Microcontrollers

PIC ARCHITECTURE & ASSEMBLY LANGUAGE PROGRAMMING

Dr. Jafar Jallad

Palestine Technical University – Kadoorie

Second semester

2019-2020

The WREG Register

- Many registers for arithmetic and logic operation.
- The WREG (WORking Register) Register is one of the most widely used registers of the PIC
  - 8-bit register → any data larger than 8 bits must be broken into 8-bits chunks before it is processed.
  - There is only one .

MOVLW

- Moves 8-bit data into WREG
  - MOVLW k; move literal value k into WREG
- Example
  - MOVLW 25H
  - MOVLW A5H

- Is the following code correct?
  - MOVLW 9H
  - MOVLW A23H

ADDLW

- ADDLW k; Add literal value k to WREG (k +WREG)

- Example:
  - MOVLW 12H
  - ADDLW 16H
    - ADDKW 11H
    - ADDLW 43H

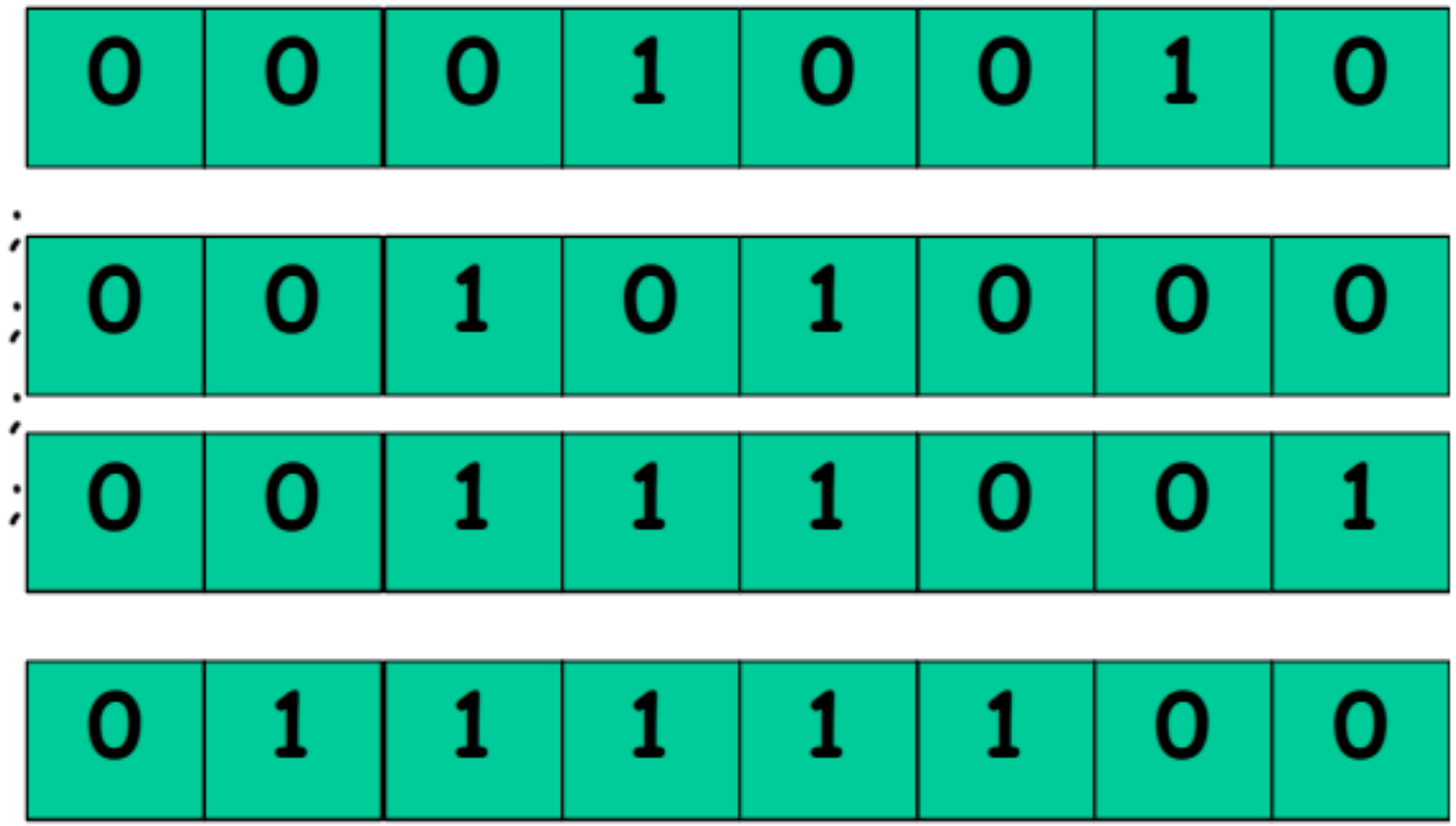
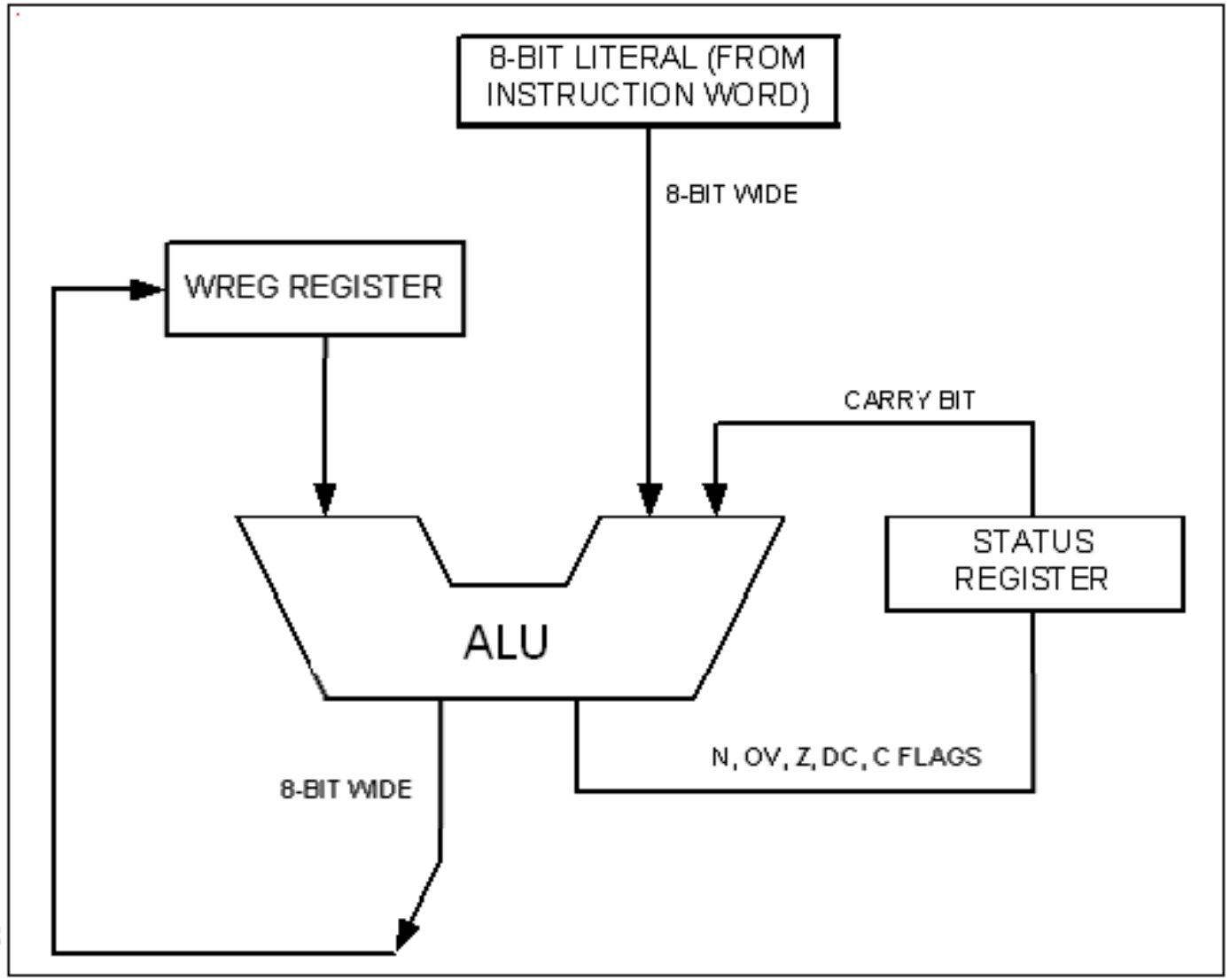
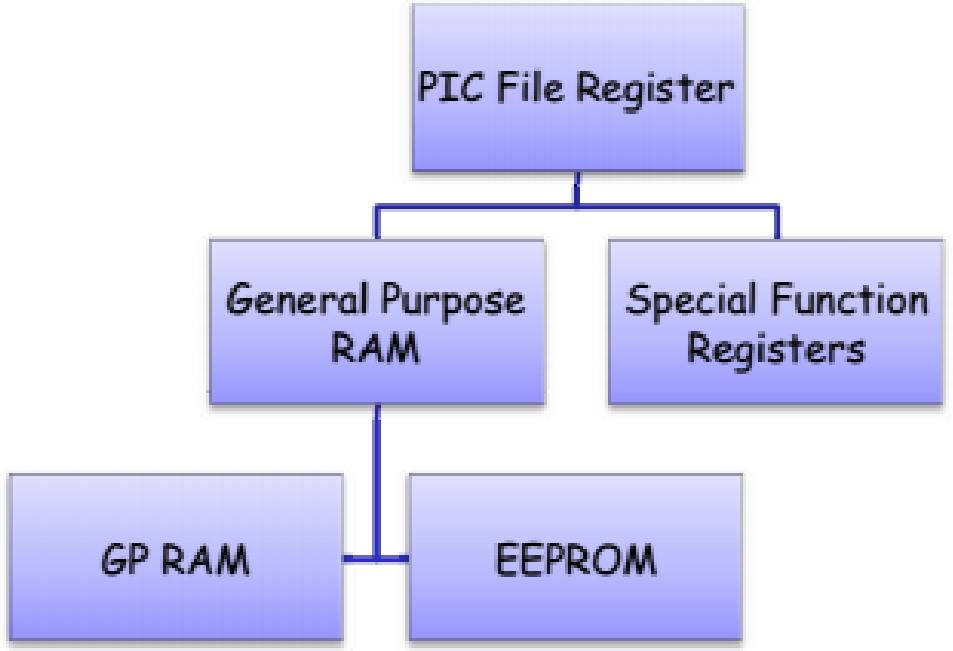


Figure 2-1. PIC WREG and ALU Using Literal Value



The PIC File Register

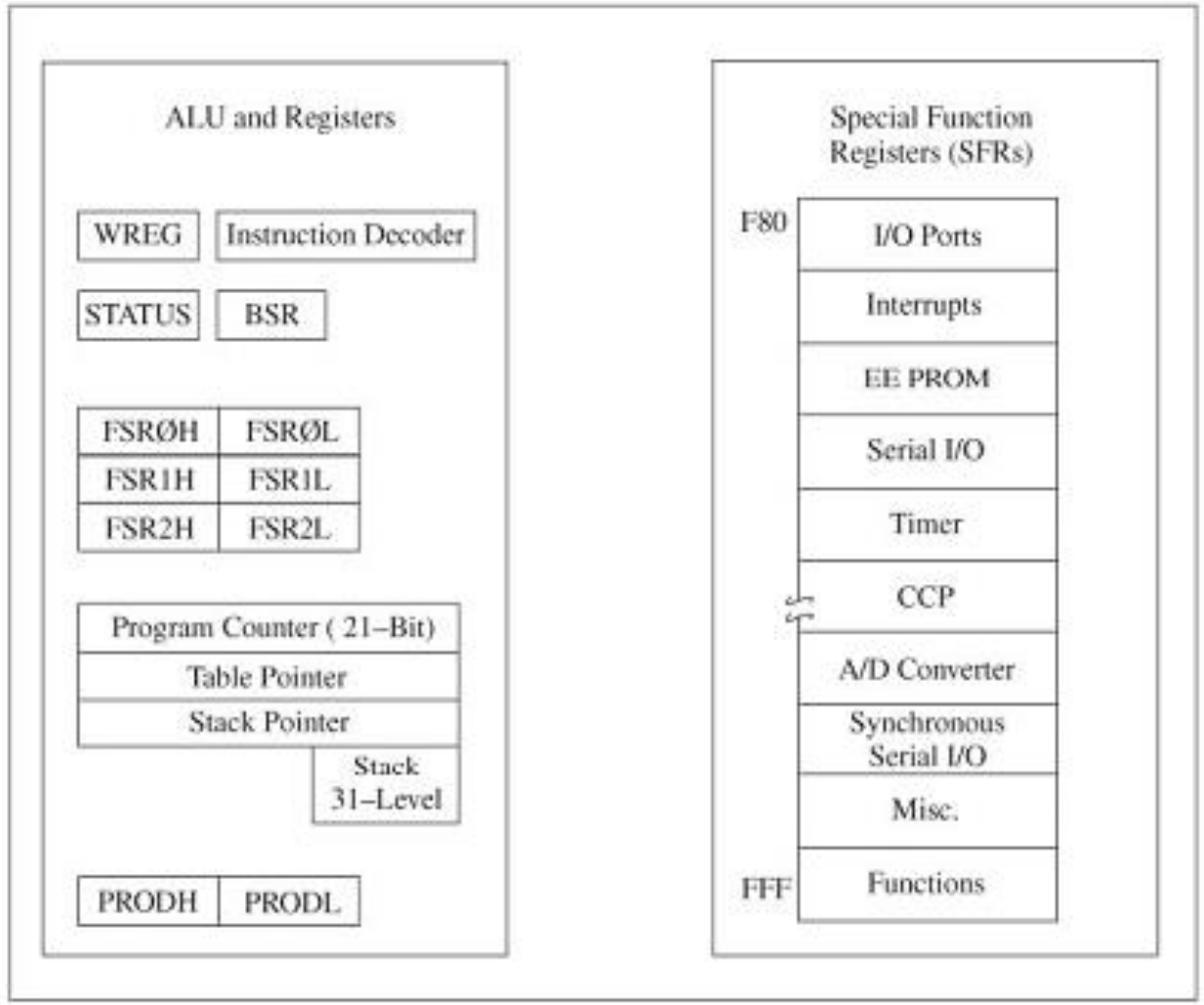
- It is the data memory.
  - Read/Write → Static RAM
  - Used for data storage, scratch pad and registers for internal use and function
  - 8-bit width



Special Function Registers

- dedicated to specific functions such as ALU status, timers, serial communication, I/O ports, ADC,…
- The function of each SFR is fixed by the CPU designer at the time of design
  - it is used for control of the microcontroller or peripheral
- 8-bit registers
- Their numbers varies from one chip to another.

PIC18F Programming Model



General Purpose RAM

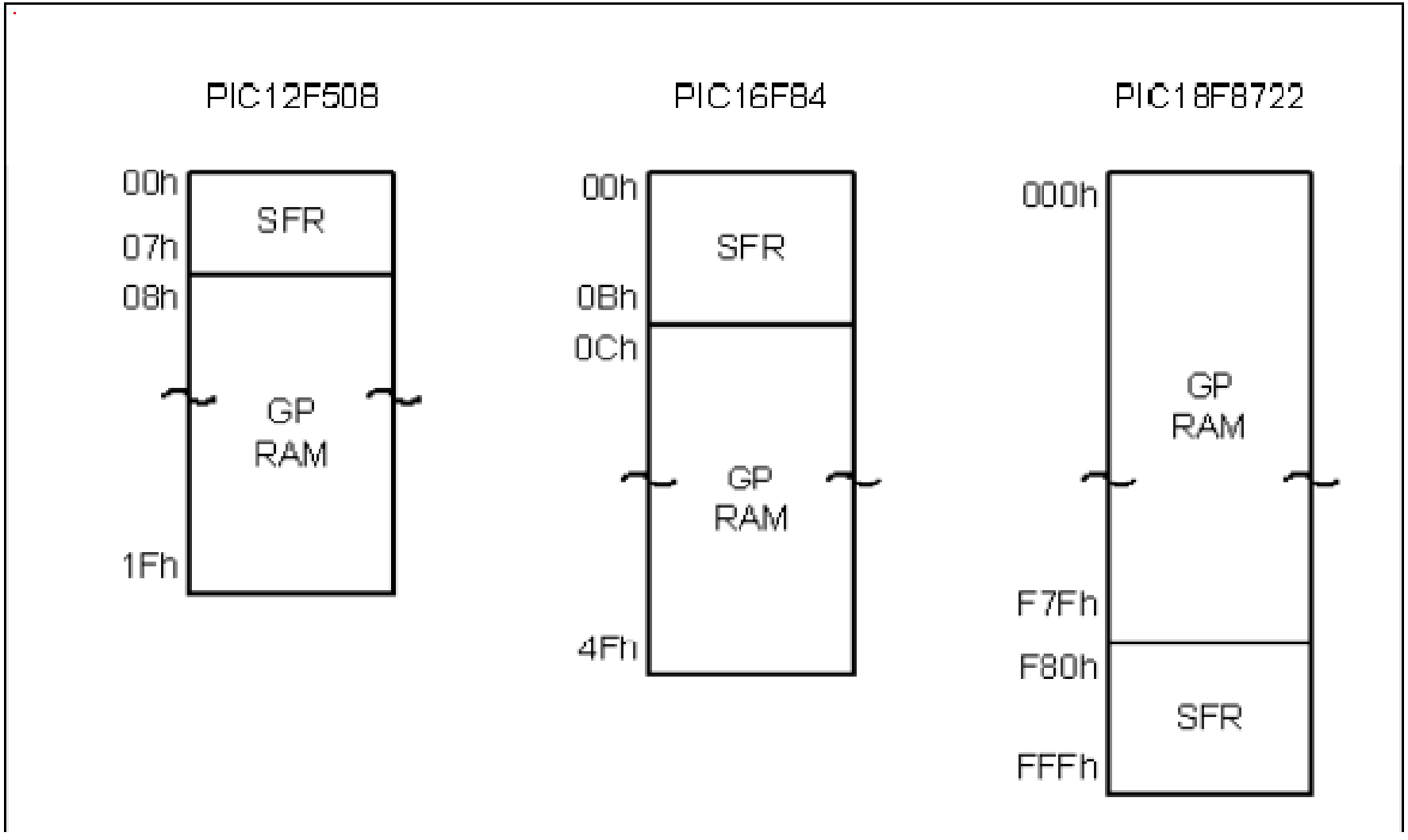
- Group of RAM locations
- 8-bit registers
- Larger than SFR
  - Difficult to manage them by using *Assembly* language
  - Easier to handle them by *C* Compiler.

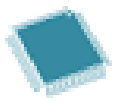
- The microchip website provides the data RAM size, which is the same as GPR size.

File Register Size

	File Register (Bytes)	=	SFR (Bytes)	+	GPR (Bytes)
PIC12F508	32		7		25
PIC16F84	80		12		68
PIC18F1220	512		256		256
PIC18F452	1792		256		1536
PIC18F2220	768		256		512
PIC18F458	1792		256		1536
PIC18F8722	4096		158		3938
PIC18F4550	2048		160		1888

Figure 2-2. File Registers of PIC12, PIC16, and PIC18



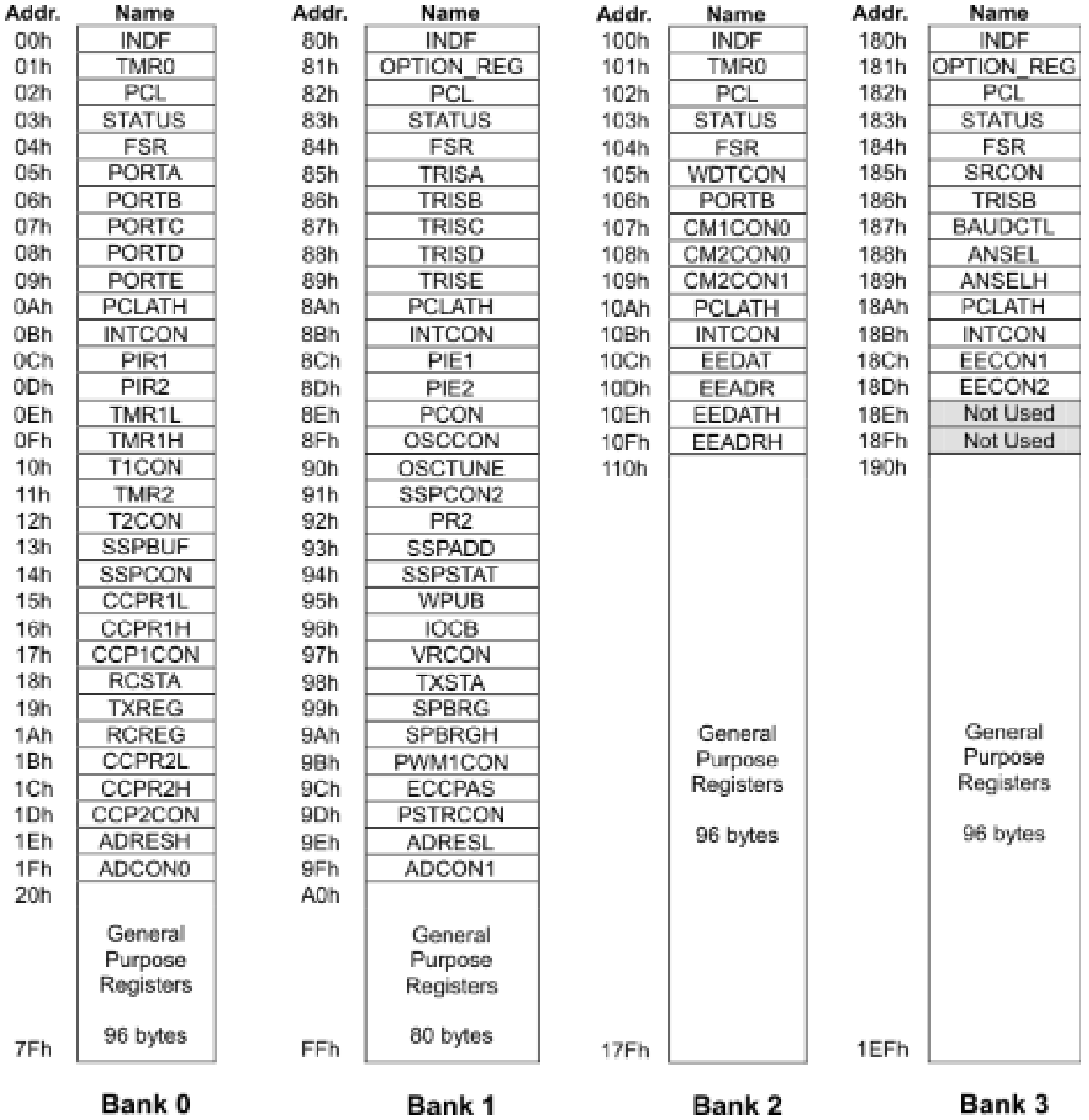


GP RAM vs. EEPROM in PIC chips

- GPRs are used by the CPU for internal data storage.
- EEPROM are considered as add-on memory that one can add externally to the ship.
- So PIC chip may have zero byte of EEPROM data memory, but impossible for a PIC have zero size for the file register.

GPRAM VS. EEPROM

- An add-on memory
- Can be zero size


Addr.
Name
00h INDF
01h TMR0
02h PCL
03h STATUS
04h FSR
05h PORTA
06h PORTB
07h PORTC
08h PORTD
09h PORTE
0Ah PCLATH
0Bh INTCON
0Ch PIR1
0Dh PIR2
0Eh TMR1L
0Fh TMR1H
10h T1CON
11h TMR2
12h T2CON
13h SSPBUF
14h SSPCON
15h CCPR1L
16h CCPR1H
17h CCP1CON
18h RCSTA
19h TXREG
1Ah RCREG
1Bh CCPR2L
1Ch CCPR2H
1Dh CCP2CON
1Eh ADRESH
1Fh ADCON0
20h
General Purpose Registers
96 bytes
7Fh
Bank 0
Addr.
Name
80h INDF
81h OPTION_REG
82h PCL
83h STATUS
84h FSR
85h TRISA
86h TRISB
87h TRISC
88h TRISD
89h TRISE
8Ah PCLATH
8Bh INTCON
8Ch PIE1
8Dh PIE2
8Eh PCON
8Fh OSCCON
90h OSCTUNE
91h SSPCON2
92h PR2
93h SSPADD
94h SSPSTAT
95h WPUB
96h IOCB
97h VRCON
98h TXSTA
99h SPBRG
9Ah SPBRGH
9Bh PWM1CON
9Ch ECCPAS
9Dh PSTRCON
9Eh ADRESL
9Fh ADCON1
A0h
General Purpose Registers
80 bytes
FFh
Bank 1
Addr.
Name
100h INDF
101h TMR0
102h PCL
103h STATUS
104h FSR
105h WDTCON
106h PORTB
107h CM1CON0
108h CM2CON0
109h CM2CON1
10Ah PCLATH
10Bh INTCON
10Ch EEDAT
10Dh EEADR
10Eh EEDATH
10Fh EEADRH
110h
General Purpose Registers
96 bytes
17Fh
Bank 2
Addr.
Name
180h INDF
181h OPTION_REG
182h PCL
183h STATUS
184h FSR
185h SRCON
186h TRISB
187h BAUDCTL
188h ANSEL
189h ANSELH
18Ah PCLATH
18Bh INTCON
18Ch EECON1
18Dh EECON2
18Eh Not Used
18Fh Not Used
190h
General Purpose Registers
96 bytes
1EFh
Bank 3

STATUS	R/W (0)	R/W (0)	R/W (0)	R (1)	R (1)	R/W (x)	R/W (x)	R/W (x)	Features
	IRP	RP1	RP0	TO	PD	Z	DC	C	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

- RP1,RP0 - Bits select register bank. They are used for direct addressing.

RP1	RP0	Active Bank
0	0	Bank0
0	1	Bank1
1	0	Bank2
1	1	Bank3

```
;BANK0
BCF STATUS,RP0; BCF STATUS,5
BCF STATUS,RP1 ; BCF STATUS,6
;BANK1
BSF STATUS,RP0
BCF STATUS,RP1
;BANK2
BCF STATUS,RP0
BSF STATUS,RP1
;BANK3
BSF STATUS,RP0
BSF STATUS,RP1
```

```
Macro Bank0
BCF STATUS,RP0;
BCF STATUS,RP1 ;
ENDM

Macro BANK1
BSF STATUS,RP0
BCF STATUS,RP1
ENDM

Macro BANK2
BCF STATUS,RP0
BSF STATUS,RP1
ENDM

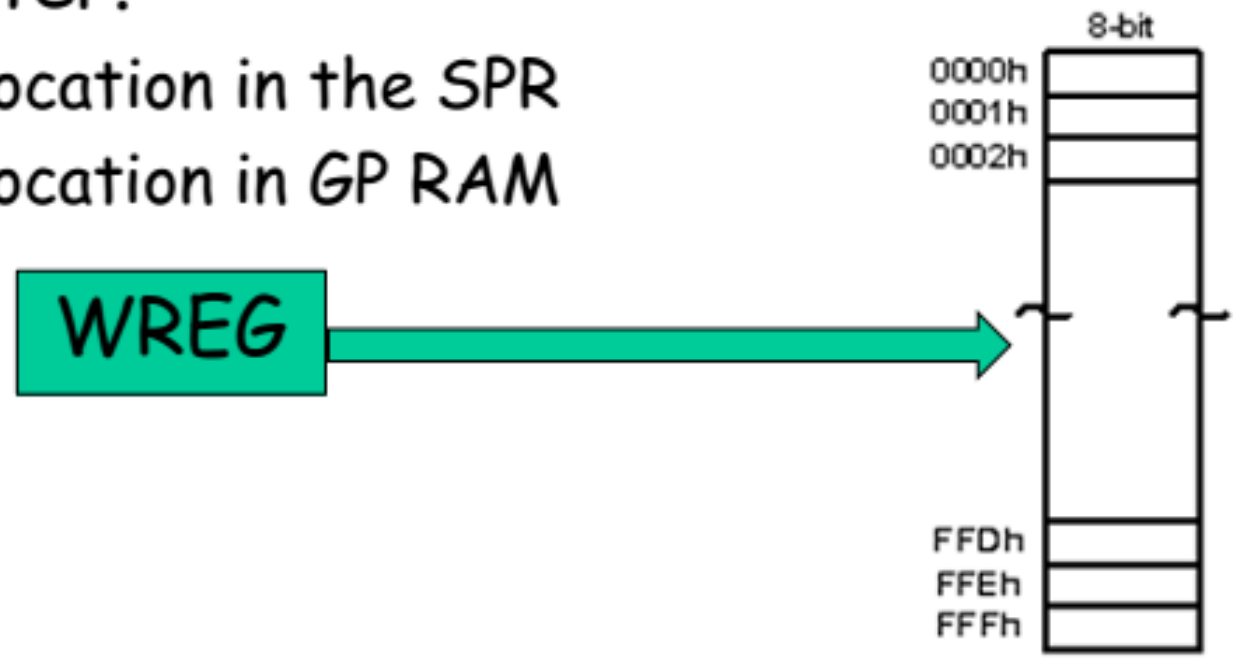
Macro BANK3
BSF STATUS,RP0
BSF STATUS,RP1
ENDM
```

MOVWF instruction

- F indicates for a file register

  MOVWF Address

- It tells the CPU to copy the source register, WREG, to a destination in the file register.
  - A location in the SPR
  - A location in GP RAM



Example 2-1

- MOVLW 99H
- MOVWF 12H
- MOVLW 85H
- MOVWF 13H
- MOVLW 3FH
- MOVWF 14H
- MOVLW 63H
- MOVWF 15H
- MOVLW 12H
- MOVWF 16H

99

85

3F

63

12

Address	Data
012H	
013H	
014H	
015H	
016H	

Address	Data
012H	99
013H	85
014H	3F
015H	63
016H	12

Note

- We cannot move literal values directly into the general purpose RAM location in the PIC16. They must be moved there via WREG.

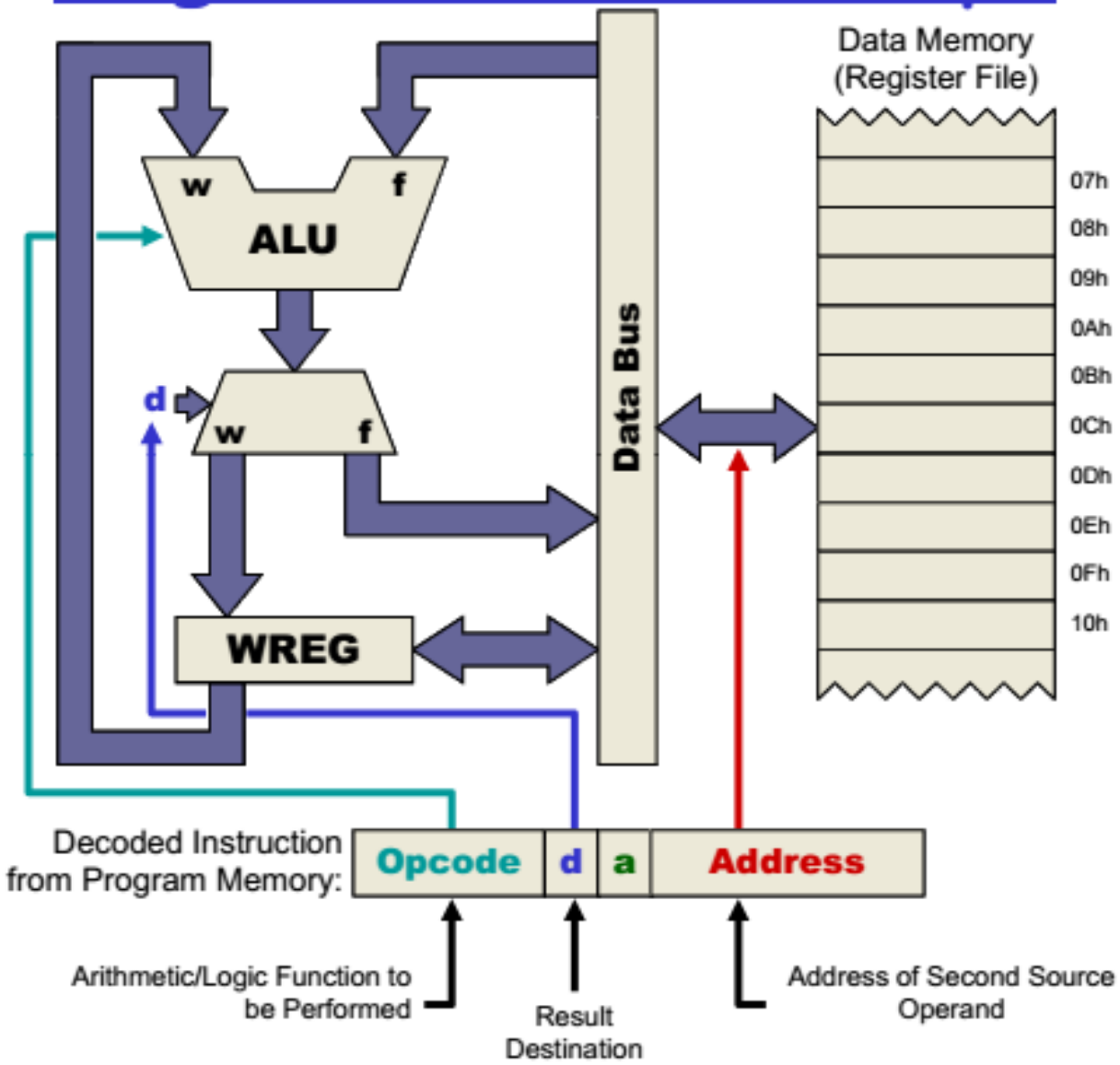
ADDWF

- Adds together the content of WREG and a file register location

  ADDWF File Reg. Address, D

- The result will be placed in either the WREG or in the file register location
  - D indicates the destination bit
- If D=0 or (D=w)
  - The result will be placed in the WREG
- If D=1 or (D=f)
  - The result will be placed in the file register

Register File Concept



- Register File Concept: All of data memory is part of the register file, so any location in data memory may be operated on directly
- All peripherals are mapped into data memory as a series of registers
- Orthogonal Instruction Set: ALL instructions can operate on ANY data memory location

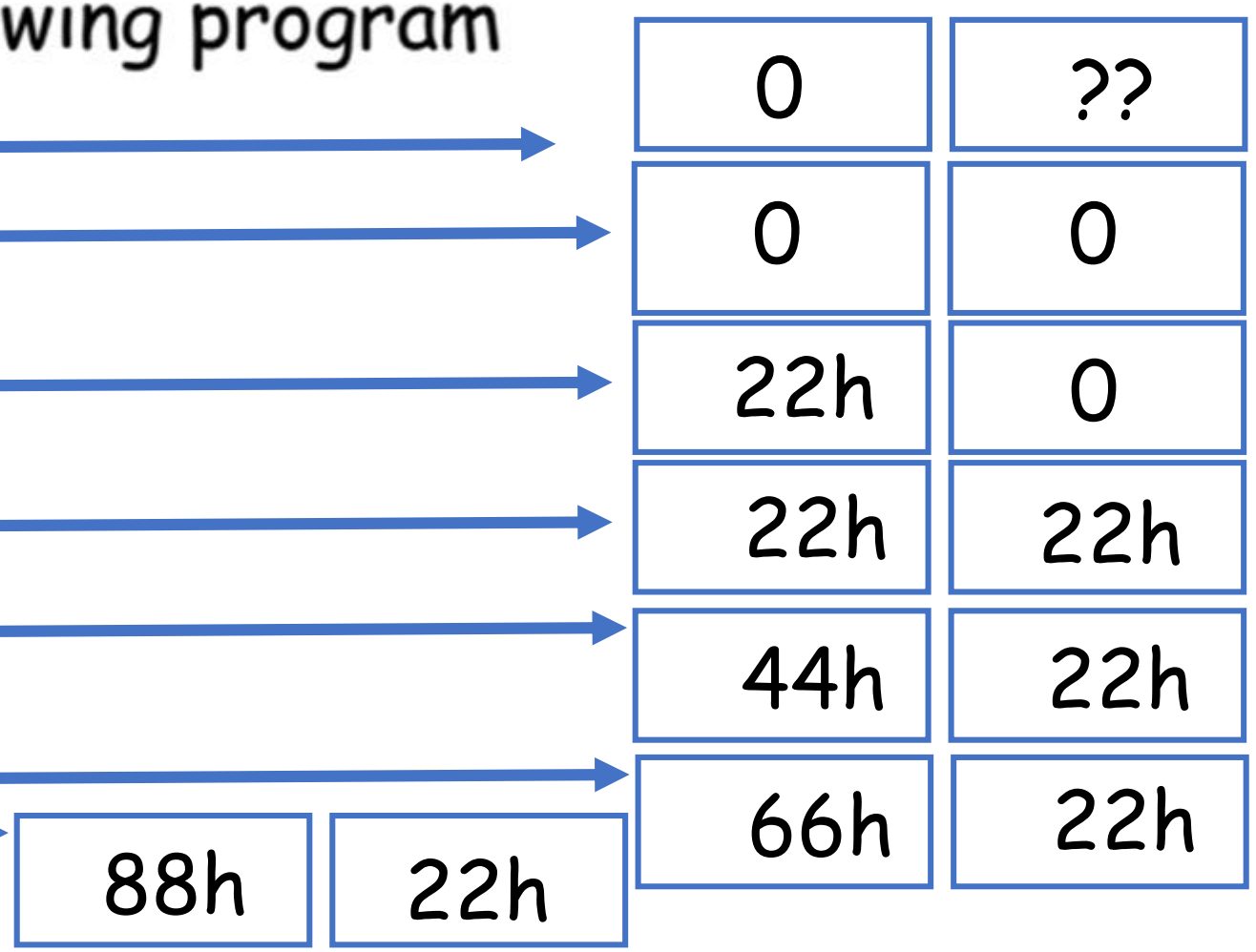
Example 2-2

- State the content of file register location and WREG after the following program

```
MOVLW 0          ;W=0H
MOVWF 12H        ;12H=0
MOVLW 22H        ;W=22H
ADDWF 12H, F     ;W=22H & 12H=22H
ADDWF 12H, F     ;W=22H & 12H=44H
ADDWF 12H, F     ;W=22H & 12H=66H
ADDWF 12H, F     ;W=22H & 12H=88H
```

Example 2-3

- State the content of file register location and WREG after the following program
  - MOVLW 0
  - MOVWF 12H
  - MOVLW 22H
  - ADDWF 12H, F
  - ADDWF 12H, W
  - ADDWF 12H, W
  - ADDWF 12H, W



COMF instruction

COMF File Reg. Address, D

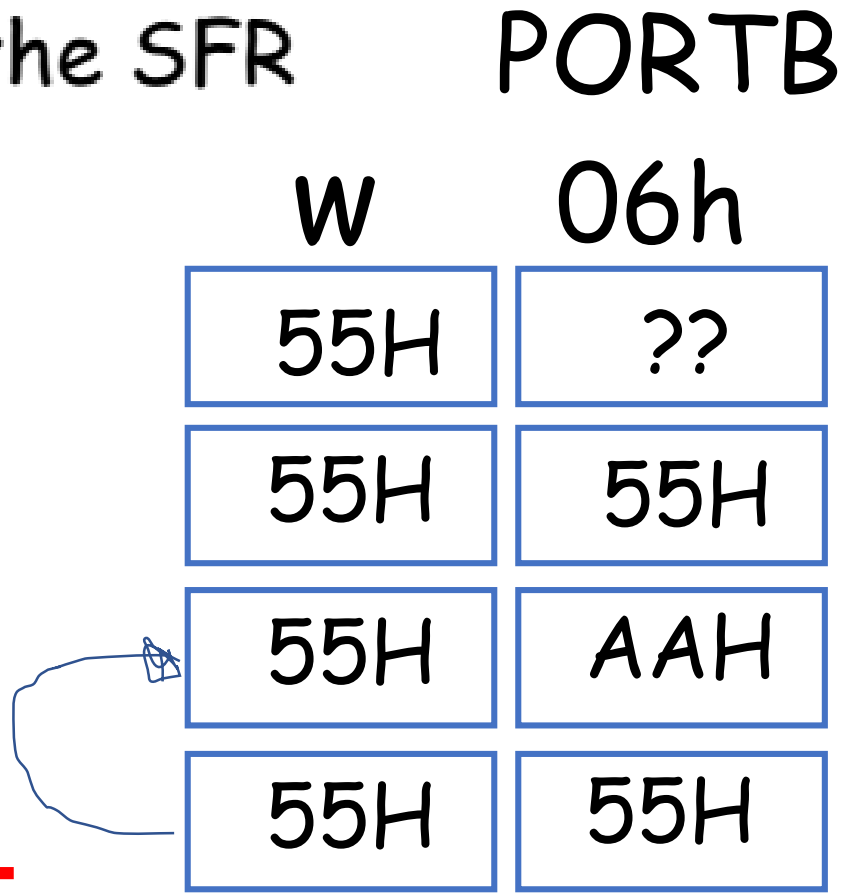
- It tells the CPU to complement the content of fileReg and places the results in WREG or in fileReg.

Example 2-4

- Write a simple program to toggle the SFR of Port B continuously forever.

Solution

```
     MOVLW   55H
     MOVWF   PORTB
B1   COMF    PORTB, F
     GOTO    B1
```



55H=0101 0101 B
AAH=1010 1010 B

DECF instruction

DECF File Reg. Address, D

- It tells the CPU to decrement the content of fileReg and places the results in WREG or in fileReg.
- Example:
  - MOVLW 3
  - MOVWF 20H
  - DECF 20H, F
  - DECF 20H, F
  - DECF 20H, F

W	20h
3H	??
3H	3H
3H	2H
3H	1H
3H	0H

DECF instruction

DECF File Reg. Address, D

- It tells the CPU to decrement the content of fileReg and places the results in WREG or in fileReg.
- Example:
  - MOVLW 3
  - MOVWF 20H
  - DECF 20H, w
  - DECF 20H, w
  - DECF 20H, w

W	20h
3H	??
3H	3H
2H	3H
2H	3H
2H	3H

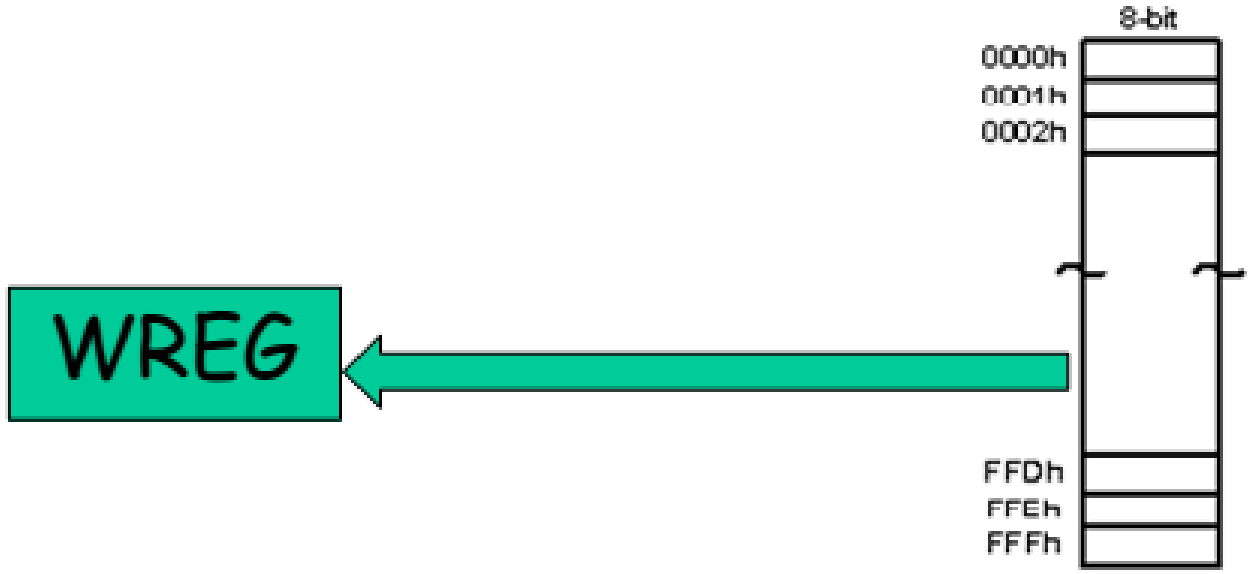
MOVF instruction

MOVF File Reg. Address, D

- It is intended to perform MOVFW
  - MOVFW isn't existed
- If D=0
  - Copies the content of fileReg (from I/O pin) to WREG
- If D=1
  - The content of the fileReg is copied to itself. (why?)

MOVF instruction

- MOVF File Reg. Address, 0



Example 2-5

- Write a simple program to get data from the SFRs of Port B and send it the SFRs of PORT C continuously.

Solution

```
AGAIN  MOVF  PORTB, W
       MOVWF PORTC
       GOTO  AGAIN
```

Example 2-6

- Write a simple program to get data from the SFRs of Port B Add the value 5 to it and send it the SFRs of PORT C
- Solution
- ```
  MOVF  PORTB,W
  ADDLW 05H
  MOVWF PORTC
  ```
  ```

PIC Status Register

- To indicate arithmetic conditions
- It is a 8-bit register
  - Five bits are used
- D0: C Carry Flag
- D1: DC Digital Carry Flag
- D2: Z Zero Flag

Example 2-8

- Show the status of the C, DC, Z flags after the following addition instruction
- MOVLW 38H
- ADDLW 2FH
- Solution
- 38H + 2FH = 67H → WREG=67H

  C=0

  DC=1

  Z=0

Example 2-9

- Show the status of the C, DC, Z flags after the following addition instruction
- MOVLW 9CH
- ADDLW 64H
- Solution
- 9CH + 64H = 100H → WREG= 00H

  C=1

  DC=1

  Z=1

PIC Data Format and Directives

- There is one data type
  - 8 bits
  - It is the job of the programmer to break down data larger 8 bits
- Data type can be positive or negative
- Data format are
  - Hex (default in PIC) 12 or 0x12 or H'12' or 12H
  - Binary B'00010010'
  - Decimal .12 or D'12'
  - ASCII A'c' or a'c'

Assembler Directives

- What is the difference between Instruction and Directives?
- EQU
  - Defines a constant or fixed address
- ORG (Origin)
- END

Rules for labels

- Unique name
- Alphabetic letters
  - Upper, lower, digits (0-9),special char. (? . @_ $)
- The first letter must be Alphabetic letters
- Not a reserved word

Introduction to PIC Assembly Language

- Difficult for us to deal with the machine code (0s and 1s)
- Assembly Language provide
  - Mnemonic: codes and abbreviations that are easy to remember
  - Faster programming and less prone error
  - Programmer must know all Reg. …etc.
- Assembler is used to translate the assembly code into machine code (object code)

Structure of Assembly Language

- Series of lines
  - Instruction
  - Directives
- Consists of four field

[label] mnemonic [operands] [;commands]

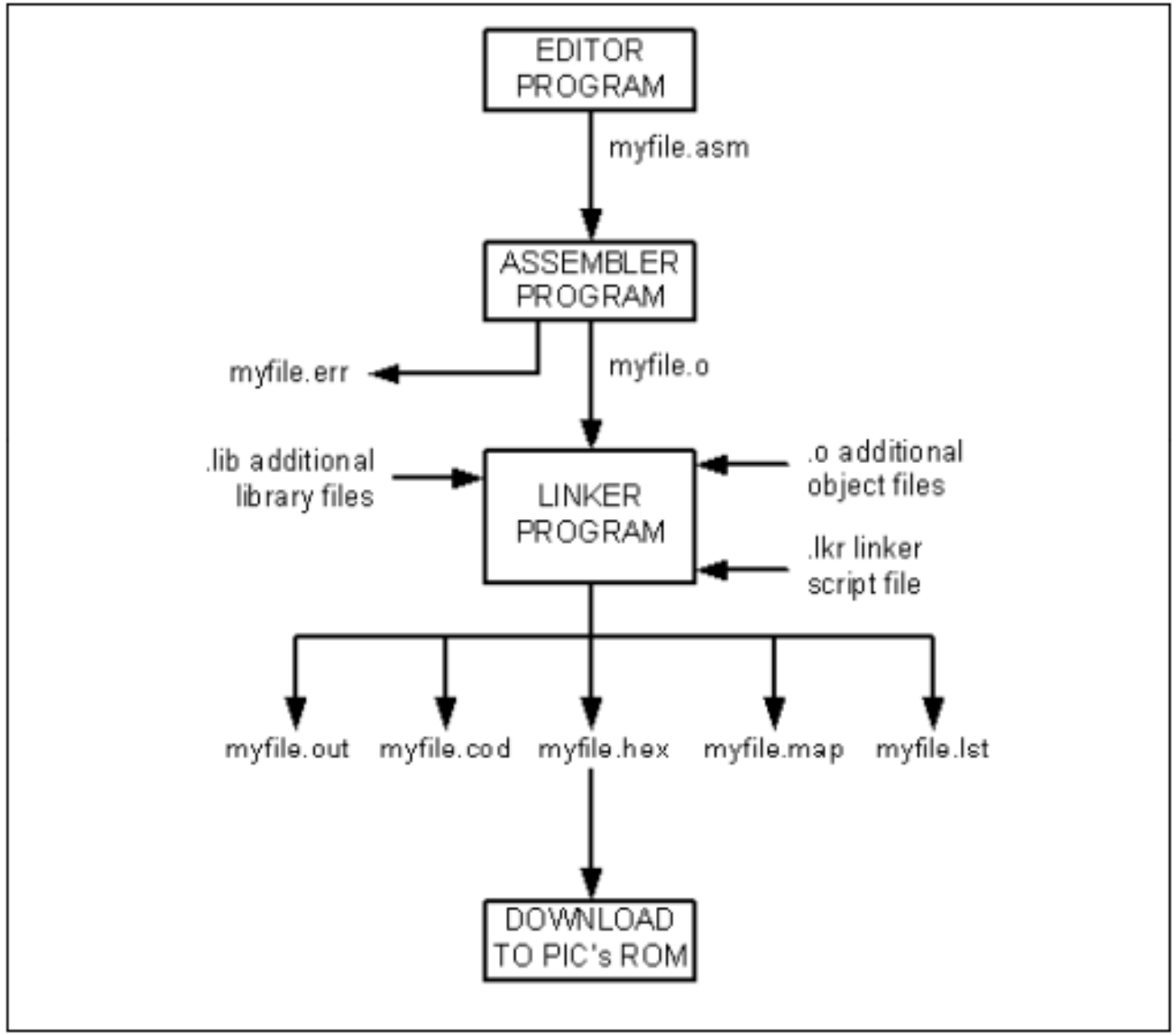
- Label: refer to line by code (certain length)
- Mnemonic and operands are task that should be executed.
  - Directive don't generate any machine code and used by assembler

Sample of Assembly Language Program

```
SUM  EQU 10H ;RAM loc 10H fro SUM
     ORG 0H; start at address 0
     MOVLW 25H ; WREG = 25
     ADDLW 0x34 ;add 34H to WREG=59H
     ADDLW 11H ;add 11H to WREG=6AH
     ADDLW D'18' ; W = W+12H=7CH
     ADDLW 1CH ; W = W+1CH=98H
     ADDLW b'00000110' ; W = W+6H=9EH
     MOVWF SUM ;save the result in SUM location
HERE GOTO HERE ;stay here forever
     END            ; end of asm source file
```

Assembling and Linking A PIC Program

Figure 2-8. Steps to Create a Program

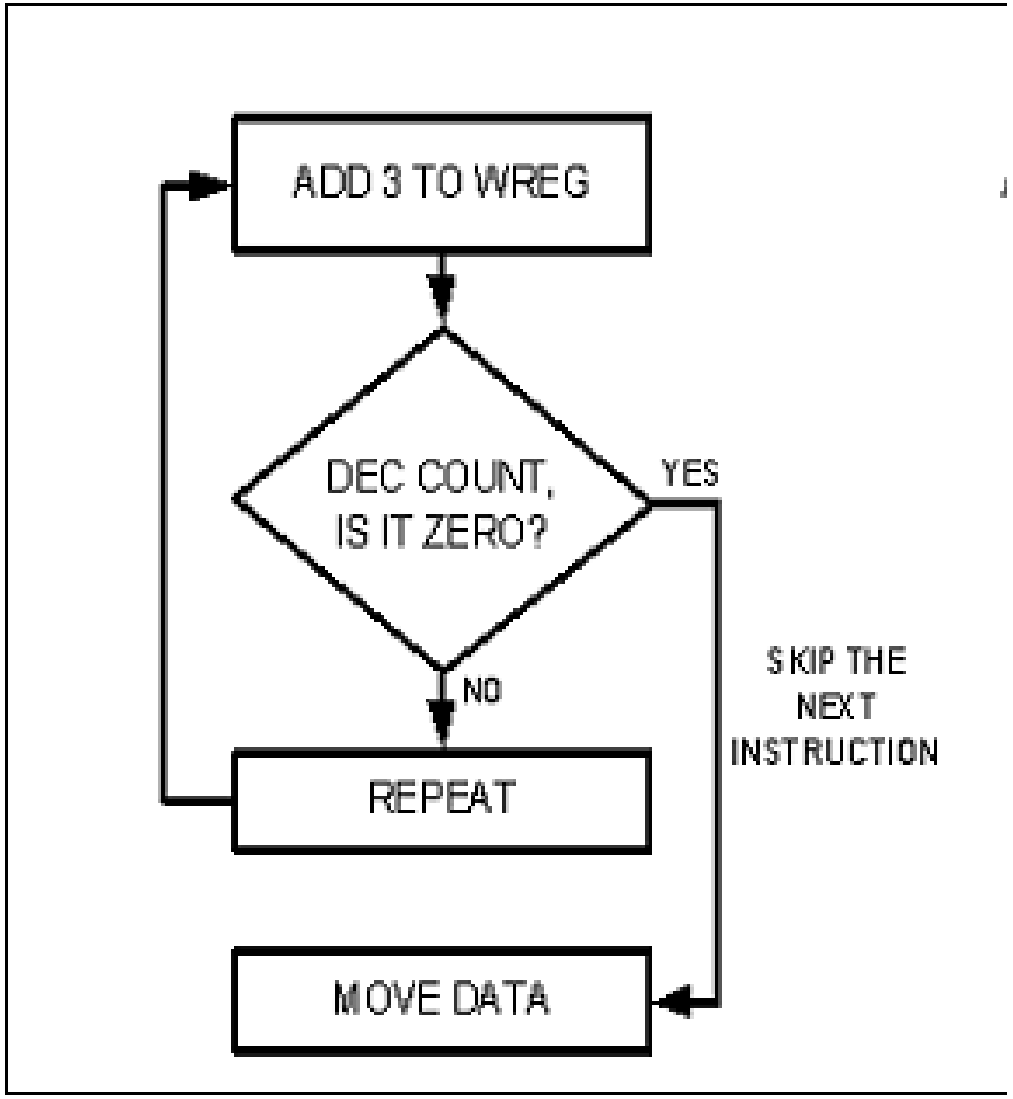


Looping in PIC

- Repeat a sequence of instructions or a certain number of times
- Two ways to do looping
  - Using DECFSZ instruction

DECFSZ instruction

- Decrement file register, skip the next instruction if the result is equal 0
- DECFSZ fileRef, d
- GOTO instruction follows DECFSZ



Example 3-1

- Write a program to
  a) Clear WREG
  b) Add 3 to WREG ten times and place the result in SFR PORTB

- Solution

```
COUNT   EQU  0x25
        MOVLW  d'10'
        MOVWF  COUNT
        MOVLW  0
AGAIN   ADDLW  3
        DECFSZ COUNT,F
        GOTO   AGAIN
        MOVWF  PORTB
```

Example 3-3

- What is the maximum number of times that the loop can be repeated?
- All locations in the FileReg are 8-bit
- The max. loop size is 255 time

Loop inside a loop

- Write a program to

a) Load the PORTB SFR register with the value 55H

b) Complement PORTB 700 times

Solution

```
R1  EQU 0X25
R2 EQU 0X26
COUNT_1 EQU .70
COUNT_2 EQU .10
        MOVLW 0X55
        MOVWF PORTB
        MOVLW COUNT_2
        MOVWF R2
LOP_2   MOVLW COUNT_1
        MOVWF R1
LOP_1   COMPF PORTB,F
        DECFSZ R1,F
        GOTO LOP_1
        DECFSZ R2,F
        GOTO LOP_2
```

- Write a program to

a) Load the PORTB SFR register with the value 55H

b) Complement PORTB 100,000 times

Solution

```
R1  EQU 0X25
R2 EQU 0X26
R3 EQU 0X27
COUNT_1  EQU .100
COUNT_2 EQU .100
COUNT_3 EQU .10
                MOVLW 0X55
                MOVWF PORTB
                MOVLW COUNT_3
                MOVWF R3
LOP_3           MOVLW COUNT_2
                MOVWF R2
LOP_2           MOVLW COUNT_1
                MOVWF R1
LOP_1           COMF PORTB,F
                DECFSZ R1,F
                GOTO LOP_1
                DECFSZ R2,F
                GOTO LOP_2
                DECFSZ R3,F
                GOTO LOP_3
```

BTFSC and BTFSS

BTFSC == Bit Test FileReg Skip if Clear
BTFSS == Bit Test FileReg Skip if Set

EX1:

```
BTFSS FileReg, 7
Inst.Set1
Inst.Set2
   .
   .
```

If D7=0 ➔ Inst.Set1 then Inst.Set2 are executed sequentially.
If D7=1 ➔ Inst.Set2 is executed only.

EX2:

```
BTFSC FileReg, 7
Inst.Set1
Inst.Set2
   .
   .
```

If D7=1 ➔ Inst.Set1 then Inst.Set2 are executed sequentially.
If D7=0 ➔ Inst.Set2 is executed only.

Example 3-5

- Write a program to determine if the loc. 0x30 contains the value 0. if so, put 55H in it.
- Solution:


```
MYLOC  EQU 0X30

       MOVF MYLOC,F
       BTFSS STATUS,Z
       GOTO Next
       MOVLW 0x55
       MOVWF MYLOC
Next
       .
       .
       .
```

GOTO to itself

- Label and $ can be used to keep uC busy (jump to the same location)
- HERE GOTO HERE
- GOTO $

Instruction Cycle time for the PIC

- What is the Instruction Cycle ?
- Most instructions take one or tow cycles
  - BTFSS can take up to **2** cycles
- Instruction Cycle depends on the freq. of oscillator
- Clock source: Crystal oscillator and on-chip circuitry
- One instruction cycle consists of four oscillator period.

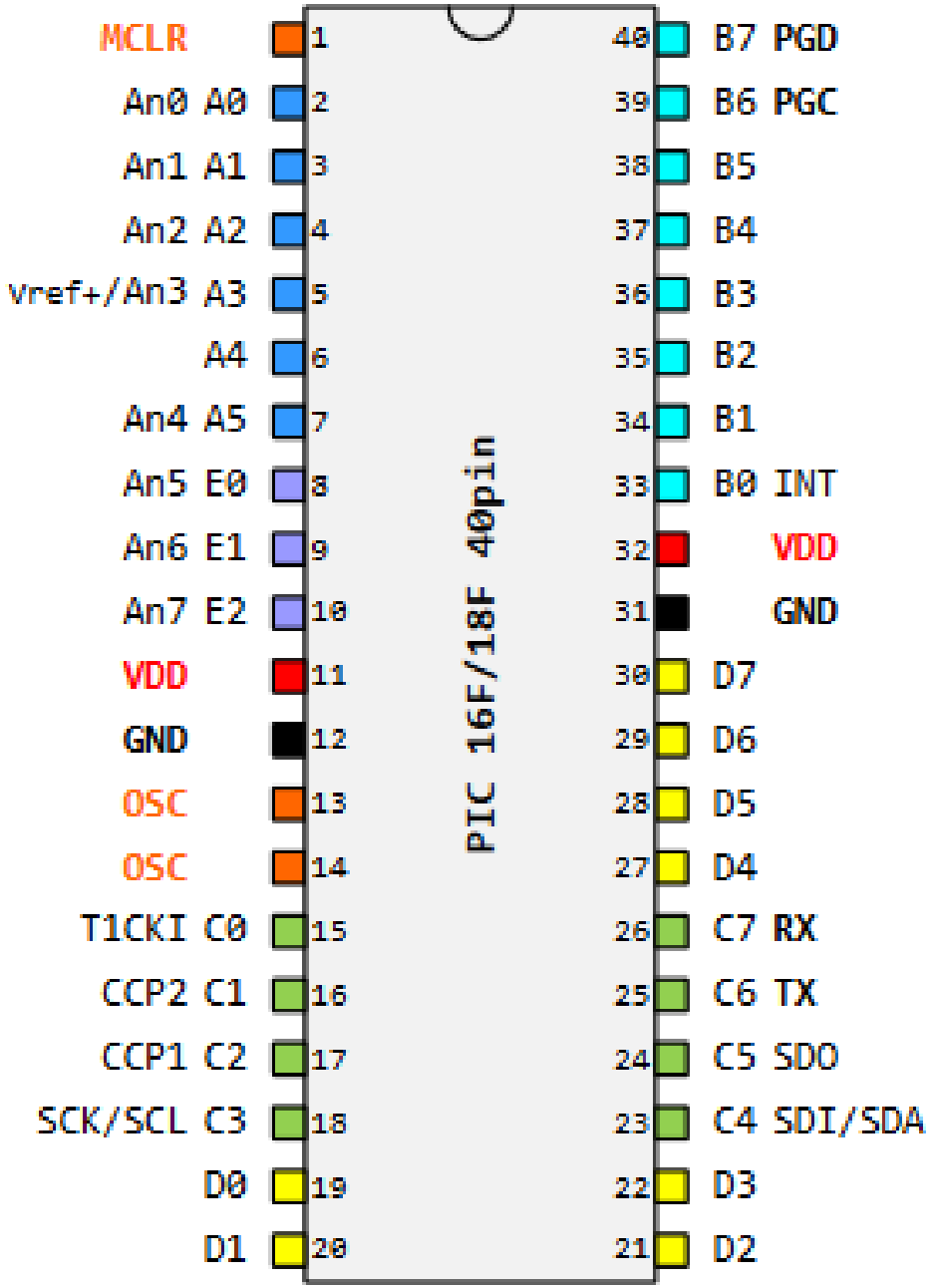
Example 3-14

- Find the period of the instruction cycle you chose 4 MHz crystal?

- Solution
- 4 MHz/4 =1 MHz
- Instruction Cycle = 1/1MHz = 1 usec

I/O Port Programming in PIC18

- PIC18 has many ports
  - Depending on the family member
  - Depending on the number of pins on the chip
  - Each port can be configured as input or output.
    - Bidirectional port
  - Each port has some other functions
    - Such as timer , ADC, interrupts and serial communication
  - Some ports have 8 bits, while others have not


MCLR 1
An0 A0 2
An1 A1 3
An2 A2 4
Vref+/An3 A3 5
A4 6
An4 A5 7
An5 E0 8
An6 E1 9
An7 E2 10
VDD 11
GND 12
OSC 13
OSC 14
T1CKI C0 15
CCP2 C1 16
CCP1 C2 17
SCK/SCL C3 18
D0 19
D1 20
PIC 16F/18F 40pin
40 B7 PGD
39 B6 PGC
38 B5
37 B4
36 B3
35 B2
34 B1
33 B0 INT
32 VDD
31 GND
30 D7
29 D6
28 D5
27 D4
26 C7 RX
25 C6 TX
24 C5 SDO
23 C4 SDI/SDA
22 D3
21 D2

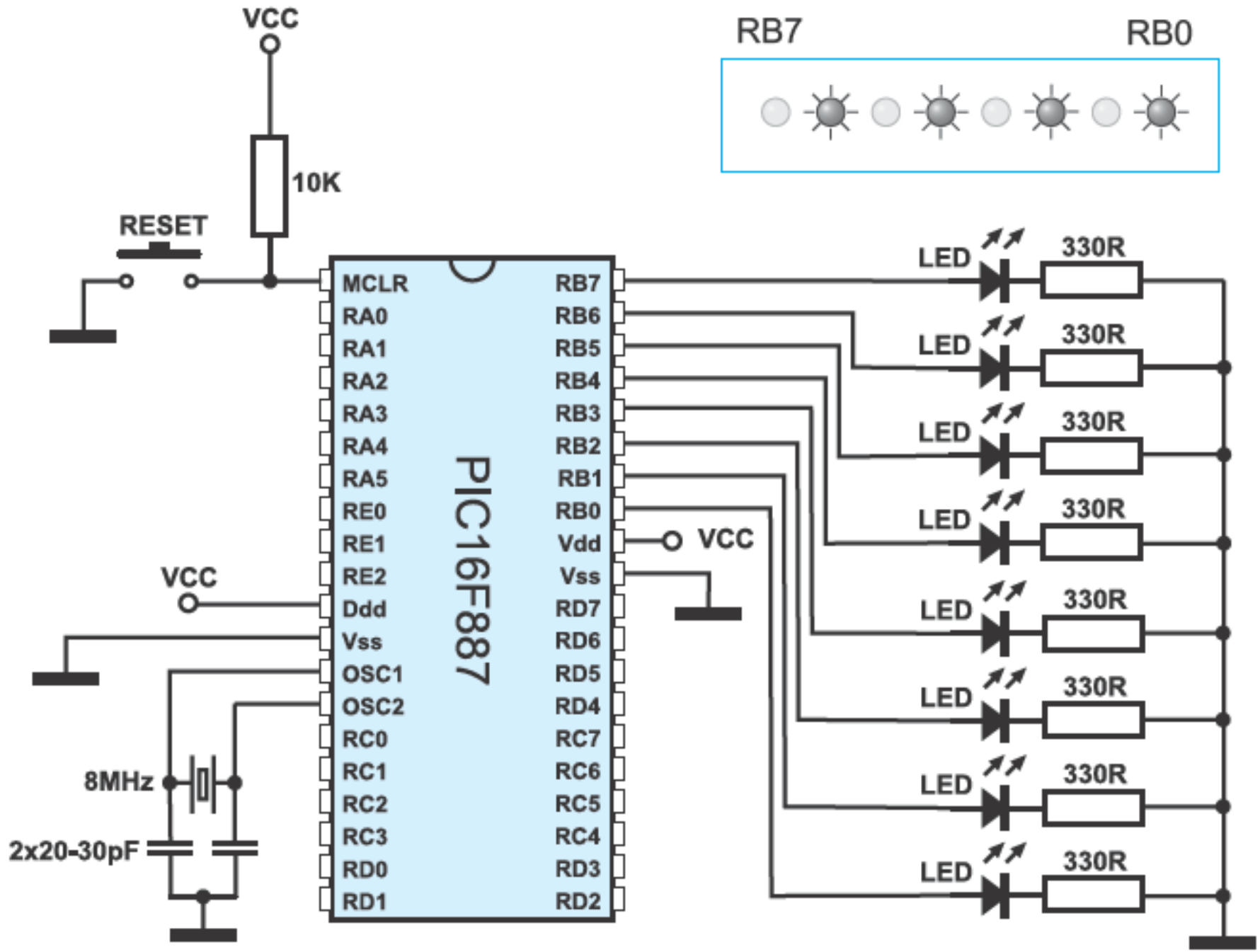
Port A

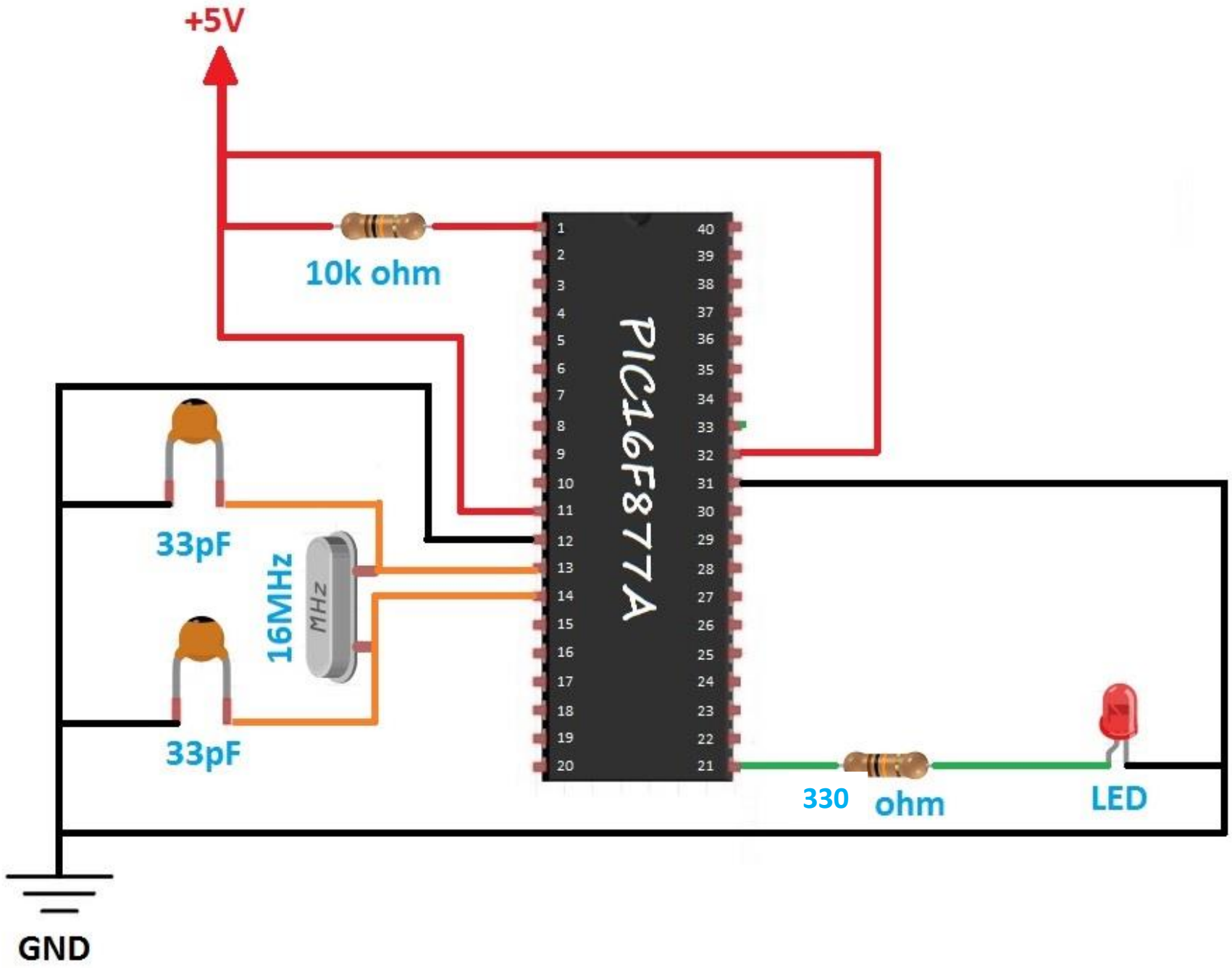
- PORTA is a 6-bit wide, bidirectional port.

- The corresponding Data Direction register is TRISA.
- Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input
- Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output
- On a Power-on Reset, these pins are configured as inputs and read as '0'.

PORT B, PORT C, PORT D and PORT E

- PORTB is 8 pins
- PORTC is 8 pins
- PORTD is 8 pins
- PORTE is 3 pins


VCC
10K
RESET
RB7
RB0
MCLR
RA0
RA1
RA2
RA3
RA4
RA5
RE0
RE1
RE2
Ddd
Vss
OSC1
OSC2
RC0
RC1
RC2
RC3
RD0
RD1
PIC16F887
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
Vdd
Vss
RD7
RD6
RD5
RD4
RC7
RC6
RC5
RC4
RD3
RD2
VCC
VCC
8MHz
2x20-30pF
LED
330R
LED
330R
LED
330R
LED
330R
LED
330R
LED
330R
LED
330R
LED
330R

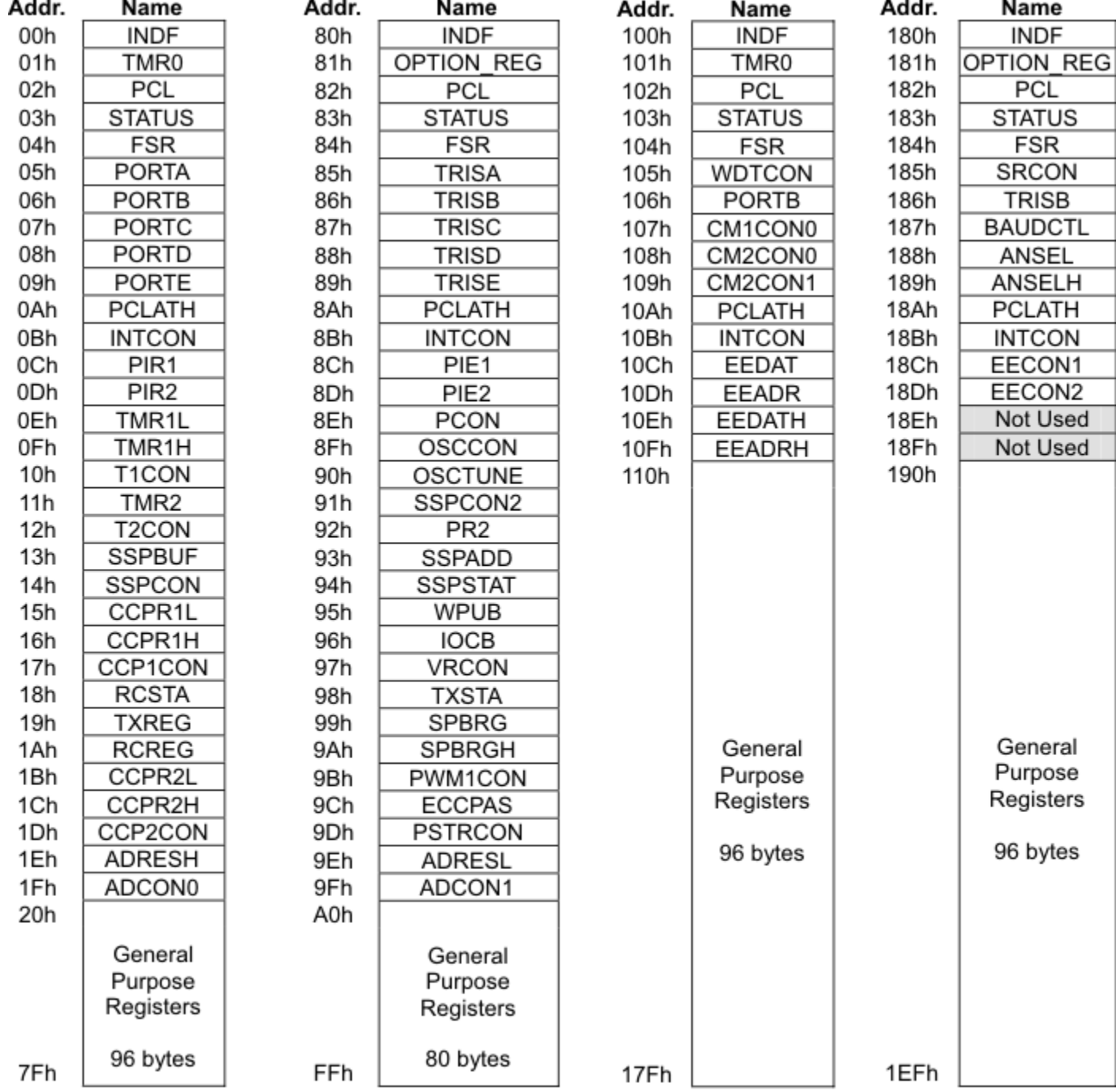

+5V
10k ohm
PIC16F877A
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
33pF
16MHz
33pF
330 ohm
LED
GND

I/O SFR

- Each port has three registers for its operation:
  - TRIS register (Data Direction register)
    - If the corresponding bit is 0 → Output
    - If the corresponding bit is 1 → Input
  - PORT register (reads the levels on the pins of the device)

TRISA→BANK1
PORTA→BANK0

Output→0
Input→1


Addr.
Name
00h INDF
01h TMR0
02h PCL
03h STATUS
04h FSR
05h PORTA
06h PORTB
07h PORTC
08h PORTD
09h PORTE
0Ah PCLATH
0Bh INTCON
0Ch PIR1
0Dh PIR2
0Eh TMR1L
0Fh TMR1H
10h T1CON
11h TMR2
12h T2CON
13h SSPBUF
14h SSPCON
15h CCPR1L
16h CCPR1H
17h CCP1CON
18h RCSTA
19h TXREG
1Ah RCREG
1Bh CCPR2L
1Ch CCPR2H
1Dh CCP2CON
1Eh ADRESH
1Fh ADCON0
20h
General Purpose Registers
96 bytes
7Fh
Bank 0
Addr.
Name
80h INDF
81h OPTION_REG
82h PCL
83h STATUS
84h FSR
85h TRISA
86h TRISB
87h TRISC
88h TRISD
89h TRISE
8Ah PCLATH
8Bh INTCON
8Ch PIE1
8Dh PIE2
8Eh PCON
8Fh OSCCON
90h OSCTUNE
91h SSPCON2
92h PR2
93h SSPADD
94h SSPSTAT
95h WPUB
96h IOCB
97h VRCON
98h TXSTA
99h SPBRG
9Ah SPBRGH
9Bh PWM1CON
9Ch ECCPAS
9Dh PSTRCON
9Eh ADRESL
9Fh ADCON1
A0h
General Purpose Registers
80 bytes
FFh
Bank 1
Addr.
Name
100h INDF
101h TMR0
102h PCL
103h STATUS
104h FSR
105h WDTCON
106h PORTB
107h CM1CON0
108h CM2CON0
109h CM2CON1
10Ah PCLATH
10Bh INTCON
10Ch EEDAT
10Dh EEADR
10Eh EEDATH
10Fh EEADRH
110h
General Purpose Registers
96 bytes
17Fh
Bank 2
Addr.
Name
180h INDF
181h OPTION_REG
182h PCL
183h STATUS
184h FSR
185h SRCON
186h TRISB
187h BAUDCTL
188h ANSEL
189h ANSELH
18Ah PCLATH
18Bh INTCON
18Ch EECON1
18Dh EECON2
18Eh Not Used
18Fh Not Used
190h
General Purpose Registers
96 bytes
1EFh
Bank 3

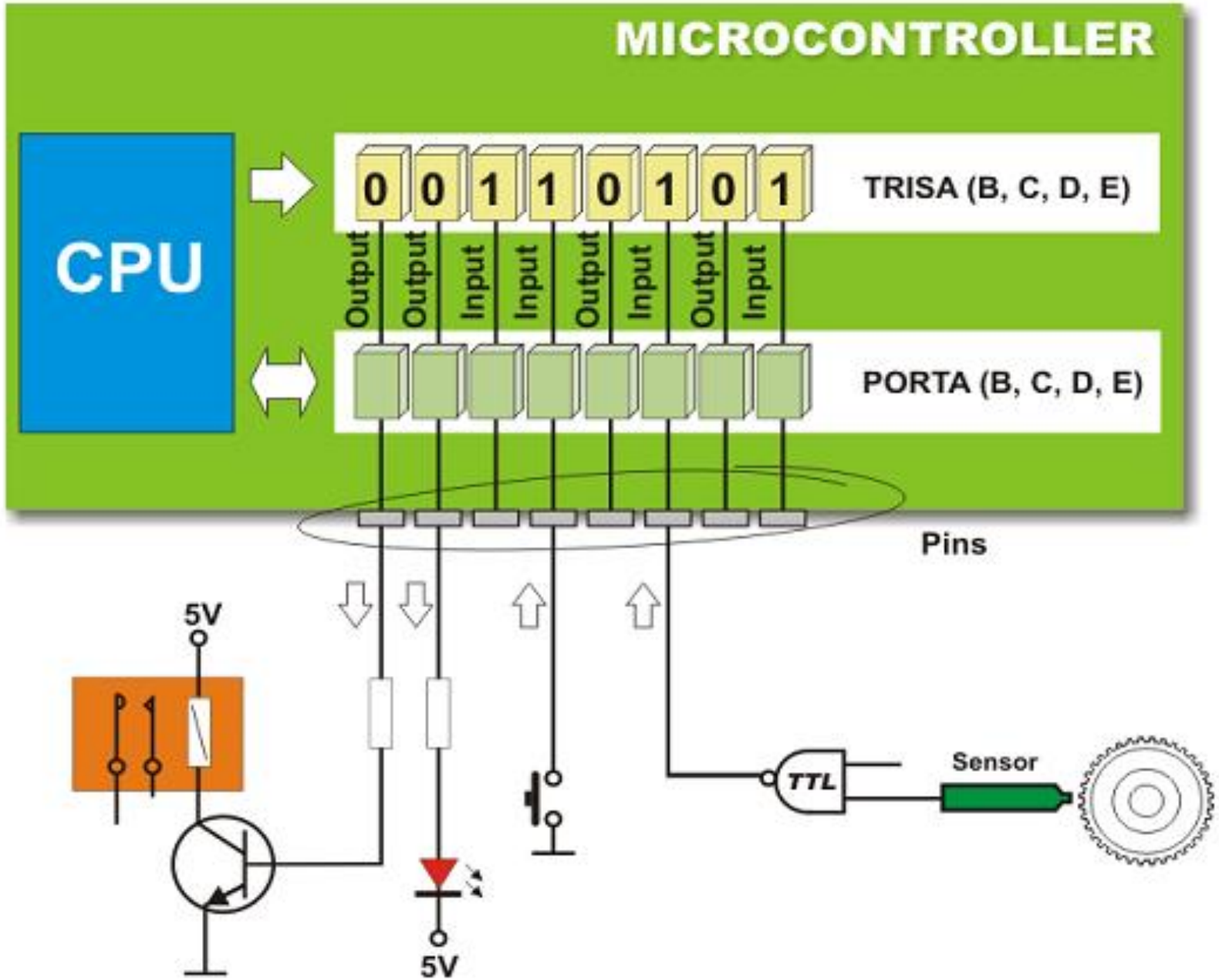


Fig. 3-1 I/O Ports

I/O Bit Manipulation Programming

- I/O ports and bit-addressability
- Monitoring a single bit
- Reading a single bit

Read followed by write operation

- Be careful
  - Don't have a two I/O operations one right after the other.
- Data Dependency
  - A NOP is needed to make that data is written into WREG before it read for outputting to PortB.

```
BANK1
CLRF TRISB
MOVLW .255
MOVWF TRISC
BANK0
L4
MOVF PORTC,W
NOP
MOVWF PORTB
GOTO L4
```

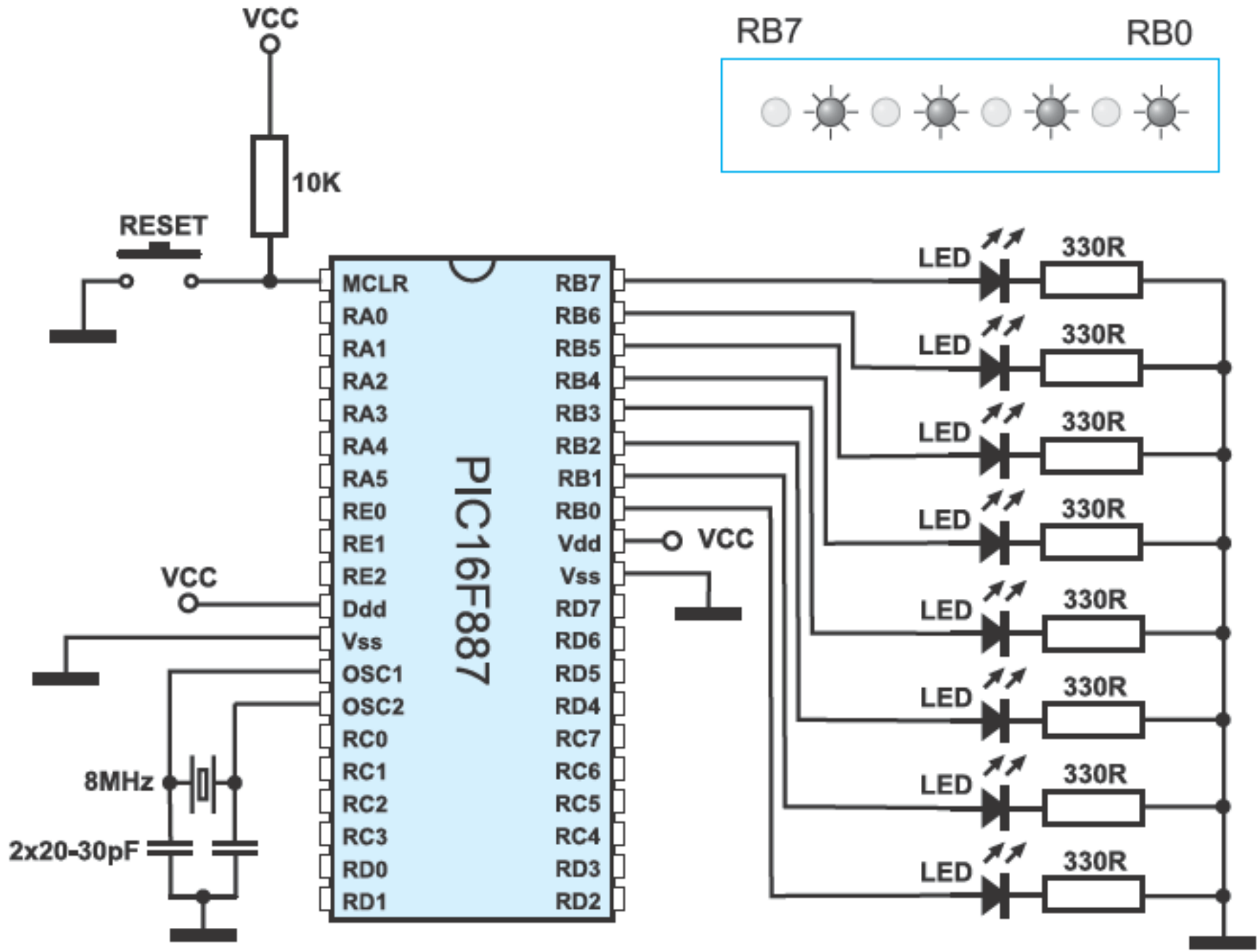
Example

Write a test program for the PIC chip to toggle all the bits of PORTB, PORTC and PORTD every 0.2 of a second. (suppose that there is a 4 MHz)

```
List p=PIC16F877A
#include P16F877A.INC
R1 EQU 0x20
R2 EQU 0X21
ORG 0
Macro BANK0
BCF STATUS,RP0
BCF STATUS,RP1
ENDM
Macro BANK1
BSF STATUS,RP0
BCF STATUS,RP1
ENDM
```

```
BANK1
CLRF TRISB
CLRF TRISC
CLRF TRISD
BANK0
MOVLW 0X55
MOVWF PORTB
MOVWF PORTC
MOVWF PORTD
L3
COMF PORTB,F
COMF PORTC,F
COMF PORTD,F
CALL QDELAY
GOTO L3
```

```
QDELAY
MOVLW .200
MOVWF R2
D2
MOVLW .250
MOVWF R1
D1
NOP
DECFSZ R1,F
GOTO D1
DECFSZ R2,F
GOTO D2
RETURN
END
```


VCC
10K
RESET
RB7
RB0
MCLR
RA0
RA1
RA2
RA3
RA4
RA5
RE0
RE1
RE2
Ddd
Vss
OSC1
OSC2
RC0
RC1
RC2
RC3
RD0
RD1
PIC16F887
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
Vdd
Vss
RD7
RD6
RD5
RD4
RC7
RC6
RC5
RC4
RD3
RD2
VCC
VCC
8MHz
2x20-30pF
LED
330R
LED
330R
LED
330R
LED
330R
LED
330R
LED
330R
LED
330R
LED
330R

```
;BANK1
BSF STATUS,RP0
BCF STATUS,RP1
MOVLW .0
MOVWF TRISB;CLRF TRISB
;BANK0
BCF STATUS,RP0
BCF STATUS,RP1
MOVLW .255; B'11111111'=0XFF=D'255'
MOVWF PORTB
END
```

Flashing LEDs

```
R1 EQU 0X22
R2 EQU 0X23
;BANK1
BSF STATUS,RP0
BCF STATUS,RP1
MOVLW .0
MOVWF TRISB;CLRF TRISB
;BANK0
BCF STATUS,RP0
BCF STATUS,RP1
START
MOVLW .255; B'11111111'=0XFF=D'255'
MOVWF PORTB
CALL DELAY
MOVLW .0; B'00000000'=0X00=D'0'
MOVWF PORTB
CALL DELAY
GOTO START
```

```
DELAY
MOVLW .200
MOVWF R2
L2
MOVLW .100
MOVWF R1
L1
NOP
NOP
DECFSZ R1,F
GOTO L1
DECFSZ R2,F
GOTO L2
RETURN
END
```

Flashing LEDs 5Times

```
R1 EQU 0X22
R2 EQU 0X23
Count EQU 0X24
;BANK1
BSF STATUS,RP0
BCF STATUS,RP1
MOVLW .0
MOVWF TRISB;CLRF TRISB
;BANK0
BCF STATUS,RP0
BCF STATUS,RP1
NOVLW .5
MOVWF Count
```

```
START
MOVLW .255
MOVWF PORTB
CALL DELAY
MOVLW .0
MOVWF PORTB
CALL DELAY
DECFSZ Count,F
GOTO START
HERE
GOTO HERE
```

```
DELAY
MOVLW .200
MOVWF R2
L2
MOVLW .100
MOVWF R1
L1
NOP
NOP
DECFSZ R1,F
GOTO L1
DECFSZ R2,F
GOTO L2
RETURN

END
```

Example

- Write the following program

  Create a square wave of 50% duty cycle on bit 0 of C

Sol:

```
BANK1
BCF TRISC,0
START
BANK0
BSF PORTC,0
CALL DELAY
BCF PORTC,0
CALL DELAY
GOTO START
```

Write the following programs:

(a) Create a square wave of 50% duty cycle on bit 0 of Port C.

(b) Create a square wave of 66% duty cycle on bit 3 of Port C.

Solution:

(a) The 50% duty cycle means that the "on" and "off" states (or the high and low portions of the pulse) have the same length. Therefore, we toggle RC0 with a time delay between each state.

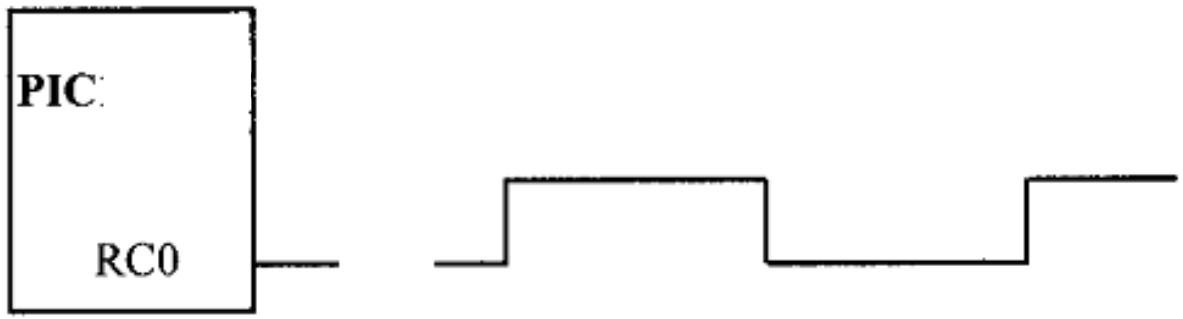
Bank1

```
      BCF    TRISC,0        ;clear TRIS bit for RC0 = out
```

Bank0

```
HERE  BSF    PORTC,0        ;set to HIGH RC0 (RC0 = 1)
      CALL   DELAY          ;call the delay subroutine
      BCF    PORTC,0        ;RC0 = 0
      CALL   DELAY
```

GOTO HERE



(b) A 66% duty cycle means that the "on" state is twice the "off" state.

Bank1

```
BCF    TRISC,3          ;clear TRISC3 bit for output
```

Bank0

```
BACK  BSF    PORTC,3     ;RC3 = 1
      CALL   DELAY       ;call the delay subroutine
      CALL   DELAY       ;twice for 66%
      BCF    PORTC,3     ;RC3 = 0
      CALL   DELAY       ;call delay once for 33%
```

GOTO BACK


PIC
RC3

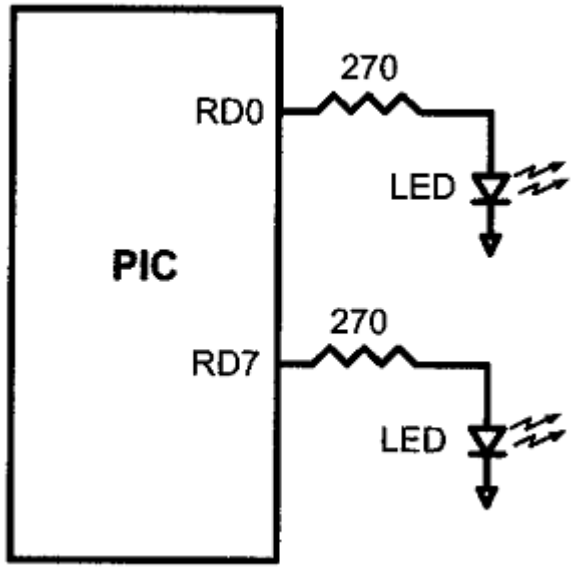
An LED is connected to each pin of Port D. Write a program to turn on each LED from pin D0 to pin D7. Call a delay module before turning on the next LED.

Bank1

```
CLRF   TRISD          ;make PORTD an output port
```

Bank0

```
BSF     PORTD,0
CALL    DELAY
BSF     PORTD,1
CALL    DELAY
BSF     PORTD,2
CALL    DELAY
BSF     PORTD,3
CALL    DELAY
BSF     PORTD,4
CALL    DELAY
BSF     PORTD,5
CALL    DELAY
BSF     PORTD,6
CALL    DELAY
BSF     PORTD,7
CALL    DELAY
```



Example

- Write a program to perform the following:

a) Keep monitoring the RB2 bit until it becomes HIGH (1)

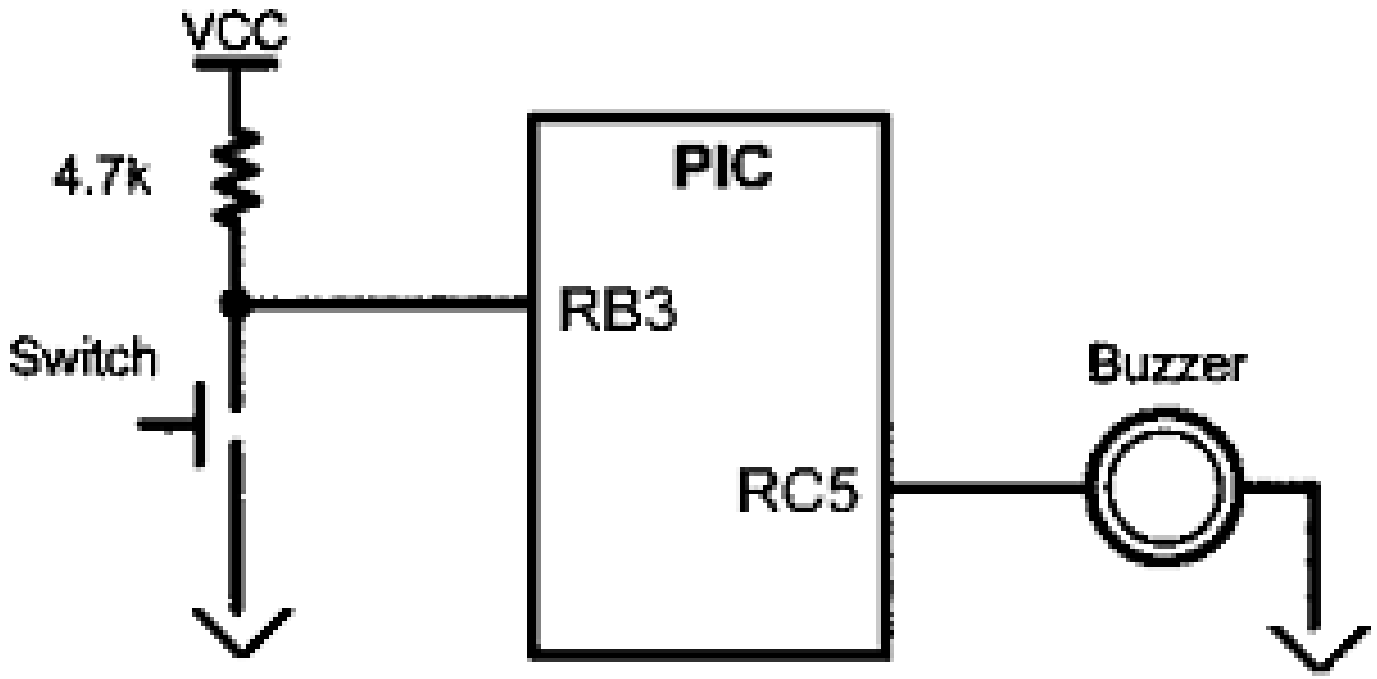
b) When RB2 becomes HIGH, write value 45H to portC and send a HIGH to LOW plus to RD3

Sol:

```
BANK1
BSF TRISB,2   ;RB2 IS INPUT
CLRF TRISC    ;PORTC OUTPUT
BCF TRISD,3   ;RD3 IS OUTPUT
BANK0
TEST
BTFSS PORTB,2
GOTO TEST
MOVLW 0X45
MOVWF PORTC
BSF PORTD,3
CALL DELAY
BCF PORTD,3
CALL DELAY
```

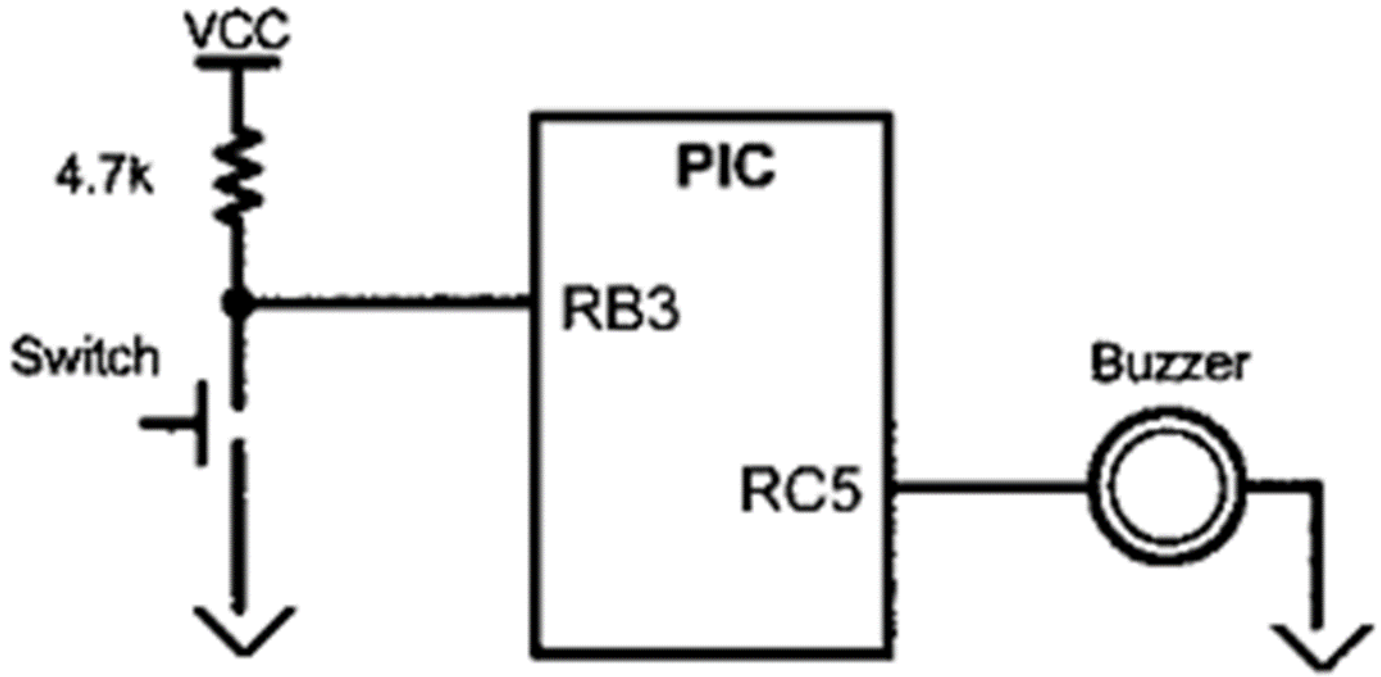
Example 4-5

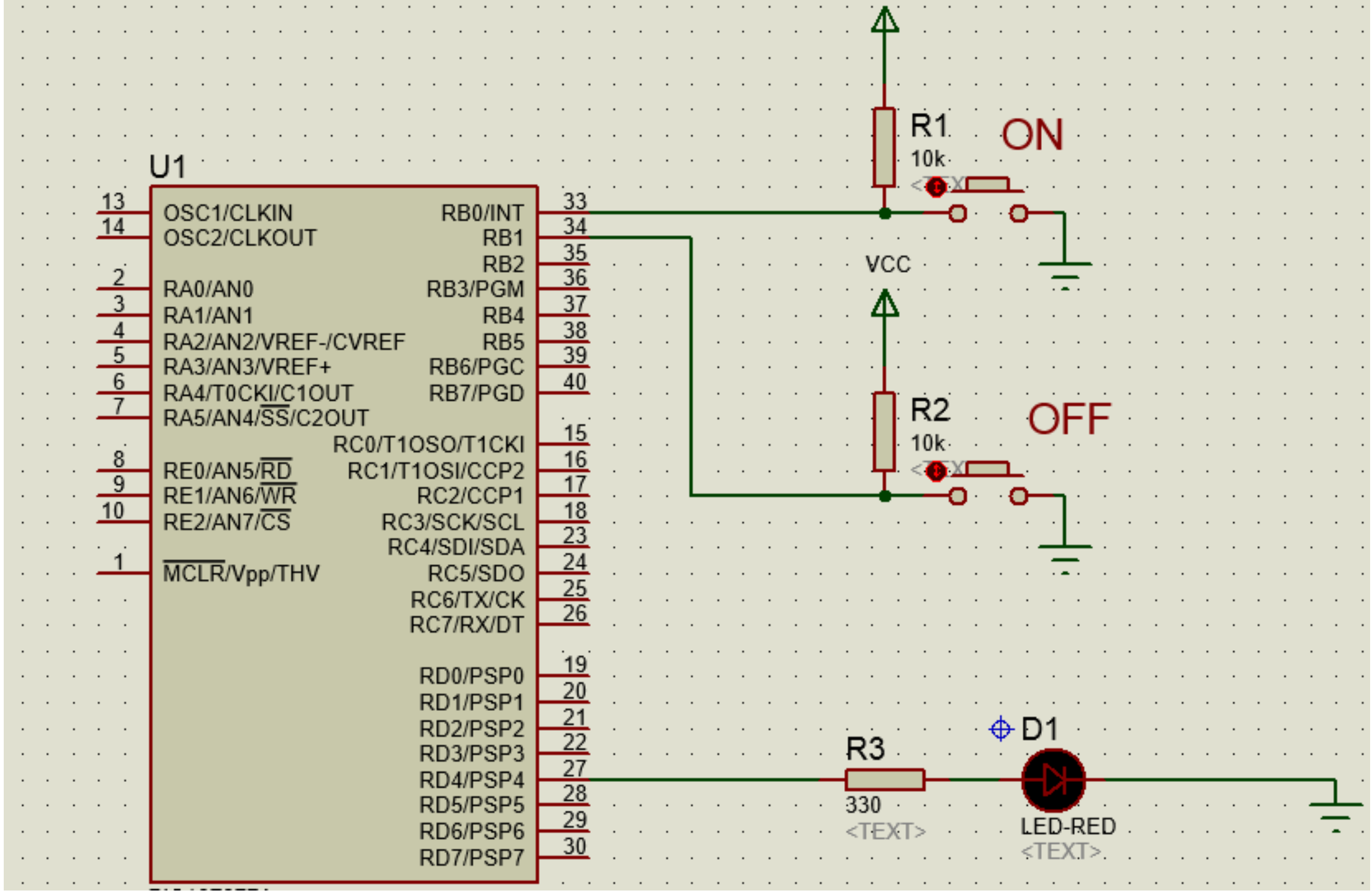
Assume that bit RB3 is an input and represents the condition of a door alarm. If it goes LOW, it means that the door is open. Monitor the bit continuously. Whenever it goes LOW, send a HIGH-to-LOW pulse to port RC5 to turn on a buzzer.



Sol:

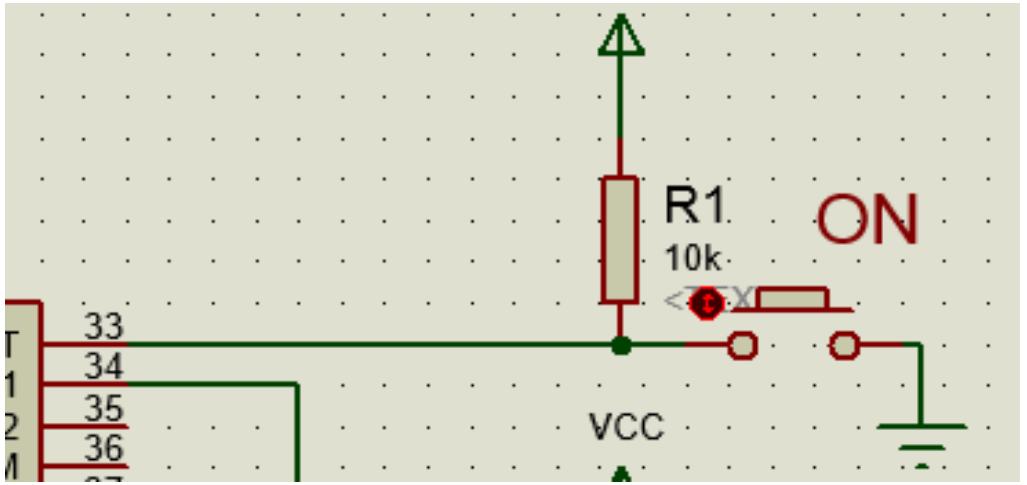
```
BANK1
BSF TRISB,3   ;RB3 IS INPUT
BCF TRISC,5   ;RC5 IS OUTPUT
BANK0
TEST
BTFSC PORTB,3
GOTO TEST
BSF PORTC,5
CALL DELAY
BCF PORTC,5
GOTO TEST
```

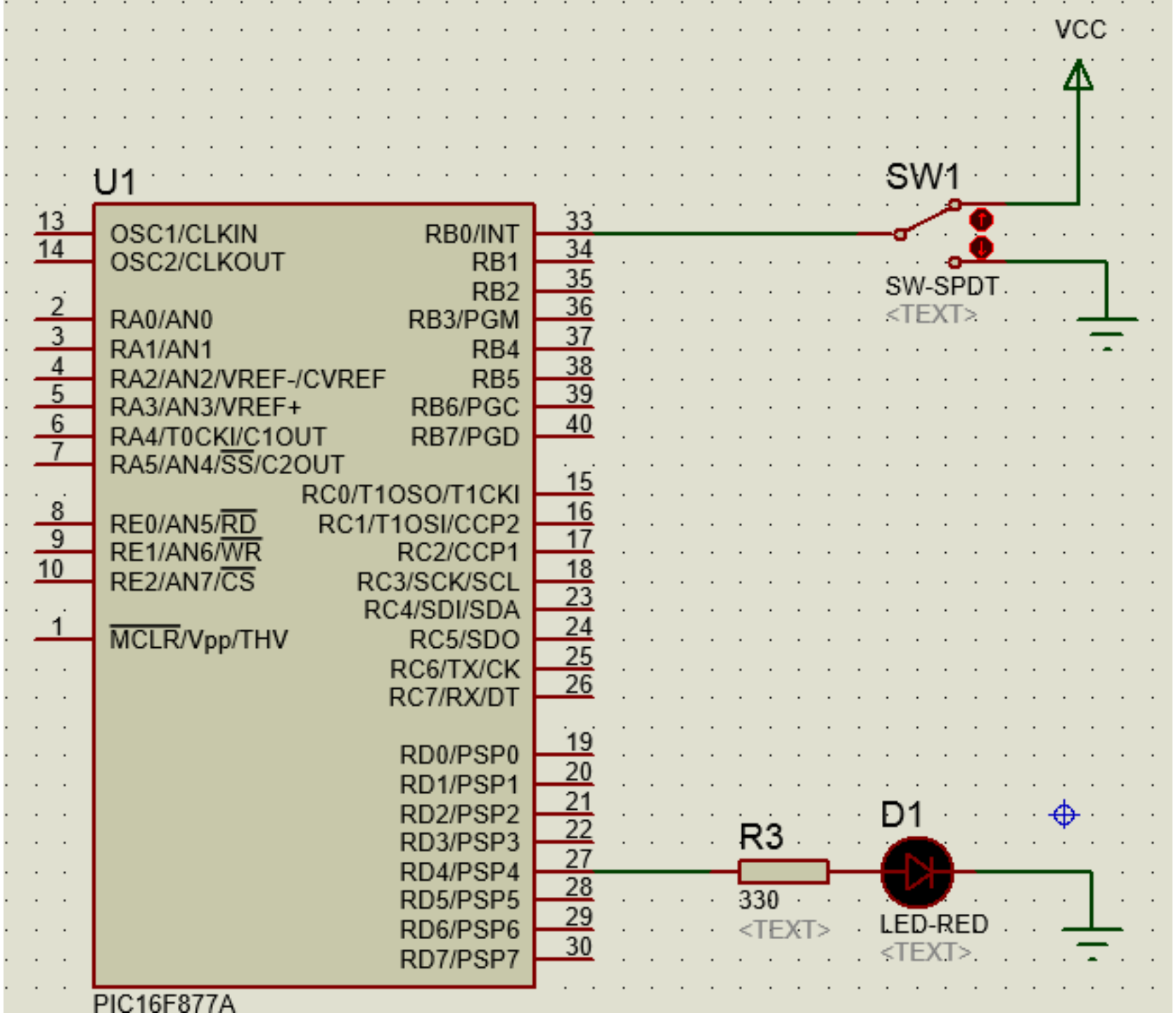



U1
13
14
2
3
4
5
6
7
8
9
10
1
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
33
34
35
36
37
38
39
40
15
16
17
18
23
24
25
26
19
20
21
22
27
28
29
30
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
R1
10k
ON
VCC
R2
10k
OFF
R3
330
<TEXT>
D1
LED-RED
<TEXT>

Sol:

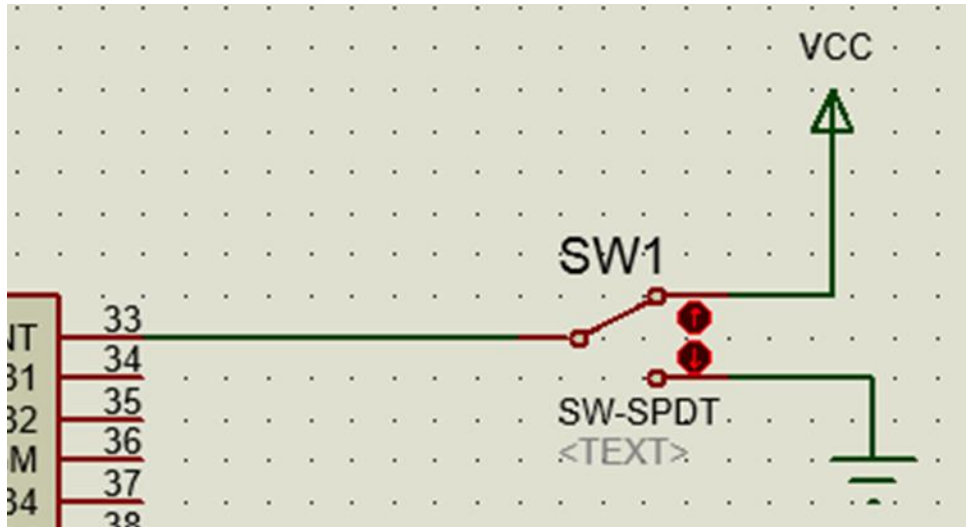
```
BANK1
BSF TRISB,0
BSF TRISB,1
BCF TRISD,4
BANK0
OFF
BCF PORTD,4
BTFSC PORTB,0
GOTO OFF
ON
BSF PORTD,4
BTFSC PORTB,1
GOTO ON
GOTO OFF
```

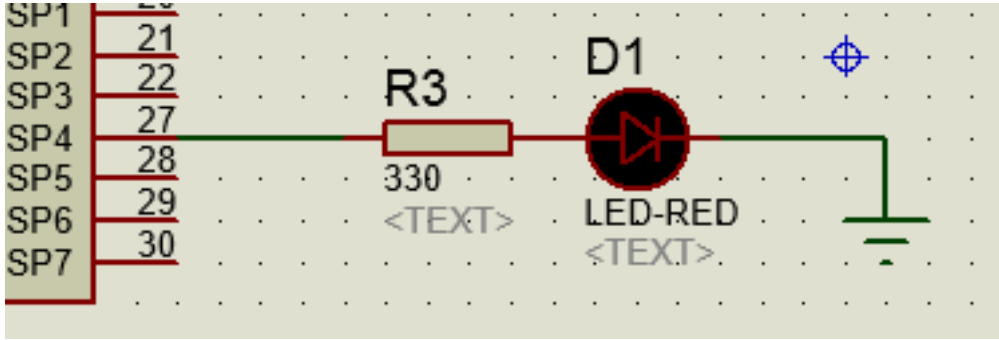



VCC
U1
SW1
13
14
2
3
4
5
6
7
8
9
10
1
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
33
34
35
36
37
38
39
40
15
16
17
18
23
24
25
26
19
20
21
22
27
28
29
30
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
SW-SPDT
<TEXT>
R3
330
<TEXT>
D1
LED-RED
<TEXT>
PIC16F877A

Sol:

```
BANK1
BSF TRISB,0
BCF TRISD,4
BANK0
START
BTFSS PORTB,0
GOTO OFF
GOTO ON
ON
BSF PORTD,4
GOTO START
OFF
BCF PORTD,4
 GOTO START
```






Thank
you!