SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
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

19ECB211 – MICROCONTROLLER PROGRAMMING & INTERFACING

II YEAR IV SEM

UNIT I – PIC MICROCONTROLLER : HISTORY , FEATURES & ARCHITECTURE

TOPIC 7 – PIC Data Formats and Directives

PIC Microcontroller 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 break down data larger than 8 bits, 00 to FFH, or 0 to 255 in decimal to be processed by the CPU.
- The data types used by the PIC can be positive or negative.

Data Format Representation

- There are 4 ways to represent a byte of data in the PIC assembler.
- The numbers can be in hex, binary, decimal or ASCII formats.

Hex Numbers:

There are 4 ways to show hex numbers:

- Can use 'h' or 'H' right after the number like this: MOVLW 88H or MOVLW 88h
- Put 0x or 0X in front of the number like this: MOVLW 0x88 or MOVLW 0X88
- Put nothing in front or back of the number like this: MOVLW 88
- Put 'h' in front of the number, but with this single quotes around the number like this: MOVLW h'88’

Data Format Representation

Some PIC assemblers might give you a warning, but no error, when you use 88H because the assembler already knows that data is in hex and there is no need to remind it.

This is a good reminder when we do coding in assembly.

Below lines of code 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

Data Format Representation

The following are invalid:

MOVLW E5H	;invalid, it must be MOVLW 0E5H
ADDLW C6	;invalid, it must be ADDLW 0C6

If the value starts with the hex digits A - F, then it must be preceded with a zero. The valid is given below.

MOVLW 0F	;valid, WREG = 0FH or 00001111 in binary

Data Format Representation – Binary Number

There is only on way to represent binary numbers in a PIC assembler. It is as follows.

MOVLW B'10011001	;WREG = 10011001 or 99 in Hex

The lowercase b will also work. ' is the single quote key, which is on the same key as the double quotes ". This is different from other assemblers such as the 8051 and x86.

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

Data Format Representation – Decimal Number

There are 2 ways to represent decimal numbers in a PIC assembler, One way is as follows.

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

The lowercase d will work also. This is different from other assemblers such as the 8051 and x86, In those assemblers, to indicate decimal numbers we simply use the decimal e.g, 12 and nothing before or after it, while in the PIC assembler, 12 is the default for hex numbers.

MOVLW D'37'	;WREG = 25H , 37 in Decimal and 25 in Hex
ADDLW D'17'	;WREG = 37 + 17 = 54, 54 in Decimal and 36H in Hex

The other way to represent decimal numbers is to use ".value"

MOVLW .12	;WREG = 00001100 = 0CH = 12

Data Format Representation – ASCII Representation

Use the letter A to represent the ASCII data in PIC assembler.

MOVLW A'2'	;WREG = 00110010 or 32 in Hex

Lowercase 'a' can also use. This is different from other assemblers such as the 8051 and x86. In those assemblers, single quotes are used for single ASCII characters and double quotes are used for a string.

MOVLW A'9'	;WREG = 39H, which is hex number for ASCII '9'
ADDLW A'1'	;WREG = 39H + 31H = 70H ;31 Hex is for ASCII '1'
MOVLW '9'	;WREG = 39H, another way for ASCII

To define ASCII strings, more then one character, we use th DB, define byte directive.

Assembler Directives EQU-Equate

- Directives, also called pseudo-instructions give directions to the assembler.
- The MOVLW and ADDLW instructions are commands to the CPU, but EQU, ORG and END are directives to the assembler.
- The widely used directives of the PIC are EQU and SET.

Assembler Directives EQU-Equate

- This is used to define a constant value or a fixed address.
- The EQU directive does not set aside storage for a data item, but associates a constant number with a data or an address label so that when the label appears in the program, its constant will be substituted for the label

Assembler Directives EQU-Equate

➢The following uses EQU for the counter constant, and then the constant is used to load the WREG register:

COUNT EQU 0x25	;total count as counter number
.....	....
MOVLW COUNT	;WREG = 25H

When executing the above instruction "MOVLW COUNT", the register WREG will be loaded with the value 25H.

Advantage of using Equate.

- Assuming that a constant, a fixed value, is used through out the program, and the programmer wants to change its value everywhere.
- By the use of EQU, the programmer can change it once and the assembler will change all of its occurrences throughout the program,

SET:-

- This directive is used to define a constant value or a fixed address.
- In this regard, the SET and EQU directives are identical.
- The only difference is the value assigned by the SET directive may be reassigned later.

Using EQU for Special Function Register Address Assignment

EQU is widely used to assign special function register (SRF) addresses.

COUNTER EQU 0x00	;counter value 00
PORT EQU 0xFF6	;Special function register Port B address
MOVLW COUNTER	;WREG = 00H
MOVWF PORTB	;Port B now has 00 too
INCF PORTB, F	;Port B has 01
INCF PORTB, F	;increment Port B, Port B = 02
INCF PORTB, F	;increment Poer B, Port B = 03

Using EQU for RAM Address Assignment

The common usage of EQU is for the address assignment of the general purpose region of the file register. Examine the following rewrite of an earlier example using EQU:

MYREG EQU 0x12	;assign RAM location to MYREG
MOVLW 0	;clear WREG, WREG = 0
MOVWF MYREG	;clear MYREG, location 12H has 0
MOVLW 22H	;WREG = 22H
ADDWF MYREG, F	;MYREG = WREG + MYREG
ADDWF MYREG, F	;MYREG = WREG + MYREG
ADDWF MYREG, F	;MYREG = WREG + MYREG
ADDWF MYREG, F	;MYREG = WREG + MYREG

References

https://www.embedded.com/the-evolution-of-embedded-devices-addressing-complex-design-challenges/

http://iamtechnical.com/org-origin-end-list-include-config-radix-directives

https://www.electronicspecifier.com/products/design-automation/embedded-systems-the-evolution-of-embedded-system-design

Mazidi M. A., McKinlay R. D., Causey D. “PIC Microcontroller And Embedded Systems” Pearson Education International, 2008(Unit I,II,III, IV & V)