PIC Microcontroller and Embedded Systems
Muhammad Ali Mazidi, Rolin McKinlay and Danny Causey

Eng. Husam Alzaq
The Islamic Uni. Of Gaza
Chapter 5: Arithmetic, logic Instruction and programs
Objective

- Define the range of numbers possible in PIC unsigned data
- Code addition and subtraction instructions for unsigned data
- Perform addition of BCD
- Code PIC unsigned data multiplication instructions and programs for division
- Code PIC Assembly language logic instructions
- Code PIC rotate instructions
Outlines

- Arithmetic Instructions
- Signed Number Concepts and Arithmetic Operations
- Logic and Compare Instructions
- Rotate instruction and data serialization
- BCD and ASCII Conversion
Arithmetic Instructions

- Unsigned numbers are defined as data in which all the bits are used to represent data
  - no bits are set aside for neg. or pos. sign

- Addition of unsigned numbers
  - ADDLW k
  - ADDWF fileReg, d, a
  - ADDWFC (adding two 16-bit numbers)

- What happens to flag register?
Example 5-3

- Add

3CE7H and 3B8DH

Store the sum in fileReg locations 6 and 7, where location 6 should have the lower byte.

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>05H</td>
<td>00</td>
</tr>
<tr>
<td>06H</td>
<td>00</td>
</tr>
<tr>
<td>07H</td>
<td>00</td>
</tr>
<tr>
<td>08H</td>
<td>00</td>
</tr>
<tr>
<td>09H</td>
<td>00</td>
</tr>
</tbody>
</table>
BCD Number System

- We use the digits 0 to 9 in everyday.
- Binary Coded Decimal
  - Unpacked BCD
    - The lower 4 bits is just used
    - Requires 1 byte
      - Example: 0000 0010
  - Packed BCD
    - A single byte has two BCD numbers
    - Efficient in storing data
      - Example: 0101 0010

<table>
<thead>
<tr>
<th>Digit</th>
<th>BCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
</tbody>
</table>
BCD

- What is the result if you add

- To correct the problem, we should add 6.
DAW, Decimal Adjust WREG

- Works only with WREG
- Add 6 to the lower or higher nibble if needed
- After execution,
  - If the lower nibble is greater than 9, or if DC = 1, add 0110 to the lower nibble.
  - If the upper nibble is greater than 9, or if C = 1, add 0110 to the upper nibble.
- Doesn’t require the use of arithmetic instructions prior the DAW execution
Subtraction of unsigned numbers

- Subtractor circuit is cumbersome. (Why?)
- PIC performs the 2's complement then uses adder circuit to the result.
- Take one Clock Cycle
- There are four sub instructions
  - SUBLW k (k - WREG)
  - SUBWF f d (destination = fileReg - WREG)
- Result may be negative (N=1 and C=1)
  - The result is left in 2's complement
Example 5-5

MOVLW 0x23
SUBLW 0x3F

\[
\begin{array}{c}
0011 1111 \\
+ \\
1101 1101 \\
1 \\
0001 1100 \\
\hline
C = 1, D7 = N = 0
\end{array}
\]
Example 5-6

- Subtract 4C - 6E?

```
| 0100 1100 |
+ 1001 0010 |
| 1101 1110 |
```

- \( C = 0, \ D_7 = N = 1 \)

```
MYREG EQU 0x20
MOVWLW 0x4C
MOVWF MYREG
MOVWLW 0x6E
SUBWF MYREG,W
BNN NEXT
NEGF WREG
NEXT
MOVWF MYREG
```

The PIC uCs
Multiplication of unsigned number

- PIC supports byte-by-byte multiplication
- One of the operand must be in WREG
- After multiplication, the result is stored in PRODH and PRODL (16 bit)
- Example
  - `MOVLW` 0x25
  - `MULLW` 0x65

The PIC uCs
Division of unsigned numbers

- There is no single instruction for the division of byte/byte numbers.
- You need to write a program
  - Repeated subtraction
  - The numerator is placed in a fileReg
  - Denominator is subtracted from it repeatedly
  - The quotient is the number of times we subtracted
  - The remainder is in fileReg upon completion
Example 5-8

Convert the hexadecimal number FDH, stored in location 0x15, into decimal.

Save the digits in locations 0x22, 0x23 and 0x24

```assembly
#include <P18F458.INC>

NUME EQU 0x15
QU EQU 0x20
RMND_L EQU 0x22
RMND_M EQU 0x23
RMND_H EQU 0x24
MYNUM EQU 0xFD
MYDEN EQU D'10'

ORG 0H
MOVFW MYDEN
MOVLW MYNUM
MOVWF NUME
CLRF QU,F
```

It is a Mistake in your book. There is no F
Example 5-8 (2)

D_1

INCF QU,F
SUBWF NUME
BC D_1
ADDWF NUME
DECF QU,F
MOVFF NUME, RMND_L
MOVFF QU, NUME
CLRF QU

D_2

INCF QU,F
SUBWF NUME
BC D_2
ADDWF NUME
DECF QU,F
MOVFF NUME, RMND_M
MOVFF QU, RMND_H
HERE
GOTO HERE
END
Signed Number Concepts and Arithmetic Operations

- The MSB is set aside for the sign (0 or -)
- The rest, 7 bits, are used for the magnitude.
- To convert any 7-bit positive number to negative use the 2’s complement
- You have 128 negative numbers and 127 positive numbers
Overflow problem in Signed Number Operations

- An overflow occurs when the result of an operation is too large for the register.
- OV flag indicate whether the result is valid or not.
  - If OV = 1, the result is erroneous.
- When is the OV flag set?
  - There is a carry from D6 to D7 but no carry out of D7.
  - There is a carry from D7 out (C = 1) but no carry from D6 to D7.
Examples

\[ +96 \quad \begin{array}{c}
\text{0110 0000}
\end{array} \\
+ \quad +70 \quad \begin{array}{c}
\text{0100 0110}
\end{array} \\
+ \quad 166 \quad \begin{array}{c}
\text{1010 0110 (N=1, OV=1 and sum=-90)}
\end{array} \]

\[ -128 \quad \begin{array}{c}
\text{1000 0000}
\end{array} \\
+ \quad - \quad 2 \quad \begin{array}{c}
\text{11 11 11 10}
\end{array} \\
+ \quad 166 \quad \begin{array}{c}
\text{1 0111 1110 (N=0, OV=1 and sum=126)}
\end{array} \]
Logic and Compare Instructions

- Widely used instructions
  - `ANDLW k`
  - `ANDFW FileReg, d`
  - `IORLW k`
  - `IORFW FileReg, d`
  - `XORLW k`
  - `XORFW FileReg, d`

- Effect only Z and N Flags
Complement Instructions

- **COMF**  FileReg,d
  - Takes the 1's complement of a file register
  - Effect only Z and N Flags

- **NEGF**  FileReg
  - Takes the 2's complement of a file register
  - Effect all Flags

**Example**
- **MYREG**  EQU  0x10
- **MOVLW**  0x85
- **MOVWF**  MYREG
- **NEGF**  MYREG
## Compare Instructions

- These instructions take 1/2 cycle(s)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPFSGT</td>
<td>Compare FileReg with WREG, skip if greater than</td>
<td>FileREg &gt; WREG</td>
</tr>
<tr>
<td>FileReg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPFSEQ</td>
<td>Compare FileReg with WREG, skip if equal</td>
<td>FileREg = WREG</td>
</tr>
<tr>
<td>FileReg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPFSLT</td>
<td>Compare FileReg with WREG, skip if less than</td>
<td>FileREg &lt; WREG</td>
</tr>
<tr>
<td>FileReg</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 5-3. Flowchart for CPFSGT
Figure 5-4. Flowchart for CPFSEQ
Figure 5-5. Flowchart for CPFSLT
Example 5-27

- Write code to determine if data on PORTB contains the value 99H. If so, write letter 'y' to PORTC; otherwise, make PORTC='N'

```assembly
CLRF TRISC
MOVLW A'N'
MOVWF PORTC
SETF TRISB
MOVLW 0x99
CPFSEQ PORTB
BRA OVER
MOVLW A'Y'
MOVWF PORTC
OVER ............
```

The PIC uCs
Rotate instruction and data serialization

- Rotate fileReg Right or Left (no Carry)
  - RRNCF fileRed, d
  - RLNCF fileRed, d
  - affect the N and Z flag

- Rotate Right or Left through Carry flag
  - RRCF fileRed, d
  - RLCF fileRed, d
  - affect the C, N and Z flag
Serializing data

- One of the most widely used applications of the rotate instructions.
  - Take less space on the PCB
- Sending a byte of data, one bit at a time through a single pin of uC.
  - Using the serial port.
  - Using a programming technique to transfer data one bit at a time and control the sequence of data and spaces between them.
Example 5-28

- Write a program to transfer value 41H serially via RB1.
- Put one High at the start and end
- Send LSB

Solution

```asm
RCNT     EQU 0x20
MYREG    EQU 0x21

BCF TRISB,1
MOVLW 0x41
MOVWF MYREG
BCF STATUS,C
MOVLW 0x8
MOVWF RCNT
BSF PORTB,1
AGAIN
RRCF MYREG,F
BNC OVER
BSF PORTB,1
BRA NEXT
OVER BCF PORTB,1
NEXT DECF RCNT,F
BNZ AGAIN
BSF PORTB,1
```

The PIC uCs
Example 5-29

- Write a program to bring in a byte of data serially via pin RC7 and save it in file register location 0x21
- The byte comes in with the LSB first

```
RCNT EQU 0x20
MYREG EQU 0x21
BSF TRISC,7
MOVLW 0x8
MOVWF RCNT
AGAIN BTFSC PORTC,7
    BSF STATUS,C
    BTFSS PORTC,7
    BCF STATUS,C
    RRCF MYREG,F
    DECF RCNT,F
    BNZ AGAIN
```
SWAPF

- Swap the lower nibble and the higher nibble

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7-D4</td>
<td>D3-D0</td>
</tr>
<tr>
<td>D3-D0</td>
<td>D7-D4</td>
</tr>
</tbody>
</table>

- In the absence of a SWAPF instruction, how would you exchange the nibbles? How many rotate instruction do you need?
BCD and ASCII Conversion

- What is ASCII?
  - What does Keyboard produce when you press any button?
- Real time clock, RTC, provide the time and date in BCD.
## BCD and ASCII Codes for digits 0-9

<table>
<thead>
<tr>
<th>Key</th>
<th>ASCII (hex)</th>
<th>Binary</th>
<th>BCD (unpacked)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30</td>
<td>0011</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td>0011</td>
<td>0000 0001</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>0011</td>
<td>0000 0010</td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td>0011</td>
<td>0000 0011</td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>0011</td>
<td>0000 0100</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>0011</td>
<td>0000 0101</td>
</tr>
<tr>
<td>6</td>
<td>36</td>
<td>0011</td>
<td>0000 0110</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>0011</td>
<td>0000 0111</td>
</tr>
<tr>
<td>8</td>
<td>38</td>
<td>0011</td>
<td>0000 1000</td>
</tr>
<tr>
<td>9</td>
<td>39</td>
<td>0011</td>
<td>0000 1001</td>
</tr>
</tbody>
</table>
Packed BCD to ASCII Conversion

- RTC provides the date and the time in packed BCD
- Data must be in ASCII to be displayed on a LCD

<table>
<thead>
<tr>
<th>packed BCD</th>
<th>Unpacked BCD</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>29H</td>
<td>02H &amp; 09H</td>
<td>32H and 39H</td>
</tr>
<tr>
<td>0010 1001</td>
<td>0011 0010</td>
<td>0011 0010</td>
</tr>
<tr>
<td>0011 1001</td>
<td>0011 1001</td>
<td>0011 1001</td>
</tr>
</tbody>
</table>
ASCII to Packed BCD Conversion

- Get rid of the high nibble (3)

<table>
<thead>
<tr>
<th>key</th>
<th>ASCII</th>
<th>Unpacked BCD</th>
<th>packed BCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>34</td>
<td>0000 0100</td>
<td>0100 0111</td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>0000 0111</td>
<td>which is 47H</td>
</tr>
</tbody>
</table>
Example 5-32

Assume that register WREG has packed BCD. Write a program to convert packed BCD to two ASCII numbers and place them in file register locations 6 and 7.

```assembly
BCD_VAL   EQU      0x29
L_ASC     EQU      0x06
H_ASC     EQU      0x07
MOVLW BCD_VAL
ANDLW 0x0F
IORLW 0x30
MOVWF L_ASC
MOVWL BCD_VAL
ANDLW 0xF0
SWAPF WREG,W
IORLW 0x30
MOVWF H_ASC
```

The PIC uCs
Chapter 5: Summary

- We discussed arithmetic instructions for both signed and unsigned data.
- We defined the logic and compare instructions.
- The rotate and swap instructions are widely used.
- We described BCD and ASCII formats and conversions.

Next:
Bank Switching, Table processing, Macros and Modules