EEE 407 MICROPROCESSOR LABORATORY

EXPERIMENTAL WORK 2

BRANCH, CALL and TIME DELAY LOOP

Objective: Learn how to implement loops and conditional branches in PIC18 assembly, and how to create simple time delays using loops and subroutines. This experiment introduces instructions for flow control: unconditional branch (goto), subroutine calls (call/return), and conditional branch instructions like decfsz (decrement and skip if zero) and bnz (branch if not zero). We will use these to repeat operations and generate delays. We will also compute how long a loop-based delay lasts, reinforcing understanding of instruction execution time.

Ex.1: Write the following Assembly language program and observe the content of 25H, 26H and File Registers by using MPLAB Simulator.

```
        list        p=18f452

        #include    p18f452.inc

COUNT equ           0x25        ; use file register 0x25 as a loop counter

SUM   equ           0x26        ; use file register 0x26 to store the final sum

        org         0           ; Reset Vector

        goto        Start


        org         0x20        ; beginning of program in memory


Start

        movlw       d'10'       ; Load WREG with 10 – this will be our loop count

        movwf       COUNT       ; Store WREG (10) into COUNT at address 0x25

        movlw       0x00        ; Load WREG with 0 – this will be the initial sum

again   addlw       d'3'        ;  Add 3 to WREG

        decfsz      COUNT,f     ; Decrement COUNT. If COUNT not yet 0, execute

                                ; next line; if it became 0, skip next line.

        goto        again       ; repeat until counter becomes 0

        movwf       SUM         ; Move final WREG value into SUM register

        end
```

Ex.2: Modifiy the previous Assembly program to use "BNZ" instruction and repeat the procedure in Ex.1

```
        list        p=18f452

        #include    p18f452.inc

COUNT equ           0x25        ; loop counter at 0x25

SUM   equ           0x26        ; final result at 0x26


        org         0x00        ; Reset Vector

        goto        Start


        org         0x0020      ; Begin program


Start

        movlw       d'10'       ; WREG=10

        movwf       COUNT       ; load the counter

        movlw       0x00        ; WREG = 0 (start sum at 0)

        movwf       SUM         ; SUM = 0

BACK  addlw         d'3'        ; add 3 to WREG

        decf        COUNT,f     ; Decrement COUNT by 1 (this affects Z flag: if
                                ; COUNT becomes 0, Z=1)

        bnz         BACK        ; Branch to 'Back' if Zero flag = 0 (i.e., if
                                ; COUNT is not zero, keep looping)

        movwf       SUM         ; When loop ends, move WREG (final sum) into SUM

        end
```

Ex.3: Write the following Assembly language program and observe the content of 20H, File Registers and Program Memory by using MPLAB Simulator. Calculate the "delay time" created in the delay subroutine.

```
        list        p=18f452

        #include    p18f452.inc

        MYREG equ        0x21  ; use file register 0x21 as loop counter


        org         0x20        ;starting point of main program


BACK    movlw       0X55  ; Load WREG with 0x55 (pattern 0101 0101 in binary)
        movwf       20H   ; Write 0x55 to file register at 0x20
        CALL        DELAY ; Call the delay subroutine
        movlw       0XAA  ; Load WREG with 0xAA (pattern 1010 1010, the inverse pattern)
        movwf       20H   ; Write 0xAA to address 0x20
        CALL        DELAY ; Call the delay again
        goto        BACK  ; Repeat forever


;-----------------This is the delay subroutine-------------------


        org         300H    ;  place the delay subroutine at address 0x300 in program memory

DELAY                                 ;   instruction cycle
        movlw       D'255'            ;           1
        movwf       MYREG             ;           1
AGAIN
        NOP                           ;           1
        NOP                           ;           1
        NOP                           ;           1
        decf        MYREG, f          ;           1
        bnz         AGAIN             ;           2
        return                        ;           2
        end
```