Chapter 5
PIC18F Architecture and Addressing Modes

Chien-Chung Ho (何建忠)

Basic Features

- Since the PIC18F CPU uses the Harvard architecture, program and data memory units use separate memory spaces along with their own buses. This allows the PIC18F to access both programs and data simultaneously.
- The PIC18F uses flash memory to store program memory while RAM contains data memory. Note that F in PIC18F indicates that the chip contains flash memory.

Basic Features

- Several registers are mapped as memory locations, and these registers are called “Special Function Registers (SFRs)”. This makes it convenient for the programmers to use the register names in the program. For example, PORTC is mapped as an SFR with address 0xF82. Note that SFRs are CPU registers and registers of the peripheral module. Typical CPU registers include Stack Pointer, STATUS register, and Program Counter. Typical peripheral registers, on the other hand, include I/O ports, Timer registers, ADC registers, and CCP registers. The PIC18F assembler and the C18 compiler can access these registers by name. They can be read and written like a variable defined in the application.

Basic Features

- The maximum size of the Program Counter (PC) of the PIC18F is 21-bit wide. Hence, the PIC18F can directly address a maximum of two Megabytes (2^{21}) of program memory space.
- The data memory address, on the other hand, is 12-bit wide. Hence, the PIC18F can directly address data memory of up to 4Kbytes (2^{12}) . However, the sizes of program memory and data memory may vary from one of the PIC18F family members to another.

Basic Features

- The sizes of program and data memories, number of Input/Output (I/O) ports, and the clock frequency may vary from one version to another.
- The PIC18F4321 contains 256 bytes of SRAM. The PIC18F4520, on the other hand, includes 1536 bytes of RAM. Two popular PIC18F family members include PIC18F4321 and PIC18F4520. Table 5.1 summarizes the basic differences between them.

Basic Features

TABLE 5.1 Basic differences of two popular members of the PIC18F Family (F in PIC18F indicates on-chip flash memory)

On-chip features	PIC18F4321	PIC18F4520
Flash memory (Program memory)	4k x 16	4k x 16
RAM data memory (bytes)	256	1536
EEPROM (bytes)	256	256
Maximum Operating Frequency	40 MHz	40 MHz
I/O Pins	36	36
Timers	4	4
Capture/Compare/PWM (CCP) modules	2	2
Serial communication interface	Yes	Yes
10-bit ADC (Can also be configured as 8-bit)	13 Channels	13 Channels
Instruction set	77 instructions; 83 with extended set enabled	75 instructions; 83 with extended set enabled
Number of pins	40, 44	40, 44

Basic Features

- The PIC18F can perform functions such as capture, compare, and pulse width modulation (PWM) using the timers and CCP (Capture / Compare / PWM) modules. The PIC18F can compute the period of an incoming signal using the capture module. The PIC18F can produce a periodic waveform or time delays using the compare module.

Basic Features

- The PIC18F's on-chip PWM can be used to obtain pulse waveforms with a particular period and duty cycle which are ideal for applications such as motor control. The PIC18F serial communication interface can be used to facilitate data transmission for serial peripheral devices which can transmit or receive data one bit at a time.

Basic Features

- The on-chip 10-bit A/D converter of the PIC18F can convert an analog signal into 10-bit binary equivalent. This is very convenient in practice since physical variables such as temperature, flow, and pressure are analog in nature and must first be converted into an analog electrical signal.

Basic Features

- All PIC18F microcontrollers use pipelining to execute instructions. The pipelining is performed in two stages which is defined as a single instruction cycle.
- The PIC18F microcontrollers use the main oscillators (OSC1) to synchronize its internal operations.

Basic Features

- The PIC18F clock input is internally divided into four non-overlapping clocks (Q1, Q2, Q3 and Q4). A machine cycle is defined as four pulses from the main oscillator (OSC1). During the time Q1 is active, the program counter is incremented by 2 (since the PIC18F instruction size is normally 16-bit) to point to the next instruction to be fetched. This instruction will be fetched during Q4. Simultaneously, the previous instruction is being executed during the whole machine cycle (from Q1 to Q4).

Basic Features

- There are three phases of an instruction execution. They are fetch, decode, and execution. During the fetching phase, the instruction is fetched from the program memory and stored into the Instruction Register (IR) during Q4. During the decoding phase, the control unit translates the contents of the IR, and determines the specified operation to be performed. Finally, during the execution phase, the microcontroller executes the instruction. The clocks and instruction execution flow are shown in Figure 5.1.

Basic Features

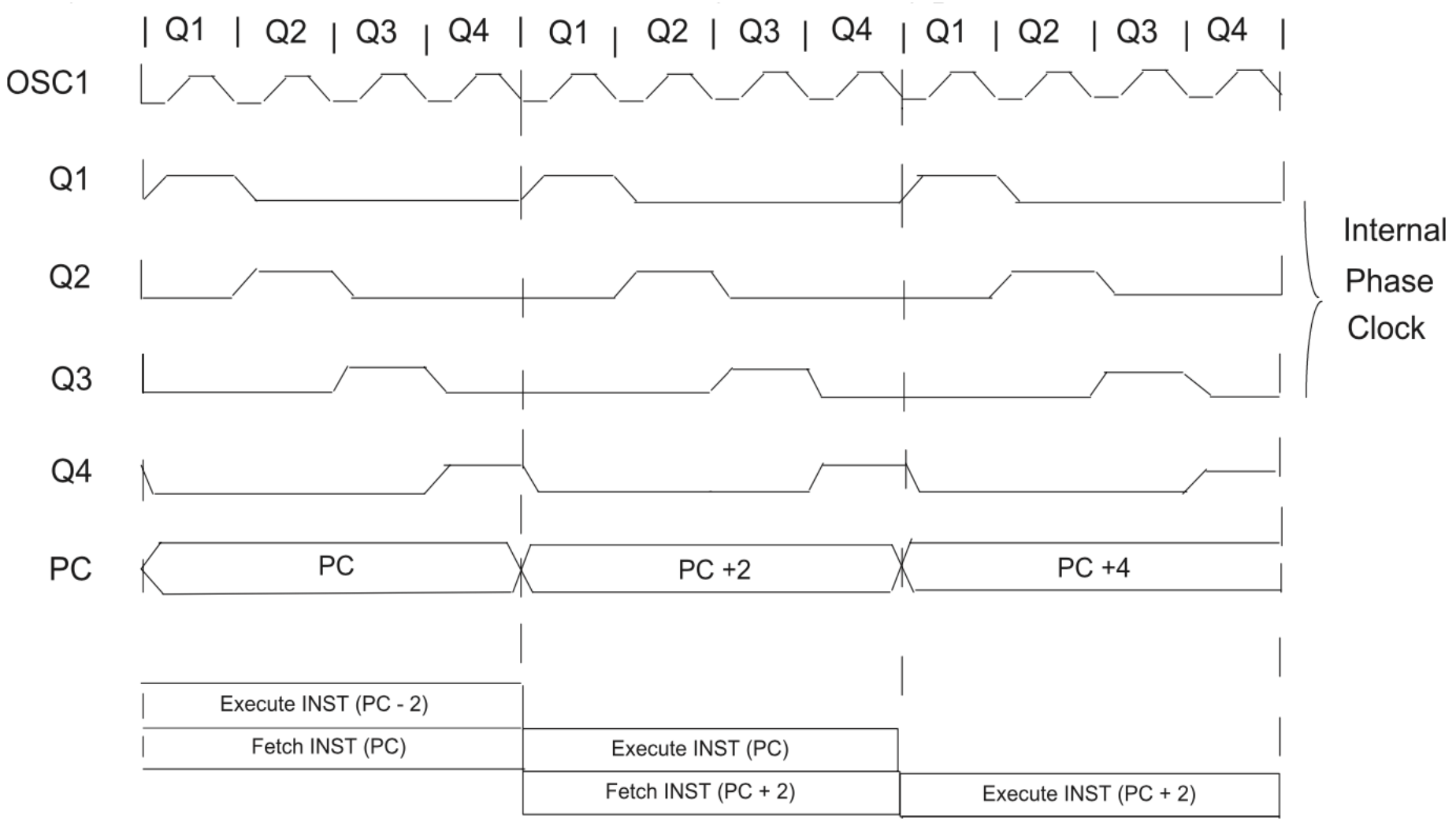
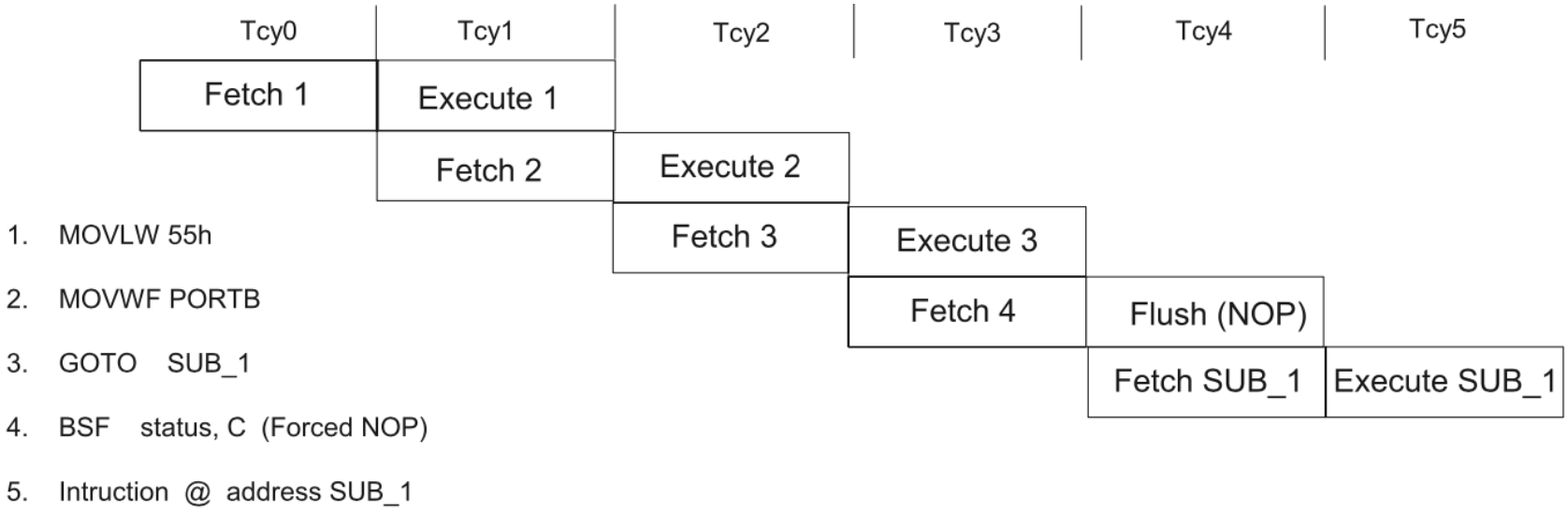


FIGURE 5.1 Clock/Instruction Cycle

Basic Features

- The PIC18F uses a two-stage pipeline. This means that execution of the previous instruction is overlapped with fetching of the current instruction.



All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed

FIGURE 5.2 Instruction Pipeline Flow

Basic Features

- An “Instruction Cycle” consists of four Q cycles: Q1 through Q4. The instruction fetch and execute cycles are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute phases take another instruction cycle.
- Consider Figure 5.2. The PIC18F fetches MOVLW instruction into IR during Tcy0. The PIC18F executes MOVLW in Tcy1 and also fetches the next instruction, MOVWF in Tcy1. …

Basic Features

- The PIC18F unconditionally jumps to address SUB_1, and executes the instruction at address SUB_1. Hence, the pipeline is flushed (NOP) in Tcy4, and the instruction at address SUB_1 is fetched in Tcy4. The instruction at address SUB_1 is executed in Tcy5. Hence, each of the instructions, MOVLW and MOVWF, is executed in a single cycle. The GOTO instruction, on the other hand, takes two cycles since the instruction BSF is flushed from the pipeline while the new instruction at address SUB_1 is fetched and then executed.

PIC18F Register Architecture

- Figure 5.3 shows the PIC18F CPU registers. All registers in the PIC18F are mapped as addresses in the data memory, and are called SFRs (Special Function Registers).
- Each register is assigned with a unique 12-bit memory address.

PIC18F Register Architecture

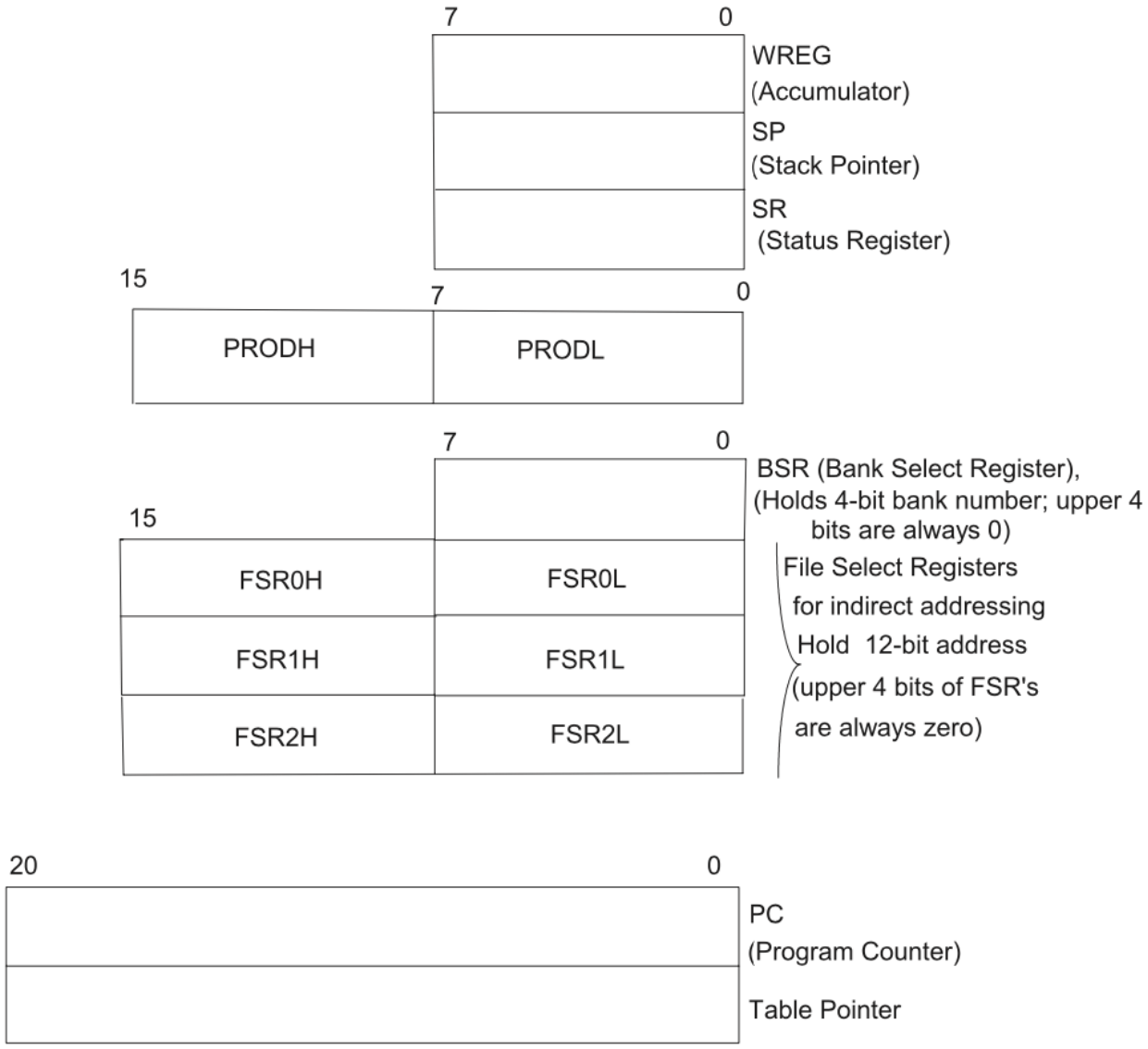


FIGURE 5.3 PIC18F registers

PIC18F Register Architecture

- **WREG**
- The WREG (Working register) is 8-bit wide. This is the PIC18F accumulator. Most arithmetic and logic operations are performed using the WREG.

PIC18F Register Architecture

- **SP**
- The SP (Stack Pointer) register (called STKPTR in the PIC18F) is 8-bit wide.
- The PIC18F hardware stack is a group of 31 21-bit registers to hold memory addresses.
  - Note that the PIC18F program counter is 21-bit wide.
- The low five bits of the STKPTR are used to address the stack. The PIC18F hardware stack is accessed from the bottom.
  - SP is incremented by 1 after PUSH and decremented by 1 after POP.

PIC18F Register Architecture

- Figure 5.4 shows the details of the SP. The PIC18F subroutine CALL and RETURN instructions push and pop the program counter using the hardware stack.

7	6	5	4	3	2	1	0
STKFUL	STKUNF	——	SP4	SP3	SP2	SP1	SP0

FIGURE 5.4 SP (Stack Pointer)

PIC18F Register Architecture

- The 31 stack registers are neither part of program memory nor data memory. The low five bits of the SP address the stack. The stack overflow bit (STKFUL, bit 7) is set to one if more than 31 registers are attempted for pushing addresses onto the stack. The stack underflow bit (STKUNF, bit 6) is set to one if more addresses that are stored in the stack are attempted to be popped. Bit 5 is not implemented and is read as 0.

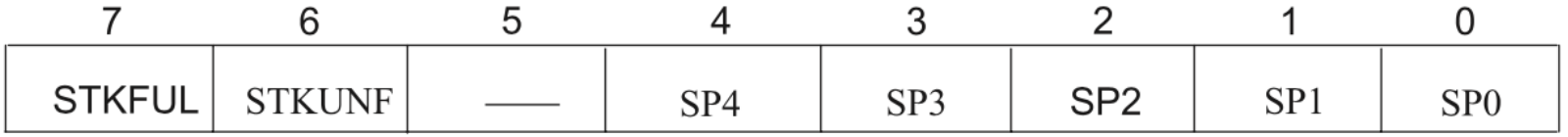


FIGURE 5.4 SP (Stack Pointer)

PIC18F Register Architecture

- **PC**
- The maximum size of the PIC18F's PC (Program Counter) is 21-bit wide. The size of the PC varies from one PIC18F family member to another. The PC normally points to the next instruction. As mentioned before, the 21-bit PC provides the PIC18F with direct addressing capability of a maximum of 2MB of program memory.

PIC18F Register Architecture

- The 21-bit PC is comprised of three 8-bit registers, namely PCL (PC Low byte), PCLATH (PC latch High byte), and PCLATU (PC Latch Upper 5 bits).
- Bits 0 through 7 in PCL, bits 8 through 15 in PCLATH, and bits 16 through 20 in the low five bits of PCATU.

PIC18F Register Architecture

- **Table Pointer**
- The PIC18F uses the 21-bit Table Pointer register as a pointer to a table in program memory for copying bytes between program memory and data memory. This register is mapped by the PIC18F as three 8-bit special function registers in the data SRAM.

PIC18F Register Architecture

- **BSR**
- The BSR (Bank Select Register) is 8-bit wide. The low four bits are used to provide the bank address from 0 to F; upper four bits of BSR are zero. The BSR provides the upper four bits of a 12-bit address of data memory. BSR is used for directly addressing the data SRAM.
  - It divides the data memory space into 16 contiguous banks (bank 0 through 15) of 256 bytes.

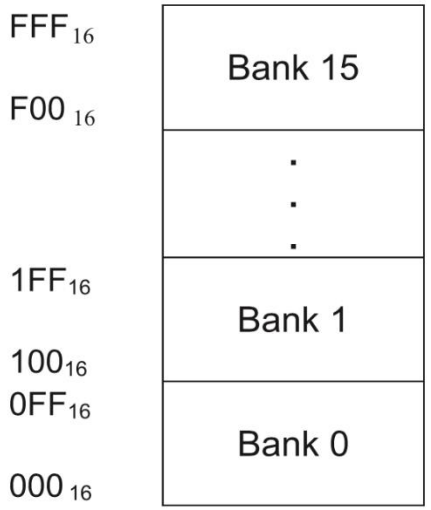


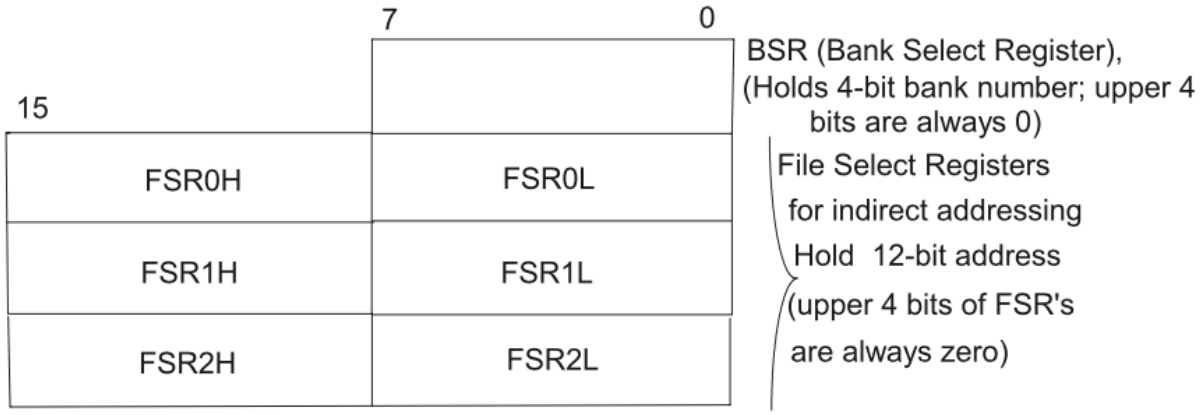
FIGURE 3.1 PIC18F data memory.

PIC18F Register Architecture

- **FSR**
- The FSR (File Select Register) consists of three 16-bit registers (FSR0, FSR1, FSR2) with the upper four bits of each FSR as zero. The low 12 bits of FSR0, FSR1, or FSR2 are used to hold the 12-bit memory address of the data SRAM. These registers are basically memory pointers. Hence, they are typically used for handling arrays and pointer-based data accessing.

PIC18F Register Architecture

- Each of these three registers are divided into two 8-bit registers as follows: FSR0H (high byte of FSR0) and FSR0L (low byte of FSR0), FSR1H (high byte of FSR1) and FSR1L (low byte of FSR1), FSR2H (high byte of FSR2) and FSR2L (low byte of FSR2).



PIC18F Register Architecture

- **PRODH / PRODL**
- Each of the PRODH and PRODL registers is 8-bit wide. The PIC18F has two 8-bit x 8-bit unsigned multiplication instructions providing 16-bit product. The upper byte of the product is stored in the PRODH register while the lower byte of the product is placed in the PRODL register.

PIC18F Register Architecture

SR The SR (Status Register) is 8-bit wide. Figure 5.5 shows the PIC18F status register which contains the flags. The meaning of these flags will be explained in the following:

- C (Carry flag) is set to 1 if there is a carry from addition or a borrow from subtraction; otherwise, C = 0.
- DC (Digit Carry flag) is set to 1 if there is a carry due to addition of the low 4 bits into the high 4 bits or a borrow due to the subtraction of the low 4 bits from the high 4 bits of a number; otherwise, DC = 0. This flag is used by BCD arithmetic instructions.
- Z (Zero flag) is set to 1 if the result is zero; Z = 0 for a nonzero result.
- OV (Overflow flag) is set to 1 if there is an arithmetic overflow (i.e., if the size of the result exceeds the capacity of the destination location); otherwise, OV = 0. Note that in overflow, $OV = C_f \oplus C_p$ where C_f is the final carry and C_p is the previous carry. Overflow is used for signed arithmetic (2's Complement).
- N (Negative flag) is set to 1 if the most significant bit of the result is 1 indicating a negative number; N = 0 if the most significant of the result is 0 indicating a positive number.
- Bits 5 through 7 are not implemented, and are read as zero.

PIC18F Register Architecture

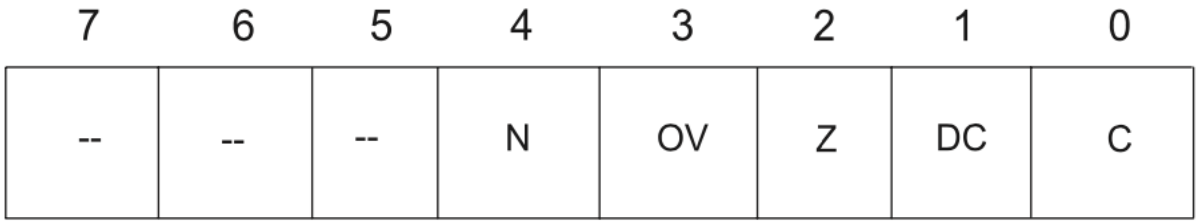
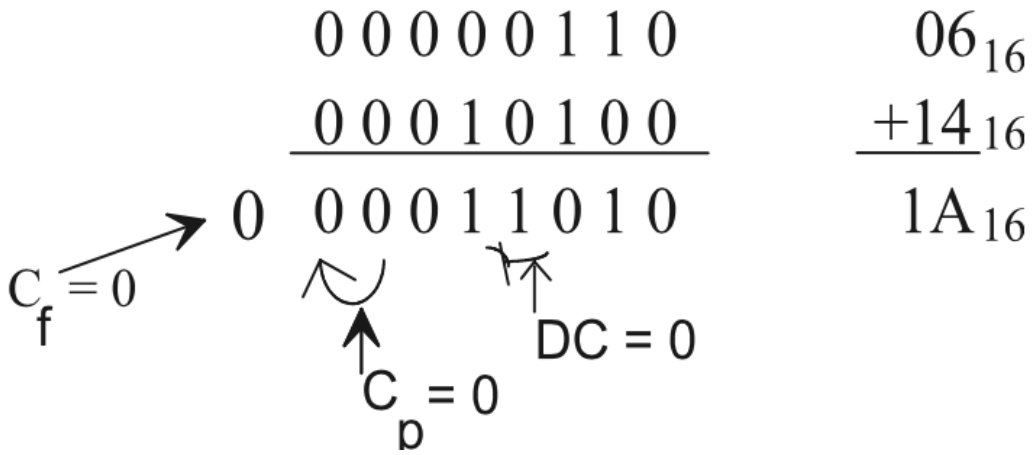


FIGURE 5.5 SR (Status Register)

PIC18F Register Architecture

Consider adding 06_{16} with 14_{16} as follows:

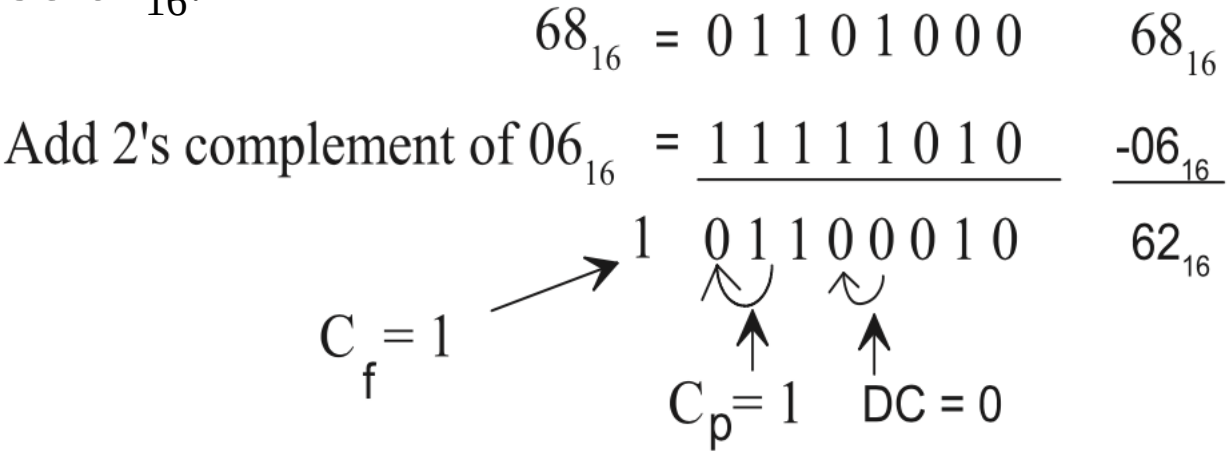


Let C_f be the final carry (carry out of the most significant bit or sign bit) and C_p be the previous carry (carry out of bit 6 or seventh bit)

$C=C_f=0$, DC=0 (No carry from bit 3 to bit 4), Z=0 (nonzero result), $OV = C_f \oplus C_p = 0 \oplus 0 = 0$ (meaning correct result), N = 0 (Most significant bit of the result is 0 indicating a positive number).

PIC18F Register Architecture

Next, consider subtracting 06_{16} from 68_{16} using 2's complement. The result will be 62_{16}.



C (Borrow) = one's complement of C= 0, DC = 0 (No carry from bit 3 to bit 4), Z = 0 (nonzero result), OV = $C_f \oplus C_p = 1 \oplus 1 = 0$ (meaning correct result), N = 0 (Most significant bit of the result is 0 indicating a positive number).

PIC18F Memory Organization

- Three types of memories are utilized in the PIC18F. They are Flash memory, SRAM and EEPROM.
- The Flash memory is used to store programs.
- The SRAM contains data. Some versions of the PIC18F family contain EEPROM along with SRAM to hold data. Note that SRAM is a volatile read/write memory. The EEPROM, on the other hand, is a nonvolatile memory.

PIC18F Memory Organization

- Data can be specified in the instruction as *immediate data* or by the *address of data*. For example, the MOVLW 2 instruction will load the WREG register with immediate data 2.
- The MOVF 0x20, W instruction will load the contents of data memory address 0x20 into WREG. Now, if the contents of 0x20 is 02H, then 02H will be loaded into WREG.

PIC18F Memory Organization

- The EEPROM is used for long-term storage of critical data. The EEPROM is normally used as a read-mostly memory since its read time is faster than write times. It is not directly mapped in either the register file or program memory space but is indirectly addressed through the Special Function Registers (SFRs). One of the main advantages of including the EEPROM in the PIC18F is that all critical data stored in the EEPROM can be protected from reading or writing by other users.

PIC18F Memory Organization

- The data memory in PIC18F devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes (2^{12}) of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each.

PIC18F Program Memory

- The Program Counter (PC) contains the address of the instruction to be fetched for execution. The maximum size of the PIC18F PC is 21 bits wide. The PC addresses bytes in the program memory.
- With a maximum of 21-bit address, the PIC18F address range goes from 21 zeros to 21 ones (addresses 0x 000000 to 0x1FFFFF). This 21-bit address provides a maximum program memory size of $2^{21}(2^{10} \times 2^{10} \times 2^{1})$.

PIC18F Data Memory

- The PIC18F data memory is also called “file register” or “data register” by Microchip. Figure 5.7 shows the data memory organization of the PIC18F4321. The size of the PIC18F4321 address is 12-bit wide and the contents are 8-bit wide. With 12-bit address, the PIC18F4321 can have a data memory of up to 4096 (2^{12}) bytes; 12-bit address (addresses from 0x000 to 0xFFF) is needed to address each location.

PIC18F Data Memory

- The data memory is divided into 16 banks of 256 bytes each with a total of 4k bytes (16 x 256 = 4096 bytes = 4k bytes). Upper four bits of the 12-bit address specifies the bank number, and is contained in the low four bits of BSR with upper four bits of BSR as 0's.

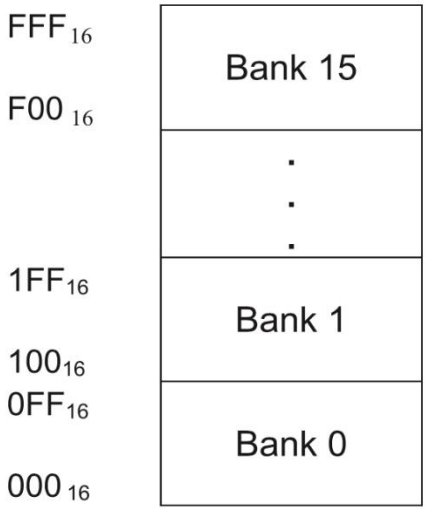


FIGURE 3.1 PIC18F data memory.

PIC18F Data Memory

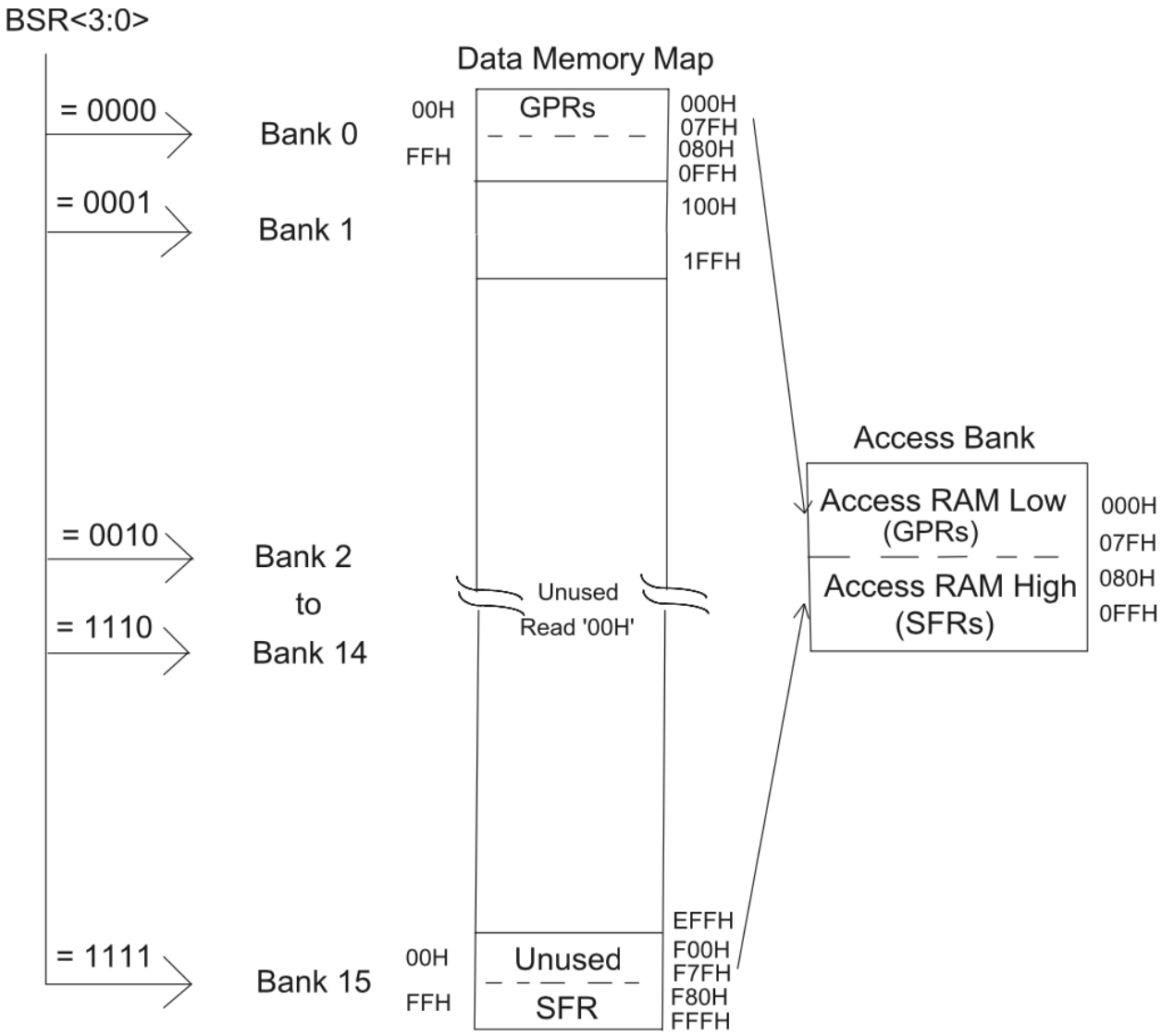


FIGURE 5.7 PIC18F4321 Data Memory Map

PIC18F Data Memory

- The low eight bits of the 12-bit address provide the 8-bit address in a bank from addresses 0x00 to 0xFF. For example, 12-bit address 0x0FE will mean that upper four bits 0H of 0x0FE will specify bank 0 and the lower 8 bits will provide the address 0xFE.

PIC18F Data Memory

- Upon hardware reset, the programmer can access a special bank called the "Access Bank". Note that the access bank consists of the LOW 128 bytes of RAM (addresses 0x00 - 0x7F) in bank 0 called "Access RAM Low", and the last 128 bytes of RAM (addresses 0x80 - 0xFF) in bank F called "Access RAM High" as shown in Figure 5.7. Also, the programmer can access the registers in the access bank without the BSR. This provides faster execution of programs. For simplicity, the access bank LOW (addresses 00H - 7FH in bank 0) will be used as data memory in this book.

PIC18F Data Memory

- The Access RAM Low contains GPRs (General Purpose Registers) while the Access RAM High contains SFRs.
- GPRs are data registers used for accessing data during programming. The purpose of SFRs was already explained earlier. A list of some of these SFRs is given in Table 5.1.

PIC18F Data Memory

TABLE 5.1 Selected Special Function Registers (SFRs)

Address	Name	Description
0xFFF	TOSU	Top of stack (Upper 5 bits)
0xFFE	TOSH	Top of stack (High byte)
0xFFD	TOSL	Top of stack (Low byte)
0xFFC	STKPTR	Stack Pointer
0xFFB	PCLATU	Program Counter Latch (Upper 5 bits)
0xFFA	PCLATH	Program Counter Latch (Upper byte)
0xFF9	PCL	Program Counter Latch (Lower byte)
0xFF8	TBLPTRU	Table Pointer (Upper 5 bits)
0xFF7	TBLPTRH	Table Pointer (High byte)
0xFF6	TBLPTRL	Table Pointer (Low byte)
0xFF5	TABLAT	Table Latch
0xFF4	PRODH	Product Register (High byte)
0xFF3	PRODL	Product Register (Low byte)
0xFEF	INDF0(**1**)	Indirect File Register 0; associated with FSR0
0xFEE	POSTINC0(**1**)	Postincrement Pointer 0; uses FSR0
0xFED	POSTDEC0(**1**)	Postdecrement Pointer 0; uses FSR0
0xFEC	PREINC0(**1**)	Predecrement Pointer 0 ; uses FSR0
0xFEB	PLUSW0(**1**)	Add FSR0 to WREG and uses as pointer for data registers
0xFEA	FSR0H	File Select Register 0 (High byte)
0xFE9	FSR0L	File Select Register 0 (Low byte)
0xFE8	WREG	Working Register (Accumulator)
0xFE7	INDF1(**1**)	Indirect File Register 1; associated with FSR1
0xFE6	POSTINC1(**1**)	Postincrement Pointer 1; uses FSR1
0xFE5	POSTDEC1(**1**)	Postdecrement Pointer 1; uses FSR1
0xFE4	PREINC1(**1**)	Predecrement Pointer 1 ; uses FSR1
0xFE3	PLUSW1(**1**)	Add FSR1 to WREG and uses as pointer for data registers
0xFE2	FSR1H	File Select Register 1 (High byte)
0xFE1	FSR1L	File Select Register 1 (High byte)
0xFE0	BSR	Branch Select Register
0xFDA	FSR2H	File Select Register 2 (High byte)
0xFD9	FSR2L	File Select Register 2 (Low byte)
0xFD8	SR	Status Register

PIC18F Data Memory

- Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. For the PIC18F, this is accomplished with a RAM banking scheme. This divides the memory space into 16 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 12- bit address, or an 8-bit low-order address and a 4-bit Bank Pointer.

PIC18F Data Memory

- To access a memory location from one bank to a memory location in a different bank, bank switching is required. For example, to access address F56H in bank F from address 150H in bank 1, we can use the MOVLB K instruction where K is an 8-bit number. The low four bits of K are used to specify the bank, and the upper four bits are always cleared to 0's. For example, in order to switch from bank 1 to bank F, the instruction MOVLB 0x0F can be used. This instruction will load 0FH into BSR.

PIC18F Data Memory

- However, programs will not work if the programmer forgets about bank switching. To facilitate access for the most commonly used data memory locations, the data memory is configured with an “Access Bank”, which allows users to access a mapped block of memory without bank switching.
- The lower half is known as the “Access RAM” and is comprised of GPRs . This upper half is also where the device’s SFRs are mapped.

PIC18F Data Memory

- These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address (Figure 5.7). It is convenient to use access bank for file registers. The user does not have to worry about bank switching. Hence, one should use access bank whenever possible. Note that upon power-up, the PIC18F uses the access bank of the file registers as the default bank.

PIC18F Addressing Modes

- The manner in which a microcontroller specifies location(s) of operand(s) and destination addresses.
  1. Literal or Immediate Addressing Mode
  2. Inherent or Implied Addressing Mode
  3. Direct or Absolute Addressing Mode
  4. Indirect Addressing Mode
  5. Relative Addressing Mode
  6. Bit Addressing Mode

PIC18F Addressing Modes

- Literal or Immediate Addressing Mode
  - The operand data is a literal or constant data. Immediate data is part of the instruction.
  - Constant data can be moved into the WREG or any other specified register such as BSR. However, constant data cannot be moved into file registers using the Literal or Immediate mode.
- An example is the MOVLW 0x2A instruction in which MOVLW uses immediate mode, and moves 8-bit data 2AH into the WREG register.

PIC18F Addressing Modes

- Inherent or Implied Addressing Mode
  - In this mode, instructions do not require operands. These instructions are also called *no-operand instructions.*
  - An example of an instruction with inherent or implied mode is the DAW instruction. The DAW is a no-operand instruction. It adjusts the sum (adds 6 for BCD correction) in the WREG register stored after addition of two 8-bit packed BCD numbers. Note that the DAW instruction implicitly or inherently uses the WREG register .

PIC18F Addressing Modes

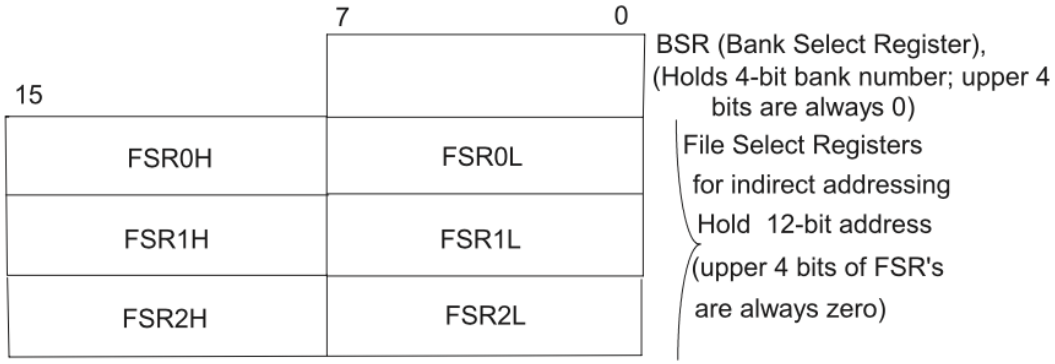
- Direct or Absolute Addressing Mode
  - The address is included as part of the instruction. This address specifies either a data register address in one of the banks of data SRAM or a location in the Access Bank as the data source for the instruction.
- Example: MOVWF 0x50. The MOVWF 0x50 instruction moves the contents of the WREG register into a file register in the data SRAM whose address is 0x50. The contents of WREG are unchanged.

PIC18F Addressing Modes

- Direct or Absolute Addressing Mode
  - Note that the 12-bit address, 0x050 in the access bank, is specified by 8-bit number 0x50 in the MOVWF instruction since the upper four bits (0H) specify the bank number 0.
  - The 8-bit address (0x50) is included in the MOVWF instruction. Addresses are specified as 8-bit numbers while the bank number is specified by the access bank or BSR.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - A register is used as a pointer to an address in the data memory. In the PIC18F, three registers, namely FSR0, FSR1, and FSR2 are used for this purpose. Note that “FSR” stands for *file select register*. Each of these 16-bit registers is divided into low byte and high byte as follows: FSR0 as FSR0H and FSR0L, FSR1 as FSR1H and FSR1L, FSR2 as FSR2H and FSR2L. Each of these registers holds a 12-bit memory address (stored in low 12 bits) to point to a data memory location.



PIC18F Addressing Modes

- Indirect Addressing Mode
  - Each FSR is associated with an SFR (Special Function Register) called INDF (Indirect File) as follows: FSR0 is associated with SFR called INDF0, FSR1 is associated with SFR called INDF1, and FSR2 is associated with SFR called INDF2.
  - The FSR's can be initialized using the LFSR (Load FSR) instruction.

PIC18F Addressing Modes

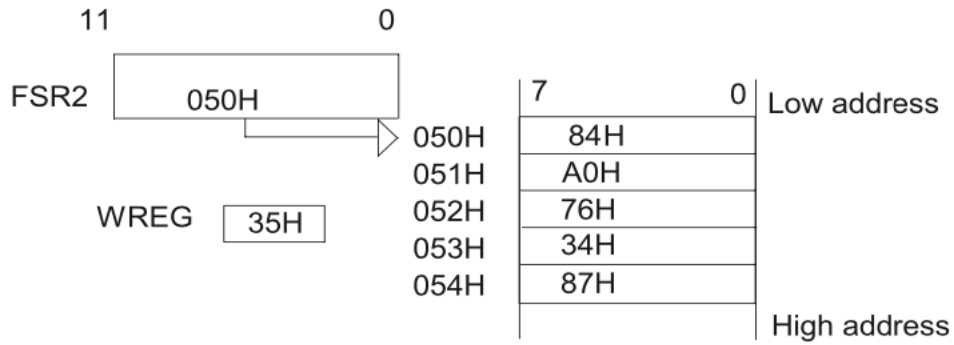
- Indirect Addressing Mode

```
LFSR    0, 0x0010       ; Load  0010H into FSR0
LFSR    1, 0x0040       ; Load 0040H into FSR1
LFSR    2, 0x0080       ; Load 0080H into FSR2
```

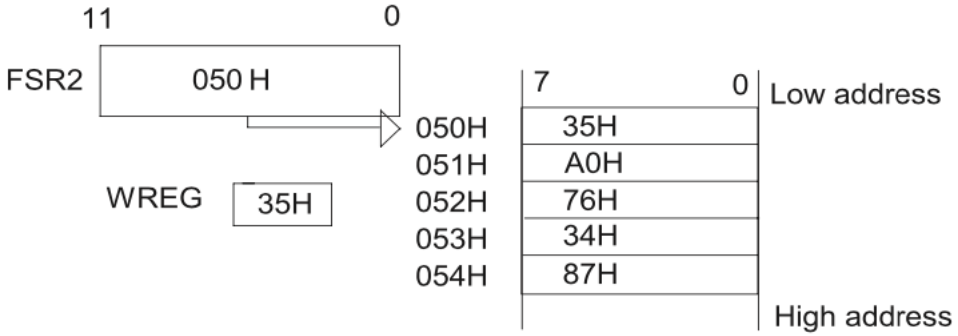
  - After initializing one of the FSR's, data can be moved into a RAM location indirectly using the associated SFR called INDF.
  - For example, in order to move the contents of the WREG register into a 12-bit data RAM location 050H using the FSR2 register indirectly, the following PIC18F instruction sequence can be used:

```
MOVLW   0x35        ; Move 35H into WREG
LFSR    2, 0x50     ; Initialize FSR2 with the RAM location 050H
MOVWF   INDF2       ; Move contents of WREG (35H) into a data RAM address pointed
                    ; to by FSR2 (address 050H) since INDF2 is associated with FSR2
```

PIC18F Addressing Modes



(a) Memory contents after execution of MOVLW and LFSR instructions in the example for indirect addressing



(b) Memory contents after execution of MOVWF in the example for indirect addressing

FIGURE 5.8 Illustration of the Indirect Addressing Mode

PIC18F Addressing Modes

- Indirect Addressing Mode
- The PIC18F provides the indirect addressing mode with four sub-modes as
  1. Indirect with postincrement mode
  2. Indirect with postdecrement mode
  3. Indirect with preincrement mode
  4. Indirect with 8-bit indexed mode

PIC18F Addressing Modes

- Indirect Addressing Mode
  - Each FSR is comprised of two 8-bit registers: FSRH and FSRL. Also, each FSR has a corresponding INDFx register (INDF0- INDF3) used for indirect addressing.
  - In addition to these registers, the four sub-modes utilize four Special Function Registers (SFRs) namely: POSTINC, POSTDEC, PREINC, and PLUSW.
  - These Special Function Registers use the contents of FSR0 through FSR2 to achieve the four sub-modes.

PIC18F Addressing Modes

- Indirect Addressing Mode
- The SFRs are utilized by the sub-modes as follows:

- Indirect with postincrement mode uses SFR's called POSTINC0 through POSTINC2 registers. POSTINC0 is associated with FSR0, POSTINC1 with FSR1, and POSTINC2 with FSR2.

- Indirect with postdecrement mode uses SFR's called POSTDEC0 through POSTDEC2. POSDEC0 is associated with FSR0, POSDEC1 with FSR1, and POSDEC2 with FSR2.

- Indirect with preincrement mode uses SFR's called PREINC0 through PREINC2 registers. PREINC0 is associated with FSR0, PREINC1 with FSR1, and PREINC2 with FSR2.

- Indirect with 8-bit indexed mode uses SFR's called PLUSW0 through PLUSW2. PLUSW0 is associated with FSR0, PLUSW1 with FSR1, and PLUSW2 with FSR2.

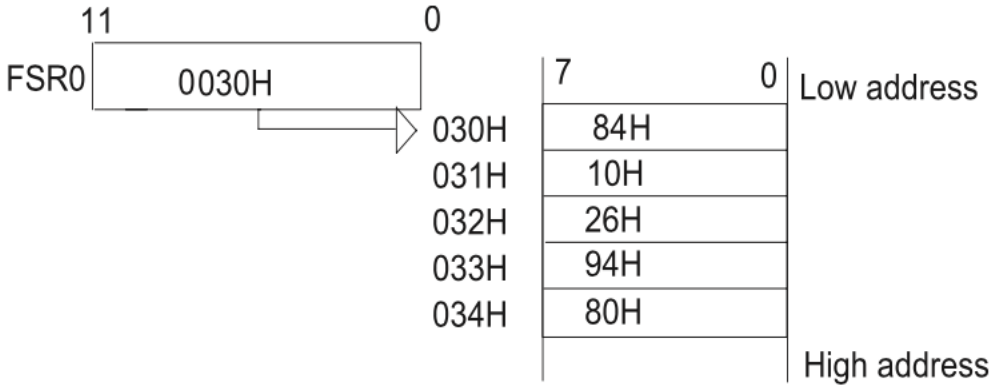
PIC18F Addressing Modes

- Indirect Addressing Mode
  - Indirect with postincrement mode reads the contents of the FSR specified in the instruction, using a low 12-bit value as the address for the operation to be performed. *The specified FSR is then incremented by 1 to point to the next address.* As mentioned before the Special Function Register, POSTINC, is used for this purpose.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - Consider CLRF POSTINC0. Prior to execution of this instruction, suppose that the 16-bit contents of FSR0 are 0030H, and the 8-bit contents of the data memory location addressed by 12-bit address 030H are 84H.
  - After execution of the instruction, CLRF POSTINC0, the contents of address 030H, and will be cleared to 00H, and the contents of FSR0 will be incremented by 1 to hold 0031H. This may be used as a pointer to the next data. This is depicted in Figure 5.9.

PIC18F Addressing Modes



(a) Memory contents before execution of CLRF POSTINC0.

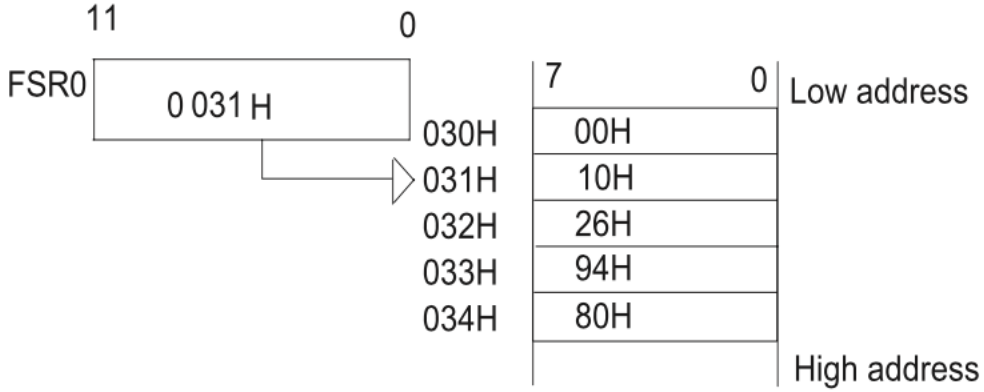


FIGURE 5. 9 Illustration of Indirect with Postincrement Mode

PIC18F Addressing Modes

- Indirect Addressing Mode
  - The postincrement mode is typically used with memory arrays stored from LOW to HIGH memory locations. For example, to clear 20 bytes starting at data memory address 030H and above, the instruction sequence in Figure 5.10 can be used.

```
        MOVLW   D'20'       ; Move 20 decimal into WREG
        MOVWF   0x10        ; Initialize Counter 0x10 with 20 decimal
        LFSR    0,0x30      ; Initialize pointer FSR0 with starting address 0x030
REPEAT  CLRF    POSTINC0    ; Clear a location to 0 and  increment FSR0 by 1
        DECF    0x10,F      ; Decrement Counter by 1
        BNZ     REPEAT      ; Branch to  REPEAT if  Zero flag = 0, otherwise
                            ; go to the next instruction
```

FIGURE 5.10 Instruction sequence for illustrating the postincrement mode

PIC18F Addressing Modes

- Indirect Addressing Mode
  - MOVLW D’20’ moves 20 decimal into WREG while MOVWF 0x10 moves the contents of WREG into address 010H. This will initialize the counter register with 20 decimal. The LFSR 0,0x30 loads FSR0 with 0030H; 030H is the address of the first byte in the array to be cleared to 0. The CLRF POSTINC0 clears the contents of the data memory addressed by FSR0 to 0. Since the 16-bit contents of FSR0 are 0030H, contents addressed by the the low 12 bits (030H) of FSR0 are cleared to 0.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - The DECF 0x10,F decrements the contents of data register 010H by one and then places the result in the data register 010H. After the first pass, data register 010H will contain 19 decimal.
  - The BNZ REPEAT instruction checks if Z flag in the flag register is 0. Note that after first iteration, Z = 0 since the contents of counter are nonzero (19) after execution of DECF. The program branches to label REPEAT, and the loop will be performed 20 times clearing 20 bytes of the array to 0's.

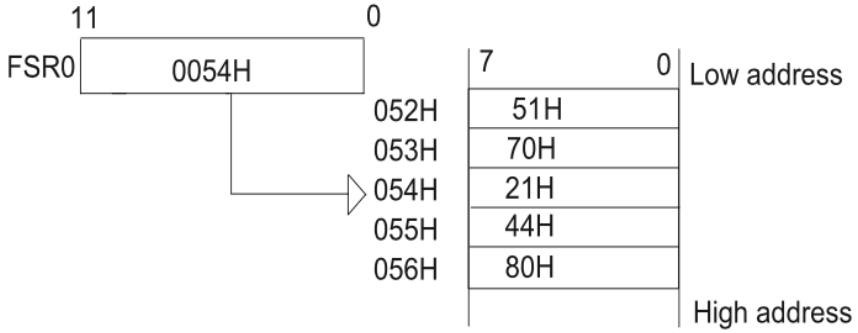
PIC18F Addressing Modes

- Indirect Addressing Mode
  - Indirect with postdecrement mode reads the contents of the FSR specified in the instruction, and uses a low 12-bit value as the address for the operation to be performed. The specified FSR is then decremented by 1. The Special Function Register, POSTDEC, is used for this purpose.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - Consider CLRF POSTDEC0. Prior to execution of this instruction, suppose that the 16-bit contents of FSR0 are 0054H, and the 8-bit contents of the data register addressed by 12-bit address 054H are 21H. After execution of the instruction, CLRF POSTDEC0, the contents of address 054H will be cleared to 00H and the contents of FSR0 will be decremented by 1 to hold 0053H.

PIC18F Addressing Modes



(a) Memory contents before execution of CLRF POSTDEC0.

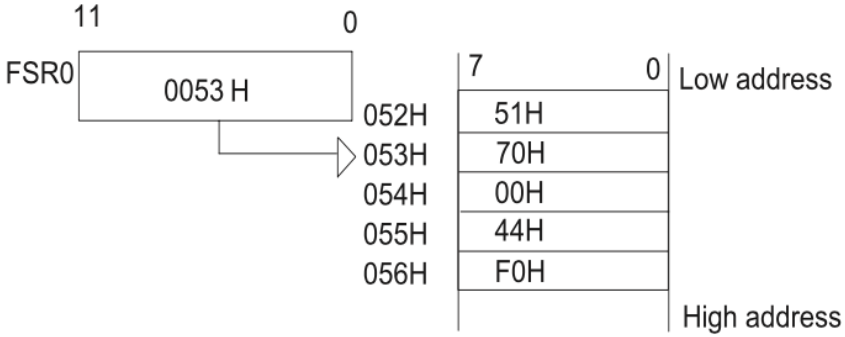


FIGURE 5.11 Illustration of Postdecrement mode

PIC18F Addressing Modes

- Indirect Addressing Mode
  - The postdecrement mode can be used with arrays stored from HIGH to LOW addresses.
  - For example, to clear 100 bytes starting at address 044H and below, the instruction sequence in Figure 5.12 can be used.

```
        MOVLW   D'100'      ; Move 100 decimal into WREG
        MOVWF   0x20        ; Initialize Counter reg (0x20) with 100 decimal
        LFSR    0,0x44      ; Initialize pointer FSR0 with starting address 044H
REPEAT  CLRF    POSTDEC0    ; Clear a location to 0 and  decrement FSR0 by 1
        DECF    0x20,F      ; Decrement Counter by 1
        BNZ     REPEAT      ; Branch to  REPEAT if  Zero flag = 0, otherwise
                            ; go to the next instruction
```

FIGURE 5.12 Instruction sequence for illustrating posdecrement mode

PIC18F Addressing Modes

- Indirect Addressing Mode
  - Indirect with preincrement mode. The contents of the FSR are incremented by 1 to contain the next address. The Special Function Register, PREINC, is used for this purpose.
  - Consider CLRF PREINC0. Prior to execution of this instruction, suppose that the 16-bit contents of FSR0 are 0030H, and the 8-bit contents of the data register addressed by 12-bit address 031H are 84H. After execution of the instruction, CLRF PREINC0, the contents of FSR0 will be incremented by one to hold 0031H. The contents of data register with address 031H will be cleared to 00H.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - Indirect with 8-bit indexed mode adds the contents of the FSR specified in the instruction with the 8-bit contents of the WREG register. The sum is used as an address of a data register in the RAM. The instruction is then executed using this data. The contents of the specified FSR and WREG are unchanged.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - Consider CLRF PLUSW2. Prior to execution of this instruction, suppose that the 16-bit contents of FSR2 are 0020H, the 8-bit contents of WREG are 04H, and the 8-bit contents of address 024H in data RAM are 37H. After execution of CLRF PLUSW2, the contents of the data register 024H will be cleared to 00H. The contents of FSR2 and WREG are 0020H (unchanged) and 04H (unchanged) respectively.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - The indirect with 8-bit indexed mode can be used for code conversion. Two 8-bit ports (Port C and Port D) of the PIC18F will be used to illustrate this example.
  - Assume that a PIC18F is interfaced to an ASCII keyboard via port C and to an EBCDIC printer via port D. Suppose that it is desired to enter numerical data via the ASCII keyboard and then print them on the EBCDIC printer.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - Note that numerical data entered into the PIC18F via the keyboard will be in ASCII code. Since the printer only understands EBCDIC code, an ASCII-to-EBCDIC code conversion program is required.
  - As discussed in Chapter 1, the ASCII codes for numbers 0 through 9 are 30H through 39H, while the EBCDIC codes for numbers 0 to 9 are F0H to F9H.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - An array can be stored in the access bank starting at address 030H; EBCDIC code F0H (decimal 0) at address 030H, EBCDIC code F1H (decimal 1) at address 031H, and so on.
  - Now, suppose that ‘1’ is pushed on the ASCII keyboard. The PIC18F can input this data via PORT C into WREG as 31H (ASCII for 1). The EBCDIC printer will print ‘1’ if the PIC18F outputs F1H to Port D. This can be accomplished by initializing one of the FSR’s (assume FSR2 in this example) with 0000H and then execute MOVFF instruction using indirect with 8-bit indexed mode as follows:

PIC18F Addressing Modes

```
LFSR     2, 0x0000          ; Load 0000H into FSR2
MOVFF    PLUSW2,PORTD       ; Add WREG to FSR2, move
                            ; the byte content  of that address to Port D
```

In the above instruction sequence, since the content of WREG, in this example, is 31H, the instruction MOVFF PLUSW2,PORTD will output the contents of address 031H (F1H), to the EBCDIC printer. The printer will then print '1'.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - Addressing modes for TBLRD* and TBLWT* instructions. In the PIC18F, the address and data sizes of the program memory and data memory are not compatible. The 16- bit contents of the program memory are addressed by 21-bit addresses while the 8-bit contents of the data memory (data registers) are addressed by 12-bit addresses.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - In order to transfer data between program and data memories, the PIC18F is provided with two Special Function Registers, namely 21-bit TBLPTR (Table Pointer) and 8-bit TABLAT (Table Latch).
  - The TABLPTR is used as a pointer for program memory. The TABLAT is used to hold a byte to be transferred from program memory to data memory.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - Two instructions, namely TBLRD* and TBLWT*, are provided with the PIC18F in order to perform data transfers between program and data memories.
  - The TABLRD* instruction reads a byte from the program memory into the TABLAT register that can be moved into a data register. The TBLWT* writes a data byte from the TABLAT register (already moved from a data register) into the program memory.

PIC18F Addressing Modes

- Indirect Addressing Mode
  - The TBLRD* and TBLWT* instructions can use register indirect, postincrement, postdecrement, and preincrement modes. These instructions along with the addressing modes can be used to transfer a block of data between program memory and data memory.

PIC18F Addressing Modes

- Relative Addressing Mode
  - All conditional and one unconditional branch (BRA) instructions in the PIC18F use relative addressing mode.
  - The conditional branch instructions in the PIC18F are based on four flags, namely C, Z, OV, and N. Each conditional branch instruction specifies an 8-bit offset. This offset is a two’s complement signed number. This means that for forward branching, the range of the offset value is from 00H to 7FH. For backward branching, this range varies from 80H to FFH.

PIC18F Addressing Modes

- Relative Addressing Mode
  - Since conditional branch instructions are 16-bit wide in the PIC18F, the PC (Program Counter) is incremented by 2 to point to the next instruction while executing the conditional branch instruction. The offset is multiplied by 2 and then added to PC+2 to find the branch address if the condition is true.
  - Note that the offset is multiplied by 2 since the contents of the PC must always be an even number for 16- and 32-bit instruction lengths.

PIC18F Addressing Modes

- Relative Addressing Mode
  - Consider BNC 0x03. BNC stands for “Branch if no carry”. If the C (carry flag) in the Status register is 0, then the PC is loaded with the (PC + 2 + 03H x 2). When the PIC18F executes the BNC instruction, the PC points to the next instruction. This means that if BNC is located at address 0050H in program memory, the PC will contain 0052H (PC + 2) when the PIC18F executes BNC. Hence, if C = 0, then after execution of the BNC 0x03 instruction, the PC will be loaded with address 0058H (0052H + 03H x 2). The program will branch to address 0058H.

PIC18F Addressing Modes

- Relative Addressing Mode
  - The unconditional branch instruction, BRA (Branch Always), also uses the relative addressing mode. However, the BRA d instruction unconditionally branches to (PC + 2 + d x 2) where offset ‘d’ is a signed 11-bit number specifying a range from -102410 to + 102310 with 0 being positive.

PIC18F Addressing Modes

- Relative Addressing Mode
  - Consider BRA 0x05 that is stored at location 0040H in the program memory. This means that the PC will contain 0042H (PC + 2) when the PIC18F executes BRA. Hence, after execution of the BRA 0x05 instruction, the PC will be loaded with address 004CH (0042H + 05H x 2). The program will branch to address 004CH.

PIC18F Addressing Modes

- Relative Addressing Mode
  - Consider the following

```
        ORG   0x100
HERE    BRA   HERE
```

The machine code for the above instruction is 1101011111111111_2 ($D7FF_{16}$). Note that 1101_2=0xD is the opcode and 11111111111_2 (Eleven 1's sign-extended = -1_{10}) is the offset. During execution of the BRA instruction, the PC points to 0x102. The target branch address = (PC + 2 + d x 2) = 0x102 + (-1) x 2 = 0x100. The instruction HERE BRA HERE unconditionally branches to address 0x100. This is equivalent to HALT instruction in other processors.

PIC18F Addressing Modes

To illustrate the concept of relative branching, consider the following PIC18F disassembled instruction sequence along with the machine code (all numbers in hex):

```
                                    1:          #INCLUDE<P18F4321.INC>
                                    2:          ORG     0x00
0000    0E02    MOVLW   0x2         3:  BACK    MOVLW   0x02
0002    0802    SUBLW   0x2         4:          SUBLW   0x02
0004    E001    BZ      0x8         5:          BZ      DOWN
0006    0E04    MOVLW   0x4         6:          MOVLW   0x04
0008    0804    SUBLW   0x4         7:  DOWN    SUBLW   0x04
000A    E0FA    BZ  0               8:          BZ      BACK
000C    0003    SLEEP               9:          SLEEP
```

Note that all instructions, addresses, and data are chosen arbitrarily. The first branch instruction, BZ DOWN (line 5) at address 0x0004, has a machine code 0xE001. Upon execution of the instruction BZ (branch if Z-flag = 1), the PIC18F branches to label DOWN if Z = 1 (condition true); the PIC18F executes the next instruction at address 0x0006 if Z = 0 (condition false).

PIC18F Addressing Modes

- Relative Addressing Mode
  - The BZ instruction uses the relative addressing mode. This means that DOWN is a positive number (the number of steps forward relative to the current program counter) indicating a forward branch. The machine code 0xE001 means that the op-code for BZ is 0xE0 and the relative 8-bit signed offset value is 0x01 (+1). T. While executing BZ DOWN at address 0x0004, the PC points to address 0x0006 since the program counter is incremented by 2. This means that the program counter contains 0x0008. The offset 0x01 is multiplied by 2 and added to address 0x0006 to find the target branch address where the program will jump if Z = 1.

PIC18F Addressing Modes

- Relative Addressing Mode

The branch address can be calculated as follows:

```
  0x0006   =   0000 0000  0000 0110
+ 0x0002   =   0000 0000  0000 0010   (0x01 is multiplied by 2, and sign-
                                       extended to 16 bits)
-------------------------------------------------------------------
                0000 0000  0000  1000 = 0x0008
```

Hence, the PIC18F branches to address 0x0008 if Z = 1. This can be verified in the instruction sequence above.

PIC18F Addressing Modes

- Relative Addressing Mode
- Consider the second branch instruction, BZ BACK (line 8). Upon execution of BZ BACK, the PIC18F branches to label BACK if Z = 1 (condition true); if Z = 0 (condition false), the PIC18F executes the next instruction. The machine code this instruction at address 0x000A is 0xE0FA, where 0xE0 is the op-code and 0xFA is the signed 8-bit offset value. The offset is represented as an 8-bit two's complement number.

PIC18F Addressing Modes

- Relative Addressing Mode
- Since 0xFA is a negative number (-610), this is a backward jump. Note that while executing BZ BACK at address 0x000A, the PC points to address 0x000C since the program counter is incremented by 2. This means that the program counter contains 0x000C. The offset -6 is multiplied by 2, and then added to 0x000C to find the address value where the program will branch if Z = 1.

PIC18F Addressing Modes

- Relative Addressing Mode

```
   0x000C  =   0000 0000  0000 1100
+  0xFFF4  =   1111  1111  1111 0100 (0xFA is multiplied by 2 and then sign-
                                       extended to 16  bits)
-------------------------------------------------------------------
                1  0000 0000  0000 0000  = 0x0000
                ↗
Ignore final carry
```

The branch address is 0x0000, which can be verified in the instruction sequence above. As mentioned in Chapter 1, in order to add a 16-bit signed number with an 8-bit signed number, the 8-bit signed number must first be sign-extended to 16 bits. The two 16-bit numbers can then be added. Any carry resulting from the addition must be discarded. This will provide the correct answer.

PIC18F Addressing Modes

- Bit Addressing Mode
  - The instructions using the Bit addressing mode directly specify a single bit to be operated on.
  - For example, BCF 0x10,3 will clear bit 3 to 0 in the data register addressed by 0x10. If the contents of the data register 0x10 are 01111000_2 (78H), then after execution of BCF 0x10,3, the contents of data register 0x10 are 01110000_2(70H).