Chapter 3: Branch, Call and Time Delay Loop

- Branch instruction and looping
- Call instruction and stack
- PIC18 Time Delay and instruction pipeline

Objective

- Code PIC Assembly language instructions to create loops and conditional branch instructions
- Code Goto instructions for unconditional jump

Looping in PIC

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

DECFSZ instruction

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

Branch instructions and looping

- Looping in PIC
- Loop inside loop
- Other conditional jumps
- All conditional branches are short jumps
- Calculating the short branch address
- Unconditional branch instruction
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
ADDLW 3
DECFSZ COUNT,F
GOTO AGAIN
MOVWF PORTB

Example 3-2

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
ADDLW 3
DECF fileReg, f
BNZ Back

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 times
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 'd'10'
COUNT_2 EQU 'd'70'

Solution

MOVLW 0x55
MOVWF PORTB
MOVLW COUNT_1
MOVWF R1

LOP_1
MOVLW COUNT_2
MOVWF R2

LOP_2
COMPF PORTB, F
DECF R2, F
BNZ LOP_2
DECF R1, F
BNZ LOP_1

Other conditional jumps

- All of the 10 conditional jumps are 2-byte instructions
- They require the target address
  - 1 byte address (short branch address)
  - Relative address
- Recall: MOVF will affect the status reg.
- In the BZ instruction, the Z flag is checked. If it is high, that is equal 1, it jumps to the target address.
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
BNZ NEXT
MOVLF 0x55
MOVWF MYLOC
NEXT ...

Example 3-6

Find the sum of the values 79H, F5H, and E2H. Put the sum in fileReg loc. 5H and 6H.

Address Data
5H (Low-Byte) 0
6H (High-Byte) 0

Solution

L_Byte EQU 0x5
H_Byte EQU 0x6
ORG 0h
MOVLW 0
MOVWF H_Byte
ADDLW 0x79
BN C N_1
INCF H_Byte,F
OVER MOVWF L_Byte
END

Example 3-7

Program Memory

<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Opcode</th>
<th>Label</th>
<th>Disassembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000</td>
<td>0E00</td>
<td></td>
<td>MOVLW 0x0</td>
</tr>
<tr>
<td>2</td>
<td>00002</td>
<td>6E06</td>
<td></td>
<td>MOVWF H_Byte</td>
</tr>
<tr>
<td>4</td>
<td>00004</td>
<td>0F79</td>
<td></td>
<td>ADDLW 0x79</td>
</tr>
<tr>
<td>6</td>
<td>00006</td>
<td>E301</td>
<td></td>
<td>INCF H_Byte,F</td>
</tr>
<tr>
<td>8</td>
<td>00008</td>
<td>2A06</td>
<td>N_2</td>
<td>ADDLW 0xE2</td>
</tr>
<tr>
<td>10</td>
<td>00010</td>
<td>E301</td>
<td>N_1</td>
<td>ADDLW 0xF5</td>
</tr>
<tr>
<td>12</td>
<td>00012</td>
<td>E301</td>
<td>N_2</td>
<td>ADDLW 0xF5</td>
</tr>
</tbody>
</table>

Example 3-8

Program Memory

<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Opcode</th>
<th>Label</th>
<th>Disassembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000</td>
<td>0E00</td>
<td></td>
<td>MOVLW 0x0</td>
</tr>
<tr>
<td>2</td>
<td>00002</td>
<td>6E06</td>
<td></td>
<td>MOVWF H_Byte</td>
</tr>
<tr>
<td>4</td>
<td>00004</td>
<td>0F79</td>
<td></td>
<td>ADDLW 0x79</td>
</tr>
<tr>
<td>6</td>
<td>00006</td>
<td>E301</td>
<td></td>
<td>INCF H_Byte,F</td>
</tr>
<tr>
<td>8</td>
<td>00008</td>
<td>2A06</td>
<td>N_2</td>
<td>ADDLW 0xE2</td>
</tr>
<tr>
<td>10</td>
<td>00010</td>
<td>E301</td>
<td>N_1</td>
<td>ADDLW 0xF5</td>
</tr>
<tr>
<td>12</td>
<td>00012</td>
<td>E301</td>
<td>N_2</td>
<td>ADDLW 0xF5</td>
</tr>
</tbody>
</table>
Question?

Which is better, to use BNZ along with DECF or DCFSNZ??

Unconditional branch instruction

- Control is transferred unconditionally to the target location (at ROM)
- Two unconditional branches
  - GOTO
  - BRA

Figure 3-4. GOTO Instruction

Figure 3-5. BRA (Branch Unconditionally) Instruction Address Range

BRA Instruction

Program Memory

GOTO to itself

- Label and $ can be used to keep uC busy (jump to the same location)
- HERE GOTO HERE
- GOTO $
- OVER BRA OVER
- BRA $
**PIC18 Call instruction**

**Section 3-2**

CALL Instruction

<table>
<thead>
<tr>
<th>PCU</th>
<th>PCH</th>
<th>PCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>k</td>
<td>0</td>
</tr>
</tbody>
</table>

- Control is transferred to subroutine
- Current PC value, the instruction address just below the CALL instruction, is stored in the stack
  - push onto the stack
- Return instruction is used to transfer the control back to the caller,
  - the previous PC is popped from the stack

**Stack and Stack Pointer (SP)**

- Read/Write Memory
- Store the PC Address
  - 21-bit (000000 to 1FFFFF)
- 5-bit stack, total of 32 locations
- SP points to the last used location of the stack
  - Location 0 doesn’t used
  - Incremented pointer

**Figure 3-7. PIC Stack 31 × 21**
Return from Subroutine

- The stack is popped and the top of the stack (TOS) is loaded into the program counter.
- If \( s = 1 \), the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, W, STATUS and BSR.
- If \( s = 0 \), no update of these registers occurs (default).

Example 3-9

- Toggle all bits of to SFR register of PORTB by sending to it values 55H and AAH continuously. Put a delay in between issuing of data to PORTB.
- Analyze the stack for the CALL instructions

Example 3-10

Address Data

The PIC uCs

Solution

Example 3-11

Address Data

The PIC uCs

RCALL (Relative Call)

- 2-Byte instruction
- The target address must be within 2K
- 11 bits of the 2 Byte is used
- Save a number of bytes.
Example 3-12

<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Opcode</th>
<th>Label</th>
<th>Disassembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0000</td>
<td>0011</td>
<td>MOVL OSS</td>
<td>MOVL OSS</td>
</tr>
<tr>
<td>2</td>
<td>0002</td>
<td>0001</td>
<td>MOVW PORT, ACCESS</td>
<td>MOVW PORT, ACCESS</td>
</tr>
<tr>
<td>3</td>
<td>0004</td>
<td>0002</td>
<td>SCALL DELAY</td>
<td>SCALL DELAY</td>
</tr>
<tr>
<td>4</td>
<td>0006</td>
<td>0001</td>
<td>COMB PINS, F, ACCESS</td>
<td>COMB PINS, F, ACCESS</td>
</tr>
<tr>
<td>5</td>
<td>0008</td>
<td>FF77</td>
<td>Faux PINS</td>
<td>Faux PINS</td>
</tr>
<tr>
<td>6</td>
<td>000A</td>
<td>FF77</td>
<td>Faux PINS</td>
<td>Faux PINS</td>
</tr>
<tr>
<td>7</td>
<td>000C</td>
<td>FF77</td>
<td>Faux PINS</td>
<td>Faux PINS</td>
</tr>
<tr>
<td>0</td>
<td>0010</td>
<td>0001</td>
<td>RET MN</td>
<td>RET MN</td>
</tr>
<tr>
<td>1</td>
<td>0012</td>
<td>0010</td>
<td>RET MN</td>
<td>RET MN</td>
</tr>
<tr>
<td>11</td>
<td>0030</td>
<td>FF77</td>
<td>NOP</td>
<td>FF77</td>
</tr>
<tr>
<td>12</td>
<td>0032</td>
<td>FF77</td>
<td>NOP</td>
<td>FF77</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

PIC18 Time Delay and instruction pipeline

Section 3-3

Delay Calculating for PIC18

- Two factors can affect the accuracy of the delay
  1. The duration of the clock period, which is function of the Crystal freq
     - Connected to OSC1 And OSC2
  2. The instruction cycle duration
     - Most of the PIC18 instructions consumes 1 cycle
       - Use Harvard Architecture
       - Use RISC Architecture
       - Use the pipeline concept between fetch and execute.

Figure 3-9. Pipeline vs. Non-pipeline

PIC multistage pipeline

- Superpipeline is used to speed up execution.
- The process of executing instructions is split into small steps
- Limited to the slowest step

Figure 3-10. Pipeline Activity After the Instruction Has Been Fetched
The PIC uCs

Figure 3-11. Pipeline Activity for Both Fetch and Execute

Instruction Cycle time for the PIC

- What is the Instruction Cycle?
- Most instructions take one or two cycles
  - BTFSS can take up to 3 cycles
- Instruction Cycle depends on the freq. of oscillator
- Clock source: Crystal oscillator and on-chip circuitry
- One instruction cycle consists of four oscillator periods.

Instruction Cycle = \( \frac{1}{f_{osc}} \)

Example 3-14

- Find the period of the instruction cycle you chose 4 MHz crystal? And what is required time for fetching an instruction?
- Solution
  - 4 MHz/4 = 1 MHz
  - Instruction Cycle = \( \frac{1}{1 \text{ MHz}} = 1 \text{ usec} \)
  - Fetch cycle = \( 4 \times 1 \text{ usec} = 4 \text{ usec} \)

Branch penalty

- Queue is needed for prefetched instruction
- If the prefetched instruction is incorrect, the CPU must flush the memory. When??

BTFSC and BTFSS

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read</td>
<td>Process</td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>Operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>
Example 3-15

Find how long it take to execute each of the following instructions for a PIC18 with 4 MHz

- MOVLW
- ADDLW CALL
- DECF GOTO
- NOP BNZ
- MOVWF

Example 3-16

Find the size of the delay in the following program if the crystal freq. is 4MHz.

```
DELAY
MOVLW 0xFF
MOVWF MYREG
AGAIN
NOP
NOP
DECF MYREG, F
BNZ AGAIN
RETURN
```

Example 3-17

```
MYREG EQU 0x08
ORG 300H
BACK MOVLW 0xF5
MOVWF PORTB AGAIN NOP
CALL DELAY NOP
MOVLW 0xAA NOP
MOVWF PORTB
CALL DELAY BNZ AGAIN
GOTO BACK RETURN
```

Example 3-20

```
DELAY_500MS
MOVLW D'20'
MOVWF R4
BACK
MOVLW D'100'
MOVWF R3
AGAIN
MOVLW D'250'
MOVWF R2
R2 EQU 0x2
R3 EQU 0x3
R4 EQU 0x4
MOVLW 0x55
MOVWF PORTB
CALL DELAY_500MS
COMF PORTB GOTO BACK
HERE NOP
DECF R2, F
BNZ HERE
DECF R3, F
BNZ AGAIN
DECF R4, F
BNZ BACK
RETURN
```

Chapter 3: Summary

- Looping in PIC Assembly language is performed using an instruction to decrement a counter and to jump to the top of the loop if the counter is not zero.
- Assembly language includes conditional and unconditional, and call instructions.
- PIC18 uses Superpipeline is used to speed up execution.

Next: Chapter 4
PIC I/O Port Programming