SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Sub: Microcontroller Programming And Interfacing

Subcode:23ECB202
Unit-I

PIC Microcontrollers: History, Features, & Architecture

Topic: PIC Data Formats and Directives

PIC data Type

The PIC microcontroller has only one data type. It is 8 bits, and the size of each register is also 8 bits.

- The programmer to break down data larger than 8 bits (00 to FFH, or 0 to 255 in decimal) to be processed by the CPU.

Data format representation

- There are four ways to represent a byte of data in the PIC assembler. The numbers can be in hex, binary, decimal, or ASCII formats.

Examples

- Use 'H' or 'h' after the number: MOVLW 99H
- Use '0x' or '0X' before the number: MOVLW 0x99
- No prefix or suffix: MOVLW 99
- Use 'h' with single quotes: MOVLW h'99'

Hexadecimal Format in PIC Assembly

Here are a few lines of code that use the hex format:

```
MOVLW 25      ; WREG = 25H
ADDLW 0x11    ; WREG = 25H + 11H = 36H
ADDLW 12H     ; WREG = 36H + 12H = 48H
ADDLW H'2A'   ; WREG = 48H + 2AH = 72H
ADDLW 2CH     ; WREG = 72H + 2CH = 9EH
```

Binary Numbers in PIC Assembly

- There is only one way to represent binary numbers in a PIC assembler

```
MOVLW B'10011001'   ; WREG = 10011001 or 99 in hex
```

The lowercase b will also work

Examples:

```
MOVLW B'00100101'   ; WREG = 25H
ADDLW B'00010001'   ; WREG = 25H + 11H = 36H
```

Decimal Numbers in PIC Assembly

One way to represent decimal numbers:

```
MOVLW D'12'    ; WREG = 00001100 or 0C in hex
```

The lowercase d will also work.

Unlike other assemblers (8051, x86), PIC requires D'12' instead of just 12.

```
MOVLW D'37'    ; WREG = 25H (37 decimal is 25 hex)
ADDLW D'17'    ; WREG = 37 + 17 = 54 (54 decimal = 36H)
MOVLW .12    ; WREG = 00001100 = 0CH = 12
```

ASCII Character Representation in PIC Assembly

To represent ASCII data in a PIC assembler, we use the letter **A** as follows:

MOVLW A'2'   ; WREG = 00110010 or 32 in hex (See Appendix F)
Lowercase 'a' will also work.
Single quotes are used for single ASCII characters.
Double quotes are used for ASCII strings.

More Examples:
MOVLW A'9'   ; WREG = 39H (hex for ASCII '9')
ADDLW A'1'   ; WREG = 39H + 31H = 70H (31H is ASCII '1')
MOVLW '9'   ; WREG = 39H (another way to represent ASCII)
To define ASCII strings (more than one character), use the DB (define byte) directive.

Assembler Directives

While instructions tell the CPU what to do, directives (also called pseudo-instructions) provide guidance to the assembler.

MOVLW and ADDLW are CPU instructions.

EQU, ORG, and END are assembler directives.

EQU (Equate)

Used to define a constant value or a fixed address. Unlike variables, **EQU** does not allocate memory; it assigns a label to a constant.

```
COUNT EQU 0x25
...
MOVLW COUNT    ; WREG = 25H
```

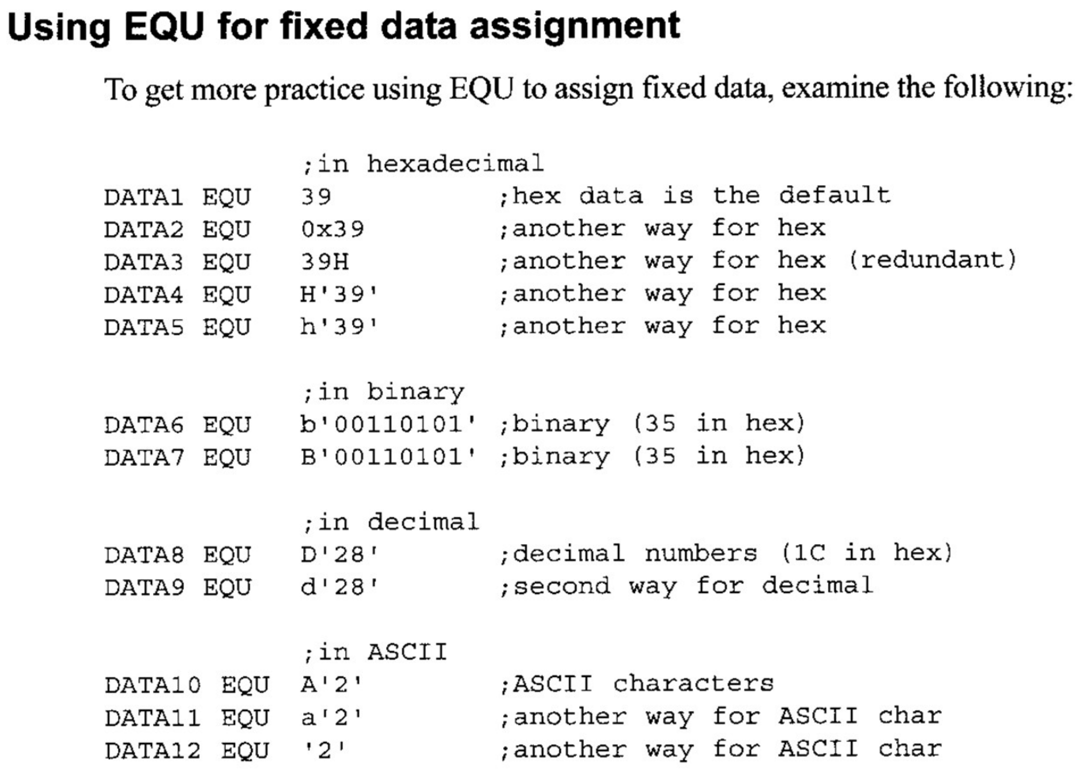
When executing MOVLW COUNT, WREG will be loaded with 25H.

- Advantage of using EQU: If the value needs to change, updating COUNT EQU will reflect the change everywhere in the program.

SET

- Also used to define a constant or fixed address.
- Difference from EQU: The value assigned using SET can be reassigned later.

Example



Using EQU for fixed data assignment

To get more practice using EQU to assign fixed data, examine the following:

```
             ;in hexadecimal
DATA1  EQU   39          ;hex data is the default
DATA2  EQU   0x39        ;another way for hex
DATA3  EQU   39H         ;another way for hex (redundant)
DATA4  EQU   H'39'       ;another way for hex
DATA5  EQU   h'39'       ;another way for hex

             ;in binary
DATA6  EQU   b'00110101' ;binary (35 in hex)
DATA7  EQU   B'00110101' ;binary (35 in hex)

             ;in decimal
DATA8  EQU   D'28'       ;decimal numbers (1C in hex)
DATA9  EQU   d'28'       ;second way for decimal

             ;in ASCII
DATA10  EQU  A'2'        ;ASCII characters
DATA11  EQU  a'2'        ;another way for ASCII char
DATA12  EQU  '2'         ;another way for ASCII char
```

ORG (Origin) Directive in PIC Assembly

The **ORG (Origin)** directive is used in PIC assembly language to specify the **starting address** for code or data in program memory or RAM. It tells the assembler where to place the following instructions in memory.

Usage of ORG

- The **default start address** for program memory in PIC microcontrollers is **0x0000**.
- When using **interrupts**, ORG can set different starting points.

```
ORG 0x00       ; Start of the main program
GOTO MAIN      ; Jump to the main program

ORG 0x04       ; Interrupt vector location
GOTO ISR       ; Jump to the interrupt service routine

ORG 0x10       ; Main program start
MAIN:
   MOVLW 0x55  ; Load 55H into WREG
   GOTO MAIN   ; Infinite loop

ORG 0x20       ; Interrupt Service Routine (ISR)
ISR:
   MOVLW 0xAA  ; Load AA into WREG
   RETFIE      ; Return from interrupt
```

END Directive

- The END directive marks the end of the assembly source code.
- It tells the assembler that no more instructions follow.
- Any lines written after END are ignored.

Example:

```
ORG 0x00
GOTO MAIN
ORG 0x10
MAIN:
  MOVLW 0x55
   GOTO MAIN
 END   ; Marks the end of the program
```

LIST Directive

The LIST directive controls assembler output formatting.

It can define processor type, number formatting, and macro expansions.

Ex: LIST option

LIST P=16F877A → Sets microcontroller type.

LIST R=DEC → Displays values in decimal format.

LIST M=ON → Enables macro expansion.