Universidad Francisco de Vitoria
UFV Madrid

Arquitectura y Organización de Computadores

Sesiones de Laboratorio curso 2019-2020

Olga Peñalba Rodríguez y Daniel León

Sesión de Laboratorio 2

En esta sesión vamos a comenzar a aprender cómo funciona la programación de bajo nivel. Ya conoces de la anterior sesión el entorno de trabajo, los materiales que vamos a utilizar y el alcance esperado del proyecto que debes realizar a lo largo de la asignatura. Ahora vamos a empezar a dar pasos para que puedas llegar a esa meta.

En esta sesión vas a construir tu primer programa en ensamblador, empezando desde cero.

Objetivos

Los objetivos generales que se persiguen son:

- Aprender, de forma general, qué es un microcontrolador y conocer el microcontrolador PIC16F886, que se utilizará para el desarrollo de todas las prácticas de programación en ensamblador.
- Entender la estructura de la memoria del microcontrolador para poder
- Familiarizarse con el repertorio de instrucciones del PIC16F886 y saber interpretar la información que proporciona sobre el mismo el Data Sheet.
- Aprender a manejar el entorno MPLAB X IDE para la edición, ensamblado y depuración de programas en ensamblador.

Resultados de aprendizaje

Tras la realización de esta sesión de laboratorio, deberás ser capaz de:

- Utilizar correctamente las instrucciones MOVF y MOVWF, para transferencia de datos, y ADDWF, ADDLW y SUBWF para cálculos aritméticos.
- Definir los diferentes elementos que forman parte de un programa en ensamblador del PIC (etiquetas, directivas, instrucciones, etc.).
- Escribir programas sencillos que calculen expresiones aritméticas.
- Verificar el funcionamiento de un programa utilizando el entorno MPLAB X IDE y depurarlo mediante ejecución paso a paso.
- Acceder a cualquier SFR y GPR del PIC y escribir y leer de los pines del microcontrolador.

Teoría

Introducción a los microcontroladores

A mediados de 2002 se superó la impresionante cifra de 1000 millones de procesadores utilizados en la fabricación de computadores. Pero esta cantidad sólo abarca entorno al 2% de los sistemas de computación; el 98% restantes son procesadores empotrados (entre los que se encuentran los microcontroladores).

Un microcontrolador (conocido en inglés por las siglas MCU) es un sistema electrónico que tiene como finalidad controlar el funcionamiento de ciertos dispositivos externos, como sensores, teclados, pantallas, interruptores, etc. Está compuesto por un procesador o núcleo de ejecución, memoria para almacenamiento de datos e instrucciones y otros módulos que permiten la comunicación con los dispositivos que ha de controlar, todo ello integrado dentro de un mismo circuito integrado. La capacidad de procesamiento de estos sistemas es muy inferior a la de un procesador de propósito general; sin embargo son muy baratos y tienen un tamaño muy reducido, lo que los hace ideales para ser insertados en multitud de máquinas y equipos, incrementando sus prestaciones.

Los microcontroladores se programan, al igual que los procesadores de propósito general, pero "una única vez". Es decir, a diferencia de los computadores, los microcontroladores ejecutan una única aplicación, que es la que realiza las acciones necesarias para el control de la máquina donde ha sido insertado.

Existen más de medio centenar de prestigiosos fabricantes de microcontroladores, y cada uno oferta una gran variedad de modelos potenciando en algunos las capacidades de memoria, en otros la velocidad de trabajo, en otros la potencia de los recursos auxiliares, etc.

Entre los campos de aplicación se encuentran: electrodomésticos, electrónica de consumo, automoción, maquinaria industrial, etc.

La empresa americana Arizona Microchip (www.microchip.com) fabrica los microcontroladores PIC, que han tenido un extraordinario éxito en todo el mundo. Actualmente es una de las empresas con mayor volumen de venta de microcontroladores, más aún tras la compra de su rival Atmel en 2016.

Cada fabricante aporta los siguientes documentos para sus microcontroladores:

- ***Data Sheet***: es el documento más importante, donde se describe al detalle la estructura del microcontrolador y las características de todos los elementos que lo integran.
- ***Family reference Manual***: Contiene información muy detallada sobre los aspectos comunes de una familia de microcontroladores.

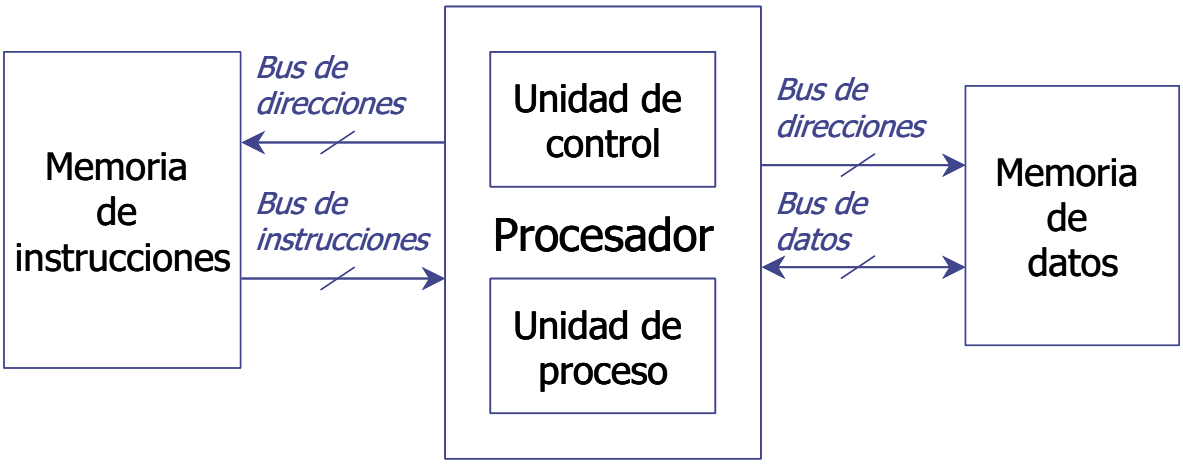
- ***Application notes***: documentos que describen aplicaciones llevadas a cabo utilizando dicho microcontrolador. Pueden incorporan los ficheros de código ensamblador utilizados para crear la aplicación.
- ***User Guides***: documentos guías para el desarrollo de aplicaciones con los microcontroladores; suelen explicar el funcionamiento de los entornos y herramientas de desarrollo (programación).

El microcontrolador PIC16F886

El microcontrolador PIC16F886 es un circuito integrado de 28 pines, que funciona a una frecuencia de reloj de hasta 20 MHz. Incorpora un procesador capaz de ejecutar 35 instrucciones diferentes, tres tipos de memoria (Flash, RAM, EEPROM) para datos e instrucciones y numerosos módulos y registros para control de los periféricos que se le conecten (temporizadores, contadores, conversores A/D y D/A, puertos de comunicación serie y paralelo etc.).

Arquitectura Harvard

Los PIC utilizan la arquitectura Harvard, según muestra la siguiente figura:



A diferencia de la arquitectura Von Neumann, la Harvard independiza la memoria de datos de la de instrucciones, y utiliza un bus dedicado para cada una de ellas, optimizando las capacidades y la longitud de sus palabras y permitiendo accesos simultáneos a ambas. En particular, el PIC16F886 contiene una memoria de instrucciones de tipo Flash, de 8K palabras, de 14 bits cada una; una memoria RAM de datos, formada por 368 bytes (divididos en cuatro páginas o bancos) y otra EEPROM de 256 bytes (todos los datos son de 1 byte).

Otra característica de este procesador es su arquitectura de 1 dirección, es decir, arquitectura de acumulador (las instrucciones aritméticas hacen referencia explícita a

un solo operando; el otro se encuentra en el acumulador), con la particularidad de que puede elegirse en cuál de los dos operandos escribir el resultado.

Incorpora los modos de direccionamiento: inmediato, directo e indirecto de registro.

Cuenta con varios registros específicos, entre los que se encuentran:

- *Contador de programa*: almacena la dirección de la siguiente instrucción a ejecutarse.
- *Registro de estado* (STATUS): de 8 bits, los últimos tres almacenan los bits Z (indica si el resultado es 0), DC (indican si ha habido un acarreo al operar los 4 bits menos significativos de los datos), y C (indica un acarreo al operar con los 8 bits del dato)

Periféricos

Un microcontrolador incluye una serie de periféricos encargados de la comunicación con el exterior que conectados al bus de datos (Entrada/Salida mapeada en memoria). Se configuran mediante una serie de registros especiales dentro de los "SFR – Special File Register", de la misma manera que se puede acceder al STATUS en la posición de memoria 0x03.

La imagen de la página siguiente muestra un diagrama de bloques general del PIC16F886 donde se detallan los periféricos (extraído de la pág. 14 del Data Sheet).

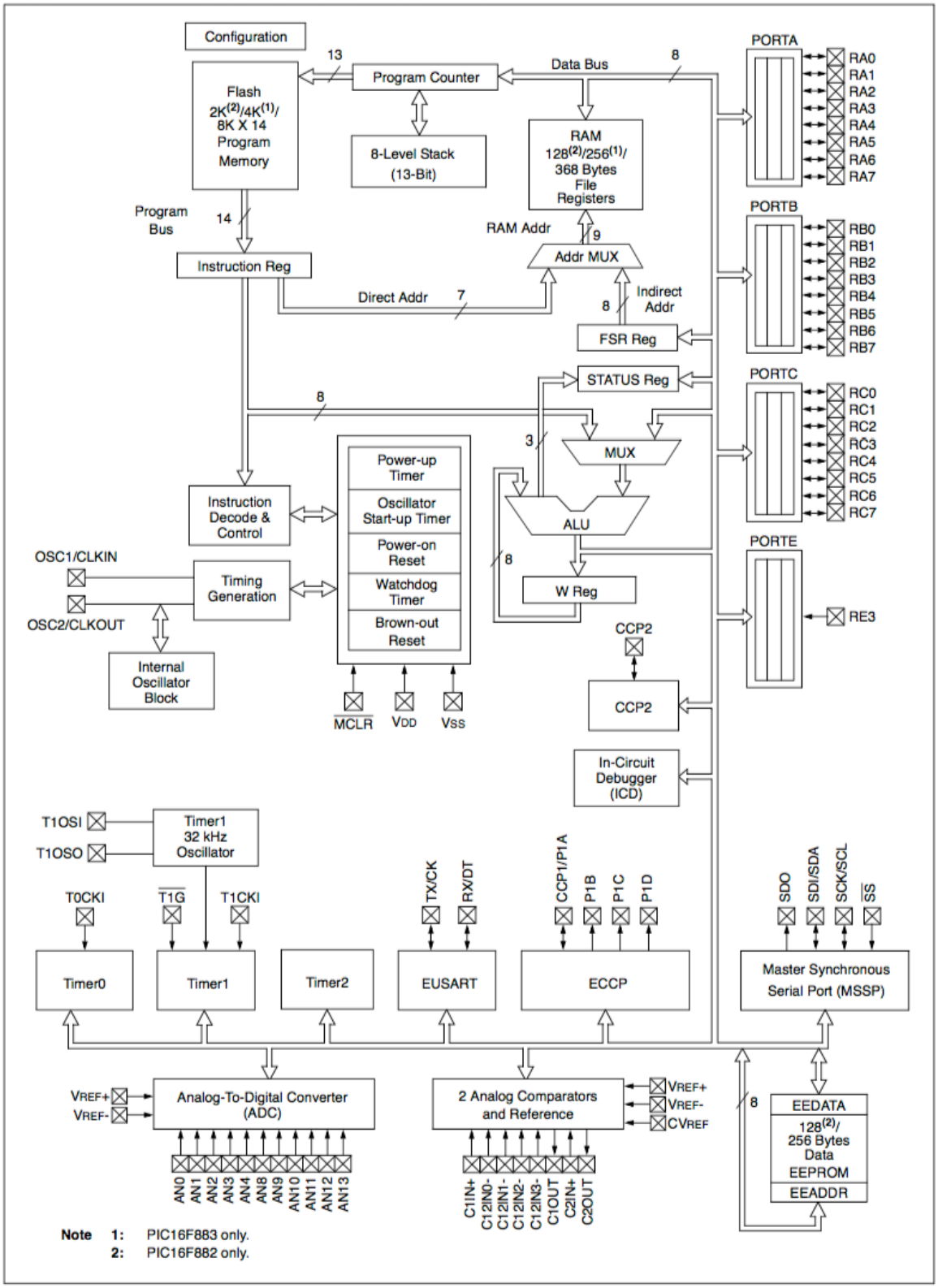
De estos periféricos, los más importantes, y en los que nos vamos a centrar en este curso, son los puertos GPIO (General-Purpose Input-Output), que están divididos en 4 bloques: PORTA (8 puertos), PORTB (8 puertos), PORTC (8 puertos) y PORTE (1 puerto). Estos puertos se corresponden con pines específicos del microcontrolador y podremos escribir o leer datos digitales de ellos (0 o 1).

Cada pin puede configurarse para entrada o salida mediante los **SFR TRISA, TRISB y TRISC** (banco 1 de memoria, como veremos más adelante). El pin E3 es siempre de entrada.

Si un pin está configurado como salida, podremos fijar un valor digital de 0 o 1 escribiendo en los **SFR PORTA, PORTB y PORTC** (banco 0 de memoria) como si se tratara de un registro normal de datos.

De la misma forma, si un pin está configurado como entrada, podremos leer el valor digital que tiene a la entrada leyendo de esos mismos registros y de **PORTE**. Hay que tener en cuenta que la lectura acepta niveles lógicos CMOS de 5v (0v-1.5v para 0 y 3.5v-5v para 1) y, aunque aplica histéresis para las transiciones, la lectura de un pin en circuito abierto será errática.

FIGURE 1-1: PIC16F882/883/886 BLOCK DIAGRAM



Note 1: PIC16F883 only.
2: PIC16F882 only.

Estructura de la memoria del PIC16F886

Como hemos visto, la memoria de 368 bytes está dividida en 3 bancos y se corresponde al siguiente esquema (Data Sheet, página 25)

FIGURE 2-6: PIC16F886/PIC16F887 SPECIAL FUNCTION REGISTERS

	File Address		File Address		File Address		File Address
Indirect addr. (1)	00h	Indirect addr. (1)	80h	Indirect addr. (1)	100h	Indirect addr. (1)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	WDTCON	105h	SRCON	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	CM1CON0	107h	BAUDCTL	187h
PORTD(2)	08h	TRISD(2)	88h	CM2CON0	108h	ANSEL	188h
PORTE	09h	TRISE	89h	CM2CON1	109h	ANSELH	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2(1)	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved	18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh	Reserved	18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	WPUB	95h		115h		195h
CCPR1H	16h	IOCB	96h	General Purpose Registers	116h	General Purpose Registers	196h
CCP1CON	17h	VRCON	97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h	16 Bytes	119h	16 Bytes	199h
RCREG	1Ah	SPBRGH	9Ah		11Ah		19Ah
CCPR2L	1Bh	PWM1CON	9Bh		11Bh		19Bh
CCPR2H	1Ch	ECCPAS	9Ch		11Ch		19Ch
CCP2CON	1Dh	PSTRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Registers 96 Bytes	20h – 3Fh / 40h – 6Fh	General Purpose Registers 80 Bytes	A0h – EFh	General Purpose Registers 80 Bytes	120h – 16Fh	General Purpose Registers 80 Bytes	1A0h – 1EFh
	70h – 7Fh	accesses 70h-7Fh	F0h – FFh	accesses 70h-7Fh	170h – 17Fh	accesses 70h-7Fh	1F0h – 1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

2: PIC16F887 only.

Para acceder a las posiciones de memoria, o a los SFR ubicados en cada banco en particular, tenemos que manipular los bits **RP1** y **RP0** (bits 6 y 5 respectivamente) del registro **STATUS**, de forma que el número representado en binario de este par de bits se corresponde con el banco seleccionado. Nótese que esto es posible porque el registro **STATUS** es accesible desde todos los bancos de memoria. La macro BANKSEL del compilador también puede utilizarse como veremos en clase.

RP1	RP0	BANCO
0	0	0
0	1	1
1	0	2
1	1	3

Este tipo de acceso se denomina "acceso directo". En sesiones posteriores veremos otra forma de acceder a la memoria, el "acceso indirecto", que se utiliza, entre otros propósitos, para recorrido de estructuras de datos en memoria.

Lenguaje ensamblador

Entre las instrucciones (35 en total) del lenguaje ensamblador del PIC16F886 podemos encontrar:

- aritmético-lógicas: ADDLW, ADDWF, SUBWF, XORWF
- transferencias de datos: MOVF, MOVWF
- manipulación de bits: BCF, BSF
- control de flujo: GOTO, BTFSC, RETURN
- Instrucciones de control de HW: SLEEP, CLRWDT

Todas se codifican con 14 bits, pudiendo tener cuatro formatos distintos, mientras que los datos son de 8 bits.

Elementos de un programa en lenguaje ensamblador

Un programa escrito en lenguaje ensamblador puede contener los siguientes elementos:

- **Instrucciones**: constituyen el núcleo del programa y son traducidas a lenguaje máquina (binario) durante el proceso de ensamblado. Están formadas por un código mnemotécnico que identifica la operación, seguido por la especificación de los operandos.
- **Directivas**: son indicaciones que se introducen en el programa para controlar el proceso de ensamblado. No se corresponden con ninguna instrucción del lenguaje máquina y, por lo tanto, no generan código en binario, pero afectan a cómo se genera ese código.

- **Etiquetas**: son un mecanismo que ofrece el lenguaje ensamblador para facilitar el proceso de creación de programas. Tienen un valor numérico, que en mucho casos representa una dirección de memoria.
- **Macros**: son conjuntos de instrucciones que están agrupadas bajo un nombre común (etiqueta). Para ejecutar esas instrucciones basta con escribir el nombre de la macro en el punto del programa donde se desee incluir las instrucciones. De esta manera se simplifica la escritura del programa. Pueden incluir datos de tipo parámetro.
- **Comentarios**: son textos introducidos con el fin de facilitar la lectura y compresión del programa.
- **Datos u operandos**: suelen ser nombres de registros, literales o direcciones de memoria.

Elementos de un programa en ensamblador del PIC16F886

Un programa en lenguaje ensamblador del PIC16F886 puede incluir cualquiera de los elementos anteriormente reseñados. Cada línea del programa puede considerarse formada por cuatro campos o bloques:

etiqueta – código – argumentos – comentarios

El único campo obligatorio es el **código**, que identifica uno de los tres elementos siguientes: una *instrucción* del lenguaje del microcontrolador, una *directiva* del ensamblador o una *macro*. Se encuentra siempre separado de la primera columna por al menos un espacio en blanco o tabulador.

Los **argumentos** aparecen o no dependiendo del *código* (algunas instrucciones, directivas y macros no los requieren). Son siempre un número, o un cálculo sobre números fijos, representado en cualquier sistema de numeración de los admitidos. Cada número puede ser sustituido por una etiqueta previamente definida con el valor de dicho número.

Las **etiquetas** están formadas por caracteres alfanuméricos hasta un máximo de 32. Tienen siempre un valor numérico. Su uso es opcional, aunque las utilizaremos habitualmente con el fin de simplificar la escritura y lectura del código. Además, mediante las etiquetas se consigue independizar el código de la zona de memoria particular donde se almacene.

Los **comentarios** pueden aparecer en cualquier lugar, siempre que vayan precedidos por el símbolo ";".

Aunque este es el esquema general de las líneas de código, también se permiten líneas en blanco dentro del programa, o líneas formadas únicamente por comentarios.

Instrucciones del PIC16FF886

Existen en total 35 instrucciones en el lenguaje máquina del PIC16F886, que pueden ser combinadas para la creación de programas. La relación de todas las instrucciones puede consultarse en el DataSheet del microcontrolador (en la página 232) o en el apartado 29 de la referencia de la familia *Midrange* de PIC.

Table 29-1: Midrange Instruction Set

Mnemonic, Operands		Description	Cycles	14-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.
3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

Directivas del MPASM

Están asociadas al programa ensamblador MPASM. De las numerosas directivas disponibles, utilizaremos de forma habitual las siguientes:

- **LIST**: Con ella habilitamos la creación de un listado de símbolos para el procesador que estamos utilizando. Lo usaremos de la forma
  LIST p=16F886
- **__CONFIG**: Indica los parámetros de configuración general del microcontrolador, como su fuente de reloj o la protección anticopia. En condiciones normales no necesitaremos cambiar.
- **INCLUDE**: Permite incluir código existente en otro fichero. Lo usaremos de la forma
  INCLUDE "PIC16F886.INC"
  para incluir todas las definiciones relativas a elementos del microcontrolador que estamos programando.
- **EQU**: Sirve para asignar valores a etiquetas. Lo usaremos de la forma
  Etiqueta EQU valor.
- **ORG**: Indica a partir de qué dirección de memoria debe situarse el código que aparece a continuación. De momento la usaremos siempre como ORG 0x05, para que sitúe nuestro programa principal a partir de la posición 5 de la memoria flash.
- **END**: Indica al ensamblador el final del código fuente, donde ha de finalizar el proceso de ensamblado.

Uso de mayúsculas y minúsculas

Son indistinguibles para las directivas e instrucciones (pueden escribirse como se deseen y serán reconocidas por el ensamblador).

Las etiquetas sí deben respetar las mayúsculas y minúsculas utilizadas en su creación, tanto las que define el propio usuario como las que se incluyen mediante el fichero PIC16F886.INC.

Con el fin de facilitar la lectura y compresión de programas, utilizaremos el siguiente convenio:

- los nombres de instrucciones se escribirán siempre en minúsculas
- los nombres de directivas se escribirán en mayúsculas (Upper Case)
- las etiquetas definidas por uno mismo se escribirán siguiendo el criterio UpperCamelCase o Pascal. En minúsculas, pero comenzando por una letra mayúscula (ejemplo: Var). Si el nombre de la etiqueta está compuesto por dos o más palabras, o trozos de palabras, se escribirá en mayúsculas la primera letra de cada palabra (ejemplo: TiempoRetardo).

En cuanto a la colocación de los distintos elementos en cada línea, se alinearán por columnas (mediante tabulaciones) todos los elementos del mismo tipo: las etiquetas irán en la primera columna (si no, el ensamblador señalará una advertencia), los códigos de instrucción o directiva en la segunda, los argumentos en una tercera y, finalmente, los comentarios en la cuarta columna.

EJEMPLO. El siguiente código es un programa que suma dos datos en memoria.

```
            ;Suma los datos en las posiciones 20h y 21h de memoria
            ;y deja el resultado en la posición 22h de memoria

            LIST    p=16F886    ;Define el procesador y habilita listado
            ORG     0x00        ;Indica al compilador que coloque el programa al
principio

ResVect     goto    Inicio      ;Indica el comienzo del programa

IntVect     retfie              ;Vector de interrupción con retorno inmediato

            ORG     0x05        ;Dirección a partir de la cual se ubican
                                ;las instrucciones siguientes
Inicio      movf    20h,0
            addwf   21h,0
            movwf   22h
            goto    Inicio      ;Vuelve al comienzo de nuevo

            END                 ;Fin del programa
```

Constantes numéricas

El ensamblador MPASM soporta los sistemas de numeración decimal, binario, octal y hexadecimal. La siguiente tabla muestra la forma de representar los números según el sistema de numeración utilizado.

TIPO	SINTAXIS	EJEMPLO
Decimal	D'<cantidad>' d'<cantidad>' **. <cantidad>**	movlw D'109' movlw d'109' movlw .109
Binario	B'<cantidad>' **b'<cantidad>'**	movlw B'11000101' movlw b'11000101'
Octal	O'<cantidad>' **o'<cantidad>'**	movlw O'155' movlw o'155'

Hexadecimal	H'<cantidad>' h'<cantidad>' **0x<cantidad>** <cantidad>H (*) <cantidad>h (*) <cantidad> (*)	movlw H'6D' movlw h'6D' movlw 0x6D movlw 6DH movlw 6Dh movlw 6D

(*) dependiendo de la versión de mpasm, estos formatos pueden dar error si el primer dígito no es un número de 0-9.

Cada número utilizado, independientemente de lo que represente, puede estar en cualquier sistema de numeración, y especificado de cualquiera de las formas recogidas en la anterior tabla.

Es importante tener en cuenta que los números pueden representar tanto datos como direcciones de datos o direcciones de instrucciones. En cada caso, el número de bits empleado en el PIC16F886 es distinto, y, por tanto, también lo es el rango de datos válido.

CONSTANTE	Nº DE BITS	RANGO
Dato	8	[00h,FFh]
Dirección de instrucción	13	[0000h,1FFFh]
Dirección de dato	9	[000h,1FFh]

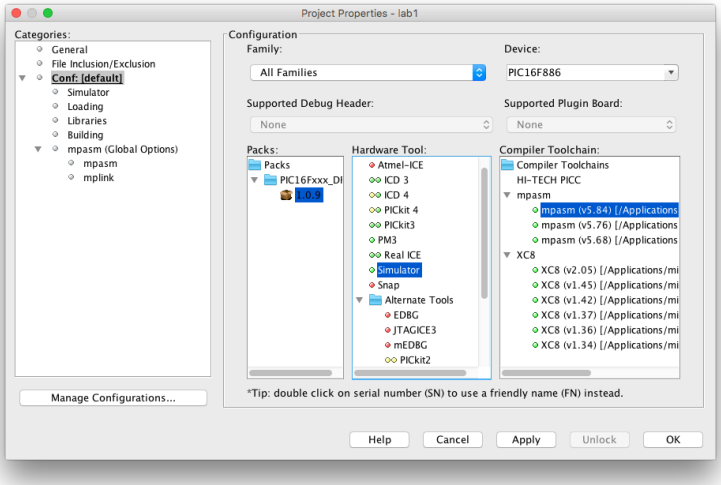
El entorno de desarrollo MPLAB X IDE

El entorno MPLAB X IDE es proporcionado por Arizona Microchip para el desarrollo de aplicaciones con microcontroladores PIC. Permite escribir programas, ensamblarlos y depurarlos, para numerosos dispositivos. Su funcionamiento se describe en el documento MPLAB X IDE User's Guide. Tanto ese manual como el software pueden descargarse a través de la página web www.microchip.com.

PASOS para la depuración en MPLAB X IDE:

1. Crear un proyecto siguiendo los pasos de la primera sesión de laboratorio.
2. Añadir un nuevo fichero con extensión asm.
3. Escribir el código del programa.
4. Añadir el fichero creado mediante el menú *Add Files to project* del menú *Project*.
5. Ensamblar mediante la opción *Build Project* o *Clean and Build Project*.

6. Una vez ensamblado con éxito, vamos a simular para verificar el correcto funcionamiento del programa. Para simular, primero elegir la herramienta de debug **simulator**, accediendo desde el Dashboard a las propiedades del proyecto.



7. Lanzar la simulación con la opción **Debug -> Debug** Project, o con el icono asociado.

8. Comprobar que el programa está correctamente cargado en memoria mostrando la memoria de instrucciones a través del menú ***Window->Target Memory View->Program Memory***.

9. Dar valores a las datos del programa. Para ello, mostrar la memoria de datos a través del menú ***Window->Target Memory View->File Registers***. Situar el cursor sobre el byte de dirección correspondiente a alguno de los datos del programa y modificar su valor.

10. Ejecutar el programa paso a paso , haciendo una pausa y avanzando con los iconos (o pulsando F7 o F8) y verificar que funciona correctamente, es decir, que en la posición de memoria correspondiente al resultado aparece el valor esperado.

11. Repetir la ejecución con diferentes valores para los datos de partida y comprobar que se genera correctamente el resultado en todos los casos.

Cuestiones y ejercicios

E1: El microcontrolador PIC16F886

Responde las siguientes preguntas sobre las características básicas del PIC16F886

1. ¿Por qué la memoria de instrucciones es de tipo Flash y no RAM?
2. ¿Cómo leerías el SFR WDTCON?
3. ¿Cómo leerías una variable colocada en la posición 0xFA de la memoria?
4. ¿Cuál es el tamaño de una instrucción?
5. ¿Cuál es el tamaño del contador de programa?
6. ¿Con qué nombre se conoce al registro acumulador?
7. ¿Qué instrucciones del PIC16F886 no llevan argumentos?
8. ¿Cuáles llevan sólo uno? ¿Qué representa ese argumento?
9. ¿Cuáles llevan dos? ¿Qué indica el segundo argumento en estos casos?
10. ¿De cuántas direcciones son las instrucciones del PIC16F886?
11. ¿Qué hace la instrucción goto?
12. ¿y la instrucción movf?

E2: Acciones básicas en lenguaje ensamblador del PIC16F886

Escribe una o varias instrucciones en ensamblador del PIC16F886 para realizar las siguientes acciones básicas. Supón en todos los casos que Var es una variable almacenada en la posición h'020' de memoria, y que las constantes del ejemplo están representadas en base decimal.

ACCIÓN	EJEMPLO	CÓDIGO ENSAMBLADOR
Inicializar una variable a 0	Var = 0	
Inicializar una variable a un valor distinto de 0	Var = 7	
Incrementar una variable	Var = Var + 1	
Decrementar una variable	Var = Var - 1	
Sumarle a una variable un	Var = Var + 10	

número constante		
Restarle a una variable un número constante	Var = Var - 15	

E3: Constantes numéricas

Determina para todas las constantes numéricas que aparecen en el siguiente programa si representan un Dato, una Dirección de dato o una Dirección de instrucción. ¿Cuáles están fuera de rango? ¿Qué ocurre con esos datos? Verifícalo utilizando MPLAB X IDE y anota las consecuencias que producen esas situaciones en compilación y en ejecución.

```
            LIST    p=16F886
            ORG     o'0'
            goto    Inicio
            ORG     0x05

Inicio      movlw   h'190'
            subwf   h'021',0
            movlw   .290
            movwf   h'022'
            goto    0Ch
            movf    h'020',0
            movwf   h'022'
            goto    b'0101'
            movlw   100h
            goto    Inicio

            END
```

P1: Programación del PIC16F886

Escribe un programa en ensamblador del PIC16F886 que permita intercambiar los datos almacenados en las posiciones 30h y 31h de memoria. Verifica el funcionamiento del programa utilizando el simulador de MPLAB X IDE.

P2: Depuración paso a paso

El siguiente programa debería calcular la expresión matemática

$$A = (C + 20) - B$$

donde A, B y C son variables en memoria, en las posiciones h'020', h'021' y h'022', respectivamente, y 20 un valor decimal constante. Sin embargo, no funciona según lo esperado. Depúralo con MPLAB X IDE y corrige los errores hasta conseguir que calcule dicha expresión. Explica cada error que te vas encontrando.

```
            ;Programa Ejercicio P2

            LIST    p=16F886
            ORG     0x00
            goto    Inicio
            ORG     0x05

Inicio      movf    h'022'
            addwf   20h
            subwf   h'021'
            movwf   h'020'
            goto    Inicio

            END
```

P3: "Blinky"

Utilizando el esqueleto de programa disponible en el Aula Virtual, realiza un programa que lea el puerto **PORTA4** y escriba ese contenido en el puerto **PORTC4**. Necesitarás utilizar las instrucciones **bcf** y **bsf**. Puedes leer el puerto A completo y escribir el puerto B completo si quieres.

Una vez que compruebes el funcionamiento con MPLAB X IDE, puedes configurar la placa, mediante cables, para que el generador de pulsos se conecte al puerto **PORTA4** y el puerto **PORTC4** se conecte a un led azul del array de la parte derecha.