**¿Qué es un sistema embebido?**

Un sistema embebido, embarcado o empotrado (integrado, incrustado) es un sistema de computación basado en un microprocesador o un microcontrolador diseñado para realizar una o algunas pocas funciones dedicadas,​ frecuentemente en un sistema de computación en tiempo real. Al contrario de lo que ocurre con los ordenadores de propósito general (como por ejemplo una computadora personal o PC) que están diseñados para cubrir una amplia gama de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas. En un sistema embebido la mayoría de los componentes se encuentran incluidos en la placa base (tarjeta de vídeo, audio, módem, etc.) y muchas veces los dispositivos resultantes no tienen el aspecto de lo que se suele asociar a una computadora. Algunos ejemplos de sistemas embebidos podrían ser dispositivos como un taxímetro, un sistema de control de acceso, la electrónica que controla una máquina expendedora o el sistema de control de una fotocopiadora entre otras múltiples aplicaciones.

Por lo general los sistemas embebidos se pueden programar directamente en el lenguaje ensamblador del microcontrolador o microprocesador incorporado sobre el mismo, o también, utilizando los compiladores específicos, pueden utilizarse lenguajes como C o C++; en algunos casos, cuando el tiempo de respuesta de la aplicación no es un factor crítico, también pueden usarse lenguajes Orientados a Objetos como JAVA.

Existen también plataformas desarrolladas por distintos fabricantes que proporcionan herramientas para el desarrollo y diseño de aplicaciones y prototipos con sistemas embebidos desde ambientes gráficos, algunos ejemplos de estas son: Arduino, mbed, Raspberry Pi, BeagleBone, etc.

Hay que señalar que las líneas reales de distinción entre microprocesador, microcontrolador y microcomputador en un solo chip están difusas, y se denominan en ocasiones de manera indistinta unos y otros.

Lenguaje ensamblador

Lenguaje de bajo nivel. Un programa utilitario llamado ensamblador es usado para traducir sentencias del lenguaje ensamblador al código de máquina del computador objetivo. La transformación del lenguaje ensamblador en código máquina la realiza un programa ensamblador, y la traducción inversa la puede efectuar un desensamblador

Características:

* El código escrito en lenguaje ensamblador posee una cierta dificultad de ser entendido ya que su estructura se acerca al lenguaje máquina, es decir, es un lenguaje de bajo nivel.
* El lenguaje ensamblador es difícilmente [portable](https://es.wikipedia.org/wiki/Portabilidad), es decir, un código escrito para un microprocesador, puede necesitar ser modificado, para poder ser usado en otra máquina distinta. Al cambiar a una máquina con arquitectura diferente, generalmente es necesario reescribirlo completamente.
* Los programas hechos por un programador experto en lenguaje ensamblador pueden ser más rápidos y consumir menos recursos del sistema (ej: memoria RAM) que el programa equivalente compilado desde un lenguaje de alto nivel. Al programar cuidadosamente en lenguaje ensamblador se pueden crear programas que se ejecutan más rápidamente y ocupan menos espacio que con lenguajes de alto nivel. Conforme han evolucionado tanto los procesadores como los compiladores de lenguajes de alto nivel, esta característica del lenguaje ensamblador se ha vuelto cada vez menos significativa. Es decir, un compilador moderno de lenguaje de alto nivel puede generar código casi tan eficiente como su equivalente en lenguaje ensamblador.[1](https://es.wikipedia.org/wiki/Lenguaje_ensamblador#cite_note-1)​
* Con el lenguaje ensamblador se tiene un control muy preciso de las tareas realizadas por un [microprocesador](https://es.wikipedia.org/wiki/Microprocesador) por lo que se pueden crear segmentos de código difíciles y/o muy ineficientes de programar en un lenguaje de alto nivel, ya que, entre otras cosas, en el lenguaje ensamblador se dispone de instrucciones del CPU que generalmente no están disponibles en los lenguajes de alto nivel.

**Ejemplo:**

en el lenguaje ensamblador para un procesador [x86](https://es.wikipedia.org/wiki/X86):

La sentencia

* MOV AL, 61h

Asigna el valor [hexadecimal](https://es.wikipedia.org/wiki/Hexadecimal) 61 (97 [decimal](https://es.wikipedia.org/wiki/Sistema_de_numeraci%C3%B3n_decimal)) al registro "AL".

El programa ensamblador lee la sentencia de arriba y produce su equivalente [binario](https://es.wikipedia.org/wiki/Binario) en [lenguaje de máquina](https://es.wikipedia.org/wiki/Lenguaje_de_m%C3%A1quina).

* Binario: 10110000 01100001 (hexadecimal: B061)

El mnemónico MOV es un *código de operación* u "[opcode](https://es.wikipedia.org/wiki/Opcode" \o "Opcode)". El *opcode* es seguido por una lista de argumentos o *parámetros*, completando una típica instrucción de ensamblador. En el ejemplo, AL es un registro de 8 bits del procesador, al cual se le asignará el valor hexadecimal 61 especificado.

El código de máquina generado por el ensamblador consiste de 2 bytes. El primer byte contiene empaquetado la instrucción MOV y el código del registro hacia donde se va a mover el dato:

1011 0000 01100001

| | |

| | +---- Número 61h en binario

| |

| +--- Registro AL

+-------- Instrucción MOV

En el segundo byte se especifica el número 61h, escrito en binario como 01100001, que se asignará al registro AL, quedando la sentencia ejecutable como:

* 10110000 01100001

La cual puede ser entendida y ejecutada directamente por el procesador.

# RISC Computador con conjunto de instrucciones reducido

es un tipo de diseño de CPU generalmente utilizado en microprocesadores o microcontroladores con las siguientes características fundamentales:

Instrucciones de tamaño fijo y presentadas en un reducido número de formatos.

Solo las instrucciones de carga y almacenamiento acceden a la memoria de datos.

RISC es una filosofía de diseño de CPU para computadora que está a favor de conjuntos de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse. El tipo de procesador más comúnmente utilizado en equipos de escritorio, el x86, está basado en CISC en lugar de RISC, aunque las versiones más nuevas traducen instrucciones basadas en CISC x86 a instrucciones más simples basadas en RISC para uso interno antes de su ejecución.

Esto se hace ya que aunque el procesamiento era cada vez más rápido, el proceso de acceder a la memoria no era tan rápido y se forma un culo de botella.

# CISC *Computador con Conjunto de Instrucciones Complejas*

es un modelo de arquitectura de computadores. Los microprocesadores CISC tienen un conjunto de instrucciones que se caracteriza por ser muy amplio y permitir operaciones complejas entre operandos situados en la memoria o en los registros internos, en contraposición a la arquitectura RISC.1​

Es capaz de ejecutar varios centenares de instrucciones complejas diferentes siendo extremadamente versátil.

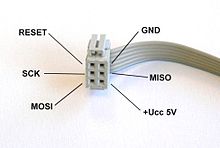
Este tipo de arquitectura dificulta el paralelismo entre instrucciones, por lo que, en la actualidad, la mayoría de los sistemas CISC de alto rendimiento implementan un sistema que convierte dichas instrucciones complejas en varias instrucciones simples del tipo RISC, llamadas generalmente microinstrucciones.

Los CISC pertenecen a la primera corriente de construcción de procesadores, antes del desarrollo de los RISC. Ejemplos de ellos son: Motorola 68000, Zilog Z80 y toda la familia Intel x86, AMD x86-64 usada en la mayoría de las computadoras personales actuales.

# AVR

Los AVR son una familia de microcontroladores RISC del fabricante estadounidense Atmel

El AVR es una CPU de arquitectura Harvard. Tiene 32 registros de 8 bits. Algunas instrucciones sólo operan en un subconjunto de estos registros. La concatenación de los 32 registros, los registros de entrada/salida y la memoria de datos conforman un espacio de direcciones unificado, al cual se accede a través de operaciones de carga/almacenamiento. A diferencia de los **microcontroladores PIC**, el stack se ubica en este espacio de memoria unificado, y no está limitado a un tamaño fijo.

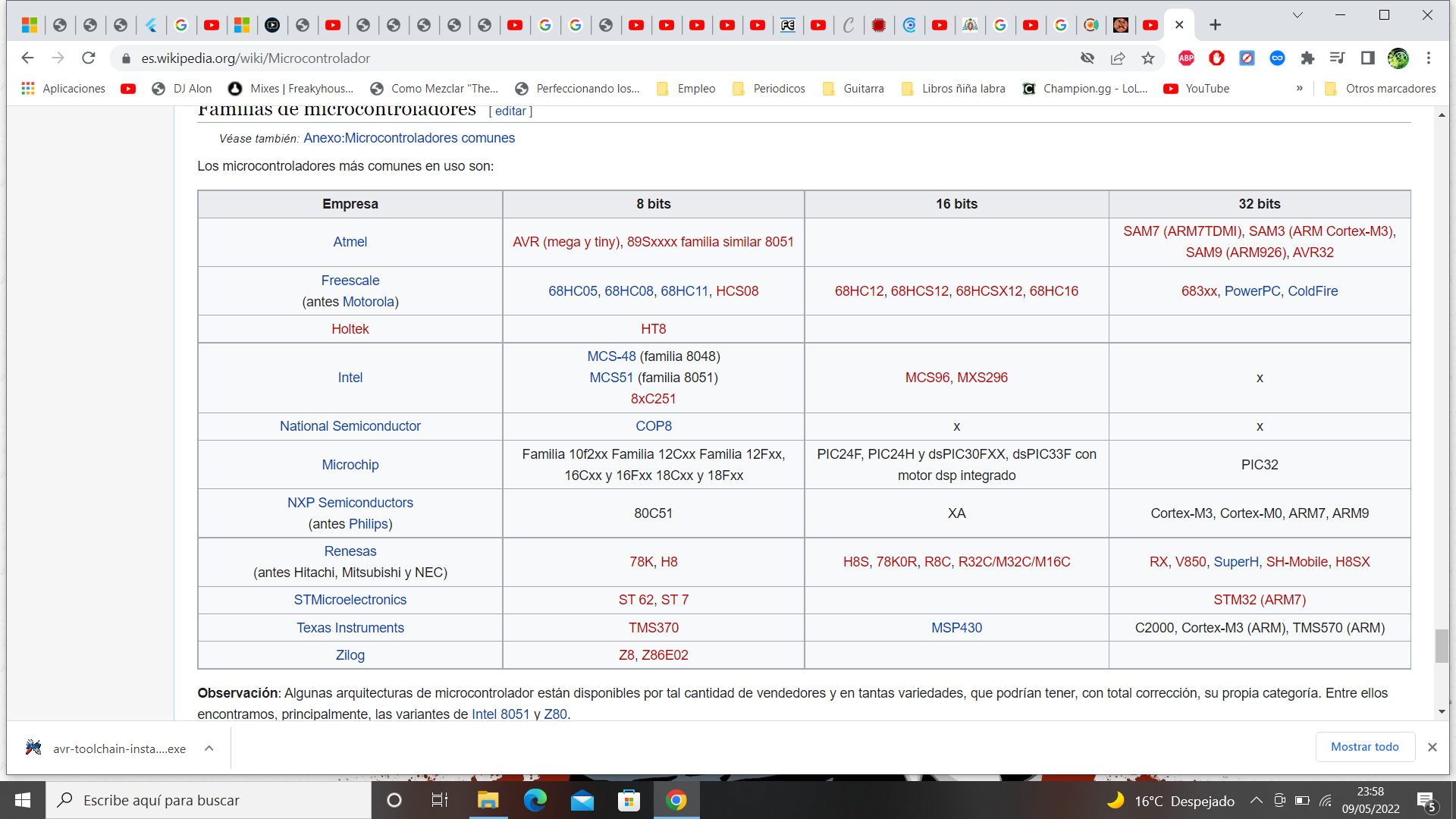
[](https://commons.wikimedia.org/wiki/File:1-3-2007-Avr2.jpg)

Descripción de pines del cable [ISP](https://es.wikipedia.org/wiki/Programaci%C3%B3n_en_el_sistema), tal y como es usado por algunos fabricantes

# Microcontrolador PIC

Los PIC son una familia de microcontroladores tipo RISC desarrollados por la empresa microship technology.

* Bajo costo
* Eficiencia
* Mucha documentación
* Fácil adquisición de los kit de desarrollo



# Microcontroladores

## Como funcionan?

Como el hardware ya viene integrado en un solo chip, para usar un microcontrolador se debe especificar su funcionamiento por software a través de programas que indiquen las instrucciones que el microcontrolador debe realizar. En una memoria se guardan los programas y un elemento llamado CPU se encarga de procesar paso por paso las instrucciones del programa. Los lenguajes de programación típicos que se usan para este fin son *ensamblador* y *C*, pero antes de grabar un programa al microcontrolador hay que compilarlo a hexadecimal que es el formato con el que funciona el microcontrolador.

Para diseñar programas es necesario conocer los bloques funcionales básicos del microcontrolador, estos bloques son:

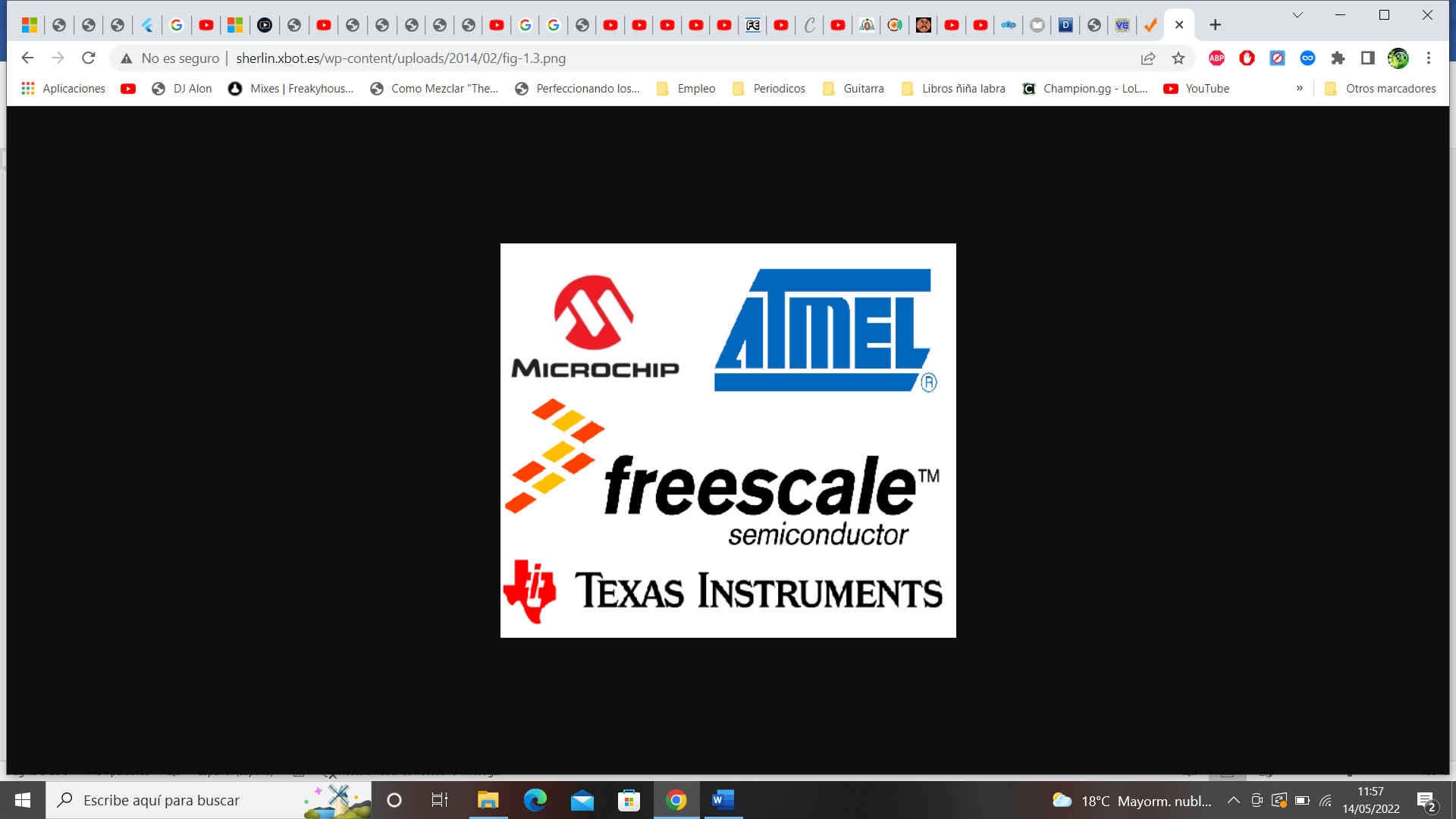
* **CPU** (Unidad central de proceso)
* **Memoria ROM** (Memoria de solo lectura)
* **Memoria RAM** (Memoria de acceso aleatorio)
* **Líneas de entrada y salida** (Periféricos)

La CPU del micro posee una memoria independiente y de acceso rápido que se usa para almacenar datos denominados registros. **Si estos registros son de 8 bits se dice que el microcontrolador es de 8 bits.**

Para programar un microcontrolador se necesitan 3 cosas:

1. Una computadora
2. Software de programación (incluyendo un compilador).
3. Un circuito programador

Si no disponemos de lo necesario para grabar un micro la alternativa es **simularlo. Con el software Proteus** podemos hacer la simulación, pero atención, las simulaciones pueden tener comportamientos diferentes a lo que ocurriría en la vida real.



# Arquitectura de los microcontroladores

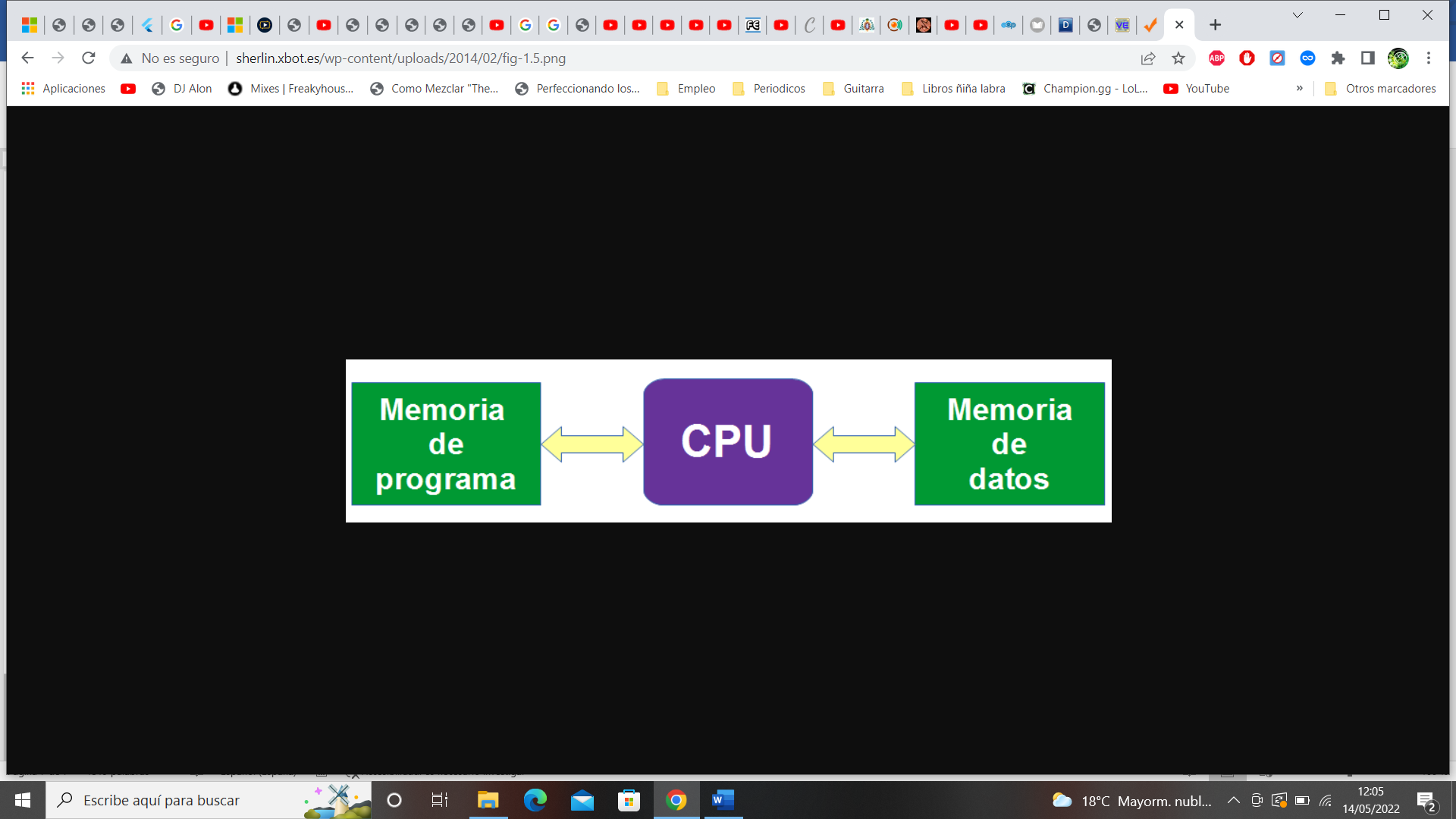
Las dos arquitecturas principales usadas en la fabricación de micros son arquitectura de Von Neumann y arquitectura Harvard. Además, estas arquitecturas pueden tener procesadores de tipo CISC o de tipo RISC.

## Arquitectura Von Neumann



Arquitectura en la que a la memoria de programa y datos se accede por un mismo bus (pues se guardan en la misma memoria), creando cuello de botella en la velocidad con la que se realizan los procesos. Su principal ventaja es el ahorro de líneas de entrada y salida.

## Arquitectura Harvard:



Posee dos memorias, una para programa y otra para datos con su bus de datos independiente, aumentando la velocidad de procesamiento en gran medida.

Actualmente, la tendencia de los microcontroladores es usar este tipo de arquitectura.

Recordamos

**Procesador de tipo CISC (Complex Instruction Set Computer)**

Un procesador que permita manejar un amplio juego de instrucciones es llamado de tipo *CISC* que en español significa «Ordenador con Juego de Instrucciones Complejo», programar en este tipo de arquitectura requiere en algunos casos del dominio de hasta centenares de instrucciones.

**Procesador de tipo RISC (Reduced Instruction Set Computer)**

Cuando un procesador está diseñado para manejar pocas instrucciones, pero sin afectar las prestaciones del ordenador es llamada de tipo *RISC* que en español significa «Ordenador con Juego de Instrucciones Reducido», esto **permite programar con mucha más facilidad** y, por si fuera poco, los circuitos de tipo *RISC* disponen de una **estructura que busca como mínimo la instrucción próxima a ejecutar mientras realiza la instrucción actual. Esta estructura permite lograr no solo mayor velocidad de proceso sino también procesar cada instrucción con la misma velocidad.**

**NOTA:***Microchip* introdujo la arquitectura *Harvard* con procesador tipo *RISC*en sus microcontroladores cuando el mercado era dominado por microcontroladores con arquitectura de Von Neumann, desde entonces las ventajas que ofreció esta nueva tecnología permitieron a Microchip sobresalir como uno de los más grandes fabricantes de microcontroladores en el mundo.

# Estructura de un código fuente:

**Inicio**

**Declaración de variables y constantes**

**Operaciones**

**Fin**

Cada programa debe tener un inicio y un fin bien determinados. Las variables y las constantes derivan de los datos y condiciones del problema a resolver, las operaciones se encargan de hacer los procesos y comunicar los resultados.

Cuando se hace un programa, se especifica **cómo se hará uso de las memorias disponibles en el microcontrolador**

* **Memoria de programa:**  memoria encargada de guardar las instrucciones.
* **Memoria de datos:**  Guarda cada resultado que surja a medida que el programa corre.

**Dado que las memorias tienen un límite de capacidad siempre hay que estar atento de no excederlo.**

# Cosas que deberíamos mirar de un microcontrolador:

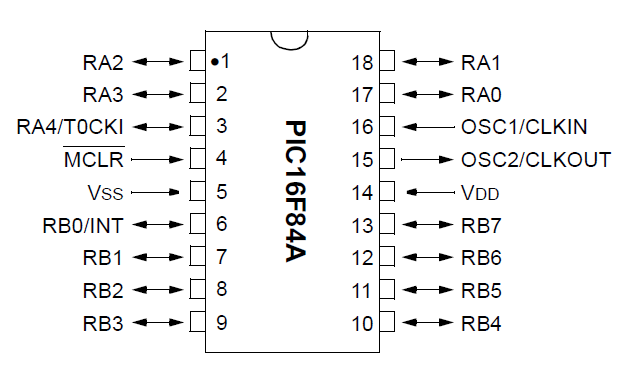
Pondremos de ejemplo el PIC16F84A:

Sus características principales son:

* Set de solo 35 instrucciones
* Memoria de programa de 1 KB (equivale a 1024 instrucciones)
* Máxima velocidad de operación: 20 MHz
* 68 Bytes de RAM
* 64 Bytes de EEPROM
* 4 fuentes de interrrupción
* 2 puertos de salida
* 13 Líneas de I/O configurables individualmente
* 25 mA de corriente por pin

**Descripción de los pines**

El PIC 16F84A contiene 13 líneas I/O (entrada o salida), 5 pertenecen al puerto A y 8 pertenecen al puerto B.

[](http://sherlin.xbot.es/wp-content/uploads/2014/02/fig-2.2.png)fig 2.2 Distribución de pines del PIC 16F84A

* **VDD:** Alimentación positiva (5 voltios)
* **VSS:** Tierra (0 voltios)
* **MCLR:** pin de resetear
* **OSC1, OSC2:** Conexión de oscilador
* **RA0, RA1, RA2, RA3, RA4:** líneas I/O del puerto A
* **RB0, RB1, RB2, RB3, RB4, RB5, RB6, RB7:** líneas I/O del puerto B

# El Oscilador externo

Todo microcontrolador requiere de un oscilador el cual es el encargado de definir la velocidad con la que se procesaràn las instrucciones. El PIC 16F84A tiene cuatro posibles configuraciones de oscilador:

**LP:** Cristal usado para bajo consumo de potencia

**XT:** Cristal de cuarzo estándar

**HS:** Cristal de alta velocidad

**RC:** Resistor/capacitor, ofrece menor precisión en el oscilador

En caso de hacer una conexión RC, habrá que saber que valores de R y C se recomiendan y en lugar de ir a los dos pines a los que iría un oscilador, iría sólo a uno. En el otro pin nos daría una salida de oscilaciones con frecuencias menores a la que nos da en el pin conectado. **En este punto nos falta información.**

## Velocidad de ejecución de un programa:

Un microcontrolador va ejecutando su programa paso por paso, a cada paso lo llamamos ciclo de máquina y la velocidad con la que se ejecuta está relacionada con el oscilador de la siguiente manera:

Donde F0 será la frecuencia con la que trabajará el microcontrolador y F la frecuencia del oscilador. Esto nos dice que por cada 4 oscilaciones del oscilador tenemos un ciclo de máquina. Algunas instrucciones requieren de un solo ciclo de máquina para ejecutarse, otras de más.

El tiempo que dura un ciclo de máquina también es importante, depende de la operación anterior y viene dado por:

### Ejemplo:

Si tenemos un micro con un oscilador de 4MHz, aplicando la fórmula nos da una velocidad de programa de 1MHz. Ahora aplicando la fórmula del tiempo nos daría un tiempo por ciclo de 1us. Conclusión, con un cristal de 4MHz una instrucción básica tardará en ejecutarse 1us.

## Fusibles:

Son configuraciones de la forma de funcionamiento de un micro. Por ejemplo el PIC 16F84A posee cuatro fusibles que son:

**1. Oscilador (OSC):**Le indica al PIC el tipo de oscilador que usará.

**2. Perro Guardián (WDT):** Permite autorresetear al PIC en caso de que quede en un bucle infinito.

**3. Temporizador de encendido (PWRT):**Se encarga de retardar la inicialización del PIC cuando se conecta a la fuente de alimentación, esto con el fin de garantizar que cuando el programa inicie el voltaje ya sea estable.

**4. Protección de código (CD):**Al habilitar esta función se impide que el programa grabado en el microcontrolador sea leído posteriormente.

Para indicarle a un PIC (importante no a todos los micros, sino a algunos PIC) que fusibles queremos habilitar o no a través de lenguaje de programación se escribe una línea de código llamada palabra de configuración.

**Ejemplo:**  \_\_CONFIG        \_XT\_OSC  &  \_WDT\_OFF  &  \_PWRTE\_ON  &  \_CP\_OFF

La anterior línea es una *palabra de configuración* escrita en lenguaje ensamblador y configura los fusibles así:

Tipo de oscilador que se usará:**XT**

Perro guardián: **deshabilitado**

Temporizador de encendido: **habilitado**

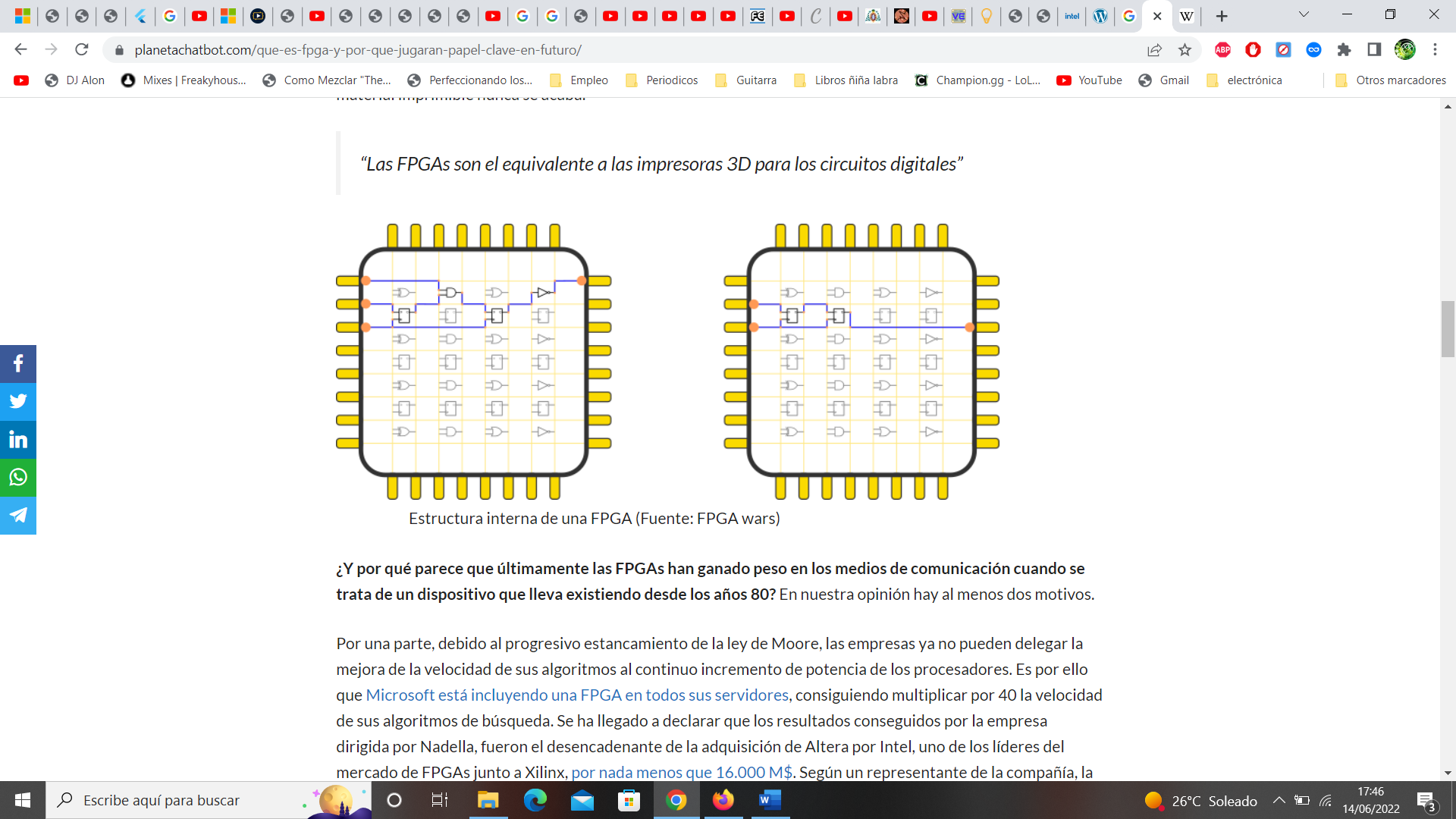
Protección de código: **deshabilitada**

**Memoria OTP:**  Aquella que solo se puede grabar 1 única vez.

## FPGAs

Las FPGAs son unos dispositivos nos permiten describir un circuito digital usando un lenguaje específico (los dos más comunes son VHDL y Verilog) y que, tras cargarlo en el integrado, es creado físicamente en el chip. Su nombre es un acrónimo inglés que significa matriz de puertas reprogramable o Field Programmable Gate Array.

Internamente se componen principalmente de cables, puertas lógicas, biestables, y puertos de entrada y salida. Todo ello sin conectar, como una plantilla en blanco, hasta que se les carga un bitstream -un archivo generado a partir de la descripción del circuito-. Un símil que nos parece muy fácil de entender, es que las FPGAs son el equivalente a las impresoras 3D para los circuitos digitales. A diferencia eso sí de las impresoras, es posible reprogramar una FPGA tantas veces como se necesite, es decir, siguiendo con la analogía, el material imprimible nunca se acaba.



**¿Y por qué parece que últimamente las FPGAs han ganado peso en los medios de comunicación cuando se trata de un dispositivo que lleva existiendo desde los años 80?**

 la integración de FPGAs y CPUs en un único integrado podría incrementar la velocidad hasta un 50% con respecto a su uso en integrados separados.

el año pasado y haciendo uso de la ingeniería inversa, [Clifford Wolf (entre otros) desarrolló y liberó](http://www.clifford.at/icestorm/) todas las herramientas necesarias para generar el bitstream de una Lattice iCE40. Nunca hasta ahora se había podido lograr esto sin depender de las herramientas privativas (y muchas veces prohibitivas en cuanto a coste) del fabricante. Esto ha significado que muchos aficionados se están introduciendo en un campo hasta ahora limitado principalmente a investigadores e industrias muy específicas.

