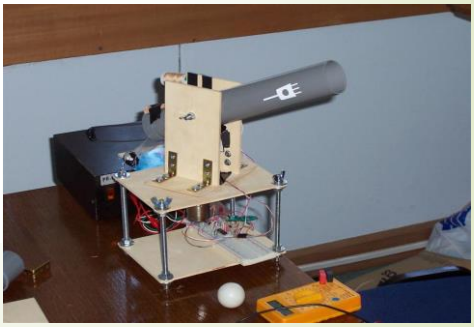
Microcontroladores

PIC18F4550



Ing. Javier Barriga Hoyle

Programación en Assembler

Contenido:

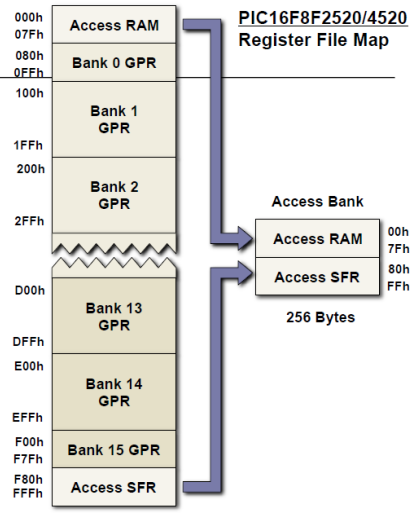
- Organización de la memoria de datos y programa.
- Registros: Wreg (W), Bank Select Register (BSR), Special Function register (SFR), Status, File Select Registers (FSR) y Program Counter (PC).
- Instruction Pipeline flow.
- Modos de direccionamiento: Inherente, Literal o Inmediato, Directo e Indirecto.
- Set de instrucciones del PIC18F.
- Ejemplos.
- Bibliografía.

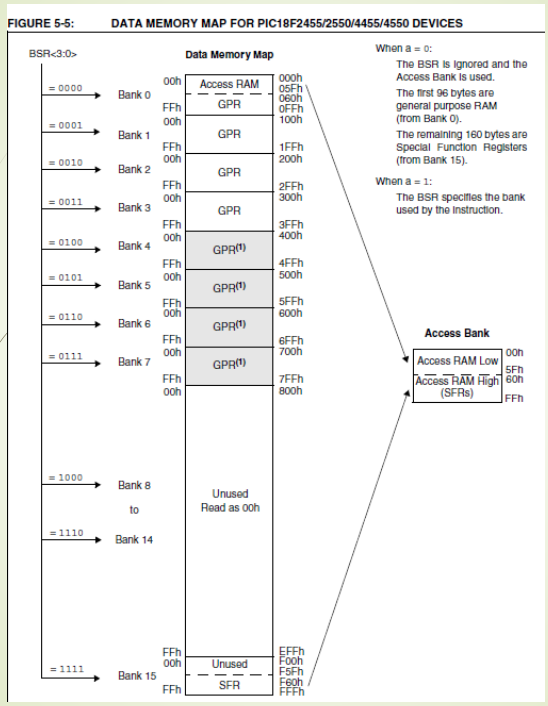
Organización de la memoria datos

Data Memory Organization

Remember:

- Data Memory up to 4k bytes
- Divided into 256 byte banks
- Half of bank 0 and half of bank 15 form a virtual bank that is accessible no matter which bank is selected





Organización de la memoria

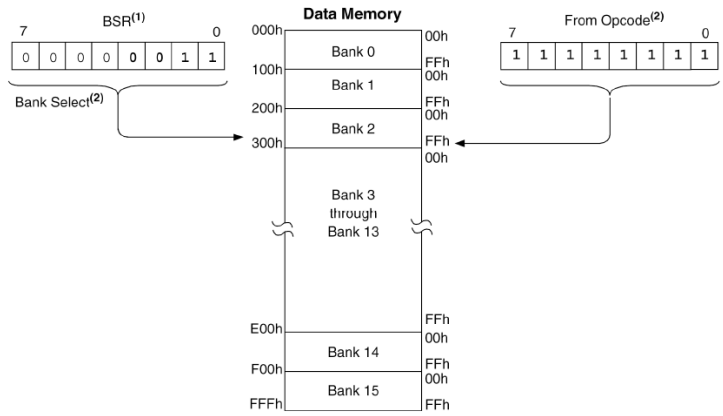
Cuando a=0:
Se ignora el BSR y usamos el acceso al banco.

Los primeros 96bytes de la RAM son de propósito general (del banco 0)

Los 160bytes restantes son registros de funciones especiales (banco 15)

Cuando a=1:
El BSR especifica el banco usado en la instrucción.

Organización de la memoria de datos

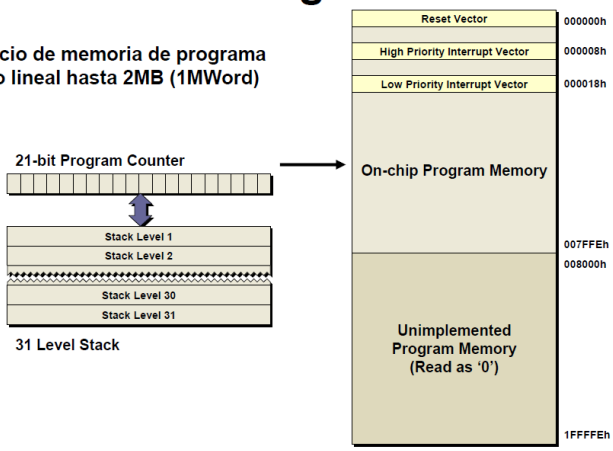


Nota: 1: El bit de acceso a RAM de la instrucción se puede usar para invalidar banco seleccionado (BSR<3:0>) del registro de acceso al banco.

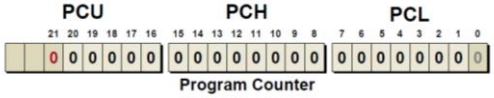
2: La instrucción MOVFF inserta los 12bit de dirección en la instrucción.

Organización de la Memoria de Programa

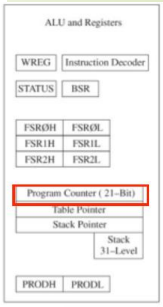
- **Un espacio de memoria de programa continuo lineal hasta 2MB (1MWord)**

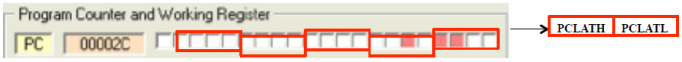


Contador de Programa



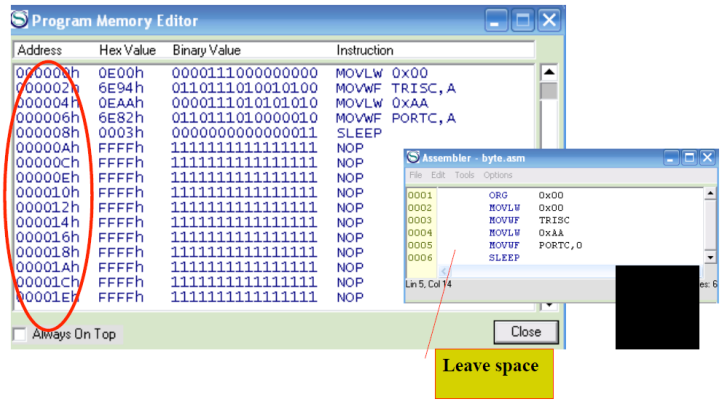
- 21-bit PC permite acceder hasta to 2^{21} = 2MB (1MWord)
- 22nd bit usado para acceder a la configuración de memoria para programar tiempo o via tabla de lectura y escritura
- Contiene la dirección de la siguiente instrucción (pipelining)
- El byte bajo es accesible en la memoria de datos como PCL
- Los byte s altos son accesibles indirectamente via PCLATH/PCLATU
- Bit 0 del PC es siempre '0' excepto cuando se lee o escribe en la memoria de programa via el mecanismo de tablas de lecturaescritura





Puntero de instrucción (PC)

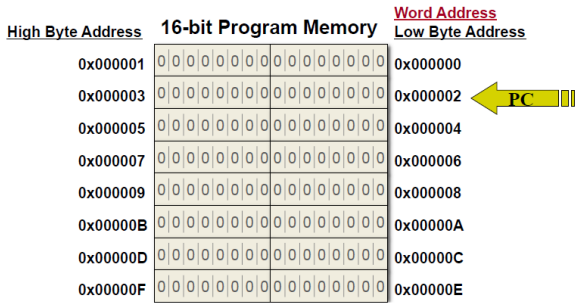
21-Bit PC Example & Program Memory



Puntero de instrucción - PC

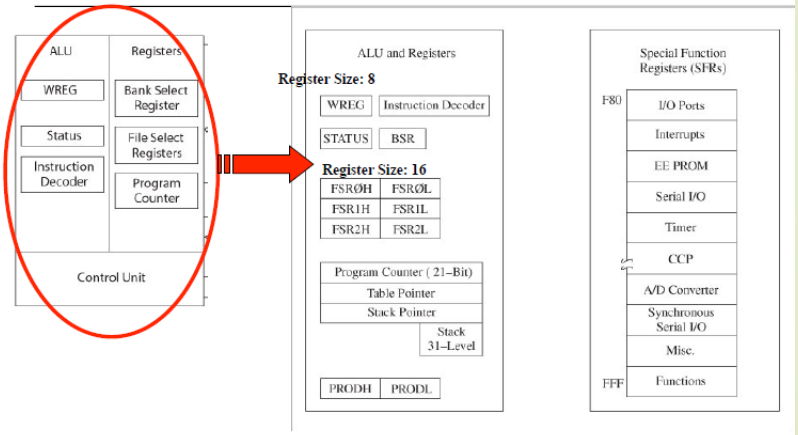
Program Memory is Byte Addressable

- Low byte has even address, high byte has odd address
- Addresses of instructions are always even
- 16-bit wide program memory is byte addressable
- All program instructions will start at an even address
- So if we are jumping 4 instructions ahead, we are actually jumping 8-bytes (or 8 word addresses) ahead



Registros: Arquitectura interna

PIC18F Programming Model (2 of 2)



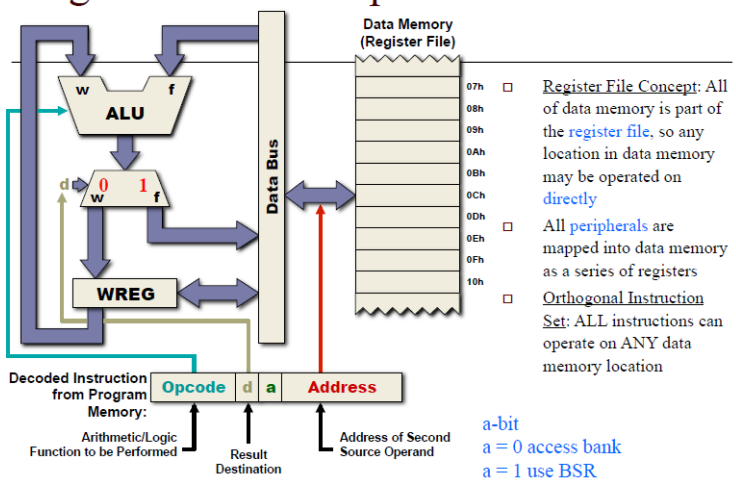
Registros: Arquitectura interna

Registers

- WREG
  - 8-bit Working Register (equivalent to an accumulator)
  - Used for arithmetic and logic operations
- BSR: Bank Select Register (0 to F)
  - 4-bit Register
    - Only low-order four bits are used to provide MSB four bits of a12-bit address of data memory.

Registros: WREG (W)

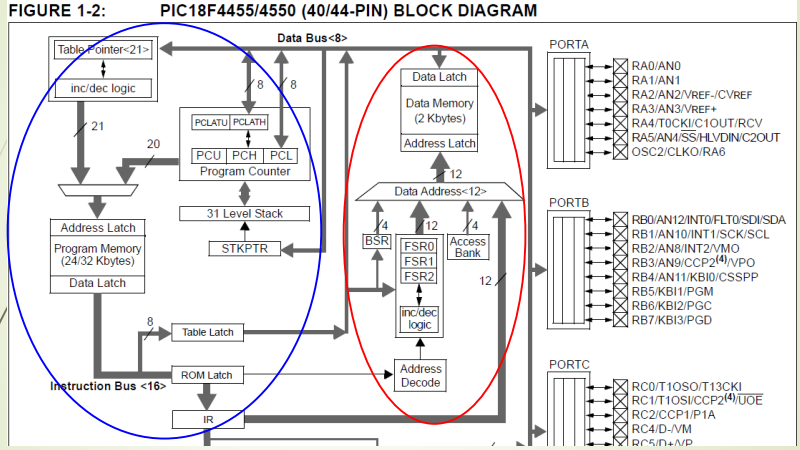
Register File Concept



- Register File Concept: All of data memory is part of the register file, so any location in data memory may be operated on directly
- All peripherals are mapped into data memory as a series of registers
- Orthogonal Instruction Set: ALL instructions can operate on ANY data memory location

Registros: BSR, FSR y PC

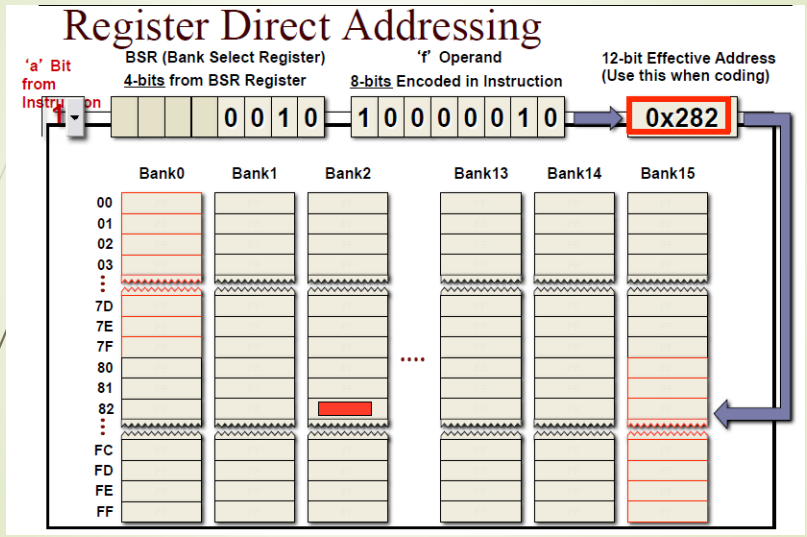
FIGURE 1-2: PIC18F4455/4550 (40/44-PIN) BLOCK DIAGRAM



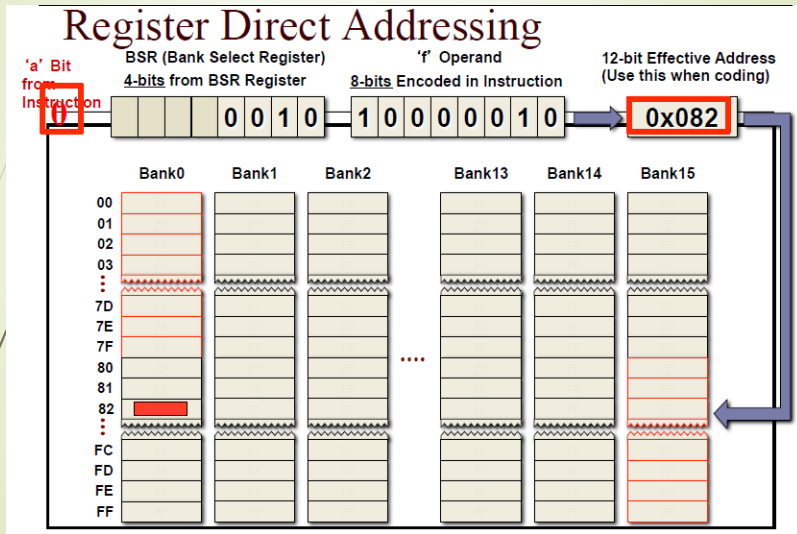
Punteros a Memoria de programa

Punteros a Memoria de datos

Registros: BSR



Registros: BSR



Registros: BSR

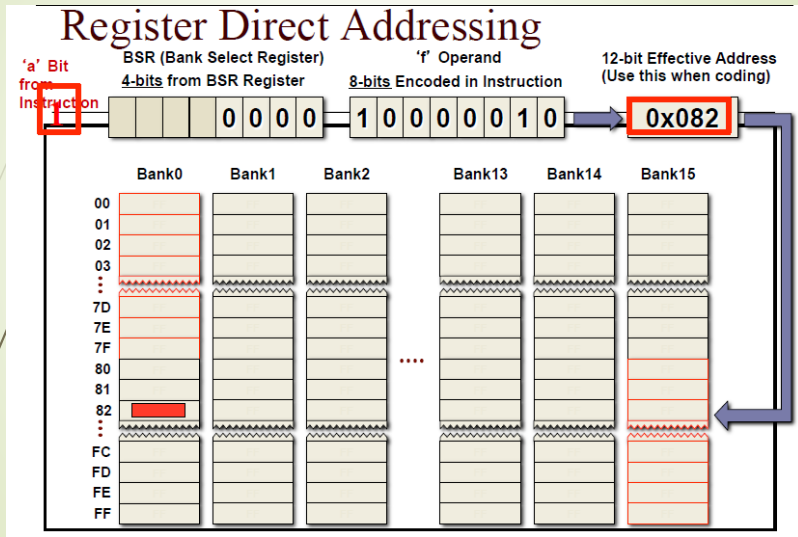


TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2455/2550/4455/4550 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(1)	FBFh	CCPR1H	F9Fh	IPR1	F7Fh	UEP15
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	CCPR1L	F9Eh	PIR1	F7Eh	UEP14
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	CCP1CON	F9Dh	PIE1	F7Dh	UEP13
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	CCPR2H	F9Ch	—(2)	F7Ch	UEP12
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	CCPR2L	F9Bh	OSCTUNE	F7Bh	UEP11
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—(2)	F7Ah	UEP10
FF9h	PCL	FD9h	FSR2L	FB9h	—(2)	F99h	—(2)	F79h	UEP9
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	—(2)	F78h	UEP8
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	ECCP1DEL	F97h	—(2)	F77h	UEP7
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS	F96h	TRISE(3)	F76h	UEP6
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD(3)	F75h	UEP5
FF4h	PRODH	FD4h	—(2)	FB4h	CMCON	F94h	TRISC	F74h	UEP4
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB	F73h	UEP3
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA	F72h	UEP2
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—(2)	F71h	UEP1
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	—(2)	F70h	UEP0
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—(2)	F6Fh	UCFG
FEEh	POSTINC0(1)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—(2)	F6Eh	UADDR
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(3)	F6Dh	UCON
FECh	PREINC0(1)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(3)	F6Ch	USTAT
FEBh	PLUSW0(1)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC	F6Bh	UEIE
FEAh	FSR0H	FCAh	T2CON	FAAh	—(2)	F8Ah	LATB	F6Ah	UEIR
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA	F69h	UIE
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—(2)	F68h	UIR
FE7h	INDF1(1)	FC7h	SSPSTAT	FA7h	EECON2(1)	F87h	—(2)	F67h	UFRMH
FE6h	POSTINC1(1)	FC6h	SSPCON1	FA6h	EECON1	F86h	—(2)	F66h	UFRML
FE5h	POSTDEC1(1)	FC5h	SSPCON2	FA5h	—(2)	F85h	—(2)	F65h	SPPCON(3)
FE4h	PREINC1(1)	FC4h	ADRESH	FA4h	—(2)	F84h	PORTE	F64h	SPPEPS(3)
FE3h	PLUSW1(1)	FC3h	ADRESL	FA3h	—(2)	F83h	PORTD(3)	F63h	SPPCFG(3)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC	F62h	SPPDATA(3)
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB	F61h	—(2)
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA	F60h	—(2)

Note 1: Not a physical register.
2: Unimplemented registers are read as '0'.
3: These registers are implemented only on 40/44-pin devices.

Registros de funciones especiales SFR

Utilizados para el manejo de puertos

Registro de estado (banderas)

Flags in Status Register

B7	B6	B5	B4	B3	B2	B1	BØ
—	—	—	N	OV	Z	DC	C
			Sign	Overflow	Zero	Digit Carry	Carry/Borrow

- C (Carry/Borrow Flag): set when an addition generates a carry and a subtraction generates a borrow
- DC (Digit Carry Flag): also called Half Carry flag; set when carry generated from Bit3 to Bit4 in an arithmetic operation
  - Used for BCD representation
- Z (Zero Flag): set when result of an operation is zero
- OV (Overflow Flag): set when result of an operation of **signed numbers** goes beyond seven bits – if the results fall outside 127 (0x7F) and -128 (0x80)
- N (Negative Flag): set when bit B7 is one

Registro de estado (banderas)

Example: PIC18 Visual Interpreter

ADD: 9F and 52. Which flags will be set?

1

1001 1111
0101 0010

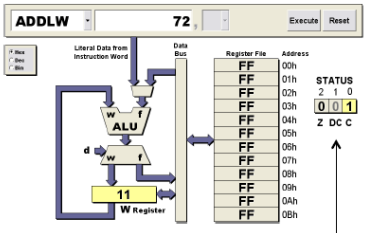
1111 0001
=F1

N=1
OV=0
Z=0
DC=1
C=0

ADD: 9F and 72. Which flags will be set?

1001 1111
0111 0010
---
1 0001 0001
=11

N=0
OV=0 // not signed numbers
Z=0
DC=1
C=1



0111 0010
0001 0001

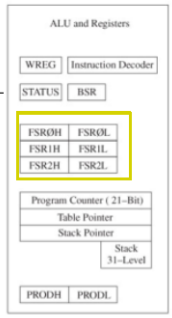
1000 0011
=82

Flag status does not change!

Registro puntero FSR

File Select Registers (FSR)

- Three registers holding 12-bit address of data registers
  - FSR0, FSR1, and FSR2
- File Select Registers composed of two 8-bit registers (FSRH and FSRL)
- Used as pointers for data registers for indirect addressing
  - Associated with index (INDF) registers

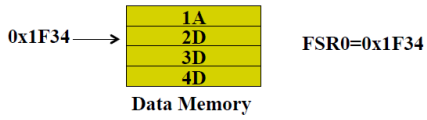


Find FSR0-FSR2 in Special Function Register – page 64 What are the File addresses for each? / How many INDF do you find?

Registro puntero FSR

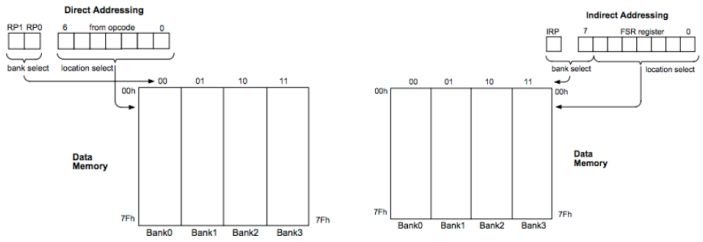
File Select Registers (FSR) – Indirect Addressing

- The main application of FSR is **Indirect Addressing**
  - FSRs will be pointing at the address of the data file and they can be incremented
  - This is much easier than using direct addressing



Registro puntero FSR

Direct and Indirect Addressing



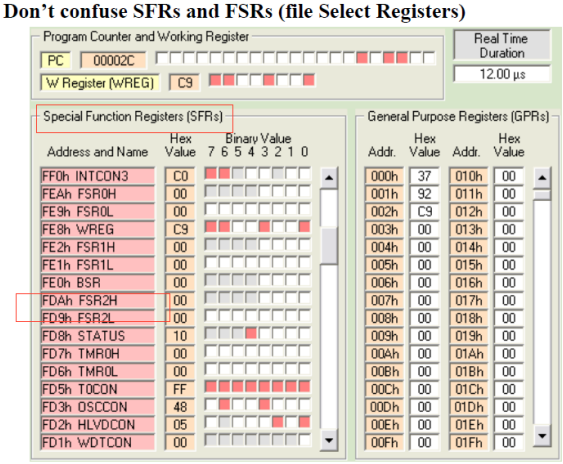
WE WILL DISCUSS THIS IN MORE DETAILS WHEN WE LEARN MORE AOUT COMMANDS!

http://ww1.microchip.com/downloads/en/DeviceDoc/31006a.pdf

Registro puntero FSR

FSR

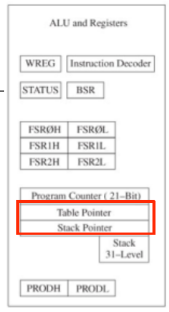
Don't confuse SFRs and FSRs (file Select Registers)



Registro Stack Pointer (SP)

Stack and Table Pointers

- Table Pointer
  - 21-bit register used as a memory pointer to copy bytes between program memory and data registers
- Stack Pointer (SP)
  - Stack is a group of 31 word-size registers used for temporary storage of memory address during execution
  - Requires 5-bit address
  - Saved in STKPTR in SFR
  - Used primarily for saving PC for next program address prior to entering subroutine

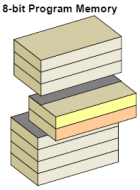


Examine Table 3-1 – Page 64

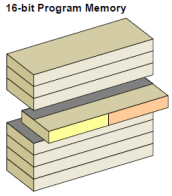
Pipeline de instrucciones

Instruccion de Palabra Larga

8-bit Program Memory



16-bit Program Memory



8-bit Instruction on typical 8-bit MCU

Example: Freescale 'Load Accumulator A':
- 2 Program Memory Locations
- 2 Instruction Cycles to Execute

```ldaa #k```

1	0	0	0	0	1	1	0
k	k	k	k	k	k	k	k

- Ancho de banda Limitado
- Incrementa los requerimientos de memoria

16-bit Instruction on PIC18 8-bit MCU

Example: 'Move Literal to Working Register'
- 1 Program Memory Location
- 1 Instruction Cycle to Execute

```movlw k```

0	0	0	0	1	1	1	0	k	k	k	k	k	k	k	k

- Separate busses allow different widths
- 2k x 16 is roughly equivalent to 4k x 8

 Slide 15

Pipelining de Instrucciones

Pre-Fetched Instruction

```movlw 0x05```

Executing Instruction

```-```

Instruction Cycles

Example Program

```
1  MAIN  movlw  0x05
2        movwf  REG1
3        rcall  SUB1
4        addwf  REG2
         ⋮
51 SUB1  movf   PORTB,w
52       return
53 SUB2  movf   PORTC,w
54       return
```

 Slide 7

Pipelining de Instrucciones

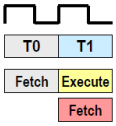
Pre-Fetched Instruction

```
movwf REG1
```

Executing Instruction

```
movlw 0x05
```

Instruction Cycles



Example Program

```
1  MAIN  movlw  0x05
2        movwf  REG1
3        rcall  SUB1
4        addwf  REG2
         ⋮
51 SUB1  movf   PORTB,w
52       return
53 SUB2  movf   PORTC,w
54       return
```


Pipelining de Instrucciones

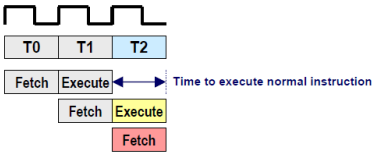
Pre-Fetched Instruction

```
rcall SUB1
```

Executing Instruction

```
movwf REG1
```

Instruction Cycles



Example Program

```
1  MAIN  movlw  0x05
2        movwf  REG1
3        rcall  SUB1
4        addwf  REG2
         ⋮
51 SUB1  movf   PORTB,w
52       return
53 SUB2  movf   PORTC,w
54       return
```

Pipelining de Instrucciones

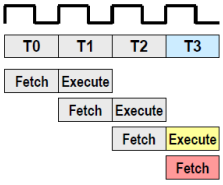
Pre-Fetched Instruction

```
addwf REG2
```

Executing Instruction

```
rcall SUB1
```

Instruction Cycles



Example Program

```
1  MAIN  movlw  0x05
2        movwf  REG1
3        rcall  SUB1
4        addwf  REG2
         ⋮
51 SUB1  movf   PORTB,w
52       return
53 SUB2  movf   PORTC,w
54       return
```


Pipelining de Instrucciones

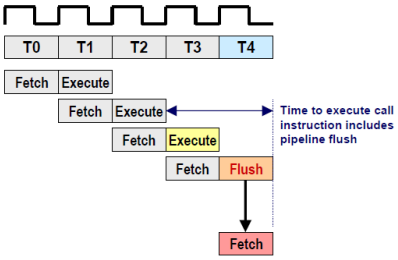
Pre-Fetched Instruction

```
movf PORTB,w
```

Executing Instruction

```
rcall SUB1
```

Instruction Cycles



Example Program

```
1  MAIN  movlw  0x05
2        movwf  REG1
3        rcall  SUB1
4        addwf  REG2
         ⋮
51 SUB1  movf   PORTB,w
52       return
53 SUB2  movf   PORTC,w
54       return
```

Pipelining de Instrucciones

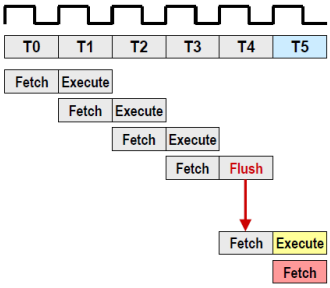
Pre-Fetched Instruction

```
return
```

Executing Instruction

```
movf PORTB,w
```

Instruction Cycles



Example Program

```
1  MAIN  movlw  0x05
2        movwf  REG1
3        rcall  SUB1
4        addwf  REG2
         :
51 SUB1  movf   PORTB,w
52       return
53 SUB2  movf   PORTC,w
54       return
```


Pipelining de Instrucciones

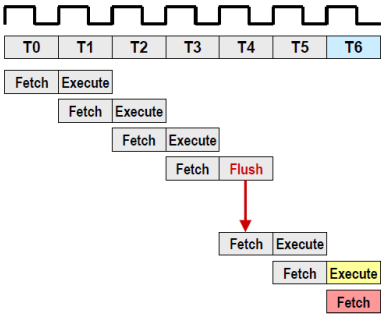
Pre-Fetched Instruction

```
movf PORTC,w
```

Executing Instruction

```
return
```

Instruction Cycles



Example Program

```
1  MAIN  movlw  0x05
2        movwf  REG1
3        rcall  SUB1
4        addwf  REG2
         :
51 SUB1  movf   PORTB,w
52       return
53 SUB2  movf   PORTC,w
54       return
```

Pipelining de Instrucciones

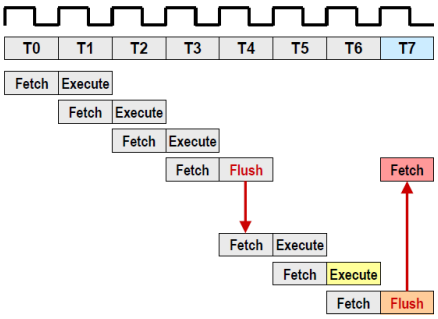
Pre-Fetched Instruction

```
addwf REG2
```

Executing Instruction

```
return
```

Instruction Cycles



Example Program

```
1  MAIN  movlw  0x05
2        movwf  REG1
3        rcall  SUB1
4        addwf  REG2
         ⋮
51 SUB1  movf   PORTB,w
52       return
53 SUB2  movf   PORTC,w
54       return
```

 Slide 14

PIC18 Modos de Direccionamiento

- **Acceso a Memoria de Datos:**

Mode	Example Syntax
Direct	`clrf <reg>, <dst>`
Indirect	`clrf INDFn, <dst>`
Auto Pre-Increment Indirect	`movff PREINCn, <dst>`
Auto Post-Increment Indirect	`movff POSTINCn, <dst>`
Auto Post-Decrement Indirect	`movff POSTDECn, <dst>`
Index Indirect	`movff PLUSWn, <dst>`
Immediate (Literal)	`movlw <const>`

 Slide 22

PIC18 Modos de Direccionamiento

- **Acceso a Memoria de Programa:**

Mode	Example Syntax
Absolute	`goto <addr>`
Relative	`bra <addr>`
Table Read / Write	`tblrd*  tblwt*`
Table Read / Write Post Increment	`tblrd*+  tblwt*+`
Table Read / Write Post Decrement	`tblrd*-  tblwt*-`
Table Read / Write Pre Increment	`tblrd+*  tblwt+*`

Set de instrucciones PIC18

Introduction to PIC18 Instruction Set

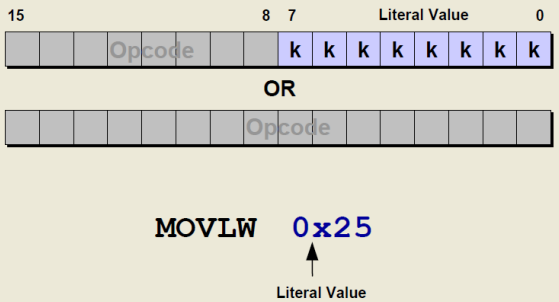
- Includes 77 instructions;
  - 73 one-word (16-bit) long
  - Four two-words (32-bit) long
- Divided into seven groups
  - Move (Data Copy) and Load
  - Arithmetic
  - Logic
  - Program Redirection (Branch/Jump)
  - Bit Manipulation
  - Table Read/Write
  - Machine Control

Move / Copy
Arithmetic
Logic
Branches
Bit Manipulation
Table Read/Write
Machine Control

Formato de instrucción

Revisión del Set de Instrucciones

Literal and Control Operations



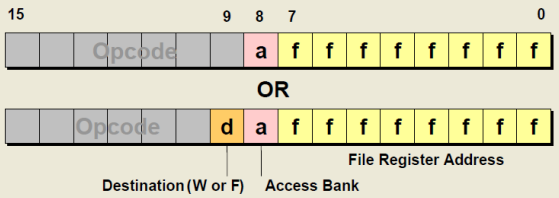

```
MOVLW   0x25
```

Literal Value

Formato de instrucción

Revisión del Set de Instrucciones

Byte Oriented Operations




```
ADDWF   0x25, W, A
```

File Register Address

Destination

Use Access Bank (Optional)

Formato de instrucción

Revisión del Set de Instrucciones

Bit Oriented Operations

15			11	9	8	7							0
Opcode			b	b	b	a	f	f	f	f	f	f	f	f

File Register Address

Bit Position (0-7)

```
BSF  0x25, 3, A
```

File Register Address — Bit Position — Access Bank (Optional)

Formato de instrucción

Revisión del Set de Instrucciones

Byte to Byte Move Operations(2 Words)

15			12	11										0
Opcode				f_s	f_s	f_s	f_s	f_s	f_s	f_s	f_s	f_s	f_s	f_s
Opcode				f_d	f_d	f_d	f_d	f_d	f_d	f_d	f_d	f_d	f_d	f_d

Source Register Address

Destination Register Address

```
MOVFF  0x125, 0x140
```

Source Address — Destination Address

Formato de instrucción

Revisión del Set de Instrucciones

Call and Goto Operations (2 Words)

15			11			8	7						0
			Opcode				n_8	n_7	n_6	n_5	n_4	n_3	n_2	n_1
	Opcode		n_{20}	n_{19}	n_{18}	n_{17}	n_{16}	n_{15}	n_{14}	n_{13}	n_{12}	n_{11}	n_{10}	n_9

```
CALL    0x1125
```

Subroutine Address

Instrucciones: Move o Load

Move / Copy
Arithmetic
Logic
Branches
Bit Manipulation
Table Read/Write
Machine Control

L → W → Fa ↔ Fb

Move and Load Instructions

- MOVLW 8-bit ;Load an 8-bit literal in WREG
- MOVLW 0 x F2
- MOVWF F, a ;Copy WREG in File (Data) Reg.
  ; If a = 0, F is in Access Bank
  ;If a = 1, Bank is specified by BSR
- MOVWF 0x25, 0 ;Copy W in Data Reg.25H
- MOVFF fs, fd ;Copy from one Data Reg. to
  ;another Data Reg.
- MOVFF 0x20,0x30 ;Copy Data Reg. 20 into Reg.30

Instrucciones aritméticas

Move / Copy
Arithmetic
Logic
Branches
Bit Manipulation
Table Read/Write
Machine Control

L W OR Fa W

Arithmetic Instructions (1 of 3)

- ADDLW 8-bit ;Add 8-bit number to WREG
- ADDLW 0x32 ;Add 32H to WREG
- ADDWF F, d, a ;Add WREG to File (Data) Reg.
  ;Save result in W if d =0
  ;Save result in F if d = 1
- ADDWF 0x20, 1 ;Add WREG to REG20 and
  ;save result in REG20
- ADDWF 0x20, 0 ;Add WREG to REG20 and
  ;save result in WREG

Instrucciones aritméticas

Fa W C

Arithmetic Instructions (2 of 3)

- ADDWFC F, d, a ;Add WREG to File Reg. with
  ;Carry and save result in W or F
- SUBLW 8-bit ;Subtract WREG from literal L-W→W
- SUBWF F, d, a ;Subtract WREG from File Reg
- SUBWFB F, d, a ;Subtract WREG from File Reg. F-W→Dest.
  ;with Borrow
- INCF F, d, a ;Increment File Reg.
- DECF F, d, a ;Decrement File Reg.
- COMF F, d, a ;Complement File Reg.
- NEGF F, a ;Take 2's Complement-File Reg.

Instrucciones aritméticas

Arithmetic Instructions (3 of 3)

- MULLW 8-bit ;Multiply 8-bit and WREG **L x W→ PROD**
  ;Save result in PRODH-PRODL
- MULWF F, a ;Multiply WREG and File Reg.
  ;Save result in PRODH-PRODL
- DAW ;Decimal adjust WREG for BCD
  ;Operations

Example:

```
MOVLW   0xA   ;W=A
DAW           ;W=10
```

Instrucciones lógicas

Logic Instructions

- ANDLW 8-bit ;AND literal with WREG
- ANDWF F, d, a ;AND WREG with File Reg. and
  ;save result in WREG/ File Reg.
- IORLW 8-bit ;Inclusive OR literal with WREG
- IORWF F, d, a ;Inclusive OR WREG with File Reg.
  ;and save result in WREG/File Reg.
- XORLW 8-bit ;Exclusive OR literal with WREG
- XORWF F, d, a ;Exclusive OR WREG with File Reg.
  ;and save result in WREG/File Reg.

Instrucciones lógicas

And, XOR, and IOR

A	B	T
0	0	0
0	1	0
1	0	0
1	1	1

(T = A · B)

```
AND   X X X X  X X X X
      0 0 0 0  1 1 1 1
      ---------------
      0 0 0 0  X X X X
      \_____/
```

Cleared to zero

```
XOR   X X X X  X X X X
      0 0 0 0  1 1 1 1
      ---------------
      X X X X  X X X X
               \_____/
```

Toggled

If they are the same→0
If they are different→1

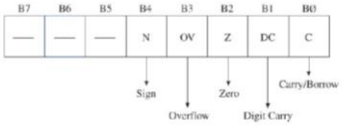
```
IOR   X X X X  X X X X
      0 0 0 0  1 1 1 1
      ---------------
      X X X X  1 1 1 1
               \_____/
```

Set to one

Instrucciones de saltos

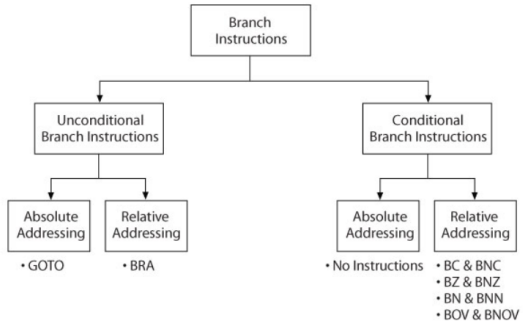
Branch Instructions

- BC n ;Branch if C flag = 1 within + or – 64 Words
- BNC n ;Branch if C flag = 0 within + or – 64 Words (NO CARRY)
- BZ n ;Branch if Z flag = 1 within + or – 64 Words
- BNZ n ;Branch if Z flag = 0 within + or – 64 Words
- BN n ;Branch if N flag = 1 within + or – 64 Words
- BNN n ;Branch if N flag = 0 within + or – 64 Words
- BOV n ;Branch if OV flag = 1 within + or – 64 Words
- BNOV n ;Branch if OV flag = 0 within + or – 64 Words
- GOTO Address: Branch to 20-bit address unconditionally



Instrucciones de saltos

Branch Instructions



Instrucciones de saltos

Call and Return Instructions

- RCALL nn ;Call subroutine within +or – 512 words
- CALL 20-bit, s ;Call subroutine
  ;If s = 1, save W, STATUS, and BSR
- RETURN, s ;Return subroutine
- ;If s = 1, retrieve W, STATUS, and BSR
- RETFIE, s ;Return from interrupt
- ;If s = 1, retrieve W, STATUS, and BSR

Instrucciones orientadas al bit

Bit Manipulation Instructions

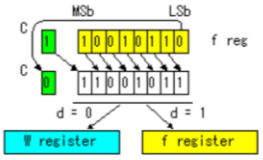
- BCF F, b, a ;Clear bit b of file register. b = 0 to 7
- BSF F, b, a;Set bit b of file register. b = 0 to 7
- BTG F, b, a ;Toggle bit b of file register. b = 0 to 7

- RLCF F, d, a ;Rotate bits left in file register through ; carry and save in W or F register
- RLNCF F, d, a ;Rotate bits left in file register ; and save in W or F register
- RRCF F, d, a ;Rotate bits right in file register through
- ; carry and save in W or F register
- RRNCF F, d, a ;Rotate bits right in file register ; and save in W or F register

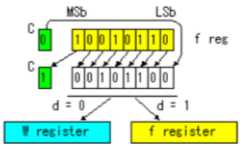
Instrucciones orientadas al bit

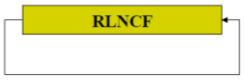
Rotations

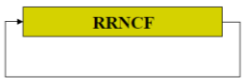
Rotate Right through Carry
RRCF



Rotate LEFT through Carry
RLCF







Instrucciones orientadas al bit

Test and Skip Instructions

- BTFSC F, b, a ;Test bit b in file register and skip the ;next instruction if bit is cleared (b =0)
- BTFSS F, b, a ;Test bit b in file register and skip the ;next instruction if bit is set (b =1)
- CPFSEQ F, a ;Compare F with W, skip if F = W
- CPFSGT F, a ;Compare F with W, skip if F > W
- CPFSLT F, a ;Compare F with W, skip if F < W
- TSTFSZ F, a ;Test F; skip if F = 0

Instrucciones orientadas a bucles

Increment/Decrement and Skip Next Instruction

- DECFSZ F, b, a ;Decrement file register and skip the ;next instruction if F = 0
- DECFSNZ F, b, a ;Decrement file register and skip the ;next instruction if F ≠ 0
- INCFSZ F, b, a ;Increment file register and skip the ;next instruction if F = 0
- INCFSNZ F, b, a ;Increment file register and skip the ;next instruction if F ≠ 0

Instrucciones básicas con literales

- **MOVLW K** ; Se carga el registro "W" con el
  ; contenido del literal "k“

Ejemplo:

```
MOVLW   .10          ; W= 0x0A
MOVLW   0x25         ; W= 0x25
MOVLW   0x86         ; W= 0x86
MOVLW   b’01010011’  ; W= 0x53
```

Instrucciones básicas con literales

- **ADDLW K** ; Suma al contenido del registro W
  ; el literal “K" (W ← W + K)
- **SUBLW K** ; Resta al contenido del registro W
  ; el literal “K" (W ← W - K)

Ejemplo:

```
MOVLW   0x25   ; W= 0x25
ADDLW   0x40   ; W= 0x40 + 0x25 = 0x65
MOVLW   .50    ; W= 0x32
SUBLW   0x18   ; W= 0x18 – 0x32 = 0xE6
```

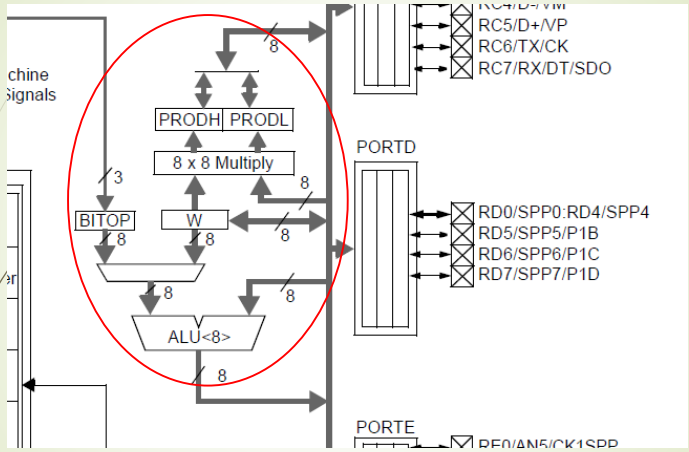
Instrucciones básicas con literales

- **MULLW K** ; PROD ← W * K ;
  ; PRODL = LSB_RESULTADO
  ; PRODH = MSB_RESULTADO

Ejemplo:

```
MOVLW   0x25   ; W= 0x25
MULLK   0x40   ; PROD ← 0x25 * 0x40
               ; PRODH:PRODL = 0x0940
               ; PRODH= 0x09, PRODL=0x40
MULLK   .2     ; PROD ← 0x25 * 0x02
               ; PRODH:PRODL = 0x004A
```

Registros: PRODH:PRODL en MULL



Instrucciones básicas con literales

- **LFSR f,K** ; FSRx ← K (número de 12 bits)
  ; FSRxL = LSB_K
  ; FSRxH = MSB_K

Ejemplo:

```
LFSR   0,0x356    ; FSR0H= 0x03, FSR0L= 0x56
LFSR   1,0x4A2    ; FSR1H= 0x04, FSR1L= 0xA2
LFSR   2,0xC10    ; FSR2H= 0x0C, FSR2L= 0x10
LFSR   FSR0,0x356 ; FSR0H= 0x03, FSR0L= 0x56
LFSR   FSR1,0x4A2 ; FSR1H= 0x04, FSR1L= 0xA2
```

Instrucciones básicas con literales

- **ANDLW K** ; W ← W (and) K
- **IORLW K** ; W ← W (or) K
- **XORLW K** ; W ← W (or) W

Ejemplo:

```
MOVLW  0x39      ; W= 0x39
ANDLW  0x25      ; W= 0x39 (and) 0x25 = 0x21
IORLW  0x88      ; W= 0x21 (or) 0x88 = 0xA9
XORLW  0x17      ; W= 0xA9 (xor) 0x17 = 0xBE
```

Instrucciones básicas entre W y REG.

- **MOVWF f,a** ; Copia el contenido del registro
  ; "W" al registro "f"
  ; Si a=0, **f** está en el banco de SRAM
  ; Si a=1, f está en el banco dado por BSR

Ejemplo 1:

```
X    EQU 0x30        ; se asigna posmem 0x30 a X
Y    EQU 0x31        ; se asigna posmem 0x31 a Y

     ORG     0x0000  ;vector de reset
     GOTO    MAIN    ;salta al programa principal
     org     0x0020  ;Zona de programa de usuario
MAIN:
;……………………………………………………………………
     MOVLW    0x25   ; W= 0x25
     MOVWF    X,0    ; [X]= [0x30]=0x25
     MOVLW    .45    ; W= 0x2D
     MOVWF    Y      ; [Y]=[0x31]= 0x2D (se puede ignorar "a"=0)
fin GOTO   fin
     END
```

Instrucciones básicas entre W y REG.

Ejemplo 2: **MOVWF f,a** ;(a=1)

```
X    EQU 0x30        ; se asigna posmem 0x30 a X
Y    EQU 0x31        ; se asigna posmem 0x31 a Y

     ORG     0x0000  ;vector de reset
     GOTO    MAIN    ;salta al programa principal
     org     0x0020  ;Zona de programa de usuario
MAIN:
;..................................................
     MOVLB    .2     ; BSR= 2
     MOVLW    0x25   ; W= 0x25
     MOVWF    X,1    ; [X]=[0x230]= 0x25
                     ; (DIR_X= 0x30 + 0x200 = 0x230)
     MOVLW    .50    ; W= 0x32
     MOVWF    Y,1    ; [Y]=[0x231]= 0x32
                     ; (DIR_Y= 0x31 + 0x200 = 0x231)
fin GOTO   fin
     END
```

Instrucciones básicas entre W y REG.

- **ADDWF f,d,a** ; Suma el contenido del registro
  ; "W" al registro "f"
  ; Si d=0, W ← Resultado de la suma,
  ; Si d=1, f ← Resultado de la suma
  ; Si d=1 ^ a=1, f está en el banco dado por BSR

Ejemplo 1: (a=0)

```
X    EQU     0x30
Y    EQU     0x31

     ORG     0x0000      ;vector de reset
     GOTO    MAIN        ;salta al programa principal
     org     0x0020      ;Zona de programa de usuario
MAIN:
     MOVLW   0x20        ; W= 0x20
     MOVWF   X,0         ; [X]=[0x30]= 0x20
     MOVLW   .50         ; W= 0x32
     ADDWF   X,0,0       ; W= 0x20 + 0x32 = 0x52
     ADDWF   X,1,0       ; [X]=[0x30]= 0x52 + 0x20 = 0x72
fin GOTO   fin
     END
```

Instrucciones básicas entre W y REG.

Ejemplo 2 (**ADDWF f,d,a**) ;(a=1)

```
X    EQU 0x30        ; se asigna posmem 0x30 a X
Y    EQU 0x31        ; se asigna posmem 0x31 a Y

     ORG     0x0000  ;vector de reset
     GOTO    MAIN    ;salta al programa principal
     org     0x0020  ;Zona de programa de usuario
MAIN:
;.................................................
     MOVLW   0x20    ; W = 0x20
     MOVWF   X,0     ; [X] = 0x20
     MOVLW   .50     ; W = 0x32
     MOVLB   .3      ; BSR = 3
     ADDWF   X,1,1   ; [X]=[0x330]= [0x330]+W = 0x00 + 0x32 = 0x32
     ADDWF   X,1,0   ; [X]=[0x30]= [0x30] + W = 0x20 + 0x32 = 0x52
     MOVLW   .101    ; W = 0x65
     XORWF   X,1,1   ; [X]=[0x330]= [0x330] xor W = 0x32 + 0x65 = 0x51

fin GOTO   fin
     END
```

Ejemplo1: Manejo de registros

```
1
2       list p=18f4550              ;Modelo del microcontrolador
3       #include<p18f4550.inc>     ;Llamo a la librería del PIC18F4550
4
5   ;Zona de los bits de configuración
6     CONFIG  PLLDIV = 5            ; PLL Prescaler Divide by 5 (20 MHz/5 = 4 MHZ)
7     CONFIG  CPUDIV = OSC1_PLL2    ; System Clock Postscaler (20 MHz/1 = 20 MHz)
8     CONFIG  USBDIV = 2            ; USB Clock Full-Speed (96 MHz/2 = 48 MHz)
9     CONFIG  FOSC = HS             ; Oscillator Selection bits (HS oscillator)
10    CONFIG  PWRT = ON             ; Power-up Timer Enable bit (PWRT enabled)
11    CONFIG  BOR = OFF             ; Brown-out Reset disabled
12    CONFIG  BORV = 3              ; Brown-out Reset Voltage (Minimum 2.05V)
13    CONFIG  WDT = OFF             ; Watchdog Timer disabled
14    CONFIG  CCP2MX = OFF          ; CCP2 MUX bit (CCP2 is multiplexed with RB3)
15    CONFIG  PBADEN = OFF          ; PORTB A/D (PORTB<4:0> configured as digital I/O)
16    CONFIG  MCLRE = ON            ; MCLR Pin Enable bit (MCLR pin enabled)
17    CONFIG  STVREN = ON           ; Stack Full/Underflow will cause Reset
18    CONFIG  LVP = OFF             ; Single-Supply ICSP disabled
19
```

- Nota: este encabezamiento de los programas se usará en todos los ejemplos que se mostrarán a continuación.
- Frecuencia del Xtal = 20 MHz, Frec_CPU = 20 MHz y Frec_USB = 48 MHz

Ejemplo1: Manejo de registros

```
X    EQU     0x30
Y    EQU     0x31

     ORG     0x0000       ;vector de reset
     GOTO    MAIN         ;salta al programa principal
     org     0x0020       ;Zona de programa de usuario
MAIN:
     MOVLW   .100         ;W = 0x64
     MOVWF   X            ;[X] = 0x64     (almacena en [X] <= W)
     MOVLW   .150         ;W = 0x96
     MOVWF   Y            ;[Y] = 0x96     (almacena en [Y] <= W)
     MOVLW   .42          ;W = 0x2A
     ANDWF   X,0          ;W = 0x20       (W <= W and [X])
     ADDWF   Y,1          ;[Y] = 0xB6     ([Y] <= W + [Y])
     SUBLW   .25          ;W = 0xF9       (W <= 0x19 - W)
     MOVF    X,0          ;W = 0x64       (carga en W <= [X])
     ADDLW   .66          ;W = 0xA6       (W <= 0x42 + W)
     XORWF   Y,0          ;W = 0x10       (W <= W xor [Y])
     IORWF   X,1          ;[X] = 0x74     ([X] <= W or [X])
     ADDLW   b'01010111'  ;W = 0x67       (W <= 0x57 + W)
     ANDLW   0x78         ;W = 0x60       (W <= 0x78 and W)
     ADDLW   .21          ;W = 0x75       (W <= 0x15 + W)
fin  GOTO    fin
     END
```

Instrucciones orientadas al bit

- **BCF f,b,a** ; Este comando pone en CERO el Bit
  ; "b" del registro "f".
  ; Si a=0, **f** está en el banco de SRAM
  ; Si a=1, f está en el banco dado por BSR
- **BSF f,b,a** ; Este comando pone en UNO el Bit
  ; "b" del registro "f".

Ejemplo:

```
BCF     PORTA,3,0    ; pone en "0" el bit 3 del PORTA
BSF     PORTA,5,0    ; pone en "1" el bit 5 del PORTA
MOVLB   .3           ; BSR= 3
BCF     0x45,5,1     ; pone en "0" el bit 5 del contenido de
                     ; la dirección 0x345.
```

Instrucciones orientadas al bit

- **BTFSC f,b,a** ; Este comando prueba el Bit "b" del registro "f" y
  ; salta una línea si f,b está en CERO.
- **GOTO k** ; salta a una dirección con el nombre "k" (etiqueta).

Ejemplo:

```
BTFSC   PORTA,0,0    ; testea el bit 0 del PORTA, y:
GOTO    vale_1       ; si es "1", ejecuta esta instrucción
GOTO    vale_0       ; si es "0", ejecuta esta instrucción
```

Instrucciones orientadas al bit

- **BTFSS f,b,a** ; Este comando prueba el Bit “b” del registro "f” y ; salta una línea si f,b está en CERO.
- **GOTO k** ; salta a una dirección con el nombre "k" (etiqueta).

Ejemplo:

```
BTFSC   PORTA,0,0   ; testea el bit 0 del PORTA, y:
GOTO    vale_1      ; si es “1”, ejecuta esta instrucción
GOTO    vale_0      ; si es “0”, ejecuta esta instrucción
```

Instrucciones orientadas al bit

- **BTG f,b,a** ; Este comando invierte el Bit “b” del registro "f”.

Ejemplo:

```
MOVLW   0b10101111  ; W= 0xAF
MOVWF   0x60,0      ; [0x60]= 0b10101111= 0xAF
BTG     0x60,3,0    ; [0x60]= 0b10100111= 0xA7
MOVLW   0b10001001  ; W= 0x89
MOVWF   LATD,0      ; LATD= 0b10001001= 0xAF
BTG     LATD,7,0    ; LATD= 0b00001001= 0x09
```

Instrucciones orientadas al byte

- **CLRW** ; Esta instrucción borra el registro w (w = 0).
- **CLRF f,a** ; Esta instrucción borra un registro específico.

Ejemplo:

```
MOVLW   0b10101111   ; W= 0xAF
CLRW                 ; W= 0x00
MOVLW   0xFF         ; W= 0xFF
MOVWF   LATB,0       ; LATB= 0xFF
CLRF    LATB,0       ; LATB= 0x00
```

Instrucciones orientadas al byte

- CALL k,s ; Llama a una subrutina llamada K
  ; guardando en la pila la dirección
  ; PC+2, antes del salto.
  - Si s=0, los valores de BSR, W y STATUS no se guardan en la pila. Si s=1, los valores se guardan en la pila.
- RETURN ; Retorno de una subrutina

Ejemplo:

```
    CALL   DELAY        ; Llama a la subrutina delay
    ......
;.....................................
DELAY                   ; programa de la subrutina
    .....
    RETURN              ; fin de la subrutina
```

Instrucciones orientadas al byte

- **DECFSZ f,d,a** ; Esta instrucción disminuye en una sola ; unidad el registro "f" en la cual, si el registro "f" queda igual a cero ; entonces salta una línea.
- **DCFSNZ f,d,a** ; Esta instrucción disminuye en una sola ; unidad el registro "f" en la cual, si el registro "f" es diferente de ; cero, entonces salta una línea.

Instrucciones orientadas al byte

- **INCFSZ f,d,a** ; Esta instrucción incrementa en una sola ; unidad el registro "f" en la cual, si el registro "f" queda igual a cero ; entonces salta una línea.
- **INFSNZ f,d,a** ; Esta instrucción incrementa en una sola ; unidad el registro "f" en la cual, si el registro "f" es diferente de ; cero, entonces salta una línea.

Bibliografía

- Los gráficos fueron obtenidos de: "Comenzando con los PIC18: Arquitectura, Set de Instrucciones, Interrupciones, Periféricos y Características Especiales"

- PIC18f4550 datasheet.
- Han-Way Huang, PIC Microcontroller: An Introduction to Software and Hardware Interfacing, Thomson Delmar Learning, 2005.