========== TEXT SECTION ============ ###
          .text
     .globl    __start

__start:
     # // Program No 2
     # int stop = 0;
     # // stop -> $t1
     li $t1, 0b0
     # int initial  = 0b01110111;
     # // initial -> $t0
     li $t0, 0b01110111
     # int right = 1;
     # // right -> $t2
     li $t2, 0b0
     main:
          # TRISD = 0x0;
          # // Virtual address of TRISD 0xBF88 60C0
          # // TRISD's address -> $t3
          lui $t3, 0xBF88
          ori $t3, 0x60C0
```

```
          # // Set the bits of TRISD 0 (output)
          sw $t1, 0($t3)
          # // Three bits of PORTA are inputs
          # TRISA = 0b111;
          li $t4, 0b111 # 0b111 -> $t4
          # // Address of TRISA 0xBF88 6000
          andi $t3, $t3, 0x0000 # // AND mask to clear lower 16
bits
          ori $t3, $t3, 0x6000 # // Address of TRISA -> $t3
          sw $t4, 0($t3) # // Store the value 0b111 in TRISA
          # PORTD = initial;
          # // Address of PORTD 0xBF88 60D0
          andi $t3, $t3, 0x0000 # // AND mask to clear lower 16
bits
          ori $t3, $t3, 0x60D0 # // Address of PORTD -> $t3
          sw $t0, 0($t3) # // Store initial in PORTD
          # while(1) {
          while1:
               # // continue until 0
               bne $0, $0, end
               # int lsb;
               # int mask;
               # if(PORTABits.RA1 == 0) {
               # // Address PORTA : 0xBF88 6010
               andi $t3, $t3, 0x0000 # // AND mask to clear lower
16 bits
              ori $t3, $t3, 0x60D0 # // Address of PORTA -> $t3
              # // load value into the $t4
              lw $t4, 0($t3)
              # // mask all bits except bit 1 to access
PORTABits.RA1
              # // Store masked value ->  $t5
              andi $t5, $t4, 0x0020
              # // Check if(PORTABits.RA1 == 0) condition
               bne $t5, $0, ifStop
                    # stop = !stop;
                    # // change value of STOP -> $t1
                    beq $t1, $0, change1:
                         li $t1, 0
                         j endChange
                    change1:
                         li $t1, 1
                    endChange:
                    # if(!stop){
                    ifStop:
                         bne $t1, $0, else # // continue if stop
is not 0
                         # // Get the PORTD's address
                         andi $t3, $t3, 0x0000 # // AND mask to
clear lower 16 bits
                         ori $t3, $t3, 0x60D0 # // Address of
PORTD -> $t3
```

```
                    # PORTD = initial;
                    sw $t0, 0($t3) # // Store initial in
PORTD
               else:
          # // check value stop
          ifStop2:
          # if(!stop){
               bne $t1, $0, else2 # // continue if stop is
not 0
               # // lsb -> $t6
               # // get PORTDs value inside $t6
               lw $t6, 0($t3)
               # lsb = PORTD & 0x1;
               andi $t6, $t6, 0x1
               # // mask -> $t7
               # mask = lsb << 7;
               addi $t7, $t6, 0
               sll  $t7, $t7, 7
               # PORTD = (PORTD >> 1) | mask;
               # // get PORTDs value inside $t5
               lw $t5, 0($t3)
               # // right shift
               srl $t5, $t5, 1
               # apply OR mask
               or $t5, $t5, $t7
               # store value back in PORTD
               sw $t5, 0($t3)
               j endElse2
          # else {
          else2:
               # PORTD = 0b11111111;
               li $t5, 0b11111111
               # // $t3 contains address of PORTD
               sw $t5, 0($t3)
          endElse2:
          # delay_ms(1000);
          li $v0, 32
          li $a0, 1000
          syscall # // one second waiting
          j while1
     end:
# // exit the Programme 2
li $v0, 10
syscall
```