

Microprocesadores

LEOPOLDO PARRA REYNADA

Red Tercer Milenio

MICROPROCESADORES

MICROPROCESADORES

LEOPOLDO PARRA REYNADA

RED TERCER MILENIO



AVISO LEGAL

Derechos Reservados © 2012, por RED TERCER MILENIO S.C.

Viveros de Asís 96, Col. Viveros de la Loma, Tlalnepantla, C.P. 54080, Estado de México.

Prohibida la reproducción parcial o total por cualquier medio, sin la autorización por escrito del titular de los derechos.

Datos para catalogación bibliográfica

Leopoldo Parra Reynada

Microprocesadores

ISBN 978-607-733-181-0

Primera edición: 2012

Revisión pedagógica: Sofía Cabrera Ruíz

Revisión editorial: Ma. Eugenia Buendía López

DIRECTORIO

Bárbara Jean Mair Rowberry
Directora General

Rafael Campos Hernández
Director Académico Corporativo

Jesús Andrés Carranza Castellanos
Director Corporativo de Administración

Héctor Raúl Gutiérrez Zamora Ferreira
Director Corporativo de Finanzas

Ximena Montes Edgar
Directora Corporativo de Expansión y Proyectos

ÍNDICE

<i>Introducción</i>	5
<i>Objetivo general de aprendizaje</i>	7
<i>Mapa conceptual</i>	8
Unidad 1: Estructuras fundamentales	9
Mapa conceptual	10
Introducción	11
1.1 Componentes básicos de una computadora digital	12
1.2 Elementos básicos de un microprocesador	17
1.3 Elementos básicos de un microcontrolador	23
1.4 Alimentación	27
1.5 Señal de reloj	28
1.6 Reset	29
1.7 Bus de datos	30
1.8 Bus de direcciones	31
1.9 Mapa de memoria	32
1.10 Interrupciones	34
Autoevaluación	37
Unidad 2: Estructura de un microprocesador	40
Mapa conceptual	41
Introducción	42
2.1 Operaciones básicas entre registros	43
2.2 Registros con elementos aritméticos básicos	45
2.2.1 Operaciones con un registro sencillo	47
2.2.2 Aritmética básica con registros	50
2.2.3 Operaciones condicionales y de salto	54
2.3 Conceptos básicos de instrucción y código de operación	59
2.4 Microprocesador hipotético	62

2.5 Operación del microprocesador, fases de búsqueda y ejecución	65
2.6 Concepto y operación del microprocesador	66
2.7 Concepto de modo de direccionamiento	71
2.8 Introducción a la programación en ensamblador	74
Autoevaluación	82
Unidad 3: Características específicas de microprocesadores de 8 bits	85
Mapa conceptual	86
Introducción	87
3.1 Análisis de las características de un microcontrolador comercial de 8 bits	88
3.2 Arquitectura del microcontrolador de 8 bits	91
3.3 Modos de direccionamiento del PIC 16F628A	95
3.4 Conjunto de instrucciones del microcontrolador PIC 16F628A	96
3.5 Primer programa en ensamblador	106
Autoevaluación	117
Unidad 4: Programación de entrada/salida	123
Mapa conceptual	124
Introducción	125
4.1 Comunicación de datos en forma paralela	126
4.2 Programación, control y direccionamiento de los elementos de entrada/salida paralelos	129
4.3 Comunicación de datos en forma serial	137
4.4 Programación, control y direccionamiento de los elementos de entrada/salida	141
4.5 Programas de aplicación	147
Autoevaluación	154
Unidad 5: Características específicas de microprocesadores de 16 y 32 bits	159

Mapa conceptual	160
Introducción	161
5.1 Breve historia de los microprocesadores de 16, 32 y 64 bits	162
5.2 Capacidad de memoria	166
5.3 Programas diseñados para 16 y 32 bits	172
5.4 Direccionamiento de unidades o bloques de memoria	177
5.5 Arquitectura y set de instrucciones para microprocesadores de 16 y 32 bits	181
Autoevaluación	188
<i>Bibliografía</i>	192
<i>Glosario</i>	193



Los microprocesadores están por todas partes.¹

Esto significa que los microprocesadores y microcontroladores se han convertido en parte de la vida diaria, y esto a su vez implica que cualquier persona interesada en el área de la electrónica o del control debe saber cómo funcionan y cómo se aplican estos dispositivos. Precisamente, el objetivo de este libro es proporcionar los principios básicos de los microprocesadores y microcontroladores, combinando la teoría y la práctica.

¹ Imágenes propiedad de sus respectivos fabricantes.

OBJETIVO GENERAL DE APRENDIZAJE

Conocer el panorama general de los microprocesadores y microcontroladores digitales, especialmente de los dispositivos de 8 bits. Identificar la estructura de un circuito de proceso lógico, así como de sus bloques internos y externos, las señales que requiere para funcionar, etc. Al finalizar el curso, el estudiante tendrá un conocimiento más profundo acerca de los circuitos de proceso lógico y de cómo aprovecharlos para distintos proyectos.

MAPA CONCEPTUAL



UNIDAD 1

ESTRUCTURAS FUNDAMENTALES

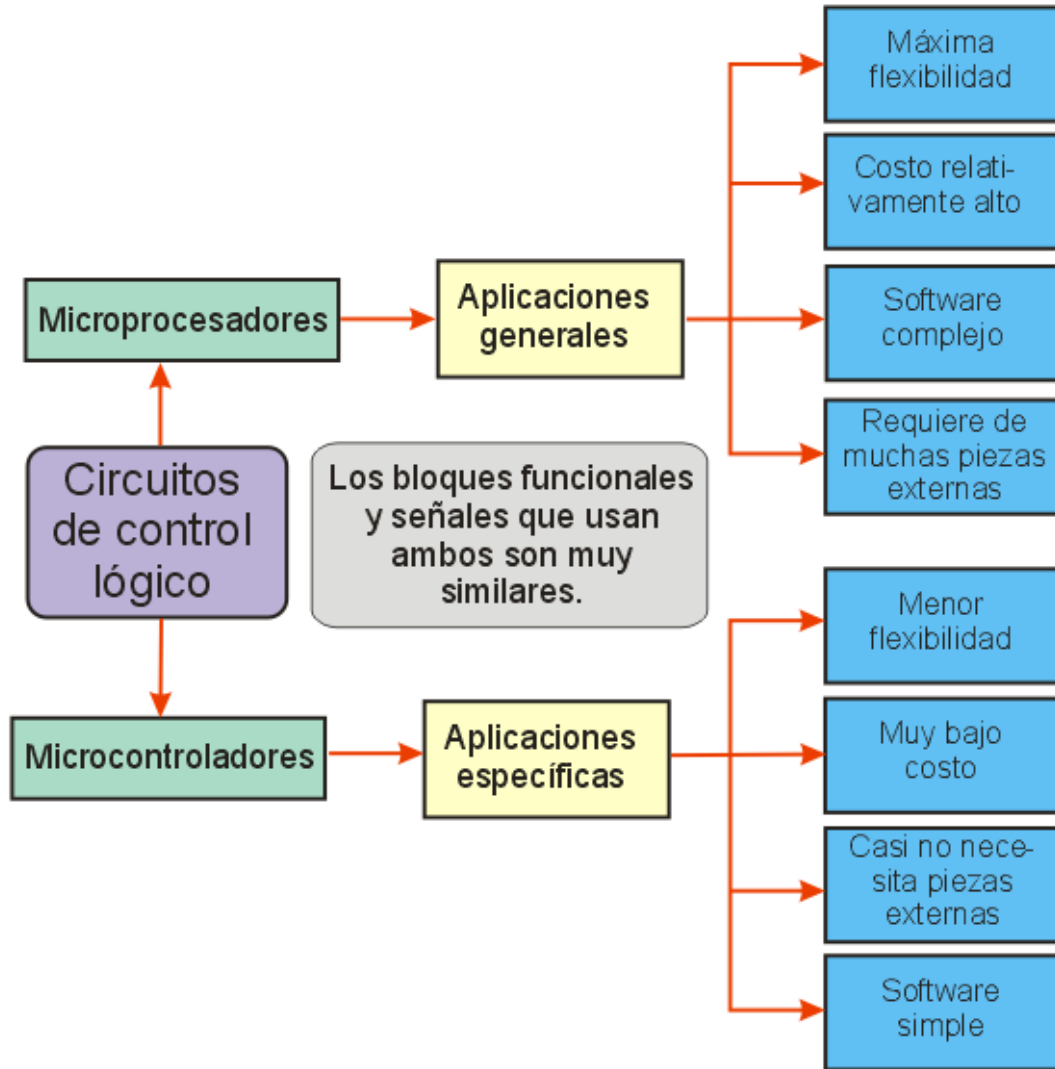
OBJETIVO

Entender el panorama general de los microprocesadores y microcontroladores, y su utilidad en la sociedad moderna. Conocer la aplicación de los circuitos de proceso lógico de información, y cómo han modificado la forma en la que los usuarios interactúan con aparatos diversos, así como comprender sus secciones principales, las señales con las que trabajan, sus bloques funcionales más importantes, etc.

TEMARIO

- 1.1 COMPONENTES BÁSICOS DE UNA COMPUTADORA DIGITAL
- 1.2 ELEMENTOS BÁSICOS DE UN MICROPROCESADOR
- 1.3 ELEMENTOS BÁSICOS DE UN MICROCONTROLADOR
- 1.4 ALIMENTACIÓN
- 1.5 SEÑAL DE RELOJ
- 1.6 RESET
- 1.7 BUS DE DATOS
- 1.8 BUS DE DIRECCIONES
- 1.9 MAPA DE MEMORIA
- 1.10 INTERRUPCIONES

MAPA CONCEPTUAL



INTRODUCCIÓN

Para comprender lo que es un microprocesador y el porqué estos dispositivos han revolucionado la tecnología del control, primero es necesario conocer su estructura interna, además de sus ventajas e inconvenientes; todo esto con el objetivo de comprender mejor los dispositivos con los que se trabajará de aquí en adelante, y para tener un panorama claro de lo que se puede esperar de un microprocesador o de un microcontrolador. Esto servirá como puerta de entrada para estudiar los procesos internos dentro de estos dispositivos, y cómo se pueden aprovechar para que el circuito se comporte exactamente como se desea.

1.1 COMPONENTES BÁSICOS DE UNA COMPUTADORA DIGITAL

Si bien las computadoras digitales han salido de los laboratorios de investigación y de las grandes empresas, para llegar a los hogares y a las oficinas; esto ha sido el resultado de largos años de trabajo de múltiples compañías alrededor del mundo, que desarrollaron componentes cada vez más confiables, pequeños y económicos, al grado de que en la



Las computadoras modernas se han convertido en parte del ambiente cotidiano.

actualidad, una computadora con una potencia de cálculo que hace un par de décadas habría costado cientos de miles de dólares, ahora se puede obtener por menos de mil dólares, convirtiéndose en una herramienta indispensable para realizar trabajos diarios.

Al observar una computadora moderna, pocos imaginan la enorme cantidad de elementos que tienen que trabajar en estrecha colaboración para que este dispositivo realice adecuadamente su labor. Desde el punto de vista exclusivamente físico, se necesitan una gran cantidad de componentes que, al ser ensamblados adecuadamente, dan el soporte para la ejecución de complejos programas de software, y la combinación de hardware y software es lo que hace de una computadora digital, un auxiliar invaluable en las labores cotidianas. En la siguiente figura, se muestran los componentes básicos de una computadora personal (PC) moderna. A continuación se describe brevemente qué es cada una de estas partes, y su función principal.



Estructura básica de una computadora digital.

En realidad, una computadora opera alrededor de un elemento central, que es la tarjeta madre (1); en este elemento se concentran todos los protocolos de comunicación que permiten la interacción de los distintos componentes, así que funciona como una especie de “estación de tránsito principal” para canalizar los datos desde y hacia todos los elementos que sean necesarios.

En esta tarjeta madre se instala el microprocesador (2), el cual es el encargado de realizar todos los cálculos indicados en el software que se estén ejecutando. Este dispositivo determina la potencia de cómputo del equipo, y se considera como la pieza individual más importante dentro de una computadora personal.

Para poder almacenar temporalmente todos los datos que necesita para trabajar, el microprocesador requiere de una memoria RAM (3), y la capacidad de esta última determina la complejidad de los programas o de los datos que puede manejar una PC. Como en la memoria RAM sólo pueden guardarse datos en forma temporal (cuando se apaga el equipo, se pierde la información que contiene), se requieren de otros medios de almacenamiento más permanentes para guardar el sistema operativo, los programas y los datos que vaya generando el usuario, y para ello está un disco duro (4) y una unidad de

discos ópticos (5). Estos tres elementos, RAM, disco duro y unidad óptica, son los principales medios de almacenamiento de datos de una computadora personal, aunque no son los únicos.

Para introducir las órdenes del usuario, es necesario tener elementos especiales, como el teclado y el ratón (6). A estos elementos recientemente se les ha añadido un micrófono, una cámara web, un escáner, etc., pero incluso en la actualidad, estos últimos componentes aún se consideran como equipo opcional, mientras que el teclado y el ratón son elementos indispensables en cualquier computadora moderna.

La computadora para poder presentar sus resultados al usuario, también necesita elementos de salida de información, que por lo general consisten en una tarjeta de video (7) a la cual está conectado un monitor (8), y una tarjeta de sonido (normalmente incluida en la tarjeta madre) de la que salen un par de bocinas (9); con estos elementos, el usuario puede ver y escuchar lo que está haciendo la máquina, y aprovecharlo ya sea para el trabajo diario o para el entretenimiento.

Todo lo anterior está contenido en un gabinete, dentro del cual también se encuentra la fuente de poder que se encarga de alimentar adecuadamente a todos los componentes de la computadora (10). Se puede indicar que esta es la estructura básica de una PC moderna, y aunque poco a poco se le han añadido más elementos externos y/o internos, los que se mencionaron son los elementos más importantes y que se consideran indispensables, para que una computadora actual sea capaz de ejecutar el software que la convertirá en una poderosa herramienta de productividad o en un centro de entretenimiento de propósito general.

Por increíble que parezca, estos mismos elementos se pueden encontrar en las computadoras portátiles, aunque reducidos y concentrados en un gabinete de pequeñas dimensiones. Si bien en



algunas computadoras portátiles modernas ya se han eliminado algunos elementos (como la unidad óptica) o se han sustituido por dispositivos más adecuados para este tipo de máquinas (como el *touchpad* en lugar del ratón), los demás componentes siguen presentes, y es por eso que el mismo software que se utiliza en máquinas de escritorio se puede ejecutar en una computadora portátil, a pesar de la enorme diferencia en su aspecto externo.



El software tiene un papel igual de importante que el hardware en la estructura de una PC moderna.

Y ya que se menciona al software, no se debe olvidar que la computadora más poderosa del mundo no sería sino un pisapapeles muy costoso si no fuera por la presencia de una serie de instrucciones programadas cuidadosamente para que todo el equipo funcione de una determinada forma, permitiendo que una misma computadora trabaje como procesador de texto, hoja de cálculo, base de datos, diseño de presentaciones, retoque fotográfico, dibujo artístico o industrial, reproductor de música o de películas, centro de entretenimiento multimedia, etc.

El hardware y el software de una computadora son un binomio inseparable, ya que uno no es nada sin el otro. Esta es una de las características que hacen especiales a los circuitos de proceso digital de información, que se puede tener un dispositivo de uso “general”, y que dependiendo de la programación que se le aplique, servirá para diversos propósitos.

Esta flexibilidad es la que ha permitido que los microprocesadores y microcontroladores invadan todos los rincones del hogar y la oficina, y cada día esto es más evidente, cuando se encuentran dispositivos como cafeteras, ventiladores, acondicionadores de aire, refrigeradores, etc., que han sustituido los tradicionales controles electromecánicos por circuitos digitales, más baratos, eficientes, confiables y seguros.

En resumen, los microprocesadores y microcontroladores no se encuentran únicamente en computadoras personales, sino que gradualmente han invadido una enorme cantidad de dispositivos que nos rodean; esto significa que en la actualidad, si se desea entrar totalmente en el mundo de la electrónica o el control, se deben conocer los fundamentos de estos dispositivos, cómo se constituyen en su interior, cómo trabajan e incluso saber aplicar algunos de ellos en el trabajo diario. En los siguientes subtemas, precisamente, se describirá la estructura básica de un microprocesador y un microcontrolador, sus partes principales y cómo interactúan, así como los fundamentos de programación interna, que permiten que este elemento se pueda utilizar en multitud de aplicaciones.

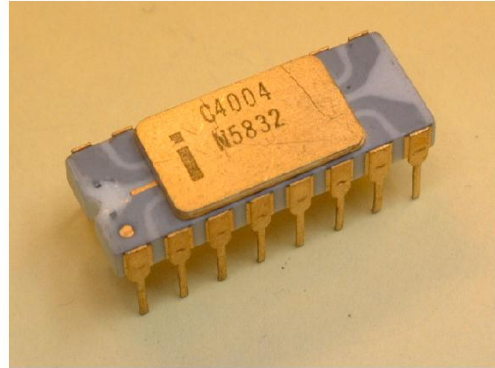
ACTIVIDAD DE APRENDIZAJE

En la actualidad, dos empresas son las principales proveedoras de microprocesadores para computadoras personales; averigua el nombre de ambas empresas y de sus dispositivos más representativos; y si tienes una computadora en casa, investiga qué tipo de microprocesador utiliza, cuánta memoria RAM posee, la capacidad del disco duro, y todos los datos que puedas sobre tu equipo. Investiga también qué versión de sistema operativo utiliza, y cuáles son los programas más usuales que trae instalados; comprobarás así que una PC es un conjunto de componentes diversos trabajando al unísono, ejecutando un software que determina qué se va a hacer en cada momento.

1.2 ELEMENTOS BÁSICOS DE UN MICROPROCESADOR

Resulta evidente la enorme importancia que tienen los dispositivos digitales de cálculo en la sociedad moderna; así, no es una exageración mencionar que el mundo sería muy distinto si estos componentes no hubieran sido desarrollados. De hecho, la historia de los microprocesadores es una

larga sucesión de acontecimientos afortunados, iniciados por dos grandes empresas a principios de la década de los setenta del siglo pasado: Intel y Busicom, la primera fabricante de circuitos integrados, y la segunda de calculadoras electrónicas. Entre ambas empresas, desarrollaron el primer microprocesador de la historia: el Intel 4004, un dispositivo de cuatro bits que servía como “cerebro” de toda una línea de calculadoras de escritorio. Seguramente, quienes diseñaron tan modesto dispositivo, el cual apenas contaba en su interior con poco más de dos mil transistores, nunca imaginaron que en unas cuantas décadas se tendrían microprocesadores corriendo a miles de millones de ciclos por segundo, y que en su interior contendrían cientos de millones de transistores trabajando al unísono.



Intel 4004, el primer microprocesador del mundo.

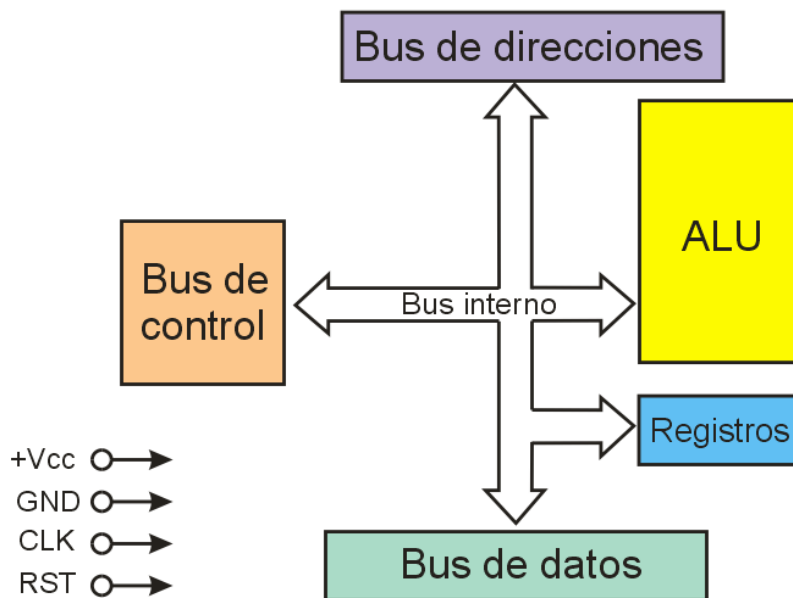


Aspecto típico de un microprocesador común: el Z-80 de Zilog.

Lo que hace tan especiales a los microprocesadores es precisamente su capacidad de manejar en diversas formas una serie de datos binarios, realizando una gran cantidad de operaciones lógicas basadas en un programa

preestablecido, y buscando con ello obtener un resultado final satisfactorio para el usuario.

Si observamos el exterior de un microprocesador típico, no tiene diferencia con otros circuitos integrados comunes; lo que lo hace especial es su estructura interna, que está diseñada para proporcionar al dispositivo una flexibilidad que no tienen otro tipo de componentes, la cual permite aplicar exactamente el mismo microprocesador en una computadora pequeña, en una consola de videojuegos, en el control de un automóvil, en el panel de un proceso industrial, etc.



Aspecto interno de un microprocesador básico.

En la figura, se observa el diagrama interno típico de un microprocesador; en el extremo derecho se ubica una etapa identificada como ALU, siglas de *Aritmetic-Logic Unit* o Unidad Aritmética-Lógica, este es el bloque en donde se llevan a cabo las operaciones dentro del microprocesador, y es el que determina la potencia de cálculo de este dispositivo. La ALU está conectada al resto del dispositivo por medio de un bus de comunicación interno, que lleva y trae señales de los otros bloques desde y hacia a la ALU; por

ejemplo, para alimentar los datos con los cuales trabajará la ALU, existen una serie de registros, que son pequeños bloques de memoria interna, donde se almacenan temporalmente los bits y bytes que usará la ALU para sus cálculos, y también ahí se colocan los resultados parciales antes de enviarse a la etapa adecuada. Generalmente, en estos registros también se guarda información adicional que agiliza la realización de los cálculos que lleva a cabo el microprocesador, como banderas de estado, bits que indican que una operación ha rebasado la capacidad del registro, etc.; adicionalmente, aquí se lleva el control de la lectura de memoria, de los saltos en la programación, de los puntos de retorno, etc. Estos registros están conectados al bus de comunicación interno, lo que permite cargar y leer datos de ellos con gran facilidad y rapidez.

También existe un bus de control, que sirve para expedir o introducir instrucciones al microprocesador, dependiendo de la operación que se vaya a realizar; por ejemplo, aquí se encuentran las líneas que determinan si un dato se va a leer o a grabar en la memoria RAM, también se ubican las líneas de interrupción para detener un programa que haya entrado en un ciclo interminable, etc. Como su nombre lo indica, estas líneas de instrucción ejercen diversas tareas de control desde el microprocesador hacia sus periféricos y viceversa, así que resultan fundamentales para la correcta operación del dispositivo.

El bus de datos es la puerta de entrada y salida de los bits que se procesarán dentro del microprocesador; por aquí se introducen las instrucciones de programación que le indican a cada momento qué hacer al dispositivo, también entran los datos que se usarán para realizar los cálculos deseados, y por ahí salen los resultados obtenidos para ser almacenados en memoria o expedidos por el puerto correspondiente. Se puede decir que este bus es por donde circula toda la información con la que trabaja el microprocesador, tanto la que entra como la que sale.

Finalmente, está el bus de direcciones, que sirve para que el microprocesador pueda leer su memoria externa, o para determinar

exactamente qué dispositivo externo se utilizará en un momento dado. Por ejemplo, cuando se está leyendo un programa, normalmente se inicia la operación desde cierta dirección de memoria, y luego el control interno del microprocesador va leyendo celdillas consecutivas de la memoria para reunir todas las instrucciones necesarias para ejecutar lo que desea el usuario, a menos que entre las instrucciones haya alguna que indique un “salto” a otra dirección de memoria distinta.

Para funcionar, lo anterior necesita de una fuente de poder externa, y también de una señal de reloj que sirva como referencia para la ejecución de todas las operaciones internas. Esta es la estructura típica de un microprocesador sencillo, aunque con el enorme desarrollo que han tenido últimamente estos dispositivos, en los microprocesadores más novedosos se han incrementado considerablemente el número y variedad de bloques internos. Sólo como referencia, en la figura anexa se muestra el diagrama a bloques de un microprocesador ARM-Cortex moderno, que se utiliza en teléfonos inteligentes o en computadoras tipo tablilla.

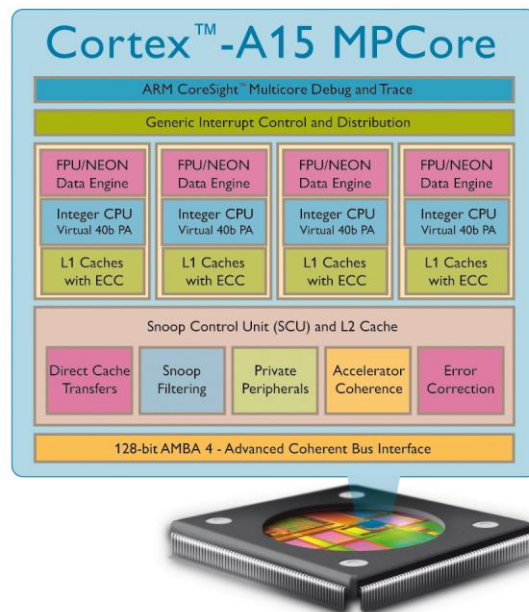


Diagrama a bloques de un microprocesador moderno.

(Imagen cortesía de ARM)

Al observar la estructura de un microprocesador, se puede distinguir una



Un microprocesador no puede hacer nada por sí solo, necesita de circuitos externos auxiliares.

de sus grandes fortalezas, pero también una gran debilidad: el microprocesador por sí mismo no sirve absolutamente para nada, ya que no posee los elementos para realizar ninguna tarea. Para

convertirse en un dispositivo completamente funcional,

requiere de una serie de elementos externos, los cuales pueden ser elegidos cuidadosamente para adaptarse a la aplicación específica que se esté buscando. Así, si el microprocesador se usara para controlar un proceso industrial, puede conectarse a sensores y actuadores que permitan el monitoreo continuo y el control de los distintos pasos del proceso; si se usa para una consola de videojuegos se rodea de los elementos necesarios para leer el juego que se desee ejecutar, recibir las órdenes del usuario, y expedir la señal de audio y video resultante hacia el televisor; y así sucesivamente. Esto significa que, a pesar de que por sí mismo el microprocesador es prácticamente inútil, cuando se rodea de los periféricos adecuados se convierte en una poderosa herramienta, completamente flexible y que puede aplicarse en múltiples situaciones muy distintas entre sí.

Sin embargo, existen muchas aplicaciones en las que los requerimientos de entrada y salida de datos están perfectamente identificadas, y que no se requiere de flexibilidad, sino de economía (económica y de espacio).

Por ejemplo, el microprocesador que controla el funcionamiento de un horno de microondas, lo único que tiene como entrada son las órdenes que el usuario le indica a través del teclado, y su única salida es el control del

calentamiento dentro del horno. En estos casos, usar un microprocesador convencional, con todos los periféricos necesarios para que pueda funcionar, resultaría completamente inadecuado. Precisamente, para cubrir este tipo de necesidades más limitadas se han desarrollado nuevos dispositivos denominados “microcontroladores”, que se describen a continuación.

Familias de microprocesadores

En la actualidad, las principales familias de microprocesadores son:

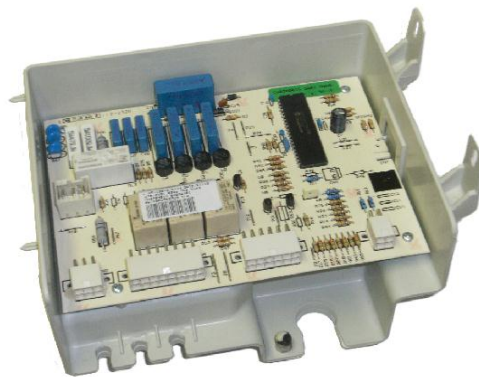
- *Microprocesadores tipo x86 y derivados*, se utilizan en prácticamente todas las computadoras personales.
- *Microprocesadores tipo ARM*, se usan ampliamente en teléfonos celulares inteligentes, computadoras tipo tablilla, sistemas de control complejos, etc.
- *Microprocesadores tipo PowerPC*, se emplean principalmente en consolas de videojuegos, pero también en sistemas de control.
- *Microprocesadores tipo 68xxx*, usados por varias computadoras tradicionales, aunque recientemente están en desuso.
- *Microprocesadores tipo SPARC*, exclusivos de las computadoras empresariales SUN, muy usados en aplicaciones de redes realmente grandes.
- *Microprocesadores tipo MIPS*, para aplicaciones gráficas y de computación realmente grandes, aunque últimamente su desarrollo se ha reducido.
- *Arquitectura i960*, exclusiva de Intel, se usa principalmente en grandes sistemas de cómputo y en sistemas de control.
- *Familia Z80*, microprocesadores muy sencillos de programar, que se siguen usando en la actualidad.

ACTIVIDAD DE APRENDIZAJE

Haz una lista de los dispositivos dentro de tu hogar que poseen procesadores lógicos en su interior. Recuerda que estos procesadores pueden ser tan sencillos como el contador de tiempo dentro de los relojes digitales, o tan complejos como el procesador central de una computadora personal, pasando por todos los circuitos que controlan el funcionamiento de los aparatos electrónicos de consumo, como televisores, reproductores de DVD, equipos de sonido, reproductores portátiles de música, hornos de microondas, etc.; incluso, los teléfonos celulares y controles remotos poseen circuitos lógicos de control en su interior, así que seguramente tu lista será mucho más larga de lo que te habías imaginado.

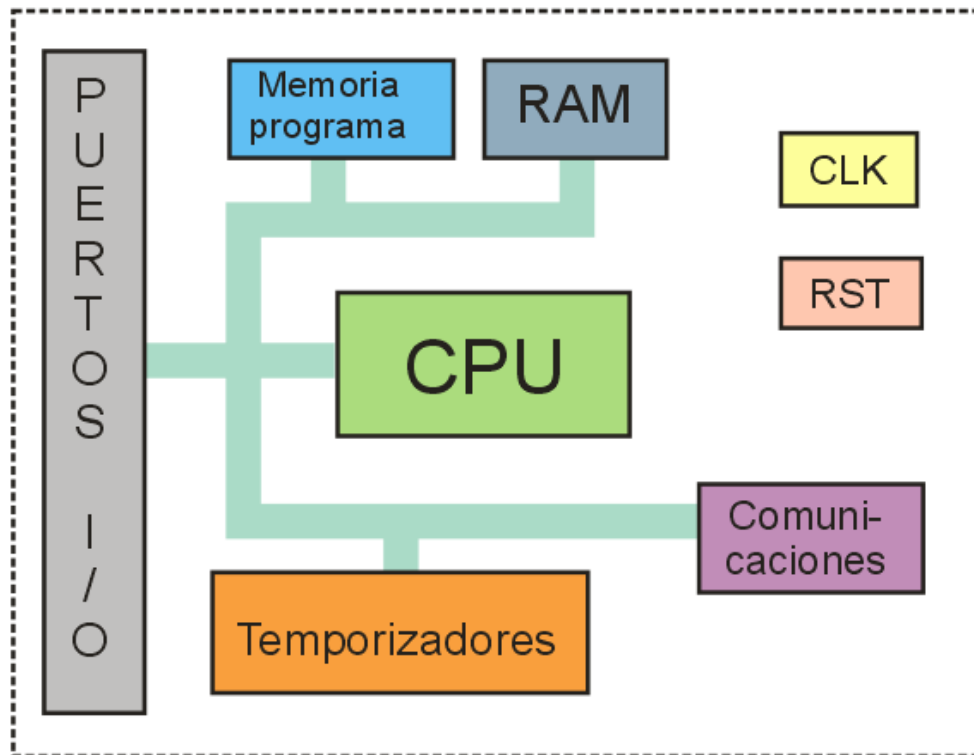
1.3 ELEMENTOS BÁSICOS DE UN MICROCONTROLADOR

Como ya se mencionó, los microprocesadores han invadido casi todos los aspectos de la vida diaria; se encuentran en los televisores, en los equipos de audio, en los teléfonos celulares, en los relojes de pulsera, en el control remoto, en las calculadoras de bolsillo, en fin, prácticamente en cualquier aparato electrónico en el hogar o la oficina; sin embargo, estos aparatos tienen necesidades tan específicas que resultaría un verdadero desperdicio colocar un microprocesador convencional, con todos sus periféricos asociados, para realizar siempre las mismas tareas una y otra vez; por ejemplo, en un control remoto, el microprocesador dentro de este dispositivo lo único que tiene que hacer es monitorear su teclado, y cuando detecta que una tecla se ha



Control de un refrigerador digital.
Observa el microcontrolador en la esquina superior derecha.

presionado, identifica cuál orden está indicando el usuario, de este modo, busca en su memoria el código correspondiente a esta orden, y expide los pulsos correspondientes para que el LED infrarrojo se encienda en cierta secuencia para enviar el comando hacia el aparato. Todo esto puede hacerlo fácilmente un microprocesador, pero rodearlo de periféricos diversos para que pueda funcionar haría que los controles remotos fueran grandes y estorbosos, además de poco eficientes.



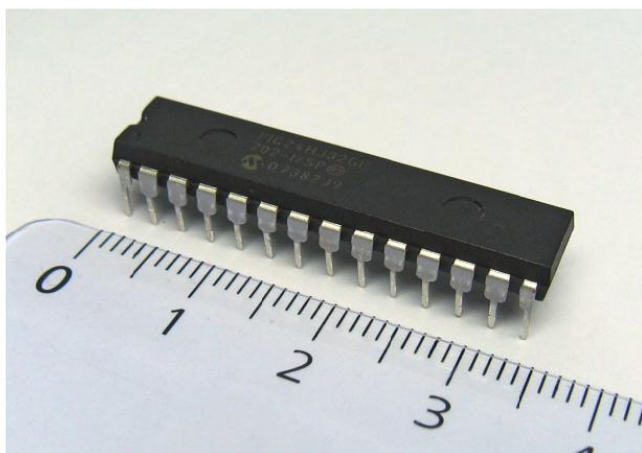
Bloques internos de un microcontrolador típico.

El número y variedad de bloques cambia según

Para evitar este desperdicio de recursos, se han diseñado los microcontroladores, que básicamente son microprocesadores completos rodeados de ciertos bloques periféricos básicos, de modo que en un chip único se puedan realizar muchas de las aplicaciones que normalmente requerirían de un microprocesador, y por lo menos de dos o tres chips adicionales más. En la

figura anexa, se muestra el diagrama a bloques de un microcontrolador típico, en el centro se ubica un microprocesador completo como el que se presentó en el apartado anterior, pero en el mismo encapsulado también se encuentra un bloque de memoria de programación, donde se puede almacenar el programa que se desee ejecutar en el micro; un bloque de memoria RAM para guardar datos temporales; una serie de puertos I/O (entrada-salida) para conectar directamente señales externas al microcontrolador, y para que éste pueda comunicarse con los elementos a su alrededor; un generador de reloj interno, una serie de temporizadores, bloques de comunicación, etc.

Todo esto se encuentra dentro de un chip sencillo, de modo que en el caso del control remoto que se expuso como ejemplo, se puede hacer que las teclas lleguen hacia los puertos de entrada del microcontrolador, y éste por medio de su programa interno reconozca la tecla que fue presionada, y consulte su memoria para extraer el código correspondiente a la orden, y luego a través de uno de sus puertos de salida envíe los pulsos necesarios para que se encienda el LED infrarrojo, y así transmitir los deseos del usuario hacia el aparato en cuestión. Resulta evidente la conveniencia que representa hacer todo esto con un chip único, en lugar de usar un micro tradicional rodeado de todos los periféricos necesarios; y es por esto que se pueden encontrar

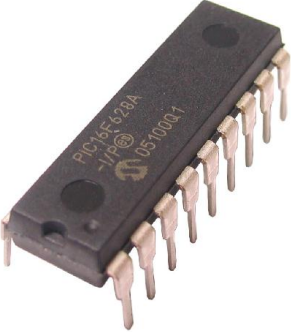


Aspecto de un microcontrolador comercial típico, de la familia PIC de Microchip.

microcontroladores en prácticamente todos los aparatos electrónicos que posean controles digitales (televisores, reproductores de DVD, equipos de sonido, hornos de microondas, teléfonos inalámbricos, ¡incluso tu despertador digital de cabecera posee un pequeño microcontrolador

en su interior! De hecho, no resulta exagerado indicar que por cada microprocesador tradicional que se fabrica en el mundo, se producen cientos o miles de microcontroladores para aplicaciones específicas, y es precisamente, por lo económicos y convenientes que resultan estos últimos, que los controles digitales han llegado a aparatos que antes se controlaban por medios electromecánicos, como lavadoras, ventiladores, cafeteras, etc.

De hecho, estos dispositivos son tan convenientes, que el chip que se tomará como base para los ejercicios de este libro, es precisamente un microcontrolador: el PIC16F628, fabricado por la compañía Microchip, el cual es fácil de conseguir y muy económico, además de que su bloque de programación interno está formado por celdas de memoria flash, esto significa que incluso si se cometen errores en la programación, o si se ha terminado con alguna práctica y se comenzará la siguiente, lo único que se debe hacer es grabarlo otra vez con el nuevo programa. Observa en el recuadro, las características básicas de este dispositivo; más adelante se detallará el significado de cada una y su utilidad en la programación.

<p><i>Características del PIC 16F628A</i></p> <ul style="list-style-type: none">• Microcontrolador de 8 bits• Frecuencia máxima: 20MHz• Terminales I/O: 16• Memoria Flash interna: 2048 bytes• RAM: 224 bytes• EEPROM: 128 bytes• USART: 1• Salida PWM: 1• Timers: 2 (8 bits)/1 (16 bits)• Comparadores: 2	 <p>PIC 16F628A, dispositivo que usaremos como base para las prácticas de programación</p>
---	--

Entonces, y resumiendo los dos puntos anteriores, un microprocesador posee en su interior estrictamente los elementos necesarios para el proceso de información, por lo que requiere de una serie de bloques funcionales externos para alimentarlo de su programa, guardar datos temporales, comunicarse con el exterior, etc. Esta estructura modular le da gran potencia y flexibilidad a un microprocesador, aunque encarece el diseño general. Por otra parte, un

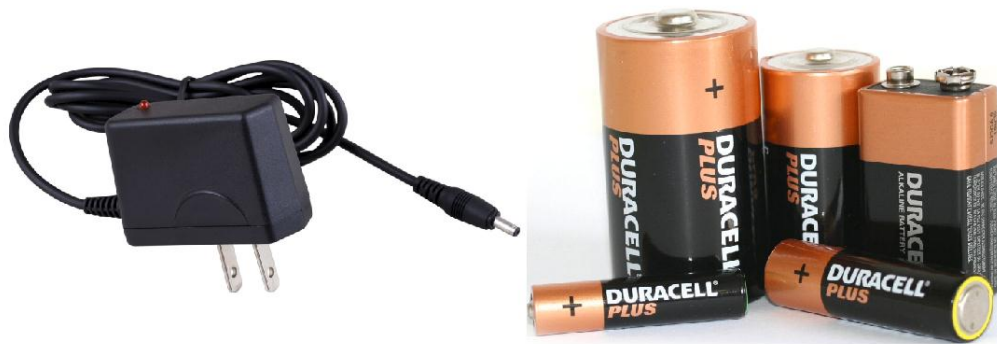
microcontrolador contiene prácticamente todos los elementos necesarios para utilizarse en aplicaciones específicas, por lo que casi no necesita de componentes externos de apoyo. Esto reduce flexibilidad y poderío, pero se gana en economía y facilidad en el diseño del circuito en general.

A continuación, se desglosarán los bloques dentro de un microprocesador, para apreciar la importancia de cada uno, y así obtener la información necesaria para comenzar con las primeras prácticas de programación de estos dispositivos.

1.4 ALIMENTACIÓN

Como cualquier circuito electrónico, un microprocesador necesita de una fuente de alimentación que le proporcione la energía eléctrica necesaria para su correcto funcionamiento. Esta fuente entrega al dispositivo el voltaje y la corriente adecuados para que pueda realizar sus tareas.

La fuente de alimentación puede ser ya sea un bloque que tome la electricidad de la línea de corriente alterna (CA) hogareña, y la transforme en el voltaje adecuado para el circuito, o una simple pila eléctrica (o varias).



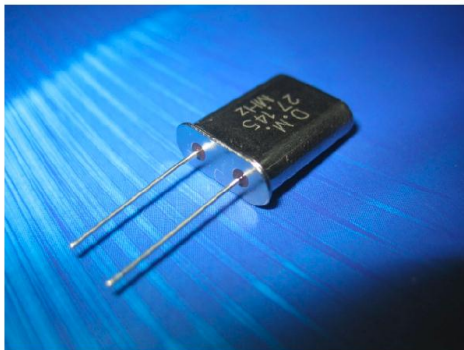
Aunque lo normal es que un microprocesador se alimente por medio de una fuente conectada a la línea de CA, cada vez son más comunes las aplicaciones que usan pilas como fuente de poder.

De manera tradicional, los microprocesadores se han alimentado con un voltaje de 5 voltios de corriente directa (5Vdc), así que lo normal es contar con

una fuente capaz de entregar esta tensión; sin embargo, existen múltiples aplicaciones en las que el microprocesador se encuentra en dispositivos portátiles, lo que implica que debe ser capaz de alimentarse con baterías, esto significa que hay muchos microprocesadores capaces de trabajar a 3V o incluso a 1.5V, lo cual a su vez trae algunas ventajas; por ejemplo, un circuito puede trabajar más rápidamente mientras mayor sea su voltaje de alimentación, pero también consume más potencia eléctrica; y al contrario, al reducir el voltaje, un circuito necesita de mucho menos corriente para funcionar, lo que resulta especialmente útil cuando la fuente de alimentación son una o dos pilas. Los microcontroladores modernos de bajo consumo pueden incluso entrar en un “modo de espera” en el cual gastan apenas unos cuantos microamperes de corriente para mantenerse siempre listos para recibir instrucciones, “despertando” cuando éstas se indican, cumpliéndolas sin el menor problema, y regresando al modo de espera automáticamente cuando han concluido su labor.

Al diseñar un circuito que requiera el uso de un microprocesador o microcontrolador, se debe tener en cuenta la fuente de alimentación asociada; sin ella el circuito no trabajará.

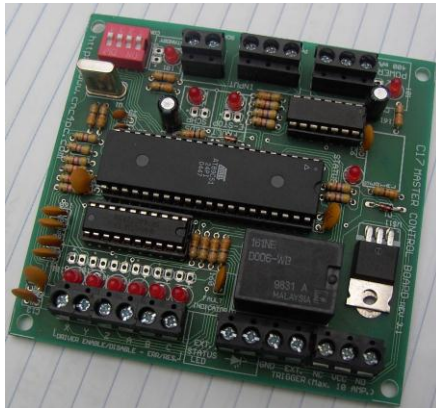
1.5 SEÑAL DE RELOJ



Aspecto típico de un cristal oscilador de cuarzo.

Los circuitos de proceso digital requieren de una señal que les indique la velocidad en la que se ejecutarán sus cálculos internos, esta señal también sirve como sincronía entre los diversos bloques periféricos necesarios para cierta aplicación. Esta señal se conoce como “reloj” o por su abreviatura en inglés: “CLK”.

Normalmente, esta señal se obtiene a través de un cristal oscilador conectado



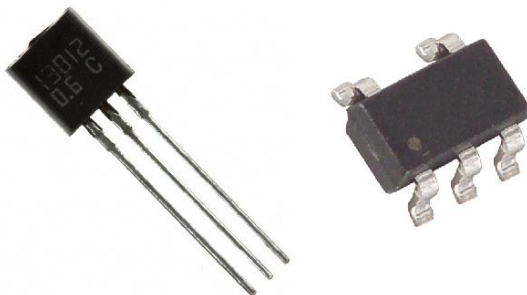
Todo circuito de proceso lógico requiere de un reloj. Observa el cristal arriba a la izquierda.
(Imagen cortesía Atmel).

directamente a una o dos de las terminales del microprocesador, aunque cada vez es más común que se pueda colocar una simple configuración *resistencia-capacitor* para generar esta oscilación, o incluso muchos microcontroladores modernos ya poseen bloques de reloj internos, lo que evita tener que utilizar componentes adicionales para que el dispositivo trabaje.

Esta señal de reloj se debe fijar dependiendo la aplicación específica del micro, pero nunca debe exceder la frecuencia máxima de operación del dispositivo, ya que de lo contrario éste no la reconocerá como tal. Esto significa que si se requiere de una aplicación donde el micro trabaje, por ejemplo, a 30MHz, será necesario conseguir un dispositivo capaz de funcionar a esa velocidad; si se coloca un reloj de 30MHz en un dispositivo diseñado para trabajar máximo a 20MHz, el chip no responderá.

1.6 RESET

Cuando se enciende un aparato que posea en su interior un microprocesador, existe un momento justo después del encendido en que el voltaje de la fuente aún no se estabiliza por completo, y por lo tanto, puede ocasionar que el



dispositivo comience a generar resultados extraños. Para evitar que el microprocesador comience a funcionar cuando su voltaje de alimentación aún no se ha estabilizado,

dispositivo de ocho bits tendrá un bus de datos de ocho líneas.

El bus de datos debe lograr funcionar en un momento dado como entrada o como salida de información, además de que en ocasiones se conectan a él circuitos que intercambian información entre ellos, sin que ésta deba afectar al micro en cuestión. Por esta razón, las terminales correspondientes poseen una configuración muy especial denominada “tri-state” o “3-



Las terminales tipo 3-state pueden servir como entrada/salida de datos conectadas a un bus común.

state”, lo que significa que estos pines pueden estar en un voltaje alto (H), uno bajo (L), o en un estado de alta impedancia (Z), en el cual los bits que circulen por el bus externo no afectarán los procesos internos del micro. Es necesario considerar esto porque cuando se desee enviar o recibir información a través del bus de datos, se deberá desactivar la señal que coloca las terminales en modo Z, y así permitir el flujo de datos desde o hacia el bus externo.

1.8 BUS DE DIRECCIONES

Cualquier microprocesador requiere de una memoria externa para guardar en ella las instrucciones básicas del programa que debe ejecutar, así como los datos iniciales con los que comenzará su proceso de cálculo; para poder acceder al contenido de esa memoria, es necesario contar con un bus de direcciones, donde precisamente el micro puede indicar a la memoria: “necesito el dato escrito en la celdilla XX”, y de este modo, el chip de memoria busca en su almacén interno y suministra al bus de datos la información deseada.

Cuando un microprocesador comienza a funcionar, de manera predeterminada ejecuta la orden contenida en la dirección 0000h de la memoria, luego se sigue con la 0001, 0002 y así sucesivamente, a menos que dentro del programa exista un “salto” que lo obligue a brincar varias de estas



direcciones y dirigirse a un punto específico de la memoria. Para tener un control estricto de la lectura de la memoria, es necesario que dentro del micro exista un bloque especializado exclusivamente en esa tarea, que es el que se detallará enseguida.

Para localizar datos e instrucciones en su memoria, el micro debe indicar la dirección exacta que desea leer, a través del bus de direcciones.

El bus de direcciones también sirve para determinar a cuál de los periféricos conectados al bus de datos irá cierta dirección, o para avisarle al micro de cuál periférico proviene cierta instrucción; así que no sólo tiene la función de controlar la memoria, sino que también es indispensable para la comunicación externa del micro con sus periféricos.

1.9 MAPA DE MEMORIA

Durante la ejecución de cualquier programa, los saltos de una dirección a otra son muy comunes, ya que se usan por ejemplo, para llamar a sub-rutinas, para ejecutar diversas acciones según lo requieran las condiciones de operación del circuito, etc. Esto implica que dentro del

1.9 MAPA DE MEMORIA

Durante la ejecución de cualquier programa, los saltos de una dirección a otra son muy comunes, ya que se usan por ejemplo, para llamar a sub-rutinas, para ejecutar diversas acciones según lo requieran las condiciones de operación del circuito, etc. Esto implica que dentro del



Cuando se programan las instrucciones de un micro, se debe aprovechar su memoria, de la forma más eficiente.

(Imagen cortesía de Atmel)

micro debe existir un control de memoria muy preciso, capaz de ir leyendo los datos e instrucciones de forma secuencial, pero también de llevar un registro exacto de los saltos realizados, y de los puntos de retorno asignados; y ese es precisamente el objetivo del control de memoria, donde se almacena un verdadero “mapa” de ésta, definiendo cuáles bloques se leerán en forma secuencial, dónde se establecen los saltos o los puntos de retorno, etc.

También es obligación del usuario, mientras diseña el programa interno del micro, llevar un mapa muy cuidadoso de las direcciones de memoria ya utilizadas, de la ubicación de las sub-rutinas, de los saltos condicionales y hacia dónde conducen, etc., ya que de lo contrario, puede suceder que al momento de ejecutar su programa, se origine un conflicto de direcciones, y en cierta localidad de memoria donde debía estar guardado cierto dato, en realidad se encuentre algo completamente distinto. Esto es especialmente importante en los microcontroladores, que generalmente tienen una cantidad de memoria bastante limitada, lo que implica que se debe aprovechar al máximo esta memoria.

Adicionalmente, tanto en microprocesadores como en microcontroladores existen localidades de memoria que tienen asignadas funciones específicas, así que el usuario no puede utilizarlas libremente, ya que en dichos espacios se guarda información específica necesaria para el buen funcionamiento del dispositivo. Esto también se debe considerar al momento de planear el mapa de utilización de memoria en cualquiera de estos dispositivos.

Esto quedará más claro en unidades posteriores, cuando se realicen ejemplos de programación en dispositivos reales.

ACTIVIDAD DE APRENDIZAJE

Los microprocesadores modernos utilizan casi siempre tres tipos de memoria: flash, RAM y EEPROM. Investiga qué significa cada una de ellas y su utilidad al programar un dispositivo lógico.

1.10 INTERRUPCIONES

Otro punto importante a considerar es qué hacer si el dispositivo entra en un ciclo interminable, pero el usuario no desea hacer algo tan drástico como apagar el micro o darle un reset, sino que desea que continúe haciendo su trabajo normal. Para esos casos, los microprocesadores cuentan con una o más señales de interrupción (INT), que como su nombre lo indica, cuando se aplican al dispositivo éste detiene lo que esté haciendo en ese momento y se dirige a una dirección de memoria preestablecida, donde deberán estar programadas las instrucciones adecuadas para que el micro recobre su estado de control normal.

Casi siempre hay dos tipos de señal de interrupción: las interrupciones “mascarables” y las “no-mascarables”. Las primeras pueden ser ignoradas por el micro en un momento dado, si así está determinado en su programación, mientras que las segundas no pueden ser ignoradas por el micro, e implican la interrupción inmediata de cualquier proceso que esté realizando, y el salto hacia la dirección de memoria predeterminada.

A continuación, se expone un ejemplo para ilustrar el uso de estas interrupciones.



Las interrupciones permiten a un micro dejar a un lado lo que esté haciendo y atender algún problema urgente.

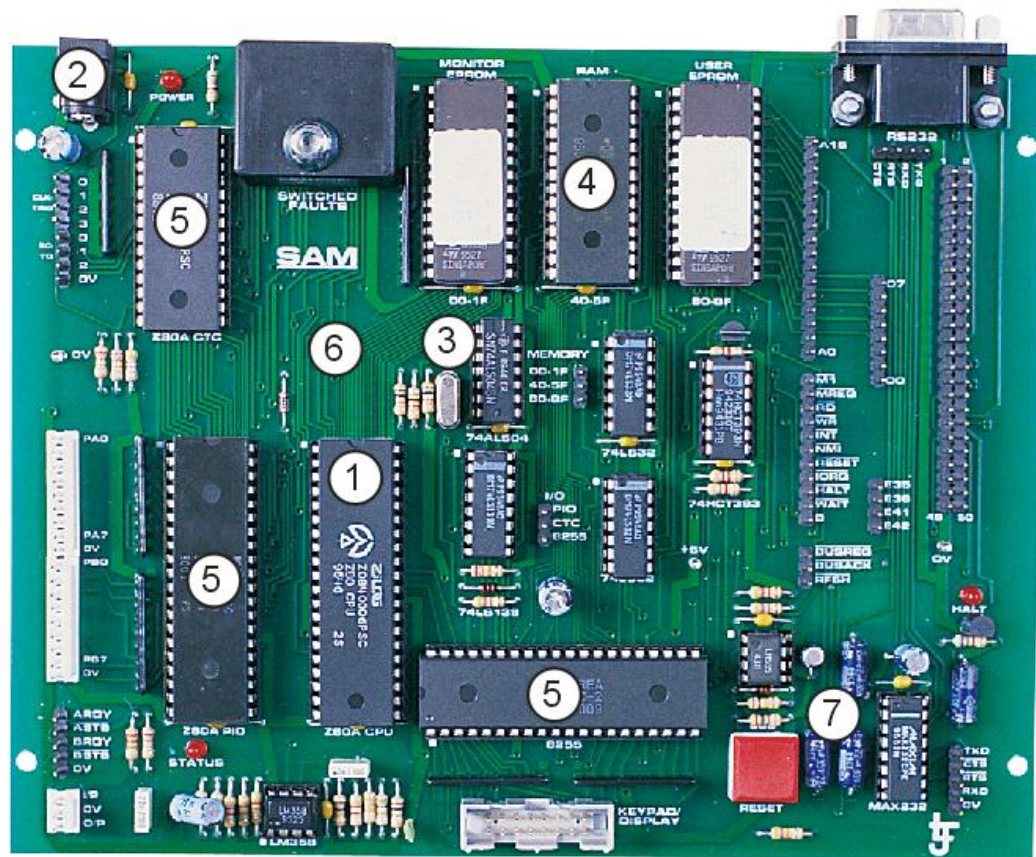
(Imagen cortesía de Motorola)

Por ejemplo, se usa un microprocesador para monitorear un proceso industrial de horneado de pan; por tanto, el micro debe ocuparse de controlar la temperatura del horno, la velocidad de la banda transportadora, la activación de los ventiladores de circulación de aire caliente, la cuenta de las piezas horneadas, etc., sin embargo, de todos estos factores, el más importante es la temperatura del horno, ya que si es

muy alta el pan saldrá quemado, y si es muy baja saldrá sin cocer. Imagina lo que sucedería, si el micro tarda algunos minutos en llevar a cabo el control de la banda, de los ventiladores, de la cuenta, etc., y durante ese tiempo no puede estar pendiente de la temperatura del horno por estar ocupado en las otras tareas, y además, la temperatura del horno tiene un cambio brusco justo al comenzar el proceso de verificación anterior. Si no hubiera algún modo de interrumpir al micro, la temperatura del horno (que es el factor más importante), podría estar fuera de especificaciones por varios minutos, antes de que el micro regrese a verificarlo; sin embargo, si en la rutina de control de los otros aspectos se implementa una interrupción conectada al termómetro del horno, cuando éste note que el parámetro sale de especificaciones, de inmediato envía la señal INT al micro, éste interrumpe de inmediato todo lo que está haciendo, y activa las medidas correctivas para que la temperatura regrese a su valor adecuado. Cuando la emergencia ha sido atendida, puede regresar a su rutina normal. Ese es el motivo principal por el cual se añadieron las señales de interrupción, así que se deben tener en cuenta al diseñar un programa, para que se activen en los momentos realmente importantes (si es necesario).

Ahora que ya se describió la estructura interna, así como los principales requerimientos de un microprocesador típico, en la siguiente unidad, se estudiarán los tipos de operaciones que se pueden realizar con un microprocesador, y cómo se pueden aprovechar para realizar tareas útiles en el hogar o la industria.

ACTIVIDAD DE APRENDIZAJE



Este es un circuito de control típico que utiliza un microprocesador. Identifica con el número correcto el componente señalado. Coloca el número que consideres adecuado entre los paréntesis.

- a) Fuente de alimentación ()
- b) Memoria ()
- c) Microprocesador ()
- d) Circuitos auxiliares I/O ()
- e) Reloj ()
- f) Reset ()
- g) Buses de comunicación ()

AUTOEVALUACIÓN

- 1.- ¿Qué es lo que da flexibilidad a los circuitos digitales de control?
- 2.- Menciona la diferencia principal entre un microprocesador y un microcontrolador:
- 3.- ¿Qué significa ALU, y para qué sirve ese bloque?
- 4.- ¿Cuáles son las dos fuentes de alimentación más comunes para microprocesadores?
- 5.- ¿Para qué sirve la señal de reloj?
- 6.- ¿Para qué sirve la señal de reset?
- 7.- ¿Cuáles son los tres buses principales que se encuentran en cualquier microprocesador?
- 8.- ¿Qué significa “terminales 3-state” y por qué es importante en microprocesadores?
- 9.- ¿Por qué es importante el mapa de memoria?
- 10.- ¿Por qué son necesarias las interrupciones en un microprocesador?

RESPUESTAS

- 1.- Que puedan usarse en distintas aplicaciones, dependiendo de su programación interna.
- 2.- Un microprocesador requiere de varios bloques auxiliares externos para trabajar, mientras que un microcontrolador contiene casi todo lo necesario para funcionar por sí mismo.
- 3.- Son siglas de *Aritmetic-Logic Unit* o Unidad aritmética-lógica, y se encarga de realizar las operaciones lógicas dentro de un microprocesador.
- 4.- Una fuente de poder conectada a la línea de alimentación hogareña, o una o más baterías.
- 5.- Para fijar la velocidad a la que se realizarán los cálculos dentro del dispositivo, así como para sincronizar la operación del microprocesador con sus bloques externos.
- 6.- Para evitar que el microprocesador comience a funcionar antes de que su voltaje de alimentación se haya estabilizado por completo; también sirve para reiniciar las operaciones del dispositivo.
- 7.- Bus de datos, bus de direcciones y bus de control.
- 8.- Son terminales que pueden tener tres estados distintos: alto, bajo o alta impedancia; y sirven para que el microprocesador se conecte a un bus general de comunicación, por el cual puedan circular datos entre otros bloques sin que esto afecte al dispositivo.
- 9.- Para llevar un control estricto de las localidades o bloques de memoria que ya han sido usados, dónde están los saltos y los puntos de retorno, la ubicación de las subrutinas, las celdas de memoria de uso exclusivo para el control de operaciones, etc. Esto permite diseñar programas más eficientes y de tamaño reducido.

10.- Para lograr detener en un momento dado lo que esté haciendo el dispositivo, para que atienda algún proceso de mayor prioridad, o para sacar al microprocesador de una situación inestable o un ciclo interminable.

UNIDAD 2

ESTRUCTURA DE UN MICROPROCESADOR

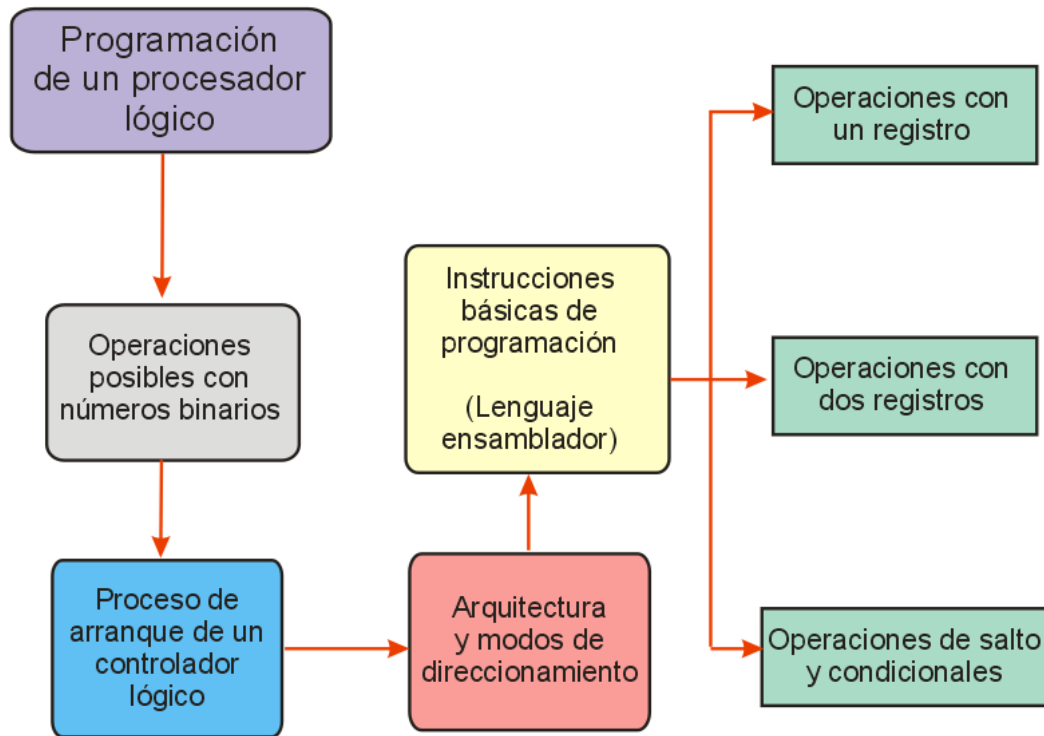
OBJETIVO

Conocer las operaciones que puede llevar a cabo un microprocesador o microcontrolador. Identificar la estructura de un microprocesador hipotético, y el proceso de encendido y búsqueda de instrucciones del mismo, así como las señales que intercambia el CPU con sus periféricos. Además, conocer los principales modos de direccionamiento dentro de un micro, así como comenzar a programar en lenguaje ensamblador.

TEMARIO

- 2.1 OPERACIONES BÁSICAS ENTRE REGISTROS
- 2.2 REGISTROS CON ELEMENTOS ARITMÉTICOS BÁSICOS
 - 2.2.1 *Operaciones con un registro sencillo*
 - 2.2.2 *Aritmética básica con registros*
 - 2.2.3 *Operaciones condicionales y de salto*
- 2.3 CONCEPTOS BÁSICOS DE INSTRUCCIÓN Y CÓDIGO DE OPERACIÓN
- 2.4 MICROPROCESADOR HIPOTÉTICO
- 2.5 OPERACIÓN DEL MICROPROCESADOR, FASES DE BÚSQUEDA Y EJECUCIÓN
- 2.6 CONCEPTO Y OPERACIÓN DEL MICROPROCESADOR
- 2.7 CONCEPTO DE MODO DE DIRECCIONAMIENTO
- 2.8 INTRODUCCIÓN A LA PROGRAMACIÓN EN ENSAMBLADOR

MAPA CONCEPTUAL



INTRODUCCIÓN

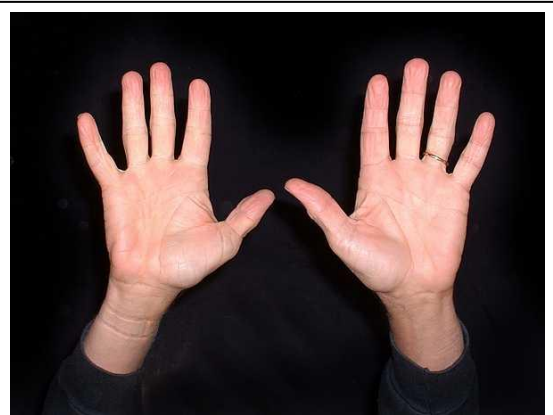
Una vez que se han conocido las diferencias entre microprocesadores y microcontroladores, ha llegado la hora de profundizar en la estructura y funcionamiento de estos dispositivos. Primero, se indicará qué tipo de cálculos se llevan a cabo dentro de estos dispositivos, y cómo pueden aprovecharse para realizar diversas funciones, desde las más sencillas hasta las más complejas. También se describirá el flujo de información dentro de un dispositivo típico; cómo se clasifican los registros, para qué sirven y cómo se pueden aprovechar las banderas dentro de un programa, etc. Para ilustrar estas explicaciones, se tomará como ejemplo el microcontrolador mencionado en la unidad anterior: el PIC 16F628A; aunque lo que se explique podrá ser fácilmente extrapolado a casi cualquier microprocesador o microcontrolador comercial moderno.

Para facilitar aún más la aplicación a otros modelos de microprocesadores, primero se mostrarán estos procesos aplicados a un dispositivo hipotético ideal, y una vez comprendido el concepto, se aterrizará al set de instrucciones del micro base, lo cual a su vez servirá como una introducción a la programación en lenguaje ensamblador. Todo esto servirá como cimiento para iniciar con las primeras prácticas de programación, que aunque en esta unidad introductoria a los microprocesadores se referirán programas sencillos y sin muchas complicaciones, en las siguientes unidades se abundará en el tema, explotando al máximo todas las características funcionales de un microprocesador o microcontrolador típico.

2.1 OPERACIONES BÁSICAS ENTRE REGISTROS

Antes de comenzar con la explicación de la estructura y funcionamiento de un microprocesador o microcontrolador, se indicarán las operaciones principales que se pueden hacer con estos dispositivos, y aclarar un concepto que en ocasiones confunde a quienes apenas comienzan en este tema: la distinción entre números decimales y números binarios, y las distintas operaciones que se pueden realizar con estos últimos. Aunque probablemente quien ya llegó a esta unidad domina el concepto de numeración binaria, no está de más dar un repaso rápido sobre el tema.

Como se sabe, las personas utilizan el sistema decimal para realizar sus operaciones diarias, desde contar las frutas y verduras, hasta los más complejos cálculos astronómicos o físicos. El sistema decimal ha demostrado su efectividad a través del tiempo, y se ha convertido en algo tan natural que parece difícil pensar que existan otras formas de realizar operaciones matemáticas; sin embargo, utilizar el sistema decimal es más fruto de la casualidad que de un razonamiento lógico, y la razón principal por la que se estableció desde el inicio de la humanidad, es



Los seres humanos usamos el sistema decimal debido a que poseemos diez dedos en ambas manos.

(Banco de imágenes propio)

debido a que tenemos diez dedos en ambas manos, así que este aspecto se utilizó como base y se creó un sistema de numeración fundamentado en potencias de diez.

Sin embargo, a mediados del siglo XX, cuando comenzaron a construirse las



En electrónica, es mucho más fácil comprender un sistema con sólo dos variables: encendido o apagado.

(Banco de imágenes propio)

primeras computadoras electrónicas, los científicos se enfrentaron a un problema: era muy difícil hacer que estas máquinas comprendieran el concepto de diez valores distintos, e incluso si trataban de implementarlo, se prestaba a confusión y a resultados falsos. En un circuito electrónico, es difícil determinar si un transistor está encendido al 10, 20 o 30%, lo cual es algo necesario si se quisiera utilizar una lógica decimal para efectuar sus cálculos; resulta mucho más sencillo simplemente tener dos estados: “encendido” o “apagado”, y esto a su vez implica que sólo se tienen dos posibilidades con las cuales trabajar, lo que simplifica muchísimo los circuitos y el tipo de operaciones que se pueden realizar con ellos. A este nuevo sistema de numeración se le llamó “binario”, y es la base sobre la cual trabaja toda la electrónica digital moderna, incluyendo los microprocesadores.

Como su nombre lo indica, la numeración binaria representa cualquier tipo de cantidad utilizando sólo dos símbolos: un “0” y un “1”. Esto podría parecer extraño a primera vista, pero una vez que se comprende la lógica detrás de la conversión, resulta fácil de entender.

Por ejemplo, al analizar los diez números naturales (del 0 al 9) que se utilizan de manera cotidiana, y cómo se pueden representar por numeración binaria; se han añadido también los números del 10 al 15 con su correspondiente representación en lenguaje hexadecimal (indicado como Ah, Bh, etc), ya que es el más utilizado al momento de programar dispositivos de 8 bits.

<i>Decimal</i>	<i>Binario</i>	<i>Decimal</i>	<i>Binario</i>
0	0	1	1
2	10	3	11
4	100	5	101
6	110	7	111
8	1000	9	1001
10 (Ah)	1010	11 (Bh)	1011
12 (Ch)	1100	13 (Dh)	1101
14 (Eh)	1110	15 (Fh)	1111

La lógica detrás de esta conversión es la siguiente: en la numeración binaria, la posición 0 representa al número 1, la posición 1 al número 2, la posición 2 al 4, la posición 3 al 8, la posición 5 al 16, y así sucesivamente; esto significa que cada vez que se añade una posición a la izquierda se está duplicando el número representado. Al observar el siguiente ejemplo de cómo se interpretan las posiciones en un número binario, y cómo convertir de binario a decimal, se puede apreciar que la posición al extremo derecho recibe el nombre de “posición 0”, la siguiente a su izquierda “posición 1” y así sucesivamente, lo que significa que en un micro de 8 bits, se tendrán registros con posiciones que van desde la 0 a la 7.

Se desea conocer cuánto representa el número 1001011101 binario en decimal.										
Posición	9	8	7	6	5	4	3	2	1	0
Valor decimal	512	256	128	64	32	16	8	4	2	1
Número binario	1	0	0	1	0	1	1	1	0	1

Precisamente, al hablar de “microprocesadores de “N” bits”, esto se refiere a la extensión de la palabra típica binaria que puede manejar este dispositivo; esto significa que un microprocesador de 8 bits puede manejar hasta 256 combinaciones distintas en sus datos; uno de 16 puede manejar más de 65,000; y uno de 32 más de 4 mil millones de combinaciones.

ACTIVIDAD DE APRENDIZAJE 2A

Convierte a binarios los siguientes números decimales, y encuentra también su equivalencia hexadecimal:

- a) 352 = _____ (binario) = _____ (hexadecimal)
- b) 129 = _____ (binario) = _____ (hexadecimal)
- c) 1017 = _____ (binario) = _____ (hexadecimal)

Encuentra el equivalente decimal de los siguientes números binarios:

- d) 10011010 = _____ (decimal)
- e) 1101110010 = _____ (decimal)
- f) 10101010 = _____ (decimal)

Encuentra el equivalente binario y decimal de los siguientes números hexadecimales:

- g) 14Ah = _____ (binario) = _____ (decimal)
- h) B7h = _____ (binario) = _____ (decimal)

2.2 REGISTROS CON ELEMENTOS ARITMÉTICOS BÁSICOS

Ahora que se recordaron los conceptos básicos de la numeración binaria, se describirán las operaciones básicas que se pueden hacer con este tipo de números, y así, más adelante, analizar lo que se puede hacer cuando se desean realizar operaciones con dos números binarios.

Como última aclaración: se utilizarán palabras de 8 bits, pero se separarán en dos grupos de cuatro, ya que así se pueden interpretar más fácilmente; también se mostrará su equivalente hexadecimal, ya que es la notación más común al programar microprocesadores de 8 bits.

2.2.1 Operaciones con un registro sencillo

Con un número binario se pueden realizar una gran cantidad de operaciones, de las cuales, bastantes están incluidas en el juego de instrucciones básicas de microprocesadores y microcontroladores; a continuación, se muestran las más representativas:

Operación: Borrar registro

Como su nombre lo indica, en esta operación lo único que se hace es llevar todas las posiciones del registro a un valor de "0".

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
0100 1110	Borrar registro	0000 0000
4Eh	Clear reg	00h

Operación: Set registro

Operación contraria a la anterior; lleva todas las posiciones a un valor de "1".

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
0100 1110	Set registro	1111 1111
4Eh	Set reg	FFh

Operación: Hacer girar un registro a la derecha

Se desplazan los valores de un registro hacia la derecha, de modo que la posición 7 pasa al lugar 6, la 6 a la 5, y así sucesivamente. Normalmente, la posición 0 se mueve hasta la 7, de modo que se hace un corrimiento circular, aunque a veces esta rotación también toma en cuenta un bit adicional llamado "carry" (el cual se describirá con más detalle un poco más adelante).

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
1001 1010	Girar registro a la derecha	0100 1101
9Ah	Rotate right reg	4Dh

Operación: Correr registro a la izquierda

Se desplazan los valores de un registro hacia la izquierda, de modo que la posición 0 pasa al lugar 1, la 1 a la 2, y así sucesivamente. Normalmente, la posición 7 se mueve hasta la 0, de modo que se hace un corrimiento circular, aunque a veces también se toma en cuenta el bit “carry”.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
1001 1010	Girar registro a la izquierda	0011 0101
9Ah	Rotate left reg	35h

Operación: Borrar un bit dentro de un registro

Se pone en “0” una posición determinada dentro del registro.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
1001 1010	Borrar bit 3	1001 0010
9Ah	Clear bit 3	92h

Operación: Set un bit dentro del registro

Se pone en “1” una posición determinada dentro del registro.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
1001 1010	Set bit 5	1011 1010
9Ah	Set bit 5	BDh

Operación: Invertir registro (complemento)

Se invierten todas las posiciones del registro; donde había un “1” se pone un “0” y viceversa. También se le llama “operación complemento”.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
1001 1010	Invertir registro	0110 0101
9Ah	Compl reg	65h

Operación: Incrementar registro

Se aumenta en “1” el valor del número almacenado en el registro.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
1001 0010	Incrementar registro	1001 0011
92h	Inc reg	93h

Operación: Decrementar registro

Se resta “1” al valor del número almacenado en el registro.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
1001 0010	Incrementar registro	1001 0001
92h	Dec reg	91h

Operación: Intercambiar nibbles

Se toman los 4 bits superiores del registro y se colocan en la parte inferior, y los inferiores se colocan en la parte superior.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
1001 1010	Intercambiar nibbles	1010 1001
9Ah	Swap nibbles	A9h

Operación: Mover un dato a un registro

Se toma el contenido de alguna posición de memoria, y se pasa hacia un registro de trabajo, para que la ALU (*Unidad Aritmética Lógica*) lo tenga disponible para operaciones posteriores.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
A=0101 1101 B=NI	Mueve A hacia B	A=0101 1101 B=0101 1101
A=5Dh B=NI Nota: NI = (no importa)	Move A to B	A=5Dh B=5Dh

Operación: Cargar un número fijo en un registro

Se coloca un número determinado dentro de un registro, para que la ALU lo tenga disponible para operaciones posteriores.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
No.=1001 1101 A=NI	Carga 9Dh en A	A=1001 1101
No.=9Dh A=NI Nota: NI = (no importa)	Move 9Dh to A	A=9Dh

Estas son las operaciones básicas que se pueden hacer con un registro; a continuación, se indica lo que se puede hacer con dos de ellos.

2.2.2 Aritmética básica con registros

Así como en numeración decimal se pueden efectuar sumas, restas y demás operaciones matemáticas, también se pueden realizar con la numeración binaria, aunque sus reglas son un algo distintas. No se profundizará en cómo se hace una suma o resta en binario, tan sólo se mencionarán las operaciones más importantes y cómo se interpretan en los registros. Se supondrá que los registros con los que se realizará la operación se identifican como [A] y [B], y el resultado se colocará en un registro especial de resultado [R].

Operación: AND entre dos registros:

Equivale a aplicar una compuerta AND a cada posición de ambos registros.

<i>No. original en los registros</i>		<i>Operación</i>	<i>Resultado</i>
Registro A	Registro B		Registro R
1001 0011	1010 1110	[A] AND [B]	1000 0010
93h	AEh		82h

Operación: OR entre dos registros:

Se aplica una compuerta OR en cada una de las posiciones de los dos registros.

<i>No. original en los registros</i>		<i>Operación</i>	<i>Resultado</i>
Registro A	Registro B		Registro R
1001 0011	1010 1110	[A] OR [B]	1011 1111

93h	A Eh		BFh
-----	------	--	-----

Operación: OR exclusiva (XOR) entre registros:

Se aplica una compuerta XOR entre cada una de las posiciones de ambos registros.

<i>No. original en los registros</i>		<i>Operación</i>	<i>Resultado</i>	
Registro A	Registro B		Registro R	
1001 0011	1010 1110	[A] XOR [B]	0011 1101	
93h	A Eh		3Dh	

Operación: Suma de dos registros. Caso 1: el resultado no excede la capacidad del registro R:

Se realiza una operación de suma binaria entre ambos registros.

<i>No. original en los registros</i>		<i>Operación</i>	<i>Resultado</i>	
Registro A	Registro B		Registro R	
0010 1011	1010 1110	[A] + [B]	1101 1001	
2Bh	A Eh		D9h	

Operación: Suma de dos registros. Caso 2: el resultado excede la capacidad del registro R:

Se realiza una operación de suma binaria entre ambos registros; como el resultado necesita más bits de los que puede dar el registro R, se activa un bit adicional (carry) para indicar esta situación.

<i>No. original en los registros</i>		<i>Operación</i>	<i>Resultado</i>	
Registro A	Registro B		Registro R	Bit Carry
1001 0011	1010 1110	[A] + [B]	0100 0001	1
93h	A Eh		41h	1

Operación: Resta entre dos registros:

Se realiza una resta binaria entre el contenido de dos registros. *Caso 1:* el registro A es mayor que el registro B.

No. original en los registros		Operación	Resultado	
Registro A	Registro B		Registro R	
1001 0100	0010 1010	[A] - [B]	0110 1010	
94h	2Ah		6Ah	

Operación: Resta entre dos registros:

Se realiza una resta binaria entre el contenido de dos registros. Caso 2: el registro A es menor que el registro B. Para indicar que el resultado es menor que cero, se activa un bit adicional (decrement carry, DC) para indicar dicha situación.

No. original en los registros		Operación	Resultado	
Registro A	Registro B		Registro R	Bit DC
1001 0011	1010 1110	[A] - [B]	1110 0101	1
93h	A Eh		E5h	1

Operación: Resta entre dos registros:

Se realiza una resta binaria entre el contenido de dos registros. Caso 3: el registro A es igual que el registro B. Para indicar que el resultado de la operación es cero, se activa un bit adicional (cero, Z) para indicar dicha situación.

No. original en los registros		Operación	Resultado	
Registro A	Registro B		Registro R	Bit Z
1001 0011	1001 0011	[A] - [B]	0000 0000	1
93h	93h		00h	1

Operación: Comparar dos registros:

Se compara el contenido binario de dos registros, y se activa el bit que determine si A es mayor que B, si A es menor que B o si A y B son iguales. Es una combinación de las tres operaciones que se acaban de mostrar. El resultado de la comparación se conoce al observar el estado de los bits DC y Z; si ambos son "0", $A > B$; para $DC=1$ y $Z=0$, $A < B$; y si $DC=0$ y $Z=1$, $A = B$.

No. original en los registros		Operación	Resultado		
Registro A	Registro B		Registro R	DC	Z
1001 0100	0010 1010	[A] COMP [B]	0110 1010	0	0
94h	2Ah	Para A > B	6Ah	0	0
1001 0011	1010 1110	[A] COMP [B]	1110 0101	1	0
93h	A Eh	Para A < B	E5h	1	0
1001 0011	1001 0011	[A] COMP [B]	0000 0000	0	1
93h	93h	Para A = B	00h	0	1

Estas son las operaciones básicas que se pueden realizar con dos registros; ya que se conocen, ahora se puede comenzar a analizar cómo se aprovechan estas operaciones en la programación de un microprocesador o microcontrolador.

ACTIVIDAD DE APRENDIZAJE 2B

Realiza las siguientes operaciones entre dos números binarios:

- a) 1010 0110 AND 1110 1101 = _____
- b) 0001 0101 OR 0101 0011 = _____
- c) 0100 0111 XOR 0110 0101 = _____

Realiza las siguientes operaciones, considerando la existencia del bit "carry":

- d) 0011 1011 + 0110 0110 = _____ C = __
- e) 1001 0110 + 0111 0101 = _____ C = __
- f) 1100 0010 + 0100 1110 = _____ C = __

Realiza las siguientes restas binarias, considera los bits DC y Z:

- g) 1001 0111 – 0101 1001 = _____ DC = __ Z = __
- h) 0111 0101 – 0111 1001 = _____ DC = __ Z = __
- i) 0110 0101 – 0110 0101 = _____ DC = __ Z = __

2.2.3 Operaciones condicionales y de salto

Existen otras operaciones básicas que cualquier microprocesador debe ser capaz de realizar, para ejecutar determinados segmentos de código en el momento en que se cumplan ciertas condiciones; es precisamente por eso que a ese tipo de instrucciones se les llama “condicionales”, y resultan fundamentales al momento de realizar el programa básico de cualquier circuito de proceso lógico de información.

Además de éstas, hay instrucciones en las que simplemente se le ordena al microprocesador que haga un “salto” en su memoria, esto es, que deje de leerla secuencialmente y en un momento dado se dirija a una posición de memoria determinada, donde probablemente exista una serie de instrucciones necesarias en ese momento, u otra serie de comandos que deben ejecutarse cuando se llega a un cierto punto del programa. A continuación, se indican cuáles son estas instrucciones condicionales y de salto, y así, tener las herramientas principales que permitirán comenzar a realizar los primeros programas de prueba.

Aquí se debe introducir un registro muy especial, llamado “contador de programa” o PC por sus siglas en inglés; en este registro, se lleva precisamente qué localidad de memoria se está leyendo en un momento dado; y en condiciones normales, cada vez que se ejecuta alguna instrucción, el PC se incrementa una unidad para leer la siguiente orden. Esta lectura secuencial sólo se interrumpe cuando existe alguna instrucción de salto, con lo cual el contenido del PC puede cambiar según la instrucción dada o según lo indiquen los comandos básicos del microprocesador empleado.

También es importante mencionar la existencia de otro registro muy especial: el *stack pointer* o almacén de direcciones, el cual se usa especialmente cuando se llaman a segmentos de código que se deben usar una y otra vez. A estos segmentos se les da el nombre de “subrutinas”, y existen instrucciones especiales para llamarlas y para regresar de ellas, y es ahí donde se aprovechan los registros stack. Si el concepto de estos dos registros no queda muy claro en este momento, más adelante, cuando se

hagan los primeros programas de prueba y se apliquen todos estos registros, su utilidad será evidente.

Operación: Decrementa un registro, y salta si es igual a cero:

Se decrementa en una unidad el número almacenado en cierto registro, pero si al hacer esto, el número llega a ser igual a cero, en ese momento se hace un salto a una posición de memoria determinada.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
1001 1100	Decrementa F, salto si = 0	1010 1011
9Ch	Dec F skip-z	9Bh

Operación: Incrementa un registro, y salta si es igual a cero:

Se incrementa en una unidad el número almacenado en cierto registro, pero si al momento de hacer esto, el número llega a ser igual a cero, en ese momento se hace un salto a una posición de memoria determinada.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro</i>
1001 1100	Incrementa F, salto si = 0	1010 1101
9Ch	Inc F skip-z	9Dh

Operación: Prueba el bit "X" en el archivo A, salta si es igual a cero:

Se revisa el estado del bit "X" en cierto archivo, y si el bit es igual a cero, se realiza un salto.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro PC</i>
1001 1100	Prueba el bit X de A, salto si = 0	Si el bit es uno, el PC sigue con su cuenta normal
9Ch	Test bit X-A, skip-z	Si el bit es cero, el PC cambia a la posición indicada en el salto

Operación: Prueba el bit "X" en el archivo A, salta si es igual a uno:

Se revisa el estado del bit "X" en cierto archivo, y si el bit es igual a uno, se realiza un salto.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el registro PC</i>
1001 1100	Prueba el bit X de A, salto si = 1	Si el bit es cero, el PC sigue con su cuenta normal
9Ch	Test bit X-A, skip-1	Si el bit es uno, el PC cambia a la posición indicada en el salto

Operación: Salto no condicional a una dirección de memoria.

Se hace un salto no condicional hacia una cierta localidad de memoria, para ejecutar algún otro segmento del programa.

<i>No. original en el registro PC (contador de programa)</i>	<i>Operación</i>	<i>No. resultante en el registro PC</i>
PC=NI	Salto a 86h	PC=86h
PC=NI	Goto 86h	PC=86h

Operación: Llama a la subrutina ubicada en la dirección X:

Se hace un salto no condicional hacia una cierta la localidad de memoria X, donde se encuentra la subrutina que se desea ejecutar en ese momento.

<i>No. original en el registro PC y stack</i>	<i>Operación</i>	<i>No. resultante en el registro PC y stack</i>
PC=24h stack=NI	Llama a la subrutina ubicada en 86h	PC=86h Stack=25h
PC=24h stack=NI	Call 86h	PC=86h Stack=25h

Operación: Regresa al programa normal desde una subrutina:

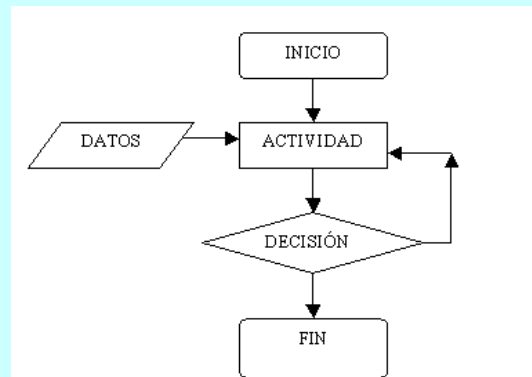
Una vez que se ha terminado de ejecutar la subrutina, y se desea regresar al programa normal, se coloca una instrucción de retorno, para que el contador de programa siga funcionando desde el punto en que se quedó antes de realizar el salto hacia la subrutina.

<i>No. original en el registro</i>	<i>Operación</i>	<i>No. resultante en el</i>
------------------------------------	------------------	-----------------------------

<i>PC y stack</i>		<i>registro PC</i>
PC=NI stack=25h	Retorno de subrutina	PC=25h
PC=NI stack=25h	Return	PC=25h

Estas son las operaciones básicas que puede realizar prácticamente cualquier microprocesador. Claro que hay dispositivos que pueden llevar a cabo más operaciones, pero casi siempre son variantes de las mencionadas.

Diagramas de flujo



Muchas veces, cuando se realiza un programa, conviene intentar representarlo mediante un diagrama de flujo, donde se especifiquen, de manera general, los procesos que se deben llevar a cabo en el transcurso de trabajo del dispositivo. Aquí se representan los saltos condicionales, el llamado a subrutinas, el uso de bloques funcionales, etc. Aunque por el momento no se profundizará en ellos, se deben tener en cuenta porque son un auxiliar invaluable al enfrentar programas de cierta complejidad.

Concepto de microprocesadores CISC y RISC



En un principio, los diseñadores de microprocesadores deseaban hacer sus dispositivos cada vez más poderosos y flexibles, por lo que los construían con un set de instrucciones interno muy avanzado, esto resultaba conveniente para quienes se enfrentaban a la programación de estos micros, ya que con una sola instrucción a veces podían hacer operaciones muy complejas; sin embargo, esta aproximación funcionaba muy bien para micros poderosos que ya contaran con lenguajes de programación de alto nivel, que “escondían” la complejidad del código interno del dispositivo bajo una interfaz de programación sencilla y fácil de usar.

Cuando los microcontroladores comienzan a utilizarse masivamente, los diseñadores se enfrentaron a un problema muy serio: dado que la memoria de programación en estos dispositivos solía ser muy limitada, eso implicaba que el programa interno tenía que realizarse usando directamente las instrucciones básicas del micro; y al tener cientos de instrucciones disponibles, la elaboración de estos programas se complicaba terriblemente.

Ante esta situación, surgió un movimiento que buscó la simplificación del set de instrucciones interno de los microprocesadores, y se creó así el concepto de CISC y RISC; CISC son las siglas en inglés de “computación con set de instrucciones complejo” y RISC son las siglas de “computación con set de instrucciones reducido”. Ambos tipos de computación son muy utilizados actualmente, y en términos generales, se puede decir que casi todos los micros de alto nivel son de tipo CISC, mientras que un buen número de los microcontroladores son de tipo RISC.

Precisamente el dispositivo que se usará como base en las explicaciones, el PIC 16F628, es un microcontrolador tipo RISC con únicamente 35 instrucciones básicas para programarlo. Esto simplifica considerablemente la elaboración de los programas internos, aunque tiene el inconveniente de que funciones que en un micro CISC podrían realizarse con una sola orden, en los RISC toma varios comandos llevarlas a cabo.

(Imágenes cortesía Intel e IBM).

2.3 CONCEPTOS BÁSICOS DE INSTRUCCIÓN Y CÓDIGOS DE OPERACIÓN

Hasta este momento, ya se indicó que para que un microprocesador pueda comenzar a trabajar, es necesario proporcionarle una serie de órdenes y datos, que detallen exactamente qué labor se desea realizar, cómo utilizar sus puertos, cómo establecer contacto con sus periféricos, etc. Sin esta serie de instrucciones, un microprocesador no sirve absolutamente para nada, pero si estos comandos están correctamente programados, el micro se convierte en una poderosa herramienta de productividad, ya sea para trabajos generales (como los microprocesadores en computadoras personales), o para aplicaciones muy específicas (como los microcontroladores incorporados en los equipos electrónicos hogareños). Además, esta serie de instrucciones permite utilizar un mismo micro para distintas aplicaciones, tan sólo cambiando la asignación de terminales, la forma de interpretar las señales de sus sensores, conectando distintos actuadores, etc. Es precisamente este programa base lo que proporciona versatilidad a los microprocesadores, y permiten aprovecharlos en muchas y muy variadas formas.

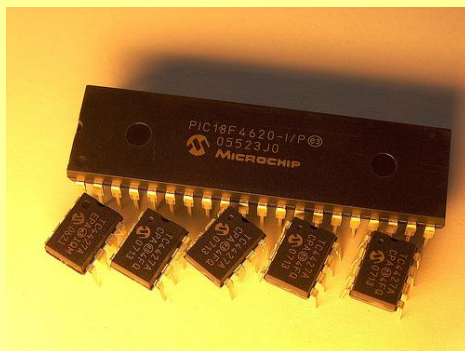
Ahora bien, este programa base debe tener cierta estructura, la cual es determinada por el microprocesador. Si por ejemplo, se programa un microcontrolador para un sistema de aire acondicionado, se supondría que como entrada se colocará un termómetro, y como salida un par de controles que enciendan ya sea el calefactor o el enfriador de aire; además, este aparato deberá tener un teclado de control para que el usuario determine la temperatura que desea, y un display donde muestre dicho valor. Con esta información básica, se puede comenzar a pensar en cómo puede ser un programa que controle el funcionamiento de este aparato; sin embargo, aquí existe un problema fundamental.

A un micro digital es imposible darle órdenes como “cuando la temperatura llegue a X valor, enciende el enfriador de aire”; al contrario, los microprocesadores poseen un lenguaje propio en el cual hay que “hablarles” para que entiendan lo que el usuario desea, y actúen según las órdenes almacenadas. Además, este lenguaje suele ser muy limitado, permitiendo que

el micro haga sólo una o dos cosas a la vez, así que se le deben dar instrucciones paso por paso, tratando de contemplar todas las situaciones posibles, y diciéndole en todo momento qué debe hacer, para que el micro pueda llevar a cabo la tarea asignada. A las órdenes válidas que se le pueden dar a un micro se les denomina “set de instrucciones”, el cual puede ser tan reducido o tan amplio como el diseñador del dispositivo lo desee.

Ahora bien, no sólo se le deben dar órdenes al micro, también se debe alimentar con datos, mismos que pueden ser fijos o variables. En el primer caso, el dato puede grabarse permanentemente en la memoria junto con el programa, pero en el segundo, se le debe asignar una posición de memoria

Juego de instrucciones básico de los microcontroladores PIC



Una de las grandes ventajas que poseen los microcontroladores de 8 bits de la familia PIC de Microchip, es que su set de instrucciones consta de sólo 35 órdenes básicas. Esto significa que si se aprende a utilizar correctamente estos comandos, se podrá programar casi cualquier microcontrolador que produzca dicha empresa, desde dispositivos de apenas 8 terminales hasta los más complejos con 40 terminales o más.

Esto contrasta con otros dispositivos que poseen juegos de instrucciones más complejos; por ejemplo, los microcontroladores AVR de Atmel que poseen un set de instrucciones de 120 comandos básicos, o como los microprocesadores poderosos que se usan en computadoras personales, que contienen un set de instrucciones que puede tener varios cientos o hasta miles de comandos. La diferencia está en que para estos micros existen lenguajes de programación muy avanzados, como el C, C++, Java, etc., mientras que los microcontroladores PIC normalmente se programan directamente en lenguaje ensamblador; más adelante se profundizará en este aspecto. (Imágenes cortesía Microchip).

RAM para que ahí se vayan guardando los datos conforme se vayan recibiendo. Los resultados del micro también pueden guardarse en la misma memoria RAM, o enviarse a alguno de los registros internos del micro, de modo que enciendan o apaguen algunas de sus terminales, para que esto se refleje en la activación o no de circuitos periféricos asociados; por ejemplo, en el caso del acondicionador de aire que se mencionó, cuando el micro se percata de que la temperatura ha subido por arriba del valor deseado por el usuario, echa a andar el enfriador para solucionar el problema; y si la temperatura cae por debajo del umbral inferior, entonces enciende el calefactor para compensar. Esto se hace encendiendo o apagando algunas de las terminales del micro, y para hacer esto, se tiene que cargar un valor determinado en ciertos registros internos del dispositivo.

Entonces, en resumen, los microprocesadores poseen un juego de instrucciones básicas con las cuales deberá estar elaborado el programa que le va diciendo a cada momento qué hacer; estas instrucciones tienen una construcción muy particular, dependiendo del microprocesador en cuestión, y pueden ser combinaciones de bits de diversa longitud, dependiendo de cómo haya sido diseñado ese dispositivo.

ACTIVIDAD DE APRENDIZAJE 2C

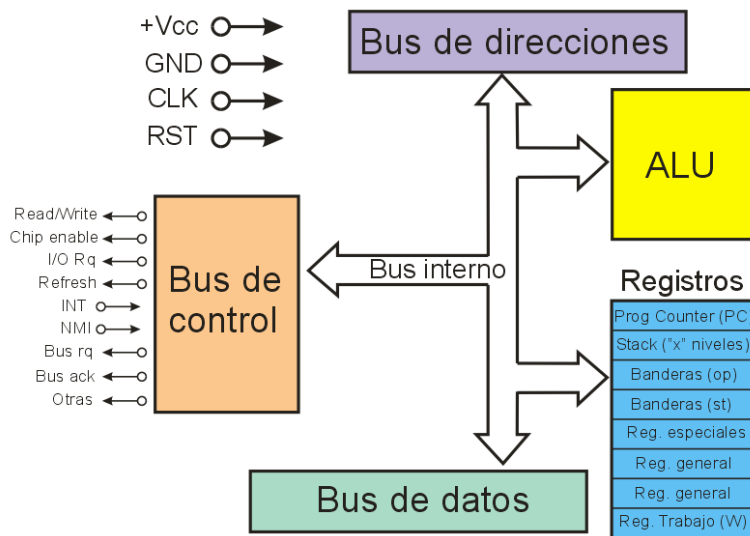
Consulta en Internet cuántos comandos individuales posee el set de instrucciones de los siguientes microcontroladores comerciales:

- Z-8 de Zilog
- MC68HC11 de Motorola
- 8051 de Intel
- PIC series 12 y 16 de Microchip
- PIC serie 18 de Microchip

También averigua si para todos ellos existen lenguajes de programación de alto nivel, como Basic o C, lo que podría simplificar mucho tu labor de diseño, si ya conoces alguno de estos lenguajes.

2.4 MICROPROCESADOR HIPOTÉTICO

Antes de comenzar las prácticas con un microcontrolador real, a continuación se indica cómo funciona un microprocesador hipotético, y esto servirá como base para comprender mejor el funcionamiento de dispositivos reales. También permitirá conocer algunos puntos importantes en el funcionamiento de estos circuitos de proceso lógico, que se aplican a cualquier marca y modelo de microprocesador o microcontrolador.



En la figura anexa se muestra un diagrama a bloques de un microprocesador hipotético; se parece mucho al mostrado en la unidad 1, pero se han detallado algunos de sus registros internos, además de las señales que maneja a través del bus de control. A continuación, se indica para qué sirven los registros señalados, y cómo funcionan durante la programación y el trabajo normal del dispositivo.

El primer registro especial que se debe considerar es el contador de programa o PC, que es donde se lleva el control de qué localidad de memoria se está leyendo en un momento determinado. Normalmente, este registro inicia con la dirección 0000h cuando se aplica alimentación al dispositivo, y de ahí comienza a contar de uno en uno, para que después de la instrucción en 0000h se ejecute la que está en 0001h, luego 0002h y así sucesivamente, a menos

que se encuentre alguna orden de dar un “salto”, con lo que el registro PC se carga con la dirección indicada en dicho comando, y a partir de ahí comienza nuevamente su cuenta de uno en uno. Entonces, el control de qué instrucción se está leyendo y ejecutando en todo momento lo lleva el registro PC.

Los registros stack son un almacén temporal de direcciones para el PC, y se usan cuando en el programa se invoca alguna subrutina. Las subrutinas son segmentos de programa que se utilizan de forma recurrente, así que para evitar tener que escribirlas una y otra vez durante la programación, se escriben una sola vez en una cierta localidad de la memoria de programa, y cada vez que se necesite, simplemente se le llama desde el programa principal, se ejecuta la subrutina, y cuando ésta concluye, el programa principal sigue funcionando como si nada. Para poder hacer esto, siempre que se llama a una subrutina, la localidad de memoria en la que estaba el PC es almacenada en el stack, el contenido del PC cambia a la localidad donde se encuentra la subrutina, ésta se ejecuta, y cuando aparece la orden de retorno, el PC recupera desde el stack la localidad en que se había quedado, pudiendo reanudar la ejecución del programa principal.

No hay sólo un registro stack, suele haber varios de ellos, para que incluso dentro de una subrutina se pueda llamar a una subrutina secundaria. Casi todos los microprocesadores poseen por lo menos cuatro posiciones de stack, lo que significa que podrían llamar hasta a cuatro subrutinas, una dentro de la otra; y en el caso específico de los microcontroladores PIC de la familia 16F, el stack es de ocho posiciones, lo que le da al programador una enorme flexibilidad al momento de diseñar su código.

El registro de banderas guarda en su interior algunos bits especiales que se usan al momento de realizar operaciones con bits y bytes; por ejemplo, aquí se encuentra el bit de “carry” (C), el de “cero” (Z), el “menos de cero” (DC), etc. Estos registros por lo general, no pueden ser modificados por el usuario, sino que lo hace la ALU dependiendo del resultado de cierta operación; y dependiendo del programa que se esté diseñando, el programador constantemente tendrá que estar consultando el estado de estos bits, para

tomar decisiones y ejecutar segmentos específicos de código, a través de los comandos de saltos condicionales.

Existen otros registros con funciones especiales, como el de estado, el de puertos, el de convertidores, etc., pero el contenido y uso de cada uno depende del microprocesador empleado y de las necesidades de programación del usuario.

Los registros generales son en donde se almacenan provisionalmente los datos con los que va a trabajar la ALU; por ejemplo, si se necesita hacer la suma de dos bytes, lo normal es cargar ambos en los registros generales y luego hacer la operación con ellos; así se evita que la ALU tenga que hacer lecturas directas a ciertas localidades de memoria, y tiene los datos siempre disponibles.

Uno de estos registros especiales suele denominarse “de trabajo” o “W”, y es donde la ALU coloca el resultado de las operaciones realizadas. Muchos dispositivos no poseen registros generales, pero el registro W es obligado prácticamente en todos (aunque no siempre recibe el nombre de W, a veces también se le conoce como “acumulador” o registro A; en micros de Atmel, se les conoce como registros X-Y-Z, y así sucesivamente).

Pero, independientemente de los nombres, lo importante aquí es recalcar la presencia de registros internos que cumplen funciones especiales dentro del micro; ya sea como control de lectura de memoria, como almacén para poder ejecutar subrutinas, como indicadores auxiliares de la ALU, como almacén para los datos con los que se está trabajando, etc. Se deben tener en cuenta, ya que resultan muy importantes al momento de diseñar el programa de un dispositivo de control lógico.

A continuación, se detallan las señales asociadas al bus de control. Observa que estas señales



Las señales del bus de control le permiten al micro comunicarse adecuadamente con sus elementos periféricos.

(Imagen cortesía Silicon Optix)

están dedicadas a controlar el funcionamiento de los circuitos periféricos conectados al microprocesador, como es una señal de lectura/escritura, que le indica a la memoria si se va a leer o a escribir un dato en alguna de sus celdillas; una señal de chip enable, que permite elegir qué periférico estará activo en un momento dado, una señal que indica si la información del bus de datos es una entrada o una salida, una señal para refrescar la memoria en caso de que se use RAM dinámica, etc. Todas estas señales se van activando o desactivando según lo requiera el programa o lo indiquen los periféricos asociados, de modo que el micro pueda estar recibiendo los datos y cifras que necesita para funcionar, y también pueda enviar las órdenes necesarias para controlar sus equipos periféricos.

Sobre los buses de direcciones y datos, no hay mucho qué decir; en el primero se van expidiendo las distintas direcciones de memoria que se están leyendo en un momento dado, o la dirección del periférico al que se desea comunicar el micro; y por el segundo, circula la información que está procesando el CPU, ya sea de entrada o de salida. Estos son los bloques básicos que debe tener un microprocesador típico, y si bien los nombres específicos de registros y/o señales cambian dependiendo de la marca y el modelo de dispositivo empleado, su función primordial sigue siendo la misma.

A continuación, se indica lo que sucede cuando apenas se enciende un microprocesador hipotético, y más adelante se detallará cómo se realiza esto en un dispositivo real.

2.5 OPERACIÓN DEL MICROPROCESADOR, FASES DE BÚSQUEDA Y EJECUCIÓN

¿Qué sucede cuando se aplica un voltaje de alimentación por primera vez a un microprocesador hipotético? En primer lugar, debido al circuito de RST que debe tener conectado en la terminal respectiva, durante algunos microsegundos el micro ya está recibiendo su voltaje de alimentación, pero no puede comenzar a funcionar hasta que la terminal RST sea activada; cuando finalmente sucede esto, el registro PC del micro está en su posición 0000h, por lo que el dispositivo comienza a buscar la primer orden de su programa en dicha

posición. Si no hubiera ningún salto directo, de ahí se pasa a la posición 0001h, 0002h, y así sucesivamente, hasta que encuentre la primera orden de salto, y se dirige hacia alguna localidad de memoria específica, esto se hace

modificando el contenido del registro PC.

En estas primeras órdenes que recibe el micro, debe estar la configuración inicial del dispositivo, que le indica cómo programar los periféricos que lo rodean; por ejemplo, tiene que indicar cuáles serán sus puertos de entrada y salida de datos, configurar adecuadamente los temporizadores asociados, indicar si se van a utilizar las interrupciones externas, etc., todo esto para que se fijen las



Cuando se aplica alimentación a un microprocesador, se echan a andar varios procesos automáticos para que comience a funcionar.

(Banco de imágenes propio)

condiciones iniciales tanto del micro como de sus circuitos asociados, y que todo quede listo para comenzar a trabajar de inmediato.

Una vez que todos los periféricos están debidamente inicializados, el programa puede comenzar a ejecutarse, y el micro puede comenzar con su labor de control numérico.

2.6 CONCEPTO Y OPERACIÓN DEL MICROPROCESADOR

Cuando el microprocesador ha comenzado a funcionar, es el momento de que inicie su labor de control; la cual puede consistir en el simple encendido/apagado de dispositivos externos siguiendo una cierta secuencia fija (por ejemplo, un semáforo, que siempre está funcionando con la misma secuencia verde-amarillo-rojo y vuelve a empezar), o si el caso lo amerita, el micro puede recibir realimentación de algún bloque externo, que sirva para

hacer funcionar otros elementos (aquí el ejemplo sería un horno de microondas, que requiere que el usuario introduzca el tiempo de operación, comprobar que los switches de la puerta están cerrados, revisar que no se han activado las protecciones térmicas, y si todo eso se cumple, en ese momento se activa el generador de microondas, el motor del plato giratorio y el ventilador de enfriamiento, para obtener el resultado final del calentamiento).

¿Cómo es capaz de hacer esto un microprocesador, usando tan sólo su set de instrucciones interno? Eso es lo que hace tan especial a estos dispositivos: sabiéndolos usar, es relativamente fácil hacer un programa que revise todos los puntos antes mencionados, y cuando se cumplan las premisas de operación, en ese momento se activan los elementos correspondientes, lo que garantiza una operación segura y confiable. En seguida, se indica cómo se podría hacer un programa básico para el horno de microondas del ejemplo anterior. Debido a que aún no se tienen las herramientas y conocimientos necesarios, se hará sin entrar en muchos detalles.

Primero, lo que se debe hacer en estos casos, es considerar las necesidades específicas del proyecto, qué entradas tiene el micro, qué salidas se requieren, qué puntos hay que verificar, etc., con esto se tendrá un panorama amplio de cómo podría ir el programa interno de dicho microcontrolador, lo que servirá como introducción precisamente al tema de programación de dispositivos de control lógico.

En la siguiente figura, se muestra un diagrama a bloques de cómo iría el circuito de control de un horno de microondas; se debe observar, que por un lado se encuentra el teclado para introducir las órdenes del usuario, además de una serie de interruptores y sensores de protección, que se aseguran que la puerta del horno esté cerrada firmemente antes de ponerse en operación, y que el aparato no tenga sobrecalentamiento. Como salida, hay una pantalla de realimentación para el usuario, los relevadores que encenderán al magnetrón (el dispositivo que genera las microondas), el motor del plato giratorio, la lámpara y el ventilador de enfriamiento. Así, se tienen los requerimientos básicos que debe contemplar el programa interno del microcontrolador.



En primer lugar, se dedica una serie de terminales del micro para el rastreo constante del teclado, y se programa internamente qué acción se debe realizar cada vez que se presiona una tecla. Por ejemplo, si se presiona algún número del panel frontal, dicho número debe aparecer en el display del horno, y almacenarse en algún registro interno para guardar el tiempo que la persona desea que el horno funcione. Una vez que se ha introducido el tiempo de operación, al activar la tecla de “inicio”, el horno debe revisar que los interruptores de protección estén cerrados, y que los sensores de sobrecalentamiento no estén activados; y sólo cuando se cumplen todas estas condiciones iniciales, entonces el micro expedirá los pulsos necesarios para encender el magnetrón, la lámpara, el motor de plato y el ventilador.

Al encenderse todo esto, el micro debe comenzar a contar “hacia atrás” el tiempo introducido por el usuario, y cuando llegue al valor de “0”, en ese momento se apagan los tres dispositivos anteriores, se genera un tono de

advertencia, y se deja el aparato en modo de espera para que el usuario retire lo que deseaba calentar y todo quede listo para comenzar de nuevo.

Con esta descripción, se pueden observar de manera general, los bloques funcionales del programa que se deben introducir a este microcontrolador:

- *Bloque inicial*, donde se fijan las condiciones operativas del micro, se configuran sus puertos I/O, se nombran variables y constantes, se activan o desactivan interrupciones, etc.
- *Bloque de arranque*, donde el micro reconoce sus periféricos asociados y los inicializa (si fuera necesario); también se hace una revisión previa de las condiciones del aparato en general.
- *Bloque de modo “en espera”*, en el cual se revisa constantemente el teclado para comprobar si hay alguna orden nueva, y se está expidiendo en el display un mensaje neutro (normalmente un “0” parpadeando), que indica que el equipo está listo para recibir nuevas órdenes.
- *Bloque de introducción de instrucciones*: se identifica cada tecla presionada, y se van almacenando los números elegidos por el usuario en localidades de memoria específicas, para que sirvan como base para el conteo regresivo durante la operación del aparato. Estos números también se expiden en el display como realimentación para el usuario.
- *Bloque de activación*: se detecta la orden de inicio, se comprueba que los interruptores de seguridad y los sensores estén en sus modos correctos, y si todo está en orden, se procede al encendido del magnetrón, el motor de plato, el ventilador y la lámpara interna.
- *Bloque de conteo regresivo*: con todo lo anterior encendido, el micro comienza un conteo regresivo segundo a segundo a partir del tiempo fijado inicialmente por el usuario, el cual debe reflejarse en el display. Cuando este contador llega a cero, entonces se apagan los cuatro elementos anteriores y se produce un tono de advertencia. Durante esta operación, en caso de que alguno de los sensores de

sobrecalentamiento se active, se genera una interrupción y todo el aparato se apaga como precaución.

- *Bloque de reinicio de operaciones*: una vez que el usuario abre la puerta para retirar la comida caliente, el micro regresa al bloque de modo en espera, y todo queda listo para reiniciar el ciclo.

Como se puede apreciar, el programa de un microcontrolador puede dividirse en una serie de “bloques funcionales” encadenados entre sí; esto simplifica considerablemente la labor del programador, ya que se puede segmentar el trabajo diseñando bloques individuales, para finalmente reunir todo y hacerlos funcionar en conjunto. Al comenzar con las prácticas de programación, convendrá seguir este método para enfrentar programas relativamente complejos.

Esto tiene otra ventaja desde el punto de vista del programador: supongamos que se deben diseñar dos o más programas distintos, pero que en todos hay un teclado y un display, aunque lo demás varía considerablemente. Si se hacen los programas “por bloques”, se podrá utilizar prácticamente el mismo bloque de control de teclado y display en todos los proyectos, y sólo cambiar aquellos aspectos específicos de cada aplicación. Esta reutilización de código es muy útil para acelerar el trabajo diario, así que siempre se debe tratar de diseñar los programas por bloques y considerar una aplicación lo más general posible, para después aterrizarla a las necesidades particulares del proyecto.

ACTIVIDAD DE APRENDIZAJE 2D



(Imagen cortesía Sony)

De forma general, imagina (como se hizo en el ejemplo anterior del horno de microondas), cuáles serían los bloques funcionales del programa necesario para el microprocesador que controla el funcionamiento de un reproductor de CD. Compáralo con el de tus compañeros para saber si consideraste todos los aspectos o alguien pensó en algo más.

2.7 CONCEPTO DE MODO DE DIRECCIONAMIENTO

Antes de entrar al tema de la programación de un microprocesador, se describirá un concepto indispensable para su correcta utilización: los modos de direccionamiento de las instrucciones. Se denomina “modo de direccionamiento” a la forma como son “llamados” los datos que se utilizarán en una operación, dependiendo de si se invoca directamente, de modo indirecto, de manera implícita, etc. Existe una amplia variedad de modos de direccionamiento de información, aunque los más utilizados por los fabricantes de microprocesadores son sólo unos cuantos; a continuación, se indican los más comunes:

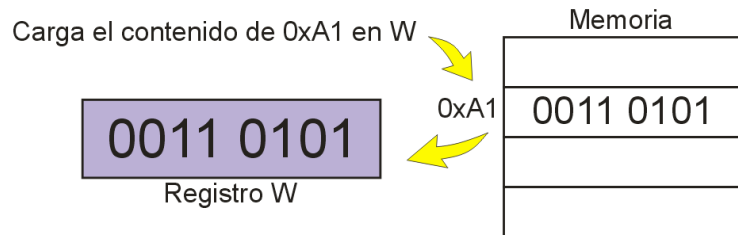
- *Direccionamiento inmediato*: en el mismo comando de la operación va el dato con el que se trabajará. Ejemplo: carga 35h en W

Carga 35h en W

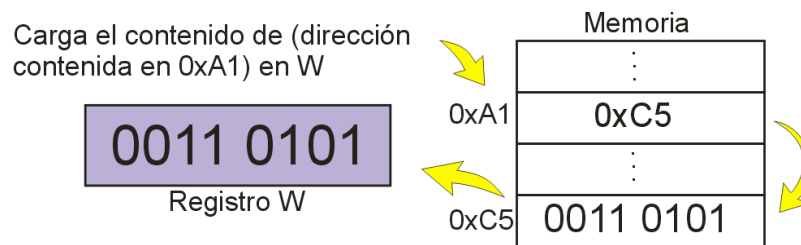
0011 0101

Registro W

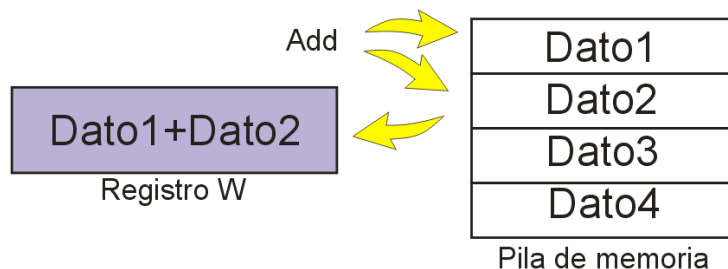
- *Direccionamiento directo:* en el comando se indica la dirección de memoria exacta donde se encuentra el dato con que se desea trabajar. Ejemplo: carga el contenido de 0xA1 en W.



- *Direccionamiento indirecto:* el comando indica una dirección de memoria, en donde se almacena una segunda dirección, que es donde está el dato necesario. Ejemplo: carga el contenido de [dirección de memoria guardada en 0xA1] en W.



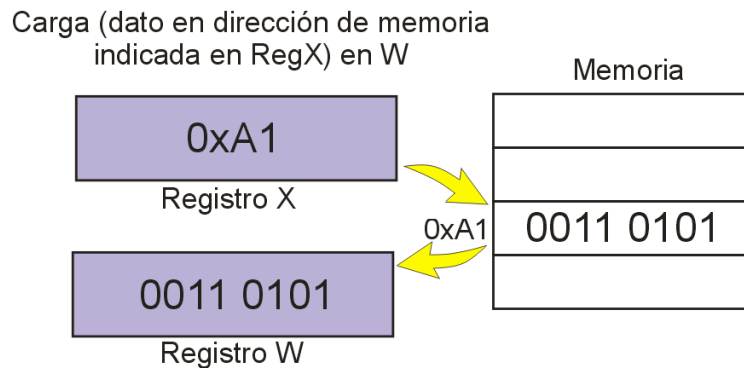
- *Direccionamiento implícito:* cuando la orden sólo indica qué se va a hacer con los operandos, y éstos se extraen automáticamente de una pila de datos. Ejemplo: teniendo una serie de operandos en la pila de datos, simplemente se da la orden “Add”, con lo que se toman los dos datos más cercanos de la pila y se suman entre sí.



- *Direccionamiento a registros:* cuando la orden invoca a un dato que ya se tiene en alguno de los registros de trabajo del micro. Ejemplo: incrementa el valor de W.



- *Direccionamiento indirecto por registro:* la orden invoca una dirección de memoria que está almacenada en uno de los registros del micro. Ejemplo: carga el dato que está en [dirección de memoria guardada en registro X] en W.



Y así sucesivamente; existen decenas de modos distintos de direccionamiento, pero muchos de ellos sólo se utilizan en los microprocesadores más complejos, y en pocas ocasiones. Como estos micros suelen tener lenguajes de programación de alto nivel (C++, Java, etc.), el usuario en realidad no se percató de estos modos, sino que escribe su programa en el lenguaje en cuestión, y el compilador se encarga de pasarlo a lenguaje de máquina, y es aquí donde aparecen dichos modos de direccionamiento. No obstante, se han hecho pruebas muy estrictas y se ha descubierto que incluso en estos casos, más del 90% de los direccionamientos son de tipo sencillo (algunos de los mencionados antes).

En el caso específico de microcontroladores PIC, prácticamente sólo existen dos o tres modos de direccionamiento, lo que simplifica la programación, aunque aumenta ligeramente el número de instrucciones requeridas.

2.8 INTRODUCCIÓN A LA PROGRAMACIÓN EN ENSAMBLADOR

Con todos estos antecedentes, ahora se describirá cómo se realiza el programa interno de un dispositivo lógico de control, lo cual conduce directamente al concepto de “lenguaje ensamblador” y a la manera de utilizarlo.

Como se ha indicado, los microprocesadores y microcontroladores poseen un set de instrucciones interno, que son las órdenes que puede identificar y ejecutar directamente el dispositivo. Cualquier orden que se intente dar y que no se encuentre en este set de instrucciones básicas, simplemente, el micro es incapaz de comprenderla, así que seguramente no hará absolutamente nada, aunque en el peor de los casos podría crear un conflicto que paralizara al sistema digital. Ahora bien, como también ya se mencionó, los microprocesadores complejos poseen varios cientos de instrucciones básicas en este set, lo que complica bastante su programación tratando de utilizar únicamente este juego de comandos.

Esto se soluciona con el desarrollo de los lenguajes de programación de alto nivel, en los cuales un usuario puede escribir su programa utilizando una serie de comandos fáciles de aprender, y una vez que el programa está hecho, se “compila” y se pasa automáticamente a lenguaje de máquina, simplificando considerablemente el diseño de programas para estas plataformas. Esto también se puede hacer porque estos micros avanzados suelen estar acompañados por grandes cantidades de espacio de almacenamiento y memoria de ejecución, así que los diseñadores de programas pueden “darse el lujo” de escribir códigos redundantes o poco optimizados, con la confianza de que contarán con el hardware necesario para su ejecución.

Sin embargo, los microcontroladores suelen estar muy limitados en ciertos recursos, siendo el principal su memoria de programación. Un microcontrolador típico posee entre 1 y 4 kilobytes de memoria, y en ella se debe colocar el código de control necesario para que el dispositivo realice todo su trabajo. Esto significa que al enfrentar una aplicación compleja, que requiera de una gran cantidad de cálculos por parte del micro, se deberá optimizar el

programa interno al máximo; y no hay mejor manera de hacer esto que programar directamente en lenguaje ensamblador.

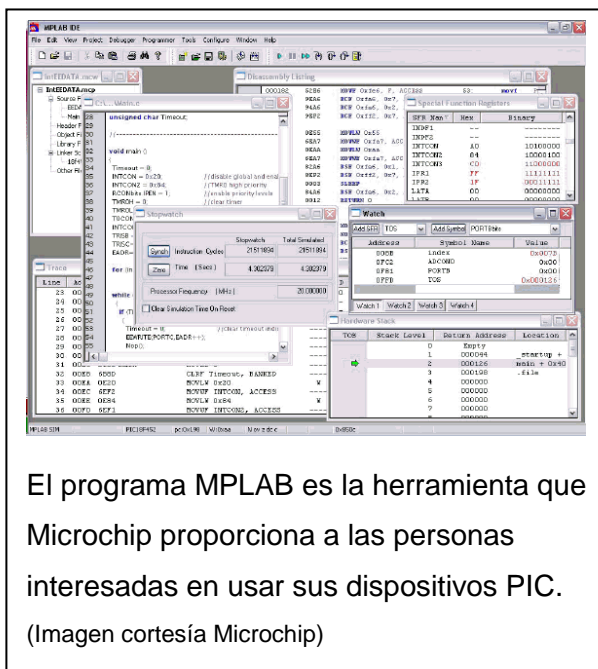
Afortunadamente, muchos fabricantes de microcontroladores comerciales se han percatado de esta situación, y han simplificado el set de instrucciones de sus dispositivos a niveles realmente mínimos. Precisamente una de las marcas que más se ha preocupado por simplificar al máximo el proceso de programación de sus dispositivos es Microchip, con su serie de microcontroladores PIC. Las series 12 y 16 de esta empresa son micros de 8 bits, cuyo set de instrucciones básico tiene apenas 35 instrucciones, lo que significa que incluso para quien apenas comienza a programar este tipo de chips, le resultará relativamente sencillo aprender todas estas instrucciones y comenzar a programar en muy poco tiempo, con resultados satisfactorios. Otros fabricantes han dotado a sus dispositivos con sets de instrucciones más sofisticados, como los chips AVR de Atmel, con un set de 120 instrucciones básicas, o los Z80 de Zilog, con poco más de 150 instrucciones; y los dispositivos CISC como el 8051 de Intel, que posee un set de comandos aún más amplio (más de 200 instrucciones).

Esta situación hace a la familia PIC ideal para dar los primeros pasos en programación de microcontroladores digitales; y es por ello que las prácticas que se realicen de aquí en adelante estarán basadas precisamente en estos dispositivos.

Pero ¿qué es la programación en ensamblador? Precisamente, consiste en tomar el set de instrucciones del dispositivo y, usando únicamente este lenguaje de muy bajo nivel, comenzar a dar las órdenes básicas que deberá ejecutar el chip para llevar a cabo su labor de control. El lenguaje ensamblador manipula directamente los bits y los bytes que está manejando el micro, para encender o apagar terminales, revisar el estado de entradas externas, enviar o recibir datos de sus periféricos, medir tiempos, activar bloques internos, etc. En el caso de los PIC, parece increíble que con 35 instrucciones básicas se puedan elaborar programas extremadamente complejos, donde el micro compara señales análogas, las convierte en señal digital, las procesa y expide

en forma de señal PWM, entra en comunicación con otros chips a través de buses de comunicación serial o paralela, etc., ¡y todo eso ocupando menos de los 2 o 4kB de memoria de programación que normalmente incluyen! Verdaderamente, programar en ensamblador puede generar un código muy poderoso y que ocupa un espacio muy reducido.

Sin embargo, programar en ensamblador tiene también cierto grado de dificultad, ya que se deben verificar manualmente todos y cada uno de los aspectos del programa, cargando los datos en posiciones específicas, alimentando registros, realizando operaciones, consultando bits de estado y banderas, etc. Esto significa que una orden que en C++ o Java ocupa una línea de código, al pasarlo a ensamblador se debe traducir en varias órdenes independientes.



El programa MPLAB es la herramienta que Microchip proporciona a las personas interesadas en usar sus dispositivos PIC. (Imagen cortesía Microchip)

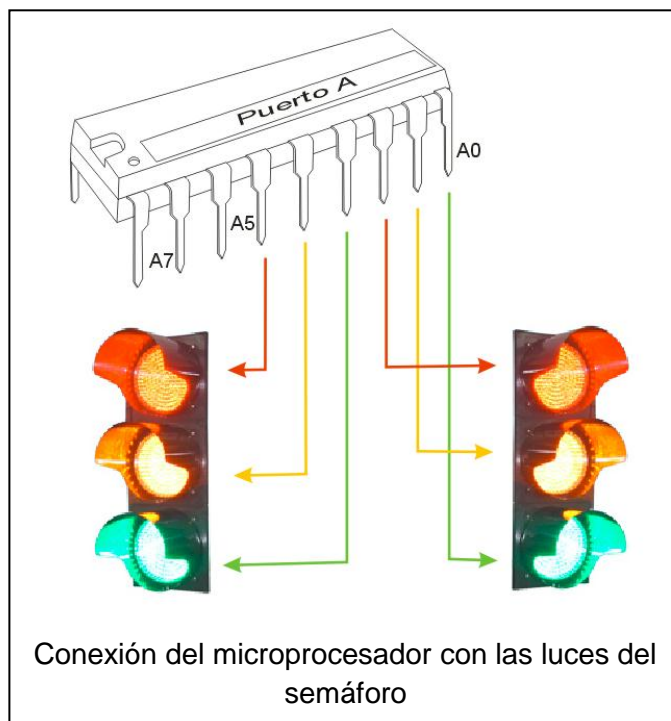
Precisamente por esta razón, casi siempre los fabricantes de microcontroladores ponen a disposición de sus clientes, de forma gratuita, alguna aplicación especializada para el desarrollo de código para sus micros. Se puede decir que se trata de un lenguaje de bajo nivel que amplía ligeramente las opciones del set de instrucciones del dispositivo, y que facilita al usuario el desarrollo del código interno que desee

programar. En el caso de los microcontroladores PIC, Microchip pone a disposición de los usuarios el programa MPLAB, el cual se muestra en la pantalla anexa. Este programa puede descargarse gratuitamente de la página de la empresa (www.microchip.com), y es en el que se basarán las prácticas de programación con dispositivos reales que se realizarán más adelante.

A continuación, se expone un ejemplo de programa en lenguaje ensamblador, usando los comandos básicos que se explicaron en el tema 2.1 de esta unidad. Debido a que apenas se está comenzando, se supondrá que se tienen a disposición algunas subrutinas básicas, y se usarán como bloques funcionales.

Se supondrá que se desea usar un microcontrolador para que encienda y apague secuencialmente las luces de un semáforo que controla el cruce de automóviles en dos calles; y para facilitar aún más esto, las luces duran exactamente lo mismo para ambas calles. Se fijarán algunos valores: luz verde: aproximadamente 40 segundos + 3 intermitentes de 1 segundo cada una; luz amarilla: aproximadamente 3 segundos; luz roja: aproximadamente 50 segundos; y se repite el ciclo. Este patrón debe repetirse en el semáforo de la otra calle, pero obviamente la luz verde se encenderá mientras la del semáforo 1 está en rojo, y viceversa.

En este caso, el microcontrolador tiene un puerto A de 8 bits, de los cuales sólo se usarán 6 como salidas para las luces de los semáforos; a la luz verde del semáforo 1 se asignará el bit A0, al amarillo1 el bit A1, rojo1 = bit A2, verde2 = bit A3, amarillo2 = bit A4 y rojo2 = bit A5; los bits 6 y 7 no se usarán en este proyecto. Observa el diagrama anexo para mayor claridad.



De este modo, ya se tienen los elementos necesarios para comenzar a trabajar; a continuación se indica cómo se puede programar en ensamblador el código para este proyecto. Algo más: mientras se está desarrollando un programa, es posible introducir comentarios que permitan saber qué se está haciendo en cada

momento; en este ejemplo, toda línea que inicie con un punto y coma (;) se considerará como comentario, y no entra en el código de programación.

Una aclaración final: siempre que se mencione código de programación, se cambiará el tipo de letra por una fuente Courier, para que se identifique claramente del resto del texto; esta indicación se mantendrá en todas las unidades restantes.

; Inicio del programa

Start

; Se inicializan las variables internas del micro

Include [Micro que se vaya a utilizar]

; En primer lugar, vamos a fijar que el puerto A sean sólo salidas

Clear PortA-pins

; Se apagan todas las salidas del puerto A

Clear PortA

; Se enciende Verde1 y Rojo2; se pone un indicador para identificar el punto

Point1 - Load 00100001 PortA

; Se cuentan 40 segundos

; Se carga el número 40 en el registro X

Load RegX 28h

; Se llama a la subrutina que introduce una espera de 1 segundo, se pone

; un indicador para identificar este punto

Point2 - Call 1seg

; Se resta una unidad del registro X; cuando llegue a cero, se hace un salto

; de una posición de memoria

Dec RegX skip-z

; Se regresa al indicador 2

Goto Point2

; Cuando se cumplen los 40 segundos, empieza el parpadeo de la luz verde1

Load 00100000 PortA

Call 1seg

Load 00100001 PortA

Call 1seg

Load 00100000 PortA

Call 1seg

Load 00100001 PortA

Call 1seg

Load 00100000 PortA

Call 1seg

Load 00100001 PortA

Call 1seg

Load 00100000 PortA

Call 1seg

; Se enciende amarillo1 por 3 segundos

Load 00100010 PortA

Call 1seg

Call 1seg

Call 1seg

; Se apaga amarillo 1 y se enciende rojo 1 y verde 2

; Se repite lo anterior, pero ahora para el semáforo 2

Load 00001100 PortA

; Se cuentan 40 segundos

; Se carga el número 40 en el registro X

Load RegX 28h

; Se llama a la subrutina que introduce una espera de 1 segundo, se pone

; un indicador para identificar este punto

Point3 - Call 1seg

; Se resta una unidad del registro X; cuando llegue a cero, se hace un salto

; de una posición de memoria

Dec RegX skip-z

; Se regresa al indicador 3

Goto Point3

; Cuando se cumplen los 40 segundos, empieza el parpadeo de la luz verde2

Load 00000100 PortA

Call 1seg

Load 00001100 PortA

Call 1seg

Load 00000100 PortA

Call 1seg

Load 00001100 PortA

Call 1seg

Load 00000100 PortA

Call 1seg

Load 00001100 PortA

Call 1seg

Load 00000100 PortA

Call 1seg

; Se enciende amarillo2 por 3 segundos

Load 00010100 PortA

Call 1seg

Call 1seg

Call 1seg

; Se regresa a la bandera inicial para repetir el ciclo

Goto Point1

; Fin del programa

End

Si se cuentan cuidadosamente los códigos utilizados, se observará que son alrededor de 50 líneas de instrucciones (todo lo que empieza con un punto y coma se considera comentario y no cuenta como código); claro que aquí se asume que hay una subrutina que da retrasos de 1 segundo, aunque en realidad, ésta también se tendría que programar en otro segmento de la memoria. A pesar de ello, resulta evidente lo fácil que es implementar un semáforo sencillo para dos calles usando un microcontrolador que normalmente cuesta menos de 100 pesos; claro que necesita de elementos auxiliares, como los relevadores que encenderán las luces; pero el circuito de control sería extremadamente económico usando un microcontrolador de los más comunes. No sólo eso; si en algún momento, alguna de las calles comienza a tener más tráfico que la otra y se requiere que el paso por ella sea más ágil (aumentando el tiempo que dura el “verde” en dicha dirección), basta con cambiar el parámetro respectivo en el programa (en lugar de 40, colocar un número mayor para el “verde” en la dirección deseada y/o reducir el valor para la que tiene menos tráfico) y el semáforo podrá adaptarse a las necesidades cambiantes de una ciudad.

De este modo, se pudo comprobar lo sencillo que resulta utilizar un microcontrolador en este tipo de aplicaciones simples. En la siguiente unidad, se comenzarán a elaborar programas reales para experimentar con los circuitos PIC; elaborar un programa de este tipo resulta bastante fácil, y poco a poco aumentará el grado de dificultad de los programas, y así se comprobará el poderío de estos dispositivos, y cómo pueden solucionar muchos problemas que anteriormente requerían de una gran cantidad de chips individuales.

AUTOEVALUACIÓN

1. ¿Por qué los seres humanos utilizamos un sistema de numeración en base 10?
2. ¿Qué son los números binarios y por qué se utilizan en circuitos electrónicos digitales?
3. Menciona tres operaciones que se pueden hacer con dos números binarios:
4. ¿Qué son los saltos condicionales y para qué sirven?
5. Cuando se indica que un microprocesador es de “N” bits, ¿qué significa exactamente?
6. ¿Cuál es la diferencia principal entre microprocesadores tipo CISC y dispositivos tipo RISC?
7. ¿Cuál es la diferencia entre el lenguaje ensamblador y los lenguajes de alto nivel?
8. ¿Por qué conviene programar en ensamblador los microcontroladores?
9. ¿Para qué sirven los diagramas de flujo al momento de comenzar a diseñar un programa informático?
10. ¿Para qué sirven los registros PC y stack dentro de un microprocesador?

RESPUESTAS

1. Debido a que poseemos diez dedos en ambas manos.
2. Son números que sólo utilizan dos estados para representar cualquier cantidad; el “1” y el “0”, y se usan en circuitos digitales porque son mucho más fáciles de manejar que la numeración decimal.
3. AND, OR, XOR, suma, resta, comparación, etc.
4. Son instrucciones para que el microprocesador deje de leer secuencialmente su memoria y realice un “salto” hasta una posición de memoria establecida, la cual debe mencionarse en la orden.
5. Al ancho máximo de las palabras con que puede trabajar ese dispositivo.
6. Los dispositivos CISC tienen un set de instrucciones muy amplio, mientras que los RISC poseen un juego de instrucciones reducido.
7. El lenguaje ensamblador es el más básico para comunicarse con el micro y puede aplicarse tal cual, mientras que los de alto nivel necesitan de un compilador para poder aplicarse al dispositivo.
8. Especialmente por su escasa memoria de programación.
9. Para visualizar de forma efectiva los procesos, las tomas de decisiones y los saltos que se deben llevar a cabo dentro de un programa.
10. El PC sirve para llevar el control de la localidad de memoria que se esté leyendo en un momento dado, y el stack para guardar la dirección en que estaba el PC antes de llamar a alguna subrutina.

Soluciones a las actividades de aprendizaje:

Actividad de aprendizaje 2A:

- a) - 352d = 101100000 = 160h b) - 129d = 10000001 = 81h
c) - 1017d = 1111111001 = 3F9h d) - 10011010 = 154d
e) - 1101110010 = 882d f) - 10101010 = 170d
g) - 14Ah = 101001010 = 330d h) - B7h = 10110111 = 183d
i) - 2F5h = 101110101 = 757d

Actividad de aprendizaje 2B:

- a) 1010 0100 b) 0001 0001 c) 0010 0010
d) 1010 0001 C=0 e) 0000 1011 C=1 f) 0001 0000 C=1
g) 0011 1110 DC=0 Z=0 h) 1111 1100 DC=1 Z=0 i) 0000 0000 DC=0 Z=1

Actividad de aprendizaje 2C:

Z8 – 83 instrucciones; 8051 – 255 instrucciones; 68HC11 – 145 instrucciones
PIC12-16 – 35 instrucciones; PIC18 – 77 instrucciones

Actividad de aprendizaje 2D:

- Fase de arranque e inicialización
- Fase de prueba de sensores internos e inicialización mecánica
- Fase de lectura inicial de disco
- Fase de espera
- Fase de espera de órdenes cuando hay disco insertado
- Fase de reproducción de disco.

UNIDAD 3

CARACTERÍSTICAS ESPECÍFICAS DE MICROPROCESADORES DE 8 BITS

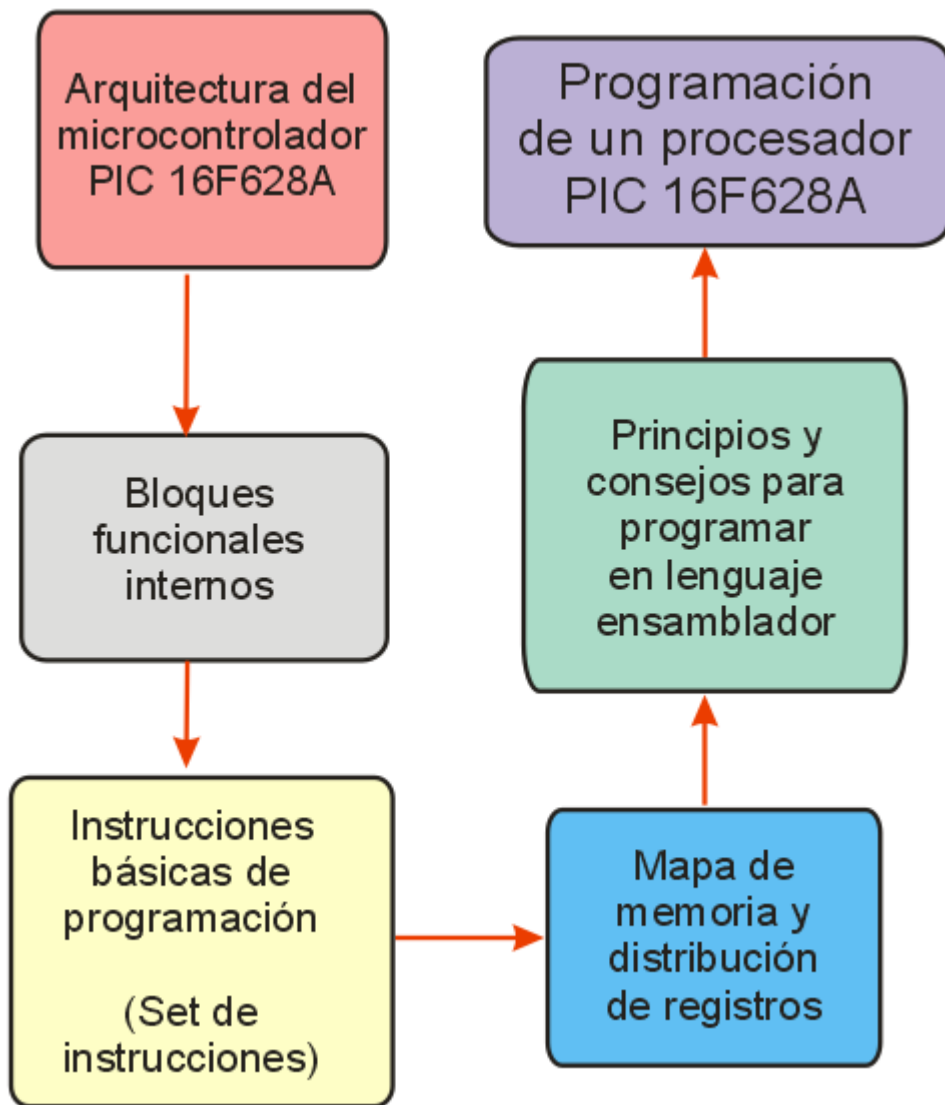
OBJETIVO

El estudiante conocerá los detalles internos y operativos del microcontrolador que se utilizará para las prácticas que se indicarán en este libro e identificará el set de instrucciones del micro para elaborar los primeros programas en ensamblador; además, reconocerá el método para cargar el programa en el microcontrolador y probarlo en un circuito externo.

TEMARIO

- 3.1 ANÁLISIS DE LAS CARACTERÍSTICAS DE UN MICROCONTROLADOR COMERCIAL DE 8 BITS
- 3.2 ARQUITECTURA DEL MICROCONTROLADOR DE 8 BITS
- 3.3 MODOS DE DIRECCIONAMIENTO DEL PIC 16F628A
- 3.4 CONJUNTO DE INSTRUCCIONES DEL MICROCONTROLADOR PIC 16F628A
- 3.5 PRIMER PROGRAMA EN ENSAMBLADOR

MAPA CONCEPTUAL



INTRODUCCIÓN

Hasta ahora se han descrito de forma muy general los microprocesadores y microcontroladores, y aunque se ha mencionado en repetidas ocasiones que se usará un dispositivo de la familia PIC de Microchip como base para las prácticas que se realizarán, aún no se ha aterrizado en la arquitectura particular de este micro, sus bloques internos, la forma de aprovecharlo, su lenguaje de programación específico, etc.

Precisamente en esta unidad se profundizará en el microcontrolador que se utilizará de ahora en adelante, el PIC 16F628A de Microchip, un microcontrolador muy poderoso y flexible, que es relativamente fácil de conseguir y cuyo costo es muy razonable, lo que significa que cualquier persona puede adquirirlo y comenzar a experimentar los pros y contras de los circuitos de proceso digital.

3.1 ANÁLISIS DE LAS CARACTERÍSTICAS DE UN MICROCONTROLADOR COMERCIAL DE 8 BITS

Como se ha mencionado a lo largo de este libro, los microcontroladores de la familia PIC 12-16 de Microchip son dispositivos de proceso digital de 8 bits, capaces de realizar desde las tareas más sencillas hasta algunas que realmente sorprenden por su complejidad. Estos dispositivos son capaces de realizar hasta 5 millones de operaciones por segundo, o bien utilizar un modo de muy baja energía en el cual apenas consumen unos cuantos microamperios de corriente (menos de una milésima parte de amperio), lo que los hace ideales para aplicaciones donde tengan que ser alimentados por baterías. La serie 12-16 de PIC posee además una amplia variedad de bloques internos, que van desde circuitos de reset (ya no es necesario colocar un chip especial para esta función), como generadores de reloj, temporizadores, comparadores analógicos, convertidores analógico-digital, circuitos de vigilancia internos, diversos puertos I/O para entrada o salida de datos, tres tipos de memoria interna: flash para programación, RAM para almacenar datos temporales y



EEPROM para guardar datos que se deseen conservar incluso cuando se retira la alimentación del dispositivo, bloques de transmisión y recepción de datos, en fin, una muy amplia diversidad de bloques funcionales que están disponibles dependiendo del modelo específico de micro que utilizará.

Estos microcontroladores se pueden adquirir en versiones que van desde las 8 a las 40 terminales o más,

para aplicaciones desde muy pequeñas hasta muy sofisticadas; y a todo lo anterior se debe añadir una muy amplia disponibilidad en el mercado electrónico, y un precio extremadamente razonable. Todas estas razones hacen

de la familia PIC 12-16 la ideal para comenzar con el mundo de los microcontroladores.

Las características principales del micro PIC16F628A que se usará en las prácticas, son las siguientes: Se trata de un dispositivo de 8 bits en arquitectura RISC (computación por set de instrucciones reducido), cuyo lenguaje ensamblador sólo posee 35 comandos individuales, lo que lo hace muy fácil de aprender y dominar. El micro posee un oscilador interno seleccionable entre 4MHz (reloj normal) o 48KHz (para aplicaciones de muy bajo consumo de potencia), lo que le permite funcionar sin la presencia de una fuente de reloj externa; pero se le puede aplicar una oscilación de hasta 20MHz para máxima velocidad de proceso. El dispositivo puede ser alimentado con un voltaje de 2 hasta 5.5V; se encuentra en un encapsulado de 18 terminales, de las cuales hasta 16 pueden usarse como entradas o salidas de señal; posee dos comparadores análogos, que pueden usar ya sea un voltaje de referencia externo o uno interno para su labor; posee 3 temporizadores, uno de 16 bits y 2 de 8 bits; un bloque PWM para expedir señales análogas simuladas, un bloque de comunicaciones consistente en un receptor-transmisor universal síncrono-asíncrono (USART); y muchos bloques más para aplicaciones más especializadas.

El 16F628A posee 2 kilobytes de memoria flash para almacenar su programa interno, además de 224 bytes de RAM para guardar datos temporales y 128 bytes de EEPROM para datos semipermanentes. La ventaja de que la memoria de programación sea flash es que, en caso de que al desarrollar el programa, se cometiera algún error, simplemente se corrige y se carga nuevamente en el micro.

La memoria flash se puede grabar hasta 100,000 veces antes de que aparezcan errores, excediendo en mucho la vida útil del dispositivo.



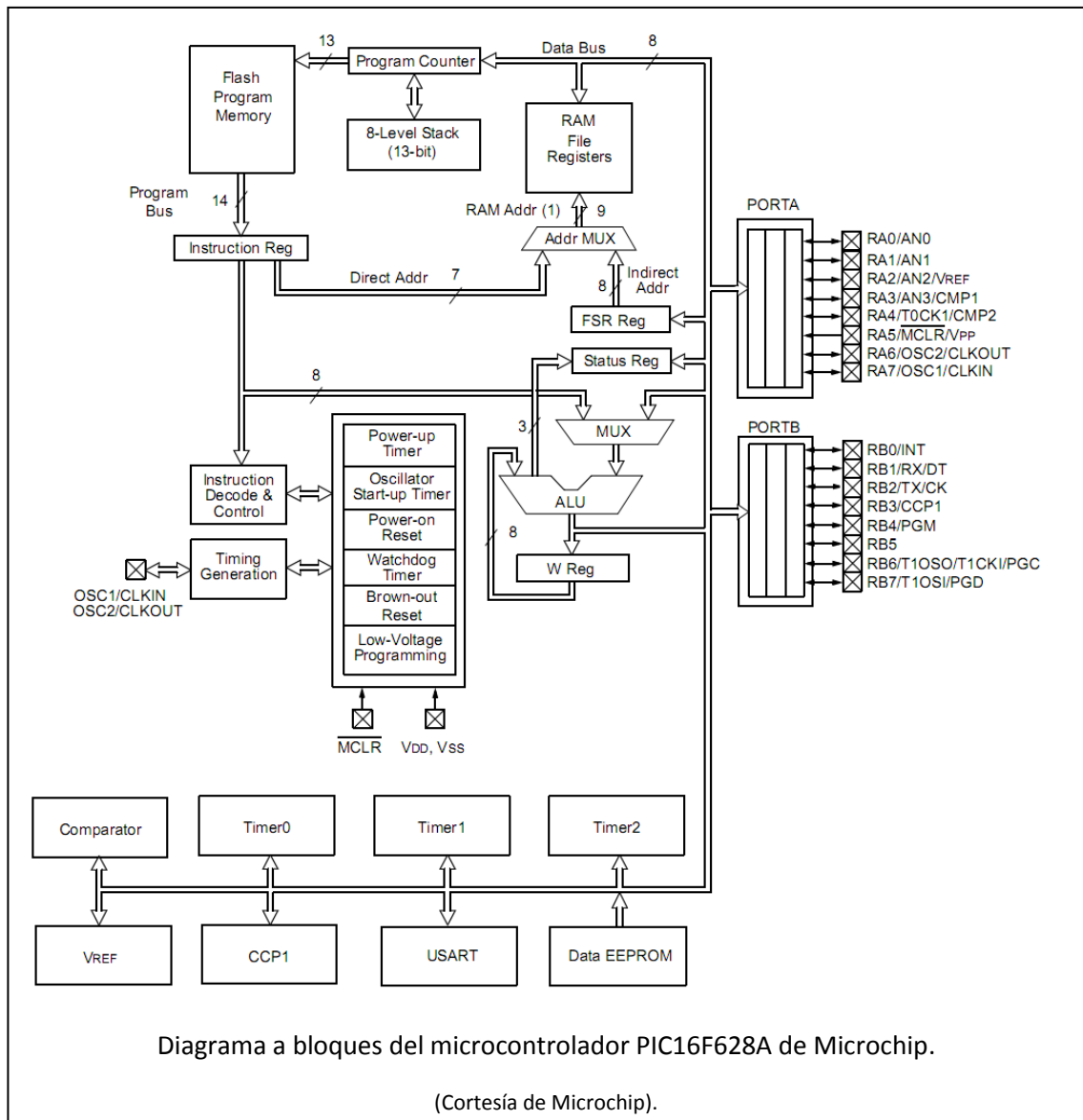
Un detalle especial que tiene la memoria, es que en el bloque de flash, cada celda tiene una extensión de 14 bits, mientras que en la RAM y la EEPROM sus celdillas son de 8 bits; esto se debe a la arquitectura Harvard de este dispositivo, que mantiene separados el bus de datos y el bus de programa; y esto le permite que en una sola orden de 14 bits, se incluya el comando en cuestión y el dato que se va a utilizar o la dirección a la que se requiere acceder. Esto se traduce en menor tiempo para ejecutar las órdenes; de forma típica, un comando utiliza un ciclo de reloj para procesarse, con la excepción de los saltos, que requieren de dos ciclos. Si se considera que el micro toma la señal del oscilador y la divide entre 4 para generar su reloj interno, y sabiendo que la máxima frecuencia del oscilador es de 20MHz, entonces este micro es capaz de realizar hasta 5 millones de operaciones sencillas por segundo, algo más que razonable para un dispositivo de este precio.

ACTIVIDAD 3A

Contesta lo siguiente:

- 1.- ¿Cuál es la empresa que fabrica el microcontrolador que se usará como ejemplo para las prácticas de este libro?
- 2.- ¿De cuántos bits es la serie PIC 12-16?
- 3.- ¿Qué tipo de arquitectura utiliza la familia PIC, Von Neumann o Harvard?
- 4.- ¿Cuál es la principal diferencia entre ambas arquitecturas?
- 5.- ¿Cuál es la ventaja de que la memoria de programación sea de tipo flash?
- 6.- ¿Cuál es la frecuencia máxima que puede manejar este dispositivo?
- 7.- ¿Es indispensable colocar un reloj externo al PIC 16F628A?
- 8.- ¿Cuántos temporizadores incluye este dispositivo?
- 9.- ¿Incluye comparadores analógicos? ¿Cuántos?
- 10.- ¿Para qué sirve el bloque de memoria EEPROM?

3.2 ARQUITECTURA DEL MICROCONTROLADOR DE 8 BITS



En la figura anexa se muestra el diagrama a bloques interno del microcontrolador PIC16F628A para tener una idea de su complejidad; se puede observar la gran cantidad de elementos que trae incorporados, permitiendo así su aplicación prácticamente inmediata, sin necesidad de rodearse de circuitos auxiliares externos. Lo realmente sorprendente es imaginar que todo eso se encuentra en el interior de un circuito integrado de 18 terminales más pequeño que un clip de oficina, y que normalmente se puede adquirir en menos de 100 pesos, lo que lo pone al alcance de casi cualquier bolsillo.

Sin embargo, y a pesar de su complejidad aparente, este dispositivo mantiene la misma estructura básica que ya se explicó en unidades anteriores; por ejemplo, se puede observar en la parte media del diagrama la unidad aritmética lógica (ALU), que se encarga de realizar todas las operaciones dentro del circuito. Exactamente debajo de ella está el registro de trabajo *W*, que es donde se carga uno de los operandos de cualquier operación de dos bytes que se deseen hacer en la ALU, pudiendo obtener el otro ya sea de la memoria flash de programa, de la RAM, de la EEPROM o incluso de alguno de sus puertos de entrada. Encima de la ALU, se observa la presencia de dos registros especiales: el de estado y el FSR, que sirve para localizar de forma indirecta los datos contenidos en alguna dirección de memoria. Arriba de estos registros se encuentra la RAM, y parte de esta memoria también se usa como registros especializados para el micro, donde se configuran las entradas/salidas de los puertos, si se van a activar los temporizadores y cómo se configuran, si se van a usar las interrupciones y cómo, si se van a usar los comparadores analógicos y cómo, etc. Más adelante, se presentará un mapa detallado de la memoria RAM de este micro, y se indicará que hay localidades que sólo se tocan durante la configuración inicial del dispositivo, y no se pueden usar para trabajo normal.

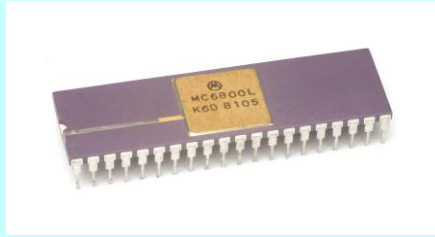
En la parte derecha del diagrama están los dos puertos I/O generales, cada uno de 8 bits; cada una de las terminales de estos puertos puede configurarse como entrada o salida, independientemente de las demás, así que en un momento dado, un puerto puede estar expidiendo órdenes por algunos pines, y recibiendo instrucciones a través de otros pines del mismo puerto. Esto da mucha flexibilidad al diseñador, para simplificar en la medida de lo posible los trazos del circuito impreso donde se montará el micro.

En la parte inferior se encuentran los bloques periféricos adicionales que incluye el 16F628A, como son los tres temporizadores, el comparador analógico, el generador de voltaje de referencia, el generador-receptor PWM (CCP1), el bloque de comunicaciones USART y la EEPROM para datos. Normalmente todo esto tendría que añadirse al circuito a través de chips

externos, pero debido a la integración lograda por los dispositivos digitales, ahora ya forman parte de la estructura interna de este microcontrolador.

En el bloque que se encuentra a la izquierda de la ALU, se encuentran una serie de etapas encargadas de garantizar el buen funcionamiento del dispositivo; por ejemplo, hay un temporizador de inicio, un oscilador de arranque, el bloque reset, un temporizador watchdog (para vigilar que el programa se ejecute correctamente), un reset “en frío” y un bloque de programación a bajo voltaje; más a la izquierda se ubica el generador de reloj y el control y decodificador de instrucciones. Finalmente, en la parte superior (junto a la RAM) se encuentra el registro PC, el Stack y la memoria flash de programación, con su respectivo registro de instrucciones. Todos estos bloques están unidos entre sí por una serie de buses internos, casi todos de 8 bits, excepto el de instrucciones, que es de 14 bits, y algunos otros de funciones especiales.

Arquitectura Von Neumann vs. Arquitectura Harvard



Los microprocesadores convencionales, como los de la serie 80xx de Intel o la 68xx de Motorola, utilizan una arquitectura tradicional Von Neumann, en la cual tanto los datos como las instrucciones se guardan en una memoria común, y ambos miden exactamente la misma cantidad de bits. Esto permite utilizar un bloque de memoria único, lo que simplifica en cierta medida el direccionamiento del mismo, aunque tiene como inconveniente que cuando se indica una orden, se tiene que dar por un lado el comando y por otro lado el dato con que se trabajará, y esto origina que cada comando necesita de varios ciclos de reloj para poder ser ejecutado.

En la arquitectura Harvard, hay un bloque independiente de memoria para instrucciones y otro para datos, lo que complica ligeramente el acceso a la memoria, pero tiene la ventaja de que ambos bloques pueden tener una extensión distinta; por ejemplo, en los microcontroladores PIC, que usan una arquitectura de este tipo, la memoria flash de programación usa celdillas de 14 bits cada una, mientras que el bloque de RAM utiliza celdillas de 8 bits cada una. Con esto, lo que se consigue es que haya muchas instrucciones en las que se combina la orden y el dato con que se va a trabajar, y por tanto, se pueden ejecutar en un ciclo de reloj único, lo que acelera considerablemente la velocidad de proceso.

(Imágenes cortesía de Motorola y Microchip).

Esta es la arquitectura interna del PIC 16F628A; se debe tener presente para saber qué esperar y qué no esperar de este dispositivo, y sus límites y alcances al aplicarlo en distintos proyectos.

ACTIVIDAD DE APRENDIZAJE 3B

Contesta lo siguiente:

- 1.- ¿Qué significa ALU y qué función tiene este bloque dentro del microcontrolador?
- 2.- ¿Hasta cuántas terminales I/O puede manejar este micro?
- 3.- ¿Cuántos registros de trabajo posee este dispositivo?
- 4.- ¿Para qué sirven los registros PC y Stack, y qué profundidad tiene este último en este PIC?
- 5.- ¿De qué extensión es la palabra en la memoria de programación de la familia PIC 12-16?
- 6.- ¿Qué significa USART y para qué sirve este bloque?
- 7.- ¿Cuál es el voltaje de operación normal de este micro?
- 8.- ¿Cuál es la función del bloque CCP1?
- 9.- ¿Cuál es la función del watchdog timer?
- 10.- ¿Para qué se necesita el bloque de memoria RAM dentro del micro?

3.3 MODOS DE DIRECCIONAMIENTO DEL PIC 16F628A

Como se mencionó en la unidad anterior, los procesadores lógicos, en su set de instrucciones, pueden manejar una amplia variedad de modos de direccionamiento para el acceso de datos hacia los registros de trabajo. Sin embargo, al considerar que la serie PIC 12-16 sólo maneja 35 instrucciones en su set básico de comandos, por lo tanto tiene un número muy reducido de modos de direccionamiento; a continuación se indican cuáles son:

- *Direccionamiento inmediato:* Se carga directamente un registro con una cantidad predeterminada: `MOVLW [literal]`. Ejemplo: `MOVLW 0x5A`; después de ejecutarse la orden, el valor 5Ah se carga en el registro W.
- *Direccionamiento directo:* Se carga un valor desde una localidad específica de memoria: `MOVF [dirección], [destino]`, donde [dirección] es el registro o localidad de memoria desde donde se lee el dato, y [destino] es hacia dónde se moverá este dato, siendo 0 = W y 1 = el mismo registro o localidad de memoria. Ejemplo: `MOVF REG1, 0`; después de ejecutarse la orden, el contenido de REG1 se carga en W.
- *Direccionamiento de registro a memoria:* Se carga el contenido de un registro en alguna localidad de memoria o en otro registro: `MOVWF REG1`; después de ejecutar la instrucción, el contenido de W se carga en el registro REG1.
- *Direccionamiento a registro:* Se hace una operación sobre la cantidad que ya está guardada en algún registro: `[COMANDO] [registro], [destino]`. Después de la instrucción, el número contenido del registro varía según la orden, y el resultado se guarda en W si el destino es "0", y en el mismo registro si el destino es "1". Ejemplo: `INCF REG1, 1`; después de la orden, el número contenido en REG1 se incrementa en una unidad, y el resultado se guarda en el mismo REG1.

¡Y eso es todo!, estos son los cuatro modos de direccionamiento que se pueden invocar en los PIC de la serie 12-16. Obviamente existen más órdenes

que utilizan estos modos de direccionamiento, pero se detallarán enseguida, cuando se describan una por una las órdenes del set de instrucciones de estos dispositivos.

ACTIVIDAD DE APRENDIZAJE 3C

Contesta lo siguiente:

- 1.- ¿Cuántas órdenes posee el set de instrucciones de los PIC 12-16?
- 2.- ¿Cuántos tipos de modos de direccionamiento posee el PIC 16F628A?
- 3.- Menciona dos de estos modos de direccionamiento.

3.4 CONJUNTO DE INSTRUCCIONES DEL MICROCONTROLADOR PIC 16F628A

Ahora, se estudiará el set de instrucciones del microprocesador 16F628A, para saber qué es lo que se puede hacer con él, y cómo aprovecharlo para que realice las tareas que se le asignen a través de su programa. Como ya se ha mencionado, este set de instrucciones sólo cuenta con 35 comandos básicos, así que no debería ser difícil aprenderlos todos de memoria (además de que hay algunos que se utilizan sólo en pocas ocasiones).

A continuación se muestran todas las órdenes del set de instrucciones de la familia PIC 12-16 con una descripción básica sobre la utilidad de cada una, esta lista puede servir como referencia rápida como auxiliar de programación, pero si se desea información detallada sobre qué hace exactamente cada orden, enseguida se hará una descripción punto por punto de qué es lo que hace cada comando, y cómo afectan a las distintas banderas de estado. Cabe aclarar que en la notación de Microchip, cuando nos referimos a una localidad de memoria específica se le identifica como "F", así que se denominará de esa forma de ahora en adelante.

No.	Comando	Descripción	Banderas
<i>Operaciones relacionadas con bytes en registros y localidades de memoria</i>			
1	ADDWF	Suma el contenido de W y una localidad de memoria F	C, DC, Z
2	ANDWF	Operación AND entre W y F	Z

3	CLRF	Borra el contenido de F	Z
4	CLRW	Borra el contenido de W	Z
5	COMF	Operación complemento a F	Z
6	DECF	Decrementa el contenido de F	Z
7	DECFSZ	Decrementa F, salta una posición si llega a cero	
8	INCF	Incrementa el contenido de F	Z
9	INCFSZ	Incrementa F, salta una posición si llega a cero	
10	IORWF	Operación OR entre W y F	Z
11	MOVF	Mueve F	Z
12	MOVWF	Mueve el contenido de W a F	
13	NOP	No operación	
14	RLF	Rotar F a la izquierda a través de carry	C
15	RRF	Rotar F a la derecha a través de carry	C
16	SUBWF	Resta entre W y F	C, DC, Z
17	SWAPF	Intercambia los nibbles de F	
18	XORWF	Operación XOR entre W y F	Z
<i>Operaciones relacionadas con bits dentro de registros o localidades de memoria</i>			
19	BCF	Borra un bit dentro de F	
20	BSF	Activa un bit dentro de F	
21	BTFSC	Prueba un bit dentro de F, salta una posición si es cero	
22	BTFSS	Prueba un bit dentro de F, salta una posición si es uno	
<i>Operaciones con literales y de control</i>			
23	ADDLW	Suma el contenido de W con una literal	C, DC, Z
24	ANDLW	Operación AND entre W y una literal	Z
25	CALL	Llama a una subrutina	TO PD
26	CLRWDT	Borra el temporizador watchdog	
27	GOTO	Salto incondicional a una dirección	
28	IORLW	Operación OR entre W y una literal	Z
29	MOVLW	Mueve una literal a W	
30	RETFIE	Regreso de una interrupción	
31	RETLW	Regresa de una subrutina con una literal en W	
32	RETURN	Regreso de una subrutina	
33	SLEEP	Entrar en modo de espera y bajo consumo	TO, PD

34	SUBLW	Subtrae W de una literal	C, DC, Z
35	XORLW	Operación XOR entre W y una literal	Z

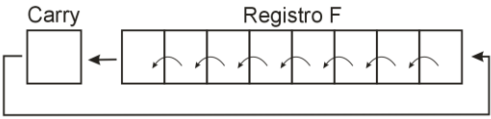
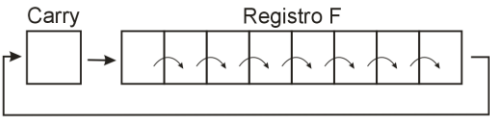
Estas son todas las órdenes que se pueden indicar a un microcontrolador PIC de la familia 12-16 (8 bits); ahora se detallará qué significa cada una y cómo se lleva a cabo. Se hará esta descripción en estricto orden alfabético, para localizar fácilmente la explicación de cualquier comando.

<p>ADDLW k</p> <p>Se suma el valor de W con una literal k, donde k = de 0 a 255, el resultado se guarda en W.</p> <p>Ejemplo: ADDLW 0x15</p> <p>Si W era igual a 0x10, después de la instrucción, W = 0x25</p> <p>Si el resultado rebasa los 8 bits, se activa el bit CARRY.</p>	<p>ADDWF</p> <p>Se suma el valor de W con el contenido en la localidad de memoria F; el resultado se guarda en F si el bit de opción es 1, y en W si es 0.</p> <p>Ejemplo: ADDWF REG1, 1</p> <p>En este caso, se suman W y REG1, y el resultado se guarda en REG1.</p> <p>Si el resultado rebasa los 8 bits, se activa el bit CARRY.</p>
<p>ANDLW</p> <p>Operación AND entre W y una literal k, donde k = de 0 a 255, el resultado se guarda en W.</p> <p>Ejemplo: ANDLW 0x5F</p> <p>Si W era igual a 0xA3, después de la instrucción, W = 0x03</p>	<p>ANDWF</p> <p>Operación AND entre W y el contenido de una localidad de memoria F; el resultado se guarda en F si el bit de opción es 1, y en W si es 0.</p> <p>Ejemplo: ANDWF REG1, 0</p> <p>En este caso, se hace una AND entre W y REG1, y el resultado se guarda en W.</p>
<p>BCF</p> <p>Borra un bit dentro de una localidad de memoria F.</p> <p>Ejemplo: BCF REG1, 7</p> <p>Se borra el bit 7 de REG1; si REG1 era 0xF1, después de la orden será 0x71</p>	<p>BSF</p> <p>Activa un bit dentro de una localidad de memoria F.</p> <p>Ejemplo: BSF REG1, 3</p> <p>Se activa el bit 3 de REG1, si REG1 era 0x51, después de la orden será 0x55</p>
BTFSC	BTFSS

<p>Prueba un bit dentro de una localidad F, y el PC salta una unidad si es cero.</p> <p>Ejemplo: BTFSC REG1, 5</p> <p>Si el bit 5 de REG1 es 1, el PC sigue con su cuenta normal, pero si es 0, salta una unidad.</p>	<p>Prueba un bit dentro de una localidad F, y el PC salta una unidad si es uno.</p> <p>Ejemplo: BTFSS REG1, 5</p> <p>Si el bit 5 de REG1 es 0, el PC sigue con su cuenta normal, pero si es 1, salta una unidad.</p>
--	---

<p>CALL</p> <p>Se llama al contenido de una subrutina.</p> <p>Ejemplo: CALL 1SEG</p> <p>El contenido del PC se guarda en el Stack, y cambia a la dirección de la subrutina. Cuando se regresa de ella, se recupera el número del Stack hacia el PC, y sigue con su cuenta normal.</p>	<p>CLRF</p> <p>Borra el contenido de una localidad de memoria F.</p> <p>Ejemplo: CLRF REG1</p> <p>Sin importar qué número estuviera guardado en REG1, después de la orden se tendrá un 0x00</p>
<p>CLRW</p> <p>Borra el contenido de una localidad de memoria F.</p> <p>Ejemplo: CLRW</p> <p>Sin importar qué número estuviera guardado en W, después de la orden se tendrá 0x00</p>	<p>CLRWDT</p> <p>Borra el temporizador Watchdog de vigilancia.</p> <p>Ejemplo: CLRWDT</p> <p>Se reinicializa el contador interno del WDT.</p>
<p>COMF</p> <p>Operación complemento en el contenido de la localidad de memoria F. Si el bit de opción es 0, el resultado se guarda en W, y en REG1 si el bit es 1.</p> <p>Ejemplo: COMF REG1, 0</p> <p>Si REG1 era 0x13, luego de la operación REG1 = 0x13 y W = 0xEC</p>	<p>DECF</p> <p>Se decrementa en una unidad el contenido de F. Si el bit de opción es 0, el resultado se guarda en W, y en REG1 si el bit es 1.</p> <p>Ejemplo: DECf REG1, 1</p> <p>Si REG1 era 0x24, después de la instrucción REG1 = 0x23</p>
<p>DECFSZ</p> <p>Se decrementa en una unidad el valor</p>	<p>GOTO</p> <p>Salto incondicional hacia cualquier</p>

<p>de F, y cuando llega a cero, el PC hace un salto de una unidad en su cuenta.</p> <p>Ejemplo: DECFSZ REG1</p> <p>Si REG1 no es cero, el PC sigue con su cuenta normal, pero si llega a cero, el PC hace un salto de una unidad.</p>	<p>dirección válida de memoria</p> <p>Ejemplo: GOTO RUTINAX</p> <p>Sin importar dónde estuviera el PC, luego de la orden irá a la dirección de la rutina-X</p>
<p>INCF</p> <p>Se incrementa en una unidad el valor dentro de F. Si el bit de opción es 1, el valor se guarda dentro de F, y si es 0, se guarda en W.</p> <p>Ejemplo: INCF REG1, 1</p> <p>Si REG1 era 0x24, después de la orden REG1 = 0x25</p>	<p>INCFSZ</p> <p>Se incrementa el valor de F, y si llega a cero, el PC hace un salto de una unidad. Si el bit de opción es 1, el valor se guarda dentro de F, y si es 0, se guarda en W.</p> <p>Ejemplo: INCFSZ REG1, 1</p> <p>Si REG1 tiene cualquier valor distinto a cero, el PC sigue su cuenta normal, pero si llega a cero, el PC hace un salto de una unidad.</p>
<p>IORLW</p> <p>Se hace una operación OR entre W y una literal k, donde k = de 0 a 255; el resultado se guarda en W.</p> <p>Ejemplo: IORLW 0x35</p> <p>Si W era 0x9A, luego de la instrucción W = 0xBF</p>	<p>IORWF</p> <p>Se hace una operación OR entre W y el contenido de F; si el bit de opción es 1, el valor se guarda dentro de F, y si es 0, se guarda en W.</p> <p>Ejemplo: IORWF REG1, 0</p> <p>Si W = 0x91 y REG1 = 0x13, luego de la instrucción W = 0x93</p>
<p>MOVLW</p> <p>Carga una literal k en el registro W, donde k = de 0 a 255.</p> <p>Ejemplo: MOVLW 0x3A</p> <p>Sin importar qué hubiera en W anteriormente, después de la orden W = 0x3A</p>	<p>MOVF</p> <p>El contenido de la dirección de memoria F se mueve a W si el bit de opción es 0, y se queda en F si el bit de opción es 1.</p> <p>Ejemplo: MOVF REG1, 0</p> <p>El valor de REG1 se pasará a W después de la orden.</p>
<p>MOVWF</p> <p>El contenido del registro W se graba en</p>	<p>NOP</p> <p>Ninguna operación; se utiliza para</p>

<p>la localidad de memoria F.</p> <p>Ejemplo: MOVWF REG1</p> <p>Si $W = 0xA1$, después de la orden este valor se cargará en REG1.</p>	<p>introducir un retardo de un ciclo de reloj, cuando se quiere llevar un tiempo muy preciso.</p> <p>Ejemplo: NOP</p>
<p>RETFIE</p> <p>Regresa desde una interrupción: cuando se recibe una interrupción, el PC hace un salto incondicional a una cierta localidad de memoria, pero guarda su valor anterior en el Stack; cuando se termina de ejecutar la rutina que se invoca por la interrupción, una orden RETFIE regresa al PC a la localidad de memoria donde estaba anteriormente.</p> <p>Ejemplo: RETFIE</p>	<p>RETLW</p> <p>Regresa de una subrutina con una literal cargada en W. Cuando termina una subrutina y se desea regresar al programa normal, pero con un dato específico en W, esta orden permite hacerlo en un solo comando.</p> <p>Ejemplo: RETLW, 0xA3</p> <p>Al regresar de la subrutina, el registro W tendrá en su interior un valor de 0xA3.</p>
<p>RETURN</p> <p>Regreso desde una subrutina. Cuando se termine de ejecutar una subrutina y se desee regresar al programa normal, una orden RETURN hace que el PC recupere el valor guardado en el Stack, y retome su cuenta normal.</p> <p>Ejemplo: RETURN</p>	<p>RLF</p> <p>Rota los bits del registro F hacia la izquierda, pasando por el bit CARRY</p>  <p>Ejemplo: RLF REG1</p>
<p>RRF</p> <p>Rota los bits del registro F hacia la derecha, pasando por el bit CARRY.</p>  <p>Ejemplo: RRF REG1</p>	<p>SLEEP</p> <p>Esta orden coloca al micro en un modo de mínima energía, mientras espera nuevas órdenes.</p> <p>Ejemplo: SLEEP</p>
<p>SUBLW</p>	<p>SUBWF</p>

<p>Subtrae el valor de W de una literal k, donde k = de 0 a 255. El resultado se guarda en W.</p> <p>Ejemplo: SUBLW 0x28</p> <p>Si W = 0x03, luego de la orden, W = 0x25. Dependiendo si W es mayor, menor o igual a la literal, se activan los bits C, Z y DC</p>	<p>Subtrae el valor de W del contenido de F. El resultado se guarda en W si el bit de opción es 0, y en F si es 1.</p> <p>Ejemplo: SUBWF REG1, 1</p> <p>Dependiendo si W es mayor, menor o igual a la literal, se activan los bits C, Z y DC</p>
<p>SWAPF</p> <p>Intercambia los nibbles (grupos de 4 bits) del contenido de F. El resultado se guarda en W si el bit de opción es 0, y en F si es 1.</p> <p>Ejemplo: SWAPF REG1, 1</p> <p>Si REG1 = 0x9A, después de la orden REG1 = 0xA9</p>	<p>XORLW</p> <p>Operación XOR entre el contenido de W y una literal k, donde k = de 0 a 255. El resultado se guarda en W.</p> <p>Ejemplo: XORLW 0xAF</p> <p>Si W era 0xB5, después de la orden W = 0x1A</p>
<p>XORWF</p> <p>Operación XOR entre W y el contenido de F. El resultado se guarda en W si el bit de opción es 0, y en F si es 1.</p> <p>Ejemplo: XORWF REG1, 0</p> <p>Si W = 0xB5 y REG1 = 0xAF, luego de la orden W = 0x1A</p>	

De este modo, se puede tener una idea más clara de para qué sirve cada una de las órdenes incluidas en el set de instrucciones de los microcontroladores PIC serie 12-16. Una gran ventaja que tienen estos dispositivos es que su set de comandos casi no ha cambiado desde que comenzaron a fabricarlos, así que un programa que se diseñe en este momento puede funcionar prácticamente sin modificaciones en otros micros más antiguos o en dispositivos que aún están en la mesa de diseño; de ahí la conveniencia de realizar los programas en forma de bloques funcionales, para que cuando

llegue algún proyecto con una etapa que ya se había diseñado para un circuito anterior, se pueda retomar haciéndole sólo modificaciones mínimas.

Antes de pasar a las prácticas de programación, se debe describir un aspecto importante de la operación de este microcontrolador: los registros especiales donde se activan las banderas de estado; y esto implica echar un vistazo al mapa de memoria del dispositivo.

A continuación se encuentra el mapa de memoria de un microcontrolador PIC 16F628A; se puede observar que varias de las localidades de memoria que posee el micro ya están ocupadas por una gran cantidad de registros, cada uno con una función específica dentro de la estructura del micro. Se describirán los más importantes, y se dejarán algunos para cuando se estudien ciertos temas específicos sobre este controlador.

Se iniciará por orden numérico: en la localidad 01h se encuentra el registro correspondiente al temporizador No. 0; por el momento no se indicará para qué sirve, ya que este bloque sólo se utiliza para ciertas funciones avanzadas del micro.

El registro PCL corresponde a los 8 bits inferiores del registro PC o contador de programa; dado que el 16F628A posee 2kB de memoria de programa, pero con 8 bits sólo se pueden direccionar 256 bits, esto implica que necesita por lo menos 3 bits adicionales para direccionar a toda su memoria, entonces, en PCL se guardan los 8 bits inferiores, y en PCLATH los adicionales.

El registro STATE o de estado guarda algunos de los bits más usuales durante la programación del dispositivo, como las banderas C (carry), Z (cero) y DC (decrement carry), además del IRP, RP0 y RP1 (elección del banco de memoria usado), TO (Time out) y PD (Power down). En programas sencillos sólo se usarán los bits C, Z y DC, pero en programas más complejos se llegan a utilizar todos los demás, especialmente los tres primeros, ya que se activan o desactivan dependiendo de en qué bloque de memoria se alojen las instrucciones del programa.

El registro OPTION_REG controla aspectos de las interrupciones y los temporizadores, pero hay un bit especial RBPU, que habilita o desactiva unas

El registro INTCON sirve para configurar las interrupciones y cómo se utilizarán (en caso de que el programa lo requiera). Los registros PIE y PIR1 sirven para manejar las interrupciones a través de bloques periféricos como los temporizadores, el USART, los comparadores, etc.

El registro PCON posee un bit importante para el diseño usando este PIC: OSCF o frecuencia del oscilador interno; si se coloca en “1”, la frecuencia será de 4MHz, en “0” será de 48kHz. Los bits POR y BOR se usan para monitorear si se ha aplicado un reset al equipo.

Los registros TRISA y TRISB sirven para determinar si las terminales de los puertos A y B se utilizarán como entrada o como salida de señal. Para hacerlo, se coloca un número binario en este registro, y las posiciones con un “1” quedarán como entradas, mientras las que tengan un “0” serán salidas. Esto se puede cambiar durante la ejecución del programa, pero normalmente se determina desde el principio de la programación cómo se usarán las terminales, y se configuran al inicio del programa.

Para leer o escribir algún dato a las terminales de los puertos, se tienen los registros PORTA y PORTB; lo que significa que si se desea poner en un nivel alto alguna terminal de un puerto, bastará con que se escriba un “1” en la posición correspondiente de este puerto, y eso se reflejará de inmediato en la salida del dispositivo. Lo mismo se puede decir de la entrada; si se desea saber si en una cierta terminal hay un “1” o un “0”, simplemente se lee el bit correspondiente del PORTA o PORTB, y con eso se sabrá qué está llegando a esa terminal.

Todos los registros posteriores son de aplicación específica para los bloques periféricos internos del micro; así hay registros para los temporizadores 1 y 2, para los comparadores, para el transmisor USART, etc. Estos registros sólo se utilizarán cuando sea necesario aprovechar estos bloques funcionales, lo cual normalmente sólo se requiere en aplicaciones complejas.

Finalmente, existe un registro muy importante, el de configuración inicial, en el cual se fijan las condiciones operativas del dispositivo. Por ejemplo, aquí se indica si se activará el watchdog timer, si se protegerá el código grabado en

el micro, qué tipo de reloj usará el dispositivo (interno, externo, RC, cristal, etc.) y otros aspectos fundamentales para la correcta operación del controlador. Este registro se debe configurar al inicio del programa.

Estos son los registros especiales que posee este micro. Se deben tener siempre presentes, ya que se utilizarán constantemente durante el desarrollo del código interno del procesador.

ACTIVIDAD DE APRENDIZAJE 3D

Contesta lo siguiente:

- 1.- ¿Cómo se dividen los comandos del set de instrucciones de los PIC?
- 2.- Menciona dos comandos relacionados con los bits dentro de los registros.
- 3.- Menciona tres comandos que sirvan para hacer operaciones con dos registros.
- 4.- ¿Cuáles son los comandos para llamar una subrutina y para volver de ella?
- 5.- ¿En cuántos bloques está dividida la memoria del 16F628A?
- 6.- ¿Dónde se encuentran los bits C, DC y Z?
- 7.- ¿Para qué sirven los registros TRISA y TRISB?
- 8.- ¿Cómo se configura en estos registros si una terminal servirá como entrada o salida?
- 9.- ¿En qué registro se coloca la información que va o viene del puerto A?
- 10.- ¿Para qué sirve el registro de configuración inicial?

3.5 PRIMER PROGRAMA EN ENSAMBLADOR

Ahora, se comenzará con el desarrollo del primer programa para un microcontrolador PIC 16F628A, y se iniciará con un programa muy simple: que sólo encienda o apague algunos LED para simular la operación de un semáforo que controla dos calles.

Antes de iniciar, es conveniente que visitar la página de Microchip (www.microchip-com) y descargar la herramienta de programación que esta

empresa ofrece a sus consumidores: el programa MPLAB; se debe instalar en la computadora, para tenerlo siempre a la mano, ya que en él se basarán las explicaciones posteriores. Esto no es estrictamente necesario, ya que el programa en sí se puede hacer en cualquier editor ASCII, como el Notepad que incluye Windows; sin embargo, el paso final de pasar el programa de lenguaje ensamblador a lenguaje de “unos y ceros” del micro sí se tiene que hacer con esta herramienta, así que es mejor tenerla disponible.

Página principal de Microchip; abajo a la izquierda está el enlace para descargar el programa MPLAB. (Cortesía de Microchip).

Existen otras opciones. Si ya se dominan lenguajes de programación de alto nivel, como Basic o C, existen versiones de éstos dedicadas exclusivamente para el PIC, algunos incluso con versiones de prueba gratuitas, aunque la mayoría de las empresas productoras suele obsequiar una versión limitada, y sólo libera todas sus funciones al adquirir la licencia correspondiente. Por el momento, como lo que se desea es saber cómo se programa en

ensamblador, las explicaciones se concentrarán estrictamente en el programa MPLAB.

A continuación, se creará en el Notepad un archivo que llevará por nombre SEMAFORO.ASM; donde se escribirá el código del semáforo que se desea implementar con el PIC 16F628A. En seguida se muestra este código ya depurado:

**; En primer lugar, hay que fijar el microprocesador empleado y
; llamar a la librería que nombra sus registros y puertos.**

```
LIST P=16F628A  
#INCLUDE <P16F628A.INC>
```

**; Este archivo está en el mismo directorio del MPLAB IDE, y ahí se
; fija la posición de los registros, para que en lugar de tener que
; poner "escribe en el registro XXX", se pueda poner "escribe en
; el registro STATUS"; simplificando la programación. También se
; nombran los puertos y sus terminales individuales. Trata de abrir
; con el notepad este archivo P16F628.INC, para que veas su
; contenido.**

**; Se utilizará el generador de reloj interno, dejando todas las
; terminales como salidas.**

```
__CONFIG _CP_OFF & DATA_CP_OFF & _LVP_OFF & _MCLRE_OFF &  
_WDT_OFF & _BODEN_OFF & _PWRTE_OFF & _INTOSC_OSC_NOCLKOUT
```

; Se definen las variables necesarias para el programa:

```
TIEMPO EQU 20H  
CONT1 EQU 21H  
CONT2 EQU 22H  
CONT3 EQU 23H
```

**; Esta línea fija la dirección 0000H como el punto de inicio del
; programa.**

```
ORG 0000H
```

**; Teóricamente, a continuación tendríamos que poner un salto, ya que
; la dirección por default que busca el micro cuando recibe una**

; interrupción es la 0004H, pero como en este proyecto no se usan
; las interrupciones, podemos seguir con el programa.

; Se ponen todas las terminales del puerto B como salidas, y se
; colocan en ceros. Nota que dado que el registro TRISB está en el
; banco de memoria 1, hay que configurar el bit RP0 del registro
; STATUS para que pueda ser modificado.

```
MOVLW    b'00000000'  
BSF      STATUS, RP0  
MOVWF    TRISB  
BCF      STATUS, RP0  
MOVWF    PORTB
```

; Ahora vamos a implementar la subrutina de espera SEG; para ello
; vamos a colocar tres contadores anidados uno dentro de otro, dos
; de 255 a cero y uno de 15 a cero; con esto se consigue un retraso ; de poco
más de 1 segundo.

```
SEG      MOVLW    b'11111111'  
        MOVWF    CONT1  
        MOVWF    CONT2  
        MOVLW    b'00001111'  
        MOVWF    CONT3  
SEGA     DECFSZ   CONT1, 1  
        GOTO     SEGA  
        DECFSZ   CONT2, 1  
        GOTO     SEGA  
        DECFSZ   CONT3, 1  
        GOTO     SEGA  
        RETURN
```

; Se enciende Verde1 y Rojo2, se coloca un indicador en este punto

```
PUNTO1   MOVLW    b'00100001'  
        MOVWF    PORTB
```

; Se carga la cantidad de tiempo en el registro respectivo

```
MOVLW    b'00101000'  
MOVWF    TIEMPO
```

; Se llama a la subrutina de retraso de SEG. Se coloca un marcador
; en este punto. Comienza a decrementarse el registro TIEMPO, cuando

; llegue a cero, se hace un salto.

```
PUNTO2          CALL          SEG  
                DECFSZ      TIEMPO, 1
```

; Si aún no se llega a cero, se regresa al PUNTO2

```
                GOTO          PUNTO2
```

; Cuando se llega a cero, inicia el parpadeo de Verde1

```
                MOVLW        b'00100000'  
                MOVWF        PORTB  
                CALL         SEG  
                MOVLW        b'00100001'  
                MOVWF        PORTB  
                CALL         SEG  
                MOVLW        b'00100000'  
                MOVWF        PORTB  
                CALL         SEG  
                MOVLW        b'00100001'  
                MOVWF        PORTB  
                CALL         SEG  
                MOVLW        b'00100000'  
                MOVWF        PORTB  
                CALL         SEG  
                MOVLW        b'00100001'  
                MOVWF        PORTB  
                CALL         SEG  
                MOVLW        b'00100000'  
                MOVWF        PORTB  
                CALL         SEG
```

**; Después de los tres parpadeos, se enciende Amarillo1 por 3
; segundos**

```
                MOVLW        b'00100010'  
                MOVWF        PORTB  
                CALL         SEG  
                CALL         SEG  
                CALL         SEG
```

; Se apaga amarillo1 y rojo 2, se encienden verde2 y rojo1

```
MOVLW      b'00001100'  
MOVWF      PORTB
```

; Se repite todo el proceso anterior, pero ahora manteniendo
; encendido rojo1

```
MOVLW      b'00101000'  
MOVWF      TIEMPO
```

; Se llama a la subrutina de retraso de SEG. Se coloca un marcador
; en este punto. Comienza a decrementarse el registro TIEMPO, cuando
; llegue a cero, se hace un salto.

```
PUNTO3     CALL      SEG  
           DECFSZ    TIEMPO, 1
```

; Si aún no se llega a cero, se regresa al PUNTO3

```
GOTO       PUNTO3
```

; Cuando se llega a cero, inicia el parpadeo de Verde2

```
MOVLW      b'00000100'  
MOVWF      PORTB  
CALL       SEG  
MOVLW      b'00001100'  
MOVWF      PORTB  
CALL       SEG  
MOVLW      b'00000100'  
MOVWF      PORTB  
CALL       SEG  
MOVLW      b'00001100'  
MOVWF      PORTB  
CALL       SEG  
MOVLW      b'00000100'  
MOVWF      PORTB  
CALL       SEG  
MOVLW      b'00001100'  
MOVWF      PORTB  
CALL       SEG  
MOVLW      b'00000100'  
MOVWF      PORTB  
CALL       SEG
```

**; Después de los tres parpadeos, se enciende Amarillo2 por 3
; segundos**

```
MOVLW          b'00010100'  
MOVWF         PORTB  
CALL          SEG  
CALL          SEG  
CALL          SEG
```

; Se regresa al inicio para encender verde1 y rojo2 y repetir el ciclo.

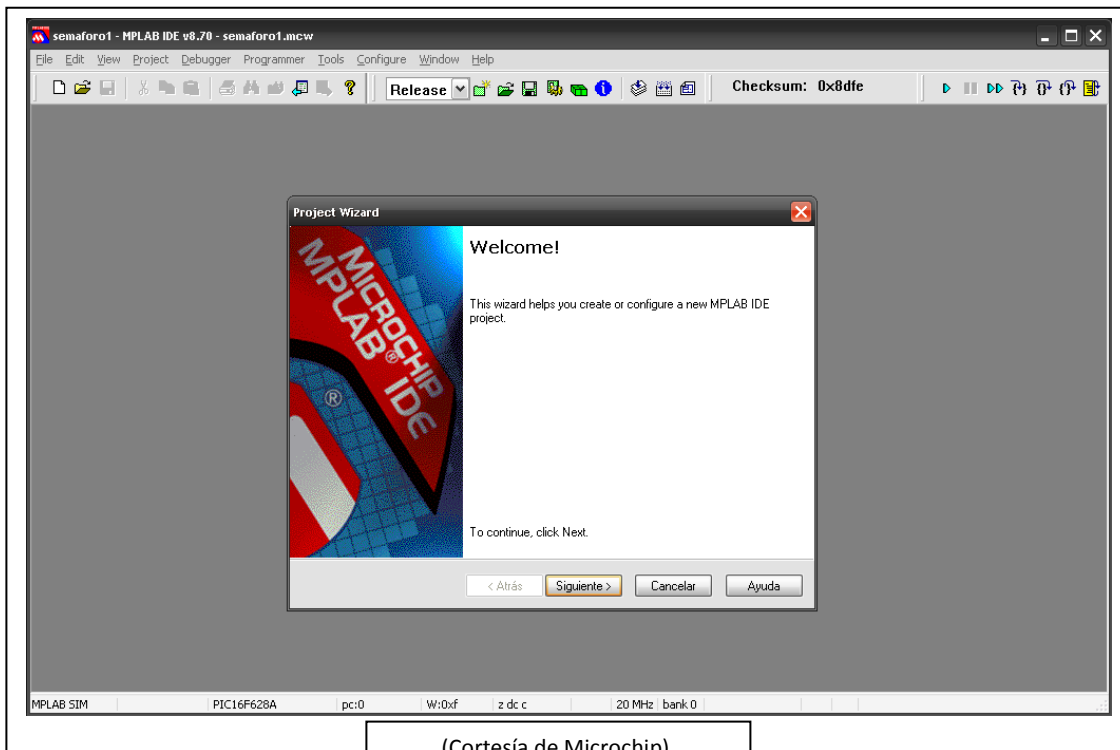
```
GOTO          PUNTO1
```

; Fin del programa

```
END
```

Se debe observar que durante todo el programa, se fueron colocando abundantes notas para saber a cada paso exactamente qué se estaba haciendo. Esto es conveniente para reaprovechar códigos en proyectos futuros, y así ahorrar trabajo al realizar nuevos programas.

Ya con el código correctamente escrito, se debe llevar al programa MPLAB-IDE; para ello, en el menú PROJECT se activa el Project Wizard; se obtendrá lo que se observa en la figura.

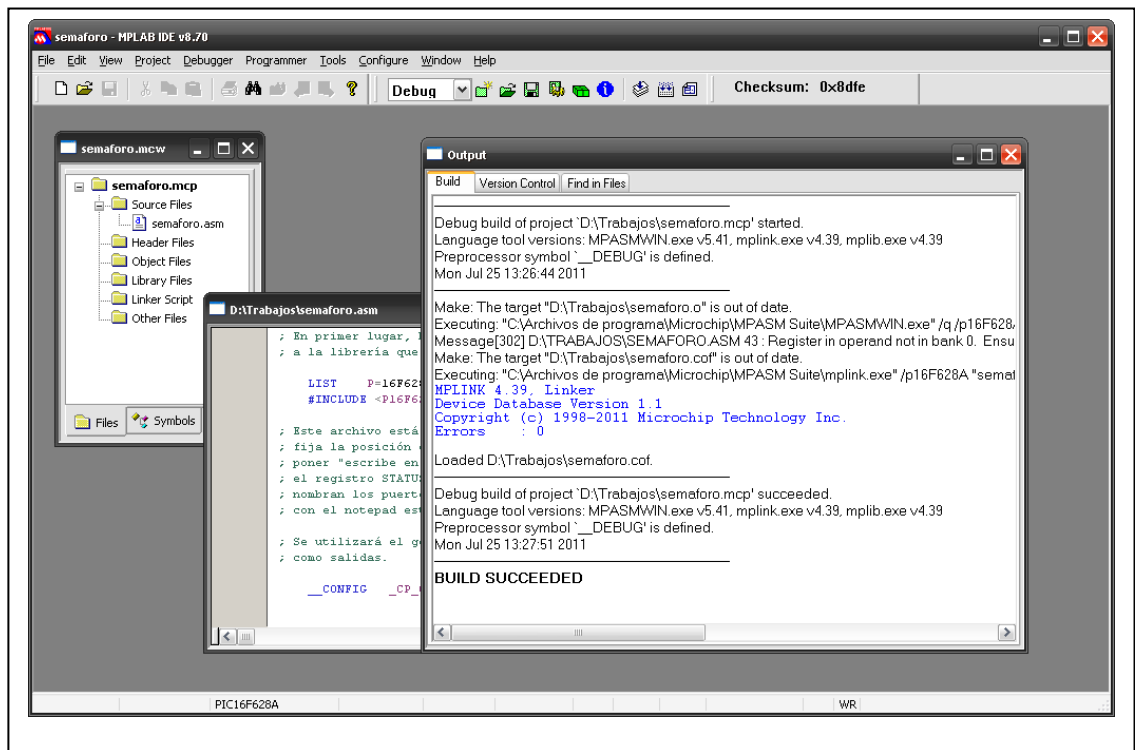


Se deberán pasar varias opciones, como elegir el PIC que se utilizará, elegir el ambiente de trabajo, cargar el archivo con el programa (aquí hay que elegir el recién creado SEMAFORO.ASM), etc. Al terminar, en el directorio elegido aparece un archivo SEMAFORO.MCP y otro SEMAFORO.MCW, que es el proyecto creado por el MPLAB.

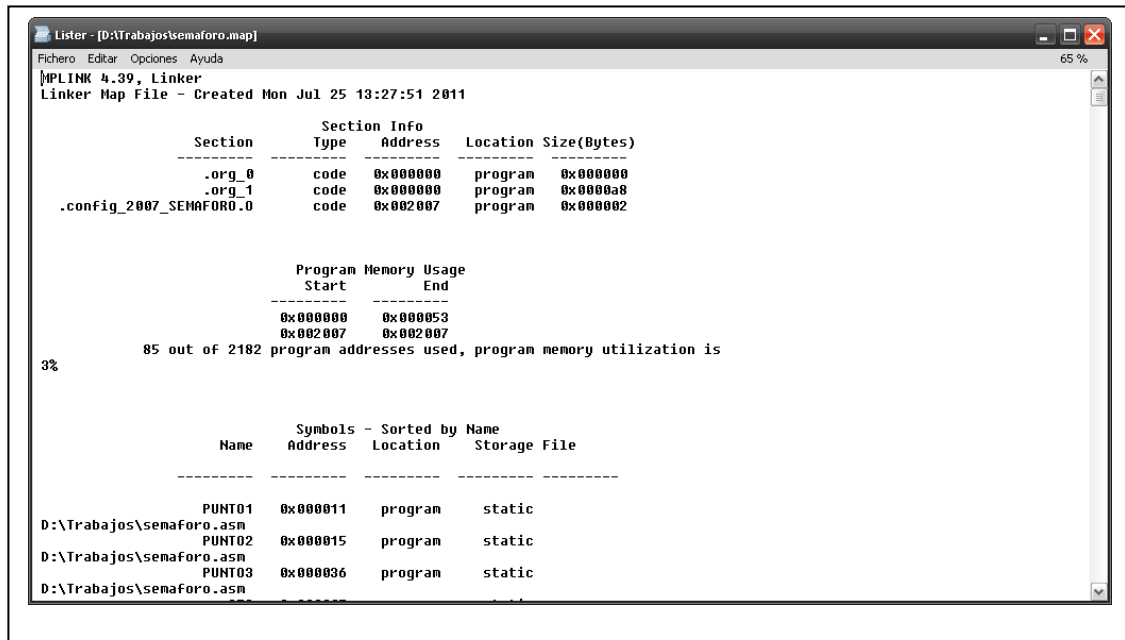
Una vez creado el proyecto, hay que revisar si todo está correcto; para ello, en la barra de tareas del programa deberá estar activada la opción DEBUG, y luego accionar la opción MAKE a la derecha del círculo azul con una "i"; si el programa no tiene errores, aparecerá un mensaje como el que se ve en la figura siguiente. Si aparece el mensaje BUILD SUCCEEDED, eso significa que el código no tiene errores, si existiera alguno, aparecerá un mensaje BUILD FAILED, lo que significa que hay que depurar el código.

Para facilitar esta tarea, el programa crea un archivo de error, donde se puede encontrar algo como lo siguiente:

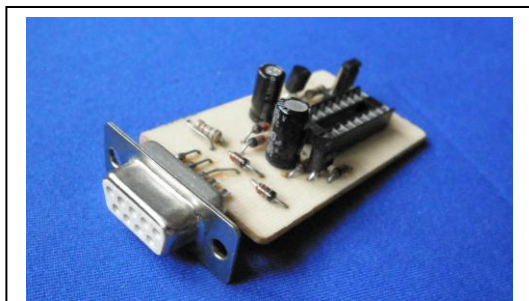
Error[128] D:\TRABAJOS\SEMAFORO.ASM 13 : Missing argument(s)
Error[128] D:\TRABAJOS\SEMAFORO.ASM 13 : Missing argument(s)
Message[302] D:\TRABAJOS\SEMAFORO.ASM 37 : Register in operand not in bank 0. Ensure that bank bits are correct.



Esto indica que hay un error en la línea 13, y que se debe verificar la condición indicada en la línea 37. Un error impedirá que el proyecto se compile, un mensaje o una advertencia no.



Una vez depurado el código, y que no existan errores, en el directorio de trabajo se creará un archivo SEMAFORO.HEX, que es precisamente lo que se cargará en el microcontrolador. También aparece un archivo SEMAFORO.MAP, donde se detalla el uso de la memoria que hace el programa (en la figura hay un fragmento de este archivo, donde claramente indica que apenas se usaron 85 de las 2182 localidades de memoria disponibles), lo que es un indicador más de la sencillez de este proyecto.

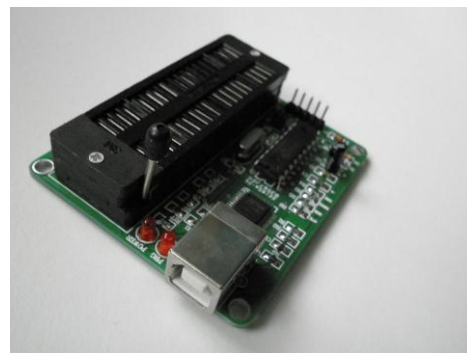


Programador JDM casero; requiere de un puerto serial para funcionar.

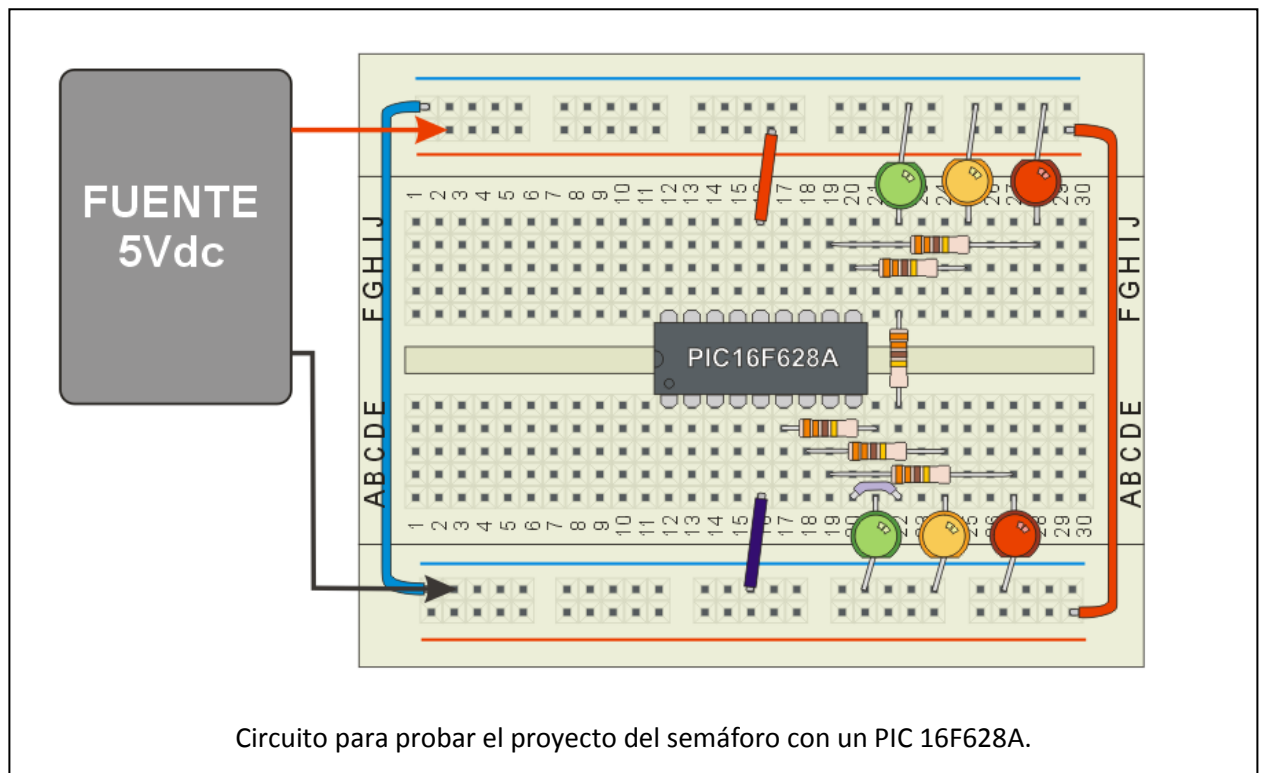
Cuando ya se tiene el archivo HEX, se debe cargar en el microcontrolador; para ello, se puede adquirir alguno de los programadores y emuladores que ofrece la misma empresa Microchip, aunque suelen ser algo costosos; sin embargo, circulan por Internet una gran cantidad de proyectos

con los cuales se puede construir un programador muy económico. Uno de los más populares es el JDM, que se muestra en la foto (armado por el autor), que tiene la ventaja de que no necesita de fuente adicional, ya que se alimenta directamente del puerto serial de la computadora. Si la computadora no posee puertos seriales, entonces lo mejor será adquirir un programador vía USB como el que se muestra, aunque esto sí implica una inversión un poco más elevada.

Para la carga del programa hacia el PIC, en el caso del programador JDM, es necesario utilizar algún programa capaz de comunicarse con él, los más populares son el IC-Prog o el Pony-Prog. En el caso del programador USB, seguramente está acompañado con su programa de carga exclusivo, así que ese es el que se debe utilizar.



Programador de PICs USB.



Circuito para probar el proyecto del semáforo con un PIC 16F628A.

Cuando el programa se haya cargado en la memoria flash del PIC, se deberá probar. En este caso, hay que conectar el micro como se muestra en la figura anexa; se observa que se han conectado los LED de colores a las salidas del puerto B, verdes para RB0 y RB3, amarillos para RB1 y RB4, y rojos para RB2 y RB5; RB6 y RB7 no tienen conexión, lo mismo que las terminales del puerto A. Lo único adicional es la entrada de voltaje y tierra, que alimentan al dispositivo.

Si todo se realizó de forma correcta, al momento de aplicar un voltaje de 5Vdc a la tablilla de pruebas, los LED comienzan a encenderse con la secuencia de un par de semáforos; ¡y todo esto se consiguió con un chip y unas cuantas líneas de código!

ACTIVIDAD DE APRENDIZAJE 3E

Contesta lo siguiente:

- 1.- ¿En dónde se puede conseguir el programa MPLAB?
- 2.- ¿Es necesario programar un PIC en ensamblador?
- 3.- ¿Se puede elaborar un programa en ensamblador sin contar con el MPLAB?
- 4.- ¿Por qué se coloca la instrucción INCLUDE al principio del programa?
- 5.- ¿Por qué conviene fijar las variables al principio del código?
- 6.- ¿Qué precaución se debe tener al configurar los registros TRISA y TRISB?
- 7.- ¿Por qué hay que colocar marcadores a lo largo del programa?
- 8.- ¿Qué hay que hacer si al ejecutar un debug en el programa, aparecen errores?
- 9.- ¿Para qué sirve el archivo HEX que genera el MPLAB?
- 10.- ¿Cómo se carga el programa en el dispositivo?

AUTOEVALUACIÓN

1. Describe cuál es la principal diferencia entre un microcontrolador de arquitectura Von Neumann y uno de arquitectura Harvard.
2. Describe los bloques periféricos incorporados dentro del PIC16F628A, con una breve explicación de la utilidad de cada uno.
3. Al considerar que este dispositivo sólo utiliza 35 comandos en su set de instrucciones, ¿es de tecnología CISC o RISC?
4. ¿De cuántos bits es la instrucción típica de un microcontrolador de la familia PIC 12-16? ¿Qué ventaja tiene esto?
5. Menciona las órdenes que se deben indicar para modificar un bit individual dentro de un registro.
6. ¿Qué precaución hay que tomar cuando se configuran los registros TRISA y TRISB? ¿Por qué?
7. En operaciones con uno o dos bytes, ¿qué significa el modificador final?
8. ¿Por qué es tan importante el registro W?
9. ¿Cuál es el archivo más importante que se genera con el MPLAB?
10. Menciona dos programas que se usan normalmente para cargar el código de un PIC usando el programador JDM.

RESPUESTAS

1. En un microprocesador Von Neumann, la memoria y el programa comparten el mismo espacio, mientras que en una arquitectura Harvard, son bloques independientes.
2. Puertos I/O: para entrada y salida de señales en general; temporizadores: para medir tiempos; comparadores: para comparar señales analógicas con un voltaje de referencia; USART: para comunicaciones seriales; CCP1: para expedir, recibir y/o comparar señales PWM; EEPROM: para guardar datos semipermanentes.
3. Es tecnología RISC, esto es, computación por set de instrucciones reducido.
4. De 14 bits, lo que le permite combinar en una sola línea tanto el comando como el dato con que se trabajará.
5. [BCF REG1, x] o [BSF REG1, x]; donde REG1 es el nombre del registro, y x el número de bit que se desea modificar.
6. Hay que asegurarse de activar el bit RP0, para que el micro busque en su segundo banco de memoria. No se debe olvidar desactivarlo después de la orden.
7. Indica la dirección hacia donde irá el resultado de la operación; si se coloca un "1", el resultado irá al mismo registro; si se coloca un "0", irá al registro W.
8. Es el registro de trabajo, donde se cargan prácticamente todos los datos que se van a procesar por medio de la ALU.

9. El archivo HEX, ya que es éste el que finalmente se cargará en el micro.

10.IC-Prog y Pony-Prog.

Soluciones a las actividades de aprendizaje:

Actividad de aprendizaje 3A:

1. Microchip.
2. De 8 bits.
3. Harvard.
4. Cómo manejan su memoria de programación y datos.
5. Que se puede reescribir las veces que sea necesario, para corregir errores o usar el micro para otro proyecto.
6. 20 MHz con un cristal externo.
7. No, posee su propio oscilador de 4MHz o de 48kHz.
8. 3, 2 de 8 bits y uno de 16 bits.
9. Sí, dos comparadores independientes.
10. Para guardar datos que necesiten mantenerse incluso si se retira la energía del micro.

Actividad de aprendizaje 3B:

1. Unidad aritmética lógica, y es la encargada de hacer las operaciones lógicas dentro del micro.
2. Hasta 16 terminales repartidas en dos puertos.
3. Sólo uno, el registro W.
4. PC sirve para llevar el contador de programa, y Stack para guardar la ubicación en que estaba el PC antes de llamar a una subrutina.
5. De 14 bits.
6. Transmisor-Receptor Universal Síncrono-Asíncrono, y sirve para comunicarse de forma serial con otros dispositivos.
7. Desde 2 hasta 5.5 voltios.
8. Es el generador, receptor y comparador de señales PWM.
9. Vigilar que el programa se esté ejecutando adecuadamente.

10. Ahí se tienen los registros y las variables, para manejar ahí los datos que necesiten estar cambiando de manera constante durante la operación del dispositivo.

Actividad de aprendizaje 3C:

- 1.- Sólo 35 comandos individuales.
- 2.- Cuatro modos de direccionamiento.
- 3.- Direccionamiento inmediato, directo, de registro a memoria y directo a registro.

Actividad de aprendizaje 3D:

1. En operaciones relacionadas con bytes, operaciones relacionadas con bits y operaciones con literales y de control.
2. BCF, BSF, BTFSC, BTFSS.
3. ADDWF, ANDWF, IORWF, SUBWF, XORWF.
4. CALL y RETURN o RETLW
5. En cuatro bloques, aunque los más importantes son los dos primeros.
6. En el registro STATUS.
7. Para determinar si las terminales de los puertos A y B serán entradas o salidas.
8. Se escribe una cifra en TRISA o TRISB; donde haya un "1", será entrada, un "0" será salida.
9. En el registro PORTA.
10. Para fijar condiciones operativas fundamentales, como el tipo de reloj empleado, si se protegerá el código interno, etc.

Actividad de aprendizaje 3E:

1. En la página de Microchip: www.microchip.com.
2. No, también se puede hacer en Basic o C, pero se necesitan programas especiales.

3. No, se puede usar cualquier editor de textos ASCII, como el Notepad de Windows.
4. Para cargar una librería que ya trae el MPLAB, con las características principales del microcontrolador empleado.
5. Para evitar introducir la dirección exacta de cada variable, en vez de eso, se usa su nombre.
6. No olvidar activar el bit RP0 en el registro STATUS, y una vez configurado el puerto, desactivarlo otra vez.
7. Para marcar puntos hacia donde irán los saltos condicionales o incondicionales del programa.
8. Se debe consultar el archivo *.ERR, para localizar los fallos y corregirlos.
9. Es el archivo binario que finalmente se cargará en la memoria del microcontrolador.
10. Por medio de un programador externo, normalmente a través de puerto serial o USB.

UNIDAD 4

PROGRAMACIÓN DE ENTRADA/SALIDA

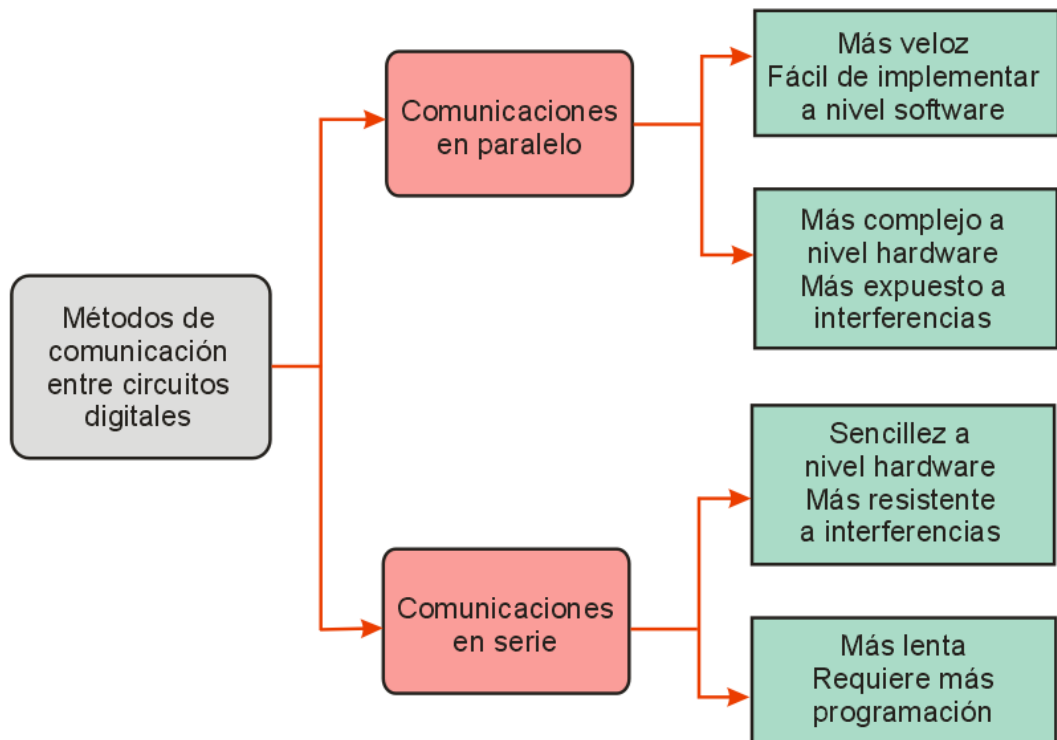
OBJETIVO

Identificar los dos métodos de comunicación más comunes en sistemas digitales, reconociendo sus ventajas y desventajas, así como la forma de implementarlos para utilizarse en diversos proyectos.

TEMARIO

- 4.1 COMUNICACIÓN DE DATOS EN FORMA PARALELA
- 4.2 PROGRAMACIÓN, CONTROL Y DIRECCIONAMIENTO DE LOS ELEMENTOS DE ENTRADA/SALIDA PARALELOS
- 4.3 COMUNICACIÓN DE DATOS EN FORMA SERIAL
- 4.4 PROGRAMACIÓN, CONTROL Y DIRECCIONAMIENTO DE LOS ELEMENTOS DE ENTRADA/SALIDA
- 4.5 PROGRAMAS DE APLICACIÓN

MAPA CONCEPTUAL



INTRODUCCIÓN

Como se expuso en la unidad anterior, los microcontroladores poseen en su interior prácticamente todos los bloques funcionales que requieren para realizar un trabajo determinado; sin embargo, cuando la aplicación requiere de una gran flexibilidad, lo mejor es utilizar un microprocesador genérico, y conectarle externamente los periféricos adecuados para el uso que se le dará. En estos casos, resulta indispensable el intercambio de información entre el micro y esos bloques periféricos, tanto para la expedición como para la recepción de datos diversos.

En circuitos digitales, cuando dos bloques necesitan intercambiar datos entre sí, normalmente se utilizan dos tipos de enlaces entre ellos: comunicación en paralelo y comunicación en serie; cada uno con ventajas y desventajas que hacen a uno u otro, adecuado para diversas aplicaciones. En esta unidad, se describirá



Cuando dos o más circuitos digitales necesitan intercambiar información entre sí, se establece un enlace entre ellos, que puede ser paralelo o serial.

(Cortesía Maxtor).

precisamente qué es una comunicación en paralelo y qué es una comunicación en serie, cuáles son sus ventajas y cuáles sus inconvenientes, así como la forma de implementar ambas en un microprocesador común (aunque se usará como práctica el mismo microcontrolador PIC de la unidad anterior). De este modo, se tendrán las bases para implementar soluciones basadas en procesadores digitales que requieran múltiples chips trabajando en conjunto, algo indispensable en aplicaciones realmente complejas.

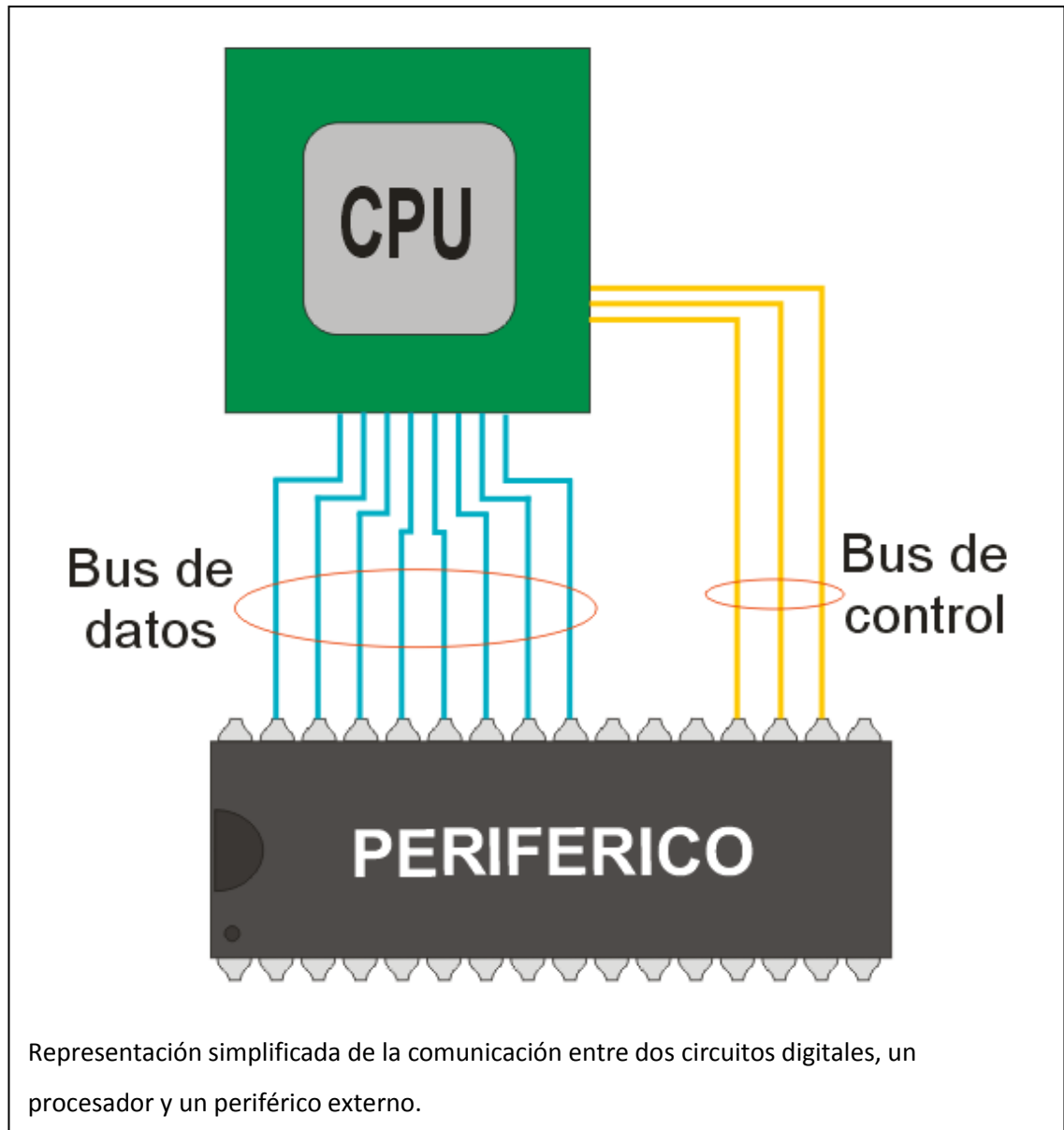
4.1 COMUNICACIÓN DE DATOS EN FORMA PARALELA

Aunque el uso de microcontroladores ahorra considerablemente la cantidad de chips empleados en un proyecto específico, existen aplicaciones muy complejas en las que resulta prácticamente imposible incluir en un solo circuito integrado todas las funciones que se requieren para cubrir cierto requerimiento; lo que obliga a colocar dos o más integrados trabajando en estrecha cooperación, intercambiando datos, repartiendo instrucciones, enviando y recibiendo información, etc.

Tradicionalmente, la forma en la que se realiza este intercambio de datos digitales es utilizando varias líneas de comunicación entre chips, y esta cantidad de líneas es equivalente a la cantidad de bits que maneja el dispositivo en cuestión; esto significa que un microprocesador de 8 bits usa 8 líneas de comunicación, uno de 16 bits usaría 16 líneas, y así sucesivamente (aunque esto no siempre se cumple). Debido a que en este tipo de comunicación digital, los bits viajan uno al lado de otro entre chips, se le ha dado el nombre de “comunicación paralela”; a continuación, se indica en qué consiste.

En la figura anexa, se muestran dos circuitos integrados, un procesador digital y un periférico externo, los cuales necesitan intercambiar información entre sí; para hacerlo, todas las líneas del bus de datos del procesador se conectan hacia el otro chip; de modo que cuando se establezca la comunicación, en cada ciclo de reloj puedan intercambiarse “N” bits entre ambos chips, donde “N” es el número de líneas que posee el bus de datos del procesador; sin embargo, aunque a través de estas líneas circula la información, es necesario añadir algunas líneas de control, que le indiquen al periférico el momento exacto en el que va a recibir o enviar sus datos, ya que incluso, si el intercambio de datos entre ambos chips es constante, habrá momentos en que el procesador desee comunicarse con otro periférico diferente, y para hacerlo, debe usar el mismo bus de datos. Esto implica que las líneas del bus de datos se comparten prácticamente con todos los integrados que forman un circuito lógico, y es a través de las líneas de control que el

procesador decide a cuál chip le estará enviando su información, o de cuál de ellos estará recibiendo datos en un momento dado.



Para lograr esto, todos los periféricos externos poseen una terminal que funciona como un “*chip enable*” o habilitador del chip, lo que significa que las terminales de ese integrado pueden estar conectadas al bus de datos general, y a través de este bus puede circular una gran cantidad de información, sin que ello afecte a este chip; sin embargo, cuando a este bloque llega la orden *chip enable*, en ese momento el chip entra en actividad, establece comunicación con

el procesador principal, recibe o envía la información, y finalmente vuelve a su estado de reposo al deshabilitarse la señal *chip enable*.



Esta señal se transmite a los periféricos a través del bus de control y del bus de direcciones, ya que por lo general, no todas las posibles combinaciones de este último se usan para direccionar localidades de memoria, también se apartan algunas para determinar hacia qué chip se estará canalizando una

comunicación. Se debe señalar, que casi todos los microprocesadores más poderosos y modernos sí tienen un bus de direcciones dedicado especialmente para la memoria, y otro para el direccionamiento de sus periféricos.

Para analizar con más detalle cómo se lleva a cabo esta comunicación, en el siguiente tema, se simularán algunos casos típicos en los que un micro tiene que comunicarse con algunos chips periféricos, y qué señales se tienen que intercambiar entre sí para lograr un enlace correcto.

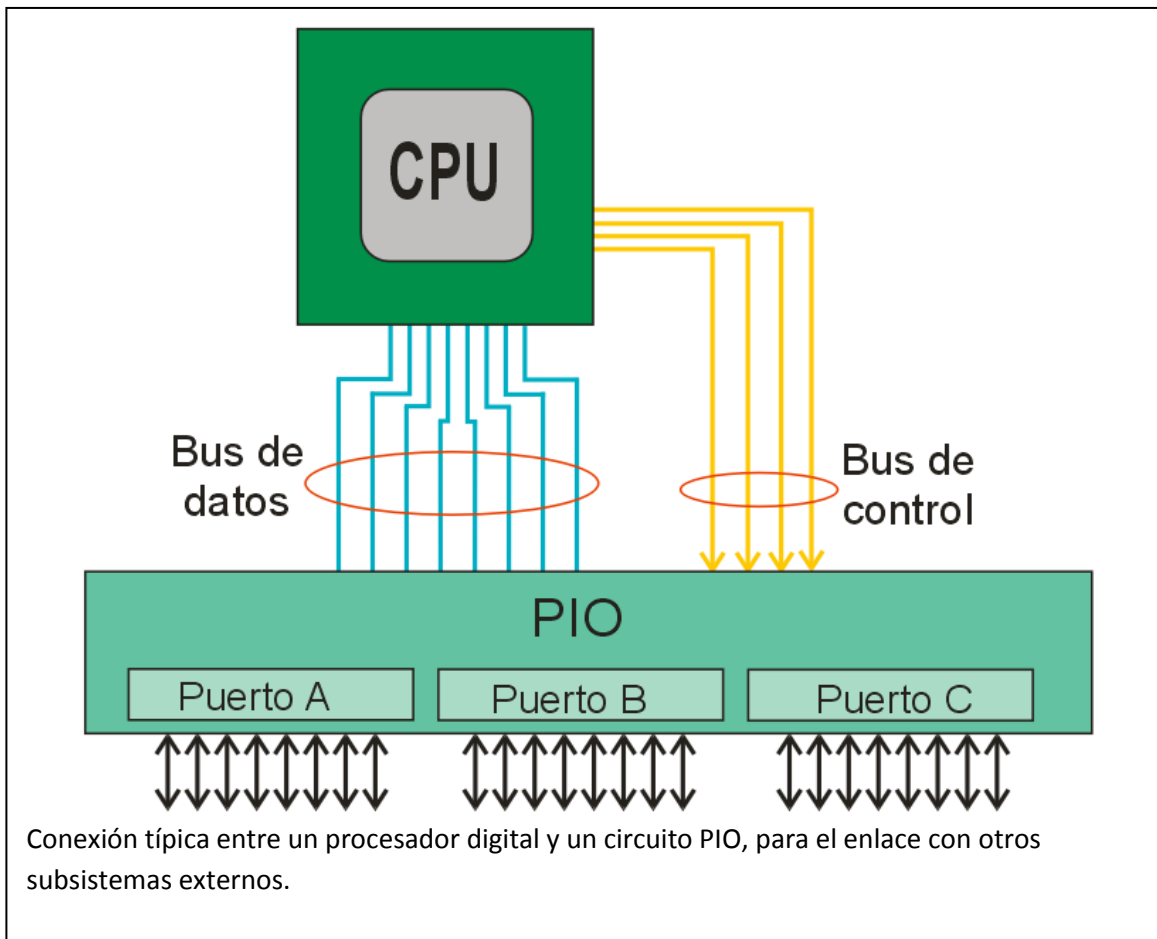
ACTIVIDAD DE APRENDIZAJE 4A

Contesta lo siguiente:

- 1.- ¿Cómo intercambian información dos o más circuitos digitales dentro de un circuito?
- 2.- ¿Cuáles son los dos tipos de comunicación más empleados en circuitos digitales?
- 3.- ¿Cuál es la característica principal de la comunicación en paralelo?
- 4.- ¿Por qué en ciertas aplicaciones conviene usar un microprocesador genérico y rodearlo de periféricos diversos?
- 5.- En un microprocesador típico de 8 bits, ¿cuántos bits tiene su bus de datos?
- 6.- ¿Para qué sirve la señal *chip enable*?
- 7.- ¿Qué hace el bus de control durante una comunicación en paralelo?
- 8.- En un microprocesador sencillo, ¿todas las opciones del bus de direcciones se usan para memoria?

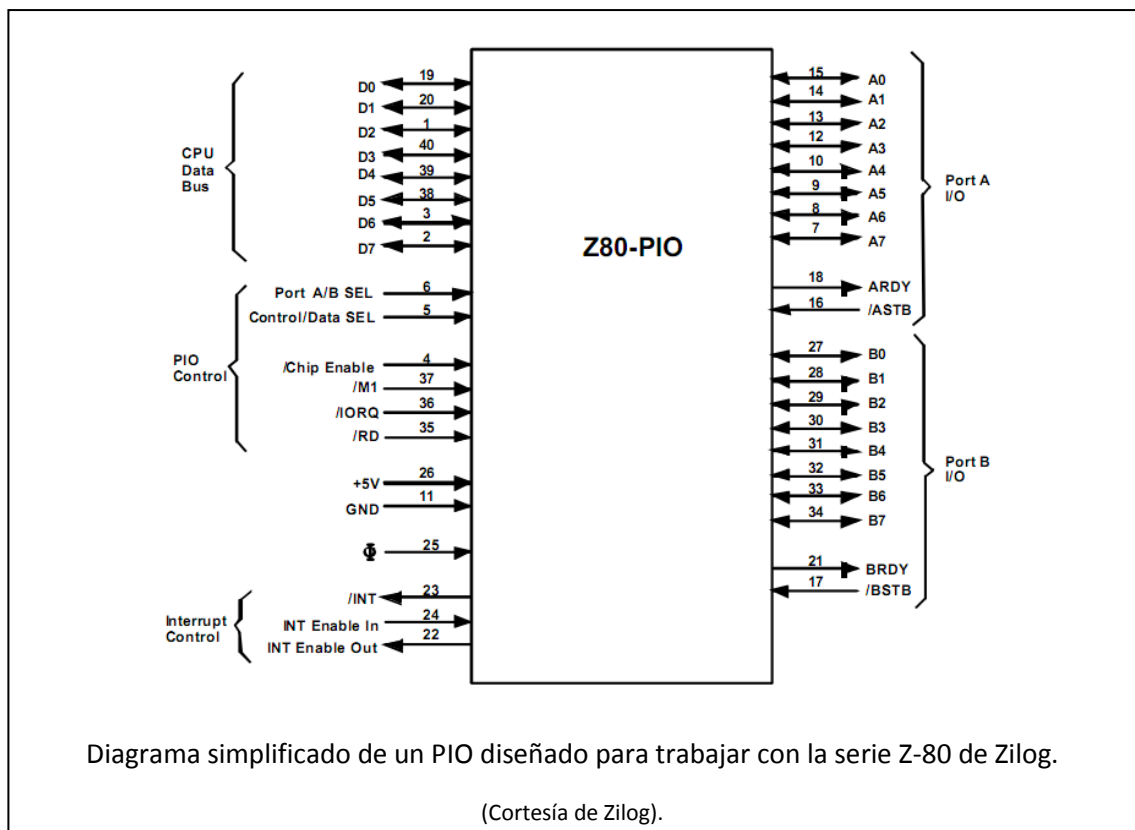
4.2 PROGRAMACIÓN, CONTROL Y DIRECCIONAMIENTO DE LOS ELEMENTOS DE ENTRADA/SALIDA PARALELOS

A continuación, se describirá con más detalle cómo se hace una comunicación paralela entre un procesador digital y algún periférico externo; para ello, se considerará un microprocesador genérico y se enlazará con un chip muy especial, que seguramente se podrá encontrar en diversos proyectos digitales: un PIO, siglas de *Parallel Input/Output* o Entrada/Salida Paralela. Estos chips suelen ser circuitos integrados relativamente grandes, de aproximadamente 32 a 40 terminales; y en ellos se encuentran dos o tres puertos I/O para expedir o introducir datos, desde y hacia el micro. A continuación, se supondrá tener un chip de 40 terminales con 3 puertos I/O, los cuales pueden funcionar como entrada o salida de datos, según lo ordene el procesador. La figura anexa se tomará como base.

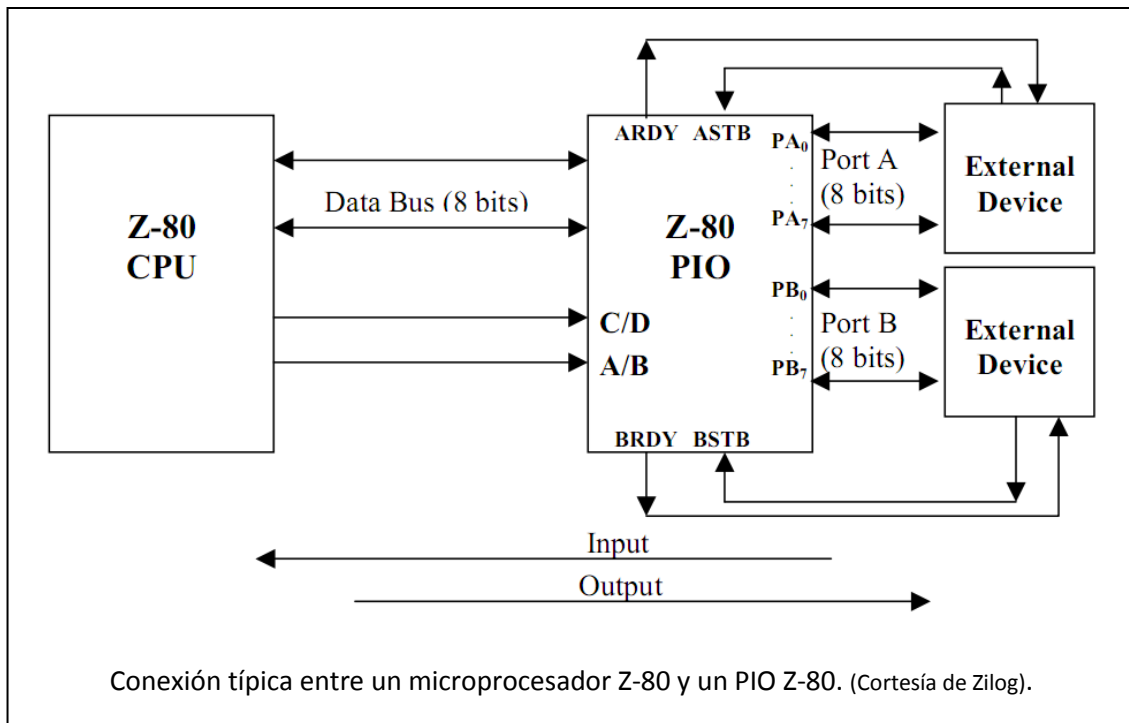


Si el PIO posee tres puertos independientes, de los cuales el micro puede recibir información o también puede enviarla (dependiendo de cómo se configure el puerto), es obvio que ya no basta una simple señal de *chip enable* para lograr manejar la comunicación entre este periférico y el CPU. Se necesitan por lo menos tres señales adicionales para que la comunicación entre ambos chips sea la adecuada:

- Un par de líneas *port select* o selección de puerto, que le indiquen al PIO cuál de los puertos se utilizará en un momento dado. En el ejemplo, se podrían tener dos líneas programadas de la siguiente forma: si en ambas líneas se tiene 00, el chip no hace nada; con un 01, activa al puerto A, con un 10 al puerto B y con un 11 al puerto C; es obvio entonces que con estas dos líneas se está eligiendo qué puerto se va a utilizar en un momento dado; de ahí su nombre.
- Una línea *write/read* o escribe/lee, que como su nombre lo indica, configurará al puerto respectivo para que funcione como entrada de datos (*read*) o como salida (*write*). En el ejemplo, se podría hacer que la línea indique que el puerto funcionara como entrada al colocarse en "1", y como salida al estar en "0".



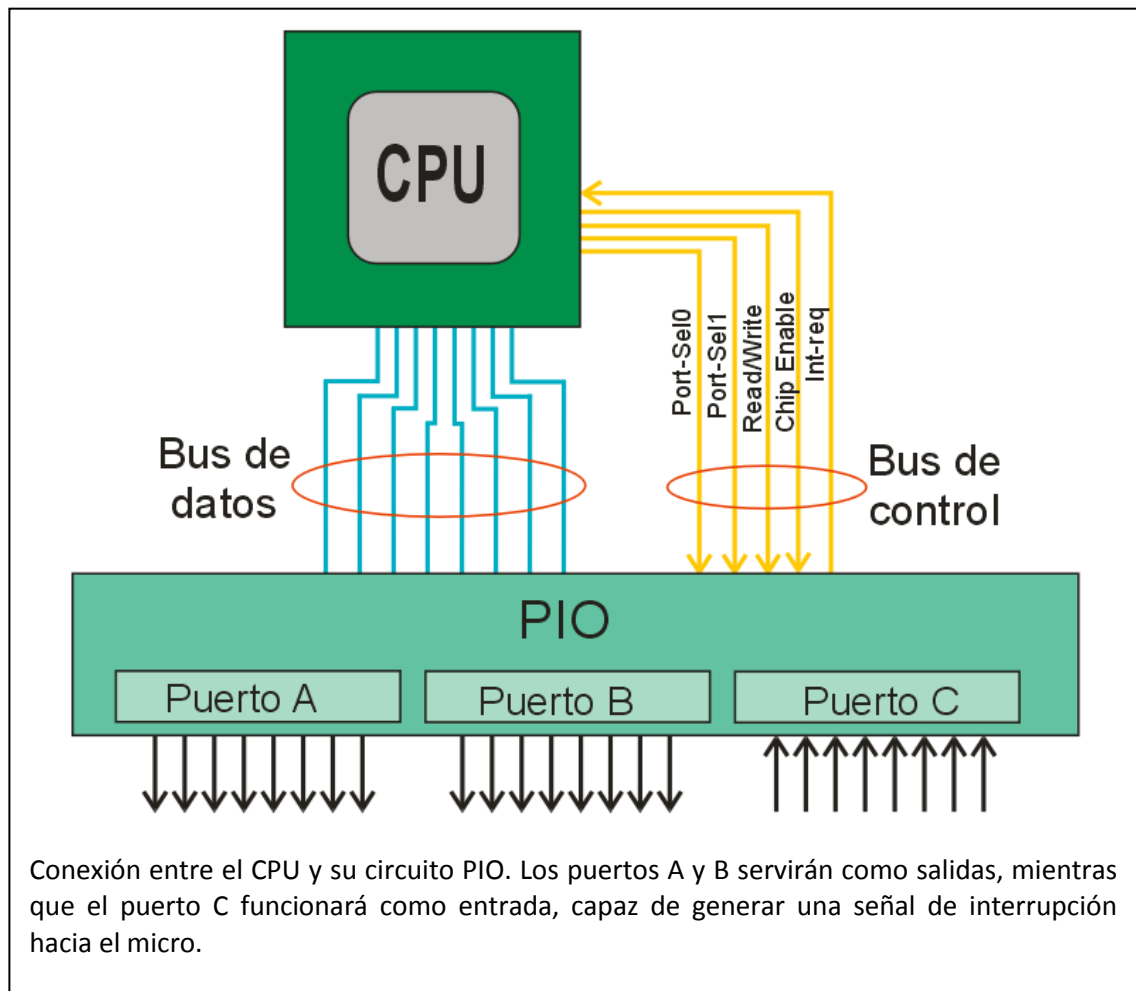
Entonces, añadiendo la línea de *chip enable*, se necesita un mínimo de cuatro líneas de control, además de las ocho líneas del bus de datos, para garantizar un intercambio de información adecuado entre el CPU y su circuito PIO. Sólo como referencia, se puede observar en la figura anexa, un diagrama simplificado de un chip PIO para la serie Z-80 de Zilog, y enseguida un diagrama de cómo se conecta este chip tanto con el micro como con sus circuitos externos.



Esto significa que, si se tiene un procesador digital, y se desea conectar con elementos periféricos utilizando el bus de datos, de control y de direcciones, se deberán programar en el código interno del micro, las instrucciones necesarias para expedir por las líneas del bus de direcciones la combinación precisa para que se establezca la comunicación con el chip deseado; después, generar por el bus de control las instrucciones necesarias para lograr el enlace entre el micro y el periférico, y finalmente, a través del bus de datos, enviar la información correspondiente o recibir los datos que esté enviando el chip.

Por ejemplo, usando un lenguaje de programación genérico, como el que se usó en la unidad dos para elaborar el primer programa en ensamblador (este lenguaje no corresponde a ningún micro, sólo se dan comandos que posteriormente podrán ser trasladados al lenguaje ensamblador del dispositivo que se empleará en un proyecto en particular).

De acuerdo con la figura del micro conectado a un PIO con tres puertos I/O; pero aumentando un poco la complejidad y con la posibilidad de que el PIO pueda generar una señal de interrupción hacia el procesador. Al suponer que dos de los puertos son salidas y uno funciona como entrada. A continuación se muestra el diagrama de esta nueva conexión:



A continuación, se muestra cómo programar el código para lograr la comunicación con estos puertos:

Para comunicarse con el puerto A:

Colocar PORT SEL 0-1 en b'01'	; se selecciona el puerto A
Colocar READ/WRITE en b'0'	; para que el bus de datos ; funcione como salida
Colocar CHIP ENABLE en b'1'	; se habilita el PIO
Colocar INFO en BUS DE DATOS	; inicia el envío de información

Para comunicarse con el puerto B:

Colocar PORT SEL 0-1 en b'10'	; se selecciona el puerto B
Colocar READ/WRITE en b'0'	; para que el bus de datos ; funcione como salida
Colocar CHIP ENABLE en b'1'	; se habilita el PIO
Colocar INFO en BUS DE DATOS	; inicia el envío de información

Para recibir datos del puerto C:

Colocar PORT SEL 0-1 en b'11'	; se selecciona el puerto C
Colocar READ/WRITE en b'1'	; para que el bus de datos ; funcione como entrada
Colocar CHIP ENABLE en b'1'	; se habilita el PIO
Colocar INFO en BUS DE DATOS	; inicia la recepción de datos

Si el PIO envía una interrupción a través de puerto C:

Colocar PORT SEL 0-1 en b'11'	; se selecciona el puerto C
Colocar READ/WRITE en b'1'	; para que el bus de datos ; funcione como entrada
Colocar CHIP ENABLE en b'1'	; se habilita el PIO
Colocar INFO en BUS DE DATOS	; inicia la recepción de datos
Llama a Subrutina INT	; se ejecuta el procedimiento

Return from interrupt ; de emergencia necesario
; regresa al programa normal

Obviamente, todo lo anterior tendría que programarse usando el lenguaje ensamblador adecuado, dependiendo del micro que se esté usando; por ejemplo, si se usara un PIC, y suponiendo que el puerto A es el que funciona como bus de datos y el puerto B como bus de control, el programa para enviar datos al puerto A del PIO quedaría más o menos así (suponiendo que el MCU ha sido configurado apropiadamente desde un inicio):

```

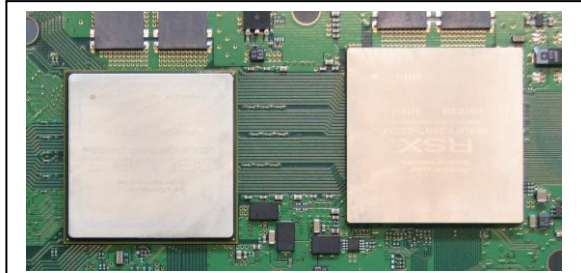
MOVWF    PORTA    ; se coloca en las
terminales
; del puerto A el dato que se
; quiere enviar
BSF      PORTB, PSEL0 ; Se pone en uno PSEL0
BCF      PORTB, PSEL1 ; se pone en cero PSEL1
BSF      PORTB, RW    ; se pone como salida PuertoA
BSF      PORTB, CEN   ; se habilita el PIO

```

De este modo, el contenido del registro W pasa a las terminales del puerto A, y de ahí llega hasta el PIO, para canalizarse hacia donde sea necesario. Obviamente, una vez enviado el dato, se debe desactivar el *chip enable* y dejar nuevamente en “ceros” las líneas de selección de puerto, para que todo quede listo para una nueva transmisión o recepción de datos.

Como se ha observado, en realidad no se requiere de demasiados pasos ni de procesos complejos para lograr que un microprocesador se comunique con algún chip periférico; lo único que se debe verificar es que las instrucciones que controlan este intercambio de datos sean siempre las correctas.

¿Cuáles son las ventajas y desventajas de la comunicación paralela? Su principal ventaja es que es muy fácil de implementar, además de que el intercambio de datos es muy rápido, ya que en pocos ciclos de reloj se mueven tantos bits como tenga el bus de datos del dispositivo, garantizando la máxima velocidad de transferencia entre el procesador y sus periféricos.



La comunicación paralela es la más veloz que existe en sistemas digitales, por lo que se usa para comunicar circuitos que requieran intercambiar datos con rapidez.

El principal inconveniente es que se necesitan muchas líneas de comunicación entre el procesador y su circuito periférico, y si ambos están relativamente alejados, esto complica el diseño de las placas de circuito impreso; además, la comunicación en paralelo está mucho más expuesta a interferencias externas, así que en aplicaciones donde es vital que la información fluya sin errores, es necesario implementar algoritmos de detección y corrección de fallas en el enlace, lo que complica aún más el diseño general.

A pesar de esto, la comunicación paralela es de las que más se utiliza en el proceso lógico de señales, y es la que seguramente se utilizará en los primeros proyectos que se diseñen, en los que haya dos o más circuitos digitales interactuando entre sí.

ACTIVIDAD DE APRENDIZAJE 4B

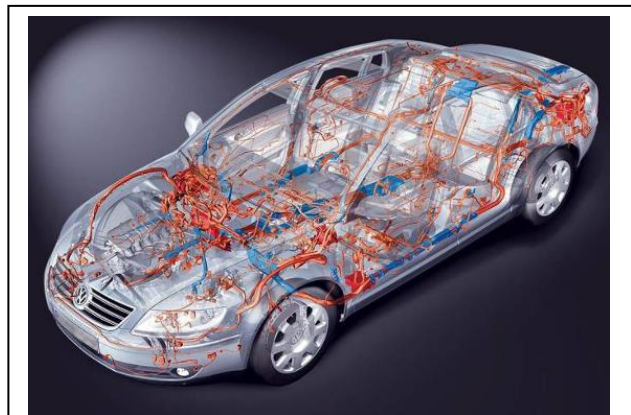
Contesta lo siguiente:

- 1.- ¿Qué significa PIO?
- 2.- ¿Para qué sirven estos circuitos periféricos?
- 3.- ¿Se pueden configurar los puertos de un PIO como entrada o salida, según la aplicación?
- 4.- ¿Qué otras señales de control se requieren para obtener una comunicación adecuada entre el microprocesador y el PIO?
- 5.- ¿Es posible manejar interrupciones a través de un PIO?
- 6.- Describe de forma general, el proceso para expedir un dato a través de uno de los puertos de un PIO.
- 7.- ¿Se puede utilizar un microcontrolador como si fuera un microprocesador, enlazándolo con periféricos externos?
- 8.- ¿Cuál es la principal ventaja de la comunicación paralela entre circuitos lógicos?
- 9.- Menciona dos inconvenientes de este tipo de enlace:
- 10.- La comunicación paralela entre chips, ¿es automática o se debe programar dentro del código del microprocesador?

4.3 COMUNICACIÓN DE DATOS EN FORMA SERIAL

A pesar de las ventajas que tiene la comunicación en paralelo, existen aplicaciones en las que no resulta muy adecuada. Por ejemplo, cada vez son más comunes los circuitos digitales que requieren intercambiar órdenes o datos entre sí, pero éstos se encuentran muy alejados, como el caso de un automóvil moderno, que posee una gran cantidad de microprocesadores y microcontroladores verificando el correcto funcionamiento de una enorme cantidad de variables, pero estos chips se encuentran muy separados entre sí. Pueden existir algunos en el tablero de instrumentos, otros en la computadora de a bordo (anexa al motor), unos más detectando el funcionamiento de las llantas, controlando el flujo de gasolina, etc. Esto significa que todos estos sistemas tienen que estar intercambiando un gran volumen de datos, pero en este caso, la comunicación en paralelo resulta poco adecuada, debido a que se

necesitaría una gran cantidad de cables corriendo por toda la extensión del automóvil. Si a eso se añade que, como ya se mencionó, la comunicación en paralelo es muy susceptible a interferencias externas, la posibilidad de que existan fallas en la comunicación entre módulos se multiplica considerablemente.



En un automóvil moderno, existen varios módulos digitales, intercambiando bastante información entre sí. (Cortesía de Volkswagen).

Para este tipo de aplicaciones, y sobre todo cuando se necesita un intercambio de órdenes y datos entre circuitos digitales, pero no importa demasiado la velocidad de este intercambio, se ha diseñado un tipo de comunicación que requiere menos recursos, menos líneas de enlace, y que correctamente implementada, es mucho más resistente contra interferencias externas que un enlace paralelo, es decir, la comunicación en serie.

En la actualidad, la comunicación serial ha encontrado una amplísima variedad de aplicaciones, desplazando a la comunicación paralela a funciones muy específicas donde la velocidad de procesamiento es vital, como el enlace entre un microprocesador y su memoria, con el chip de video, o con sus medios de almacenamiento; sin embargo, casi todo el resto de las comunicaciones



El popular puerto USB utiliza precisamente una comunicación serie, de ahí su nombre, (*Universal Serial Bus*). (Banco de imágenes del autor).

entre el microprocesador central y sus elementos periféricos, se realiza a través de enlaces seriales, ya que estos últimos presentan varias ventajas en comparación a las líneas en paralelo. A continuación, se indican las características de una comunicación serial y porqué ha prevalecido en los últimos años.

En primer lugar, y como su nombre lo indica, una comunicación en serie sólo utiliza una línea de comunicación entre emisor y receptor

(aunque en realidad, casi siempre se trata de dos líneas, una para datos y otra para el reloj de lectura/escritura); esto significa que si un microprocesador de 8 bits desea enviar cierta información a alguno de sus periféricos, y emplea una comunicación tipo serie, será necesario ir colocando en la línea de datos los bits que se desean enviar de uno en uno, para ir mandándolos en forma de una cadena de bits a través del único hilo de enlace; y aquí aparece uno de los principales inconvenientes de este tipo de comunicación: se requieren de muchos ciclos de reloj para ir colocando los bits de uno en uno, y esto hace a los enlaces seriales considerablemente más lentos que la comunicación paralela.

Sin embargo, la comunicación serial tiene una enorme ventaja en comparación con la paralela: incluso los datos de control necesarios para el correcto intercambio de información, pueden viajar en el mismo hilo de enlace; y

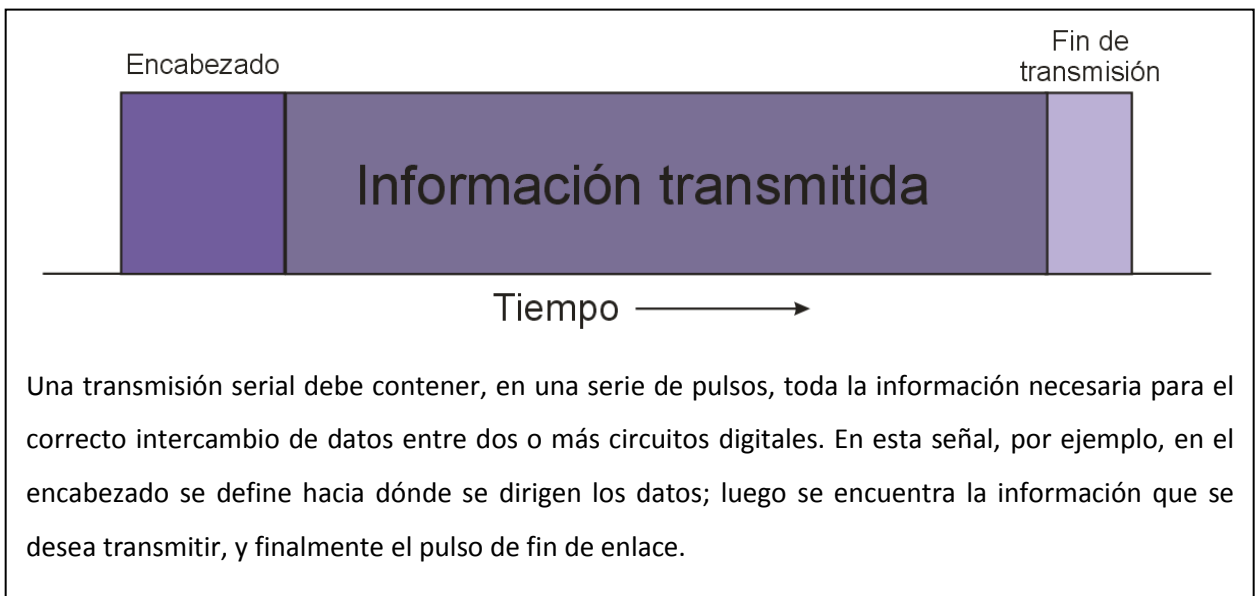
esto hace que este tipo de transmisiones sean ideales para atravesar grandes distancias, o que materialmente sea imposible colocar más de una línea de comunicación.

Por ejemplo, ahora se analizará cómo el control remoto de un televisor o equipo de sonido envía sus órdenes hacia un aparato; estos controles envían sus comandos mediante una luz infrarroja, que es captada por el equipo, e interpretada para dar cumplimiento a las órdenes del usuario; sin embargo, en esa serie de pulsos infrarrojos deben reunirse todos los elementos



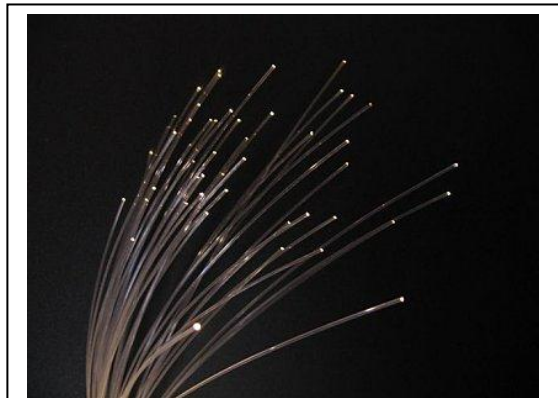
necesarios para el correcto transporte de información:

- Un encabezado que identifique hacia qué aparato va el comando.
- El reloj de datos que sincronice la señal.
- Los datos que se necesitan transmitir.
- Una señal de fin de transmisión.



¿Cómo se puede enviar tal cantidad de datos en una única línea? Si se pudiera observar una transmisión de datos serial graficada contra el tiempo, se encontraría algo semejante a lo que se muestra en la figura anexa: al principio de la señal, aparece un encabezado con una serie de pulsos identificadores, que indican hacia qué aparato o periférico se está enviando la señal; a continuación se encuentra la información que se desea transmitir desde un circuito hacia el otro; y finalmente, se tiene una porción de “fin de señal”, que sirve para dejar todo en condiciones iniciales, listos para otra transmisión.

Tan efectiva resulta la transmisión de datos en forma serial, que es la que se utiliza prácticamente para todas las comunicaciones globales, incluyendo telefonía, Internet, transmisiones de TV digital, etc. Esto se ha favorecido por el desarrollo de circuitos de procesamiento de señal cada vez más veloces, lo que compensa en gran medida el defecto principal de las comunicaciones seriales (su velocidad comparativamente más lenta que un



La fibra óptica utiliza comunicaciones tipo serie, muy parecidas a las que usa un control remoto, pero a una velocidad millones de veces superior. (Imagen Dow Corning)

enlace paralelo). Incluso, el método de comunicación más rápido que se conoce en la actualidad, la transmisión por fibra óptica, utiliza enlaces tipo serial.

Por todo lo anterior, la comunicación serial se ha convertido en parte cotidiana del diseño de circuitos digitales, así que también es necesario saber cómo se lleva a cabo, para utilizarla en caso necesario.

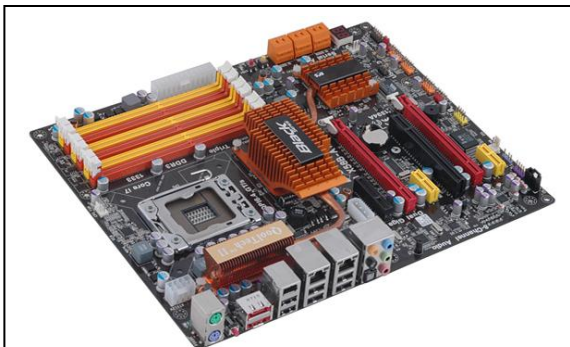
ACTIVIDAD DE APRENDIZAJE 4C

Contesta lo siguiente:

- 1.- ¿Por qué es más conveniente la comunicación serial cuando se deben enlazar dos o más chips que se encuentran relativamente alejados?
- 2.- ¿Cuántas líneas se requieren para una comunicación serial?
- 3.- ¿Cómo viajan los bits en una comunicación serial?
- 4.- ¿Cuál es el principal defecto de la comunicación serial?
- 5.- Menciona dos ejemplos de enlaces seriales usados regularmente:
- 6.- Describe cómo debe ser la estructura de una transmisión serial de datos:
- 7.- ¿Para qué sirve el bloque inicial en una transmisión serie?
- 8.- ¿Cómo se compensa la lentitud relativa de las transmisiones seriales?
- 9.- ¿Por qué la comunicación serie es ideal cuando se necesita transmitir a largas distancias?
- 10.- Investiga si los discos duros modernos usan comunicación paralela o en serie.

4.4 PROGRAMACIÓN, CONTROL Y DIRECCIONAMIENTO DE LOS ELEMENTOS DE ENTRADA/SALIDA

De acuerdo a lo anterior, resulta evidente que si se utilizará un microprocesador simple como núcleo de un circuito de proceso lógico, es



La tarjeta madre de una computadora incluye la mayoría de los periféricos que el microprocesador requiere para convertir a la máquina en una poderosa herramienta de productividad. (Cortesía ECS).



Incluso, el microprocesador más poderoso resulta inútil si no se rodea de los periféricos adecuados para realizar su labor. (Cortesía Intel).

indispensable rodearlo de elementos periféricos que lo apoyen en su labor de cálculo, ya que el procesador, por sí mismo, es incapaz de realizar prácticamente cualquier tarea. Como bloques indispensables para el funcionamiento del circuito, está la memoria, ya sea de datos, de programación, o ambas de forma combinada; también se requiere un circuito generador de reloj y uno que envíe el pulso de reset cada vez que se encienda el sistema; además, si el micro va a interactuar con otros elementos, se necesita por lo menos un circuito PIO que sirva como puerto de entrada y salida de señales digitales. Existen otros bloques funcionales que, por su uso tan común, ha originado que los fabricantes diseñen circuitos especiales para realizar dicha labor; así, están los circuitos temporizadores o *timers*, los chips de comunicaciones DART o USART (*Dual Asynchronous Receiver Transmitter* o *Universal Synchronous Asynchronous Receiver Transmitter*), los convertidores digital-analógico o analógico-digital (DAC o ADC), etc.

Entonces, resulta evidente que un microprocesador es tan poderoso y flexible como sean los periféricos que puedan conectarse a él; esta es la razón principal por la que las computadoras personales incorporan microprocesadores de última generación, capaces de manejar la avalancha de información que un usuario promedio requiere, incluso para una tarea tan sencilla como escribir una carta para enviarla por correo electrónico. A continuación se muestran de manera general, los procesos involucrados dentro de una PC mientras realiza incluso la labor más simple.

La mayoría de los microprocesadores modernos poseen un bus especial para comunicarse con su memoria, así que el bus de datos y el de direcciones prácticamente se utiliza tan sólo para el intercambio de datos con sus circuitos periféricos. Entonces, si todos los circuitos conectados al microprocesador comparten un mismo bus de datos, de direcciones y de control, ¿cómo puede el micro, en un momento dado, enviar cierta información hacia un determinado puerto USB, para que de ahí se envíe a la impresora?, ¿cómo hace para que ese dato no termine en la tarjeta de sonido o en el chip de red?

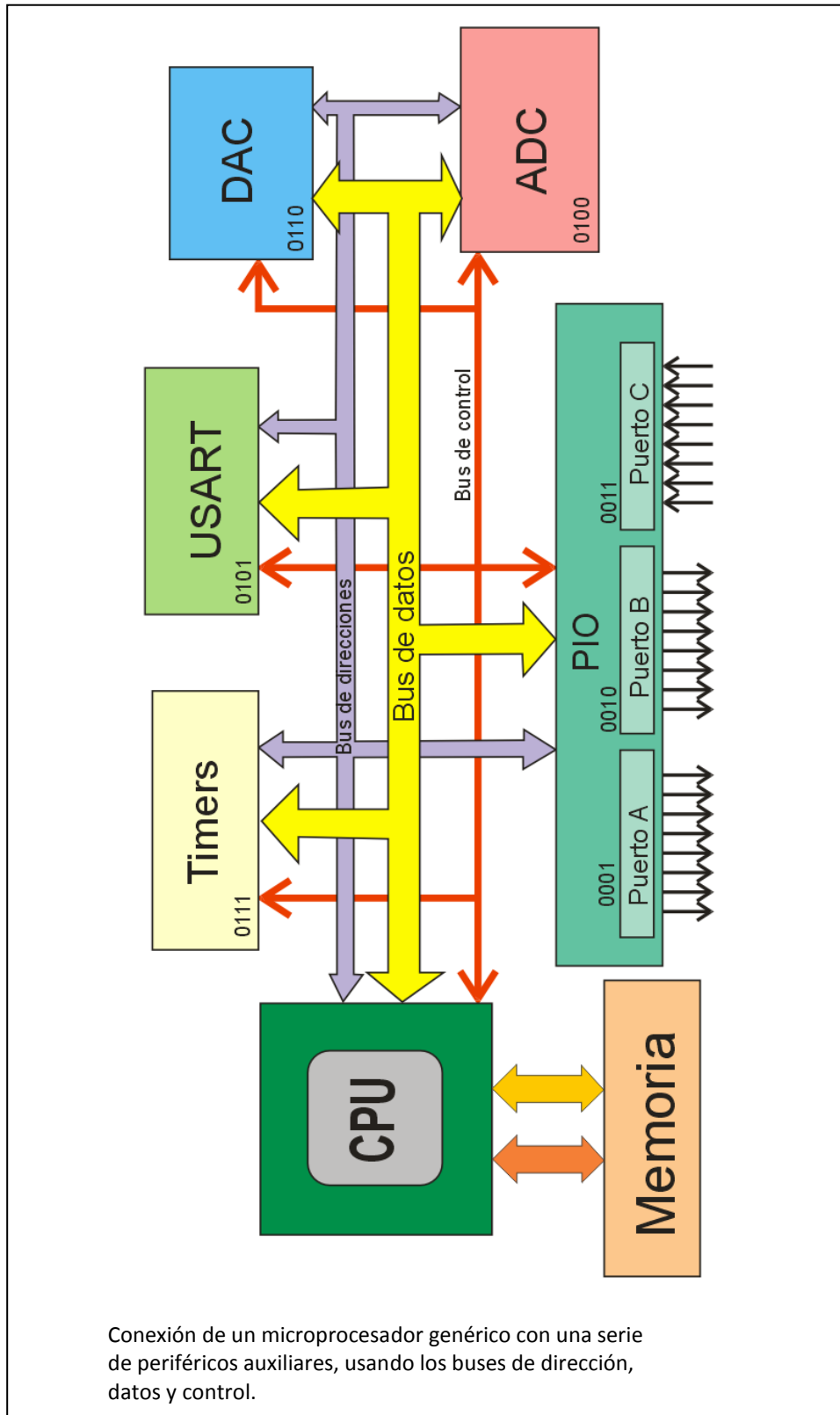
En primer lugar, el número de bits que posea el bus de direcciones determina el número máximo de periféricos que se pueden conectar a un micro, ya que a cada una de las combinaciones posibles de unos y ceros que se puedan expedir por dicho bus, se le puede asignar algún chip específico. Al suponer un microprocesador sencillo, cuyo bus de direcciones tiene sólo 4 bits, lo que implicaría que se pueden conectar a él hasta 15 dispositivos distintos (casi siempre se reserva una de las combinaciones, generalmente la 0h o la Fh, para que en ella no se active ninguno de los chips periféricos).

Ahora, se indica cómo hace el CPU para intercambiar información con cada uno de los bloques mostrados en la siguiente figura; cada bloque trae indicada la dirección a la que el micro debe enviar o desde la que debe recibir sus datos, así que bastará con que en el bus de direcciones se coloque la combinación correcta de bits, para lograr un enlace adecuado entre el micro y sus periféricos.

Por ejemplo, si el micro desea expedir cierta información por el puerto A del PIO, los pasos que debe seguir son:

- Colocar en el bus de datos la combinación b'0001'.
- Activar la línea *write* en el bus de control.
- Colocar en el bus de datos la información que desea expedir por el puerto A.

- Activar la línea *chip enable* en el bus de control.



De este modo, los datos que estaban en el bus de datos pasarán a las salidas del puerto A, y de ahí hacia el circuito al que esté conectado. El mismo procedimiento se sigue para el puerto B, sólo que en el primer paso, la combinación para comunicarse con dicho puerto será b'0010'.

Para recibir datos del puerto C, el procedimiento será:

- Colocar en el bus de direcciones la combinación b'0011'.
- Activar en el bus de control la opción *read*.
- Activar en el bus de control la opción *chip enable*.
- Comenzar a recibir los datos desde el puerto C.

Si ahora el CPU desea intercambiar ciertos datos hacia el *USART*, el procedimiento será el siguiente:

- Colocar en el bus de direcciones la combinación b'0101'.
- Si se enviarán datos, activar en el bus de control la opción *write*.
- Activar en el bus de control la opción *chip enable*.
- Comenzar a enviar los datos que se desean transmitir a través del bus de datos.
- Si se va a recibir información, el proceso es el mismo, pero activando la opción *read* en el bus de control en lugar de *write*.

Hay una situación especial en los bloques DAC y ADC, ya que el primero es un periférico exclusivamente de salida, mientras que el segundo sólo funciona como entrada, esto significa que cuando se requiera recibir un dato que venga del ADC, se coloca en bus de direcciones la combinación b'0100', se activa en el bus de control la opción *read* y se activa la opción *chip enable*, con lo que el micro puede leer el dato que le esté enviando este bloque. Para usar el DAC, se coloca en bus de direcciones la combinación b'0110', se activan en el bus de control las opciones *write* y *chip enable*, y de este modo, el micro

podrá enviar al DAC el valor binario que desea se convierta en un valor analógico equivalente a su salida.

Otro caso especial es el bloque *timer*, que como su nombre lo indica, posee en su interior uno o más temporizadores que le permiten al micro medir lapsos de tiempo muy específicos; en este caso, lo primero que tiene que hacer el micro es introducir en el *timer* una cantidad equivalente al número de pulsos que desea que cuente este bloque, así que lo primero que tiene que hacer el CPU es transmitir dicha cantidad al temporizador. Esto significa que deberá colocar en el bus de direcciones la combinación b'0111', luego, colocar en el bus de control la opción *write*, y luego colocar en el bus de datos la cantidad deseada; finalmente se activa la opción *chip enable*, para que el dato se cargue en el *timer* y éste comience su cuenta regresiva. Cuando se cumple el lapso de tiempo deseado, el bloque *timer* envía una interrupción hacia el micro, entonces éste interrumpe lo que esté haciendo y realiza la labor que tenía que hacer después del lapso indicado.

Por lo tanto, la comunicación entre un microprocesador y sus circuitos periféricos no resulta demasiado compleja, lo único que se debe hacer es colocar en el bus de direcciones la combinación adecuada para el periférico con que se desee enlazar, luego por el bus de control se activan las opciones correctas, y se puede comenzar a intercambiar datos entre el CPU y los bloques que lo rodean. En el siguiente tema, se indicarán un par de ejemplos usando como CPU el PIC 16F628A, para mostrar que en realidad no resulta muy difícil el enlace entre dos circuitos digitales.

ACTIVIDAD DE APRENDIZAJE 4D

Contesta lo siguiente:

- 1.- ¿Sirve de algo un microprocesador sin la compañía de circuitos periféricos?
- 2.- Entonces, ¿cuál es la ventaja de usar un microprocesador?
- 3.- Menciona dos de los bloques periféricos que generalmente acompañan a un microprocesador:
- 4.- ¿Cuáles son los buses que enlazan a un microprocesador con sus periféricos?
- 5.- ¿Qué señales circulan por el bus de control?
- 6.- ¿Qué es lo que se determina a través del bus de direcciones?
- 7.- Un DAC, ¿es un bloque de entrada o salida de datos?
- 8.- ¿Para qué sirven los bloques DART o USART?
- 9.- ¿Qué función tiene el bloque *timer*?
- 10.- ¿Cuáles son los pasos para expedir cierto dato a través de uno de los puertos del PIO?

4.5 PROGRAMAS DE APLICACIÓN

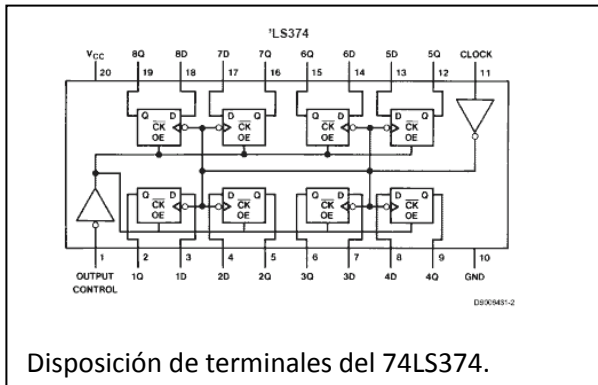
A continuación, se realizarán un par de prácticas en las cuales se simulará que el PIC 16F628A es un microprocesador, y que por tanto, se necesitará que se comunique con algunos bloques periféricos para poder funcionar. Se usará el puerto A como bus de datos, y el puerto B como bus de control, y como circuito periférico se usará un chip de la familia TTL, el 74LS374, que es un óctuple flip-flop tipo D, que puede funcionar como un puerto de salida de



74LS374, chip adicional que se usará en las prácticas de comunicación.

(Cortesía TI)

datos, ya que la información que aparezca a la entrada de los flip-flops pasará tal cual hacia la salida al momento de recibir un pulso de reloj.



En la figura anexa, se muestra la disposición de terminales de este integrado; las terminales de entrada están marcadas como “xD”, donde “x” puede ir del 1 al 8; existen un par de señales de control, una OE o habilitador de salida, que funciona

como una especie de selección de chip, y una de reloj que funcionará como señal *chip enable*. Para facilitar aún más el ejercicio, se mantendrá la terminal OE permanentemente activada, y el “bus de control” se limitará al pulso de activación de los flip-flops, los cuales pasan la información de su entrada a su salida con el flanco ascendente del pulso.

Otra consideración: se creará este programa como si fuera una subrutina (AOUT1) de otro más grande, para evitar que el código sea excesivamente largo. Entonces, el programa no funcionará tal y como se indique aquí, pero sí lo hará si se añade como una subrutina en algún código posterior.

Después de colocar los puertos A y B del micro como salidas, de configurar que use su reloj interno, y teniendo una subrutina adicional que provoca un retraso de un segundo (rutina SEG, mostrada en la unidad anterior), ahora se deberá suponer que en las salidas del 74LS374 aparezca el contenido de un registro SALIDA1, y que permanezca así incluso si en el puerto A se están expidiendo otras señales distintas. El pulso de activación del 74LS374 se expedirá por la terminal 7 del puerto B. El código para realizar esta tarea será:

```

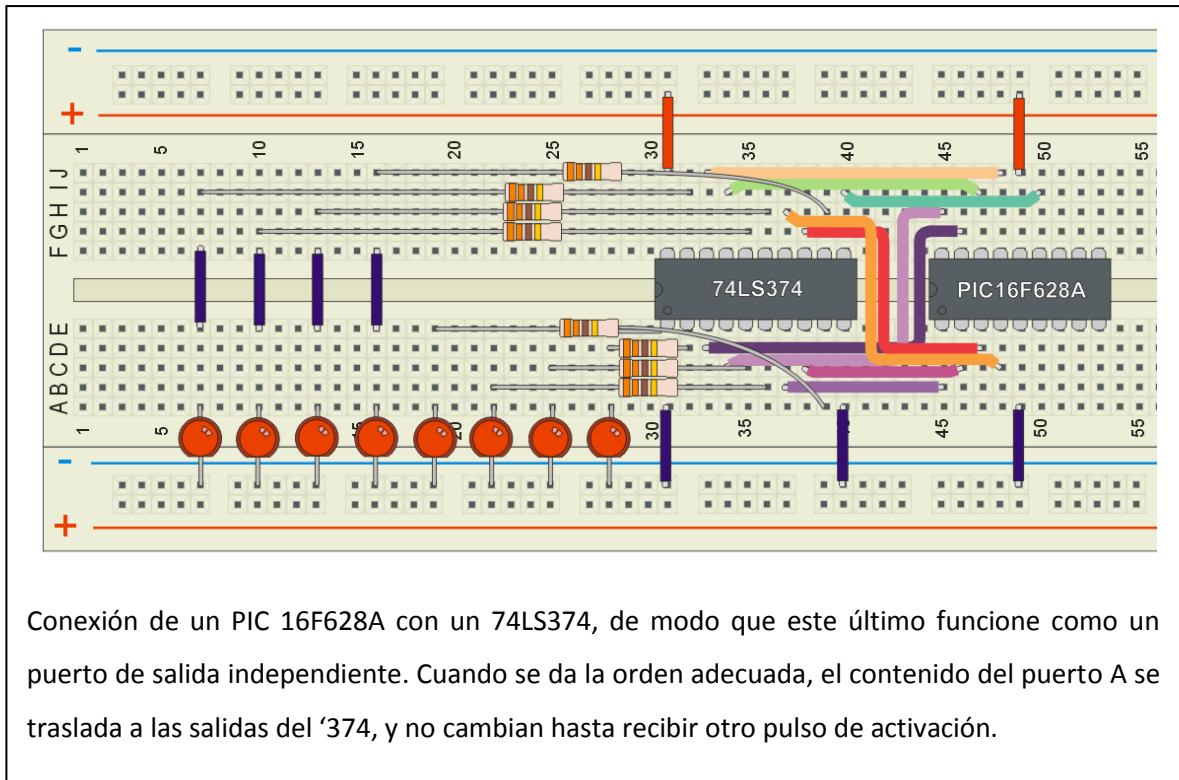
AOUT1          MOVF SALIDA1, 0 ; se carga el contenido del registro
                  ; SALIDA1 en W
MOVWF          PORTA          ; se envía hacia el puerto A del
PIC
BSF           PORTB, RB7 ; se coloca en 1 RB7, generando el
    
```

```

; pulso que necesita el 374 para
; transferir los datos
NOP
NOP ; un par de NOP para que el pulso no
; sea demasiado rápido
BCF PORTB, RB7 ; se regresa a cero RB7
RETURN ; regresa al programa principal

```

¡Y eso fue todo! Si durante la ejecución de un programa, se requiere que cierta información permanezca disponible en la salida, a pesar de que el MCU esté haciendo otras cosas, ésta es una solución fácil, rápida y económica. De hecho, si se extiende esta situación, se pueden tener hasta ocho '374 conectados al puerto A, y que cada pin del puerto B active a uno de ellos, con lo que con un PIC 628 de 18 terminales, se podrían controlar ¡hasta 64 terminales de salida! En la siguiente figura, se muestra cómo quedaría este proyecto en un protoboard. En los LED, aparecerá exactamente la palabra que se envíe hacia el puerto A, una vez que se haya activado el pulso de *chip enable*; y después de eso, las salidas del puerto A pueden cambiar nuevamente, sin que eso afecte a los LED, hasta que se accione otro pulso de activación. Se han simulado cables de distinto color, para observar fácilmente de dónde sale un cable y hasta dónde llega, aunque obviamente esto no es indispensable.



Ahora se muestra, cómo se podría implementar una comunicación en serie entre el PIC y el '374. Para ello, se hará que la salida serial esté en RA0; en RB7, estará la señal de reloj que acompaña a esta información serial; y en la salida del '374, aparecerá la palabra que se envió de forma serial, con el encendido y apagado de los LED. Nuevamente, se supondrá que lo que se desea transmitir se encuentra en un registro SALIDA1, y que ya se cuenta con una subrutina de espera SEG, de modo que se esté enviando aproximadamente un bit cada segundo hacia el '374. Se efectúan las mismas suposiciones que en el caso anterior, ubicando todo este programa como una subrutina de otro más grande. El código para lograr esto quedaría así:

```

AOUT1          MOVLW    b'00001000'      ; se carga un "8" en W
               MOVWF    CONT4             ; se envía a un registro
CONT4
PUNTO1        BTFSC    SALIDA1, 0        ; se checa si hay un "cero" en
               ; el bit 0 de SALIDA1

```

```

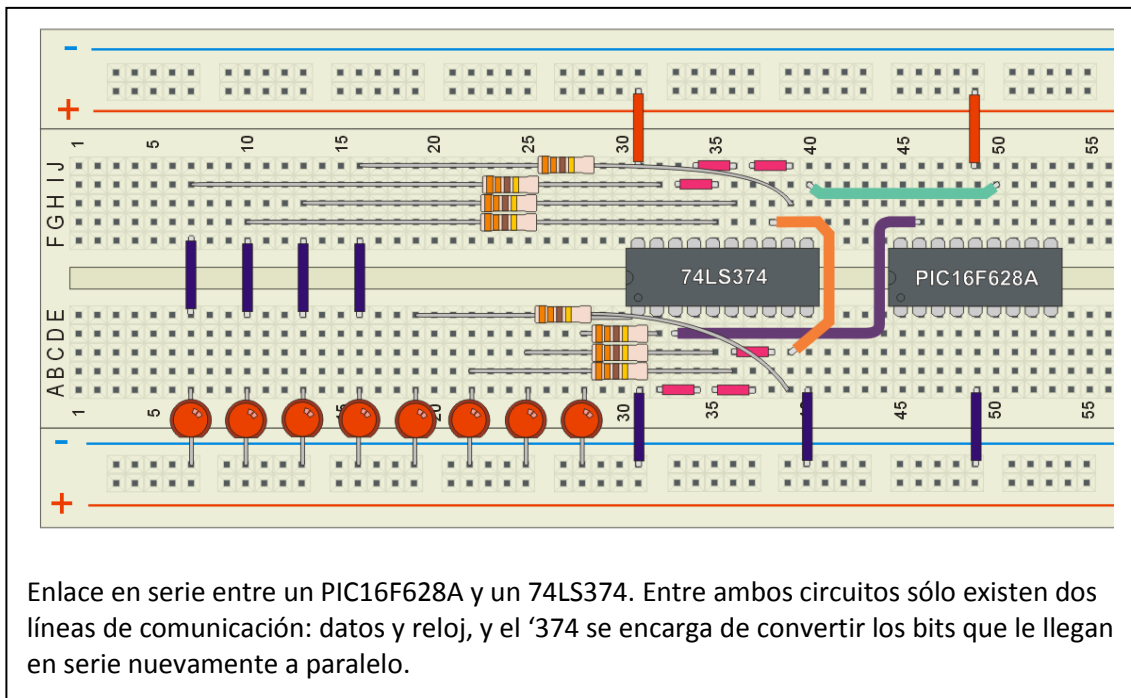
                BTFSS SALIDA1, 0      ; se checa si hay un "uno" en
                                        ; el bit 0 de SALIDA1
                GOTO SEND0            ; si hay un cero, ve a SEND0
                GOTO SEND1            ; si hay un uno, ve a SEND1
SEND0          BCF  PORTA, RA0        ; se pone en cero RA0
                BSF  PORTB, RB7        ; se envía el pulso de activación
                NOP
                NOP                    ; pequeño retardo
                BCF  PORTB, RB7        ; se regresa a RB7 a cero
                GOTO ROTATE            ; ve a ROTATE
SEND1          BSF  PORTA, RA0        ; se pone en uno RA0
                BSF  PORTB, RB7        ; se envía el pulso de activación
                NOP
                NOP                    ; pequeño retardo
                BCF  PORTB, RB7        ; se regresa a RB7 a cero
ROTATE        RRF  SALIDA1, 1        ; se desplazan los bits de
                                        ; SALIDA1 una posición a la
                                        ; derecha
                DECFSZ  CONT4          ; Se decrementa una unidad
CONT4
                CALL  SEG              ; se llama a la subrutina SEG
                GOTO  PUNTO1           ; regresa al punto 1 para repetir
                                        ; el proceso
                RETURN                 : regresa al programa principal

```

Este proceso, lo que hace es cargar un "8" en un registro CONT4, luego lee el contenido del bit cero de SALIDA1 y lo envía hacia RA0, activando el pulso para el '374: luego recorre los bits a la derecha, de modo que el bit1 pasa a ser el bit0, el bit2 pasa a ser el bit1, y así sucesivamente, y luego regresa al punto inicial para repetir el proceso. Una vez que se han transmitido los 8 bits de SALIDA1, termina la subrutina y se regresa al programa normal.

Sin embargo, aquí el detalle está en cómo está conectado el '374 para convertir los datos que le van llegando de forma serial en una salida paralela. En el siguiente diagrama, se puede analizar cómo se logra esto.

Se puede notar fácilmente que el alambrado de este circuito es menos complejo que el anterior, eso es porque sólo existen dos líneas de comunicación entre el PIC y el '374: la línea de datos que sale de RA0 y se dirige a D1, y la línea de reloj, que sale de RB7 y llega a CLK. Aquí el secreto está en la conexión de los bloques internos del '374; se debe observar que la salida Q1 se conecta a D2, la salida Q2 va a D3, y así sucesivamente. Esto significa que cuando llegue un pulso en CLK, el nivel que se encuentre en RA0 pasará a Q1; el que estaba anteriormente en la salida Q1 pasará a la salida Q2; Q2 pasará a Q3; y así sucesivamente, hasta que al llegar a ocho pulsos de reloj, los bits que llegaron en serie a través del enlace entre RA0 y D1, ahora se muestran en paralelo, a través de los LED.



Sin embargo, incluso este ejemplo tan sencillo, sirve para resaltar la gran diferencia entre ambos tipos de comunicación: el enlace paralelo es muy rápido y sencillo de implementar a nivel software, pero implica un hardware más

complejo (una gran cantidad de cables corriendo entre ambos chips); al contrario, el enlace serial resulta muy sencillo de implementar a nivel hardware, pero el software de control se complica bastante, además de que toma muchísimos más ciclos de reloj poder efectuar la transmisión de datos en modo serial que en paralelo.

Hasta este momento, se ha descrito la forma de utilizar un microcontrolador de 8 bits, aprovechando sus bloques internos y simulando con él un microprocesador convencional. Se recomienda practicar la programación del PIC 16F628A, y aplicarse en diversos proyectos electrónicos; es sorprendente lo que se puede hacer con un circuito tan sencillo y económico.

ACTIVIDAD DE APRENDIZAJE 4E

Contesta lo siguiente:

- 1.- Investiga qué significa TTL, y cuáles son las características principales de esta familia de circuitos lógicos:
- 2.- ¿Qué función tiene el 74LS374?
- 3.- Investiga cuál es la característica principal de un flip-flop tipo D:
- 4.- ¿Se puede usar un '374 como puerto auxiliar de salida de un procesador lógico?
- 5.- ¿Cuántos bits puede transmitir a la vez el PIC al '374 durante una transmisión en paralelo?
- 6.- ¿Qué tan complejo es el código para una transmisión en paralelo?
- 7.- ¿Se puede usar el '374 como receptor de una transmisión en serie?
- 8.- ¿Cuántas líneas de comunicación se necesitan entre el PIC y el '374 en una transmisión en serie?
- 9.- ¿Qué tan complejo es el código para una transmisión serial?
- 10.- De acuerdo a los ejemplos, ¿cuál transmisión será más veloz, la paralela o la serial?

AUTOEVALUACIÓN

- 1.- ¿Cuáles son los dos tipos de comunicación más utilizados en circuitos digitales?
- 2.- Menciona las principales ventajas de la comunicación en paralelo:
- 3.- Menciona la principal ventaja de la comunicación en serie:
- 4.- Menciona dos ejemplos de transmisiones digitales en serie:
- 5.- ¿Cómo se ha compensado el principal problema de la comunicación en serie?
- 6.- ¿A través de cuáles buses se comunica un microprocesador con sus circuitos periféricos?
- 7.- ¿Qué hace el bus de direcciones? ¿Y el de control?
- 8.- ¿Por qué es importante la señal *chip enable*?
- 9.- ¿Qué es un PIO? ¿Para qué sirve?
- 10.- Menciona otros dos bloques funcionales, que normalmente se encuentran conectados a un microprocesador:

RESPUESTAS

- 1.- Comunicación en paralelo y comunicación en serie o serial.
- 2.- Es más fácil de implementar, y es mucho más veloz que la serial.
- 3.- Se necesitan menos líneas entre el procesador y sus periféricos, simplificando el diseño general.
- 4.- Los controles remotos de los aparatos electrónicos, el puerto USB de una computadora, la interfaz SATA de los discos duros, los enlaces por fibra óptica, etc.
- 5.- Utilizando circuitos de proceso digital cada vez más rápidos y poderosos.
- 6.- Bus de datos, bus de direcciones y bus de control.
- 7.- El de direcciones determina hacia cuál de los periféricos se enlazará el microprocesador, mientras que el de control garantiza que el intercambio de datos sea adecuado entre ambos.
- 8.- Es la señal final con la que el procesador avisa que está listo para enviar o recibir los datos hacia o desde sus periféricos.
- 9.- Es un puerto de entrada y/o salida de datos; funciona como puerto de uso general para el intercambio de datos digitales.
- 10.- Temporizadores, convertidores digital-analógico o analógico-digital, bloques de transmisión DART o USART, etc.

Solución de las actividades de aprendizaje:

Actividad de aprendizaje 4A

- 1.- Transmitiendo los bits y bytes que deseen compartir a través de líneas de comunicación.
- 2.- Comunicación en paralelo y comunicación en serie.
- 3.- Que mueve una gran cantidad de bits en poco tiempo, lo que la hace muy veloz.
- 4.- Porque así se aumenta la flexibilidad del circuito, colocando los periféricos adecuados para la aplicación específica.
- 5.- De 8 bits.
- 6.- Para habilitar al circuito periférico con el que desee interactuar el procesador.
- 7.- Maneja el intercambio de datos, para garantizar que se lleve a cabo de forma fluida y sin problemas.
- 8.- No, algunas se aprovechan para direccionar a sus circuitos periféricos.

Actividad de aprendizaje 4B

- 1.- Parallel Input-Output, o entrada-salida paralela.
- 2.- Funcionan como puertos genéricos de entrada o salida de información digital.
- 3.- Sí, algunos de los puertos pueden ser entrada y otros salida, según se necesite.
- 4.- Las señales de selección de puertos, la que determina si se leerá o se escribirá en el puerto, la señal *chip enable* y las interrupciones de realimentación.
- 5.- Sí, sobre todo cuando un puerto funciona como entrada de datos.
- 6.- Se direcciona el circuito PIO, se selecciona el puerto que se va a utilizar, se envía la señal de *write* para expedir datos por él, se colocan los datos en el bus de datos, y se activa la señal *chip enable*.

- 7.- Sí, aunque lo mejor es aprovechar al máximo sus bloques internos.
- 8.- La gran velocidad con la que se consigue el intercambio de datos.
- 9.- Se necesitan muchas líneas de comunicación entre el procesador y sus periféricos, y son más susceptibles a ser afectadas por ruido e interferencia.
- 10.- Se tiene que programar en el código del procesador.

Actividad de aprendizaje 4C

- 1.- Porque requiere sólo de una o dos líneas de enlace entre ambos, lo que simplifica el diseño y hace más segura la transmisión.
- 2.- Una o dos, dependiendo si se requiere de una señal de reloj independiente.
- 3.- Uno detrás de otro, como en una cadena de bits.
- 4.- Que es mucho más lenta que la comunicación paralela.
- 5.- Puertos USB, el control remoto de aparatos electrónicos, la fibra óptica, etc.
- 6.- Se tiene un encabezado que indica hacia dónde se dirigen los datos, un cuerpo con la información que se está enviando, y un pulso de fin de transmisión.
- 7.- Para identificar el bloque hacia donde se está enviando la información.
- 8.- Usando circuitos de proceso digital más rápidos y poderosos.
- 9.- Porque sólo se necesita una línea para la transmisión, reduciendo la complejidad del sistema; además de que es menos susceptible a ruidos e interferencias.
- 10.- Los discos SATA utilizan comunicación tipo serial.

Actividad de aprendizaje 4D

- 1.- No, no puede hacer absolutamente nada.
- 2.- Le da mayor flexibilidad a un diseño, permitiendo elegir los periféricos que se usarán.
- 3.- Memoria, PIO, USART, DART, *timers*, ADC, DAC, etc.
- 4.- Bus de datos, bus de direcciones y bus de control.
- 5.- La señal *chip enable*, *write/read*, *port select*, etc.
- 6.- Con cuál de sus periféricos va a interactuar el procesador.

- 7.- De salida, convierte un dato digital en un nivel de voltaje.
- 8.- Para comunicarse con otros circuitos digitales.
- 9.- Contadores de tiempo que miden lapsos predeterminados.
- 10.- Se selecciona el chip a través del bus de direcciones, se elige el puerto que se usará, se determina cuál será entrada o salida de datos, se colocan los datos en el bus respectivo, y se activa *chip enable*.

Actividad de aprendizaje 4E

- 1.- Lógica transistor a transistor, se trata de una familia muy veloz y segura de circuitos digitales.
- 2.- Es un óctuple flip-flop tipo D, ideal para transferencias de datos.
- 3.- Cuando recibe un pulso de reloj, el nivel a su entrada lo pasa a su salida.
- 4.- Sí, tanto en transmisiones paralelas como en serie.
- 5.- 8 bits, el ancho de su bus de datos.
- 6.- Es muy sencillo, sólo unas líneas de código para colocar el dato en la salida de un puerto, y una señal de activación.
- 7.- Sí, encadenando los flip-flops uno detrás de otro.
- 8.- Dos, una de datos y otra de reloj.
- 9.- Es bastante más complicado que el de la transmisión en paralelo.
- 10.- La transmisión paralela es mucho más veloz.

UNIDAD 5

CARACTERÍSTICAS ESPECÍFICAS DE MICROPROCESADORES DE 16 Y 32 BITS

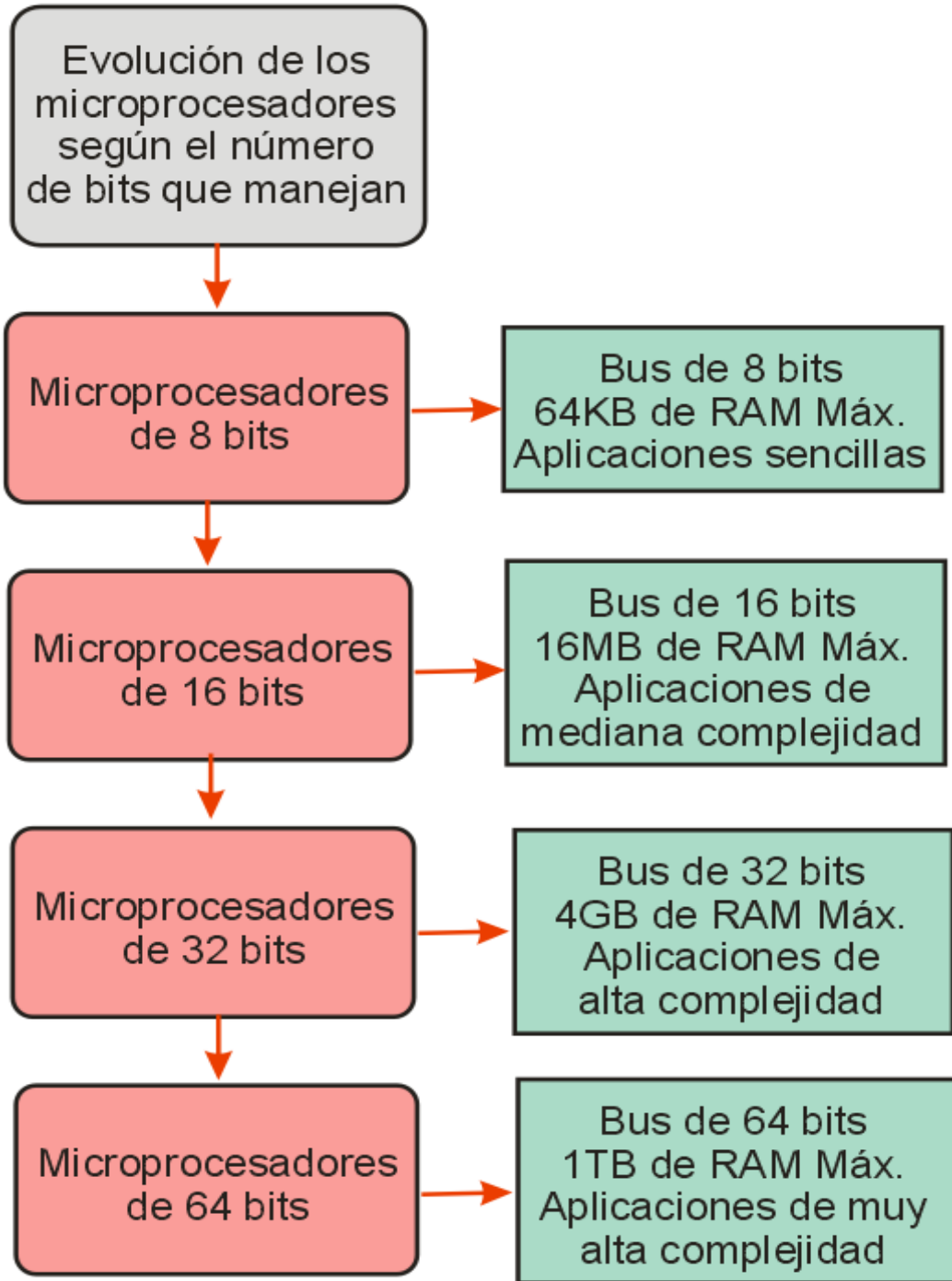
OBJETIVO

El alumno identificará las principales características de los microprocesadores de 16, 32 y 64 bits, así como las ventajas de cada uno de ellos y las razones para elegir uno u otro en alguna aplicación específica.

TEMARIO

- 5.1 BREVE HISTORIA DE LOS MICROPROCESADORES DE 16, 32 Y 64 BITS
- 5.2 CAPACIDAD DE MEMORIA
- 5.3 PROGRAMAS DISEÑADOS PARA 16 Y 32 BITS
- 5.4 DIRECCIONAMIENTO DE UNIDADES O BLOQUES DE MEMORIA
- 5.5 ARQUITECTURA Y SET DE INSTRUCCIONES PARA MICROPROCESADORES DE 16 Y 32 BITS

MAPA
CONCEPTUAL



INTRODUCCIÓN

Hasta este momento, se ha estudiado con cierto detalle la arquitectura y el funcionamiento de procesadores digitales de 8 bits, que a pesar de su poco poderío, se siguen utilizando en diversas labores de control y en proyectos electrónicos variados; sin embargo, el avance de la electrónica digital y las crecientes necesidades de los fabricantes y los usuarios, han obligado a los diseñadores de procesadores a mejorar cada vez más sus dispositivos, haciéndolos más poderosos y flexibles, adecuados para los requerimientos del software actual. Estos procesadores también se utilizan como centro de control para los avanzados aparatos electrónicos modernos, como televisores, reproductores de DVD, impresoras, multifuncionales, teléfonos celulares, etc., así que los procesadores de 16 y 32 bits se han convertido en los más empleados en la actualidad.

5.1 BREVE HISTORIA DE LOS MICROPROCESADORES DE 16, 32 Y 64 BITS

Si bien los microprocesadores de 8 bits fueron toda una revolución en su momento, convirtiéndose en piedra angular para el desarrollo de las primeras computadoras personales exitosas, al cabo de muy poco tiempo estos dispositivos comenzaron a mostrar sus limitaciones, que impedían utilizarlos para labores realmente complejas. Ante la demanda de más poder y flexibilidad, los diseñadores de estos circuitos decidieron romper con la barrera de los 8 bits, lo que le daría al dispositivo mayor capacidad de



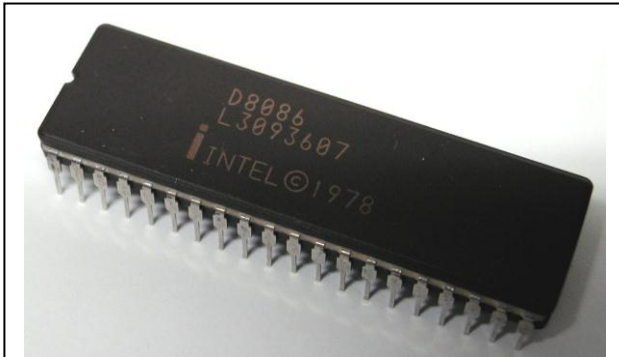
Computadora Tandy TRS-80 de finales de la década de 1970; utilizaba un microprocesador Zilog Z-80 de 8 bits. (Cortesía Radio Shack).

cálculo, acceso directo a una mayor cantidad de memoria, la posibilidad de ejecutar programas mucho más complejos y fáciles de manejar, etc.

La primera empresa en lanzar al mercado un microprocesador de 16 bits fue Intel, con su chip i8086, presentado al público en 1978. Este micro buscaba quitarle el predominio al Z80 de Zilog, así que los diseñadores de Intel mejoraron considerablemente sus características operativas. En la siguiente tabla se muestra una comparación rápida entre ambos microprocesadores:

<i>Característica</i>	<i>Zilog Z80</i>	<i>Intel 8086</i>
Ancho del bus de datos	8 bits	16 bits
Velocidad típica de operación	4 MHz	4.7 MHz
Memoria RAM máxima	64 kB	1 MB
Terminales multiplexadas	No	Si
Interrupciones	2	5
Encapsulado	DIL-40	DIL-40
Coprocesador matemático	No	8087

Transistores	8,500	29,000
--------------	-------	--------



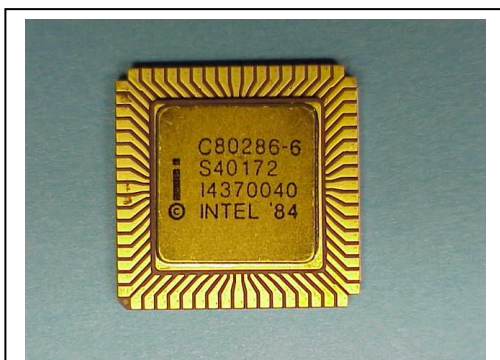
Microprocesador 8086 de Intel, el primer procesador de 16 bits en el mundo, y la base sobre la que se construyó la arquitectura x86, usada en las computadoras tipo PC.

(Cortesía Intel)

Es fácil apreciar la gran diferencia entre ambos dispositivos, sobre todo en la complejidad de su arquitectura interna (lo que se refleja en el número de transistores empleados). El 8086 permitió la construcción de las primeras computadoras pequeñas, lo suficientemente poderosas como para que las empresas comenzaran a considerarlas como opción para aumentar su

productividad, y una variante de este microprocesador, el 8088, fue adoptada por IBM para la construcción de su primera PC-XT, la máquina que dio el impulso final al concepto de computación personal.

Sin embargo, el límite de 1MB de RAM que imponía el 8086-8088 pronto demostró ser un obstáculo importante para el posterior crecimiento de la plataforma, por lo que Intel diseñó un nuevo microprocesador que, a pesar de seguir trabajando con 16 bits, permitía el acceso directo a mayor cantidad de memoria. Aparece así el 80286 en 1982, y una de sus principales ventajas era su capacidad de direccionar hasta 16MB de RAM de forma directa.



Intel 80286, primer procesador de 16 bits que pudo manejar más de 1MB de RAM. (Cortesía Intel)

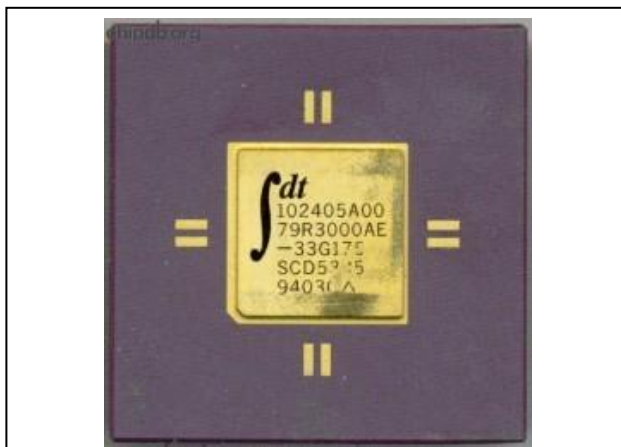
A mediados de la década de 1980, los usuarios de computadoras personales ya estaban alcanzando los límites de lo que podían ofrecer los

microprocesadores de 16 bits, así que Intel presentó en 1985 el primer microprocesador de 32 bits en el mundo, el 80386. Este dispositivo tenía muchísimas características que lo hacían más avanzado, poderoso y flexible que sus antecesores, como son: posibilidad de direccionar hasta 4GB de RAM de forma directa; memoria protegida, lo que permite ejecutar más de un programa a la vez sin interferirse entre sí (multitarea real); más y mejores



Intel 80386, primer microprocesador de 32 bits. (Cortesía Intel)

instrucciones en su set de comandos básicos, lo que amplía considerablemente la potencia del software que se puede ejecutar con él; en fin, todo lo necesario para convertir a las computadoras personales en poderosas herramientas de productividad y entretenimiento. Cuando surgieron las primeras PC que utilizaban este dispositivo, se desarrolló también el primer ambiente de trabajo gráfico exitoso en esta plataforma, el ambiente Windows de Microsoft, que revolucionó por completo la forma como el usuario interactuaba con su computadora.



El MIPS R3000, fabricado por IDT bajo licencia de MIPS Technologies; primer microprocesador de 64 bits en el mundo. (Cortesía IDT)

La arquitectura de 32 bits fue tan exitosa, que por algún tiempo los diseñadores de Intel pensaron que no sería necesario el desarrollo de microprocesadores con mayor número de bits; es por ello que el primer microprocesador de 64 bits fue diseñado y producido en 1991

por una empresa poco conocida hasta el momento: MIPS Technologies, y fue construido como “cerebro” para las computadoras Silicon Graphics, especializadas en labores de diseño industrial. A este club pronto se unió DEC, en 1992 con sus microprocesadores serie Alpha, Sun en 1995 con sus micros Sparc e IBM en 1997 con sus PowerPC; y hasta 2003, los microprocesadores de 64 bits llegan a las computadoras personales, con AMD y su serie Athlon-64. Los últimos microprocesadores de Intel, la serie Core i5-i7, también son dispositivos de 64 bits. Esto significa que toda computadora moderna que incluya cualquiera de estos dispositivos ya tiene en su interior todo el poderío de un procesador de 64 bits. No hay hasta el momento ningún microprocesador comercial de mayor capacidad.

Este ha sido un recorrido rápido por la evolución de los procesadores digitales; a continuación se indican algunas características específicas de los dispositivos de 16 y 32 bits, para saber qué se puede esperar de ellos.

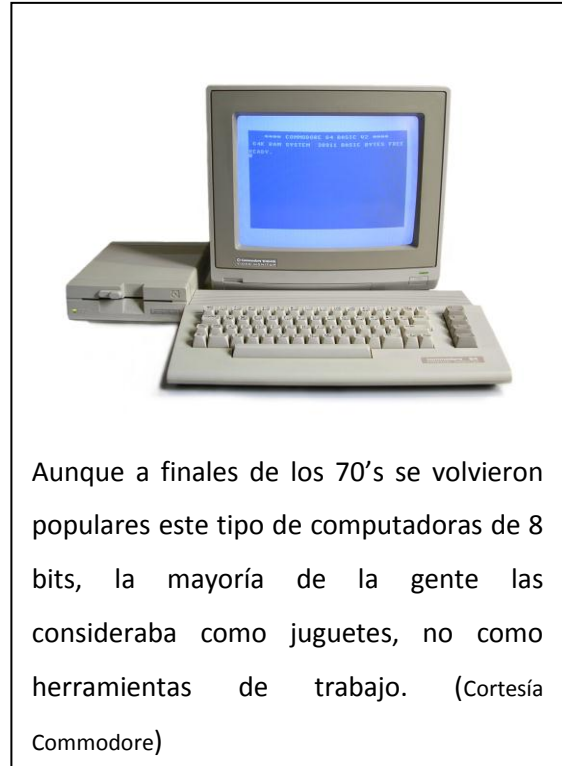
ACTIVIDAD DE APRENDIZAJE 5A

Contesta lo siguiente:

- 1.- ¿Cuál microprocesador de 8 bits dominó el mercado a finales de la década de 1970?
- 2.- ¿Cuál fue el primer microprocesador de 16 bits?, ¿qué empresa lo diseñó?
- 3.- ¿Cuál era la capacidad máxima de memoria de este dispositivo?
- 4.- ¿Cuál variante de este micro utilizó IBM en la recién creada plataforma PC?
- 5.- ¿Qué microprocesador rompió la barrera de 1MB que existía en plataforma PC?
- 6.- ¿Cuánta memoria podía manejar este dispositivo?
- 7.- ¿Cuál fue el primer microprocesador de 32 bits? ¿Qué empresa lo diseñó?
- 8.- ¿Cuál es el límite teórico de memoria que puede manejar este dispositivo?
- 9.- ¿Cuál fue el primer microprocesador de 64 bits? ¿Qué empresa lo diseñó?
- 10.- Menciona dos microprocesadores de 64 bits utilizados en la actualidad en computadoras personales:

5.2 CAPACIDAD DE MEMORIA

Uno de los puntos más importantes que se mejoraron conforme han evolucionado las generaciones de microprocesadores, es su manejo de memoria. Por ejemplo, un microprocesador típico de 8 bits por lo general posee un bus de direcciones de 16 bits, lo que le permite manejar un máximo de 64kB de RAM. Aunque a finales de la década de 1970 esa cantidad de memoria parecía suficiente para las computadoras caseras, la verdad es que una capacidad de almacenamiento temporal tan baja obligaba a utilizar programas muy poco sofisticados, que apenas podían realizar el trabajo encomendado; e incluso en algunos casos, la computadora era incapaz de manejar todo el volumen de información que se requería (por ejemplo, la nómina de una gran empresa; si no puede ser cargada en los 64kB disponibles, no se puede trabajar con ella).



Los microprocesadores de 16 bits compensaron este problema añadiendo más líneas al bus de direcciones; así, el 8086 y el 8088 poseen 20 líneas de direccionamiento, lo que les da la capacidad de manejar directamente hasta 1MB de RAM. Esa cantidad, que ahora parece tan limitada, a principios de la década de 1980 se

consideraba más que suficiente, y de hecho, cuando apenas apareció la plataforma PC, el máximo de memoria que se podía colocar en una de estas máquinas era de tan sólo 640kB; y en ese espacio se tenía que cargar tanto el programa como los datos con los que se deseara trabajar.

Conforme han ido creciendo las necesidades de los usuarios, los diseñadores de microprocesadores han tratado de satisfacerlas; entonces, cuando los 640kB de RAM que se podían instalar en una computadora XT

comenzaron a mostrar sus limitaciones, Intel presentó el procesador 80286, el cual ya tenía un bus de direcciones de 24 bits, lo que le da acceso directo a un máximo de 16MB de RAM. Por su parte, Zilog presentó el Z8000, una versión de 16 bits de su popular Z80, capaz de manejar hasta 8MB de RAM de forma directa debido a un bus de direcciones de 23 bits, mientras que Motorola presentó el



Las computadoras que usaban el 80286, podían manejar hasta 16MB de RAM, aunque raras veces se colocaba más de 2MB. (Cortesía IBM).

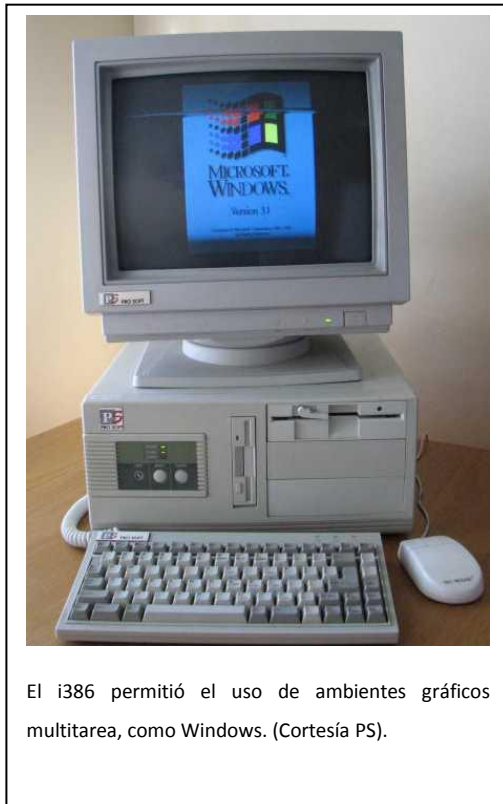
MC68000 con un bus de direcciones de 24 bits y un máximo de 16MB de RAM; sin embargo, debido a su inclusión en la plataforma PC, la arquitectura x86 pronto prevaleció, y aunque tanto la plataforma 68xx como la Z8xx han prevalecido hasta nuestros días, los microprocesadores más conocidos y populares en la actualidad son los de la familia x86; por esta razón, las explicaciones posteriores se concentrarán en esta familia.

Aunque 16MB de RAM parecían más que suficientes para las necesidades de la época, el 80286 aún tenía un inconveniente muy grave desde el punto de vista del usuario avanzado: no tenía forma de proteger segmentos de memoria cuando estuvieran siendo utilizados por alguna aplicación. Esto significa que si al usuario se le ocurría ejecutar dos programas al mismo tiempo, y ambos trataban de acceder a un mismo bloque de memoria,

se ocasionaba un conflicto que, por lo general, implicaba una máquina completamente bloqueada. Por esta razón, en computadoras 8088 y 80286

resulta casi imposible tener más de una aplicación abierta al mismo tiempo.

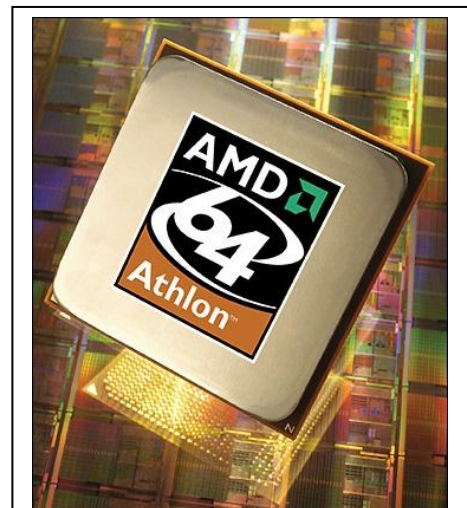
Esto se solucionó con la aparición del microprocesador 80386, conocido posteriormente tan sólo como i386 (y todos sus clónicos conocidos). Este microprocesador ya podía manejar palabras de 32 bits de extensión, y su bus de direcciones tenía también 32 bits de extensión, lo que le permite la instalación de hasta 4GB de RAM, con la ventaja adicional de que esta memoria puede trabajar en “modo protegido”, esto es, si el usuario abre más de una aplicación a la vez, el procesador segmenta la memoria y le asigna a cada programa un “pedazo” de



El i386 permitió el uso de ambientes gráficos multitarea, como Windows. (Cortesía PS).

la misma, para que puedan trabajar sin interferencias entre sí. Esto abrió la puerta al concepto de multitarea, tan común actualmente.

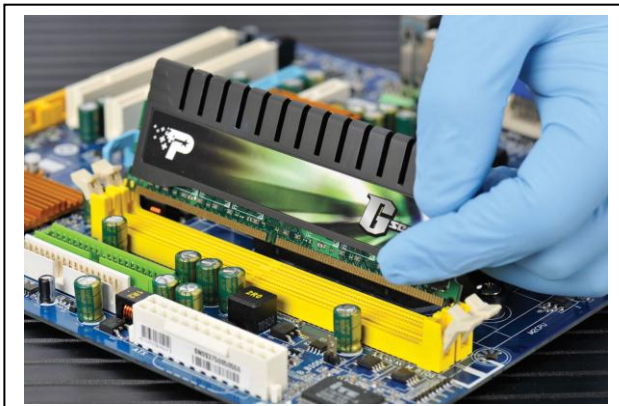
Si bien a mediados de la década de 1980, cuando fue presentado el i386, 4GB parecían inalcanzables, el rápido desarrollo de los circuitos digitales y la necesidad cada vez más apremiante de programas más complejos y poderosos han hecho que en la actualidad, cualquier computadora más o menos avanzada ya pueda incluir 4GB o más de RAM. Esto no preocupó a los diseñadores de microprocesadores para PC por mucho tiempo, de modo que los procesadores de 4ª, 5ª, 6ª y 7ª generación (desde el 486 hasta



Primer microprocesador de 64 bits en plataforma PC: Athlon-64 de AMD. (Cortesía AMD).

los Pentium-4), siguieron siendo dispositivos de 32 bits con un límite máximo de memoria de 4GB. Así, hasta fechas relativamente recientes, es que los dispositivos de 64 bits llegan a la plataforma PC, trayendo consigo todo el poderío de esta nueva generación de procesadores.

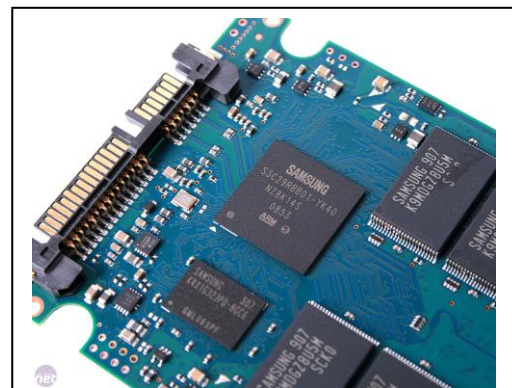
Los primeros microprocesadores de 64 bits que aparecieron en computadoras de la plataforma PC fueron los Athlon-64 de AMD; estos micros poseen un bus de direcciones de 40 bits, lo que teóricamente les permitiría manejar hasta 1TB de RAM de forma directa, aunque en realidad este parámetro suele estar limitado por el número de ranuras de memoria disponibles en la tarjeta madre. Algo similar ocurre con los microprocesadores Core i3-i5-i7 de Intel, cuyo máximo de memoria teórico es muy elevado, aunque los dispositivos reales suelen tener un límite de alrededor de 32GB de RAM instalada.



La cantidad de memoria que se puede instalar en una computadora depende más de la capacidad de la tarjeta madre que del microprocesador. (Cortesía Patriot Memory)

programas que requieran de varios GB de espacio disponible para almacenarse, y esto evidentemente se refleja en una mayor cantidad de RAM que necesitan para su ejecución. A esto se debe añadir que los usuarios modernos gustan de mantener

¿Por qué es importante tener tanta memoria instalada? Principalmente por los siguientes factores: los programas de software son cada vez más poderosos y sofisticados, pero esto implica un crecimiento exponencial en su tamaño. No es raro encontrar en la actualidad



Los procesadores de la serie ARM son muy utilizados en sistemas de control, así como en teléfonos inteligentes y computadoras tipo tablilla. (Cortesía Samsung).

abiertas varias aplicaciones a la vez, y cada una de ellas consume una buena cantidad de RAM. Si a ello se suma el hecho de que los archivos con los que se trabaja normalmente también han aumentado de tamaño (ahora no es raro que alguien edite en su PC imágenes de muy alta resolución con un peso de varias decenas de MB, o fragmentos de video de varios cientos de MB de extensión), entonces se explica fácilmente porqué los usuarios de computadoras modernas están forzados a instalar la mayor cantidad de RAM posible en sus sistemas, para que todo funcione adecuadamente.

Aparte del ámbito de las computadoras PC, los microprocesadores de 32 bits siguen siendo los más utilizados; se usan como circuito principal de teléfonos celulares “inteligentes”, en circuitos de control para equipos electrónicos diversos, en paneles de sistemas industriales, etc. De entre todas las arquitecturas existentes, hay una que últimamente ha sobresalido, convirtiéndose en un estándar por derecho propio: los microprocesadores ARM. Estos micros suelen usarse en smartphones, en computadoras tipo tablilla, como control en impresoras multifuncionales, etc. Es importante considerar esta arquitectura, ya que su potencialidad apenas comienza a vislumbrarse.



Los microcontroladores también han pasado de ser dispositivos simples de 8 bits a poderosos chips de 16 o 32 bits, aptos para un gran número de aplicaciones. (Cortesía Microchip).

También en el ámbito de los microcontroladores se ha producido el paso de arquitecturas sencillas de 8 bits a dispositivos más complejos de 16 y 32 bits; por ejemplo, en la familia PIC de Microchip, la serie 18-24 ya son micros de 16 bits, mientras que la serie 32 son dispositivos de 32 bits. En el caso de Atmel y sus microcontroladores AVR, la serie normal es de 8 bits, pero su serie AT32UC3 ya es de 32 bits. Por el lado de Zilog, su línea Z8 es de 8 bits, pero la ZNEO es de 16 bits. Intel

maneja una amplia línea de microcontroladores de 16 y 32 bits, basados en la popular arquitectura x86; y así sucesivamente. En todos estos microcontroladores, también la memoria interna de programación ha tenido un incremento notable, pasando de unos cuantos kilobytes hasta chips que poseen decenas o cientos de kilobytes, e incluso más.

Resulta obvio por lo tanto, que entre mayor cantidad de bits utilice un procesador digital, más memoria podrá manejar, y tendrá la capacidad de ejecutar software más complejo y sofisticado. Esto se debe considerar en caso de encontrar alguna aplicación donde un micro de 8 bits no sea suficiente; siempre existe la opción de dar el salto y utilizar dispositivos de 16 o 32 bits, seguramente alguno cubrirá los requerimientos necesarios.

ACTIVIDAD DE APRENDIZAJE 5B

Contesta lo siguiente:

- 1.- ¿De cuántas líneas es el bus de direcciones de un procesador típico de 8 bits?
- 2.- ¿Cuánta memoria puede manejar como máximo con ese bus?
- 3.- ¿De qué ancho era el bus de direcciones de los procesadores 8086?
- 4.- ¿Cuánta memoria podía manejar con ese bus?
- 5.- ¿De cuántos bits es el bus de direcciones en un i386?
- 6.- ¿Cuánta memoria puede manejar (en teoría)?
- 7.- ¿De cuántos bits era el bus de un Pentium-4? ¿Cuánta RAM podía manejar?
- 8.- ¿De cuántos bits es el bus de datos en un Athlon-64?
- 9.- ¿Cuánta memoria máxima podría manejar (teóricamente)?
- 10.- ¿Qué es lo que limita en la realidad la cantidad de RAM que puede instalarse en una computadora?

5.3 PROGRAMAS DISEÑADOS PARA 16 Y 32 BITS

El mayor poder de los microprocesadores de 16 y 32 bits generalmente se acompaña por un aumento significativo en el número de comandos básicos que forman su set de instrucciones; y esto es natural si se considera que en los micros de 8 bits, la extensión de esta palabra limita el número de instrucciones básicas a un máximo de 256 (las combinaciones posibles con 8 bits); cuando los microprocesadores alcanzan 16 bits, esto significa que los diseñadores tienen más de 65,000 combinaciones distintas, así que pueden crear nuevos comandos que faciliten alguna tarea especialmente compleja, o que permita ahorrar pasos al momento de realizar alguna operación (obviamente aprovechar todas las combinaciones posibles no es el objetivo de los diseñadores, pero sí hacen uso del “espacio extra” para mejorar el set de instrucciones del dispositivo).

adc Add with carry flag	add Add two numbers	and Bitwise logical AND
call Call procedure or function	cbw Convert byte to word (signed)	cli Clear interrupt flag (disable interrupts)
cwd Convert word to doubleword (signed)	cmp Compare two operands	dec Decrement by 1
div Unsigned divide	idiv Signed divide	imul Signed multiply
in Input (read) from port	inc Increment by 1	int Call to interrupt procedure
iret Interrupt return	j?? Jump if ?? condition met	jmp Unconditional jump
lea Load effective address offset	mov Move data	mul Unsigned multiply
neg Two's complement negate	nop No operation	not One's complement negate
or Bitwise logical OR	out Output (write) to port	pop Pop word from stack

Sólo como ejemplo, se muestra el set de instrucciones del 8086 de Intel, el cual es la base de la plataforma x86, la más empleada en computadoras personales. Este set de instrucciones a primera vista parece simple; aparentemente sólo tiene 40 comandos básicos; el detalle está al expandir este set de instrucciones; por ejemplo, la orden "JUMP" tiene más de 30 variantes distintas, los comandos para hacer operaciones con dos cifras (AND, ADD, COMP, etc.) cada una tiene muchas variantes dependiendo de cuáles serán los operandos y en dónde se guardará el resultado; y así sucesivamente. Esto significa que el set aparentemente simple de 40 instrucciones, al final se convierte en varios cientos de comandos distintos.

Un set de instrucciones tan complejo evidentemente hace más difícil la tarea de los programadores, sobre todo si desean realizar su código en

lenguaje ensamblador, ya que tienen que lidiar con cientos y cientos de órdenes distintas, cada una diseñada para realizar su labor de cierta forma específica. Aunque no es imposible programar en ensamblador estos procesadores, lo normal en estos casos es que se recurra a una herramienta adicional, que simplifica considerablemente el desarrollo de las aplicaciones para estos dispositivos: los lenguajes de programación de alto nivel.

Prácticamente desde que surgió la computación, a mediados del siglo XX, aparecieron los primeros lenguajes de programación, que facilitaban el desarrollo de los programas que serían ejecutados por las enormes y primitivas máquinas de aquella época. Hay que recordar que en esos tiempos, si alguien deseaba que una computadora hiciera cierto



En las primeras computadoras, prácticamente se debían programar todas las tareas que se deseaban realizar con ellas. (Cortesía IBM).

cálculo, prácticamente tenía que realizar el programa por sí mismo, ya que las computadoras eran tan escasas y costosas, que no existía una industria de desarrollo de software, y cada programa estaba dedicado a una marca y modelo de equipo en particular.

Sin embargo, pronto comenzaron a aparecer lenguajes de programación “estándar”, con los cuales, un programa podía transportarse de una computadora a otra con relativa facilidad; surgen así lenguajes míticos como ALGOL, COBOL, Pascal, ADA, Fortran, etc., pero definitivamente, desde el punto de vista de microprocesadores modernos, dos lenguajes se han convertido en los más utilizados por los desarrolladores en todo el mundo: Basic y C. Existen versiones de estos lenguajes para prácticamente todos los microprocesadores y microcontroladores en producción, y esto tiene como ventaja que una aplicación programada en uno de estos lenguajes, puede ser

transportada a otras plataformas con relativa facilidad. Esto significa que se puede diseñar un programa en C o Basic, y luego conseguir el compilador adecuado, y cargarlo prácticamente sin modificaciones en un PIC, un AVR, un eZ8, un 8085, o el procesador que se utilizará.

En resumen, aunque todavía es posible programar en lenguaje ensamblador un microprocesador de 16, 32 o incluso 64 bits, la complejidad de su set de instrucciones interno hace que sea muy poco práctico; es aquí donde brillan con luz propia los lenguajes de programación de alto nivel, que con comandos estándar relativamente fáciles de aprender, permiten desarrollar de forma rápida y sencilla el código necesario para que el procesador realice la tarea deseada. Sólo como ejemplo, aquí se muestra el famoso programa "Hola mundo", escrito en tres lenguajes distintos: Basic, C y Java.

Ejemplos del programa "Hola mundo"

Basic:

```
print "Hola Mundo";
```

C:

```
#include <iostream.h>
using namespace std;

int main() {
    cout << ";Hola, mundo!" <<
endl;
    return 0;
}
```

Java:

```
import static
java.lang.System.out;

public class HolaMundo {
    public static void
main(String[] args)
    {
        out.print("Hola Mundo");
    }
}
```

De esto se podría desprender la idea de que todos los procesadores de 16 o 32 bits deben programarse usando lenguajes de alto nivel; sin embargo, en el caso de los microcontroladores, existen algunos cuyo set de instrucciones es tan simple, que aún es factible programarlos en ensamblador; un buen ejemplo de ello son los PIC 18-24, que a pesar de ser dispositivos de 16 bits, aún pueden programarse sin demasiado trabajo usando su set de instrucciones básico; no obstante, los mismos diseñadores de Microchip recomiendan que, si se desea utilizar un PIC32, lo mejor es usar las versiones de C que ponen a disposición en su página.

Por todo lo anterior, si se piensa explorar la posibilidad de utilizar microprocesadores o microcontroladores de 16 bits o más, sería muy

conveniente aprender algún lenguaje de alto nivel, como Basic o C; con esto, la tarea de desarrollar el código interno para ese dispositivo se simplifica de manera considerable.

ACTIVIDAD DE APRENDIZAJE 5C

Contesta lo siguiente:

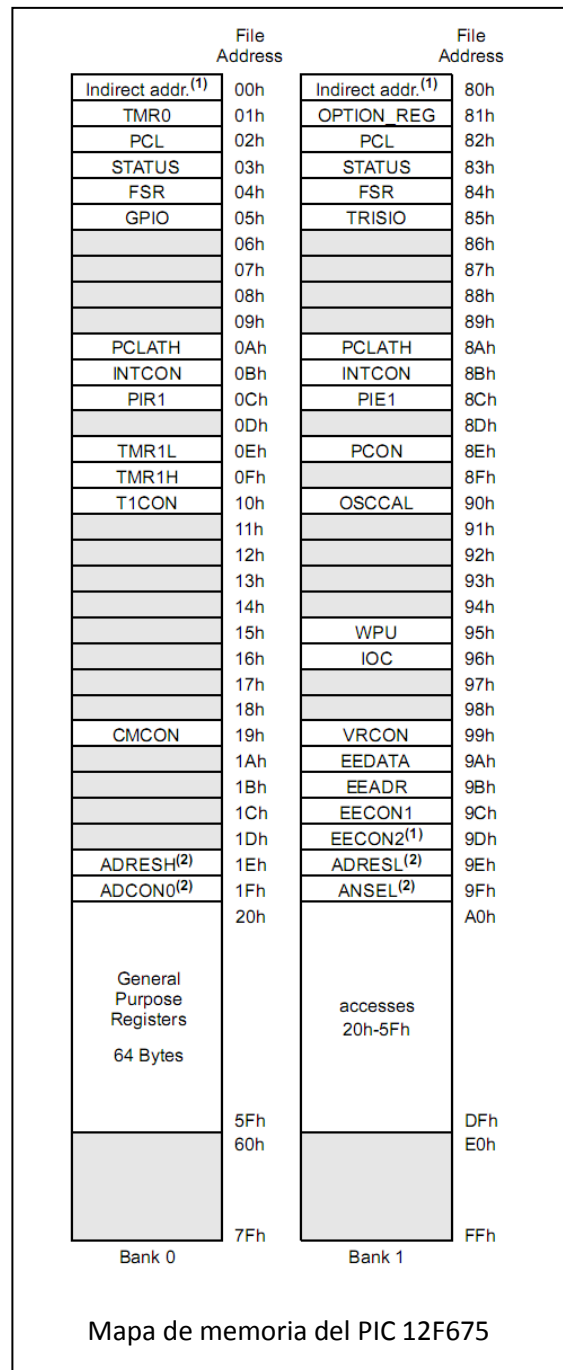
- 1.- ¿Por qué los microprocesadores de 8 bits deben tener menos de 256 comandos en su set de instrucciones?
- 2.- ¿Cuál es el número máximo teórico de instrucciones que podría tener un microprocesador de 16 bits?
- 3.- ¿Cuántas instrucciones básicas (sin variantes) posee el set de instrucciones de un 8086?
- 4.- ¿Es posible programar un microprocesador de 16 bits o más en ensamblador?
- 5.- ¿Qué son los lenguajes de programación de alto nivel?
- 6.- Menciona los dos lenguajes de programación de alto nivel más empleados en procesadores digitales:
- 7.- ¿Se pueden programar los microcontroladores en estos lenguajes?
- 8.- ¿Cuál es la ventaja de programar en Basic o C en lugar de ensamblador?
- 9.- ¿Qué se necesita para pasar un programa de C o Basic al lenguaje específico del microprocesador que se va a utilizar?
- 10.- ¿Cuáles lenguajes se recomienda conocer para programar código que funcione casi con cualquier tipo de procesador digital?

5.4 DIRECCIONAMIENTO DE UNIDADES O BLOQUES DE MEMORIA

Cuando un procesador necesita acceder a la memoria donde se guarda ya sea el programa o las variables de éste, lo normal es que la RAM tenga que estar dividida en bloques para llevar un mejor control de los procesos de lectura y escritura; esto significa que durante el desarrollo del código correspondiente, se deberá tener mucho cuidado de direccionar la unidad o bloque de memoria adecuada, dependiendo de la tarea que desee realizar.

Se puede decir que este problema ya se ha enfrentado en unidades anteriores, específicamente en la unidad 3, donde se describió el ejemplo del semáforo programado en lenguaje ensamblador para el PIC; ahora se usará un PIC aún más sencillo para mostrar qué significa el direccionamiento de bloques de memoria, y por qué es tan importante tomarlo en cuenta al desarrollar el código de un microprocesador o microcontrolador.

En la figura anexa se muestra el mapa de registros de un PIC 12F675, el cual es un microcontrolador de 8 bits en un encapsulado de apenas 8 terminales. De esos 8 pines, 2 se usan para alimentar al micro, así que quedan 6 para actuar como puertos generales I/O. Esto significa que habrá en la memoria un registro llamado GPIO que es donde se escribirán o leerán los datos que van o vienen de dichas



terminales, y un registro TRISIO que permite determinar si una terminal funcionará como entrada o como salida de datos.

Sin embargo, se puede observar que el registro TRISIO se encuentra en el segundo banco de memoria (dirección 85h), mientras que el GPIO está en el primero (dirección 05h). Existen registros importantes como el PC, STATUS, INTCON, etc., que se repiten en ambos bancos; mientras que hay otros que sólo están en el banco 0 o en el banco 1.

Si se analiza el contenido del registro STATUS, se encontrará lo siguiente:

Registro STATUS							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRP	RP1	RP0	TO	PD	Z	DC	C

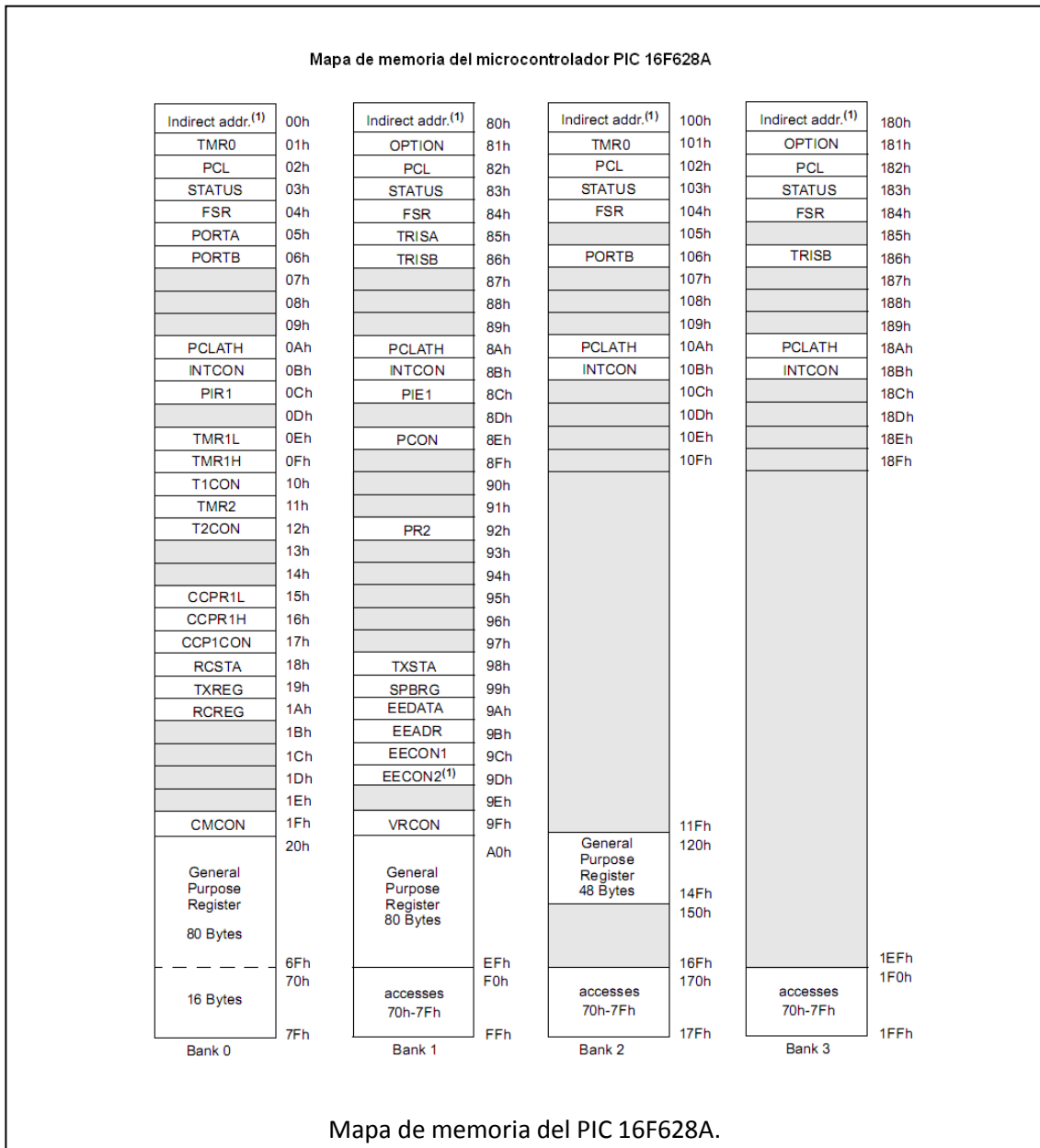
Y al consultar en el manual para qué sirve cada bit, se indica que la selección de banco de memoria se lleva a cabo con RP0. Si este bit está en “0”, se tendrá acceso al banco de memoria 0, pero si está en “1”, ahora se estará trabajando con el banco de memoria 1. Esto significa que si al principio del programa se desea configurar si las terminales del micro serán entradas o salidas, se tendrán que colocar sendos “1” o “0” en el registro TRISIO, pero para poder hacer esto, primero se debe seleccionar el banco de memoria adecuado. En código de PIC, esto se haría de la siguiente forma:

```

; Cambiamos al banco de memoria 1
      BSF      STATUS, RP0
; Ya ubicados en el banco 1, colocamos 3 pines como entrada y 3
; como salida
      MOVLW   b'00111000'
      MOVWF   TRISIO
; Terminada la configuración de terminales, regresamos al banco 0
      BCF      STATUS, RP0
; Continúa el programa

```

Entonces, resulta obvia la importancia de la correcta selección del banco de memoria que se está utilizando en un momento dado. Se debe tener especial cuidado en este aspecto sobre todo si se utilizarán las funciones complejas del microcontrolador, como los convertidores A/D, la memoria EEPROM, etc., ya que los registros con los que se controlan estas funciones están ubicados tanto en banco 0 como en banco 1, así que durante la configuración hay que ser muy cuidadosos de qué registro se está modificando a cada momento.



Obviamente, si se usa un procesador más poderoso (como el 16F628A que se ha venido utilizando), cuya memoria esté dividida en una mayor cantidad de bloques, entonces tendrá que aumentar el número de bits que se necesitará modificar para poder acceder correctamente a esos bancos. En el caso del 16F628, se cuenta con 4 bancos de RAM, y por esta razón, ahora el cambio entre ellos se realiza modificando las opciones RP1 y RP0 del registro STATUS, de modo que si RP1-RP0 son "00", se tiene acceso al banco 0; una combinación "01" da acceso al banco 1, "10" al banco 2, y finalmente un "11" al banco 3.

Sin embargo, existe un caso especial de registros de propósito general que pueden ser utilizados prácticamente desde cualquier banco. Se puede notar que en el mapa de memoria del 16F628 se señalan a los últimos 16 bytes del banco 0 (los que van de dirección 70h a 7Fh) como especiales, y es que se puede acceder a esas localidades de RAM desde cualquiera de los otros bancos; las direcciones F0h a FFh, 170h a 17Fh y 1f0h a 1FFh, redireccionan hacia 70h-7Fh; y esto implica que si se define alguna de las variables dentro de este lapso (70h-7Fh), se tendrá acceso a esa variable independientemente del banco de memoria.

Por todo lo anterior, resulta fundamental para la correcta ejecución de un código, poner especial cuidado en el uso de los bancos de memoria, y seleccionar siempre el adecuado para la función que se desee hacer o modificar. Esta situación no es exclusiva de los circuitos PIC, se encuentra también en prácticamente todos los microprocesadores de otras marcas, así que se debe tener mucho cuidado con este aspecto, sobre todo al programar en lenguaje ensamblador (lenguajes de alto nivel, como el Basic o el C, por lo general te piden elegir el procesador que vas a usar, y al momento de compilar el código, automáticamente hacen la conmutación de bancos cuando es necesario, quitándote esa preocupación de encima).

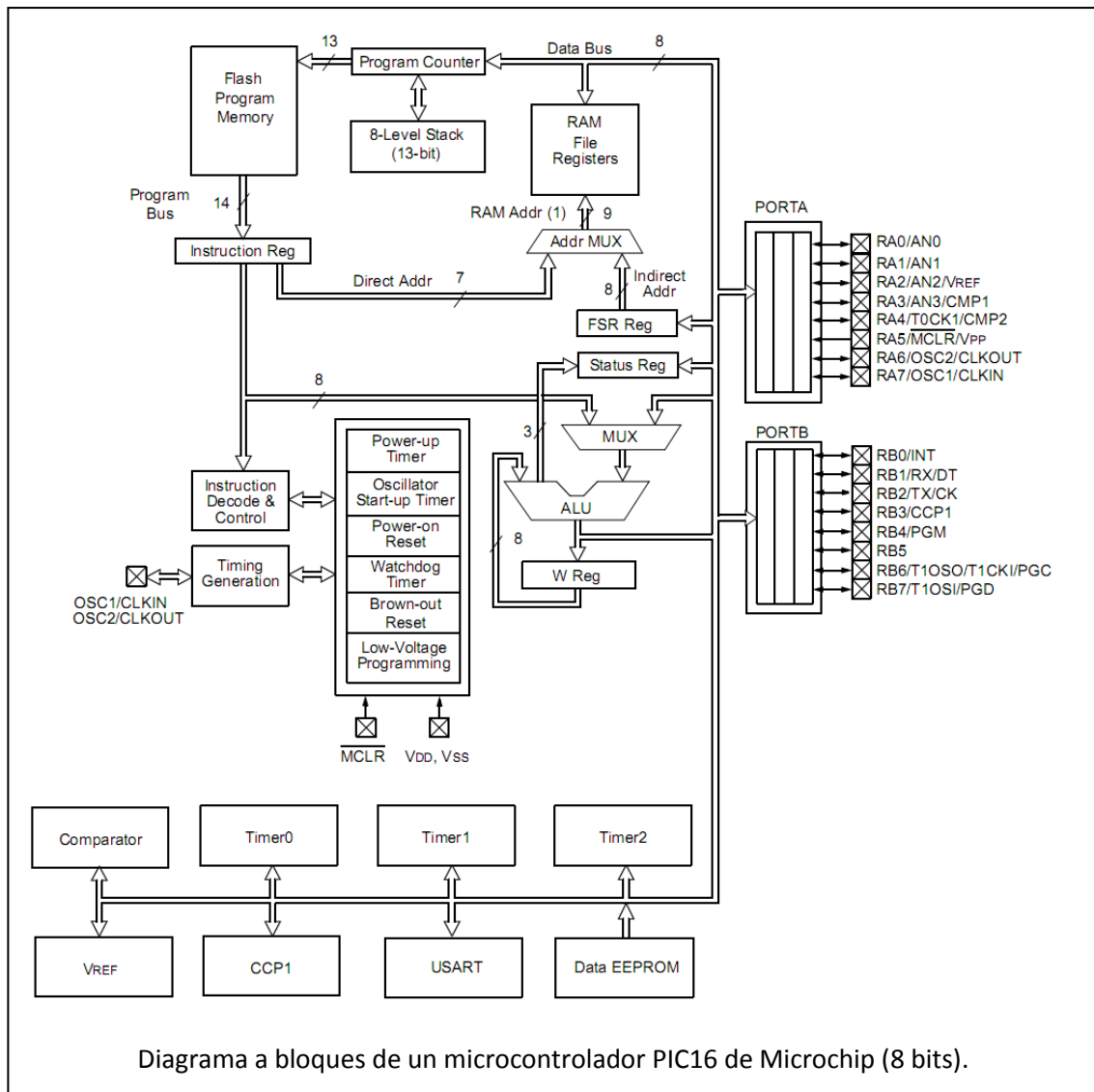
ACTIVIDAD DE APRENDIZAJE 5D

Contesta lo siguiente:

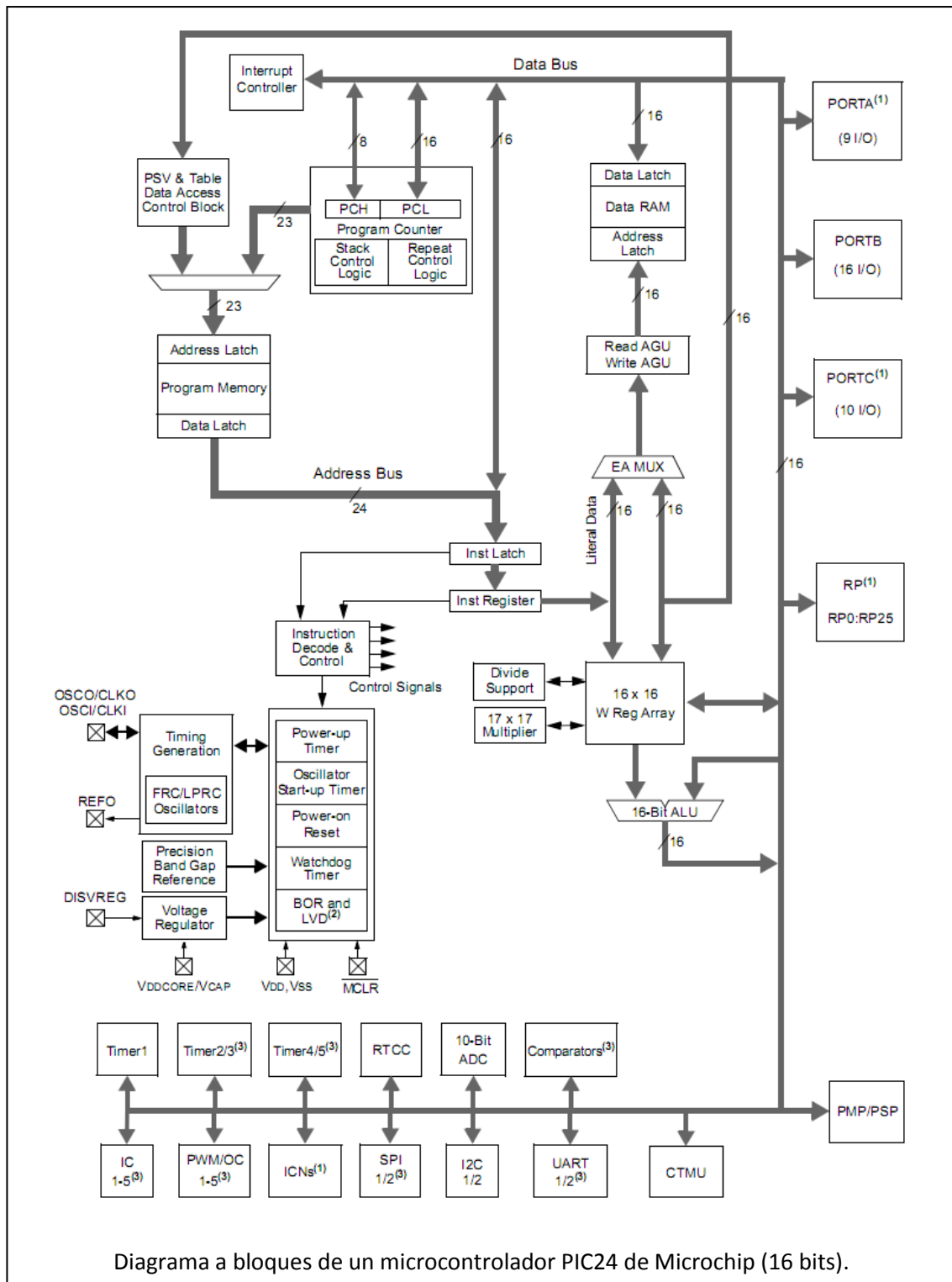
- 1.- ¿Cómo se encuentra normalmente la memoria de un procesador digital, como un bloque único o dividido en unidades?
- 2.- La RAM interna de los procesadores PIC, ¿está dividida?
- 3.- ¿En qué registro se encuentran los bits para seleccionar el bloque de memoria utilizado?
- 4.- ¿Cuáles son los bits que se usan para hacer la selección de bloque de RAM?
- 5.- ¿Por qué hay registros que se repiten en dos o más de los bloques de memoria?

5.5 ARQUITECTURA Y SET DE INSTRUCCIONES PARA MICROPROCESADORES DE 16 Y 32 BITS

El aumento en el número de bits que puede manejar un microprocesador no sólo se traduce en un mayor número de comandos en su set de instrucciones; generalmente también está acompañado por un incremento en la complejidad de la arquitectura interna del micro en sí, aumentando la cantidad de registros, añadiendo bloques funcionales, mejorando los métodos de intercambio de datos internos, etc., y todo esto se traduce en dispositivos más flexibles y poderosos, capaces de realizar muchas más tareas en menos tiempo que circuitos equivalentes de generaciones anteriores.

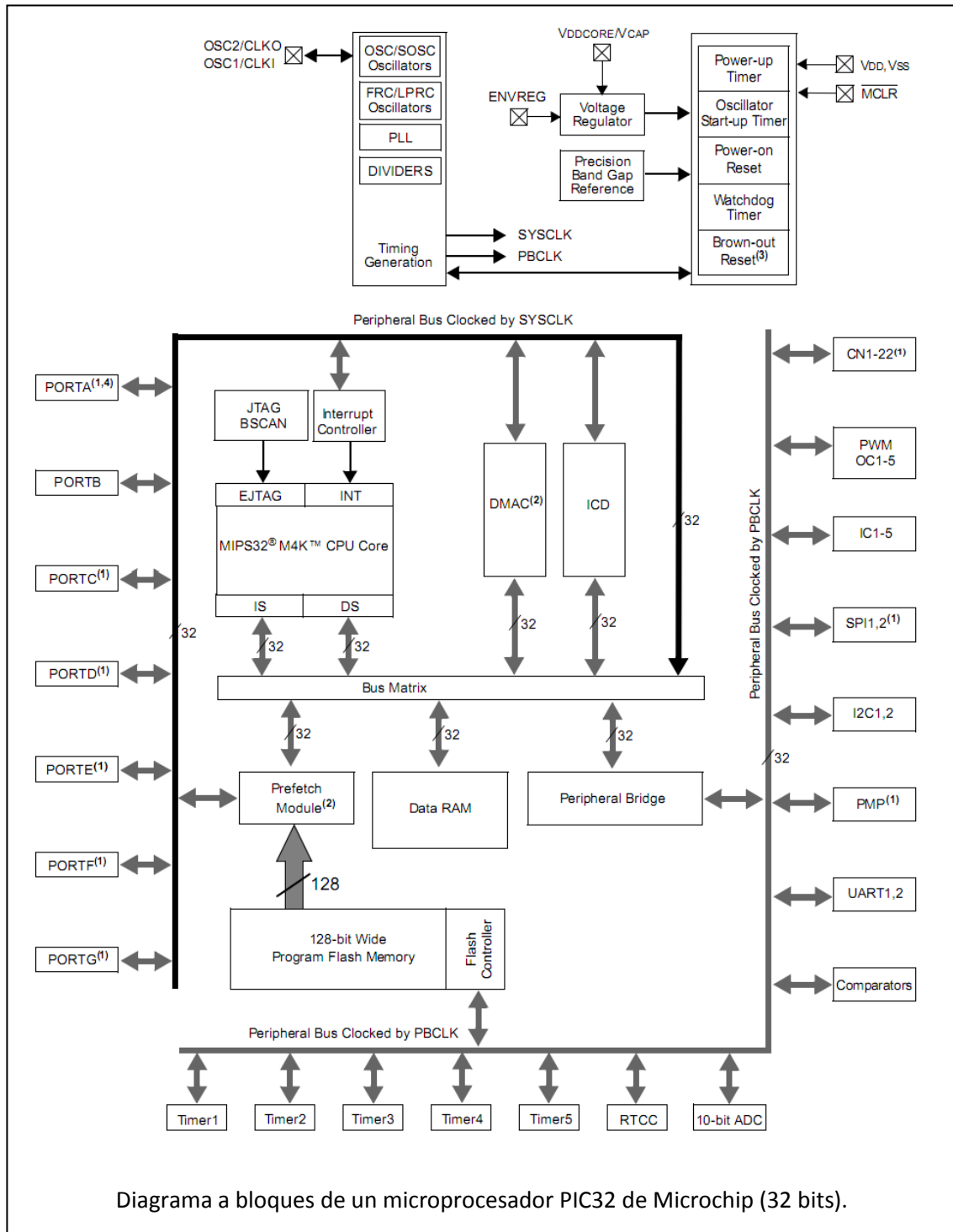


Este aumento en la complejidad interna del micro casi siempre está acompañado por un incremento proporcional del número de comandos básicos en su set de instrucciones, como ya se había comentado en los puntos anteriores. Esto hace que en el caso de los microprocesadores PIC, los dispositivos de 8 bits tengan un set de instrucciones de apenas 35 comandos simples; la serie PIC18-24 de 16 bits maneje más de 70 comandos básicos, mientras que la serie PIC32 posee más de 120 comandos básicos, algunos con variantes que aumentan su complejidad.



Para tener una idea de cómo ha ido evolucionando la arquitectura interna de la familia PIC, a continuación se muestran los diagramas a bloques de un

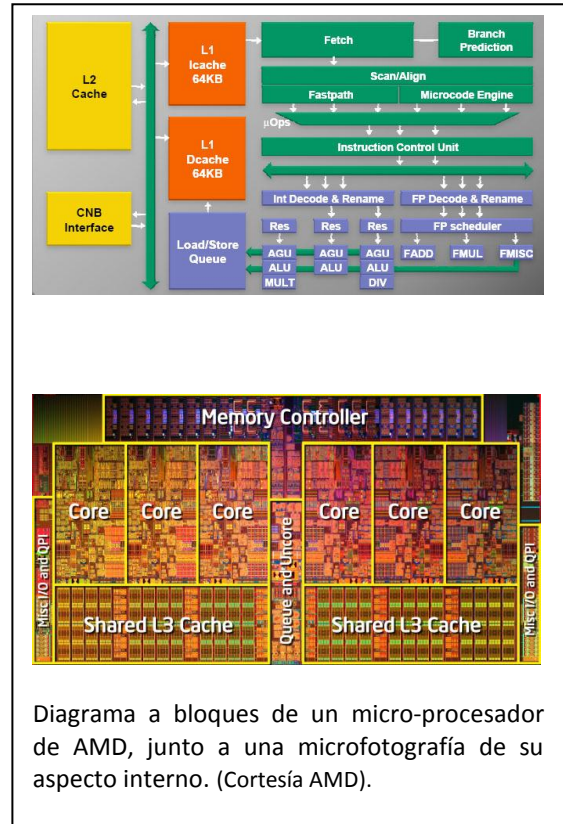
PIC16 de 8 bits, así como de un PIC24 de 16 bits, y finalizando con uno de la familia PIC32, de 32 bits.



Basta dar un vistazo a estos diagramas para apreciar cómo ha ido creciendo la complejidad interna de estos dispositivos, aunque son

microcontroladores relativamente económicos para aplicaciones diversas, que van desde proyectos escolares a tableros de control industriales, pasando por experimentos de robótica, sistemas de seguridad, etc.,

¡Imagínate cómo habrán evolucionado los microprocesadores que usan las computadoras personales! Generalmente, por estos dispositivos los usuarios están dispuestos a pagar una buena cantidad de dinero con tal de poseer un procesador rápido y poderoso, además de que las dos empresas principales que los fabrican, AMD e Intel, están en una cerrada competencia para ver quién ofrece el mejor micro y al menor precio. Esto ha hecho que la estructura interna de un micro moderno sea extremadamente compleja, tanto que ya se integran en



su interior muchos bloques que antes eran circuitos periféricos, todo con el objeto de mejorar el desempeño de la PC, sin que eso afecte demasiado su precio final.

Sin embargo, aquí hay una situación particular: cuando los microcontroladores aumentan el número de bloques internos, por lo general esto implica la adición de nuevos periféricos con tareas de lo más diversas; sin embargo, los microprocesadores de computadoras personales teóricamente no deberían tener ningún periférico incorporado, limitándose a cumplir su labor de realizar los cálculos matemáticos y lógicos solicitados. Entonces, ¿cómo puede aumentar la complejidad de estos micros según avanzan las generaciones de PC? Lo que están haciendo los fabricantes es aumentar el número de bloques auxiliares internos, como líneas de ejecución, memoria caché, interfaces con

dispositivos externos (como la RAM), e incluso algunos ya están rompiendo la regla e incorporando dentro del micro bloques que normalmente estaban afuera del mismo, como controladoras de memoria, procesadoras de gráficos, etc. Un microprocesador para PC moderno es toda una pieza de ingeniería y diseño, en cuyo interior hay cientos de millones de transistores trabajando al unísono.

Regresando a la cuestión del set de instrucciones de los procesadores. A continuación se muestra un ejemplo de cómo son las instrucciones en lenguaje ensamblador para un microprocesador de 16 bits de la serie x86; se tienen algunas líneas extraídas de programas reales, con una explicación rápida de qué significa cada una.

Comando	Significado
xor cx, [59507]	Operación en modo directo (XOR entre CX y la palabra en DS:E873)
push word [bx]	Operación en modo de registro indirecto (coloca la palabra que está en DS:BX en el Stack)
mov ax, [bp-4]	Operación en modo base (mueve la palabra en SS:(BP-4) dentro de AX)
sub [si+2], bx	Operación en modo indexado (Subtrae BX de la palabra en DS:(SI+2))
not byte [bp+di]	Operación en modo base indexado (Invierte los bits del byte en SS:(BP+DI))
add [bx+si+2], dx	Operación en modo base indexado con desplazamiento (Suma DX a la palabra en DS:(BX+SI+2))

Resulta obvio al observar la complejidad de estas instrucciones, que intentar programar estos dispositivos utilizando directamente lenguaje ensamblador resulta una tarea titánica, y esta es la razón por la que generalmente, los microprocesadores de 16 bits o más se programan usando algún lenguaje de alto nivel, generalmente C o Basic. Sin embargo, y como se ha mencionado, eso no impide que alguien intente programar estos dispositivos directamente en

Ejemplo de un programa para PIC24 escrito en ensamblador:

```
.title " Muestra de código en
ensamblador para dsPIC"
.sbttl " LED parpadeante"

.include "p24HJ12GP201.inc"
.text
.global __reset
__reset:
mov #0x0a00,W8
mov W8,SPLIM
mov #0x0980,W15
mov #0xfc,W4
mov W4,TRISB

mloop: mov #0x02,W5
mov W5,PORTB
call delay

mov #0x01,W5
mov W5,PORTB
call delay

bra mloop

delay: mov #0x7fff,W6
dloop: dec W6,W6
cp0 W6
bra NZ,dloop
return
.end
```

ensamblador, si se siente con la habilidad necesaria para hacerlo. Como ejemplo, en el cuadro que se ubica a un lado, se muestra un programa muy simple para un microcontrolador de la familia PIC24, escrito en ensamblador, que lo único que hace es que un LED colocado en una de sus terminales parpadee; se puede apreciar que, a pesar de lo sencillo de la tarea, el programa resulta relativamente complejo.

Así, finaliza esta unidad y este libro, en este momento se deberá tener una visión mucho más amplia y precisa de lo que son los microprocesadores y microcontroladores digitales, así como sus usos y alcances.

ACTIVIDAD DE APRENDIZAJE 5E

Contesta lo siguiente:

- 1.- Si aumenta el número de bits de una plataforma, ¿su set de instrucciones queda igual?
- 2.- ¿Cambia la arquitectura interna de un procesador lógico al aumentar el número de bits que maneja?
- 3.- ¿Qué bloques se han añadido recientemente a los microprocesadores de plataforma PC?
- 4.- Investiga qué es una línea de ejecución, y por qué conviene que un microprocesador tenga varias funcionando al mismo tiempo:
- 5.- ¿De cuántos bits es la familia PIC12-16? ¿Y la familia PIC24?

AUTOEVALUACIÓN

- 1.- ¿Cuál fue el primer microprocesador de 16 bits? ¿Y el de 32 bits? ¿Y el de 64 bits?
- 2.- ¿Cuál fue el primer microprocesador que rompió la barrera de 1MB de RAM?
- 3.- ¿Cuánta memoria puede direccionar teóricamente un microprocesador de 32 bits?
- 4.- ¿A qué se le denomina “memoria protegida” y por qué es importante?
- 5.- ¿Por qué conviene tener mucha memoria instalada en una computadora?
- 6.- ¿Qué tipo de microprocesador usan los smartphones y las computadoras tipo tablilla?
- 7.- ¿Por qué no es conveniente tratar de programar un procesador de 16 bits o más en lenguaje ensamblador?
- 8.- Entonces, ¿cómo se deben programar?
- 9.- ¿Qué precauciones hay que tomar cuando un procesador se comunice con su memoria, si ésta se encuentra dividida en bloques?
- 10.- El aumento en el número de bits de un microprocesador, ¿afecta su arquitectura interna?, ¿cómo?

RESPUESTAS

- 1.- 16 bits – 80286 de Intel; 32 bits – 80386 de Intel; 64 bits – R3000 de MIPS Technologies.
- 2.- El 80286 de Intel, primer microprocesador en alcanzar 16MB de RAM.
- 3.- Hasta 4GB de memoria.
- 4.- A la capacidad que tiene el microprocesador de asignar bloques de memoria distintos a diferentes programas, logrando así una multitarea sin problemas.
- 5.- Para lograr procesar los programas cada vez más complejos, y para poder tener varias aplicaciones abiertas a la vez.
- 6.- Procesadores tipo ARM.
- 7.- Porque su set de instrucciones es demasiado largo y complejo, difícil de dominar.
- 8.- Usando lenguajes de programación de alto nivel, como el Basic o el C.
- 9.- Activar adecuadamente los bits de selección de bloque de memoria.
- 10.- Normalmente, los diseñadores añaden bloques funcionales, para hacer más poderosos sus dispositivos.

Soluciones de las actividades de aprendizaje:

Actividad de aprendizaje 5A

- 1.- El Z-80 de Zilog.
- 2.- El 8086 diseñado por Intel.
- 3.- Hasta 1MB de RAM.
- 4.- El 8088.
- 5.- El 80286 de Intel.
- 6.- Hasta 16MB de RAM.
- 7.- El 80386 de Intel.
- 8.- Hasta 4GB de RAM.
- 9.- El R3000 de MIPS Technologies.
- 10.- El Athlon-64 y el Phenom de AMD, o el Core i3, i5 o i7 de Intel.

Actividad de aprendizaje 5B

- 1.- De 16 líneas.
- 2.- Hasta 64KB de RAM.
- 3.- De 20 líneas.
- 4.- Hasta 1MB de RAM.
- 5.- De 32 líneas.
- 6.- Hasta 4GB de RAM.
- 7.- 32 líneas, hasta 4GB.
- 8.- De 40 líneas.
- 9.- Hasta 1TB de RAM.
- 10.- La capacidad y número de ranuras en la tarjeta madre.

Actividad de aprendizaje 5C

- 1.- Porque ese es el número máximo de combinaciones con 8 bits.
- 2.- Más de 65,000 instrucciones.
- 3.- 40 instrucciones básicas.
- 4.- Sí, aunque es muy complejo.

5.- Son aplicaciones que permiten programar en un lenguaje estándar, y luego pasarlo al lenguaje del micro específico que se esté usando.

6.- Basic y C.

7.- Sí, existen versiones para casi todos los microcontroladores.

8.- Que el programa se puede trasladar fácilmente a otros micros.

9.- Un compilador.

10.- Basic y C.

Actividad de aprendizaje 5D

1.- Casi siempre se encuentra dividido en bloques o unidades.

2.- Sí, por lo menos en dos bloques.

3.- En el registro STATUS.

4.- Los bits RP0 y RP1.

5.- Para poder acceder a ellos sin importar en qué bloque se encuentre el programa.

Actividad de aprendizaje 5E

1.- No, por lo general se diseñan nuevas instrucciones para dar más flexibilidad al dispositivo.

2.- Sí, se aumentan bloques funcionales para hacer más poderoso al procesador.

3.- Controladora de memoria, memoria caché, controladora de gráficos, nuevos núcleos de ejecución, etc.

4.- Una línea de ejecución es una serie de bloques capaces de realizar operaciones lógicas dentro de un procesador; si hay varias de ellas, se pueden procesar varias instrucciones al mismo tiempo.

5.- Familia PIC12-16 – 8 bits; familia PIC24 – 16 bits.

BIBLIOGRAFIA

Atmel, *AVR microcontroller family*, Atmel Corporate.

Bestofmedia Group, *Tom's microprocessors guide*, TG Publishing.

Freescale, *68HC11E Family Datasheet*, Freescale Semiconductor.

Intel, *805x User Manual*, Intel Corporation.

Intel, *386 Programmer's Reference Manual*, Intel Corporation.

Intel, *8080A Microprocessor Datasheet*, Intel Corporation.

Microchip, *PIC 16 Family Reference Manual*, Microchip Technology.

Microchip, *PIC 16F627A/628A/648A Manual*, Microchip Technology.

Mueller, Scott, *Mantenimiento y reparación de PCs*, Anaya Multimedia, 2010.

Zilog, *Z-80 CPU User Manual*, Zilog Press.

GLOSARIO

ADC: Convertidor de análogo a digital; circuito que puede convertir una señal análoga en su equivalente en numeración binaria.

ALU: Arithmetic-Logic Unit o unidad aritmética-lógica; bloque interno de un procesador que se encarga de realizar las operaciones binarias necesarias para el funcionamiento de ese circuito.

ARM: Familia de microprocesadores que se ha popularizado recientemente, debido a su uso intensivo en teléfonos inteligentes, computadoras tipo tablilla y procesos de control avanzados.

AVR: Familia de microcontroladores fabricada por Atmel, muy populares por su bajo costo y alto poder de cálculo, además de que son muy fáciles de programar.

Bit: Unidad mínima de información en lógica binaria; sólo puede tomar el valor de "0" o "1".

Bus: Conjunto de líneas que entran o salen de un circuito de proceso de datos, y que sirven para transportar información o instrucciones entre dos o más circuitos digitales.

Byte: Conjunto de 8 bits que se manejan como un bloque funcional.

CEN: Chip Enable o habilitador de chip; señal que le indica a un circuito periférico que debe comenzar a funcionar.

CLK: Otro nombre con el que se conoce a la señal de reloj, necesaria para que funcione un circuito de proceso digital de datos.

DAC: Convertidor de digital a análogo, circuito que recibe un número digital y lo transforma en un voltaje análogo equivalente.

EEPROM: *Electrically Erasable Programmable Read Only Memory* o memoria de sólo lectura borrable eléctricamente; tipo de memoria que puede grabarse y borrarse según lo requiera el programa, pero que no pierde sus datos en caso de que se elimine la fuente de alimentación al chip.

Ensamblador: Lenguaje de programación que utiliza únicamente el set de instrucciones básico de un circuito de proceso digital.

Flash: Tipo de memoria que puede grabarse y borrarse varias veces, pero que no pierde sus datos cuando se retira la alimentación del circuito. Se utiliza en microcontroladores genéricos, como los PIC, los AVR y otros más.

GND: Referencia de tierra, nombre que se da genéricamente a la terminal (-) de la fuente de poder.

Interrupciones: En lenguaje de procesadores digitales, señales que le indican al circuito que tiene que atender algún proceso urgente, dejando de lado temporalmente lo que esté haciendo en ese momento.

Lenguajes de alto nivel: Lenguajes de programación que permiten diseñar el código de control de un circuito de proceso digital, usando comandos relativamente fáciles de aprender. Los más usados en microprocesadores y microcontroladores son C y Basic, aunque hay muchos más.

MCU: Siglas de *Micro Controller Unit* o unidad de micro-control. Otro nombre que reciben los microcontroladores.

Microcontrolador: Dispositivo de proceso de información digital, que incluye en su interior diversos bloques que le permiten interactuar de forma directa con su entorno, e incluso su programa principal se encuentra grabado en una memoria interna. Esto hace que sean dispositivos de aplicación muy específica, pero reduce considerablemente su costo de implementación.

Microprocesador: Dispositivo de proceso de información digital, capaz de realizar diversas operaciones con números binarios, para obtener un resultado que depende del programa que se esté ejecutando en ese momento. Son dispositivos de aplicación general, donde el software determina qué se estará haciendo en un momento determinado. Por lo general, necesita de una gran cantidad de elementos periféricos para poder comunicarse con su entorno.

MPU: Siglas de *Micro Processor Unit* o unidad de micro-proceso. Otro nombre que reciben los microprocesadores.

Periféricos: Circuitos que apoyan a los microprocesadores para realizar sus tareas, permitiéndoles comunicarse con su entorno, o almacenando las órdenes del software que se esté ejecutando.

PIC: Familia de microcontroladores fabricada por Microchip, son de los más populares entre los aficionados a la electrónica, por su bajo costo y facilidad de programación, combinada con un poder de cálculo razonable.

Puertos I/O: Bloque que permite a un microprocesador comunicarse con otros circuitos adyacentes, a través de varias terminales de entrada y salida de información.

PWM: Siglas de *Pulse Width Modulation* o modulación por ancho de pulso; método que se usa para simular una señal analógica por medio de una salida digital.

R/W: Nombre que recibe la línea que determina si se leerá o escribirá un dato en un momento dado (Read/Write). Normalmente forma parte del bus de control.

RAM: *Random Access Memory* o memoria de acceso aleatorio. Principal medio de almacenamiento temporal de datos en un circuito de proceso digital.

Registros: Localidades de memoria RAM que sirven para tener disponibles, de manera inmediata, los datos con que esté trabajando un circuito de proceso digital.

Reloj: Señal pulsante que sirve como base de tiempo para el funcionamiento de un circuito de proceso digital.

Reset: Pulso que se da al inicio de operación de un circuito de proceso digital, y que sirve para evitar que funcione mientras no se estabiliza el voltaje de alimentación. También sirve para reiniciar el sistema en caso necesario.

Set de instrucciones: Conjunto de comandos básicos que puede ejecutar un circuito de proceso digital de información.

USART: Siglas de *Universal Serial Asynchronous Receiver Transmitter* o transmisor-receptor asíncrono serial universal; periférico que se utiliza para la comunicación serial entre dos o más circuitos digitales.

Vcc: Voltaje de alimentación; es el voltaje que le proporciona al procesador la energía necesaria para realizar su trabajo. Nombre que recibe genéricamente la terminal (+) de una fuente de poder.

x86: Nombre que recibe la familia de microprocesadores iniciada por el 8086 de Intel, y que se extiende hasta los dispositivos más recientes, como los Core i3-i5-i7 de Intel o los Athlon-Phenom de AMD. Prácticamente, todas las computadoras personales de la actualidad utilizan microprocesadores de esta familia.