UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA

ESCUELA DE CIENCIAS BASICAS TECNOLOGIA E INGENIERIA

PROGRAMA DE INGENIERIA ELECTRONICA

309696 - MODULO DE MICROPROCESADORES & MICROCONTROLADORES

HECTOR URIEL VILLAMIL GONZALEZ
(Director Nacional)

MIGUEL PINTO APARICIO
Acreditador

CHIQUINQUIRA
Julio de 2009

INDICE DE CONTENIDO

LISTADO DE TABLAS

LISTADO DE GRÁFICOS Y FIGURAS

ASPECTOS DE PROPIEDAD INTELECTUAL Y VERSIONAMIENTO

El modulo en su versión 2004 fue diseñado por el Ingeniero Ing. Freddy Reinaldo Téllez Acuña, docente de la UNAD. El Ingeniero Freddy Reinaldo es Ingeniero Electricista y Magister en Potencia Eléctrica.

El presente módulo es un nuevo diseño enfocado en lograr un aprendizaje practico en la programación de microprocesadores y microcontroladores incorporando en su contenido algunos apartes significativos del anterior modulo y muchos temas nuevos entre ellos los referentes a microcontroladores Motorola Freescale y ejemplos de aplicación práctica en lenguaje assembler para Debugger, programación básica de HC08 y programación básica y avanzada con microcontroladores PIC. Este modulo ha sido desarrollado en Julio de 2009 por el Ing. Héctor Uriel Villamil González. URIEL VILLAMIL, se ha desempeñado como tutor de la UNAD en el CEAD de CHIQUINQUIRA, desde el año 2007 y se desempeña actualmente como director de curso a nivel nacional.

Este mismo año el Ing. MIGUEL PINTO APARICIO, tutor del CEAD Bucaramanga, apoyó el proceso de revisión de estilo del módulo y dio aportes disciplinares, didácticos y pedagógicos en el proceso de acreditación de material didáctico desarrollado en el mes de JULIO de 2009.

INTRODUCCIÓN

Los microprocesadores y microcontroladores representan la maravilla del desarrollo de la tecnología electrónica en más de medio siglo, los aparatos que los incorporan han cambiado la forma de trabajar e investigar de la humanidad, en la historia ninguna herramienta creada por el hombre, tenía la capacidad de crear otras y acelerar su evolución, en la actualidad muchos instrumentos, electrodomésticos y en general cualquier dispositivo electrónico utiliza alguno de estos dos componentes para optimizar su funcionamiento.

Los nuevos dispositivos traen consigo nuevas tecnologías aplicadas tanto a la electrónica del hardware como al desarrollo del software que lo controla, se relacionan entonces varios componentes que parten del microprocesador, tal es el caso del microcontrolador, que también es objeto de estudio de este curso. No solo encontramos microprocesadores y microcontroladores, también existen evoluciones de estos aplicados a situaciones particulares en la industria y consumo, tal es el caso de los DSP (Procesadores Digitales de Señales) y PLC (Controladores Lógicos Programables).

Las aplicaciones de control, medición, instrumentación, entretenimiento y consumo, son las promotoras y fuente del creciente mercado tanto de microprocesadores como de microcontroladores. El panorama es alentador por la expectativa de nuevos productos, donde las aplicaciones están limitadas por el ingenio y la imaginación de los programadores.

UNIDAD 1

Nombre de la Unidad	**MICROPROCESADORES**
Introducción	En esta unidad se presenta el microprocesador como dispositivo que ha definido la revolución tecnología de los últimos años. Con el desarrollo de cada lección se encamina al estudiante por la evolución histórica de este dispositivo, pasando por su estructura, arquitectura, unidades funcionales, sistemas numéricos, para llegar a tomar como puntos de referencia algunos dispositivos particulares categorizados en su arquitectura desde los 8 bits hasta los 64 bits. Finalmente, se toca el tema de la programación con assembler y algunos ejemplos para aplicados buscando el desarrollo de habilidades.
Justificación	La temática expuesta es de gran importancia en las áreas de ingeniería electrónica y afines como sistemas, telecomunicaciones, industrial, audio entre otras, es de notar que los microprocesadores hacen parte de toda la tecnología empleada y como profesionales de las áreas mencionadas es conveniente profundizar en su estudio, aplicación y programación.
Intencionalidades Formativas	Con esta Unidad se pretende que el estudiante conozca en profundidad los microprocesadores, su construcción, arquitectura, evolución, aplicaciones y programación.
Denominación de capítulos	Capitulo 1: Microprocesador, principios básicos. Capitulo 2: Familias de microprocesadores. Capitulo 3: Ensamblador (Assembler).

CAPÍTULO 1: MICROPROCESADOR, PRINCIPIOS BASICOS

LECCIÓN 1: INVENCIÓN Y EVOLUCIÓN HISTÓRICA DEL MICROPROCESADOR.

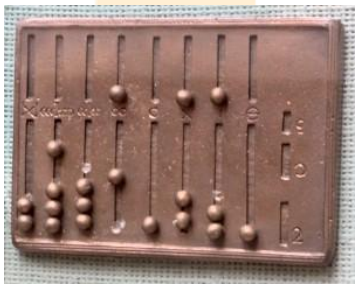
En los comienzos de la implementación de circuitos lógicos el diseñador debía poseer amplios conocimientos tanto en lógica digital como en los componentes y dispositivos que debía acoplar, esto presentaba inconvenientes, la llamada lógica cableada no permitía hacer modificaciones sin afectar la construcción física del circuito, en 1970 esto cambia con la aparición del microprocesador, con lo que aparece lo que se denomina Lógica programable, la cual permite modificar el comportamiento lógico digital del circuito sin tener que cambiar su configuración física.

Partiendo de las técnicas digitales consolidadas en los años sesenta, se comienza a desarrollar y profundizar en el estudio de las técnicas y desarrollo de aplicaciones para los microprocesadores junto con la programación en lenguaje maquina y la programación en assembler. Con esta breve introducción es hora de iniciar una exploración por los momentos históricos que han sido claves para la invención, evolución y desarrollo del microprocesador.

Primeros dispositivos para realizar cálculos

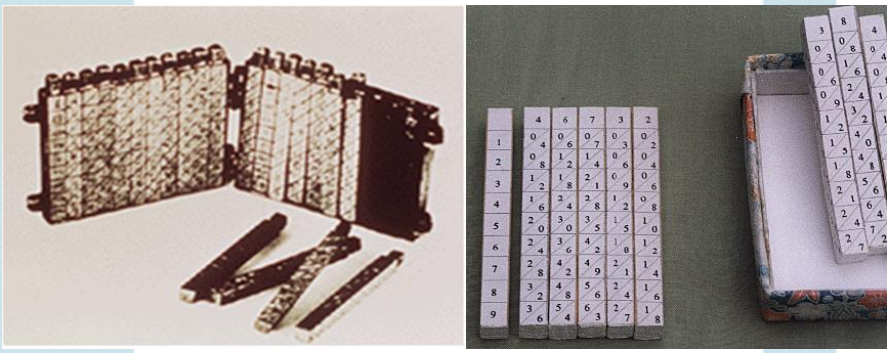
El dispositivo más reconocido es el Abaco, su origen se pierde en el tiempo algunos estiman cerca de 3000 años A.C otros entre 600 y 500 A.C en Egipto o China, su efectividad ha superado el pasar de los años.

Figura 1. ***Ábaco romano, Reconstrucción hecha por el RGZ Museum en Mainz, 1977. El original es de bronce y está en manos de la Biblioteca Nacional de Francia, en Paris***[1]



A comienzos del siglo XVI, el sistema decimal desplazó al sistema romano en la realización de cálculos complejos, John Naiper (1550-1617), matemático escoses, descubre los logaritmos y construye las primeras tablas de multiplicar.

Figura 2. ***Los "Naiper Bones"***[2]

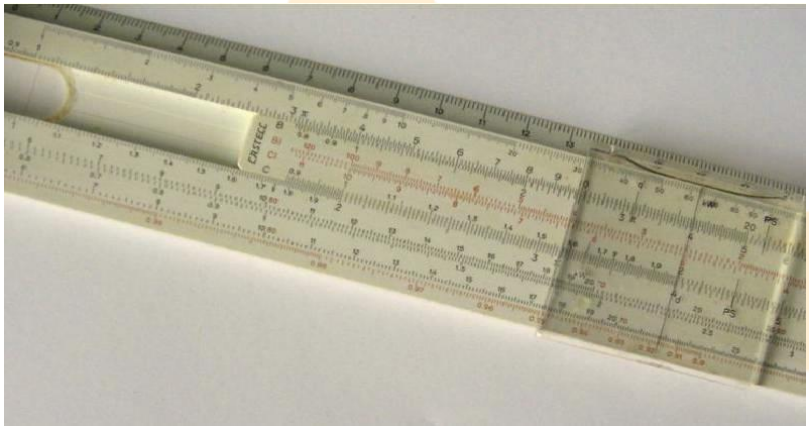


Dispositivos mecánicos

Basados en los logaritmos, surge las primeras máquinas analógicas de calculo derivadas de prototipos construidos por Edmund Gunter (1581-1626), matemático y astrónomo ingles y William Ougthred (1574-1660).

[1] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:RomanAbacusRecon.jpg
[2] Extraído el 10 de Julio de 2009 desde http://helmutsy.homestead.com/files/computacion/Historia/historia_computadores.htm

Figura 3. ***Los "Slide-rule" o regla de calculo*** [3]



El filosofo y matemático francés Blas Pascal (1623-1662) construye la primera maquina mecánica llamada "Pascalina", su funcionamiento se basaba en engranajes y ruedas con capacidad de sumar.

Figura 4. ***Pascalina***[4]

La pascalina se perfecciono en una maquina llamada "Calculadora Universal", desarrollada por Gottfried Leibniz (1646-1716), tenia la capacidad de realizar las cuatro operaciones aritméticas básicas.

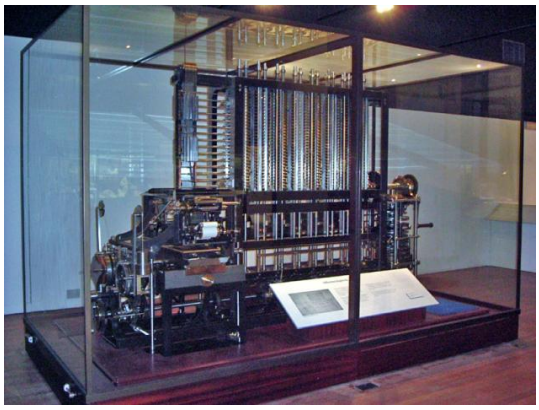
Otros avances significativos fueron la calculadora de 12 dígitos (1774) construida por Philipp-Matthaus Hahn; el telar de Jacquard (1801) controlado por tarjetas

[3] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Slide_rule_12.jpg
[4] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Arts_et_Metiers_Pascaline_dsc03869.jpg

perforadas para reproducir patrones de tejidos, por Joseph Jacquard; el Arithmometer (1820), primera calculadora construida a nivel industrial, ideada por el francés Xavier Charles Thomas de Colmar (1785-1870).

Charles Babbage (1791-1871), inventa una maquina llamada “Máquina de Diferencias o Máquina de Babbage”, este inventor británico elaboro los principios de la computadora digital moderna y junto con la matemática británica Augusta Ada Byron (1815-1852), se consideran los inventores de la computadora digital moderna. Babbage también ideo la “Maquina Analítica” (1833), como una maquina de calculo general aunque no se pudo construir debido a la complejidad y limitaciones tecnológicas de la época.

Figura 5. ***La réplica de la máquina diferencial del Museo de Ciencias de Londres***[5]

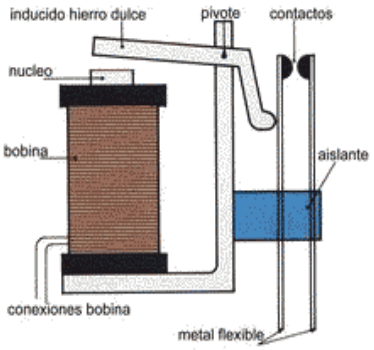


Era Electromagnética

Con el surgimiento de la electricidad y dispositivos como el motor eléctrico, los contactos eléctricos y dispositivos electromagnéticos como los relay o relevos o contactores los cuales al paso de la corriente eléctrica abren o cierran contactos permitiendo el bloqueo o paso de la corriente con lo que se puede representar dos estados estables, encendido y apagado, que en lógica digital son el uno (1) y cero (0), estos estados junto con el trabajo realizado por George Boole (1815-1864), con su teoría de lógica booleana llamada “El algebra de Boole”, son la base matemática en la lógica de los computadores modernos.

[5] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:050114_2529_difference.jpg

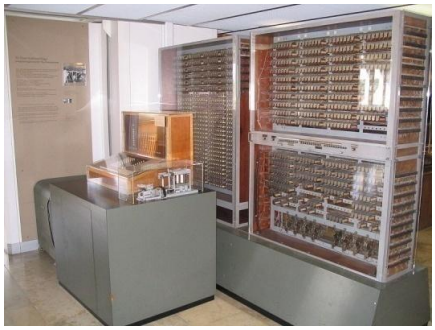
Figura 6. ***Relevo o relay***[6]



Se destaca la maquina “Tabuladora” o “Maquina de Hollerith” (1886), que registra datos utilizando contactos eléctricos. Hollerith creo una empresa para la distribución de esta maquina, esta empresa en 1924 paso a llamarse “International Bussiness Machines” o IBM.

Las maquinas que utilizaban motores eléctricos para movilizar mecanismos basados en la calculadores de Blaise Pascal dominaron el procesamiento de la información hasta la aparición de la “Z3” (1941) creada por Konrad Zuse, como la primera maquina programable y automática construida con 2300 relés, frecuencia de reloj de aproximadamente 5Hz y longitud de palabra de 22 bits, realizaba cálculos con aritmética de punto flotante en sistema binario.

Figura 7. ***Z3 (1941)***[7]



El primer computador electromecánico denominado “Mark I” (1944), construido por Howard H. Aiken, tenia 760.000 ruedas y 800 Kilómetros de cable, su funcionamiento se basa en la maquina analítica de Charles Babbage.

Era eléctrica

La invención del tubo de rayos catódicos o CTR (1897) por Karl Braun (1850-1918) y el diodo de válvula de vacio (1904) por Jhon A. Fleming que permite el

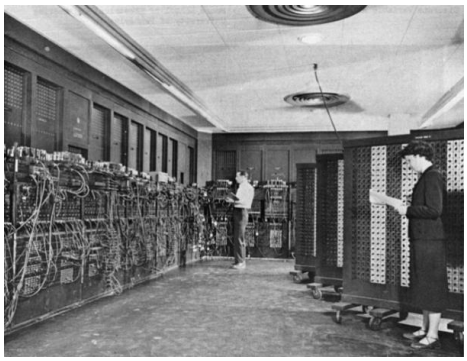
[6] Extraído el 10 de Julio de 2009 desde http://electricidadmectaronica.blogspot.com/2008/10/conceptos.html
[7] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Z3_Deutsches_Museum.JPG

mejoramiento en las comunicaciones, preceden la invención que define la tecnología del microprocesador, se hable del tubo al vacio (1906) por Lee de Forest (1873-1961), basado en el efecto Edison, este dispositivo es un diodo de válvula de vacio con una rejilla de control adicional creando la válvula de tres electrodos, con este dispositivo se hace posible la radio, la telefonía, la telegrafía inalámbrica, la televisión, etc. Impulsando también el desarrollo de la electrónica.

Colossus fue el primer dispositivo calculador y sistema de computo primitivo que uso tubos al vacio, utilizado por los británicos para descifrar comunicaciones alemanas durante la segunda guerra mundial, de este se fabricaron varios modelos que mejoraban el desempeño del anterior, llegando a utilizar 4200 válvulas.

ENIAC (Electronic Numerical Integrator And Computer), primera computadora de propósito general, desarrollada en la Universidad de Pennsylvania por Jhon Presper Eckert y John William Mauchly, completamente digital ejecutaba sus instrucciones en lenguaje maquina, la ENIAC se reprogramaba recableando sus circuitos, lo que tardaba varios días e incluso semanas, construida con mas de 17000 válvulas, mas de 7000 diodos de cristal, 1500 relés, 70.000 resistencias, 10.000 condensadores, con un peso de 27 toneladas, temperatura de operación de 50°C y consumo de 160 KW.

Figura 8. ***ENIAC (1946)***[8]



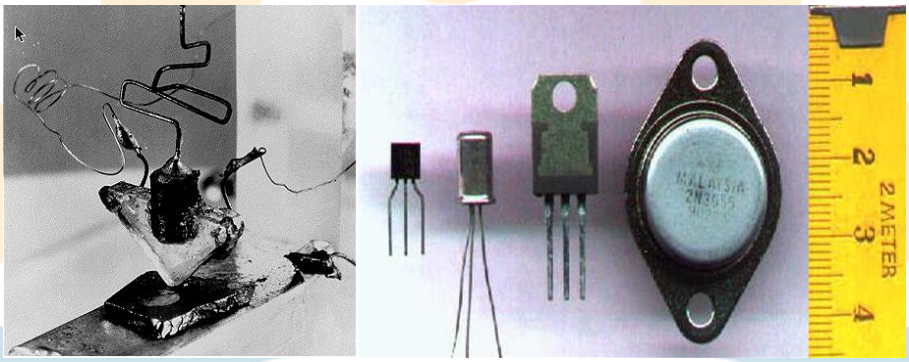
Era Electrónica

La revolución del transistor comienza con su desarrollo e invención en los laboratorios de Bell Telephone en 1948, su utilización comenzó hasta 1950, para

[8] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Eniac.jpg

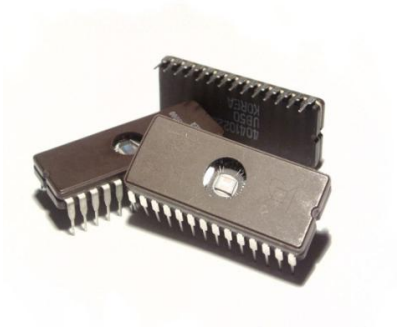
la fabricación de aparatos de radio, televisión, sonido, militar y la naciente tecnología de computadores.

Figura 9. ***Transistor (1948)***[9]



En 1958 se desarrolla el Circuito Integrado, por el Ingeniero Jack Kilby (1923-2005), consistía en un dispositivo de germanio que integraba seis transistores en una misma base semiconductora formando un oscilador de rotación de fase. El circuito integrado consiste en una pastilla pequeña de unos cuantos milímetros de área, construida a partir de silicio en la que se fabrican circuitos electrónicos basados en circuitos semiconductores, estos dispositivos se protegen del exterior por una cubierta de cerámica o plástico y se comunica al exterior por medio de unos contactos metálicos, llamados pines, patillas o contactos.

Figura 10. ***Circuito Integrado (1958)***[10]



El desarrollo del transistor, conduce al desarrollo de la primera familia lógica RTL (Lógica Resistencia Transistor), con la utilización de tecnologías de integración surge la tecnología de circuitos integrados digitales. Robert Noyce después de renunciar a Fairchild en 1968 en compañía de Gordon Moore y Andrew Grove, con el respaldo económico de Arthur Rock funda la compañía INTEL donde se dan los primeros pasos para el desarrollo del microprocesador, comenzando por el

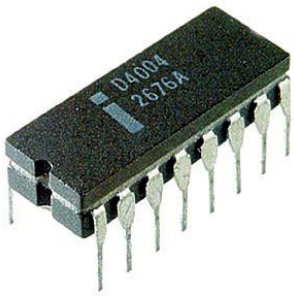
[9] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Transistor-photo.JPG
[10] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Microchips.jpg

desarrollo de una memoria basada en semiconductores y la lógica de un Flip-Flop como elemento de memoria capaz de almacenar un bit, se crea el primer circuito de memoria RAM llamado 1103 (1969) con una capacidad de 1024 bits.

Finalmente aparece el primer microprocesador el Intel 4004, lanzado al mercado en 1971, con una CPU de 4 bits, 2300 transistores, 16 pines (DIP-Dual In-line Package), máxima velocidad de reloj 740 KHz, con arquitectura Harvard, 46 instrucciones, 16 registros de 4 bits cada uno, stack de 3 niveles.

En 1972 se lanza al mercado el Intel 8008, empleaba direcciones de 14 bits, direccionando hasta 16KB de memoria, con la limitante de 18 pines en DIP, tiene un bus compartido de datos y direcciones de 8 bits, velocidad 0,5 a 0,8 MHz.

Figura 11. ***Circuito Integrado (1958)***[11]



El primer “PC” o computador personal se construye en IBM y se lanza al mercado en 1981, aunque en el mercado ya existían otros computadores fabricados principalmente por Apple, el modelo IBM PC causa gran impacto debido al concepto de compatibilidad, es decir, la estrategia de fabricar computadores con partes de distintos fabricantes con compatibilidad IBM, en su interior tenia un microprocesador 8088 de Intel.

Figura 12. ***IBM PC (1981)***[12]

[11] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Intel_4004.jpg
[12] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:IBM_PC_5150.jpg

Microprocesadores de Intel y otros fabricantes

Intel, Con el desarrollo del 8080 de 8 bits lanzado en abril de 1974 y posteriormente el 8085 binariamente compatible con el 8080 pero que tenía menos exigencia de hardware sigue una serie de avances y desarrollo de nuevos microprocesadores con características que mejoran el desempeño con respecto a sus predecesores. En los 80's aparecen los microprocesadores de 16 bits más potentes con la incursión al mercado del 8086 de Intel en 1978, utilizado por IBM para la fabricación de la gamma de IBM PC serie PS/2.

Intel prosigue el desarrollo del 8088 para IBM XT, 80186, el 80286 utilizado para IBM AT, el 80386, 80486, Pentium que fue un nombre comercial para evitar la duplicación de su nomenclatura por parte de AMD y Cyrex, mas adelante sigue con el desarrollo y producción de Pentium II, Pentium III, Pentium IV, hasta los actuales microprocesadores multinucleo.

Zilog, nace como una empresa fundada por personal que trabajo para Intel en el 8080, construyen el Z-80 que incorpora un conjunto de instrucciones mas extenso y compatible con el 8080.

Motorola, en la misma época del 8080 Motorola desarrolla el microprocesador de 8 bits 6800, su diseño es completamente distinto al 8080 pero tiene características similares, Motorola perfecciona el 6800 hacia el 6809 considerado uno de los mejores microprocesadores de la época aunque no tuvo éxito comercial, pero el 6502 derivado el 6800 logro posicionarse como pieza clave en la fabricación de las primeras computadoras personales PET de Commodore y Apple II de Apple Computer Inc. En un consorcio entre Apple, IBM y Motorola, se desarrolla una nueva familia de microprocesadores "los Power PC" los cuales se utilizan en las computadoras Apple de IBM actuales.

AMD, comienza a producir chips en 1969, en 1975 empieza con el desarrollo de memorias RAM, este mismo año introduce un clon del 8080 de Intel utilizando ingeniería inversa, su trabajo se redujo a procesadores embebidos logrando cierto éxito en los 80's, al no lograr el éxito esperado se concentra en el desarrollo de memorias Flash y procesadores Intel, empezando a desarrollar procesadores como el 386 que superaba ampliamente las prestaciones del Intel 80x386, siguió con el 486 para comenzar fabricando su primer procesador propio AMD K5, K6, K6-II, K6-III, Athlon y la ultima gamma multinucleo.

Cyrix, comienza a operar en 1988 como proveedor de coprocesadores matemáticos para el 286 y 386, la compañía fue fundada por Jerry Roges y ex trabajadores de Texas Instruments, al igual que AMD comienza clonando

procesadores de Intel comenzando con el 386 denominado por Cyrix 486SLC y 486 DLC lanzados en 1992 situando su rendimiento entre un 386 y 486 de Intel, siguen con los 486 compatibles con los de Intel, Cx5X86 compatible con Pentium y el 6x86 primer microprocesador de su línea en superar las prestaciones de Intel Pentium. En 1996 Cyrix lanza el microprocesador MediaGX y en 1997 se fusiona con National Semiconductor.

Cronología de los microprocesadores

- 1971: Intel 4004. Nota: Fue el primer microprocesador comercial. Salió al mercado el 15 de noviembre de 1971.
- 1972: Intel 8008
- 1974: Intel 8080, Intel 8085
- 1975: Signetics 2650, MOS 6502, Motorola 6800
- 1976: Zilog Z80
- 1978: Intel 8086, Motorola 68000
- 1979: Intel 8088
- 1982: Intel 80286, Motorola 68020
- 1985: Intel 80386, Motorola 68020, AMD80386
- 1987: Motorola 68030
- 1989: Intel 80486, Motorola 68040, AMD80486
- 1993: Intel Pentium, Motorola 68060, AMD K5, MIPS R10000
- 1995: Intel Pentium Pro
- 1997: Intel Pentium II, AMD K6, PowerPC G3, MIPS R120007
- 1999: Intel Pentium III, AMD K6-2, PowerPC G4
- 2000: Intel Pentium 4, Intel Itanium 2, AMD Athlon XP, AMD Duron, MIPS R14000
- 2003: PowerPC G5
- 2004: Intel Pentium M
- 2005: Intel Pentium D, Intel Extreme Edition con hyper threading, Intel Core Duo, AMD Athlon 64, AMD Athlon 64 X2, AMD Sempron 128.
- 2006: Intel Core 2 Duo, Intel Core 2 Extreme, AMD Athlon FX
- 2007: Intel Core 2 Quad, AMD Opteron Quad Core, AMD Quad FX
- 2008: Intel Core 2 Extrem, AMD Phenom X4, Athlon II X2
- 2009: Intel Core i7, i7 Extrem, AMD Phenom II

Los MIPS en los microprocesadores

El acrónimo MIPS proviene de millones de instrucciones por segundo, es una forma de medir la capacidad de procesamiento de los microprocesadores con el mismo juego de instrucciones, las comparativas presentan valores picos por lo que no son muy realistas.

En la siguiente tabla se expone la evolución de los microprocesadores en el tiempo siguiendo el comportamiento de los MIPS.

Tabla 1. ***Evolución en el tiempo de las instrucciones por segundo***[13]**.**

Procesador	IPS	Reloj	Año
Intel 8080	640 KIPS	2 MHz	1974
Intel 8086	800 KIPS	4.77 MHz	1974
Motorola 68000	1 MIPS	8 MHz	1979
Intel 486DX	54 MIPS	66 MHz	1992
PowerPC 600s (G2)	35 MIPS	33 MHz	1994
ARM 7500FE	35.9 MIPS	40 MHz	1996
PowerPC G3	525 MIPS	233 MHz	1997
ARM10	400 MIPS	300 MHz	1998
Zilog eZ80	80 MIPS	50 MHz	1999
Sony "Allegrex"(de la PSP)	32 MIPS	333MHZ	2002
Pentium 4 Extreme Edition	9726 MIPS	3.2 GHz	2003
ARM Cortex A8	2000 MIPS	1.0 GHz	2005
Xbox360 IBM "Xenon" Single Core	6400 MIPS	3.2 GHz	2005
AMD Athlon 64	8400 MIPS	2.8 GHz	2005
AMD Athlon FX-57	12000 MIPS	2.8 GHz	2005
AMD Athlon 64 Dual Core	18500 MIPS	2.2 GHz	2005
AMD Athlon 64 3800+ X2 (Dual Core)	18900 MIPS	2.2 GHz	2005
Overclocked AMD Athlon 64 3800+ X2 (Dual Core)	25150 MIPS	2.8 GHz	2005
Cell (cada PPE)	6400 MIPS	3.2 GHz	2006
Procesador Cell de la PlayStation 3	21800 MIPS	3.2 GHz	2006
AMD Athlon FX-60 (Dual Core)	22150 MIPS	2.6 GHz	2006
Overclocked AMD Athlon FX-60 (Dual Core)	24300 MIPS	2.8 GHz	2006
Overclocked AMD Athlon FX-60 (Dual Core)	27100 MIPS	3.0 GHz	2006

[13] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/MIPS

LECCIÓN 2: BASES NUMÉRICAS, BITS Y BYTES.

Bases numérica decimal

La forma de interpretar los números decimales es a través de su posición (lugar que ocupa el digito en el número total. i.e unidad, decena, centena, etc), que se pueden relacionar con las denominaciones de unidades, decenas, centenas, etc. Desde un punto de vista analítico, el sistema decimal tiene 10 símbolos representativos (0,1,2,3,4,5,6,7,8,9), para representar símbolos como “12” se usa una combinación de los símbolos mencionados. El principio del cálculo posicional puede hacerse extensivo a los demás sistemas numéricos mediante la siguiente ecuación:

$\sum_{n=0}^{\infty} D_n.B^n$ Donde **n** es la posición del número comenzando desde cero e incrementando con enteros positivos en la posición desde el punto decimal hacia la izquierda y con enteros negativos desde el punto decimal a la derecha, **D** es el número y **B** representa la base numérica de la cifra a convertir.

Ejemplo:

El decimal “125” podría representarse con punto decimal así, “125.00” entonces tomando desde el punto decimal hacia la izquierda y comenzando desde cero, el numero cinco (5) tendrá la posición cero (0), el numero dos (2) la posición uno (1) y finalmente el numero uno la posición (2), siguiendo lo establecido por la ecuación:

$\sum_{n=0}^{\infty} D_n.B^n = 5_0 \text{ x}10^0 + 2_1 \text{ x}10^1 + 1_2 \text{ x}10^2 = 5\text{x}1 + 2\text{x}10 + 1\text{x}100 = 5+20+100=125$

Convertir un binario a decimal

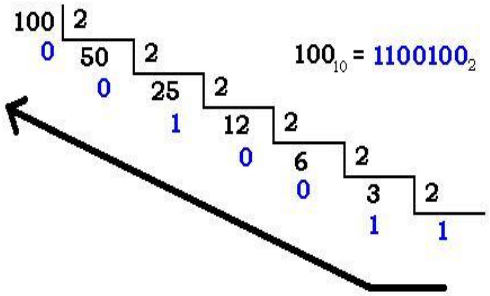
El sistema binario se basa únicamente en dos condiciones: encendido (1) o apagado (0), por lo tanto su base es 2, la anterior formula es aplicable a la conversión de cualquier base a la base decimal cada posición de los dígitos binarios representa un valor particular de “2” elevado a la potencia “n”. Es

importante tener claro que el digito ubicado mas a la derecha es llamado digito menos significativo (LSB ó de menos valor posicional numérico) y el digito ubicado mas a la derecha es el más significativo (MSB ó de más valor posicional numérico).

Convertir un número decimal a binario

Existen dos maneras de convertir un número de decimal a binario, la primera es utilizar una tabla de equivalencias de binario a decimal siguiendo las potencias de 2 (dos), la segunda es válida para convertir de decimal a cualquier base numérica y consiste en dividir sucesivamente la parte entera entre dos tomando sus módulos (residuos) y el último resultado, la parte decimal se multiplica por la base y se toma el numero entero.

Figura 13. ***Conversión decimal-binario***[14]




```
0,3125 (decimal)   => 0,0101 (binario).
Proceso:
0,3125 x 2 = 0,625 => 0
0,625  x 2 = 1,25  => 1
0,25   x 2 = 0,5   => 0
0,5    x 2 = 1     => 1
En orden: 0101     -> 0,0101 (binario)
```

Los dígitos son cada una de las cifras que componen un número en cualquier sistemas numérico, para el sistema binario los dígitos binarios hacen referencia a las cifras individuales representadas por un “uno” o un “cero” que en conjunto forman un número binario, en el ejemplo anterior cada residuo y último resultado es una cifra o digito binario, en conjunto forman el numero binario “1100100” que equivale al numero decimal “100”.

[14] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Sistema_binario

El sistema Hexadecimal

La base decimal tiene 10 dígitos representativos que van del 0 al 9 con lo que puede representarse cualquier valor, en la base hexadecimal se tiene 16 dígitos que van del 0 al 9 y de la letra A hasta la F las que representan los números del 10 al 15. La conversión entre números binarios y hexadecimales, es sencilla notando que cada cuatro binarios equivalen a un hexadecimal y cada hexadecimal equivale a cuatro (4) binarios, esta agrupación de cuatro dígitos binarios se debe comenzar desde la posición menos significativa desde el punto separador de enteros y fracciones a la izquierda o a la derecha.

Figura 14. ***Tabla de Conversión***[15]

$0_{hex} = 0_{dec} = 0_{oct}$	0	0	0	0
$1_{hex} = 1_{dec} = 1_{oct}$	0	0	0	1
$2_{hex} = 2_{dec} = 2_{oct}$	0	0	1	0
$3_{hex} = 3_{dec} = 3_{oct}$	0	0	1	1
$4_{hex} = 4_{dec} = 4_{oct}$	0	1	0	0
$5_{hex} = 5_{dec} = 5_{oct}$	0	1	0	1
$6_{hex} = 6_{dec} = 6_{oct}$	0	1	1	0
$7_{hex} = 7_{dec} = 7_{oct}$	0	1	1	1
$8_{hex} = 8_{dec} = 10_{oct}$	1	0	0	0
$9_{hex} = 9_{dec} = 11_{oct}$	1	0	0	1
$A_{hex} = 10_{dec} = 12_{oct}$	1	0	1	0
$B_{hex} = 11_{dec} = 13_{oct}$	1	0	1	1
$C_{hex} = 12_{dec} = 14_{oct}$	1	1	0	0
$D_{hex} = 13_{dec} = 15_{oct}$	1	1	0	1
$E_{hex} = 14_{dec} = 16_{oct}$	1	1	1	0
$F_{hex} = 15_{dec} = 17_{oct}$	1	1	1	1

El sistema Octal

El sistema octal también es utilizado en la representación de información y datos en el microprocesador, se compone de ocho dígitos (0,1,2,3,4,5,6,7) los cuales siguen una regla similar a la de los hexadecimales la cual establece que tres dígitos binarios equivalen a un dígito octal y un octal a tres binarios.

[15] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Sistema_hexadecimal

Tabla de conversión entre sistemas y codificaciones utilizadas en los sistemas basados en microprocesadores y microcontroladores

Tabla 2. ***Conversión y códigos*** [16]

Decimal	Binario	Hexadecimal	Octal	BCD	Exceso 3	Gray o Reflejado
0	0	0	0	0	11	0
1	1	1	1	1	100	1
2	10	2	2	10	101	11
3	11	3	3	11	110	10
4	100	4	4	100	111	110
5	101	5	5	101	1000	111
6	110	6	6	110	1001	101
7	111	7	7	111	1010	100
8	1000	8	10	1000	1011	1100
9	1001	9	11	1001	1100	1101
10	1010	A	12	0001 0000		1111
11	1011	B	13	0001 0001		1110
12	1100	C	14	0001 0010		1010
13	1101	D	15	0001 0011		1011
14	1110	E	16	0001 0100		1001
15	1111	F	17	0001 0101		1000

Representación de números negativos

Para representar números negativos en el sistema numérico binario, se realiza mediante la técnica denominada complemento a dos, esta técnica se define como

[16] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Sistema_binario

el numero obtenido después de convertir todos los ceros a unos y todos los unos a ceros, al resultado se le suma uno.

Decimal 125

Equivale a 1111101 → complemento a dos → 0000010 + 0000001 = 0000011

Bits y Bytes

Un bit es la unidad mas pequeña de información, pero en su forma más simple un bit representa un digito binario, es decir, un “uno” o un “cero”, físicamente los bits requieren de un espacio o celda donde almacenar una cantidad de energía definida en dos estados estables que representen los dígitos binarios, esta energía es llamada voltaje, que es la energía necesaria para mover una carga eléctrica (electrón) de un punto a otro.

La Unidad Central de Proceso o CPU, se encarga de procesar la información dentro de celdas denominadas registros, cada registro se compone de una o varias celdas la cuales almacenan una entidad llamada “bit”, en un comienzo las celdas de almacenamiento se fabricaban a partir de flip-flops o también llamados basculadores o biestables, generalmente los registros se disponen en un conjunto de 8 o 16 bits. Los flip-flops tienen la capacidad de almacenar dos niveles de voltaje, uno bajo típicamente 0,5 Volts que es interpretada como cero “0” o apagado y un nivel alto típicamente 5 Volts interpretado como uno “1” o encendido, estos estados son los conocidos como “bit” (BInary digiT o digito binario).

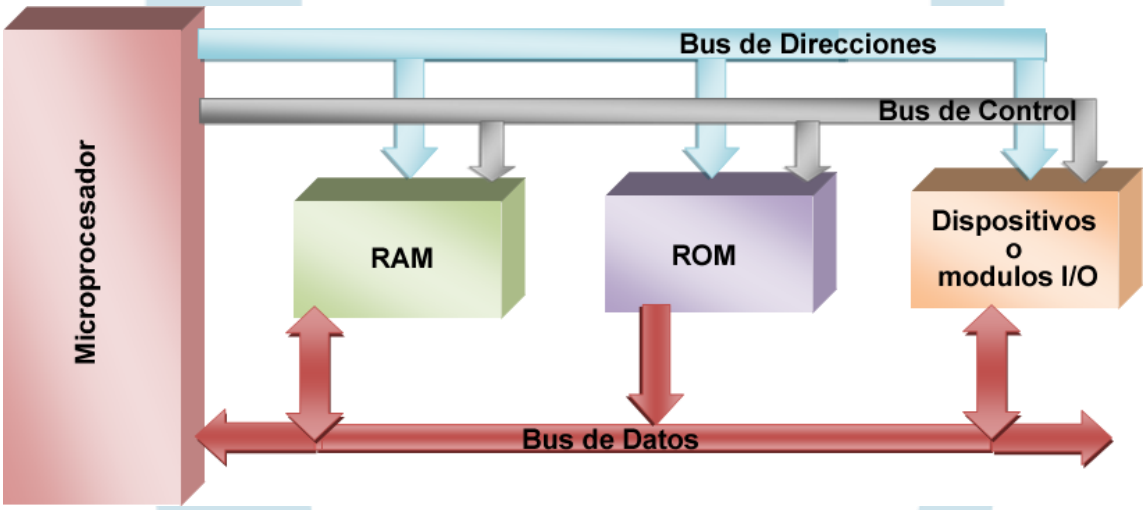
A un grupo de 4 bits se le conoce como Nibble, dos Nibbles u 8 bits se conocen como Byte y dos Bytes o 16 bits se denominan como palabra, existen también, la doble palabra y la cuádruple palabra. Como ejemplo de estos registros en los microprocesadores esta un registro denominado AX el cual es de 16 bits pero puede ser utilizado como de 8 bits, este registro AX puede representar 65536 números binarios entre 0000000000000000 (2^0) y 1111111111111111(2^{15}).

LECCIÓN 3: ESTRUCTURA INTERNA Y FUNCIONAMIENTO.

Los sistemas de computo basados en microprocesadores se componen de tres bloques fundamentales, la Unidad Central de Procesamiento – CPU, Dispositivos de memoria y Puertos de Entrada / Salida, de igual forma el microprocesador esta conformado por varios bloques, entre ellos están la Unidad Aritmético – lógica (ALU), una Unidad de Control (UC) y un bloque o matriz de registros.

La unidad de entrada conformada dispositivos encargados de recibir la información del mundo exterior, estos dispositivos llamado de entrada / salido o I/O (*Input/Output*), la información es llevada hacia la unidad de memoria para ser procesada posteriormente. La información luego es llevada desde la unidad central de proceso o CPU, hacia circuitos o periféricos utilizando el bus de datos. La unidad de memoria se encarga de almacenar los datos y los programas que operan sobre esos datos. La CPU obtiene las instrucciones y los datos colocando una dirección en el bus de direcciones para posteriormente ser transferidos a través del bus de datos cuando la CPU lo solicite.

Figura 15. *Sistema de cómputo*[17]



Existen dos sistemas diferentes de memoria, la de almacenamiento primario y la del almacenamiento secundario. La memoria de almacenamiento primario se refiere a la memoria que almacenan los programas que se van a ejecutar y los datos utilizados durante la ejecución del programa. La memoria secundaria es la

[17] Extraído el 10 de Julio de 2009 desde Fuente (CEKIT, 2002)

encargada de almacenar grandes cantidades de datos que no se requieren con frecuencia, este tipo de dispositivo son los discos duros y discos 3,5". También se distinguen tres categorías, la ROM (Red Only Memory) o memoria de solo lectura donde se almacenan cierto tipo de programas como la BIOS, la memoria RAM (Random Access Memory) o memoria de lectura y escritura, donde se almacenan datos que los programas en ejecución van generando, como tercera categoría se encuentra la "cache" con tiempo de acceso muy rápido muy cercana al núcleo del procesador.

La ejecución de un programa se realiza de forma secuencial, las instrucciones almacenadas en la memoria de programa modifican los datos almacenados en la memoria u obtenidos a través de un dispositivo de entrada, los datos que se obtienen después del procesamiento pueden ser almacenados en la memoria o ser enviados a los periféricos de salida utilizando el bus de datos, la CPU utiliza el bus de control para establecer la lectura y/o escritura de los datos y con el bus de direcciones determina el destino de los datos.

La Unidad Central de Proceso - CPU

En la práctica la Unidad Central de Proceso o CPU se encuentra en forma de un circuito integrado llamado microprocesador. El microprocesador es un circuito digital que acepta o lee datos aplicados a un cierto numero de líneas de entrada, los procesa de acuerdo a las instrucciones secuenciales de un programa almacenado en su memoria y suministra o escribe los resultados del proceso con cierto numero de salida. La evolución de los microprocesadores ha hecho que evolucionen los computadores ampliando sus prestaciones, servicios, capacidad y facilidad de interacción con el medio. Los microprocesadores se aplican a otras situaciones de control y supervisión en el control industrial, maquinaria, motores, telemetría, robótica, automatización entre otros.

Los datos de entrada pueden provenir de interruptores, sensores, convertidores A/D, teclados, etc; los datos de salida son dirigidos a display, LCD, CTR, convertidores D/A, impresoras, etc. El soporte físico de las instrucciones del programa es la memoria, la cual almacena los datos para que sean procesados, los niveles lógicos (unos y ceros) en las líneas de salida de un microprocesador no solo dependen del programa, también dependen de la historia o estado anterior de las señales de entrada. Como se observa es clave la cantidad de pines o conexiones de entra/salida que tenga el microprocesador para interactuar con los periféricos y la memoria por eso la mayoría de microprocesadores utilizan las mismas líneas para entrada y salida. Las líneas de control sincronizan las

operaciones con los componentes externos conectados y ejercen un control global de los buses de datos y direcciones.

Los programas realizan funciones o aplicaciones limitadas por la imaginación del programador y la capacidad de los dispositivos, el programa que se ejecuta debe alojarse en determinadas posiciones de memoria y ser escrito en un lenguaje que la CPU entienda, es decir, lenguaje binario. La CPU entonces lee de forma secuencial la lista de instrucciones, las interpreta y controla la ejecución de cada una de ellas, para ejecutar cada instrucción la CPU realiza los siguientes pasos:

1. Leer de la memoria la instrucción a ejecutar y guardarla en un registro interno de la CPU.

2. Identificar la instrucción que acaba de leer.

3. Comprobar si la instrucción necesita datos que se encuentran en memoria, si es el caso, determina la localización de los mismos.

4. Buscar los datos en memoria y guardarlos en registro dentro de la CPU.

5. Ejecutar la instrucción.

6. El resultado puede ser almacenado en memoria o retenido esperando la siguiente instrucción o incluso generar comunicación con otros elementos externos a la CPU.

7. Comienza un nuevo ciclo empezando con el primer paso.

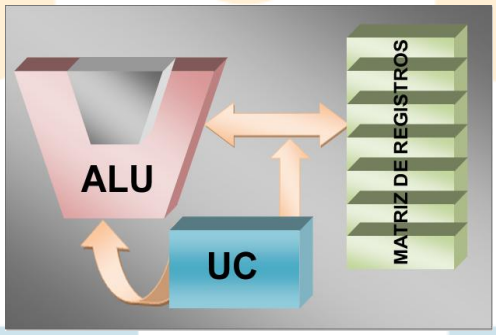
La ejecución de cada instrucción implica un movimiento de datos a partir de pulsos electrónicos que siguen una secuencia y orden establecido por una señal generada en un circuito que genera pulsos periódicos, este circuito denominado “reloj”, es simple en su construcción pero vital para el funcionamiento del microprocesador.

Arquitectura del microprocesador

La arquitectura en un microprocesador se refiere a las unidades estructurales que lo conforman, su diseño y organización, es difícil definir un modelo de microprocesador pues cada uno tiene una lógica de funcionamiento propia. Las principales unidades básicas de un microprocesador son la Unidad Aritmética y Lógico – ALU, la matriz de registros y la Unidad de Control. Aunque existen otras

unidades auxiliares y secciones que trabajan en conjunto con las anteriores para formar un microprocesador especifico.

Figura 16. ***Unidades Básicas de un MP***[18]



Sin importar la forma física que tome la CPU, su operación fundamental es ejecutar una secuencia de instrucciones denominadas “programa”, utilizando cuatro pasos básicos leer, decodificar, ejecutar, y escribir.

Figura 17. ***Pasos en la ejecución de una instrucción***[19]

El microprocesador junto con sus unidades básicas realiza las principales tareas en un sistema de cómputo:

1. Transferencia de datos desde y hacia: la memoria, periféricos y/o dispositivos.
2. Operaciones aritméticas y lógicas para ello el microprocesador cuenta con una unidad interna aritmético – lógica (hace sumas y comparaciones y con base a estas realiza las demás operaciones, complementos, restas, etc).
3. Un control sobre el flujo de un programa, este programa se está ejecutando y el microprocesador esta accionando sobre el programa comunicación con la memoria.

El poder del microprocesador radica en:

[18] Extraído el 10 de Julio de 2009 desde Fuente (CEKIT, 2002)
[19] Extraído el 10 de Julio de 2009 desde http://organizacioncomputadoras.blogspot.com/2009/05/31-funciones-de-la-unidad-de-control.html

1. La capacidad de ejecutar millones de instrucciones por segundo (MIPs), que provienen de un programa o software almacenado en la memoria de programa.
2. La capacidad para tomar decisiones simples basadas en hechos numéricos, como signos, comparaciones o resultados lo que permite modificar el flujo del programa.

Buses en los microprocesadores

A partir del desarrollo de la arquitectura Von Newman, se introdujo en los microprocesadores el concepto de programa almacenado y la ejecución secuencial del programa almacenado, para que el microprocesador pueda comunicarse con los diversos periféricos que tiene existen unos medios llamados buses.

Los buses permiten la conexión con los periféricos incluida la memoria y permiten la transmisión de la información, físicamente los buses consisten en líneas conductoras que permiten la circulación de los pulsos eléctricos. Generalmente el número de líneas de entrada es igual al número de líneas de salida, este numero define la longitud de la palabra de datos del microprocesador que comúnmente es 4, 8, 16, 32 y 64 bits aunque la unidad básica de almacenamiento y procesamiento es de un Byte.

Bus de datos, es el encargado de llevar datos e instrucciones hacia y desde el microprocesador, es un bus bidireccional en donde los datos varían en tamaño desde 8 a 64 bits, por el bus de datos las instrucciones y los datos son transferidos al microprocesador y los resultados de una operación son enviados desde el microprocesador.

Bus de direcciones, contiene la información digital que envía el microprocesador a la memoria y demás dispositivos direccionales del sistema para seleccionar una posición de memoria, una unidad de entrada/salida o un registro particular de la misma, la cantidad de líneas del bus de direcciones determina la capacidad máxima de la memoria que puede direccionar el sistema basado en microprocesador. Los primeros microprocesadores utilizados en sistemas de computo tenían 16 líneas de direcciones con lo que podían direccionar hasta $2^{16}=2^{10}*2^{6}=1024 * 64 = 65536 = 64KB$.

Cuantos más bits tenga el bus de direcciones mas capacidad de direccionamiento de memoria tendrá el sistema, el numero de líneas de dirección se ha incrementado a 20, 24, 32 y 36 bits este ultimo permite direccionar:

$2^{36}= 2^{10}*2^{10}*2^{10}*2^{6}=1024 * 1024 * 1024 * 1024 * 64 = 1K*1K*1K*1K*64 = 1M * 1M * 64$

$2^{36}= 1G * 64 = 64$ GB.

Bus de control, se utiliza para coordinar, sincronizar y comunicarse con los dispositivos externos, también se utiliza para insertar estados de espera para los dispositivos más lentos y evitar congestión en el bus, contiene líneas que seleccionan la memoria o dispositivo E/S y los habilita para que haga operaciones de lectura o escritura, particularmente en los microprocesadores 80x86 existen cuatro líneas que son:

- $\overline{MRDC}$ control de lectura de memoria.
- $\overline{MWTC}$ control de escritura de memoria.
- $\overline{IORC}$ control de lectura en un puerto.
- $\overline{IOWC}$ control de escritura en un puerto.

Unidad Aritmético lógica – ALU

Los datos en el ALU se presentan en registros, posteriormente en ellos se almacenan los resultados de las operaciones, estos registros son posiciones de memoria temporales, internas en el microprocesador y conectados a la ALU, los procesos aritméticos y lógicos pueden activar indicadores o también llamadas banderas (flags), que son indicadores del estado del microprocesador, normalmente estos indicadores son almacenados en un registro especial dentro del microprocesador (registro de estado), por ejemplo cuando el resultado de una operación es “cero” su indicador o bandera (*flag*) de cero cambian su estado.

Figura 18. ***Interconexión de la ALU***[20]

[20] Extraído el 10 de Julio de 2009 desde http://www.computacion.geozona.net/cpu.html

La unidad de control proporciona las señales que gobiernan el funcionamiento de la ALU y la transferencia de datos dentro y fuera de ella.

Esta unidad es la encargada de realizar las operaciones elementales de tipo aritmético (generalmente sumas o restas) y de tipo lógico (generalmente comparaciones).

Para realizar su función, consta de un banco de registros (BR), este banco está constituido por 8, 16 ó 32 registros de tipo general que sirven para situar datos antes de cada operación, para almacenar datos intermedios en las operaciones y para operaciones internas del procesador.

Como se había establecido antes los datos que maneja la ALU tienen formato de complemento a dos, haciendo más sencillo el trabajo y las operaciones con números negativos.

En general los bits restantes en una operación que involucre números con signo siguen lo establecido en la tabla 3.

Tabla 3. ***Representación con signo en la ALU***[21]

RANGO DE VALORES	-2^{n-1} a $2^{n-1}-1$
Nº de representaciones del cero	Una
Negación	Ejecutar el complemento booleano de cada bit y sumar 1 al patrón de bits resultante.
Ampliación de la longitud de bits	Las posiciones de bit extra se añaden a la izquierda, rellenándolas con el valor del bit de signo original
Regla para el desbordamiento	Si se suman dos números de igual signo (ambos positivos o ambos negativos), hay desbordamiento si, y sólo si, el resultado tiene signo opuesto.
Regla para la resta	Para restar B de A, se toma el complemento a dos de B y suma a A.

La representación del número cero tienen un bit de signo cero (0), una magnitud cero. El costo de manipular números negativos es la restricción de valores representables, por ejemplo para registros de 8 bits, el valor máximo representable sin signo es de 00000000 a 11111111, o equivalente a 255 en decimal, el numero con signo, en formato positivo no puede ser mayor a 01111111 binario o 127 decimal, por cuanto el siguiente numero binario valido seria 10000000 y al tener encendido el bit mas significativo es evidente que representa un numero negativo

[21] Stallings, 2000

que correspondería al -128. Entonces para representar con “n” bits el rango de números positivos y negativos se establece como números positivos desde cero hasta 2^{n-1}-1, para los negativos se representan desde -1 hasta -2^{n-1}.

Matriz de Registros

Un sistema de cómputo exige de la CPU la realización de varias funciones, captar instrucciones, interpretar instrucciones, captar datos, procesar datos y escribir datos; para que un microprocesador pueda realizar las tareas en un sistema de cómputo, es necesario que la CPU almacene alguna información de forma temporal. Esta información debe contener en un almacenamiento temporal la instrucción a ejecutar, los datos sobre los que debe operar, la dirección de la instrucción o dato a cargar en la ALU, las direcciones de salto o llamado, de forma que pueda saber dónde ir a buscar la siguiente instrucción. Lo anterior hace necesario que la CPU requiera de una memoria interna, es así como en la CPU, existe un conjunto de registros funcionando como un nivel de memoria que esta por encima de la memoria principal y la cache, estos registros se conocen como la matriz de registros y pueden ser de dos tipos:

1. **Registros visibles para el usuario:** Permiten al programador de lenguaje máquina o ensamblador, minimizar las referencias a memoria principal optimizando el uso de estos registros, estos registro se pueden referenciar mediante lenguaje maquina, están disponibles para todos los programas (de sistemas y aplicación).

2. **Registros de control y de estado:** Utilizados por la Unidad de Control para controlar las operaciones del procesador, y por los programas o rutinas con privilegios del Sistema Operativo para controlar la ejecución del programa.

1. Registros visibles para el usuario

Se pueden clasificar en las siguientes categorías:

- **Registros de Propósito general**: El programador puede asignarlos a varias funciones, como almacenamiento de datos o direccionamiento. Lo que nos permite subdividirlos en:
    - Registros de datos: que solo contienen datos y no se emplean en cálculos de direcciones de operando, estos registros deben poder almacenar la mayoría de tipos de datos. Ocasionalmente algunas maquinas permiten que registros contiguos puedan ser usados como uno solo para contener valores de longitud doble.
    - Registro de dirección: pueden usarse para almacenar datos aunque su función principal esta dedicada a atender un modo de

direccionamiento, lo que implica tener la capacidad de almacenar la dirección mas grande.

- **Indicadores de Condición**: son bits ajustados por el hardware de la CPU resultado de alguna operación, tal es el caso de desbordamientos, resultado nulo, positivo o negativo, esto implica que la ejecución de una instrucción no solo puede dar por resultado la modificación o almacenamiento en un registro, también se obtiene un código de condición. Estos códigos de condición se reúnen por lo general en uno o mas registros, normalmente en un registro de control, estos bits son leídos por las instrucciones maquina por referencia implícita aunque no pueden ser alterado por el programador.

- **Acumulador**: es un registro (o registros) donde se almacena temporalmente los resultados aritméticos y lógicos intermedios que serán tratados por la ALU. Su longitud puede ser de 8, 16 o 32 bits. Es un registro indispensable para toda operación de I/O (Entrada / Salida ó *Input / Output*)y para operaciones de cadenas

- **Punteros de segmento**: cuando la memoria se divide en segmentos, una referencia a la memoria consta de una referencia a un segmento y un desplazamiento de segmento, por ejemplo la combinación CS:IP (Segmento de código : Apuntador de instrucción) da como resultado la siguiente instrucción a ejecutar.

- **Registro índice**: Estos registros se utilizan generalmente en operaciones con cadenas, también en operaciones como el cálculo de direcciones sumando un índice a un valor base para obtener la dirección efectiva en direccionamiento indirecto, también es utilizado en el pase de parámetros a otras rutinas

- **Puntero de pila (SP)**: son registros que apuntan a alguna localidad de memoria, en particular este registro indica la siguiente posición de memoria disponible en la pila. La pila o Stack es un área reservada utilizada principalmente para el almacenamiento de direcciones de vuelta y contenido de registros. La pila es de tipo LIFO – ultimo en entrar primero en salir, se utiliza durante las llamadas a subrutina y durante las interrupciones.

2. Registros de control y de estado

Estos registros sirven para controlar el funcionamiento de la CPU o para evidenciar el estado particular de la misma respecto a una instrucción, la mayoría en gran parte de dispositivos no son visibles al usuario, algunos pueden ser

visibles mediante instrucciones maquina ejecutadas en modo de control o de sistema operativo.

- **Contador de programa (PC)**: contiene la dirección de la instrucción a ejecutar, la CPU actualiza el contador de programa después de captar cada instrucción, apuntando a la siguiente instrucción. Las instrucciones de salto o bifurcación modifican el contenido del contador de programa, la instrucción captada se carga en el registro IR.

- **Registro de instrucción (IR)**: contiene la instrucción captada mas reciente, en este registro se analiza el código de operación y los campos del operando.

- **Registro de dirección de memoria (MAR)**: contiene la dirección de una posición de memoria.

- **Registro intermedio de memoria (MBR)**: contiene el dato a escribir en la memoria o la palabra leída mas reciente. En un sistema con organización de bus, MAR se conecta directamente al bus de direcciones y MBR directamente al bus de datos. Los registros visibles por el usuario intercambian repetidamente datos con MBR.

- **Registro de estado**: Este registro reporta el estado de alguna operación aritmética o lógica, se le conoce como registro de estado (status register) o PSW (Program Status Word), los bits que lo conforman tienen dos estados posibles que tienen una significación especial acorde al procesador particular. Los indicadores de condición que lo conforman son conocidos como “flags” o “banderas” que son utilizados por instrucciones de bifurcación para la toma de decisiones, los campos mas comunes son:

    - **Signo:** Contiene el bit de signo del resultado de la última operación aritmética.

    - **Cero:** Puesto a uno cuando el resultado es 0.

    - **Acarreo:** Puesto a uno si una operación da lugar a un acarreo (suma) o préstamo (resta) del bit más significativo.

    - **Igual:** puesto a uno si el resultado de una comparación lógica es la igualdad.

    - **Desbordamiento:** usado para indicar un desbordamiento aritmético.

    - **Interrupciones habilitadas/deshabilitadas:** usado para permitir o inhabilitar interrupciones.

- **Supervisor:** indica si la CPU funciona en modo de supervisor o usuario. Únicamente en modo supervisor se pueden ejecutar ciertas instrucciones privilegiadas y se puede acceder a ciertas áreas de memoria.

Unidad de control

La unidad de control (UC) es el centro nervioso de la CPU, desde ella se controla y gobiernan todas las operaciones (búsqueda, decodificación, y ejecución de la instrucción), proporcionando las señales de control para la entrada o salida de datos en el microprocesador y la sincronización de la ejecución de las instrucciones, estas tareas básicas se pueden resumir en:

- **Secuenciamiento:** hace que el procesador avance a través de una serie de microoperaciones, basadas en el programa que esté ejecutando.
- **Ejecución:** la unidad de control hace que se ejecute cada microoperación.

El trabajo que realiza la Unidad de control implica:

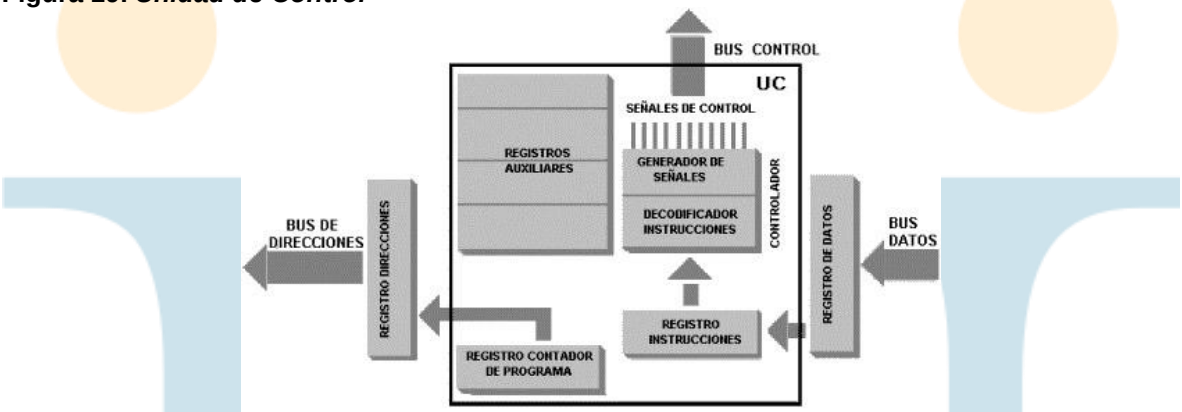
1. **Microoperaciones:** para entender el concepto de microoperación hay que tomar los siguientes conceptos.
   - **Ciclo de reloj**, consiste en una serie de pulsos periódicos generados por reloj local interno o externo, este ciclo permite organizar la secuencia interna de eventos en la CPU.
   - **Ciclo maquina**, consiste en uno o varios ciclos de reloj necesarios para ejecutar una instrucción, algunas instrucciones utilizan mas de un ciclo maquina para completar sus actividades, tal es el caso de instrucciones de decisión-salto.
   - **Ciclo de instrucción**, compuesto en forma general por captación, ciclo indirecto, ejecución e interrupción, cada uno de estos contienen pasos que involucran a registros del microprocesador, estos pasos son las **microoperaciones**.

2. **Señales de control**
   Las señales de control se componen de señales de entrada que permiten determinar el estado del sistema y salidas que permiten controlar el comportamiento del mismo.

- **Señales de entrada.**
  - **Reloj**: Es el encargado de generar pulsos periódicos utilizados por la Unidad de Control para ejecutar una o varias microoperaciones en cada pulso de reloj, este pulso se denomina “período de reloj” o “tiempo de ciclo del procesador”.

- **Registro de instrucción**: contiene el código de la instrucción con la que la Unidad de control determina el conjunto de microoperaciones a realizar durante el ciclo de ejecución.
- **Indicadores**: son bits que utiliza la Unidad de Control para determinar el estado del procesador y el resultado de la operación anterior en el ALU
- **Señales de control del bus de control**: señales de control desde el bus del sistema (señales de interrupción y de reconocimiento).

Figura 19. ***Unidad de Control***[22]



- **Señales de salida.**
  - **Señales de control internas al procesador**:
    - Utilizadas para transferir datos de un registro a otro.
    - Activar funciones específicas de la ALU.
  - **Señales de control hacia el bus de control**:
    - Señales de control de la memoria.
    - Señales de control de las E/S.

Figura 20. ***I/O en la Unidad de Control***[23]

[22] Extraído el 10 de Julio de 2009 desde http://www.computacion.geozona.net/cpu.html
[23] Extraído el 10 de Julio de 2009 desde http://organizacioncomputadoras.blogspot.com/2009/05/31-funciones-de-la-unidad-de-control.html

LECCIÓN 4: REGISTROS Y SEGMENTOS.

Registros

- **Registros de uso general**: En los microprocesadores x86 s tienen cuatro registros duales, lo que significa que pueden ser manipulados como 8 y 16 bits, estos registros en modo de 16 bits se conocen como AX, BX, CX y DX. En la modalidad de 8 bits responden a una letra distintiva que los divide en parte baja (L) desde el bit 0 al bit 7 y la parte alta (H) desde el bit 8 hasta el bit 15, es decir, AH+AL=AX, BH+BL=BX, CH+CL=CX y DH+DL=DX.
- **Registros apuntadores**: son registros que apuntan a una posición de memoria, entre ellos están:
    - **BP**, Base Pointer – apuntador base utilizado para manipular la información de la pila o Stack.
    - **SP**, Stack Pointer – apuntador de pila, que en conjunto con el registro de segmento de memoria se utiliza para crear una estructura de memoria llamada pila o Stack.
- **Registros índice**: son muy utilizados en operaciones con cadenas, estos registros se identifican como SI (Source Index – Índice fuente) y DI (Destination Index – Índice destino).

Segmentos

Cualquier operación que accede a la memoria se hace mediante registros que seleccionan un área contigua de memoria que puede ser de 64 KB, conocida como segmento, estos segmentos en su interior tienen desplazamientos internos para acceder la información. El procesador generalmente tiene cuatro registros llamados registros de segmentos:

- **CS-Code segment o registro de segmento de código**: se encuentra asociado al código del programa, es decir, controla el código de los programas y tiene como complemento el registro IP. La combinación CS:IP da como resultado la siguiente instrucción a ser ejecutada.
- **DS-Data Segment o registro del segmento de datos**: es donde se guardan los datos del programa.
- **ES-Extra Segment o registro de segmento extra**: sirve como apoyo al segmento de datos ofreciendo una ampliación del mismo.
- **SS-Stack Segment o registro del segmento de pila**: tiene la función de controlar el área donde se creará la pila.

La pila

Es un bloque de memoria utilizado para almacenar información bajo el esquema de ultimo en entrar primero en salir (*LIFO* - Last Input First Out), este bloque de memoria es conocido como pila ó "*stack*" y es necesario que exista, pues la CPU lo requiere, en la pila los datos son empujados para posteriormente ser extraídos comenzando con el ultimo que fué introducido, la pila proporciona a la CPU un mecanismo para que pueda almacenar el contexto de un programa, es decir las direcciones de retorno después de invocar alguna rutina o llamado y el manejo de alguna interrupción, también sirve como área temporal de almacenamiento.

La arquitectura x86 obliga a usar en combinación un registro de segmentos (CS, DS, ES, SS) y un desplazamiento para obtener la localidad absoluta de un byte de datos o instrucción localizado en la memoria (RAM o ROM), en el caso del 8086 el limite para cada segmento es de 64KB.

LECCIÓN 5: MODOS DE DIRECCIONAMIENTO.

Modos de direccionamiento

Antes de establecer los modos de direccionamiento se considera definir qué es una dirección de memoria y como se calcula.

- **Direcciones de memoria**: Las direcciones de memoria son localidades o ubicaciones del registro que almacena cierta información (datos o instrucciones). El esquema de direccionamiento que impone la CPU obliga a usar registros de segmentos en cada operación, por ejemplo el 8086 tiene un bus de 20 bits y registros de 16 bits como se observa el registro no tiene la capacidad de representar 20 bits, lo que hace necesario dividirla en dos partes, el segmento y el desplazamiento dentro del segmento, cada uno de 16 bits esto se representa en Hexadecimal como SSSS:DDDD, donde SSSS representa el registro de segmento, dos puntos, DDDD que representa el desplazamiento. Por ejemplo 0050:0002 es la dirección del tercer byte dentro del segmento 0050H.

- **Calculo de direcciones de memoria**: La forma de calcular una dirección real de memoria es recorrer el registro de segmento 4 bits a la izquierda y

sumarle el desplazamiento. Por ejemplo 2B45:0032 si solo se suman las cantidades anteriores se obtendría 2B77H, que tiene 16 bits (recordar que un hexadecimal representa 4 bits) y no 20 como se necesita, si se corre el registro de segmento 4 bits a la izquierda se obtiene 2B450H que claramente es de 20 bits y se le suma el desplazamiento con lo que se obtiene 2B450H + 0032H = 2B482H, de esta forma se obtiene la dirección absoluta de memoria, es posible que una o mas combinaciones de segmentos y desplazamientos originen la misma dirección.

La CPU ofrece varios métodos para calcular direcciones de memoria, los accesos a memoria pueden categorizarse de dos modos:

- Accesos para obtener la siguiente instrucción a ejecutarse utilizando la combinación CS:IP.
- Para obtener algún dato. Se utilizan varias modalidades de direccionamiento:
  - **Transferencia de registro a registro**: es una de las más sencillas, en la cual se realiza una copia del registro fuente al registro destino dejando intacto el primero. Ejemplo *MOV AX, BX*. La instrucción *MOV* hace referencia a mover datos, en este caso desde un registro *BX* que es la fuente desde donde se copia el dato al registro de destino AX. No hay acceso a memoria.
  - **Inmediata**: el operando se incluye como parte de la instrucción. Por ejemplo MOV AX, 5, el número 5 forma parte de la instrucción en su totalidad, el número se especifica como una constante numérica en la instrucción y no hay necesidad de acceder a memoria, moviendo 5 al registro AX.
  - **Directa**: Un valor de 16 bits forma parte de la instrucción y es interpretado como un acceso a la memoria. Por ejemplo MOV AX, VALOR; donde VALOR es una localidad de memoria donde se encuentra la información que se moverá al registro AX, mas específicamente con respecto a la instrucción anterior tendríamos MOV AX, [120H], que mueve el contenido almacenado en la localidad de memoria [120H] al registro AX, *los corchetes indican localidad de memoria*.
  - **Indirecta**: Esta modalidad emplea el contenido de los registros índice SI (Source Index – índice fuente) y DI (Destination Index – índice destino), para calcular la dirección de memoria. Se puede incluir un operando

inmediato en la misma instrucción para realizar el calculo. Por ejemplo MOV AX, [SI], donde si se considera que SI tiene el valor 0270H, la instrucción toma el contenido del registro SI (0270H) como una localidad de memoria, accediendo a dicha localidad, obteniendo el valor y almacenándolo en el registro AX. Es posible utilizar el operando inverso, por ejemplo MOV [SI-0100], BX. En esta modalidad todos los desplazamientos se hacen en función del registro de segmentos DS.

- **Base relativa**: Esta modalidad utiliza dos registros, el BX y el BP, para calcular una dirección de memoria, el desplazamiento indicado por el registro BX se toma en función al registro de segmento DS y el desplazamiento indicado por el registro BP se toma en función al registro de segmento SS, su funcionalidad se relaciona con la anterior modalidad a excepción del registro BP. Al usarse BP debe incluirse un segmento operando, ya sea un índice o un operando inmediato, por ejemplo MOV AX, [BP], será ensamblada como MOV AX, [BP+0].

CAPÍTULO 2: FAMILIAS DE MICROPROCESADORES

LECCIÓN 1: PRINCIPALES FAMILIAS DE MICROPROCESADORES.

Familia de Microprocesadores Zilog

Zilog, es un fabricante de microprocesadores de 8 bits, siendo su producto más reconocido el Zilog Z80. Fue fundado en California en 1974 por Federico Faggin, quien trabajó perfeccionando el primer microprocesador de Intel, el Intel 4004.

En 1976 la compañía crea el Zilog Z80 que era un microprocesador basado en el Intel 8080 con algunas mejoras. Con este nuevo producto Faggin realizó una gira por el mundo buscando potenciales clientes. Un año después sale al mercado el primer computador que hace uso del Z80, el TRS-80 Model 1 con un Z80 a 1,77 MHz y 4 KB de RAM. En 1989 sale la videoconsola *Game Boy* con un Z80 a 1,05 MHz y posteriormente la *Sega Game Gear* y la *Sega Genesis*, que también lo usan.

En el año 1995 crean el V-Chip y en el 2001 Zilog lanza el eZ80.

Familia Microprocesadores Motorola

Uno de sus principales exponentes es el Motorola 6800, el cual fue lanzado al mercado en 1975, poco después del Intel 8080. Su conjunto de instrucciones está formado por 78 instrucciones, posiblemente es el primer microprocesador que contó con un registro índice. El 6800 normalmente se fabricaba en un encapsulado DIP de 40 pines.

Varios de los primeros microordenadores de los años 1970, que usualmente eran vendidos por correo (en piezas sueltas o ensamblados), usaron el 6800 como procesador principal. Entre ellos se encuentran el SWTPC 6800 (el primero en usarlo) y el MITS Altair 680.

Figura 21. ***Motorola 6800***[24]

Partiendo del 6800 se crearon varios procesadores derivados, siendo uno de los más potentes el Motorola 6809, que fue usado en el sistema de videojuego Vectrex y en el ordenador Tandy TRS-80, entre otros.

También se han producido varios microcontroladores basados en el 6800, como el Motorola 6805, 6807, 6808, 68HC11 y el 68HC12.

Familia de Microprocesadores SPARC

SPARC (del inglés Scalable Processor ARChitecture) es una arquitectura RISC, es decir, una arquitectura con un conjunto reducido de instrucciones.

Fue originalmente diseñada por Sun Microsystems en 1985 y se basa en los diseños RISC I y II de la Universidad de California en Berkeley que fueron definidos entre los años 1980 y 1982.

La empresa Sun Microsystems diseñó esta arquitectura y la licenció a otros fabricantes como Texas Instruments, Cypress, Fujitsu y LSI Logic.

SPARC es la primera arquitectura RISC abierta y como tal, las especificaciones de diseño están publicadas, así otros fabricantes de microprocesadores pueden desarrollar su propio diseño.

La CPU SPARC está compuesta por una unidad entera (Integer Unit) que procesa la ejecución básica y una unidad de punto flotante (Floating Point Unit) que ejecuta las operaciones y cálculos de reales.

Aunque no es una parte formal de la arquitectura, las computadoras basadas en sistemas SPARC de Sun Microsystems tienen una unidad de manejo de memoria (MMU) y un gran caché de direcciones virtuales (para instrucciones y datos) que están dispuestos periféricamente sobre un bus de datos y direcciones de 32 bits.[25]

[24] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Motorola_MC6800L_SC7718I_top.jpg
[25] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/SPARC

Figura 22. *Sun Ultra EparcII*[26]

La siguiente tabla presenta algunas de las generaciones del SPARC.

Tabla 4. *Generaciones Spark*[27]

Generación	Familia	Especificación general
Primera generación (1987)	SPARC	**Frecuencia de reloj de 16 a 50MHz. Diseño escalar**
Segunda generación (1992)	SUPER SPARC	**Frecuencia de reloj de 33 a 50MHz. Diseño super escalar.**
Tercera generación (1996)	**ULTRA SPARC II**	**Arquitectura super escalar de 4 etapas y de 64 bits. Cinco unidades de punto flotante. Velocidades entre 250 y 300 Mhz**

Familia de Microprocesadores Power PC

PowerPC se presenta como una arquitectura de computadoras de tipo RISC (*Reduced Instruction Set Computer*) creada por la Alianza AIM, compuesto por las empresas Apple, IBM y Motorola, de cuyas primeras letras, surgió la sigla. Los procesadores de esta familia son producidos por IBM y Freescale Semiconductor que es la división de semiconductores y microprocesadores de Motorola, siendo utilizados principalmente en ordenadores o computadores Macintosh de Apple Computer.

Este microprocesador está diseñado con base a la arquitectura POWER de IBM con algunos componentes tomados del microprocesador Motorola 68000 para darle compatibilidad con arquitectura de los ordenadores de Apple.

[26] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Sun_UltraSPARCII.jpg
[27] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Sun_SPARC

Figura 23. ***Contrucción y Power PC 601***[28]

Familia de Microprocesadores AMD

AMD surgió por la división que tuvo Intel en el proceso de integración de sistemas. Anteriormente AMD estaba asociada con Intel y creaban microprocesadores. Los procesadores del tipo 8088, 8086 o 80286, por ejemplo, incluían a AMD con su logotipo y después a Intel como marca registrada. Con la separación de Intel, AMD empezó a incursionar directamente en el campo de los microprocesadores.

Actualmente AMD es una empresa que desarrolla procesadores compatibles Intel, permitiendo disponer de un sistema accesible con un alto rendimiento, imágenes y gráficos 3D reales, sonido y video de pantalla completa.

Tipos de procesadores AMD

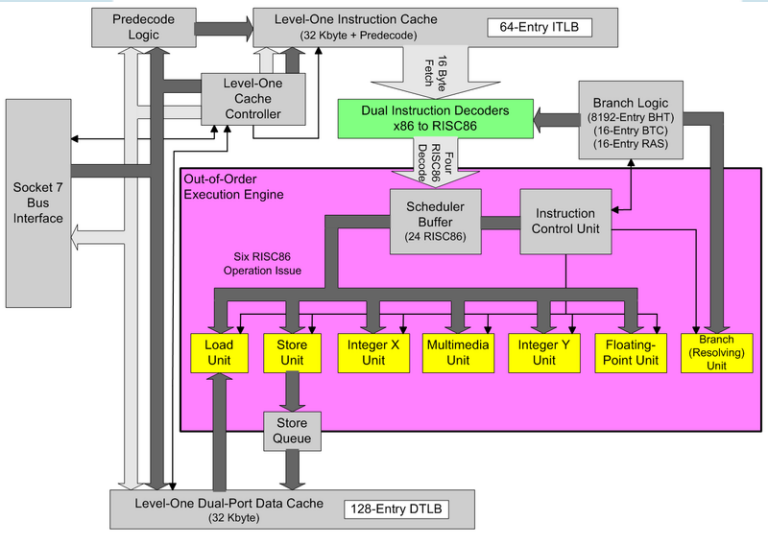
- **Am9080**: *Es una copia, originalmente sin licencia, del Intel 8080. Posteriormente se consiguió un acuerdo con Intel para fabricarlo y fue renombrado a Am8080. Las primeras versiones del Am9080 fueron puestas a la venta en abril de 1974, y funcionaban a una velocidad de reloj de 2 MHz*[29].
- **Am286**: *Es un procesador copia del Intel 80286, creado con permiso de Intel, debido a que IBM quería que Intel tuviese una segunda fuente para poder suplir la demanda en caso de problemas. Por lo tanto el Am286 es idéntico al Intel 80286. Posteriormente AMD lo vendió como procesador embebido*[30].
- **Am386**: Fue creado por AMD en 1991. Era un procesador con características semejantes al Intel 80386 y compatible 100% con este último, lo que le valió varios recursos legales de Intel por copiar su

[28] Extraído el 10 de Julio de 2009 desde http://appledifferent.com/wordpress/tag/power-pc/ y http://es.wikipedia.org/wiki/Archivo:IBM_PowerPC601_PPC601FD-080-2_top.jpg
[29] Extraído el 10 de Julio de 2009 desde http://www.stormloader.com/users/megaterran/9080.htm
[30] Extraído el 10 de Julio de 2009 desde http://www.diclib.com/cgi-bin/d1.cgi?l=es&base=es_wiki_10&page=showid&id=48428

tecnología. Tenía una velocidad de hasta 40 MHz lo que superaba a su competidor que sólo llegó a los 33 MHz.

- **Am486:** Fue presentado en 1993 por Advanced Micro Devices (AMD). Es un procesador compatible x86, comparable al Intel 80486.
- **AMD 5x86**: *Es un procesador compatible x86 presentado en 1995 por Advanced Micro Devices destinado a ser utilizado en ordenadores basados en un 486. Presentado en noviembre de 1995, el AMD 5x86 es un procesador 486 estándar con un multiplicador interno a 4x, permitiéndole funcionar a 133 MHz en sistemas para procesadores 486 DX2 o DX4 sin multiplicador. Tenía una memoria caché L1 de 16 Kbyte. Esta combinación permitió al 5x86 igualar e incluso sobrepasar ligeramente un procesador Pentium a 75 MHz. Además, como fue concebido en base a un 486, era compatible con sistemas más antiguos, lo que perjudicaba ligeramente a su rival más rápido, el Cyrix Cx5x86*[31].
- **AMD K6**: *fue lanzado en 1997. Este procesador estaba diseñado para funcionar en placas base Pentium. La principal ventaja del AMD con respecto al Pentium era su precio, bastante económico con las mismas prestaciones. El K6 tuvo una gran aceptación en el mercado presentándose como un rival fuerte para Intel. Su sucesor fue el microprocesador K6-2*[32]. El K6 cuenta con una gama que va desde los 166 hasta los 300 MHz y con el juego de instrucciones MMX, que ya se ha convertido en un estándar.

Figura 24. ***AMD K6***[33]



- **AMD Duron**: Es una gama de microprocesadores de bajo costo compatibles con los Athlon, por lo tanto con arquitectura x86. Fueron diseñados para competir con la línea de procesadores Celeron de Intel. La

[31] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/AMD_Am5x86
[32] Extraído el 10 de Julio de 2009 desde http://www.slideshare.net/dElyya/soquet-y-microprocesador-presentation
[33] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Amdk6_arch.png

diferencia principal entre los Athlon y los Duron es que los Duron solo tienen 64 Kbyte de memoria caché L2, frente a los 256 Kbyte de los Athlon.

- **El Athlon**: Originalmente el Athlon *Classic*, fue el primer procesador x86 de séptima generación y en un principio mantuvo su liderazgo de rendimiento sobre los microprocesadores de Intel. El procesador Athlon se lanzó al mercado el 21 de agosto de 1999. El primer núcleo del Athlon, conocido en clave como "K7", estuvo disponible inicialmente en versiones de 500 a 650 MHz, pero después alcanzó velocidades de hasta 1 GHz. Internamente el Athlon es un rediseño de su antecesor, al que se le mejoró sustancialmente el sistema de coma flotante, esto es, 3 unidades de punto flotante que pueden trabajar simultáneamente y se le aumentó la memoria caché de primer nivel (L1) a 128 KB (64 KB para datos y 64 KB para instrucciones). Además incluye 512 KB de caché de segundo nivel (L2) externa al circuito integrado del procesador y funcionando, por lo general, a la mitad de velocidad del mismo.
- **El AMD Opteron**: *Fue el primer microprocesador con arquitectura x86 que usó conjunto de instrucciones AMD64, también conocido como x86-64. También fue el primer procesador x86 de octava generación. Fue puesto a la venta el 22 de abril de 2003 con el propósito de competir en el mercado de procesadores para servidores, especialmente en el mismo segmento que el Intel Xeon. La ventaja principal del Opteron es la capacidad de ejecutar tanto aplicaciones de 64 bits como de 32 bits sin ninguna penalización de velocidad. Las nuevas aplicaciones de 64 bits pueden acceder a más de 18 Hexabytes de memoria, frente a los 4 gigabytes de las de 32 bits.*[34]
- **Sempron:** Es un procesador de bajo costo con arquitectura X86 fabricado por AMD. El AMD Sempron reemplaza al procesador Duron, siendo su principal competidor el procesador Celeron de Intel. Las primeras versiones fueron lanzadas al mercado en agosto de 2004. Las versiones iníciales de este procesador estaban basadas en el núcleo *Thoroughbred*/*Thorton* del Athlon XP, con una caché de segundo nivel de 256 KB y un bus de 333 MHz. Posteriormente el Sempron se basó en el núcleo Barton del Athlon XP. Esta versión tenía un índice de prestaciones relativas de 3000+ y poseía una caché de segundo nivel de 512 KB.
- **AMD Turion 64:** Es una versión de bajo consumo del procesador AMD Athlon 64 destinada a los ordenadores portátiles y constituye la respuesta comercial de AMD a la plataforma Centrino de Intel. Se presentan en dos series, ML con un consumo máximo de 35 W y MT con un consumo de 25 W, frente a los 27 W del Intel Pentium M. Es compatible con el Socket 754 de AMD y dispone de 512 o 1024 KB de cache L2 y un controlador de memoria de 64 bit integrado.
- **Phenom:** nombre dado por AMD a la generación de procesadores de tres y cuatro núcleos, basando su arquitectura en el K10, estos reemplazan la línea de los athlon dos núcleos. Los Phenom, otorgan alta definición,

[34] Extraído el 10 de Julio de 2009 desde http://www.taringa.net/posts/info/2026465/Microprocesadores.html

rendimiento avanzado en procesos multitarea y consumo eficiente de energía. Existen Phenom de dos, tres y cuatro núcleos, basados en tecnología de 65 nanómetros, utilizan socket AM2+, incorpora 450 millones de transistores, controlador de memoria DDR integrado, tecnología 3D NOW, SSE, SSE2, SSE3, SSE4a.

Familia de Microprocesador CYRIX

Ha sido el tercero entre los procesadores Intel compatibles. Sus procesadores se han caracterizado por tener una regular unidad de coma flotante, por lo que no es una buena opción para los que utilicen programas CAD, 3D, e incluso juegos. Además de esto, se ha caracterizado también por sus diseños avanzados y "originales" lo que le ha provocado más de un inconveniente por falta de compatibilidad.

Sus primeras versiones de 6x86 Cyrix tuvieron serios problemas debido a su alto consumo, lo que generaba un calentamiento excesivo en los reguladores de tensión de la placa base.

Figura 25. ***Cyrix 6x86***[35]

Estos dispositivos presentaban también un problema con Windows NT4, ya que dicho sistema operativo desactivaba la caché del procesador, y por tanto éste se ejecutaba muy lentamente.

- **6x86MX:** Este es el primer microprocesador de Cyrix que lleva implementado el juego de instrucciones MMX. Sus desventajas son el bajo rendimiento de su coprocesador matemático, lo que conlleva que por lo menos dos de sus procesadores trabajen con una velocidad de bus de 75 y 83 MHz. De todas formas, para compensar este posible problema, Cyrix ha

[35] Extraído el 10 de Julio de 2009 desde http://www.cpu-world.com/CPUs/6x86/

implementado multiplicadores de x2, x2.5, x3 y x3.5, y garantizan que siempre se podrá trabajar con una frecuencia de bus más normal y ajustar el multiplicador para que la CPU trabaje a una frecuencia parecida a la autorizada.

- **MII:** Su diseño es idéntico al del 6x86MX, y sólo consigue imponerse a aquel por la mayor velocidad de sus nuevos modelos.

Figura 26. ***Cyrix MII***[36]

El problema de este procesador es tener una FPU poco potente. Este problema se agudiza con los actuales juegos 3D y unas cada vez mayores necesidades de este tipo de cálculos, lo que lo condena a quedar relegado a entornos ofimáticos.

Una de las ventajas es que funciona con cualquier placa preparada para MMX, no necesita de placas de última generación con voltajes más bajos de 2,9. Lo anterior permite actualizar la máquina a 300 MHz sin necesidad de cambiar de placa.

Familia de Microprocesadores Intel

Intel Corporation es una compañía pionera en el desarrollo y comercialización de microprocesadores. Actualmente, tiene una cuota del mercado mundial de los microprocesadores muy grande, ya que sus procesadores se utilizan en la mayoría de los computadores compatibles PC.

- **x86** *es la denominación genérica dada a ciertos procesadores de la familia Intel, sus compatibles y a la arquitectura básica de estos procesadores, por la terminación de sus nombres: 8086, 80286, 80386 y 80486. Los sucesores del 80486 pasaron a ser llamados por nombres no numéricos, bajo la denominación Pentium; sin embargo todavía se los llama*

[36] Extraído el 10 de Julio de 2009 desde http://www.cpu-world.com/CPUs/MII/TYPE-MII.html

procesadores de la familia x86. A continuación se describen los microprocesadores más representativos[37].

- **Intel 4004:** fue el primer microprocesador de 4 bits en un simple chip, así como el primero disponible comercialmente.
- **Intel 4040**: fue el sucesor del Intel 4004. Fue lanzado al mercado en 1974. El 4040 fue usado principalmente en juegos, pruebas, desarrollo, y equipos del control. El paquete del 4040 era más de dos veces el ancho del 4004 y tenía 24 pines en lugar de los 16 del 4004. El 4040 agregó 14 instrucciones, un espacio más grande para la pila (7 niveles en vez de 3), un espacio para programas de 8KB, 8 registros adicionales, y habilidades de interrupción.
- **8080 y 8085:** son procesadores de 8 bits fabricados por Intel a mediados de los 70. El 8085 era binariamente compatible con el 8080, pero exigía menos soporte de hardware; de esta forma permitía unos sistemas de microordenadores más simples y más económicos. El número 5 de la numeración del procesador 8085 proviene del hecho que solamente requería una alimentación de 5 voltios, no como el 8080 que necesitaba unas alimentaciones de 5 y 12 voltios. Entre las características del 8085 se destaca su bus de datos multiplexado, es decir, comparte los pines del bus de datos con los del bus de direcciones.
- **8086 y 8088**: son dos microprocesadores de 16 bits diseñados por Intel en 1978, iniciadores de la arquitectura x86. La diferencia entre el 8086 y el 8088 es que el 8088 utiliza un bus externo de 8 bits, para el empleo circuitos de soporte al microprocesador más económicos, en contraposición al bus de 16 bits del 8086.
- **80186 y 80188**: estos dos microprocesadores fueron desarrollados por Intel alrededor de 1982. Los 80186 y los 80188 son una mejora del Intel 8086 y del Intel 8088 respectivamente. Al igual que el 8086, el 80186 tiene un bus externo de 16 bits, mientras que el 80188 lo tiene de 8 bits como el 8088, para hacerlo más económico. La velocidad de reloj del 80186 y del 80188 es de 6 MHz. Ambos microprocesadores no fueron muy usados en ordenadores personales, sino que su uso principal fue como procesadores empotrados. Con el 80186 y el 80188 se introdujeron ocho nuevas instrucciones al conjunto de instrucciones x86.
- **80286**: (llamado oficialmente iAPX 286, también conocido como i286 o 286) es un microprocesador de 16 bits de la familia x86, que fue lanzado al mercado por Intel el 1 de febrero de 1982. Las versiones iníciales del i286 funcionaban a 6 y 8 MHz, pero acabó alcanzando una velocidad de hasta 20 MHz. El i286 fue el microprocesador más empleado en los IBM PC y compatibles entre mediados y finales de los años 80.
- **IA32**: Es la arquitectura de microprocesadores de 32 bits de Intel (Intel *Architecture* 32). Son los microprocesadores más usados en los ordenadores personales (PC). Esta gama de microprocesadores comenzó con el 80386, conocido luego popularmente como 386 o x86 para denominar a toda la gama. Los procesadores de Intel que siguieron y

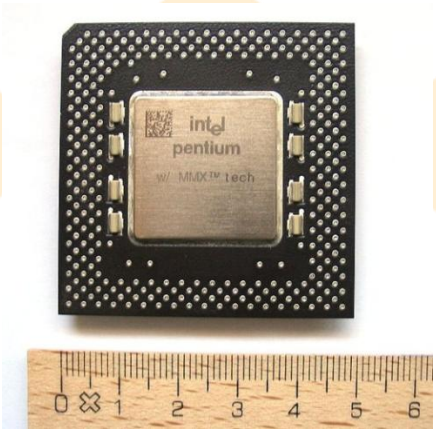
[37] Extraído el 10 de Julio de 2009 desde http://www.conocimientosweb.net/portal/term3873.html

mantuvieron la compatibilidad son el 486, Pentium (ó 586), Pentium II (ó 686), Pentium III y Pentium IV. La novedad de estos procesadores con respecto a sus predecesores es que incluyen gestión de memoria avanzada (segmentación, paginación, soporte de memoria virtual), unidad de punto flotante (FPU), y a partir del Pentium MMX cuentan con soporte para operaciones matriciales complejas, muy usadas en aplicaciones gráficas y multimedia.

- **80386**: (i386, 386) Es un microprocesador CISC con arquitectura x86. Durante su diseño se le llamó 'P3', debido a que era el prototipo de la tercera generación x86. El i386 fue empleado como la unidad central de proceso de muchos ordenadores personales desde mediados de los años 80 hasta principios de los 90.
- **80486**: (i486, 486) son una familia de microprocesadores de 32 bits con arquitectura x86 diseñados por Intel. Las principales diferencias con el i386 son las siguientes: tienen un conjunto de instrucciones optimizado, una unidad de coma flotante y un caché unificado e integrado en el propio circuito del microprocesador y una unidad de interfaz de bus mejorada.
- **Pentium**: este procesador se lanzó al mercado el 22 de marzo de 1993, sucediendo al procesador Intel 80486. Intel no lo llamó 586 sino Pentium, para poderlo registrar y así evitar que la competencia siguiera utilizando los mismos números que Intel para sus procesadores equivalentes (AMD 486, IBM 486, etc). También es conocido por su nombre clave P54C. El procesador Intel Pentium está formado por 3,1 millones de transistores y direcciona memoria con 64 bits. Integra dos memorias caché de 8 Kbyte (una para datos y otra para código) y tiene dos unidades aritmético lógicas (ALU), lo que le permite hacer tratamiento paralelo. Por tanto el Pentium puede ejecutar hasta dos instrucciones por ciclo de reloj. Está optimizado para ejecutar código de 16 bits.
- **Evolución del Pentium.** *En 1997, Intel presentó una evolución de su procesador Pentium, llamado Pentium MMX. Este se basaba en el mismo núcleo del Pentium original, pero se le añadió una memoria caché L1 de 32 Kbyte (frente a los 16 Kbyte del Pentium común), siendo 16 KB para datos y 16 KB para instrucciones, ordenadas en 4 vías de 4 KB cada una, y 57 nuevas instrucciones multimedia (de coma flotante), llamadas MMX, con el fin de ejecutar más rápidamente las futuras aplicaciones interactivas. Este procesador funcionaba entre 166 y 233 MHz. El nombre Pentium fue conservado por Intel para las generaciones siguientes de sus procesadores (Pentium Pro, Pentium II, Pentium III, Pentium IV y actualmente Pentium D), aunque exista una evolución importante en las arquitecturas.*[38]

[38] *Extraído el 10 de Julio de 2009 desde http://www.netzweb.net/html/print/pc/cpu/pentium.pdf*

Figura 27. ***Intel Pentium***[39]



- **Pentium II**: este microprocesador fue al mercado el 7 de mayo de 1997. Está basado en una versión modificada del núcleo P6, usado por primera vez en el Intel Pentium Pro. Los cambios fundamentales fueron mejorar el rendimiento en la ejecución de código de 16 bits, añadir el conjunto de instrucciones MMX y eliminar la memoria caché de segundo nivel del núcleo del procesador, colocándola en una tarjeta de circuito impreso junto a éste.

  El Pentium II se comercializó en versiones que funcionaban a una frecuencia de reloj de entre 166 y 450 MHz. La velocidad de bus era originalmente de 66 MHz, pero en las versiones a partir de los 333 MHz se aumentó a 100 MHz.

  Poseía 32 KB de memoria caché de primer nivel repartida en 16 KB para datos y otros 16 KB para instrucciones. La caché de segundo nivel era de 512 KB.

  Como novedad respecto al resto de procesadores de la época, el Pentium II se presentaba en un encapsulado SEC, con forma de cartucho. El cambio de formato de encapsulado se hizo para mejorar la disipación de calor. Este cartucho se conecta a las placas base de los equipos mediante una ranura Slot 1. Este microprocesador integra 7,5 millones de transistores.

[39] *Extraído el 10 de Julio de 2009 desde http://upload.wikimedia.org/wikipedia/commons/thumb/2/22/Pentium-mmx.jpg/606px-Pentium-mmx.jpg*

Figura 28. ***Intel Pentium II***[40]

- **Celeron**: es el nombre que lleva la línea de procesadores de bajo costo de Intel. El primer Celeron fue lanzado en agosto de 1998, y estaba basado en el Pentium II. Posteriormente, salieron nuevos modelos basados en las tecnologías Pentium III y Pentium IV. Los procesadores Celeron se dividen en dos grandes clases:

  P6: Basada en los procesadores Pentium II y Pentium III

  Netburst: Basada en los procesadores Pentium IV

- **Pentium III**: es un microprocesador de arquitectura i686. Fue lanzado el 26 de febrero de 1999. Las primeras versiones eran muy similares al Pentium II, siendo la diferencia más importante la introducción de las instrucciones SSE. Al igual que con el Pentium II, existía una versión Celeron de bajo costo y una versión Xeon para quienes necesitaban gran poder de cómputo. Esta línea ha sido eventualmente reemplazada por el Pentium IV, aunque la línea Pentium M está basada en el Pentium III.

Figura 29. ***Intel Pentium III***[41]

[40] Extraído el 10 de Julio de 2009 desde http://www.cpu-world.com/CPUs/Pentium-II/index.html

[41] Extraído el 10 de Julio de 2009 desde http://www.cpu-world.com/CPUs/Pentium-III/index.html

- **Pentium IV**: es un microprocesador de séptima generación basado en la arquitectura x86 y manufacturado por Intel. Es el primer microprocesador con un diseño completamente nuevo desde el Pentium Pro de 1995. El Pentium IV original, denominado *Willamette*, trabajaba a 1,4 y 1,5 GHz; y fue lanzado en noviembre de 2000.

  Para la sorpresa de la industria informática, el Pentium IV no mejoró el diseño P6 según las dos tradicionales formas para medir el rendimiento: velocidad en el proceso de enteros u operaciones de coma flotante. La estrategia de Intel fue sacrificar el rendimiento de cada ciclo para obtener a cambio mayor cantidad de ciclos por segundo y una mejora en las instrucciones SSE. Al igual que los demás procesadores de Intel, el Pentium IV se comercializa en una versión para equipos de bajo presupuesto (Celeron) y una orientada a servidores de gama alta (Xeon).

- **Pentium IV EDICIÓN EXTREMA**: en septiembre de 2003, Intel anunció la edición extrema (*Extreme Edition*) del Pentium IV. Se destacó en el área de la codificación multimedia, ya que superaba la velocidad de los anteriores Pentium 4 y a toda la línea de AMD.

- **Pentium M**: Fue Introducido en marzo de 2003. Este microprocesador cuenta con una arquitectura x86 (i686). Originalmente fue diseñado para uso en ordenadores portátiles. Su nombre en clave antes de su introducción era '"Banias"'. Todos los nombres clave del Pentium M son lugares de Israel, la ubicación del equipo de diseño del Pentium M.

  El Pentium M es una versión fuertemente modificada del diseño del Pentium III. Está optimizado para un consumo de potencia eficiente, una característica vital para ampliar la duración de la batería de los ordenadores portátiles. Funciona con un consumo medio muy bajo y desprende mucho menos calor que los procesadores de ordenadores de sobremesa. El Pentium M funciona a una frecuencia de reloj más baja que los procesadores Pentium IV normales, pero con un rendimiento similar, y puede igualar o superar el rendimiento de un Pentium IV Prescott a 3 GHz. Este procesador forma parte de la plataforma Intel Centrino.

- **Pentium D**: fue introducido por Intel en 2005. Un chip Pentium D consta básicamente de dos procesadores Pentium IV (de núcleo Prescott) en una única pieza de silicio con un proceso de fabricación de 90nm. El nombre en clave del Pentium D antes de su lanzamiento era "Smithfield". Incluye una tecnología DRM (Digital Rights Management) para hacer posible un sistema de protección anticopia de la mano de Microsoft.

- **Xeon**: es la actual generación de microprocesadores Intel para servidores PC. El primer procesador Xeon apareció en 1998 como Pentium II Xeon. El

Pentium II Xeon utilizaba tanto el chipset 440GX como el 450NX. En el año 2000, el Pentium II Xeon fue reemplazado por el Pentium III Xeon

El último miembro añadido a la familia Xeon es el procesador Xeon MP, lanzado en 2002, que combinaba la tecnología Hyper-Threading con NetBurst. Sus chipsets utilizan el socket 603 y tiene versiones GC-LE (2 procesadores, 16Gb de memoria direccionable) y GC-HE (4 procesadores o más, 64Gb direccionables), todos usando un bus de 400MHz.

Como la familia x86/IA-32 estándar de Intel de procesadores PC de escritorio, la línea de procesadores Xeon es de 32 bits. Se planea una versión de 64 bits de Xeon que complementará (o reemplazará) a la CPU Itanium de Intel. El 26 de junio de 2006, Intel anunció la nueva generación de Xeon con tecnología de doble núcleo. Intel afirma que este nuevo procesador brindará un 80% más de rendimiento por vatio y que será hasta un 60% más rápido que la competencia.

- **Intel Core Duo**: es el microprocesador de Intel que cuenta con dos núcleos de ejecución. Fue lanzado en enero del 2006. El microprocesador Intel Core Duo está optimizado para las aplicaciones de subprocesos múltiples y para la multitarea. Puede ejecutar varias aplicaciones simultáneamente, como juegos con gráficos o programas que requieran muchos cálculos, al mismo tiempo que puede descargar música o analizar su PC con su antivirus en el segundo plano. Este microprocesador implementa 2Mb de caché compartida para ambos núcleos más un bus frontal de 667Mhz; además implementa un nuevo juego de instrucciones para multimedia (SSE3) y mejoras para las SSD y SSD2; sin embargo, el desempeño con enteros es ligeramente inferior debido a su caché con mayor latencia. Intel Core Duo es el primer microprocesador de Intel usado en las computadoras Apple Macintosh.

  El Core Duo contiene 151 millones de transistores. El núcleo de ejecución del procesador contiene un pipeline de 12 etapas con velocidades previstas de ejecución entre 2.33 y 2.50 GHz. La comunicación entre la caché L2 y los dos núcleos de ejecución es controlada por un módulo de bus árbitro que elimina el tráfico de coherencia a través del bus frontal (FSB), con el costo de elevar la latencia de la comunicación de núcleo a L2 de 10 ciclos de reloj. El incremento de la frecuencia de reloj contrapesa el impacto del incremento en la latencia.

  Las nuevas características de administración de energía incluyen control mejorado de temperatura, así como escalado independiente de energía entre los 2 núcleos, lo que resulta en un manejo de energía mucho más eficiente que los diseños anteriores.

- **Core 2 Quad:** microprocesador de cuatro núcleos, basado en micro arquitectura de 45 nanómetros, presenta cache hasta de 12 MB, bus frontal de 1333 MHz.

- **Core 2 Extreme**: microprocesador de cuatro núcleos, tecnología de 45 nanómetros, velocidad de 3,2 GHZ, 12 MB de cache total, bus frontal de 1600 MHz.

- **Core i7:** utilizan micro arquitectura Nehalem, sucesor de la línea de los Core 2, soporta memoria DDR3, velocidad de reloj hasta de 4GHz, para socket LGA1366.

LECCIÓN 2: MICROPROCESADORES DE 8 BITS.

La línea de microprocesadores de 8 bits, es considerada como el núcleo para otras familias y fabricantes, el INTEL 8080 e INTEL 8085, fueron utilizados en computadores a finales de los 70s y comienzos de los 80s, específicamente el diseño del microprocesador 8080 fue la base para la creación del Z80 microprocesador que supero ampliamente las prestaciones y mercado del 8080.

En la actualidad hay muchas versiones clónicas de esos microprocesadores inclusive núcleos de microcontroladores siguen como diseño base en la CPU el diseño de los Z80. A continuación se hará una descripción general, de dos de los microprocesadores de 8 bits, más representativos.

Características Generales

El **Intel 8080** fue el sucesor del Intel 8008; esto se dio fácilmente ya que eran compatibles en muchos aspectos y porque utilizaban el mismo conjunto de instrucciones. El i8080 salió al mercado en el año de 1974, con un empaquetado más grande, DIP de 40 pines, lo que permitió al 8080 proporcionar un bus de direcciones de 16 bits y un bus de datos de 8 bits, permitiendo el fácil acceso a 64 KB de memoria. Tenía siete registros de 8 bits, seis de los cuales se podían combinar en tres registros de 16 bits, un puntero de pila en memoria de 16 bits, que reemplazaba la pila interna del 8008, y un contador de programa de 16 bits.

El 8080 podía manejar 256 puertos de E/S, los cuáles eran accedidos por los programas mediante instrucciones dedicadas de E/S. Este esquema usaba un espacio de direcciones separado para las entradas/salidas, pero ahora es menos

común que el de mapeo de memoria para dispositivos o puertos de E/S. El 8080 fue implementado en muchos de los primeros microcomputadores, tales como la Altair 8800 de MITS y el IMSAI 8080, formando la base para las máquinas que corrían el sistema operativo CP/M. El primer microcomputador en una simple tarjeta fue construido en base al 8080. Por medio de su arquitectura de conjunto de instrucciones (ISA), el 8080 hizo un impacto duradero en historia del computador.

El **Intel 8085** fue presentado en el año de 1976 y se consideró un estándar de la industria de microprocesadores durante mucho tiempo. Tenía 20.000 transistores, funcionaba a 1 MHz y estaba encapsulado en un chip de 40 pines.

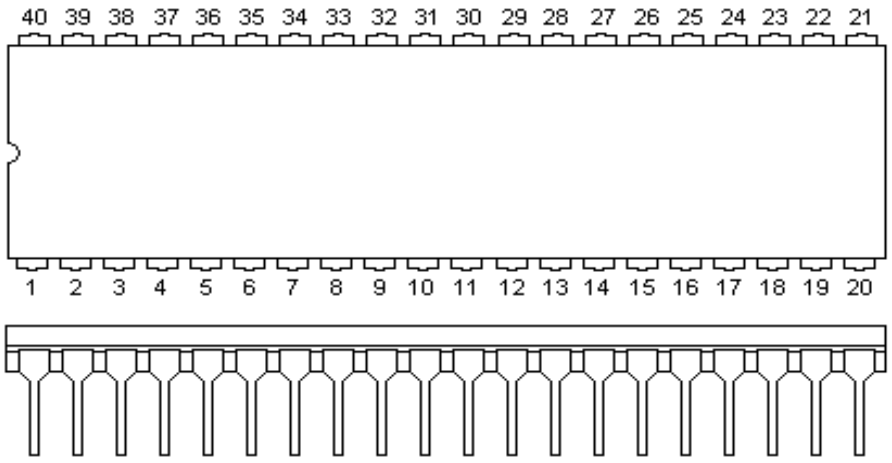
Este microprocesador tiene un total de 16 líneas de dirección. Las 8 líneas de dirección más significativas están conectadas a los pines A_8 y A_{15}, y se conectan directamente al bus de direcciones. El bus de datos de 8 bits está compartido con las 8 líneas de dirección menos significativas. Los pines de alimentación V_{CC} y V_{SS} deben ser conectados a una fuente de +5V. El 8085 tiene circuitería de reloj permanente y solo necesita conectar los pines de entrada X_1 y X_2 a un cristal.

Funciones y diagrama de pines

El **microprocesador 8080** es un chip de 40 pines con encapsulado de doble línea o DIP. Tiene 16 líneas de direcciones (pin 1, pines 25 al 27 y pines 29 al 40) y 8 líneas bidireccionales para el de bus datos (pines 3 al 10). Todas estas líneas se encargan del direccionamiento de la memoria y los dispositivos de entrada/salida, y del manejo de los datos.

A continuación se presenta el diagrama de pines del i8080 y se anexa una tabla con la descripción de cada una de sus funciones.

Figura 30. ***Intel 8080 pines***[42]



[42] Extraído el 10 de Julio de 2009 desde http://www.alpertron.com.ar/8080.HTM

Tabla 5. ***Intel 8080 descripción de pines***[43]

Pin	Nombre	Descripción
1	**A10**	Bus de direcciones
2	**GND**	Referencia de tierra. Todas las tensiones se miden con respecto a este punto.
3	**D4**	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal de control que indica salida a periférico.
4	**D5**	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal que indica si el uP está en ciclo de búsqueda de instrucción.
5	**D6**	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal de control que indica entrada de periférico.
6	**D7**	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal de control que indica lectura de memoria.
7	**D3**	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal que indica que el uP se ha detenido.
8	**D2**	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal que indica que se realiza una operación con el stack.
9	**D1**	Si SYNC = 0: Bus de datos. Si SYNC = 1: Modo lectura/escritura.
10	**D0**	Si SYNC = 0: Bus de datos. Si SYNC = 1: Señal de reconocimiento de interrupción.
11	**-5V**	Una de las tres patas de alimentación del 8080.
12	**RESET**	Señal de borrado de todos los registros internos del 8080. Para ello, ponerlo a uno durante tres ciclos de reloj como mínimo.
13	**HOLD**	Sirve para poner los buses en alta impedancia para el manejo de DMA (acceso directo a memoria).
14	**INT**	Señal de pedido de interrupción.
15	**CLK2**	Señal de reloj (debe venir del generador de reloj 8224).
16	**INTE**	Señal de aceptación de interrupción.
17	**DBIN**	Indica que el bus de datos está en modo lectura.
18	**/WR**	Indica que el bus de datos está en modo escritura.

[43] Extraído el 10 de Julio de 2009 desde http://www.alpertron.com.ar/8080.HTM

19	**SYNC**	Este pin se pone a uno cuando comienza una nueva instrucción.
20	**+5V**	Una de las tres patas de alimentación del 8080.
21	**HLDA**	Reconocimiento de HOLD.
22	**CLK1**	Señal de reloj (debe venir del generador de reloj 8224).
23	**READY**	Sirve para sincronizar memorias o periféricos lentos (detiene al 8080 mientras se lee o escribe el dispositivo).
24	**WAIT**	Cuando vale "1", el 8080 está esperando al periférico lento.
25-27	**A0 - A1**	Bus de direcciones.
28	**+12V**	Una de las tres patas de alimentación del 8080.
29-40	**A3 - A11**	Bus de direcciones.

Tabla 6. ***Descripción general de pines***[44]

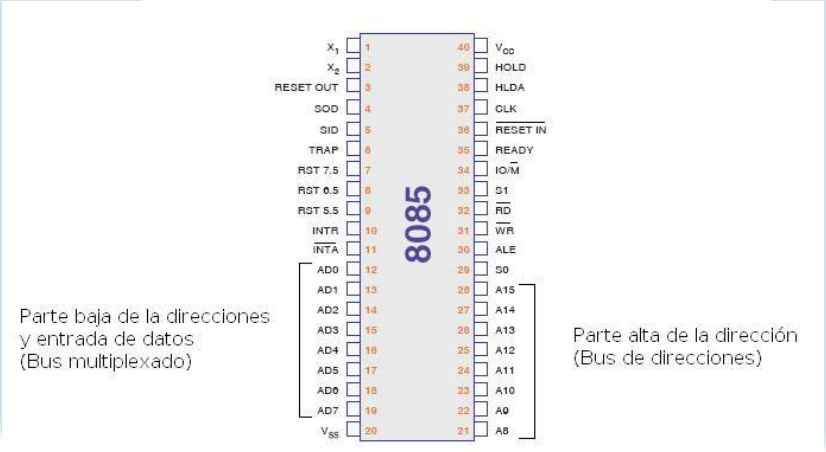
NOMBRE DEL PIN	TIPO	DESCRIPCIÓN
Líneas de dirección (A_{15} - A_0)	Salida	Provee la dirección de memoria o número de dispositivo entrada/salida. A_0 denota el bit menos significativo
Líneas de datos (D_7 - D_0)	Entrada/Salida	Provee comunicación bidireccional de instrucciones o datos entre la CPU y la memoria o dispositivos de entrada/salida. D_0 es el bit menos significativo.
Señal de sincronismo (SYNC)	Salida	La señal de sincronismo indica el comienzo de cada ciclo máquina.
Señal de entrada de datos (DBIN)	Salida	Indica a los dispositivos externos que el bus de datos está en modo de entrada.
Listo (READY)	Entrada	Indica al 8080a que hay un dato válido de la memoria o dispositivo de entrada/salida en el bus de datos. Esta señal permite sincronizar la CPU con dispositivos más lentos. Si después de enviar una dirección no se recibe la señal de READY, la CPU entra en estado de espera hasta recibirla.
Espera (WAIT)	Salida	Indica que la CPU se encuentra en estado de espera.
Escritura ($\overline{WR}$)	Salida	Señal de control de escritura para la memoria o dispositivo de entrada/salida.
Reposo (HOLD)	Entrada	La CPU entra en estado de reposo, es decir, permite que un dispositivo externo tenga el control del bus de direcciones y de datos.
Reconocimiento de reposo (HLDA)	Salida	Esta señal aparece en respuesta a la señal HOLD e indica que el bus de datos y direcciones entran en estado de alta impedancia.

[44] Téllez, 2007

Habilitación de interrupción (INTE)	Salida	Indica el estado interno del flip-flop de habilitación de interrupción. Cuando es 0 se deshabilita la interrupción y cuando es igual a 1 es habilitada. Cuando la CPU se reinicia las interrupciones son deshabilitadas.
Petición de interrupción (INT)	Entrada	La CPU reconoce la petición de interrupción y la atiende al final de la actual instrucción o mientras esté detenido. Si se encuentra en estado de reposo o deshabilitadas las interrupciones no atiende la petición.
Reinicio (RESET)	Entrada	Mientras la señal esté activa el contador de programa es borrado. Después se reanudará en la posición de memoria 0.
Tierra (V_{SS})	Alimentación	Tierra o referencia
V_{DD}	Alimentación	$+12V \pm 5\%\ V$
V_{CC}	Alimentación	$+5V \pm 5\%\ V$
V_{BB}	Alimentación	$-5V \pm 5\%\ V$
Fases de reloj (ϕ_1, ϕ_2)	Reloj	2 fases externas de reloj suplementarias

Al igual que el i8080, el **microprocesador 8085**, su sucesor, es un procesador de 8 bits con un encapsulado DIP de 40 pines. Su diagrama de pines se ilustra en la siguiente figura. En la tabla anexa se resumen los nombres y descripciones de sus pines.

Figura 31. ***Intel 8085***[45]



[45] Extraído el 10 de Julio de 2009 desde http://upload.wikimedia.org/wikipedia/commons/7/75/I8085_Buses.JPG

Tabla 7. Intel 8085 Descripción de pines[46]

NOMBRE DEL PIN	TIPO	DESCRIPCION
(AD_7 - AD_0)	Entrada/salida	Bus de direcciones y datos, tres estados.
(A_{15} - A_8)	Salida	Bus de direcciones, tres estados.
ALE	Salida	Señal de habilitación del cerrojo de dirección.
$\overline{RD}$	Salida	Control de lectura.
$\overline{WR}$	Salida	Control de escritura.
IO/ $\overline{M}$	Salida	E/S o indicador de memoria.
S_0, S_1	Salida	Indicadores del estado del bus.
READY	Entrada	Petición de estado de espera.
SID	Entrada	Entrada de datos serie.
SOD	Salida	Salida de datos serie
HOLD	Entrada	Petición de reposo.
HLDA	Salida	Conocimiento de reposo.
INTR	Entrada	Petición de interrupción.
TRAP	Entrada	Petición de interrupción no enmascarable.
RST5.5	Entrada	Petición de interrupción hardware vectorizadas.
RST6.5	Entrada	Petición de interrupción hardware vectorizadas.
RST7.5	Entrada	Petición de interrupción hardware vectorizadas.
$\overline{INTA}$	Salida	Conocimiento de interrupción.
$\overline{REINICIALIZACIÓN}$ IN	Entrada	Reinicialización del sistema.
REINICIALIZACIÓN OUT	Salida	Reinicialización de los periféricos
X_1, X_2	Entrada	Conexiones al cristal o RC.
CLK	Salida	Señal de reloj
V_{CC}, V_{ss}	Alimentación	+ 5V, tierra.

Con las funciones añadidas del 8085, los 40 pines del chip no eran suficientes para todas las entradas y salidas. Por esta razón, el fabricante utiliza los pines 12 al 19 como líneas de propósito doble: de direcciones y datos (AD_7 - AD_0). Se dice entonces que esta unidad tiene un bus multiplexado de datos y direcciones.

El 8085 tiene una señal especial para informar a los periféricos cuándo el bus de direcciones/datos envía una dirección y cuando funciona como bus de datos. La señal especial se denomina señal de control de habilitación del cerrojo de direcciones (ALE). Se debe observar que los pines del bus de direcciones/datos son bidireccionales o pueden ser también de tres estados o alta impedancia.

Las salidas de control de lectura ($\overline{RD}$) y escritura ($\overline{WR}$) son utilizadas para informar a la memoria o dispositivos de entrada/salida cuando hay que enviar o recibir datos vía el bus de datos. La entrada de reinicialización ($\overline{REINICIALIZACIÓN}$) actúa como el reset de la CPU, ya que el contador de

[46] Téllez, 2007

programa se ubica en la posición 0000H de la memoria. Los buses de datos y direcciones y líneas de control están en triestado durante la reinicialización. El contenido de los registros internos también puede ser alterado durante una reinicialización. El pin REINICIALIZACIÓN OUT está asociado con la operación de reinicialización. Como la CPU se está reinicializando, el pin REINICIALIZACIÓN OUT envía una señal a los periféricos que les informa que el sistema se va reiniciar.

La entrada de petición de interrupción (INTR) al 8085 es una interrupción de propósito general. Sin embargo, puede ser habilitada o deshabilitada por instrucciones de software. Además de la petición regular de interrupciones, la CPU tiene otras cuatro entradas de interrupción. Estos son los pines de entrada TRAP, RST7.5, RST6.5 y RST5.5. TRAP es la interrupción de mayor prioridad. La interrupción INTR salta a una dirección dictada por una instrucción especial recibida, de un dispositivo periférico, cuando es activada la salida de conocimiento de interrupción ($\overline{INTA}$) de la CPU.

EL pin de entrada HOLD (reposo) notifica a la CPU que otro dispositivo quiere utilizar los buses de direcciones y datos. Esto puede ocurrir durante operaciones de acceso directo a memoria. Después de recibir una entrada HOLD, la CPU completará las transferencias actuales de datos en los buses. Entonces los pines de direcciones, datos, $\overline{RD}$, $\overline{WR}$ e IO/$\overline{M}$ del 8085 se ponen en alta impedancia para no interferir con las transferencias de datos en los buses. Una salida HLDA (conocimiento de reposo) indica a un periférico que se ha recibido una petición HOLD y que el microprocesador abandonará el control de los buses en el siguiente ciclo de reloj.

Tabla 8. ***Señales de control del 8085***[47]

Señales de control del 8085			***Status de ciclos de máquina***
IO/$\overline{M}$	S_0	S_1	
0	0	1	**Escritura en memoria**
0	1	0	**Lectura de memoria**
1	0	1	**Escritura de E/S**
1	1	0	**Lectura de E/S**
0	1	1	**Búsqueda del código de operación**
1	1	1	**Conocimiento de interrupción**
*****	0	0	**Alto**
*****	X	X	**Reposo**
*****	**X**	**X**	**Reinicialización**

* = condición de tres estados X = no especificado.

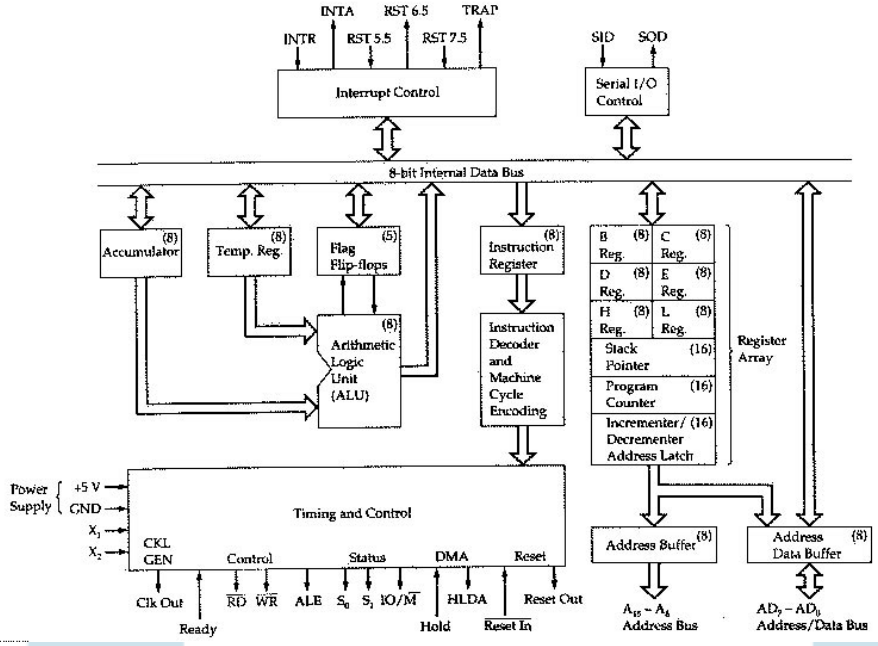
[47] Téllez, 2007

Las salidas IO/$\overline{M}$, S_0 y S_1 son señales de control que notifican a los periféricos el tipo de ciclo de máquina que está realizando la CPU. Los tipos de ciclo de máquina aparecen a la derecha de la tabla. La combinación adecuada de las señales de salida de los pines IO/$\overline{M}$, S_0 y S_1 se detalla en la columna de la izquierda.

Arquitectura interna

La organización interna o arquitectura del **microprocesador 8080** se muestra en la siguiente figura.

Figura 32. ***Arquitectura Intel 8080***[48]



Registros

El formato de datos e instrucciones que se manejan, es de enteros de 8 bits. Todas las transferencias sobre el bus de datos del sistema se realizan con este formato.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Las instrucciones de programa pueden tener tamaño de uno, dos o tres bytes de longitud. Los múltiples bytes de instrucciones deben ser guardados en palabras sucesivas en la memoria del programa. El formato de las instrucciones dependerá de la operación ejecutada en particular. Las entradas o salidas de datos a la

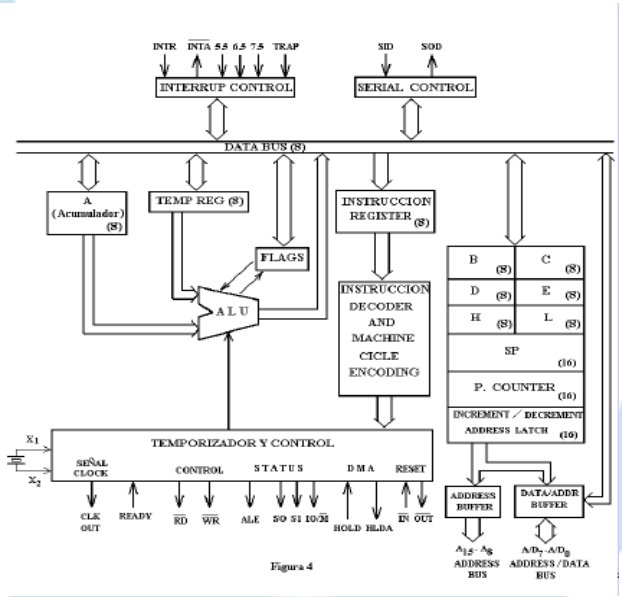
[48] Datasheet Intel 8080

unidad, se realizan a través del bus de datos interno de 8 bits. Los datos pueden fluir desde el bus de datos interno al acumulador de 8 bits o a los registros temporales, registro de instrucciones, unidad de control de interrupción, unidad de control de E/S o cualquiera de los registros de propósito general (B, C, D, E, H y L). Cuenta también con un puntero de pila de 16 bits, un contador de programa de 16 bits y un buffer de datos/direcciones.

La unidad aritmético lógica está alimentada por dos registros de 8 bits (registro acumulador y registro temporal). Los flip-flops señalizadores tienen cinco indicadores de estado. El registro de instrucciones alimenta al decodificador de instrucciones, el cual interpreta la instrucción actual y determina el microprograma que debe seguir. El decodificador de instrucciones también indica a las secciones de control y temporización sobre la secuencia de eventos que deben realizarse. La sección de temporización y control coordina las acciones del procesador y los periféricos.

El **microprocesador 8085**, cuenta con las mismas unidades estructurales del i8080: una unidad de control, una unidad lógico aritmética, un conjunto de registros y un bus interno de interconexión; pero incluye dos componentes adicionales bastante interesantes: una unidad para el control de las interrupciones y una comunicación serie sencilla. También utiliza registros de 8 y 16 bits. Seis de estos registros pueden utilizarse como registros de 8 bits o como tres registros de 16 bits. A continuación se presenta el diagrama interno o la arquitectura del **microprocesador 8085**.

Figura 33. ***Arquitectura Intel 8085***[49]



[49] Extraído el 10 de Julio de 2009 desde http://www.monografias.com/trabajos32/microprocesador-8085/microprocesador-8085.shtm

Algunos de los registros más importantes, son los siguientes:

- El acumulador: es el centro de todas las operaciones de la ALU, la cual incluye instrucciones aritméticas, lógicas, de carga y de E/S. Es un registro de 8 bits.
- Los registros de propósito general BC, DE y HL pueden ser utilizados como seis registros de 8 bits o como tres registros de 16 bits dependiendo de la instrucción que se vaya a ejecutar. El registro HL llamado puntero de datos puede ser utilizado para apuntar a direcciones. Pocas instrucciones utilizan los registros BC y DE como punteros de dirección, ya que normalmente se utilizan como registros de datos de propósito general.

- El contador de programa (PC): Contiene una dirección de 16 bits.

- El puntero de pila (SP): Es un registro de 16 bits.
- El registro de señalizadores (Flags) contiene cinco señalizadores de 1 bit con la información del estado de la CPU. Estos señalizadores son utilizados generalmente por instrucciones de bifurcación condicional y llamada y vuelta de subrutina. Este registro se presentan a continuación:

S	**Z**		**AC**		**P**		**CY**
B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0

- Flag de arrastre (CY) es puesto a 1 ó 0 por operaciones aritméticas, y su estado es examinado por las instrucciones del programa. Un desbordamiento de una suma de 8 bits genera en el registro un set (1) en la bandera CY. En la resta el señalizador actúa como un señalizador de "préstamo", indicando que el minuendo es menor que el sustraendo si el señalizador está en 1.
- Flag de paridad (P) examina el número de bits 1 del acumulador. Si éste contiene un número par de 1, se dice que existe paridad par y el señalizador de paridad se pone a 1. Sin embargo, si el acumulador contiene un número impar de 1 (paridad impar), el señalizador de paridad se pone en 0.
- Flag de arrastre auxiliar (AC) indica desbordamiento o arrastre del bit 3 del acumulador de la misma forma que el señalizador de arrastre indica desbordamiento del bit 7. Esta señalizador es utilizado comúnmente en aritmética BCD (decimal codificado binario).
- Flag de cero (Z) se pone a 1 si el resultado de ciertas operaciones es cero. El señalizador de cero está a 0 si el resultado no es cero.
- Flag de signo (S) indica la condición del bit más significativo del acumulador después de la ejecución de instrucciones aritmético-lógicas. Estas instrucciones utilizan el MSB del dato para representar el signo del número contenido en el acumulador. Un señalizador de signo a 1 representa un número negativo, mientras que si es 0 representa un número positivo.

Generador del reloj interno: el 8085 incorpora un generador completo de reloj en el mismo chip. Requiere solamente la adición de un cristal de cuarzo para establecer la temporización de su operación. El pin de salida CLK es una salida de reloj que tiene una frecuencia de una vez y media la frecuencia del cristal.

Modos de direccionamiento

|En los microprocesadores i8080 e i8085, se tienen principalmente cinco modos de direccionamiento. Estos son:

- **Direccionamiento inmediato:** Por ejemplo, se tiene la instrucción, " ADDI: suma inmediata", El microprocesador busca el código de operación en la memoria del programa. Después de decodificar la instrucción encuentra el dato inmediato en la siguiente posición consecutiva de la memoria del programa, después del código de operación. En este caso, el dato inmediato se suma al contenido del acumulador y el resultado es guardado en el acumulador.
- **Direccionamiento directo:** este tipo de instrucciones se especifican utilizando formato de instrucción de 3 bytes. El byte 1 contiene el código de operación para la instrucción de direccionamiento directo. El byte 2 contiene el byte de orden inferior de la dirección del operando. El byte 3 de la instrucción contiene el byte de orden superior de la dirección del operando. Por ejemplo, la instrucción, "LDA: cargar A". En este caso el código de operación es 3AH. Los 2 siguientes bytes de la memoria son ensamblados por el microprocesador en una dirección de 16 bits. Esta dirección de la memoria de datos es accedida por la CPU y su contenido cargado en el acumulador. as instrucciones que utilizan direccionamiento directo a veces son evitadas ya que requieren de mucho espacio en la memoria de programa, al igual que tiempo de ejecución relativamente largo, debido a los numerosos accesos a memoria que se necesitan.
- **Direccionamiento indirecto de registro:** las instrucciones de registro indirecto referencian la memoria utilizando el contenido de un registro par para señalar la dirección del operando. Por ejemplo, la instrucción, "ADDM: sumar memoria" Esta instrucción suma el contenido del acumulador con el contenido de la posición de memoria indicado por la dirección del registro HL de la CPU.
- **Direccionamiento implicado:** el modo de direccionamiento de ciertas instrucciones es implicado o inherente a, por la función de la instrucción. Por ejemplo, la instrucción: "STC: poner a 1 el señalizador de arrastre", Esta instrucción se relaciona con el señalizador de arrastre solamente y no con otros registros o posiciones de memoria.
- **Direccionamiento de registro:** cuando se utilizan el tipo de instrucciones que utilizan direccionamiento de registro, se especifican la fuente del

operando y la operación. Consideremos la ejecución de la instrucción, "ADD C: sumar el registro C con el acumulador" Una vez que se ejecuta la instrucción el resultado se deposita en el acumulador. Se considera del tipo de direccionamiento de registros porque ambos operandos estaban localizados en registros internos del microprocesador. Estas instrucciones son muy eficientes ya que solamente utilizan un espacio de la memoria de programa de 1 byte. También se ejecutan rápidamente porque no tienen que buscar operandos en memoria. Los accesos de memoria aumentan el tiempo de ejecución de la instrucción.

- **Modos de direccionamiento combinados:** algunas instrucciones son de direccionamiento combinado, como por ejemplo la llamada a subrutina CALL, en la cual se tiene direccionamiento directo e indirecto de registro. En este caso el direccionamiento directo especifica la dirección de la subrutina deseada y el direccionamiento indirecto es el del puntero de pila.

Repertorio de instrucciones

El grupo de instrucciones que un microprocesador puede ejecutar, se denomina su repertorio de instrucciones. Todas las instrucciones de un programa son almacenadas en un área denominada memoria de programa. El conjunto de instrucciones del 8080 es el mismo del 8085; aunque este último cuenta con dos nuevas instrucciones en su repertorio.

Figura 34. ***Set instrucciones 8085***[50]

[50] Extraído el 10 de Julio de 2009 desde http://www.monografias.com/trabajos32/microprocesador-8085/microprocesador-8085.shtml

LECCIÓN 3: MICROPROCESADORES DE 16 BITS.

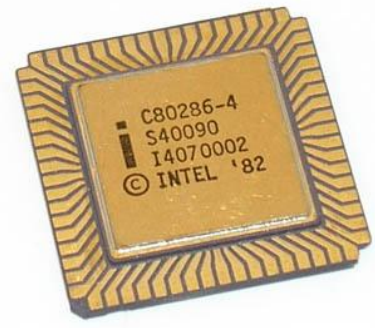
El Intel 80286 es un microprocesador de 16 bits, su relevancia radica en la implementación de este microprocesador en los primeros equipos de cómputo personal y compatible, se puede decir que la computadora personal nació con la implementación de estos microprocesadores como núcleos del sistema.

Características Generales

*El **Intel 80286** (llamado oficialmente iAPX 286, también conocido como i286 o 286) es un microprocesador de 16 bits de la familia x86. Fue lanzado al mercado por Intel el 1 de febrero de 1982. Las versiones iníciales del i286 funcionaban a 6 MHz y a 8 MHz, pero acabó alcanzando una velocidad de hasta 20 MHz. El i286 fue el microprocesador más empleado en los IBM PC y compatibles entre mediados y finales de los años 80*[51].

El i286 funciona el doble de rápido que su predecesor, el Intel 8086, y puede direccionar hasta 16 Mbyte de memoria RAM, en contraposición a 1 Mbyte del i8086. En máquinas DOS, esta memoria adicional solo podía ser accedida a través de emulación de memoria expandida previamente habilitada mediante software la memoria extendida. De todos modos, pocos ordenadores basados en el i286 tuvieron más de 1 Mbyte de memoria.

Figura 35. *80286*[52]



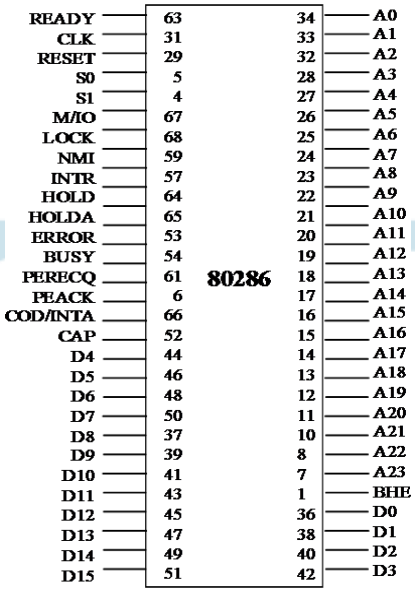
El i286 fue diseñado para ejecutar aplicaciones multitarea, incluyendo comunicaciones, control de procesos en tiempo real y sistemas multiusuario. A pesar de su gran popularidad, hoy en día quedan pocos ordenadores con el i286 funcionando.

[51] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Intel_80286
[52] Extraído el 10 de Julio de 2009 desde http://www.cpu-world.com/CPUs/80286/index.html

Funciones y diagrama de pines

La siguiente figura presenta la asignación de terminales del microprocesador 80286. La descripción de sus pines más importantes se hará en conjunto con el microprocesador i80386DX.

Figura 36. ***Pines del 80286***



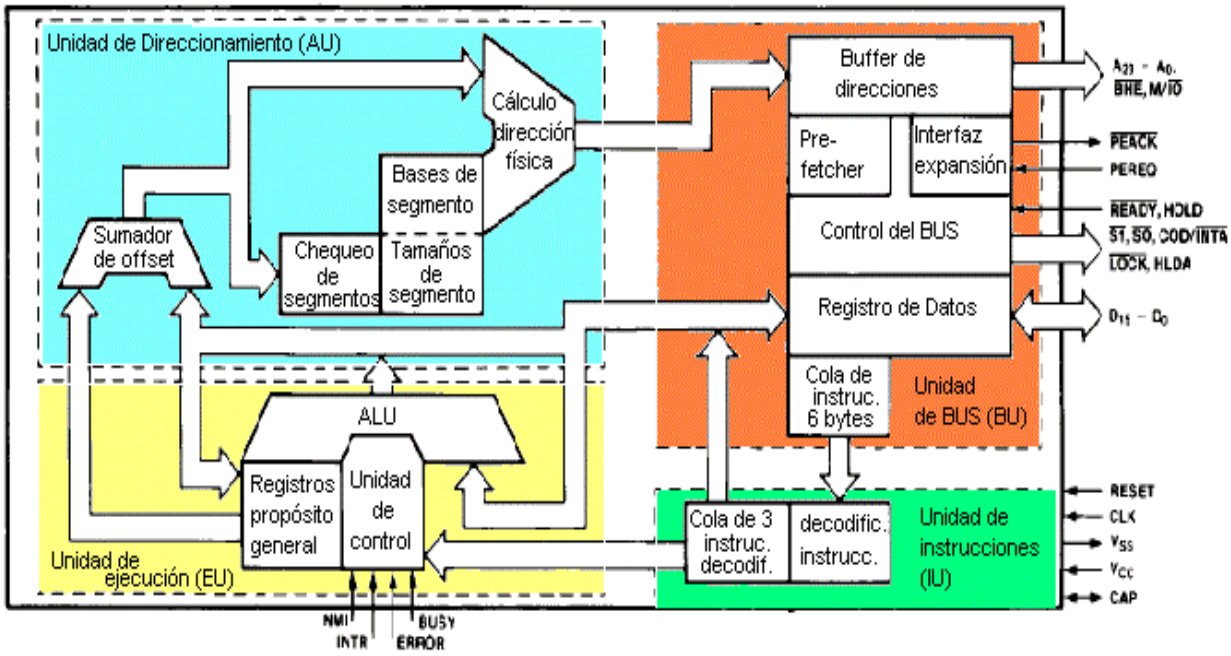
Arquitectura interna

La siguiente figura muestra el diagrama de bloques interno del microprocesador 80286. Este microprocesador no incorpora periféricos internos; en lugar de ello, contiene una unidad de administración de memoria (MMU), referida como unidad de direccionamiento en el diagrama de bloques.

Las terminales A_{23}-A_0, $\overline{BUSY}$, CAP, $\overline{ERROR}$, $\overline{PEREQ}$ y $\overline{PEACK}$ son terminales nuevas o adicionales que no aparecen en el microprocesador 8086. Las señales $\overline{BUSY}$, CAP, $\overline{ERROR}$, $\overline{PEREQ}$ y $\overline{PEACK}$ son utilizadas con la extensión del microprocesador, o coprocesador, del cual el 80387 es un ejemplo.

El bus de direcciones es de 24 bits para controlar los 16 MB de memoria física. Las unidades básicas del microprocesador se conservan, aunque su complejidad y rendimiento han aumentado considerablemente. (comparados con los anteriores microprocesadores de 8 bits)

Figura 37. ***Arquitectura del 80286***[53]



El conjunto de registros del i80286 se describe a continuación. Cabe anotar que casi todos los registros son iguales a los utilizados por los microprocesadores 8086/8088 y 80186/80188. Sin embargo se observa que el 80286 contiene un registro adicional de palabra de status de máquina (MSW), en donde el bit MSW controla si el 80286 está en el modo real o protegido.

El registro de señalizadores del 80286 se aprecia a continuación.

	NT	IO PL	OF	DF	IF	TF	SF	ZF		AF		PF		CF
15	14	13 -	12	11	10	9	8	7	6	5	4	3	2	1

[53] Extraído el 10 de Julio de 2009 desde http://www.geocities.com/nachoenweb/intel.html

Los principales señalizadores son:

Tabla 9. ***Bit en registro señalizador del 8085***[54]

bit número	nombre
11	desbordamiento
10	señalizador de dirección
9	habilita interrupción
8	señalizador de trampa
7	signo
6	cero
4	arrastre auxiliar
2	paridad
0	arrastre

Modos de direccionamiento

La CPU (i80286 o i80386) puede operar en dos modos. El primero es el modo de **direcciones reales,** en el cual puede acceder hasta 1 Mbyte de memoria, igual que los microprocesadores 8086 y 80186. El segundo es el modo de direcciones virtuales protegidas, que también se denomina **modo protegido.** En el modo protegido, la CPU formatea 1 Gbyte de direcciones virtuales por tarea, en un espacio de direcciones reales de 16 Mbyte. El modo protegido se denomina así ya que proporciona protección de memoria para aislar a los datos y programas de las tareas individuales. Esto permite la multitarea y una fácil conexión en red de los computadores. El modo protegido proporciona acceso a la gestión de memoria, paginación y capacidades de privilegio de CPU.

La memoria virtual es un espacio de memoria mayor (1GB para el 80286) en un espacio de memoria física mucho más pequeño (16 MB para el 80286), lo que permite que un sistema muy grande pueda ser ejecutado en sistemas de memoria física menor. Esto es realizado por medio de un intercambio de datos y programas entre el sistema de memoria de disco duro y la memoria física. El direccionamiento de un sistema de memoria de 1GB es realizado por los descriptores en el microprocesador 80286. Cada descriptor del 80286 describe un segmento de

[54] Téllez, 2007

memoria de 64KB. El i80286 acepta hasta 16K descriptores, lo que corresponde a 1GB de memoria.

Los descriptores describen al segmento de memoria en el modo protegido. El 80286 tiene descriptores que definen códigos, datos, segmentos de pila, interrupciones, procedimientos y tareas. Los accesos a descriptores son realizados cargando un registro de segmento con un selector en el modo protegido. El selector accede a un descriptor que describe un área de memoria.

LECCIÓN 4: MICROPROCESADORES DE 32 BITS.

La relevancia de los microcontroladores 80386 además de sus 32 bits, fue el prototipo que marco el inicio de la tercera generación de microprocesadores, se destaca la incorporación de una unidad de traslación de páginas, haciendo posible la implementación de Sistemas Operativos que emplean memoria virtual. A continuación se presentan las generalidades de estos dos microprocesadores claves en la evolución tecnológica de procesamiento de datos.

Características Generales

El **Intel 80386** (i386 o 386) es un microprocesador CISC de 32 bits con arquitectura x86. Durante su diseño se le llamó 'P3', debido a que era el prototipo de la tercera generación x86. El i386 fue empleado como la unidad central de proceso de muchos ordenadores personales desde mediados de los años 80 hasta principios de los 90.

El procesador i386 fue lanzado al mercado el 16 de octubre de 1985. El i386 añadió una arquitectura de 32 bits y una unidad de translación de páginas, lo que hizo mucho más sencillo implementar sistemas operativos que emplearan memoria virtual.

Figura 38. ***Intel 80386***[55]

[55] Extraído el 10 de Julio de 2009 desde http://www.cpu-world.com/CPUs/80386/index.html

Debido al alto grado de compatibilidad, la arquitectura del conjunto de procesadores compatibles con el i386 suele ser llamada arquitectura i386. El conjunto de instrucciones para dicha arquitectura se conoce actualmente como IA-32.

Después de que comenzara la producción del 80386, Intel introdujo el i80386SX. El i386SX fue diseñado como una versión económica del i386. Los i386SX, como todos los i386, tienen una arquitectura de 32 bits, pero se comunican con el exterior mediante un bus externo de 16 bits. Esto hace que sean el doble de lentos al acceder al exterior, pero por el contrario el diseño los circuitos auxiliares del microprocesador es mucho más sencillo.

El i386 original fue renombrado a Intel 80386DX para evitar la confusión. Desde un punto de vista comercial, el i386 fue importante debido a que fue el primer microprocesador disponible desde una única fuente.

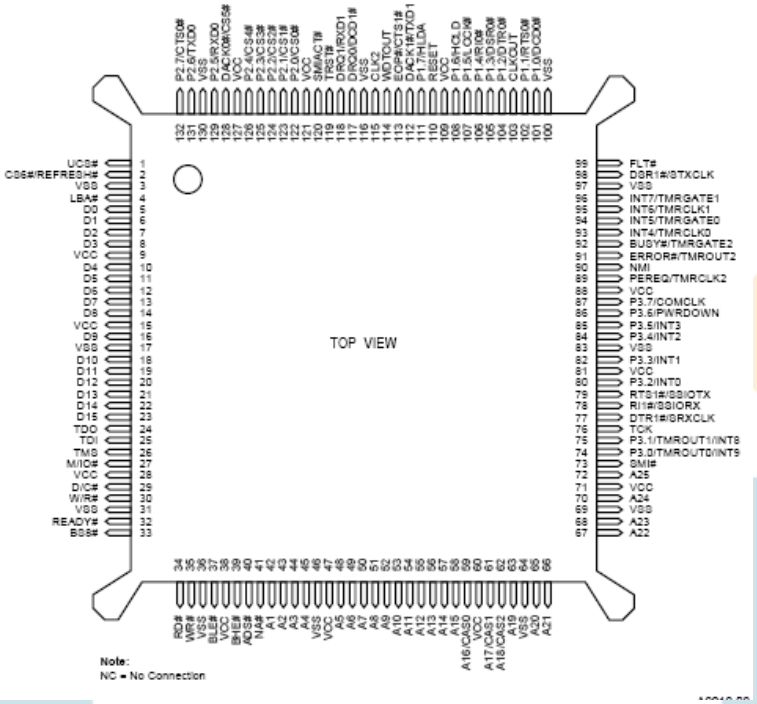
Funciones y diagrama de pines

Al igual que en las versiones anteriores de los microprocesadores de la familia Intel, el 80386 requiere solamente una fuente de alimentación de +5V. La corriente promedio de la fuente de alimentación es 550 mA para la versión de 25MHz, 500mA para la versión de 20MHz y 450mA para la versión de 16MHz. Este dispositivo contiene múltiples conexiones de V_{cc} y V_{ss} , las cuáles deben estar conectadas, respectivamente, a +5V y tierra para una correcta operación. Algunas terminales están etiquetadas como N/C y no deben ser conectadas.

Cada terminal de salida del 80386 es capaz de proporcionar 4mA (conexiones de dirección y datos) o 5mA (otras conexiones). Esto representa un incremento en la corriente de salida en comparación con los 2mA disponibles en los terminales de salida de los anteriores 8086, 8088 y 80286. Cada Terminal de entrada representa una carga pequeña, requiriendo aproximadamente 10 μA de corriente.

En algunos sistemas, exceptuando a los más pequeños, estos niveles de corriente requieren buffers de bus. La siguiente figura muestra la asignación de terminales del microprocesador 80386DX. El 80386DX está encapsulado en un empaque PGA (arreglo reticular de terminales) de 132 pines.

Figura 39. ***Pines del 80386***[56]



A continuación se presenta la función de cada grupo de terminales del 80386DX:

- **A31-32**: Las conexiones del bus de direcciones permiten acceder a cualquiera de las localidades de memoria de 1GX32 bits encontradas en el sistema de memoria de los 80386. Las señales A_0 y A_1 son codificadas en las señales de habilitación de bus de 32 bits. Debido a que el 80386SX posee un bus de datos de 16 bits en vez de uno de 32, las señales de selección de banco son sustituidas por $\overline{BHE}$ y $\overline{BLE}$. La señal $\overline{BHE}$ habilita la mitad superior del bus de datos; la señal $\overline{BLE}$ habilita la mitad inferior del bus de datos.

- **D31-D0**: Las conexiones del bus de datos transfieren datos entre el microprocesador y su sistema de memoria y E/S.
- $\overline{BE3} - \overline{BE0}$**:** Las señales de habilitación de banco seleccionan el acceso de un byte, una palabra, o doble palabra de datos. El microprocesador genera internamente estas señales a partir de los bits de dirección A_1 y A_0.

- **M/**$\overline{IO}$**:** El Terminal de memoria/ES selecciona un dispositivo de memoria cuando tiene un valor 1 lógico, o un dispositivo de E/S cuando tiene un valor de 0 lógico. Durante la operación de E/S, el bus de direcciones contiene una dirección de E/S de 16 bits en las conexiones de dirección A15-A2.

[56] Fuente (Téllez, 2007)

- **W/$\overline{R}$:** Escritura/lectura indica que el ciclo de bus actual es de escritura cuando contiene 1 lógico, o de lectura cuando contiene un 0 lógico.

- $\overline{ADS}$: El pulso de direccionamiento está activo cuando el 80386 haya emitido una dirección válida de memoria o de E/S. Esta señal es combinada con la señal W/$\overline{R}$ para generar las señales separadas de lectura y escritura existentes en los sistemas anteriores, basados en los microprocesadores 8086 y 80286.

- **RESET**: la restauración inicializa al 80386, que comienza a ejecutar el software a partir de la localidad de memoria FFFFFFF0H. El 80386 es restaurado en el modo real, y las 12 líneas de dirección más significativas permanecen con el valor de 1 lógico (FFFH), hasta que un salto lejano o llamada lejana es ejecutada. Esto establece la compatibilidad con los microprocesadores anteriores.

- **CLK2**: El reloj por 2 es alimentado por una señal de reloj que tiene una frecuencia igual al doble de la frecuencia de operación del 80386.

- $\overline{READY}$: Ready controla el número de estados de espera insertados en un ciclo de acceso para aumentar el tiempo de acceso a memoria.

- $\overline{LOCK}$: El Terminal de candado asume un valor 0 lógico siempre que una instrucción tenga el prefijo LOCK. Esto se utiliza con mayor frecuencia durante los accesos de DMA.

- **D/$\overline{C}$** : Datos/control indica que el bus de datos contiene datos para, o desde la memoria o E/S cuando tiene un valor 1 lógico. Si D/$\overline{C}$ es un 0 lógico, el microprocesador es detenido o ejecuta una aceptación de interrupción.

- $\overline{BS16}$: El tamaño de bus 16 selecciona ya sea un bus de datos de 32 bits ($\overline{BS16}$=1), o un bus de datos de 16bits ($\overline{BS16}$=0).

- $\overline{NA}$: Siguiente dirección ocasiona que el 80386 emita la dirección de la siguiente instrucción, o a los datos del ciclo de bus actual.

- **HOLD**: Detención solicita una acción de DMA.

- **HOLDA**: La solicitud de coprocesador pide al 80386 que ceda el control, y es una conexión directa al coprocesador aritmético 80387.

- $\overline{PEREQ}$: Reconocimiento de detención indica que el 80386 está actualmente en una condición de detención.

- $\overline{BUSY}$: Ocupado es una entrada utilizada por la instrucción WAIT o FWAIT que espera a que el coprocesador se desocupe. Esta es también una conexión directa entre el 80387 y el 80386.

- $\overline{ERROR}$: indica al microprocesador que el coprocesador ha detectado un error.

- **INTR:** es utilizada por los circuitos externos para solicitar una interrupción.

- **NMI**: Una interrupción no enmascarable es solicitada igual que en las versiones anteriores del microprocesador.

Arquitectura interna

La siguiente figura presenta el diagrama de bloques de la estructura interna del microprocesador 80386DX. El i80386DX direcciona 4GB de memoria por medio de su bus de direcciones de 32 bits y tiene un bus de datos de 32 bits. El 80386SX es más parecido al 80286, y direcciona 16MB de memoria con su bus de direcciones de 24 bits y cuenta con un bus de datos de 16 bits. El 80386SX es encontrado en muchas aplicaciones que no requieren una versión completa del bus de 32 bits.

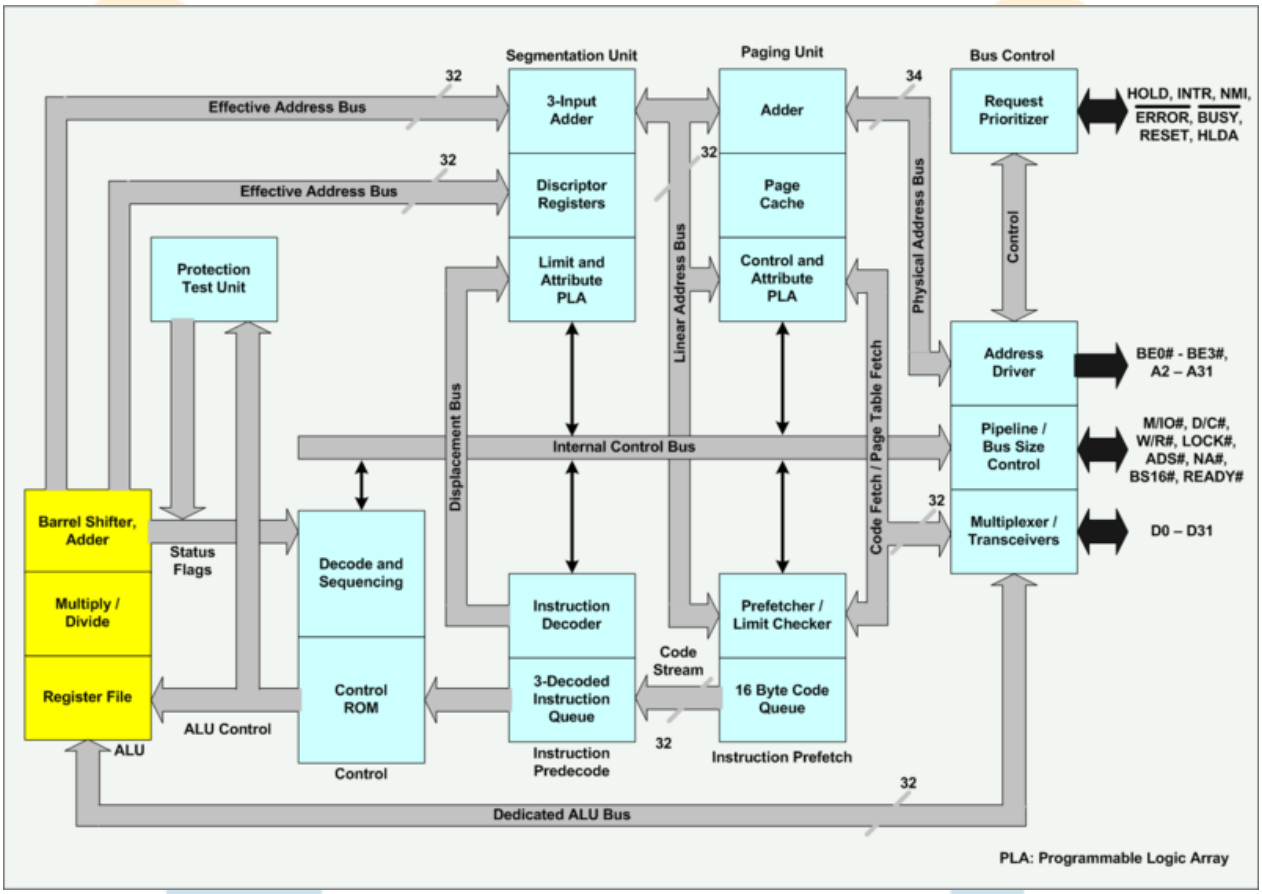
El sistema de memoria

El sistema de memoria física del 80386DX tiene una capacidad de 4GB y es diseccionado como tal. Si se recurre a direccionamiento virtual, la unidad de administración de memoria y los descriptores mapean 64 TB en los 4 GB de espacio físico.

La memoria está dividida en cuatro bancos de memoria de 8 bits, y cada uno con una capacidad hasta 1 GB de memoria. Esta organización de memoria de 32 bits permite el acceso directo a bytes, palabras o dobles palabras de datos de memoria. El 80386DX transfiere un número de hasta 32 bits en un solo ciclo de memoria, mientras el 8088 requiere cuatro ciclos para completar la misma transferencia, y los 80286 y el 80386SX requieren dos ciclos. Actualmente, el ancho de datos es importante, especialmente para el manejo de números de punto flotante con precisión sencilla de 32 bits. Normalmente, el software de alto nivel

utiliza números de punto flotante para el almacenamiento de datos, así que las localidades de memoria de 32 bits aceleran la ejecución de software de alto nivel, sacando así el máximo provecho de esta memoria más ancha.

Figura 40. ***Arquitectura del 80386***[57]



La CPU del i80386 contiene registros en las siguientes categorías:

- Registros de propósito general.
- Registros de segmento.
- Puntero de instrucción y señalizadores.
- Registros de control.
- Registros de direcciones del sistema.
- Registros de depuración.
- Registros de prueba.

[57] Extraído el 10 de Julio de 2009 desde http://commons.wikimedia.org/wiki/File:80386DX_arch.png

Los registros de **propósito general** de 32 bits, los registros selectores de segmento de 16 bits y el puntero de instrucciones y señalizadores se muestran en la siguiente figura.

Reservado	VM	RF	0	NT	IO PL	O F	DF	IF	TF	SF	ZF	0	AF	0	CF

Tabla 10. ***Bit en registro señalizador del 80386***[58]

bit	nombre
VM	modo virtual
RF	señalizador de resumen
NT	señalizador de tareas anidadas
IOPL	nivel de privilegio de E/S
OF	desbordamiento
DF	señalizador de dirección
IF	habilitación de interrupción
TF	señalizador de trampa
SF	señalizador de signo
ZF	señalizador de cero
AF	arrastre auxiliar
PF	señalizador de paridad
CF	señalizador de arrastre

[58] Téllez, 2007

Los otros tipos de registro (control, direcciones del sistema, depuración y prueba) son más usados por el software del sistema. Por ejemplo, el registro MSW del 80286 es parte del registro de control de la CPU del 80386.

Pipelines y cachés. La memoria caché es una memoria buffer que permite que el 80386 funcione más eficientemente con velocidades de DRAM más bajas. Un ***pipeline*** es una forma especial de manejar accesos a memoria con tiempos de acceso de 50 ns o menos para operar a máxima velocidad. De hecho actualmente las DRAMs más rápidas tienen un tiempo de acceso de 60 ns o mayor. Esto significa que debe encontrarse alguna técnica para establecer una interfaz con estos dispositivos de memoria, que son más lentos a lo requerido por el microprocesador. Tres técnicas están disponibles: memoria entrelazada, uso de memoria caché y un pipeline.

Cuando una instrucción es obtenida de la memoria, el microprocesador dispone frecuentemente de un tiempo adicional antes de leer la siguiente instrucción. Durante este tiempo adicional, la dirección de la siguiente instrucción es emitida por el bus de direcciones en forma anticipada. Este tiempo adicional (un período de reloj) es utilizado para emitir un mayor tiempo de acceso a componentes de memoria más lentos. En términos generales, el ***pipeline*** es una característica que reduce costos y que disminuye el tiempo de acceso requerido por el sistema de memoria en sistemas de baja velocidad.

Una **caché** es un sistema de memoria de alta velocidad ubicado entre el microprocesador y el sistema de memoria DRAM. Los dispositivos de memoria caché son normalmente componentes de memoria RAM estática, con tiempos de acceso menores a 25 ns. La capacidad de la memoria caché está determinada más por la aplicación, que por microprocesador. Si un programa es pequeño y hace referencia a pocos datos de memoria, una caché pequeña es benéfica. Si un programa es grande y hace referencia a bloques grandes de memoria, es recomendable que el tamaño de caché sea lo más grande posible.

Un sistema de **memoria entrelazada** es otro método para mejorar la velocidad del sistema. Su única desventaja es que debido a su estructura requiere considerablemente más memoria. Los sistemas de memoria entrelazada existen en algunos sistemas, para que puedan alargarse los tiempos de acceso sin necesidad de tiempos de espera; además requiere dos o más conjuntos completos de buses de direcciones y un controlador que proporcione las direcciones a cada bus.

Modos de direccionamiento

El microprocesador 80386 también tiene dos modos de operación: el modo de direcciones reales (modo real) y el modo de direcciones virtuales protegidas (modo protegido). En el modo real, el 80386 opera como un microprocesador 8086 muy rápido. En el modo protegido proporciona acceso a la gestión de memoria, paginación y capacidades de privilegio de la CPU.

Para asegurar un alto rendimiento, la interfaz del bus del 80386 ofrece encauzamiento de direcciones (pipeling), bus dinámico de datos y señales directas de habilitación de bytes para cada byte del bus de datos. El microprocesador puede ejecutar de esta forma unas tres millones de instrucciones por segundo.

Además soporta diferentes tipos de datos. Entre éstos se encuentran: bit, campo de bits, cadena de bits, byte, entero, entero grande, cuádruple palabra con signo, cadena, BCD y BCD empaquetado. Cuando el 80386 se conecta al coprocesador numérico 80387, se soportan los números con signo en punto flotante.

Repertorio de instrucciones

A continuación se muestra una selección del conjunto de instrucciones de los microprocesadores 80286 y 80386, el cual contiene el código de operación nemotécnico y una breve descripción del propósito de la instrucción.

ARPL (ajusta el nivel de privilegio solicitado)

Formato: ARPL reg, reg.

0	**1**	**1**	**0**	**0**	**0**	**1**	**1**
o	**o**	**r**	**r**	**r**	**m**	**m**	**m**

	80286	**80386**
Ciclos	10	20
señalizadores	Z	

BOUND (verifica arreglo contra límite)

Formato: BOUND reg, mem.

0	1	1	0	0	0	1	0
o	o	r	r	r	m	m	m

	80286	80386
Ciclos	13	10
señalizadores	ninguno	

BSF (exploración de bit en sentido directo)

Formato: BSF reg, reg.

0	0	0	0	1	1	1	1
1	0	1	1	1	1	0	0

	80286	80386
Ciclos	-	10+3n
señalizadores	Z	

BT (verificación de bit)

Formato: BT reg, inm8.

0	0	0	0	1	1	1	1
1	0	1	1	1	0	1	0

	80286	80386
Ciclos	-	3
señalizadores	C	

CALL (llamar procedimiento(subrutina))

Formato: Call etiqueta.

1	1	1	0	1	0	0	0

	80286	80386
Ciclos	7	3
señalizadores	ninguno	

CMP (compara)

Formato: CMP reg, reg.

0	0	1	1	1	0	d	w

	80286	80386
Ciclos	2	2
señalizadores	S, Z, A, P, C.	

CWDE (convierte palabra en doble palabra extendida)

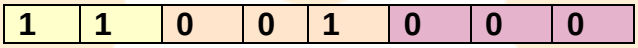
Formato: CWDE

1	0	0	1	1	0	0	0

	80286	80386
Ciclos	-	3
señalizadores	ninguno	

ENTER (crea un marco de pila)

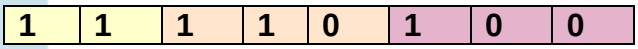
Formato: ENTER inm, 0.



	80286	80386
Ciclos	11	10
señalizadores	ninguno	

HLT (para)

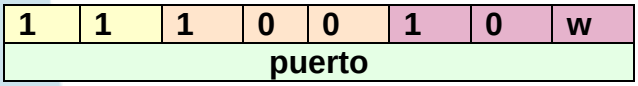
Formato: HLT



	80286	80386
Ciclos	2	5
señalizadores	ninguno	

IN (Lee dato de puerto)

Formato: IN acc, pt.



	80286	80386
Ciclos	5	12
señalizadores	ninguno	

JMP (salta)

Formato: JMP etiqueta

1	**1**	**1**	**0**	**1**	**0**	**1**	**1**

	80286	**80386**
Ciclos	7	7
señalizadores	ninguno	

LFS (carga apuntador lejano a FS y registro)

Formato: LFS reg, mem.

0	**0**	**0**	**0**	**1**	**1**	**1**	**1**
1	**0**	**1**	**1**	**0**	**1**	**0**	**0**
dato							

	80286	**80386**
Ciclos	-	7
señalizadores	ninguno	

LOOP (repite hasta que CX=0 o ECX=0)

Formato: LOOP etiqueta.

1	**1**	**1**	**0**	**0**	**0**	**1**	**0**
dato							

	80286	**80386**
Ciclos	8/4	11
señalizadores	ninguno	

MOVS (mueve dato de cadena)

Formato: MOVSB

1	0	1	0	0	1	0	w

	80286	80386
Ciclos	5	7
señalizadores	ninguno	

OUT (escribe dato a puerto)

Formato: OUT pt, acc.

1	1	1	0	0	1	1	w
puerto							

	80286	80386
Ciclos	3	10
señalizadores	ninguno	

POP (recupera dato de pila)

Formato: POP reg

0	1	0	1	1	r	r	r

	80286	80386
Ciclos	5	4
señalizadores	ninguno	

RET (regreso de procedimiento)

Formato: RET

1	**1**	**0**	**0**	**0**	**0**	**1**	**1**

	80286	**80386**
Ciclos	11	10
señalizadores	ninguno	

SET condition (establece condicionalmente)

Formato: SETcnd, reg8.

0	**0**	**0**	**0**	**1**	**1**	**1**	**1**
1	**0**	**0**	**1**	**c**	**c**	**c**	**c**
o	**o**	**0**	**0**	**0**	**m**	**m**	**m**

	80286	**80386**
Ciclos	-	4
señalizadores	ninguno	

SMSW (Almacena palabra de estado de máquina (80286))

Formato: SMSW reg.

0	**0**	**0**	**0**	**1**	**1**	**1**	**1**
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**
o	**o**	**1**	**0**	**0**	**m**	**m**	**m**

	80286	**80386**
Ciclos	2	10
señalizadores	ninguno	

TEST (verifica operandos (comparación lógica))

Formato: TEST reg,reg.

0	**1**	**0**	**1**	**1**	**r**	**r**	**r**

	80286	**80386**
Ciclos	2	2
señalizadores	C=0,P, Z, S	

WAIT (espera al coprocesador)

Formato: WAIT

1	**0**	**0**	**1**	**1**	**0**	**1**	**1**

	80286	**80386**
Ciclos	3	6
señalizadores	ninguno	

LECCIÓN 5: MICROPROCESADORES DE 64 BITS.

Los microprocesadores de 64 bits constituyen un modelo, una arquitectura y el tamaño de los datos o direcciones de memoria, que ahora se compone de 8 bytes, la tecnología de 64 bits no es nueva en el sentido de su aplicación practica, esta tecnología esta presente desde 1960 en supercomputadoras, también en servidores y estaciones de trabajo basadas en arquitectura RISC implementados en los 90´s.

La introducción de esta tecnología en computadores personales, se hace desde 2003 con arquitecturas x86-64 y procesadores PowerPC G5. El término 64 bits puede referirse a la CPU, al tamaño de los buses o al tamaño de las instrucciones,

pero realmente se habla de tecnología de 64 aplicada a un equipo de cómputo si el equipo cumple con estos tres parámetros:

- La CPU internamente debe ser de 64 bits.
- Los buses deben tener un ancho de 64 bits.
- Las instrucciones llegan a tamaños de 64 bits.

Ventajas y desventajas de la arquitectura de 64 bits

Ventajas, La más notoria es el aumento de la capacidad de memoria tanto RAM como virtual con respecto a la tecnología de 32 bits ya que estos últimos tienen una limitante de 4 GB máximo de memoria.

La cantidad máxima de memoria soportada por las versiones de 64 bits de Windows es:

- Windows XP professional 64 bits: 16 GB de RAM.
- Windows Vista Home Basic 64 bits: 8GB de RAM.
- Windows Vista Home Premium 64 bits: 16 GB de RAM.
- Windows Vista (otras versiones) 64 bits: 128 GB de RAM.

Otra ventaja de los sistemas de 64 es su mayor capacidad de procesamiento, estabilidad y seguridad de la información.

Desventajas, Los sistemas operativos de 64 bits, son de uso mas profesional que domestico, tiene inconvenientes con compatibilidad de software:

- No son compatibles con programas o software de 16 bits
- Inclusive algunos antivirus y aplicaciones de 32 bits presentan incompatibilidad.
- Existen problemas de Drivers para tecnología 64 bits.

Arquitectura

Como se ha expuesto la tecnología de 64 bits se ha extendido desde los años 60’s hasta la actualidad, es muy amplia y compleja las arquitecturas que incorporan tecnología de 64 bits por lo que el estudio requiere una profunda conceptualización en los términos tanto de hardware como de software

involucrados, igualmente en la microelectrónica empleada por lo que se invita a los estudiantes a profundizar en esta tecnología comenzando con la literatura que se encuentra en línea y en los fabricantes de estos productos, la siguiente es una condensación de las principales arquitecturas.

Arquitectura DEC Alpha*: la organización de sus registros es de uso general con una arquitectura que se puede encuadrar como de registro-registro. Esto hace que la memoria de sus instrucciones opere sobre los registros, haciendo uso de la memoria RAM solo para instrucciones de carga y almacenamiento. La razón es que se intenta minimizar los accesos a memoria, puesto que suponen el cuello de botella para los procesadores actuales, la longitud de palabra de los registros es de 64 bits, ya sea desde el PC (Contador de Programa), pasando por los registros de enteros, punto flotante, etc.… el primer procesador que hizo gala de la tecnología Alpha fue el 21064*

Arquitectura IA-64*: Línea de procesadores Itanium e Itanium2. Representan el diseño de producto más complejo del mundo con más de 1700 millones de transistores. Esto permite obtener sólidas capacidades de virtualización, mejorar la confiabilidad y niveles de rendimiento líderes del mercado.*

La arquitectura AMD64*: El conjunto de instrucciones del AMD 86x-64 (renombrada posteriormente como AMD64) es una extensión directa de la arquitectura del x86 a una arquitectura de 64 bits, motivado por el hecho de que los 4GB de memoria que son direccionables directamente por una CPU de 32 bits ya no es suficiente para todas las aplicaciones. El primer procesador con soporte para este conjunto de instrucciones fue el Opteron lanzado en 2003. Posteriormente ha sido implementado en múltiples variantes del Athlon 64 y el Pentium 4 de Intel, en este último caso bajo una versión de Intel llamada EM64T.*

La arquitectura SPARC*: Es la primera arquitectura RISC abierta y como tal las especificaciones de diseño son públicas, así otros fabricantes de microprocesadores pueden desarrollar su propio diseño. Una de las ideas innovadoras de esta arquitectura es la ventana de registros que permite hacer fácilmente compiladores de alto rendimiento y una significativa reducción de memoria en las instrucciones load/restore en relación con otras arquitecturas RISC. Las ventajas se aprecian sobre todo en programas grandes.*

La CPU SPARC esta compuesta de una unidad entera, UI (Integer Unit) que procesa la ejecución básica y una FPU (Floating-Point Unit) que ejecuta las operaciones y cálculos reales. La UI y la FPU pueden o no estar integradas en el mismo Chip.

Aunque no es una parte formal de la arquitectura, las computadoras basadas en sistema SPARC de Sun Microsystem tienen una unidad de manejo de memoria (MMU) y una gran cache de direcciones virtuales (para las instrucciones y los

datos) que están dispuestos periféricamente sobre un bus de datos y direcciones de 32 bits.

La arquitectura POWER: Es usada en servidores IBM, pero sin embargo hay muchos microprocesadores que son derivados o variantes de este que se encuentra en gran variedad de equipos que van desde computadores para automóviles hasta consolas de videojuegos, su nombre proviene de "**P**erformance **O**ptimization **W**ith **E**nhanced **R**ISC".

La arquitectura PA-RISC: Una característica interesante de PA-RISC es que la mayoría de sus microprocesadores no tienen cache L2. En su lugar se implementa un cache L1 mayor, formada por chips separados conectados al microprocesador a través de un bus (actualmente esta integrada en el propio chip). Solo el modelo PA-7300LC tiene cache L2. Otra innovación de esta arquitectura fue la adición de un repertorio de instrucción multimedia (SIMD) conocido como MAX e introducido por primera vez en el 7100LC.

También se suelen referir a ella como la arquitectura HP/PA, Hewlett Packard Precision Architecture. PA se desarrolla en Palo Alto, donde se encuentra la central de HP.

CAPÍTULO 3: ENSAMBLADOR (ASSEMBLER)

LECCIÓN 1: FUNDAMENTOS DE PROGRAMACIÓN.

Para mantener un control sobre los periféricos la CPU utiliza pequeños periodos de tiempo en los cuales realiza una serie de acciones cuyo propósito es cumplir tareas que han sido organizadas de forma secuencial, estas tareas son las instrucciones y la forma secuencial es el programa, este concepto se aplica a cualquier sistema basado en microprocesadores o Microcontroladores.

Programas

Los programas contienen una secuencia de instrucciones, estas instrucciones debe poder ser almacenada en algún lugar del sistema, se habla de “memoria de programa” como el lugar donde se almacenaran las instrucciones cada una de ellas dispuesta en uno o varios contenedores denominados registros, compuestos a su vez por celdas de almacenamiento, cada una de ellas con capacidad de almacenar un “**bit**” que puede presentar uno de dos estados estables posibles “**cero**” o “**uno**”.

Las instrucciones y datos del programa al ejecutarse en la CPU generan datos con significado dentro del contexto de la solución del problema, estas instrucciones y datos se expresan en conjuntos de “**bits**”, como los denominados “**Nibbles**” (4 bits), “**Bytes**” (8 bits), “**W**-palabra” (16 bits), “**DW**- doble palabra” (32 bits) y “**QW-** cuádruple palabra” (64 bits).

El programa es el centro nervioso de los sistemas electrónicos modernos, en los que es igualmente importante el desarrollo de componentes, su interconexión y los programas orientados a la solución de problemas utilizando dichos componentes.

Los algoritmos

En términos generales los algoritmos son el conjunto finito de instrucciones o la serie de pasos que deben seguirse para solucionar un problema, en la programación de microprocesadores o Microcontroladores, se debe cumplir que cada paso o instrucción del algoritmo debe estar bien definido sin ambigüedades, la claridad es vital para que cualquier persona pueda entenderlo y realizarlo. Aunque hay varias formas de solucionar un problema, cada paso o instrucción debe regirse por la claridad y simpleza de su acción y la efectividad en conjunto para llegar a una solución.

En la programación de microprocesadores un algoritmo debe tener un principio y un final, se espera que la entrada sea procesada y produzca una salida. En los sistemas basados en microcontroladores los algoritmos pueden seguir trayectorias cerradas bien definidas haciéndolos parecer que no tuvieran un final.

Programador

El programador es el sujeto que interfiere en el proceso de creación de programas teniendo como bases fundamentales la lógica, los algoritmos y el conocimiento del lenguaje con el que se pretende desarrollar la solución. Entones el objetivo del programador es el de entender el problema a solucionar, diseñar el algoritmo que lo resuelve, escribir la secuencia de instrucciones requeridas en el lenguaje apropiado, almacenar la secuencia de las instrucciones en la memoria interna del dispositivo con capacidad de procesamiento automático para finalmente ejecutarlo.

Lenguaje maquina

Los microprocesadores y los circuitos digitales que los conforman y complementan responden a un conjunto de operaciones que se denomina programa maquina, el cual contiene un conjunto de instrucciones o set de instrucciones que realizan operaciones sobre un bit o los bits que conforman las palabras, las instrucciones se expresan también como conjuntos de bits ya que estos conforman el lenguaje propio de la maquina.

Se entiende que el lenguaje máquina solo se escribe con “ceros” y “unos”, llamado lenguaje binario, por lo que la escritura de los datos se hace compleja e incómoda. Este inconveniente impulso la idea de representar la secuencias de “unos” y “ceros” con símbolos que fueran más comprensibles y fáciles de manipular, utilizando equivalencias en hexadecimal u octal.

Un ejemplo de código maquina y una forma compacta de expresarlo con código hexadecimal se muestra a continuación:

Tabla 11. *Código maquina*[59]

DIR.SEGMENTO	DIR. MEM	COD.BINARIO	COD.HEXA
17A3	0100	1011010000001000	B408
17A3	0102	100000001100010000000011	80C403
17A3	0105	100000001110110000000100	80EC04
17A3	0108	1100110100100000	CD20

Como se observa aunque se disminuye el campo de símbolos, sigue existiendo la complejidad en su interpretación por lo que se desarrolla otra técnica para identificar las instrucciones permitiendo una más fácil interpretación.

Ventaja del lenguaje maquina: Como principal ventaja esta la velocidad de ejecución que es superior a la de cualquier otro lenguaje, esto se debe a que el programa es cargado directamente en memoria sin necesidad de una “traducción” previa.

Desventajas del lenguaje maquina:

- Una de las desventajas se presenta del lado del programador porque es muy difícil recordar la secuencia o cadena de caracteres binario que identifican una instrucción y no se puede identificar una relación con la operación que realiza.

[59] CEKIT, 2002

- Otra desventaja se presenta por las diferencia entre el set de instrucciones entre uno y otro procesador, incluso de la misma fábrica.
- Generalmente un programa debe ser ingresado a la maquina, lo que genera lentitud y dificultad en la codificación, lo que lo hace susceptible a errores, a lo anterior se suma los inconvenientes en el proceso de depuración de errores y puesta a punto del programa.

Como se observa las desventajas superan ampliamente las ventajas por lo que el lenguaje maquina no es recomendable para desarrollar aplicaciones.

Lenguaje ensamblador

Buscando solventar los inconvenientes de la programación con solo símbolos binarios, se desarrollan intérpretes y fichas mnemotécnicas que son el origen del lenguaje ensamblador, hoy las fichas mnemotécnicas se han sustituido por símbolos o compresiones de palabras que provienen de la síntesis significativa de la instrucción u operación.

El lenguaje ensamblador es considerado un lenguaje de bajo nivel, que es más fácil de utilizar que su predecesor el lenguaje maquina, pero mantiene la dependencia con respecto al tipo de procesador.

Las instrucciones de lenguajes maquina son simbolizadas por mnemotécnicos, los cuales emplean términos que permiten una fácil asociación con la operación básica que realiza dicha instrucción.

Tabla 12. ***Código maquina y nemotécnico***[60]

DIR.SEGMENTO	DIR. MEM	COD.BINARIO	COD.HEXA	MNEMOTECNICO
17A3	0100	1011010000001000	B408	MOV AH, 08
17A3	0102	100000001100010000000011	80C403	ADD AH, 03
17A3	0105	100000001110110000000100	80EC04	SUB AH, 04
17A3	0108	1100110100100000	CD20	INT 20

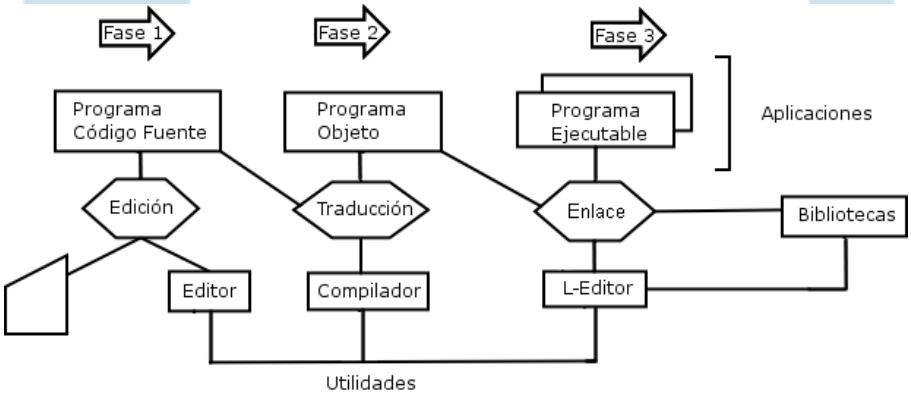
[60] CEKIT, 2002

Programa en ensamblador

Los programas escritos en lenguaje ensamblador deben pasar por una fase donde se "traducen" a lenguaje máquina o lenguaje binario para que el computador o microcomputador lo ejecute directamente, esta tarea la realiza un software especializado llamado "programa ensamblador". El proceso de crear un programa se divide en tres partes:

1. Editar el programa, en su formato fuente
2. Ensamblarlo y enlazarlo
3. Realizar las correcciones y depuraciones necesarias, el ciclo se puede cerrar en esta parte para comenzar de nuevo en la primera.

Figura 41. ***Fases de realización de un programa***[61]



Código fuente:

Es el conjunto de líneas de texto que representan las instrucciones que debe seguir la CPU para ejecutar el programa, este código describe completamente el funcionamiento del programa, aunque no es directamente ejecutable teniendo que pasar por un compilador para ser traducido a lenguaje maquina o código objeto.

Código objeto:

El código objeto es el resultado de la compilación del código fuente, consiste en lenguaje maquina que puede estar distribuido en varios archivos correspondientes

[61] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Fuente.png

a varios códigos fuente que son enlazados con un programa "enlazador" o "**linker**" para producir el programa ejecutable.

Código ejecutable:

Es común que se confunda el código objeto con el código ejecutable, la diferencia radica en que el código objeto presenta una estructura comprendida como símbolos, mientras el código ejecutable es un empaquetado listo para ser ejecutado en una computadora, generalmente tiene la extensión ".COM" o ".EXE", es el resultado de la compilación y el enlazamiento "**linker**" de los códigos objetos,

Compilador e intérprete

Los intérpretes son programas encargados de procesar y traducir cada instrucción o sentencia de un programa que esta escrito en lenguaje de alto nivel a código maquina para después ejecutarla, cada instrucción se ejecuta después de traducir y comprobar que no posee errores en sintaxis. Las tareas de un intérprete son leer, examinar, traducir y ejecutar cada instrucción del programa fuente, los intérpretes incorporan un editor por medio del cual se hace la edición, interpretación y ejecución del programa, el principal inconveniente de los intérpretes surge con la lentitud ocasionada en sucesivas ejecuciones del programa. Como ejemplo de intérprete tenemos el Debugger.

Los compiladores también se encargan de traducir el programa fuente en código maquina, pero a diferencia de los interpretes la traducción se realiza sobre la totalidad del programa fuente que después de convertirse a código maquina es ejecutado, las ejecuciones sucesivas no necesitan generar nuevamente el código maquina lo que acelera el tiempo de ejecución. Uno de los compiladores mas conocidos en la programación de microcontroladores es el MASM o versiones similares como FASM, TASM y NASM, se concluye entonces que los compiladores tienen lo mejor de los ensambladores y los interpretes.

Existen dos formas de crear programas:

1. La primera consiste en usar un ensamblador profesional, como MASM (Macro Assembler, de Microsoft), u otras versiones como FASM, TASM y NASM, pero existen problemas de compatibilidad con Windows Vista.

2. Utilizar Debugger como intérprete y punto de partida en la programación de microcontroladores implementando instrucciones y pequeños programas que prueben la potencia del código.
3. Utilizar el emulador SimuProc. Que emula un procesador hipotético compatible x86, con este emulador y sus especificaciones se adquiere habilidades y se comprende con más detenimiento el funcionamiento de cada instrucción y las microoperaciones que involucran.

LECCIÓN 2: DIAGRAMA DE FLUJO O BLOQUES.

Como se ha establecido en la lección anterior una de las dificultades de utilizar lenguaje ensamblador es el hecho que cada microprocesador o microcontrolador posee su propio lenguaje ensamblador y programa(s) compilador(es). La experiencia ha demostrado que el aprendizaje de los fundamentos básicos en la programación de determinado microprocesador o microcontrolador y la habilidad en desarrollar la lógica de los algoritmos que componen la aplicación o solucionan el problema en cualquier lenguaje, es suficiente para que el programador transfiera estas habilidades a un sistema diferente.

Pseudocódigo

Es una mezcla de lenguaje de programación y lenguaje nativo (español), se emplea en la programación para realizar el diseño del algoritmo, consiste en una estructura narrativa respecto a los pasos a seguir en un algoritmo para solucionar un problema. El pseudocódigo no sigue la rigidez de la sintaxis de cada instrucción, tampoco la fluidez del lenguaje humano, es un lenguaje intermedio que permite codificar de forma sencilla al lenguaje de programación utilizado, este lenguaje se emplea en las primeras etapas del análisis o diseño del software, en lo que se conoce como diseño del algoritmo.

Para realizar un pseudocódigo se debe considerar partir de una serie de pasos simples que lleven a la solución del problema estos pasos se convierten en secuencias simplificadas utilizando pocas palabras o frases sencillas describiendo tareas simples, observe el ejemplo de Pseudocódigo: para poder hallar el área de un cuadrado.

```
INICIA:
        LADO, AREA: ENTERO
        PONER EN BLANCO LA PANTALLA
        ESCRIBIR: "DIGITE LA LONGITUD DEL LADO"
        LEER: LADO
        AREA = LADO * LADO
        ESCRIBIR: "AREA DEL CUADRADO" = AREA
TERMINA
```

El paso siguiente depende de la complejidad o entendimiento que el programador tenga del algoritmo pudiendo utilizar los dos o uno de los siguientes pasos:

1. La sencillez de algunos programas pueden solo requerir compactar el pseudocódigo convirtiendo cada estructura simple en una instrucción equivalente en lenguaje ensamblador.
2. Puede asociar el pseudocódigo a un diagrama de bloques o un diagrama de flujo que permitan observar la estructura general del algoritmo en un lenguaje aproximado al utilizado para programar en este caso al ensamblador con el set de instrucciones propio del dispositivo a utilizar.

Ventajas del Pseudocódigo:

- Mejora la comprensión del problema y dilucida la posible solución.
- Facilita la solución y desarrollo del problema.
- Representa en forma sencilla las operaciones repetitivas o cíclicas.
- Se obtiene la estructura general del programa.
- En el proceso de aprendizaje facilita que el estudiante este mas cerca del paso siguiente, (la realización del diagrama de flujo o bloques).

Diagrama de bloques

Los diagramas de bloque son representaciones graficas del funcionamiento de un sistema que facilita a los programadores visualizar las características de los componentes que hacen parte de la solución del problema. Consisten en una serie de bloques que puede estar conectados por flechas o números que indiquen el orden del proceso, dentro de cada bloque se encuentra un componente simple del pseudocódigo que mas adelante se convertirá en la instrucción en ensamblador.

```
INICIA:
        LADO, AREA: ENTERO
        PONER EN BLANCO LA PANTALLA
        ESCRIBIR: "DIGITE LA LONGITUD DEL LADO"
        LEER: LADO
        AREA = LADO * LADO
        ESCRIBIR: "AREA DEL CUADRADO" = AREA
TERMINA
```

La representación de un algoritmo utilizando diagramas de bloques puede no ser lo suficientemente clara para representar el algoritmo de programación, en la mayoría de casos se elije el diagrama de flujo el cual es mas compatible y se adecua a las necesidades y requerimiento de representación de las operaciones simples que realiza el microprocesador o microcontrolador.

Figura 42. ***Fases de realización de un programa***[62]

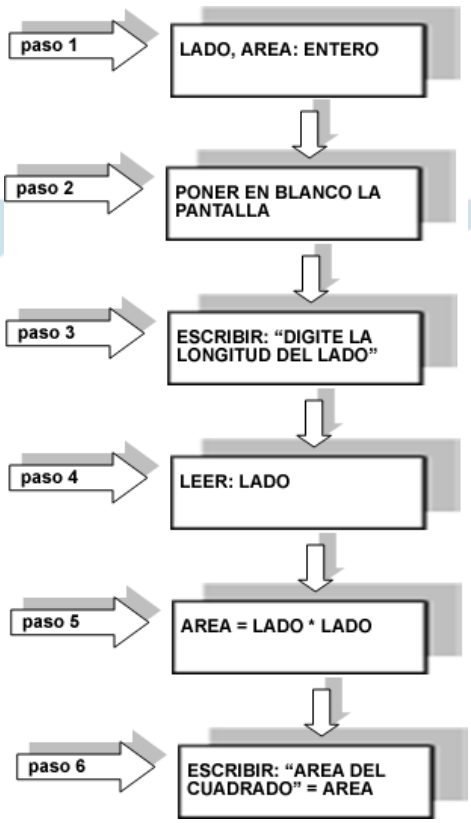


Diagrama de Flujo

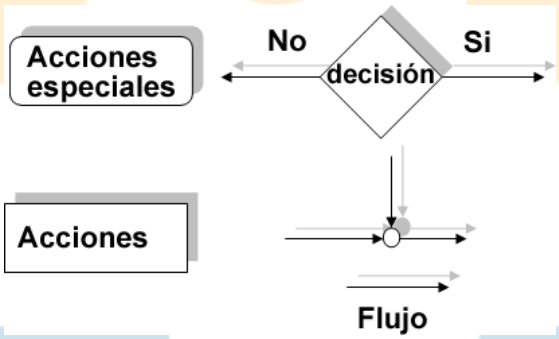
El diagrama de flujo representa más detalladamente el algoritmo de un programa en lenguaje ensamblador y en general de programación de computadores, consiste en una serie de símbolos gráficos que representan por si solos acciones o decisiones unidos por flechas que determinan el flujo del programa, los símbolos de acciones o decisiones contienen en su interior un paso de la secuencia generada por el pseudocódigo.

El análisis del problema puede conducir inmediatamente al desarrollo de un diagrama de flujo que integra en un solo paso el pseudocódigo y el diagrama

[62] CEKIT, 2002

mismo, aunque los diagramas de flujo contienen símbolos diversos para representar las acciones, se utilizaran básicamente cuatro de ellos

Figura 43. ***Símbolos más comunes para diagramas de flujo*** [63]



- **Acciones especiales**, conformadas por.
  - "inicio" que indica el comienzo del programa.
  - "fin" que indica que el programa ha terminado.
- **Acciones**, son las instrucciones que involucran movimiento de datos, operaciones aritméticas, lógicas, comparaciones sin decisión, llamado o saltos.
- **Decisiones**, son las instrucciones que involucran comparaciones con saltos, es decir, generan bifurcaciones (caminos) dentro del flujo del programa
- **Flechas**, las cuales simbolizan el flujo del programa.

Los caminos que describen los diagramas de flujo deben conducir a una solución, solo existe un único "inicio" y "final" de programa, aunque en los sistemas con microcontroladores en su funcionamiento pareciera que nunca tienen fin y siguen ciclos estructurados que mantienen el flujo del programa. Se debe evitar sumideros infinitos o espacios que sin entradas retornan alguna salida, es recomendable etiquetar procesos que posteriormente representan las subrutinas del programa, con esto se evitan errores en el flujo del programa final.

Ventajas de los diagramas de flujo

- Son una herramienta flexible utilizada ampliamente en el aprendizaje y capacitación en programación de sistemas con microprocesadores o microcontroladores.

[63] CEKIT, 2002

- Ayuda en la comprensión mental del programa al mostrarlo como un dibujo que es más susceptible de ser relacionado por el cerebro humano.
- Un diagrama de flujo bien diseñado sustituye líneas de texto extensas.
- Mediante un diagrama de flujo se pueden establecer redundancias cíclicas, bifurcaciones, flujos inconsistentes y errores.
- Permiten un análisis que pueden identificar problemas y mejorar el algoritmo.

Desarrollo de un diagrama de flujo

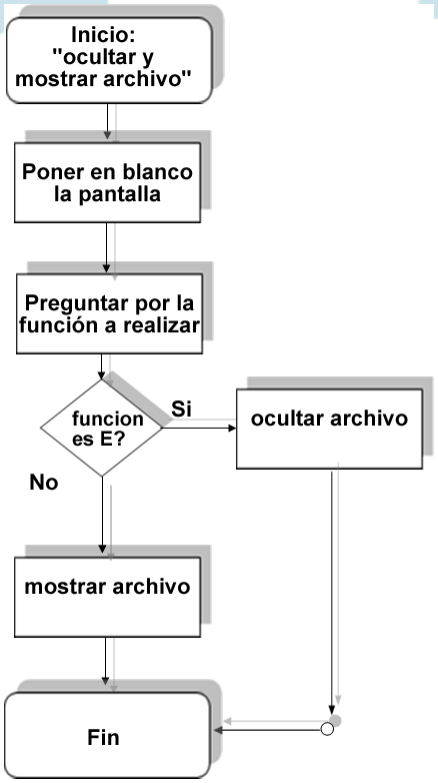
Para desarrollar un diagrama de flujo es necesario tener claridad en la lógica digital y el proceso de desarrollo de algoritmos, en general un diagrama de flujo exige ciertas acciones previas:

- Se debe tener el algoritmo desarrollado para la solución del problema, este debe especificar, las entradas (definir variables y constantes), el proceso (acciones, decisiones o bifurcaciones) y el resultado (almacenado en memoria, acción sobre periféricos (visualizar, actuar, activar)).
- Es recomendable tener el algoritmo expresado en pseudocódigo, de tal forma que este estructurado en unidades que representen operaciones simples (movimiento de datos, sumas, restas, and, or, xor, saltos, decisiones, comparaciones, llamados).
- Comenzar con una acción inicial y terminar con una final.
- Utilizar adecuadamente la simbología grafica acorde con la acción determinada en la secuencia del pseudocódigo.
- Representar el diagrama de arriba hacia abajo y/o de izquierda a derecha.
- Los símbolos se conectan por flechas que indican la dirección del flujo del programa, es recomendable utilizar solo líneas horizontales y verticales.
- Los símbolos pueden tener más de una línea de entrada a Excepción del símbolo de “fin”.
- Los símbolos de decisión deben contener al menos dos líneas de salida (si/no; falso/verdadero; cumple/no cumple).
- Los textos que se digitan y que son el pseudocódigo deben estar escritos de forma clara y legible evitando usar muchas palabras.
- Se recomienda no cruzar líneas de flujo pues se confunde con un nodo.
- Los nodos son puntos donde entran y salen líneas de flujo, los nodos no representan ninguna acción solo un punto de unión.
- No dejar líneas de flujo sin conectar.
- En caso que el diagrama de flujo haga un llamado a otro programa representarlo como una acción que se dirija a esa subrutina seguida de otra que retorne el resultado de la misma.
- Se recomienda nombrar cada programa y subprograma (subrutina) de manera que al convertirlo en lenguaje ensamblador, es acorde con el diagrama de flujo.

Ejemplo de diagrama de flujo:

```
INICIA:
        PONER EN BLANCO LA PANTALLA
        PREGUNTAR POR LA FUNCION A REALIZAR
        PREGUNTAR POR EL NOMBRE DEL ARCHIVO
        IF FUNCION ES 'E'
                THEN OCULTAR ARCHIVO
        ELSE
                MOSTRAR EL ARCHIVO
        END IF
TERMINA
```

Figura 44. ***Diagrama de flujo para mostrar u ocultar un archivo***[64]**.**



LECCIÓN 3: PROGRAMACIÓN CON DEBUG Y ASSEMBLER.

Existe un depurador muy poderoso llamado Code view, en el curso se trabaja con el Debugger bajo Windows. Es recomendable que los estudiantes utilicen primero el debugger que aun que su simplicidad hace que se tenga que repetir el

[64] ROJAS, 1997

programa cada vez que se modifica esto se transforma en una ventaja pues la repetición constante hace que se adquiera habilidad y se entienda mejor el contexto del programa y el sentido de assembler.

DEBUG

Bug es un término utilizado en ingles para hacer referencia a una falla o error en la secuencia de instrucciones de un programa que impide la normal ejecución del mismo. Esta falla puede provenir de un error sintáctico o lógico, por lo que depurar es la acción de corregir estos errores o fallas.

El Debugger o Debug es una utilería que ayuda a realizar tres tareas:

1. Ver el contenido de las memorias RAM y ROM
2. Ejecutar un programa, ya sea en su totalidad o una instrucción a la vez.
3. Ensamblar y ejecutar programas sobre la marcha.

El uso de Debugger nos permite trabajar directamente en lenguaje ensamblador con lo que se puede apreciar lo poderoso que este puede ser. Recordemos que lenguaje ensamblador, se trata de un lenguaje que se encuentra un paso antes del lenguaje real máquina (código binario), es un lenguaje donde se hace necesario suministrar con exactitud las instrucciones que debe ejecutar la CPU, a diferencia de los lenguajes de alto nivel y tal vez como punto en contra el lenguaje ensamblador requiere mas instrucciones para ejecutar lo mismo, pero a favor cuenta con la posibilidad de manipular y acceder directamente a todo el hardware del sistema cosa que no pueden hacer en su totalidad los lenguajes de alto nivel.

Primeros pasos con Debugger

Debugger se encuentra localizado en la mayoría de computadores con sistema basado en DOS o Windows, el programa suele estar localizado en el directorio de Windows en System32 y puede identificarse como debug.com o debug.exe, dependiendo de la versión del DOS o Windows que tenga en su equipo. El debug puede ser invocado mediante la ventana de línea de comandos (símbolo de sistema), que generalmente se encuentra en Inicio/Todos los programas/Accesorios/Símbolo de sistema.

Es hora de comenzar:

- Iniciar Debugger digitando en línea de comando Debug + enter, así, C:\ Debug [enter].
- Debe aparecer un guión "-"que indica que debug esta esperando un comando.
- Digitar – r [enter], recordar que el guión aparece por defecto. "r" es el comando de "register" del debug este comando permite exhibir o modificar uno o mas registros de la CPU.
- Como resultado se despliega el contenido de los siguientes registros ya estudiados, AX, BX, CX, DX, DS, ES, CS, SS, SI, DI, IP, SP, BP Y F.

¿Qué es la sintaxis? La sintaxis de se refiere a las reglas que determinan si un conjunto de caracteres o string son validos o no dentro de un programa.

Comando "**r**", la sintaxis de este comando es "r [registro]", donde [registro] es la notación de un registro valido de la CPU que constan de dos caracteres alfabéticos siguiendo la nomenclatura de los procesadores 808x y 80x86.

Los datos desplegados mediante el comando "**- r**" despliegan 14 registros internos, cada uno de 16 bits, los primeros cuatro AX, BX, CX y DX, son registros de uso general y se pueden usar como registros de 8 bits, es decir se parte el registro en dos partes de ocho bits la parte alta (ejemplo AH) y la parte baja (ejemplo AL). La totalidad de registros y sus nombres se exponen nuevamente en forma simplificada:

- AX (acumulador). Generalmente se utiliza para almacenar resultados de operaciones, lectura o escritura desde o hacia los puertos y como área de memoria temporal (scratch pad).
- BX (registro base). Sirve como registro apuntador base o índice.
- CX (registro contador). Se utiliza como constante en operación de iteración, como un contador que automáticamente se incrementa o decrementa de acuerdo con el tipo de instrucción usada.
- DX (registro de datos). Se usa comúnmente como puente para el acceso de datos.
- DS (registro de segmento de datos). Cualquier datos sea variable o no debe encontrarse dentro del segmento.
- ES (registro de segmento extra). Este registro permite operaciones sobre cadenas pero puede ser una extensión del DS.
- SS (registro del segmento de pila). Maneja la posición de memoria donde se encuentra la pila o stack. Esta estructura se utiliza para almacenar datos en forma temporal, tanto de un programa como de las operaciones internas del PC.
- CS (registro del segmento de código). En el CS es donde se encuentra el código ejecutable de cada programa ligado a los diferentes modelos de memoria.

- BP (registro de apuntadores base). Manipula la pila sin afectar el registro de segmento SS.
- SI (registro índice fuente). En conjunto con DI se utiliza para manejar bloques de cadenas en memoria siendo SI el primero y DI el segundo.
- DI (registro índice destino). Usado en conjunto con DI, SI representa la dirección donde se encuentra la cadena y DI la dirección donde será copiada.
- SP (registro del apuntado de la pila). Apunta a un área especifica de memoria que sirve para almacenar datos bajo la estructura LIFO (last in, first out: ultimo en entrar primero en salir), mejor conocido como pila (stack).
- IP (registro apuntador de la siguiente instrucción). Apunta a la siguiente instrucción que será ejecutada en memoria.
- F (Registro de banderas o flags). Es un registro de 16 bits aunque no todos son utilizados, estas son las banderas y sus significados:
    - Overflow: NV = no hay desbordamiento; OV= si lo hay.
    - Direction: UP = hacia adelante; DN = hacia atrás.
    - Interrupts: DI = desactivadas las interrupciones; EI = activadas.
    - Sign: PL = positivo; NG = negativo.
    - Zero: NZ = no es cero; ZR = si es cero.
    - Auxiliary Carry: NA = no hay acarreo auxiliar; AC = hay acarreo auxiliar.
    - Parity: PO = paridad non ; PE = paridad par
    - Carry: NC = no hay acarreo CY = si lo hay.

Estructura del ensamblador

En lenguaje ensamblador toda línea de código consta de dos partes.

1. la primera siempre contiene un comando de 2, 3 o 4 letras, estos comandos de llaman mnemónicos o códigos de operación porque representan una función que debe realizar la CPU.
2. La segunda parte son los operandos o datos sobre los cuales se va a trabajar, no todos los códigos de operación requieren operandos.

Ejemplos:

- mov ax,01 ➔ “mov” es el código de operación y “ax,01” son los operandos
- push ➔ no requiere comandos

Ensamblando un programa

Para crear al instante un programa en ensamblador con debugger se utiliza el comando “A” (Assemble), este comando permite introducir código en forma de nemónicos y su sintaxis es “A[dirección]”, donde dirección es la ubicación de

memoria a partir de la cual se empezara a ensamblar, sino se especifica la dirección inicial a partir de la cual se debe ensamblar, ensambla a partir de la localidad especificada por CS:IP.

Ejecutado el comando “A”, debug preguntará en forma sucesiva y secuencial por la siguiente instrucción a ensamblar, cada instrucción se ensambla en el momento de ser digitada y cada byte generado se almacena en la memoria en la dirección inicial y en secuencia sucesiva.

Digite “A [enter]”, como resultado se obtiene:

-A

17A3:0100

Debug se encuentra listo para aceptar las instrucciones:

mov ah, 8 [enter]

add ah, 3 [enter]

sub ah, 4 [enter]

int 20 [enter]

[enter]

Figura 45. *Entrando a Debugger*[65]

```
C:\Users\TURION>debug
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A3  ES=17A3  SS=17A3  CS=17A3  IP=0100   NV UP EI PL NZ NA PO NC
17A3:0100 0000          ADD     [BX+SI],AL                          DS:0000=CD
-a
17A3:0100 mov ah, 8
17A3:0102 add ah, 3
17A3:0105 sub ah, 4
17A3:0108 int 20
17A3:010A
-
```

Las localidades de memoria o segmento pueden ser distintas en cada PC. Para ejecutar el programa anterior se usa el comando “G [dirección]”, que ejecuta instrucciones a partir de una dirección. Se observa que nuestro programa comienza en el segmento 17A3:0100 y termina en el segmento 17A3:0108, si se desea ejecutar todas las instrucciones a partir de CS:IP (17A3:0100) hasta la instrucción en 17A3:0108, se digita “g108”, como resultado se obtiene:

[65] ROJAS, 1997

Figura 46. ***Ejecutar un programa con Debugger***[66]

```
C:\Users\TURION>debug
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A3  ES=17A3  SS=17A3  CS=17A3  IP=0100   NV UP EI PL NZ NA PO NC
17A3:0100 0000          ADD     [BX+SI],AL                      DS:0000=CD
-a
17A3:0100 mov ah, 8
17A3:0102 add ah, 3
17A3:0105 sub ah, 4
17A3:0108 int 20
17A3:010A
-g108

AX=0700  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A3  ES=17A3  SS=17A3  CS=17A3  IP=0108   NV UP EI PL NZ NA PO NC
17A3:0108 CD20          INT     20
-
```

Debug despliega los registros con los resultado intermedios, se observa comparando el estado inicial y final de los registros con respecto a AX en su parte alta, es decir, AH que tiene un valor inicial AX=0000, pasando por AX=0800, sumando 3, AX=0B00 y restando 4, AX=0700 que es el resultado final de este programa. Debug ejecuta la interrupción 20 “int 20” termina el programa y regresa el control al DOS.

Desensamblar

Para desensamblar un programa que se acaba de escribir se utiliza el comando “U” que desensambla lo que se digita partiendo de la localidad de memoria especificada y la cantidad de bytes especificados con “L” que significa longitud (length), lo que muestra tanto el código digitado con nemotécnicos como el código en hexadecimal de cada instrucción.

Figura 47. ***Desensamblar un programa con Debugger***[67]

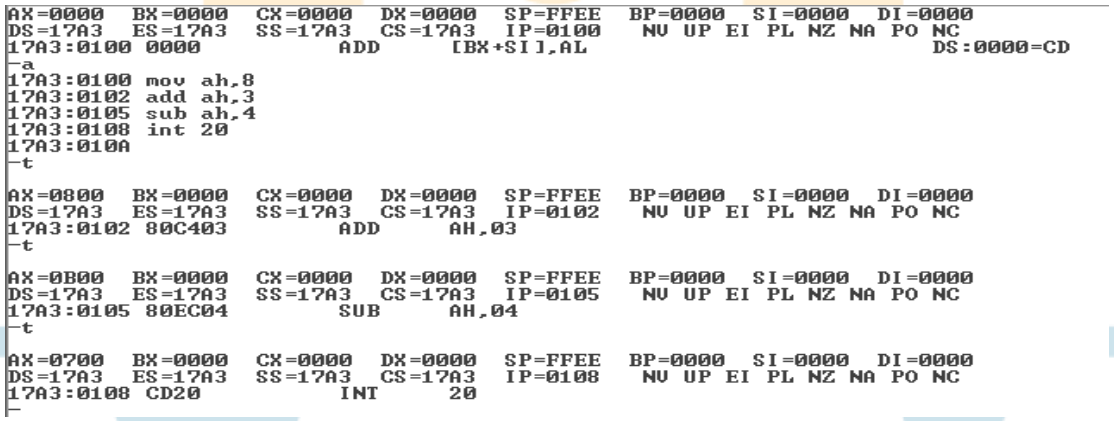
```
-a
17A3:0100 mov ah, 4
17A3:0102 add ah, 3
17A3:0105 sub ah, 4
17A3:0108 int 20
17A3:010A
-g108

AX=0300  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A3  ES=17A3  SS=17A3  CS=17A3  IP=0108   NV UP EI PL NZ NA PE NC
17A3:0108 CD20          INT     20
-u 100 L9
17A3:0100 B404          MOV     AH,04
17A3:0102 80C403        ADD     AH,03
17A3:0105 80EC04        SUB     AH,04
17A3:0108 CD20          INT     20
-
```

[66] ROJAS, 1997
[67] ROJAS, 1997

Existe también un comando que ayuda a rastrear la ejecución del programa haciendo un paso a paso este comando es el “Trace (t)”.

Figura 48. ***Comando “Trace” o de rastreo de ejecución en Debugger***[68]



Guardar y cargar un programa

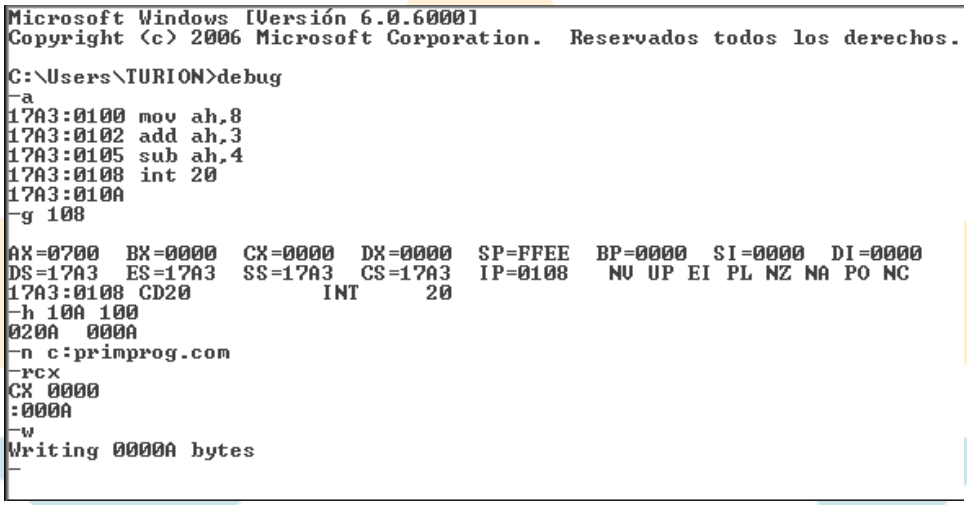
Lo hecho hasta ahora ha sido operar un programa en memoria, para guardarlo en disco y recuperarlo después, siga los siguientes pasos, tomando el código ya trabajado:

1. Averigüe la longitud del programa restando la dirección final (ultima instrucción del programa) de la dirección inicial (primera instrucción del programa), la longitud se expresa en hexadecimal. → - h 10A 100, lo que resulta en “020A 000A”.
2. Crear un nombre para el programa, que incluya la vía y la extensión. → - n c:primprog.com. no más de ocho caracteres para el nombre.
3. Poner la longitud del programa en el registro CX. → - rcx, lo que despliega el contenido del registro CX, CX 0000, se coloca la longitud del programa restando la dirección final 010A de la inicial 0100 que se obtiene 000A, se digita “:000A”.
4. Darla orden de escritura. La orden de escritura se da con “w”, - w aparece el mensaje “writing 000A bytes”, escribe la cantidad de byte en el registro CX.

El programa se almacena en Windows Vista queda guardado en C:/Users/”nombre del usuario”.

[68] ROJAS, 1997

Figura 49. ***Procedimiento que guarda un programa desde Debugger***[69]

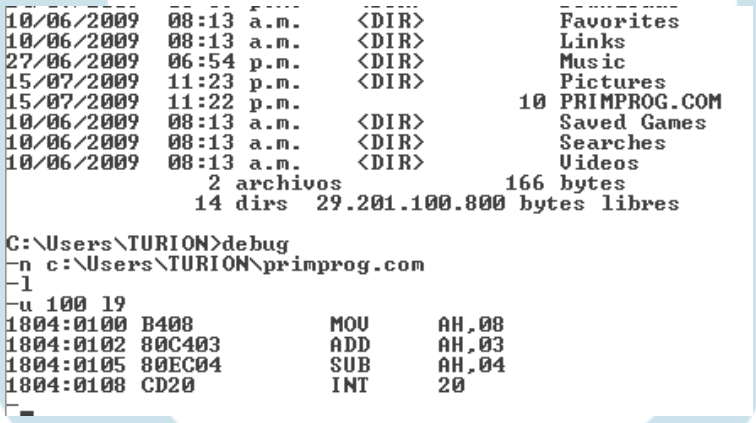


Para cargar el archivo anterior u otro archivo, se debe seguir los siguientes pasos:

1. Proporcionar el nombre del programa que se cargará usando el comando "n". → -n c:\ primprog.com
2. Cargarlo mediante comando (L) load. → -l
3. Para verificar que fue cargado, desensamble a partir de la localidad 100H. → -u 100 L9.

Para estar seguro de la ubicación antes de entrar a debug ejecute el comando "dir" para mostrar los archivos presentes en el lugar donde guardo el archivo.

Figura 50. ***Procedimiento para cargar un programa desde Debugger***[70]



Salir de debug, para salir de debug digite el comando "q", es decir, -q

[69] ROJAS, 1997
[70] ROJAS, 1997

LECCIÓN 4: INSTRUCCIONES BÁSICAS EN ASSEMBLER.

Movimiento de datos con respecto a la memoria

Las instrucciones que se encargan de mover datos en relación con la memoria se pueden categorizar por tareas específicas:

- Las que transfieren datos de un registro a otro o entre la memoria y los registros.
- Las que preparan registros para ingresar a una localidad de memoria.
- Las que manipulan la pila.
- Las que mueven grandes bloques de datos.
- Las que interactúan con dispositivos periféricos a través de puertos.

La instrucción de movimiento mas común es **"mov** "que requiere de dos operadores, su sintaxis es:

***Mov** destino, fuente*.

El destino no puede ser el registro de segmentos CS, no se puede mover un valor inmediato a un registro de segmentos para poder hacerlo utilice un registro intermedio como AX, ejemplo

Mov AX, 3

Mov DS, AX

Adicional a la instrucción "**mov**", existen otras que operan bajo circunstancias diferentes con el mismo propósito, la instrucción **"echo**" intercambia el contenido de la fuente con el destino. Otra instrucción útil es "**xlat**" (traslate), se utiliza comúnmente al trabajar con tablas o arreglos, esta instrucción reemplaza el valor de AL por un nuevo valor tomado de la tabla que se compone de un conjunto de localidades contiguas en memoria, y supone que el registro BX contiene la dirección del primer elemento de la tabla.

En resumen:

mov src, dest

mov dest, src

src : fuente: Inmediata. Registro. Memoria

dest : destino: Registro. Memoria

movz src, dest

movzx dest, src

src : fuente: Registro. Memoria

*dest :*destino: Registro

movs src, dest

movsx dest, src

src : fuente: Registro. Memoria

*dest :*destino: Registro

xchg src, dest

xchg dest, src

src : fuente: Registro. Memoria

dest : destino: Registro. Memoria

Administración de la pila

La pila es usada automáticamente por las instrucciones "**call**", "**int**", "**ret**" y "**iret**", para guardar o restaurar la dirección de retorno antes de ejecutar las rutinas indicadas por dichas instrucciones. Otros servicios que presta es para pasar parámetros entre rutinas, o bien de un lenguaje de alto nivel al ensamblador, la pila se manipula mediante las instrucciones "**push**" (empuja o almacena en la pila) y "**pop**" (extrae o saca de la pila).

Por ejemplo la instrucción "**push [1125]**" empuja a la pila el contenido de la localidad de memoria **1125.**

Otro ejemplo, las instrucciones "**push ax**", "**push bx**" almacenan en la pila los valores de los registros AX y BX, para después sacarlos de la pila con las instrucciones "**pop ax**", "**pop bx**".

Movimiento bloques de datos

En la programación de alto nivel normalmente se deben inicializar arreglos o copiar cadenas de caracteres, en lenguaje ensamblador existen instrucciones que permiten trabajar con bloques de datos, estas son "**movs**", "**stos**" y "**lods**" los dos

primeros se utilizan con un prefijo de repetición, como *REP*, lo que permite repetir una operación tantas veces como se estipule en el registro CX, lo que conlleva a una minimización del código.

Existen otros prefijos como *REPE* (repite hasta encontrar la igualdad) o *REPZ* (repite hasta que el valor sea cero), *REPNE (repite mientras el valor no sea igual)* o REPNZ (repite mientras el valor no sea cero) usados en combinación con las instrucciones "**cmp**" (comparar) y "**scas**" (escanerar-cadena).

Trabajo con bloques, El movimiento de grandes bloques de memoria y su inicialización se basan en fuente-destino controlados por SI-DI y un bit en el registro de flags que especifica la dirección o sentido en la cual se debe trabajar: "**cdl**" hacia adelante o "**std**" de atrás hacia adelante. Las instrucciones "**movs**" tiene dos modalidades, la primera puede mover bytes "**movsb**" o palabras "**movsw**" estando en función de los registros pares DS:SI como fuente y ES:DI como destino.

Instrucciones lógicas y aritméticas

Las operaciones aritméticas y lógicas son parte del repertorio de instrucciones de cualquier lenguaje, en x86 se incorporar las instrucciones suficientes para efectuar las operaciones aritméticas básicas (suma, resta, multiplicación y división), las instrucciones tienen la capacidad de trabajar con operando de 8 – 16 bits con y sin signo, se debe recordar que el bit mas significativo del byte o palabra es el signo.

Suma, se pueden utilizar tres instrucciones las cuales a su vez se agrupan dos que funcionan con dos operandos y una con solo un operando:

- Con dos operandos:
  - "**add**" o suma. Sintaxis "**add** destino, fuente", ejemplo ADD DX, 5.
  - "**adc**" o suma con acarreo. La instrucción realiza la suma y automáticamente toma en cuenta el acarreo, usa generalmente 32 bits para sus operaciones, las operaciones se realizan sobre el registro DX:AX. Sintaxis "**adc** destino, fuente", ejemplo ADC DX, 5.

- Un solo operador:
  - "**inc**" o incremento. Es una operación de incremento en uno para cualquier registro o localidad de memoria. Sintaxis "**inc** [registro o posmen]", ejemplo INC AX. La desventaja es que "**inc**" trata a su

operando como operando sin signo, si un byte contiene **FFH** o todos los bits a uno, aplicando "**inc**" todos los bits pasan a cero.

Resta, son la contraparte de la suma también presenta tres instrucciones agrupadas en dos con dos operandos y una con uno solo.

- Con dos operandos:
  - "**sub**" o resta. Sintaxis "**sub** destino, fuente", ejemplo SUB AX, DX.
  - "**sbb**" o resta con acarreo. La instrucción realiza la resta y automáticamente toma en cuenta el acarreo. Sintaxis "**sbb** destino, fuente", ejemplo SBB AX, DX.

- Un solo operador:
  - "**dec**" o decremento. Es una operación de decremento en uno para cualquier registro o localidad de memoria. Sintaxis "**dec** [registro o posmen]", ejemplo DEC DX.

Existe una variación de la operación resta, la instrucción "**neg**" resta el operando de cero. Si digitamos "**NEG AX**" restará el contenido AX a la cantidad cero "0", como resultado se obtiene la negación, es decir, el cambio de todos los unos por cero y todos los ceros por unos y al resultado se le suma AX.

Multiplicaciones, es un caso singular de sumas repetitivas, se presentan dos instrucciones:

- "**mul**" para multiplicar valores con signo. Sintaxis "**mul** [multiplicador]".
- "**imul**" para multiplicar valores sin signo. Sintaxis "**imul** [multiplicador]".

Las restricciones que tiene la multiplicación son, que el multiplicando no se define explícitamente, debe encontrarse en el registro AX y tener el mismo tamaño del multiplicando (8-16 bits), la multiplicación de 8 bits no debe producir un resultado mayor a 16 bits y una operación de 16 bits no puede producir un resultado mayor a 32 bits, el resultado se encuentra en el registro par DX:AX.

Ejemplo 1: Pone en AL el máximo valor representable en 8 bits y lo multiplica por 6, dejando el resultado en AX.

Ejemplo 2: AX = CX=65535, multiplica AX y CX y deja el resultado en el registro par DX:AX

Figura 51. ***Como multiplicar***[71]

```
C:\Users\TURION>debug
-a
17A3:0100 mov al,ff
17A3:0102 mov cl,9
17A3:0104 mul cl
17A3:0106 int 20
17A3:0108
-g106

AX=08F7  BX=0000  CX=0009  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A3  ES=17A3  SS=17A3  CS=17A3  IP=0106   OV UP EI PL NZ NA PO CY
17A3:0106 CD20           INT      20
-
```

```
C:\Users\TURION>debug
-a
17A3:0100 mov ax,ffff
17A3:0103 mov cx,ffff
17A3:0106 mul cx
17A3:0108 int 20
17A3:010A
-g108

AX=0001  BX=0000  CX=FFFF  DX=FFFE  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A3  ES=17A3  SS=17A3  CS=17A3  IP=0108   OV UP EI PL NZ NA PO CY
17A3:0108 CD20           INT      20
-
```

División, de igual forma que la multiplicación la división presenta dos instrucciones:

- "**div**" para dividir valores con signo. Sintaxis "**div** [divisor]".
- "**idiv**" para dividir valores sin signo. Sintaxis "**idiv** [divisor]".

En la división el dividendo se almacena en AX, siguiendo las mismas pautas que en la multiplicación, para un divisor de 8 bits se espera un dividendo de 16 bits en AX, pero un divisor de 16 bits requiere un divisor de 32 bits en el registro par DX:AX, siendo la palabra alta DX y la palabra baja AX. Cuando la división es un byte, el cociente se almacena en el registro AL y el residuo en AH. Si el divisor es una palabra, el cociente se almacena en AX y el residuo en DX.

Comparación, se utiliza la instrucción "**cmp**" la comparación es una resta simple que no afecta el contenido del operando fuente, pero el resultado afecta al registro de banderas. Ejemplo "CMP BX,AX".

En resumen: donde arg1 y arg2 son argumentos o registros.

cmp arg1, arg2

[71] ROJAS, 1997

cmp arg2, arg1

Comparación de bits, se utiliza "**test**", la cual tiene la capacidad de comparar bits para poder tomar decisiones. Ejemplo "TEST AL, 1", esto prueba el estado del bit 1 en el registro AL.

Comparando cadenas, las dos instrucciones utilizadas son "**scans**" – scan string o buscar cadenas y "**cmps**" – compare string o comparar cadena, este tiene dos variantes "**cmpsb**" que compra el byte y "**cmpsw**" compara palabra. Estas instrucciones se utilizan en conjunto con los prefijos:

- "**repe**" o "**repz**", repite la instrucción mientras la bandera de cero se encuentra prendida o hasta que el registro CX sea mayor que cero.
- "**repn**" o "**repnz**" es el inverso del anterior, repitiendo la instrucción mientras la bandera de cero sea cero o hasta que CX sea mayor que cero.

Instrucciones lógicas, existen cuatro tipos de operaciones lógicas: "**and**, **not**, **or** y **xor**" estas efectúan la operación bit por bit sobre sus operandos. "**and**, **or** y **xor**" trabajan sobre operandos destino – fuente, y la operación "**not**" complementa todos los bits de un único operando.

Desplazamiento de bits, assembler cuenta con instrucciones que permiten desplazar o rotar bits dentro de un registro (byte o palabra).

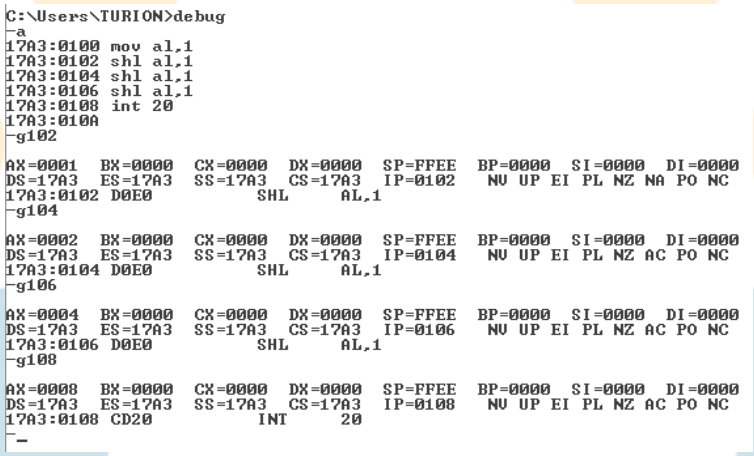
- Sirven para desplazar bits:
  - La instrucción "**shl**" Shift left : desplazamiento a la izquierda.
  - La instrucción "**sal**" Shift arithmetic left : desplazamiento aritmético a la izquierda.
  - La instrucción "**sht**" Shift right : desplazamiento a la derecha.
  - La instrucción "**sar**" Shift arithmetic right : desplazamiento arimetico a la derecha.

- Sirven para rotar bits:
  - La instrucción "rcl" rotate left thru carry : rotación con acarreo a la izquierda.
  - La instrucción "rcr" rotate rigth thru carry : rotación con acarreo a la derecha.
  - La instrucción "rol" rotate left : rotación a la izquierda.

- La instrucción “ror” rótate right . rotación a la derecha.

Ejemplo: el valor inicial de AL=1 y al aplicar la instrucción “**shl**” se presenta un desplazamiento a la izquierda con lo que equivale a multiplicar por 2.

Figura 52. ***Desplazamiento de bits con instrucción “SHL”***[72]**.**



Instrucciones de control de flujo, son las encargadas de generar saltos absolutos o condicionales y llevar a diversas partes del programa principal denominadas rutinas o subrutinas.

- Los saltos incondicionales: “**jmp** loc” la siguiente dirección a ejecutar se localiza en la localidad de memoria “loc”.
- Salto de igualdad: “**je** loc” si el resultado de la operación anterior “**cmp**” arroja una igualdad en los operandos se lleva a la dirección especificada por “loc”.
- Salto de no igualdad: “**jne** loc” si el resultado de la operación anterior “**cmp**” arroja que los operando no son iguales se lleva a la dirección especificada por “loc”.
- Saltar si mayor:
  - “loc **jg**” si primer operador de la anterior instrucción “**cmp**” es mayor que el segundo.
  - “**jge** loc” si primer operador de la anterior instrucción “**cmp**” es mayor o igual que el segundo.
  - “**ja** loc” similar a “**jg**”.
  - “**jae** loc” similar a “**jge**”.
- Saltar si menor:
  - “**jl** loc” si primer operador de la anterior instrucción “**cmp**” es menor que el segundo.

[72] ROJAS, 1997

  - "**jle** loc" si primer operador de la anterior instrucción "**cmp**" es menor o igual que el segundo.
  - "**jb** loc" similar a "**jl**".
  - "**jbe** loc" similar a "**jle**".
- Salto de desbordamiento: "**jo** loc" si la expresión aritmética anterior tuvo un desbordamiento.
- Ir a cero:
  - "**jnz** loc" si una operación aritmética no resultó en cero, salta a la localización de memoria dada por "loc".
  - "**jz** loc" si una operación aritmética resultó en cero, salta a la localización de memoria dada por "loc".
- Funciones de llamado:
  - "**call** proc", utilizado en subrutinas, coloca la dirección siguiente de la instrucción "**call**" y ejecuta la subrutina identificada por la etiqueta o dirección "proc".
  - "**ret** [valores]" retorna de una subrutina con un valor cargado en registro.
- Instrucciones de bucle.
  - "**bucle** arg" utiliza CX como contador que es decrementado para ejecutar un bucle.
  - "**loopx** arg" con x representando a "**loope**, **loopne**, **loopnz** y **loopz**", esta instrucción decrementa CX en el bucle y salta a la dirección especificada por "arg".
- Otras instrucciones de control:
  - "**nop**" no hace nada, solo consume ciclos maquina.
  - "**wait**" espera a que la CPU realice el ultimo calculo.

Interrupciones

Las interrupciones generan interrupciones en el flujo normal del programa para atender un evento interno o externo y ejecutar un programa de interrupción, al finalizar el programa o atender la interrupción se devuelve el control al programa principal en ejecución. "**int** arg" es la instrucción y su sintaxis utilizada para referirse a una interrupción (específicamente de software), existen interrupciones varios tipos de interrupciones:

- Interrupción por hardware, como por ejemplo en teclados.
- Interrupción por software, usadas para transferir el control al núcleo del sistema operativo, esta es la que se activa con la instrucción "**int**". Las interrupciones de software se dividen en:
  - Interrupciones de BIOS (Basic Input / Output System : Sistema básico de entrada / salida), estas controlan en su mayoría los

periféricos, aunque también pueden abarcar algunas funciones de disco.
    - Interrupciones DOS (Disk Operating System : Sistema Operativo en Disco), encargadas de administrar la memoria, el disco y algunas que manejan entrada y salida de información.
- Interrupciones excepcionales, como dividir por cero o acceder a un área de memoria protegida.

Las interrupciones mas usuales para trabajar con Debugger, MASM u otro ensamblador son:

Int 20h - Terminar programa MS-DOS.
Int 21h - MS-DOS. Esta tiene una serie de funciones: que puede visualizar en los archivos del curso.

Las interrupciones DOS se usan cargando el parámetro de la función en el registro "**AH**" e invocando la interrupción correspondiente, es de aclarar que no todas las interrupciones DOS tienen funciones.

Igualdades, sirven para definir con un nombre simbólico a una constante. Esto permite que el programa sea más legible.

```
LlamadaMSDOS        equ     21h
```

La anterior igualdad hace que en lugar de escribir en el cuerpo del programa "*int 21hN"* se pueda escribir "*int LlamadaMSDOS*".

LECCIÓN 5: EJEMPLOS DE APLICACIÓN.

Como herramientas de programación en ensamblador tenemos el Debugger, MASM y el Simuproc. Es recomendable comenzar con Debugger digitando y probando los códigos de programa que aquí se exponen. Luego es recomendable tomar la documentación suministrada sobre las especificaciones del programa Simuproc hacer una lectura analítica y comenzar por introducir instrucciones simples y notar su funcionamiento, mas adelante podrá realizar pequeños programas adecuándolos desde Debugger a Simuproc y hacer la ejecución del mismo, hasta llegar al punto de producir sus propios programas antes de pasar a trabajar con MASM, el cual no se utiliza en profundidad en este curso dada su incompatibilidad y problemas de ejecución en los sistemas Windows vista que gran parte de la población utiliza.

Es importante que los estudiantes cada vez que generen un programa en Debugger o MASM, sean consientes que pueden al digitar o probar algún programa quedar en un ciclo infinito o bloquear el sistema, por lo que se aconseja no tener abierto documentos o programas con información relevante susceptible de ser perdida en el momento de realizar los ejercicios, es conveniente destinar un equipo y espacio de tiempo para realizar los programas que se exponen a continuación, cuanto sucedan eventos de iteración infinita o bloqueo reiniciar el equipo.

Si se utiliza SimuProc dado que es un emulador aunque también puede generar algún “bug” la solución es menos drástica, en este caso cerrando el programa y volviéndolo a ejecutar.

Controlar condiciones, iteraciones y bifurcaciones.

Las estructuras básicas de programación utilizan las iteraciones, condiciones y bifurcaciones proporcionándole al sistema cierto grado de inteligencia y decisión:

El siguiente programa despliega 10 veces una cadena de caracteres, como características especiales se utiliza la INT 21H y su función 9, exige que el registro par DS:DX contenga la dirección inicial de la cadena a desplegar, la cadena debe terminar con el signo “$”, la instrucción para control de iteración es LOOP que salta a la localidad que repite la INT 21H y decrementa a CX hasta que llegue a cero.

Figura 53. ***Ejemplo de iteración utilizando la instrucción “LOOP”***[73]

```
C:\Users\TURION>DEBUG
-A
17A3:0100 JMP 135
17A3:0102
-E 102 'MICROPROCESADORES Y MICROCONTROLADORES 2009' 0D 0A '$'
-A 135
17A3:0135 MOV CX, 000A
17A3:0138 MOV DX, 0102
17A3:013B MOV AH, 9
17A3:013D INT 21
17A3:013F LOOP 013D
17A3:0141 INT 20
17A3:0143
-G141
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009

AX=0924  BX=0000  CX=0000  DX=0102  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A3  ES=17A3  SS=17A3  CS=17A3  IP=0141   NV UP EI PL NZ NA PO NC
17A3:0141 CD20          INT     20
```

[73] ROJAS, 1997

Observe que la cadena se encuentra entre comillas simples, seguida por un espacio, el valor hexadecimal 0DH (retorno de carro), otro espacio, el valor hexadecimal 0AH (código de línea nueva), otro espacio y finalmente el símbolo “$” (terminación de cadena para ensamblador).

Variante del programa 1.

Una variante del anterior programa utiliza dos instrucciones para controlar el bucle, la primera sirve para usar el registro BX como una variable, que se carga con la cantidad de veces que se quiere desplegar la cadena e ir decrementando de uno en uno, utilizando la instrucción “DEC BX”, para poder bifurcar en función del resultado de la bandera “NZ”, la instrucción para generar la bifurcación es “JNZ” (Jump if NotZero) si no es igual a cero despliega nuevamente la cadena, si es igual a cero termina el programa con la interrupción “INT 20”.

Figura 54. ***Iteración con “DEC BX” y condición de salto “JNZ”***[74]

```
C:\Users\TURION>DEBUG
-A
17A3:0100 JMP 135
17A3:0102
-E 102 'MICROPROCESADORES Y MICROCONTROLADORES 2009' 0D 0A '$'
-A 135
17A3:0135 MOV BX, 0A
17A3:0138 MOV DX, 102
17A3:013B MOV AH, 9
17A3:013D INT 21
17A3:013F DEC BX
17A3:0140 JNZ 13D
17A3:0142 INT 20
17A3:0144
-G142
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009

AX=0924  BX=0000  CX=0000  DX=0102  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A3  ES=17A3  SS=17A3  CS=17A3  IP=0142   NV UP EI PL ZR NA PE NC
17A3:0142 CD20          INT     20
-
```

Variante del programa 2.

Esta variación hace uso del registro CX como contador, también utiliza la instrucción JCXZ (Jump if CX = 0) que controla la condición de salto, con esto se ha observado que existen varias formas en assembler de llegar al mismo resultado o hacer la misma función.

[74] ROJAS, 1997

Figura 55. ***Iteración utilizando el contador "CX" y el salto condicional "JCXZ"***[75]

```
C:\Users\TURION>DEBUG
-A
17A3:0100 JMP 135
17A3:0102
-E 102 'MICROPROCESADORES Y MICROCONTROLADORES 2009' 0D 0A '$'
-A 135
17A3:0135 MOV DX, 0102
17A3:0138 MOV CX, 000A
17A3:013B MOV AH, 09
17A3:013D INT 21
17A3:013F DEC CX
17A3:0140 JCXZ 0144
17A3:0142 JMP 013D
17A3:0144 INT 20
17A3:0146
-G144
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009
MICROPROCESADORES Y MICROCONTROLADORES 2009

AX=0924  BX=0000  CX=0000  DX=0102  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A3  ES=17A3  SS=17A3  CS=17A3  IP=0144   NV UP EI PL ZR NA PE NC
17A3:0144 CD20          INT     20
-_
```

El ejercicio y practica constante con la programación en lenguaje ensamblador permite con el tiempo adquirir habilidades y competencias, con las que el programador puede determinar las instrucciones, su disposición y la estructura general del programa buscando mejorar la velocidad de ejecución y estructura del código.

Interrupciones

Las interrupciones son el medio para que la CPU tenga "conciencia" de su entorno, oprimir una tecla, es un evento que genera una interrupción y que le dice a la CPU que desde el teclado se esta generando información que debe ser atendida. La atención de la interrupción en este caso consiste en detener el programa en ejecución y atender el teclado, obtener el código de la tecla presionada, guardando este código en una memoria intermedia (memoria intermedia de teclado – type ahead buffer). Es de notar que hay otro tipo de interrupciones que ocurren o suceden automáticamente 18 veces por segundo (aproximadamente) que es la que actualiza la hora del día en el sistema.

Las interrupciones son tan habituales y ocurren tan seguido que la CPU debe tener un modo eficiente de tratarlas, por lo que hace uso de un circuito integrado (chip) que le permite establece la prioridad de la interrupción a nivel de hardware.

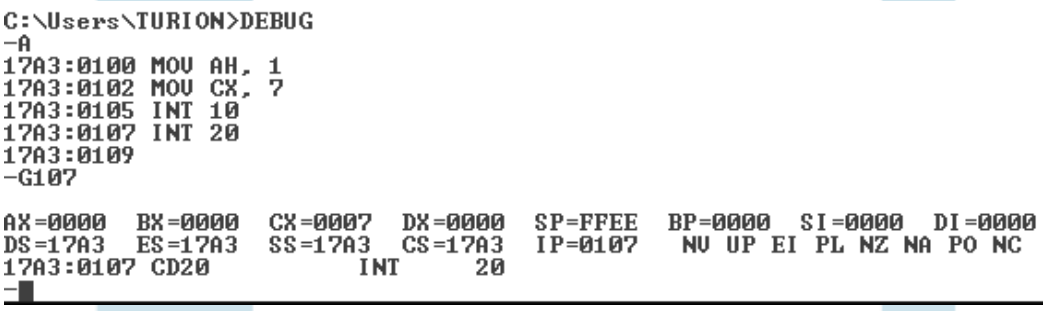
[75] ROJAS, 1997

En MS-DOS se mantiene una tabla de vectores de interrupción en la memoria baja, que empieza por la localidad de memoria “cero” y termina en la “256” (decimal). Los vectores de interrupción apuntan a localidades de memoria donde la CPU ejecuta el programa que atiende la interrupción, esta técnica se denomina “darle servicio a la interrupción”. Se puede definir la interrupción como una bifurcación a una localidad de memoria RAM o ROM donde la CPU inicia la ejecución de una serie de instrucciones y al terminar regresa a la siguiente localidad de memoria de la instrucción donde se ocasionó la interrupción.

Programa utilizando interrupciones

Este programa utiliza la función 1 de la interrupción BIOS, la cual cambia la forma del cursor, la función a utilizar se carga en el registro AH y el código del cursor en CX.

Figura 56. ***Uso de las Interrupciones***[76]



Para regresar el cursor a su modalidad predefinida, usualmente una raya al piso, se cambia la línea que dice “MOV CX, 0607”, valido para las tarjetas adaptadoras compatibles con color. Al salir de Debugger (Q), el curso regresa a su estado normal.

El siguiente programa utiliza la interrupción 21H de DOS, empleando dos funciones, la primera “**AH=1**” lee el teclado y la segunda “**AH=2**” escribe en la pantalla, el programa lee caracteres del teclado hasta encontrar <CR> (retorno de carro).

[76] ROJAS, 1997

Figura 57. ***Uso de la interrupción 21H de DOS***

```
C:\Users\TURION>DEBUG
-A
17A3:0100 MOV AH, 1
17A3:0102 INT 21
17A3:0104 CMP AL, 0D
17A3:0106 JNE 100
17A3:0108 MOV AH, 2
17A3:010A MOV DL, AL
17A3:010C INT 21
17A3:010E INT 20
17A3:0110
-G110
BIENVENIDOS AL CURSO DE MICROPROCESADORES Y MICROCONTROLADORES
Program terminated normally
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=17A3  ES=17A3  SS=17A3  CS=17A3  IP=0100   NV UP EI PL NZ NA PO NC
17A3:0100 B401          MOV     AH,01
```

El siguiente programa de ejemplo sirve para afianzar los temas expuestos, ensamblado, ejecución, guardar y cargar el archivo, desensamblar y ejecutarlo nuevamente, este programa permite ocultar o mostrar un archivo, después de ocultarlo tendrá que recurrir a otro medio para visualizarlo.

Figura 58. ***Programa que permite ocultar o mostrar un archivo***[77]**.**

```
C:\Users\TURION>DEBUG
-A 100
17A3:0100 JMP 138
17A3:0102
-E 102 'FUNCION A REALIZAR (E) O (M) : $'
-E 122 'NOMBRE DEL ARCHIVO : $'
-E 250 D 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-E 246 0D 0A '$'
-A 138
17A3:0138 MOV AH, 0
17A3:013A MOV AL, 2
17A3:013C INT 10
17A3:013E MOV DX, 102
17A3:0141 MOV AH, 9
17A3:0143 INT 21
17A3:0145 MOV AH, 0
17A3:0147 INT 16
17A3:0149 MOV [249], AL
17A3:014C MOV DX, 246
17A3:014F MOV AH, 9
17A3:0151 INT 21
17A3:0153 MOV DX, 122
17A3:0156 MOV AH, 9
17A3:0158 INT 21
17A3:015A MOV AH, 0A
17A3:015C MOV DX, 250
17A3:015F INT 21
17A3:0161 MOV AL, [249]
17A3:0164 CMP AL, 45
17A3:0166 JNZ 16D
17A3:0168 MOV CX, 2
17A3:016B JMP 170
17A3:016D MOV CX, 0
17A3:0170 MOV DI, 252
17A3:0173 MOV AL, 0D
17A3:0175 SCASB
17A3:0176 JNZ 175
17A3:0178 MOVE BYTE PTR [DI-01], 00
17A3:017C MOV AH, 43
17A3:017E MOV DX, 252
17A3:0181 MOV AL, 01
17A3:0183 INT 21
17A3:0185 INT 20
17A3:0187
-H 0187 0100
0287  0087
-N C:MPMC01.COM
-RCX
CX 0100
:0187
-W
Writing 00187 bytes
-N C:\Users\TURION\MPMC01.COM
-L
-U 100 186
```

[77] ROJAS, 1997

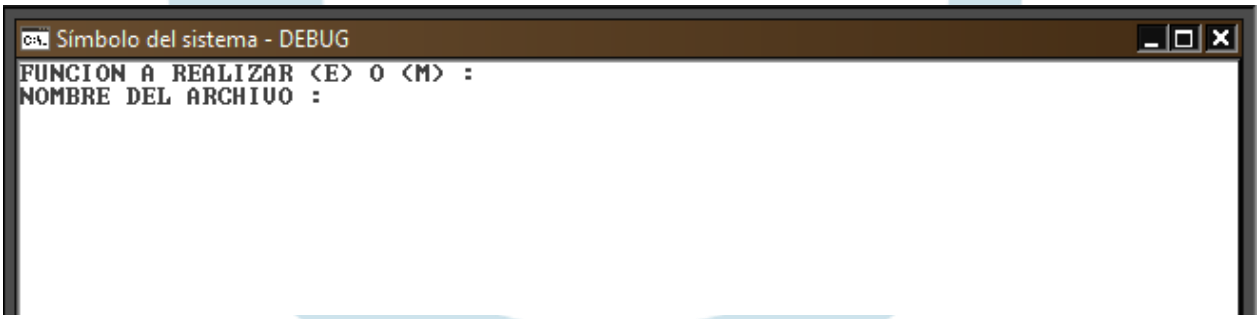
El anterior código está basado en un algoritmo que previamente se ha diseñado, como toda fase de diseño de un proyecto hay que tener primero un conocimiento absoluto de las instrucciones y modo de ejecución, posteriormente se emplea los conocimiento en programación para diseñar el algoritmo correspondiente para finalmente convertirlo en el lenguaje que se utiliza para lograr el objetivo del proyecto.

Para este caso presentamos primero el código para que el estudiante se acerque más a las instrucciones y se familiarice con ellas, de esta forma se puede comenzar un análisis de cada instrucción, con respecto al algoritmo siguiendo pautas de ingeniería inversa. Este es el algoritmo en pseudocódigo.

```
INICIA:
        PONER EN BLANCO LA PANTALLA
        PREGUNTAR POR LA FUNCION A REALIZAR
        PREGUNTAR POR EL NOMBRE DEL ARCHIVO
        IF FUNCION ES 'E'
                THEN OCULTAR ARCHIVO
        ELSE
                MOSTRAR EL ARCHIVO
        END IF
TERMINA
```

En el caso del Windows, como alternativa para visualizar archivos ocultos se puede recurrir al explorador de Windows, menú /Herramientas/Opciones de carpeta, pestaña VER y seleccionar "mostrar todos los archivos y carpetas ocultas", con esto podrá ver el archivo oculto y con clic derecho/Propiedades deseleccione la opción "oculto".

Figura 59. ***Interfaz del programa anterior, ventana DOS***[78]**.**



[78] ROJAS, 1997

ACTIVIDADES DE AUTOEVALUACIÓN DE LA UNIDAD

1. Realizar una investigación mediante consulta en internet sobre los últimos microprocesadores para servidores, equipos de escritorio y portátiles.
2. Investiga cual es el ranking de las mejores supercomputadoras.
3. Consulta las características de los últimos procesadores ofrecidos por las empresas INTEL y AMD.
4. Disponer de un equipo de computo y del tiempo necesario para probar cada programa propuesto en la lección “Ejemplos de aplicación” en el Capítulo 3 utilizando el interprete Debugger, no se debe estar trabajando en otros programas o tener información importante susceptible de ser perdida, al utilizar Debugger pueden generarse ciclos infinitos o bloquear el sistema. Si sucede un bloqueo reinicie la maquina. La utilización de Debugger dentro de parámetros controlados como los ejercicios propuestos no implican perdida de información, es recomendable consultar la bibliografía propuesta si quiere profundizar en estos temas.
5. Descargar el emulador “SimuProc” e instalarlo en el equipo, con compatibilidad “Windows XP”, descargar la documentación respecto a SimuProc, realizar lectura del documento.
6. Proceder a ingresar de forma manual cada instrucción estudiando el comportamiento y las microoperaciones involucradas.
7. SimuProc tiene una serie de programas de demostración los cuales puede abrir y ejecutar, es conveniente que los estudie paso a paso para entender su funcionamiento.
8. Realizar el algoritmo, diagrama de flujo e introducir en el intérprete Debugger, guardar, cargar y ejecutar programas sencillos que involucren movimiento de registros, operaciones aritméticas, comparaciones y bifurcaciones. por ejemplo:
    a. Área de un cuadrado, triangulo, círculo o alguna superficie geométrica.
    b. Serie de Fibonacci.
    c. Factorial de un número.

9. Realizar el algoritmo, diagrama de flujo, código fuente, compilar y ejecutar programas sencillos en SimuProc que involucren movimiento de registros, operaciones aritméticas, comparaciones y bifurcaciones. por ejemplo:
    a. Área de un cuadrado, triangulo, círculo o alguna superficie geométrica.
    b. Serie de Fibonacci.
    c. Factorial de un número.
10. Comparar las experiencias, sacar conclusiones y debatirlas en el grupo.

BIBLIOGRAFIA

Ureña, López Alfonso, Sánchez, Solano Antonio Miguel, Martin, Valdivia María Teresa & MANTAS, Ruiz Jose Miguel. (1999). Fundamentos de informática. Editorial Alfaomega & ra-ma. Santafé de Bogotá.

Rojas, Ponce Alberto. (1997). "Ensamblador Básico". Editorial Computec. AlfaOmega Santafé de Bogotá.

CEKIT. (2002). Curso Práctico de Microcontroladores: Teoría, Programación, Diseño, Prácticas Proyectos completos. Editorial Cekit. Pereira-Colombia.

Téllez, Acuña Freddy Reynaldo. (2007). Modulo de Microprocesadores y Microcontroladores. UNAD.

Stallings, William. "Organización y Arquitectura de Computadores". (5ª edición). Editorial Prentice-Hall. Madrid, 2000.

Tokheim, Roger. "Fundamentos de los Microprocesadores". (2ª edición). Editorial Mc Graw Hill. México, 1985.

Uruñuela, José Mª. "Microprocesadores: Programación e Interconexión". (2ª edición). Editorial Mc Graw Hill. España, 1995.

UNIDAD 2: MICROCONTROLADORES

Nombre de la Unidad	**MICROCONTROLADORES**
Introducción	La Segunda Unidad de este curso, está enfocada al tema de los microcontroladores donde el estudiante adquiere los conocimientos básicos de construcción, arquitectura interna y diferencias con los microprocesadores, como punto cumbre de esta unidad se presentan dos de las principales familias de microcontroladores exponiendo sus características más relevantes junto con sus conjuntos de instrucciones con lo que el estudiante contara con las herramientas necesarias para empezar a programar estos dispositivos producto de la evolución de los microprocesadores y que son utilizados para aplicaciones especificas de control.
Justificación	Igual como se justifica la unidad 1, el estudio de la unidad 2 está dispuesto para la Ingeniería Electrónica y profesiones afines, donde con estos dispositivos se buscan aplicaciones de control robusto para infinidad de proyectos que dependen o están limitados por el ingenio del diseñador y por el problema a resolver.
Intencionalidades Formativas	Con el desarrollo de esta Unidad y teniendo muchos puntos y temas de apoyo heredados de la unidad anterior se enfoca a los estudiantes en el concepto de microcontrolador y sus diferencias con el microprocesador, pasando por la arquitectura, disposición de pines, direccionamiento, registros y set de instrucciones, estableciendo las bases para hacer las primeras pruebas e implementación de proyectos simples.
Denominación de capítulos	Capitulo 4: Introducción a los microcontroladores. Capitulo 5: Microcontroladores de 8 bits PIC16F84 / PIC16f877. Capitulo 6: Microcontroladores de 8 bits Motorola Freescale MC68H(R)C908/JL3/JK3/JK1

CAPÍTULO 4: INTRODUCCIÓN A LOS MICROCONTROLADORES

Con la aparición de los microprocesadores y las necesidades de control para distintos dispositivos tanto industriales (instrumentación, automatización, telemetría, etc.), comerciales (automóviles, periféricos, juguetes) y domésticos (electrodomésticos, audio, video), aparece la necesidad tecnológica de incorporar en un solo “chip” la estructura básica de un sistema de computo, este “microcomputador” debería contar con tres unidades funcionales de cualquier equipo de computo: CPU, memoria y unidades I/O (Entrada/salida). Es así como se da origen a los microcontroladores, pequeños dispositivos producto de la microelectrónica generalmente de arquitectura cerrada que fusionan en una misma pastilla de silicio las tres unidades funcionales de una computadora, aplicados a situaciones especificas de control y capaces de incorporar unidades adicionales que amplían su capacidad de interacción con el medio incluso llegando a comportarse como sistemas abiertos (el caso de los microprocesadores).

LECCIÓN 1: GENERALIDADES DE LOS MICROCONTROLADORES.

Origen

En 1969, ingenieros de la compañía japonesa BUSICOM, buscan soluciones para fabricar con pocos componentes sus dispositivos (calculadoras), esta proposición se le hizo a INTEL quien en un proyecto dirigido por Marcian Hoff y apoyado por Federico Faggin, logro fabricar un bloque integrado denominado “microprocesador” adquiriendo los derechos de la compañía BUSICOM y entregando al mercado en 1971 el primer microprocesador el 4004 de 4 bits. Como ya se ha mencionado le siguieron el i8008, i8080, el Motorola 6800, Z80, i8085.

En 1976 aparece en el mercado un nuevo dispositivo que incorpora una CPU, memoria RAM - ROM y puertos de I/O, este dispositivo es llamado "microcontrolador" que son microcomputadoras en un solo chip, dos de los mas representativos y primeros microcontroladores fueron:

- Intel 8048, con arquitectura Harvard modificada con programa ROM en el mismo chip, RAM de 64 a 256 bytes e interfaz I/O (entrada/salida).
- Motorola 6805R2.

En la década de los 80's comienza la ruptura de desarrollo y evolución tecnológico entre microprocesadores y microcontroladores. Los microprocesadores han evolucionado buscando la solución al manejo de grandes volúmenes de información, mientras los microcontroladores incorporan unidades funcionales con capacidades superiores de interacción con el medio físico en tiempo real, un mejor desempeño y robustez en aplicaciones industriales.

En los años posteriores aparecen nuevos microcontroladores que son utilizados generalmente para controlar dispositivos periféricos de computadores y algunas aplicaciones de control particulares.

Microcontrolador

Es un dispositivo programable con capacidad de ejecutar operaciones, tareas y procesos a gran velocidad, lo que permite su uso en aplicaciones en tiempo real, como sensores, sistemas remotos, automatismos, sistemas de control en maquinas y aplicaciones industriales.

En síntesis el microcontrolador es una pequeña computadora utilizada para aplicaciones puntuales, esto quiere decir que el microcontrolador debe incluir ciertas unidades fundamentales y comunes en cualquier computadora, estas unidades son:

- CPU, Unidad Central de proceso, los microcontroladores generalmente se basan en el núcleo de un microprocesador como por ejemplo el Intel 8080, Z80, Motorola 6800 entre otros.
- Memoria ROM y RAM, dentro del microcontrolador se construyen bloques de memoria necesaria para almacenar el programa, los datos y demás registros necesarios para implementar el proceso de control. Coexisten dos tipos de memoria:
    - ROM, es el sitio donde se almacena el programa (memoria de programa), consta de unos cuantos KBytes de memoria, suficientes para almacenar el programa en código maquina.

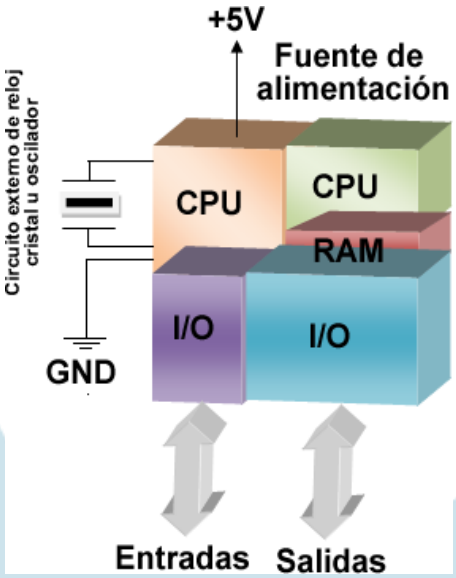
- RAM, en ella se almacenan datos temporales (memoria de datos), usualmente es de poca capacidad, porque las aplicaciones de control, instrumentación y automatización no requieren grandes espacios de almacenamiento temporal.

- Puertos I/O, puertos de entrada / salida, son pines del microcontrolador destinados a comunicar el microcontrolador con el entorno, usualmente los pines pueden tener varias funciones las cuales se configuran por registros internos que varían entre familias de fabricantes y entre la gamma de la familia.

Los microcontroladores representan la gran mayoría de chips de computadoras vendidos en el mundo, de estos más del 50% son microcontroladores básicos y el restante son DSP o Procesador Digital de Señales, una variante de los microcontroladores con gran capacidad de procesamiento de señales, usualmente se encuentran en equipos de audio.

En general hay conciencia de la existencia de los procesadores por los computadores personales y la publicidad que se maneja, pero aunque pasa desapercibido, en nuestro entorno estamos rodeados de mas microcontroladores que de microprocesadores, cada electrodoméstico moderno, equipos de audio, video, telefonía, entretenimiento, automóviles, equipo industrial, instrumentación, etc están compuestos de microcontroladores especializados para cada tarea.

Figura 60. ***Estructura de microcontrolador***[79]



[79] CEKIT, 2002

Clasificación de los Microcontroladores.

Los microcontroladores tienen una clasificación similar a la de los microprocesadores, es decir, se clasifican de acuerdo a la longitud de palabra desde los 4bits, 8bits, 16bits y los últimos que han salido al mercado son los poderosos de 32bits.

Figura 61. ***Microchip 32 bits***[80]

Figura 62. ***Motorola 16, 32 bits***[81]

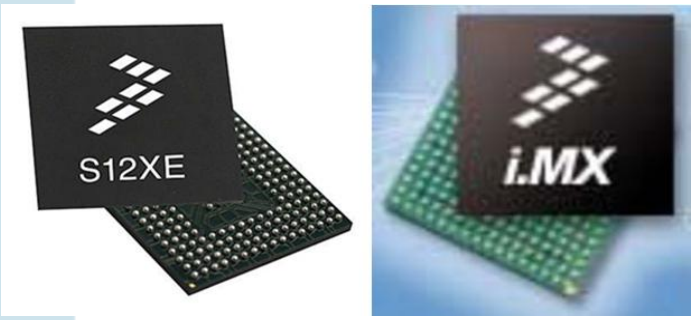


Figura 63. ***AVR-Atmel 32 bits***[82]

[80] Extraído el 10 de Julio de 2009 desde http://www.electronicosonline.com/noticias/images/uploads/PIC-32-Microchip.jpg
[81] Extraído el 10 de Julio de 2009 desde http://img.directindustry.com/images_di/photo-g/microcontroller-193829.jpg

http://img.directindustry.com/images_di/photo-g/microprocessor-193830.jpg
[82] Extraído el 10 de Julio de 2009 desde http://www.anatronic.com/images/Noticias/An2583.jpg

La razón del porque siguen en el mercado microcontroladores de 4 y 8 bits, es simple, todavía existen muchas aplicaciones que los implementan, no tiene sentido aplicar un microcontrolador potente como los de 16bits o 32 bits donde el control lo puede hacer uno de 8 bits mucho mas económico, de hecho, los microcontroladores de 8 bits dominan el mercado actual.

Además de la clasificación en bits, dentro de cada familia, existe una gran variedad de modelos que incorporan unidades funcionales como temporizadores (timer), conversores análogo-digitales (ADC), moduladores de ancho de pulso (PWM), puertos de comunicación I^2C, USART, puertos serial síncronos (SSP), puertos seriales asíncronos (BSSP), puerto paralelo esclavo (PSP), control de motores, soporte de interfaz universal de comunicaciones (USB), soporte controlador de Ethernet, soporte controlador IRDA (comunicaciones infrarrojas) entre otros.

Las anteriores unidades funcionales se incorporan en distintos modelos o gammas dentro de la misma familia, llegando a decenas e incluso cientos, en configuraciones de 8, 18, 20, 28 y 40 pines en DIP, SOIC, SSOP,TQFP, TSOP YJW, que son los empaques o presentaciones disponibles, para el caso de los utilizados en montajes de prueba y aprendizaje se utilizan del tipo DIP (Dual In-line Package).

Hablando de técnicas de fabricación, prácticamente todos los microcontroladores se fabrican utilizando tecnología CMOS (Complementary Metal Oxide Semiconductor), estas técnicas son mas económicas e inmunes a ruido.

Aplicaciones de los Microcontroladores

La fabricación actual de microcontroladores supera en millones de unidades de un mismo modelo en una semana, por su utilización en multitud de aplicaciones y la capacidad que tienen de comunicación entre ellos, se implementan sistemas los cuales cada microcontrolador se encarga de una función especifica y mantiene comunicación y coordinación de tareas con otros microcontroladores que atienden otras funciones mediante un microcontrolador mas potente o un microprocesador, este caso lo podemos apreciar en las computadoras personales en sus periféricos (controlados por microcontroladores) y CPU (microprocesador).

En la actualidad el mercado del microcontrolador se extiende muy rápidamente por la necesidad de incorporar estos chips en diversos productos buscando aumentar sus prestaciones, mejorar la fiabilidad, disminuir costo, consumo y tamaño.

Microcontroladores y el mercado tecnológico

Generalmente a los ojos del común, se escucha hablar de microprocesadores acaparando toda la atención, lo cierto es que se venden más microcontroladores que microprocesadores. Mientras los microprocesadores evolucionan de 32 bits a 64 y múltiples núcleos dejando atrás modelos de 8, 16 y 32 bits y tecnologías consideradas obsoletas, en los microcontroladores el mercado no deja extinguir los primeros modelos de 4 y 8 bits y aceptan de igual manera los últimos de 32 bits.

El sector que más ha impulsado la producción de microcontroladores ha sido el sector automovilístico el cual promovió el desarrollo de microcontroladores de uso especial para adaptarse a las condiciones extremas de vibración, ruido, choques, etc. Conservando fiabilidad para evitar fallos que causen accidentes, estos chips serian adaptados para usos generales. Otros sectores como el de telecomunicaciones, informático, instrumentación e industrial también han favorecido el desarrollo y producción de nuevos microcontroladores capaces de adaptarse al medio. Las ventas de chips están ubicadas en sectores bien definidos:

- Computadores y periféricos
- Aplicaciones de consumo como electrodomésticos, audio, video, entretenimiento, televisión, etc).
- En dispositivos de comunicación (telefonía, radio comunicaciones)
- Aplicaciones industriales en automatización, telemetría, sensores, etc.
- Industria de automotores.
- Aplicaciones militares.

Los recientes microcontroladores de 32 bits están siendo bien recibidos en el mercado, ocupando el interés de áreas como la de comunicaciones, procesos industriales, procesamiento de imágenes y control de dispositivos de almacenamiento masivo.

Microcontroladores vs Microprocesadores

Tabla 13. ***Microcontroladores vs Microprocesadores***[83]

Microcontroladores	Microprocesadores
Los dispositivos genéricos son de 8bits, actualmente hay desarrollo de 16 y 32 bits	Tienen mayores longitudes de palabra (16, 32, 64bits)
Incorpora en una misma pastilla capacidad de memoria y manejo de puertos.	Necesitan chips externos para apoyar las funciones de transferencia y almacenamiento.
La memoria es limitada aunque algunos permiten incorporar KB externos.	Manejo de mayor capacidad de memoria
Tarjetas simples alojadas en espacios pequeños.	Gran volumen (físico) en su adecuación y funcionamiento.
Consumo de miliwatts o unos pocos watts es posible alimentarlo con baterías.	Consumo de potencia en cientos de watts
Dispositivos complejos de comunicación que permiten control en tiempo real.	Manejo de dispositivos periféricos informáticos.
Robustos e inmunes a ruidos industriales	Susceptibles a ruido industrial
Sistemas mínimos	Sistemas informáticos

LECCIÓN 2: SISTEMAS MICROCONTROLADOS.

La posibilidad de manejar señales de entrada y salida, procesar datos a gran velocidad, tomar decisiones en tiempo real, bajo consumo y ser robustos e inmunes al ruido convierten al microcontrolador en uno de los componentes mas utilizados, versátiles y vendidos en la actualidad.

Dispositivos de entrada

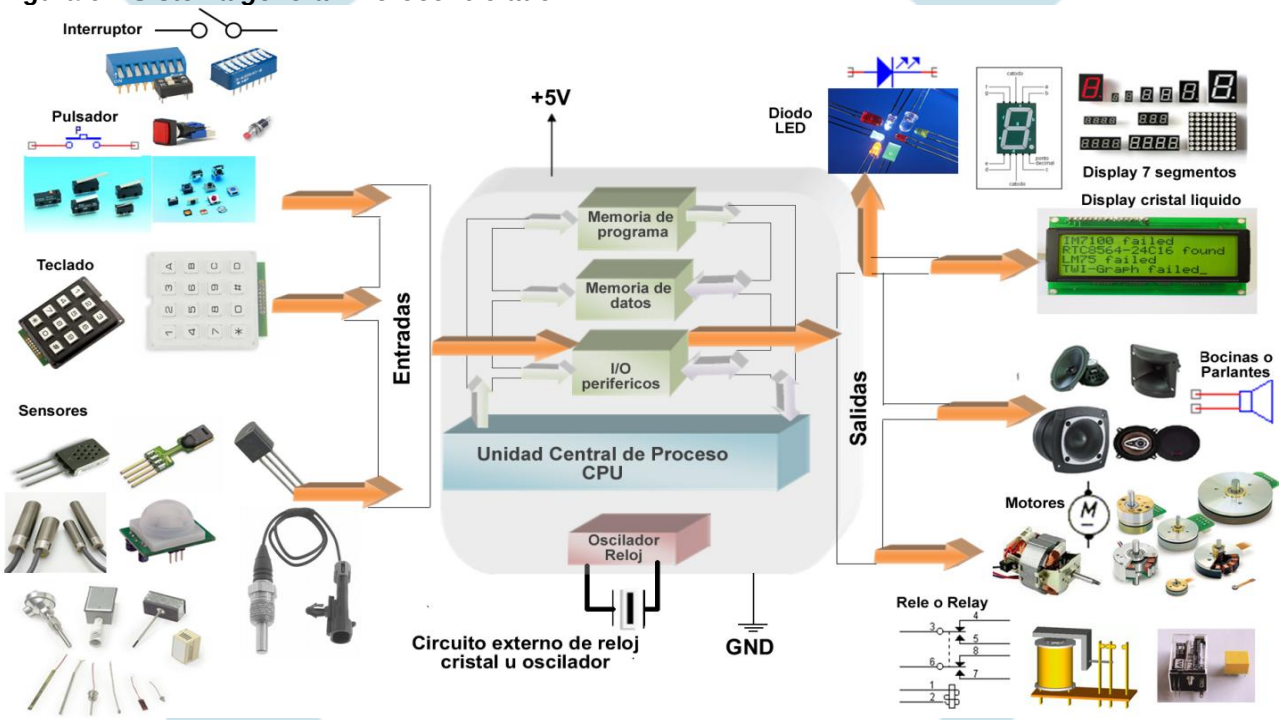
Conforman este grupo todos los dispositivos capaces de cambiar su estado como respuesta a un evento que los afecte y generar una señal que pueda ser utilizada por el microcontrolador para ejecutar una operación de control o de decisión.

[83] CEKIT, 2002

Dentro de este grupo se encuentran, interruptores (switch), pulsadores, teclados, transductores y toda la gamma de sensores (temperatura, presión, humedad, luz, ph). Cada uno de ellos requiere ser adecuado eléctricamente a nivel de hardware, para que produzca los dos estados estables necesarios (uno y cero) a nivel de software. En los interruptores, pulsadores y teclado se busca generalmente que la pulsación genere "uno lógico", es decir, exista un voltaje de 5 Voltios, la no pulsación debe mantener a "cero lógico" la entrada, aunque si sucede lo contrario es fácilmente adecuado a nivel de software, utilizando lógica negativa.

Para los transductores y la mayoría de sensores, se requiere convertir la señal eléctrica producida por ellos a una señal digital, para lo cual se utilizan conversores ADC (analog-to-Digital Conversion), externos o incorporados en el microcontrolador.

Figura 64. ***Sistema general microcontrolador***[84]



Dispositivos de salida

Los dispositivos de salida se encargan de recibir las señales del microcontrolador y reproducir acciones particulares, entre los dispositivos mas usuales están los visuales, auditivos y actuadores.

[84] Extraído el 10 de Julio de 2009 desde http://img.directindustry.es/images_di/photo-g/boton-pulsador-282573.jpg

- Los dispositivos visuales mas comunes son los LED (Light-Emmiting Diode – Diodo emisor de luz) en forma individual o agrupados en los denominados “siete segmentos” y “matrices de LEDs”. También son utilizadas lámparas incandescentes desde bulbos de 1,2 Voltios hasta de cientos de Watts mediante relevos o dispositivos de estado solido (SCR - TRIACs).
- LDC (Liquid Cristal Display – Display de Cristal Liquido), los cuales se fabrican en presentaciones de 1x16 (1 línea, 16 caracteres), 2x16 y gráficos.
- Audibles, como parlantes o bocinas, zumbador (buzzer), que funcionan como emisores de sonidos de alarma, indicadores de actividad y de ingreso de información.
- Actuadores:
    - Rele o Relay, dispositivo electromecánico, ampliamente utilizado para el accionamiento de grandes cargas, es decir, dispositivos de gran consumo de corriente o voltaje, son accionados por etapas denominadas “driver”, construidas con semiconductores como los transistores.
    - Motores continuos, utilizando dispositivos de estado solido o relevos para su accionamiento y puentes “H” para el control de giro.
    - Motores alternos, generalmente activados con relevos o dispositivos de cuatro capas como SCR o TRIACs.
    - Electroválvulas, dispositivo electromecánico, de apertura y cierre para el paso de líquidos o gases

Selección del microcontrolador

La elección de un microcontrolador es una acción que requiere considerar una serie de variables y parámetros antes de proponerse a implementar el diseño, el desarrollador de proyectos con microcontroladores se encontrara con muchos fabricantes, con familias de microcontroladores, cada una de ellas provee una amplia gamma y variedad de dispositivos, clasificados por sus características particulares (memoria, puertos, interrupciones, temporizadores, etc), cantidad de pines y tipos de empaques. Esta gran variedad hace de la selección del microcontrolador una tarea que requiere mucha práctica que se consigue con el trabajo individual sobre los microcontroladores de uso general de cada familia.

Consideraciones generales:

- La documentación existente sobre el producto, manuales, datasheet, textos especializados, ejemplos de desarrollo.

- Disponibilidad en el mercado, que se consiga localmente o en ciudades cercanas a precios cómodos.
- Herramientas de desarrollo, constituidas por ensambladores, emuladores, simuladores, programadores y entrenadores.
- Costo del producto, es de considerar que algunos fabricantes por mantenerse en el mercado disminuyen sus precios pero los soportes son insuficientes.
- Las características del microcontrolador como memoria y su capacidad, temporizadores, interrupciones, velocidad de reloj, ADC, PWM, etc. que dependen del diseño de hardware, software y tipo de empaquetado con el cual se estima desarrollar el producto final.

Consideraciones de Aplicación

La selección de un microcontrolador para una aplicación es una tarea que requiere experiencia y conocimiento de la familia de dispositivos sobre la que se piensa implementar, teniendo presente las consideraciones generales antes mencionadas y las consideraciones de aplicación que a continuación se exponen:

- **Entradas y salidas**, en el proceso de diseño se debe establecer la cantidad y tipo de entradas y la cantidad y tipo de salidas, junto con el microcontrolador seleccionado, para lo cual se emplean diagramas esquemáticos simples para identificar rápidamente las entrada, salidas y dispositivos de hardware necesarios. Se procede al análisis del diagrama donde se ratifica el microcontrolador propuesto o el cambio por otro mas adecuado.

- **Memoria**, para establecer los requerimientos de memoria que debe cumplir el microcontrolador, se separa en:
  - Memoria de programa – ROM, es la memoria que se utiliza para el almacenamiento del programa, su tamaño es determinado por la cantidad de Bytes que ocupa el programa en lenguaje maquina.
  - Memoria de datos – RAM, es la memoria que contiene los registros de propósito especial y los registros de propósito general (utilizados por el programador para definir variables y dinamizar el flujo del programa).
  - Memoria no volátil, de tipo EEPROM y FLASHRAM, son pocas localidades de memoria que almacenan datos de calibración, parámetros iníciales, estado anterior o numero de serie.

- **velocidad del reloj**, influye en la velocidad de procesamiento y capacidad de trabajo en tiempo real, para aplicaciones generales se utilizan velocidades menores o iguales a 4MHz, muchas de las familias soportan varias decenas de MegaHertz en el circuito de reloj.

- **Datos y procesamiento**, se recomienda seleccionar el microcontrolador con menor ancho de palabra que satisfaga las necesidades de aplicación, generalmente se manejan datos de ancho cercano o igual a 8 bits (1 Byte), con lo que se prefiere los microcontroladores de 8bits, aunque todavía se utilizan microcontroladores de bus de datos de 4 bits. En aplicaciones que requieren espacio de direccionamiento grande, velocidad de proceso y longitudes de palabra mayores a 8bits, se elijen microcontroladores de 16 o 32 bits.

- **Precisión**, si la necesidad radica en la precisión de los datos a procesar puede que un microcontrolador de 8 bits no sea suficiente, indicando que se debe optar por uno de 16bits, 32bits o uno de coma flotante.

- **Consumo**, los microcontroladores tienen consumos muy bajos pudiendo ser alimentados por baterías, el consumo aumenta dependiendo de los componentes periféricos al microcontrolador, si el consumo es un elemento crítico de diseño, es recomendable utilizar dispositivo que permitan “estado de bajo consumo” y utilizar celdas de iones de litio como fuente de alimentación.

- **Diseño de la placa de circuito**, el diseño de la placa esta condicionada por el tipo de microcontrolador utilizado. Dependiendo de las entradas y salidas y los periféricos conectados a ellas como teclados, Display 7 segmentos, LCD, Motores CC, etc, hacen que un microcontrolador sencillo con pocos pines, incremente el costo de la implementación. Sin embargo utilizar un microcontrolador complejo que incorpore unidades funcionales y capacidades como manejadores de display LCD, PWM, comunicaciones, etc. Necesita un software complejo para controlar sus capacidades internas y la multiplexación de sus pines.

LECCIÓN 3: DIFERENCIAS ENTRE SISTEMAS BASADOS EN MICROPROCESADORES Y MICROCONTROLADORES.

Entre las tecnologías de microprocesadores y microcontroladores existen varias diferencias. Cada tecnología tiene ventajas y desventajas donde su uso depende de la aplicación particular, por lo que es importante en esta lección estudiar y aclarar estas diferencias.

CPU

La CPU de un microprocesador es más simple, sus instrucciones están orientadas al manejo y operación de las líneas de entrada y salida, generalmente utilizan un conjunto reducido de instrucciones.

RAM

La capacidad de direccionamiento de memoria en los microprocesadores es mucho más elaborada pudiendo acceder a grandes bancos de memoria acordes a los requerimientos del sistema. En un microcontrolador la memoria RAM o memoria de datos, que se incorpora dentro del chip es de baja capacidad, puesto que las aplicaciones de control no necesitan almacenar grandes cantidades de datos temporales.

ROM

Los microprocesadores y su sistema de chips periféricos permiten el acceso a dispositivos de almacenamiento externo de gran capacidad ROM, RAM, FLASH, Magnéticos y ópticos de distintas tecnologías. La memoria ROM o memoria de programa en los microcontroladores es de capacidad limitada y esta alojada dentro del mismo chip, es utilizada para el almacenamiento del programa en lenguaje maquina, se distinguen varios tipos de memoria ROM:

- **EPROM**, Las memorias EPROM en los microcontroladores son memorias utilizadas para el desarrollo de prototipos, esta memoria tienen la capacidad de ser programada y borrada por luz ultravioleta cientos de veces, los chips que las incorporan tienen una ventana traslucida que permite su borrado por exposición a luz ultravioleta durante un periodo de 20 a 30 minutos. Para que el programa no se borre se debe cubrir la ventana, la exposición a luz día, solar, fluorescente e incandescente con la ventana descubierta hace que en cuestión de horas, días o semanas se borre el programa.

- **OTP** (One Time Programable), son memorias programables una sola vez. Cuando un prototipo con microcontroladores se termina y ha sido completamente probado además se quiere su producción en masa, se elije este tipo de memoria por su bajo costo.

- **EEPROM** o **E^2PROM**, estas memorias en los microcontroladores se caracterizan por su capacidad de ser borradas eléctricamente, este proceso se hace en unas pocas fracciones de segundo, donde se escribe o borra

una celda a la vez. con respecto a las **EPROM** no necesitan dispositivos especiales para ser borradas y pueden repetir el proceso de borrado y reprogramación muchas mas veces.

- **FLASH**, este tipo de memoria es una variedad mas sofisticada de la **E^{2}PROM**. Los microcontroladores que la implementan, se pueden borrar eléctricamente, con pulsos de voltaje mas bajos, la velocidad de escritura o borrado es mucho más rápida pues toma varias celdas de almacenamiento a la vez, soportan millones de veces de programación y borrado, su consumo es menor con respecto a sus predecesoras.

Implementación

La implementación con microprocesadores involucra placas impresas multicapa, para permitir la interconexión de una compleja red de circuitos y chips individuales relacionados con la codificación y decodificación, memoria, almacenamiento y periféricos, convirtiéndolo en algo tan complejo como la mainboard de un computador personal.

Con los microcontroladores la implementación es mas sencilla, usualmente requiere solo una capa en el circuito impreso, para aplicaciones simples el circuito involucra el reloj generalmente conformado por un cristal de cuarzo y un par de condensadores, unos pocos LEDs, resistencias, pulsadores, filtros de alimentación y conexión de baterías.

Tipo de sistema, abierto y/o cerrado

Un ejemplo de sistema abierto es el microprocesador el cual dispone de líneas o pines externos accesibles al diseñador, estos comunican el hardware exterior con los buses de dirección, datos y control en el microprocesador.

La arquitectura de sistema cerrado es propia de los microcontroladores, generalmente no permiten que el diseñador tenga acceso a los buses de dirección, datos y control internos de la CPU, no obstante, muchos microcontroladores permiten acceso a los buses a través de los puertos de entrada / salida utilizando señales de sincronización que permiten la conexión de chips RAM o ROM para expandir su capacidad de memoria.

Velocidad de operación

Actualmente los microprocesadores tienen velocidades de operación del orden de los GigaHertz con varios núcleos, en los microcontroladores la velocidad de operación es mucho más lenta, de hasta varias decenas de MegaHertz por encima de los 50MHz, pero es suficiente para controlar sistemas en tiempo real. El circuito responsable de la velocidad del reloj que esta estrechamente relacionado con la velocidad de operación puede ser interno o externo, cuando es externo los pulsos o ciclos por segundo (Hertz) son producidos por un oscilador digital, cuando es interno puede ajustarse el numero de ciclos por segundo utilizando varios dispositivos y configuraciones conectadas a los pines del oscilador en el chip microcontrolador, entre los dispositivos y configuraciones mas usuales están, la red interna o externa de resistencia-condensador

Sistemas de desarrollo

Para los microprocesadores se considera como sistema de desarrollo el hardware conectado al micro en conjunto con los paquetes de software como, intérpretes de comandos, compiladores, programas de bajo a alto nivel, encargados de escribir, ensamblar, depurar y ejecutar los programas en lenguaje maquina.

En los microcontroladores se requiere un sistema de desarrollo diferente para cada familia de microcontroladores, estos sistemas de desarrollo se componen de:

- **Software**, con capacidad para editar, ensamblar, compilar y simular los estados y comportamiento del microcontrolador.
- **Hardware**, para programar o “quemar” el microcontrolador, es decir, gravar en la memoria del microcontrolador el programa.

Diversidad de periféricos

Los sistemas basados con microprocesadores incorporan una amplia variedad de periféricos que satisfacen necesidades donde la capacidad y potencia de un microcontrolador no es suficiente y donde la implementación es muy compleja. Se hace referencia al procesamiento de datos, que tienen que ver con audio, video,

información, bases de datos, comunicaciones, entretenimiento, aplicaciones industriales y espaciales entre otras.

Para proyectos y aplicaciones con una complejidad manejable dentro de las capacidades de los microcontroladores actuales, con la movilidad y portabilidad que los caracteriza, existe uno o varios microcontroladores con características singulares que permiten la implementación con relativa simplicidad.

- Si se requiere controlar fenómenos de naturaleza análoga como temperatura, humedad, voltaje etc, se utilizan dispositivos que incorporen un convertidor análogo-digital.
- Cuando se requiere medir o generar periodos de tiempo, tonos o frecuencias, se busca tener en el chip, temporizadores programables (timers).
- La interacción con el entorno puede requerir respuestas en tiempo real a eventos que deben ser monitoreados por el microcontrolador a través de sus pines por lo que se requiere que el chip contenga fuentes de “interrupción”.
- Las necesidades de comunicación con otros microcontroladores o con un microprocesador se satisfacen con dispositivos que incorporen soporte para comunicaciones seriales (RS-232), paralelas, de red etc.
- Para controlar actuadores como motores o cargas resistivas debe tener incorporado un modulador de ancho de pulso o PWM.

Conclusión

En conclusión, un sistema basado en microcontroladores tiene ciertas ventajas notables con respecto al los sistemas basados en microprocesadores:

- El circuito impreso y su montaje es más sencillo pues el microcontrolador incorpora muchos de los componentes internamente, reduciendo también el costo.
- Al integrarse la mayoría de componente sensible al ruido en un solo chip, el microcontrolador es prácticamente inmune al ruido lo que lo hace ideal para aplicaciones en casi cualquier campo de desarrollo.
- El tiempo de desarrollo de un sistema basado en microcontroladores se reduce notablemente.

- Sin embargo, cuando la complejidad de un sistema supera las prestaciones y características de un microcontrolador como capacidad de memoria, longitud de palabra, capacidad de manejo de datos, velocidad de proceso, numero de pines, puertos de I/O, etc. Se opta por un sistema basado en microprocesadores.

LECCIÓN 4: ARQUITECTURA INTERNA.

La arquitectura en los microcontroladores se refiere a la forma como la CPU accede a la memoria y a la cantidad o set de instrucciones de cada familia.

Arquitectura Von Neumann

Con esta arquitectura se hace el diseño conceptual y la estructura operacional de la mayoría de microprocesadores y de computadoras de uso personal que se utilizan desde su aparición a la fecha. Esta arquitectura esta basada en el concepto de programa almacenado propuesto por el matemático Von Neumann y propuesto por Jhon Presper Ecker, Jhon William Mauchly, Arthur Burks, entre otros en el periodo de construcción de la ENIAC. En la arquitectura Von Neumann la CPU se conecta a una memoria principal única generalmente del tipo RAM, donde se almacenan los datos y el programa, accediendo a través de un sistema de buses único, como son el bus de dirección, control y datos.

El ancho del bus que comunica la memoria con la CPU determina el tamaño de la unidad de datos o instrucciones, un microprocesador de 8 bits con bus de 8 bits tendrá que manejar datos o instrucciones de 8 bits de longitud. Para el acceso a datos o instrucciones de más de 8bits tendrá que realizar más de un acceso a la memoria.

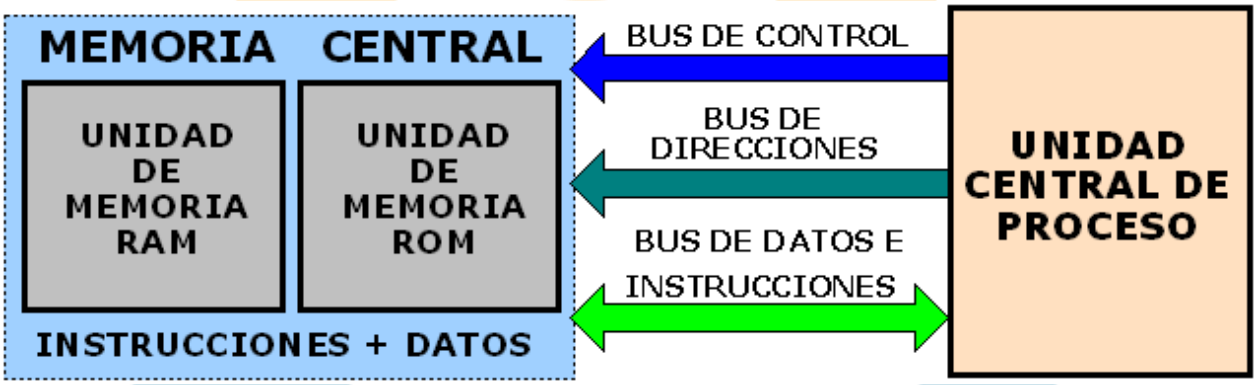
La arquitectura Von Neumann tiene varias limitaciones:

- La longitud de las instrucciones están limitadas por el bus de datos, lo que hace que el ejecutar una instrucción compleja requiera varios accesos a memoria.

- El microprocesador es mas lento en su respuesta, la velocidad de operación se afecta por tener un único bus para datos e instrucciones lo que impide

acceder a la memoria de datos y de instrucciones simultáneamente, es decir, no permite superponer tiempos de acceso.

Figura 65. ***Arquitectura Von Neumann***[85]



Arquitectura Harvard

El termino proviene de la Harvard Mark I, la cual almacenaba los datos en cintas perforadas y las instrucciones mediante interruptores, la arquitectura Harvard se caracteriza por tener separados los bloques de memoria de datos e instrucciones y acceder a ellos por buses independientes de dirección, datos y control. La independencia de buses permite tener accesos simultáneos e independientes a la memoria de datos e instrucciones, el contenido y longitud de las localidades de memoria pueden ser distintos para los datos e instrucciones, esto permite una optimización en el uso de la memoria.

Los diseñadores aprovechan este concepto donde la memoria de datos puede por ejemplo de 8bits, mientras la memoria de instrucciones se adecua a la longitud de las instrucciones buscando que cada instrucción se aloje en una posición de memoria, con lo que la ejecución de una instrucción puede hacerse en un solo ciclo maquina, permitiendo también la superposición de tiempos de acceso, por tanto en el mismo lapso que busca y ejecuta una instrucción puede estar realizando una acción de lectura o escritura en la memoria de datos. Esta característica es explotada por microprocesadores y microcontroladores con conjunto de instrucciones reducido (RISC).

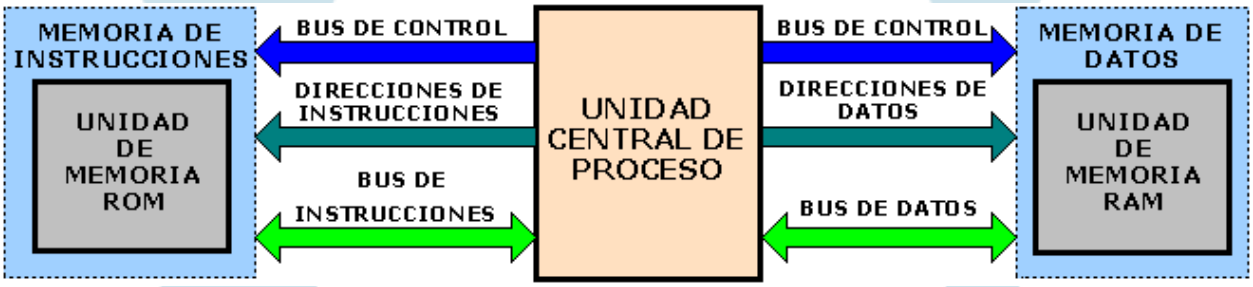
[85] Extraído el 10 de Julio de 2009 desde http://perso.wanadoo.es/pictob/microprg.htm

La arquitectura Harvard tiene ventajas significativas con respecto a la arquitectura Von Neumann, las más significativitas son:

- El tamaño de las instrucciones no esta relacionado con el tamaño de los datos permitiendo optimizar la memoria haciendo que cada instrucción ocupe una única posición de memoria, esto hace que la longitud de programa puede ser menor.

- La posibilidad de superponer tiempos de acceso, es decir, poder acceder a la memoria de programa y a la memoria de datos en el mismo ciclo maquina, esta característica y la anterior permiten una velocidad de operación mas alta.

Figura 66. ***Arquitectura Harvard***[86]



Arquitectura CISC

Complexa Instrucción Set Computing o Computadores de juego de instrucciones complejo, la mayoría de CPUs utilizada en microcontroladores están basados en esta arquitectura, dentro de las características mas relevantes están:

- Un gran numero de instrucciones de longitud variable.
- Generalmente más de 80 instrucciones en su set de instrucciones.
- Instrucciones muy sofisticadas y potentes, que actúan como macros.
- Instrucciones que requieren un numero de múltiplos de ciclo maquina.
- Modos de direccionamiento múltiple.
- Pequeño numero de registros de trabajo de propósito general.

Esta arquitectura dificulta el paralelismo entre instrucciones, en la actualidad los sistemas con CISC de alto rendimiento implementan sistemas que convierten

[86] Extraído el 10 de Julio de 2009 desde http://perso.wanadoo.es/pictob/microcr.htm

instrucciones complejas en simples del tipo RISC, denominadas microinstrucciones.

Arquitectura RISC

Reduced Instruction Set Computer, Computadores con set de instrucciones reducido, esta arquitectura se implementa con gran éxito actualmente en microcontroladores PIC, como características principales están:

- Conjunto de instrucciones reducido, generalmente menor o igual a 120.
- Típicamente un numero reducido de modos de direccionamiento, que son las formas como el procesador utiliza la memoria, básicamente cuatro (4) modos.
- El procesador tiene un numero superior de registros de propósito general típicamente de 32 registros.
- Todas las instrucciones se ejecutan típicamente en un solo ciclo maquina, compuesto de unos pocos ciclos de reloj, generalmente cuatro ciclos de reloj.
- La longitud de las instrucciones tiende a ser fija y pequeña entre 12 y 32 bits con un número reducido de formatos.

Arquitectura SISC

Specific Instruction Set Computer, computado con juego de instrucciones específico, los microcontroladores que son destinados a aplicaciones muy concretas tienen un juego de instrucciones además de reducido y “especifico”, es decir que se adaptan a una aplicación predefinida.

Núcleo del microcontrolador

El núcleo se refiere a las características fundamentales que son requeridas para que el “micro” realice las operaciones básicas, entre ellas están:

- **CPU**: Unidad Central de Proceso, es la responsable de tomar las instrucciones desde la memoria de programa, ejecutarlas y así controlar la operación de todo el sistema. Esta conformada por:

    - **ALU**, Unidad Aritmético Lógica encargada de interactuar con la memoria de datos en las operaciones aritméticas y lógicas.
    - **UC**, Unidad de control, busca las instrucciones en la memoria de programa, las decodifica y las ejecuta.
    - **Matriz de registros**, los conforman por registros visibles al usuario como el acumulador o registro de trabajo, temporizador, entre otros y los registros de control y estado como el registro de estado, contador de programa, registro de interrupciones entre otros.

- **Mapa de memoria**, compuesto por la disposición en bloque o bancos de memoria. La arquitectura Von Neumann generalmente maneja un solo bloque de memoria mientras que la arquitectura Harvard maneja hasta cuatro bancos de memoria, que contienen las funciones particulares de algunas localidades y los registros de usuario.

- **Pila**, conformada por una pequeña porción de memoria ubicada generalmente en la memoria de programa en la arquitectura Harvard, esta determina la profundidad o la cantidad de llamados sucesivos a subrutinas. Por ejemplo una pila de 2 niveles indica que pueden realizarse un llamado a una subrutina dentro de otra subrutina, superar la profundidad o niveles de pila implica una perdida de secuencia en el programa causando errores, esto es llamado “desbordamiento de pila”.

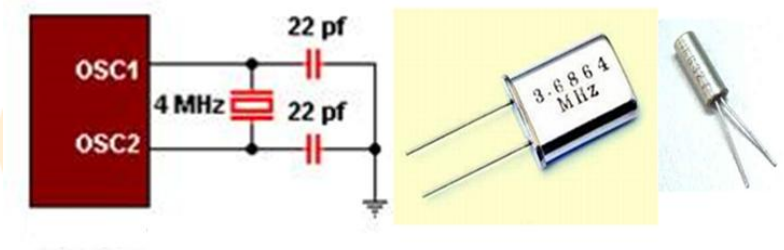
- **Circuito Oscilador**: es un circuito encargado de generar pulsos o señal de reloj necesaria para que el microcontrolador sincronice y ejecute las instrucciones y funciones adecuadamente en los periféricos. Los circuitos osciladores mas comunes están basados en componentes que determinan la frecuencia entre ellos podemos mencionar los siguientes:

    - **INTRC**: Es una red de resistencia – condensador interna, es la mas económica y permite tener un par de pines extra para I/O, no todos los microcontroladores lo implementan, es necesario recurrir a las hojas de especificaciones para comprobar su disponibilidad.

    - **RC**: Red resistencia – condensador externo, es una solución económica pero inestable y poco precisa.

    - **Cristal de cuarzo**: son redes compuestas por cristales de cuarzo, generalmente con un par de condensadores conectados entre tierra (GND) y cada terminal del cristal y el microcontrolador.

Figura 67. ***Oscilador con Cristal de cuarzo***[87]



- **LP**: cristal de baja potencia, tiene consumo bajo de corriente, con frecuencias entre 32KHz y 200KHz.
- **XT**: cristal / Resonador con un consumo mas elevado que el anterior y frecuencias entre 100KHz y 4MHz.
- **HS**: cristal / Resonador de alta frecuencia permite trabajar a gran velocidad pero también incrementa el consumo, tiene frecuencias mayores a 8MHz.

- **Sistema de Reanudación o Reset**: es usado para llevar al microcontrolador a un estado conocido, el vector de reset se direcciona generalmente a la localidad de memoria 0000H. En este estado se establecen condiciones iníciales estables con las que siempre inicia el sistema y con las que se garantiza un buen funcionamiento de todas las tareas.
- **Conjunto de interrupciones**, generalmente los microcontroladores implementan varias fuentes de interrupción las cuales son atendidas utilizando vectores que apuntan a localidades específicas dentro de la memoria de programa.
- **Conjunto de instrucciones**, cada instrucción en lenguaje ensamblador se divide en un código de operación “OPCODE”, que especifica el tipo de instrucción y uno o mas operandos que especifican la operación de la instrucción, por ejemplo la instrucción “MOVLW 07H” implementada en un PIC con arquitectura Harvard de 13 bits en longitud de bus de instrucciones se divide como se ilustra a continuación:

Tabla 14. ***Instrucciones y assembler***[88]

	bits	
Longitud	13 . . . 8	7 . . .0
Instrucción	OPCODE	K (literal)
mnemónico	MOVLW	07H
Hexadecimal	30	07

[87] Extraído el 10 de Julio de 2009 desde http://grupodcti.blogspot.com/2008_10_01_archive.html
[88] CEKIT, 2002

Periféricos

Son los elementos, unidades funcionales, módulos o soportes que el microcontrolador tiene para interactuar con el exterior, son los que hacen la diferencia entre microprocesadores y microcontroladores. Los periféricos se encargan de la comunicación entre el "micro" y el mundo exterior mediante los puertos de I/O, manejadores de LCD, conversores ADC, PWM, etc. En general existen muchos tipos de periféricos entre los que podríamos mencionar los siguientes:

- **Pines o línea de entrada / salida (I/O) de propósito general**, estos permiten al microcontrolador comunicarse con el mundo exterior enviando y recibiendo señales, son utilizados para supervisar y controlar dispositivos conectados al microcontrolador, muchos de estos pines son multiplexados para que tengan funciones alternas, es decir, pueden funcionar como pines de entrada, de salida o incorporar funciones adicionales.
- **Temporizadores**, los microcontroladores implementan uno o varios temporizadores que establecen bases de tiempo confiables, intervalos, tiempo entre eventos, etc.
- **Módulos de conversión**, estos módulos permiten convertir señales y almacenarlas en el "micro" para su posterior procesamiento y toma de decisiones o enviar señales al exterior adecuadas para cierto tipo de dispositivos, entre estos están los módulos ADC y PWM.
- **Módulos de comunicación**, como su nombre lo indica son módulos que permiten la comunicación entre el micro y el exterior y/o viceversa, entre los mas usuales están, la comunicación serial síncrona, USART y comunicación por puerto paralelo.
- **Comparadores**, estos módulos se implementan para comparar señales análogas y servir como referencia para la toma de decisiones y control de procesos desde el microcontrolador.

Características especiales

Las características especiales ayudan a disminuir costos de implementación, sencillez en el montaje, funcionalidad, flexibilidad y convierten el sistema basado en microcontroladores en un sistema robusto aplicado a casi cualquier ambiente de trabajo, entre las características mas sobresalientes están:

- **Bits de configuración de dispositivo**, estos bits permiten al usuario personalizar el modo de trabajo del microcontrolador, como tipo de oscilador, encriptación o protección de programa entre otros.

- **Sistema de protección e inicio**, muchos microcontroladores cuentan con circuitos que vigilan el estado de la alimentación eléctrica generando un auto-reset en caso de encontrar variaciones o cambios drástico en la fuente de poder, también implementan circuitos que direccionan el vector de reset para garantizar el comienzo del programa en la localidad de memoria preestablecida.
- **Temporizadores al encendido o en funcionamiento**, los temporizadores al encendido generan tiempos de espera para permitir que se estabilice la señal de alimentación y los pulsos generados por el reloj, durante el funcionamiento pueden generar eventos de “reset” en periodos definidos evitando bloqueos, situaciones no deseadas o fallas en el proceso.
- **Modo de bajo consumo**, generalmente los microcontroladores dispones de instrucciones que permiten entrar en un estado de bajo consumo o “sleep” apropiado para situaciones donde la duración de las baterías es un punto critico.
- **Oscilador interno**, el circuito oscilador requiere típicamente dos (2) pines del microcontrolador para conectar los componentes que generan la base de tiempos para el pulso de reloj, en ocasiones estos dos pines son necesarios para utilizarlos como control de algún dispositivo o como líneas I/O, como característica adicional y para tener un par de pines extra se implementan en algunos modelos de microcontroladores osciladores internos.
- **Programación serial dentro del circuito**, actualmente la mayoría de modelos de microcontroladores permiten ser programas dentro del mismo circuito de aplicación.

LECCIÓN 5: FAMILIAS DE MICROCONTROLADORES.

Existen muchas familias fabricantes de microcontroladores, entre las más comunes están:

Atmel (AVR), Hitachi (H8), Intel de 8 bits (8XC42, MCS51, 8xC251) o Intel de 16 bits (MCS96, MXS296), National Semiconductor (COP8), Microchip, Motorola de 8 bits (68HC05, 68HC08, 68HC11) o de 16 bits (68HC12, 68HC16) o de 32 bits (683xx), NEC (78K), Texas Instruments (TMS370) y Zilog (Z8, Z86E02).

Sin embargo en nuestro medio se destacan sólo dos de ellas: la empresa Motorola y la empresa Microchip.

La familia Motorola Freescale

Esta familia, desarrollada por la casa Motorola, se divide en las siguientes subfamilias:

- **Familia HC05:**
  Esta familia es una de las más utilizadas en la gran mayoría de aplicaciones por su versatilidad de recursos y fácil programación. Sin embargo, presenta una propiedad con mayor importancia y es su compatibilidad con familias más avanzadas, por ejemplo con la familia HC08, lo que permite hacer migración de diseños hacia dispositivos de más alto rendimiento de una manera muy fácil y rápida. Sus principales ventajas son:

  - Un timer robusto
  - Memoria EEprom de 256
  - Memoria de programa desde 4k hasta 32 k
  - Memoria RAM desde 176 hasta 528 bytes.
  - Ocho canales A/D
  - Comunicación serial síncrona y asíncrona.

(VESGA, 2007).

Tabla 15. *Familia Motorola*[89]

Familias 68HC05-C y 68HC05-D	• Timer de 16 bits, acompañado de modulo de captura y comparación • Memoria de programa desde 4k a 16k • Comunicación SCI (75Hz-131KHz) • Interfaz SPI 4 hilos • Watchdog
Familias 68HC05-J y 68HC115-K	• Bajo costo • Encapsulado de 16 y 20 pines • Memoria de programa 0,5K a 2K • Memoria RAM 32 hasta 128 bytes
Familia 68HC05-P	• Controlador para manejo de LCD • Memoria de programa 0,6K hasta 24 K • Memoria RAM 32 hasta 768 bytes • Timer de 16 bits con módulos de captura y comparación. • Comunicación seria síncrona y asíncrona.

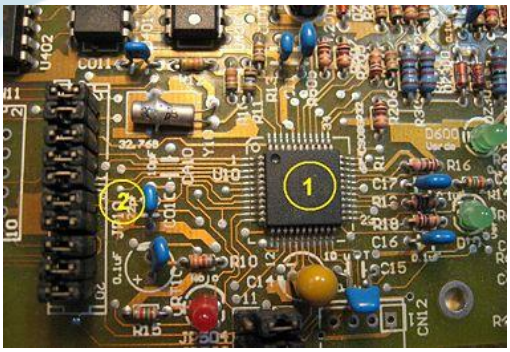
[89] VESGA, 2007

- **Familia HC08**

  Son microcontroladores de propósito general. Cada miembro de esta familia cuenta con diferentes periféricos internos, pero con una CPU común que permite migrar aplicaciones entre ellos, facilitando con ello el diseño.

  El 68HC08 es un microcontrolador de 8 bits y arquitectura Von Neumann, con un solo bloque de memoria. Es conocido también simplemente por HC08.
  Entre los periféricos internos con los que cuentan estos microcontroladores, están: conversores analógicos-digitales, módulo de control de tiempos y sistemas de comunicación como SPI, I²C, USB o SSCI entre otros.

Figura 68. ***Microcontrolador Motorola Freescale***[90]



- **Familia 68HC11** (abreviado HC11 o 6811)

  Es una familia de microcontroladores de Motorola, derivada del microprocesador Motorola 6800. Los microcontroladores 68HC11 son más potentes y costosos que los de la familia 68HC08 y se utilizan en múltiples dispositivos empotrados.

  Siguen la arquitectura Von Newman. Internamente, el conjunto de instrucciones de la familia 68HC11 es compatible con la de la mayoría de sus predecesores. La familia 68HC11 emplea instrucciones de longitud variable y se considera que emplea una arquitectura CISC. Tienen dos acumuladores de ocho bits (A y B), dos registros índice de 16 bits (X e Y), un registro de banderas, un puntero de pila y un contador de programa.

  Los 68HC11 tienen cinco puertos externos (A, B, C, D y E), cada uno de ocho bits excepto el E, que es generalmente de seis bits. El puerto A se emplea en captura de eventos, salida comparada, acumulador de pulsos y

[90] Extraído el 10 de Julio de 2009 desde http://es.wikipedia.org/wiki/Archivo:Microcontrolador_HC08_en_Impreso_Comentado_V1.JPG

otras funciones de reloj. El puerto D para E/S serie y el puerto E como conversor analógico-digital.

La familia 68HC11 puede funcionar tanto con memoria interna o externa. En caso de emplear memoria externa, los puertos B y C funcionan como bus de datos y direcciones respectivamente. Últimos Microcontroladores de la Familia Freescale.

- Familia "ultra bajo costo" RS08, con sus modelos **MC9RS08KA2 y MC9RS08KA1.**
- Dispositivos de la familia HC908 con capacidades de memoria hasta de 16k, entre los cuales están**. MC68HC908QTxA/QYxA, MC68HC908QLxx, MC68HC908QC16xx, MC68HC908GRxxA, MC68HC908QB8 y MC68HC908JL16.**
- Existen novedades en la familia **HC9S08,** de bajo consumo con los dispositivos **MC9S08QG8/4 y MC9S08AWxx.**

Figura 69. *HC08*[91]

La literatura existente que trata el tema de los microcontroladores Motorola Freescale, es muy extensa, generalmente en ingles, este material esta basado en una fuente en español, se podría asegurar que casi la única en su estilo, como texto complementario y especializado en este tema esta el libro del Ingeniero Juan Carlos Vesga titulado "Microcontroladores Motorola Freescale: programación, familias y sus distintas aplicaciones en la industria" por lo que se sugiere al estudiante su consecución para complementar estos conocimientos.

La familia PIC

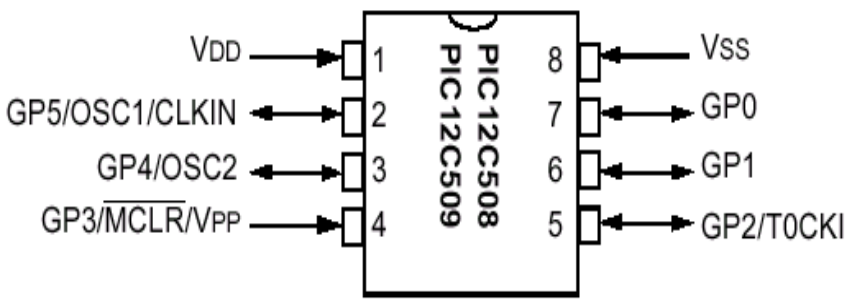
Esta familia, desarrollada por la casa Microchip, se divide en varias gamas: enana, baja, media y alta.

[91] Extraído el 10 de Julio de 2009 desde http://image.absoluteastronomy.com/images/topicimages/f/fr/freescale_68hc11.gif

Las principales diferencias entre estas gamas radica en el número de instrucciones y su longitud, el número de puertos y funciones, lo cual se refleja en el encapsulado, la complejidad interna y de programación, y en el número de aplicaciones.

- **Gama enana**
  Su principal característica es su reducido tamaño, al disponer todos sus componentes de 8 pines. Se alimentan con un voltaje de corriente continua comprendido entre 2,5 V y 5,5 V, y consumen menos de 2 mA cuando trabajan a 5 V y 4 MHz. El formato de sus instrucciones puede ser de 12 o de 14 bits y su repertorio es de 33 o 35 instrucciones, respectivamente. En la Figura se muestra el diagrama de pines de uno de estos PIC.

Figura 70. ***PIC gamma baja o enana PIC12Cxxx***[92]



Aunque los PIC enanos sólo tienen 8 pines, pueden destinar hasta 6 como líneas de E/S para los periféricos porque disponen de un oscilador interno R-C, lo cual es una de su principales características.

En la siguiente tabla se presentan las principales características de los modelos de esta subfamilia. En los modelos 12C5xx el tamaño de las instrucciones es de 12 bits; mientras que en los 12C6xx sus instrucciones tienen 14 bits. Los modelos 12F6xx poseen memoria Flash para el programa y EEPROM para los datos.

[92] Extraído el 18 Octubre de 2009 desde
http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuatro.htm

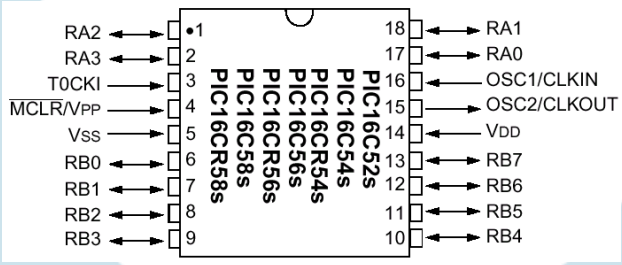
Tabla 16. ***Gamma Enana PIC***[93]

Modelo	Mem. de programa	Mem. de datos	Frecuencia	Líneas de E/S	ADC de 8 bits	Temporizador
PIC12C508	512x12	25x8	4MHz	6	---	TRM0+WDT
PIC12C509	1024x12	41x8	4MHz	6	---	TRM0+WDT
PIC12C670	512x14	80x8	4MHz	6	---	TRM0+WDT
PIC12C671	1024x14	128x8	4MHz	6	2	TRM0+WDT
PIC12C672	2048x14	128x8	4MHz	6	4	TRM0+WDT
PIC12F680	512x12 FLASH	80x8 16x8 EEPROM	4MHz	6	4	TRM0+WDT
PIC12F681	1024x14 FLASH	80x8 16x8 EEPROM	4MHz	6	4	TRM0+WDT

- **Gama baja**

  Se trata de una serie de PICs de recursos limitados, pero con una de la mejores relaciones costo/prestaciones. Sus versiones están encapsuladas con 18 y 28 pines y pueden alimentarse a partir de una tensión de 2,5 V, lo que los hace ideales en las aplicaciones que funcionan con pilas, teniendo en cuenta su bajo consumo (menos de 2 mA a 5 V y 4 MHz). Tienen un repertorio de 33 instrucciones cuyo formato consta de 12 bits. En la Figura se muestra el diagrama de pines de uno de estos PICs.

Figura 71. ***PIC gamma media PIC16Cxx***[94]



[93] Extraído el 18 Octubre de 2009 desde http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuatro.htm

[94] Extraído el 18 Octubre de 2009 desde http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuatro.htm

Al igual que todos los miembros de la familia PIC16/17, los componentes de la gama baja se caracterizan por poseer los siguientes recursos: Sistema Power On Reset, Perro guardián (*Watchdog* o WDT), Código de protección, etc. Sus principales desventajas o limitaciones son que la pila sólo tiene dos niveles y que no admiten interrupciones. En la siguiente tabla se presentan las principales características de los modelos de esta subfamilia.

Tabla 17. *Gamma Baja PIC*[95]

Modelo	Mem. de programa	Mem. de datos	Frecuencia	Líneas de E/S	Temporizador	pines
PIC16C52	384	25 bytes	4 MHz	12	TRM0+WDT	18
PIC16C54	512	25 bytes	20 MHz	12	TRM0+WDT	18
PIC16C55	512	24 bytes	20 MHz	20	TRM0+WDT	28
PIC16C56	1K	25 bytes	20 MHz	12	TRM0+WDT	18
PIC16C57	2K	72 bytes	20 MHz	20	TRM0+WDT	28
PIC16C58A	2K	73 bytes	20 MHz	12	TRM0+WDT	18

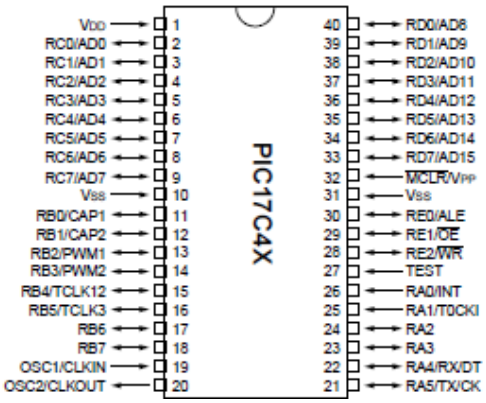
- **Gama media**

Es la gama más variada y completa de los PIC. Abarca modelos con encapsulado desde 18 pines hasta 68, cubriendo varias opciones que integran diversos periféricos. En esta gama sus componentes añaden nuevas prestaciones a las que poseían los de la gama baja, haciéndoles más adecuados en las aplicaciones complejas. Admiten interrupciones, poseen comparadores de magnitudes analógicas, convertidores A/D, puertos serie y diversos temporizadores.

El repertorio de instrucciones es de 35, compatible con el de la gama baja. Sus distintos modelos contienen todos los recursos que se precisan en las aplicaciones de los microcontroladores de 8 bits. También dispone de interrupciones y una pila de 8 niveles que permite el anidamiento de subrutinas.

[95] Extraído el 18 Octubre de 2009 desde
http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuatro.htm

Figura 72. ***PIC gamma alta PIC17CXX***[96]



En la siguiente tabla se presentan las principales características de algunos de los modelos de esta familia.

Tabla 18. ***Gamma Media PIC***[97]

Modelo	PINES	I / O	EPROM	RAM	Interrup	Voltaje (V)
PIC16C61	18	13	1Kx14	36x8	3	3.0-6.0
PIC16C62	28	22	2Kx14	128x8	10	2.5-6.0
PIC16C63	28	22	4Kx14	192x8	10	3.0-6.0
PIC16C64	40	33	2Kx14	128x8	8	3.0-6.0
PIC16C65	40	33	4Kx14	192x8	11	3.0-6.0
PIC16C620	18	13	512x14	80x8	4	3.0-6.0
PIC16C621	18	13	1Kx14	80x8	4	3.0-6.0
PIC16C622	18	13	2Kx14	128x8	4	3.0-6.0

Modelo	PINES	I / O	EPROM	RAM	Interrup	Canales A / D
PIC16C70	18	13	512x14	36x8	4	4 canales
PIC16C71	18	13	1Kx14	36x8	4	4 canales
PIC16C72	28	22	2Kx14	128x8	8	5 canales
PIC16C73	28	22	4Kx14	192x8	11	5 canales
PIC16C74	40	33	4Kx14	192x8	12	8 canales

96 Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/DeviceDoc/30412c.pdf

97 Extraído el 18 Octubre de 2009 desde
http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuatro.htm

Modelo	Mem. de programa RAM / EEPROM		Mem. de programa	Mem. de datos	INT	Líneas de E/S	Temporizador
PIC16F83	36	64	512X14 FLASH	25 bytes	4	13	TRM0+WDT
PIC16C84	36	64	1KX14 EEPROM	25 bytes	4	13	TRM0+WDT
PIC16F84	68	64	1KX14 FLASH	25 bytes	4	13	TRM0+WDT

- **Gama alta**: **PIC17CXXX**

  En esta gama se alcanzan las 58 instrucciones de 16 bits en el repertorio y sus modelos disponen de un sistema de gestión de interrupciones vectorizadas muy potente. También incluyen variados controladores de periféricos, puertos de comunicación serie y paralelo con elementos externos, un multiplicador hardware de gran velocidad y mayores capacidades de memoria, que alcanza los 8K palabras en la memoria de instrucciones y 454 bytes en la memoria de datos.

  Quizás la característica más destacable de los componentes de esta gama es su arquitectura abierta, que consiste en la posibilidad de ampliación del microcontrolador con elementos externos. Para este fin, los pines comunican al exterior las líneas de los buses de datos, direcciones y control, a las que se conectan memorias o controladores de periféricos. Esta facultad obliga a estos componentes a tener un elevado número de pines comprendido entre 40 y 44. Esta filosofía de construcción del sistema es la que se empleaba en los microprocesadores y no suele ser una práctica habitual cuando se emplean microcontroladores.

  En la siguiente tabla se presentan las características más relevantes de los modelos de esta gama, que sólo se utilizan en aplicaciones espaciales.

Tabla 19. ***Gamma Alta PIC***[98]

Modelo	CAP	PWM	Multiplica hardware	Mem. de programa	Mem. de datos RAM	Líneas de E/S	Temp.	pines
PIC17C42A	2	2	8X8	2KX16	232	33	4+WDT	18
PIC17C43	2	2	8X8	4KX16	454	33	4+WDT	18
PIC17C44	2	2	8X8	8KX16	454	33	4+WDT	18
PIC17C52	4	1	8X8	8KX16	454	50	4+WDT	18
PIC17C56	4	1	8X8	16KX16	902	50	4+WDT	28

La literatura sobre PIC es extensa, se encuentra material aceptable en la red y en librerías especializadas.

Familia de microcontroladores Intel MCS 51

El primer microcontrolador en esta familia fue el 8048, en su interior se alojaba una memoria RAM pero el programa se debía almacenar en un dispositivo externo, unos años mas tarde se desarrolla el 8051 la cual es la piedra angular de una serie de dispositivos con características especiales para aplicaciones especificas.

Las versiones existentes de esta familia se basan en el núcleo del 8051, de ahí que se tomo como referencia en la denominación oficial de Intel para la familia de microcontroladores basados en el 8051, esta denominación es MCS51.

El 8051 se caracteriza por tener 4K de memoria ROM, posteriormente se implementa el 8751 con una memoria EPROM dando la posibilidad de la reprogramación, borrando el dispositivo de memoria por exposición a luz ultravioleta. La característica más sobresaliente de estos dispositivos es la capacidad de expansión de memoria, es decir, tienen puertos habilitados para

[98] Extraído el 18 Octubre de 2009 desde http://www.ufps.edu.co/materias/ucontrol/htdocs/conte/usuarios.lycos.es/sfriswolker/pic/cuatro.htm

direccionar hasta 64K de memoria externa RAM y ROM esta ultima con la capacidad de almacenar el programa de control.

El núcleo del 8051 es usado en mas de 100 tipos de microcontroladores por mas de 20 fabricantes (Atmel, Dallas Semiconductor, Philips, etc).

Figura 73. ***Familia MSC51***[99]

En el mercado se encuentran versiones de microprocesadores como el 8086 y 8088 que permiten aprovechar las herramientas desarrollada para PC, estos microcontroladores, son el 80186, 80188 y 80386EX.

Memoria

Estos dispositivos a diferencia de los microprocesadores, tiene un espacio para las direcciones de datos tanto para la lectura como para la escritura y otro para las direcciones de programa, es decir, emplea una arquitectura Harvard. La capacidad de direccionamiento de memoria llega hasta los 64K, de los cuales en versiones ROM, EPROM y EEPROM, los 4K, 8K o 16K inferiores son alojados en el microcontrolador. El microcontrolador tiene la memoria interna dividida en dos partes:

Adicionalmente, el microcontrolador contiene una memoria interna, dividida en dos partes:

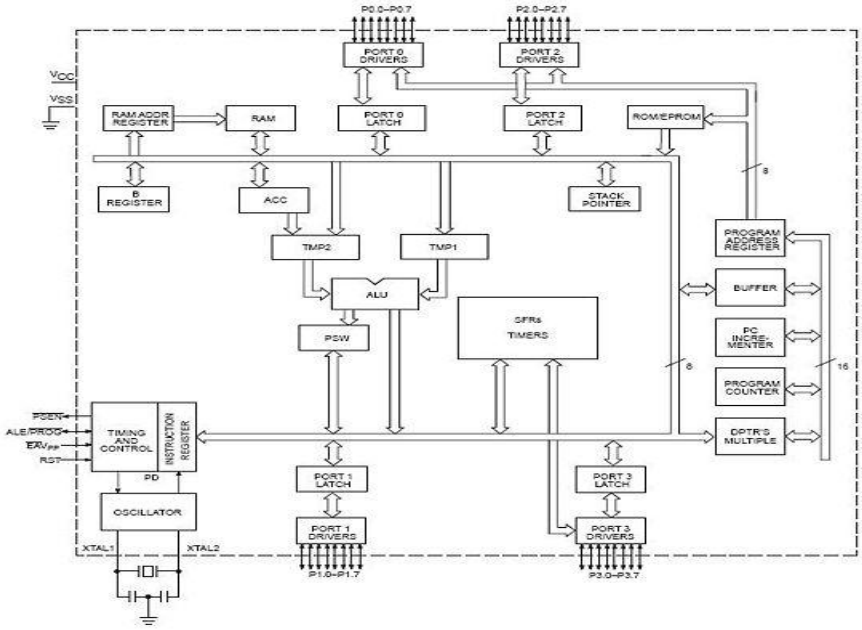
- SFR, (Special Function Registers), son los registros proporcionados por el microcontrolador, y tienen asignadas direcciones en esta memoria interna.
- Memoria de propósito general.

El acceso a esta memoria interna es más rápido que el acceso a la memoria externa, pero es de tamaño limitado. Parte de esta memoria interna además se usa como pila durante las llamadas a función y el proceso de interrupciones.

[99] Extraído el 18 Octubre de 2009 desde http://es.wikipedia.org/wiki/Archivo:KL_Intel_P8051.jpg

Actualmente están disponible versiones del 8051 con memoria de tipo FLASH, lo que permite una programación rápida y con una capacidad mayor de reprogramaciones, se evidencia la complejidad de su programador por lo que se hace difícil implementar estos proyectos al incluir costos adicionales del programador y paquetes de programación.

Figura 74. ***Arquitectura interna MSC51***[100]



Características

- Permite operaciones a nivel de bit, por la inclusión de una unidad booleana.
- Tiene cuatro conjuntos separado de registros.
- Pueden incluir una o dos UARTs, puerto Transmisor-Receptor Asíncrono Universal.
- Inclusión de dos o tres temporizadores.
- 128 a 256 de RAM interna
- 0K y 54K de memoria de programa.
- Watchdog.
- Compatibilidad con I^2C, SPI, USB.
- Generadores PWM
- Conversor A/D y D/A.

[100] Extraído el 18 Octubre de 2009 desde http://es.wikipedia.org/wiki/Archivo:Diag_bloques_8051.JPG

Conjunto de instrucciones

Todos los miembros de la familia ejecutan las mismas instrucciones optimizando aplicaciones de control 8 bits, con capacidad de varios modos de direccionamiento, soporte para control de programa basados en operaciones de un (1) bit.

Las instrucciones se dividen en instrucciones aritméticas, instrucciones lógicas e instrucciones de transferencia de datos. No se hace profundiza en el estudio y aplicaciones con este dispositivo dado la complejidad y costo del sistema de desarrollo (programador). En este curso se hará énfasis en los microcontroladores de la familia Motorola Freescale y Microchip PIC.

Familia de microcontroladores ATMEL

Esta empresa maneja microcontroladores basados en arquitectura RISC, las CPUs llegan hasta 32 bits, existen varios grupos de microcontroladores:

- Microcontroladores basados en el 8051 Intel, incorporan una memoria de programa Flash.
- Microcontroladores AT91, soportan compilaciones en C, emulator.
- Microcontroladores AVR, con arquitectura RISC y CPU de 8 bits, incorpora módulos USART, SPI, ADC, etc. Implementado sobre arquitectura Harvard.

Figura 75. ***Familia ATMEL***[101]

[101] Extraído el 18 Octubre de 2009 desde http://es.wikipedia.org/wiki/Archivo:ATMEL-AT90S2333.jpg

CAPÍTULO 5

MICROCONTROLADORES DE 8 BITS PIC16F84 y PIC16F877

LECCIÓN 1: MICROCONTROLADORES PIC.

Los códigos que identifican a un microcontroladores PIC, también entregan varias características importantes al usuario:

- El tipo de memoria utilizado se identifica por la letra después de la serie de gamma del microcontrolador:
  - **C**, como PIC16CXXX utiliza memoria EPROM.
  - **CR**, como PIC16CRXXX utiliza memoria ROM.
  - **F**, como PIC16FXXX utiliza memoria tipo Flash.
- El rango de voltaje tanto estándar como extendido se identifica así:

Tabla 20. PIC *Estándar y Extendido*[102]

Rango de Voltaje	EPROM		ROM		FLASH	
Estándar	16CXXX	4,5 - 6,0 V	16CRXXX	4,5 - 6,0 V	16FXXX	4,5 - 6,0 V
Extendido	16LCXXX	2,5 - 6,0 V	16LCRXXX	2,5 - 6,0 V	16LFXXX	2,0 - 6,0 V

[102] CEKIT, 2002

Características generales de los microcontroladores PIC16F84 y PIC16F877

- **CARACTERÍSTICAS GENERALES PIC16F84**
  - Bajo consumo de energía
  - Frecuencia de reloj externa máxima de 10MHz
  - No posee conversores analógicos/digitales ni digitales/analógicos
  - Pipe-line de 2 etapas, 1 para búsqueda de instrucción y otra para la ejecución
  - de la instrucción (los saltos ocupan un ciclo más).
  - Repertorio de instrucciones reducido (RISC), con tan solo 35 instrucciones
  - 4 tipos distintos de instrucciones: orientadas a byte, orientadas a bit,
  - operación entre registros y de salto.
  - 1024 palabras de memoria de programa
  - Memoria de datos RAM de 68 bytes
  - Memoria de datos EEPROM de 64 bytes
  - Palabras de instrucción de 14 bits
  - Bytes de datos de 8 bits
  - 15 registros especiales de función hardware
  - 8 niveles de pila
  - Modos de direccionamiento: directo, indirecto y relativo
  - Cuatro fuentes de interrupción
  - TMR0 de 8 bits con pre-escala programable
  - Perro guardián (watchdog)

- **CARACTERÍSTICAS GENERALES PIC16F877**

  El microcontrolador PIC16F877 tiene como características generales:
  - Arquitectura RISC y 35 instrucciones de palabra sencilla
  - Todas las instrucciones son de un solo ciclo, excepto los saltos (dos ciclos)
  - Velocidades de operación: DC hasta 20MHz con entrada de reloj y DC hasta
  - 200ns ciclo de instrucción.
  - Hasta 8Kx14 palabras de memoria programable FLASH, 368x8 bytes de
  - memoria de datos RAM y 256x8 bytes de memoria de datos EEPROM.
  - Capacidad de interrupción (hasta 14 fuentes).
  - 8 niveles de pila
  - Perro guardián (watchdog)
  - Modos de direccionamiento: directo, indirecto y relativo
  - Conversor analógico digital multicanal de 10 bits.
  - TRM0, TRM2 de 8 bits contador/ preescalar.
  - TRM1 de 16 bits contador/ prescalar.
  - Tiene dos módulos de comparación, captura de 16 bits y PWM de 10 bits.
  - Receptor/transmisor USART / SCI.
  - Puerto paralelo esclavo de 8 bits con líneas de control externo.

LECCIÓN 2: MODOS DE DIRECCIONAMIENTO Y DIAGRAMA DE PINES.

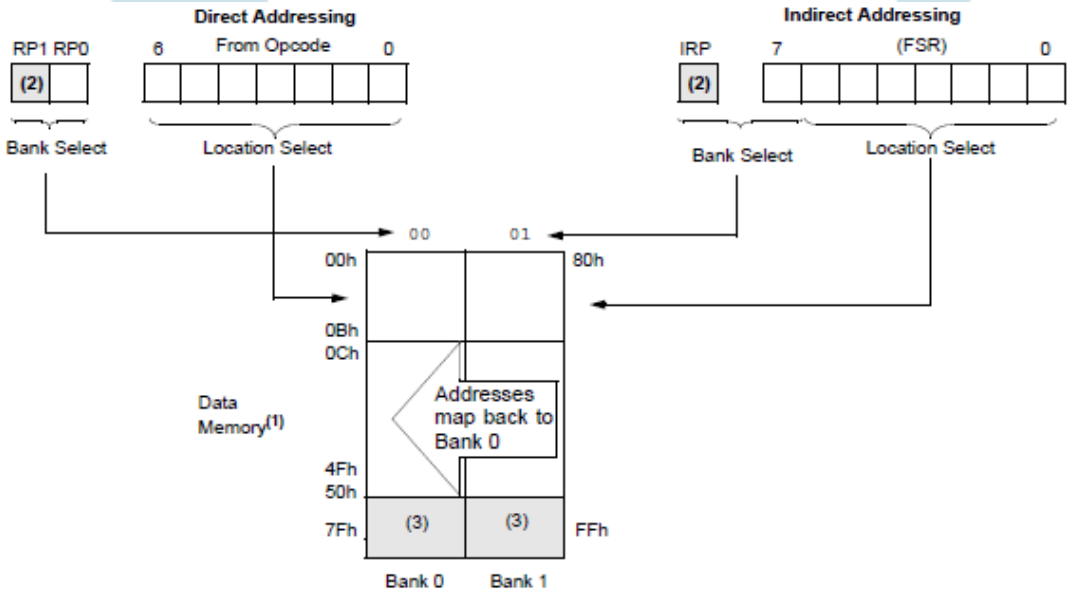
Modos de direccionamiento

Existen tres modos de direccionamiento en los microcontroladores PIC: directo, indirecto y relativo. En el direccionamiento directo se utilizan los valores de RP1 y RP0 para seleccionar el banco y la localización a través del formato de instrucción (OPCODE). En el caso particular de esta familia, el direccionamiento indirecto está determinado por los registros INDF y FSR. Hay que destacar que el registro INDF no es un registro físico, pero a través de el es que se realiza el direccionamiento indirecto. Cualquier instrucción que utilice el registro INDF accede al registro puntero de selección de fila, FSR.

Si se lee solamente el registro INDF el valor de FSR es igual a 0, es decir, la dirección leída es igual 00H. Los 9 bits de dirección efectiva son el resultado de concatenar los 8 bits del registro FSR y el bit 7 (IRP) del registro de estado.

El direccionamiento relativo se logra sumando el contenido del registro de trabajo W al registro contador de programa (PC). Este direccionamiento se utiliza ampliamente en la elaboración de tablas de saltos. La siguiente figura ilustra la diferencia entre el direccionamiento directo y el indirecto.

Figura 76. ***Diferencia entre direccionamiento directo e indirecto***[103]

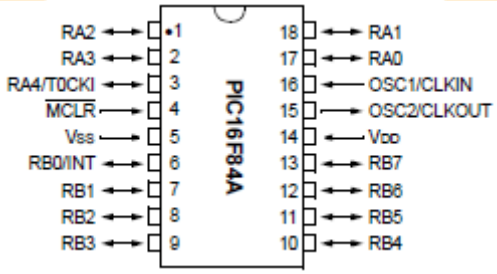


[103] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/devicedoc/30430c.pdf

Funciones y diagrama de pines para el PIC16F84

La siguiente figura muestra el diagrama de pines del PIC16F84 en encapsulado PDIP, SOIC.

Figura 77. ***PIC16F84A***[104]



La descripción de las funciones de los pines de este microcontrolador se presenta en la siguiente tabla.

Tabla 21. Descripción de pines PIC16F84[105]

NOMBRE DEL PIN	TIPO	DESCRIPCION
RA0 a RA3	Entrada/salida	Líneas de E/S digitales del Puerto A
RA4/T0CKI	Entrada/salida	E/S digital o entrada de reloj externo para el TMR0.
RB0/INT a RB7	Entrada/salida	E/S digitales del Puerto B. RB0/INT puede actuar como INT externa. RB4-RB7 pueden provocar una interrupción cuando cambian de estado.
OSC1/CLKIN	Entrada	Entrada de cristal oscilador / entrada de reloj externo.
OSC2/CLKOUT	Salida	Salida del cristal oscilador. En el modo RC, la salida del pin de OSC2/CLKOUT, tiene ¼ de la frecuencia de OSC1 (ciclo de instrucción)
V_{SS}	Alimentación	Tierra para los pines lógicos y de E/S
V_{DD}	Alimentación	Fuente de tensión positiva (Típicamente 5V)
$\overline{MCLR}/V_{PP}$	Entrada	Entrada maestra de borrado (Reset)/ voltaje de programación. El reset del dispositivo es activo bajo.

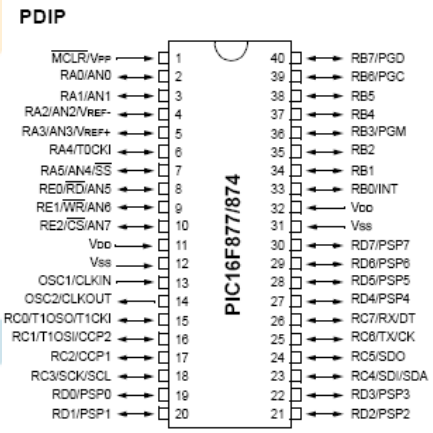
[104] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/devicedoc/30430c.pdf

[105] Téllez, 2007

Funciones y diagrama de pines para el PIC16F877

La asignación de pines del PIC16F877 se muestra en la siguiente figura.

Figura 78. ***PIC16F877/874***[106]



El PIC16F877 viene en una pastilla integrada de 40 pines. 33 pines conforman los cinco puertos bidireccionales que posee, mientras que los siete pines restantes se emplean para la aplicación del voltaje de alimentación (4), el circuito oscilador (2) y el circuito de Reset.

La descripción de las funciones de los pines del microcontrolador PIC16F877 se presenta a continuación.

Tabla 22. ***Descripción de pines PIC16F877***[107]

NOMBRE DEL PIN	TIPO	DESCRIPCION
RA0/AN0 - RA2/AN2	Entrada/salida	Líneas de E/S digitales del Puerto A, o entradas analógicas.
RA3/AN3/VREF	Entrada/salida	E/S digital, analógica o entrada externa de referencia
RA4/T0CKI	Entrada/salida	E/S digital o entrada de reloj externo para TMR0.
RA5/AN4/SS	Entrada/salida	E/S digital, analógica o selección del puerto síncrono

[106] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf
[107] Téllez, 2007

RB0/INT - RB7	Entrada/salida	E/S digitales del Puerto B. RB0/INT puede actuar como entrada de interrupción externa. RB4-RB7 pueden provocar una interrupción cuando cambian de estado.
RC0/T1OSO/T1CKI	Entrada/salida	E/S digital del Puerto C. Conexión del oscilador externo para el temporizador TMR1 o entrada de reloj para el TMR1.
RC1/T1OSI/CCP2	Entrada/salida	Conexión del oscilador externo para el TMR1 o salida del módulo 2 de captura/comparación.
RC2/CCP1	Entrada/salida	Salida del módulo 1 de captura/comparación.
RC3/SCK/SCL	Entrada/salida	E/S de reloj para el Puerto Serie Síncrono (SSP) en los módulos SPI o I2C.
RC4/SDI/SDA	Entrada/salida	E/S digital. Entrada de datos serie en el modo SPI. E/S de datos serie en modo I2C.
RC5/SDO	Entrada/salida	E/S digital. Salida de datos serie en modo SPI
RC6/TX/CK	Entrada/salida	E/S digital. Transmisión serie asíncrona. Entrada de reloj para comunicación serie síncrona.
RC7/RX/DT	Entrada/salida	E/S digital. Recepción serie asíncrona. Línea de datos en la comunicación serie síncrona.
RD0/PSP0 - RD7/PSP7	Entrada/salida	E/S digitales del Puerto D. Este puerto puede trabajar como puerto paralelo esclavo para interconexión con un bus de datos de 8 bits de otro microprocesador.
RE0/RD/AN5	Entrada/salida	E/S digital del Puerto E. Señal de lectura del puerto paralelo esclavo. Entrada analógica.
RE1/WR/AN6	Entrada/salida	E/S digital del Puerto E. Señal de escritura del puerto paralelo esclavo. Entrada analógica.
RE2/CS /AN7	Entrada/salida	E/S digital. Señal de activación del puerto paralelo esclavo. Entrada analógica.
OSC1/CLKIN	Entrada	Entrada del cristal / entrada de reloj externo.
OSC2/CLKOUT	Salida	Salida del cristal oscilador. Conecta el cristal o el resonador en el modo cristal oscilador. En el modo RC, la salida del pin de OSC2 CLKOUT, tiene ¼ de la frecuencia de OSC1
V_{SS}	Alimentación	Tierra para los pines lógicos y de E/S
V_{DD}	Alimentación	Fuente de Tensión Positiva
$\overline{MCLR}$	Entrada	Entrada maestra de borrado (Reset)/ voltaje de programación. El reset del dispositivo es activo bajo.

LECCIÓN 3: ARQUITECTURA, FUNCIONAMIENTO Y SET DE INSTRUCCIONES.

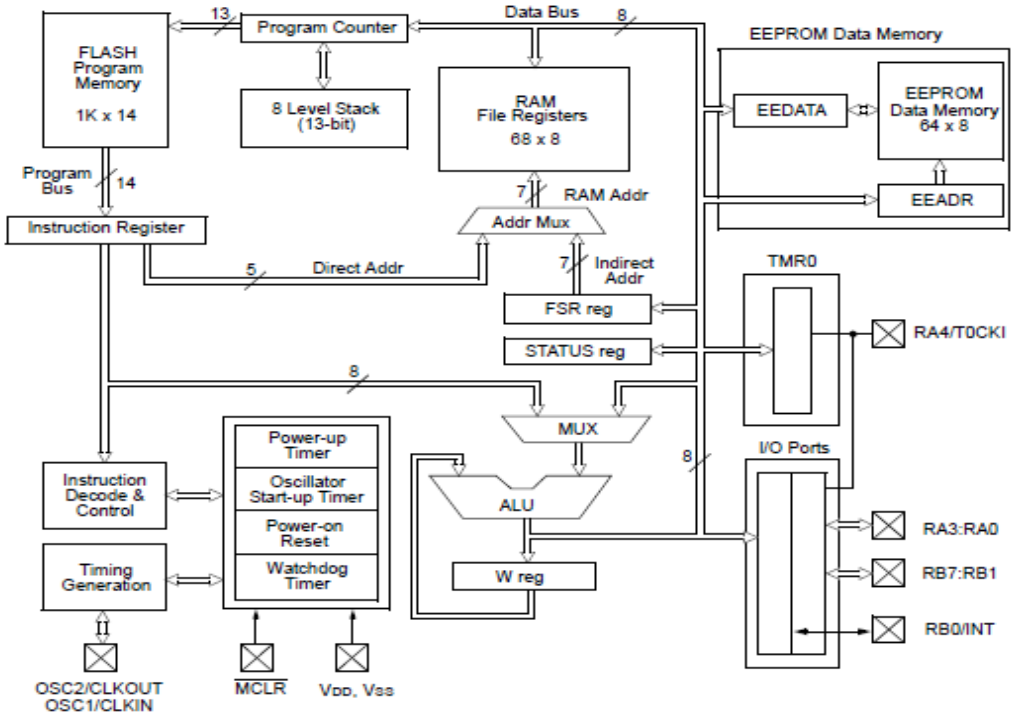
Arquitectura interna del pic16F84 y pic16F877

Como se ha mencionado, el microcontrolador emplea una arquitectura Harvard, en la cual, programa y datos se acceden de memorias separadas usando diferentes buses. Esto trae como consecuencia la existencia de dos buses independientes y la posibilidad de tener un ancho diferente. En este caso, el bus de datos es de 8 bits y el bus de programa es de 14 bits.

La secuencia de las instrucciones de un programa, está controlada por el registro contador de programa (Program Counter). Este registro se incrementa en cada paso para ejecutar el programa grabado en la memoria ROM del microcontrolador. El contador de programa también está conectado a los registros de Stack o pila, que son los que soportan llamadas consecutivas a una subrutina.

El diagrama de bloques de la estructura interna del PIC16F84 se presenta en la siguiente figura.

Figura 79. ***Arquitectura del PIC16F84***[108]

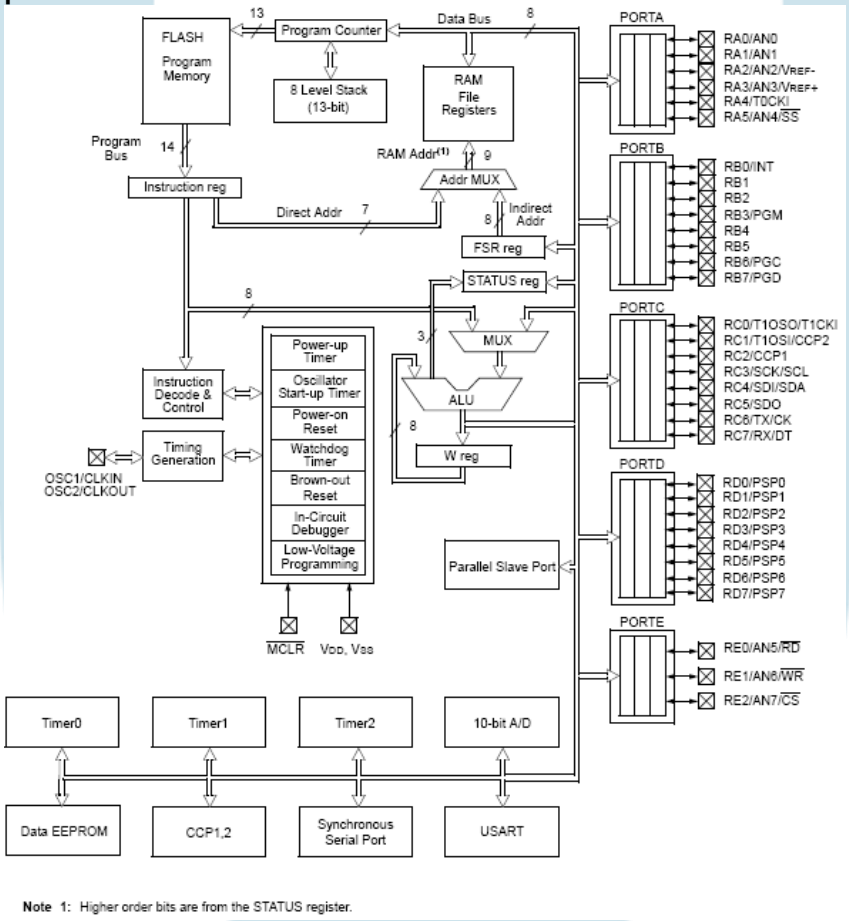


[108] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/devicedoc/30430c.pdf

El microcontrolador también posee una unidad lógica aritmética (ALU) de 8 bits y un registro de trabajo (W). La ALU ejecuta funciones aritméticas y booleanas entre datos del registro de trabajo y cualquier otro registro o constante. También es capaz de realizar operaciones lógicas, adiciones y sustracciones entre datos de 8 bits.

El registro de trabajo (W), es un registro de 8 bits de gran importancia para el buen funcionamiento del microcontrolador. Se emplea intensamente en las operaciones que requieren transferencia interna de datos, como por ejemplo, en las operaciones de la ALU o en las comunicaciones entre registros. Para la ejecución de un programa, las palabras de la memoria de programa se llevan al registro de instrucciones, el cual, las comunica al decodificador de instrucciones, en donde se da la orden de iniciar su ejecución. A continuación se presenta una breve descripción de los principales componentes del microcontrolador. El diagrama de bloques de la estructura interna del PIC16F877 se presenta en la siguiente figura.

Figura 80. Arquitectura del PICF877[109]



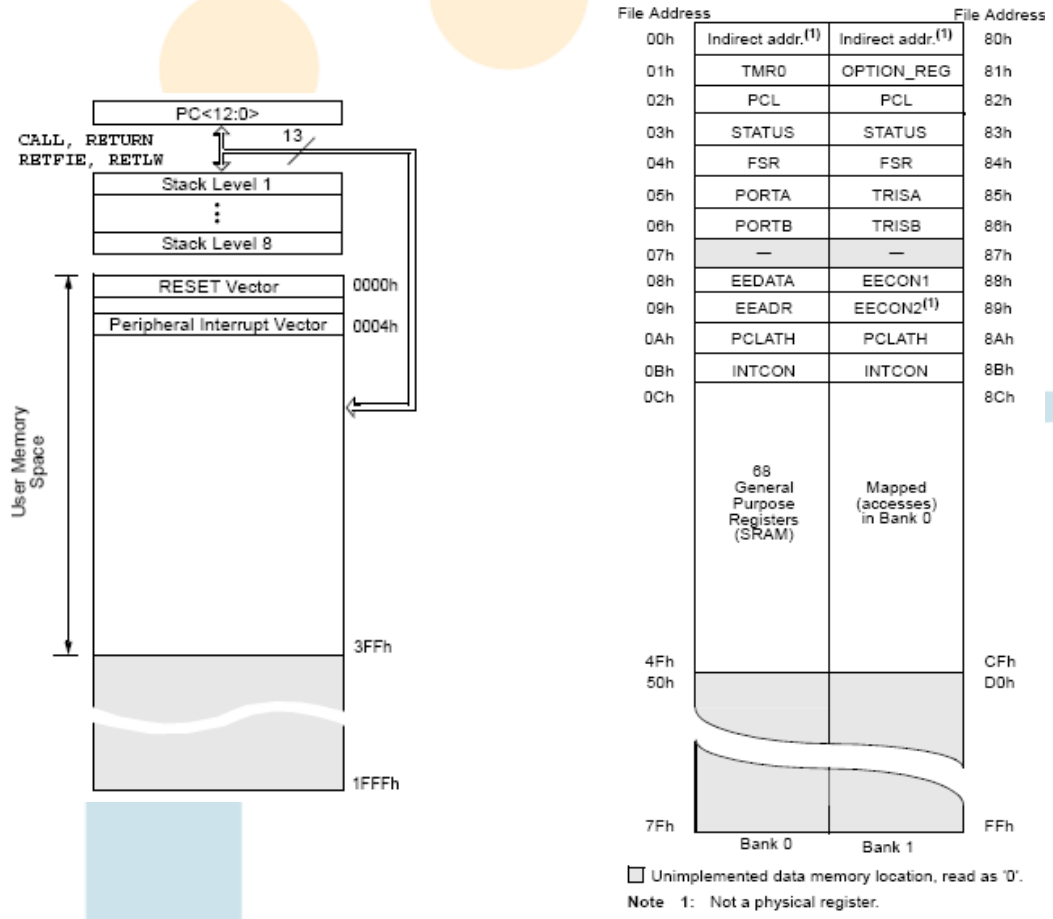
Note 1: Higher order bits are from the STATUS register.

[109] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

Organización de la memoria

- **Organización de la memoria para los PIC16F84**

Figura 81. ***Organización de la memoria***[110]



La memoria de programa tiene implementado 1024 palabras de 14 bits cada una, para el almacenamiento de las instrucciones del programa a ejecutar. También cuenta con dos vectores empleados para el manejo de las Interrupciones y el estado de Reinicialización (reset) del microcontrolador.

La memoria de datos está particionada en dos áreas. La primera es para los registros de funciones especiales (SFR), y la segunda área es para los registros de propósito general (GPR). Los registros especiales controlan las operaciones del dispositivo.

Toda la memoria de datos puede ser accedida ya sea por direccionamiento directo utilizando direcciones absolutas de cada registro de fila o indirecto a través del registro de selección de fila (FSR).

[110] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/devicedoc/30430c.pdf

- **Organización de la memoria en los PIC16F877**

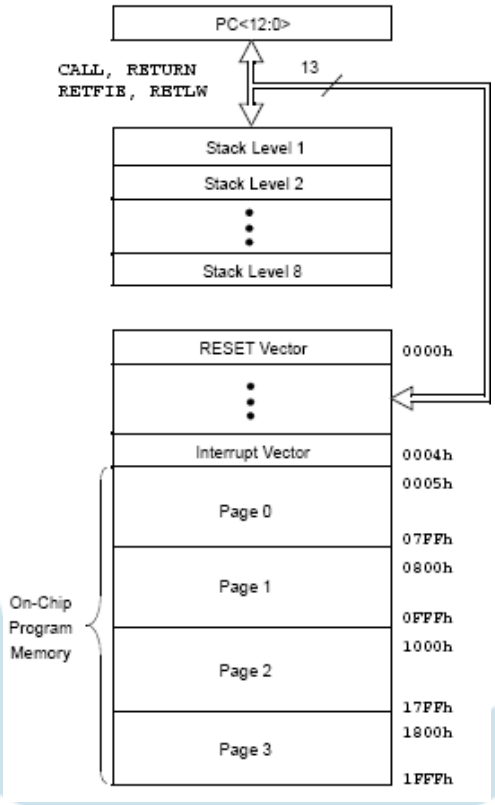
Existen principalmente dos bloques de memoria en el interior del microcontrolador: la memoria de datos y la memoria de programa, cada una con su propio bus.

- **Organización de la Memoria de Programa**

El microcontrolador tiene un contador de programa de 13 bits capaz de direccionar un espacio de memoria de programa de 8K x 14, es decir 8192 palabras de memoria FLASH.

El vector de RESET está en la dirección 0000h y el vector de interrupción en la 0004h. Además cuenta con 8 niveles de pila como lo muestra la figura.

Figura 82.Organizacion de la memoria de programa[111]

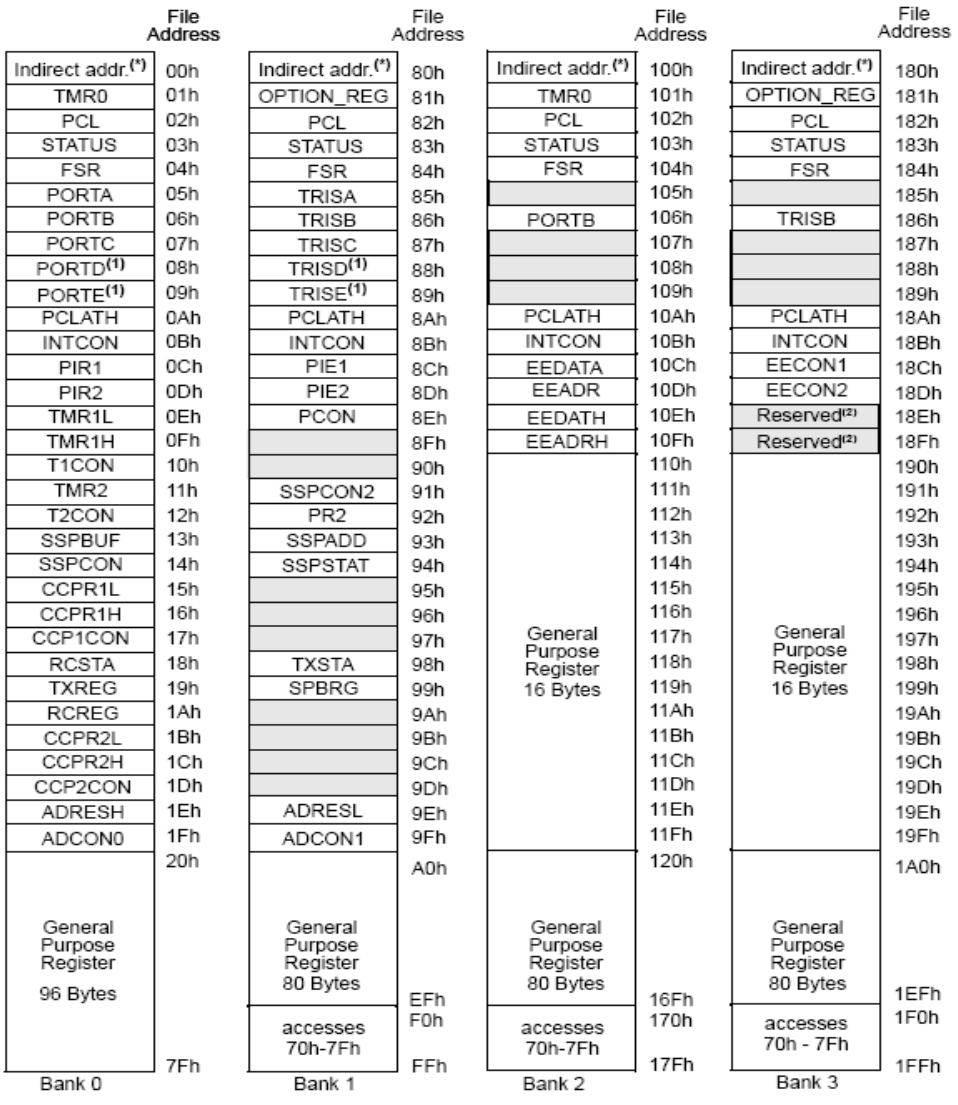


[111] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

- **Organización de la memoria de datos**

La memoria de datos está particionada en múltiples bancos (4) que contienen registros de propósito general y registros de función especial.

Figura 83. ***Organización de la memoria de datos***[112]



Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
2: These registers are reserved, maintain these registers clear.

[112] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

Los bits RP1 (bit 6) y RP0 (bit 5) del registro de estatus seleccionan los bancos de memoria como lo muestra la siguiente figura.

Figura 84.*Cambio de bancos*[113]

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Registros de uso genera

Registro W: es un registro no direccionable de 8 bits utilizado para las operaciones de la ALU.

Registro STATUS(03h y 83h): El registro de estado contiene el estado aritmético de la ALU, el estado del proceso de re inicialización y los bits de selección de bancos de memoria, este registro puede trabajar como cualquier otro registro de almacenamiento, pero si la instrucción afecta matemáticamente, los bits Carry, Digit Carry y Bandera de Cero, se inhabilitan. El registro STATUS está formado por 8 bits:

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	/TO	/PD	Z	DC	C
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

R/W significa que el bit correspondiente se puede leer y escribir, mientras que R significa que solamente puede ser leído. También se indica el estado que se establece tras un reset.

Bit 7, IRP: Selección del banco en direccionamiento indirecto. Este bit junto con el de más peso del registro FSR sirven para determinar el banco de la memoria de datos seleccionado. En el PIC16X84 al disponer de dos bancos no se usa y debe programarse como 0.

[113] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

Bit 6 y 5, RP0 y **RP1**: Register Bank Select. Selección de página o banco de la memoria con direccionamiento directo. Cada página contiene 128 bytes. Como el PIC16X84 sólo tiene dos bancos únicamente se emplea RP0 de forma que cuando vale 0 se accede al banco 0 y cuando vale 1 se accede al banco 1. Después de un reset, RP0 se pone automáticamente a 0. RP1 debe mantenerse a 0. El bit RP1 deberá ser puesto a cero, ya que si no nos saldríamos del rango de memoria.

Bit 4 (flag), TO: Time Out (Tiempo acabado)

1. Se pone a 1 tras conectar la alimentación o al ejecutar CLRWDT o SLEEP.

0. Se pone a 0 por desbordamiento del Perro Guardián WDT.

Bit 3 (flag), PD: Power Down (Apagado).

1. Se pone automáticamente a 1 tras conectar la alimentación Vdd o ejecutar CLRWDT, que resetea el contador WatchDog.

0. Se pone a 0 al ejecutar la instrucción SLEEP.

Bit 2 (flag), Z: Cero

1 = El resultado de una operación aritmética o lógica es 0.
0 = El resultado es distinto de 0.

Bit 1 (flag), DC (Digit Carry). Acarreo en el 4º bit de menos peso. Funciona igual que el bit de Carry descrito a continuación. De interés en operaciones en BCD

Bit 0 (flag), C (Carry). Acarreo en el 8º bit o bit de mas peso. Es el bit de "acarreo" en operaciones de suma AADWF y ADDLW así como también el bit de "llevada" en las instrucciones de sustracción SUBWF y SUBLW. También lo emplean las instrucciones RLF y RRF de rotación de bits.

Suma

1. Se pone a 1 cuando se ha producido acarreo en la suma en el bit de mayor peso con las operaciones AADWF y ADDLW.

0. Se pone a 0 si en la suma no se ha producido acarreo.

Resta

1. Se pone a 1 si en la resta no se ha producido llevada.

0. Se pone a 0 cuando se ha producido llevada en la resta con las operaciones SUBWF y SUBLW.

Registro FSR (04h y 84h): El contenido del FSR se utiliza para el direccionamiento indirecto junto, este registro contiene 8 bits

Registro PORTA y PORTB (05h y 06h):]Cada puerto tiene vinculado un registro que almacena la información que entra o sale del microcontrolador.

Registro EEDATA (08h): El registro EEDATA (Datos de EEPROM) guarda el contenido de una posición de la memoria EEPROM de datos antes de su escritura o después de su lectura, según leamos o escribamos en ella. Para leerla se sigue un proceso especial. La memoria EEPROM es bastante lenta, dato que tendremos en cuenta cuando accedamos a ella para escribirla, pues tarda unos 10 ms en completar el proceso.

Registro EEADR (09h): El registro EEADR (Dirección de EEPROM) guarda la dirección de la posición de memoria EEPROM cuando se accede a ella, bien para su lectura, o bien para su escritura. El registro EEADR puede direccionar como máximo 256 bytes de los cuales sólo los 64 primeros están disponibles, con lo que los dos bits de mayor peso han de tener el valor de '0'.

Registro PCLATH (0Ah y 8Ah): El registro PCLATH (Contador de Programa Alto), guarda la parte alta de la dirección de apuntamiento no accesible por el programador.

Registro INTCON (0Bh y 8Bh): Este registro contiene varios bits de selección de fuentes de interrupción, el bit de activación global de interrupciones y varios flag

que indican la causa de una interrupción. Sirve para el control global de las interrupciones y para indicar la procedencia de algunas de ellas, gracias a los bits de estado. Se dispone de cuatro potenciales recursos de interrupción:

- Una fuente externa a través del pin RB0/INT.
- El desbordamiento del temporizador 0 (TMR0).
- Un cambio de estado en los pines RB4 a RB7.
- Programación de la EEPROM de datos.

Cada bit del registro INTCON tiene un significado concreto que se muestra en la siguiente tabla:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Bit 7, GIE: Habilitación global de interrupciones (Global Interrupt Enable).

1: Concede el permiso de interrupciones.

0: Cancela el permiso de las interrupciones.

Bit 6, EEIE: Habilitación de las Interrupciones de la memoria EEPROM.

1: Permite que se produzcan interrupciones debidas al fin de escritura de la EEPROM, etc.

0: Este tipo de interrupciones estarán inhibidas.

Bit 5, T0IE: Habilitación de la interrupción del temporizador por desbordamiento (Timer 0 Interrupt Enable).

1: Autoriza las interrupciones debidas al desbordamiento del temporizador.

0: Interrupción del temporizador deshabilita de manera que cuando se produzca el flag correspondiente permanecerá inactivo.

Bit 4, INTE: Habilitación de la entrada de interrupción externa (Interrupt Enable) por patilla RB0/INT.

1: Autoriza las interrupciones provocadas RB0/INT del puerto B.

0: Interrupción externa deshabilita de manera que cuando se produzca una interrupción externa el flag correspondiente permanecerá inactivo.

Bit 3, RBIE: Habilitación de las interrupciones del puerto B (RB Interrupt Enable).

1: Autoriza las interrupciones provocadas por un cambio de estado de las líneas RB4 a RB7 del puerto B.

0: Interrupción del puerto B deshabilitada.

Bit 2 (flag), T0IF: Bit de interrupción de desbordamiento del TMR0.

1: El TMR0 ha rebosado. Se borra por software.

0: El TMR0 no ha rebosado.

Bit 1 (flag), INTF: Bit de interrupción de la Entrada de Interrupción INT (patilla RB0/INT).

1: La entrada de interrupción se ha activado (patilla RBO/INT del puerto B). Se borra por software.

0: No hay interrupción externa.

Bit 0 (flag), RBIF: Bit de interrupción del puerto B.

1: Cambio de estado en una de las líneas de RB4 a RB7 del puerto B. Se borra por software.

0: Ninguna línea de RB4 a RB7 del puerto B ha cambiado.

Cada flag o bandera individual debe ponerse a cero por software.

Solamente hay un vector de interrupción en la memoria de programa (dirección 0004h), por lo que se deben comprobar los bits de INTCON en la subrutina de interrupción para saber cual es la fuente de la misma. Cuando llega una interrupción, el PIC pone el bit GIE a cero, de forma que no se perturbe el tratamiento de la interrupción en curso, debido a otras interrupciones eventuales. Este bit se pone automáticamente a uno al terminar la subrutina de interrupción, con la ejecución de la instrucción RETFIE.

Los indicadores de interrupciones correspondientes permanecen funcionales incluso cuando no se han autorizado. En este caso también pueden leerse y escribirse todos los bits que componen este registro.

Registro OPTION (80h): El registro OPTION (o registro de opciones) se emplea para programar las opciones del temporizador TMR0, el tipo de flanco con el que se detecta una interrupción y la activación de las resistencias de polarización del puerto B. Ocupa la posición 81h de la página 1 del banco de registros. Debe escribirse usando la instrucción especial OPTION. Esta instrucción carga el contenido de W en el registro OPTION.

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
/RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 7, /RBPU (RB Pull Up). Conexión de las resistencias de polarización del Puerto B. Se conectan todas cuando el puerto B actúa como entrada.

1: Todas las resistencias son desconectadas.

0: Las resistencias se activan de forma individual.

Bit 6, INTDEG (INTerrupt EDGe). Selecciona el tipo de flanco para la interrupción externa. Este bit indica el tipo de flanco de la señal externa que ha de provocar una interrupción en la patilla RB0/INT.

1: La interrupción es producida por el flanco ascendente o de subida.

0: La interrupción es producida por el flanco descendente o de bajada.

Bit 5, T0CS (Timer 0 Signal Source). Selección de la fuente de reloj para el TMR0.

1: TMR0 se usa en modo contador de los pulsos introducidos a través de RA4/T0CKI

0: TMR0 se usa en modo temporizador haciendo uso de los pulsos de reloj internos (Fosc/4).

Bit 4, T0SE (Timer 0 Signal Edge). Tipo de flanco activo de T0CKI (patilla RA4/T0CKI).

1 = El TMR0 se incrementa con el flanco descendente de la señal aplicada a RA4/T0CK1.

0 = El TMR0 se incrementa con el flanco ascendente.

Bit 3, PSA (PreScaler Assignement). Se usa para la asignación del divisor de frecuencias o Prescaler.

1 = El divisor de frecuencia se asigna al WDT.

0 = El divisor de frecuencia se asigna a TMR0.

Bits 0, 1 y 2, PS0, PS1 y PS2 (Prescaler Rate Select Bits). Configura la tasa del valor del divisor de frecuencia o prescaler. Difiere dependiendo que se haya asignado al TMR0 o al WDT.

PS2	PS1	PS0	Divisor TMR0	Divisor WDT
0	0	0	1:2	1:1
0	0	1	1:4	1:2

0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

Registro TRISA y TRISB (85h y 86h): Estos registros son idénticos para el puerto A y el puerto B, con la diferencia de que uno será de 5 bits y otro de 8 bits, el mismo número de bits que tiene cada puerto. Los registros TRIS, también son llamados así, sirven para configurar si los bits de cada puerto serán de entrada o de salida:

1: La patilla del puerto correspondiente será de entrada

0: En este caso la patilla actuará como una salida.

Registro EECON1 (88h): Este registro contiene configuraciones importantes acerca de la escritura y la lectura de la EEPROM de datos. En concreto tiene 5 bits de control, cuya distribución y significado es el siguiente.

U-0	U-0	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
-	-	-	EEIF	WRERR	WREN	WR	RD
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

U (Unimplemented), No implementado. Se lee como 0.

Bit 4 (flag): EEIF. Bit de interrupción de escritura en la memoria EEPROM (EEPROM Interrupt Flag)

1: Este bit se pone a uno al terminar la operación de escritura en la EEPROM, y debe ponerse a cero por software

0: No se ha completado la operación de escritura o no ha empezado.

Bit 3 (flag), WRERR. Bit de error de escritura (Write Error)

1: Este bit se pone a 1 si se produce un error de escritura de forma prematura (Reset o Watchdog). En este caso, los contenidos de EEADR y EEDATA no varían, de manera que el proceso pueda ser repetido correctamente.
0: Se ha completado la operación de escritura.

Bit 2, WREN. Bit de habilitación de escritura. (Write Enable)

1: Este bit debe ser habilitado para poder escribir en la EEPROM
0: Deshabilita la escritura de datos en la memoria EEPROM.

Bit 1, WR. Bit de control de escritura (Write Data)

1: Indica que se ha iniciado una operación de escritura. Este bit debe ponerse a uno para escribir un dato.
0: Indica que se ha completado una operación de escritura. El PIC lo pone automáticamente a cero

Bit 0, RD. Bit de control de lectura (Read Data)

1: Inicia una lectura de la memoria EEPROM. Este bit debe ponerse a uno para poder leer un dato.
0: No se ha iniciado una lectura de la EEPROM. El PIC lo pone automáticamente a cero

Registro EECON2 (89h)

Este registro no está implementado físicamente, por lo cual no se puede leer. Tan sólo sirve para un proceso de protección de escritura que consiste en copiar en él unos datos específicos, con el fin de evitar que un programa por error pueda programar la EEPROM, manipulando simplemente los bits del EECON1.

Set de instrucciones

Cada instrucción de los PIC16F84 y PIC16F877 es una palabra de 14 bits, dividida entre el OPCODE, la cual especifica el tipo de instrucción y uno o más operandos según la instrucción. Se presenta a continuación el listado de instrucciones de estos microcontroladores.

INSTRUCCIONES QUE MANEJAN REGISTROS				
Nemónicos	**operandos**	**Descripción**	**Ciclos**	**Flags**
ADDWF	f,d	Suma W y f	1	C, DC, Z
ANDWF	f,d	AND W con f	1	Z
CLRF	f	Borra f	1	Z
CLRW	---	Borra W	1	Z
COMF	f,d	Complementa f	1	Z
DECF	f,d	Decrementa f	1	Z
DECFSZ	f,d	Decrementa f, si es 0 salta	1 (2)	Ninguno
INCF	f,d	Incrementa f	1	Z
INCFSZ	f,d	Incrementa f, si es 0 salta	1	Ninguno
IORWF	f,d	OR entre W y f	1	Z
MOVF	f,d	Mueve f	1	Z
MOVWF	f	Mueve W a f	1	Ninguno
NOP	---	No opera	1	Ninguno
RLF	f,d	Rota f a la izqda. a través del carry	1	C
RRF	f,d	Rota f a la dcha. a través del carry	1	C
SUBWF	f,d	Resta a f el reg. W	1	C, DC, Z
SWAPF	f,d	Intercambia f	1	Ninguno
XORWF	f,d	XOR de W con f	1	Z
INSTRUCCIONES QUE MANIPULAN BITS				
BCF	f,b	Borra bit de f	1	Ninguno

BSF	f,b	Pone a 1 el bit de f	1	Ninguno
BTFSC	f,b	Comprueba un bit de f y salta si es 0	1 (2)	Ninguno
BTFSS	f,b	Comprueba un bit de f y salta si es 1	1 (2)	Ninguno
INSTRUCCIONES DE CONTROL Y DE OPERANDOS INMEDIATOS				
ANDLW	k	AND inmediato con W	1	Z
CALL	k	Llamada a subrutina	2	Ninguno
CLRWDT	k	Borra Watchdog	1	TO, PD
GOTO	k	Salto incondicional	2	Ninguno
IORLW	k	OR inmediato con W	1	Z
MOVLW	k	Mueve a W un valor inmediato	1	Ninguno
OPTION	k	Carga el registro OPTION	1	Ninguno
RETLW	k	Retorno y carga de W	2	Ninguno
SLEEP	---	Pasa a estado de reposo	1	TO, PD
TRIS	f	Carga el registro	1	Ninguno
XORLW	k	OR exclusiva a W	1	Z

LECCIÓN 4: PUERTOS I/O Y PRINCIPALES MÓDULOS EN LOS PIC.

Puertos de Entrada/Salida

- **Microcontrolador PIC16F84**

El microcontrolador PIC16F84, cuenta con dos puertos bidireccionales: Puerto A y Puerto B. Algunos pines de estos puertos son multiplexados con una función alternativa de los periféricos del dispositivo. En general, cuando un periférico es habilitado, el pin correspondiente no puede ser utilizado como pin de E/S de propósito general. A continuación se describen los puertos de este microcontrolador.

Puerto A: este puerto bidireccional tiene un tamaño de 5 bits (RA4:RA0). Tiene además 2 registros asociados que se muestran en la siguiente tabla y que corresponden a:

- Control de dirección de los pines del puerto A (TRISA)
- Estado de los pines del puerto A (PORTA)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are unimplemented, read as '0'.

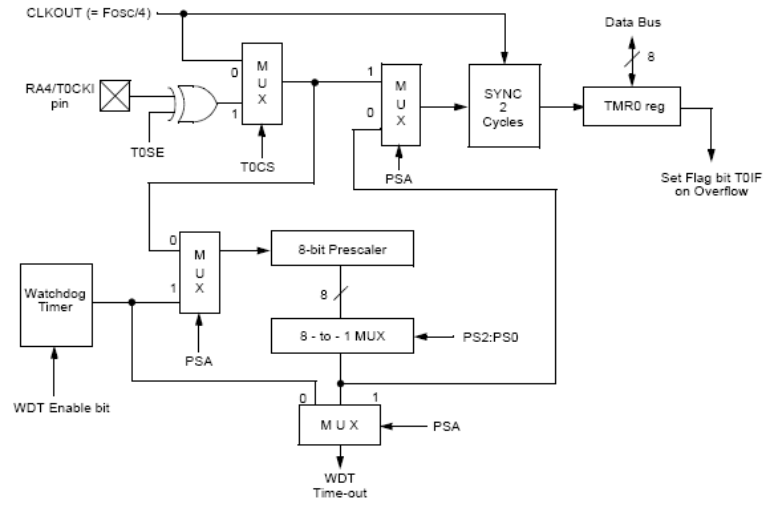
Puerto B: este puerto bidireccional tiene un tamaño 8 bits (RB7:RB0). Tiene además 4 registros asociados que se muestran en la siguiente tabla.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
0Bh,8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

Timer0: el módulo temporizador/contador de 8 bits TMR0 puede ser leído y escrito continuamente y se puede seleccionar un reloj interno o uno externo para trabajar con el. Además cuenta con preescalador programable de 8 bits, y con interrupción por desbordamiento desde FFh a 00h. El diagrama de bloques del módulo se muestra en la siguiente figura y los registros relacionados con el Timer0 se presentan posteriormente.

Figura 85. ***Timer0***[114]



Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
01h	TMR0	Timer0 Module Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	---1 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

Configuración del oscilador

El PIC16F84 puede operar con cuatro configuraciones de oscilador. El usuario puede seleccionar la más conveniente, según su necesidad

- LT: baja potencia de cristal
- XT: cristal/ resonador
- HS: cristal de alta velocidad/ resonador
- CS: resistor/ capacitor

La siguiente tabla muestra los valores de capacitor para las configuraciones: LP, XT y HS; junto con el valor del rango de frecuencia de resonancia.

[114] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/devicedoc/30430c.pdf

Mode	Freq	OSC1/C1	OSC2/C2
LP	32 kHz 200 kHz	68 - 100 pF 15 - 33 pF	68 - 100 pF 15 - 33 pF
XT	100 kHz 2 MHz 4 MHz	100 - 150 pF 15 - 33 pF 15 - 33 pF	100 - 150 pF 15 - 33 pF 15 - 33 pF
HS	4 MHz 20 MHz	15 - 33 pF 15 - 33 pF	15 - 33 pF 15 - 33 pF

Interrupciones

El PIC16F84 tiene cuatro fuentes de interrupción:

- Interrupción externa a través del pin RBO/INT
- Interrupción por desbordamiento del TMR0
- Interrupción PORTB (RB7:RB4)
- Interrupción de la memoria EEPROM cuando la escritura ha finalizado.

***Watchdog* (perro guardián)**

El temporizador *watchdog* es un recurso de vigilancia del microcontrolador. Cuenta libremente con el oscilador RC del chip, por lo tanto no necesita componentes externos. Tiene un período nominal de finalización de 18ms.

Durante la operación normal del microcontrolador, una finalización del temporizador *watchdog* genera un reinicio o RESET; pero si el dispositivo está en modo SLEEP provoca que se retome nuevamente la operación normal del dispositivo. El wachtdog puede ser habilitado o deshabilitado por software.

- **Microcontrolador PIC16F877**

Puertos de Entrada/Salida

El microcontrolador 16F877 cuenta con cinco puertos de E/S (A, B, C, D, E), los cuáles suman 33 pines bidireccionales para el trabajo con diversas señales externas. Hay que destacar que algunos pines de E/S están multiplexados con una función alternativa de los periféricos del dispositivo. En general, cuando un

periférico es habilitado el pin correspondiente no puede ser utilizado como pin de E/S de propósito general.

Puerto A: este puerto bidireccional tiene un tamaño de 6 bits (RA5:RA0). Tiene además 3 registros asociados que se muestran en la tabla y que corresponden a:

Estado de los pines del puerto A (PORTA)

Control de dirección de los pines del puerto A (TRISA)

El periférico de conversión analógico digital (ADCON1)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.
Shaded cells are not used by PORTA.

Puerto B: este puerto bidireccional tiene un tamaño de 8 bits (RB7:RB0). Tiene además 3 registros asociados que se muestran en la figura y que corresponden a:

Estado de los pines del puerto B (PORTB)

Control de dirección de los pines del puerto B (TRISB)

Interrupción externa pin RB0 (OPTION-REG bit INTEDG)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

Puerto C: este puerto bidireccional tiene un tamaño de 8 bits (RC7:RC0). Tiene además 2 registros asociados que se muestran en la figura y que corresponden a:

Estado de los pines del puerto C (PORTC)

Control de dirección de los pines del puerto C (TRISC)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

Puerto D: este puerto bidireccional tiene un tamaño de 8 bits (RD7:RD0) con entradas bufferadas de *schmit trigger* (disparo). Puede ser configurado como puerto paralelo esclavo y en este modo los buffers de entrada son TTL. Tiene además 3 registros asociados que se muestran en la figura y que corresponden a:

Estado de los pines del puerto D (PORTD)

Control de dirección de los pines del puerto D (TRISD)

Control de configuración puerto paralelo esclavo (TRISE)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

Puerto E: este puerto bidireccional tiene un tamaño de 3 bits (RE2:RE0), con entradas bufferadas de *schmit trigger* (disparo). Puede ser configurado como puerto paralelo esclavo, en este modo los buffers de entrada son TTL. Tiene además 3 registros asociados que se muestran en la figura y que corresponden a:

Operación de lectura / escritura de los pines RE cuando son entradas analógicas o puerto paralelo esclavo (PORTE)

Control de configuración puerto paralelo esclavo y control de la dirección de los pines RE (TRISE)

Configuración de entrada/salida digital (ADCON1)

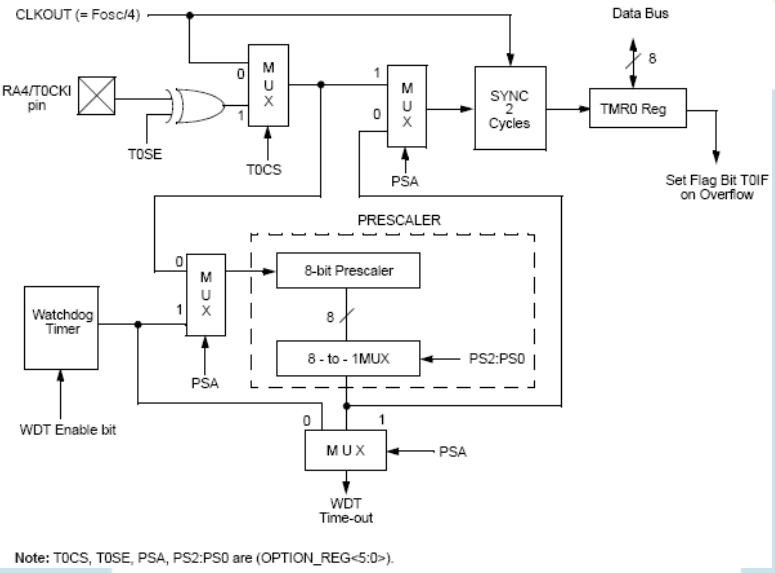
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

Módulo Timer0

Este temporizador/contador de 8 bits, tiene las características de ser de escritura y lectura, preescalador de 8 bits programable por software, selección de reloj interno o externo e interrupción por desbordamiento. La siguiente figura muestra el diagrama de bloques y la tabla con los registros relacionados con el Timer0.

Figura 86. ***Timer0 PIC16F877***[115]



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
01h,101h	TMR0	Timer0 Module's Register								xxxx xxxx	uuuu uuuu
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
81h,181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.
Shaded cells are not used by Timer0.

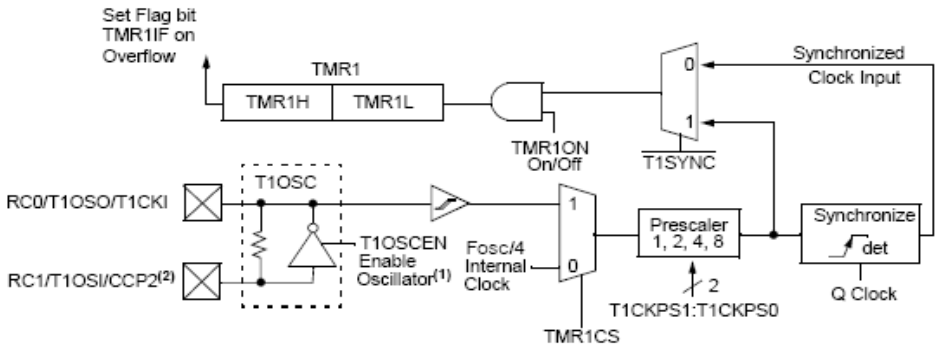
Módulo Timer1

El timer1 es un módulo contador/temporizador de 16 bits que consta de dos registros de 8 bits, los cuales pueden ser escritos y leídos continuamente. El módulo puede operar como temporizador o como contador. En el modo

[115] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

temporizador el TIMER1 se incrementa cada ciclo de instrucción, en el modo contador se incrementa con el flanco de bajada de la entrada de reloj externa. El diagrama de bloques del módulo se muestra en la siguiente figura y la tabla muestra los registros relacionados con el Timer0.

Figura 87. ***Timer1 PIC16F877A***[116]



Note 1: When the T1OSCEN bit is cleared, the inverter is turned off. This eliminates power drain.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.
Note 1: Bits PSPIE and PSPIF are reserved on the PIC16F873/876; always maintain these bits clear.

Módulo de Puerto Serial Maestro Síncrono

Es una interfaz serial utilizada para comunicarse con otros periféricos o microcontroladores. Estos dispositivos periféricos pueden ser memorias EEPROM seriales, registros de desplazamiento, manejadores de display, conversores A/D entre otros. El módulo MSSP puede operar en dos modos: interfaz serial periférica (SPI) ó circuito inter-integrado (I^2C)

a) Interfaz Serial Periférica (SPI): El SPI permite transmitir y recibir datos de 8 bits simultáneamente. Además soporta las cuatro modalidades que son:

[116] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

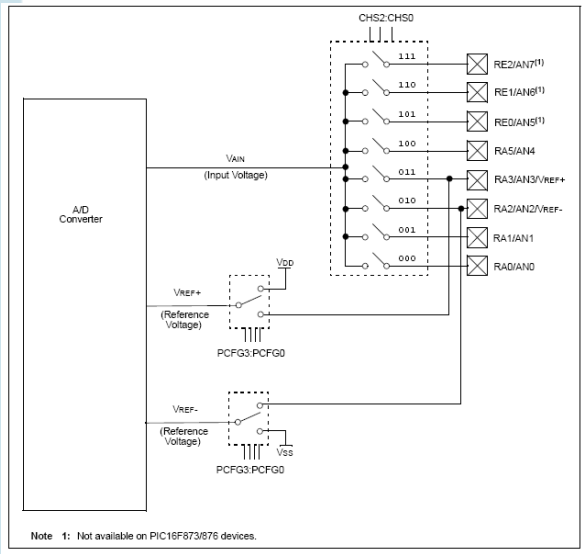
- Salida serial de dato (SDO)
- Entrada serial de dato (SDI)
- Reloj serial (SCK)
- Selección esclavo ($\overline{SS}$)

b) Circuito inter- integrado (I^2C): El módulo soporta todas las configuraciones maestro y esclavo, provee interrupciones por hardware de los bits de arranque y parada y determina cuando libera el bus.

Conversor analógico digital

El conversor analógico/digital del PIC16F877 tiene 8 entradas multiplexadas y una resolución máxima de 10 bits.

Figura 88. ***Conversión Análoga - Digital***[117]



La señal analógica de entrada es muestreada y cargada en un capacitor a la entrada del conversor; el cual genera como resultado un valor digital del nivel analógico a través de aproximaciones sucesivas.

[117] Extraído el 18 Octubre de 2009 desde http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf

La conversión analógico/digital de la señal analógica de entrada tiene una resolución máxima de 10 bits. El módulo cuenta también con un nivel alto y bajo de voltaje de referencia programado por software, escogiendo entre los pines V_{SS}, V_{DD}, RA3 y RA2.

El conversor A/D tiene la característica de operar mientras el microcontrolador se encuentra en modo SLEEP. El reloj del A/D es derivado del oscilador interno del conversor.

El tiempo mínimo de adquisición es de 19,72 μs. El tiempo de conversión del A/D por bit se denomina T_{AD}; en el caso del módulo se necesitan 12 T_{AD} para 10 bits. La fuente del reloj de conversión se selecciona por software y pueden ser: $2T_{OSC}$, 8 T_{OSC}, 32 T_{OSC} ó módulo oscilador interno del ADC. La siguiente tabla muestra los registros asociados con el A/D.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on $\overline{MCLR}$, WDT
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	0000 00-0
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	--0u 0000
89h(1)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
09h(1)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.
Note 1: These registers/bits are not available on the 28-pin devices.

LECCIÓN 5: HERRAMIENTAS DE DESARROLLO Y EJERCICIOS BÁSICOS.

Microchip y otras empresas ofrecen varias herramientas de desarrollo que permiten y facilitan la elaboración y depuración de los programas, las herramientas de desarrollo de microchip operan en el entorno de programación de MPLABIDE (IDE-“Entorno de Desarrollo Integrado”), algunas de las herramientas mas usuales se dividen en:

- Generación de código.

  - MPASM, MPLAB-C y MP-DriveWay.
- Depuración de los programas.
  - PICMASTER y ICEPICt emuladores en circuito (In-Circuit) y MPLABSIM como simulador de programas.
- Dispositivos programadores.
  - PICSTART Plus y el PROMATE II de Microchip
  - ICPROG con varios programadores de libre distribución (Pro-Pic Program).

  Los anteriores dispositivos permiten almacenar el programa en la memoria del microcontrolador, para esto utilizan usualmente los puertos serie o paralelo del computadora personal usando un programa especialmente diseñado para enviar el código maquina ejecutable hacia la memoria del microcontrolador, estos programadores en sus versiones mas sofisticadas y su respectivo hardware permiten seleccionar la referencia del dispositivo a programar.

- Tarjetas de evaluación de los productos y sistemas de desarrollo. Las tarjetas de demostración permiten probar y evaluar las prestaciones de cada dispositivo, son ampliamente utilizadas en laboratorios de enseñanza y para práctica de usuarios aprendices. Los sistemas de desarrollo son combinaciones de hardware y software que permiten realizar todo el proceso de desarrollo de un prototipo desde la edición, compilación, depuración, y grabación del programa en la memoria del micro, además permite establecer las conexiones requeridas con los periféricos de entrada-salida dispuestos sobre la tarjeta, cuando se llega al punto donde el sistema funciona adecuadamente se procede a la construcción del sistema final sobre una tarjeta de circuito impreso para la fabricación en serie.

Ejercicios básicos con microprocesadores

La comprensión y desarrollo práctico de estos ejercicios permite una mayor asimilación, aprendizaje y habilidad en la programación, prueba y ejecución de aplicaciones basadas en microcontroladores.

- **Intermitencia de un LED**: este ejercicio explora las directivas básicas de programación y la utilización de las instrucciones más habituales.

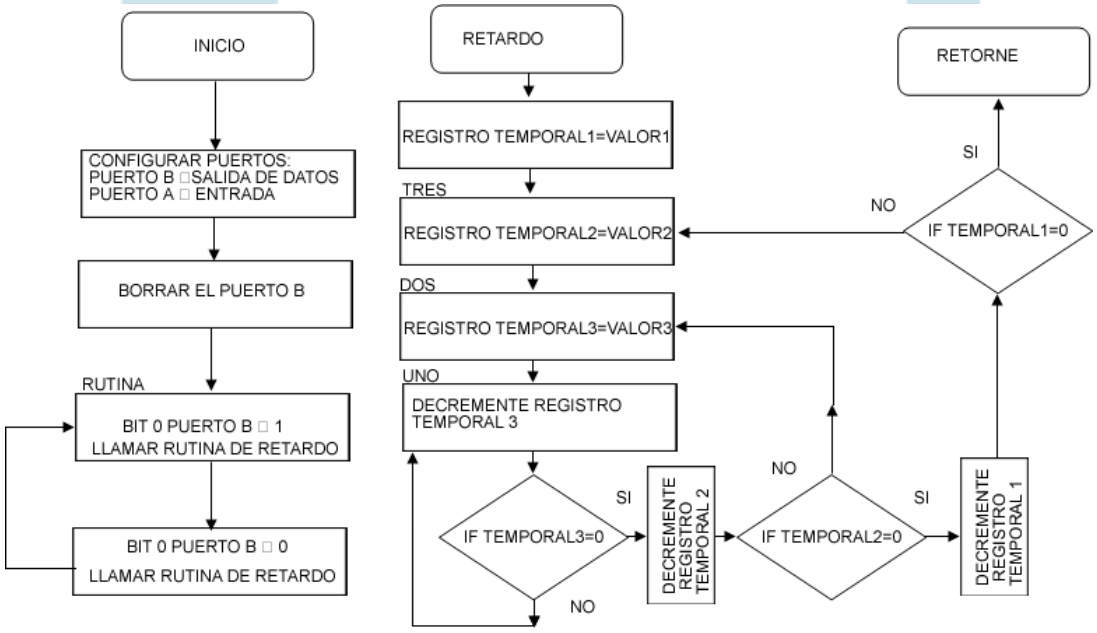
Entradas y salidas: en este caso no hay entradas solo una salida donde se conectara un diodo LED, elegimos el puerto “B” en su pin RB0 como salida de datos.

Pseudocódigo:

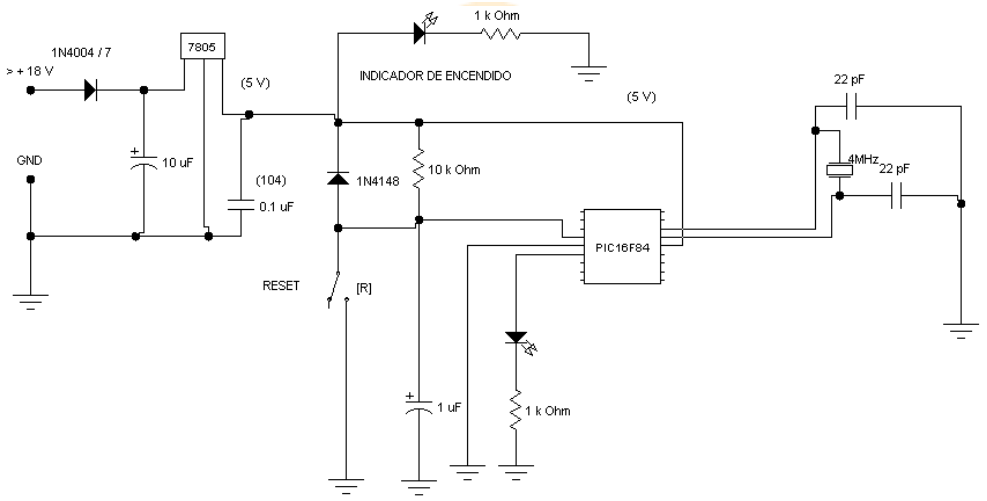
INICIA:
CONFIGURAR PUERTOS:
PUERTO B →SALIDA DE DATOS
PUERTO A → ENTRADA
RUTINA: BORRAR EL PUERTO B
BIT 0 PUERTO B → 1
LLAMAR RUTINA DE RETARDO
BIT 0 PUERTO B → 0
LLAMAR RUTINA DE RETARDO
SALTAR A RUTINA
TERMINA

SUBRUTINA "RETARDO":
REGISTRO TEMPORAL1=VALOR1
TRES REGISTRO TEMPORAL2=VALOR2
DOS REGISTRO TEMPORAL3=VALOR3
DECREMENTE REGISTRO TEMPORAL 3
UNO IF TEMPORAL3=0
THEN DECREMETNE TEMPORAL2
IF TEMPORAL2=0
THEN DECREMENTE TEMPORAL1
IF TEMPORAL1=0
THEN RETORNE
ELSE SALTAR A → TRES
ELESE SALTAR A → DOS
ELSE SALTAR A → UNO
TERMINA

Diagrama de Flujo



Montaje:



Código ensamblador

```
TITLE "PIC16F84A EEPROM PROGRAM"
        LIST P=16F84A,F=INHX32
        #INCLUDE <P16F84A.INC>
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;ENCENDIDO INTERMITENTE DE UN LED (PARA EFECTOS PRACTICOS SE INCORPORA UN RETARDO)
;PINES DE SALIDA LEDS ==> B0
;PINES DE ENTRADA ==> "NINGUNO"
;-----------------------------------------------------------------------------------------
;REGISTROS DE PROPOSITO ESPECIAL MAS UTILIZADOS
INDF            EQU     00H     ;REGISTRO PARA DIRECCIONAMIENTO INDIRECTO
TMR0            EQU     01H     ;REGISTRO TIMER 0
PCL             EQU     02H     ;PROGRAM COUNTER PARTE BAJA
STATUS  EQU     03H     ;REGISTRO DE ESTADO DEL PIC
FSR             EQU     04H     ;REGISTRO UTILIZADO COMO APUNTADOR EN DIR. INDIRECTO
PTA             EQU     05H     ;PUERTO A DEL PIC
PTB             EQU     06H     ;PUERTO B DEL PIC
EEDATA  EQU     08H     ;DATO IN/OUT EN EEPROM
EEADR           EQU     09H     ;DIRECCION DE IN/OUT DE EEPROM
PCLATCH         EQU     0AH     ;CERROJO PARTE ALTA DEL PROGRAM COUNTER
INTCON          EQU     0BH     ;REGISTRO DE INTERRUPCIONES
OPCION  EQU     81H     ;OPCIONES DE CONFIGURACION INT TMP PULLUP PREESCALER
TRISA           EQU     85H     ;RA0,RA1,RA2,RA3,RA4 ENTRADAS
TRISB           EQU     86H     ;RB0,RB1,RB2,RB3,RB4,RB5,RB6,RB7 COMO SALIDAS
EECON1          EQU     88H     ;B4=EEIF/FIN WR/*B3=WRERR/ERROR WR/*B2=WREN/PERMISO
WR/*B1/WR/*B0/RD/
EECON2          EQU     89H     ;REG SEGURIDAD PROCESO EEPROM
;-----------------------------------------------------------------------------------------
;DEFINICION REGISTROS DE PROPOSITO GENERAL PARA ESTE PROGRAMA
TMP1    EQU     0CH     ;REGISTRO TEMPORAL 1 SIRVE AL CICLO MAS EXTERNO DEL RETARDO
TMP2    EQU     0DH     ;REGISTRO TEMPORAL 2 SIRVE AL CICLO INTERMEDIO DEL RETARDO
TMP3    EQU     0EH     ;REGISTRO TEMPORAL 3 SIRVE AL CICLO MAS INTERNO DEL RETARDO
;-----------------------------------------------------------------------------------------
;DEFINICION DE BITS
W       EQU     0       ;REGISTRO DE TRABAJO
F       EQU     1       ;REGISTRO
C       EQU     0       ;FLAG DE CARRY
Z       EQU     2       ;FLAG DE CERO
RD      EQU     0       ;RD=1 CICLO DE LECTRURA DE LA EEPROM
```

```
WR      EQU     1       ;WR=1 CICLO DE ESCRITURA DE LA EEPROM
WREN    EQU     2       ;WREN=1 AUTORIZA PERMISO DE ESCRITURA EN LA EEPROM
B0      EQU     0       ;BIT 0
B1      EQU     1       ;BIT 1
B2      EQU     2       ;BIT 2
B3      EQU     3       ;BIT 3
B4      EQU     4       ;BIT 4
B5      EQU     5       ;BIT 5
B6      EQU     6       ;BIT 6
B7      EQU     7       ;BIT 7
;-------------------------------------------------------------------------------
;DEFINICION DE CONSTANTES
VAL1    EQU     20H     ;CONSTANTE DE RETARDO RECARGA AL REGISTRO TMP1
VAL2    EQU     30H     ;CONSTANTE DE REATRDO RECARGA AL REGISTRO TMP2
VAL3    EQU     40H     ;CONSTANTE DE RETARDO RECARGA AL REGISTRO TMP3
;-------------------------------------------------------------------------------
;MACROS
#DEFINE BANK1   BSF     STATUS, 5       ;ENCARGADO DE PASAR AL BANCO DE MEMORIA CERO
#DEFINE BANK0   BCF     STATUS, 5       ;ENCARGADO DE PASAR AL BANCO DE MEMORIA UNO
;-------------------------------------------------------------------------------
;INICIO DEL PROGRAMA
;-------------------------------------------------------------------------------
        ORG     00H
;-------------------------------------------------------------------------------
;CONFIGURACION DE PUERTOS
;-------------------------------------------------------------------------------
        BANK1
        MOVLW   00H     ;SALIDA DE DATOS POR LOS PINES B7 B6 B5 B4 B3 B2 B1 B0
        MOVWF   TRISB
        MOVLW   1FH     ;ENTRADA DE DATOS A4 A3 A2 A1 A0
        MOVWF   TRISA
        BANK0
;-------------------------------------------------------------------------------
;PROGRAMA DE INICIO
;-------------------------------------------------------------------------------
INICIO  CLRF    PTB     ;BORRAR EL PUERTO SE APAGAN LOS LEDs
RUTIN   BSF     PTB,0   ;COLOCA UN UNO EN EL BIT CERO DE PUERTO B
        CALL    RET     ;LLAMAR A RUTINA DE RETARDO
        BCF     PTB,0   ;COLOCA UN CERO EN EL BIT CERO DE PUERTO B
        CALL    RET     ;LLAMAR A RUTINA DE RETARDO
        GOTO    RUTIN   ;COMENZAR DE NUEVO LA RUTINA
;-------------------------------------------------------------------------------
;LLAMA LA RUTINA DE RETARDO ESPECIAL VARIABLE
;-------------------------------------------------------------------------------
RET     MOVLW   VAL1            ;VALOR MAXIMO / MINIMO VARIABLE DE TEMPORIZADOR
        MOVWF   TMP1            ;REGISTRO TEMPORAL 1 TIEMPOS GRANDES
TRES    MOVLW   VAL2            ;VALOR FIJO
        MOVWF   TMP2            ;REGISTRO TEMPORAL 2 TIEMPOS MEDIANOS
DOS     MOVLW   VAL3            ;VALOR FIJO
        MOVWF   TMP3            ;REGISTRO TEMPORAL 3 TIEMPOS PEQUEÑOS
UNO     DECFSZ  TMP3,   F
        GOTO    UNO
        DECFSZ  TMP2,   F
        GOTO    DOS
        DECFSZ  TMP1,   F
        GOTO    TRES
        RETURN
;-------------------------------------------------------------------------------
        END
;-------------------------------------------------------------------------------
```

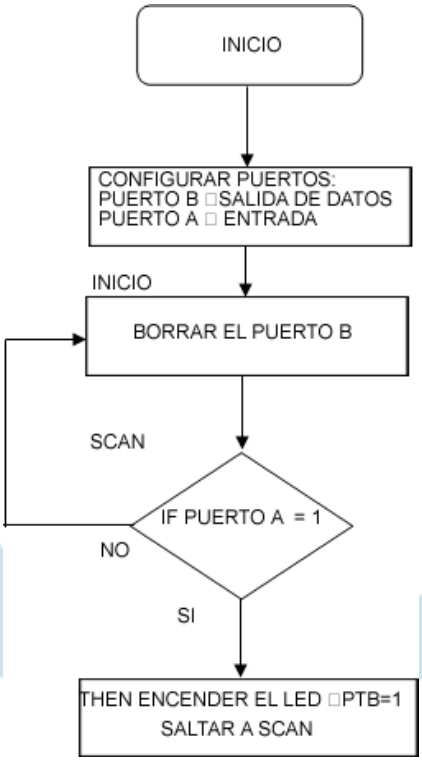
- **Encendido y apagado de un LED**: Este ejercicio propone recibir la señal de un pulsador y reflejar este estado mediante el encendido de un LED en caso de ser pulsado y apagar el LED en caso de soltar el pulsador.

Entradas y salidas: para este ejercicio existe un pulsador el cual es conectado al puerto a en su pin RA0, hay una salida donde se conectara un diodo LED, elegimos el puerto "B" en su pin RB0 como salida de datos.

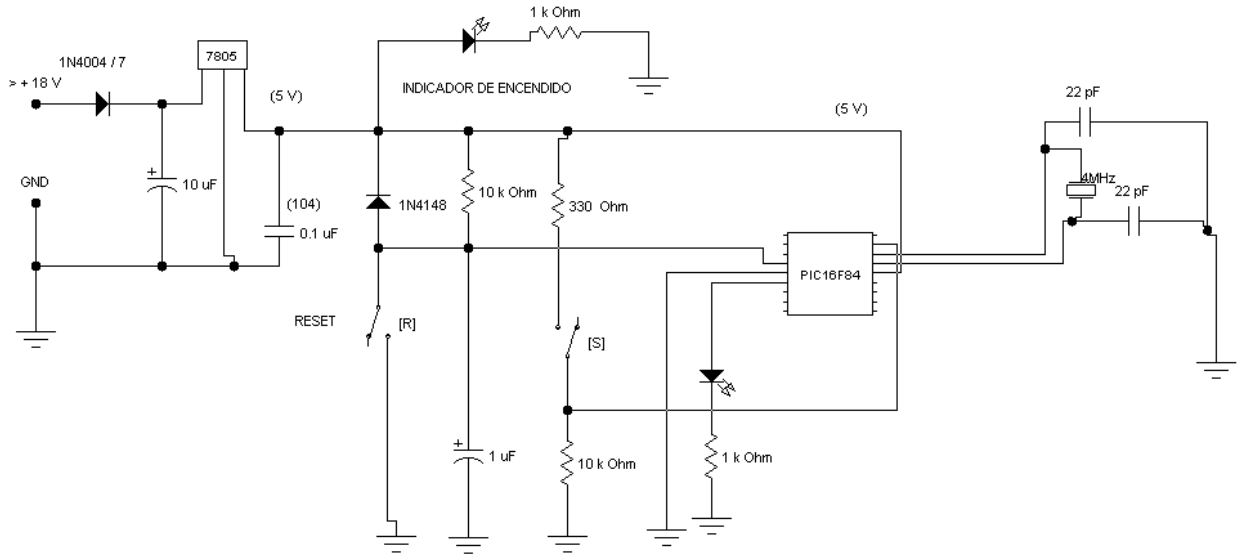
Pseudocódigo:

```
INICIA:
        CONFIGURAR PUERTOS:
                PUERTO B →SALIDA DE DATOS
                PUERTO A → ENTRADA
        INICIO:  BORRAR EL PUERTO B
        SCAN:           IF PUERTO A  → 1
                                THEN ENCENDER EL LED →PTB=1
                                SALTAR A SCAN
                        ELSE    SALTAR A INICIO
TERMINA
```

Diagrama de Flujo



Montaje:



Código ensamblador

```
TITLE "PIC16F84A EEPROM PROGRAM"
        LIST P=16F84A,F=INHX32
        #INCLUDE <P16F84A.INC>
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;ENCENDIDO DE UN LED POR ACCION DE UN PULSADOR(PARA EFECTOS PRACTICOS SE INCORPORA UN
RETARDO)
;PINES DE SALIDA LEDS ==> B0
;PINES DE ENTRADA ==> A0 PULSADOR
;-----------------------------------------------------------------------------------------
;REGISTROS DE PROPOSITO ESPECIAL
INDF    EQU     00H     ;REGISTRO PARA DIRECCIONAMIENTO INDIRECTO
TMR0    EQU     01H     ;REGISTRO TIMER 0
PCL     EQU     02H     ;PROGRAM COUNTER PARTE BAJA
STATUS  EQU     03H     ;REGISTRO DE ESTADO DEL PIC
FSR     EQU     04H     ;REGISTRO UTILIZADO COMO APUNTADOR EN DIR. INDIRECTO
PTA     EQU     05H     ;PUERTO A DEL PIC
PTB     EQU     06H     ;PUERTO B DEL PIC
EEDATA  EQU     08H     ;DATO IN/OUT EN EEPROM
EEADR   EQU     09H     ;DIRECCION DE IN/OUT DE EEPROM
PCLATCH         EQU     0AH     ;CERROJO PARTE ALTA DEL PROGRAM COUNTER
INTCON  EQU     0BH     ;REGISTRO DE INTERRUPCIONES
OPCION  EQU     81H     ;OPCIONES DE CONFIGURACION INT TMP PULLUP PREESCALER
TRISA   EQU     85H     ;RA0=PULSADOR,RA1,RA2,RA3,RA4 ENTRADAS
TRISB   EQU     86H     ;RB0,RB1,RB2,RB3,RB4,RB5,RB6,RB7 COMO SALIDAS
EECON1 EQU      88H     ;B4=EEIF/FIN WR/*B3=WRERR/ERROR WR/*B2=WREN/PERMISO WR/*B1/WR/*B0/RD/
EECON2 EQU      89H     ;REG SEGURIDAD PROCESO EEPROM
;-----------------------------------------------------------------------------------------
;DEFINICION REGISTROS DE PROPOSITO GENERAL
TMP1    EQU     0CH     ;REGISTRO TEMPORAL 1 SIRVE AL CICLO MAS EXTERNO DEL RETARDO
TMP2    EQU     0DH     ;REGISTRO TEMPORAL 2 SIRVE AL CICLO INTERMEDIO DEL RETARDO
TMP3    EQU     0EH     ;REGISTRO TEMPORAL 3 SIRVE AL CICLO MAS INTERNO DEL RETARDO
```

```
;-------------------------------------------------------------------------------
;DEFINICION DE BITS
W       EQU     0       ;REGISTRO DE TRABAJO
F       EQU     1       ;REGISTRO
C       EQU     0       ;FLAG DE CARRY
Z       EQU     2       ;FLAG DE CERO
RD      EQU     0       ;RD=1 CICLO DE LECTRURA DE LA EEPROM
WR      EQU     1       ;WR=1 CICLO DE ESCRITURA DE LA EEPROM
WREN    EQU     2       ;WREN=1 AUTORIZA PERMISO DE ESCRITURA EN LA EEPROM
B0      EQU     0       ;BIT 0
B1      EQU     1       ;BIT 1
B2      EQU     2       ;BIT 2
B3      EQU     3       ;BIT 3
B4      EQU     4       ;BIT 4
B5      EQU     5       ;BIT 5
B6      EQU     6       ;BIT 6
B7      EQU     7       ;BIT 7
;-------------------------------------------------------------------------------
;DEFINICION DE CONSTANTES
VAL1    EQU     20H     ;CONSTANTE DE RETARDO RECARGA AL REGISTRO TMP1
VAL2    EQU     30H     ;CONSTANTE DE REATRDO RECARGA AL REGISTRO TMP2
VAL3    EQU     40H     ;CONSTANTE DE RETARDO RECARGA AL REGISTRO TMP3
;-------------------------------------------------------------------------------
;MACROS
#DEFINE BANK1   BSF     STATUS, 5       ;ENCARGADO DE PASAR AL BANCO DE MEMORIA CERO
#DEFINE BANK0   BCF     STATUS, 5       ;ENCARGADO DE PASAR AL BANCO DE MEMORIA UNO
;-------------------------------------------------------------------------------
;INICIO DEL PROGRAMA
;-------------------------------------------------------------------------------
        ORG     00H
;-------------------------------------------------------------------------------
;CONFIGURACION DE PUERTOS
;-------------------------------------------------------------------------------
        BANK1
        MOVLW   00H     ;SALIDA DE DATOS POR LOS PINES B7 B6 B5 B4 B3 B2 B1 B0
        MOVWF   TRISB
        MOVLW   1FH     ;ENTRADA DE DATOS A4 A3 A2 A1 A0
        MOVWF   TRISA
        BANK0
;-------------------------------------------------------------------------------
;PROGRAMA DE INICIO
;-------------------------------------------------------------------------------
INICIO  CLRF    PTB             ;BORRADO DEL PUERTO B
SCAN    BTFSS   PTA,    B0      ;PRUEBE BIT 0 DEL PUERTO A (RA0)
        GOTO    INICIO          ;SI ES CERO SALTA A INICIO Y NO ENCIENDE EL LED
        BSF     PTB,    B0      ;SI ES UNO ENCIENDE EL LED
        GOTO    SCAN            ;SALTAR A LA RUTINA DE TESTEO

;-------------------------------------------------------------------------------
        END
;-------------------------------------------------------------------------------
```

CAPÍTULO 6: MICROCONTROLADORES DE 8 BITS MOTOROLA FREESCALE MC68H(R)C908/JL3/JK3/JK1

LECCIÓN 1: FAMILIAS DE MICROCONTROLADORES MOTOROLA FREESCALE.

La elección de un MICROCONTROLADOR FREESCALE frente a otros más conocidos como el 80XX de Intel, el PIC de Microchip, el ST-62XX de SGS-Thomson o el Z86XX de Zilog, se debe a características como su bajo precio, velocidad, reducido consumo de energía, tamaño, facilidad de uso, fácil programación y lo mejor de todo son los recursos que la gran mayoría de estos microcontroladores presentan a la hora de diseñar cualquier aplicación. Es por ello que los microcontroladores Freescale se encuentran hoy en día en la gran mayoría de aplicaciones industriales, de comunicaciones y control. Si se desea investigar al respecto, por ejemplo, en el caso de la industria automotriz, la cual en la actualidad es una de las que requiere mayor precisión en el desarrollo de procesos de control, instrumentación, entre otras. Casi el 90% de sus componentes son gobernados por microcontroladores Freescale, debido a sus bondades, estabilidad, inmunidad al ruido y otros factores importantes que hacen decisiva su elección frente a otras marcas. (VESGA, 2007).

Tabla 23. ***Variación de la familia MC68H(R)C908/JL3/JK3/JK1***[118]

Device	Oscillator Option	FLASH Memory Size	Pin Count
MC68HC908JL3E	X-TAL	4096 bytes	28
MC68HRC908JL3E	RC		
MC68HC908JK3E	X-TAL	4096 bytes	20
MC68HRC908JK3E	RC		
MC68HC908JK1E	X-TAL	1536 bytes	20
MC68HRC908JK1E	RC		

[118] VESGA, 2007

Características generales del microcontrolador Motorola Freescale

Características de los Microcontroladores MC68H (R) C908JL3E:

- EMC versión mejorada de MC68H (R) C908JL3/JK3/JK1
- Arquitectura de alto rendimiento M68HC08
- Totalmente compatible con el código objeto de las familias M6805, M146805, y M68HC05
- Diseño de baja potencia; totalmente estática con los modos de parada y espera
- Máxima frecuencia de bus interno:
  - 8 MHz en la tensión de 5V
  - 4-MHz 3V en la tensión de funcionamiento
- Oscilador opciones:
  - Oscilador de cristal de MC68HC908JL3E/JK3E/JK1E
  - RC oscilador para MC68HRC908JL3E/JK3E/JK1E
- Programa de usuario en memoria FLASH
  - 4096 bytes para MC68H (R) C908JL3E/JK3E
  - 1536 bytes para MC68H (R) C908JK1E
- 128 bytes de memoria RAM on-chip
- 2 canales, temporizador de 16 bits Módulo de interfaz (TIM)
- 12 canales, 8 bits de analógico a digital (ADC)
- 23 de propósito general puertos I / O para MC68H (R) C908JL3E:
  - 7 interrupciones con teclado desplegable interior hasta (6 interrupciones para MC68HC908JL3E)
  - 10 drivers para LED
  - 2 × 25 mA open-drain E / S con pull-up
- 15 puertos I / O de propósito general para MC68H (R) C908JK3E/JK1E:
  - 1 interrupción por teclado con pull-up interno (MC68HRC908JK3E/JK1E solamente)
  - 4 drivers LED
  - 2 × 25 mA open-drain E / S con pull-up
  - ADC de 10 canales
- Características del Sistema de protección:
  - Optional computer operating properly (COP) reset
  - Opcional, de detección de bajo voltaje seleccionable con reset y puntos seleccionables de operación entre 3V y 5V
- Pin maestro de reset con pull-up internos y power-on reset
- IRQ1 con entrada Schmitt-trigger y pull-up programables
- Empaquetado de 28-pines PDIP, 28-pines SOIC, y de 48-pines LQFP para MC68H (R) C908JL3E
- Empaquetado de 20-pin PDIP y 20-pin SOIC para MC68H (R) C908JK3E/JK1E
- Características principales de la CPU08 :
  - Modelo de programación ampliado de HC05
  - Amplias funciones de control de bucle
  - 16 modos de direccionamiento (ocho más que la HC05)
  - Registro índice de 16-bit y apuntador de pila
  - Transferencia de datos memoria – memoria.
  - Instrucciones de binario – código decimal (BCD)
  - Soporte para lenguaje C

LECCIÓN 2: MODOS DE DIRECCIONAMIENTO Y DIAGRAMA DE PINES.

Modos de direccionamiento

Los Microcontroladores Freescale usan seis modos de direccionamiento que son:

Inherente Las instrucciones de direccionamiento inherente no tienen ningún operando, ya que el operando se define en el 'opcode' de 8-bits. Por ejemplo para borrar el acumulador, se usa la instrucción CLRA, que es una instrucción de un sólo ciclo para el HC08; el HC05 toma 3 ciclos. EJ. CLRA

Inmediato Las instrucciones de direccionamiento inmediato tienen los operandos, que siguen inmediatamente al 'opcode', de 8-bits o de 16-bits. Cargar el acumulador con 20 es una instrucción de dos bytes, que se ejecuta en ciclos de bus. Ej. LDA #20

Extendido Las instrucciones de direccionamiento extendido proporcionan direccionamiento absoluto a cualquier posición en los 64K del mapa de memoria, sin paginar. Requieren en total tres bytes para el 'opcode' más la dirección de 16-bits del operando. La mayoría de los ensambladores usan el modo directo más corto automáticamente, para cualquier acceso a los primeros 256 bytes del mapa de memoria. Ej. LDA $4000

Directo Las instrucciones de direccionamiento directo tienen la dirección de 8-bits del operando que sigue inmediatamente al 'opcode'. Por consiguiente, las instrucciones acceden directamente a los primeros 256 bytes de la memoria. Anteriormente, se hizo referencia a este tipo de acceso como página directa o página cero. Cargar el acumulador con el operando a la posición de memoria 40, es una instrucción de dos bytes que se ejecuta en 3 ciclos de bus. Ej. LDA $40

Indexado Los modos de direccionamiento indexado son claves para direccionar tablas y otras estructuras de datos de una manera eficaz. El direccionamiento indexado sin desplazamiento ('offset'), se refiere a lo que en la mayoría de otras arquitecturas es el direccionamiento del puntero indirecto. El valor en el registro de índice es la dirección o el puntero del operando. Los modos de direccionamiento con 'offset' proporcionan al 68HC08 una gran eficacia de código comparado con otras arquitecturas con sólo direccionamiento indirecto o direccionamiento indirecto con otro registro de 'offset'

- sin desplazamiento Ej. LDA ,X
- con desplazamiento de 8 bits Ej. LDA $40, X

- con desplazamiento de 16 bits Ej. LDA $4000, X

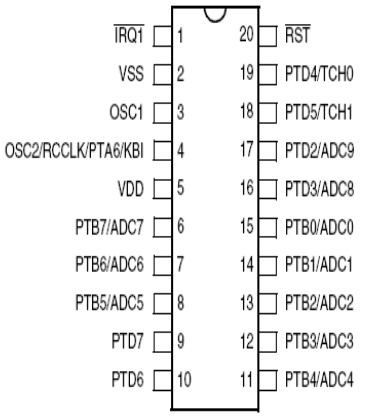
Relativo. Todas las instrucciones de bifurcación condicional usan el direccionamiento relativo Si la condición de bifurcación es verdad, el contador de programa se agrega al byte con signo, que sigue inmediatamente al 'opcode' de bifurcación. Esto da un rango de -128 a +127 bytes, para la bifurcación. Las instrucciones de salto o de salto a subrutina, se pueden usar para mover el contador de programa en cualquier parte de los 64K del mapa de memoria. Ej. BLT LOOP.

Como parte del proceso de aprendizaje y practica con microcontroladores Motorola Freescale, se hace referencia al material bibliográfico "Microcontroladores Motorola Freescale: Programación, familias y sus distintas aplicaciones industriales" que es uno de los pocos textos que tratan el tema en forma didáctica en español y de fácil consecución en las librerías del país.

Funciones y diagrama de pines

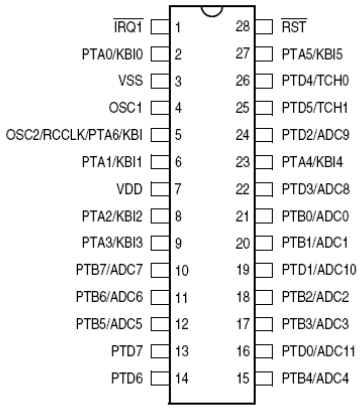
En la siguiente figura se observa la distribución de pines del MC68H(R)C908L3 y MC68H(R)C908JK3/JK1

Figura 89. ***JL3,JK3 y JK1***[119]



MC68H(R)C908JK3E/JK1E

Pins not available on 20-pin packages	
PTA0/KBI0	PTD0/ADC11
PTA1/KBI1	PTD1/ADC10
PTA2/KBI2	
PTA3/KBI3	
PTA4/KBI4	
PTA5/KBI5	
Internal pads are unconnected.	



MC68H(R)C908JL3E

[119] Vesga, 2007

Las siguiente tabla se discrimina la función(es) de pines, en el MC68H(R)C908JK3E/JK1E, no están disponibles los siguientes pines *PTA0, PTA1, PTA2, PTA3, PTA4, PTA5, PTD0, and PTD1*

Figura 90. ***Configuración de pines***[120]

PIN NAME	PIN DESCRIPTION	IN/OUT	VOLTAGE LEVEL
VDD	Power supply.	In	5V or 3V
VSS	Power supply ground	Out	0V
$\overline{RST}$	RESET input, active low. With Internal pull-up and schmitt trigger input.	Input	VDD
$\overline{IRQ1}$	External IRQ pin. With software programmable internal pull-up and schmitt trigger input. This pin is also used for mode entry selection.	Input	VDD to VDD+V_{HI}
OSC1	X-tal or RC oscillator input.	In	Analog
OSC2	MC68HC908JL3E/JK3E/JK1E: X-tal oscillator output, this is the inverting OSC1 signal.	Out	Analog
	MC68HRC908JL3E/JK3E/JK1E: Default is RC oscillator clock output, RCCLK. Shared with PTA6/KBI6, with programmable pull-up.	In/Out	VDD
PTA[0:6]	7-bit general purpose I/O port.	In/Out	VDD
	Shared with 7 keyboard interrupts KBI[0:6].	In	VDD
	Each pin has programmable internal pull-up device.	In	VDD
	PTA[0:5] have LED direct sink capability	In	VSS
PTB[0:7]	8-bit general purpose I/O port.	In/Out	VDD
	Shared with 8 ADC inputs, ADC[0:7].	In	Analog
PTD[0:7]	8-bit general purpose I/O port.	In/Out	VDD
	PTD[3:0] shared with 4 ADC inputs, ADC[8:11].	Input	Analog
	PTD[4:5] shared with TIM channels, TCH0 and TCH1.	In/Out	VDD
	PTD[2:3], PTD[6:7] have LED direct sink capability	In	VSS
	PTD[6:7] can be configured as 25mA open-drain output with pull-up.	In/Out	VDD

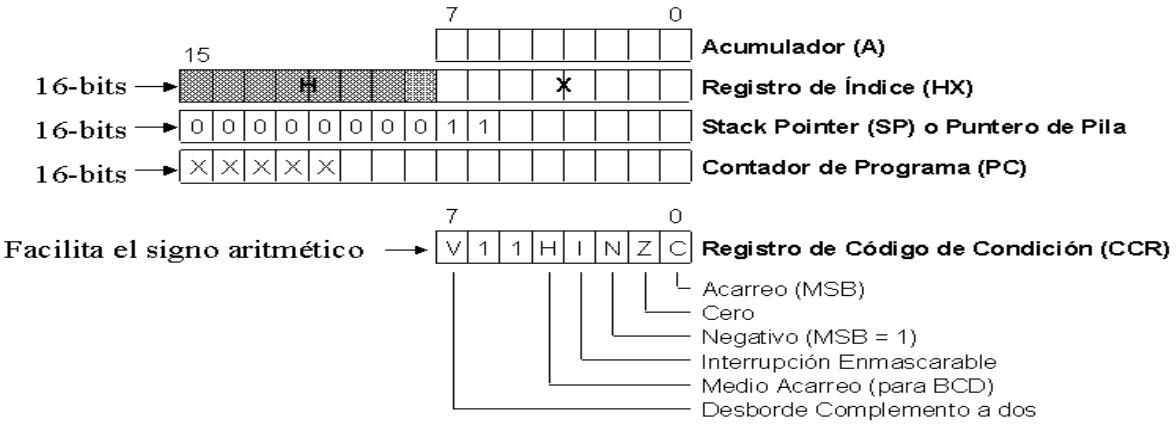
LECCIÓN 3: ARQUITECTURA, FUNCIONAMIENTO Y SET DE INSTRUCCIONES.

Arquitectura interna

El modelo de programación de la CPU del 68HC08 es una evolución del modelo de CPU del 68HC05 Cada 68HC08 implementa este modelo de CPU sin tener en

[120] Vesga, 2007

cuenta el tamaño o el conjunto de características propias. A continuación se verán las mejoras específicas que se incluyen en la arquitectura 68HC08:

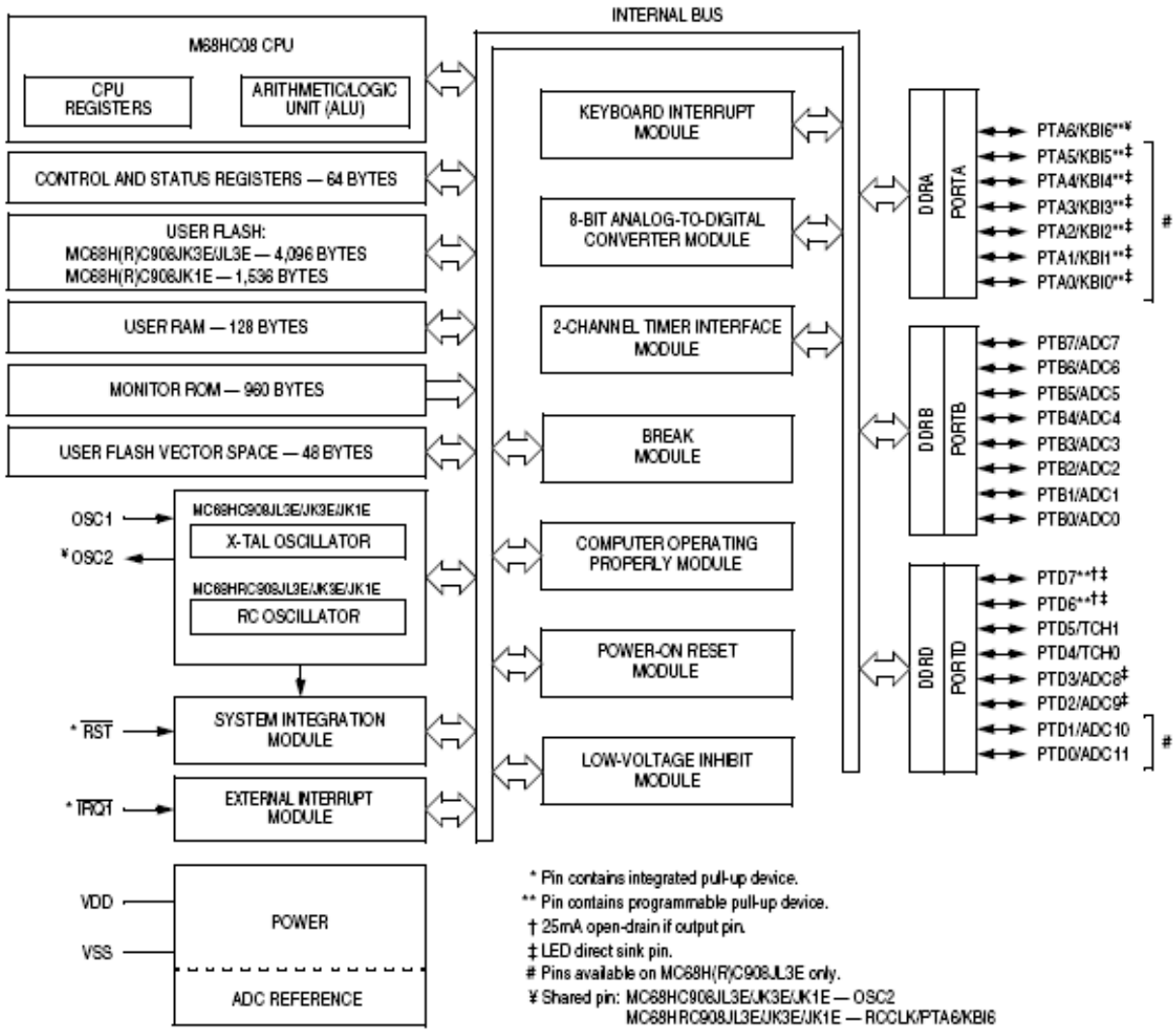


- El *registro de índice* se ha ampliado a 16-bits, ayudando a manejar mejor las tablas o estructuras de datos que son mayores de 256 bytes.
- El *puntero de pila* y el *contador de programa* son registros de 16-bits, sin tener en cuenta la memoria interna disponible. Más adelante se verá la pila (‘stack’).
- En *Power-On Reset* , el 68HC08 se parece al 68HC05. Durante el Reset, los 8 bits más altos del registro de índice HX, de los 16 bits que tiene, se ponen a cero y el puntero de pila se inicializa a $00FF como en el 68HC05. Considerando que la mayoría de 68HC08 tienen mayor cantidad de RAM disponible, es probable que el usuario relocalizará el puntero de pila (‘stack pointer’). Sin embargo, esta característica ayuda a mantener la compatibilidad operacional con el software existente del 68HC05.
- En la CPU del 68HC08 el *bit V,* del registro de código de condición (CCR) facilita los cálculos aritméticos con signo. Esta mejora permite a los programadores de lenguaje ensamblador compiladores, realizar cálculos de direccionamiento mucho mejor

El 68HC08 utiliza cuatro fases del reloj interno en cada ciclo de ejecución de la CPU. Si el 68HC08 está gobernado por un cristal, el ciclo de ejecución es un cuarto de la frecuencia del cristal.

A este ciclo se le llama ciclo de bus o ciclo de instrucción En el 68HC08, todos los tiempos de cada instrucción se especifican en ciclos de bus. Por ejemplo, un reloj de entrada de 32 MHz producirá una frecuencia de bus de 8 MHz. Un ciclo de bus de una instrucción se ejecutará en 125 ns o 1 dividido por 8 MHz.

Figura 91. ***Arquitectura***[121]



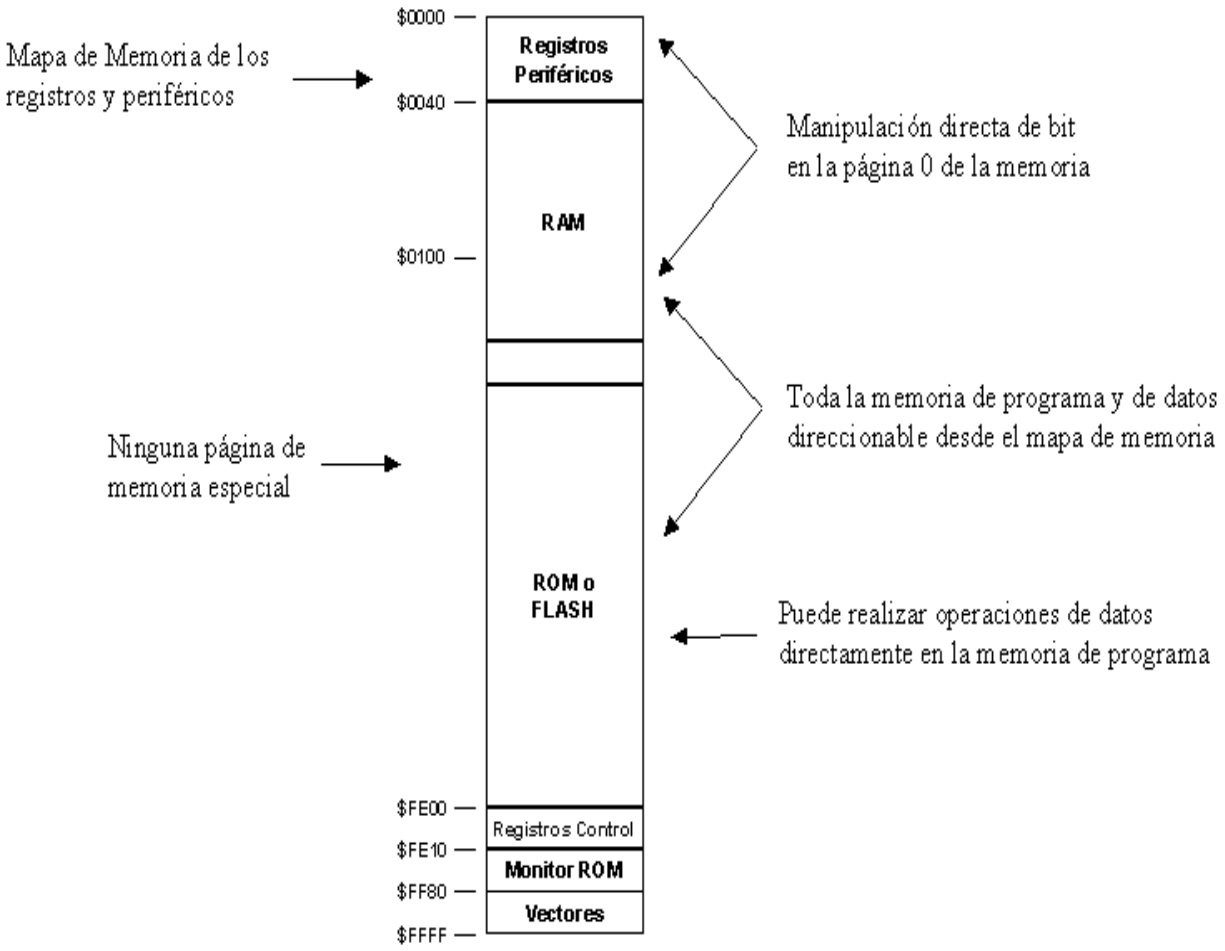
Organización de la memoria

La figura siguiente muestra el mapa de memoria del 68HC08. Este mapa de memoria es idéntico al empleado para el 68HC05, sólo que se ha implementado hasta 64 KBytes, sin tener en cuenta los diferentes tamaños de memoria de cada 68HC08 disponible En el 68HC08, el mapa de memoria empieza con los registros de los periféricos con 64 bytes. A continuación le sigue el espacio para la RAM. La ROM o FLASH ocupa la parte superior de la memoria que precede a $FE00, donde están los Vectores, el programa Monitor y los Registros de control.

[121] Vesga, 2007

En el medio del mapa de memoria, entre la RAM y la ROM/FLASH, hay una zona de memoria no utilizable por el usuario que realiza las verificaciones de direcciones ilegales.

Figura 92. ***Bloque de memoria***[122]

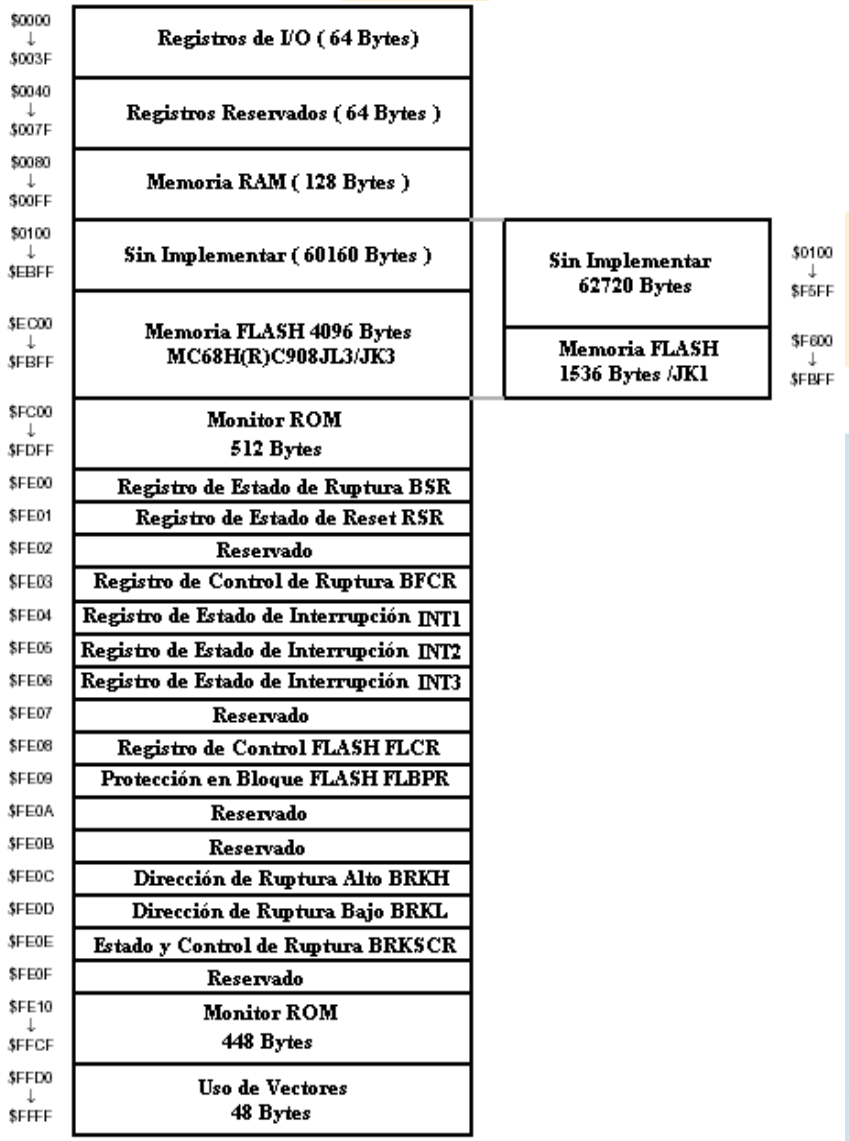


Todas las MCU de la familia 68HC08 usan este modelo, aunque algunas de ellas tienen una dirección de inicio y final para las áreas específicas del monitor ROM o los registros de control

Desde las direcciones $0000 hasta la $003F se encuentran todos los registros de control para entrada y salida de datos, configuración de los puertos A, B y D, Configuración y estado de los TIMERS, de los canales de conversión A/D, etc (VESGA, 2007).

[122] Vesga, 2007

Figura 93. ***Memoria en los Motorola Freescale, Fuente*** [123]



Registros de uso general

ACUMULADOR (A) : Es un registro de 8 bits de propósito general de lectura y escritura utilizado por la CPU para el almacenamiento temporal de los operandos y los resultados de las operaciones aritméticas y lógicas realizadas por la ALU.

[123] VESGA, 2007

REGISTRO ÍNDICE (H:X) : Es un registro de 16 bits que permite direccionar hasta 64 K de memoria en forma indexada, esta dividido en dos, el primero denominado H con el byte alto (8 bits mas altos) y el byte bajo (8 bits mas bajos) se denomina X, con lo cual se construye un registro de 16 bits, o 2 bytes H:X lo que permite apuntar a cualquiera de las 64 K de memoria.

REGISTRO ÍNDICE (X) : El registro índice es usado para los modos de direccionamiento indexados o bien puede ser usado como un acumulador auxiliar, está constituido por 8 bits, debe recordarse que su contenido determina la dirección efectiva del operando.

PUNTERO DE PILA (SP) : (Stack Pointer), Es un registro de 16 bits que contiene la dirección de la próxima posición de 8 bits dentro del Stack (pila).

CONTADOR DE PROGRAMA (PC) : Es un registro de 16 bits que contiene la dirección de la siguiente instrucción u operación a procesar, la CPU aumenta el valor del contador en forma secuencial, guardando la siguiente posición de memoria.

REGISTRO DE BANDERAS (CCR): Conocido en otros microcontroladores como el registro de estado, Es un registro de 8 bits que contiene el bit habilitador de interrupción y 5 banderas o flags de estado, tomando como referencia el material bibliográfico “Microcontroladores Motorola Freescale” (VESGA,2007) se ilustran el diagrama y funciones de los flags.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer								
Escribir	V	1	1	H	I	N	Z	C

V : Bandera de rebosamiento o bit indicador de sobre flujo

La CPU coloca esta bandera en 1 cuando al efectuar el complemento a dos ocurre un rebosamiento.

H : Bandera de Medio Carry o bit de acarreo intermedio

La CPU coloca este bit en 1 cuando ocurre un carry entre los bits 3 y 4 durante una suma con o sin carry, el medio carry es requerido cuando se utiliza codificación en BCD.

I : Mascara de interrupciones o bit habilitador de interrupciones

Cuando este bit se coloca en 1 lógico todas las interrupciones son deshabilitadas, y son habilitadas nuevamente cuando este bit es colocado en 0 lógico.

N : Bandera de valor negativo o bit de indicador negativo

Este bit es colocado en 1 lógico cuando el resultado de una operación aritmética es **Negativa**.

Z : Bandera de Cero

Este bit es colocado en 1 lógico cuando el resultado de una operación aritmética o lógica de cómo resultado **CERO.**

C : Bandera de Carry o bit de acarreo

Este bit es colocado en 1 lógico cuando el resultado de una operación aritmética produce Carry después del bit 7**.**

LECCIÓN 4: PUERTOS I/O Y PRINCIPALES MÓDULOS EN LOS MICROCONTROLADORES MOTOROLA FREESCALE.

Puertos de entrada salida

Los Puertos del microcontrolador son de gran importancia pues a través de ellos se controlan cargas, dispositivos o sirven para recibir información del entorno del microcontrolador. En el Microcontrolador MC68H(R)C908JL3E, 23 pines pueden

ser configurados de manera bidireccional (I/O), a través de tres puertos paralelos. Todos los pines pueden ser configurados como entrada o salida.

Se Debe tener en cuenta que en la gran mayoría de familias de Microcontroladores, los puertos de entrada/salida no solamente cumplen funciones de envío y recepción de señales digitales, sino que además comparten recursos internos con el Microcontrolador. (VESGA, 2007).

Como todo dispositivo basado en tecnología FET (Transistor de efecto de Campo) es recomendable terminar los pines no utilizados en las aplicaciones, es decir conectarlos a cero voltios (0 V) o a cinco voltios (5 V).

REGISTRO PORT A (PTA) Dirección $0000

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer	0							
Escribir		PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0

Es un registro de datos, compuesto por 7 bits de lectura y escritura, se pueden configurar como 7 entradas externas de interrupción para el manejo de teclados , todos implementan resistencias internas pull-ups con capacidad de conectarlas o desconectarlas mediante configuración del software. Los terminales entre PTA= y PTA5 poseen fuentes limitadas de corriente destinadas al manejo de diodos LED, es decir no es necesario las resistencias limitadoras de corriente.

REGISTRO DE CONFIGURACION DEL PORT A (DDRA) Dirección $0004

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer	0							
Escribir		DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0

Mediante este registro de 8 bits, de los cuales a solo 7 bits se puede acceder, se efectúa la configuración de los bits del registro PORT A, ya sea como entradas, con un uno (1) lógico o salidas con un cero (0) lógico. La acción de reset borra todos los bits de control, programando todo el puerto como entrada.

REGISTRO PTAPUE Dirección $000D

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer								
Escribir	PTA6EN	PTAPUE6	PTAPUE5	PTAPUE4	PTAPUE3	PTAPUE2	PTAPUE1	PTAPUE0

Este registro es el que permite habilitar o deshabilitar las resistencias de Pull-Up para cada uno de los pines del puerto A. Esta configuración es valida siempre que los pines estén configurados como entradas, la CPU desconecta las pull-up automáticamente cuando se configure el pin como salida.

REGISTRO PORT B (PTB) Dirección $0001

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer Escribir	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

El registro PORTB, asociado a 8 terminales bidireccionales, permite la manipulación de señales digitales entre sus terminales tal como ocurre con el registro PORTA, adicionalmente cada pin de este puerto se asocia a un canal del modulo de conversión Análogo/Digital incorporado en el dispositivo.

REGISTRO DE CONFIGURACION DEL PORT B (DDRB) Dirección $0005

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer Escribir	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0

Este registro es el responsable de la configuración entrada / salida del puerto B se siguen los mismos parámetros utilizados en el registro de configuración DDRA.

REGISTRO PORT D (PTD) Dirección $0003

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer Escribir	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Función 1	LED	LED	TCH1	TCH0	LED	LED		
Función 2	25mA	25mA			ADC8	ADC9	ADC10	ADC11

El registro PORTD esta compuesto por 8 bits de lectura escritura, poseen funciones especiales de entrada y salida digital, dos de las terminales asociadas comparten su función con la interfaz del módulo temporizador (TCH1 y TCH2) mientras que otras cuatro comparten entradas del modulo de conversión Análoga / Digital (A/D) y dos pines (PTD6 y PTD7) con driver se corriente (25 mA) y resistencias pull-up programables.

REGISTRO DE CONFIGURACION DEL PORT D (DDRD) Dirección $0007

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer / Escribir	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0

Este registro de 8 bits es el responsable de la configuración de los bits del registro PORT D, como entradas o salidas.

REGISTRO DE CONTROL DEL PORT D (PDCR) Dirección $000A

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Leer	0	0	0	0				
Escribir					SLOWD7	SLOWD6	PTDPU7	PTDPU6

Su función es muy similar al del registro PTAPUE, habilita o deshabilita las resistencias pull-up, junto con el manejo de los drivers de corriente para los pines PTD6 Y PTD7.

FUNCION OSCILADOR

La función del OSC2 se configura cuando se escoge la opción de oscilador RC.

1 : El OSC2 es configurado para utilizar el pin PTA6 como un pin de I/O, con las funciones de interrupción y configuración de resistencias de Pull-Up.

0 : El OSC2 es configurado como oscilador de tipo RC

Set de instrucciones

El 68HC05 tiene un total de 85 instrucciones que en la actualidad todavía forman un juego básico de instrucciones funcional y potente. El juego de instrucciones del 68HC08 se ha construido sobre la base de las instrucciones del 68HC05.

El 68HC08 añade 28 instrucciones al juego de instrucciones del 68HC05, remarcadas en la tabla siguiente. Muchas de estas instrucciones soportan el registro del índice extendido a 16 bits y el puntero de pila (‘stack pointer’) es totalmente relocalizadle

Listado de prefijos comúnmente utilizados en representaciones numéricas.

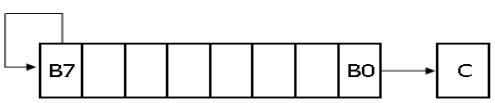
PREFIJO	Tipo de valor que representa
t	Decimal
$	Hexadecimal
@	Octal
%	Binario
Apóstrofe '	Carácter ASCII

Conjunto de Instrucciones

Tabla 24. ***Instrucciones Motorola Freescale***[124]

INSTRUCCIÓN	OPERACIÓN	No. CICLOS
ADC #OPR	SUMA CON CARRY	2
ADC OPR		3
ADC OPR,X	A = A+(M)+C	3
ADC ,X	M = Dato o Valor Almacenado	2
ADC OPR,SP	C = Carry	4
ADD #OPR	SUMA SIN CARRY	2
ADD OPR		3
ADD OPR,X	A = A+(M)	3
ADD ,X	M = Dato o Valor Almacenado	2
ADD OPR,SP		4
AIS #OPR	SP=SP+DATO	2
AIX #OPR	H:X = H:X + DATO	2
AND #OPR	FUNCION AND	2
AND OPR		3
AND OPR,X	A = A&(M)	3
AND ,X	M = Dato o Valor Almacenado	2
AND OPR,SP		4
ASL OPR	Desplazamiento Aritmético a la Izquierda	4
ASLA		1
ASLX		1
ASL OPR,X		4
ASL,X		3

[124] VESGA, 2007

ASL OPR,SP		5
ASR OPR ASRA ASRX ASR OPR,X ASR OPR,SP	Desplazamiento Aritmético a la Derecha 	4 1 1 4 5
BCC Etiqueta	Saltar a la etiqueta si el Bit de Carry es 0	3
BCLR N,OPR	Borrar el bit N del registro OPR	4
BCS Etiqueta	Saltar a la etiqueta si el Bit de Carry es 1	3
BEQ Etiqueta	Saltar a la etiqueta si es Igual (Bit Z=1)	3
BGE OPR	Saltar si es Igual o mayor que OPR	3
BGT OPR	Saltar si es mayor que OPR	3
BHCC Etiqueta	Saltar a la etiqueta si el bit de Carry Medio es 0 (H)	3
BHCS Etiqueta	Saltar a la etiqueta si el bit de Carry Medio es 1 (H)	3
BHI Etiqueta	Saltar a la etiqueta si es Mayor	3
BHS Etiqueta	Saltar si es Mayor o Igual	3
BIH Etiqueta	Saltar si el Pin IRQ está en Alto	3

BIL Etiqueta	Saltar si el Pin IRQ está en Bajo	3
BIT #OPR		2
BIT OPR	Probar bits	3
BIT OPR,X		3
BIT ,X	A & (M)	2
BIT OPR,SP	M = Dato o Valor Almacenado	4
BLE OPR	Saltar si es Igual o Menor que OPR	3
BLO ETIQ	Saltar a la etiqueta si es Menor	3

BLS ETIQ	Saltar a la etiqueta si es Menor o igual	3
BLT OPR	Saltar si es Menor que	3
BMC ETIQ	Saltar si la bandera de interrupción esta en 0	3
BMI ETIQ	Saltar si el resultado de una operación es Negativo	3
BMS ETIQ	Saltar si la bandera de interrupción esta en 1	3
BNE ETIQ	Saltar a la etiqueta si no es igual	3
BPL ETIQ	Saltar si el resultado de una operación es Positivo	3
BRA ETIQ	Saltar a la Etiqueta siempre	3
BRCLR N,OPR,ETIQ	Saltar a la etiqueta si el bit N del registro OPR esta en 0	5
BRN ETIQ	Nunca Saltar	3
BRSET N,OPR,ETIQ	Saltar a la etiqueta si el bit N del registro OPR esta en 1	5
BSET N,OPR	Poner en 1 el bit N del registro OPR	4
BSR ETIQ	Saltar a Subrutina	4
CBEQ OPR,ETIQ		5
CBEQA #OPR,ETIQ	Comparar el valor de A con el valor #OPR o el dato	4
CBEQX #OPR,ETIQ	almacenado en OPR y saltar si son iguales a la etiqueta	4
CBEQ OPR,X+,ETIQ		5
CBEQ OPR,SP,ETIQ		5
CLC	Borrar el Bit de Carry	1
CLI	Borrar el Bit de Interrupción o Bandera de Interrupción	2
CLR OPR		3
CLRA		1
CLRX	Borrar	1

CLRH		1
CLR OPR,X		3
CLR,X		2
CLR OPR,SP		4

CMP #OPR		2
CMP OPR	Comparar el valor de A con el valor #OPR o el dato	3
CMP OPR,X	almacenado en OPR	3
CMP ,X		2
CMP OPR,SP		4
COM OPR		4
COMA		1
COMX	Complemento a uno	1
COM OPR,X		4
COM ,X		3
COM OPR,SP		5
CPHX #OPR	Comparar el valor de H:X con el valor #OPR o el dato	3
CPHX OPR	almacenado en OPR	4
CPX #OPR		2
CPX OPR	Comparar el valor de X con el valor #OPR o el dato	3
CPX ,X	almacenado en OPR	3
CPX OPR,X		2
CPX OPR,SP		4
DAA	Ajustar a decimal el registro A	2
DBNZ OPR,ETIQ		5

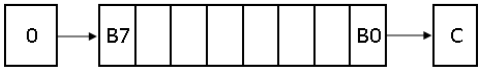
DBNZA ETIQ		3
DBNZX ETIQ	Decrementar y saltar si no es Cero	3
DBNZ OPR,X,ETIQ		5
DBNZ X,ETIQ		4
DBNZ OPR,SP,ETIQ		5
DEC OPR		4
DECA		1
DECX	Decrementar y saltar si no es Cero	1
DEC OPR,X		4
DEC ,X		3
DEC OPR,SP		5
DIV	Dividir A=(H:A)/X (A = Cociente , H = Residuo)	7
EOR #OPR		2
EOR OPR	OR EXCLUSIVA	3
EOR OPR,X		3
EOR ,⊕	A = A (M)	2
EOR OPR,SP	M = Dato o Valor Almacenado	4
INC OPR		4
INCA		1
INCX	Incrementar	1
INC OPR,X		4
INC ,X		3
INC OPR,SP		5

JMP OPR JMP OPR,X JMP ,X	Saltar a la dirección OPR	2 3 2
JSR OPR JSR OPR,X JSR ,X	Saltar a Subrutina	4 5 4

LDA #OPR
LDA OPR
LDA OPR,X
LDA ,X
LDA OPR,SP
	Cargar en A el valor #OPR o el dato almacenado en OPR	2
		3
		3
		2
		4

LDHX #OPR LDHX OPR	Cargar en H:X el valor #OPR o el dato almacenado en OPR	3 4

LDX #OPR
LDX OPR
LDX OPR,X
LDX ,X
LDX OPR,SP
	Cargar en X el valor #OPR o el dato almacenado en OPR	2
		3
		3
		2
		4

LSL OPR LSLA LSLX LSL OPR,X LSL ,X LSL OPR,SP	Desplazamiento Lógica a la Izquierda (Igual que ASL)	4 1 1 4 3 5
LSR OPR LSRA LSRX	Desplazamiento Lógica a la Derecha 	4 1 1

LSR OPR,X		4
LSR ,X		3
LSR OPR,SP		5
MOV OPR,OPR		5
MOV OPR,X+	Mover Fuente, Destino	4
MOV #OPR,OPR		4
MOV X+,OPR		4
MUL	Multiplicación sin Signo (X:A = X* A)	5
NEG OPR		4
NEGA		1
NEGX	Complemento a Dos	1
NEG OPR,X		4
NEG ,X		3
NEG OPR,SP		5
NOP	No Operación	1
NSA	Intercambiar Nibles de A (A=(A[3:0]:A[7:4]))	3

ORA #OPR		2
ORA OPR	FUNCION OR	3
ORA OPR,X		3
ORA ,X	A = A | (M)	2
ORA OPR,SP	M = Dato o Valor Almacenado	4
PSHA	Insertar A en el Stack	2
PSHH	Insertar H en el Stack	2

PSHX	Insertar X en el Stack	2
PULA	Sacar A del Stack	2
PULH	Sacar H del Stack	2
PULX	Sacar X del Stack	2
ROL OPR		4
ROLA	Rotar a la Izquierda a través del Carry	1
ROLX		1
ROL OPR,X		4
ROL ,X		3
ROL OPR,SP		5
ROR OPR		4
RORA	Rotar a la Derecha a través del Carry	1
RORX		1
ROR OPR,X		4
ROR ,X		3
ROR OPR,SP		5
RSP	Reset al Stack Pointer	1
RTI	Retornar de una Interrupción	7
RTS	Retornar de una Subrutina	4
SBC #OPR		2
SBC OPR	RESTA CON CARRY	3
SBC OPR,X		3
SBC ,X	A = A - (M) - C	2
SBC OPR,SP	M = Dato o Valor Almacenado	4
SEC	Colocar el bit de Carry en 1	1
SEI	Colocar el bit de Interrupción en 1 I = 1	2
STA OPR		3

STA OPR,X	Asignar A en el registro OPR	4
STA ,X	M = A	2
STA OPR,SP	M = Dato o Valor Almacenado en dirección OPR	4
STHX OPR	Asignar H:X en el registro OPR	4
STOP	Habilitar el pin IRQ, detener el Oscilador	1
STX OPR		3
STX OPR,X	Asignar X en el registro OPR	4
STX ,X		2
STX OPR,SP		4

SUB #OPR		2
SUB OPR	RESTAR	3
SUB OPR,X		3
SUB ,X	A = A - (M)	2
SUB OPR,SP	M = Dato o Valor Almacenado	4
SWI	Interrupción por Software	9
TAP	Transferir A al CCR , CCR = A	2
TAX	Transferir A a X , X = A	1
TPA	Transferir CCR a A ; A = CCR	1
TST OPR		3
TSTA		1
TSTX	Probar si la cantidad es negativa o Cero	1
TST OPR,X		3
TST ,X		2
TST OPR,SP		4

TSX	Transferir SP a H:X , H:X = SP + 1	2
TXA	Transferir X a A , A = X	1
TXS	Transferir H:X a SP , SP = H:X - 1	2

LECCIÓN 5: HERRAMIENTAS DE DESARROLLO Y EJERCICIOS BÁSICOS.

Igual como en los microcontroladores PIC, en la familia de los Motorola Freescale tienen un "Entorno de Desarrollo Integrado", llamado WINIDE. Motorola y otras empresas independientes han desarrollado varias herramientas como emuladores, analizadores lógicos, programadores, tarjetas de evaluación, tarjetas de desarrollo, simuladoras, compiladoras en lenguaje C, ensambladores y depuradores.

Ejercicios básicos con microprocesadores

El desarrollo e implementación de los siguientes ejercicios básicos, permitirán una apropiación más efectiva de los conceptos y habilidades en la programación y ejecución de aplicaciones basadas en el microcontrolador Motorola Freescale.

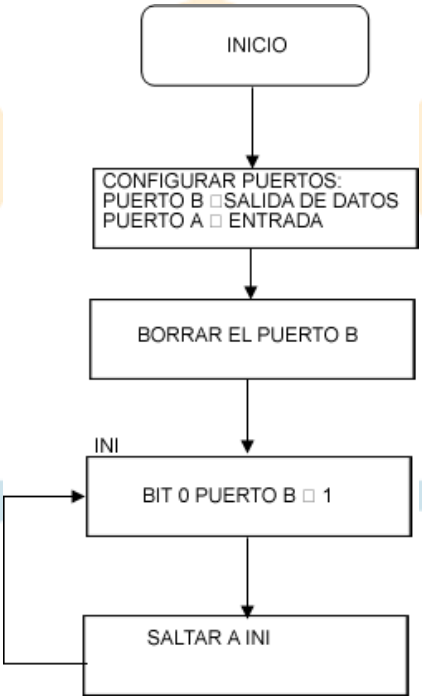
- **Encender un LED**: El objetivo es encender un LED conectado a una línea del puerto D en su pin PTD7 del microcontrolador 68HC908JK3.

Entradas y salidas: en este caso no hay entradas solo una salida donde se conectara un diodo LED, elegimos el puerto "D" en su pin PTD7 como salida de datos.

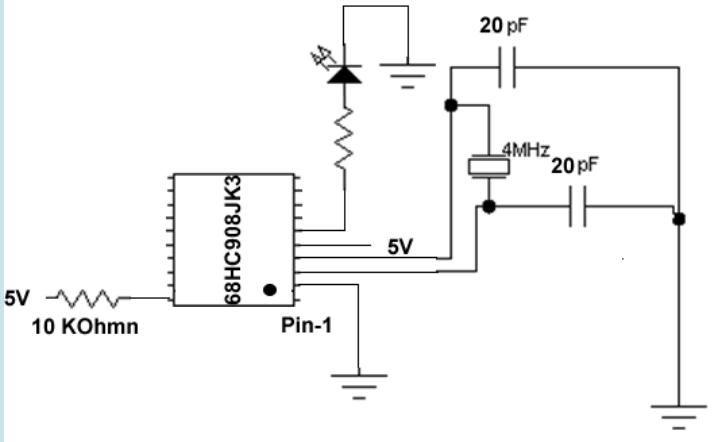
Pseudocódigo:

```
INICIA:
        CONFIGURAR PUERTOS:
                PUERTO D →SALIDA DE DATOS
        INI:    BORRAR EL PUERTO D
                ACTIVAR EL BIT 7→ 1
                SALTAR A RUTINA INI
TERMINA
```

Diagrama de Flujo



Montaje:



Código ensamblador

```
$INCLUDE 'JL3REGSG.INC'              ;ARCHIVO INCLUIDO DONDE SE DEFINEN LOS  REGISTROS
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;ENCENDIDO DE UN LED
;PINES DE SALIDA LEDS ==> D7
;PINES DE ENTRADA ==> "NINGUNO"
;-------------------------------------------------------------------------------
;REGISTROS DE PROPOSITO ESPECIAL MAS UTILIZADOS
```

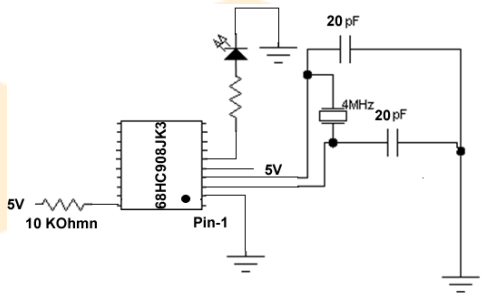
```
RAMSTART        EQU     $0080   ;DIRECCIONES RAM Y ROM EN LA MEMORIA LINEAL
ROMSTART        EQU     $EC00
VECTORES        EQU     $FFDE
;ROMFIN         EQU     $FB00
RESET_VEC       EQU     $FFFE
;-------------------------------------------------------------------------------
ORG     RAMSTART                ;DEFINICION DE LAS VARIABLES EN MEMORIA RAM
CONTADOR        RMB     1       ;RESERVACION DE MEMORIA PARA LAS VARIABLES
REGIS1          RMB     1
REGIS2          RMB     1
;-------------------------------------------------------------------------------
;INICIO DEL PROGRAMA
;-------------------------------------------------------------------------------
ORG     ROMSTART                ;INICIO DE LA MEMORIA DE PROGRAMA
;-------------------------------------------------------------------------------
;CONFIGURACION DE PUERTOS
;-------------------------------------------------------------------------------
INICIO  RSP                             ;INICIALIZAR EL PUERTO A, LA PILA (SP)
        BSET    COPD,   CONFIG1         ;DESHABILITA EL COP (WATHDOG)
        CLRA                            ;INICIALIZAR ACUMULADOR
        CLRX                            ;INICIALIZAR REGISTRO X
        MOV     #$80,   DDRD            ;CONFIGURAR EL PIN7 DEL PUERTO D COMO SALIDA
INICIO  CLR     PORTD                   ;INICIAR PORTD
        BSET    7,      PORTD           ;ACTIVO EL BIT 7 DEL PUERTO D, DONDE SE ENCUENTRA EL LED
        JSR     INI                     ;CERRAR EL CICLO RETORNANDO A “INI”
        ORG     RESET_VEC
        DW      INICIO                  ;AL DARSE RESET SALTAR A INICIO
;-------------------------------------------------------------------------------
        END
;-------------------------------------------------------------------------------
```

Contrario al lo que sucede en los microcontroladores PIC, la arquitectura de los Motorola Freescale, poseen un mapa lineal de memoria, por lo que no se necesita direccionar bancos de memoria. La directiva “include” permite incluir archivos adicionales que definan subrutinas o definiciones estandarizadas que son invocadas dentro del programa.

- **Intermitencia de un LED**: El objetivo es encender de forma intermitente un LED conectado a una línea del puerto D en su pin PTD del microcontrolador 68HC908JK3.

Entradas y salidas: en este caso no hay entradas solo una salida donde se conectara un diodo LED, elegimos el puerto “D” en su pin PTD como salida de datos, mas un retardo. El programa se basa en el anterior e incorpora una rutina de retardo.

Montaje:



Código ensamblador

```
$INCLUDE 'JL3REGSG.INC'          ;ARCHIVO INCLUIDO DONDE SE DEFINEN LOS  REGISTROS Y BITS
;INGENIERO ELECTRONICO HECTOR URIEL VILLAMIL GONZALEZ
;ENCENDIDO DE UN LED
;PINES DE SALIDA LEDS ==> D7
;PINES DE ENTRADA ==> "NINGUNO"
;-------------------------------------------------------------------------------
;REGISTROS DE PROPOSITO ESPECIAL MAS UTILIZADOS
RAMSTART        EQU     $0080    ;DIRECCIONES RAM Y ROM EN LA MEMORIA LINEAL
ROMSTART        EQU     $EC00
VECTORES        EQU     $FFDE
;ROMFIN          EQU     $FB00
RESET_VEC       EQU     $FFFE
;-------------------------------------------------------------------------------
ORG     RAMSTART                 ;DEFINICION DE LAS VARIABLES EN MEMORIA RAM
CONTADOR        RMB     1        ;RESERVACION DE MEMORIA PARA LAS VARIABLES
REGIS1          RMB     1
REGIS2          RMB     1
;-------------------------------------------------------------------------------
;INICIO DEL PROGRAMA
;-------------------------------------------------------------------------------
ORG     ROMSTART                 ;INICIO DE LA MEMORIA DE PROGRAMA
;-------------------------------------------------------------------------------
;CONFIGURACION DE PUERTOS
;-------------------------------------------------------------------------------
INICIO  RSP                              ;INICIALIZAR EL PUERTO A, LA PILA (SP)
        BSET    COPD,   CONFIG1          ;DESHABILITA EL COP (WATHDOG)
        CLRA                             ;INICIALIZAR ACUMULADOR
        CLRX                             ;INICIALIZAR REGISTRO X
        MOV     #$80,   DDRD             ;CONFIGURAR EL PIN7 DEL PUERTO D COMO SALIDA
INICIO  CLR     PORTD                    ;INICIAR PORTD
        BSET    7,      PORTD            ;ACTIVO EL BIT 7 DEL PUERTO D, DONDE SE ENCUENTRA EL LED
        JSR     RET10S                   ;SALTO A LA SUBRUTINA DE RETARDO
        BCLR    7,      PORTD            ;UNA VEZ TERMINADO EL RETARDO, SE PROCEDE A APAGAR EL
LED
        JSR     RET10S                   ;LLAMADO A RETARDO
        JSR     INI                      ;CERRAR EL CICLO RETORNANDO A "INI"
;-------------------------------------------------------------------------------
RET01   LDA     #128T                    ;GENERACION DE RETARDO
ET2     CLRX                             ;CARGA DEL ACUMULADOR CON EL NUMERO
ET3     DECX                             ;COMIENZA LA ESTRUCTURA REPETITIVA
        BNE     ET3                      ;REGISTRO DECREMENTANDO HASTA LLEGAR A CERO
        DECA
        BNE     ET2                      ;SE RETORNA UN "POSMEN" DESPUES DEL LLAMADO
        RTS
        ORG     RESET_VEC
        DW      INICIO                   ;AL DARSE RESET SALTAR A INICIO
;-------------------------------------------------------------------------------
        END
;-------------------------------------------------------------------------------
```

ACTIVIDADES DE AUTOEVALUACIÓN DE LA UNIDAD

1. Investigue en textos e internet las principales familias de microcontroladores y genere puntos de debate con sus compañeros acerca de la variedad de productos y los últimos modelos.
2. Determine cuales son las unidades fundamentales comunes a un microcontrolador y microprocesador y genere puntos de comparación para justificarlos al interior de su grupo de trabajo.
3. Con respecto a los diversos componentes utilizados como dispositivos de entrada, investigue la información particular del dispositivo que puede estar en hojas de características (datasheet), voltajes de alimentación, señal de entrada, señal de salida, rango de voltajes o corrientes, comparta con su grupo de trabajo la información referida a los siguientes componentes:
    a. Interruptores.
    b. Pulsadores.
    c. Teclados
    d. Sensores de temperatura, humedad, ph, ópticos y de movimiento.
4. Con respecto a los diversos componentes utilizados como dispositivos de salida y/o actuadores, investigue la información particular del dispositivo que puede estar en hojas de características (datasheet), voltajes de alimentación, señal de entrada, señal de salida, rango de voltajes o corrientes, comparta con su grupo de trabajo la información referida a los siguientes componentes:
    a. LEDs.
    b. Display siete segmentos.
    c. Display de Cristal liquido, LCD.
    d. Relé o relay
    e. Bocinas o parlantes.
    f. Motores AC, DC, paso a paso y Brushless
5. Investigue la simbología electrónica, funcionamiento nomenclatura y montaje de los componentes mas usuales en los montajes de sistemas basados en microcontroladores, mas precisamente, resistencias, capacitores (condensadores), bobinas, transformadores, diodos de unión, diodos zener, puentes rectificadores, LEDs, Resonador de Cristal,

transistores BJT (NPN,PNP), transistores de efecto de campo FET, tiristores SCR, DIAC, TRIAC. Comparta con sus compañeros de grupo comentarios, inquietudes y documentación al respecto. Recuerde que los tutores en el área de electrónica pueden ayudarlos en este aspecto.

6. Realice una investigación acerca de las técnicas de construcción de circuitos impresos y con su grupo de trabajo realizar la implementación en protoboard de circuitos simples como:
    a. Circuito de resistencias en paralelo, serie y mixto.
    b. Montaje de circuitos simples con el temporizador CI555, en conjunto con resistencias, capacitores, pulsadores, switch, LEDs y fuente de alimentación.
7. Tomando un PIC16F84 o un Motorola Freescale HC08, determine con ayuda de este modulo, la hoja de especificaciones, textos especializados e internet, la arquitectura interna del microcontrolador.
8. Instale los programas de simulación y desarrollo para el PIC o Freescale e ingrese a la página oficial de “proteus” para acceder a una versión gratuita.
    a. PIC16F84: MPLAB, PICDevelopment Studio, ICPROG.
    b. HC08: WINIDE,
    c. Proteus: http://www.ieeproteus.com/descarga.html
9. Aprenda el set o juego de instrucciones de los microcontroladores PIC y Motorola Freescale, practique la ejecución individual en papel (prueba de escritorio) de cada instrucción para comprender su funcionamiento, genere puntos de debate y discusión al respecto con sus compañeros y tutor.
10. Digite los programas en ensamblador, compílelos e implemente los montajes propuestos y ejecútelos, observe el comportamiento, comparta la experiencia con el grupo y tutor.

BIBLIOGRAFIA

Vesga, Ferreira Juan Carlos. (2007). Microcontroladores Motorola – Freescale: Programación, familias y sus distintas aplicaciones en la industria.

CEKIT. (2002). Curso Práctico de Microcontroladores: Teoría, Programación, Diseño, Prácticas Proyectos completos. Editorial Cekit. Pereira-Colombia.

González, Vásquez José Adolfo. (1992). Introducción a los microcontroladores: hardware, software y aplicaciones. Editorial McGraw-Hill.

Téllez, Acuña Freddy Reynaldo. (2007). Modulo de Microprocesadores y Microcontroladores. UNAD.

Valdivia, Miranda Carlos. (n. d). Arquitectura de equipos y sistemas informáticos. Editorial Paraninfo.

Angulo, Usategui José María. (n. d). Microcontroladores PIC. Diseño practico de aplicaciones.

UNIDAD 3:

Nombre de la Unidad	**MICROPROCESADORES Y MICROCONTROLADORES, PROGRAMACION Y DESARROLLO DE PROYECTOS**
Introducción	Esta Unidad ratifica su importancia pues está enfocada en la implementación practica de sistemas basados en microcontroladores en este caso especial de microcontroladores PIC, mostrando varias estrategias y modelos de programación de gran utilidad en muchas aplicaciones, con estos conceptos y ejemplos se refuerza el aprendizaje y se desarrollan habilidades en la programación de este dispositivo, con lo que permite tener capacidad de programar cualquier otro en la misma familia o diferente familia.
Justificación	Tras dios unidades donde se exploran las bases fundamentales de los microprocesadores, los microcontroladores y su programación, esta unidad tiene por objetivo lograr desarrollar habilidades y competencias en el ejercicio de programación para llegar al diseño, desarrollo e implementación basada en soluciones con dispositivos de control.
Intencionalidades Formativas	Lograr profundidad en el aprendizaje y desarrollo de competencias y habilidades que motiven el diseño de soluciones que beneficien su desempeño laboral y profesional.
Denominación de capítulos	Capitulo 7: Programación en los microprocesadores y microcontroladores. Capitulo 8: Primeros pasos en la programación de PICs Capitulo 9: Proyectos de Aplicación

CAPITULO 7: PROGRAMACION EN LOS MICROPROCESADORES Y MICROCONTROLADORES

LECCIÓN 1: CONCEPTOS BÁSICOS DE PROGRAMACIÓN EN MICROCONTROLADORES.

La solución a un problema de control electrónico, basado en microcontroladores, incluye dos etapas fundamentales:

- Escribir el programa en lenguaje ensamblador
- Generar el archivo ejecutable que debe cargarse en la memoria del microcontrolador

El ensamblador para microcontroladores está conformado por varios módulos independientes cada uno de los cuales cumple una función específica.

Ensamblador básico:

Genera a partir del código fuente, un archivo binario relocalizable, este archivo se puede almacenar en cualquier segmento disponible de memoria del microcontrolador.

Enlazador (linker): a partir del archivo binario relocalizable, se crea un archivo binario ejecutable, el cual es grabado en la memoria del micro.

Control de librerías: permite la creación de archivos binarios que pueden ser unidos (enlazados) con otros bloques de código binario, lo que facilita la reutilización de partes de programas generados en otros proyectos.

Desarrollo de proyectos

El ejercicio de desarrollar un control o proyecto basado en microcontroladores, esta sujeto a un ciclo que es representado por:

- Requisitos del problema.
- Análisis de la solución.
- Diseño de recursos.
- Codificación.
- Pruebas.
- Mantenimiento.

Este ciclo se resume en una serie de actividades familiares a un programador de proyectos con microcontroladores, este ciclo está conformado por:

- Programador, en esta etapa se establecen las entradas/salidas, se diseña el algoritmo y su diagrama de flujo.
- Editor de texto, utilizando los Entornos de Desarrollo Integrado – IDE o cualquier editor básicos, se procede a convertir el algoritmo o diagrama de flujo en instrucciones en lenguaje ensamblador compatible con el dispositivo escogido.
- Código fuente, utilizando los IDE se procede a generar el código fuente mediante la compilación del archivo editado.
- Programa ensamblador, este genera los archivos .OBJ, .LST y .HEX a partir del código fuente.
- Archivo ejecutable, como resultado del proceso de compilación se obtiene un archivo ejecutable con extensión .HEX el cual es tomado para grabar en la memoria del microcontrolador.

Método de trabajo para el desarrollo de aplicaciones

Los pasos que se mencionan a continuación pueden ser adecuados, la secuencia responde a un ejercicio constante de verificación y correcciones tanto en el código fuente y ejecutable, como en el diseño base del algoritmo. El método es simple y se recomienda seguir los pasos para lograr los objetivos:

- Requisitos del problema: se establecen las entradas, salidas y componentes periféricos como LEDs, Display 7 segmentos, LCD, teclados, relés, bocinas, motores, etc. Con lo que se obtiene un posible microcontrolador para la aplicación.

- Análisis de la solución: se reevalúa el microcontrolador y dispositivos periféricos pudiendo elegir un chip con capacidades acordes que permita sencillez en la implementación, se tiene en cuenta la precisión, longitud y velocidad de proceso del microcontrolador a elegir. Se pretende tener al final de esta etapa el pseudocódigo, algoritmos y diagramas de flujo.
- Diseño de los recursos: Se establece el listado de componentes, se procede a buscar en hojas de especificaciones las características propias de cada dispositivo y la forma de adecuar sus señales de entrada o salida a las señales digitales o analógicas que el microcontrolador recibe o entrega.
- Código fuente: con lo logrado hasta ahora se procede a convertir el algoritmo en código ensamblador a fin de obtener el código fuente del programa.
- Ensamblador: con el código fuente ya terminado se procede a ensamblar en programa fuente convirtiéndolo en código binario (HEX) que se grabara en la memoria del micro mediante el programador (hardware y software).
- Código binario: como resultado del ensamblado del código fuente se obtiene el código binario en un archivo con extensión .HEX, el cual contiene las instrucciones y secuencias de decisión y control, es el fruto del trabajo de las primeras dos etapas.
- Grabación en memoria: consiste en tomar el código binario, archivo .HEX, y grabarlo en la memoria del microcontrolador utilizando programas especiales vinculados a través de puerto USB, paralelo o serial a un hardware diseñado para alojar el microcontrolador y recibir los pulsos que permiten la grabación del programa en la memoria ROM en su interior.
- Prueba del sistema: en este punto se prueba el programa en el montaje implementado en protoboard y se establece si deben haber correcciones o se debe mejorar, lo que puede implicar saltar al paso “análisis de la solución”, para volver a reevaluar la solución, generar el algoritmo, etc.
- Producto final: esta es la ultima etapa en el proceso de trabajo para el desarrollo de aplicaciones con microcontroladores, en este punto se tiene establecido y completamente probado el programa y el circuito, es decir se tiene el software y hardware de control, el paso final es pasar el circuito montado a un circuito impreso definitivo, para su empaque y distribución.

LECCIÓN 2: ENSAMBLADOR EN LOS MICROCONTROLADORES.

Además del código binario, el ensamblador genera un conjunto de archivos adicionales, gracias a los cuales es posible controlar la evolución del proyecto. La extensión que acompaña al nombre del archivo indica el tipo de información que contiene.

Tabla 25. ***Archivos generados por ensamblador***

Tipo de archivo	Extensión	Ejemplo
Código fuente del programa	ASM	programa.asm
Código binario ejecutable	HEX	programa.hex
Listado del programa	LST	programa.lst
Lista de errores	ERR	programa.err
Código objeto ejecutable	OBJ	programa.obj
Archivo de librería	LIB	programa.lib

Los archivos con extensión .ASM contienen el código fuente del programa, este archivo puede ser digitado dentro de los programas IDE suministrados por los fabricantes o puede ser editado en un editor de texto simple como el "WordPad", es de considerar no colocar un nombre demasiado largo, la extensión se coloca generalmente de forma manual precedida de un punto (.asm o .ASM). Los nombres pueden ser una combinación de caracteres validos como letras, números y raya al piso, tenga presente que algunos sistemas no permiten extensión de nombre de mas de 8 caracteres. Los archivos de tipo HEX y OBJ contienen el código binario ejecutable y es derivado de archivos de tipo ASM. El proceso de ensamblado puede generar también archivos con extensiones como:

- LST con el contenido del programa fuente formateado, es decir, permite determinar la localización o posición de memoria donde se escribe la instrucción, la tabla de símbolos para la instrucción, la numeración de pagina, fecha y hora de ensamblado, una características que hace que el archivo LST sea particularmente importante es mostrar los errores o advertencias derivadas del código fuente.
- ERR, es un archivo donde se listan los errores y su localización en el programa, esto facilita el proceso de depuración y corrección del programa.

Términos utilizados en programación de microcontroladores

En la programación de microcontroladores como en todo sistema de aplicación informática, cuenta con términos o nombres particulares utilizados para referirse a partes del programa.

- **Registro**, es un término genérico el cual hace referencia a un localidad de la memoria que permite almacenar información temporal, cada registro esta formado por celdas las cuales tiene la capacidad de almacenar un bit. Generalmente se nombran los registros con la etiqueta "EQU" para identificarlos plenamente en el mapeo de memoria o en el programa.
- **Literal**, es un valor contante, se podría pensar como en "letras" pero realmente es un valor expresado en un sistema numérico particular y bien definido, generalmente binario, octal, hexadecimal y decimal. Es posible asignar un valor numérico a un conjunto de letras mediante el uso de la etiqueta "EQU" o similar. Ejemplo VAL EQU 12H, asigna el valor "12" expresado en hexadecimal al conjunto de letras "VAL".
- **Etiqueta**, es un nombre con el cual se identifica una posición de memoria del microcontrolador, sirve para marcar puntos o subprogramas dentro del programa principal, las etiquetas se escriben en la primera columna de línea al borde izquierdo su longitud no debe sobrepasar los 31 caracteres, debe estar compuesta por caracteres como letras, números y raya al piso, se debe corroborar restricciones de tamaño, símbolos como la "ñ o Ñ" y comenzar con caracteres numéricos.
- **Instrucción**, estas hacen referencia a operaciones básicas que pueden ejecutar un microcontrolador, se debe tener a la mano el set o lista de instrucciones validas junto con su sintaxis, operandos, operación, flags que afecta y ciclos maquina empleados.
- **Operando**, son elementos utilizados por una instrucción, algunas instrucciones no utilizan operandos, otras más complejas utilizan uno y finalmente las mas complejas utilizan un máximo de dos. El primer operando, se denomina "operando fuente", el segundo recibe el nombre de "operando destino", la información entonces fluye desde el operando fuente al destino.
- **Comentario**, los comentarios son bloques de texto que se digitan después de ";" para que el programador documente el programa, los comentario son ignorados por el ensamblador.
- **Encabezado**, es el primer componente del programa donde se definen características o funcionalidades especiales que necesita el programa y/o el ensamblador, estas directivas se establecen al comienzo del programa y dependen del tipo de IDE y familia de microcontroladores que se este utilizando. Ejemplo "List p=16C84", "$include 'jl3regsg.inc".
- **Definición de origen**, sirve para indicar explícitamente el sitio de la memoria donde se comienza la escritura de la pila, memoria de programa, interrupciones, y subprogramas (subrutinas).
- **Instrucciones de programa**, en la sección de instrucciones de programa se digitan en columna las instrucciones separando cada columna por un espacio contante generado por la tecla de tabulación "TAB", donde cada columna debe tener una estructura uniforme típicamente como sigue:
    1. Etiqueta, nombre del programa, puto de llegada de salto, subprograma o subrutina.
    2. Instrucción, es el "OPCODE" de la operación simple a realizar.

3. Operando fuente. Conformado por un registro, literal o constante.
4. Operando destino. Conformado por un registro, literal o constante, o destino de llegada del flujo de información.
5. Comentario, identificado por frases simples digitadas después de ";"

- **Fin del programa**, contiene una instrucción simple que le dice al ensamblador que en este punto termina el programa, es una palabra reservada por cada Entorno de Desarrollo Integrado para la familia de microcontroladores, por ejemplo "END" para el MPASM significa que es el final del programa.

LECCIÓN 3: MODOS DE DIRECCIONAMIENTO.

Existen varios modos de direccionamiento comunes tanto para microprocesadores como para microcontroladores, observemos como se presentan los modos de direccionamiento en los microcontroladores.

Direccionamiento Implícito

Estas instrucciones se caracterizan por no requerir el uso de operandos, debido a que no necesitan acceder a la memoria de datos, por ejemplo la instrucción "NOP", esta instrucción no produce ningún efecto visible pero es útil al brindar un retardo de una cantidad de ciclos maquina para ajustar retardos por software.

Direccionamiento inmediato

Se presenta cuando el dato no proviene de la memoria. El dato está incluido dentro de la misma instrucción, por ejemplo "MOVLW 5AH", esta instrucción almacena el literal 5AH en el registro de trabajo "W", este tipo de instrucciones utilizan datos constante o ya establecidos.

Direccionamiento directo

Es utilizado cuanto el dato se transfiere hacia, o desde, una posición de memoria, por ejemplo “MOVWF 0x0C”, la anterior instrucción accede a la localidad de memoria 0x0C y copia el dato en el registro de trabajo “W”.

Direccionamiento relativo

Las instrucciones de salto permiten alterar la ejecución secuencial de un programa, este tipo de instrucciones mediante el uso de un calculo simple suman un valor al contenido del registro “contador de programa” el cual contiene la dirección de salto. Este direccionamiento es aplicado en las “Tablas”.

Direccionamiento indexado

La dirección de destino se calcula utilizando como base un registro especial, por ejemplo “LDA $300, X” carga el registro A con el dato almacenado de la dirección 300 de memoria mas el contenido del registro “X”.

Direccionamiento extendido

Permite acceder a todo el espacio de memoria del microcontrolador, la dirección se almacena en dos o tres bytes del operando, de modo que sin importar el tamaño de la dirección destino se puede acceder directamente a ella. Por ejemplo, “LDA $3A4F”, carga el registro A utilizando el dato almacenado en la dirección absoluta “3A4F”.

Direccionamiento Indirecto

La dirección de una localidad de memoria se calcula mediante una doble referencia, en primer lugar se obtiene el dato contenido en una posición de memoria y con base en este dato, se obtiene la dirección efectiva de la posición de memoria deseada. Por ejemplo “LDA ($300)”, esta instrucción carga el registro A utilizando como base el dato contenido en la dirección “300”.

LECCIÓN 4: PROGRAMACIÓN EN MICROCONTROLADORES.

Para determinar el camino mas adecuado para desarrollar un proyecto con requiere del estudio del microcontroladores elegido tomando parámetros generales que se encuentran en hojas de especificadores o documentación especializada. A continuación se mencionan los parámetros generales y básicos a tener en cuenta en la fase de “análisis del problema”.

Elementos

Son los componentes básicos y especiales que tiene el microcontrolador en particular, que serán utilizados en el desarrollo e implementación, entre ellos están la arquitectura, los registros especiales, los registros de propósito general, el registro de estado, la memoria de datos, la memoria de programa, la pila y los puertos.

Arquitectura

Es necesario y conveniente estudiar la arquitectura y disposición lógica del dispositivo, algunos se basan en la arquitectura Harvard y otros en la Von Neumann por lo que el formato y tamaño del set de instrucciones puede variar, además de sus modos de direccionamiento, tamaño del bus de datos y de programa, velocidad de proceso y ciclo maquina.

Registros de trabajo o acumulador

Son los registros que interactúan directamente con la CPU, su posición respecto a esta última determina la forma de operación de las instrucciones, la cantidad de instrucciones y la complejidad del programa.

Registro de estado

Es el registro que contiene los bits indicadores de estado o flags que muestran el estado del dispositivo o el resultado de la ultima operación, además son utilizados también para el mapeo de memoria.

Memoria de programa

Los microcontroladores cuentan con un segmento de memoria ROM donde se aloja el programa en código binario, este segmento de memoria tiene subdivisiones explicitas referidas al vector de reset, el cual determina el punto donde comienza el programa o el valor que debe contener el contador de programa (PC) para iniciar después del "reset"; el vector de interrupción, determina el punto donde el contador de programa atenderá la solicitud de interrupción.

Memoria de datos

Es una memoria compuesta por un conjunto de registros, completamente accesibles al programador con lo que se optimiza el programa, en estos registros se almacenan las "variables" del programa que están continuamente alterándose por la secuencia de las instrucciones, normalmente se le asignan nombres al comienzo del programa en una sección dedicada a la "definición de constantes y/o variables" utilizando por ejemplo la etiqueta y palabra reservada "EQU". Dentro de la memoria de datos también se definen los registros de propósito especial los cuales alojan registros correspondientes a la matriz de registros de la CPU del microcontrolador, puertos, interrupciones y funciones especiales de comunicación, comparación o conversión.

Pila o Stack

Es un segmento de memoria con varios niveles dedicado al almacenamiento de la dirección de retorno de los llamados que se realizan a subrutinas dentro del programa principal. No se debe superar los niveles o profundidad de la pila, esto causa un desbordamiento y falla del programa. Lo anterior quiere decir que no deben anidarse llamados más allá del numero de niveles de la pila.

- **Subprograma**, El subprograma es una división del programa principal, esta ejecuta instrucciones como respuesta a instrucciones de llamado o interrupción, debe devolverse después de terminar su secuencia de instrucciones mediante instrucciones de retorno**.**

Puertos

Los puertos son elementos con los que el microcontrolador establece comunicación con el exterior, la cantidad de puertos y el tamaño de los mismos definen los componentes conectados a ellos que enviaran información al microcontrolador o la recibirán. Es de notar que en la gran variedad del microcontroladores tiene los puertos de doble vía, sirven como entradas o salidas, también existen algunos que tienen varias funciones es decir, que multiplexan sus entrada/salidas para lo cual existen registros de propósito especial que deben ser configurados.

LECCIÓN 5: EJERCICIOS DE PROGRAMACIÓN EN MPLAB.

En los proyectos que se describen en esta unidad se hace uso del software de uso libre como MPLAB, El motivo es que hemos de entender y aprender a utilizarlo para ensamblar nuestros proyectos, hacerlo fácil y esto es lo que vamos a describir aquí. Las instrucciones presentadas son generales y pueden variar en la localización de las opciones es recomendable hacer una exploración completa antes de comenzar los ejercicios. La última versión de MPLAB IDE se puede descargar en forma gratuita desde la página de Microchip.

El Organizador de Proyectos (Proyect Manager).

El organizador de proyectos (Proyect Manager) es parte fundamental de MPLAB. Sin crear un proyecto Usted no puede realizar depuración simbólica. Con el Organizador de Proyectos (Proyect manager) puede utilizar las siguientes operaciones:

- Crear un proyecto.
- Agregar un archivo de programa fuente de proyecto.
- Ensamblar o compilar programas fuente.
- Editar programas fuente.
- Reconstruir todos los archivos fuente, o compilar un solo archivo.
- Depurar su programa fuente.

Software ensamblador:

El software ensamblador que presenta Microchip viene en dos presentaciones, una, para entorno DOS llamado MPASM.EXE y la otra, para entorno Windows llamado MPASMWIN.EXE

Las dos presentaciones soportan a TODOS los microcontroladores de la familia PIC de Microchip. El conjunto de instrucciones de los microcontroladores PIC es en esencia la base del lenguaje ensamblador soportado por este software.

Directivas de uso frecuente:

Son instrucciones para el compilador.
#DEFINE
ej. #define <nombre> [<valor a remplazar>]
explicación: declara una cadena de texto como substituto de otra
END
ej. end
explicación: indica fin de programa
EQU
ej. status equ 05
explicación: define una constante de ensamble
INCLUDE
ej. include <PIC16F84.h>
explicación: incluye en el programa un archivo con código fuente
ORG
ej. org 0x100
explicación: ensambla a partir de la dirección especificada

Ejemplo en MPLAB

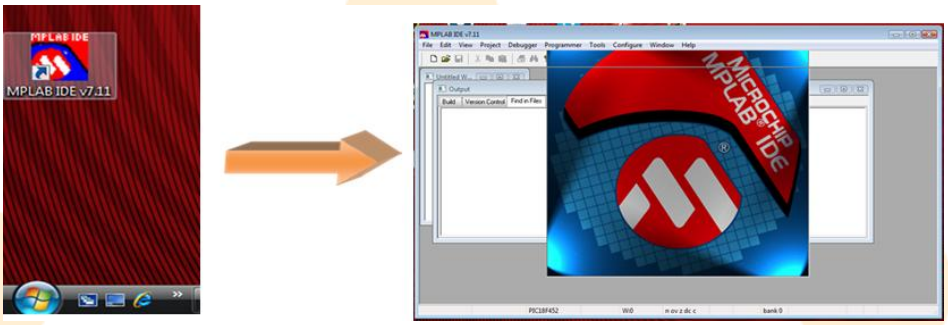
Para información más completa referirse a la guía rápida del MPASM. Una vez instalado adecuadamente el MPLAB, para realizar la simulación de un programa deben seguirse los siguientes pasos:

- Edite en un archivo de texto el siguiente programa:

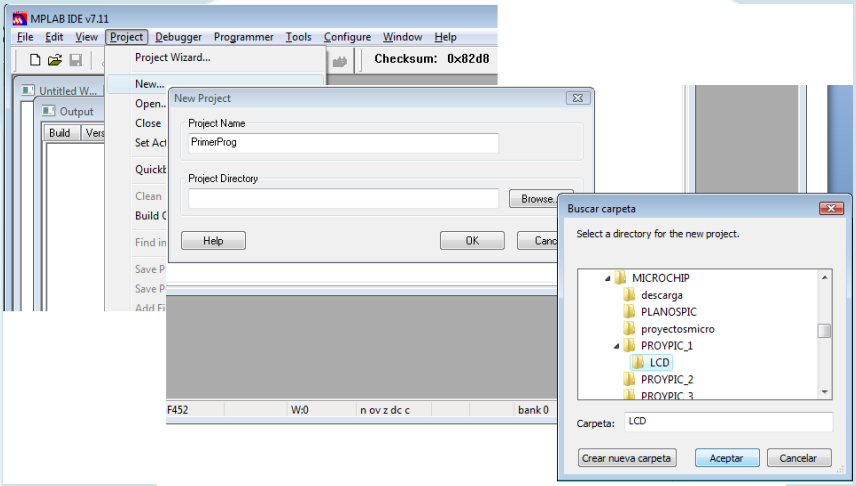
```
status  equ   0x03     ;hace equivalencia entre el símbolo status indicándolo como 3 en hexadecimal
Cont  equ   0x20
F     equ   1
      org 0           ;indica posición de memoria desde donde se ensambla
Inicio
      movlw  0x0F     ;carga de w con el valor constante 15 (literal)
      movwf  Cont     ;el contenido de w se pasa al reg. CONT
Loop
      decfsz  Cont,F  ;decremento de Cont y elude siguiente si=0
      goto  Loop      ;salto incondicional a Loop
      goto  $         ;Salto incondicional aquí mismo
      end             ;Fin del código
```

Lista de pasos:

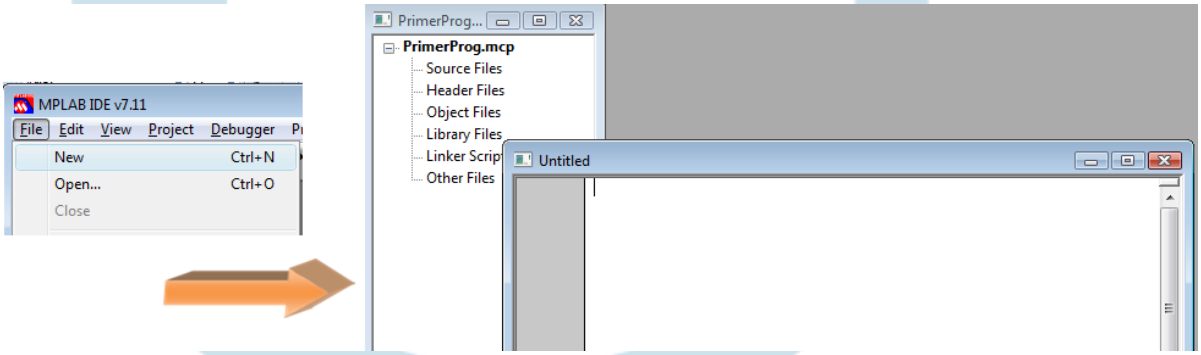
1. Haga doble clic en el ícono correspondiente a MPLAB.



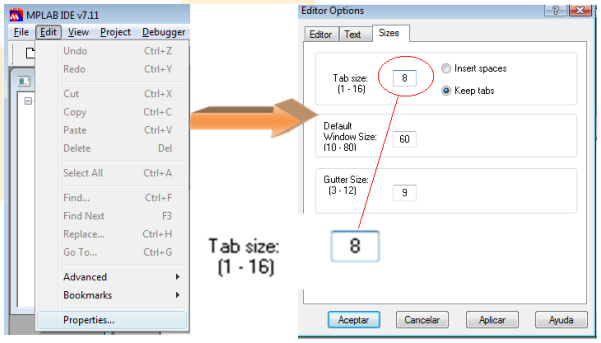
2. Crear el nuevo proyecto menú/Project/New … dar un nombre al proyecto y seleccionar el directorio donde se guardará.



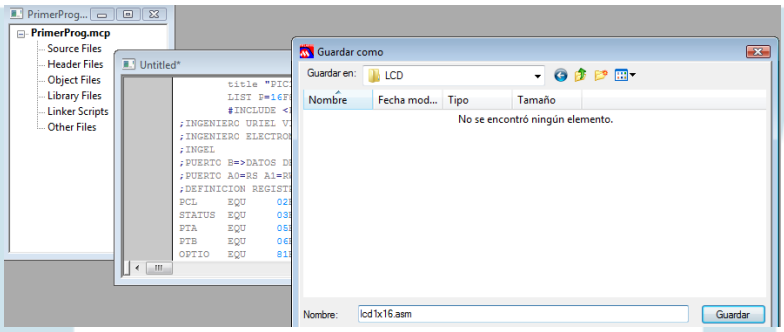
3. Crear un archivo nuevo. Menu/New

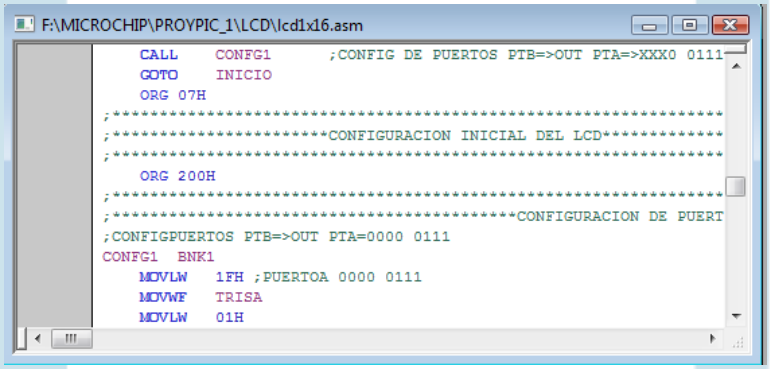


4. Configurar las propiedades del editor. Menu/Edit/Properties. Pestaña sizes/Tab size de 8



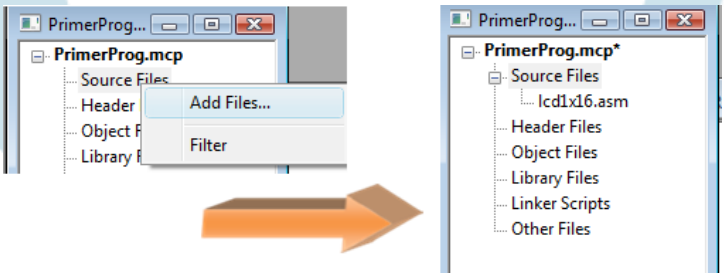
5. Salvar el archivo (con extensión .ASM) una vez terminada su edición. Menú/File/Save.



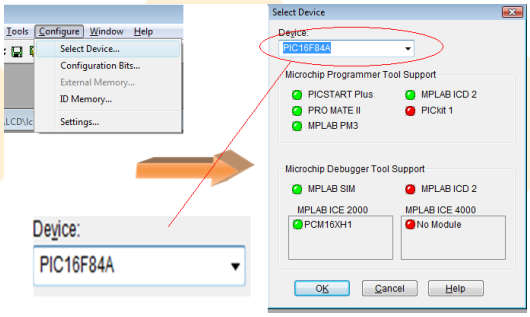


El cambio de color en el texto indica que se toma como “programa fuente”.

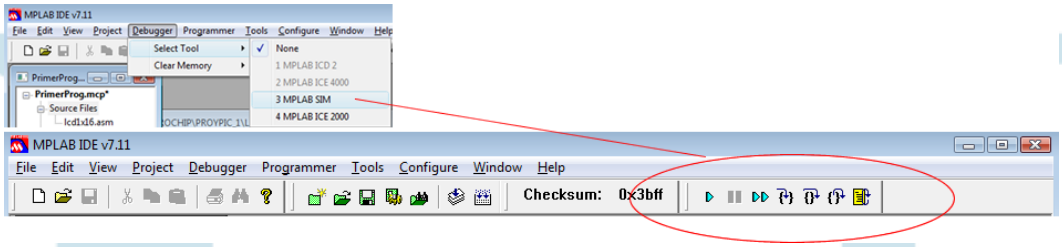
6. En la ventana de proyecto activo .MCW seleccionar SourceFiles/ clic derecho AddFiles y seleccionar el archivo .ASM para incorporarlo al proyecto.



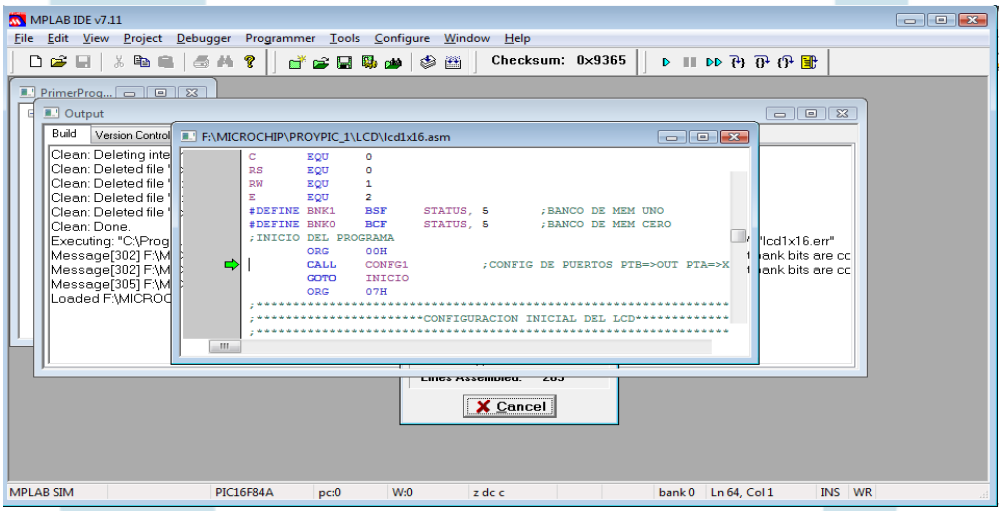
7. Seleccione el componente de la lista de modo que quede el “PIC16F84A”. Menu/Configure/SelecDevice.



8. Para comenzar con el simulador. Menu/Debugger/Select Tool/MPLAB SIM, con lo que aparece una barra de herramientas para la simulación.



9. En el menú/Project/Build All, se realiza la compilación del o "construcción de todo el proyecto".



10. Para realizar una simulación paso a paso, oprima “F7” o utilice las opciones en la barra de simulación: Run (correr programa), Halt (pausa), Animate (animación secuencial) , StepInto (paso a paso), StepOver (salto la subrutinas), StepOut (sale de subrutina), Reset (reinicia).

11. Menu/View/Watch, visualizan los registros desde esta ventana se pueden elegir los registros tanto “Add SFR” o de propósito especial y “Add Symbol” o los símbolos que se han asignados. De la ventana desplegable se selecciona el registro o símbolo y se da un clic en los botones “Add SFR” o “Add Symbol”.

12. Menu/View/Disassembly Listing, desensambla el programa a su código en lenguaje hexadecimal.
13. Menu/View/Hardware Stack, visualiza la pila de 8 niveles para el caso de PIC16F84A.
14. Menu/View/Program Memory, visualiza la memoria de programa, el OPCODE y las instrucciones desensambladas.
15. Menu/View/Files Registers, visualiza los registros de la memoria de datos.
16. Menu/View/EEPROM, muestra los datos almacenados en la EEPROM.
17. Menu/View/Memory Usage Gauge, muestra el estadístico de uso de memoria de datos y de programa.
18. Menu/View/Special Function Registers, muestra los registros de propósito especial o FSR y su contenido.
19. Menu/Debugger/Stimulus Controller/New Scenario, permite simular pulsos externos a los pines del microcontrolador, estos estímulos pueden guardarse como un archivo mas del proyecto.
20. Menu/File/Save Work Space, guarda el espacio de trabajo, lo que permite guardar las ventanas activas de un proyecto y cuando se abra el proyecto nuevamente en otra ocasión visualizar las ventanas que tenia.
21. Menu/Project/Save Project, se encarga de guardar el proyecto activo.
22. Hacer doble clic al lado izquierda de una línea de instrucción dentro del programa fuente después de que se compile, genera “BreakPoints” o puntos de ruptura donde el simulador cuando esta corriendo (Run), para y espera una señal de salto como por ejemplo “F7” para seguir paso a paso. Esta opción es muy utilizada para depurar el programa o hallar errores de decisión o lógicos dentro del programa de control.
23. En caso de tener un programador compatible con MPLAB puede configurar los “bits” de programación del microcontrolador accediendo a Menu/Configure/Configuration Bits, aparece una ventana que despliega las opciones de configuración para el tipo de Oscilador, encendido o apagado del WatchDog Timer, Control al encendido “Power Up Timer” y el código de protección “Code Protect”.
24. Las demás funciones son funciones mas complejas para programación avanzada, recurrir al manual de MPLAB suministrado por Microchip.

CAPITULO 8: PRIMEROS PASOS EN LA PROGRAMACION DE PICs

LECCIÓN 1: INTRODUCCIÓN E IMPLEMENTACIÓN DE CIRCUITOS.

Para poder implementar adecuadamente un proyecto con microcontroladores es necesario tener conocimientos básicos en electrónica análoga y digital, en montajes electrónicos en protoboard y en el manejo de instrumentos de medición, para este capitulo se hará una presentación general de la electrónica involucrada en los proyectos de desarrollo.

Definiciones básicas

La electrónica básica aplicada requiere tener conocimiento y diferenciar claramente los conceptos fundamentales:

- Carga eléctrica: es la cantidad de electrones responsable de los fenómenos eléctricos esta tiene una unidad denominada Coulomb (C). Se establece que la carga eléctrica en un protón es igual a la de un electrón, solo que la del electrón es de signo negativo, este valor es igual 1.6021 e-19 C para un protón y se conoce como carga elemental (q)
- Corriente eléctrica: es entonces la razón o cambio del flujo de carga eléctrica por unidad de tiempo, que pasa por un punto dado. Entonces la corriente se expresaría como la cantidad de cargas **'q'** que pasan por un área determinada en la unidad de tiempo, su unidad de medida es el Amperio, en honor a Andre-Marie Ampere y su símbolo es (I).
- Voltaje eléctrico: El voltaje a través de un elemento es el trabajo necesario para mover una carga positiva de 1 coulomb de la primera a la segunda terminal a través del elemento. La unidad de voltaje es el volt, "V". Entonces hay una caída de voltaje si una carga positiva entrega energía al elemento cuando se desplaza.
- Potencia eléctrica: se mide como la razón de transformación de energía, (en física se define como la cantidad de trabajo en la unidad de tiempo) se simboliza con una " p ", además la energía es la capacidad de realizar trabajo, su símbolo es "w" vatio.

Los instrumentos de medida son utilizados para testear medidas en los componentes electrónicos involucrados en el montaje, recuerde que los microcontroladores tienen un rango de voltaje par su funcionamiento:

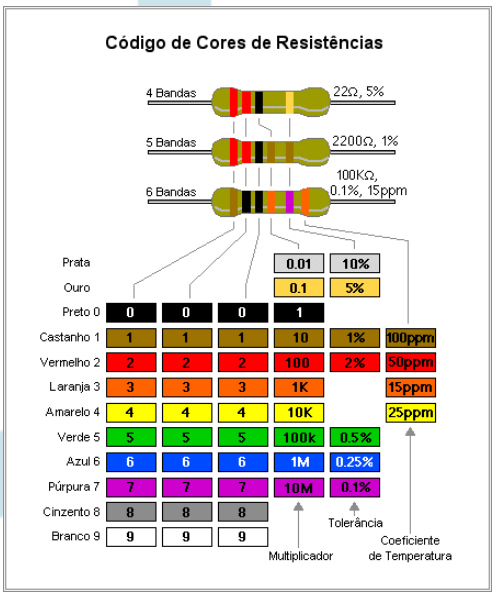
Voltímetros y amperímetros: as mediciones de corriente y voltaje se efectúan con medidores de lectura directa (analógicos) o digitales, en los medidores de lectura directa tienen una aguja indicadora cuya deflexión angular depende de la magnitud de la variable que mide. Un medidor digital muestra una serie de dígitos que indican el valor de la variable medida.

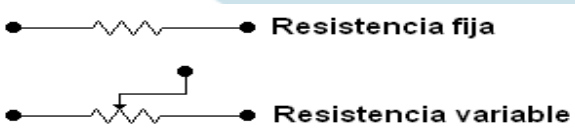
Amperímetro: un amperímetro ideal mide la corriente que fluye por un elemento el modo de conexión es enserie con el elemento, además idealmente tiene caída de voltaje cero entre sus terminales, aunque los instrumentos prácticos presenten caídas de tensión insignificantes comparadas con las tensiones presentes en un circuito eléctrico. Un amperímetro se coloca entre el elemento y la línea de alimentación eléctrica, es decir la corriente circula por el amperímetro.

Voltímetro: un voltímetro ideal mide el voltaje entre sus terminales y tiene en sus terminales una corriente igual a cero, el voltímetro practico presenta corriente en sus terminales, pero también se hace muchas veces insignificantes en comparación con las demás corrientes existentes en el circuito o con la corriente que este circulando por el elemento que este siendo medido. La manera como debe conectarse un voltímetro es en paralelo con el elemento, el voltímetro se coloca entre los terminales del elemento que se esta midiendo. Los componentes electrónicos se dividen en activos y pasivos

Elementos pasivos: son los elementos que absorben energía este es el caso de los siguientes componentes:

- Resistencias: Es la propiedad física de un elemento o dispositivo que impide el flujo de corriente; se representa así: R. La unidad de resistencia se expresa en ohm (Ω), generalmente los valores de miles o millones de resistencia se expresan en kiloOhms (KΩ) o MegaOhms (MΩ) respectivamente. Los valores nominales de los resistores en operación están generalmente basados en 25 ^{o}C





- Condensadores: son elementos conformados por placas de conductoras en paralelo separadas por un material aislante, su unidad de medida son los Faradios, generalmente expresados en microfaradios (uF), nanofaradios (nF) o picofaradios (pF), se simboliza con una “C”
- Inductancias: están fabricadas por vueltas de alambre aislado enrollado en núcleos vacios o de material ferroso, su unidad es el Henry (H), se simboliza por “L”.

Elementos Activos: un elemento activo, es activo si es capaz de entregar energía, tal es el caso de las fuentes de alimentación.

- Corriente Continua a C.C, compuesta por pilas, baterías y adaptadores que convierten la corriente A.C en C.C.
- Corriente Alterna A.C, compuesta por la energía eléctrica proveniente de la toma común en la alimentación domestica.

Elementos Semiconductores: dentro de esta familia esta la gama de diodos, transistores y dispositivos de cuatro capas.

- Diodos, son dispositivos semiconductores formados por una capa tipo N con exceso de electrones con carga eléctrica negativa, llamada “Cátodo” y unida con una capa tipo P con exceso de huecos con carga eléctrica positiva llamada “Ánodo”. Los diodos dejan circular corriente eléctrica en un solo sentido cuando el “Ánodo” se conecta al terminal más positivo y el “Cátodo” al más negativo. Dentro de las familias de diodos se encuentran:
  - Diodos de unión utilizados para rectificación y puentes rectificadores.
  - Diodos zener como referencias de voltaje para regulares eléctricos.
  - LEDs los cuales pueden esta de forma individual y emiten un haz de luz o en grupo como matrices o display de siete segmentos.
- Transistores, son dispositivos de tres capas de los cuales se distinguen los siguientes:
  - BJT, compuesto por dos capas de tipo N entre una tipo P, creando el transistor NPN y también dos capas tipo P entre una tipo N, creando el transistor PNP, estos transistores tienen tres terminales.
    - La Base: por la cual llega la señal de poca intensidad, debe tener la misma polaridad del tipo de capa.
    - El Colector: se conecta a la terminal opuesta al tipo de capa en ella se conecta la carga a manejar que requiere mayor potencia para su activación.
    - El Emisor: se conecta al terminal que coincide con la polaridad del tipo de capa, este es el que recoge el flujo de corriente.
  - FET, son transistores de efecto de campo, los hay de canal N y canal P, funcionan de forma similar a los BJT solo que el consumo de potencia es menor aunque son mas delicados por la tecnología CMOS con la que se fabrican

- Dispositivos de cuatro capas, estos dispositivos se construyen con cuatro capas alternas de material tipo N y P con lo que se obtienen tres dispositivos muy conocidos:
  - SCR, o tiristor utilizado para controlar cargas alimentadas con Corriente Alterna o Corriente Continua.
  - TRIAC es un dispositivo que permite el control de cargas alimentadas por C.A.
  - DIAC utilizado como dispositivo de disparo para los SRC o TRIACs este dispositivo funciona en C.A o C.C y permiten la conducción cuando supera un voltaje de disparo.

Símbolos eléctricos, electrónicos y digitales más usuales en la implementación de proyectos:

Figura 94. ***Símbolos de resistencias, Condensadores, bobinas Diodos e interruptores***

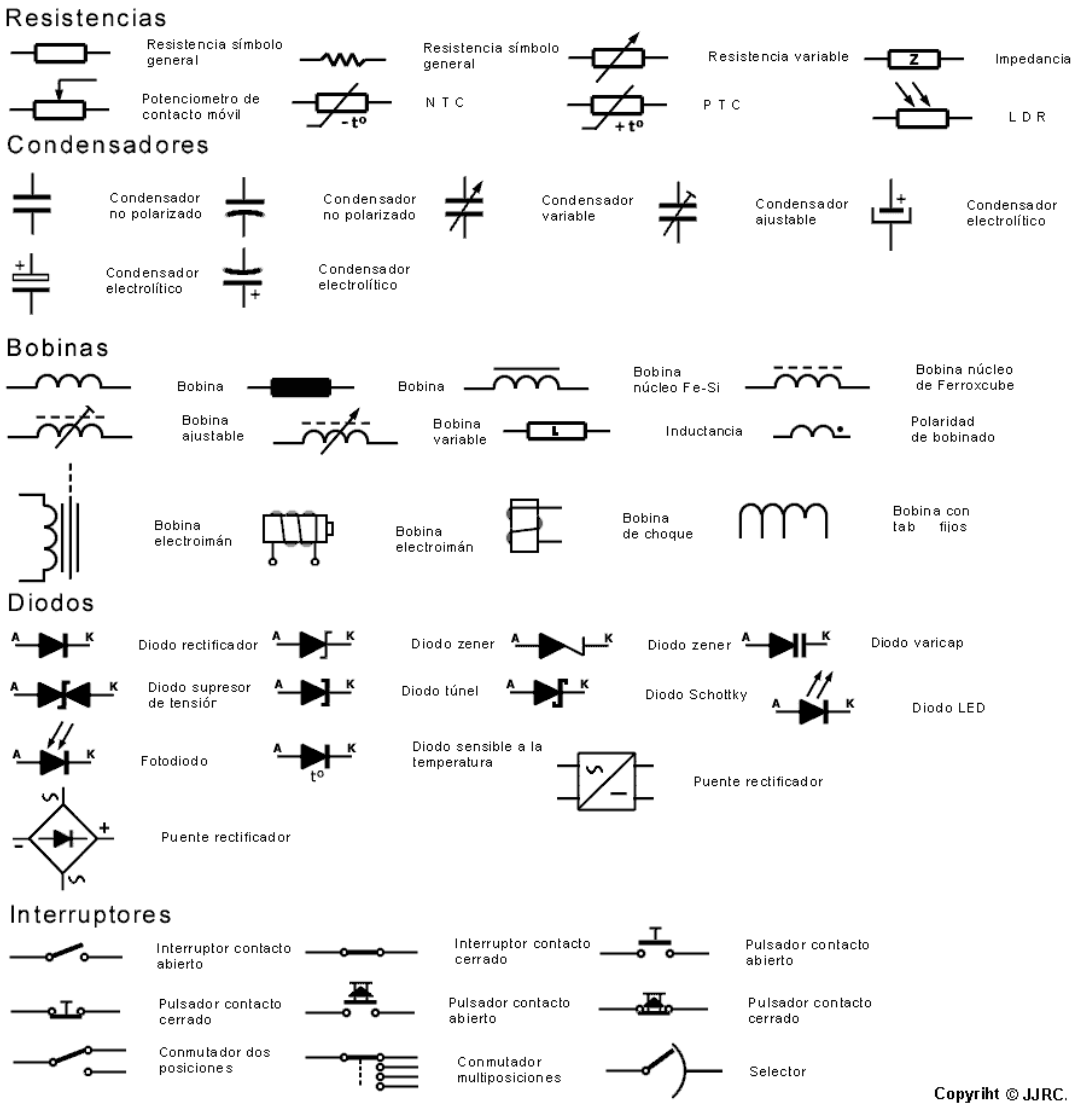


Figura 95. ***Símbolos de instrumentación, dispositivos activos, transistores, tiristores y electrónica Digital***

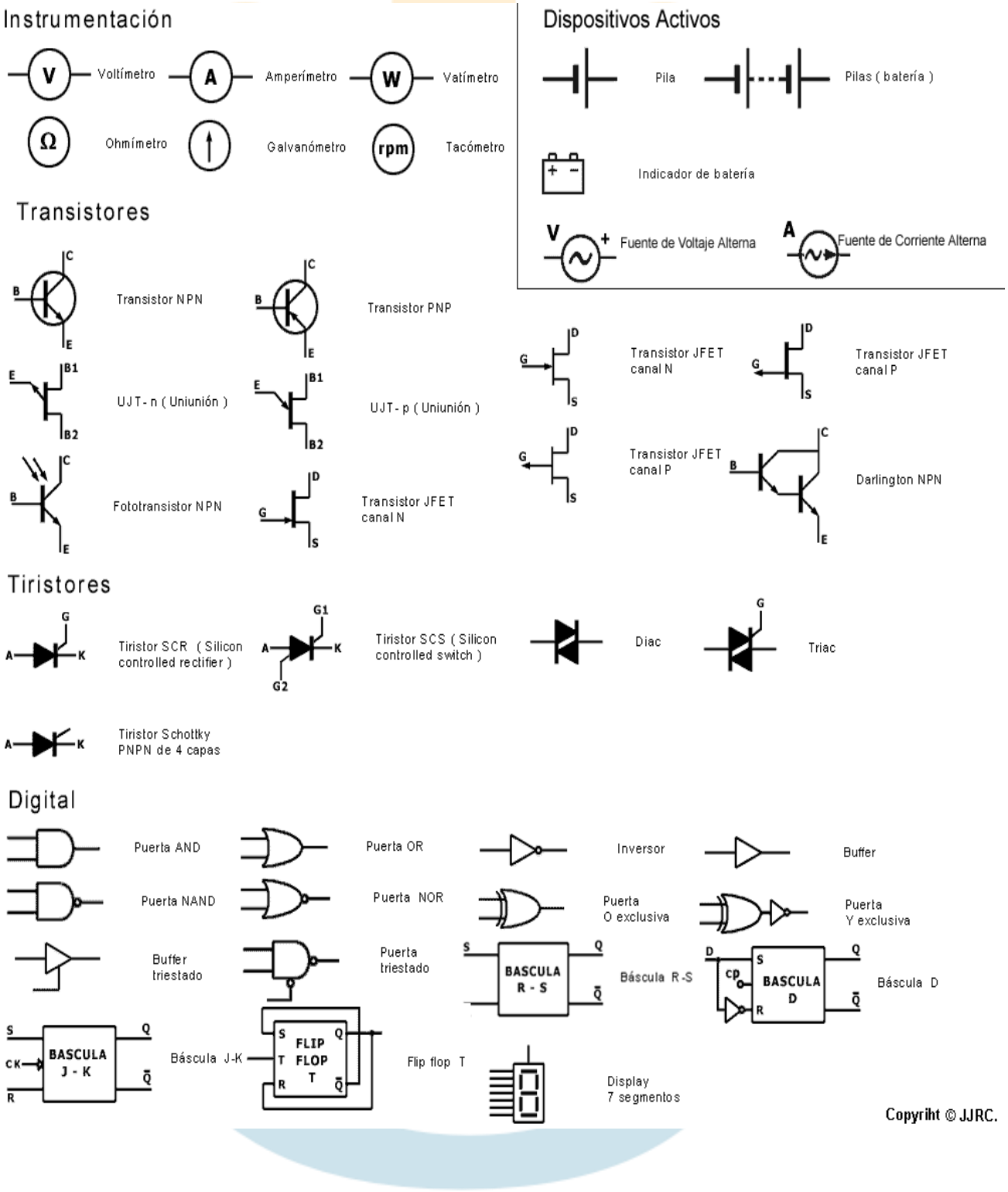
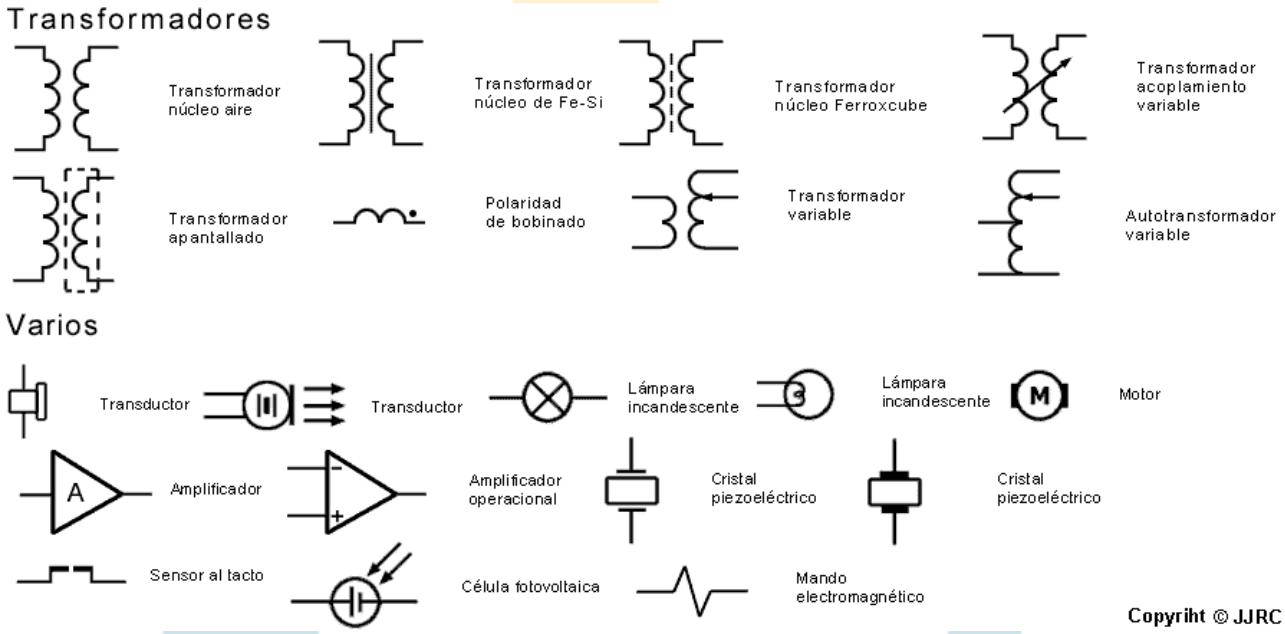


Figura 96. ***Símbolos de transformadores y otros de uso común.***



LECCIÓN 2: SUBRUTINAS Y LLAMADOS.

El programa fuente está compuesto por una sucesión de líneas de programa. Cada línea de programa está compuesta por 4 campos separados por uno o más espacios o tabulaciones. Estos campos son:

[Etiqueta] Comando [Operando(s)] [;Comentario]

La etiqueta es opcional. El comando puede ser un mnemónico del conjunto de instrucciones. El operando está asociado al comando, si no hay comando no hay operando, e inclusive algunos comandos no llevan operando. El comentario es opcional para el compilador aunque es buena práctica considerarlo obligatorio para el programador.

La etiqueta, es el campo que empieza en la primera posición de la línea. No se pueden insertar espacios o tabulaciones antes de la etiqueta sino será considerado comando. Identifica la línea de programa haciendo que el compilador le asigne un valor automáticamente. Si se trata de una línea cuyo comando es una instrucción de programa del microcontrolador, se le asigna el valor de la dirección

de memoria correspondiente a dicha instrucción (location counter). En otros casos se le asigna un valor de una constante, o la dirección de una variable, o será el nombre de una macroinstrucción, etc.

El comando puede ser un código mnemónico de instrucción del microcontrolador, o una directiva o pseudoinstrucción para el compilador. En el primer caso será directamente traducido a código de máquina, en el segundo caso será interpretado por el compilador y realizara alguna acción en tiempo de compilación como ser asignar un valor a una etiqueta, etc.

El campo de parámetros puede contener uno o más parámetros separados por comas. Los parámetros dependen de la instrucción o directiva. Pueden ser números o literales que representen constantes o direcciones.

El campo de comentario debe comenzar con un carácter punto y coma. No necesita tener espacios o tabulaciones separándolo del campo anterior, e incluso puede empezar en la primera posición de la línea. El compilador ignora todo el texto que contenga la línea después de un carácter punto y coma. De esta manera pueden incluirse líneas que contengan solo comentarios, y es muy buena práctica hacer uso y abuso de esta posibilidad para que los programas resulten autodocumentados.

Importancia de las rutinas

La mayoría de los microcontroladores incluyen en su repertorio de instrucciones algunas que permiten saltar a una rutina y, cuando se complementa su ejecución, retornar al programa principal

El empleo de subrutinas aporta muchas ventajas entre las que se destacan las siguientes:

1. Se pueden escribir como subrutinas secciones de código y ser empleadas en muchos programas (por ejemplo, la subrutina de exploración de un teclado).
2. Dan a los programas un carácter modular, es decir, se pueden codificar diferentes módulos para usarlos en cualquier programa.
3. Se reduce notablemente el tiempo de programación, la detección de errores, usando repetidamente una subrutina.
4. El código es más fácil de interpretar, dado que las instrucciones de las subrutinas no aparecen en el programa principal. Solo figuran las llamadas CALLs.

Las instrucciones call y return

La instrucción CALL (llamada la subrutina) consigue que la ejecución del programa continúe en la dirección donde se encuentra la subrutina a la que hace referencia. Es similar a GOTO pero coloca en la pila la dirección de la siguiente instrucción que se debe ejecutar después de la CALL.

La subrutina finaliza con la instrucción RETURN (Retorno de la subrutina) que retoma la dirección guardada en le pila y la coloca en el contador del programa PC continuando el flujo de control con la instrucción que sigue a la CALL.

En la familia PIC de gama media la pila tiene ocho niveles de memoria del tipo FIFO (primero en entrar, último en salir). Si se produce la llamada a una subrutina durante la ejecución de otra subrutina, la dirección de retorno de esta segunda es colocada en la cima de la pila sobre la dirección anterior. Esta segunda dirección es la primera en salir de la pila mediante la instrucción RETURN.

Con la pila de ocho niveles, una subrutina puede llamar a otra y ésta, a su vez, llamar a otra hasta un máximo de ocho. La gama baja sólo puede realizar dos llamadas de este tipo al poseer una pila de sólo dos niveles.

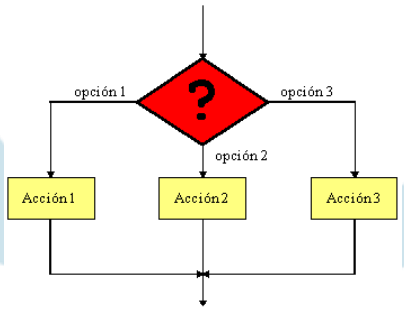
Las subrutinas deben colocarse al comienzo de las páginas debido a que el bit 8 del contador del programa es puesto a 0 por la instrucción CALL (o por cualquier instrucción que modifica el PC). Las subrutinas deben colocarse en la mitad inicial de las páginas (las 256 palabras).

LECCIÓN 3: RAMIFICACIONES EN LOS PROGRAMAS CON PIC.

Ramificación múltiple

Cuando se tiene que solucionar un diagrama de flujo como el de la figura, en el cual tenemos tres posibles respuestas a una pregunta, se plantean las soluciones aquí presentadas.

Figura 97. ***Tres posibilidades para una pregunta***



Primera Solución

Una de las formas de solucionar en un programa este problema es:

Determinando para la opción 1, la opción 2 y la opción 3 un valor consecutivo como:

```
opción1   equ   0
opción2   equ   1
opción3   equ   2
```

Uno de estos posibles valores llevarlos a W y en una parte del programa tratarlos así:

```
Decisión:                ;sitio en donde la pregunta "?" tendría solución
   addwf   PCL,1
   goto     Acción1
   goto     Acción2
   goto     Acción3
Acción1:
.....                    ;instrucciones correspondientes a la Acción 1

..... .....
   goto encuentro
Acción2:
.....                    ;instrucciones correspondientes a la Acción 2
.....
.....
   goto encuentro
Acción3:
.....                    ;instrucciones correspondientes a la Acción 3
.....
.....
encuentro:               ;sitio de encuentro luego de una de las acciones
.....                    ;continuación del programa
.....
```

Segunda Solución

Otra forma posible es comparando una por una los valores de las diferentes opciones almacenadas en memoria RAM en una variable llamada **OPCION**

```
   movlw    Opción1
   xorwf    OPCION,0    ;se realiza la verificación del contenido de OPCION con respecto a W
   btfsc     STATUS,Z    ;Verificando la bandera Z
   goto      Acción1
   movlw    Opción2
   xorwf    OPCION,0    ;se realiza la verificación del contenido de OPCION con respecto a W
   btfsc     STATUS,Z    ;Verificando la bandera Z
   goto      Acción2
   movlw    Opción3
   xorwf    OPCION,0    ;se realiza la verificación del contenido de OPCION con respecto a W
   btfsc     STATUS,Z    ;Verificando la bandera Z
```

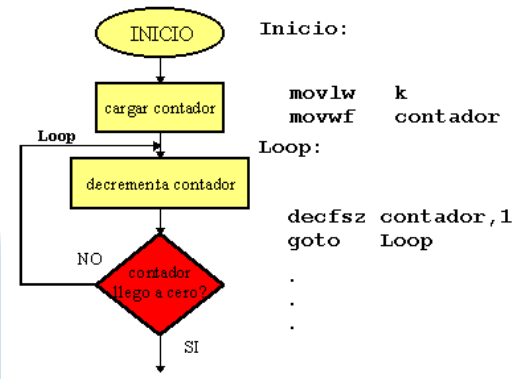
```
   goto      Acción3
Acción1:
.....     ;instrucciones correspondientes a la Acción 1
.....
.....
   goto encuentro
Acción2:
.....     ;instrucciones correspondientes a la Acción 2
.....
.....
   goto encuentro
Acción3:
.....     ;instrucciones correspondientes a la Acción 3
.....
.....
encuentro:          ;sitio de encuentro luego de una de las acciones
.....     ;continuación del programa
.....
```

Aunque este último método es más largo que el anterior, permite que los valores de las diferentes opciones no sean consecutivos entre si.

Ciclos de Temporización por software

Existen momentos dentro de la programación en los que se necesita realizar un retardo de tiempo. Los retardos de tiempo se pueden obtener mediante hardware o por medio de ciclos repetitivos basados en software. La precisión de los retardos generados por software depende en esencia del tipo de oscilador que se utilice como base de tiempo en el microcontrolador, la mayor precisión se obtiene de los cristales de cuarzo.

Figura 98. ***Temporizador***



La velocidad a la que se ejecuta el código (instrucciones) depende de la velocidad del oscilador y del número de ciclos de máquina ejecutados. Las instrucciones necesitan 1 ó 2 ciclos de máquina para ser ejecutadas. Un ciclo de máquina es un tiempo utilizado por el microcontrolador para realizar sus operaciones internas y

equivale a cuatro ciclos del oscilador. Por tanto: $T_{ciclo\ máq.} = 4 * T_{osc}$ □ □ □ $T_{ciclo\ máq} = 4 / f_{osc}$ El número de ciclos de máquina utilizados por una instrucción para ser ejecutada depende de la misma. Las instrucciones que modifican el contador de programa necesitan dos (2) ciclos de máquina, mientras que todas las demás necesitan tan solo uno (1).

El hecho de generar ciclos repetitivos por medio del programa y calcular el tiempo total de ejecución nos puede ayudar a generar tiempos precisos.

Ciclo repetitivo de retardo

Tomando el ciclo repetitivo de retardo mostrado en el diagrama de flujo, se tomará un número de ciclos así:

Operación	**# de ciclos**
la carga de k en W	1
la carga de W en el contador	1
el decremento del contador mientras no llegue a cero	k-1
el decremento del contador cuando llegue a cero	2
el salto a Loop	2 * (k-1)
Total:	**3*k+1**

Por cada instrucción agregada debe incluirse en la cuenta total el número de ciclos correspondiente a dicha instrucción. Trabajando a 4 Mhz y asumiendo que k se remplaza por el valor 15_d en el ejemplo tendríamos

n tiempo igual a:
Número de ciclos = (3*15) +1 = 46 ciclos de máquina,
$T_{ciclo\ máq.} = 4 / 4$ Mhz = 1 µ segundo, el tiempo total del ejemplo entonces será 46 µsegundos.

LECCIÓN 4: CONSULTAS DE TABLAS.

En muchas ocasiones es necesario para un programador efectuar una coincidencia entre alguna cantidad de valores conocidos y un número desconocido que se tiene como índice, por ejemplo, basados en el contenido de una posición de memoria RAM (índice) se puede obtener de una serie consecutiva de datos

almacenados en memoria de programa (a estos datos "conocidos" almacenados se le denomina tabla), el dato desplazado ***n*** posiciones adelante del comienzo de esta tabla, este número ***n*** corresponde al contenido de la posición de memoria RAM ó índice.

Programa ejemplo:

```
offset   equ   0Ch    ;posición de memoria RAM
w        equ   0       ;destino W
f        equ   1       ;destino F
.................
.................
.................
   movf     offset,w  ;tomamos a W el número n utilizado como índice
   call     tabla     ;posición en donde se encuentra la serie de datos
                      ;en este sitio luego del retorno de la subrutina se tiene en W el dato leído
e la tabla
.................
.................
.................
tabla
   addwf    PCL,f   ;se suma al PC el contenido de W obteniendo como resultado un salto Indexado
   retlw     30h    ;sí el contenido de W sumado al PCL es 0 se retorna en esta posición, =30h
   retlw     31h    ;sí el contenido de W sumado al PCL es 1 se retorna en esta posición,  =31h
   retlw     32h    ;sí el contenido de W sumado al PCL es 2 se retorna en esta posición, =32h
   retlw     33h    ;sí el contenido de W sumado al PCL es 3 se retorna en esta posición, =33h
   retlw     34h    ;sí el contenido de W sumado al PCL es 4 se retorna en esta posición, =34h
   retlw     35h    ;sí el contenido de W sumado al PCL es 5 se retorna en esta posición, =35h
   .                ;...
```

Finalmente y luego de observar el ejemplo anterior, podemos anotar que antes de hacer el llamado a la subrutina **tabla**, se debe cargar en el registro de trabajo W el valor del índice y una vez se retorne de dicha subrutina, es en este mismo registro de trabajo en donde se obtiene el resultado de la consulta a la tabla (vemos que la sucesión de instrucciones **retlw k** se encuentra en memoria de programa).

En los microcontroladores PIC las tablas deben estar dentro de los primeros 256 Bytes de memoria de programa, esto es porque solo se tiene acceso y modificación a los primero 8 bits del registro “PC o contador de programa”, es decir, desde el bit cero al bit siete.

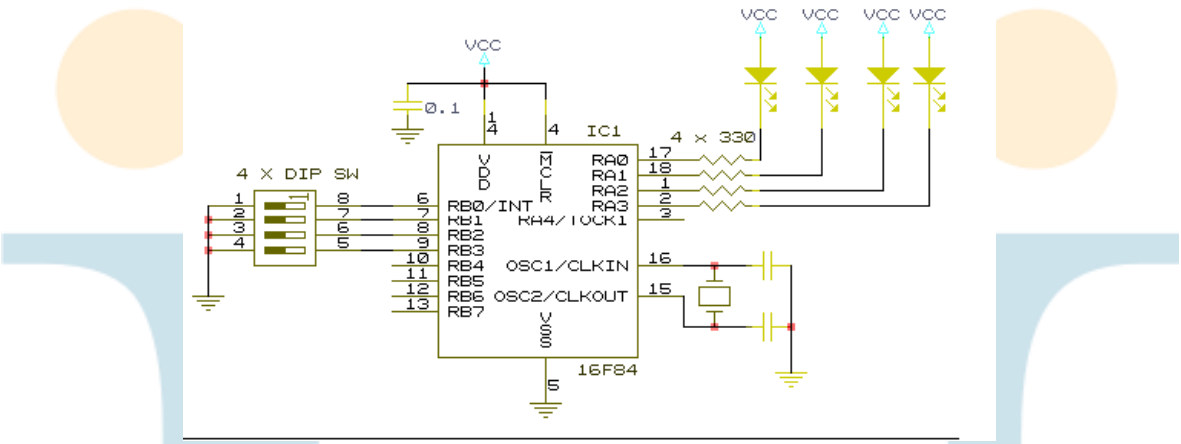
LECCIÓN 5: OPERACIÓN DE ENTRADA/SALIDA.

Para la operación de entrada/salida se debe tener presente:

- Verificar el modo en el que se debe programar el sentido de los puertos

- Realizar la entradas por puerto mediante la lectura de interruptores "dip-switch"
- Escribir sobre un puerto de salida visualizando sobre LEDs

Figura 99. ***Hardware para ejercicio Entrada/Salida***



Procedimiento:

En el proceso de utilización de un puerto debe tenerse en cuenta como primera instancia la programación del sentido en que dicho puerto va a utilizarse. Una vez energizado el microcontrolador todos y cada uno de los puertos quedan programados como entrada, entonces, tan solo deben programarse los que se quieren utilizar como salida.

En el hardware de la figura se observa que se han colocado 4 dip switch al puerto B y estos no poseen resistencia de pull-up lo cual nos obliga a habilitar las resistencias internas con las que cuenta el microcontrolador PIC16F84, el programa debe entonces en un repetitivo infinito leer el nivel lógico que colocan los switch y pasar este resultado al puerto A complementando el estado de la información puesto que de acuerdo a la disposición de los LEDs un estado bajo en el puerto enciende el LED correspondiente y por ende un estado alto en el puerto, apaga el LED.

Programa:

```
status   equ  03h
optionr  equ  81h
trisa    equ  85h
porta    equ  05h
trisb    equ  86h
portb    equ  06h
;
Inicio:
```

```
    bsf   status,5  ;se pasa al banco 1 de RAM
    clrf  trisa     ;se programa el puerto A como salida
    movlw 0Fh       ;dato para la programación del puerto B
    movwf trisb     ;parte alta como salida y parte baja como entrada
    bcf   optionr,7 ;se habilitan resistencias de Pull Up
    bcf   status,5  ;se pasa al banco 0 de RAM
Loop:
    comf  portb,0   ;se lee el puerto B, se complementa su valor y el    ;resultado pasa a W
    movwf  porta    ;se pasa el resultado de W al puerto A
    goto   Loop     ;ejecuta un ciclo infinito
    end
```

En un proceso de lectura de interruptores, casi siempre cuando se detecta un cambio en el estado, es aconsejable amortiguar la lectura de estos con un retardo de software. Dependiendo de la calidad del interruptor el tiempo del retardo puede estar alrededor de 50 mS. En el caso de este ejercicio en particular no es requerido puesto que un cambio en el interruptor debe reflejarse inmediatamente en el puerto de salida. Se debe tener en cuenta que nunca una entrada debe quedar al aire puesto que los microcontroladores PIC son hechos con tecnología CMOS. Es por este motivo que en el programa se programó la parte alta del puerto B como salida.

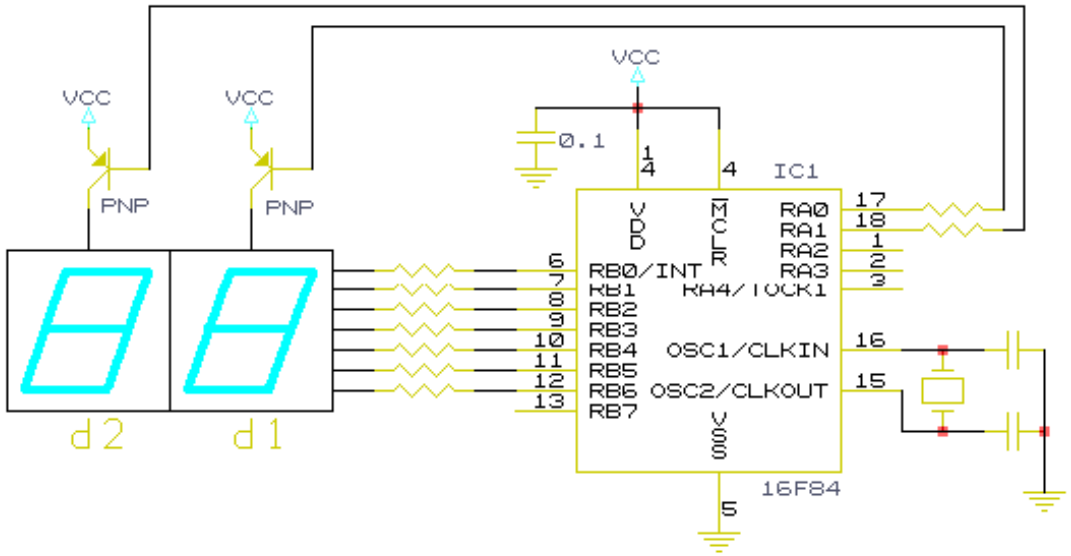
CAPITULO 9: PROYECTOS DE APLICACION

LECCIÓN 1: MANEJO DE DISPLAY 7 SEGMENTOS.

Objetivos:

- Realizar la decodificación de BCD a 7 segmentos por software
- Multiplexar en el tiempo la información para 2 dígitos 7 segmentos

Figura 100. ***Diagrama eléctrico para visualización dinámica en display 7 segmentos de dos dígitos***



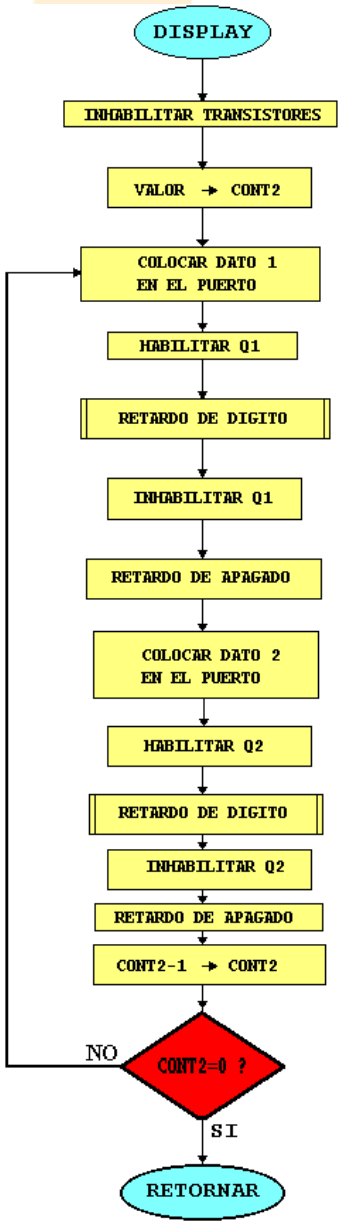
Procedimiento:

El hecho de visualizar datos en display de 7 segmentos de la forma en que se muestra en la figura anterior obliga a interconectar entre si los pines

correspondientes a los segmentos del dígito 1 (d1) con los pines de los segmentos del dígito 2 (d2), de esta manera se ahorran líneas de conexión. El método utilizado para la visualización dinámica consiste en visualizar cada dígito durante un instante de tiempo y a una velocidad tal que gracias a la persistencia del ojo el efecto final es que todos los dígitos están encendidos al tiempo.

```
;Sacar al puerto el Dato 2 por un tiempo específico

        movf    Dato2,W     ;Dato para decodificar
        call    Tabla       ;Decodificación del dato
        movwf  PORTB        ;Dato decodificado a puerto
        bcf     PORTA,1     ;Habilita Q dato 2
        call    RetDig      ;Retardo de dígito
         bsf    PORTA,1     ;Inhabilita Q dato 2
         nop                ;Retardo de apagado
         nop
         nop
         nop                                                ;
        decfsz  Cont2,F     ;Decrementa Cont2, elude sig. sí cero
         Goto   LoopDisp    ;Repite ciclo
         return
Tabla
        addwf  PCL,F
        retlw  0x01         ;Cuando el dígito es 0
        retlw  0x4F         ;Cuando el dígito es 1
        retlw  0x12         ;Cuando el dígito es 2
        retlw  0x06         ;Cuando el dígito es 3
        retlw  0x4C         ;Cuando el dígito es 4
        retlw  0x24         ;Cuando el dígito es 5
        retlw  0x20         ;Cuando el dígito es 6
        retlw  0x0F         ;Cuando el dígito es 7
        retlw  0x00         ;Cuando el dígito es 8
        retlw  0x04         ;Cuando el dígito es 9
        retlw  0x08         ;Cuando el dígito es A
        retlw  0x60         ;Cuando el dígito es B
        retlw  0x31         ;Cuando el dígito es C
        retlw  0x42         ;Cuando el dígito es D
        retlw  0x30         ;Cuando el dígito es E
        retlw  0x38         ;Cuando el dígito es F
RetDig
        movlw  2
        movwf  Del1
Loop1
        movlw  .50
        movwf  Del2
Loop2
        decfsz Del2,F
        goto   Loop2
        decfsz Del1,F
        goto   Loop1
        return
```

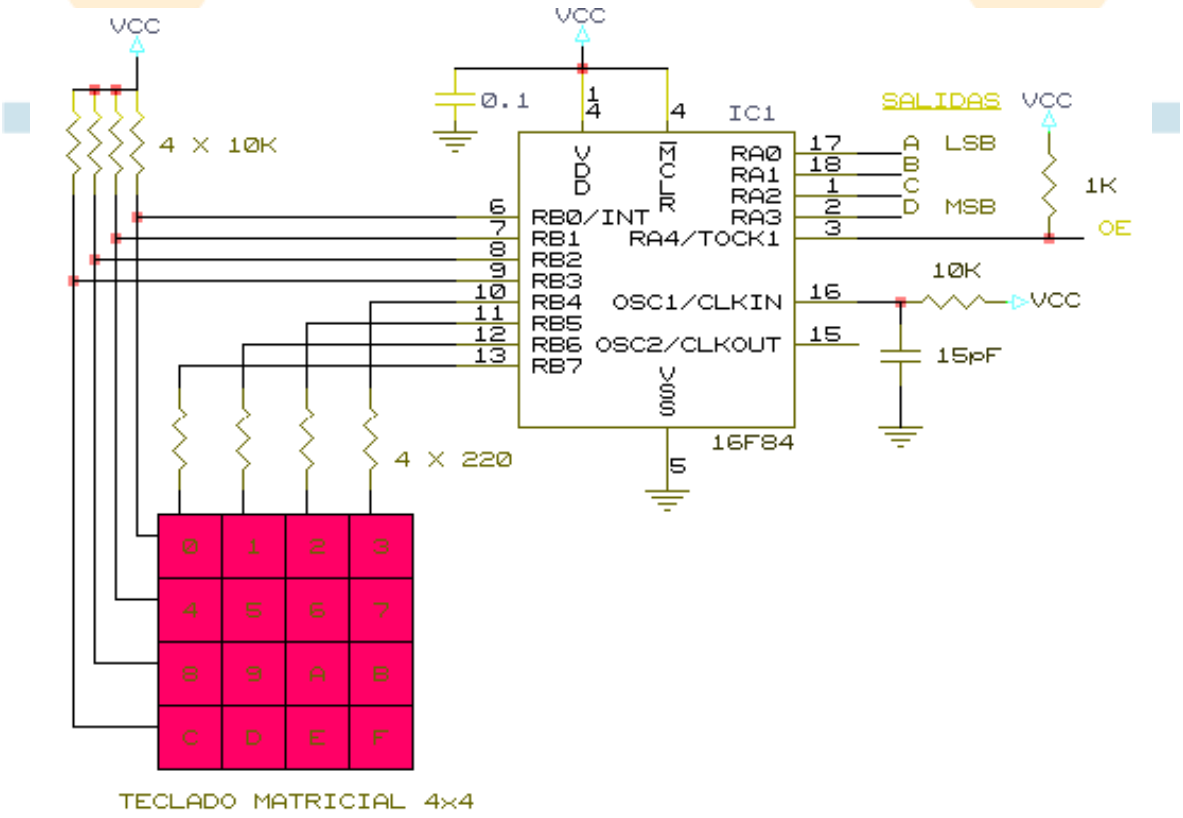


LECCIÓN 2: EXPLORACIÓN DE TECLADO.

Exploración de teclado

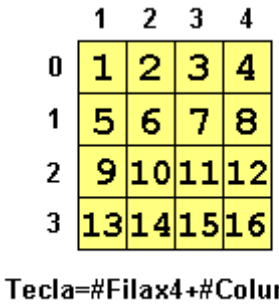
Para la lectura del teclado debemos tener en cuenta la disposición de las filas y las columnas como se observa en la figura con la cual realizando la operación allí descrita se debe obtener un número consecutivo de las teclas en la organización mostrada.

Figura 101. ***Hardware correspondiente al experimento de exploración de teclado***



Luego, mediante el acceso a una tabla se decodifica la tecla leída para obtener el patrón final observado en el diagrama del hardware de la figura anterior. Ej. Sí se oprime la tecla C del teclado, el código de exploración correspondiente a esta es el 13d que debe ser representado como el 1100b en las salidas DCBA.

Figura 102. ***Distribución del teclado, numeración en filas y columnas y la fórmula para hallar la tecla oprimida***



En este experimento se realiza la emulación del integrado decodificador de teclado 74C922 en cuanto a su funcionamiento, pero de acuerdo a la configuración de hardware.

Listado del programa para exploración de teclado:

```
   list p=16f84
#include <p16f84.inc>    ;archivo de encabezado por Microchip®
;
;ESTE PROGRAMA EMULA UN 74C922 DECODIFICADOR DE TECLADO
;
CONTFIL EQU  0x12      ;Contador de Filas
CONTCOL EQU  0x13      ;Contador de Columnas
COLKBD EQU  0x14       ;DATO EN COLUMNAS
Temp EQU 0x15
R1 EQU 0x16           ;Variable para Retardo
R2 EQU 0x17           ;Variable para Retardo
R3 EQU 0x18           ;Variable para Retardo
R4 EQU 0x19           ;Variable para Retardo
COUNT EQU 0x1A
CHAR EQU 0x1B         ;Almacenamiento temporal SCAN
AUX EQU 0x1C          ;Variable Auxiliar
#define _z STATUS,2
#define _c STATUS,0
#define OE PORTA,4
#define BANK0  bcf STATUS,RP0
#define BANK1  bsf STATUS,RP0
;
;.....................................................................
   ORG 0x00
MAIN
   BANK1
   CLRF TRISA
   BANK0
   CLRF PORTA
   NOP
   NOP
Muestre
   BCF OE
RUSCAN
```

```
    CALL SCAN
    XORLW 0x00      ;espera una Tecla
    BTFSC _z
    GOTO Muestre
    MOVWF PORTA
    MOVLW .50
    MOVWF COUNT
LOOPSCAN
    CALL DEL5MS     DECFSZ COUNT,1
    GOTO LOOPSCAN
    GOTO RUSCAN
;*********************************************************************************
DEL5MS
    MOVLW .12
    MOVWF R1
    MOVLW 7
    MOVWF R2
    MOVLW 1
    MOVWF R3
    MOVLW 1
    MOVWF R4
LOOPDEL5
    DECFSZ R1,F
    GOTO LOOPDEL5
    DECFSZ R2,F
    GOTO LOOPDEL5
    DECFSZ R3,F
    GOTO LOOPDEL5
    DECFSZ R4,F
    GOTO LOOPDEL5
    NOP

    RETURN
;*********************************************************************
;RETORNA W=00 NO HAY TECLA OPRIMIDA,
;RETORNA W=COD SI TECLA OPRIMIDA.
;*********************************************************************
SCAN
    BANK1
    MOVLW 0x0F        ;el puerto que lee teclado <0:3> filas (in)
    MOVWF TRISB
    BANK0
    MOVLW 0x01
    MOVWF CONTCOL
    MOVLW 0x7F
    MOVWF COLKBD
RSTFIL
    CLRF CONTFIL      ;RESET CONT FILAS
    MOVF COLKBD,W
    MOVWF PORTB       ;COLOCAR UN CERO EN COLUMNAS
    nop
    nop
    nop
    MOVF PORTB,W      ;LEER FILAS DE TECLADO
    MOVWF AUX
    RLF AUX,F
    RLF AUX,F
    RLF AUX,F
    RLF AUX,F
TESTFIL
    RLF AUX,F
    BTFSS _c
```

```
    GOTO ACERTADO
    INCF CONTFIL,F
    MOVF CONTFIL,W
    XORLW 0x04
    BTFSS _z
    GOTO TESTFIL
    BSF _c
    RRF COLKBD,F      ;rotacion del cero a colocar
    INCF CONTCOL,F
    MOVF CONTCOL,W
    XORLW 0x05
    BTFSS _z
    GOTO RSTFIL
RETLW 0x00
ACERTADO
    MOVF CONTFIL,W
    XORLW 0x00
    BTFSC _z
    GOTO ESCERO
    MOVLW 0x00
MUL
    ADDLW 0x04
    DECFSZ CONTFIL
    GOTO MUL SUMACOL
    ADDWF CONTCOL,W
    CALL TABKBD
    RETURN
TABKBD
   addwf PCL,F
    retlw 0       ;inválido
    retlw 0x10
    retlw 0x11
    retlw 0x12
    retlw 0x13
    retlw 0x14
    retlw 0x15
    retlw 0x16
    retlw 0x17
    retlw 0x18
    retlw 0x19
    retlw 0x1A
    retlw 0x1B
    retlw 0x1C
    retlw 0x1D
    retlw 0x1E
    retlw 0x1F
ESCERO
    MOVLW 0x00
    GOTO SUMACOL
    end
```

LECCIÓN 3: INTERFAZ PIC – DISPLAY LCD.

La interface que se va a implementar, para el trabajo con el *display* de cristal líquido, es la del bus de datos de 8-*bits*. Antes de presentar un programa que

controla el comportamiento del *display*, se van a describir en el siguiente cuadro las instrucciones típicas para su manejo y configuración.

Tabla 26. ***Pines y funciones LCD***

Instrucción	RS	R/W	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0	Descripción
Limpiar *Display*	0	0	0	0	0	0	0	0	0	1	limpia el *display* y lleva el cursor a la dirección cero
Cursor *at Home*	0	0	0	0	0	0	0	0	1	*	lleva el cursor a la dirección cero sin cambiar el contenido
Modo *entry set*	0	0	0	0	0	0	0	1	I/D	S	controla el movimiento del cursor y activa el desplazamiento del *display*
Control ON/OFF	0	0	0	0	0	0	1	D	C	B	activa o desactiva el *display, el* cursor y el parpadeo del cursor
Cursor / *Display*	0	0	0	0	0	1	S/C	R/L	*	*	mueve el cursor o desplaza el *display* sin cambiar al contenido de la pantalla
Función *SET*	0	0	0	0	1	DL	N	F	*	*	define el ancho de interface, el número de líneas del *display* y caracteres
Dirección DDRAM	0	0	0	1	ACG						define la dirección en el *display* del RAM del generador de caracteres
Dirección DDRAM	0	0	1	ADD							define la dirección en el *display* del RAM de datos del *display*
Busy Flag / *Addres Read*	0	1	BF	AC							indica si la operación interna está en proceso y lee direcciones de contenido
Escritura de Datos	1	0	Escritura de Datos								escribe los datos en el DDRAM o en el CGRAM
Lectura de Datos	1	1	Lectura de Datos								lee datos del DDRAM o del CCRAM

En el cuadro de instrucciones del *display* se emplearon las siguientes abreviaturas:

* : posición no válida.
I / D : dirección del cursor. 1, hacia la derecha. 0, hacia la izquierda.
S : desplazamiento del *display*. 1, activado. 0, desactivado.
D : *display*. 1, activado. 0, desactivado.
C : cursor. 1, activado. 0, desactivado.
B : parpadeo del cursor. 1, activado. 0, desactivado.
S /C : desplazamiento / cursor. 1, desplazamiento del *display*.

0, movimiento del cursor.
R / L : desplaza el *display*. 1, hacia la derecha. 0, hacia la izquierda.
DL : ancho de la interface. 1, 8-*bits*. 0, 4-*bits*.
N : líneas del *display*. 1, dos líneas. 0, una línea.
F : puntos de los caracteres. 1, 5x10 puntos. 0, 5x7 puntos.
BF : 1, operación interna en proceso. 0, instrucción aceptada.
DDRAM : RAM de datos del *display*.
CGRAM : RAM del generador de caracteres.
ACG : dirección del CGRAM
ADD : dirección del DDRAM.
AC : contador de direcciones.

Después de inicializar el display, se realiza el programa de interface, el cual permite que se visualicen los mensajes, datos y resultados que se necesiten o se quieran presentar durante la aplicación. En el siguiente ejemplo se presenta la interface de 8-bits PIC - display, en la que se indica la forma de inicializar el display y de visualizar un mensaje.

```
; En este programa se emplean tres subrutinas de gran ayuda para el envío
; y ejecución de las instrucciones que manejan el display: LLEVAR, BUSY
; y SEND_CHAR.

STATUS          equ     3
RP0             equ     5
PORTD           equ     8       ; el Puerto D maneja el bus de datos
TRISD           equ     88
PORTE           equ     9       ; el Puerto E maneja el bus de control.
TRISE           equ     89
E               equ     0       ; bits del Puerto E
RS              equ     1
R_W             equ     2

CHAR            equ     21      ; registros auxiliares
```

```
; inicialización y configuración del display

DISPLAY_INIT    movlw   b'00111000'   ; Función SET
                call    LLEVAR
                movlw   b'00001000'   ; Display OFF
                call    LLEVAR
                movlw   b'00001100'   ; Display ON
                call    LLEVAR
                movlw   b'00000110'   ; Modo entry set
                call    LLEVAR
                movlw   b'00000001'   ; Limpiar display
                call    LLEVAR
                movlw   b'10000001'   ; DDRAM a la posición 2
                call    LLEVAR

; se va a enviar el mensaje:  'H', 'O', 'L', 'A'

                movlw   'H'
                call    SEND_CHAR
                movlw   'O'
                call    SEND_CHAR       ; los demás mensajes se
                movlw   'L'
                call    SEND_CHAR       ; envían de igual forma
                movlw   'A'
                call    SEND_CHAR
```

```
BUSY    bsf     STATUS,RP0  ; esta subrutina se encarga
        movlw   b'11111111' ; de verificar si el caracter o
        movwf   TRISD       ; instrucción ya se ha recibido
        bcf     STATUS,RP0  ; y procesado en el módulo del
        bcf     PORTE,RS    ; display.
        bsf     PORTE,R_W
        bsf     PORTE,E
        bcf     PORTE,E
        movf    PORTD,w
        movwf   TEMP
        btfsc   TEMP,7
```

LECCIÓN 4: SERVICIO DE INTERRUPCIÓN.

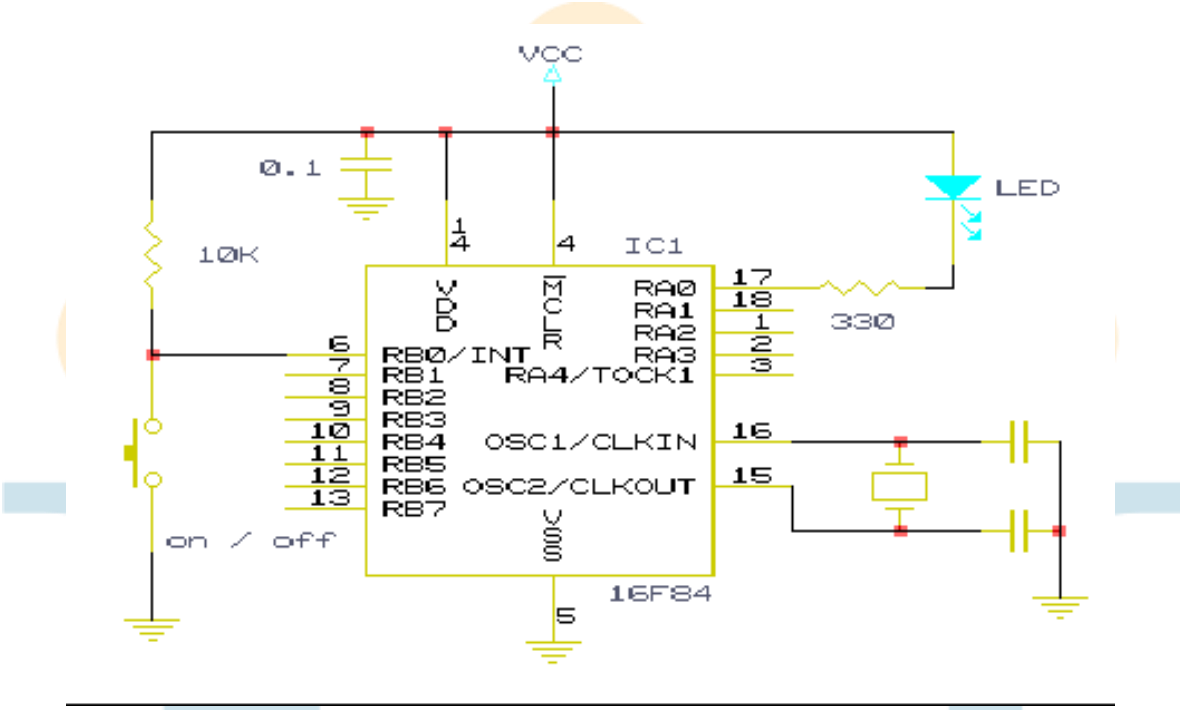
Servicio de interrupción Objetivos:

- Encender y apagar un LED como respuesta a un estímulo de interrupción
- Determinar la forma en que debe colocarse el código en el programa fuente para aceptar y atender una rutina de interrupción

Procedimiento:

Para ejemplificar el uso de un servicio de interrupción se ha dispuesto el hardware de la siguiente figura se decide utilizar la interrupción externa **INT** (**RB0**) en un PIC16F84, esta interrupción está vectorizada a la dirección de memoria de programa 004h, dentro de la atención a esta interrupción se opta por complementar el estado del LED colocado al puerto RA0 cada vez que esta sea atendida.

Diagrama electrónico:



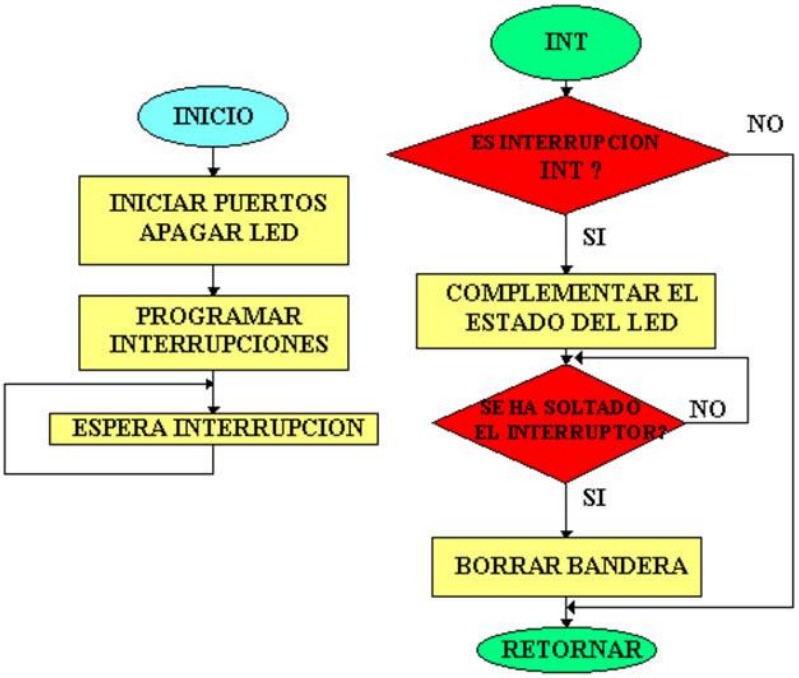
Listado del programa en assembler:

;Programa para realizar el apagado y encendido de un LED colocado en el ;Puerto A0 basado en la interrupción externa INT (RB0)

```
;
list p=16F84
status   equ   0x03
porta    equ   0x05
portb    equ   0x06
intcon   equ   0x0B
optionr  equ   0x81
trisa    equ   0x85
trisb    equ   0x86
#define  LED   porta,0
#define  BANK1 bsf status,5
#define  BANK0 bcf status,5
        org   000h    ;Indica al ensamblador la dirección de memoria de

                ;la sig. instrucción
        goto  Inicio
        org   004h    ;Indica al ensamblador la dirección de memoria de
                ;la sig. instrucción
Interrupcion
```

Diagrama de flujo:




```
    btfss intcon,1    ;es interrupción INT?
    retfie            ;retorna de interrupción y GIE=1
    btfsc porta,0     ;probar estado actual del LED
    goto  Prender     ;va a encender el LED
Apagar
    bsf   porta,0     ;apaga el LED
    goto  Espera
Prender
   bcf    porta,0     ;enciende el LED
Espera
   btfss  portb,0     ;espera a que se suelte el pulsador
   goto   Espera
   bcf    intcon,1    ;borra bandera INT
   retfie             ;retorna de interrupción y GIE=1
;Programa principal
Inicio     BANK1              ;selección del banco 1

   bcf    trisa,0     ;selecciona porta,0 como salida
   BANK0              ;selección de banco 0
   bsf    porta,0     ;apagar LED
;programación de interrupción
   bsf    intcon,4    ;activar interrupción INT     BANK1              ;selección banco 1
   bcf    optionr,6   ;selección del flanco de bajada en el pin INT
   BANK0
   bsf    intcon,7    ;Habilitar interrupciones globales
   goto   $           ;queda a la espera de interrupción
   end
```

El símbolo $ significa la dirección de memoria de programa en donde se encuentra éste (ciclo Infinito de espera)

Debe notarse la ubicación de la rutina de interrupción a partir de la posición de memoria de programa 004h.

LECCIÓN 5: COMUNICACIÓN SERIAL.

Objetivos:

- Verificar la comunicación serial síncrona y asíncrona
- Comprobar los algoritmos de comunicación serial

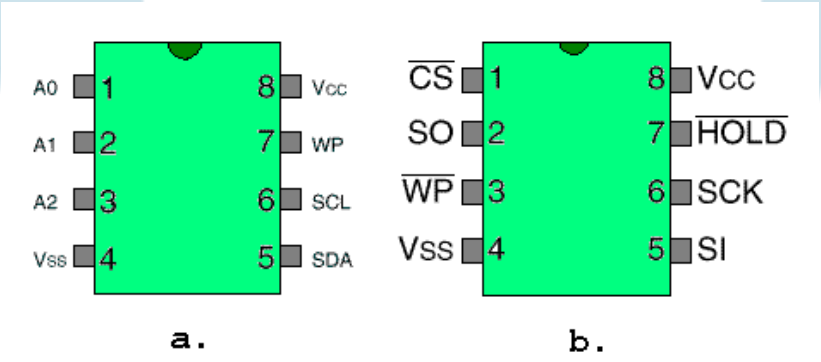
En vista de que algunos de los elementos de la familia PIC16CXXX no poseen periféricos de comunicación serial, este capítulo hará referencia al desarrollo del algoritmo como tal simulando los pines de comunicación serial con puertos del microcontrolador.

Comunicación serial síncrona: La comunicación síncrona se caracteriza porque los pulsos de sincronización deben ser transmitidos a lo largo de la línea de comunicación entre el transmisor y el receptor. Dentro de los varios tipos de comunicación serial síncrona vamos a notar el protocolo I^2C ó de dos hilos y el protocolo SPI ó de tres hilos.

Nomenclatura de los pines de comunicación (síncronos)

	Línea(s) de datos	**Línea de reloj**
I^2C	SDA (serial data)	SCL
SPI	SO (serial out), SI (serial in)	SCK

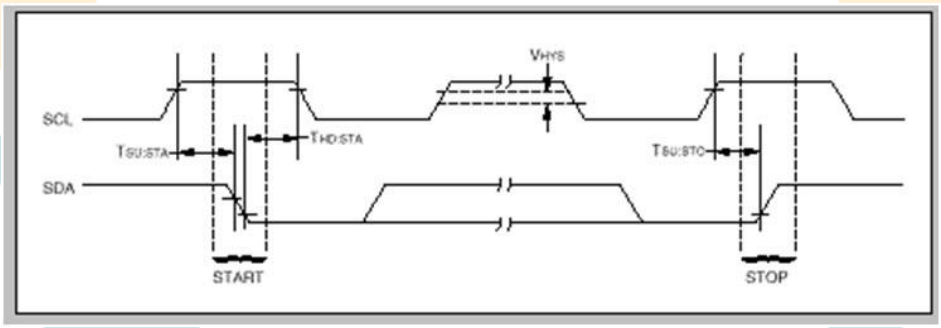
Figura 103. ***Disposición típica de pines en dispositivos síncronos (a) I^2C, (b) SPI***



I^2C

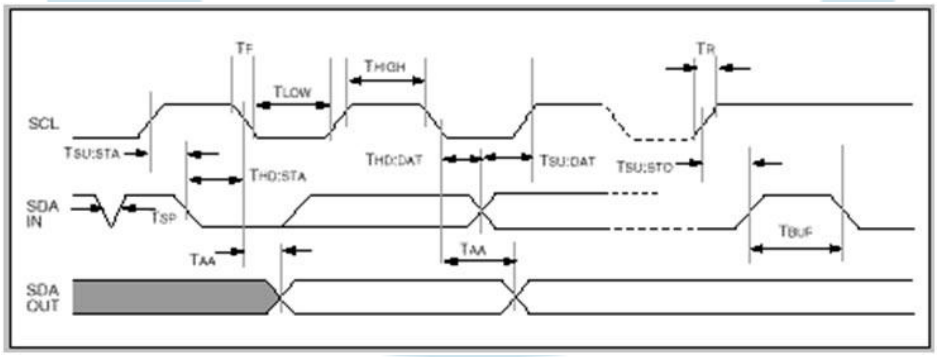
El bus I^2C es un bus diseñado para que sobre éste puedan colocarse varios dispositivos dentro de la misma tarjeta electrónica (comunicación multipunto), cada dispositivo tendrá una dirección lógica asignada físicamente mediante los pines A0, A1 y A2 de acuerdo al nivel lógico al que estos sean alambrados. ver estos pines en la figura **4.7.1 a**.

Figura 104.*Bits de START y STOP del protocolo I^2C*



En las figuras anteriores se observa la forma en que las señales SCL y SDA deben ser manejadas. Para iniciar la comunicación sobre un dispositivo I^2C debe realizarse la secuencia denominada bit de START que consiste en pasar la línea de datos SDA de nivel alto a bajo mientras que la línea SCL permanece en alto. Para la culminar la comunicación con el dispositivo I^2C debe ejecutarse la secuencia denominada bit de STOP la cual consiste en pasar la línea de datos SDA de nivel bajo a alto mientras que la línea de reloj SCL permanece en alto. Un bit de datos es aceptado por el dispositivo mientras que sobre la línea de datos SDA permanece el nivel adecuado al bit en cuestión, y sobre la línea de reloj SCL se lleva a cabo un pulso, es decir, el paso de nivel de bajo a alto y luego de alto a bajo. Los tiempos implicados en esta secuencia dependen básicamente del fabricante del dispositivo.

Figura 105. *Temporización en el bus I^2C*



SPI

El bus SPI es un bus diseñado para que sobre éste se coloque un dispositivo maestro y un dispositivo esclavo (comunicación punto a punto. Con relación al bus I2C podemos notar que éste soporta mayor velocidad de comunicación.

Figura 106. ***Entrada de datos a dispositivo SPI***

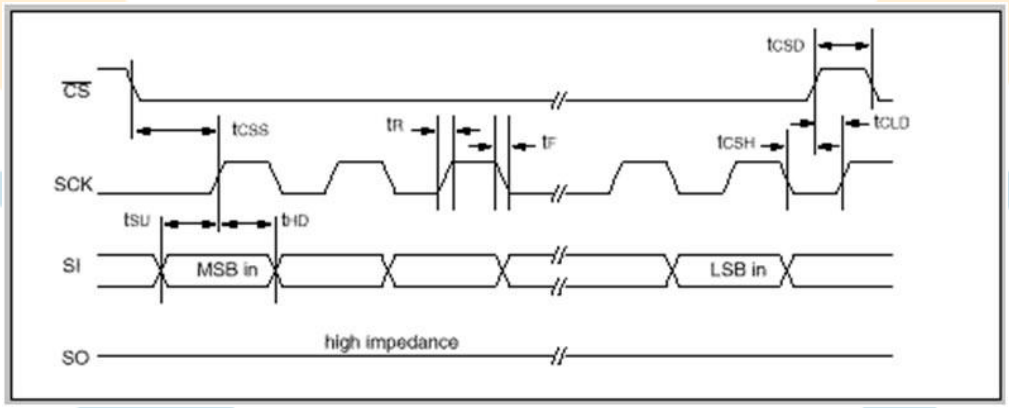
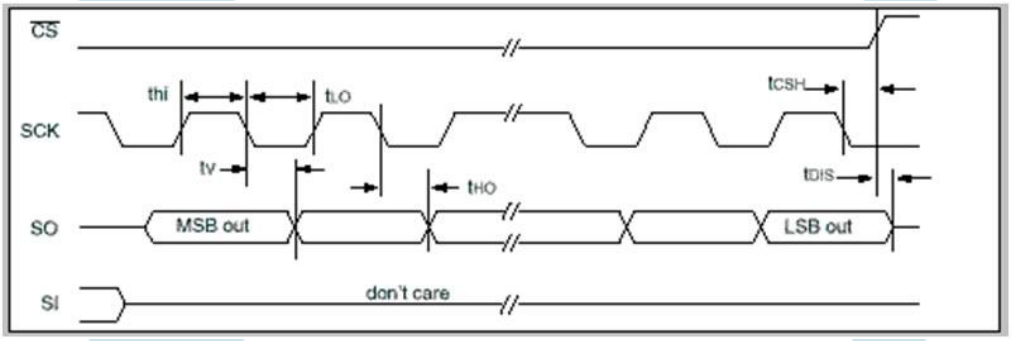
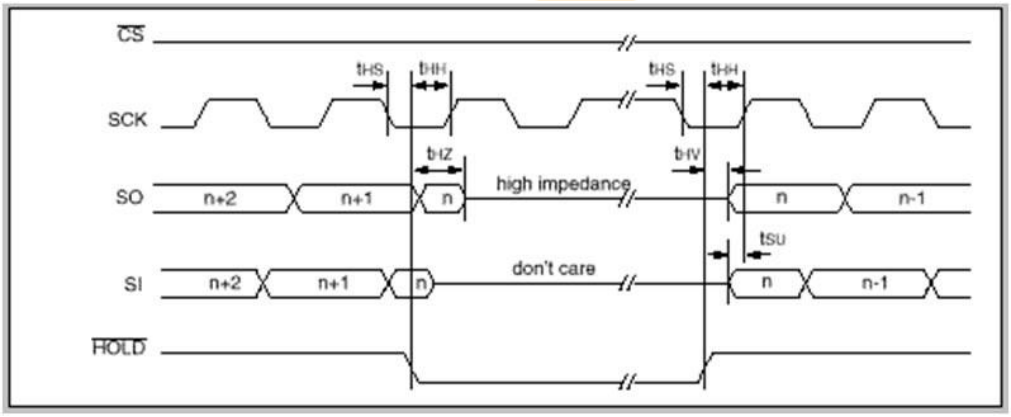


Figura 107. ***Salida de datos de dispositivo SPI***



El dispositivo SPI posee como una línea de selección CS la cual debe pasar al nivel lógico activo (en este caso bajo) para poder realizar la comunicación con el dispositivo. Desde este punto de vista podríamos colocar sobre un bus de este tipo varios dispositivos, pero utilizando un dispositivo decodificador adicional. Otra línea podemos observar es la línea HOLD la cual permite al procesador detener momentáneamente la comunicación.

Figura 108. ***Mapa de tiempos***



ACTIVIDADES DE AUTOEVALUACIÓN DE LA UNIDAD

Teniendo como precedente los contenidos de las unidades anteriores se proponen las siguientes actividades:

1. Con referencia a los ejemplos planteados en el capitulo 3 de la unidad 2, realice un recorrido por “los pasos en el método de trabajo para el desarrollo de aplicaciones”, identifique cada paso y justifíquelo en su grupo colaborativo.
2. Profundice e investigue respecto a las extensiones de archivos de ensamblador mas usuales, en foro grupal genere su propia explicación para los términos más usuales en microcontroladores.
3. Con respecto a los modos de direccionamiento, seleccione varias instrucciones dentro del set de instrucciones de microchip o Motorola y determine el modo de direccionamiento que utiliza, comparta esta experiencia con el grupo.
4. Tome el microcontrolador PIC16F84 o JK3, buque el datasheet o la documentación de cada dispositivo, determine sus funciones, elementos, arquitectura y puertos que dispone.
5. Descargue el MPLABIDE de la página oficial de Microchip o una versión más liviana y proceda a editar y generar el código fuente, identificando las partes que lo constituyen.
6. Teniendo como precedente y soporte los temas tratados en las lecciones del capitulo 2 de la unidad 3, implemente los programas de manejo de puertos, display 7 segmentos, exploración de teclado y display LCD.
7. Implemente en protoboard los circuitos anteriores y pruebe su funcionamiento.
8. Desarrolle el ejercicio de atención a interrupción y modifíquelo para manejar una secuencia de más de cuatro LEDs.
9. Diseñe e implemente un programa que responda al control de motor paso a paso con variación de giro y secuencia de pasos. Las secuencias de pasos se refieren a las “bobinas” que son activadas, esta se dividen en:

a. Pasos simples cuando se energiza una “bobina” a la vez.
b. Pasos dobles cuando se energiza dos “bobinas” a la vez, con un corrimiento de paso.
c. Medios pasos cuando se combinan las dos anteriores.
Referencia:
http://www.todorobot.com.ar/informacion/tutorial%20stepper/stepper-tutorial.htm

10. Diseñe e implemente un proyecto simple que involucre la mayoría de conceptos vistos en el curso.

BIBLIOGRAFIA

Vesga, Ferreira Juan Carlos. (2007). Microcontroladores Motorola – Freescale: Programación, familias y sus distintas aplicaciones en la industria.

CEKIT. (2002). Curso Práctico de Microcontroladores: Teoría, Programación, Diseño, Prácticas Proyectos completos. Editorial Cekit. Pereira-Colombia.

González, Vásquez José Adolfo. (1992). Introducción a los microcontroladores: hardware, software y aplicaciones. Editorial McGraw-Hill.

Téllez, Acuña Freddy Reynaldo. (2007). Modulo de Microprocesadores y Microcontroladores. UNAD.

Angulo. (n.d). Microcontroladores PIC, la solución en un chip. Sección 5.1

Valdivia, Miranda Carlos. (n. d). Arquitectura de equipos y sistemas informáticos. Editorial Paraninfo.

Angulo, Usategui José María. (n. d). Microcontroladores PIC. Diseño practico de aplicaciones.

SOFTWARE LIBRE

Software, utilidades y varias recursos adicionales son suministrados dentro del aula de curso en el “Herramientas y Sistemas de desarrollo”, algunas de ellas son:

MASM32 DSK version 10: http://www.masm32.com/

SimuProc14, simulador hipotético de un microprocesador x86:

http://gratis.portalprogramas.com/SimuProc.html

MPLABIDE: www.microchip.com

PicDevelopment Studio:

http://sourceforge.net/projects/picdev/files/picdev/PicDevelopmentStudio-1.1.exe/download

ICPROG: www.ic-prog.com/download.html

PROTEUS: http://www.ieeproteus.com/descarga.html

RECURSOS AUDIOVISUALES

Videos, animaciones y Objetos Virtuales de Aprendizaje, localizados en los recurso “Salón Virtual (videos, tutoriales, animaciones)” y “Objetos Virtuales de Aprendizaje - OVAs” localizados en el aula del curso virtual, uno de los vínculos externos es:

Video tutoriales sobre PIC: http://tutorialesvirtuales.com/

GLOSARIO DE TÉRMINOS

ACUMULADOR: Registro de la unidad aritmética y lógica del ordenador que almacena los resultados intermedios.

ADC: Acrónimo de *Analog to Digital Converter* (Conversor análogo a digital)

ADMINISTRADOR DE MEMORIA EXTENDIDA: Programa que impide que distintas aplicaciones utilicen la misma área de memoria extendida al mismo tiempo.

ALGORITMO: Procedimiento o conjunto de procedimientos que describen una asociación de datos lógicos destinados a la resolución de un problema. Los algoritmos permiten automatizar tareas.

ANCHO DE BANDA: Es la cantidad de información, normalmente expresada en bits por segundo, que puede transmitirse en una conexión durante la unidad de tiempo elegida. Es también conocido por su denominación inglesa: *bandwith*.

ARCHIVO DE SISTEMA: Archivo que contiene la información necesaria para ejecutar el sistema operativo.

AREA DE MEMORIA ALTA: (HMA) Los primeros 64 KB de memoria extendida. Esta área es utilizada por algunas aplicaciones, entre ellas Windows.

AREA DE MEMORIA SUPERIOR: Área de 384 KB adyacentes a los 640 KB de memoria convencional. Esta área se reserva usualmente para el funcionamiento de los componentes de hardware del sistema, tal como el monitor, y no se

considera parte de la memoria total, ya que las aplicaciones no pueden almacenar información en ella.

ASCII: Siglas de *American Standar Code for Information Exchange*. Se trata de un código casi universal para caracteres, números y símbolos, asignándole un número entre el 0 y el 255 a cada uno de ellos, como por ejemplo, el 65 para la letra A.

ASYNC: Acrónimo de *Asyncronous* (Asíncrono). Es el tipo de comunicación por el cual los datos se pasan entre dispositivos de forma asíncrona o sea que la transmisión de un caracter es independiente del resto de los demás caracteres.

BANDERA: Indicador interior del programa que da información sobre una condición determinada.

BAUDIO: Unidad que mide la velocidad de transmisión de los datos. Normalmente representa el número de bits por segundo.

BIT DE INICIO: Primer bit en un conjunto de datos que indica que los siguientes son datos.

BIT DE PARADA: Último bit en un conjunto de datos que indica que los anteriores son datos.

BIT DE PARIDAD: Se trata del método más elemental de detección de errores. Consiste en un único bit que indica si el número de bits enviados es par o impar.

BITS POR SEGUNDO: Se abrevia usualmente como 'bps'. Es el número de bits de datos enviados por segundo y es la auténtica velocidad de transmisión.

BLOQUES DE MEMORIA SUPERIOR: (UMOS) Zonas no utilizadas del área de memoria superior. Si se dispone de una computadora con un procesador 80386 o 80486, es posible copiar en los bloques de memoria superior información procedente de otros tipos de memoria, liberando así más memoria convencional (los primeros 640 KB).

BUCLE: Conjunto de instrucciones que se repiten varias veces seguidas.

BUFFER: Área de almacenamiento temporal de información.

BUS: (Línea) Cableado utilizado para transmitir un conjunto de señales de información entre dispositivos de un computador. De su amplitud (expresada en bits simultáneos) depende su velocidad.

CACHÉ DE DISCO: Una porción de la memoria reservada para almacenar temporalmente información leída en un disco.

CAMPO: Colección de caracteres que forman un grupo distinto, como un código de identificación, un nombre o una fecha. Generalmente un campo forma parte de una información.

CHECKSUM: (Suma de comprobación). Es el más elemental de los controles de errores. Consiste normalmente en un único byte obtenido mediante una o varias operaciones matemáticas realizadas sobre todos los bytes de un bloque de datos con el fin de controlar los posibles errores que se produzcan durante una transmisión.

COMPILADOR: Programa que pasa otro programa escrito en un lenguaje de alto nivel (parecido al humano) al lenguaje de la máquina de modo que ésta lo entienda perfectamente.

COMUNICACIÓN SíNCRONA: Es el tipo de comunicación por el cual los datos se pasan entre dispositivos de forma síncrona o sea que la transmisión depende de la meticulosa sincronización de los datos transmitidos, enviados y de sus propios mecanismos de transmisión. No requiere ni bit de inicio ni bit de parada, a diferencia de la comunicación asíncrona.

CPU: Acrónimo de *Central Processing Unit* (unidad central de procesamiento) Es el procesador central del computador encargado de controlar rutinas, realizar funciones aritméticas, y otras tareas propias. Cada vez se le suele descargar de más tareas, consiguiendo de esta forma un mayor rendimiento.

DAC: Acrónimo de *Digital to Analog Converter* (Conversor de digital a analógico)

DEPURAR: (*debug*). Eliminar los errores de un programa.

DIAGRAMA DE FLUJO (FLOW-CHART): Trazado de la escritura y curso de un programa en el que se utilizan formas diferentes, como un rectángulo o un cuadrado para indicar una acción del ordenador, y un rombo para las decisiones tomadas por éste. Normalmente, se suele hacer el gráfico o diagrama del programa antes de introducir una sola línea de éste en el computador.

DIRECCIÓN: Número que se refiere a la posición, generalmente en la memoria del computador, en la que se almacena la información.

DIRECCIONES DE E/S: (Entrada /Salida) Posiciones dentro del espacio de dirección de entrada/salida de la computadora que son utilizadas por un dispositivo, como puede ser una impresora o un módem. La dirección es utilizada para la comunicación entre el software y el dispositivo.

DMA: Acrónimo de *Direct Memory Access* (Acceso directo a memoria).

DOBLE PALABRA: Agrupación de 32 bits consecutivos equivalente a cuatro bytes.

E/S: Acrónimo de 'Entrada/Salida'. Suele aplicarse al flujo de datos. También es conocido por su acrónimo inglés 'I/O'.

EMULADOR DE MEMORIA EXPANDIDA: Utilidad que convierte memoria extendida en memoria expandida.

ENSAMBLADOR (ASSEMBLER): Es un programa que convierte otro programa escrito en lenguaje de ensamble (*assembly language*) en un código que el microprocesador puede ejecutar directamente.

ENTRADA (INPUT): Toda la información que se introduce en un programa durante su ejecución.

FIRMWARE: Son los programas que funcionan dentro del computador, relacionados íntimamente con el hardware. El firmware puede alterarse, hasta cierto punto, por medio del software.

I/O: Acrónimo de Input/Output (Entrada/Salida). Suele aplicarse al flujo de datos. También es conocido por su acrónimo castellano 'E/S' (Entrada/Salida).

INSTRUCCIÓN: Elemento del código de programación que le dice al computador que lleve a cabo una tarea determinada. Una instrucción en lenguaje de ensamble, por ejemplo, podría ser ADD, con lo que le dice al ordenador que realice una suma.

INTEL: Uno de los mayores fabricantes de procesadores, chips y circuitos integrados del mundo, de nacionalidad estadounidense. Sus 'CPUs' más conocidas son: 8086, 8088, 80286, 80386, 80486, Pentium y su co-procesador matemático: 80387.

INTERFAZ: Conexión mecánica o eléctrica que permite el intercambio de información entre dos dispositivos o sistemas. Habitualmente se refiere al *'software'* y *'hardware'* necesarios para unir dos elementos de proceso en un sistema o bien para describir los estándares recomendados para realizar dichas interconexiones. Es más conocido por su denominación inglesa: 'interface'.

INTERRUPCIÓN: Señal que un dispositivo envía a la computadora cuando dicho dispositivo, por ejemplo, no está listo para aceptar o enviar información.

IRQ: Acrónimo de *Interrupt ReQuest* (Petición de interrupción). Señal que envía un dispositivo para ser atendido por el programa encargado de gestionarlo. En un PC (no inferior a un 80286) existen 15 IRQs de las cuales sólo los números 10, 11, 12 y 15 están realmente libres ya que las otras las ocupa el propio sistema. Conocido simplemente como: interrupción.

K: Abreviatura de kilobyte; 1 K= 1.024 bytes .

KB/S: Acrónimo de Kilobytes per second (Kilo-bytes por segundo)

KBIT/S: Acrónimo de Kilobits per second (Kilobits por segundo)

LENGUAJE DE ALTO NIVEL: Lenguaje de programación que utiliza instrucciones escritas con palabras comunes.

LENGUAJE DE BAJO NIVEL: Lenguaje parecido al utilizado en el computador.

LENGUAJE DE MÁQUINA: Es el escalón inferior al lenguaje de bajo nivel; es el lenguaje que el ordenador entiende directamente .

LÍNEAS DE INTERRUPCIÓN REQUERIDA: (IRO) Líneas del hardware sobre las cuales los dispositivos pueden enviar señales de interrupción, las que indican si el dispositivo está listo para aceptar o enviar información. Por lo general, cada uno de los dispositivos conectados a la computadora utiliza una línea IRQ independiente.

MAINFRAME: Computador muy potente al que se conectan una o varias estaciones de trabajo.

MB/S: Acrónimo de Megabytes per second (Mega-bytes por segundo]

MBIT/S: Acrónimo de Megabits per second (Megabits por segundo)

MEGABYTE: Equivale a: 1024 Kilo-bytes ó 1.048.576 bytes (2 elevado a la 20).

MEMORIA CONVENCIONAL: Primeros 640 KB de memoria que utiliza el MS-DOS para ejecutar aplicaciones.

MEMORIA EXPANDIDA: Memoria que utilizan algunas aplicaciones No-Windows además de la memoria convencional. La memoria expandida es un estándar antiguo que se está sustituyendo hoy por la memoria extendida. Sólo el software compatible con EMS puede utilizar memoria expandida. Cuando Windows se ejecuta en el modo extendido del 386, simula memoria expandida para aquellas aplicaciones que la necesiten. También denominada Memoria EMS.

MEMORIA EXTENDIDA: Memoria más allá de un megabyte (MB) en las computadoras basadas en los procesadores 80286, 80386 y 80486. Windows utiliza memoria extendida para administrar y ejecutar aplicaciones. Por lo general, la memoria extendida no podrá ser utilizada por las aplicaciones No-Windows ni por MS-DOS.

MEMORIA VIRTUAL: Sistema de administración de memoria utilizado por Windows en el modo extendido del 386, que permite ejecutar Windows como si hubiese más memoria de la que existe en realidad. La cantidad de memoria virtual disponible será igual a la cantidad de memoria RAM libre sumada al volumen de espacio de disco asignado a un archivo de intercambio que Windows utiliza para simular memoria RAM adicional.
MEMORIA VOLATIL: Un mecanismo de memoria que pierde sus contenidos cuando se le corta la energía eléctrica.

MICRO: Abreviatura de microcomputador.

MICROCHIP: Diminuto circuito de silicio que funciona como memoria, procesador, etc .

MICROPROCESADOR: El chip que hace de corazón del computador o de su parte pensante.

MODO EXTENDIDO DEL 386: Modo en el que se ejecuta Windows para tener acceso a las capacidades de memoria virtual del procesador Intel 80386. En este modo, Windows parece utilizar más memoria de la disponible físicamente y puede trabajar en el modo de multitarea con aplicaciones No-Windows.

MODO PROTEGIDO: Modo de funcionamiento de la computadora que permite tener acceso directamente a la memoria extendida.

MSB: Acrónimo de *Most Significant Bit* (Bit más significativo)

MULTITAREA: Capacidad que tiene una computadora de ejecutar varias aplicaciones al mismo tiempo.

MULTIUSUARIO: Capacidad de permitir el acceso a varios usuarios simultáneos a un determinado programa, aplicación, sistema o servicio telemático en línea sin destruir la integridad de mismo.

NIBBLE: Agrupación de 4 bits consecutivos equivalente a medio byte.

PALABRA: *Word*. Agrupación de 16 bits consecutivos equivalente a dos bytes.

PAQUETE: Conjunto de caracteres enviados conjuntamente durante una comunicación. Los bloques más comunes suelen ser de 64, 128 ó 1024.

PC: Acrónimo de *Personal Computer* (Computadora personal) Computador presentado por 'IBM' el 12 de agosto de 1981 en EE.UU. y comercializado en 1982. Fue el primer ordenador que se vendió masivamente en todo el mundo siendo su denominación original: 'IBM PC'. Con la idea de que el ordenador debía transportarse, surgió su primer competidor: 'Compaq' que lanzó en 1982 su propio portátil. El despegue real del 'PC' llegó en 1983 cuando 'IBM' anunció un auténtico estándar: el 'PC XT' siendo posteriormente culminado por el 'PC AT'.

PERIFÉRICO: Dispositivo externo o interno que se conecta al computador.

POR DEFECTO: Dícese del disco, el programa, etc. que entra en funcionamiento cuando no se da ninguna otra instrucción.

PUERTO: Canal o interfaz que une un periférico a un microprocesador. Puede ser serie (envía los datos bit a bit) o paralelo (envía los datos byte a byte).

REGISTRO: Parte de la memoria del computador que contiene datos de uso frecuente .

SALIDA (OUTPUT): Todos los datos producidos por el computador mientras está procesando, ya aparezcan estos datos en la pantalla, ya los entregue a la impresora o los utilice internamente.

SERVIDOR: Computadora que suministra espacio de disco, impresoras u otros servicios a computadoras conectadas con ella a través de una red.

TERMINAL: Dispositivo de entrada/salida de datos. El terminal "tonto" (a veces denominado 'TTY') maneja simplemente dicha entrada/salida mientras que el "inteligente" utiliza un microprocesador para realizar esa tarea con capacidades adicionales no disponibles en el anterior.

TIEMPO DE ESPERA: Cantidad de tiempo que deberá esperar la computadora cuando un dispositivo no ejecute una determinada tarea, antes de presentar un mensaje de error.

TIEMPO REAL: Modalidad de funcionamiento de un sistema de proceso de datos que controla una actividad en curso, con un tiempo de respuesta prácticamente nulo a la recepción de las señales de entrada.

UNIDAD: Dispositivo de un computador que acepta discos o cintas en los que lee o graba datos.

UNIDAD CENTRAL DE PROCESAMIENTO: Es el corazón del computador, en donde se llevan a cabo las funciones aritméticas, lógicas, de procesamiento y de control.

VLSI: Acrónimo de *Very Large Scale Integration* (Integración a muy gran escala) Este tipo de integración permite diseñar circuitos cada vez más reducidos en tamaño lo que, en la práctica, hace que nuestros computadores, módems, etc sean cada vez más diminutos, aumentando paradójicamente sus prestaciones.

FUENTES DOCUMENTALES

DOCUMENTOS IMPRESOS

Ureña, López Alfonso, Sanchez, Solano Antonio Miguel, Martin, Valdivia María Teresa & MANTAS, Ruiz Jose Miguel. (1999). Fundamentos de informática. Editorial Alfaomega & ra-ma. Santafé de Bogotá.

Rojas, Ponce Alberto. (1997). “Ensamblador Básico”. Editorial Computec. AlfaOmega Santafé de Bogotá.

CEKIT. (2002). Curso Práctico de Microcontroladores: Teoría, Programación, Diseño, Prácticas Proyectos completos. Editorial Cekit. Pereira-Colombia.

Téllez, Acuña Freddy Reynaldo. (2007). Modulo de Microprocesadores y Microcontroladores. UNAD.

Vesga, Ferreira Juan Carlos. (2007). Microcontroladores Motorola – Freescale: Programación, familias y sus distintas aplicaciones en la industria.

González, Vásquez José Adolfo. (1992). Introducción a los microcontroladores: hardware, software y aplicaciones. Editorial McGraw-Hill.

Angulo. (n.d). Microcontroladores PIC, la solución en un chip. Sección 5.1

Valdivia, Miranda Carlos. (n. d). Arquitectura de equipos y sistemas informáticos. Editorial Paraninfo.

Angulo, Usategui José María. (n. d). Microcontroladores PIC. Diseño practico de aplicaciones.

Stallings, William. "Organización y Arquitectura de Computadores". (5ª edición). Editorial Prentice-Hall. Madrid, 2000.

Tokheim, Roger. "Fundamentos de los Microprocesadores". (2ª edición). Editorial Mc Graw Hill. México, 1985.

Uruñuela, José Mª. "Microprocesadores: Programación e Interconexión". (2ª edición). Editorial Mc Graw Hill. España, 1995.

DIRECCIONES DE SITIOS WEB

Teoría de computadores. Extraído el 10 de Julio de 2009 desde

http://www.computacion.geozona.net/teoria.html

Dispositivos lógicos microprogramables, extraído el 10 de Julio de 2009 desde

http://perso.wanadoo.es/pictob/indicemicroprg.htm

Curso de Microcontroladores Motorola, extraído el 10 de Julio de 2009 desde http://www.geocities.com/moto_hc08/index.html