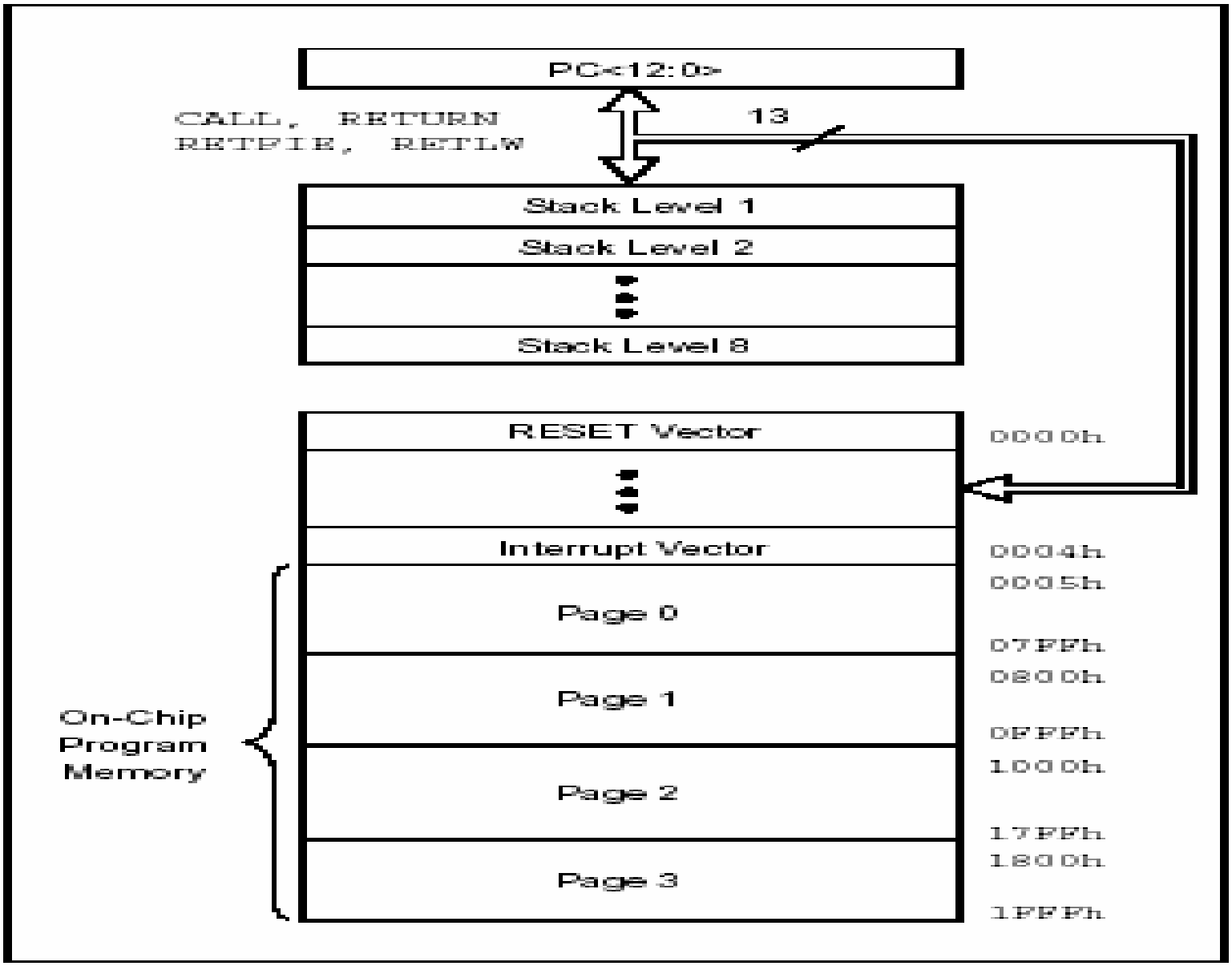
Microcontroller Programming

Dr. Habib-ur Rehman
College of Engineering
UAE University

Registers Memory

- Basic Registers
- Banks Selection
- Variable Registers definition

PIC16F877/876 Program Memory Map & Stack



PIC16F877 / 876 Register File Map

Indirect addr	File Address
PCL	02h
STATUS	03h
FSR	04h
PCLATH	0Ah
INTCON	0Bh

Indirect addr	File Address
OPTION_REG	81h
PCL	82h
STATUS	83h
FSR	84h
PCLATH	8Ah
INTCON	8Bh

Indirect addr	File Address
TMR0	101h
PCL	102h
STATUS	103h
FSR	104h
PCLATH	10Ah
INTCON	10Bh

Indirect addr	File Address
OPTION_REG	181h
PCL	182h
STATUS	183h
FSR	184h
PCLATH	18Ah
INTCON	18Bh

Data Memory Organization

- Bits RP1(STATUS<6>) and RP0(STATUS<6>) are the Bank Select bits

RP1:RP0	Bank
00	0
01	1
10	2
11	3

INSTRUCTION SET

- Introduction to instruction set

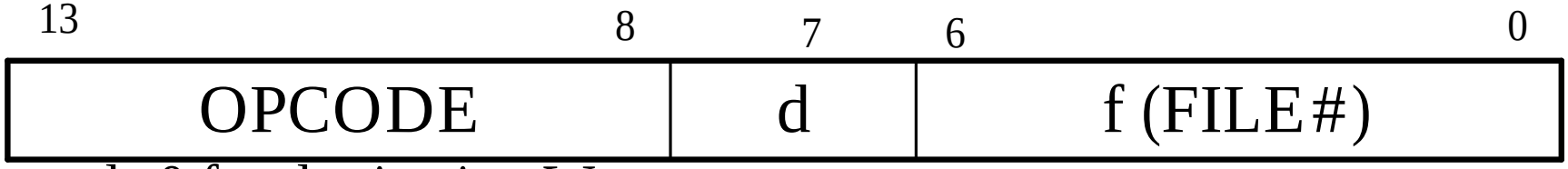
- Instructions examples & Addressing Modes

OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
w	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
PC	Program Counter
TO	Time-out bit
PD	Power-down bit

GENERAL FORMAT FOR INSTRUCTIONS

- Byte-oriented file register operations

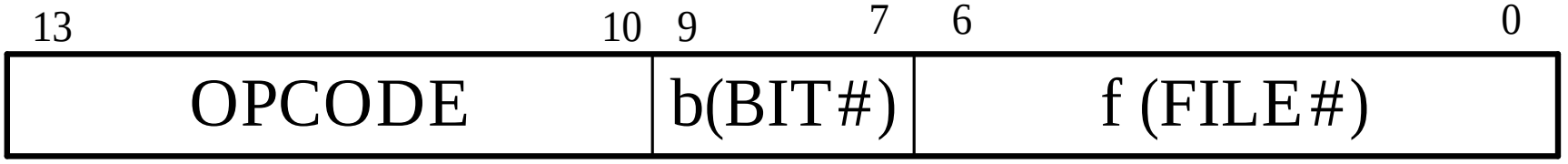


d=0 for destination W

d=1 for destination file register

f=7-bit file register address

- Bit-oriented file register operations

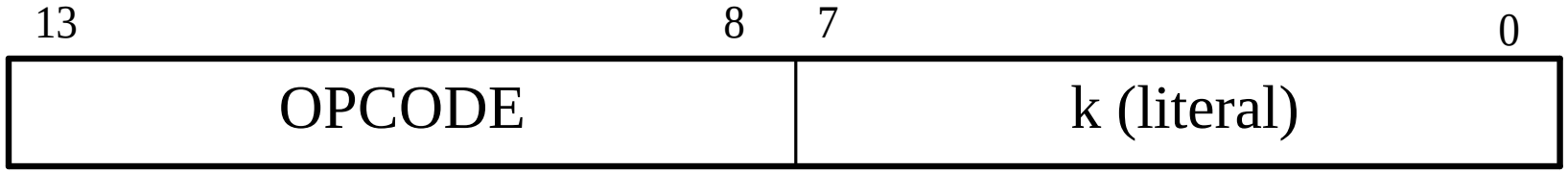


b=3-bit address

f=7-bit file register address

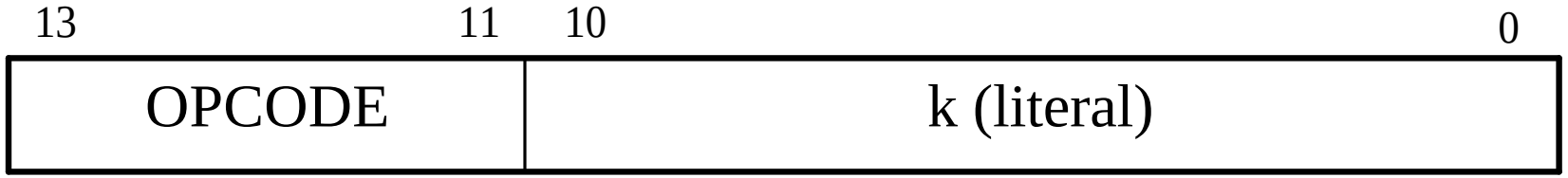
Instruction Format Continue…

- Literal and control operations
  - General



k=8-bit intermediate value.

  - CALL and GOTO instructions only



k=11-bit intermediate value.

PIC16F87X INSTRUCTION SET

Mnemonic Operands	Description	Cycles	14-bit Opcode MSb LSb	Status Affected
BYTE-ORIENTED FILE REGISTER OPERATIONS				
ADDWF f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z
ANDWF f, d	AND W with f	1	00 0101 dfff ffff	Z
CLRF f	Clear f	1	00 0001 1fff ffff	Z
CLRW -	Clear W	1	00 0001 0xxx xxxx	Z
COMF f, d	Complement f	1	00 1001 dfff ffff	Z
DECF f, d	Decrement f	1	00 0011 dfff ffff	Z
DECFSZ f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff	
INCF f, d	Increment f	1	00 1010 dfff ffff	Z
INCFSZ f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff	
IORWF f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z
MOVF f, d	Move f	1	00 1000 dfff ffff	Z

Instruction Set Continue …

Mnemonic Operands	Description	Cycles	14-bit Opcode MSb LSb	Status Affected
MOVWF f	Move W to f	1	00 0000 1fff ffff	
NOP -	No Operation	1	00 0000 0xx0 0000	
RLF f, d	Rotate Left through Carry	1	00 1101 dfff ffff	C
RRF f, d	Rotate Right through Carry	1	00 1100 dfff ffff	C
SUBWF f, d	Subtract W from f	1	00 0010 dfff ffff	Z,DC,Z
SWAPF f, d	Swap nibbles in f	1	00 1110 dfff ffff	
XORWF f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z
BIT-ORIENTED FILE REGISTER OPERATIONS				
BCF f, b	Bit Clear f	1	01 00bb bfff ffff	
BSF f, b	Bit Set f	1	01 01bb bfff ffff	
BTFSC f, b	Bit Test f, Skip if clear	1(2)	01 10bb bfff ffff	
BTFSS f, b	Bit Test f, Skip if set	1(2)	01 11bb bfff ffff	
LITERAL AND CONTROL OPERATIONS				
ADDLW k	Add literal and W		11 111x kkkk kkkk	C,DC,Z

Instruction Set Continue …

Mnemonic Operands	Description	Cycles	14-bit Opcode MSb LSb	Status Affected
ANDLW k	AND literal with W	1	11 1001 kkkk kkkk	Z
CALL k	Call subroutine	2	10 0kkk kkkk kkkk	
CLRWDT -	Clear Watchdog Timer	1	00 0000 0110 0100	TO, PD
GOTO k	Go to address	2	10 1kkk kkkk kkkk	
IORLW k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z
MOVLW k	Move literal to W	1	11 00xx kkkk kkkk	
RETFIE -	Return from interrupt	2	00 0000 0000 1001	
RETLW k	Return with literal in W	2	11 01xx kkkk kkkk	
RETURN -	Return from Subroutine	2	00 0000 0000 1000	
SLEEP -	Go into standby mode	1	00 0000 0110 0011	TO, PD
SUBLW k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z
XORLW k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z

Instruction Examples and Addressing Modes

- Immediate or Inherent Addressing.
- Direct Addressing.
- Indirect Addressing
- Index Addressing.

Immediate Addressing

- In this addressing mode the data is directly provided in the instruction, no memory location is used.

Direct Addressing

- The direct addressing mode accesses data from a memory location located in one of the four data banks.
- The bank is selected by suing IRP0 and IRP1 bits of the status register.
- The data is then read or written by providing the address of the instruction.

Sample Instructions & Addressing Modes

ADDLW — Add Literal and W_(Immediate Addressing)

Syntax: [label] ADDLW k

Operands: 0 = k = 255

Operation: (W) + k → (W)

Status Affected: C, DC, Z

Encoding:

11	111x	kkkk	kkkk

Description: The contents of the W register are added to the eight bit literal ‘k’ and the result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process data	Write to W

Example: ADDLW 0x15

Before Instruction

W = 0 x 10

After Instruction

W = 0 x 25

Sample Instructions Continue..

ANDLW AND Literal with W_ _(Immediate Addressing)

Syntax: [label] ANDLW k

Operands: 0 = k = 255

Operation: (W) . AND . (k) → (W)

Status Affected: Z

Encoding:

11	1001	kkkk	kkkk

Description: The contents of the W register are AND'ed with the eight bit literal 'k' and the result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W

Example: ANDLW 0 x 5F

Before Instruction

W = 0 x A3

After Instruction

W = 0 x 03

Sample Instructions Continue..

ADDWF — Add W and f (Direct Addressing)

Syntax: [label] ADDWF f, d

Operands: 0 = f = 127 d ? [0,1]

Operation: (W) + (f) → (destination)

Status Affected: C, DC, Z

Encoding:

00	0111	dfff	ffff

Description: Add the contents of the W register with register ‘f ’. If ‘d’ is 0 the result is stored in the W register. If ‘d’ is 1 the result is stored back in register ‘f ’.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f ’	Process data	Write to destination

Example:

```
ADDWF FSR,  0
Before Instruction
        W   = 0 x 17
        FSR = 0 x C2
After Instruction
        W   = 0 x D9
        FSR = 0 x C2
```

Sample Instructions Continue..

ANDWF AND W with f (Direct Addressing)

Syntax: [label] ANDWF f, d

Operands: 0 = f = 127

Operation: (W) . AND . (k) → (destination)

Status Affected: Z

Encoding:

00	0101	dfff	ffff

Description: AND the W register with register ‘f ’. If ‘d’ is 0 the result is stored in the W register. If ‘d’ is 1 the result is stored back in register ‘f ’.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f ’	Process data	Write to destination

Example: ANDWF FSR, 1

Before Instruction

W = 0 x 17

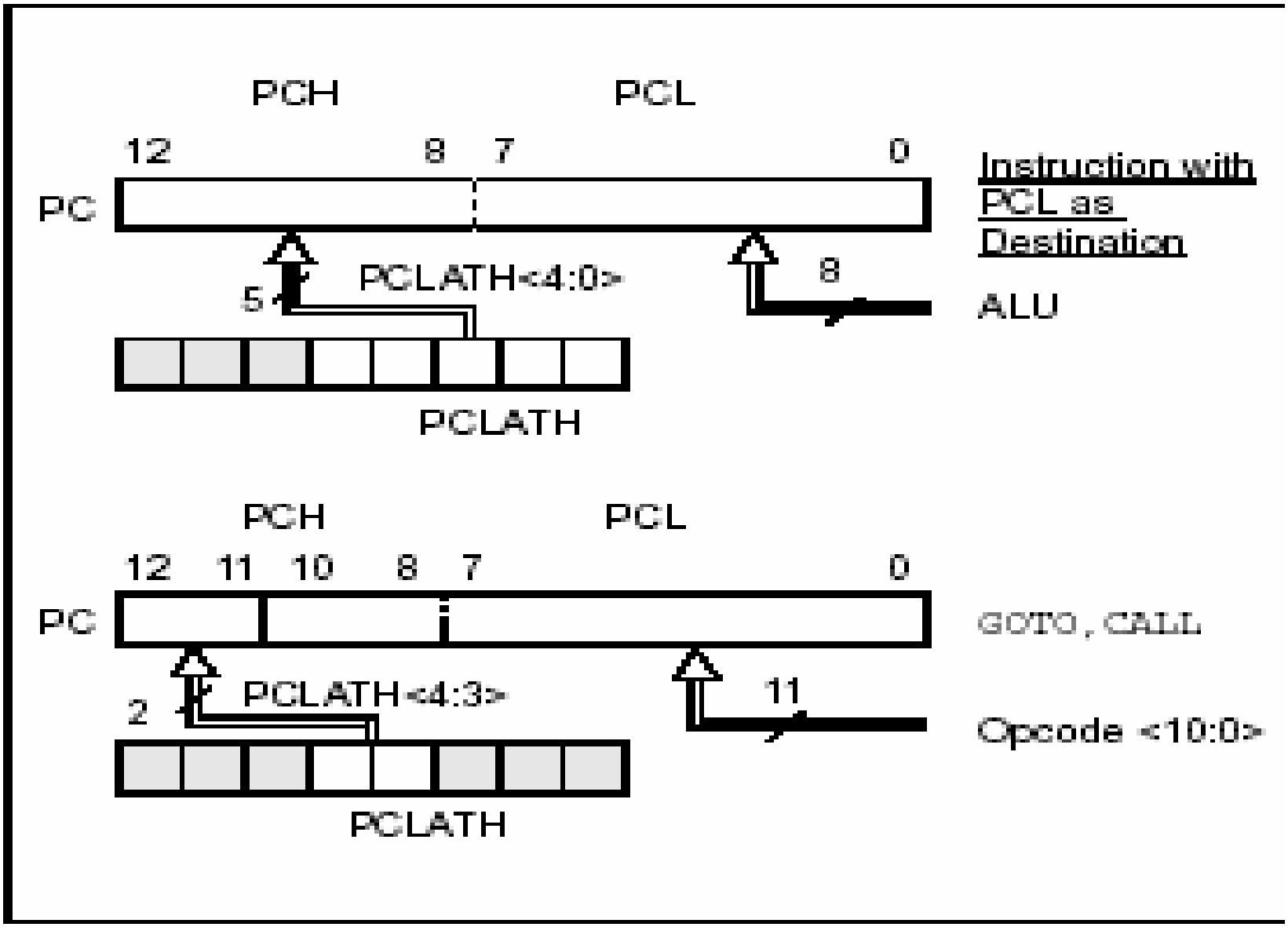
FSR= 0 x C2

After Instruction

W = 0 x 17

FSR= 0 x 02

LOADING OF PC IN DIFFERENT SITUATIONS



PCL and PCLATH

- The program counter (PC) is 13-bits wide.
- The low byte comes from the PCL register, which is a readable and writable register.
- The upper bits (PC<12:8>) are not readable, but are indirectly writable through the PCLATH register.
- The upper example in the figure shows how the PC is loaded on a write to PCL (PCLATH<4:0> ? PCH).
- The lower example in the figure shows how the PC is loaded during a CALL or GOTO instruction (PCLATH<4:3> ? PCH).

COMPUTED GOTO

- A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL).

- When doing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256 byte block). Refer to the application note, "*Implementing a Table Read*" (AN556) for more details.

Program Memory Paging

- When doing a CALL or GOTO instruction, the upper 2 bits of the address are provided by PCLATH<4:3>.
- When doing a CALL or GOTO instruction, the user must ensure that the page select bits are programmed so that the desired program memory page is addressed.
- If a return from a CALL instruction (or interrupt) is executed, the entire 13-bit PC is popped off the stack. Therefore, manipulation of the PCLATH<4:3> bits is not required for the return instructions (which POPs the address from the stack).

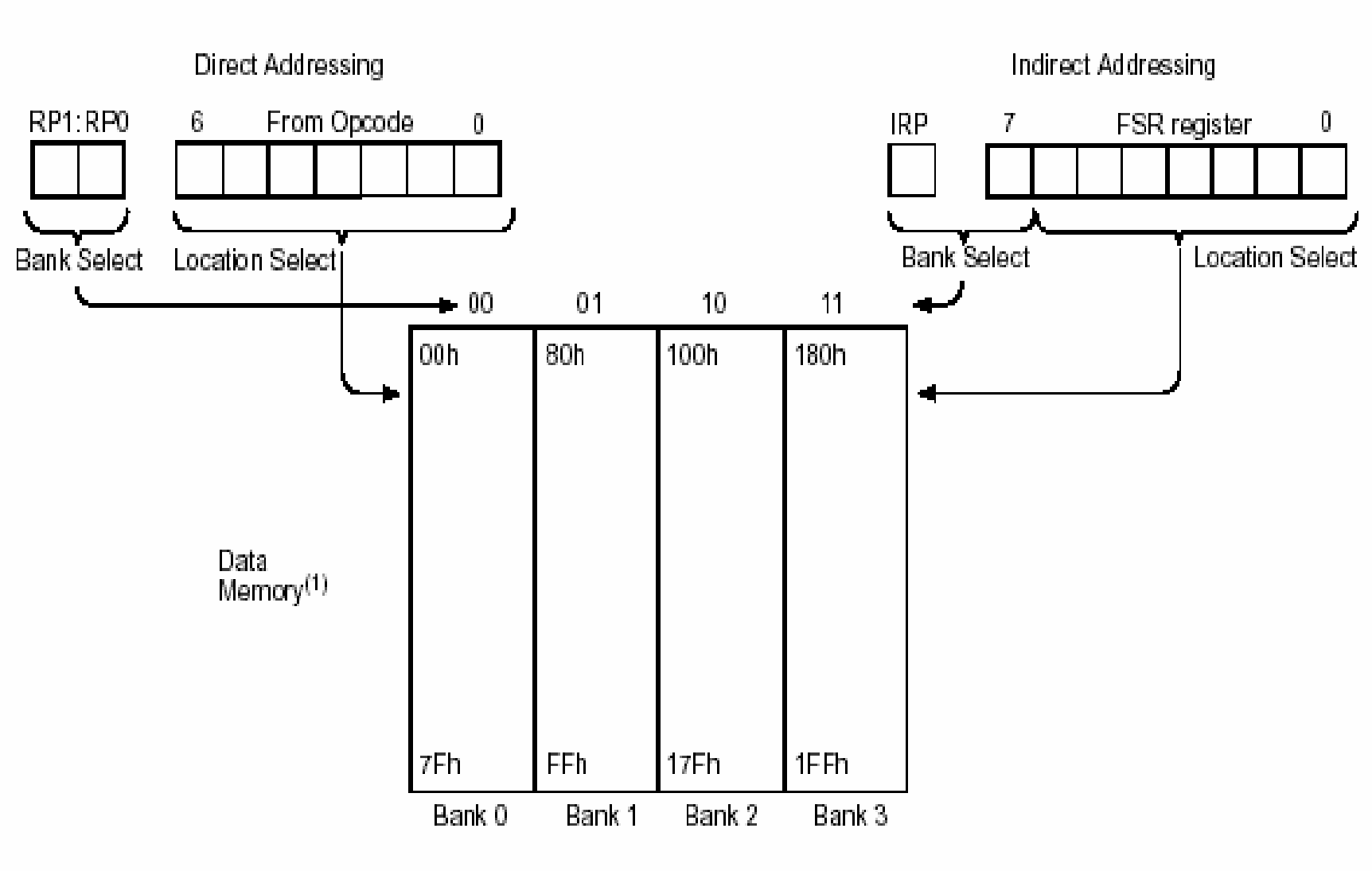
CALL OF A SUBROUTINE IN PAGE 1 FROM PAGE 0

```
            ORG  0x500
            BCF  PCLATH,4
            BSF  PCLATH,3    ;Select page 1
                             ;(800h-FFFh)
            CALL SUB1_P1     ;Call subroutine in
            :                ;page 1 (800h-FFFh)
            :
            ORG  0x900       ;page 1 (800h-FFFh)
SUB1_P1
            :                ;called subroutine
                             ;page 1 (800h-FFFh)
            :
            RETURN           ;return to
                             ;Call subroutine
                             ;in page 0
                             ;(000h-7FFh)
```

Indirect Addressing, INDF and FSR Registers

- The INDF register is not a physical register.
- Addressing the INDF register will cause indirect addressing.
- Any instruction using the INDF register actually accesses the register pointed to by the File Select Register, FSR.
- Reading the INDF register itself, indirectly (FSR = ’0’) will read 00h.
- An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 2-6.

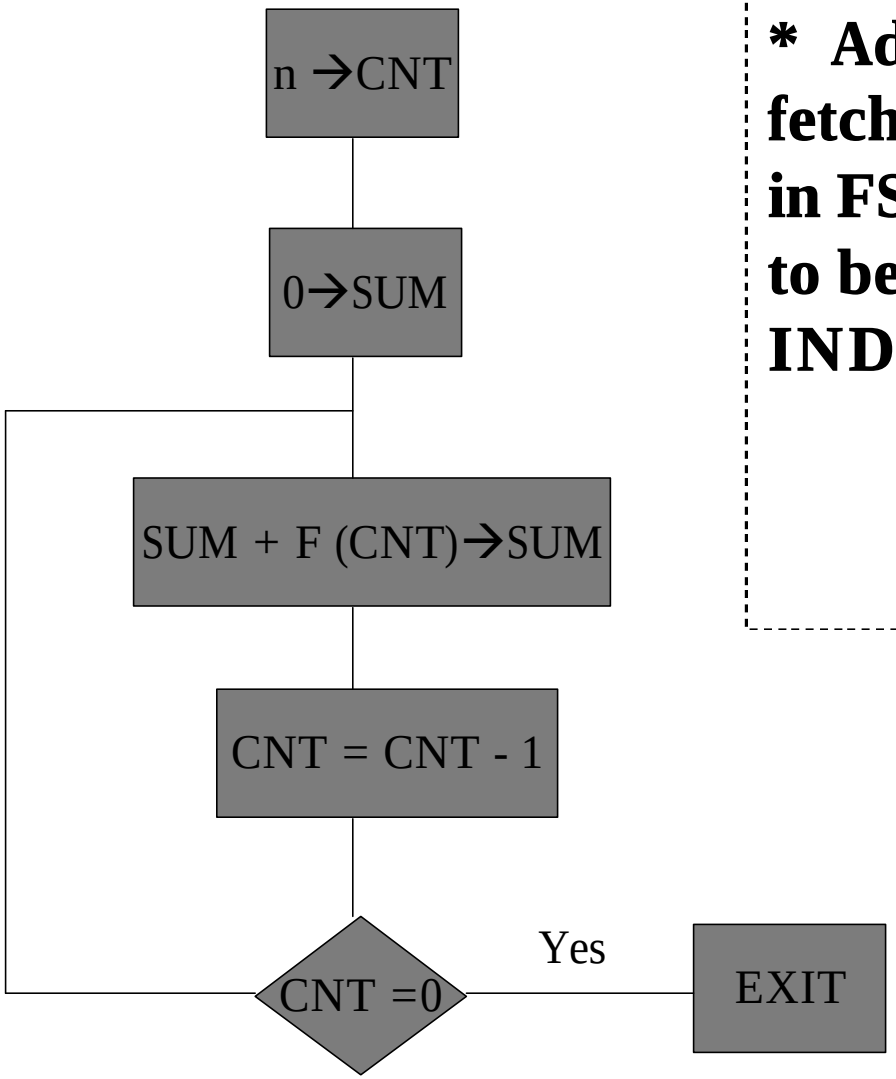
DIRECT/INDIRECT ADDRESSING



INDIRECT ADDRESSING

```
        MOVLW  0x20     ;initialize pointer
        MOVWF  FSR      ;to RAM
NEXT    CLRF   INDF     ;clear INDF register
        INCF   FSR,F    ;inc pointer
        BTFSS  FSR,4    ;all done?
        GOTO   NEXT     ;no clear next
CONTINUE
        :               ;yes continue
```

Index Addressing of Data Register



*** Address to be fetched is stored in FSR and needs to be loaded using INDF**

Index Addressing Continue..

```
        MOVLW   n           ; n →(Immediate)
        MOVWF   CNT         ; n →CNT
        CLRF    SUM         ; 0 →SUM
        MOVLW   n
        MOVWF   FSR         ; n →FSR
LOOP    MOVF    SUM,W       ;SUM →W
        ADDWF   INDF,W      ;SUM+
                            ;Reg (n)→SUM
        MOVWF   SUM         ;W →SUM
        DECF    FSR, F      ; n-1→FSR
        DECF    CNT,F       ;CNT-1 →CNT
        BTFSS   Status, z   ;sleep if z is set
        GOTO    LOOP
```

Implementing a Table Read

```
        .
        .
        movlw offset          ;load offset in w registe
        call  Table
        .
        .
        .
Table:  addwf  PCL,F          ;add offset to pc to generate
                              ;a computed go to
        retlw  'A'            ;return the ASCII char A
        retlw  'B'            ;return the ASCII char B
        retlw  'C'            ;return the ASCII char C
        .
        .
        .
```

IMPLEMENTATION EXAMPLES

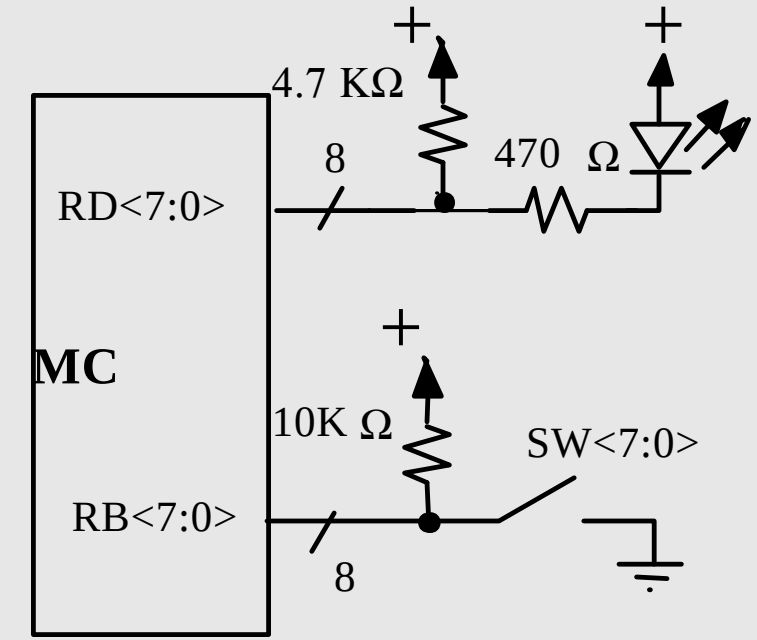
1. I/O Interfacing
2. Designing a wait loop
3. Designing table look up
4. Using TIMER
5. Interrupt Realization

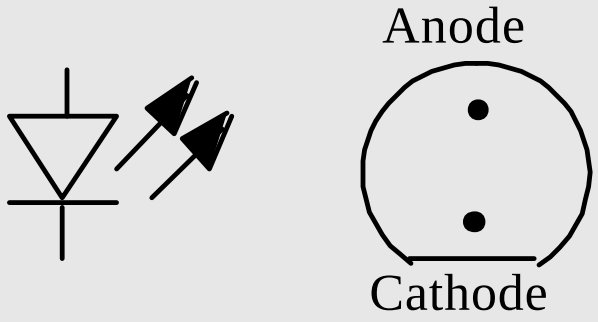
Laboratory Equipment

Parts List:

1. Breadboard
2. PIC 16F877 Microcontroller
3. 8 LEDs
4. 8 470 ohms Resisters
5. 8 10 kilo ohms Resisters
6. 8 position DIP switch

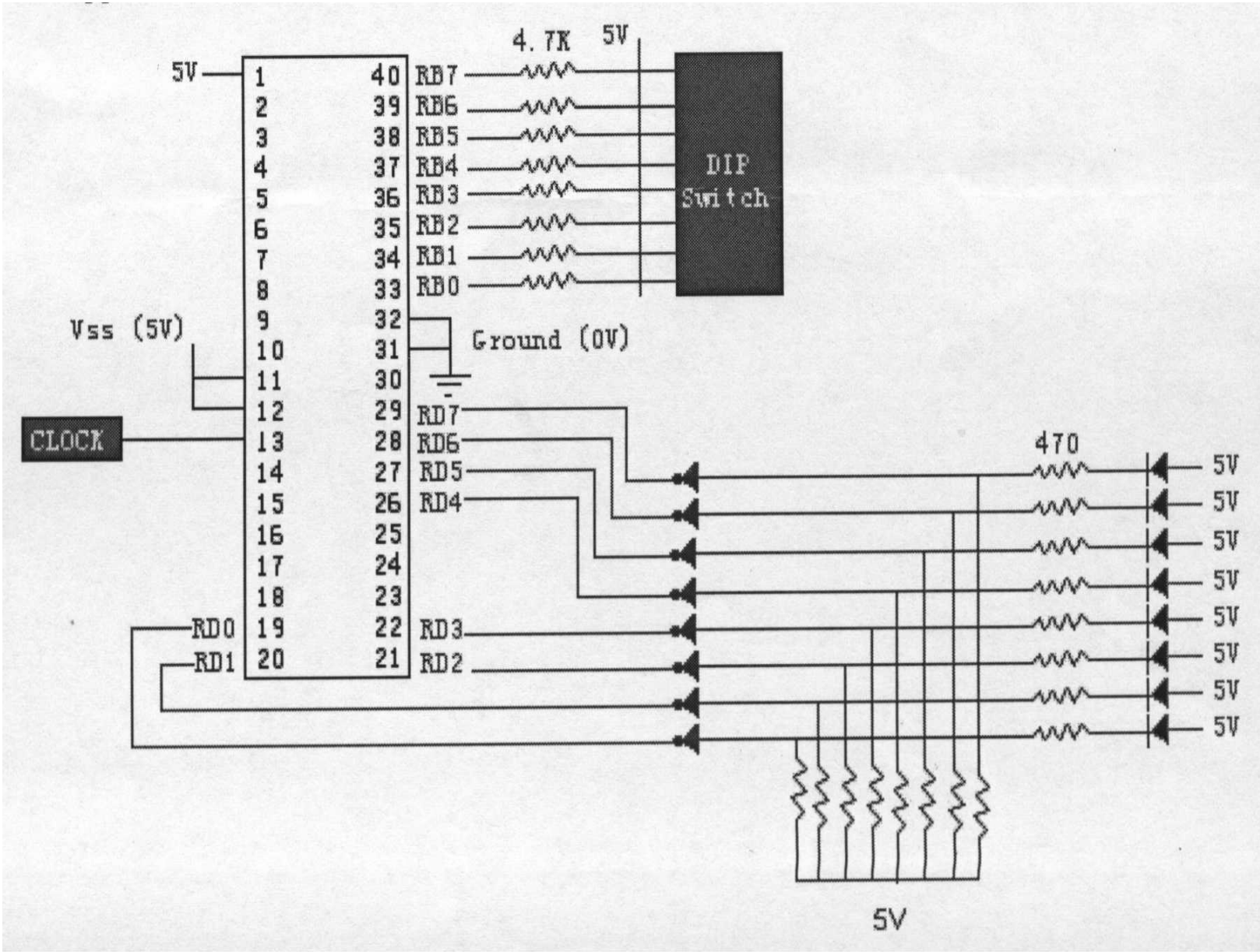
Circuit Diagram





Bottom View

Circuit Diagram



Program 1 (I/O Interface, Read and Write)

```
;This program is written for PIC16F877 microcontroller to read the value of 8
;DIP switches connected to PORTB and display their value on 8 LEDs connected
;to PORTD.
;================================================
;declaration of variable and main registers in the registers memory ROM
;================================================
    LIST P=16F877                    ;determine the type of the PIC
;------------------------------------------
;***BANK1 REGI    STER SET
;------------------------------------------
OPTIONR  EQU  81        ;define the registers by assigning the address of them
TRISA    EQU  85        ;in the RAM to their label
TRISB    EQU  86        ;i.e. TRISA register is referring to program memory
TRISC    EQU  87        ;location of 85 in Hex
TRISD    EQU  88
```

Program 1 Continue …

```
TRISE    EQU      89
SPBRG    EQU      99
;-----------------------------------------------
;***BANK 0 REGISTER SETS
;-----------------------------------------------
TMR0     EQU      1
PC       EQU      2
STATUS   EQU      3
FSR      EQU      4
PORTA    EQU      5
PORTB    EQU      6
PORTC    EQU      7
PORTD    EQU      8
PORTE    EQU      9
INTCON   EQU      0B
PIR1     EQU      0C
RCSTA    EQU      18
RCREG    EQU      1A
```

Program 1 Continue …

```
;-----------------------------------------------
;***BANK 0 REGISTER SETS
;-----------------------------------------------
TEMP     EQU      30     ;these are special function registers, which you can use
TIME     EQU      31     ;to save some data, like variable declaration in C and
TIME2    EQU      32      ;java and other programming language
COUNT    EQU      33
RATE     EQU      34
RATEBAK EQU      35
;-----------------------------------------------
;***STATUS BITS
;-----------------------------------------------
C          EQU      0
DC         EQU      1
Z          EQU      2
RP0        EQU      5
RP1        EQU      6
```

Program 1 Continue …

```
;-----------------------------------------------
;**** OPTION BIT
;-----------------------------------------------
TOCS    EQU     5
TOSE    EQU     4
PS0     EQU     0
PS1     EQU     1
PS2     EQU     2
;-----------------------------------------------
;**** INTCON BITS
;-----------------------------------------------
GIE     EQU     7
TOIE    EQU     5
TOIF    EQU     2
```

Program 1 Continue …

```
;-------------------------------------------------
    ;***W OR FILE INDICATORS
;-------------------------------------------------
W         EQU     0
F         EQU     1


;-------------------------------------------------
;****OTHER BITS
;-------------------------------------------------
PEIE      EQU     6
GIE       EQU     7
RCIE      EQU     5
RCIF      EQU     5
```

Program 1 Continue …

```
;================================================================
;The main program body
;================================================================
;----------------------------------------------
;RESET vector where the program will always start
;----------------------------------------------
        ORG  0              ;this ORG 0 means that you are at memory location 0000
RESET   GOTO   MAIN         ;now we tell the MC to go to the main body
;----------------------------------------------
;INTERRUPT vector where the PC will jump when interrupt comes
;----------------------------------------------
        ORG 04
INT     CALL INTHND         ;when interrupt comes call interrupt handler routine
        RETFIE              ;return from interrupt after doing work in
                            ;INTHND
```

Program 1 Continue …

```
;-----------------------------------------------
;the main program body
;-----------------------------------------------
            ORG 20
MAIN     CLRF     INTCON ;clear INTCON to disable all interrupts
;-----------------------------------------------
;**** INTILIZE PORTS
;-----------------------------------------------
            BSF        STATUS, RP0      ;get to Bank1 of memory (RP1:RP0 = 01)
            BCF        STATUS, RP1      ; because TRIS  registers are in Bank1
            MOVLW  0XFF                 ;put FF in TRISB so it is all ones, means
            MOVWF  TRISB                ; all pins input, 0 means output, 1 =  input pin
            CLRF      TRISD             ;clear TRISD, all zeros, all output
            BCF        STATUS,RP0
```

Program 1 Continue …

```
;-------------------------------------------------------------------
;            LOOP READ SWITCH  AND WRITE ON LEDS
;-------------------------------------------------------------------
LOOP    NOP
        NOP
        MOVF     PORTB,W        ;move the value of the switch to w using port b
        MOVWF   PORTD           ;move w to LEDs through PORTD
        GOTO     LOOP           ;keep circulating in the same loop
        END                     ;end of the program
```

Program 2(Flash LEDs Using Wait loop)

```
    LIST P=16F877                ;determine the type of the PIC
         ORG     0                ;Reset Vector
RESET    GOTO    MAIN             ;now we tell the MC to goto the main body
         ORG 04                   ;Interrupt vecotor
INT      CALL INTHND  ;when interrupt comes call interrupt handler routine
         RETFIE       ;return from interrupt
;-----------------------------------------------
;the main program body
;-----------------------------------------------
         ORG 20
MAIN     CLRF    INTCON           ;clear INTCON to disable all interrupts
;-----------------------------------------------
;**** INTILIZE PORTS
;-----------------------------------------------
         BSF     STATUS,RP0       ;get to bank1 of memory (RP1:RP0 = 01)
         BCF     STATUS,RP1       ; because TRIS registers are there
```

Program 2 Continue …

```
        MOVLW   0XFF            ;put FF in TRISB so it is all ones, means all pins input
        MOVWF   TRISB           ;0 means output, while 1 make the pin an input pin
        CLRF    TRISD           ;clear TRISD, all zeros, all ouput
        BCF     STATUS,RP0       ;go back to bank0
        BCF     STATUS,RP1
        ;-----------------------------------------------
        ;**** PUT FF IN TEMP REGISTER
        ;-----------------------------------------------
        MOVLW  0XFF         ;put FF in TEMP register which is a variable register
        MOVWF  TEMP         ;move ff first to W register then from W to TEMP
        MOVLW  0XFF         ;put FF in COUNT register which is a variable register
        MOVWF  COUNT        ;move FF first to W register then from W to COUNT
        MOVLW  0XFF         ;put FF in COUNT1 register which is a variable register
        MOVWF  COUNT1       ;move FF first to W register then from W to COUNT
        ;-----------------------------------------------
        ;**** complementing loop
        ;-----------------------------------------------
LOOP    COMF    TEMP,F      ;complement TEMP and put the result in TEMP " 0 or FF"
        MOVF    TEMP,W      ;move TEMP to PORTD, first move TEMP to w then to
        MOVWF  PORTD        ;PORTD
```

Program 2 Continue …

```
        MOVLW  0XFF     ;now again fill COUNT with FF for the next waiting loop
        MOVWF  COUNT
        MOVLW  0XFF     ;now again fill COUNT1 with FF for the next waiting loop
        MOVWF  COUNT1
        MOVLW  0X05     ;now again fill RATE with 05 for the next waiting loop
        MOVWF  RATE
        CALL      WAIT    ;call wait loop
        GOTO      LOOP

WAIT    DECFSZ  COUNT,F     ;decrement count and check if it is zero go check count1,
                            ;else
        GOTO     WAIT       ;goto wait and keep decrementing it untill it become 0
        DECFSZ  COUNT1,F
        GOTO     WAIT
        DECFSZ    RATE,F
        GOTO     WAIT
        RETURN              ;if all counting registers become zero, then return to loop
        END                 ;end of the program
```

Program 3 (Flash LEDs Using Wait loop & Using Different rates)

```
            LIST P=16F877                ;determine the type of the PIC
            ORG      00
RESET       GOTO  MAIN                   ;now we tell the MC to go to the main body
;-----------------------------------------------
;the main program body
;-----------------------------------------------
            ORG 20
MAIN        CLRF     INTCON              ;clear INTCON to make sure all interrupts are disabled
;-----------------------------------------------
;**** INTILIZE PORTS
;-----------------------------------------------
            BSF      STATUS,RP0          ;get to bank1 of memory (RP1:RP0 = 01) because TRIS
            BCF      STATUS,RP1          ;registers are there
            MOVLW    0XFF                ;put FF in TRISB so it is all ones, means all pins input
            MOVWF    TRISB               ;0 means output, while 1 make the pin an input pin
            CLRF     TRISD               ;clear TRISD, all zeros, all output
            BCF      STATUS,RP0          ;go back to bank0
            BCF      STATUS,RP1
```

Program 3 Continue…

```
;-----------------------------------------------
;**** PUT FF IN TEMP REGISTER
;-----------------------------------------------
     MOVLW      0XFF          ;put FF in TEMP register which is a variable register
     MOVWF      TEMP          ;move ff first to W register then from W to TEMP
     MOVLW      0XFF          ;put FF in COUNT register which is a variable register
     MOVWF      COUNT         ;move FF first to W register then from W to COUNT
     MOVLW      0XFF          ;put FF in COUNT1 register which is a variable register
     MOVWF      COUNT1        ;move FF first to W register then from W to COUNT
;-----------------------------------------------
;** complementing loop
;-----------------------------------------------
LOOP  COMF      TEMP,F        ;complement TEMP and put the result in TEMP "TEMP
                              either 0 or FF"
       MOVF     TEMP,W        ;move TEMP to LEDs through PORTD, first move
                              TEMP to w then to
      MOVWF     PORTD         ;PORTD
      CALL      READS         ;call read switch
      MOVWF     RATE          ;when returned w will be filled with the desired rate
      MOVWF     RATEBAK       ;thus we move it to rate and rate backup registers
```

Program 3 Continue..

```
        MOVLW  0XFF             ; fill COUNT with FF for the next waiting loop
        MOVWF  COUNT
        MOVLW  0XFF             ; fill COUNT1 with FF for the next waiting loop
        MOVWF  COUNT1
        MOVF   RATEBAK,W        ; fill RATE with RATE BACKUP
        MOVWF  RATE
        CALL   WAIT             ;call wait loop
        GOTO   LOOP
;-----------------------------------------------
;read the switch value and save it in RATE
;-----------------------------------------------
READS   MOVF   PORTB,W          ;move PORTB to w to take the switch values
        ANDLW  0X07             ;get only the less three bits of w as the rate
        ADDWF  PC,F             ;add w to pc to
        RETLW  0X01             ;if w = 0, pc = pc, then return with 0x01 in W
        RETLW  0X02             ;if w = 1, pc = pc+1, then return with 0x02 in W
        RETLW  0X04             ;if w = 2, pc = pc+2, then return with 0x04 in W
        RETLW  0X06             ;if w = 3, pc = pc+3, then return with 0x06 in W
        RETLW  0X08             ;if w = 4, pc = pc+4, then return with 0x08 in W
        RETLW  0X0A             ;if w = 5, pc = pc+5, then return with 0x0A in W
```

Program 3 Continue..

```
    RETLW       0X0C            ;if w = 6, pc = pc+6, then return with 0x0C in W
    RETLW       0X0E            ;if w = 7, pc = pc+7, then return with 0x0E in W
;-----------------------------------------------
;wait loop
;-----------------------------------------------
WAIT    DECFSZ    COUNT,F       ;decrement count and check if it is zero go check
                                ;count1, else
        GOTO WAIT               ;go to wait and keep decrementing it until it is 0
        DECFSZ   COUNT1,F
        GOTO WAIT
        DECFSZ   RATE,F
        GOTO WAIT
        RETURN                  ;if all counting registers become zero, then return to
                                ;loop

        END                     ;end of the program
```

Program 4 (Flash LEDs Using Wait loop (using TMR0) & Different rates)

```
            LIST P=16F877                ;determine the type of the PIC
            ORG     00
RESET       GOTO    MAIN
;-----------------------------------------------
;the main program body
;-----------------------------------------------
            ORG     20
MAIN        CLRF    INTCON               ;clear INTCON to make sure all interrupts are disabled
;-----------------------------------------------
;**** INTILIZE PORTS
;-----------------------------------------------
            BSF     STATUS,RP0                  ;get to bank1 of memory (RP1:RP0 = 01)
            BCF     STATUS,RP1                  ; because TRIS  registers are there
            MOVLW   0XFF                        ;put FF in TRISB so it is all ones,  all input
            MOVWF   TRISB                       ;0 means output, while 1 mean input
            CLRF    TRISD                       ;clear TRISD, all zeros, all output
            BCF     STATUS,RP0                  ;go back to bank0
            BCF     STATUS,RP1
```

Program 4 Continue …

```
;-----------------------------------------------
;**** INTILIZE TIMER FROM OPTION Register
;-----------------------------------------------
        BSF     STATUS,RP0      ;get access to bank1 if we are not there
        BCF     STATUS,RP1
        BCF     OPTIONR, TOCS   ;set TMR0 increment every internal instruction cycle
        BCF     OPTIONR, TOSE   ;TMR0 increment on low-to-high transition of clock
        BSF     OPTIONR, PS0    ;set the pre-scalar for TMR0 to overflow after 256
        BSF     OPTIONR, PS1            ;instruction cycle or FF in hex
        BSF     OPTIONR, PS2
        BCF     STATUS,RP0              ;go back to bank0
        BCF     STATUS,RP1
```

Program 4 Continue …

```
;-----------------------------------------------
;**** PUT FF IN TEMP REGISTER
;-----------------------------------------------
            MOVLW  0XFF              ;put FF in TEMP register which is a variable register
            MOVWF  TEMP              ;move ff first to W register then from W to TEMP
            MOVLW  0XFF              ;put FF in COUNT register which is a variable register
            MOVWF  COUNT             ;move FF first to W register then from W to COUNT
            MOVLW  0XFF              ;put FF in COUNT1 register which is a variable register
            MOVWF  COUNT1            ;move FF first to W register then from W to COUNT
;-----------------------------------------------
;**** complementing loop
;-----------------------------------------------
LOOP        COMF     TEMP,F          ;complement TEMP and put the result in TEMP "0 or FF"
            MOVF     TEMP,W          ;move TEMP to  PORTD, first move TEMP to w then to
            MOVWF  PORTD             ;PORTD
            CALL     READS           ;call read switch
```

Program 4 Continue …

```
        MOVWF   RATE            ;when returned w will be filled with the desired rate
        MOVWF   RATEBAK         ;thus we move it to rate and rate backup registers
        MOVLW   0XFF            ;now again fill COUNT with FF for the next waiting loop
        MOVWF   COUNT
        MOVLW   0XFF            ;now again fill COUNT1 with FF for the next waiting loop
        MOVWF   COUNT1
        MOVF    RATEBAK,W       ;now again fill RATE with RATE BACKUP for the next
                                ;waiting loop
        MOVWF   RATE
        CALL    WAIT            ;call wait loop
        GOTO    LOOP
;-----------------------------------------------
;read the switch value and save it in RATE
;-----------------------------------------------
READS   MOVF    PORTB,W         ;move PORTB to w to take the switch values in W register
        ANDLW   0X07            ;get only the less three bits of w as the rate (00000111)
        ADDWF   PC,F            ;add w to pc to specify which instruction will be execute
        RETLW   0X01            ;if w = 0, pc = pc, then return with 0x01 in W
        RETLW   0X02            ;if w = 1, pc = pc+1, then return with 0x02 in W
        RETLW   0X04            ;if w = 2, pc = pc+2, then return with 0x04 in W
        RETLW   0X06            ;if w = 3, pc = pc+3, then return with 0x06 in W
```

Program 4 Continue …

```
        RETLW  0X08          ;if w = 4, pc = pc+4, then return with 0x08 in W
        RETLW  0X0A          ;if w = 5, pc = pc+5, then return with 0x0A in W
        RETLW  0X0C          ;if w = 6, pc = pc+6, then return with 0x0C in W
        RETLW  0X0E          ;if w = 7, pc = pc+7, then return with 0x0E in W
;-----------------------------------------------
;wait loop
;-----------------------------------------------
WAIT    BTFSS INTCON,TOIF    ;decrement count and check if itis zero go
                             ;check count1, else
        GOTO WAIT            ;go to wait and keep decrementing it untill it
                             ;become 0
        DECFSZ COUNT1,F
        GOTO WAIT
        DECFSZ RATE,F
        GOTO WAIT
        RETURN               ;if all counting registers become zero, then
                             ;return to loop
        END                  ;end of the program
```

Program 5 (Flash LEDs Interrupts)

```
RESET     GOTO MAIN                 ;now we tell the MC to goto the main body
;-------------------------------------------------------------------------------
;INTERRUPT vector where the PC will jump when interrupt comes
;-------------------------------------------------------------------------------
          ORG 04
INT       CALL INTHND               ;when interrupt comes call interrupt handler routine
          RETFIE                    ;return from interrupt after doing work in INTHND
;---------------------------------
;the main program body
;---------------------------------
          ORG 20
MAIN      CLRF      INTCON          ;clear INTCON to make sure all interrupts are disabled
;---------------------------------
;**** INTILIZE PORTS
;---------------------------------
          BSF       STATUS,RP0      ;get to bank1 of memory (RP1:RP0 = 01) because TRIS
          BCF       STATUS,RP1      ;registers are there
          MOVLW  0XFF               ;put FF in TRISB so it is all ones, means all pins input
          MOVWF  TRISB              ;0 means output, while 1 make the pin an input pin
          CLRF      TRISD           ;clear TRISD, all zeros, all output
```

Program 5 Continue …

```
;-----------------------------------------
;***** INTILIZE TIMER IN INTCON
;-----------------------------------------------
        BSF     INTCON, GIE     ;set general interrupt enable bit to enable all interrupt
        BSF     INTCON, TOIE    ;set TMR0 interrupt enable bit to enable TMR0
                                ;interrupt
        BCF     INTCON, TOIF    ;clear TMR0 flag, TMR0 doesn't overflow yet
;-------------------------------------------------------------
;**** INTILIZE TIMER FROM OPTION Register
;-------------------------------------------------------------
        BSF     STATUS,RP0      ;get access to bank1 if we are not there
        BCF     STATUS,RP1
        BCF     OPTIONR, TOCS   ;set TMR0 increment every internal instruction cycle
        BCF     OPTIONR, TOSE   ;TMR0 increment on low-to-high transition of clock
        BSF     OPTIONR, PS0    ;set the pre scalar for TMR0 to overflow after 256
        BSF     OPTIONR, PS1    ;instruction cycle or FF in hex
        BSF     OPTIONR, PS2
        BCF     STATUS,RP0      ;go back to bank0
        BCF     STATUS,RP1
```

Program 5 Continue …

```
;-----------------------------------------------
;**** PUT FF IN TEMP REGISTER
;-----------------------------------------------
        MOVLW  0XFF             ;put FF in TEMP register which is a variable register
        MOVWF  TEMP             ;move ff first to W register then from W to TEMP
        MOVLW  0XFF             ;put FF in COUNT register which is a variable register
        MOVWF  COUNT            ;move FF first to W register then from W to COUNT
;-----------------------------------------------
;**** LOOP TO DO NOTHING
;-----------------------------------------------
LOOP    NOP                     ;this loop will take you to the READS every time and it
        NOP                     ;is not doing else, but wait until interrupt
        NOP                     ;comes, then the program counter PC will jump
        BCF     INTCON,TOIF     ;automatically to interrupt vector where the flashing task
        BSF     INTCON,GIE      ;is gone be done.. the other instruction is to enable the
        BSF     INTCON,TOIE     ;interrupt every time
        CALL    READS           ;call READS to get the RATE from the switches
        MOVWF   RATE            ;when returned w will be filled with the desired rate
        MOVWF   RATEBAK         ;thus we move it to rate and rate backup registers
        GOTO    LOOP
```

Program 5 Continue …

```
;-----------------------------------------------
;read the switch value and save it in RATE
;-----------------------------------------------
READS   MOVF    PORTB,W         ;move PORTB to w to take the switch values in W register
        ANDLW   0X07            ;get only the less three bits of w as the rate (00000111)
        ADDWF   PC,F            ;add w to pc to specify which instruction will be execute
        RETLW   0X01            ;if w = 0, pc = pc, then return with 0x01 in W
        RETLW   0X02            ;if w = 1, pc = pc+1, then return with 0x02 in W
        RETLW   0X04            ;if w = 2, pc = pc+2, then return with 0x04 in W
        RETLW   0X06            ;if w = 3, pc = pc+3, then return with 0x06 in W
        RETLW   0X08            ;if w = 4, pc = pc+4, then return with 0x08 in W
        RETLW   0X0A            ;if w = 5, pc = pc+5, then return with 0x0A in W
        RETLW   0X0C            ;if w = 6, pc = pc+6, then return with 0x0C in W
        RETLW   0X0E            ;if w = 7, pc = pc+7, then return with 0x0E in W
;-----------------------------------------------
;INTHND function that perform the flashing task
;-----------------------------------------------
INTHND DECFSZ   COUNT,F ;decrement count1 skip if zero, if count1 = zero, then
       GOTO     LOOP1   ;we go decrement rate. if not we go to Loop1
        DECFSZ      RATE,F      ;if count = 0, then we decrement rate and check if it is zero
```

Program 5 Continue …

```
        GOTO LOOP1              ;then we go and do the work, otherwise we goto
                                ;loop1
                                ;if both COUNT and RATE are becomes 0, then this mean
                                ;this is the time to complement the LEDs through PORTD

WORK    COMF    TEMP,F          ;complement TEMP and put the result in TEMP "
                                0 or FF"
        MOVF    TEMP,W          ;move TEMP to PORTD, first move TEMP to w then to
        MOVWF   PORTD           ;PORTD
        MOVLW   0XFF            ;now again fill COUNT with FF for the next waiting loop
        MOVWF   COUNT
        MOVF    RATEBAK,W       ;get the value of the RATE from the RATE bakup register
        MOVWF   RATE

LOOP1   BCF     INTCON,TOIF     ;loop1 do nothing but enable the interrupt again and get
        BSF     INTCON,GIE      ; you out of the INTHND subroutine
        BSF     INTCON,TOIE
        RETURN                  ;return from the INTHND subroutine
        END                     ;end of the program
```