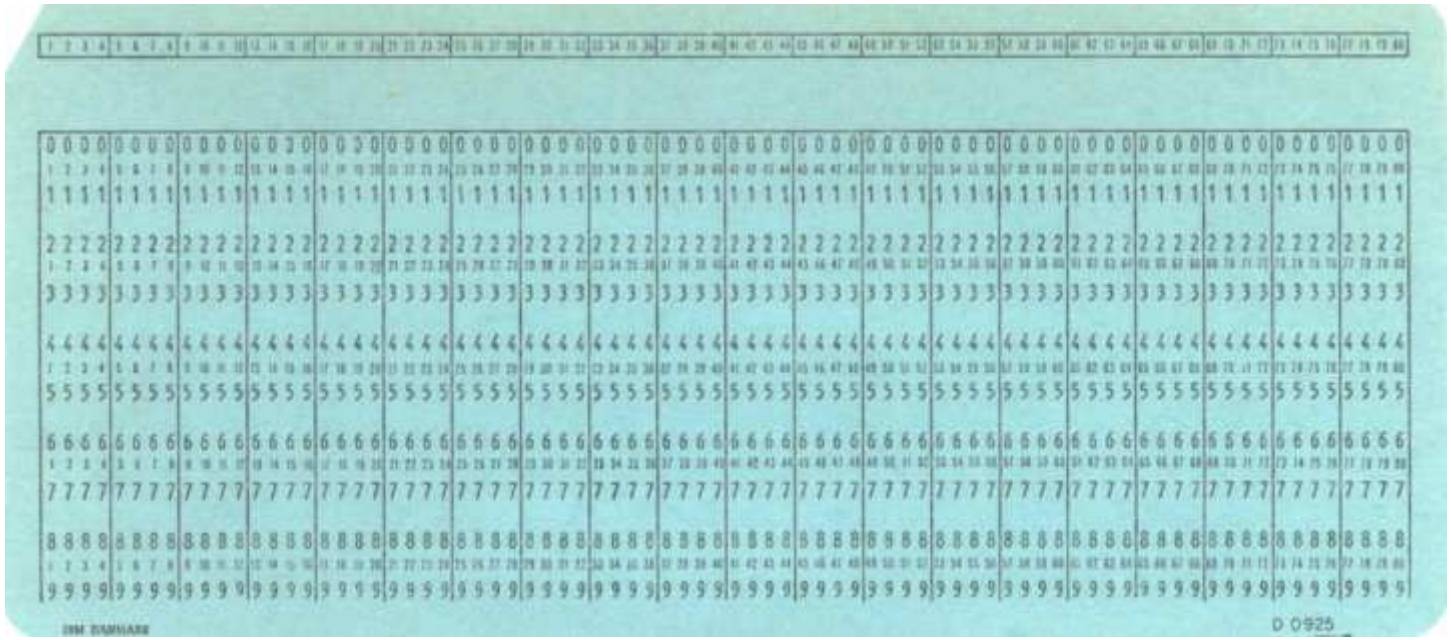
Microprocesadores, Tema 3:

Programación del Microcontrolador PIC18 a Bajo Nivel



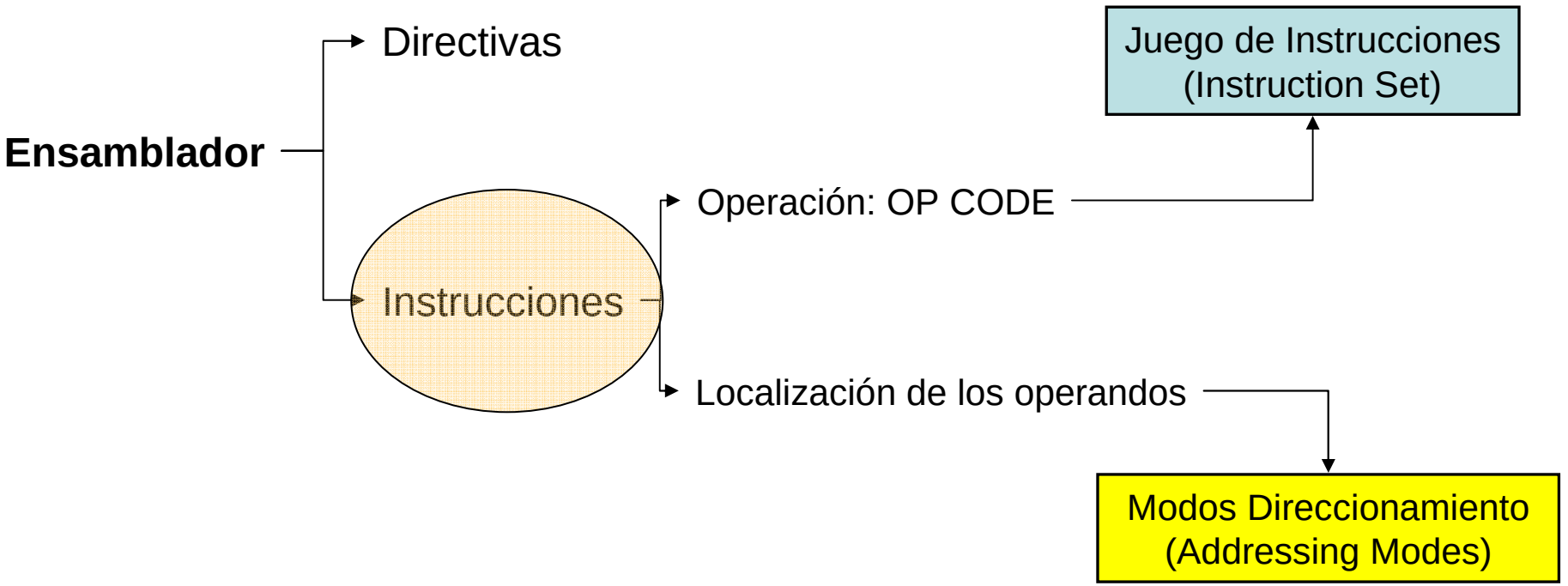
Guillermo Carpintero, guiller@ing.uc3m.es
Universidad **Carlos III** de Madrid

Lenguajes de Programación

Nos permiten expresar la secuencia de operaciones que deseamos realice la máquina

Código máquina	10010100111101
Ensamblador Códigos nemónicos Correspondencia univoca con código máquina	add A,B
Alto nivel Próximo a lenguaje natural	A = A + B A += 1

Programación Ensamblador



Programación Ensamblador

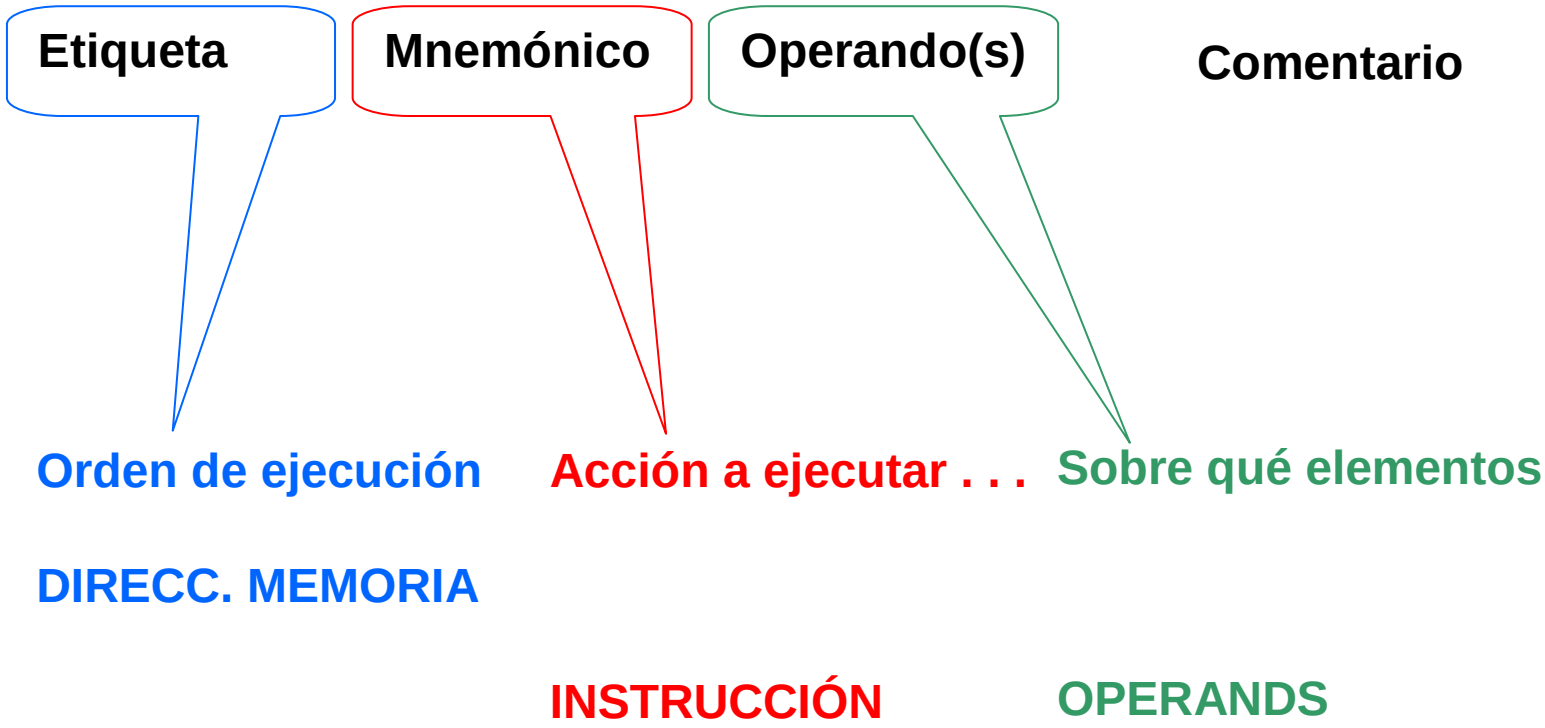
El ensamblador es un conjunto de órdenes simples . . . .

. . . . . que pueden ser ejecutadas en la ruta de datos.

Programación Ensamblador

Sentencias en ensamblador

Cada línea de código puede englobar cuatro campos:



Programación Ensamblador: Instrucciones

Clasificación de las Instrucciones de la máquina

Instrucciones de Transferencia de Datos

Movimiento (Move)
Alteración Datos (Clear, Inc, Dec)
Rotación Bits (Shift, Rotate)

Instrucciones Aritméticas (Add, Sub, Mult, Div)

Instrucciones Lógicas (And, Or, Xor)

Instrucciones Booleanas (Set bit, Clear bit, Jump if bit set, Jump if bit clear)

Instrucciones de Salto

Control (Jump, **Conditional jumps**)
Ralacionadas con Subrutinas (Push, Pull)
Relacionadas con Interrupción (Retorno de Int.)

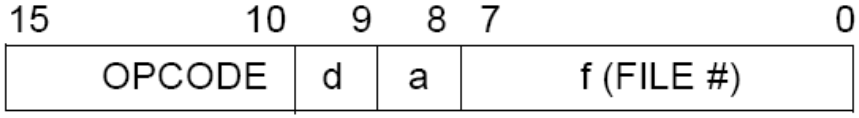
Juego de Instrucciones del PIC18

Tablas resumen del Fabricante

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb LSb	Status Affected	Notes
BYTE-ORIENTED OPERATIONS						
ADDWF	f, d, a	Add WREG and f	1	0010 01da0 ffff ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and CARRY bit to f	1	0010 0da ffff ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001 01da ffff ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110 101a ffff ffff	Z	2
COMF	f, d, a	Complement f	1	0001 11da ffff ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110 001a ffff ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110 010a ffff ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110 000a ffff ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000 01da ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010 11da ffff ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100 11da ffff ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010 10da ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011 11da ffff ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100 10da ffff ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001 00da ffff ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101 00da ffff ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 ffff ffff ffff 1111 ffff ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110 111a ffff ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000 001a ffff ffff	None	1, 2
NEGF	f, a	Negate f	1	0110 110a ffff ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011 01da ffff ffff	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100 01da ffff ffff	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	1	0011 00da ffff ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100 00da ffff ffff	Z, N	
SETF	f, a	Set f	1	0110 100a ffff ffff	None	1, 2
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101 01da ffff ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	0101 11da ffff ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101 10da ffff ffff	C, DC, Z, OV, N	
SWAPF	f, d, a	Swap nibbles in f	1	0011 10da ffff ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110 011a ffff ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001 10da ffff ffff	Z, N	

Formato de las Instrucciones **del PIC18**

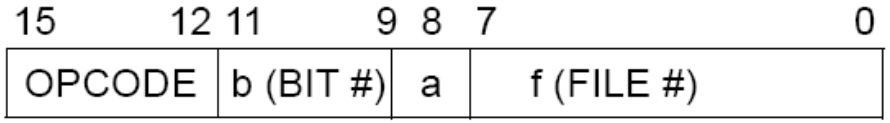
Byte-oriented file register operations



d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

MOVWF	**Move W to f**
Syntax:	MOVWF f {,a}
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(W) → f
Status Affected:	None
Encoding:	0110 111a ffff ffff

Bit-oriented file register operations

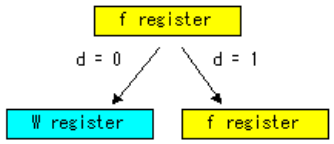


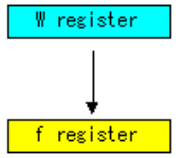
b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

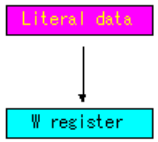
BCF	**Bit Clear f**
Syntax:	BCF f, b {,a}
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]
Operation:	0 → f<b>
Status Affected:	None
Encoding:	1001 bbba ffff ffff

Instrucciones Ensamblador

http://hobby_elec.piclist.com/

MOVF	Move f
Form	**[label]▵MOVF▵f, d** (label is omitable, ▵ shows SPACE code)
Operands	f : Register file addesss (00(00h) to 127(7Fh)) d : Destination select (0 or 1)
Operation	It moves (copy) the contents of the f register. d = 0 : store result in W d = 1 : store result in f Seemed to move from the f register to the f register no meaning, but its being used in the purpose to set a flag. 
Flag	When the result is 0, it sets 1 to the Z flag. When the result is not 0, it sets 0 to the Z flag.
Instruction cycles	1 cycle

MOVWF	Move W to f
Form	**[label]▵MOVWF▵f** (label is omitable, ▵ shows SPACE code)
Operands	f : Register file addesss (00(00h) to 127(7Fh))
Operation	It moves (copy) the contents of the W register to the f register. 
Flag	No change
Instruction cycles	1 cycle

MOVLW	Move literal to W
Form	**[label]▵MOVLW▵k** (label is omitable, ▵ shows SPACE code)
Operands	k : literal field (00(00h) to 255(FFh))
Operation	It sets literal data to the W register. 
Operation	No change
Instruction cycles	1 cycle

NOP	No Operation
Form	**[label]▵NOP** (label is omitable, ▵ shows SPACE code)
Operands	None
Operation	It is moved to the next instruction without processing anything. This instruction is used when it adjusts a processing time as the timer processing, and so on. 
Flag	No change
Instruction cycles	1 cycle

Instrucciones Ensamblador

Operandos y el uso de etiquetas

Literales `MOVLW 0xF0` hexadecimal

`0d10` decimal

`0b10100101` binario

a, d, f `MOVWF PORTA, A`

`BANKED`

`BCF PORTA, BIT5, A`

¿Cuál es el valor de

- **PORTA**, si indica la dirección del puerto A?
- **BIT5**, si indica que queremos borrar el bit 5?

Instrucciones Ensamblador

Escribir un programa (LIMPIAR_RAM) que borre N posiciones de la memoria RAM a partir de la dirección M.

N está almacenada en el registro W
M está almacenada en el registro FSR0

Necesitaremos:

Código: Escribir el código (**DEBERES**)

Inicio: Decirle al micro dónde está el código a ejecutar

Parada: Decirle al micro qué hacer cuando finalicemos

Instrucciones Ensamblador

CÓDIGO

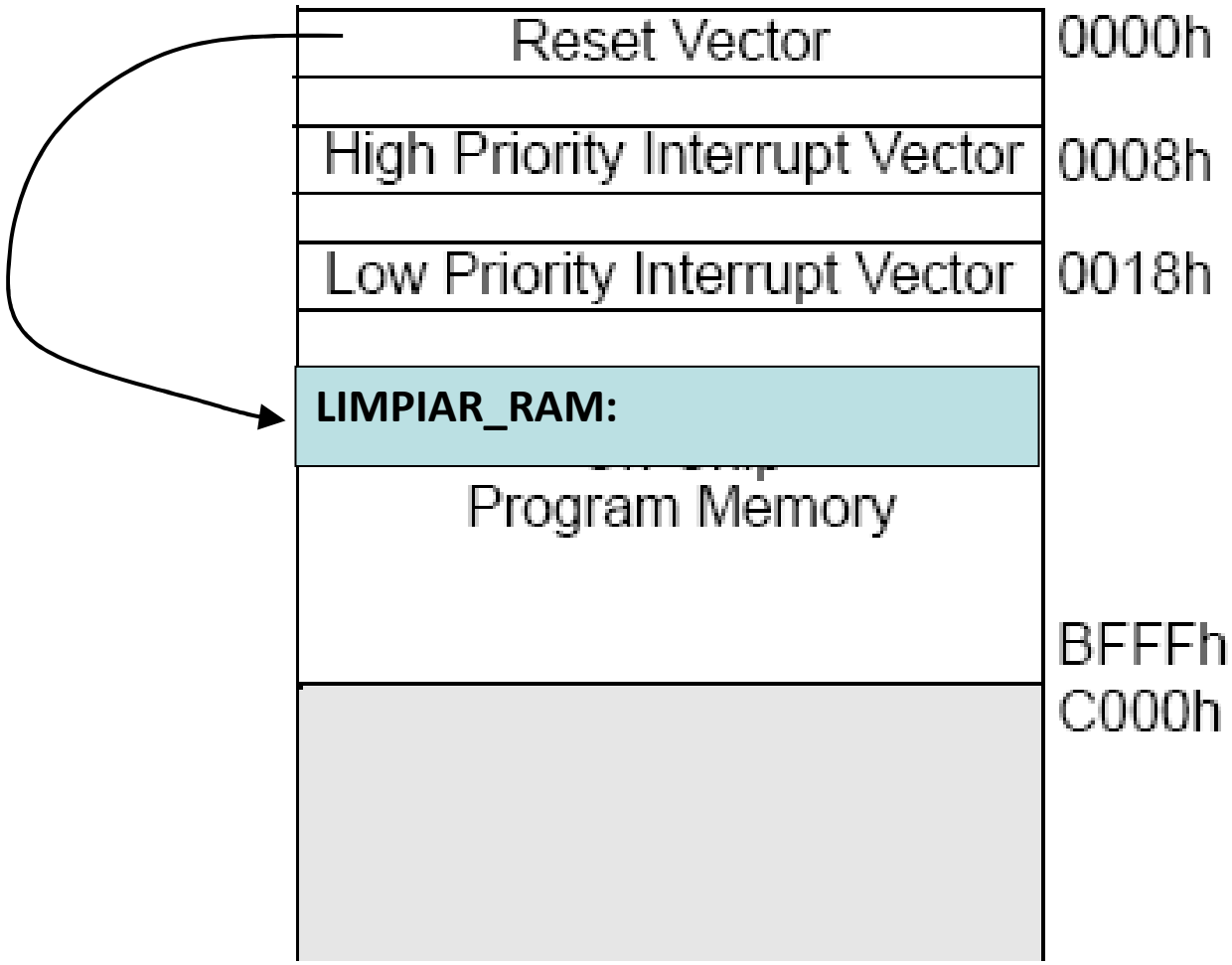
LIMPIAR_RAM:

Instrucciones

(Para la próxima semana)

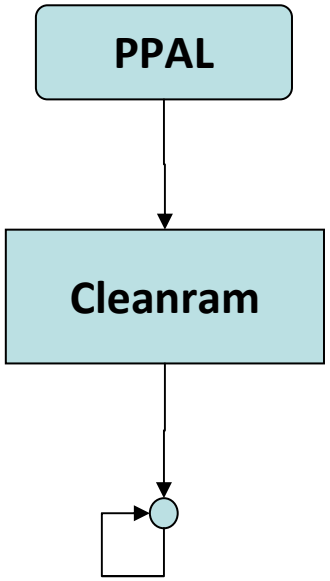
Instrucciones Ensamblador

INICIO



Instrucciones Ensamblador

PARADA



El programa DEBE estar controlado. No puede deambular por la memoria.
La ejecución se DEBE limitar a las líneas escritas por el programador

Figura del “PIC18F2525/2620/4525/4620 Data Sheet”
Con permiso de MICROCHIP

Assembler Instructions

Instrucciones de salto. INCONDICIONAL

BRA	**Unconditional Branch**
Syntax:	BRA n
Operands:	-1024 ≤ n ≤ 1023
Operation:	(PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1101 | 0nnn | nnnn | nnnn
Description:	Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction.
Words:	1
Cycles:	2

GOTO	**Unconditional Branch**
Syntax:	GOTO k
Operands:	0 ≤ k ≤ 1048575
Operation:	k → PC<20:1>
Status Affected:	None
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)	1110 | 1111 | $k_7$kkk | kkkk$_0$ 1111 | k_{19}kkk | kkkk | kkkk$_8$
Description:	GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-cycle instruction.
Words:	2
Cycles:	2

Instrucciones Ensamblador

INICIO

Establecemos la dirección de las instrucciones mediante la directiva del ensamblador ORG

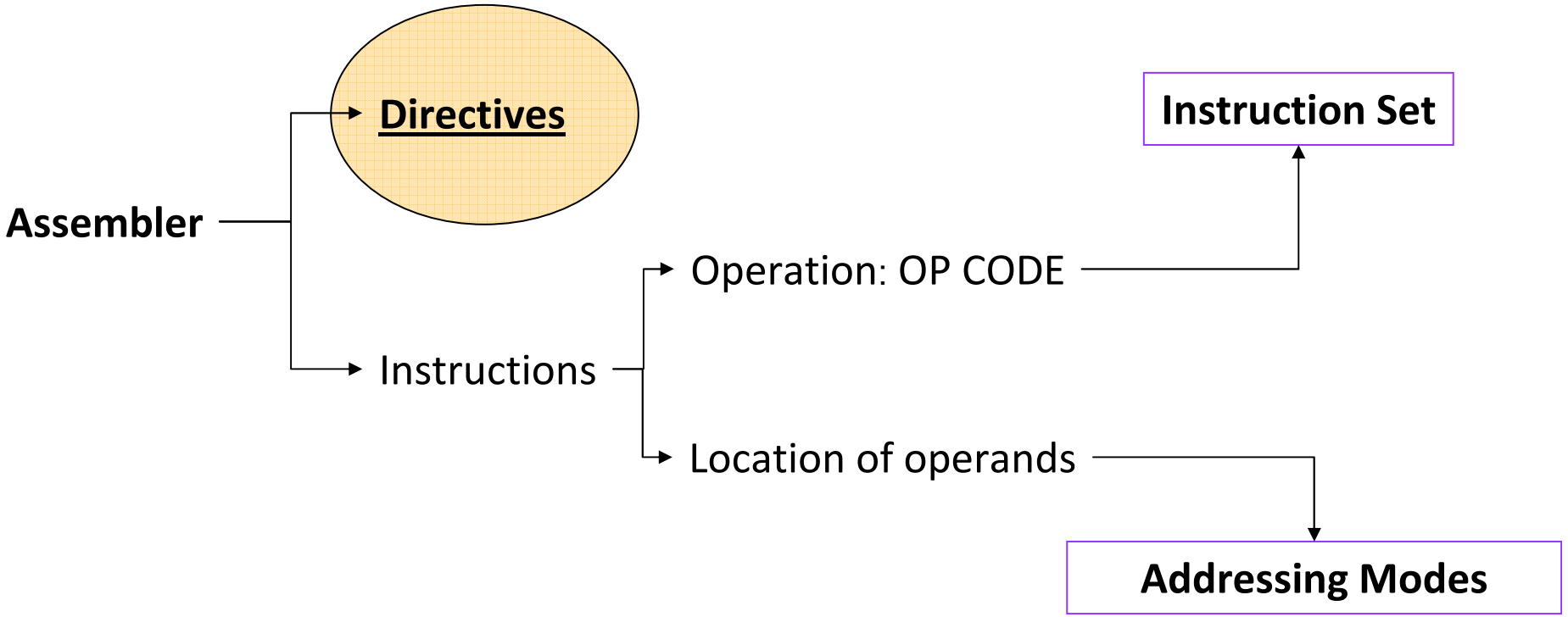
```
ORG 0x0000h
BRA LIMPIAR_RAM
```

PARADA

Bucle infinito

```
End     BRA     End
```

Fundamentos del ensamblador



Directivas de ensamblador. Código

#define <name> [<string>] **#define PORTA 80**

Esta directiva sustituye una cadena por un texto. Siempre que se encuentre **<name>** en el código, **<string>** lo sustituirá.

#include <include file> **#include <p18f2525.inc>**

Esta directiva incluye un fichero fuente. El fichero especificado es leído como código fuente. En la práctica es como se incluyéramos todo el texto del fichero referido en nuestro código.

[<label>] org <expr> **Reset ORG 0000h**

Esta directiva nos indica la dirección de memoria de la instrucción que le sigue.

Directivas de ensamblador. Datos

<label> EQU <expr>

Define una constante. Siempre que aparezca el nombre de la etiqueta, será sustituido por <expr>.

[<label>] data <expr>,[,<expr>, . . ., <expr>]

[<label>] data "<text_string>"[, "<text_string>"] **data "hola mundo"**

Inicializa una o más posiciones de memoria con datos. Los datos pueden estar en forma de constantes, etiquetas o expresiones. Cada <expr> es almacenada en una palabra.

[<label>] db <expr>[, <expr>, . . ., <expr>] **DB '1',1,0x03**

Reserva palabras de memoria de programas encapsulándolas en valores de 8 bits.

[<label> dw <expr> [, <expr>, . . ., <expr>] **DW '1',1,0x03**

Reserva palabras de memoria de programas encapsulándolas en valores de 16 bits.

Figura del "PIC18F2525/2620/4525/4620 Data Sheet"
Con permiso de MICROCHIP

Instrucciones Ensamblador

Instrucciones más importantes:

Instrucciones de salto. CONDICIONALES

FLAGS del registro de estado

BC	**Branch if Carry**
Syntax:	BC n
Operands:	-128 ≤ n ≤ 127
Operation:	if CARRY bit is '1' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 0010 nnnn nnnn

BN	**Branch if Negative**
Syntax:	BN n
Operands:	-128 ≤ n ≤ 127
Operation:	if NEGATIVE bit is '1' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 0110 nnnn nnnn