EEE 407 MICROPROCESSOR LABORATORY

EXPERIMENTAL WORK 1

INTRODUCTION TO PIC ARCHITECTURE AND ASSEMBLY LANGUAGE PROGRAMMING

Objective: In this experiment, registers and architecture of PIC18F452 microcontroller are represented. Furthermore, basic programming techniques are applied on MPLAB Simulator Environment to be able to observe the content of working and file registers.

Ex.1: In this first example, we will load a literal value into WREG and then copy that value into several file registers. Then, a subroutine will clear those registers. The program will loop indefinitely, allowing us to observe the changes continuously by using MPLAB simulator.

```
      list        p=18f452    ; Specify PIC18F452 processor
      #include    p18f452.inc ; Include device-specific definitions
DEST  equ         0x26        ; Define a constant label for memory address 0x26
      org         0x00        ; Reset Vector
      goto        Start       ; Jump to start of program (skip interrupt vectors)

      org         0x20        ; Main program start address (in code memory)

Start
      movlw       19h         ; Load literal 0x19 into WREG (WREG = 0x19)
      movwf       21h         ; Move WREG content into file register 0x21
      movwf       22h         ; Move WREG content into file register 0x22
      movwf       23h         ; Move WREG content into file register 0x23
      movwf       24h         ; Move WREG content into file register 0x24
      movwf       25h         ; Move WREG content into file register 0x25
      movwf       DEST        ; Move WREG content into file register 0x26
      call        ClearMemory ; Call subroutine to clear registers 0x21-0x25
      goto        Start       ; Loop back to Start label (repeat forever)

ClearMemory
      clrf         21h         ; Clear file register 0x21 (set to 0x00)
```

```
        clrf         22h        ; Clear file register 0x22
        clrf         23h        ; Clear file register 0x23
        clrf         24h        ; Clear file register 0x24
        clrf         25h        ; Clear file register 0x25
        return                  ; Return from subroutine to the caller
        end
```

Ex.2: In this example, we practice arithmetic instructions using WREG and observe how results are stored and how the Status flags might be affected. We add several constants to WREG and store a final sum in a file register. We also use the addwf and subwf instructions to perform register-to-register addition and subtraction, and examine the effect on WREG and the target register.

```
        list        p=18f452
        #include    p18f452.inc
SUM     equ         0x0B       ; Define a label for memory address 0x0B (will hold a sum)
SIX     equ         B'00000110' ; Define constant 6 in binary form (0b00000110)

        org         0x00        ; Reset Vector
        goto        START

        org         0x20        ; Begin program execution here
START
        movlw       25h         ; WREG = 0x25
        addlw       0x34        ; WREG = WREG + 0x34 -> (0x25 + 0x34 = 0x59)
        addlw       11H         ; WREG = WREG + 0x11 -> (0x59 + 0x11 = 0x6A)
        addlw       D'12'       ; WREG = WREG + 12 (dec)->(0x6A + 0x0C =0x76)
        addlw       SIX         ; WREG = WREG + 6 -> (0x76 + 0x06 = 0x7C)
        movwf       SUM         ; Store WREG (0x7C) into memory address 0x0B (SUM)
        addwf       SUM, F      ; Add WREG to SUM, result stored in SUM :
                                ;       - Before: WREG = 0x7C, SUM = 0x7C
                                ;       - After: SUM = SUM + WREG = 0x7C + 0x7C = 0xF8
```

```
                                    ; (WREG remains 0x7C)
        subwf       SUM, F      ; Subtract WREG from SUM, result stored in SUM:
                                ;       - Before: WREG = 0x7C, SUM = 0xF8
                                ;       - Operation: SUM = SUM - WREG = 0xF8 - 0x7C =0x7C
                                ;       - After: SUM back to 0x7C, WREG still 0x7C
        subwf       SUM, W      ; SUM (0x0B) = 82H, WREG = 0
        clrf        SUM         ; Clear SUM (set memory 0x0B to 0x00)
HERE    goto        HERE        ; stay here forever
        end                     ; end of asm source file
```