UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVISION DE ESTUDIOS DE POST-GRADO

DESARROLLO DE UN CURSO DE MICROCONTROLADORES QUE INCLUYE: EL PLAN DE ESTUDIOS, APUNTES Y LISTA DE PRACTICAS DE LABORATORIO

TESIS

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS
DE LA INGENIERIA ELECTRICA
CON ESPECIALIDAD EN ELECTRONICA

QUE PRESENTA

RENE BRIONES LARA

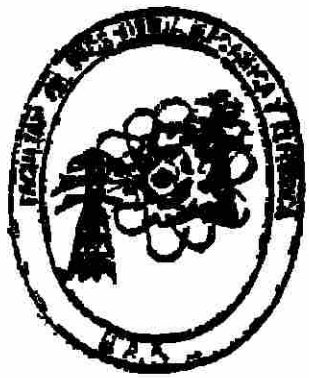
SAN NICOLAS DE LOS GARZA, N. L. SEPTIEMBRE DE 1994

DESARROLLO DE UN CURSO DE MICROCONTROLADORES QUE INCLUYE: EL PLAN DE ESTUDIOS, APUNTES Y LISTA DE PRACTICAS DE LABORATORIO

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

DIVISION DE ESTUDIOS DE POST-GRADO



DESARROLLO DE UN CURSO DE MICROCONTROLADORES QUE INCLUYE. EL PLAN DE ESTUDIOS, APUNTES Y LISTA DE PRACTICAS DE LABORATORIO

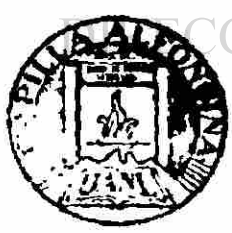
TESIS

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS
DE LA INGENIERIA ELECTRICA
CON ESPECIALIDAD EN ELECTRONICA

QUE PRESENTA

RENE BRIONES LARA

SAN NICOLAS DE LOS GARZA N. L. SEPTIEMBRE DE 1994



UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVISION DE ESTUDIOS DE POST-GRADO

Los miembros del comité de tesis recomendamos que la presente tesis realizada por el Ing. René Briones Lara sea aceptada como opción para obtener el Grado de Maestro en Ciencias de la Ingeniería Eléctrica con Especialidad en Electrónica.

El Comité de Tesis

M.C. Ronald López Gómez.
Asesor

M.C.Juan Diego Garza Gonzalez
Coasesor

M.C.Luis Manuel Camacho Velázquez
Coasesor

División de Estudios de Postgrado
M.C.Marco Antonio Méndez Cavazos

San Nicolás de los Garza, N.L. Septiembre de 1994

Agradecimiento

La tarea de escribir este curso resultó algunas veces agotadora, pero siempre fue instructiva y gratificante. Durante el desarrollo de esta actividad hubo que apartarse de amigos; pero el esfuerzo es compensado por los logros obtenidos.

Hablando de esfuerzos, quiero hacer patente mi profundo agradecimiento al **Ing. Ronald López** por su apoyo, sugerencias y tiempo que me brindó durante el desarrollo de esta tesis.

INDICE

I. PROLOGO

Hoy más que nunca en forma acelerada y sistemática, la educación asume un papel protagónico ante la globalización de la economía de mercado. El proceso de cambio que afecta al mundo, y que alcanza a todas las esferas de la actividad humana, promueve la incentividad del progreso técnico como expresión de una verdadera revolución científico-tecnológica, para adaptarlo a las tendencias que exige el desarrollo.

En el camino hacia la próxima centuria hay una palabra clave: **Globalización.** Si adherimos al principio de que el mundo se globaliza y que no vamos a poder ser una isla, entonces, ya no se trata de determinar como formar profesionistas en un país determinado, sino como formarlos para el mundo.

Apartir de ahora no vamos a poder ser tan originales: **nuestros profesionistas se tienen que parecer cada vez más a los de otros sitios.** Además, así como hay un proceso de movimiento de los productos, también hay un proceso de movimientos de los factores. Por eso, nuestros ingenieros deberán estar preparados para ser transferibles, es decir, que si no trabajan aquí, lo harán en alguna parte de la economía global. Y aunque no fuera así, igual debemos formarlos de ese modo. Porque para poder competir, nuestros profesionistas deberán de estar capacitados para construir productos con los estándares que se están usando en todo el mundo.

Con la finalidad de actualizar los conocimientos, y tomando como base las razones anteriormente expuestas, fundamento la justificación de elaborar mi tesis "Desarrollo de un Curso de Microcontroladores, que incluye: El plan de Estudios, Apuntes y un Listado de Prácticas de Laboratorio", el cuál espero sea utilizado en el Sistema Nacional de Institutos Tecnológicos .

II. SINTESIS

La presente tesis presenta un Curso de Microcontroladores desde un punto de vista totalizador, pues toma en cuenta tanto la preparación teórica como la práctica.

Aplicando una nueva **Metodología de la Enseñanza** que incorpora nuevos modelos en educación, es elaborado **El Programa de Estudios** para orientar las actividades y esfuerzos, para lograr los objetivos generales del curso, es decir:

- Aprender a programar microcontroladores,
- Aprender la forma de conectarlos adecuadamente a dispositivos de entrada o de salida.
- Realizar diseños de equipo electrónico usando como elemento básico un microcontrolador.

Los Apuntes de Microcontroladores toman como base principalmente el Microcontrolador MC68HC11A8. Los apuntes son desarrollados como fundamentación teórica, y en él se presentan:

- Los conceptos generales de microcontroladores estableciendo las diferencias con los microprocesadores.
- La arquitectura del hardware de un microcontrolador MC68HC11A8.
- El Conjunto de Instrucciones y sus Modos de Direccionamiento.
- Las técnicas y herramientas utilizadas como Sistema de Desarrollo de Hardware.
- La utilización de los Puertos para comunicación Paralelo en modo directo y con protocolo de comunicación.
- La descripción y uso del convertidor Análogo-Digital.

El Listado de Prácticas de Laboratorio sirve de guía para la capacitación en el uso de microcontroladores . El desarrollo de las mismas proporciona la experiencia en la manipulación del conjunto de instrucciones y del soporte de hardware, además, mediante el uso de la computadora, se adquiere la habilidad en el manejo del Programa Ensamblador y del Programa Monitor, permitiendo al estudiante adquirir seguridad en la funcionalidad de los microcontroladores para poder emprender proyectos de mayor magnitud.

III. INTRODUCCION

Para preparar profesionistas cuya función social y desempeño, contribuyan al desarrollo y mejoramiento de las condiciones de vida en el país, la Escuela de Graduados tiene como una de sus metas: la formación de personal docente a Nivel Licenciatura. Esto impone la exigencia de revisar y elevar la calidad de la educación.

El mejoramiento de la calidad de la educación atañe entre otros, a las Escuelas de Graduados, sobre todo ante los cambios acelerados de la tecnología contemporánea, que con frecuencia propicia la obsolescencia de los conocimientos o técnicas aprendidas, así como el uso de cursos, apuntes y prácticas de laboratorio también anticuados. Esta tesis, pretende contribuir en ofrecer un Curso de Microcontroladores a nivel licenciatura, para ser utilizado principalmente en el Sistema Nacional de Institutos Tecnológicos, e incluye como factores:

A. Un adecuado enfoque en el Programa de Estudios.
B. La información teórica necesaria en unos Apuntes de Microcontroladores.
C. La capacitación práctica mediante Prácticas de Laboratorio.

Sostengo la idea de que el Programa de Estudios, los Apuntes y el Desarrollo de Prácticas de Laboratorio, son el medio en que convergen y se sintetizan los elementos de un modelo académico y que a partir de él se definen y orientan las interacciones que caracterizan la docencia.

Diseñar o rediseñar un Curso de Microcontroladores se concibe desde este punto de vista, como una actividad totalizadora que exige preparación y capacitación teórica y práctica o sea: aplicación de teorías, conceptos y estrategias de información pertinentes.

IV. METODOLOGIA DE LA ENSEÑANZA.

La Enseñanza Tradicional se ha ido transformando para dar paso a la llamada "Sistematización de la Enseñanza". Esta última tiene como característica, incorporar todas aquellas ideas que se han experimentado en educación al desarrollo de modelos en los que el funcionamiento de la tecnología educativa es óptima, es decir, modelos en los que la interacción entre los recursos humanos, materiales y técnicos, es fundamental para el logro de objetivos instruccionales. Además, el énfasis del proceso instruccional está centrado especialmente en la actividad del estudiante.

La Sistematización de la Enseñanza se manifiesta concretamente, en la elaboración de programas por objetivos, formulados de acuerdo al enfoque conductista, programas en los que el elemento central es el objetivo operacional. Es precisamente en este elemento del modelo académico que doy forma al proyecto que hoy me ocupa. Intento dar forma al Curso de Microcontroladores como resultado de este enfoque de la tarea educativa, del aprendizaje y de la enseñanza.

Se trata de que en la realización del proceso educacional programado, se obtenga un verdadero desarrollo personal de los estudiantes y maestros, a la vez que produzca aprendizajes significativos .

Este Curso de Microcontroladores es una propuesta de aprendizajes que guardan una relación directa y concreta entre: el programa de estudios, los apuntes y el desarrollo de prácticas de laboratorio. La validez de los aprendizajes propuestos por lo tanto, se realizan en función del valor que estos tienen para desarrollar la formación de esta disciplina en particular y en su contribución al cumplimiento del curso bajo estudio.

El Programa de Estudios es una guía fundamental, pues funciona como criterio de orientación para la selección y diseño de las actividades de aprendizaje. Además, es un elemento de comunicación entre docentes y alumnos, porque al ser un instrumento de trabajo, genera compromisos para ambos al orientar sus actividades y esfuerzos, al logro de los aprendizajes que se han fijado como necesarios a partir del objetivo general del curso.

Los Apuntes y las Prácticas de Laboratorio del Curso de Microcontroladores corresponden a la instrumentación didáctica, que junto con el programa de estudios, ocupan momentos diferentes pero complementarios y necesarios de un mismo proceso; es decir, si bien la instrumentación didáctica (apuntes y práctica de laboratorio) deben hacer acopio de la experiencia y posibilidades personales tomando como base fundamental el programa de estudios, pretendiendo, que aún con las diferencias individuales, todos los estudiantes logren los conocimientos propuestos.

De aquí que se planteen como **Elementos del Programa** los *siguientes* puntos:

1. DATOS GENERALES DE IDENTIFICACION DEL CURSO.

1.1. Nombre del Curso. Es la denominación oficial que aparecerá en la carrera.

1.2. Carrera. Se especifica la(s) carrera(s) a la(s) que va dirigida el curso.

1.3. Lugar y fecha. Se anotan los datos correspondientes, mes y año en que se elaboró el programa.

2. HISTORIA DEL PROGRAMA

2.1. Lugar y fecha de revisión.

2.2. Participantes.

2.3. Cambios y justificación. Se enuncian los materiales de los que parte para elaborar o revisar el programa. Así como las modificaciones que se realizan y su justificación. Este apartado es acumulativo y deben conservarse en él, todos los cambios y justificaciones que se le hagan al programa, desde su primera elaboración, hasta su última revisión para darse cuenta de su evolución.

3. UBICACION DE LA ASIGNATURA. En este apartado se especifica:

3.1. Las relaciones que a nivel de tema tiene esta asignatura con otras *anteriores y posteriores*.

3.2. Las aportaciones concretas de la asignatura al desempeño profesional.

4. OBJETIVOS GENERALES DEL CURSO. Esto corresponde a los enunciados de los conocimientos, así como a las habilidades y aptitudes globales, que concretizan los productos de aprendizajes que se esperan lograr al final del curso y delimitan su profundidad y alcance; orientando el desglose de las unidades de aprendizaje.

5. **TEMARIO.** Es la relación de temas y subtemas, que los contenidos con los que se trabajará para lograr los objetivos generales .

6. **APRENDIZAJES REQUERIDOS.** En este apartado se enlistan los aprendizajes de las asignaturas ya cursadas y que son necesarios para que el alumno tenga una mayor probabilidad de lograr los aprendizajes planteados en los objetivos del curso.

7. **UNIDAD DE APRENDIZAJE.** En estas se observa lo siguiente:
Su determinación se hace con criterios que toman en cuenta, tanto la estructura lógica de la ciencia (estructura conceptual) así como las capacidades y aprendizajes ya logrados por el estudiante en otros cursos. También se toma encuenta lo que se proyecta lograr en esta asignatura y que está delineado en los objetivos generales.

Las Unidades de Aprendizaje son un elemento resultante de la división del curso tomando en cuenta los criterios anteriores; debido a esto, su contenido temático puede constituirse de uno o más temas, o ser un conjunto de subtemas agrupados, cuando la unidad temática es muy compleja o extensa y que para fines de aprendizajes, se necesita y se posibilita su división.

La estructura lógica de la asignatura y los aprendizajes propuestos en los objetivos generales, dan lugar a que la última unidad haya sido diseñada de manera que permite evaluar en forma integra, los aprendizajes expuestos en el objetivo general. En la redacción de las Unidades de Aprendizaje se especifican los siguientes componentes:

7.1. Número progresivo que le corresponde en el programa
7.2. Contenido temático de la unidad.
7.3. Objetivo Educacional.
7.4. Aprendizajes Intermedios.
7.5. Bibliografía.

A continuación se detalla los tres últimos componentes.

OBJETIVO EDUCACIONAL. El aprendizaje logrado representa una modificación de las estructuras del conocimiento existente, originados por la asimilación y acomodación de nuevos conocimientos y experiencias; tales procesos internos se manifiestan con ejecuciones concretas, que el que aprende es capaz de realizar. A estas manifestaciones se les denomina "Productos de Aprendizajes", de los que son ejemplos los siguientes: resolver

problemas de cierto tipo, diseñar proyectos, elaborar trabajos bibliográficos, expresar argumentos, comparar teorías, etc.

Al formular los objetivos educacionales, se han relacionado estas ejecuciones o capacidades complejas con contenidos temáticos, para así obtener enunciados que señalen a maestros y alumnos el resultado esperado de cada unidad.

La complejidad, significado y relación de estos aprendizajes, es tomada en cuenta para que la redacción sea clara y bien definida, de tal manera, que funcionan como guía para diseñar los aprendizajes intermedios y permiten seleccionar actividades de enseñanza que propician el aprendizaje. Así mismo, funcionan como instrumentos y situaciones de evaluación válidos, para verificar si los objetivos han sido logrados por los estudiantes.

APRENDIZAJES INTERMEDIOS. En los objetivos educacionales se enuncian los aprendizajes complejos en términos de productos de aprendizajes, por lo que se hace necesario un desglose de estos en otros más sencillos, que son llamados Aprendizajes Intermedios. Estos se obtienen para dar respuesta a la siguiente pregunta: Que debe conocer y saber hacer el alumno para lograr el objetivo educacional? Se enlistan entonces estos componentes más sencillos en una secuencia y graduación para que el estudiante adquiera el aprendizaje completo. Así mismo, son enunciados claramente y son suficientes en cantidad para que con su combinación se logre el aprendizaje global.

BIBLIOGRAFIA. Aquí se especifican los materiales bibliográficos relacionados con los contenidos de la unidad y suelen ser libros, partes de libros, artículos de revistas, etc.

V. DEFINICION DEL CURSO.

El presente curso de Microcontroladores incluye: El Programa de Estudios, Apuntes y un Listado de Prácticas de Laboratorio. Está dirigido principalmente a estudiantes de nivel Licenciatura de Ingeniería Electrónica. Sin embargo, también puede ser utilizado por todos aquellos estudiantes, profesionistas y técnicos con conocimiento de electrónica y circuitos digitales, que tengan necesidad y deseos de conocer como funcionan y se aplican los microcontroladores.

En este curso se analizan las diferentes técnicas disponibles para el diseño de sistemas digitales en base a microcontroladores. El curso está basado principalmente en la familia de microcontroladores MC68HC11 de Motorola. Se debe de contar con un sistema de desarrollo de software consistente de un Procesador de Texto, Programas Ensambladores y Monitores residentes en equipo compatible IBM y hardware (micromódulos) para que el alumno realice sus prácticas y proyectos.

Al terminar el curso, el alumno será capaz de:

- Planear la solución a un problema digital en base a microcontroladores.
- Diseñar el hardware necesario tomando como base un modulo comercial y realizar adiciones a este.
- Diseñar el software utilizando ensambladores y sistemas de desarrollo. Integrar todo el conjunto, probarlo y corregirlo.

El curso está estructurado para un semestre realizando las siguientes actividades:

- Asistir 4 hs. de clase teórica por semana y estudiar el material.
- Asistir 2 hs. por semana al laboratorio para hacer una tarea práctica.
- Presentar cuando menos 2 examenes parciales teóricos y examen final de tipo práctico.

Dado lo anterior, el alumno debe dedicar 10 horas en promedio por semana, además de las horas de clases, para realizar actividades directamente relacionadas con el curso.

VI. CONCLUSIONES

El tratar de realizar un Curso de Microcontroladores es desde mi punto de vista, inherentemente mas difícil de desarrollar y planear, sobre todo si lo comparamos con otros cursos como: matemáticas, circuitos eléctricos, etc. Además es más difícil de elegir la secuencia de los temas.

El presente Curso de Microcontroladores representaba un reto que fue resuelto aplicando Tecnologías Educativas que involucran la interacción entre los recursos humanos, materiales y técnicos.

Considero que alcance con éxito el objetivo originalmente planteado de desarrollar un Curso de Microcontroladores, pues para lograr que el estudiante adquiera los conocimientos suficientes sobre microcontroladores, realicé como guía fundamental El Programa de Estudios , y desarrollé como instrumentación didáctica: los Apuntes y el Listado de Prácticas de Laboratorio.

Sin embargo, los apuntes no deben ser tomados como definitivos, pues la tecnología referente a microcontroladores está avanzando a pasos agigantados, lo cual hace necesario su revisión periódica, para así mantenerlos actualizados conforme avance la tecnología, y de esa manera evitar su obsolescencia

Si bien intenté ser lo suficiente claro al exponer cada tema, puede ser necesario para afianzar los conocimientos, el volver a leer algunos temas después de haber avanzado en algunos capítulos, especialmente en lo referente a los capítulos 3 y 4. Sin embargo esto es común en este tipo de material de alta complejidad.

El constante avance en la tecnología y la necesidad de mantener los conocimientos actualizados, fue la principal razón para elaborar un Curso de Microcontroladores, el cual espero sea de gran utilidad para muchos estudiantes.

VII. REFERENCIAS Y BIBLIOGRAFIA

Tokheim,Roger L.
Fundamentos de los Microprocesadores
Ed. McGraw Hill , Segunda Edición 1991

MacKenzie I. Scott
The 8051 Microcontroller
Ed. Merril, 1992

Bishop, Ron
Basic Microprocessors and the 6800
Ed. Hayden Book Company, Inc., 1979

Gilmore, Charles M.
Principios de Microprocesadores
Ed. Limusa, 1989

Gil Padilla, Antonio J.
Electrónica General 1. Dispositivos y Sistemas Digitales
Ed. MacGraw Hill, 1989

Motorola
HCMOS Single-Chip Microcontroller MC68HC11E9
Motorola Inc., 1988

Motorola
M68HC11 Reference Manual
Motorola Inc., 1990

Dirección General De Institutos Tecnologicos
Documento Técnico sobre los Programas de Estudios
Elaborados con Unidades de Aprendizaje, 1985

Nerici, G. Imideo
Metodología de la Enseñanza
Ed. Kapelusz, 1985

Peaget,Jean
Teoría de psicogénesis del conocimiento. Problemas de Psicología Genética., 1980

VIII. ANEXOS

VIII.1. PROGRAMA DE ESTUDIOS

1. DATOS GENERALES DE IDENTIFICACION DEL CURSO.

1.1. Nombre del curso: **MICROCONTROLADORES.**
1.2. Nivel: **LICENCIATURA.**
1.3. Carrera(s):**INGENIERIA ELECTRONICA.**
1.4. Lugar y fecha de elaboración: Mty., N.L. Junio 1994.

2. HISTORIA DEL PROGRAMA.

2.1. Lugar y fecha de revisión:
2.2. Participantes:
2.3. Cambios y justificaciones:

3. UBICACION DE LA ASIGNATURA.

3.1.1. Relación con Asignaturas Anteriores:
- Lógica Digital.
- Programación.
- Teoría de Control.

3.1.2. Relación con Asignaturas Posteriores:
- Microprocesadores II.

3.2. Aportación al Desempeño Profesional:
- Conocer el funcionamiento y uso de microcontroladores y otros circuitos para aplicaciones en aparatos electrónicos.

4. OBJETIVOS GENERALES DEL CURSO.

- Programar microcontroladores usando programas y métodos actuales (Ensambladores, simuladores, Lenguajes de bajo nivel).
- Conectar adecuadamente elementos o dispositivos de entrada, salida e interfaces al exterior.
- Diseñar dispositivos electrónicos usando microcontroladores como elemento básico.

5. TEMARIO.

UNIDAD 1.-CONCEPTOS GENERALES DE MICROCONTROLADORES.
-Introducción a los microcontroladores.
-Tipos de Computadoras y Microprocesadores vs. Microcontroladores.
-El *Microcontrolador como circuito integrado.*
-Conexiones.
-Organización de memoria.

UNIDAD 2.-CONJUNTO DE INSTRUCCIONES

-Modelo del programador.
-Modos de Direccionamiento.
-Instrucciones del Microcontrolador.
-Programa Fuente y Editores.
-Programa Objeto y Ensambladores.
-Programa Monitor e Interconexiones.

UNIDAD 3.-PUERTOS PARALELO Y CONVERTIDOR ANALOGO-DIGITAL

-Puertos
-Comunicación Paralelo.
-Proceso de Conversión.
-Operación en canal único y multicanal.
-Registro de Control.

UNIDAD 4.-PROYECTO.

6. APRENDIZAJES REQUERIDOS.

- Diseño de Circuitos Combinacionales.
- Diseño de Circuitos Secuenciales.
- Unidad Lógica-Aritmética, Unidad de Control, Unidades
- Entrada/Salida.
- Algoritmos computacionales.

7. UNIDADES DE APRENDIZAJE

UNIDAD I.- CONCEPTOS GENERALES DE MICROCONTROLADORES.

OBJETIVO EDUCACIONAL.- *Describir el funcionamiento y arquitectura de un* microcontrolador. Desarrollando además un vocabulario de trabajo, en donde sea capaz de utilizar apropiadamente la terminología, siglas, abreviaciones y modismos necesarios.

APRENDIZAJES INTERMEDIOS.

Subtema.-Introducción a las computadoras.

1.1. Describir los componentes básicos y el funcionamiento de una computadora.

Subtema.-Tipos de Computadoras y Microprocesadores vs. Microcontroladores

1.2. Clasificar las computadoras de acuerdo a sus características. Describir las diferencias en la arquitectura, aplicaciones y conjunto de instrucciones entre un microprocesador y un microcontrolador.

Subtema.-El Microcontrolador como circuito integrado.

1.3 Establecer las características generales de un microcontrolador y explicar la función que realiza cada uno de los bloques (subsistemas) que lo conforman.

Subtema.-Conexiones y Terminales..

1.4. Expresar la función que realiza cada terminal de un microcontrolador y reconocer el significado de las siglas que las representan.

Subtema.-Organización de la memoria.

1.5. Describir la organización de la memoria en cada modo de operación.

UNIDAD II.-CONJUNTO DE INSTRUCCIONES.

OBJETIVO EDUCACIONAL.- Aplicar el conjunto de instrucciones de un microcontrolador en la elaboración de programas en lenguaje ensamblador, convertirlos a lenguaje de máquina, transferirlos al microcontrolador y hacerlos funcionar.

APRENDIZAJES INTERMEDIOS.

Subtema.-Modelo del Programador.

2.1. Comprender la función del CPU y los registros internos que son utilizables por el programador.

Subtema.-Modos de Direccionamiento.

2.2. Comprender el significado de los diversos modos de direccionamiento.

Subtema.-Instrucciones del Microcontrolador.

2.3..Entender la acción que realiza cada instrucción y su efecto en los contenidos de los registros y memoria antes y después de ser ejecutadas.

Subtema.-Programa Fuente y Editores.

2.4. Realizar un programas fuente en lenguaje ensamblador y usar un Editor de Texto para convertirlos a lenguaje ASCII.

Subtema.-Programa Objeto y Ensambladores.

2.5. Utilizar un programa ensamblador para obtener el programa objeto en el lenguaje de máquina.

Subtema.-Programa Monitor e Interconexión con el microcontrolador.

2.6. Utilizar un programa monitor para transferir el programa objeto al microcontrolador y realizar las conexiones necesarias para probar su funcionamiento.

UNIDAD III.-PUERTOS PARALELO Y CONVERTIDOR ANALOGO-DIGITAL

OBJETIVO EDUCACIONAL.-Utilizar los puertos paralelo para transmisión y recepción de datos mediante programas en ensamblador, y aplicar el convertidor análogo-digital en la manipulación de señales.

APRENDIZAJES INTERMEDIOS.

Subtema.-Puertos

3.1. Programar las terminales de los puertos como: entradas, salidas, bidireccionales o para aplicaciones especiales.

Subtema.-Comunicación Paralelo.

3.2. Realizar Comunicación Paralelo en: Modo Directo, Strobe Simple y Protocolo Total (Full Handshake).

Subtema.-Operación en canal único y multicanal.

3.3 Realizar programas para la operación de conversión en canal único o en multicanal.

Subtema.-Registros de Control.

3.4 Entender el efecto de los diversos registros de control del convertidor Análogo-Digital.

UNIDAD 4.-PROYECTO.

OBJETIVO EDUCACIONAL .- Aplicar los conceptos desarrollados durante el curso, a la solución de un problema digital. Tomar como base el hardware de soporte del MC68HC11A8 y hacerle las adiciones necesarias. Además, diseñar el software requerido utilizando el Editor de Texto, el Programa Ensamblador y el Programa Monitor PCbug11 para integrar todo el conjunto, probarlo y corregirlo.

VIII.2. LISTA DE PRACTICAS DE LABORATORIO

La elaboración de Prácticas de Laboratorio proporciona la experiencia en la manipulación del conjunto de instrucciones y en el hardware de soporte del MCU. Además, mediante el uso de la computadora se adquiere la habilidad en el manejo de los Programas Ensamblador y del Programa Monitor. Lo anterior permitirá al estudiante el poder emprender proyectos de mayor magnitud con la plena seguridad de su funcionalidad.

Es recomendable que durante el desarrollo y como reporte de cada práctica, el alumno realice lo siguiente:

- Descripción de la práctica.
- Elaboración del Diagrama de Flujo.
- *Edición del Programa.*
- Ensamblado del Programa.
- Elaboración del diagrama de conexiones con el hardware de soporte del MCU.
- Depuración y puesta en operación mediante el uso del Programa Monitor.
- Observaciones y Conclusiones del desarrollo de la práctica.

A continuación se presenta una lista de las prácticas sugeridas:

Practica #1. Descripción del Sistema de Desarrollo: Editor de Texto, Programa Ensamblador, Programa Monitor, Hardware de soporte del MCU.

Practica #2. Programa que realice la suma y resta de dos números.

Práctica #3. Programa que lea las señales de entrada proporcionada por interruptores y envíe a la salida para mostrar las condiciones mediante el encendido y apagado de leds

Práctica #4. Elaboración de un programa que encienda leds de izquierda a derecha en forma secuencial, con la posibilidad de variar la rapidez de encendido.

Práctica #5. Diseño de una interfaz para mostrar números hexadecimales en un display.

Practica #6. Diseño de una interfaz para lectura de un teclado hexadecimal y mostrar los valores en un display.

Práctica #7. Conversión de un voltaje analógico a señal digital.

Práctica #8. Programa que realíce operaciones de suma, resta y multiplicación de doble precisión.

Práctica #9. Programa que realíce operaciones con tablas de senos y cosenos.

Práctica #10 Verificación de la operación del proyecto del curso.

VIII.3 APUNTES

CURSO DE MICROCONTROLADORES

ING. RENE BRIONES LARA

San Nicolás de los Garza, N.L. Septiembre de 1994

PREFACIO

Estos apuntes son desarrollados para formar parte de un Curso de Microcontroladores, y examinan las características del hardware (equipo) y del software (programación) de los microcontroladores en general; pero esencialmente de los dispositivos de la familia M68HC11 de Motorola. Estos apuntes están dirigidos a estudiantes de Ingeniería Electrónica y a todos aquellos estudiantes, profesionistas y técnicos con conocimientos de electrónica y circuitos digitales, que tengan deseos de conocer como se aplican los microcontroladores.

En el desarrollo de estos apuntes se ha tomado como prototipo el microcontrolador MC68HC11A8, dada su amplia aplicación en procesos de control industrial. Se pretende que al final del curso, el estudiante realice un proyecto que involucre: diseñar, implementar y documentar un proceso controlado por el microcontrolador MC68HC11A8, además que incorpore hardware y software con su propia originalidad.

Dado que el MC68HC11A8, así como todos los microcontroladores contienen un alto grado de funcionalidad, se hace énfasis en su arquitectura y programación en vez de en los detalles eléctricos. Los tópicos de software son desarrollados en función de aplicar: el Programa Ensamblador AS11 y el Programa Monitor PCbug11.

El capítulo 1 sirve como introducción a los microcontroladores, con énfasis en las diferencias entre los microcontroladores y los microprocesadores. El capítulo 2 describe la arquitectura del hardware del microcontrolador MC68HC11A8 que forma parte de la familia M68HC11. El capítulo 3 introduce el Conjunto de Instrucciones, empezando con definiciones de los Modos de Direccionamiento del MC68HC11A8. El Conjunto de Instrucciones ha sido dividido en grupos de instrucciones (aritméticas, lógicas, de transferencia de datos, etc.), lo cual facilita la presentación paso a paso. Numerosos ejemplos breves se presentan para muchas instrucciones, incluyendo sus modos de direccionamiento.

Los capítulos 4, 5 y 6, progresan a través de las características del MC68HC11A8, no debe precipitarse en estos capítulos, pues es esencial un sólido entendimiento de la arquitectura del hardware y del conjunto de instrucciones del microcontrolador al ir avanzando en estos tópicos.

El capítulo 4 presenta las técnicas y herramientas que son utilizadas en un Sistema de Desarrollo de Software, es importante la elaboración de prácticas en este punto, pues la experiencia en la manipulación del conjunto de

instrucciones nunca podrá ser sobre enfatizada. Los estudiantes deben ser animados en completar las prácticas utilizando la computadora, observando la salida y los mensajes de error provenientes del Programa Ensamblador AS11 y el Programa Monitor PCbug11, y viendo la utilidad de la conversión del programa fuente a hexadecimal.

El capítulo 5 trata sobre Puertos Paralelo para realizar la comunicación o interacción con el mundo real. Y en el capítulo 6 es presentada la conversión análogo-digital.

La culminación de un Curso de Microcontroladores mediante un proyecto es quizá lo mas importante, pues desarrolla en el estudiante el potencial de emprender proyectos de gran escala. Tópicos de programación estructurada, ensamblado, monitoreo , depuración, y eslabonamiento de un programa, producirán un cambio significativo en muchos estudiantes.

En estos apuntes se hace un amplio uso de la literatura de Motorola, especialmente de los dispositivos de la familia M68HC11. En el apéndice se hace un listado completo del Conjunto de Instrucciones del MC68HC11A8.

Existe una dificultad muy especial en la terminología utilizada para describir el funcionamiento de los microcontroladores, sobre todo si se intenta traducirla. Dado que es más común utilizar los términos en inglés, opté por utilizar los términos originales en muchos casos.

La tecnología referente a microcontroladores, y a todos los campos de la electrónica en general, está avanzando a pasos agigantados; por lo que opté por numerar las hojas de cada capítulo en forma independiente, empezando la numeración con el número del capítulo, por ejemplo: 2.1, 2.2, 2.3, etc. Esto permitirá modificar cada capítulo en forma independiente, para así actualizar estos apuntes conforme avance la tecnología y de esa manera evitar su obsolescencia.

Deseo expresar mi profundo agradecimiento al Ing. Ronald López por sus valiosas sugerencias, apoyo y tiempo, que me brindó durante el desarrollo del escrito de este curso.

TABLA DE CONTENIDOS

TABLA DE CONTENIDOS continuación...

TABLA DE CONTENIDOS continuación...

CAPITULO 1

CONCEPTOS GENERALES DE MICROCONTROLADORES.

1.1 INTRODUCCION.

El impacto de las computadoras a sido muy profundo a pesar de haber aparecido hace solo algunas décadas, pues se han venido utilizando en forma general desde 1950. Su presencia es sentida por todos debido a la gran cantidad de publicidad que aparece en periódicos, radio y televisión.

Generalmente las computadoras son consideradas como máquinas "procesadores de datos" o realizando operaciones matemáticas con inagotable competencia. Nosotros confrontamos las computadoras desde un punto de vista bastante distinto, es decir, realizando su trabajo en forma silenciosa, eficiente y cuya presencia generalmente es desapercibida.

Como componente central de una computadora encontramos un circuito integrado llamado microprocesador. Como un concepto nuevo a surgido recientemente un circuito llamado microcontrolador, el cual realiza funciones de control por interactuar con el mundo real, encendiendo o apagando interruptores o realizando actividades de monitoreo. Algunas aplicaciones del microcontrolador son: controladores industriales, hornos de microondas, lavadoras automáticas, automóviles, juguetes, video caseteras, estéreos, etc.

1.2 ORGANIZACION DE LA COMPUTADORA.

Una **computadora** puede ser definida como un sistema de procesamiento de datos, que puede ser programada para operar sin la intervención humana, y que tiene además la habilidad de almacenar y obtener datos. Un sistema básico de computadora se muestra en la figura 1.2.1. La **Arquitectura de la Computadora** es el nombre dado a ésta organización funcional y consta de 5 unidades:

1. **CPU** (Central Procesing Unit).- Unidad de Procesamiento Central.
2. **RAM** (Random Access Memory).- Memoria de Acceso Aleatorio.
3. **ROM** (Read Only Memory).- Memoria de Solo Lectura.

4. **BUSES.-** Conjunto de alambres que transportan información con un propósito común.
5. **PUERTOS I/O** (Input/Output).- Puertos de Entrada/Salida.

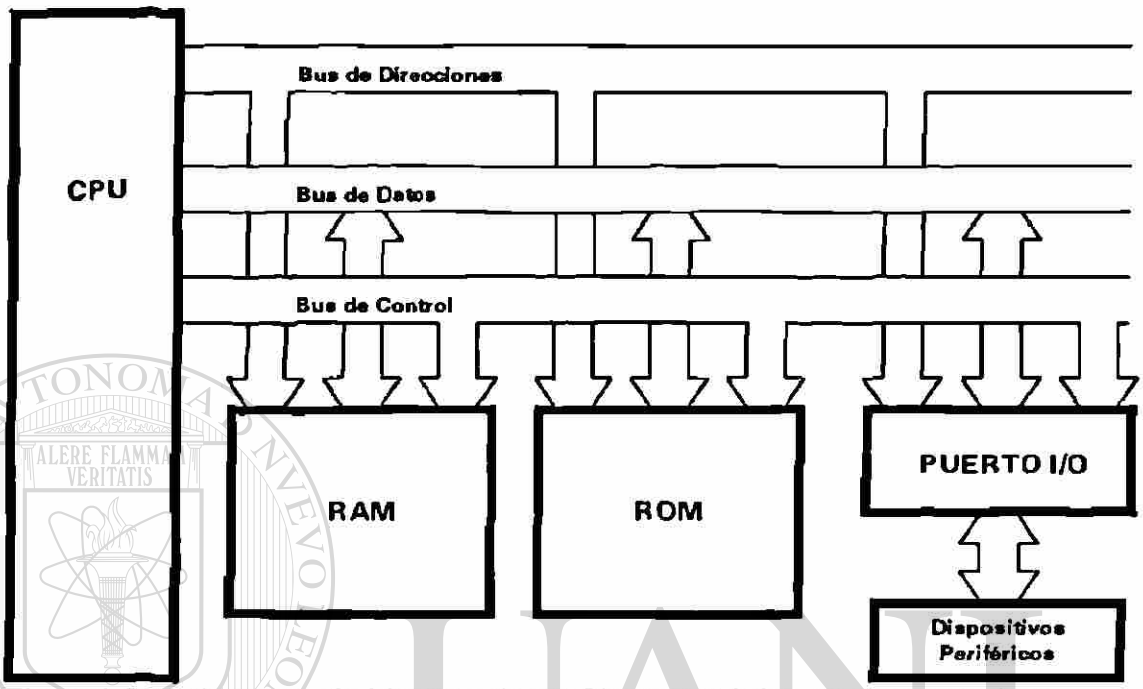


Figura1.2.1 Diagrama de bloques de un Sistema Básico de Computadora.-

El **hardware** es el equipo y constituye las unidades físicas representadas por los bloques de la figura 1.2.1. Para que el hardware sea útil, la memoria del programa debe decir al CPU lo que tiene que hacer. La **Programación** es la preparación de la lista de instrucciones. El **Software** es un término general utilizado para denominar a todos los programas. **Firmware** es el nombre dado al software que es almacenado permanentemente en la memoria de programa.

La **CPU** o Unidad de Procesamiento Central controla todas las unidades del sistema por medio de las líneas o **bus de control.** El **bus de direcciones** es un conjunto de líneas que selecciona una cierta localidad de memoria, puerto de entrada o puerto de salida. El **bus de datos** es el conjunto de líneas de doble sentido o dirección, que se utiliza para introducir o sacar datos de la CPU. Es importante observar que la CPU puede enviar o recibir datos de la memoria utilizando el bus de datos.

La **ROM** (Read Only Memory) o Memoria de Solo Lectura habitualmente se utiliza para almacenar el programa en forma permanente. La **RAM** (Random **Access Memory**) o memoria de acceso aleatorio es una memoria de lectura/escritura en donde se almacena en forma temporal los datos o algunos

programas de naturaleza temporal. Por claridad, es costumbre omitir en los diagramas de bloques: las fuentes de alimentación, relojes y algunas líneas de realimentación del CPU.

1.3 OPERACION DE LA COMPUTADORA.

En la figura 1.3.1 se muestra un ejemplo de como opera una computadora y en él se muestra el siguiente procedimiento:

1. Pulsar la tecla A.
2. Almacenar en memoria la letra A.
3. Sacar la letra A en la pantalla del monitor de tubo de rayos catódicos.

El procedimiento de entrada-almacenamiento-salida esbozado en la figura 1.3.1 es una operación común de un sistema de computadora. Para comprender éste procedimiento de transferencia de datos y explicar el uso de las diferentes unidades de la computadora, un diagrama más detallado se muestra en la figura 1.3.2

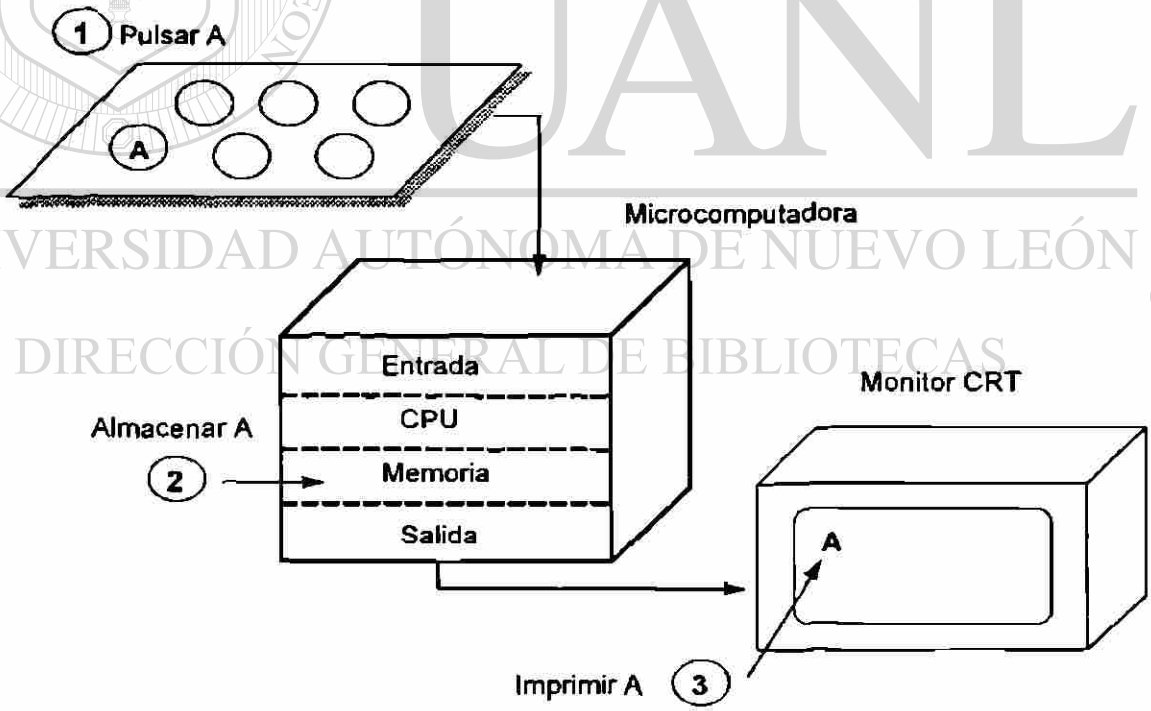


Fig. 1.3.1. Operación típica de la microcomputadora

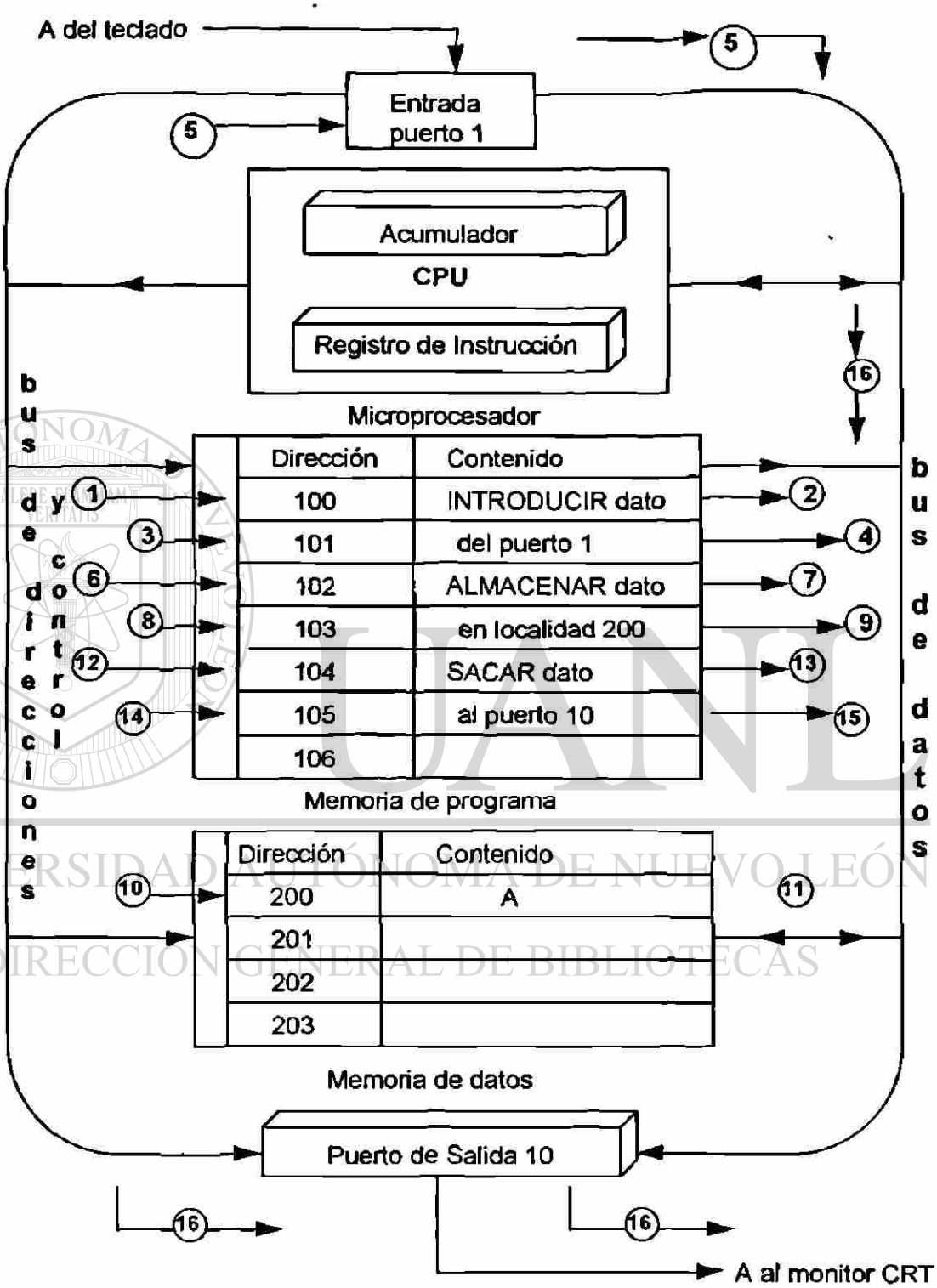


Fig. 1.3.2. Operación paso a paso de una microcomputadora cuando ejecuta las instrucciones de la memoria del programa

Observe cuidadosamente la figura 1.3.2. y vea los contenidos de la memoria de programa. Verá que las instrucciones ya se han cargado en las seis primeras localidades de memoria. Estas instrucciones que están en la memoria de programa son:

1. **INTRODUCIR el dato,** del puerto de entrada 1.
2. **ALMACENAR el dato** del puerto 1, en la localidad 200 de la memoria de datos.
3. **SACAR el dato,** al puerto de salida 10.

Observar que solamente hay tres instrucciones en el programa anterior, aunque puede parecer que hay seis en la memoria de programa de la figura 1.3.2. La razón para pensar esto es que las instrucciones se descomponen habitualmente en varias partes. La primera parte de la instrucción 1 consiste en **INTRODUCIR el dato.** La segunda parte indica de donde proviene el dato **del puerto de entrada 1.** La primera parte de la instrucción se denomina **operación** y la segunda parte **operando.** Operación y operando están ubicados en localidades separadas en la memoria de programa.

Para la primera instrucción de la figura 1.3.2 la localidad 100 de la memoria de programa contiene la operación **INTRODUCIR el dato,** mientras que la localidad 101 contiene el operando **del puerto de entrada 1** que es desde donde se introduce la información.

En el CPU de la figura 1.3.2 se identifican dos nuevas secciones denominadas el registro acumulador y el registro de instrucción.

La secuencia de eventos que ocurren en la computadora en el ejemplo de introducir-almacenar-sacar de la figura 1.3.1, se esboza en la figura 1.3.2. El flujo de instrucciones y datos puede seguirse en el diagrama por los números inscritos en los círculos. Se debe recordar que el CPU es el centro de todas las transferencias de datos y operaciones. En la figura 1.3.2 se pueden seguir los pasos que a continuación se detallan.

Paso 1. La CPU envía la localidad o dirección 100 al bus de direcciones. Una línea de control habilita (conecta) la entrada de lectura en la memoria de programa (leer significa copiar información de una localidad de memoria). Este paso se simboliza en la figura por un 1 inscrito en un círculo.

Paso 2.- La memoria de programa anuncia la primera instrucción **"INTRODUCIR dato"** al bus de datos, y la CPU acepta este mensaje codificado. La instrucción se coloca en una posición de memoria especial de la CPU denominada registro de instrucción. La CPU decodifica o interpreta la instrucción, y determina que necesita el operando de la instrucción (¿De dónde proviene el dato?).

Paso 3.- La CPU envía la localidad 101 al bus de direcciones y la línea de control habilita la entrada de lectura de la memoria de programa.

Paso 4.- La memoria de programa coloca el operando **"del puerto 1"** en el bus de datos. El operando estaba localizado en la localidad 101 de la memoria de programa. Este mensaje codificado **"la dirección del puerto 1"** se acepta en el bus de datos y se coloca en el registro de instrucción. La CPU decodifica ahora la instrucción completa **"INTRODUCIR el dato, del puerto 1"**.

Paso 5.- La CPU hace que se abra el puerto 1 utilizando el bus de direcciones y las líneas de control en la unidad de entrada. La forma está codificada en el acumulador de la CPU.

Es importante observar que la CPU sigue siempre una secuencia de busca-decodifica-ejecuta. Primero *busca* la instrucción en la memoria de programa; segundo, la decodifica y tercero la ejecuta. Trate de observar esta secuencia de busca-decodifica-ejecuta en las dos instrucciones siguientes y continúe con el programa listado en la memoria de programa de la figura.

Paso 6.- La CPU direcciona la localidad 102 en el bus de direcciones; después habilita la entrada de lectura en la memoria de programa utilizando las líneas de control.

Paso 7.- El código de la instrucción **"ALMACENAR el dato del puerto 1"** es leído en el bus de datos y es aceptado por la CPU en el registro de instrucción.

Paso 8.- La CPU decodifica la instrucción **"ALMACENAR el dato del puerto 1"** y determinar que necesita el operando. La CPU direcciona la siguiente localidad de memoria 103 y habilita la entrada de lectura de la memoria de programa.

Paso 9.- El código para **"en la localidad de memoria 200"** es colocado en el bus de datos por la memoria de programa. La CPU acepta este operando y lo almacena en el registro de instrucción. La instrucción completa **"ALMACENAR el dato del puerto 1, en la localidad de memoria 200"** ha sido buscada en memoria y decodificada.

Paso 10.- Ahora comienza el proceso de ejecución. La CPU envía la dirección o localidad 200 al bus de direcciones y habilita la entrada de escritura de la memoria de datos . Escribir significa copiar datos en una localidad de memoria.

Paso 11.- La CPU pone la información almacenada en el acumulador en el bus de datos (la forma codificada de A). La A se escribe en la localidad 200

de la memoria de datos. La segunda instrucción ha sido ejecutada. El proceso de ALMACENAR no destruye el contenido del acumulador, éste contiene todavía la forma codificada de A.

Paso 12.- La CPU debe buscar la información siguiente. Direcciona la localidad 104 y habilita la entrada de lectura de la memoria de programa.

Paso 13.- El código de la instrucción **"SACAR el dato"** se coloca en el bus de datos. La CPU acepta la instrucción en el registro de instrucción, la decodifica y determina que necesita un operando.

Paso 14.- La CPU coloca la localidad 105 en el bus de direcciones y habilita la entrada de lectura de la memoria de programa.

Paso 15.- La memoria de programa pone el código del operando **"al puerto 10"** en el bus de datos. La CPU acepta este código en el registro de instrucción.

Paso 16.- La CPU decodifica la instrucción completa **"SACAR el dato, al puerto de salida 10"** y activa al puerto 10 utilizando el bus de direcciones y las líneas de control en la unidad de salida. La A es transmitida desde el puerto 10 al monitor CRT.

La mayoría de las CPU's transfieren información de forma similar a la detallada en la figura 1.3.2. Las variaciones más grandes están probablemente en las secciones de entrada y salida. A veces se necesitan algunos pasos para lograr que las secciones de entrada y salida operen adecuadamente.

Es importante observar que la CPU es el centro y control de todas las operaciones y sigue la secuencia de busca-decodifica-ejecuta. Siempre las operaciones reales de la CPU son dictadas por las instrucciones listadas en la memoria del programa.

1.4 TIPOS DE COMPUTADORAS.

Las computadoras considerando su tamaño y potencia de cómputo pueden ser clasificadas en:

1. Microcomputadoras.
2. Minicomputadoras.
3. Computadoras de tablero Principal.

Las Microcomputadoras tienen como característica principal su tamaño reducido y el empacado del CPU, el cuál está contenido en un solo circuito integrado llamado microprocesador. Típicas **microcomputadoras** son las computadoras personales, como por ejemplo: IBM PC, Apple Macintosh, y la Commodore Amiga, que incorporan a un microprocesador como su CPU. La RAM, la ROM y los puertos de entrada y salida requieren de muchos circuitos integrados. Conforme se incrementa la cantidad de circuitos, se incrementa la potencia de cómputo. Los circuitos de interfaz varían considerablemente dependiendo de los dispositivos I/O (de entrada/salida). El manejo de la bocina contenida en muchas micro computadoras requiere solo un acoplamiento de compuertas lógicas, pero la interfaz del disco usualmente requiere muchos circuitos integrados. Otras características de las microcomputadoras son que interactuan con un solo usuario y ejecutan un solo programa a la vez.

Las Minicomputadoras son más complejas en su arquitectura y tienen un CPU consistente de varios circuitos integrados, lo cuál es necesario para obtener mayor velocidad y potencia de cómputo, además las minicomputadoras son multiusuario y ejecutan simultáneamente varios programas, lo cual es una ilusión resultante de un recurso de la CPU llamado "tiempo compartido".

Las Computadoras de Tablero Principal son de gran tamaño y potencia de cómputo, su arquitectura es muy compleja y su CPU consiste de varias tablillas de circuito impreso con múltiples circuitos integrados, además tienen gran velocidad y potencia de cómputo y puede realizar varias tareas al mismo tiempo.

1.5 MICROPROCESADORES VS. MICROCONTROLADORES.

Como fue indicado en el punto anterior, las microcomputadoras tienen un microprocesador como circuito integrado único que realiza las funciones de la CPU. Entonces, ¿Qué es lo que hace al microcontrolador diferente de un microprocesador? Esta pregunta será contestada desde 3 puntos de vista:

1. Arquitectura del hardware.
2. Aplicaciones.
3. Conjunto de instrucciones

Arquitectura del Hardware.

Para resaltar la diferencia entre los microcontroladores y los microprocesadores, la figura 1.2.1. es redibujada mostrando más detalles en la figura 1.5.1.

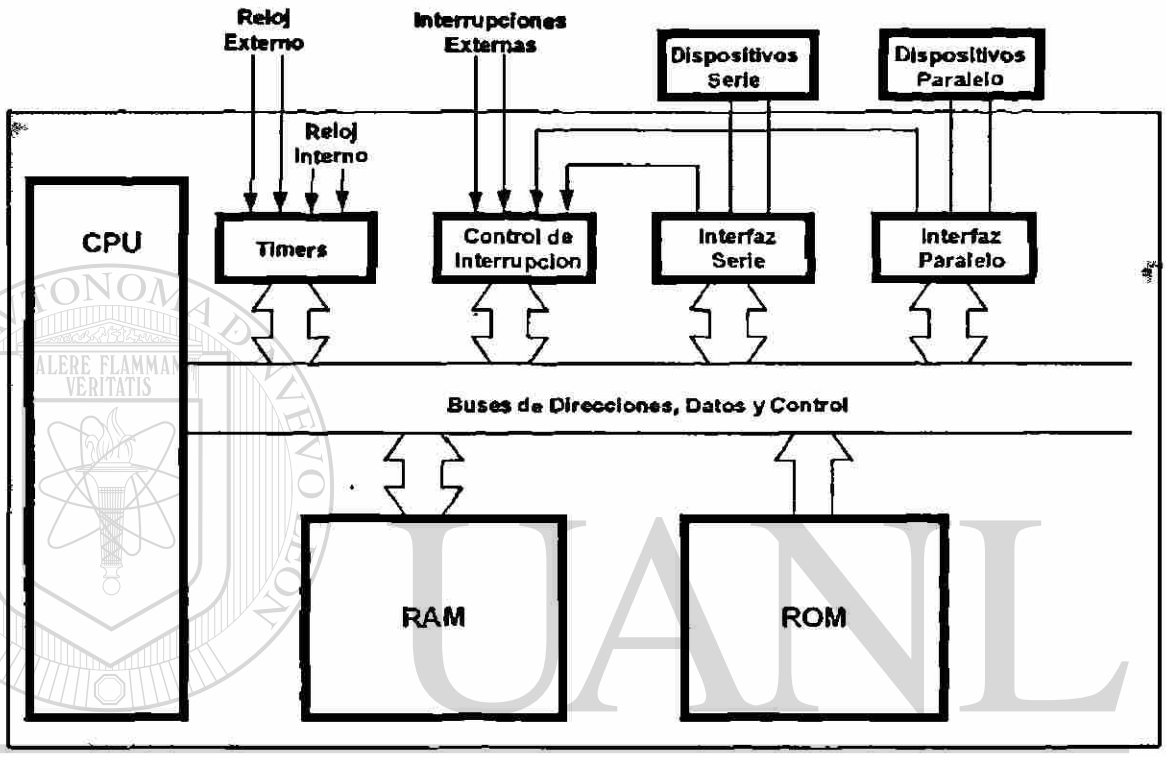


Fig.1.5.1. Diagrama de Bloques de un sistema de Microcomputadora.

Mientras que el microprocesador es un circuito integrado que solo realiza las funciones de CPU, el microcontrolador contiene en un solo circuito integrado al CPU y mucho del circuito restante de un sistema completo de micro- computadora. Los componentes dentro del área sombreada en la figura1.5.1. son una parte integral de muchos circuitos integrados microcontroladores. Además de la CPU, el microcontrolador incluye: RAM, ROM, interfaz serie, interfaz paralelo, timer, y circuitos con un programa de interrupciones, todo en el mismo circuito integrado. Por supuesto, la cantidad de RAM en el chip o pastilla no se aproxima a la que incluye el más modesto sistema de microcomputadora, pero como aprenderemos después, esto no es una limitante, dado que los microcontroladores son intentados para aplicaciones bastante diferentes.

Una importante característica de los microcontroladores es el sistema de interrupciones interconstruido. Como dispositivos orientados al control, los

microcontroladores son generalmente llamados a responder a estímulos externos en tiempo real. Ellos deben realizar rápidas interrupciones, suspendiendo un proceso y ejecutando otro en respuesta a un "evento".

El abrir la puerta de un horno de microondas, es un ejemplo de un evento que debe causar una interrupción en un producto basado en un microcontrolador. Por supuesto, muchos microcontroladores pueden también implementar potentes esquemas de interrupción, pero componentes externos son requeridos. Los circuitos en el microcontrolador incluyen todos las circuitos necesarios para manejar interrupciones.

Aplicaciones.

Los microprocesadores son comúnmente usados como CPU en los sistemas de micro computadoras, para eso fueron diseñados y es donde su mayor potencial descansa. Los microcontroladores por otro lado, son encontrados en pequeños diseños, donde mínima cantidad de componentes realizan actividades orientadas a control. Estos diseños fueron implementados en el pasado con docenas o cientos de circuitos integrados digitales, y ahora, un microcontrolador ayuda en reducir la cantidad total de componentes. Todo lo que es requerido es un microcontrolador, un pequeño número de componentes de soporte y un programa de control en ROM. Los microcontroladores son adecuados para control de dispositivos I/O (entrada/salida) en diseños donde son requeridos una mínima cantidad de componentes, mientras que los microprocesadores son adecuados para procesamiento de información en sistemas de computadoras.

El Conjunto de Instrucciones.

Un Microcontrolador esta previsto para control de entradas y salidas. Las interfaces para muchas entradas y salidas usan un solo bit. Por ejemplo, un motor puede ser encendido o apagado por un solenoide energizado por 1 bit de un puerto de salida. Los microcontroladores tienen instrucciones para activar o desactivar bits individuales y realiza operaciones orientadas a un solo bit como: AND, OR, OR-exclusivo, y realizar saltos cuando un bit es activado o desactivado. Esta potente característica es raramente presente en microprocesadores, los cuales son usualmente diseñados para operar en bytes o con grandes unidades de datos.

En el control y monitoreo de dispositivos quizá con interfaces de 1 bit, los microcontroladores tienen inter-construidos circuitos e instrucciones para operaciones de entrada/salida, sincronización de eventos y habilitación y fijación de niveles de prioridad para interrupciones causadas por estímulos

externos. Los microprocesador generalmente requieren circuitos adicionales (circuitos integrados para interfaz serie, controladores de interrupción, timers, etc.) para realizar operaciones similares. No obstante, la capacidad de procesamiento completo de un microcontrolador nunca se aproxima a la de un microprocesador, dado que la mayor parte del estado real de los circuitos integrados es consumido por las funciones en el chip (pastilla), en detrimento del potencial de procesamiento

Dado que el estado real en el chip es apremiante en los microcontroladores, las instrucciones deben ser extremadamente compactas, con la mayoría de ellas implementadas en un solo byte. Un criterio general de diseño es que el programa de control debe ser fijado en la ROM incluida en el chip, dado que el agregar alguna ROM externa aumentaría mucho el costo del producto final. Un esquema codificado compacto para el conjunto de instrucciones es esencial. Esto es raramente una característica de los microprocesadores; sus potentes modos de direccionamiento traen con ellos instrucciones codificadas menos compactas.

1.6 CPU UNIDAD DE PROCESAMIENTO CENTRAL.

La **CPU (Central Procesing Unit)** o Unidad de Procesamiento Central es el "cerebro" de la computadora, administra todas las actividades en el sistema y realiza todas las operaciones con los datos. La CPU es solamente una colección de circuitos lógicos que realizan en forma continua solo dos operaciones: reconocer instrucciones (fetch instructions) y ejecutar instrucciones (executing).

El **Conjunto de Instrucciones** es un conjunto de códigos binarios que el CPU (Inherentemente en su diseño) es capaz de entender y ejecutar. Estas instrucciones normalmente se agrupan en:

1. Instrucciones Aritméticas (suma, resta, multiplicación, división).
2. Instrucciones Lógicas (AND, OR, NOT, etc.).
3. Instrucciones de Transferencia de datos (Cargar, almacenar, transferir, etc.)
4. Instrucciones de Salto y bifurcación (Saltos Condicionales e Incondicionales.
5. Instrucciones de llamada y retorno de subrutinas.
6. Instrucciones Misceláneas (Interrupciones, no operación, espera, etc.)

La figura 1.6.1 es una vista simplificada del interior de una CPU. En esta figura se pueden observar:

1. **Registros** para almacenar temporalmente la información.
2. **ALU= Arithmetic and Logic Unit (Unidad Aritmética-Lógica)** para realizar operaciones sobre la información.
3. **Decodificador de Instrucciones y Unidad de Control** que determina la operación a realizar y el conjunto de movimientos y acciones necesarias para efectuarlas.

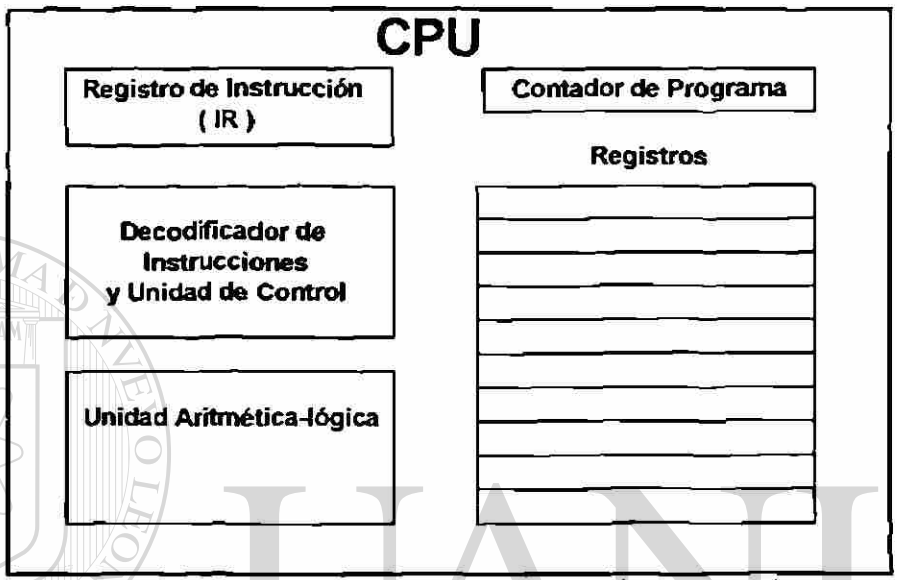


Figura 1.6.1 Unidad de Procesamiento Central CPU

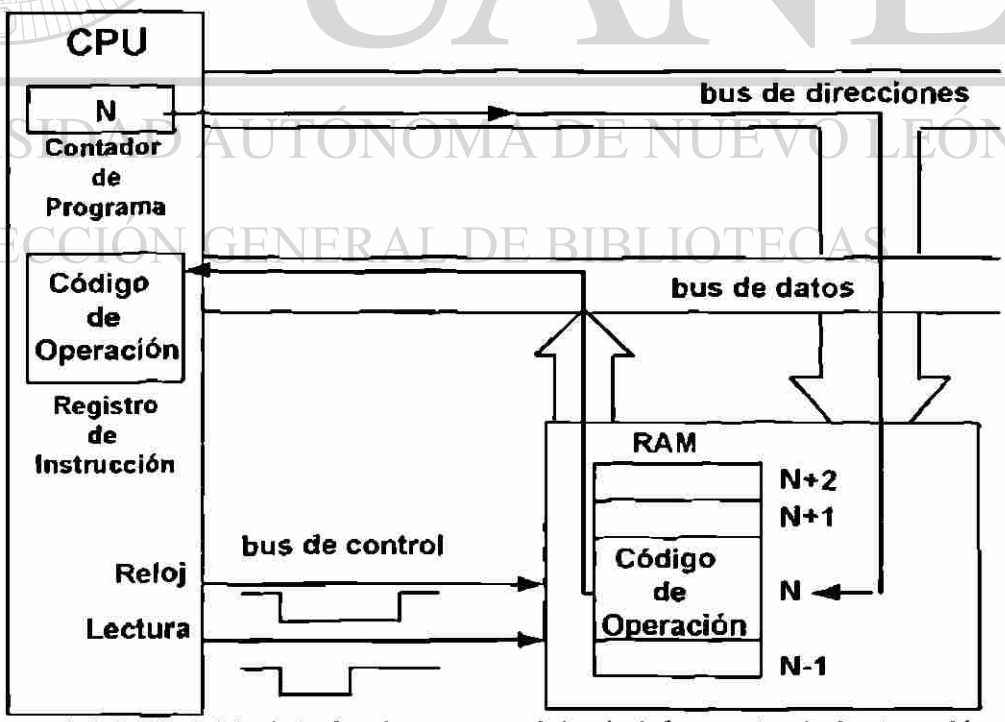


Figura 1.6.2. Actividad de los buses en ciclo de búsqueda de instrucción.

La figura 1.6.2 ilustra el flujo de información para una búsqueda de instrucción. El **buscar una instrucción** (fetch instruction) desde la memoria RAM involucra los siguientes pasos.

a. El contenido del contador de programa es colocado en el bus de direcciones.
b. Una señal de control de lectura es activada.
c. Un dato o código de operación de la instrucción es leído de la RAM y es colocado en el Bus de datos.
d. El código de operación es fijado en el Registro de Instrucciones interno del CPU
e. El Contador de Programa es incrementado para preparar el siguiente reconocimiento de instrucción desde la memoria.

La Ejecución de una instrucción involucra :
a. Decodificar es decir descifrar el código de operación.
b. Generar señales de control.
c. Activar registros internos, entradas y salidas de la ALU (unidad aritmética-lógica).
d. Señalar a la ALU que realice una operación específica.

El **Software** o programa es una serie de instrucciones combinadas para realizar un trabajo importante. El grado en que el trabajo es eficiente y correctamente desarrollado es determinado en su mayor parte por la calidad del software y no tanto en lo elaborado del CPU. Los programas "manejan" al CPU y si hacen algo impropio, entonces imitan la fragilidad de los autores. La frase "la computadora hizo un error" es equivocada. Aunque las fallas del equipo son inevitables, los errores son generalmente un signo de malos programas y errores del operador.

1.7 MEMORIAS.

Los programas y los datos son almacenados en memoria. Las variedades de memorias para computadora son bastas, la terminología que las acompaña es abundante y la tecnología toma rumbos distintos frecuentemente, tal que extensos y continuos estudios son requeridos para mantenerse al tanto de los últimos desarrollos.

La **memorias** son circuitos capaces de almacenar grandes cantidades de información mediante un determinado número de registros que utilizan entradas y salidas comunes para el acceso a todos ellos. Cada uno de estos registros de la memoria recibe el nombre de **localidad** y generalmente están constituidos por los elementos necesarios para almacenar 1, 4 u 8 bits. Un **bit**

es un dígito binario, esta palabra se forma funcionando binary digit y es la mínima cantidad de información que se puede almacenar o manejar.

Para cada localidad existe una dirección concreta expresada en forma numérica. Cuando se desea acceder a una determinada localidad es necesario direccionarla, es decir, activar la dirección correspondiente.

La mayoría de los sistemas digitales basados en microprocesadores o microcontroladores actualmente operan con paquetes de 8 bits y las memorias que se utilizan en la actualidad suelen estar constituidas por localidades direccionables que almacenan 8 bits. El conjunto de 8 bits se conoce con el nombre de **byte**. Un **nibble** esta formado por 4 bits, por lo tanto un byte tiene dos nibble.

Write o escritura es la operación de almacenar datos en memoria. **Read** o lectura es la operación de obtener el valor previamente almacenado en una localidad de memoria. Tanto la operación de escritura como de lectura en memoria se realiza seleccionando la localidad deseada a través del bus de direcciones y transportando la información que se desea (escribir o que se ha leído) mediante el bus de datos.

1.7.1 SISTEMAS DE NUMERACION BINARIO Y HEXADECIMAL.

Una determinada localidad de memoria se selecciona generando una combinación binaria, formada por tantos bits como líneas en el bus de dirección existan. Lo más normal es que un sistema digital programable necesite un circuito de memoria de gran capacidad; como consecuencia el número de líneas de direccionamiento será elevado. Utilizar largas expresiones matemáticas formadas por unos y ceros para designar cada localidad, resulta difícil, sobre todo cuando es necesario escribir el programa que hace funcionar al sistema. Los datos almacenados en memoria o aquellos con los que opera el sistema suelen ser de generalmente de 8 o 16 bits, y como consecuencia presentan el mismo problema.

El Sistema de Numeración Hexadecimal se utiliza para facilitar la tarea de programación y representación de localidades y datos. Un mismo número expresado en dicho sistema está formado por la cuarta parte de cifras de las que tendría expresado en binario.

El sistema de numeración hexadecimal (en base 16) utiliza 16 signos diferentes. En la figura 1.7.1 se muestran dichos signos y su equivalencia con el sistema decimal (en base 10) y con el sistema binario (en base 2)

Decimal base 10	Hexadecimal base 16	Binario base 2
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Figura 1.7.1 Equivalencia entre los sistemas de numeración decimal, hexadecimal y binario.

Las transformaciones entre números binarios y hexadecimales son muy sencillas. Cada cuatro bits o cifras binarias pueden ser representadas por un solo signo hexadecimal. Para pasar de binario a hexadecimal, los bits del número binario se agrupan en bloques de cuatro bits comenzando por la derecha. Si el último grupo resulta tener menos de cuatro bits, se podrá completar añadiendo ceros a la izquierda y cada grupo se sustituye por el signo hexadecimal correspondiente.

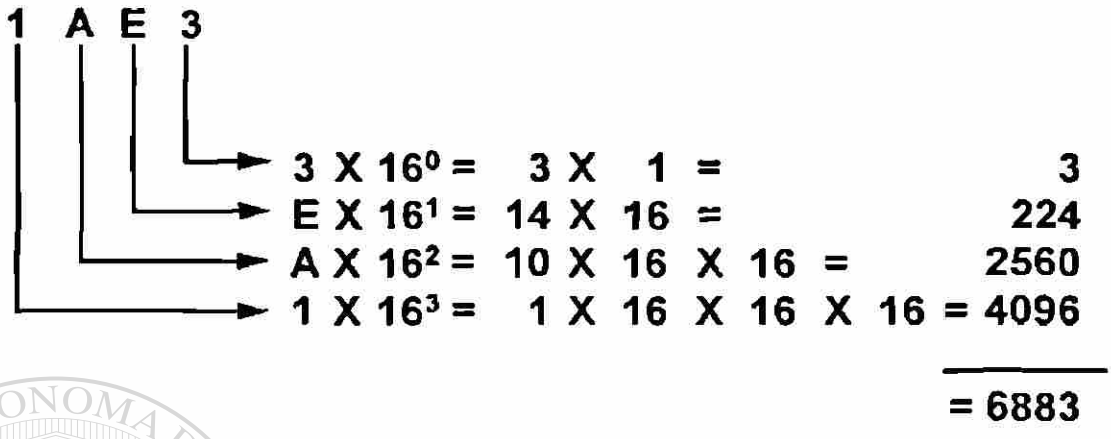
Los números binarios de cuatro cifras son fácilmente identificables con su equivalente en base 10 y estos últimos se relacionan de manera sencilla con los hexadecimales. Por lo tanto, se trata de realizar un proceso mental que consiste en pasar cada bloque binario a decimal y a continuación éste a base 16.

El siguiente ejemplo disipará cualquier duda: Pasar el número binario 1.1010.1110.0011 a hexadecimal:

0001	1010	1110	0011	binario
1	10	14	3	decimal
1	A	E	3	hexadecimal

Para pasar de hexadecimal a binario el proceso es el inverso al seguido en el ejemplo anterior.

Para obtener el equivalente decimal apartir del número hexadecimal, se sigue el siguiente procedimiento.



Las expresiones que utilizaremos en este y en los siguientes capítulos para denominar direcciones de localidades de memoria estarán comprendidas entre los números hexadecimales 0000 y FFFF. Aunque evidentemente, existen sistemas digitales. que necesitan números de más cifras para direccionar todas las localidades que constituyen su memoria.

Los microprocesadores o los microcontroladores utilizan comúnmente datos de 8 bits y para su representación en hexadecimal se utilizan números comprendidos entre el 00 y el FF. Para indicar que un número esta expresando en el sistema hexadecimal colocaremos $ antes del mismo, por ejemplo: $34.

1.7.2 CLASES DE MEMORIAS.

Son muy abundantes los tipos o clases de memorias disponibles en catálogo. Aquí serán referidos exclusivamente aquellos que están fabricados con materiales semiconductores y más concretamente los circuitos integrados **LSI** (Large Scale Integration) de gran escala de integración. También se hace referencia a dispositivos **MSI** (Medium Scale Intregration) de mediana escala de integración que tienen capacidad de almacenar información, como los registros de desplazamiento o corrimiento.

En la figura 1.7.2.1 se muestra una clasificación de las memorias atendiendo en este caso el siguiente orden: modo de acceso, forma de almacenamiento y tecnología de fabricación.

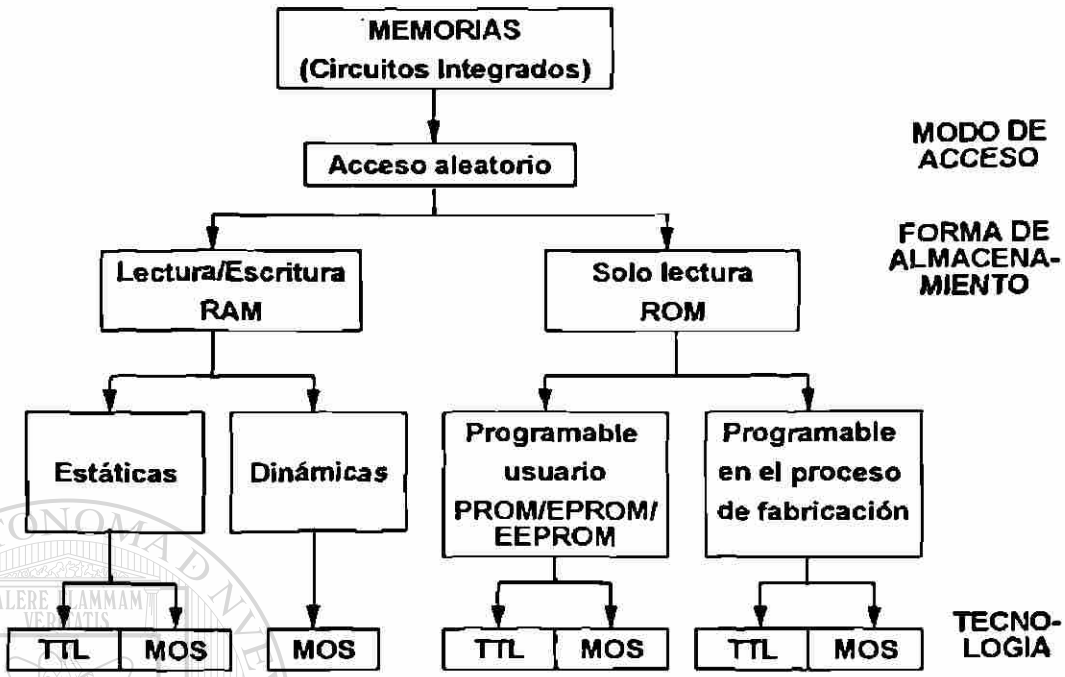


Figura 1.7.2.1 Clases de Memoria de Acceso Aleatorio

Aunque existe una gran variedad, las de mayor uso son las denominadas de acceso aleatorio es decir las RAM y ROM. En estos dispositivos se emplea el mismo tiempo para acceder a cada una de las localidades, sea cual sea su ubicación.

Las **RAM** (Random Access Memory) son memorias de acceso aleatorio en las que es posible la escritura y lectura de datos. Estas memorias son volátiles, es decir, pierden su información cuando se desconectan de la tensión de alimentación. Se utilizan generalmente para guardar resultados intermedios o finales del proceso y también para almacenar programas procedentes de dispositivos externos tales como discos. Son conocidas también como memorias de trabajo.

Las memorias RAM se dividen en estáticas y dinámicas. Las estáticas mantienen la información siempre y cuando se mantenga la alimentación eléctrica. Las dinámicas, de tecnología MOS exclusivamente, tienen un mayor número de registros por circuitos que las anteriores, y necesitan una señal de refresco controlada por el programa principal para poder mantener la información.

Las **ROM** (Read Only Memory) son memorias (de acceso aleatorio) de solo lectura, estas son no volátiles, es decir, mantienen la información aunque se desconecten de la alimentación eléctrica. Este dispositivo se utiliza para almacenar el programa de control del sistema.

Las memorias ROM de solo lectura se dividen a su vez en:

- **ROM**: Memorias programadas por máscara en el proceso de fabricación.
- **PROM:** Memorias programadas por el usuario en el momento de diseñar el sistema. Solo pueden ser programadas una vez.
- **EPROM**: Memorias de Solo Lectura Reprogramables, esta memoria pueden ser borrada con luz ultravioleta para ser reprogramada con nuevos patrones de bits.
- **EEPROM**: (Electrical Erasable Programmable ROM). Esta memoria puede ser borrada y programada electricamente.

1.8 BUSES DE DIRECCION, DATOS Y CONTROL.

Un **bus** es un conjunto de alambres que transportan información con un propósito común. El acceso a los circuitos de control alrededor del CPU es proporcionado por tres buses: El bus de direcciones, el bus de datos y el bus de control.

El **bus de direcciones** es utilizado para operaciones de lectura y escritura, el CPU especifica la localidad del dato (o instrucción) colocando una dirección en el bus de direcciones y entonces activa una señal en el bus de control indicando si la operación es de lectura o escritura.

La **Operación de Lectura** recupera un byte de dato de una localidad especifica en la memoria y la coloca en el bus de datos. El CPU lee el dato y lo coloca en uno de sus registros internos.

En la **Operación Escritura**, el CPU saca un dato y lo coloca en el bus de datos, debido a la señal de control, la memoria reconoce la operación como un ciclo de escritura y almacena el dato en la localidad especificada.

Dadas **n** líneas de dirección, cada una con la posibilidad de ser uno (1) o cero (0), 2^n localidades pueden ser direccionadas.

Un bus de direcciones de 16 lineas puede direccionar 2^{16}=65,536 localidades. Un bus de direcciones de 20 líneas pueden direccionar 2^{20}=1,048,576 localidades. La abreviación K (por kilo) es estandarizada para 2^{10}=1024, y por lo tanto 16 líneas pueden direccionar 2^6x2^{10}=64k localidades, mientras que 20 lineas pueden direccionar $2^{10}x2^{10}$=1024K (1Mega) localidades.

El **bus de datos** transporta información entre el CPU y la memoria o entre el CPU y dispositivos I/O (Entrada/Salida). Un intenso esfuerzo de investigación es utilizado en acortar el tiempo de ejecución de las actividades que realiza una computadora. Las computadoras utilizan dos terceras partes de su tiempo simplemente en mover datos. La mayoría de las operaciones de movimiento son entre los registros de CPU y las memorias RAM y ROM, por lo que el número de líneas o ancho del bus de datos es importante para el funcionamiento total. Esta limitación es un cuello de botella. Puede haber una vasta cantidad de memoria en el sistema, y el CPU puede poseer una tremenda capacidad de cómputo, pero el acceso de datos o el movimiento de datos entre la memoria y el CPU vía el bus de datos es el cuello de botella debido al ancho del bus de datos.

Esta característica es tan importante que es común agregar un prefijo que indica el cuello de botella ,es decir, el ancho del bus de datos. La frase "Computadora de 16 bits" se refiere a una computadora con 16 líneas en su bus de datos. Muchas computadoras caen en la clasificación de 4, 8, 16 o 32 bit, la capacidad de cómputo se incrementa conforme el ancho o número de líneas del bus de datos se incrementa.

En la figura 1.2.1 se muestra que el bus de datos es bidireccional, mientras que el bus de direcciones es unidireccional. La información de dirección siempre es proporcionada por el CPU, como es indicado por el sentido de la flecha de la figura 1.2.1., mientras que el dato puede viajar en ambas direcciones dependiendo si la operación es de lectura o escritura. Nota: el término dato es utilizado en sentido general, la información puede ser un dato o una instrucción usada por el programa.

El **bus de control** es una mezcla de señales, cada una con un rol específico en el control de la actividad del sistema. Como regla, las señales de control son señales de sincronización proporcionadas por el CPU hacia los circuitos externos para sincronizar el movimiento de información entre los buses de dirección y de datos.

Para el movimiento de datos entre el CPU y la memoria generalmente se consideran tres tipos de señales tales como: CLOCK (reloj), READ (lectura) y WRITE (escritura). Los nombres y operación de estas señales son altamente dependientes del CPU específico. Las hojas de datos del fabricante deben ser consultados para mayores detalles.

1.9 PUERTOS I/O

Un **PUERTO I/O** es un conjunto de terminales utilizadas para entrada (Input) o salida (Output) de datos en un microcontrolador. Es una práctica generalizada que los puertos estén formados por terminales que realizan funciones semejantes. Entre estas funciones se encuentran:

1. Entradas y/o salidas de propósito general
2. Operación con el sistema del timer (Temporizador)
3. Comunicación paralelo con Strobe (Control) simple
4. Comunicación paralelo con protocolo total (Full handshake)
5. Comunicación serie
6. Conversión análogo digital

Dependiendo de lo elaborado del microcontrolador, es posible que los puertos puedan realizar una o varias de las funciones mencionadas, lo cuál depende de diversos registros de control.

CAPITULO 2

EL MICROCONTROLADOR COMO CIRCUITO INTEGRADO.

2.1 INTRODUCCION.

Los microcontroladores son circuitos integrados LSI (Large Scale Integration) altamente desarrollados, cuyo diseño esta enfocado hacia actividades de control. En este capítulo serán descritos los subsistemas o bloques que comúnmente se encuentran dentro del chip o pastilla de un microcontrolador, además serán descritas las funciones que realiza cada una de sus terminales, y se hará una descripción de la organización de la memoria en el Modo de Operación Cargador.

Se utilizan a partir de este capítulo como prototipo, a los microcontroladores de la familia M68HC11 fabricados por Motorola, debido a que su conjunto de instrucciones es capaz de ejecutar todas las instrucciones *de las familias* M6800 y M6801 más 91 nuevos códigos de operación. Esto facilita entender el uso de microprocesadores fabricados con anterioridad por Motorola. No se ha utilizado para la descripción microcontroladores fabricados por INTEL, porque generalmente cada uno de sus dispositivos microcontroladores o microprocesadores, utilizan diferentes conjuntos de instrucciones, si bien cabe aclarar que sus caracteristicas y manera de operar son bastante semejantes.

2.2 CARACTERISTICAS GENERALES.

Las siguientes son algunas características relevantes de los microcontroladores de la familia M68HC11:

Características del Hardware:

- 12K bytes de ROM.
- 512 bytes de EEPROM.
- 512 bytes de RAM relocalizables.

- Sistema de Timer de 16 bits con: Preescalador programable de 4 etapas y la posibilidad de: a).- tres funciones de capturas de entrada/cinco de salidas comparadas o, b).- cuatro funciones de captura de entrada/cuatro de salidas comparadas seleccionables.
- Circuito Acumulador de pulsos de 8 bits.
- Interfaz de Comunicación Serie NRZ (No Return to Zero).
- Interfaz Periférico Serie.
- Convertidor Análogo-Digital de 8 bit, 8 canales.
- Circuito de interrupción de Tiempo Real.
- Sistema de Vigilancia de Operación Apropiada de Cómputo.
- Disponible en diferentes tipos de encapsulado.

Características del Software:

- Conjunto de instrucciones mejoradas respecto a M6800/M6801.
- Enteros 16 x 16 y características de División Fraccional.
- Manipulación de bit.
- Modo de Operación de ESPERA (WAIT).
- Modo de Operación de ALTO (STOP).

La familia de microcontroladores M68HC11 esta compuesta de varios miembros, en la figura 2.1 se explica como los números de parte de estos productos es construido.

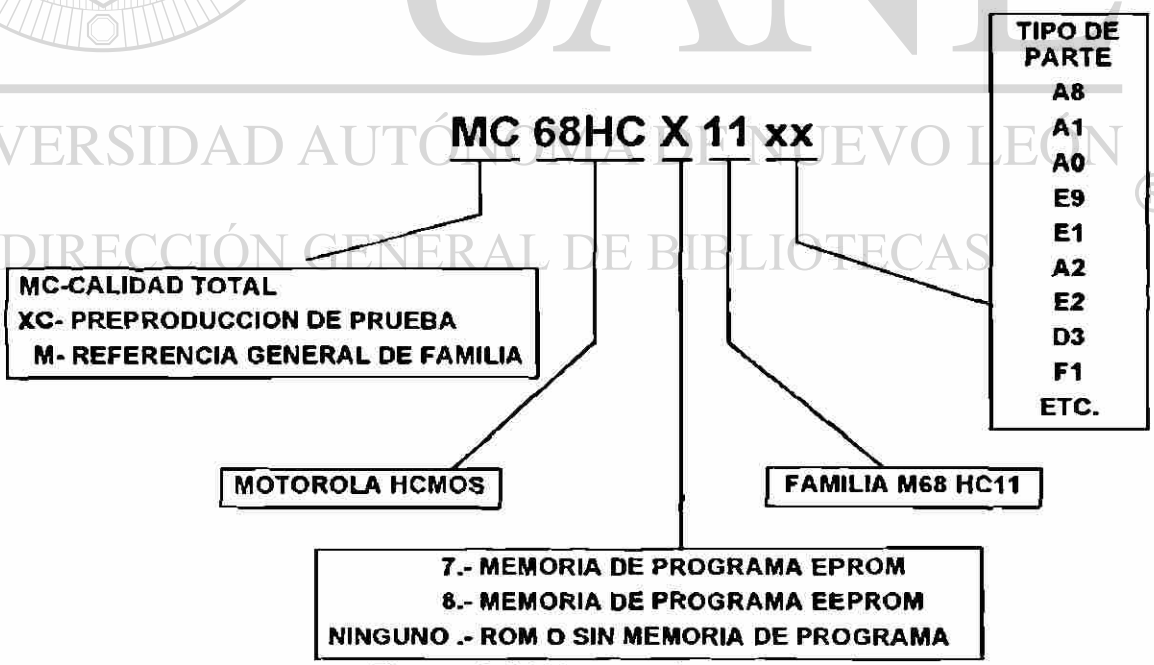


Figura 2.1 Números de parte

Los miembros de la familia M68HC11 difieren principalmente en el tipo y cantidad de memoria. En la tabla 2.1. se resumen las características. Note que los mayores cambios los tienen las variaciones x8, x9, x1, x0. Estas variaciones usan idéntico chip. El registro de configuración (CONFIG) es implementado con celdas EEPROM y es usado para semipermanentemente desactivar la ROM en la variación x1. La ROM y la EEPROM es desactivada en la variación x0.

Tabla 2.1 Microcontroladores de la Familia MC68HC11

No. Parte	EPROM	ROM	EEPROM	RAM	CONFIG 2	COMENTARIOS
MC68HC11A8	—	8K	512	256	$0F	Familia construida alrededor de este dispositivo.
MC68HC11A1	—	—	512	256	$0D	'A8 con ROM deshabilitada
MC68HC11A0	—	—	—	256	$0C	'A8 con ROM y EEPROM deshabilitada
MC68HC811A2	—	—	2K[1]	256	$FF	Sin ROM para sistema expandido
MC68HC811A8	—	—	8K+512	256	$0F	Emulador de EEPROM para A8
MC68HC11E9	—	12K	512	512	$0F	Cuatro entradas de captura
MC68HC11E1	—	—	512	512	$0D	E9 con ROM deshabilitada
MC68HC11E0	—	—	—	512	$0C	E9 con ROM y EEPROM deshabilitado
MC68HC811E2	—	—	2K[1]	256	$FF	Igual que A2 con Timer E9
MC68HC711E9	12K	—	512	512	$0F	
MC68HC11D3	—	4K	—	192	N/A	Versión de 40 terminales de bajo costo
MC68HC711D3	4K	—	—	192	N/A	Versión de D3 con un timer programado
MC68HC11F1	—	—	512[1]	1K	$FF	Alta funcionalidad

Notas::

1. La EEPROM es relocalizable en la parte alta de cualquier página de un bloque de memoria de 4K. La relocalización es hecha con los 4 bits superiores del registro CONFIG.
2. Los valores del registro CONFIG en esta tabla, reflejan el valor programado antes de ser enviado por Motorola

2.3 DESCRIPCION GENERAL DEL HARDWARE.

La tecnología CMOS de alta densidad (HCMOS = High-density Complementary Metal Oxide Semiconductor) es utilizada para producir microcontroladores avanzados altamente sofisticados y con capacidad periférica incluida en el chip, nuevas técnicas combinan pequeño tamaño y alta velocidad con bajo consumo de potencia e inmunidad al ruido. En la figura 2.3.1 se muestra un diagrama de bloques que muestran los subsistemas mayores que a continuación se describen.

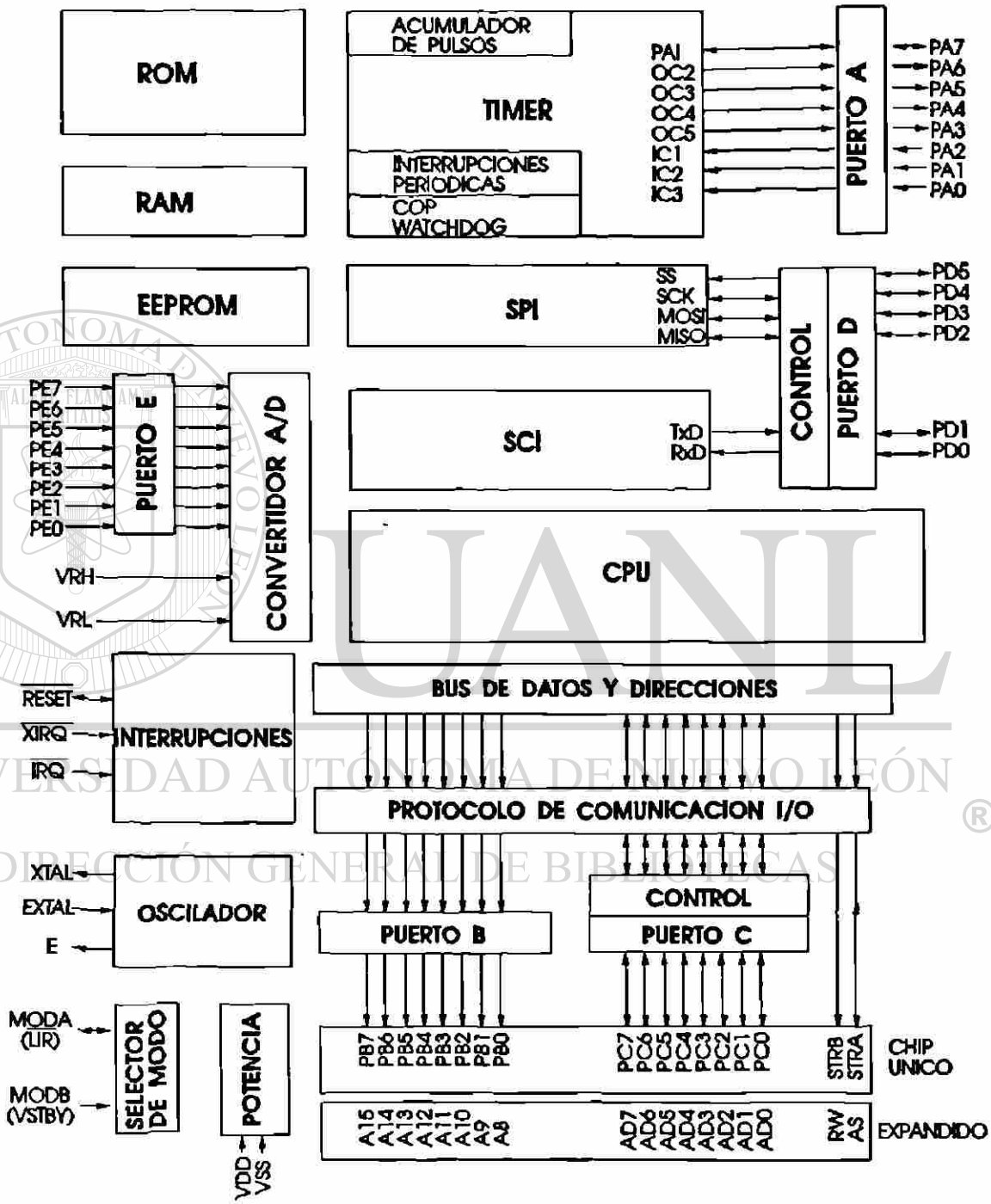


Figura 2.3.1. Diagrama de Bloques.

Memoria ROM.

La memoria ROM es programada por máscara en la fábrica y puede incluir hasta 12k de memoria. Es utilizada para almacenar permanentemente el programa del usuario, y contiene 192 bytes del Programa Cargador (bootstrap) para el Modo de Operación Cargador (Special Bootstrap).

Memoria RAM.

Son disponibles hasta 521 bytes de memoria RAM de celdas estáticas y pueden ser relocalizadas al inicio de cualquier bloque de memoria de 4k. Las aplicaciones de la memoria RAM son muy diversas y generalmente se utiliza para guardar resultados intermedios del proceso, o sea, como memoria de trabajo.

Memoria EEPROM.

Hasta 2k bytes de memoria EEPROM son disponibles en algunos miembros de la familia de microcontroladores M68HC11. La memoria EEPROM puede ser utilizada en la misma forma que la memoria ROM, pero algunas interesantes posibilidades se alcanzan que no son posibles con memorias RAM y ROM. Una vez que la información es programada en la memoria EEPROM incluida en el chip , esta permanece sin cambio aunque la fuente de energía sea removida indefinidamente. A diferencia de la información en ROM, la información en la EEPROM puede ser borrada y reprogramada bajo del control de las instrucciones del software.

Convertidor Análogo-Digital.

Un convertidor Análogo-Digital de ocho canales con una resolución de ocho bits es incluido. El sistema de conversión usa la técnica de redistribución de carga totalmente capacitivo. Dos líneas VRL y VRH son proporcionadas para las entradas de voltaje de referencia, lo cual permite aceptar entradas análogas en el rango de VRL a VRH. Pequeños rangos de entradas análogas pueden ser obtenidos ajustando VRL y VRH a los deseados límites superior e inferior.

Interfaz de Comunicación Serie.

El microcontrolador contiene una Interfaz de Comunicación Serie Asíncrono SCI (Serial Communications Interface) que funciona con un formato standard NRZ (no retorno a cero) con un bit de arranque, ocho o nueve bits de datos y un bit de stop. Una variedad de velocidades de transmisión o baud rate son disponibles, esto permite mediante esta interfaz, que se hagan conexiones a terminales de tubos de rayos catódicos, computadoras personales, etc.

Interfaz de Comunicación Serie Síncrono.

Este subsistema también conocido como Interfaz Periférico Serie ó **SPI** (Serie Peripheral Interface), como su nombre implica, es primordialmente usado para permitir que el microcontrolador se comunique con dispositivos periféricos. El SPI es capaz de realizar la comunicación con otros procesadores en un sistema maestro múltiple. Los dispositivos periféricos pueden ser tan simples como los registros de corrimiento **TTL** (Transistor Transitor Logic) o tan complejos como subsistemas completos, tales como driver de display, diodos de cristal líquido o subsistemas convertidores análogo ó digital. El sistema SPI es lo suficiente flexible como interfaz directo con numerosos productos standard periféricos de varios fabricantes.

Timer (Temporizador).

El sistema Completo del Timer Principal incluye varias secciones:

a. Un contador de 16 bit continuo (free-running) con un prescalador programable de cuatro etapas que es utilizado para cuenta de eventos o para generar retardos de tiempo por software.
b. Un Circuito de Interrupción Periódico Programable también llamado Interruptor de Tiempo Real, el cual es comúnmente usado para dar la pauta para la ejecución de subrutinas.
c. Sistema de Vigilancia de Apropiada Operación de Cómputo (Computer Operating Properly [**COP**] Watchdog System) el cual es utilizado para detectar errores de procesamiento de software.
d. Acumulador de Pulsos que se usa para: la cuenta de eventos, medición de frecuencias, interruptor de transición de pulsos, etc.
e. Generador de Baud rate para establecer la rapidez de comunicación serie.

Oscilador.

Este subsistema genera las señales de reloj que controlan la operación del microcontrolador. Las señales de reloj dependerán de la frecuencia del cristal utilizado. Además contiene un sistema de monitoreo de la señal de reloj para resetear el microcontrolador si ésta es perdida o corre demasiado lenta.

Entradas/Salidas.

Este subsistema de comunicación I/O (Input/Output) o de Entrada/Salida tiene un total de 40 terminales repartidas en 5 puertos y que junto con las líneas de control llamadas strobe, permiten la comunicación (lectura y escritura) con el exterior.

CPU.

La **CPU** (Central Processing Unit) o Unidad de Procesamiento Central, como fue explicado anteriormente, es el "cerebro" que administra todas las actividades en el sistema y realiza todas las operaciones con los datos.

2.4 CONEXIONES O TERMINALES.

En esta sección ampliaremos el conocimiento de la arquitectura del hardware desde una perspectiva externa, es decir, desde sus terminales. Una breve descripción de la función que realiza cada terminal es presentada, dejando para después en los capítulos correspondientes, el hacer una explicación más detallada. Desde esta sección conviene acostumbrarse a utilizar las siglas en inglés que representan: terminales, registros, secciones, elementos, dispositivos, etc. Por ejemplo: **MCU** (Micro Controller Unit) se utilizará en lugar de microcontrolador.

Como prototipo para la descripción de las terminales se ha elegido al microcontrolador MC68HC11A8 y en la *figura* 2.4.1 se muestra la asignación de las terminales para el encapsulado **PLCC** (Plastic Leaded Chip Carrier) de 52 terminales y el encapsulado **DIP** (Dual In-line Package) de 48 terminales. La diferencia es que en el encapsulado DIP de 48 terminales, 4 terminales del Convertidor Análogo-Digital no salen al exterior. A continuación se describen las terminales.

VDD=Entrada de Voltaje y VSS=Tierra.

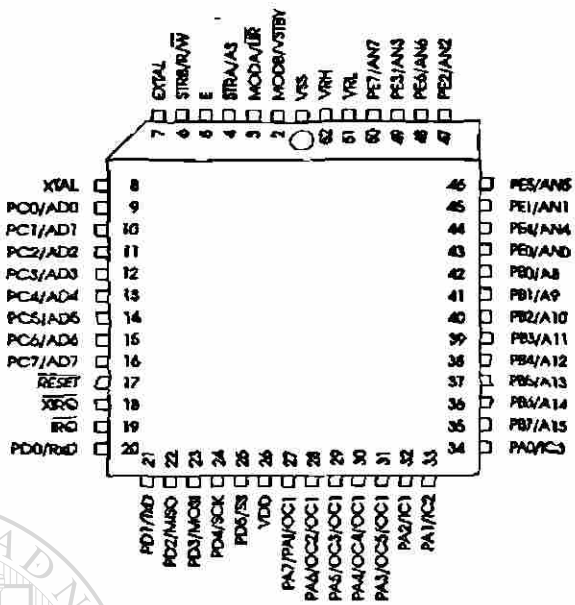
El suministro de Voltaje al MCU es realizado usando estas terminales. VDD es la entrada positiva y VSS es tierra.

RESET.

Esta terminal de control bidireccional activa en baja, es decir, cuando la señal de entrada es cero, es usada como entrada al MCU para inicializarlo en un estado de arranque conocido y actúa como salida de drenador abierto para indicar que una falla interna ha sido detectada, ya sea por el monitor del reloj o por el Sistema de Vigilancia de Operación Apropiada de Cómputo (Computer Operating Properly (**COP**) Watchdog System).

XTAL= Entrada de Cristal y EXTAL= Entrada de Reloj Externo.

Estas dos entradas proporcionan la interfaz para la conexión del cristal o para una señal externa de reloj compatible CMOS y controlan el circuito interno generador de la señal de reloj. La frecuencia aplicada a estas terminales deben ser cuatro veces la **señal de reloj E** deseada.



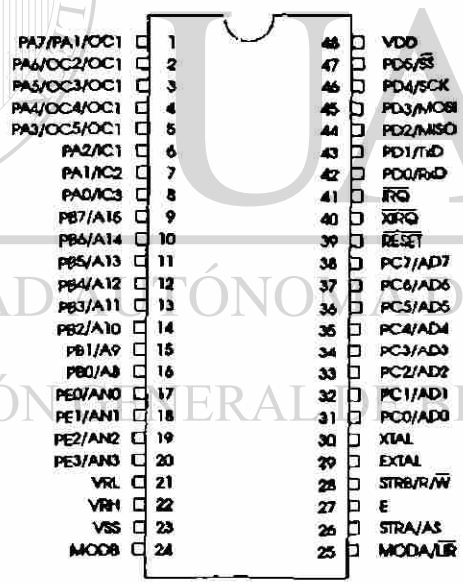


Figura 2.4.1. Asignación de terminales MC68HC11A8

E = Salida de Reloj E.

Esta es la conexión de salida para la señal de reloj E generada internamente, la cual es usada como referencia de tiempo o de sincronización. Cuando la salida de reloj E es baja, un proceso interno se está realizando, y cuando es alta, el dato esta siendo accesado. La señal de reloj E es detenida cuando el MCU esta en modo STOP.

IRQ = Interrup Request

La entrada IRQ o de Requisición de Interrupción proporciona un medio para requerir una interrupción asíncrona para el MCU. La entrada IRQ puede ser programada por el Registro OPTION para que actúe siendo sensible al nivel o siendo sensible a la transición de caída de la señal. Después de reset, IRQ es configurada por default u omisión para operación sensible al nivel.

XIRQ = Non-maskable Interrup Request

La terminal XIRQ o Interrupción No Enmascarable proporciona un medio para requerir interrupciones no enmascarables después de inicialización por reset. Durante reset, el bit X es activado en el Registro del Código de Condición CCR (Condition Code Register), y cualquier interrupción es enmascarada hasta que el software del MCU la habilita. Dado que la entrada XIRQ es sensible al nivel, esta puede ser conectada a una red OR con entradas múltiples con una resistencia de pull-up externa. La terminal XIRQ es generalmente usada como interrupción que detecta pérdida de señal.

MODA/LIR y MODB/VSTBY.

Las terminales MODA/LIR y MODB/VSTBY realizan funciones alternas. Solamente durante el reset, las terminales MODA y MODB son usadas para seleccionar uno de cuatro modos de operación del MCU de acuerdo a la tabla 2.4.1.

Tabla 2.4.1 Selección de Modos de Operación.

MODB	MODA	Modo Seleccionado
0	0	Cargador.
0	1	Prueba.
1	0	Chip Unico.
1	1	Multiplexado Expandido.

En este curso, solamente el Modo de Operación Cargador (Special Bootstrap) es utilizado, pues permite cargar en el MCU los programas del usuario para su verificación, corrección y puesta en operación. En la figura

4.6.4.1. del capítulo 4 se muestra un circuito que puede ser utilizado para operar el MCU en este modo de operación.

Después de que el modo de operación ha sido seleccionado, entonces la terminal MODA asume la función alterna LIR(Load Instruction Register) la cuál proporciona una salida de drenador *abierto para indicar que una* instrucción esta iniciándose. Todas las instrucciones son realizadas durante una serie de ciclos de reloj E. La señal LIR va hacia abajo durante el primer *ciclo de reloj* E de cada instrucción (reconocimiento del ciclo de operación). Esta salida se proporciona como una ayuda para la depuración o debug del programa.

Después de que el modo de operación ha sido seleccionado, entonces la terminal MODB asume la función alterna VSTBY que se usa como entrada para potencia de sostenimiento o standby de la RAM, *para* que su contenido no se pierda cuando VDD no está presente.

VRH-VRL Voltajes de Referencia para el Convertidor Análogo-Digital.

Estas dos terminales proporcional el voltaje de referencia para el circuito Convertidor Análogo-Digital.

STRB - R/W (Strobe B y Read/Write).

En el Modo de Operación Cargador, la terminal STRB (Strobe B) actúa como salida de control programable para Comunicación Paralelo con otros dispositivos externos.

STRA/AS (Strobe A - Address Strobe).

En el Modo de Operación Cargador, esta terminal actúa como entrada de control programable para Comunicación Paralelo con otros dispositivos externos

PUERTO A.

El puerto A puede ser configurado para:

a. Cuatro funciones de captura de entradas (IC1, IC2, IC3, IC4) y tres funciones de salida comparada (OC2, OC3, OC4), y ya sea una para una entrada acumuladora de pulsos o bien otra salida comparada (OC1).

b. Tres funciones de captura de entrada (IC1, IC2, IC3), y cuatro funciones de salida comparada (OC2, OC3, OC4, OC5), y ya sea para una entrada acumuladora de pulsos o bien otra salida comparada (OC1).

Cualquier terminal del Puerto A que no sea usada para su función alterna del TIMER puede ser usada como línea de entrada o salida de *propósito general.*

PUERTO B.

En el Modo de Operación Cargador (Bootstrap), todas las terminales del Puerto B funcionan como terminales de salida de propósito general. Durante lectura a este puerto, el nivel sensado en el lado de entrada de los drivers de salida del puerto B es leído. El puerto B también puede ser usado en el modo de salida de control (strobe) simple, ocasionando que un pulso de salida aparezca en la terminal STRB cada vez que un dato es escrito en el Puerto B.

PUERTO C.

En el Modo de Operación Cargador, las terminales del Puerto C pueden realizar funciones alternas que incluyen:

a. Entradas/Salidas de Propósito General.
b. Comunicación Paralelo con Control (Strobe) Simple.
c. Comunicación Paralelo con Protocolo Total.

PUERTO D.

Las terminales 0 a 5 del puerto D son usadas como señales I/O (Input/Output = Entrada/Salida) de propósito general. Las terminales del puerto D alternativamente sirven como Interfaz de Comunicación Serie **SCI** (Serial Communications Interface) o como Interfaz Periférica Serie **SPI** (Serial Perifheral Interface) cuando esos subsistemas son habilitados.

La terminal PD0 es la señal receptora de datos **RxD** (Receive Data) para SCI. La terminal PD1 es la señal de salida de transmisión de datos **TxD** (Transmit Data) para SCI. Las terminales PD2 hasta PD5 son dedicadas a SPI. Las terminales PD2 es para la señal **MISO** (Master-In-Slave-Out).

2.5 MODOS DE OPERACION.

Como se mencionó anteriormente, mediante las terminales MODA y MODB se pueden seleccionar cuatro modos de operación para los MCU's de la familia M68HC11, los cuáles son:

a. Cargador (Special Bootstrap)
b. Prueba (Special test)
c. Chip Unico (Single Chip)
d. Multiplexado Expandido (Expanded Multiplexed)

Modo de Operación Cargador (Special Bootstrap)

En este curso solamente este modo de operación es utilizado, pues permite cargar en el MCU los programas del usuario para su verificación, corrección y puesta en operación.

Cuando el MCU es reseteado en el Modo de Operación Cargador, una parte de la ROM de 192 byte que ocupa desde la dirección $BF40 a $BFFF, es habilitada en el chip y el MCU procede a ejecutar el Programa Cargador contenido en esa área. El Programa Cargador que está instalado permanentemente en la ROM inicializa el sistema SCI (Serial Communications Interface), checa opciones de seguridad y acepta que dependiendo del tipo de MCU sean almacenados hasta 512 byte en las localidades $0000 a $01FF. Después de que el byte final es recibido, el control es automáticamente transferido al programa que se inicializa en la localidad $0000.

El Modo de Operación Cargador es muy versátil, dado que esencialmente no hay limitaciones en el tipo de programa del usuario que puede ser cargado en la RAM interna del MCU.

Modo de Operación Prueba (Special Test).

Este modo de operación es primordialmente intentado para pruebas de fábrica . Este modo de prueba no es recomendado para ser utilizado por el usuario debido al reducido sistema de seguridad.

Modo de Operación Chip Unico (Single Chip)

En el Modo de Operación Chip Unico, el Puerto B y el Puerto C funcionan como I/O (Entradas/Salidas) de propósito general o para comunicación paralelo con protocolo de comunicación, donde las terminales strobe A y strobe B proporcionan las señales de control.

Modo de Operación Multiplexado Expandido.

En este modo de operación el MCU tiene capacidad de accesar un espacio de direcciones de 64 k byte. Este espacio total de direcciones incluye la memoria incluida dentro del chip usada en el Modo de Operación Chip Unico más la posibilidad de direccionar localidades para accesar dispositivos periféricos y memoria externa. La expansión del bus de direcciones es realizada utilizando el puerto B y el puerto C y las señales de control AS (Address Strobe) y R/W (Read/Write).

2.6. ORGANIZACION DE LA MEMORIA.

La organización de la memoria para los modos de operación Chip Unico y Cargador para el MC68HC11 es mostrada en la figura 2.6.1. Las localidades de memoria interna son mostradas sombreadas y su contenido es indicado a la derecha. Las localidades marcadas EXT no son utilizadas en estos modos de operación.

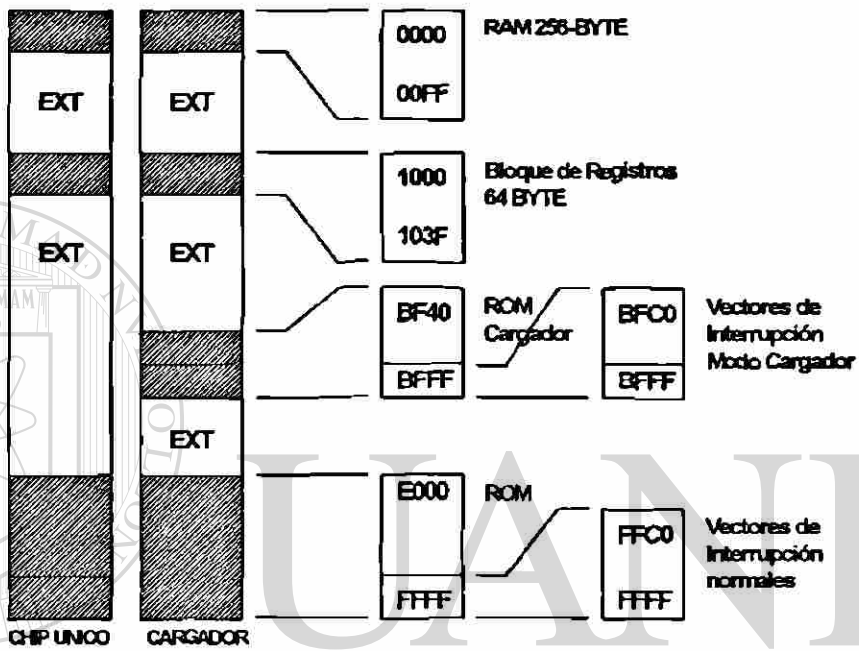


Fig. 2.6.1 MAPA DE MEMORIA EN EL MODO CHIP UNICO Y CARGADOR

En la organización de la memoria de la figura 2.6.1 se observan las siguientes secciones: RAM, EXT, Bloque de Registros, EEPROM, ROM del Programa Cargador y ROM del Programa del Usuario. A continuación se describen estas secciones.

2.6.1. RAM

La RAM interna del microcontrolador MC68HC11A8 consta de 256 byte y después de resetear el MCU empieza en la localidad $0000. Esta memoria es utilizada como memoria de trabajo para almacenar programas y resultados intermedios o finales de un proceso. Está implementada con celdas estáticas que pierden su información cuando se desconecta la tensión V_{DD}. Para mantener la información se requiere una pequeña corriente en la terminal MODB/V_{STBY}.

2.6.2. EXT

Las localidades marcadas EXT son utilizadas para direccionar memoria externa o dispositivos periféricos externos cuando el MCU está en el Modo de Operación Multiplexado Expandido o de Prueba. Las localidades marcadas EXT **no son utilizadas** en el modo de operación Cargador ni en Chip Unico y direccionar hacia esas localidades puede producir resultados impredecibles

2.6.3. BLOQUE DE REGISTROS.

Los Puertos y Registros de Control en los microcontroladores de la familia M68HC11 están caracterizados por estar localizados en el mapa de memoria y pueden ser direccionados de la misma forma que cualquier localidad de memoria. El Bloque de Registros está constituido con los puertos y los Registros de Control y forma un bloque de 64 localidades de memoria que ocupa desde la dirección $1000 hasta la $103F después de resetear el MCU en el Modo de Operación Cargador. En la tabla 5.2.1 del capítulo 5 se muestra la ubicación de los Puertos y los Registros de Control.

2.6.4. EEPROM.

El MC68HC11A8 incluye en su interior 512 byte de memoria EEPROM del tipo CMOS, y ocupa de la dirección $B600 hasta la $B7FF. Esta memoria es usada en la misma forma que la memoria ROM, pero tiene algunas interesantes posibilidades que no son posibles con memorias ROM y RAM. Una vez que la información es programada en la memoria EEPROM del MCU, esta permanece sin cambio aunque la fuente V_{DD} sea removida indefinidamente. Y a diferencia de la memoria ROM, la información en la EEPROM puede ser borrada y reprogramada mediante el control del software. La EEPROM puede ser borrada y reprogramada mediante un circuito interno alimentado por la fuente V_{DD}, por lo que no es necesaria ninguna fuente de energía especial.

2.6.5. ROM.

La ROM es una memoria de solo lectura y en el chip consta de dos partes: A).- La ROM del Programa Cargador . B).- La ROM del Programa del Usuario.

La **ROM del Programa Cargador** consta de 192 byte que ocupan de la dirección $BF40 hasta la $BFFF. Esta ROM contiene el Programa Cargador que controla el proceso de carga en el Modo de Operación Cargador (Special Bootstrap).

La ROM del Programa del Usuario es utilizada primordialmente para contener las instrucciones del programa de aplicación del usuario y no pueden ser cambiadas porque son *programadas* en el MCU cuando es fabricado. El usuario debe desarrollar y depurar sus programas de aplicación antes de ordenar la producción de MCU's. El MC68HC11A8 dispone de 8 Kbyte de ROM ubicados desde la localidad $E000 hasta la $FFFF.

CAPITULO 3

CONJUNTO DE INSTRUCCIONES

3.1. INTRODUCCION

En este capítulo se hace una descripción del Modelo del Programador, es decir, de los registros internos de CPU que intervienen en la ejecución de las instrucciones del MCU. También se incluye una descripción de los diferentes Modos de Direccionamiento, o sea, la manera en que son localizados los datos para ser manipulados por el CPU.

El Conjunto de Instrucciones completo que pueden ejecutar los MCU's de la familia M68HC11 es presentado en el Apéndice, y no es necesario analizarlo completo para entenderlo en su totalidad. Por esa razón solo es examinado el número suficiente de instrucciones con todos los modos de direccionamiento aplicables expresados brevemente, además son listados los códigos de operación para cada modo de direccionamiento. Muchos ejemplos son presentados para ilustrar el contenido de los registros del CPU y las localidades de memoria aplicables antes y después de la ejecución de una instrucción.

3.2. MODELO DEL PROGRAMADOR.

En la fig. 3.2.1. se observa una vista simplificada del interior del CPU. El CPU es "el cerebro" del MCU , administra todas las actividades en el sistema, y además realiza todas las operaciones con los datos.

La CPU es una colección de circuitos lógicos que realizan en forma contínua solo dos operaciones: buscar instrucciones (fetch instructions) y ejecutar instrucciones (executing instructions).

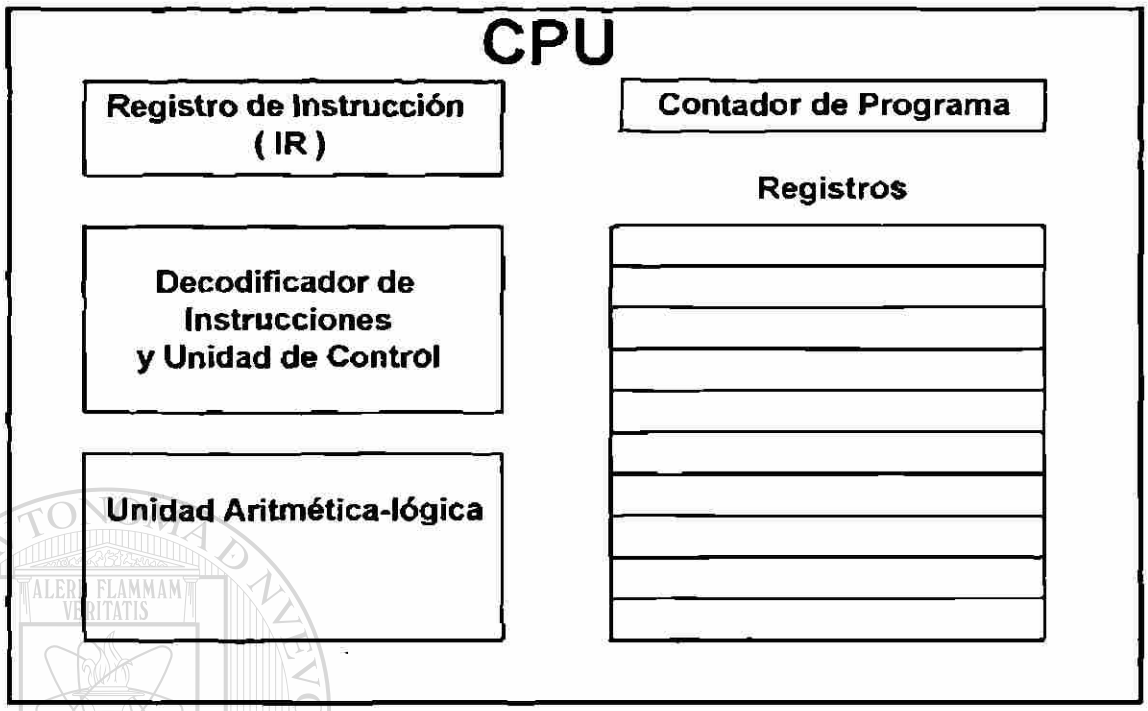


Figura 3.2.1. Modelo del Programador

El **buscar** (fetch) una instrucción desde la RAM involucra los siguientes pasos:

a. El contenido del Contador del Programa PC (Program Counter) es colocado en el bus de direcciones.
b. Una señal de control de lectura es activada.
c. Un dato o un código de operación de una instrucción es leído de la RAM y es colocado en el Bus de datos.
d. El Código de Operación **Opcode** (**Operation Code**) es fijado en el Registro de instrucciones **IR** (Instruction Register).
e. El *Contador de Programa* **PC** (*Program Counter*) es incrementado para preparar la siguiente búsqueda de instrucción desde la memoria.

La **Ejecución** de una instrucción involucra:

a. Decodificar, es decir, descifrar el Código de Operación.
b. Generar señales de Control.
c. Activar registros internos, entradas y salidas de la Unidad Aritmética - Lógica **ALU**.
d. Señalar a la Unidad Aritmética Lógica ALU que realice una operación específica (sumar, restar, etc.)

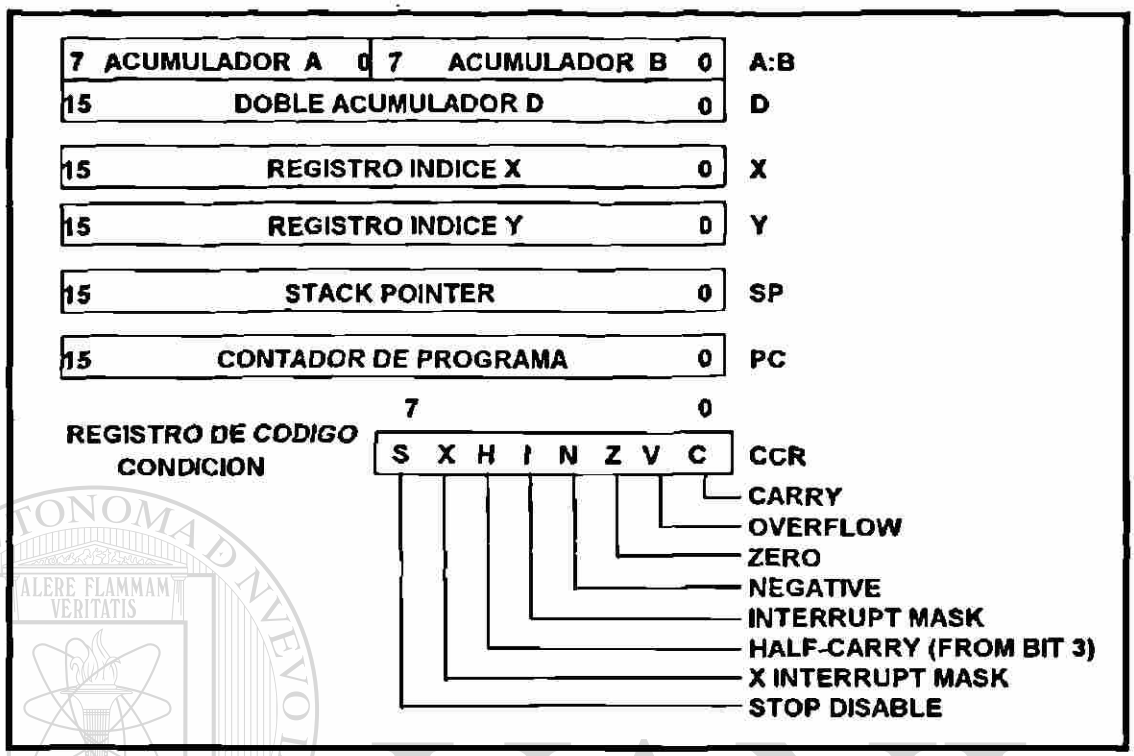


Figura 3.2.2. M68HC11 Modelo del programador

Dada la arquitectura de este CPU, todos los puertos I/O (Entradas/Salidas) y las localidades de memoria son tratados en forma idéntica en el mapa de memoria de 64K bytes. En cambio, los registros del CPU forman parte integral del mismo y no son direccionables, es decir, *no pueden ser* tratados como localidades de memoria. El **Modelo del Programador** es el *nombre dado* al conjunto de registros internos a que tiene acceso el programador. En la figura 3.2.2. se observa el conjunto de registros internos que forman parte del modelo del Programador, los cuales se listan y describen enseguida:

1. Acumulador A, B y D.
2. Registros Indice X e Y.
3. Apuntador de Pila **SP** (Stack Pointer).
4. Contador de Programa **PC** (Program Counter).
5. Registro del Código de Condición **CCR** (Condition Code Register)

Acumuladores A, B y D. Los acumuladores A y B son registros de 8 bits de propósito general usados para mantener los operandos y los resultados de cálculos aritméticos o de manipulación de datos. Los dos acumuladores pueden ser concatenados en un solo acumulador de 16 bit llamado Acumulador D.

Los Registros Indice X e Y son de 16 bits, y son usados para el Modo de Direccionamiento Indexado. Este modo de direccionamiento es usado para el manejo de Tablas.

El Apuntador de Pila SP (Stack Pointer) es un registro de 16 bits que contiene la dirección de la siguiente localidad libre en el stack (pila). El stack es configurado como un registros de lectura/escritura de secuencia **LIFO** (Last-In- First-Out), en este tipo de memoria, el último dato en entrar es el primero en salir. En el Stack Pointer son almacenados datos importantes durante interrupciones o llamadas a subrutina.

Es sumamente importante que el programador en las primeras instrucciones del Programa apunte el Stack Pointer a una área utilizable de la RAM con suficiente espacio.

El Contador de Programa PC (Program Counter) es un registro de 16 bit que apunta siempre a la localidad de la siguiente instrucción que va a ser ejecutada.

El Registro de Código de Condiciones CCR (Condition Code Register) también recibe el nombre de **Registro de Banderas**, pues se les da el nombre de **Banderas** (flags) a los bits contenidos en el registro de condiciones. El registro CCR tiene: 5 indicadores de status (H, N, Z, V, C), 2 bits de interrupciones (I,X) y un bit habilitador de STOP (S). En seguida se describe cada una de las banderas.

S stop disable.- (deshabilitador de STOP).- El bit S es controlado por medio de Software. Cuando es 1 se deshabilita la instrucción STOP, cuando es 0 la instrucción STOP es habilitada.

H half carry.- Este bit de cargo intermedio es activado o puesto en 1, cuando un cargo (carry) ocurre entre el bit 3 y 4 en la Unidad Aritmética Lógica, y es usado para corrección en operaciones aritméticas BCD.

I interrupt mask .- El bit I de esta máscara de interrupción es activado o puesto en 1 por medio de hardware, y por medio de una instrucción de programa se deshabilitan o enmascaran todas las fuentes de interrupción enmascarables.

Z zero.- El bit cero es activado si el resultado de la última operación (aritmética, lógica o al manipular datos) es cero.

C carry/borrow.- El bit C es activado cuando un cargo o préstamo ocurre en una operación aritmética. El bit C también es afectado durante instrucciones de corrimiento o rotación.

3.3. MODOS DE DIRECCIONAMIENTO

Los **Modos de Direccionamiento** son las diferentes formas en que se especifican el origen o destino de los datos que son manipulados por la CPU. Los MCU's de la familia M68HC11 pueden usar 6 modos de direccionamiento:

1. Inmediato
2. Directo
3. Extendido
4. Indexado
5. Inherente o Implícito
6. Relativo

En los ejemplos de los modos de direccionamiento se aplica lo siguiente:

a. Cuando está presente el símbolo #, indica direccionamiento inmediato.

b. Cuando no está presente el símbolo #, el número siguiente al código de operación es una dirección, excepto en el modo de direccionamiento Inherente.

c. El símbolo $ indica que el número que le sigue es hexadecimal.

En el **Modo de Direccionamiento inmediato**, la instrucción contiene los datos en el byte o bytes que están inmediatamente enseguida del código de operación.

Ejemplo del Modo de Direccionamiento Inmediato:

LDAA #$25 cargar el acumulador A, con el dato 25 en hexadecimal que se ubica inmediatamente en la siguiente localidad de memoria

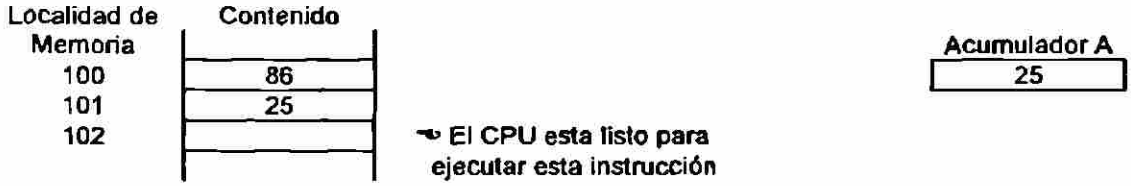
Antes de ejecutar la instrucción: LDAA #$25

Localidad de Memoria	Contenido
100	86
101	25
102	

☜ El CPU esta listo para ejecutar esta instrucción

Acumulador A
00

Después de ejecutar la instrucción: LDAA #$25

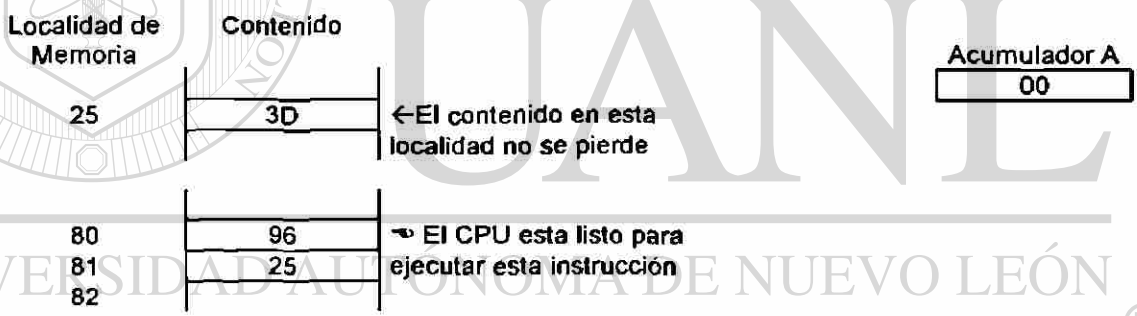


En el **Modo de Direccionamiento Directo** también llamado Direccionamiento de Página Cero, el byte que sigue al código de operación contiene la dirección de la localidad de memoria donde se encuentra el dato; esto es, en los primeros 256 bytes, desde $00 a $FF. En la dirección antes mencionada solo se especifica el byte menos significativo, pues el byte más significativo se asume igual a $00.

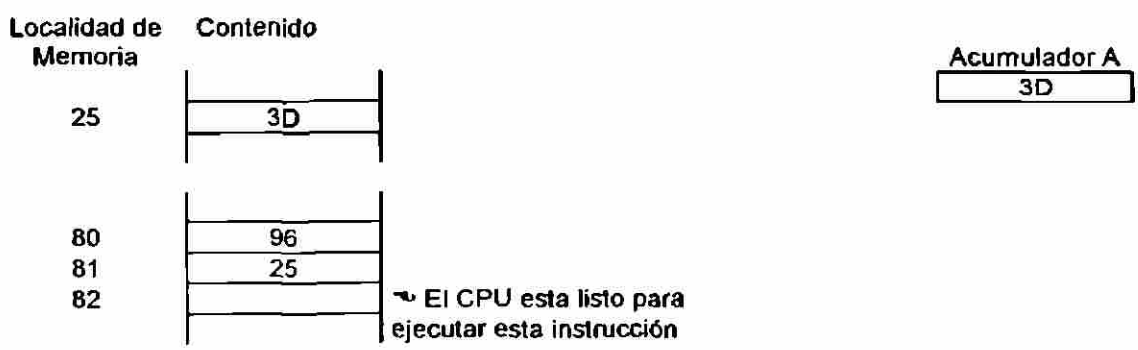
Ejemplo del Modo de Direccionamiento Directo:

LDAA $25 Cargar el acumulador A con el contenido de la localidad de memoria 25 en hexadecimal.

Antes de ejecutar la instrucción: LDAA $25



Después de ejecutar la instrucción: LDAA $25

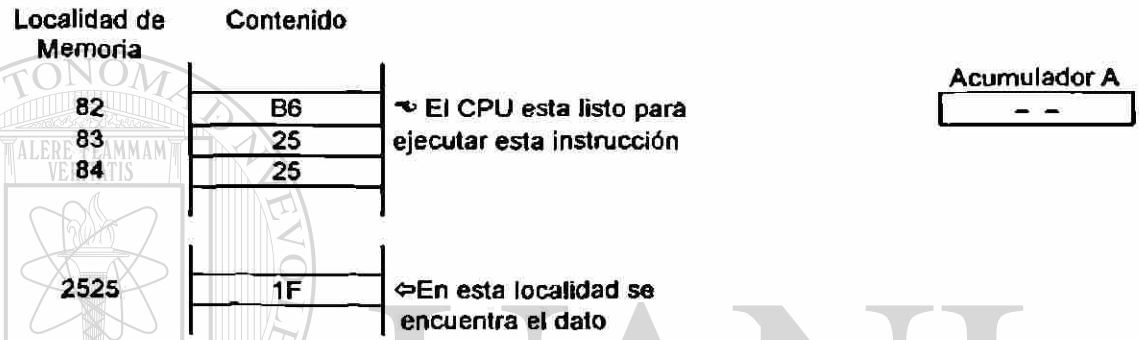


En el **Modo de Direccionamiento Extendido** el segundo y tercer byte enseguida del código de operación contienen la dirección del operando (dato). En este modo de operación se requieren dos bytes para establecer la dirección que puede ser cualquiera de los 64k bytes de memoria.

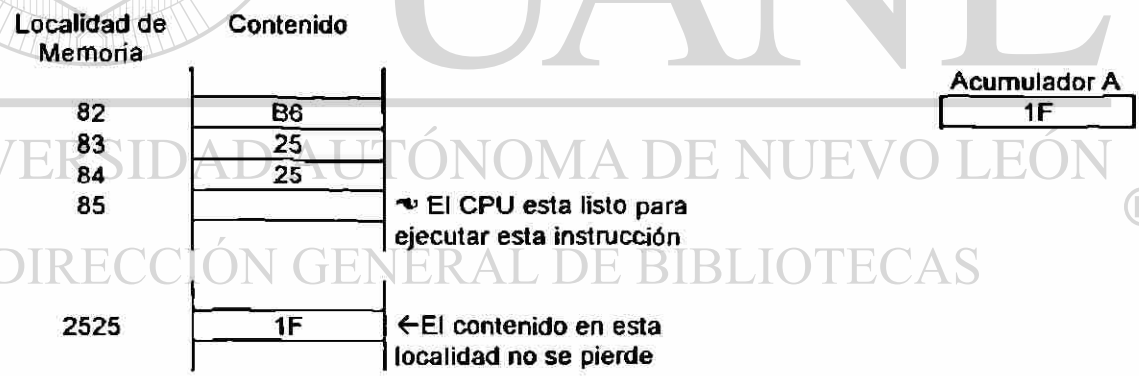
Ejemplo del Modo de Direccionamiento Extendido:

LDAA $2525 cargar el acumulador A con el contenido de la localidad de memoria 2525 hexadecimal.

Antes de ejecutar la instrucción: LDAA $2525



Después de ejecutar la instrucción: LDAA $2525



En el **Modo de Direccionamiento Indexado** uno de los registros índice X o Y es usado para calcular la dirección efectiva. **La dirección efectiva** es un valor de dos bytes, que es generado internamente en cada modo de direccionamiento (excepto en el inherente) para establecer la localidad origen o destino de los datos. En este modo de direccionamiento la dirección efectiva es variable y depende de 2 factores:

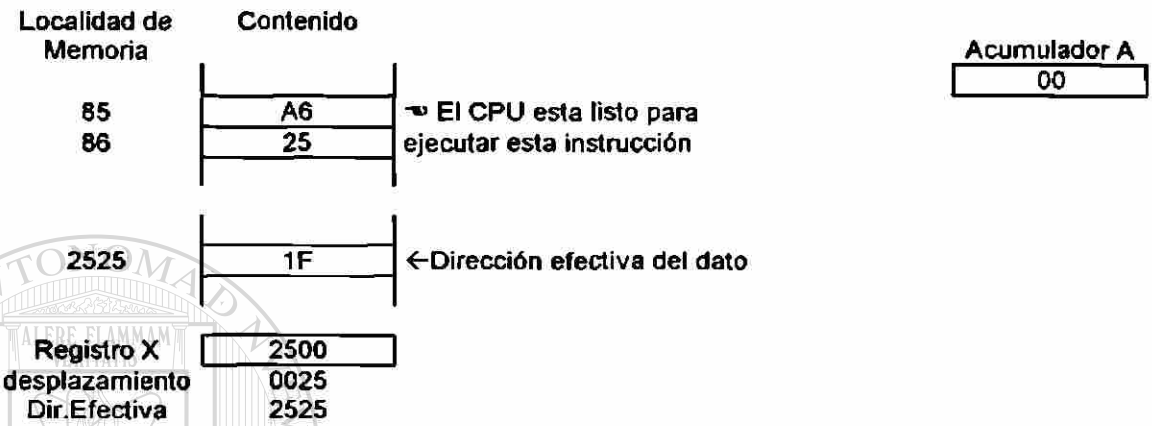
1. El contenido actual del registro índice (X o Y) que está siendo usado.

2. Los 8 bits de desplazamiento (offset) contenidos en la instrucción.

Ejemplo del Modo de Direccionamiento Indexado.

LDAA $25,X cargar el acumulador A, con el contenido de la localidad de la dirección efectiva obtenida al sumar 25 hex al contenido del Registro Indice X.

Antes de ejecutar la instrucción: LDAA $25,X



Después de ejecutar la instrucción: LDAA $25,X

El **Modo de Direccionamiento Inherente o Implícito** se utiliza con instrucciones que involucran exclusivamente los registros internos de MCU y está caracterizado por no requerir operandos, pues la instrucción contiene toda la información necesaria.

El Modo de Direccionamiento Relativo es utilizado en instrucciones de salto o bifurcación condicional. El salto solo puede ejecutarse desde -127 a +128 bytes relativos (con respecto) a la instrucción de bifurcación. En el siguiente capítulo veremos como el programa Ensamblador, realiza en forma totalmente transparente para el programador, el cálculo de localidades necesarias de salto.

3.4. CONJUNTO DE INSTRUCCIONES.

El **Conjunto de Instrucciones** es el grupo de instrucciones que un microprocesador o microcontrolador específico puede ejecutar. El Conjunto de *instrucciones* no están estandarizados debido al individualismo de cada fabricante, y a la diferencia de arquitectura y uso de cada dispositivo. En ésta sección será revisado el Conjunto de instrucciones para los MCU's de la familia M68HC11.

Un número suficiente de instrucciones es analizado, lo cual permitirá entender la totalidad del Conjunto de Instrucciones. Todos *los modos de* direccionamiento para esas instrucciones es presentado brevemente y el código de operación **hex** (abreviación de hexadecimal) para cada modo de direccionamiento es listado.

Todas las direcciones y ejemplos utilizados incluyen registros y contenidos de memoria que han sido seleccionados al azar y son mostrados para propósitos ilustrativos solamente.

3.4.1. NOMENCLATURA.

En la explicación de la función que realizan las instrucciones se hace uso de la siguiente nomenclatura:

1 *Los Registros se representan con las siguientes letras:*

A = Acumulador A
B = Acumulador B.
D = Doble Acumulador: Acumulador A concatenado con Acumulador B
CCR = Registro del Código de Condición.
X = Registro Indice X
Y = Registro Indice Y
PC = Contador de Programa (Program Counter).
SP = Apuntador de Pila (Stack Pointer)
M = Localidad de Memoria.

2 En el Registro del Código de Condición **CCR** (Condition Code Register) se usa la siguiente nomenclatura para indicar la actividad de los bits: S, X, H, I, N, Z, V, C.

\- = bit no afectado o no importa su valor si proviene de una instrucción anterior.
0 = bit forzado a cero.
1 = bit forzado a uno.

⇳ = bit es puesto en uno o en cero de acuerdo al resultado
⇩ = el bit puede cambiar de uno a cero, permanece en uno, pero no puede cambiar de cero a uno.

3 Todos los números utilizados son hexadecimales.

4 Un símbolo como éste ☜ indica la siguiente instrucción que será ejecutada por la CPU.

5 Una flecha como ésta ⇦ indica que "es colocado en" o "transferido a".

6 Los paréntesis son usados para indicar "el contenido de". Por ejemplo: (B) significa el contenido del acumulador B; (2720 hex) significa el contenido de la localidad de memoria 2720 hex.

7 Las letras mayúsculas entre comillas que aparecen al inicio de la explicación de una instrucción corresponden a su mnemónico. Los ***mnemónicos*** son abreviaturas o siglas de palabras (en inglés) que facilitan el recordar la función que realiza una instrucción y se utilizan para aproximar la lectura y escritura de programas al lenguaje habitual (en inglés).

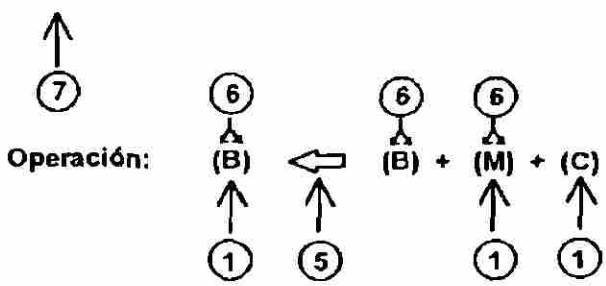
8 Cuando está presente el símbolo # indica direccionamiento *inmediato*. Cuando no está presente el símbolo #, el número siguiente al código de operación es una dirección.

9 El símbolo $ indica que el número siguiente es hexadecimal.

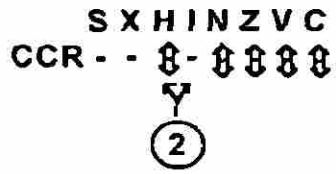
En el ejemplo siguiente relacionando los números inscritos en los círculos, con los números de los incisos correspondientes de la nomenclatura, se puede observar la aplicación de la misma.

EJEMPLO DESCRIPTIVO DE LA APLICACION DE LA NOMENCLATURA

"ADCB" Add with Carry to B



Descripción: Suma el contenido del Registro B al contenido de la localidad de memoria M y le agrega el contenido del bit C, el resultado es colocado en el Registro B, el contenido de la localidad de memoria M no cambia.



Mnemónico: ADCB

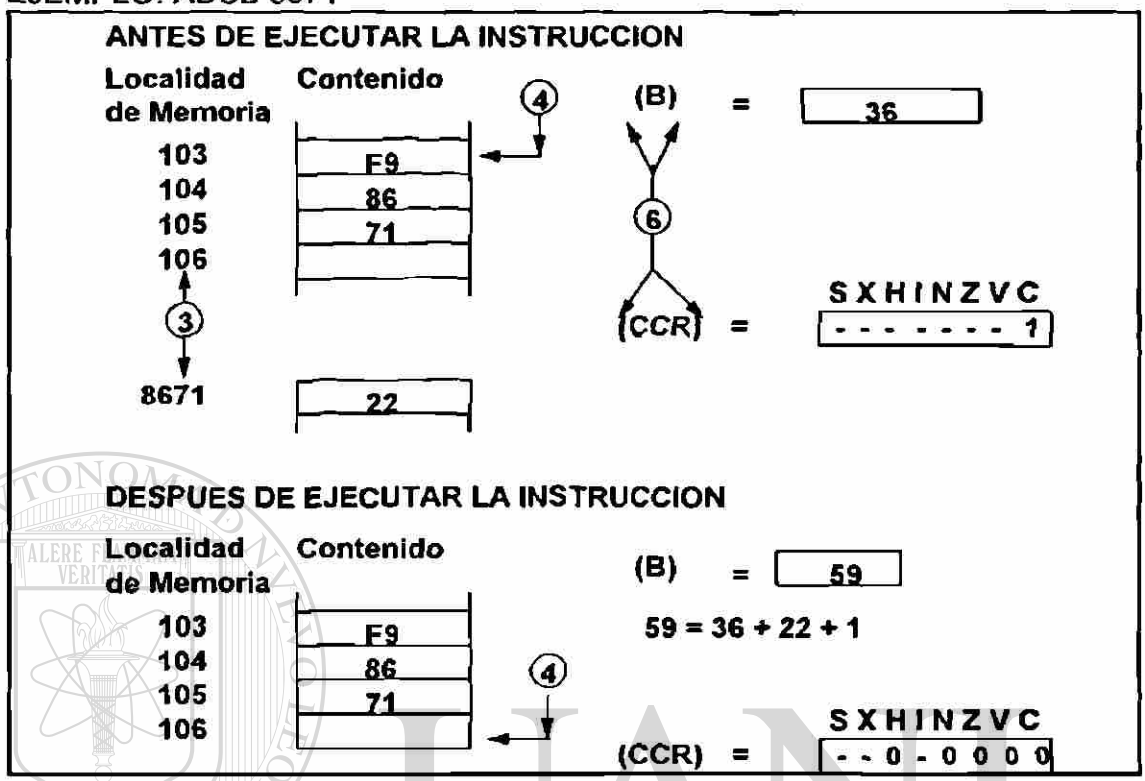
Modo de Direccionamiento	Código de Operación	Listado Fuente	Nota de Explicación
Inmediato	③ ⇩ C9	⑧ ⇩ ADCB #$36 ⇧ ⑨ ↙	1
Directo	D9	ADCB $91	2
Extendido	F9	ADCB $8671	3 Ver ejem.
Indexado X	E9	ADCB $31, X	4
Indexado Y	18E9	ADCB $D2, Y	5

NOTAS DE EXPLICACION:

1. (B) ⇦ (B) + 36 hex + (C)
2. (B) ⇦ (B) + (91 hex) + (C)
3. (B) ⇦ (B) + 8671 hex) + (C)

⑥
↙↘
4. (B) ⇦ (B) + ((X) + 31 hex) + (C)
↑
⑤
5. (B) ⇦ (B) + ((Y) + D2 hex) + (C)

EJEMPLO: ADCB 8671



3.4.2. CLASIFICACION DE LAS INSTRUCCIONES.

Las instrucciones del conjunto pueden clasificarse de diversas formas, en este capítulo se organizan en las siguientes categorías:

1. Instrucciones Aritméticas: sumar, restar, incrementar, decrementar, comparar, complemento a 2, multiplicar, dividir, etc.
2. Instrucciones Lógicas: AND, OR, OR-exclusivo, etc.
3. Instrucciones de Transferencia de datos: cargar, almacenar, transferir, intercambiar, etc.
4. Instrucciones de bifurcación o salto: bifurcaciones condicionales e incondicionales.
5. Instrucciones de llamada y retorno de subrutinas.
6. Instrucciones Misceláneas: manipulación de interrupciones, No operación, Espera, STOP, Prueba, etc.

3.4.2.1. INSTRUCCIONES ARITMETICAS.

En la Tabla 3.4.1. se listan las instrucciones Aritméticas en orden alfabético de acuerdo a los mnemónicos, observe como aparecen realzadas algunas letras de la función en Inglés que corresponden con los mnemónicos.

Tabla 3.4.1. Lista de Instrucciónes Aritméticas

Mnemónico	Función en Inglés	Función en Español
ABA	Add B to A	Sumar B a A
ABX	Add B to X	Sumar B a X
ABY	Add B to Y	Sumar B a Y
ADCA	Add with Carry to A	Sumar con Cargo a A
ADCB	Add with Carry to B	Sumar con Cargo a B
ADDA	Add Memory to A	Sumar Memoria con A
ADDB	Add Memory to B	Sumar Memoria con B
ADDD	Add Memory to D	Sumar Memoria con D
CBA	Compare B to A	Comparar B con A
CMPA	Compare A to Memory	Comparar A con Memoria
CMPB	Compare B to Memory	Comparar B con Memoria
CPD	Compare D to Memory	Comparar D con Memoria
CPX	Compare X to Memory	Comparar X con Memoria
CPY	Compare Y to Memory	Comparar Y con Memoria
DAA	Decimal Adjust A	Ajuste Decimal de A
DEC	Decrement Memory	Decrementar Memoria
DECA	Decrement A	Decrementar A
DECB	Decrement B	Decrementar B
DES	Decrement Stack	Decrementar Stack
DEX	Decrement X	Decrementar X
DEY	Decrement Y	Decrementar Y
FDIV	Fractional Divide	División Fraccional
IDIV	Integer Divide	División Entera
INC	Increment Memory	Incrementar Memoria
INCA	Increment A	Incrementar A
INCB	Increment B	Incrementar B
INS	Increment Stack	Incrementar Stack
INX	Increment X	Incrementar X
INY	Increment Y	Incrementar Y
MUL	Multiply 8 by 8	Multiplicar de 8 por 8
NEG	Negate (2's Complement) memory	Negativo(Complemento a 2)de Memoria
NEGA	Negate (2's Complement) A	Negativo (Complemento a 2) de A
NEGB	Negate (2's Complement) B	Negativo (Complemento a 2) de B
SBA	Subtract B to A	Resta B de A
SBCA	Subtract with Carry from A	Resta con Acarreo de A
SBCB	Subtract with Carry from B	Resta con Acarreo de B
SUBA	Subtract Memory from A	Resta Memoria de A
SUBB	Subtract Memory from B	Resta Memoria de B
SUBD	Subtract Memory from D	Resta Memoria de D

"ABA" Add Accumulator B to Accumulator A:
Operación: (A) ⇦ (A) + (B)
Descripción: Suma el contenido del acumulador B al contenido del acumulador A y el resultado es colocado en el acumulador A. El contenido del acumulador B no cambia.

	S	X	H	I	N	Z	V	C
CCR	-	-	⇕	-	⇕	⇕	⇕	⇕

Mnemónico: ABA
Código de Operación: 1 B

Antes de ejecutar la instrucción:ABA

Localidad de Memoria	Contenido
100	1B
101	

(A)=	A3
(B)=	34
	S X H I N Z V C
(CCR)=	- - - - - - - -

Después de ejecutar la instrucción: ABA

Localidad de Memoria	Contenido
100	1B
101	

(A)=	D7
(B)=	34
	S X H I N Z V C
(CCR)=	- - 0 - 1 0 0 0

Debido a que en la instrucción ABA es posible afectar los 5 bits de status (H, N, Z, V y C) del Registro de Código de Condiciones CCR, se presenta en seguida una serie de ejemplos donde se muestra: la suma en hexadecimal, su equivalente binaria, los bits afectados del CCR y algunas notas aclarativas.

Ejemplo de Suma donde es afectado el bit H del CCR.

SUMA HEXADECIMAL	SUMA BINARIA EQUIVALENTE
	1⇦half carry
18	00011000
+ 18	+ 00011000
30	00110000

CCR después de la operación:

S	X	H	I	N	Z	V	C
NA	NA	1	NA	NC	NC	NC	NC

NA= No Afectado
NC= No Cambia en este ejemplo

Nota: El bit H (half carry) es puesto en 1 porque se produce un acarreo intermedio al realizar la suma de los bit 3 de los sumandos. El bit H se utiliza en conjunto con la instrucción DAA para la corrección de números en código BCD.

Ejemplo de Suma donde son afectados los bits N y V del CCR.

SUMA HEXADECIMAL

```
  78
+ 50
----
  C8
```

SUMA BINARIA EQUIVALENTE

```
  01111000
+ 01010000
----------
  11001000
     ⇧
bit más significativ
        o
```

CCR después de la operación:

S	X	H	I	N	V	C
NA	NA	NC	NA	1	1	NC

NA = No Afectado
NC= No Cambia en este ejemplo

Notas:

1. En los números con signo, cuando el bit más significativo vale 0 (cero) indica número positivo, cuando el bit más significativo vale 1 (uno) indica número negativo. En éste ejemplo el bit más significativo tomó el valor de 1 después de la suma, lo cual ocasionó que el bit N se hiciera 1.

2. El bit V de sobreflujo (overflow) se activa porque al operar con números con signo, el máximo número positivo es 7F hex (127 decimal) y el resultado supera este valor.

Ejemplo de suma donde es afectado el bit C del CCR.

SUMA HEXADECIMAL

```
  78
+ A0
 11
  8
```

SUMA BINARIA EQUIVALENTE

```
Carry ⇨ 111
        01111000
      + 10100000
Carry ⇨ 100011000
```

CCR después de la operación:

S	X	H	I	N	Z	V	C
NA	NA	NC	NA	NC	NC	NC	1

NA = No Afectado
NC= No Cambia en este ejemplo

Nota: El bit C (carry) es puesto en 1 porque se produce un acarreo *(cargo = carry) al realizar la suma de los bits 7 de los sumandos.*

"ABX" Add accumulator **B** to Index Register **X**.
Operación: (X)⇦ (X) + (B)
Descripción: Suma el contenido del acumulador B no-signado, al contenido del Registro Indice X considerando el acarreo (carry), el resultado es colocado *en el Registro Indice X y el acumulador B no cambia.*

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	-	-	-	-

Mnemónico: ABX
Código de Operación: 3A

Antes de ejecutar la instrucción: ABX

Localidad de Memoria	Contenido
101	3A

(X)=	3124
(B)=	51
X+B=	3175

Después de ejecutar la instrucción: ABX

Localidad de Memoria	Contenido
101	3A

(X)=	3175
(B)=	51

"DECA" Decrement Accumulator A
Operación: (A) ⇦ (A) - 1
Descripción: Resta 1 del contenido del acumulador A y el resultado es colocado en el acumulador A. Los bits N, Z y V en el CCR son puestos en 1 ó 0 de acuerdo al resultado de la operación, el bit C no es afectado por la operación. Cuando opera con valores no signados, solo se puede contar con las instrucciones de valores de complemento a 2, además todas las instrucciones de bifurcación son disponibles.

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇕	⇕	⇕	-

Mnemónico: DECA
Código de Operación: 4A

Antes de ejecutar la instrucción: DECA

Localidad de Memoria	Contenido
106	4A
107	

(A)= 38

Después de ejecutar la instrucción: DECA

Localidad de Memoria	Contenido
106	4A
107	

(A)= 37

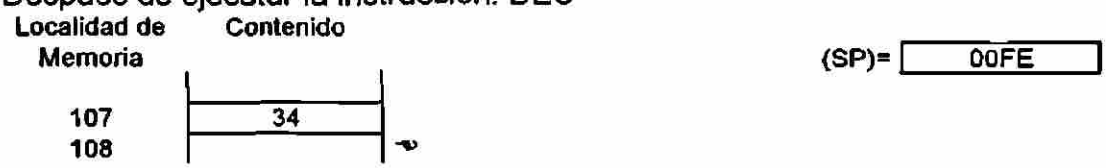
"DES" Decremento Stack Pointer
Operación: (SP) ⇦ (SP) - $0001
Descripción: resta 1 del contenido del Stack Pointer y el resultado es colocado en el Stack Pointer.

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	-	-	-	-

Mnemónico: DES
Código de Operación: 34

Antes de ejecutar la instrucción: DES

Después de ejecutar la instrucción: DES



"CMPA" Compare Accumulator A to Memory

Operación: (A) - (M)

Descripción: *resta al contenido del acumulador* A el *contenido* de la localidad de memoria M con el propósito de poner en 1 o 0 los bits N, Z, V y C del CCR para su uso en instrucciones de bifurcación.

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇕	⇕	⇕	⇕

Mnemónico: CMPA

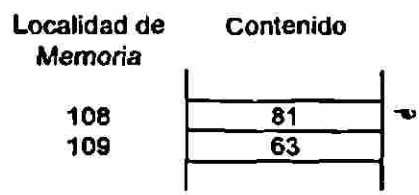
Código de Operación: 34

Modo de Direccionamiento	Código de Operación	Listado Fuente	Notas de Explicación
Inmediato	81	CMPA #$63	1 ver ejem.
Directo	91	CMPA $82	2
Extendido	B1	CMPA $8671	3
Indexado X	A1	CMPA $23, X	4
Indexado Y	18A1	CMPA $23, Y	5

NOTAS:

1.- (A) - 63 hex
2.- (A) - (82 hex)
3.- (A) - (8671 hex)
4.- (A) - ((X) + 23 hex)
5.- (A) - ((Y) + 23 hex)

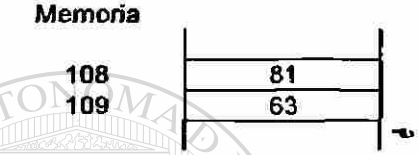
Antes de ejecutar la instrucción: CMPA #$63



(A)= 18 = 00011000
(109)= 63 = 01100011

SXH I NZVC
(CCR) = ————

Después de ejecutar la instrucción: CMPA #$63



(A)= 18 = 00011000
(109)= 63 = 01100011

SXHINZVC
(CCR) = —0-1000

"INC" (Increment Memory Byte)
Operación: (M) ⇦ (M) + $01
Descripción: *suma 1 al contenido de la localidad de memoria M, y el resultado se coloca en la localidad de memoria M. Los bits N, Z y V en el CCR son puestos en 1 ó 0 de acuerdo al resultado de la operación, el bit C, no es afectado por la operación. Cuando opera con valores no signados, solo se puede contar con las instrucciones de bifurcación BEQ y BNE. Cuando opera con valores de complemento a 2, todas las instrucciones de bifurcación son disponibles.*

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇕	⇕	⇕	-

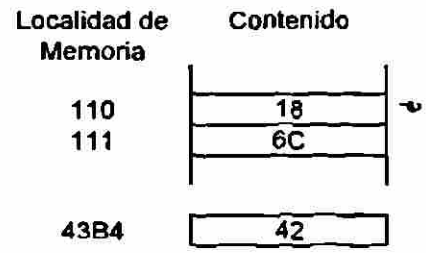
Mnemónico: INC

Modo de Direccionamiento	Código de Operación	Listado Fuente	Notas de Explicación
Extendido	7C	INC $1823	1
Indexado X	6C	INC $14, X	2
Indexado Y	186C	INC $ AC,Y	3

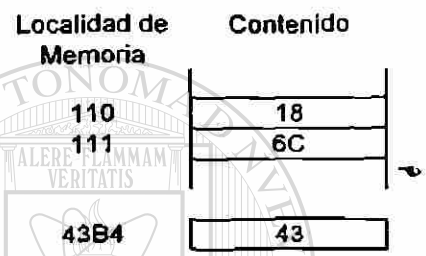
NOTAS:

1.- (1823 hex) ⇦ (1823 hex) + 1
2.- ((X) + 14 hex) ⇦ ((X) + 14 hex) + 1
3.- ((Y) + AC hex) ⇦ ((Y) + AC hex) + 1

Antes de ejecutar la instrucción: INC $AC,Y



Después de ejecutar la instrucción: INC $AC,Y

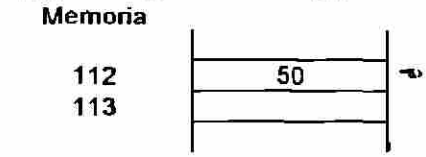


"NEGB" Negate B
Operación: (B) = - (B) = $00 - (B)
Descripción: Remplaza el contenido del acumulador B con su complemento a 2; si el valor es $80 permanece sin cambio.

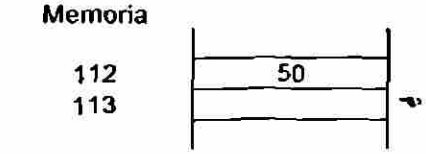
	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇕	⇕	⇕	⇕

Mnemónico: NEGB
Código de Operación: 50

Antes de ejecutar la instrucción: NEGB



Después de ejecutar la instrucción: NEGB



"SBCA" Subtract with Carry from A
Operación: (A) ⇦ (A) - (M) - (C)
Descripción: resta el contenido de A y coloca el resultado en A. Para esta instrucción el bit C del CCR representa un préstamo (borrow).

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇳	⇳	⇳	⇳

Mnemónico: SBCA

Modo de Direccionamiento	Código de Operación	Listado Fuente	Notas de Explicación
Inmediato	82	SBCA #$23	1 ver ejem.
Directo	92	SBCA $23	2
Extendido	B2	SBCA $D323	3
Indexado X	A2	SBCA $16,X	4
Indexado Y	18A2	SBCA $38,Y	5

NOTAS:
1.- (A) ⇦ (A) - 23 hex - (C)
2.- (A) ⇦ (A) - (23 hex) - (C)
3.- (A) ⇦ (A) - (D323 hex) - (C)
4.- (A) ⇦ (A) - ((X) + 16 hex) - (C)
5.- (A) ⇦ (A) - ((Y) + 38 hex) - (C)

Antes de ejecutar la instrucción: SBCA #$23

Localidad de Memoria	Contenido
113	82
114	23

(A) = 18
SXHINZVC
(CCR) = ——1

Después de ejecutar la instrucción: SBCA #$23

Localidad de Memoria	Contenido
112	82
113	23

(A) = -12
SXHINZVC
(CCR) = ——1001

3.4.2.2. INSTRUCCIONES LOGICAS

En la tabla 3.4.2. se listan las instrucciones lógicas en orden alfabético de acuerdo a los mnemónicos, observe como aparecen realzadas algunas letras de la función en inglés que corresponden a los mnemónicos.

Tabla 3.4.2. Instrucciones Lógicas

Mnemónico	Función en Inglés	Función en Español
ASL	Arithmetic Shift Left	Correr Aritmeticamente a la Izq.
ASLA	Arithmetic Shift Left A	Correr A Aritmeticamente a la Izq.
ASLB	Arithmetic Shift Left B	Correr B Aritmeticamente a la Izq
ASLD	Arithmetic Shift Left D	Correr D Aritmeticamente a la Izq.
ASR	Arithmetic Shift Right	Correr Aritmeticamente a la Der.
ASRA	Arithmetic Shift Right A	Correr A Aritmeticamente a la Der.
ASRB	Arithmetic Shift Right B	Correr B Aritmeticamente a la Der.
BITA	Bits Test A with Memory	Prueba los bits de A con Memoria
BITB	Bits Test B with Memory	Prueba los bits de B con Memoria
COM	1's Complement Memory	Complemento a 1 de Memoria
COMA	1's Complement A	Complemento a 1 de A
COMB	1's Complement B	Complemento a 1 de B
EOR	Exclusive OR A with Memory	OR Exclusivo de A con Memoria
EORB	Exclusive OR B with Memory	OR Exclusivo de B con Memoria
LSL	Logical Shift Left	Corrimiento Lógico a la Izquierda
LSLA	Logical Shift Left A	Corrimiento Lógico a la Izq. de A
LSLB	Logical Shift Left B	corrimiento Lógico a la Izq. de B
LSLD	Logical Shift Left D	Corrimiento Lógico a la Izq. de D
LSR	Logical Shift Right	Corrimiento Lógico a la Derecha
LSRA	Logical Shift Right A	Corrimiento Lógico a la Der. de A
LSRB	Logical Shift Right B	Corrimiento Lógico a la Der. de B
LSRD	Logical Shift Right D	Corrimiento Lógico a la Der. de D
ORAA	OR A	OR Inclusive de A con Memoria
ORAB	OR B	OR Inclusive de B con Memoria
ROL	Rotate Left	Rotación a la Izquierda
ROLA	Rotate Left A	Rotar A a la Izquierda
ROLB	Rotate Left B	Rotar B a la Izquierda
ROR	Rotate Right	Rotación a la Derecha
RORA	Rotate Right A	Rotar A a la Derecha
RORB	Rotate Right B	Rotar B a la Derecha

"ANDA" AND A *with Memory*
Operación: (A) ⇦ (A) . (M)
Descripción: Realiza el AND lógico entre el contenido del acumulador A y el contenido de la localidad de memoria M, y coloca el resultado en el acumulador A. Cada bit del acumulador A es "ANDeado" con el correspondiente bit del contenido de la localidad N.

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇳	⇳	0	-

Mnemónico: "ANDA"

Modos de Direccionamiento	Código de Operación	Listado Fuente	Notas de Explicación
Inmediato	84	ANDA #$18	1
Directo	94	ANDA $36	2
Extendido	B4	ANDA $1836	3
Indexado X	A4	ANDA $08,X	4 Ver Ejem.
Indexado Y	18A4	ANDA #E9,Y	5

NOTAS: AND Lógico se representa con " · "
1.- (A) ⇦ (A) ·18 hex
2.- (A) ⇦ (A) · (36 hex)
3.- (A) ⇦ (A) · (1836 hex)
4.- (A) ⇦ (A) · ((X) + 08 hex)
5.- (A) ⇦ (A) · ((Y) + E9 hex)

Antes de ejecutar la instrucción: ANDA $08,X

Localidad de Memoria	Contenido
120	A4
121	08
122	
0338	9D

(X) = 0330
(A)= 5A = 01011010
(0338)= 9D = 10011101

SXHINZVC
(CCR) = ————

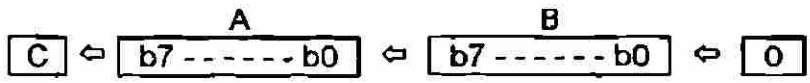
Después de ejecutar la instrucción: ANDA $08,X

Localidad de Memoria	Contenido
120	A4
121	08
122	
0338	9D

(X) = 0330
(A)= 18 = 00011000

SXHINZVC
(CCR) = ——000–

"ASLD" Arithmetic Shift Left D

Operación: 

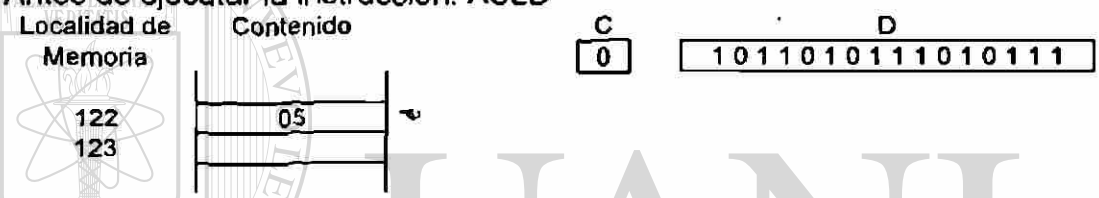
Descripción: Desplaza todos los bits del acumulador D un lugar hacia la izquierda. El bit 0 es cargado con un cero. El bit C en el CCR es cargado con el bit más significativo del acumulador D.

CCR

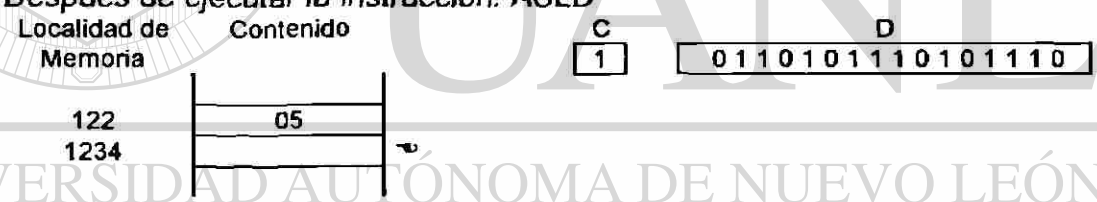
S	X	H	I	N	Z	V	C
-	-	-	-	⇕	⇕	⇕	⇕

Mnemónico: ASLD
Código de Operación: 05

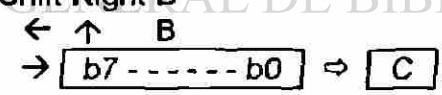
Antes de ejecutar la instrucción: ASLD



Después de ejecutar la instrucción: ASLD



"ASRB" Arithmetic Shift Right B

Operación: 

Descripción: Desplazar todos los bits del acumulador B un lugar hacia la derecha. El bit 7 es mantenido constante. El bit 0 es cargado en el bit C del CCR.

CCR

S	X	H	I	N	Z	V	C
-	-	-	-	⇕	⇕	⇕	⇕

Mnemónico: ASRB
Código de Operación: 57

Antes de ejecutar la instrucción: ASRB

Después de ejecutar la instrucción: ASRB

"EORA" Exclusive OR A with Memory
Operación: (A) ⇦ A ⊕ M
Descripción: Realiza la función lógica OR exclusiva entre el contenido del acumulador A y el contenido de la localidad de memoria M y coloca el resultado en el acumulador A. Con cada bit del acumulador A se realiza la función OR exclusiva con el correspondiente bit de la localidad de memoria M.

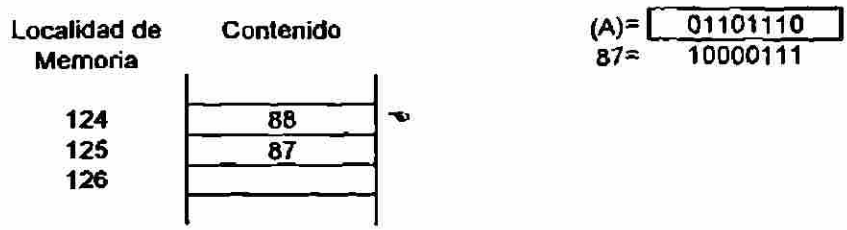
	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇳	⇳	0	⇳

Mnemónico: EORA

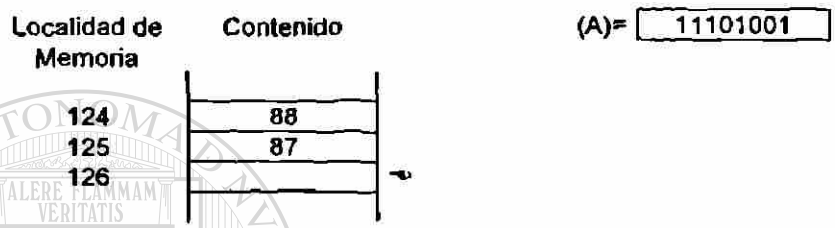
Modo de Direccionamiento	Código de Operación	Listado Fuente	Notas de Explicación
Inmediato	88	EORA #$87	1 ver ejem.
Directo	98	EORA $36	2
Extendido	B8	EORA $D344	3
Indexado X	A8	EORA $31,X	4
Indexado Y	18A8	EORA $71,Y	5

NOTAS: OR exclusive se representa con " ⊕ "
1.- (A) ⇦ (A) ⊕ 87 hex
2.- (A) ⇦ (A) ⊕ (36 hex)
3.- (A) ⇦ (A) ⊕ (D344 hex)
4.- (A) ⇦ (A) ⊕ ((X) + 31 hex)
5.- (A) ⇦ (A) ⊕ ((Y) + 71 hex)

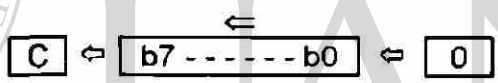
Antes de ejecutar la instrucción: EORA #$87



Después de ejecutar la instrucción: EORA #$87



"LSL" Logical Shift Left = ASL

Operación: 

Descripción: Desplaza todos los bits de la localidad de memoria M un lugar a la izquierda. El bit 0 es cargado con cero. El bit C es cargado con el bit más significativo de la localidad de memoria M.

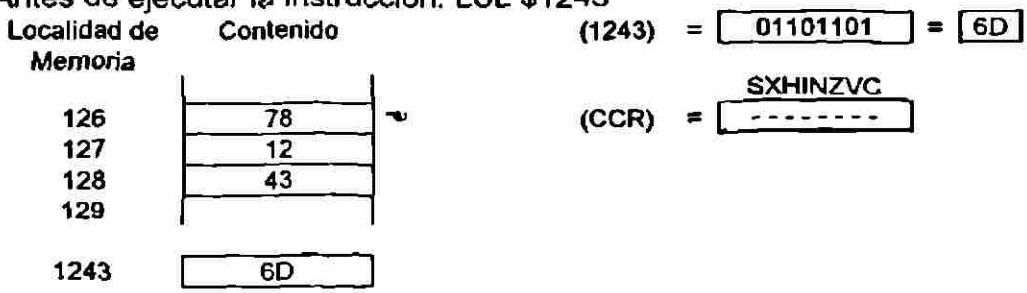
	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇕	⇕	⇕	⇕

Mnemónico: LSL

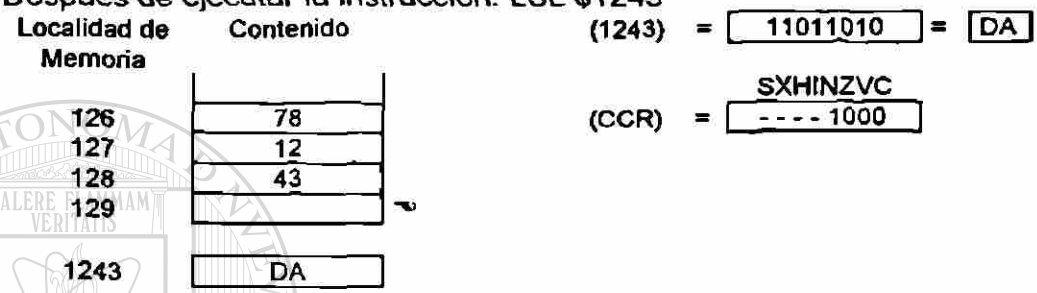
Modo de Direccionamiento	Código de Operación	Listado Fuente	Notas de Explicación
Extendido	78	LSL $1243	1 ver ejem.
Indexado X	68	LSL $18,X	2
Indexado Y	1868	LSL $15,Y	3

NOTAS:
1. Desplaza hacia la Izquierda todos los bits de la localidad 1243 hex.
2. Desplaza hacia la Izquierda todos los bits de la localidad X + 18 hex.
3. Desplaza hacia la Izquierda todos los bits de la localidad Y + 15 hex.

Antes de ejecutar la instrucción: LSL $1243



Después de ejecutar la instrucción: LSL $1243



"ORAB" OR Accumulator B

Operación: (B) ⇦ (B) + (M)

Descripción: Realiza la función lógica OR entre el contenido de B y M y el resultado es colocado en A. Con cada bit del acumulador B se realiza la función OR con los correspondientes bits del contenido de la localidad M.

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇳	⇳	0	-

Mnemónico: ORAB

Modo de Direccionamiento	Código de Operación	Listado Fuente	Notas de Explicación
Inmediato	CA	ORAB #$27	1
Directo	DA	ORAB $48	2 ver ejem.
Extendido	FA	ORAB $E316	3
Indexado X	EA	ORAB $07,X	4
Indexado Y	18EA	ORAB $10,Y	5

NOTAS: OR se representa con " + "

1.- (B) ⇦ (B) + 27 hex

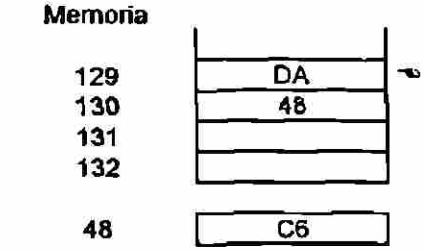
2.- (B) ⇦ (B) + (48 hex)

3.- (B) ⇦ (B) + (E316 hex)

4.- (B) ⇦ (B) + ((X) + 07 hex)

5.- (B) ⇦ (B) + ((Y) + 10 hex)

Antes de ejecutar la instrucción: ORAB $48



(B) = 01101101

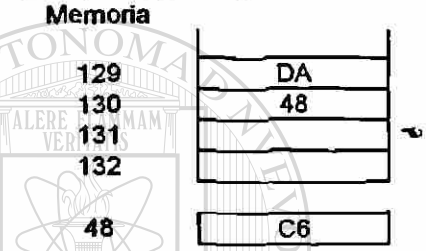
(48) = 11000110

B+48 = 11101111

SXHINZVC

(CCR) = --------

Después de ejecutar la instrucción: ORAB $48



(B) = 11101111

SXHINZVC

(CCR) = ----1000

"ROLA" Rotate Left Accumulator A

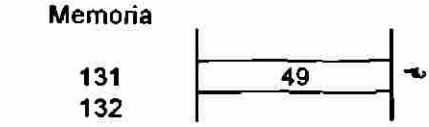
Descripción: Desplaza todos los bits de A un lugar hacia la izquierda. El bit 0 es cargado con el bit C y el bit C es cargado con el bit más significativo del acumulador A.

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇕	⇕	⇕	⇕

Mnemónico: ROLA

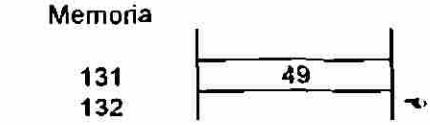
Código de Operación: 49

Antes de ejecutar la instrucción: ROLA



C	A
0	11011101

Después de ejecutar la instrucción: ROLA



C	A
1	10111010

3.4.2.3. INSTRUCCIONES DE TRANSFERENCIA DE DATOS.

En la Tabla 3.4.3. se listan las instrucciones que se consideran forman parte del grupo de Instrucciones de Transferencia de Datos.

Tabla 3.4.3 Instrucciones de Transferencia de Datos

Mnemónico	Función en Inglés	Función en Español
LDAA	Load A	Cargar A
LDAB	Load B	Cargar B
LDD	Load D	Cargar D
LDS	Load Stack Pointer	Cargar Stack
LDX	Load X	Cargar X
LDY	Load Y	Cargar Y
PSHA	Push A onto Stack	"Empujar" A al Stack
PSHB	Push B onto Stack	"Empujar" B al Stack
PSHX	Push X onto Stack	"Empujar" X al Stack
PSHY	Push Y onto Stack	"Empujar" Y al Stack
PULA	Pull A from Stack	"Jala" A desde el Stack
PULB	Pull B from Stack	"Jala" B desde el Stack
PULX	Pull X from Stack	"Jala" X desde el Stack
PULY	Pull Y from Stack	"Jala" Y desde el Stack
STAA	Store A	Almacenar A
STAB	Store B	Almacenar B
STD	Store D	Almacenar D
STS	Store Stack Pointer	Almacenar Stack
STX	Store X	Almacenar X
STY	Store Y	Almacenar Y
TAB	Transfer A to B	Transferir A a B
TAP	Transfer A to CCR	Transferir A a CCR
TBA	Transfer B to A	Transferir B a A
TPA	Transfer CCR to A	Transferir CCR a A
TSX	Transfer Stack to X	Transferir Stack a X
TSY	Transfer Stack to Y	Transferir Stack a Y
TXS	Transfer X to Stack	Transferir X a Stack
TYS	Transfer Y to Stack	Transferir Y a Stack
XGDX	Exchange D with X	Intercambiar D con X
XGDY	Exchange D with Y	Intercambiar D con Y

"LDAB" Load Accumulator B
Operación: (B) ⇦ M
Descripción: Carga el contenido de la localidad de memoria **M** en el acumulador B.

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇳	⇳	0	-

Mnemónico: LDAB.

Modo de Direccionamiento	Código de Operación	Listado Fuente	Notas de Explicación
Inmediato	C6	LDAB #$18	1 ver ejem.
Directo	D6	LDAB $31	2
Extendido	F6	LDAB $1431	3
Indexado X	E6	LDAB $3,X	4
Indexado Y	18E6	LDAB $8,Y	5

NOTAS:
1.- (B) ⇦ 18 hex
2.- (B) ⇦ (31 hex)
3.- (B) ⇦ (1431 hex)
4.- (B) ⇦ ((X) + 3 hex)
5.- (B) ⇦ ((Y) + 8 hex)

Antes de ejecutar la instrucción: LDAB #18

Localidad de Memoria	Contenido
150	C6
151	18
152	

(B) = -36

SXHINZVC
(CCR) = - - - - - - - -

Después de ejecutar la instrucción: LDAB #$18

Localidad de Memoria	Contenido
150	C6
151	18
152	

(B) = 18

SXHINZVC
(CCR) = - - - - 000-

"LDS" Load Stack Pointer
Operación: SPH ⇦ (M) SPL ⇦ (M+1)
Descripción: Carga el byte más significativo del Stack Pointer con el contenido de la localidad M, y carga el byte menos significativo del Stack Pointer con el contenido de la localidad M + 1.

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	⇳	⇳	0	-

Mnemónico: LDS

Modo de Direccionamiento	Código de Operación	*Listado* Fuente	*Notas de* Explicación
Inmediato	8E	LDS #$0116	1
Directo	9E	LDS $26	2 ver ejem.
Extendido	BE	LDS $2122	3
Indexado X	AE	LDS $08,X	4
Indexado Y	18AE	LDS $16,Y	5

1.- (SPH) ⇦ 1 hex (SPL) ⇦ 16 hex
2.- (SPH) ⇦ (26 hex) (SPL) ⇦ (27 hex)
3.- (SPH) ⇦ (2122 hex) (SPL) ⇦ (2123) hex)
4.- (SPH) ⇦ ((X) + 08 hex) (SPL ⇦ ((X) + 09 hex)
5.- (SPH) ⇦ ((Y) + 16 hex) (SPL) ⇦ ((Y) + 17 hex)

Antes de ejecutar la instrucción: LDS $26

Localidad de Memoria	Contenido
152	9E
153	26
154	
26	10
27	21

(SP)= 0000

SXHINZVC
(CCR)= - - - - - - - -

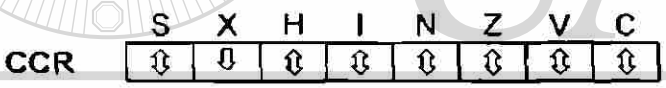
Después de ejecutar la instrucción: LDS $26

Localidad de Memoria	Contenido
152	9E
153	26
154	
26	10
27	21

(SP)= 1021

SXHINZVC
(CCR)= ---- 000-

"TAP" Transfer from Accumulator **A** to Condition Code Register.
Operación: (CCR) ⇦ A
Descripción: Transfiere el contenido del acumulador A desde las posiciones del bit 7 al bit 0 a las correspondientes posiciones del CCR. El contenido del acumulador A permanece sin cambio. El bit X del CCR puede ser puesto en cero como resultado de la instrucción TAP pero no puede ser puesto en uno.



Mnemónico: TAP

3.4.2.4. INSTRUCCIONES DE BIFURCACION O SALTO.

Las instrucciones de bifurcación o salto incondicional (Jump) permiten que el control sea transferido a cualquier dirección en el mapa de memoria de 64 kbytes, en cambio las instrucciones de bifurcación o salto condicional (branch) permiten que la CPU tome decisiones basadas en el contenido de uno o varios de los bits del CCR para realizar la bifurcación. Todos los bloques de decisión en una diagrama de flujo deben corresponder a alguna instrucción de salto o bifurcación condicional. En la tabla 3.4.4. aparecen las instrucciones de Bifurcación o Salto.

Las instrucciones de salto o bifurcación condicionales tienen un rango limitado de -128 a +127 localidades. Mediante el uso de etiquetas en la elaboración del programa en lenguaje ensamblador, el cálculo de cuantas localidades hay que saltar es transparente al usuario.

Tabla 3.4.4. Instrucciones de Bifurcación o Salto

Mnemónico	Función en Inglés	Función en Español	Condición de Salto
BCC	Branch if carry clear	Saltar si carry = 0	C = 0
BCS	Branch if carry set	Saltar si carry = 1	C = 1
BEQ	Branch if equal zero	Saltar si es igual a cero	Z = 1
BGE	Branch if greater than or equal to zero	Saltar si es mayor que o igual a cero	N ⊕ V = 0
BGT	Branch if greater than zero	Saltar si es mayor que cero	Z . (N ⊕ V) = 0
BHI	Branch if higher	Salta si es más grande	C + Z = 0
BHS	Branch if higher or same	Salta si es más grande o lo mismo	C = 0
BLE	Branch if less than or equal to zero	Salta si es menor que o igual a cero	Z +(N ⊕ V) = 1
BLO	Branch if lower	Salta si es más pequeño	C = 1
BLS	Branch if lower or same	Salta si es menor o lo mismo	C + Z = 1
BIT	Branch if less than zero	Salta si es menor que cero	N ⊕ V = 1
BMI	Branch if minus	Salta si es negativo	N = 1
BNE	Branch if not equal to zero	Salta si no es igual a cero	Z = 0
BPL	Branch if plus	Salta si es positivo	N = 0
BRA	Branch Always	Salta siempre	
BRCLR	Branch if bits clear	Salta si bits = 0	
BRN	Branch never	No salta nunca	Ninguna
BRSET	Branch if bits set	Salta si bits = 1	
BVC	Branch if overflow clear	Salta si sobre flujo = 0	V = 0
BVS	Branch if overflow set	Salta si sobre flujo = 1	V = 1
JMP	Jump	Salto incondicional	Ninguna

Cuando se avanza a través de la memoria de programa ejecutando una instrucción cada vez, frecuentemente se encuentran instrucciones de salto condicional (BRANCH). El estado de ciertos bits en el Registro de Código de Condición CCR indican al MCU si el salto condicional toma lugar o no. Estos bits mediante una instrucción previa a la instrucción BRANCH deben ser puestos en "1" o "0" . Para que un salto condicional se realice, el estado de los bits debe ser favorable, si no es así, se ejecuta entonces la siguiente instrucción. En seguida se muestran algunos ejemplos donde aparecen las instrucciones de salto condicional.

Instrucciones	Estado de los bits del CCR después de ejecutar la instrucción								¿Ocurre la condición de salto?
	S	X	H	I	N	Z	V	C	
1.-LDAB $FF	NA	NA	NA	NA	1	0	0	NA	
COMB	NA	NA	NA	NA	0	1	0	1	
BCC	NA	NA	NA	NA	0	1	0	1	No
2.- LDAA #$0	NA	NA	NA	NA	0	1	0	NA	
ADDA #$80	NA	NA	0	NA	1	0	0	0	
BHI	NA	NA	0	NA	1	0	0	0	Si
3.- LDAA #$01	NA	NA	NA	NA	0	0	0	NA	
ASRA	NA	NA	NA	NA	0	1	1	1	
BLE	NA	NA	NA	NA	0	1	1	1	Si
4.- LDAA #$85	NA	NA	NA	NA	1	0	0	NA	
LDAB #$05	NA	NA	NA	NA	0	0	0	NA	
SBA	NA	NA	NA	NA	1	0	0	0	
BMI	NA	NA	NA	NA	1	0	0	0	SI
5.- LDX #$01	NA	NA	NA	NA	0	0	0	NA	
DEX	NA	NA	NA	NA	0	1	0	NA	
BEQ	NA	NA	NA	NA	0	1	0	NA	SI
6.-LDAA #$7F	NA	NA	NA	NA	0	0	0	NA	
RORA	NA	NA	NA	NA	0	0	1	1	
BVS	NA	NA	NA	NA	0	0	1	1	SI
7.-LDAA #$7F	NA	NA	NA	NA	0	0	0	NA	
ADDA #$01	NA	NA	NA	NA	1	0	1	0	
COMA	NA	NA	NA	NA	0	0	0	1	
BGE	NA	NA	NA	NA	0	0	0	1	SI
8.-LDAA #$7F	NA	NA	NA	NA	0	0	0	NA	
LDAB #$7E	NA	NA	NA	NA	0	0	0	NA	
SBA	NA	NA	NA	NA	0	0	0	0	
LSRA	NA	NA	NA	NA	0	1	1	1	
BCS	NA	NA	NA	NA	0	1	1	1	SI
9.- LDAB #$FF	NA	NA	NA	NA	1	0	0	NA	
TSTB	NA	NA	NA	NA	1	0	0	0	
BCC	NA	NA	NA	NA	1	0	0	0	SI

3.4.5. INSTRUCCIONES DE LLAMADA Y RETORNO DE SUBRUTINA.

Conceptos Generales:

Las *Subrutinas* son un bloque de instrucciones que son utilizadas repetidamente, y permiten de una manera sencilla, dividir el trabajo de un programa en secciones. El MCU automatiza el proceso de recordar la dirección en el Programa Principal, donde la ejecución debe reasumirse después de terminar la subrutina. En la fig. 3.4.5.1. se ilustra el uso repetido de una subrutina.

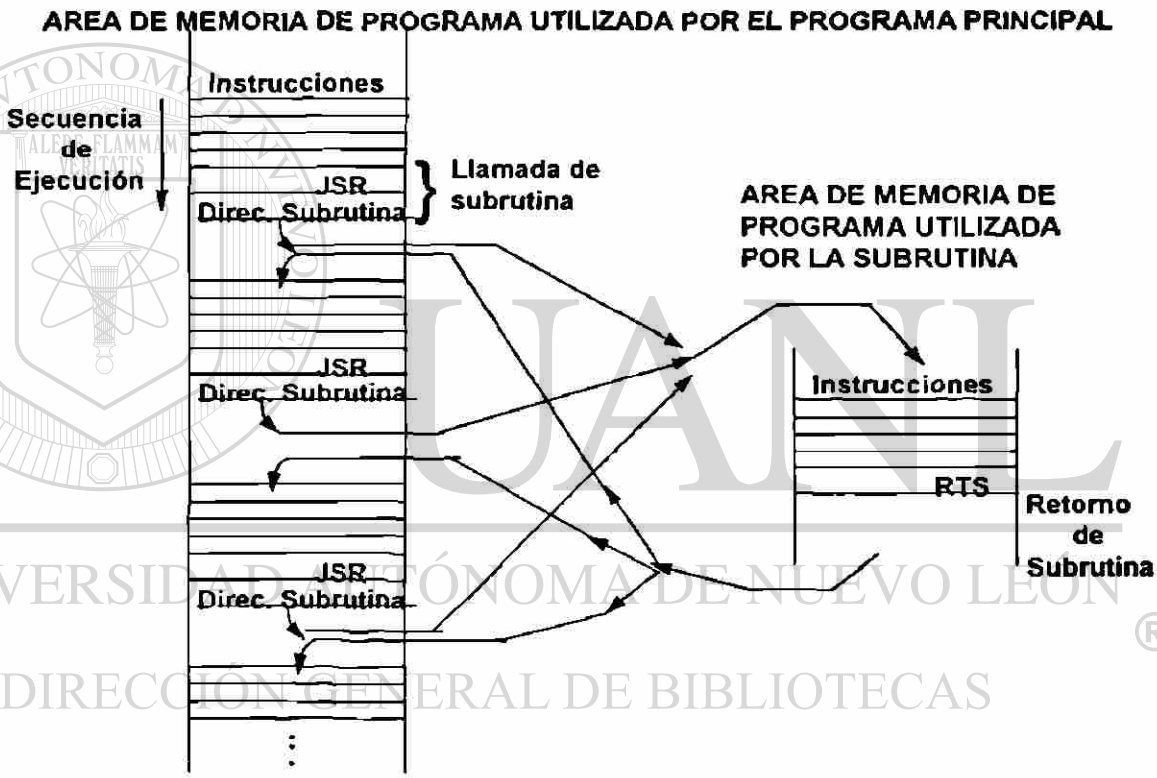


Fig. 3.4.5.1. Uso repetido de una subrutina

El Stack o Pila es un área seleccionada de memoria RAM, utilizado para almacenar en forma temporal datos o direcciones. El Stack o Pila recibe su nombre debido al hecho de que puede visualizarse como una pila (Stack) de datos, en donde los últimos datos en entrar son los primeros en salir.

El Stack Pointer (SP) o Apuntador de Pila es el registro utilizado para direccionar el Stack. El Stack Pointer o SP almacena (apunta hacia) la dirección de *la última localidad vacía utilizable para almacenar datos*. Es

sumamente importante que el Programador en las primeras instrucciones del programa, apunte el Stack Pointer a un área utilizable de la RAM con suficiente espacio, para evitar que se traslape con otros usos de la RAM que pueda dar lugar a fallas en el programa.

Para los MCU's de la familia M68HC11 conviene imaginar que las localidades o direcciones del Stack se numeran en forma semejante a como se numeran los renglones de una hoja de papel. Pero existe una diferencia importante, en el Stack la escritura de datos se empieza en las localidades con números mayores (correspondientes a los renglones inferiores) y se continua hacia las localidades con números menores (que corresponde a los renglones superiores), la lectura o extracción de datos se realiza en forma semejante tanto en el Stack como en una hoja, pero cada vez que se extrae un dato del Stack, se considera que tal localidad queda vacía, esto es ilustrado en la fig. 3.4.5.2.

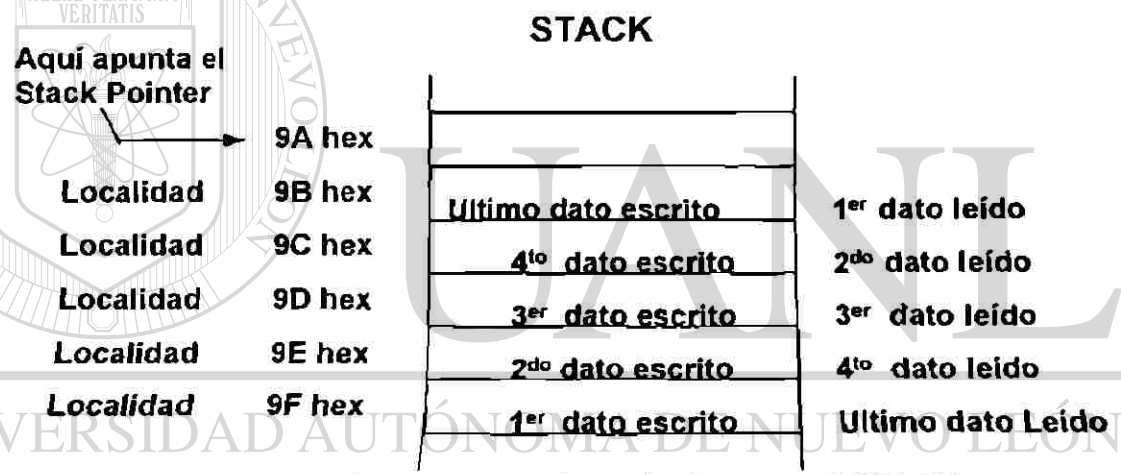


Fig. 3.4.5.2. Lectura y escritura de datos en el STACK

Solo dos operaciones son permitidas en el Stack: 1).- Escritura en la parte de arriba del Stack, a lo que normalmente se le conoce como **push.** 2).- Leer en la parte de arriba del Stack, a lo que normalmente se le conoce como **pop o pull.**

Una operación push o de escritura en el stack causa que un dato de algún registro del CPU sea escrito en la localidad de memoria a la que apunta el Stack Pointer (SP). El Stack Pointer (SP) es entonces automáticamente decrementado apuntando hacia la siguiente localidad libre en la parte de arriba del Stack.

En una operación de lectura (pop o pull) el Stack Pointer o SP es decrementado para que apunte a la dirección del último dato escrito en la parte

de arriba de la pila (stack) y entonces se realiza la operación de lectura y el dato es retornado a algún registro del CPU, después de que se realiza la lectura se considera que la localidad queda vacía.

"JSR" Jump to Subroutine
Operación: paso a paso.

1a.- (PC) ⇦ (PC) + $0003 Para direccionamiento extendido.
1b.- (PC) ⇦ (PC) + $0002 Para direccionamiento Directo o Indexado X.
2.- ((SP)) ⇦ PCL
3.- (SP) ⇦ (SP) - $0001
4.- ((SP)) ⇦ PCH
5.- (SP) ⇦ (SP) - $0001
6.- (PC) ⇦ Dirección Efectiva.

Descripción: paso a paso.

1. El Contador de Programa PC es incrementado en 3 ó 2 dependiendo del modo de direccionamiento. Esta será la dirección de retorno de la subrutina.

2. El byte menos significativo del Contador de Programa PCL o sea el byte menos significativo de la dirección de retorno de la subrutina es escrito en el Stack en la dirección a la que apunta el Stack Pointer SP.

3. El Stack Pointer es decrementado en uno.

4. El byte más significativo del Contador de Programa PCH o sea el byte más significativo de la dirección de retorno de la subrutina es escrito en el stack, en la localidad a donde apunta el más reciente valor del Stack Pointer SP resultante del paso 3.

5. El Stack Pointer SP es decrementado en uno.

6. La Dirección Efectiva es obtenida de acuerdo a las reglas de *direccionamiento* (Extendido, Directo o Indexado), y es almacenado en el Contador de Programa PC. Un salto ocurre a la dirección efectiva a donde ahora apunta el Contador de Programa y ahí se inicia la ejecución de la subrutina.

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	-	-	-	-

Mnemónico: JSR

Modo de Direccionamiento	Código de *Operación*	Listado *Fuente*	Notas de Explicación
Directo	9D	JSR $45	1
Extendido	BD	JSR $B700	2 ver ejem.
Indexado X	AD	JSR $23,X	3
Indexado Y	18AD	JSR $54,Y	4

Antes de ejecutar la instrucción: JSR $B700

PROGRAMA PRINCIPAL

Localidad de Memoria	Contenido
0160	BD
0161	B7
0162	00

(SP)= 01FF

(PC)= 0160

SUBRUTINA

Localidad de Memoria	Contenido
B700	instrucción #1
B701	instrucción #2
	•
	•
	•

STACK

localidad	contenido	
01FD		
01FE		
01FF		⇦aquí apunta el SP

Después de ejecutar la instrucción: JSR $B700

PROGRAMA PRINCIPAL

Localidad de Memoria	Contenido
0160	BD
0161	B7
0162	00

(SP)= 01FD

(PC)= B700

SUBRUTINA

Localidad de Memoria	Contenido
B700	instrucción #1
B701	instrucción #2
	•
	•
	•

STACK

localidad	*contenido*	
01FD		⇦aquí apunta el SP
01FE	01	
01FF	63	

"RTS" Return to Subrutine
Operación: paso a paso.

1.- (SP) ⇦ (SP) + $0001
2.- (PC) ⇦ PCH = ((SP))
3.- (SP) ⇦ (SP) + $0001
4.- (PC) ⇦ PCL = ((SP))

Descripción: paso a paso.

1. El Stack Pointer SP es incrementado en uno.

2. El contenido de la localidad de memoria hacia donde apunta el nuevo Stack Pointer es el byte más significativo del Contador Programa PCH y corresponde al byte más significativo de la dirección de retorno de la subrutina. Este contenido es leído (pulled) y almacenado como parte del Contador de Programa PC.

3. El Stack Pointer SP es otra vez incrementado en 1.

4. El contenido de la localidad de memoria a donde ahora apunta el Stack Pointer SP es el byte menos significativo del Contador de Programa PCL y corresponde al byte menos significativo de la dirección de retorno. Este contenido es leído (pulled) y almacenado en el Contador de Programa PC que ahora ya tiene su valor completo que corresponde a la dirección de retorno. El programa principal continua entonces su ejecución.

Nota: Una subrutina siempre debe terminar con una instrucción RTS.

	S	X	H	I	N	Z	V	C
CCR	-	-	-	-	-	-	-	-

Mnemónico: RTS.
Código de Operación: 39

Antes de ejecutar la instrucción: RTS

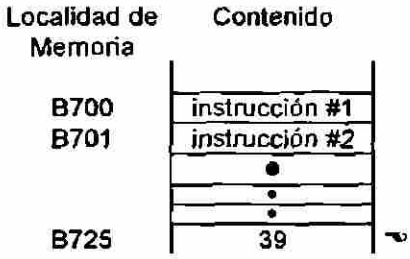
PROGRAMA PRINCIPAL

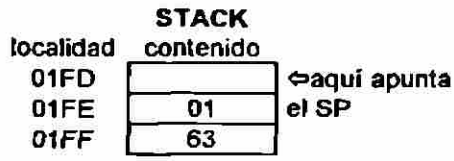
Localidad de Memoria	Contenido
0160	BD
0161	B7
0162	00

(SP)= 01FD

(PC)= B725

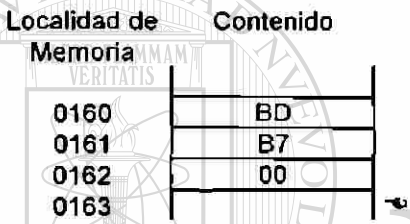
SUBRUTINA

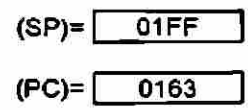




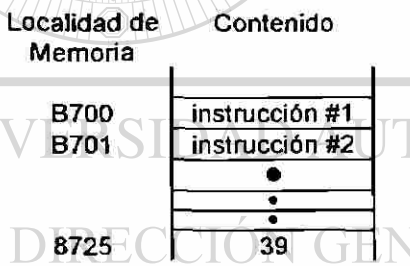
Después de ejecutar la instrucción: JSR $B700

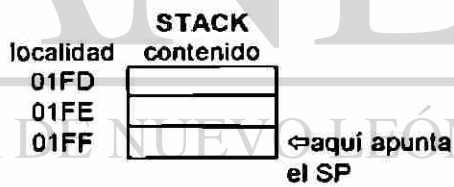
PROGRAMA PRINCIPAL





SUBRUTINA





3.4.2.6. INSTRUCCIONES MISCELANEAS.

Aquí se incluyen algunas instrucciones diversas que no corresponden a las clasificadas anteriormente y que permiten: la manipulación de bits, la manipulación de interrupciones, generar leves retardos, etc. Estas instrucciones son listadas en la Tabla 3.4.6.1.

Tabla 3.4.6.1 Instrucciones Misceláneas

Mnemónico	Función en Inglés	Función en Español
BCLR	Clear Bits in Memory	Ceros en Memoria
BSET	Set Bits in Memory	Unos en Memoria
CLC	Clear Carry bit	Cero en bit de Cargo
CLI	Clear Interrup Mask bit	Cero en bit de Interrupción
CLV	Clear Overflow bit	Cero en bit de Sobreflujo
SEC	Set Carry bit	Uno en bit de Cargo
SEI	Set Interrupt Mask bit	Uno en bit de Interrupción
NOP	No Operation	No Operación
RTI	Return from Interrupt	Retorno de Interrupción
S	Stop	Alto
SWI	Software Interrupt	Interrupción por Software
TEST	Test	Prueba
WAI	Wait for Interrupt	Espera de Interrupción

CAPITULO 4

SISTEMA DE DESARROLLO DE SOFTWARE

4.1. INTRODUCCION.

El objetivo de este capítulo es describir algunos conceptos básicos implicados en el desarrollo de software (programas). Además, aprender el manejo de algunas herramientas que nos faciliten el transferir y depurar un programa para hacer que lo ejecute apropiadamente un MCU.

La **Programación** es la preparación de la lista de instrucciones que una computadora, microprocesador o microcontrolador debe ejecutar. La programación es un trabajo de diseño que requiere un alto grado de creatividad. Cuando un programa funciona inapropiadamente generalmente se debe a que está mal elaborado y pocas veces se debe a fallas del equipo.

El **Software** o programa está constituido por la lista de instrucciones y se almacena temporal o permanentemente en la memoria de programa. Software es un término generalmente utilizado para denominar a todos los programas. **Firmware** es el nombre dado al software que es almacenado permanentemente en la memoria ROM.

El **Sistema de Desarrollo de Software** es el conjunto de técnicas y herramientas que el programador utiliza en el proceso de desarrollo de software y consta de:

- Técnicas de Programación.
- Lenguajes de Programación.
- Editor de Texto.
- Programa Ensamblador
- Programa Monitor.

En la figura 4.1 se muestra la secuencia en el Sistema de Desarrollo de Software. El procedimiento puede seguirse en la figura por medio de los números dentro de los círculos y los pasos que a continuación se detallan.

Paso 1.- Aplicando técnicas de programación se elabora el diagrama de flujo estableciendo las actividades que se desea realice el programa.

Paso 2.- Utilizando una PC que tenga instalado un Editor de Texto se teclea el listado de instrucciones para obtener el Programa Fuente (Programa.ASM) y se guarda en un disco para usarlo después.

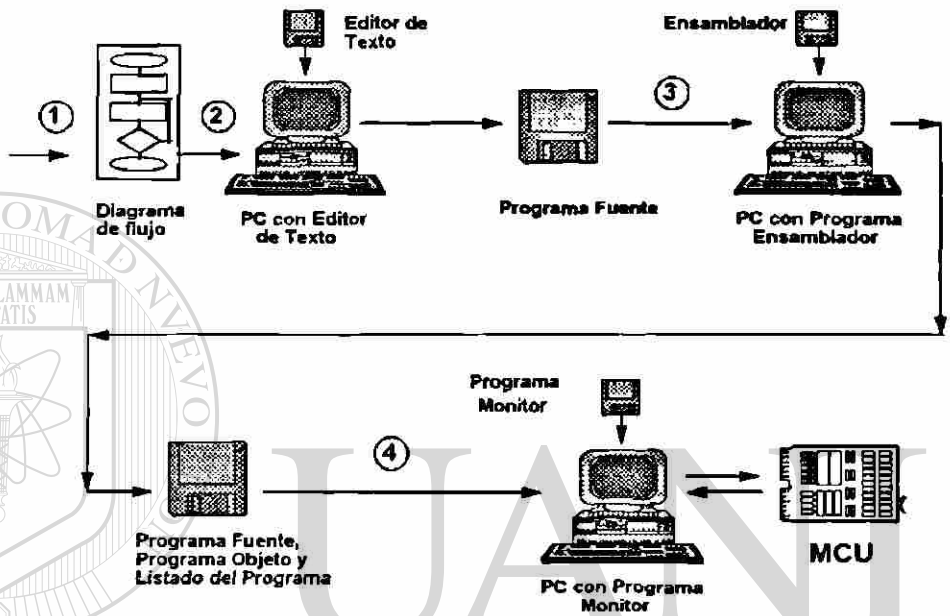


Figura 4.1. Secuencia en el Desarrollo de Software.

Paso 3.- Utilizando una PC que tenga instalado el Programa Ensamblador, se transforma o ensambla el Programa Fuente para obtener: El Programa Objeto (Programa.S19) y el Listado del Programa (Programa.LST)

Paso 4.- El Programa Monitor instalado en la PC permite depurar y corregir el Software desarrollado hasta obtener el comportamiento adecuado. El Programa Monitor realiza las siguientes funciones:

- Carga el Programa Objeto en el MCU
- Corre el Programa Objeto
- Lee y modifica memorias y registros internos del MCU

4.2. TECNICAS DE PROGRAMACION.

La primer pregunta que hay que responder antes de hacer cualquier esfuerzo de programación es: ¿Qué se quiere que haga el programa? La falta de responder esta pregunta con propiedad y eficiencia es con frecuencia uno de los obstáculos más graves y equivale a iniciar un viaje sin conocer el destino. ¡Este es un error muy común que se comete en el desarrollo de software¡ Después de decidir lo que hará el programa, se puede continuar con el proceso de desarrollo del software.

La **Programación Estructurada** es la técnica de programación más utilizada, mediante un diagrama muy parecido a la estructura jerárquica de una corporación, se establece en forma descendente como se desea que el programa realice su trabajo. El organizar y codificar programas en forma estructurada permite reducir su complejidad, mejora la claridad y facilita su depuración y modificación.

El Diagrama de Flujo es una representación gráfica que ilustra los pasos lógicos, cálculos y decisiones que en secuencia deben ser realizados para cumplir con una tarea específica. Antes de empezar a escribir un programa, es práctica común generar un Diagrama de Flujo de las actividades que se desea que realice el programa. Después de generar un diagrama de flujo, la escritura del programa es relativamente simple. En la figura 4.2. aparecen los símbolos comunes de un diagrama de flujo.

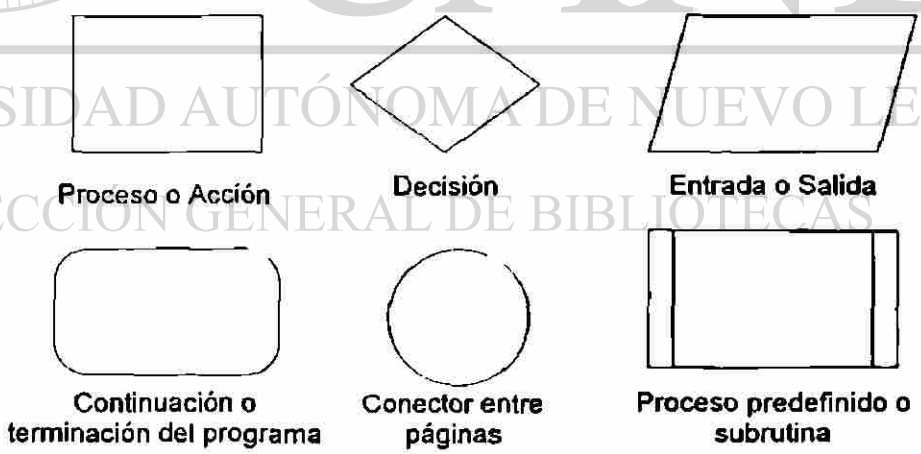


Fig. 4.2 . Símbolos comunes de un diagrama de flujo

4.3 LENGUAJES DE PROGRAMACION

En la actualidad existen diferentes tipos de lenguajes que son utilizados por el programador para realizar software, pero al aumentar el nivel de complejidad del lenguaje utilizado, los detalles de la estructura interna del microprocesador o microcontrolador se escapan a la vista del usuario. Los Lenguajes de Programación pueden ser clasificados como:

- Lenguaje de Máquina (de bajo nivel)
- Lenguaje Ensamblador (de nivel medio)
- Lenguaje de Alto Nivel

El Lenguaje de Máquina es un lenguaje binario pues está constituido exclusivamente por unos y ceros, y es el único lenguaje que reconocen los MCU's.

Un Programa en Lenguaje de Máquina contiene los códigos binarios (normalmente expresados en notación hexadecimal) que representan las instrucciones y datos de un MCU. **Programa Objeto** es el nombre dado a un programa ya expresado en lenguaje de máquina.

El **Lenguaje Ensamblador** es una colección de símbolos mnemónicos que representan: operaciones (mnemónicos de instrucciones del MCU y directivas para el ensamblador), etiquetas, operadores y símbolos especiales.

Un **Programa Fuente o Programa en Lenguaje Ensamblador** es el texto donde los mnemónicos (ayudas de memorización) del fabricante son utilizados para representar las instrucciones del MCU. Programa Fuente es otro nombre dado a un Programa en Lenguaje Ensamblador y no es ejecutable por el MCU.

Los **Lenguajes de Alto Nivel** utilizan proposiciones en inglés que representan desde unas cuantas, hasta muchas instrucciones equivalentes de lenguaje ensamblador o lenguaje de máquina. Algunos lenguajes de alto nivel son: BASIC, FORTRAN, Pascal, C, COBOL, etc. El nombre de Programa Fuente también se aplica al programa escrito mediante las instrucciones de los lenguajes de alto nivel.

4.4. EDITOR DE TEXTO.

Un **Editor de Texto** es un programa de computadora que es utilizado para escribir el Programa Fuente. Un Programa en Lenguaje Ensamblador es el texto de una Programa Fuente. Para que el Programa Fuente sea útil y legible

al MCU, debe ser transformado en Programa Objeto mediante un Programa Ensamblador.

El formato o sintaxis utilizado al escribir el programa fuente mediante el Editor de Texto, debe corresponder al utilizado por el Programa Ensamblador.

El Editor de Texto es un programa residente en computadora que permite usarla como una máquina de escribir eléctrica. En lugar de un Editor de Texto puede ser utilizado un Procesador de Palabras que produzca la salida de texto ASCII, también conocida como salida no-documentada. El Modo no-documentado produce un archivo sin los caracteres de control no imprimibles que son usados en documentos formateados, los cuales pueden producir errores al ensamblar el programa.

Debido a las diferencias de cada Editor de Textos, el usuario deberá elegir el de su preferencia para aprender a operarlo, pues aquí no será explicado ninguno de ellos por no ser parte del curso.

4.5. PROGRAMA ENSAMBLADOR.

El Ensamblador es un programa que procesa el Programa Fuente (escrito en lenguaje ensamblador) y lo traduce en Programa Objeto (escrito en lenguaje de máquina) el cual es ejecutable por el MCU, además produce un listado del Programa.

Motorola proporciona como herramienta un Ensamblador de 8 bit de dos pasos llamado AS11.EXE, el cual puede ser utilizado en computadoras IBM-PC o compatibles y produce el Programa Objeto para los MCU's de la familia M68HC11, existen otros ensambladores diferentes pero aquí solo veremos el AS11.EXE.

4.5.1. FORMATO O SINTAXIS DE LAS LINEAS FUENTE DEL PROGRAMA.

El Programa Fuente consiste de una secuencia de líneas o estatutos. En cada línea solamente es permitida una instrucción o directiva de ensamblado. Cada línea de ensamblado termina al oprimir la tecla ENTER. El formato o sintaxis de las líneas puede incluir algunos o todos los cuatro campos siguientes:

Etiqueta Operación Operando(s) Comentarios

4.5.1.1. El Campo de la Etiqueta. El primer campo de una línea o estatuto es el Campo de la Etiqueta y tiene las siguientes características:

1. Si no se requiere ninguna etiqueta en el estatuto, entonces el primer carácter debe ser un espacio en blanco, lo cuál indica que el campo de la etiqueta esta vacío.

2. Si el primer carácter en el campo de la etiqueta es un asterisco (*), entonces el resto del estatuto es un comentario. Los comentarios son ignorados por el Ensamblador y son impresos en el listado del programa solamente para información.

3. Los símbolos o caracteres utilizados para escribir etiquetas son:
   - Las letras mayúsculas y minúsculas del alfabeto, las cuales son consideradas distintas.
   - Los dígitos del 0 al 9.
   - Los caracteres especiales: Punto (.), signo de dólar ($) y subrayado (_).

4. Las etiquetas consisten de 1 a 16 caracteres y el primero debe ser alfabético, punto (.) o subrayado (_).

5. Las etiquetas no deben ser repetidas en otras líneas o un error será indicado.

6. Opcionalmente la etiqueta puede terminar con dos puntos (:), pero estos no forman parte de la etiqueta y solo sirven para separarla del resto de la línea. Los siguientes fragmentos de programa son equivalentes:

```
AQUI: DECA
      BNE AQUI

AQUI DECA
     BNE AQUI
```


4.5.1.2. El Campo de Operación se encuentra enseguida del Campo de la Etiqueta y tiene las siguientes características:

1. Debe ser precedido al menos por un espacio en blanco.

2. El Campo de Operación debe contener el mnemónico de una instrucción del *MCU* o alguna *Directiva del Ensamblador (las directivas son explicadas más adelante).*

3. Las letras mayúsculas son convertidas a letras minúsculas antes de ser checadas como operaciones válidas, por lo tanto "NOP", "NoP" y "nop" son reconocidos como el mismo mnemónico.

4.5.1.3. El Campo del Operando es utilizado para establecer el operando de una instrucción o para especificar el modo de direccionamiento de una instrucción. El Campo del Operando tiene las siguientes características:

1. Debe estar enseguida del Campo de Operación y debe ser precedido al menos por un espacio en blanco.

2. El Campo del Operando puede contener: una etiqueta, una expresión, o una combinación de etiquetas y expresiones separadas por comas (,).

3. El formato de los operandos utilizados por los MCU's de la familia M68HC11 es mostrado en la Tabla 4.5.1.3.

Tabla 4.5.1.3. Formato de los Operandos

Formato del Operando	Modo de Direccionamiento
No operando	Inherente
expresión	Directo, Extendido, Relativo
# expresión,	Inmediato
expresión, X	Indexado X
expresión, Y	Indexado Y
expresión 1 expresión 2	Bits en 1's o O's
expresión 1 expresión 2 expresión 3	Prueba de bits o Salto Condicional

Notas: expresión1 expresa direccionamiento directo o indexado.
expresión2 establece el byte de máscara.
expresión3 genera una dirección relativa.

4.5.1.3.1. Expresiones. Una expresión es una combinación de: operadores (algebraicos o lógicos), etiquetas y constantes. Las expresiones son evaluadas de izquierda a derecha y no esta previsto el uso de paréntesis. La aritmética es realizada con signo en complemento a 2 y con precisión entera de 16 bits.

Los operadores algebraicos y lógicos utilizados en las expresiones, así como la función que representan, son mostrados enseguida:

+	suma
-	resta
x	multiplicación
/	división
%	residuo después de la división
&	AND
I	OR
^	OR EXCLUSIVE

Las **etiquetas** son asociadas con un valor entero de 16 bit, el cual es usado en lugar de la etiqueta durante la evaluación de la expresión.

Las **constantes** representan cantidades que no varían en valor durante le ejecución del programa. Las constantes pueden ser representadas en alguno de los siguientes formatos: decimal, hexadecimal, binario, octal o ASCII.

Los prefijos utilizados antes de cada número para indicar su formato son:

$	Hexadecimal
%	Binario
@	Octal
'	ASCII

El Ensamblador interpreta las constantes sin prefijo como decimales. Además, convierte todas las constantes a código binario y las muestra en el listado en forma hexadecimal. Las constantes debe caer en el rango decimal de 0 a 65535 inclusive o su equivalente en cualquier otro sistema numérico.

Ejemplos de constantes decimales:

válidos	inválidos	razón de invalidez
12	123456	fuera de rango
12345	12.3	Carácter inválido

Ejemplos de constantes hexadecimales

válidos	inválidos	razón de invalidez
$12	ABCD	falta prefijo $
$ABCD	$G2A	carácter inválido
$001F	$2F018	fuera de rango

Ejemplos de constantes binarias

válidos	inválidos	razón de invalidez
%00101	1011	falta prefijo %
%1	%210101	dígito inválido

Ejemplos de constantes octales

válidos	inválidos	razón de invalidez
@17634	@2317234	fuera de rango
@377	@23914	dígito inválido

Ejemplos de constantes ASCII

valido	inválido	razón de invalidez
'*	'VALID	demasiado largo

4.5.1.4. El Campo de Comentarios es el último campo de un estatuto, este campo es opcional, y es impreso solamente en el listado del programa para propósitos de información. El Campo de Comentarios debe ser separado por al menos un espacio en blanco del campo que le precede, además puede contener cualquier carácter ASCII imprimible.

4.5.2. DIRECTIVAS DEL ENSAMBLADOR.

Las Directivas controlan el funcionamiento del Ensamblador y no son transferidas al Programa Objeto. La notación utilizada para describir las Directivas es la siguiente:

XYZ Las letras mayúsculas realzadas se usan para indicar el mnemónico de las directivas.

( ) Los paréntesis denotan un elemento opcional.

< > Los paréntesis angulares se usan para nombrar elementos y son escritos con letras minúsculas. Todos los elementos fuera de estos paréntesis '< >' deben ser especificados como están.

Por ejemplo: (<número>,), número es un elemento opcional y se requiere especificar la coma.

Lo que significan los siguientes elementos es mostrado a continuación:

<comentario>	Estatuto del Campo de Comentario.
<etiqueta>	Etiqueta.
<expresión>	Expresión del Ensamblador.
<expr>	Expresión del Ensamblador.
<número>	Constante numérica.
<string>	Cadena de Caracteres ASCII.
<delimitador>	Delimitador de string.
<opción>	Opción del Ensamblador.

En la descripción de las directivas se sigue el siguiente orden:

- Mnemónico y Nombre de la Directiva en Inglés.
- Formato o sintaxis.
- Descripción de la directiva.

4.5.2.1. BSZ (Block Storage Zeros)

(<etiqueta>) BSZ <expresión> (<comentario>).

La directivas BSZ y ZMB causan que el Ensamblador distribuya un bloque de bytes, y a cada byte le es asignado el valor inicial de cero. El número de bytes es especificado por el valor de la expresión en el campo del operando. Un error será indicado si la expresión tiene un valor de cero y si contiene etiquetas indefinidas más adelante.

4.5.2.2. EQU (Equate label to a value)

<etiqueta> EQU <expresión> (<comentario>).

La directiva EQU asigna a la etiqueta el valor de la expresión del Campo del Operando. Las etiquetas no deben ser redefinidas. Las expresiones con referencias adelante serán indicadas como error de fase.

4.5.2.3. FCB (Form Constant Byte)

(<etiqueta> FCB <expr> (,<expr>,...,<expr>) (<comentario>)

La directiva FCB puede tener uno o más operandos separados por comas. El valor de cada operando es truncado a 8 bits y es almacenado en un solo byte del Programa Objeto. Los caracteres múltiples son almacenado en bytes sucesivos. Los operandos pueden ser constantes numéricas, constantes de carácter, etiquetas o una expresión.

Ejemplo:

```
TABLA1  FCB  $00,$30,$5F.
```

4.5.2.4. FCC (Form Constant Character String)

(<etiqueta>) FCC <delimitador><string><delimitador>(<comentario>).

La directiva FCC es usada para almacenar una cadena de caracteres ASCII en bytes consecutivos de memoria. El almacenaje de bytes empieza en el valor actual del Contador de Programa y a la etiqueta se le asigna el valor del primer byte. Cualquier carácter ASCII imprimible puede formar parte de la cadena de caracteres (string). Como delimitadores se pueden usar dos caracteres idénticos ASCII.

Ejemplo:

```
Etiqueta1  FCC  ,ABC,
```

4.5.2.5. FDB (Form Double Byte Constant)

(<etiqueta>) FDB <expr>(,<expr>,...,<expr>) (<comentario>).

La directiva FDB puede tener uno o más operandos separados por comas. Un valor de 16 bits correspondiente a cada operando es almacenado en dos bytes consecutivos del Programa Objeto. El almacenaje empieza en el valor actual del Contador de Programa, y a la etiqueta se le asigna el primer valor de 16 bits. Los operandos múltiples son almacenados en bytes sucesivos. Los operandos pueden ser: constantes numéricas, constantes de carácter, etiquetas, o una expresión.

Ejemplo:

TABLA2 FDB $1004,$1030,$10FD.

4.5.2.6. FILL (Fill memory)

(<etiqueta>) FILL <expr>, <expr>.

La directiva FILL causa que el ensamblador inicialice un área de memoria con un valor constante. La primera expresión significa el valor de un byte que será colocado en memoria, y debe estar en el rango de 0 a 255. La segunda expresión indica el total de bytes que serán inicializados. Las expresiones no pueden contener etiquetas indefinidas o referenciadas más adelante en el programa.

4.5.2.7. OPT (Assembler Output Options).

OPT <opción>(,<opción>,...,<opción>) (<comentario>).

La directiva OPT es usada para controlar el formato de la salida del Ensamblador. Las opciones especificadas en el Campo del Operando son separadas por comas. Todas las opciones tienen una condición por omisión (default). Algunas opciones pueden ser inicializadas en la línea de comando que se utiliza para correr el Ensamblador, sin embargo las opciones contenidas en el Programa Fuente toman prioridad. En las siguientes descripciones aparece entre paréntesis (DEFAULT) si la opción es una condición por omisión. Todas las opciones deben ser escritas con letra minúscula.

c Habilita la cuenta de ciclos en el listado.

cre Imprime una tabla de referencia de cruce al final del listado. Esta opción debe ser especificada antes de la primera etiqueta utilizada en el programa Fuente.

l Imprime el listado desde el punto en que aparece.

noc (DEFAULT) Deshabilita el conteo de ciclos desde este punto en el listado.

nol (DEFAULT) No imprime el listado desde el punto en que aparece.

s Imprime la tabla de etiquetas al final del listado del Programa Fuente.

4.5.2.8. ORG (Set Program Counter to Origin).

ORG <expresión> (<comentario).

La directiva ORG cambia el valor del Contador de Programa al especificado en el Campo del Operando. Los estatutos que se encuentran después de la directiva ORG son ensamblados en localidades de memoria que empiezan en el nuevo valor del Contador de Programa. El Contador de Programa se inicializa en cero si no se encuentra la directiva ORG. La expresión no puede contener etiquetas indefinidas ni referencias hacia adelante.

4.5.2.9. PAGE (Top of Page).

PAGE

La directiva PAGE hace que el Ensamblador avance el papel hasta la parte superior de la siguiente página. Si el listado del programa no es producido, la directiva PAGE no tiene efecto. Esta directiva no es impresa en el listado del programa.

4.5.2.10. RMB (Reserve Memory Byte).

(<etiqueta>) RMB <expresión> (<comentario>).

La directiva RMB es utilizada para reservar espacio de memoria para etiquetas y es colocada después de la directiva ORG para que el Ensamblador asigne el espacio de memoria en la dirección especificada por la directiva ORG.

Ejemplo:

```
        ORG $100
TEMP    RMB 3
```

Descripción: la localidad 100 se *reserva para* la etiqueta TEMP, la localidad 101 se reserva para la etiqueta TEMP1 y la localidad 102 para la etiqueta TEMP2.

4.5.2.11. ZMB (Zero Memory Bytes).

(<etiqueta>) ZMB <expresión> (<comen*tario*>).

La directiva ZMB es idéntica a la directiva BSZ.

4.5.3. INVOCACION DEL ENSAMBLADOR.

Para correr el Programa Ensamblador teclee en la computadora la siguiente línea de comando:

AS11 *FUENTE1* (FUENTE2...) (-OPCION1 OPCION2...) >FUENTE1.LST

- **FUENTE1.ASM, FUENTE2.ASM**, etc. son los nombres de los programas fuente que se desean ensamblar. Convencionalmente utilizaremos la extensión .ASM para los nombres de los programas fuente

- Los elementos entre paréntesis "( )" son opcionales. Use un espacio en blanco antes del signo menos para separar las opciones del último nombre del programa fuente.

- Las **OPCIONES** son una o más de las siguientes:

l *Habilita el listado del programa.*

nol Deshabilita el listado del programa (default).

cre Habilita la creación de una tabla de referencia cruzada.

s Habilita la generación de la tabla de etiquetas.

c Habilita la cuenta de ciclos.

noc Deshabilita la cuenta de ciclos (default)

- El **Listado del Programa** o los mensajes de error pueden ser salvados como un archivo para su examen posterior. El Listado del Programa se obtiene agregando el comando de redireccionamiento indicado por el símbolo > (mayor que), seguido por el nombre del programa. Por costumbre se le coloca la extensión .LST al listado del programa.

El Programa Objeto creado, es salvado automáticamente con la extensión .S19, y se le da el nombre del primer programa fuente en la línea de comando. Si el Programa Objeto *no* es creado, *entonces* debe consultar el Listado del Programa para verificar que errores de ensamblado se produjeron. Si hay algún error aparece un mensaje explicativo. Los errores marcados Warning no cancelan el ensamblado pero son indicativos de un posible problema.

El Listado del Programa tiene el siguiente formato:

#LINEA DIR BYTES CODIGO OBJ. [#CICLOS] LINEA FUENTE

#LINEA es un número decimal de 4 dígitos usado como referencia en la referencia cruzada.

DIR es el valor hexadecimal de la dirección para el primer byte del código objeto de ésta instrucción.

BYTES CODIGO OBJ. son el código objeto de la línea fuente en hexadecimal. Si una línea fuente causa más de 6 bytes, estos son listados en líneas sucesivas sin que les preceda ninguna dirección.

#CICLOS aparece solo si está la opción "c". Los ciclos aparecen entre paréntesis rectangulares para distinguirlos de listado fuente.

LINEA FUENTE es reimpresa como en el Programa Fuente..

La Tabla de Etiquetas tiene el siguiente formato:

ETIQUETA DIR

La **ETIQUETA** es tomada directamente del campo de la etiqueta.

DIR es la dirección hexadecimal de la localidad que corresponde a la etiqueta.

La Tabla de Referencia Cruzada es impresa cuando es seleccionada la opción "cre", y tiene el siguiente formato:

ETIQUETA DIR *LOC1 LOC2 LOC3...

La ETIQUETA y DIR tienen el mismo significado que fue descrito antes. La primer localidad LOC1 en el listado, es marcada con un asterisco para indicar la línea donde la etiqueta fue definida. Las siguientes localidades LOC2 LOC3... son el número de la línea decimal del listado donde ocurre la etiqueta.

4.5.4. EJERCICIO DE ESCRITURA Y ENSAMBLADO DE UN PROGRAMA.

Mediante el Editor de Texto de su preferencia copie la escritura en letra realzada del siguiente programa que llamaremos SUMAYCCR.ASM. La escritura es realizada aplicando las reglas de sintaxis del ensamblador AS11 y se han hecho las siguientes consideraciones:

1. Se han separado ocho columnas para el Campo de la Etiqueta.
2. Se deja al menos un espacio en blanco de separación entre la etiqueta y la instrucción o directiva.
3. Se han separado ocho columnas para el Campo de Operación (instrucción o directiva).
4. Se deja al menos un espacio en blanco de separación entre la instrucción o directiva y el operando.
5. Se han separado ocho columnas para el Campo del Operando.
6. El Campo de Comentarios no es utilizado.

ETIQUETA		OPERACION		OPERANDO
*PROGRAMA QUE SUMA DOS NÚMEROS Y				
*MUESTRA EL EFECTO EN EL CCR				
		ORG		$40
		LDS		$9F
PUERTOB		EQU		$1004
REPITE		LDAA		#$18
		LDAB		#$13
		ABA		
		TPA		
		STAA		PUERTOB
		JMP		REPITE
FIN		BRA		FIN

Guarde en un disco el programa con el nombre SUMAYCCR.ASM

A continuación es revisado y descrito línea por línea este programa de ejemplo:

Línea 1 y línea 2.- *PROGRAMA QUE SUMA DOS NUMEROS Y *MUESTRA EL EFECTO EN EL CCR. Estas dos líneas empiezan con un asterisco en la columna 1 porque las líneas completas son comentarios.

Línea 3.- ORG $40. El Campo de la Etiqueta está vacío (en total 9 espacios en blanco). La Directiva Origen (ORG) indica que el programa empieza en la localidad $40.

Línea 4.- LDS $9F. El Campo de la Etiqueta está vacío. La instrucción LDS establece el valor inicial del Stack Pointer en la localidad $9F.

Línea 5.- PUERTOB EQU $1004. La etiqueta PUERTOB empieza en la columna 1 y es separada por 2 espacios en blanco de la directiva EQU que está en el Campo de Operación. La directiva EQU hace que la etiqueta PUERTOB sea igual; al valor $1004.

Línea 6.- REPITE LDAA #$18. La etiqueta REPITE empieza en la columna 1 y se utiliza para formar un ciclo en el programa. La etiqueta es separada por medio de 2 espacios en blanco de la instrucción LDAA. La instrucción LDAA es utilizada para cargar el acumulador A en forma inmediata con el número $18.

Línea 7.- LDAB #$13. El Campo de la Etiqueta está vacío. La instrucción LDAB permite cargar el acumulador B en forma inmediata con la cantidad $13.

Línea 8.- ABA. La instrucción ABA no requiere operando y es utilizada para sumar los contenidos de los acumuladores A y B. El resultado es colocado en el acumulador A y el registro CCR cambia debido a la operación de suma.

Línea 9.- TPA. Esta instrucción realiza la transferencia del contenido del registro CCR al acumulador A.

Línea 10.- STAA PUERTOB. La instrucción STAA envía el contenido del acumulador A al PUERTOB de salida.

Línea 11.- JMP REPITE. La instrucción JMP (jump) se utiliza para formar un ciclo al realizar un salto hacia donde se encuentra la etiqueta REPITE.

Línea 12.- FIN BRA FIN. Es conveniente utilizar ésta línea al final de un programa cuando se utiliza el Programa Monitor PCbug11.

Mediante el Programa Ensamblador AS11 ensamble el programa de ejemplo SUMAYCCR.ASM, utilice la siguiente línea de comando en la computadora:

AS11 SUMAYCCR.ASM -L S C CRE >SUMAYCCR.LST

Consulte el listado del programa y si existe algún error de ensamblado corríjalo hasta que no aparezcan errores.

A continuación se muestra como aparece el listado del programa después de ejecutar la línea de comando anterior:

```
0001                                    *PROGRAMA QUE SUMA DOS NUMEROS Y
0002                                    *MUESTRA EL EFECTO EN EL CCR
0003   0040                                      ORG     $40
0004   0040   9e   9f              [ 4 ]     LDS     $9F
0005   1004                                    PUERTOB EQU     $1004
0006   0042   86   18              [ 2 ]     REPITE  LDAA    #$18
0007   0044   c6   13              [ 2 ]             LDAB    #$13
0008   0046   1b                   [ 2 ]     ABA
0009   0047   07                   [ 2 ]     TPA
0010   0048   b7   10   04         [ 4 ]     STAA    PUERTOB
0011   004b   7e   00   42         [ 3 ]     JMP     REPITE
0012   004e   20   fe              [ 3 ]     FIN     BRA     FIN
0013                                              BRA     FIN

FIN     004e    *0012
PUERTOB
REPITE  0042    *0006
```

4.6 PROGRAMA MONITOR.

4.6.1. INTRODUCCION.

El Programa Monitor es colocado como herramienta final del Sistema de Desarrollo de Software, porque solamente puede ser utilizado hasta después de que el Programa Fuente escrito con un Editor de Texto, ha sido trasladado a Programa Objeto mediante un Programa Ensamblador. El Programa Monitor es utilizado para modificar y verificar que el software desarrollado por el usuario realiza la función esperada.

El **Programa Monitor PCbug11 de Motorola** es un paquete de software que permite correr los programas del usuario en cualquier MCU de la familia M68HC11, además facilita el examen de los periféricos internos bajo condiciones especificas, para la depuración de los programas hasta obtener el comportamiento deseado.

4.6.2. COMO TRABAJA EL PCBUG11.

El Programa Monitor PCbug11 es un paquete de software que permite las siguientes funciones:

- Realiza la interacción entre el MCU y la PC por medio de un programa de comunicación de bajo nivel que se instala en el MCU.
- Realiza la función de Monitoreo es decir: correr programas, leer y modificar memorias y registros. Y lo hace mediante un programa simple instalado en la PC que aprovecha las ventajas del hardware que sirve de soporte al MCU y lo sofisticado de la microcomputadora.

Existen dos métodos generales para usar los programas de comunicación que se instalan en el MCU:

1. El Método Cargador de RAM interna.
2. El Método Cargador de ROM.

4.6.2.1. Método Cargador de RAM Interna

En el **Método Cargador de RAM interna** el programa de comunicación es instalado en la RAM interna del MCU cada vez que se corre el PCbug11. Esto se realiza operando el MCU en el Modo de Operación Special Bootstrap

(El hardware de la figura 4.6.4.1 establece este modo de operación al conectar las terminales MODA y MODB a tierra). En este modo de operación, cada vez que se resetea el MCU se corre un programa instalado permanentemente en la ROM interna en las localidades $BF40 a $BFFF. Este programa establece la comunicación SCI (Serial Communications Interface) y hace posible que el PCbug11 instale el programa de comunicación en la RAM interna a partir de la localidad $00 y el programa de comunicación asume el control. Con el programa de comunicación instalado en la RAM es posible instalar programas en las memorias RAM, EEPROM y EPROM.

4.6.2.2. Método Cargador en ROM

En el Método Cargador en ROM, el programa de comunicación se instala en algún tipo de memoria ROM interna o externa. Para el MC68HC11A8 se dispone de memoria RAM desde la localidad $00 hasta $FF y memoria EEPROM desde $B600 hasta $B7FF. Debido a este limitado espacio de memoria, se recomienda para el desarrollo de prácticas instalar el programa de comunicación en la EEPROM, y éste será el método descrito. El MCU opera en el Modo de Operación Special Bootstrap pero en lugar de cargar un programa de comunicación que empiece en la localidad $00, al correr el PCbug11 con la **opción "programa de comunicación ya instalado"**, automáticamente se hace un salto a la localidad $B600 que es donde se inicia el programa de comunicación que previamente debe ser instalado en la EEPROM.

Más adelante en la sección 4.6.8. describe el procedimiento para instalar en la EEPROM el programa de comunicación TALKEREE.S19. Este programa se instala en las localidades de $B600 a $B6FF de la EEPROM, y utiliza parte de la RAM para instalar los vectores de interrupción en las localidades de $AO a $FE, además de establecer por default el Stack Pointer en la localidad $3F, dejando como AREA del Stack de $00 a $3F. Los programas del usuario deben respetar el uso de estas áreas para evitar fallas de comunicación. Vea el contenido del archivo TALKEREE.MAP para checar esas condiciones.

Estando instalado el Programa de Comunicación TALKEREE.S19 en la EEPROM, entonces el usuario dispone desde la localidad $40 hasta $9F en la RAM para poder correr, verificar y depurar sus programas hasta obtener el comportamiento deseado. Recuerde que los programas y datos almacenados en la RAM se pierden al desenergizar el MCU y que la EEPROM se utiliza para instalar programas en forma permanente (o casi permanente) y que no se pierden al desenergizar el MCU.

4.6.3. COMANDOS DEL PCBUG11.

A continuación se describen brevemente los comandos de PCbug11, deberá referirse al Manual de Usuario del PCbug11 para una explicación más detallada de los comandos. En la Tabla 4.6.3. los términos entre paréntesis "( )" son opcionales y las teclas especiales se representan entre paréntesis angulares "< >".

TABLA 4.6.3. Comandos del PCbug11.

Comando	Descripción
ASM addr (mneldir)	Ensamblador de líneas con opción de insertar mnemónicos o directivas.
BAUD (rate).	Muestra o fija el Baud rate (velocidad de comunicación).
BF addr1(addr2) byte/word	Llena un bloque de memoria con byte o palabra.
BL	Lista los puntos de quiebre (breakpoints).
BR (addr (macroname)).	Muestra o establece puntos de quiebre (con opcional ejecución de macros).
CALL addr.	Ejecuta la subrutina en la dirección addr.
CLRM	Elimina todos los macros en el MCU.
CLS	Limpia la ventana principal.
CONTROL (parámetros).	Muestra o cambia los parámetros del sistema PCbug11.
DASM addr1 (addr2).	Desensambla desde la dirección addr1 (hasta la dirección addr2).
DB startaddr (endaddr).	Muestra la Memoria del MCU desde la dirección inicial startaddr (hasta la dirección final endaddr).
DEBUG	Palabra reservada.
DEFINE symbol value/address.	Define el valor de un símbolo o etiqueta.
DEFM macroname/TRACE/ AUTOSTART	Define un comando, el macro de Trazo o el macro de AUTO arranque.
DELM macroname/TRACE/ AUTOSTART	Borra un comando, el macro Trazo o el macro de Auto arranque.
DIR (mask).	Muestra el directorio del disco o el especificado por mask.
DOS (command).	Llama al sistema operativo DOS o ejecuta un comando de DOS.
EDIT macroname.	Edita un macro.
EEPROM (startaddr (endaddr)).	Muestra, desactiva o fija un rango de direcciones al cual se aplica el algoritmo EEPROM para modificación o escritura de esta memoria.

TABLA 4.6.3. Comandos del PCbug11. Continuación...

Comando	Descripción
EEPROM DELAY option.	Fija el tiempo de borrado o programación en la EEPROM.
EEPROM ERASE (option)(addr).	Muestra o cambia la función de borrar antes de escribir en la EEPROM.
EPROM (startaddr (endaddr)).	Muestra, desactiva o fija un rango de direcciones al cual se aplica el algoritmo EPROM para modificación o escritura de esta memoria.
EPROM DELAY option.	Fija el tiempo de borrado o programación en la EPROM.
FIND byte/word addr1 addr2.	Encuentra las veces que ocurre un byte o palabra entre las direcciones addr1 y addr2.
FIND mnemonic addr1 addr2.	Encuentra las veces que ocurre un mnemónico entre las direcciones addr1 y addr2.
G (addr).	Ejecuta el programa del usuario que empieza en la dirección addr.
HELP (command).	Muestra información de ayuda.
KLE	Elimina el último mensaje de error.
LOADM (filename (macroname)).	Carga definiciones de macros del usuario o por omisión del archivo PCBUG11.MCR.
LOADS filename (loadaddr).	Carga en la memoria de MCU los Programas Objeto en Código Máquina.
LS symbol.	Lista los símbolos o etiquetas.
LSTM (Mname/TRACE/ AUTOSTART).	Lista los nombres o definiciones de macros.
MD staraddr (endaddr).	Muestra la memoria del MCU.
MM addr.	Modifica la memoria en la dirección addr.
MOVE addr1 addr2 addr3.	Mueve la memoria del MCU entre las direcciones addr1 y addr2 a la dirección addr3.
MS addr byte/word (byte/word).	Fija en la memoria bytes o palabras.
MSG (string).	Muestra mensajes en la ventana principal.
NOBR (address).	Remueve todos los puntos de quiebre o los especificados.
PAUSE (ms).	Espera que cualquier tecla sea oprimida o espera un tiempo de retardo.
PRINT.	Muestra el número de la versión del PCbug11.
PROTECT (startaddr (endaddr)).	Muestra, desactiva o activa un rango de direcciones protegido contra escritura.
QUIT (Y).	Termina la sesión del PCbug11 (sin requerir confirmación).

TABLA 4.6.3. Comandos del PCbug11. Continuación...

Comando	Descripción
RD (T).	Muestra el valor actual en la ventana de registro, con el parámetro T los actualiza en el modo de Trazo.
RESET (addr).	Resetea el hardware de MCU con nuevos o actuales vectores de reset.
RESTART (option).	Inicializa el PCbug11 con las mismas o nuevas opciones.
RM.	Modifica los Registros.
RS.	Fija valores de los registros.
S.	Detiene la ejecución del programa.
SAVEM (filame).	Salva los macros.
SHELL (command).	Ejecuta comandos del Sistema Operativo.
T (addr).	Este comando permite la ejecución de las instrucciones deteniéndose en cada una.
TERM (X1 Y1 X2 Y2).	Emulador de la terminal en las coordenadas X1,Y1,X2,Y2
TYPE filename.	Muestra el contenido de los archivos en la ventana principal.
UNDEF symbol.	Remueve símbolos o etiquetas.
VER.	Muestra el número de la versión del PCbug11.
VERF filename (memaddr).	Verifica el programa objeto en el disco contra la memoria.
VERF ERASE addr1 (addr2).	Verifica que la memoria este borrada (su contenido debe ser FF hex).

4.6.4. PREPARACION DEL MCU Y DEL PCBUG11.

Antes de usar el MCU con el paquete de software de PCbug11 debe hacer lo siguiente:

- Preparar el circuito (hardware) del MCU. El circuito debe suministrar un voltaje regulado de 5v al MCU y debe tener instalado un circuito de interfaz RS-232. En la fig. 4.6.4.1. se muestra un posible circuito.

- Conectar la interfase al puerto 1 ó 2 de una computadora IBM PC o compatible.

- Instalar el paquete de software del Programa Monitor PCbug11 en la computadora.

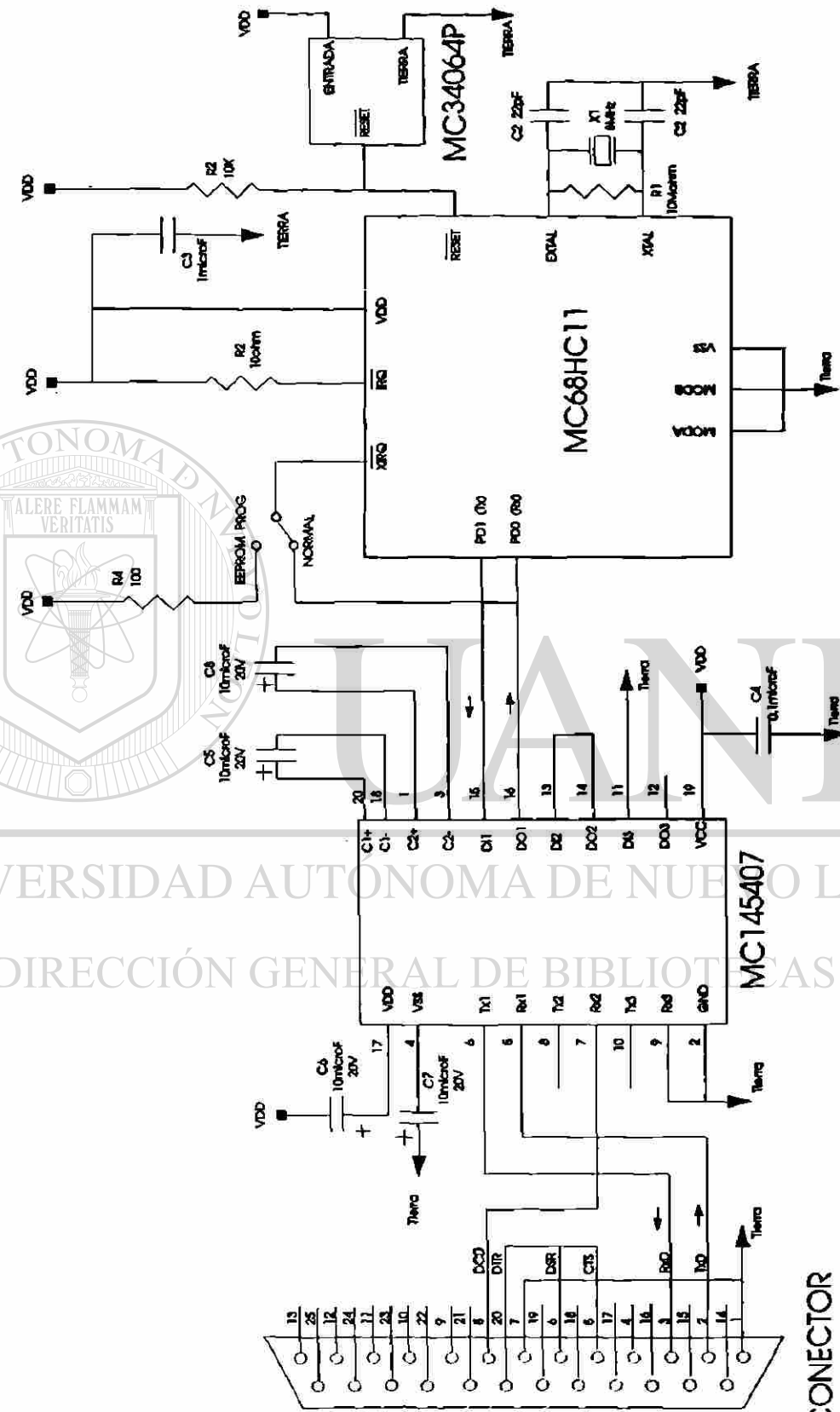


FIGURA 4.6.4.1. Diagrama del Hardware de Soporte del MC68HC11 para usarse con el Programa Monitor PCbug11.

4.6.5. INVOCACION DEL PCBUG11.

El PCbug11 es un paquete de software que hace posible muchas opciones: la elección del puerto utilizado, la elección de cristales de diferente frecuencia, el manejo de macros, la instalación de los programas del usuario en la memoria (RAM, EPROM o EEPROM) del MCU, etc.

En las descripciones siguientes: la *letra itálica* representa el texto de la computadora, la **letra negrita (Bold)** representa el texto que debe teclear como respuesta (seguido de la tecla ENTER en algunos casos) y el texto en letra normal precedido por una flecha, proporciona una explicación extra para el PCbug11 o el texto tecleado.

Para correr el PCbug11 en la computadora haga lo siguiente:

a. Realice la preparación del MCU y del PCbug11 indicada en el punto 6.4.

b. Encienda la computadora con el sistema operativo.

c. Coloque el disco con el paquete del software del PCbug11 en el drive A, B o C, aquí asumiremos que se instala en el drive B y que estamos en el drive C.

C:\> **B:** ⇦ se hace el cambio al drive B.

B:\> **PCBUG11** ⇦ se inicializa el PCbug11 y después de algunos segundos aparecen en la pantalla unos letreros solicitando respuesta a las opciones.

PCbug11 Ver 3.2.4a -M68HC11 Monitor for PC Hosts (c) Motorola Ltd 1991.

PCBUG11 Command Line Compiler.

Is the talker installed on your board? (Y/N) : ⇦ debido a que es la primera vez que usamos el PCbug11 supondremos que no hay instalado ningún talker previamente, teclee la letra **N.**

Do you wish to use the XIRQ interrupt? (Y/N) : ⇦ teclee la letra **N** (en el hardware del MCU deben estar conectadas juntas las terminales XIRQ y PD0).

MCU boot talkers available.

A 68HCIIK4	*B 68HC8IIA2*	*C 68HCIIE0*	*D 68HCIIE1*	*E 68HCIIE9*
F 68HC7IIE9	*G 68HCIIFI*	*H 68HCIIG5*	*I 68HCIIG7*	*J 68HCIIL6*
K 68HCIID3	*L 68HCIID0*	*M 68HCIIA0*	*N 68HCIIA1*	*O 68HCIIA8*
P 68HCIIE2	*Q 68HCIIA8*	*R 68HCIIE9*		

Wich device are you using?: ___

⇦ oprima la letra correspondiente al MCU que usa, no se preocupe si aparece otra letra en la pantalla, por ejemplo si oprimimos N aparece - A.

Do you wish to load a macro automatically? (Y/N): ___

⇦ por ser la primera vez que usamos el PCbug11 no debe haber ningún macro instalado, teclee la letra N.

PC communications part:

1. COM 1
2. COM 2

Wich communications port are you using?: __

⇦ teclee 1 ó 2 de acuerdo a su instalación y aparecerá en la pantalla port = 1 o port = 2 según corresponda.

Are you using an 8 MHz cristal? (Y/N): __

⇦ si la respuesta es N entonces preguntará la frecuencia en KHz del cristal utilizado en el Hardware. Supondremos que la frecuencia es 8 MHz, por lo tanto teclee la letra Y.

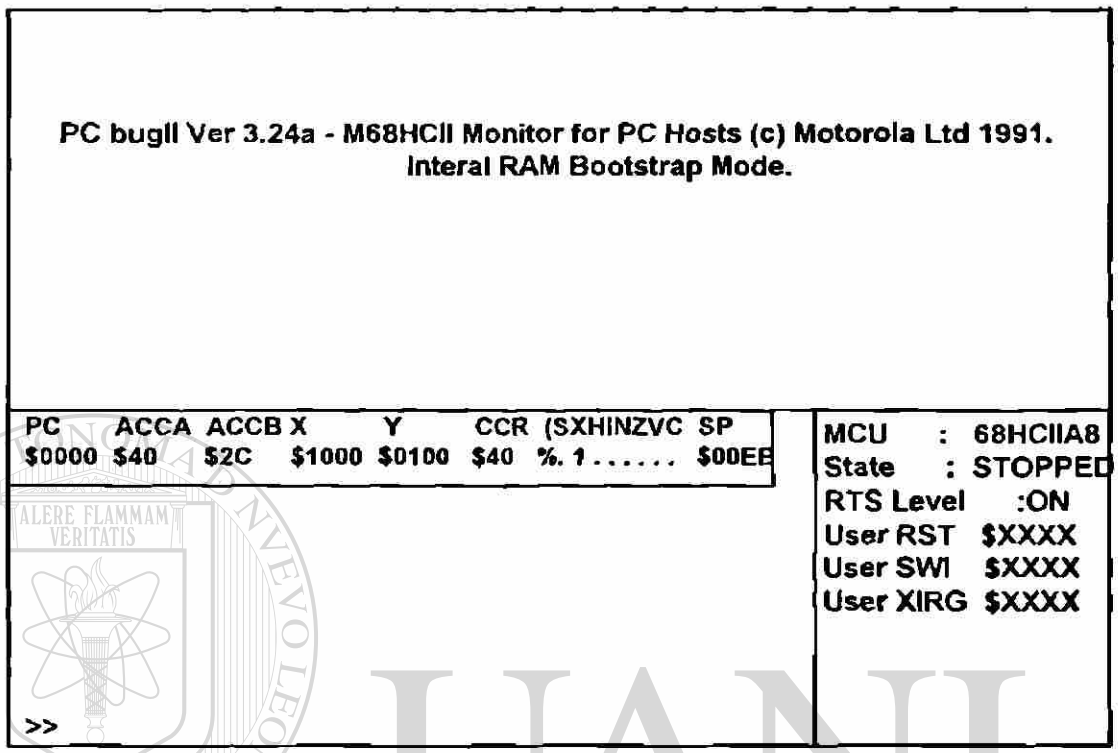
Command Line: **PCBUG11 - A**

⇦ Tecleando lo que aparece aquí subrayado despues del prompt *B:\>*, se puede correr el PCbug11 sin necesidad de teclear nuevamente todas las opciones: pero si cambia de MCU, de cristal, etc., entonces conteste las opciones. El botón de RESET del hardware del MCU debe ser oprimido antes de oprimir la tecla ENTER en la PC.

Press ESC to quit or any key to run PCBUG11

⇦ Antes de correr el PCbug11 debe oprimir el botón RESET del hardware de MCU. Presione cualquier tecla para correrlo (o la tecla ESC para salir).

Al correr el PCbug11 aparecen las siguientes ventanas en la pantalla.



4.6.6. Ventanas del PCbug11 en la Pantalla.

El estado del MCU y del hardware donde está soportado aparece en la pantalla de la computadora. Esta pantalla consiste de cuatro grandes áreas llamadas "Ventanas" que se describen enseguida:

Ventana Principal. Esta ventana está en la mitad superior de la pantalla, en ella aparece el texto en blanco fondo azul. En esta ventana aparece la mayor cantidad de información acerca de la operación del PCbug11, por ejemplo: el resultado de comandos, contenido de la memoria, ensamblado de códigos de operación (mnemónicos de instrucciones), macros, etc.

Ventana de Registros. Esta ventana se encuentra en el centro de la pantalla, con el texto en color blanco o amarillo sobre fondo rojo. En esta ventana se muestra el último contenido grabado en los registros del MCU. Tome nota que los valores en la Ventana de Registros son solamente los últimos datos ya sea después de arranque o cuando son requeridos por el usuario. Los comandos del PCbug11 le permiten modificar el contenido de los registros.

Ventana de Status se encuentra al centro a la derecha de la pantalla y en ella aparece el texto en color blanco con fondo morado. Esta ventana muestra: el tipo de MCU en uso, el estado del MCU (corriendo, detenido, trazo), el estado de la línea RS232 y el estado actual del conjunto de los vectores de interrupción del usuario.

Ventana de Comandos está en la parte inferior izquierda de la pantalla y en ella aparece el texto en color blanco sobre fondo negro. Use esta ventana para introducir y leer los comandos del PCbug11. El cursor de comandos es el carácter **">>"** y aparece en la línea de abajo de esta ventana; los comandos que se introducen mediante el teclado, aparecen después del curso de comandos. Los comandos previos y el último mensaje de error también aparecen en la ventana de comandos.

Hay dos ventanas adicionales temporales las cuales aparecen sobrepuestas en la ventana principal.

Ventana de Error. Esta ventana indica cualquier error u operación de comunicación incorrecta con el MCU. El texto aparece en color rojo con el fondo negro. Para limpiar la ventana inmediatamente, presione cualquier tecla o espere algunos segundos y se limpiará sola.

Ventana de Ayudas. Esta ventana muestra información de ayuda requerida por medio del comando HELP, el texto aparece en color blanco sobre fondo negro. Para desplazarse a través de la información use las teclas: ⇧ (up-arrow); ⇩ (down arrow), page-up, page-down. Para limpiar la pantalla use la tecla ESC.

4.6.7. EDICION DE LA LINEA DE COMANDOS.

Para hacer uso del Programa Monitor PCbug11 se utiliza el teclado de la computadora para escribir y editar en la línea de comandos. En la tabla 4.6.7.1. aparecen las teclas de edición con una breve explicación, un buffer retiene los últimos 16 comandos introducidos.

Las cuatro líneas arriba de la línea de comandos sirven para hacer un rastreo de los últimos cuatro comandos introducidos. La quinta *línea arriba de* la línea de comando muestra los puntos de quiebre (break points) y los códigos de error:

0: No error
1: Error de Verificación
2: Error MS o BF
3: Falla de comunicación del talker.

TABLA 4.6.7.1. Teclas de Edición de la Línea de Comandos.

TECLA	FUNCION
⇦ "left arrow"	Mueve el cursor un carácter hacia atrás.
⇨ "right arrow"	Mueve el cursor un carácter hacia adelante.
Home	Mueve el cursor al primer carácter.
End	Mueve el cursor al último carácter.
Delete left	Borra el carácter a la izquierda del cursor.
Del	Borra en la posición del cursor.
<CTRL> End	Borra desde la posición del cursor hasta el final de la línea.
Ins	Inserta en la posición del cursor.
⇧ Up arrow	Llama al comando previo en orden inverso
⇩ Down arrow	Llama al último comando en el buffer.
ESC	Limpia la línea de comando o termina muchos comandos en progreso (ASM, DASM, MD, etc.

4.6.8. EJEMPLO DE COMO INSTALAR UN PROGRAMA EN LA EEPROM.

En este ejemplo se instalará el programa de comunicación TALKEREE.S19 en la EEPROM, y se verá el uso de algunos comandos seguidos de una breve explicación. En las descripciones siguientes: la *letra itálica* representa el texto de la computadora, la **letra negrita (Bold)** representa el texto que debe teclear como respuesta, seguido de la tecla ENTER en algunos casos. El texto en letra normal precedido por una flecha, proporciona una explicación extra para el PCbug11 o el texto tecleado.

Procedimiento para instalar un programa en la EEPROM:

- Realice todos los pasos de la sección 4.6.5. Invocación de PCbug11, hasta que aparezcan las ventanas.

- Introduzca los siguientes comandos.

>>CONTROL BASE HEX ⇦ cambia la base numérica por default que es 10, a base hexadecimal y hace innecesario el poner el carácter $ antes de escribir cada número.

>>EEPROM B600 B6FF ⇦ habilita la escritura en un rango de B600 a B6FF (256 bytes para el programa de comunicación en la EEPROM) y hace que el *proceso de escritura sea automático* (transparente al usuario).

>>EEPROM ERASE ENABLE	⇦ habilita la función borrar antes de escribir. Si bien, ésta es una función por omisión (default).
>>LOADS TALKEREE.S19	⇦ carga el programa de comunicación TALKEREE.S19 en la EEPROM (cuando un programa ya tiene la extensión .S19, no es necesario teclear esta extensión).
>>VERF TALKEREE.S19	⇦ verifica que el programa fue cargado sin errores.
>>QUIT Y	⇦ sale de PCBUG11 sin pedir confirmación.

Ahora ya esta instalado el programa de comunicación TALKEREE en la EEPROM y se ha dejado un espacio libre en la EEPROM (de B700 en adelante) para programas del usuario.

Para usar el talker de comunicación TALKEREE instalado en la EEPROM, teclee en la computadora.

B:\> **PCBUG11 TALKEREE**	⇦ automáticamente se realiza la comunicación por medio del TALKEREE instalado en la EEPROM.

CAPITULO 5

PUERTOS PARALELO

5.1. INTRODUCCION.

La interacción del microcontrolador con el mundo real, se realiza por medio de la terminales de entrada/salida, que también son conocidas como terminales I/O (Input/Output). Todas estas terminales realizan funciones múltiples, que dependen de los datos en los Registros de Control y del Modo de Operación, permitiendo que la comunicación puede ser realizada en las diferentes modalidades de Comunicación. En las figuras 5.1.1. y 5.1.2. se puede observar la ubicación de las terminales.

5.2. PUERTOS.

Un Puerto es un conjunto de terminales utilizado para entrada o salida de datos en un MCU. Los Registros de Control permiten configurar los Puertos para que realicen diversas funciones, eso permite una mayor versatilidad y flexibilidad en el uso de las terminales, dando una mayor eficiencia al MCU.

Los puertos en los microcontroladores de la familia M68HC11, están caracterizados por estar localizados en el mapa de memoria, y pueden ser direccionados de la misma forma que cualquier localidad de memoria. Los Puertos y los Registros de Control forman un bloque de 64 localidades, después de resetear el MCU en el Modo de Operación Cargador (Special Bootstrap), ocupan desde la dirección $1000 hasta $103F. En la tabla 5.2. se muestra la ubicación de los Puertos, Registros de Control y la asignación de los bits de control.

Las Terminales I/O se agrupan en:

a. Terminales de habilitación o señalización (Strobe A y Strobe B).
b. Puertos (A, B, C, D y E).

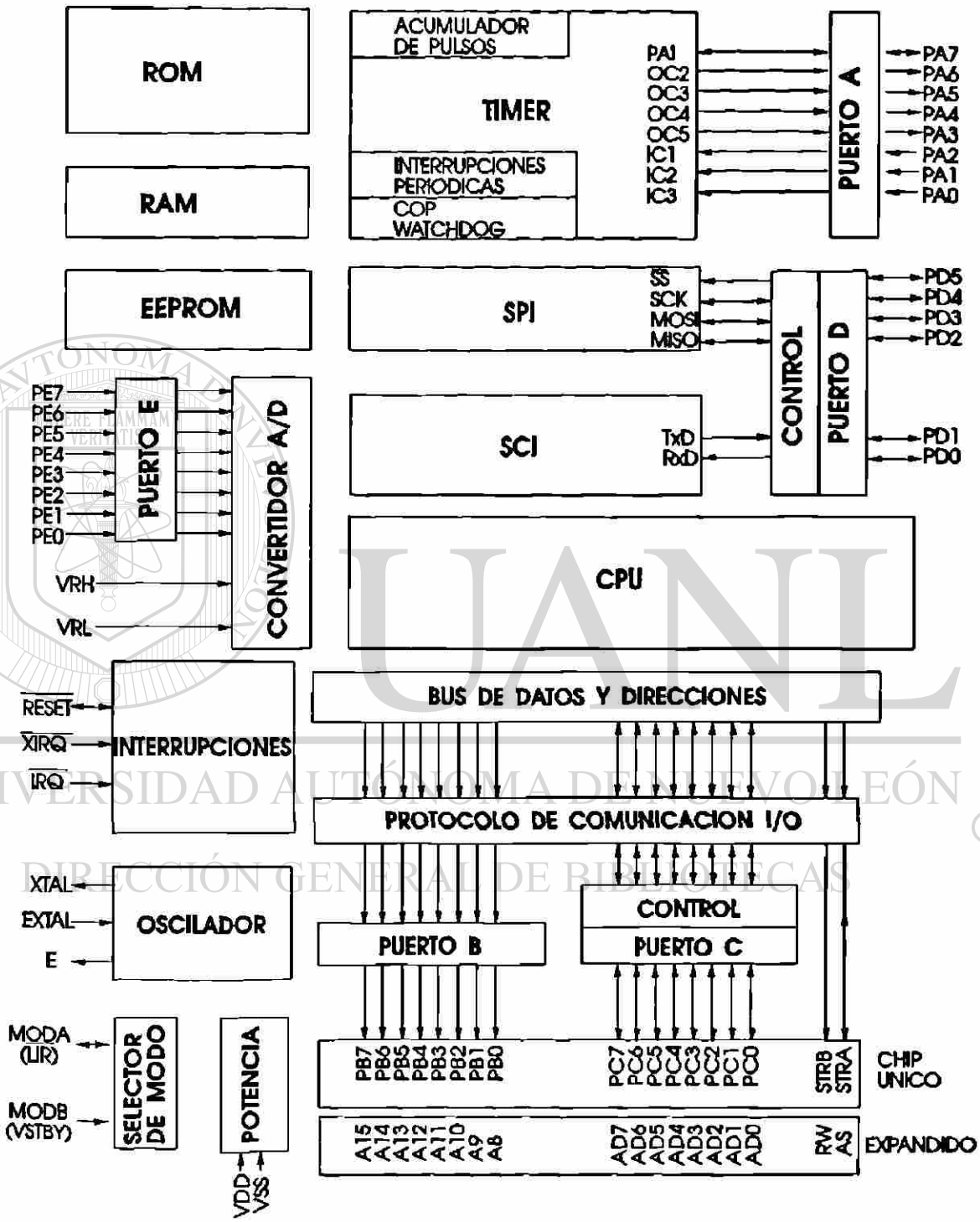
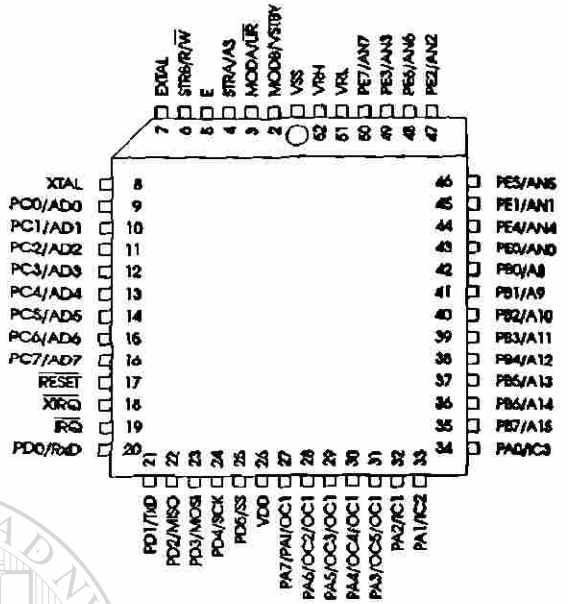


Figura 5.1.1. Diagrama de Bloques.



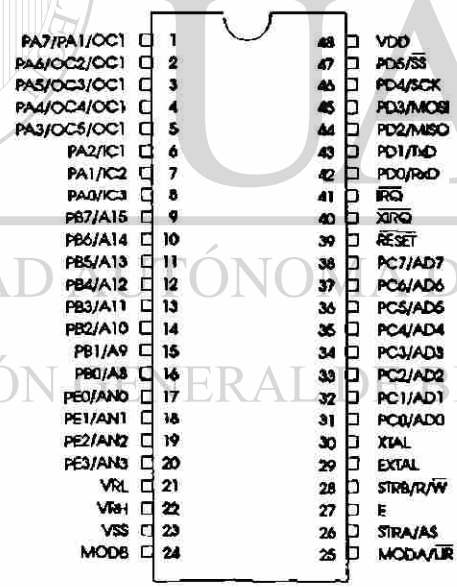


Figura5.1.2. Asignación de terminales MC68HC11A8

TABLA 5.2. BLOQUE DE REGISTROS DEL MC68HC11

localidad	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	Registro
$1000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PUERTO A
$1001									RESERVADO
$1002	STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB	PIOC
$1003	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PUERTO C
$1004	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	PUERTO B
$1005	PCL7	PLC6	PCL5	PCL4	PPCL3	PCL2	PCL1	PCL0	PUERTO CL
$1006									RESERVADO
$1007	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDCR
$1008	0	0	PD5	PD4	PD3	PD2	PD1	PD0	PUERTO D
$1009	0	0	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0	DDRD
$100A	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	PUERTO E
$100B	FOC1	FOC2	FOC3	FOC4	FOC5	0	0	0	CFORC
$100C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3	0	0	0	OC1M
$100D	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3	0	0	0	OC1D
$100E	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	TCNT(high)
$100F	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	TCNT(low)
$1010	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	TIC1(high)
$1011	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	TIC1(low)
$1012	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	TIC2(high)
$1013	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	TIC2(low)
$1014	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	TIC3(high)
$1015	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	TIC3(low)
$1016	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	TOC1(high)
$1017	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	TOC1(low)
$1018	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	TOC2(high)
$1019	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	TOC2(low)
$101A	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	TOC3(high)
$101B	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	TOC3(low)
$101C	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	TOC4(high)
$101D	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	TOC4(low)
$101E	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	TI4O5(high)
$101F	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	TI4O5(low)
$1020	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5	TCTL1
$1021	EDG4B	EDG4A	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A	TCTL2
$1022	OC1I	OC2I	OC3I	OC4I	I4O5I	IC1I	IC2I	IC3I	TMSK1
$1023	OC1F	OC2F	OC3F	OC4F	I4O5F	IC1F	IC2F	IC3F	TFLG1
$1024	TOI	RTII	PAOVI	PAII	0	0	PR1	PR0	TMSK2
$1025	TOF	RTIF	PAOVF	PAIF	0	0	0	0	TFLG2
$1026	DDRA7	PAEN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0	PACTL
$1027	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	PACNT
$1028	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
$1029	SPIF	WCOL	0	MODF	0	0	0	0	SPSR

Continuación...

localidad	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	Registro
$102A	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	SPDR
$102B	TCLR	0	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0	BAUD
$102C	R8	T8	0	M	WAKE	0	0	0	SCCR1
$102D	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SCCR2
$102E	TDRE	TC	RDRF	IDLE	OR	NF	FE	0	SCSR
$102F	R7/T7	R6/T6	R5/T5	R4/T4	R3/T32	R2/T2	R1/T1	R0/T0	SCDR
$1030	CCF	0	SCAN	MULT	CD	CC	CB	CA	ADCTL
$1031	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	ADR1
$1032	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	ADR2
$1033	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	ADR3
$1034	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	ADR4
$1035	0	0	0	PTCON	BPRT3	BPTR2	BPRT1	BPTR0	BPROT
$1036-8									RESERVAD0
$1039	ADPU	CSEL	IRQE	DLY	CME	0	CR1	CR0	OPTION
$103A	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	COPRST
$103B	ODD	EVEN	0	BYTE	ROW	ERASE	EELAT	EEPGM	PPROG
$103C	RBOOT	SMOD	MDA	IRV	PSEL3	PSEL2	PSEL1	PSEL0	HPRIO
$103D	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0	INIT
$103E	TILOP	0	OCCR	CBYP	DISR	FCM	FCOP	TCON	TEST1
$103F	EE3	EE2	EE1	EE0	1	NOCOP	1	EEON	CONFIG

5.2.1. PUERTO A.

EL PUERTO A consta de 8 terminales; de PA0 a PA7, que corresponden del bit 0 al bit 7 (ver las figuras 5.1.1. y 5.1.2. para su ubicación). Estas terminales realizan funciones alternativas que incluyen:

1. I/O de Propósito General.
2. Operación en el Sistema del Timer (Temporizador).
3. Sistema Acumulador de Pulsos.

La principal aplicación del PUERTO A y sus Registros de Control es en operaciones que involucran funciones que dependen del tiempo como: frecuencia, sincronización, cuenta de eventos, etc.

5.2.2. PUERTO B.

EL PUERTO B consta de 8 terminales de salida de dirección fija, que corresponden de PB0 a PB7 (ver las figuras 5.1.1. y 5.1.2. para su ubicación) y realizan funciones alternativas que incluyen:

1. I/O de Propósito General.
2. Comunicación Paralelo de Control (Strobe) Simple.

El Puerto B es direccionado en la localidad $1004 como se muestra en la Tabla 5.2.1. En la sección 5.3. de este capítulo se hace una explicación más detallada de este puerto y sus registros de control.

5.2.3. PUERTO C.

El Puerto C consta de 8 terminales bidireccionales que corresponden de PC0 a PC7, además posee un registro interno de captura de datos (latch) llamado PUERTO CL. Las terminales del PUERTO C realizan funciones alternativas que incluyen:

1. I/O bidireccionales de Propósito General.
2. Comunicación Paralelo de Control Simple o Strobe Simple.
3. Comunicación Paralelo con Protocolo Total (full handshake).

El Puerto C es direccionado en la localidad $1003 y el PUERTO CL en la localidad $1005. En la sección 5.3. de este capítulo se da una explicación más detallada de estos puertos y sus registros de control.

5.2.4. PUERTO D.

EL PUERTO D consta de 6 terminales bidireccionales que corresponden de PD0 a PD5, las funciones alternativas que pueden realizar son:

1. I/O bidireccionales de propósito general.
2. Dos terminales (D0 y D1) para Comunicación Serie Asíncrona.
3. Cuatro terminales (D2 a D5) para Comunicación Serie Síncrona.

EL PUERTO D es direccionado en la localidad $1003 como muestra la Tabla 5.2.1. La función principal de este puerto es para comunicación serie.

5.2.5. PUERTO E.

EL PUERTO E consta de 8 terminales de entrada de dirección fija que corresponden de PE0 a PE7. Las funciones alternativas que puede realizar este puerto son:

1. I (Input) de propósito general.
2. Canales de Entrada para conversión ANALOGO-DIGITAL (A/D).

EL PUERTO E es direccionado en la localidad $100A como muestra la Tabla 5.2.1. En el capítulo 6 se verá una explicación más detallada de este puerto y sus registros de control. La aplicación principal de este puerto es como convertidor Análogo/Digital.

5.3. COMUNICACION PARALELO.

La Comunicación Paralelo permite que varias señales de salida o de entrada sean manipuladas simultáneamente. Por ejemplo: las 8 terminales del PUERTO B manejan simultáneamente cada una de ellas una señal de salida.

La Comunicación Paralelo puede subdividirse en dos grandes modos:

1. Comunicación Directa.
2. Protocolo de Comunicación o Handshake.

5.3.1. COMUNICACION PARALELO EN MODO DIRECTO.

La Comunicación Paralelo en Modo Directo tiene las siguientes características:

a Es el método de comunicación más simple y es el que se utiliza en las prácticas complementarias de este curso.

b Las terminales de los puertos son utilizadas como I/O de Propósito General. Es decir, que en las terminales de los puertos configurados como salidas, el MCU escribe los datos o señales de salida; y en las terminales configuradas como entradas, el MCU lee los datos o señales de entrada.

c El software establece en que momento se leen o escriben los datos en las terminales sin necesidad de señales externas adicionales.

d Los dispositivos (leds, relevadores, actuadores, etc.) conectados a las terminales de salida, deben ser lo suficientemente rápidos para "seguir" los cambios de las señales de salida.

e Las variaciones en las señales de entrada deben ser lo suficientemente lentas, para que el MCU las puede procesar mediante control del software antes de que cambien.

Hay un cuidado que se debe tener, y es que el MCU no debe enviar o escribir datos en un puerto habilitado como entrada, pues no tiene sentido. Sin embargo el PUERTO C es un caso especial, pues posee un registro de captura (latch) donde se almacena el dato escrito, y se envía a las terminales cuando se habilitan como salida.

En la Tabla 5.3.1. se muestra la configuración de las terminales de cada puerto después de resetear el M68HC11 en el Modo de Operación Cargador (Special Bootstrap).

TABLA 5.3.1. Configuración de las terminales después de resetear en el Modo de Operación Cargador (Special Bootstrap).

PUERTO	TERMINALES	CONFIGURACION
A	A0, A1, A2, A3, A7	Entradas.
	A4, A5, A6	Salidas.
B	B0 a B7	Salidas.
C	C0 a C7	Entradas.
D	D0, D1	Entradas y Salidas de la interfaz de Comunicación Serie SCI.
	D2 a D5	Entradas
E	E0 a E7	Entradas.

5.3.2. MODO DE COMUNICACION PARALELO CON PROTOCOLO DE COMUNICACION.

Este modo de comunicación está caracterizado porque, además de las terminales utilizadas como salidas o entradas, se requieren otras terminales para señalización o habilitación.

Los Modos de Comunicación Paralelo con Protocolo de Comunicación o Handshake, puede subdividirse en 2 tipos:

1 Control Simple o Strobe Simple.
2 Protocolo Total (Full handshake).

En el Modo de Comunicación Paralelo con Protocolo de Comunicación o Handshake se utilizan: El Registro de Control Paralelo I/O **PIOC (Parallel I/O** Control Register), el PUERTO B y el PUERTO C.

Debido al efecto que producen en el control y configuración de la comunicación paralelo, a continuación se describe el Registro PIOC y la función de cada uno de sus bits.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
PIOC	STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB	$1002
reset	0	0	0	0	0	0	1	1	

INVB-Invert Strobe B. Este bit selecciona que nivel de la señal de salida en la terminal STRB (strobe B) se considera activa

INVB=0. La señal strobe B es activa en baja (0 lógico).

INVB=1. La señal strobe B es activa en alta (1 lógico).

EGA-Active Edge for Strobe A. Este bit selecciona cuál transición de la señal es reconocida en la terminal de entrada STRA (strobe A).

EGA=0. La transición de caída es reconocida, y la de elevación ignorada.

EGA=1. La transición de elevación es reconocida, y la de caida ignorada.

PLS-Pulse/Interlocked Handshake Operation. Este bit permite seleccionar dos tipos de señales de salida en la terminal STRB (strobe B). Este bit no tiene significado si el bit HNDS=0

PLS=0. La operación Interlocked Handshake es seleccionada. La señal STROB (Strobe B) una vez activada, permanece activa hasta que la transición seleccionada de strobe A es detectada.

PLS=1. La Operación Pulsed Handshake es seleccionada. La señal strobe B es un pulso que permanece durante 2 ciclos de reloj E.

OIN-Output or Input Handshaking. Este bit se utiliza para seleccionar: Entrada con Protocolo Total,o Salida con Protocolo Total. No tiene significado cuando el bit HNDS=0.

OIN=0. Selecciona Entrada con Protocolo Total.

OIN=1. Selecciona Salida con Protocolo Total.

HNDS-Handshake Mode. Este bit se utiliza para seleccionar el tipo de protocolo de comunicación.

HNDS=0. Selecciona el Modo de Strobe Simple. La señal strobe A actúa para habilitar la captura (latch) de datos en el registro PUERTO CL. La señal strobe B produce un pulso de salida después de escribir en el PUERTO B.

HNDS=1. Selecciona el Modo de Entrada o Salida de Protocolo Total, involucra el PUERTO C, el bit OIN y las terminales STRA y STRB.

CWOM-Port C Wire-Or Mode. Este bit afecta las 8 terminales del Puerto C, y selecciona si las salidas actúan en forma normal CMOS para operación en Wire-Or, o funcionan como salidas drenador abierto.

CWON=0. Las salidas del Puerto C son CMOS normales.

CWOM=1. Las salidas del Puerto C actúan como drenador abierto.

STAI-Strobe A Interrup Enable Mask. Este bit permite habilitar interrupciones.

STAI=1. Cuando el bit I del registro CCR es cero, al activar el bit STAF se hace una requisición de interrupción.

STAI=0. Deshabilita la requisición de interrupción.

SATF-Strobe A Interrup Status Flag. Este bit se activa cuando una transición seleccionada ocurre en la terminal Strobe A.

5.3.2.1. Modo de Comunicación Paralelo con Strobe Simple.

A continuación mediante un ejemplo, se describe la operación de salida del PUERTO B del MCU1 y la operación de entrada del PUERTO C del MCU2 en Strobe Simple.

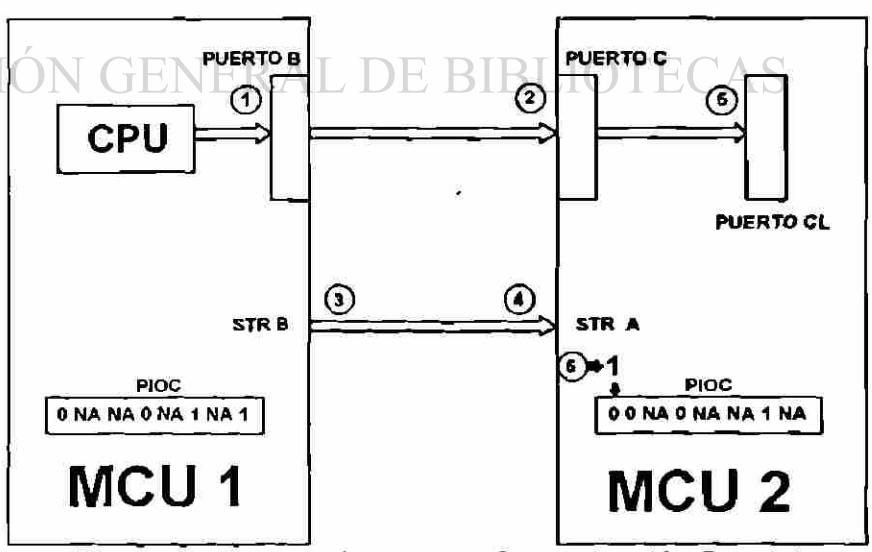


Fig.5.3.2.1.1.Diagrama de conexiones para Comunicación Paralelo

con Control Simple.

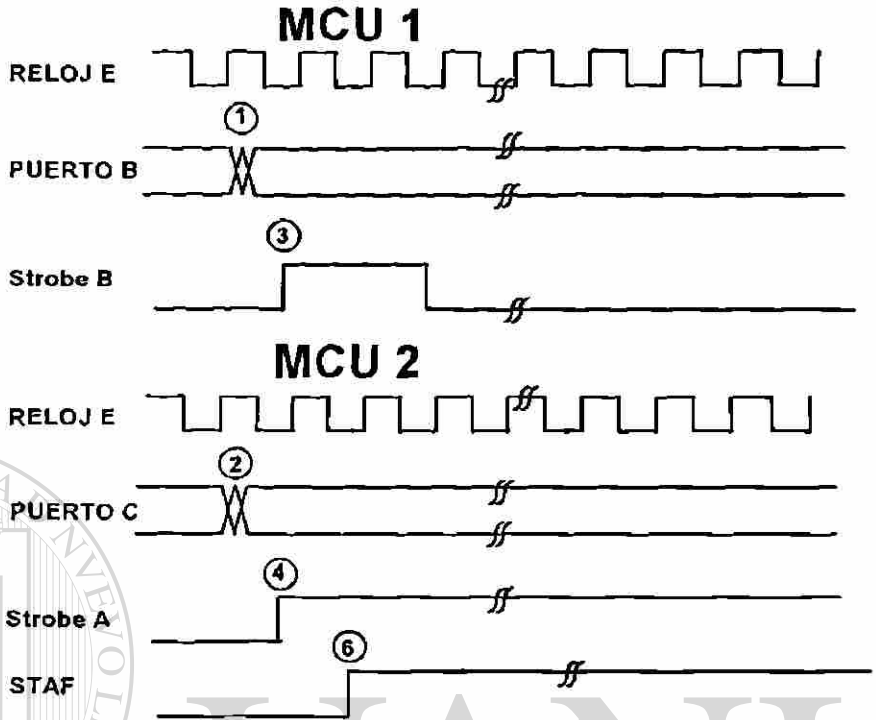


Figura 5.3.2.1.2 Diagrama de tiempos para las señales de los MCU y MCU2

En la tabla 5.3.2.3. se muestra el efecto de los valores de los bits de control de los Registro PIOC, que establecen las condiciones de operación para el Modo de Comunicación Strobe Simple de los MCU's de la figura 5.3.2.1.

Tabla 5.3.2.3. Efecto de los valores del los bits del Registro PIOC

M C U 1

Valores de los bits del registro PIOC	Efecto
STAF=0	Se mantiene en cero pues no recibe señal STRA
STAI=NA	No afecta
CWOM=NA	No afecta al Puerto B
HNDS=0	Comunicación Strobe Simple
OIN=NA	No afecta
PLS=1	Señal Strobe B pulsada
EGA=NA	No afecta la señal de salida.
INVB=1	Señal Strobe B activa en alta.

M C U 2

Valores de los bits del registro PIOC	Efecto
STAF=0	Se pone en 1 al recibir la señal STRA
STAI=0	No solicita interrupción al activarse STAF
CWOM=NA	No afecta
HNDS=0	Comunicación Strobe Simple
OIN=NA	No afecta.
PLS=NA	No afecta.
EGA=1	Captura (latch) la señal de entrada en el Puerto CL en la transición de elevación de la señal STRA.
INVB=NA	No afecta.

Para la descripción de la operación en el Modo de Comunicación Strobe Simple refiérase a las figuras 5.3.2.1.1. y 5.3.2.1.2. Obsérvese la concordancia entre los números encirculados con los siguientes incisos:

1. El programa del MCU1 escribe un dato en las terminales de salida del PUERTO B.

2. El dato es recibido en las terminales del PUERTO C del MCU2.

3. El MCU1 produce la señal strobe B con duración de 2 ciclos de reloj E.

4. La señal de salida Strobe B en la terminal STRB del MCU1 es recibida como señal de strobe A en la terminal STRA del MCU2. Como EGA=1 en el registro PIOC del MCU2, entonces se reconoce como señal la transición de elevación.

5. La señal strobe A ocasiona que los datos sean capturados (latched) en los registros del PUERTO CL.

6. Para el MCU2 las señales recibidas están fuera de sincronía respecto a sus señales de reloj E, por lo que activa el bit STAF para sincronizarla,e indicar que el dato ha sido recibido y está disponible.

5.3.2.2. Modo de Comunicación Paralelo con Protocolo Total.

Este modo de comunicación permite una comunicación más eficiente con otros dispositivos como: impresoras, computadoras, otros MCU's, etc. Pues además de los datos puede incluir señales que contienen toda o parte de la siguiente información:

- Que el MCU está listo para enviar el dato.
- El receptor contesta que está listo para recibir el dato.
- El MCU coloca el dato en las terminales de salida.
- El MCU envía una señal para que el receptor capture el dato.
- El receptor envía una señal confirmando haber recibido el dato.
- Después de procesar el dato, el receptor envía una señal indicando que esta listo para recibir otro dato.

A continuación se describe un ejemplo de Comunicación Paralelo con Protocolo Total (Full Handshake). En este ejemplo, la salida del MCU se utiliza como señal para una impresora. En la tabla 5.3.2.2.1. se muestra el efecto de los valores del Registro PIOC para configurar el microcontrolador para Comunicación con Protocolo Total.

Tabla 5.3.2.2.1. Efecto de los valores de los bits del Registro PIOC.

Valores de los bits del registro PIOC	Efecto
STAF=1	Este bit es puesto en 1 después de recibir la transición seleccionada en la terminal Strobe A
STAI=1	Habilita una interrupción para que al activarse el bit STAF otro dato sea transmitido.
CWON=0	Configura el Puerto C para salidas normales.
HNDS=1	Comunicación Paralelo con Handshake
OIN=1	Selecciona Salida con Protocolo Total.
PLS=1	La señal Strobe B es un pulso
EGA=0	La transición de caída es reconocida en la señal de Strobe A.
INVB=0	La señal Strobe B es activa en baja

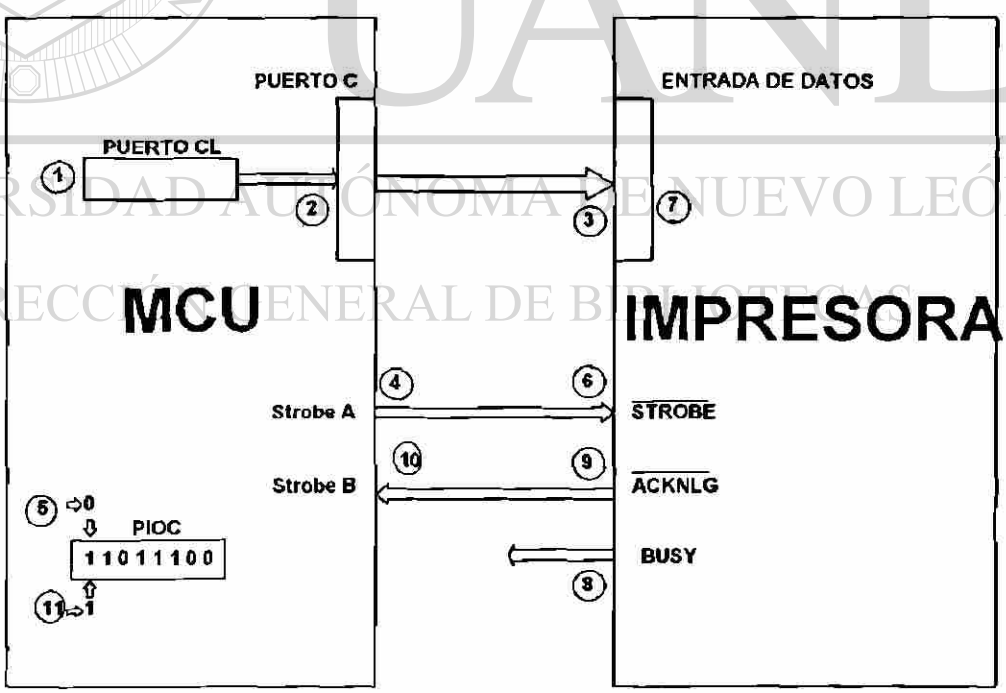


Figura 5.3.2.2.1. Diagrama de conexiones entre el MCU y la Impresora.

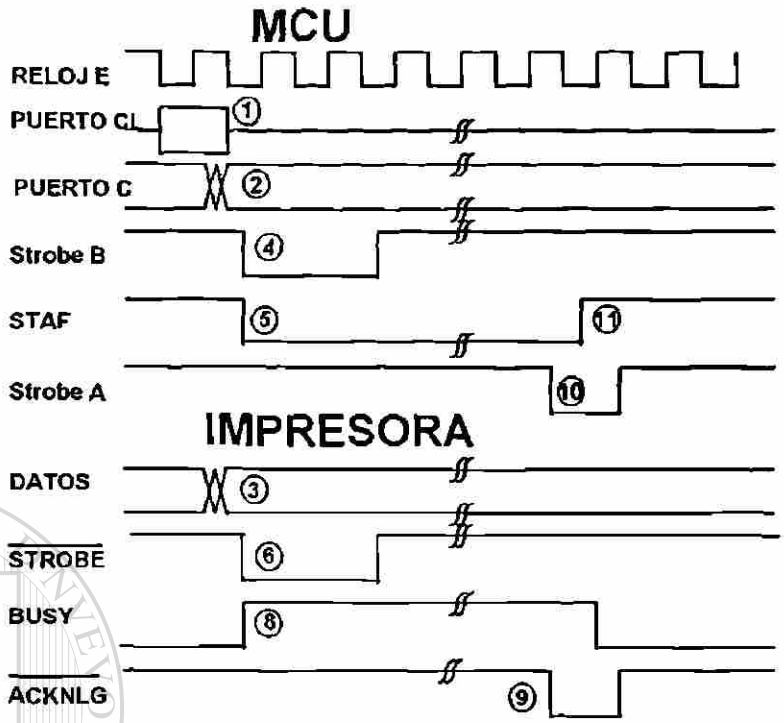


Figura.5.3.2.2.2. Diagrama de tiempos para el MCU y la impresora.

En la figura 5.3.2.2.1. se muestra el diagrama de conexiones entre el microcontrolador y la impresora. En la figura 5.3.2.2.2. se muestra el diagrama de tiempos para las señales transmitidas y recibidas en el MCU y la impresora. Observe la concordancia entre los números encirculados en ambas figuras con la descripción en los siguientes incisos.

1. El programa escribe un dato en el PUERTO CL.
2. Se envía el dato a las terminales del PUERTO C.
3. El dato aparece en las terminales de ENTRADA DE DATOS de la impresora.
4. Después de un tiempo suficiente para que se estabilice el dato, el MCU genera la señal Strobe B.
5. Se desactiva el bit STAF.
6. La señal Strobe B es recibida en la terminal $\overline{\text{STROBE}}$ de la impresora.
7. El dato es capturado (latch) por la impresora.
8. Se activa la señal BUSY indicando que la impresora está ocupada.

9. Después de imprimir el dato, la señal $\overline{\text{ACKNLG}}$ se activa en baja, indicando que puede recibir otro dato la impresora.

10. La señal $\overline{\text{ACKNLG}}$ se recibe en la terminal Strobe A, indicándo al MCU que puede enviar otro dato. Nota: la señal BUSY puede ser utilizada en lugar de la señal ACKNLG.

11. Debido a que la señal en Strobe A puede estar fuera de sincronía, hay que esperar a que el bit STAF se active en sincronía para que el MCU pueda enviar otro dato.

CAPITULO 6

CONVERTIDOR ANALOGO-DIGITAL

6.1.- INTRODUCCION.

Todas las variables físicas son de naturaleza analógica, y nos hemos acostumbrado a *movernos en este* amplio espectro de valores analógicos. Sin embargo, los circuitos internos de los MCU's son de naturaleza digital y manejan señales digitales. Por lo que se requiere que las señales pasen por un convertidor de valores analógicos a valores digitales.

6.2.- CONCEPTOS GENERALES.

En la figura 6.2.1. se puede observar una señal formada por un Tren de Pulsos de duración ts y período T. Estas dos señales después de ser manipuladas por un Modulador de Pulsos, producen una señal análoga muestreada, que consiste de un tren de pulsos modulados en amplitud y que también tienen una duración ts y período T. Precisamente este tren de pulsos modulados en amplitud, son manipulados por el Convertidor A/D (*Análogo/Digital*) para obtener la señal digitalizada, presentada generalmente en código binario o hexadecimal.

Es conveniente observar las siguientes diferencias entre la señal análoga continua y la señal digitalizada resultante:

a) La Señal análoga continua se representa gráficamente, y su valor puede ser conocido en cualquier instante de tiempo.

b) La Señal digitalizada es una lista de números, que nos proporciona el valor de la señal en instantes de tiempo específicos, en este caso cada T segundos.

En la figura 6.2.2. se muestra una vista amplificada de la señal análoga muestreada. El punto a corresponde al valor instantáneo que deseamos convertir a valor digital, y ese sería el valor convertido si el muestreador fuera ideal, es decir si ts=0. Pero el convertidor A/D del MC68HC11A8 utiliza un valor de ts=12 ciclos de Reloj E (ts=6μs para una frecuencia de Reloj E = 2 MHz). El valor en el punto b al final del tiempo de muestreo, es el valor que realmente es convertido a valor digital. Esta diferencia en valores debe ser tomada en cuenta cuando la señal análoga cambia muy rápidamente de valor.

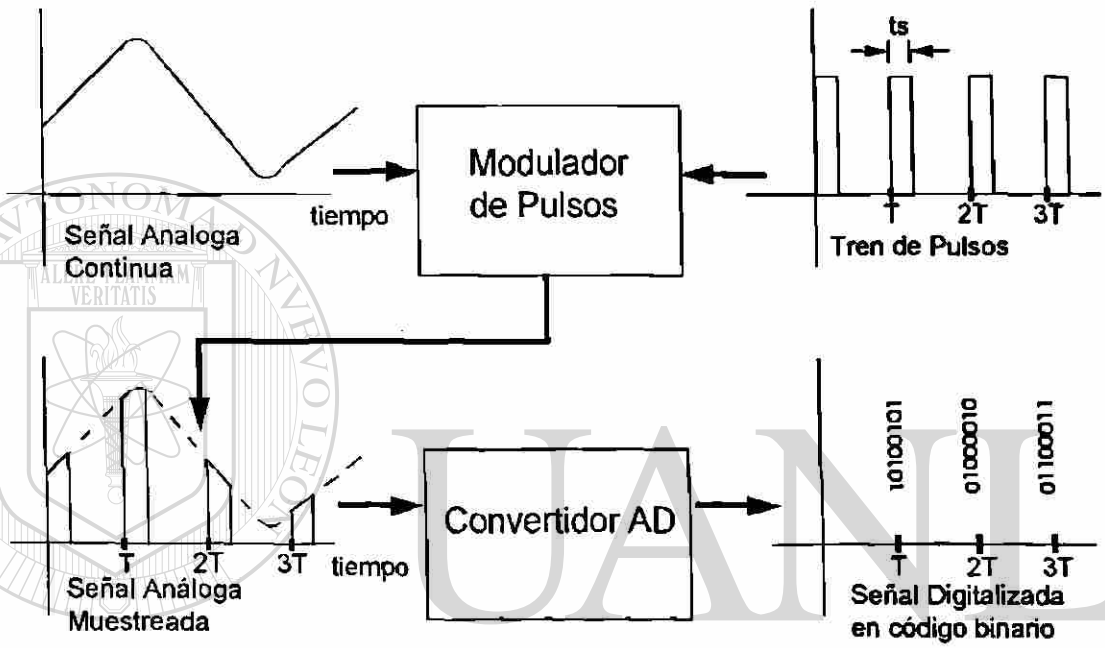


Figura 6.2.1 Proceso de Conversión Análogo Digital

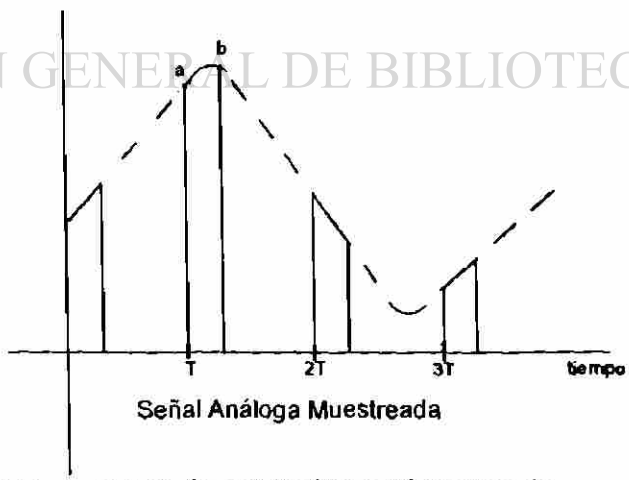


Figura 6.2.2. Señal Análoga Muestreada.

Después de muestrear la señal, el valor del punto b es mantenido durante todo el tiempo que toma la conversión, es decir 18 ciclos de Reloj E (para una frecuencia de Reloj E = 2MHz, el tiempo de conversión es de 9 µs). Después de realizada la conversión, el MCU utiliza 2 ciclos de Reloj E para almacenar el resultado en el Registro de Resultados. Un total de 32 ciclos de Reloj E, es el tiempo total que toma el proceso de muestreo-conversión-almacén de resultados (16 µs para una frecuencia de Reloj E de 2MHz).

Existen dos entradas para establecer voltajes de referencia analógicos para la conversión: V_{RH} y V_{RL}. El rango de conversión se establece de desde V_{RL} hasta V_{RH} que se convierte a un valor $FF de 8 bits (256 en decimal). Todos los valores analógicos que son convertidos digitalmente se expresan como un valor de 8 bits que puede variar de $00 a $FF. Esto corresponde a realizar un conteo de 256 partes, por lo que el error de conversión es de 1 bit que equivale a 1/256.

En la Tabla 6.2.1. se puede observar que el valor de salida digitalizado equivale a diferentes valores reales, lo cual depende de los valores de V_{RH} y V_{RL}.

TABLA 6.2.1.

V_{RH} = 5V V_{RL} = 0V	V_{RH} = 5V V_{RL} = 2.5 V	Valor Digital en Binario	Valor Digital en Hexadecimal
Valor Real Rango 0 - 5	Valor Real Rango 2.5. - 5		
5	5.0	11111111	FF
4	4.5	11001100	CC
3	4.0	10011001	99
2	3.5	01100110	66
1	3.0	00110011	33
0	2.5	00000000	00

El sistema A/D (Análogo/Digital) del MC68HC11A8 consiste de:

a. Dos terminales V_{RL} y V_{RH} utilizadas como entradas de voltaje de referencia.

b. Un solo convertidor A/D de redistribución de cargas totalmente capacitivo de aproximaciones sucesivas.

c. Un Multiplexador de Entrada para seleccionar uno de 16 canales (incluyendo los 8 canales asociados con cada terminal del PUERTO E).

d. Un Circuito de Control.

e. Cuatro Registro de Resultados.

f. Un oscilador interno RC.

6.3.- TERMINALES V_{RL} Y V_{RH}.

Estas terminales son utilizadas para proporcionar los voltajes de referencia para el Convertidor A/D y deben mantenerse dentro de los siguientes rangos para mayor precisión.

$V_{RH} \leq V_{DD}$ $V_{RL} \geq V_{SS}$ $5 \geq V_{RH} - V_{RL} \geq 2.5$.

6.4.- MULTIPLEXADOR DE ENTRADA.

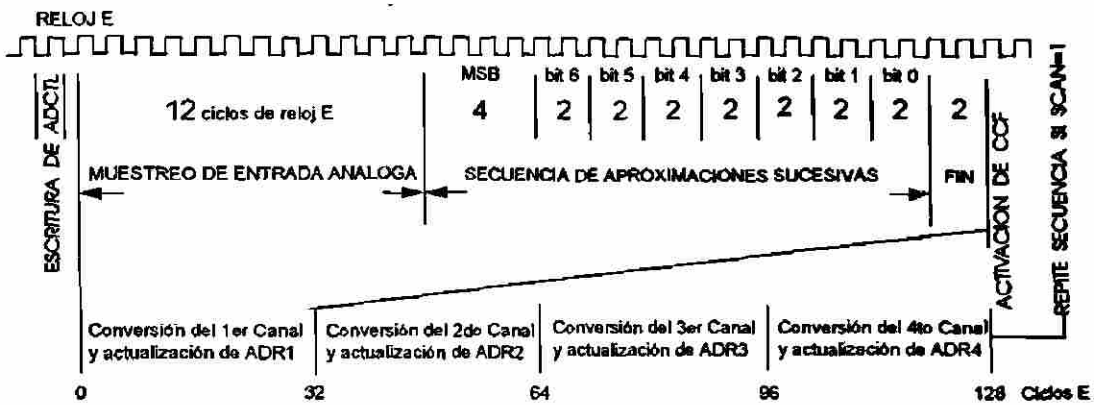
El Sistema Convertidor A/D posee un Multiplexador de Entrada que permite que el único Convertidor A/D seleccione una de 16 señales análogas, 8 de estos canales corresponden a las líneas de entrada del PUERTO E. De los 8 restantes: 4 son para puntos de referencia interna para pruebas y 4 son reservadas para usos futuros. Vea la tabla 6.5.1.1. para la asignación de canales.

6.5.- CIRCUITO DE CONTROL.

El Circuito de Control del Sistema de Conversión A/D permite que mediante diversos registros se realicen las siguientes funciones:

- Selección de canales para conversión A/D.
- Operación de Conversión en un Solo Canal o Multicanal, ambos con las modalidades: sin barrido continuo o con barrido continuo.
- Almacenamiento de Resultados A/D.
- Activación del Sistema A/D y Selección de un Oscilador RC interno.

En la figura 6.5.1. se muestra un diagrama de tiempos para la secuencia de Conversiones A/D, donde se puede observar que se toman como base de sincronía los pulsos de reloj E.



NOTAS: -El proceso de conversión se inicia después de escribir en el registro ADCTL
-Los resultados de las conversiones son colocados en el registro SAR y son transferidos al registro ADRX (X=1,2,3,4) al FINal del periodo de cada conversión
-La bandera CCF es puesta en 1 durante el FINal de la cuarta conversión.
-En esta figura se asume que CSEL=0 en el registro OPTION, por lo que los pulsos de reloj E actúan como sincronizadores.
-Si MULT=0 las cuatro conversiones en secuencia se realizan en el mismo canal

Figura 6.5.1 Secuencia de Conversión Análogo-Digital

Los registros involucrados en el proceso de conversión A/D son mostrados enseguida con la asignación de sus bits.

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	
$ 1030	CCF		SCAN	MULT	CD	CC	CB	CA	ADCTL
reset	0	0	I	I	I	I	I	I	I=Indefinido
$ 1031	-	-	-	-	-	-	-	-	ADR1
$ 1032	-	-	-	-	-	-	-	-	ADR2
$ 1033	-	-	-	-	-	-	-	-	ADR3
$ 1034	-	-	-	-	-	-	-	-	ADR4
$ 1039	ADPU	CSEL	IRQE	DLY	CME	-	CR1	CR0	OPTION
reset	0	0	0	1	0	0	0	0	

6.5.1.- SELECCION DE CANALES PARA CONVERSION A/D.

En el Registro A/D de Control y Status **ADCTL** (A/D CONTROL/STATUS REGISTER), se encuentran cuatro bits selectores de canales: CA, CB, CC y CD, que permiten seleccionar 1 de 16 canales de conversión A/D. En la tabla 6.5.1. se muestra la asignación de cada bit para seleccionar cada canal.

Cuando el modo multicanal es seleccionado haciendo el bit MULT=1, los dos bits menos significativos selectores de canal CA y CB, no tienen efecto, y los bits CC y CD especifican cual grupo de 4 canales es convertido.

Tabla 6.5.1.1. Asignación de canales Análogo-Digital.

CD	CC	CB	CA	Señal del Canal	Resultado en ADRx si MULT=1
0	0	0	0	AN0	ADR1
0	0	0	1	AN1	ADR2
0	0	1	0	AN2	ADR3
0	0	1	1	AN3	ADR4
0	1	0	0	AN4 **	ADR1
0	1	0	1	AN5 **	ADR2
0	1	1	0	AN6 **	ADR3
0	1	1	1	AN7 **	ADR4
1	0	0	0	Reservada	ADR1
1	0	0	1	Reservada	ADR2
1	0	1	0	Reservada	ADR3
1	0	1	1	Reservada	ADR4
1	1	0	0	VRH Pin*	ADR1
1	1	0	1	VRL Pin*	ADR2
1	1	1	0	(VRH)/2*	ADR3
1	1	1	1	Reservada	ADR4

* Este grupo de canales es usado durante pruebas de fábicación.
**No disponibles en versión empacada en 48 terminales.

6.5.2.- MODOS DE OPERACION DE CONVERSION.

La **Operación de Conversión en un Solo Canal** tiene dos variaciones: Sin Barrido Continuo y con Barrido Continuo.

La **Conversión A/D en un Solo Canal** Sin **Barrido Continuo** se obtiene por medio del registro ADCTL (los bits MULT=0 y SCAN=0). En este caso un solo canal es convertido 4 veces consecutivas, almacenando el primer resultado en el registro de resultados **ADR1** (**A/D** Result Register **1**), el segundo resultado en ADR2, el tercero en ADR3 y el cuarto en ADR4. Después de que se termina la cuarta conversión, se activa el bit de bandera de Conversión Completa **CCF** (Conversions Complete Flag) y se detiene toda actividad de conversión hasta que un nuevo proceso de conversión es ordenado al escribir en el registro ADCTL.

La **Conversión A/D de un Solo Canal Con Barrido Continuo** se obtiene con el registro ADCTL (los bits MULT=0 y SCAN=1). En esta variación la conversión continua es realizada, y el resultado del canal seleccionado en la quinta conversión es almacenado en el Registro de Resultados ADR1, remplazando el primer resultado de la conversión, la sexta conversión remplaza a la que existía en ADR2, y así sucesivamente.

La **Operación de Conversión Multicanal** tiene dos variantes: sin barrido continuo y con barrido continuo.

La **Conversión A/D Multicanal Sin Barrido Continuo** se obtiene con el registro ADCTL haciendo los bits MULT=1 y SCAN=0. El grupo de 4 canales es seleccionado con los bits CC y CD de acuerdo a la Tabla 6.5.1.1. y los bit CA y CB no tienen efecto. Los 4 canales seleccionados son convertidos uno cada vez, con el primer resultado almacenado en el registro ADR1, avanzando hasta que la cuarta conversión es almacenada en ADR4.

La **Conversión A/D Multicanal con Rastreo Continuo** se obtiene con el registro ADCTL (los bits MULT=1 y SCAN=1).

6.5.3. ACTIVACION DEL SISTEMA A/D

La activación del Sistema A/D se realiza mediante el bit 7 **ADPU (A/D Power Up)** del registro OPTION. Cuando ADPU es 1, el sistema A/D es habilitado y un retardo de 100 µs se requiere para que se estabilice.

6.5.4. SELECCION DEL OSCILADOR RC INTERNO.

La Selección de un Oscilador RC interno es requerida cuando la frecuencia de Reloj E es menos de 750KHz. Mediante el bit 6 **CSEL (Clock Select)** del Registro OPTION se selecciona la operación del Oscilador RC interno. Cuando CSEL=0 el sistema A/D usa el sistema de Reloj E, cuando CSEL=1 se activa el Oscilador RC interno para que maneje el sistema A/D a una frecuencia de 1.5 MHz.

GLOSARIO

Acumuladores A, B y D. Los acumuladores A y B son registros de 8 bit de propósito general usados para mantener los operandos y los resultados de cálculos aritméticos o de manipulación de datos. Estos dos acumuladores pueden ser concatenados en un solo acumulador de 16 bit llamado Acumulador D.

Apuntador de Pila SP (Stack Pointer) es un registro de 16 bit que contiene la dirección de la siguiente localidad libre en el stack (pila).

Arquitectura de la Computadora es el nombre dado a la organización funcional de un sistema básico de computadora y consta de 5 unidades:CPU o Unidad de Procesamiento Central. RAM o Memoria de Acceso Aleatorio. ROM o Memoria de Solo Lectura. BUSES o Conjunto de alambres que transportan información con un propósito común, y PUERTOS I/O o Puertos de Entrada/Salida.

AS11.EXE es un programa ensamblador de la compañia Motorola el cual puede ser utilizado en computadoras IBM-PC o compatibles y produce el Programa Objeto para los MCU's de la familia M68HC11,

Bit es un dígito binario, esta palabra se forma funcionando **binary digit** y es la mínima cantidad de información que se puede almacenar o manejar.

Bus de control es un conjunto de líneas con una mezcla de señales, cada una con un rol específico en el control de la actividad del sistema. Como regla, las señales de control son señales de sincronización proporcionadas por el CPU hacia los circuitos externos para sincronizar el movimiento de información entre los buses de dirección y de datos.

Bus de datos es el conjunto de líneas de doble sentido o dirección, que se utiliza para introducir o sacar datos de la CPU. El bus de datos transporta información entre el CPU y la memoria o entre el CPU y dispositivos I/O (Entrada/Salida).

Bus de direcciones es un conjunto de líneas que selecciona una cierta localidad de memoria, puerto de entrada o puerto de salida. El bus de direcciones es utilizado para operaciones de lectura y escritura,

Bus es un conjunto de alambres que transportan información con un propósito común. El acceso a los circuitos de control alrededor del CPU es proporcionado por tres buses: El bus de direcciones, el bus de datos y el bus de control.

Byte es el conjunto de 8 bits o dos nibble.

CCR (Condition Code Register) Registro de Código de Condiciones también recibe el nombre de **Registro de Banderas**, pues se les da el nombre de **Banderas** (flags) a los bits contenidos en el registro de condiciones. El registro CCR tiene: 5 indicadores de status (H, N, Z, V, C), 2 bits de interrupciones (I,X) y un bit habilitador de STOP (S). En seguida se describe cada una de las banderas.

Computadora puede ser definida como un sistema de procesamiento de datos, que puede ser programada para operar sin la intervención humana, y que tiene además la habilidad de almacenar y obtener datos.

Comunicación Paralelo permite que varias señales de salida o de entrada sean manipuladas simultáneamente.

Conjunto de Instrucciones es el grupo de instrucciones que un microprocesador o microcontrolador específico puede ejecutar. Está formado por un conjunto de códigos binarios que el CPU (Inherentemente en su diseño) es capaz de entender y ejecutar.

Constantes representan cantidades que no varian en valor durante le ejecución del programa.

Convertidor A/D (Análogo/Digital) es un dispositivo utilizado para obtener la señal digitalizada, presentada generalmente en código binario o hexadecimal, apartir de un tren de pulsos modulados en amplitud.

CPU (Central Procesing Unit) o Unidad de Procesamiento Central es el "cerebro" de la computadora, administra todas las actividades en el sistema y realiza todas las operaciones con los datos. La CPU es solamente una colección de circuitos lógicos que realizan en forma contínua solo dos operaciones: reconocer instrucciones (fetch instructions) y ejecutar instrucciones (executing).

Diagrama de Flujo es una representación gráfica que ilustra los pasos lógicos, cálculos y decisiones que en secuencia deben ser realizados para cumplir con una tarea específica.

DIP (Dual In-line Package) tipo de encapsulado de circuitos integrados

Directivas son instrucciones que controlan el funcionamiento del Programa Ensamblador y no son transferidas al Programa Objeto.

E señal de reloj usada como referencia de tiempo o de sincronización.

Editor de Texto es un programa residente en computadora que permite usarla como una máquina de escribir eléctrica. y es utilizado para escribir el Programa Fuente.

EEPROM: (Electrical Eraseble Programmable ROM) tipo de memoria puede ser borrada y programada electricamente.

Ensamblador es un programa que procesa el Programa Fuente (escrito en lenguaje ensamblador) y lo traduce en Programa Objeto (escrito en lenguaje de máquina) el cual es ejecutable por el MCU, además produce un listado del Programa.

EPROM: Memorias de Lectura Exclusiva Reprogramables, esta memoria pueden ser borrada con luz ultravioleta para ser reprogramada con nuevos patrones de bits.

Escritura, Operación de el CPU saca un dato y lo coloca en el bus de datos, debido a la señal de control, la memoria reconoce la operación como un ciclo de escritura y almacena el dato en la localidad especificada.

Expresión es una combinación de: operadores (algebraicos o lógicos), etiquetas y constantes.

Firmware es el nombre dado al software que es almacenado permanentemente en la memoria de programa.

Hardware es el equipo y constituye las unidades físicas

HCMOS = (High-density Complementary Metal Oxide Semiconductor) tecnología CMOS de alta densidad utilizada para fabricar circuitos integrados.

IRQ o de Requisición de Interrupción, terminal que proporciona un medio para requerir una interrupción asíncrona para el MCU.

K es la abreviación de kilo y es estandarizada para 2^{10}=1024,

Lectura Operación de recupera un byte de dato de una localidad especifica en la memoria y la coloca en el bus de datos. El CPU lee el dato y lo coloca en uno de sus registros internos.

Lenguaje de Máquina es un lenguaje binario que está constituido exclusivamente por unos y ceros, y es el único lenguaje que reconocen los MCU's.

Lenguaje Ensamblador es una colección de símbolos mnemónicos que representan: operaciones (mnemónicos de instrucciones del MCU y directivas para el ensamblador), etiquetas, operadores y símbolos especiales.

Lenguajes de Alto Nivel utilizan proposiciones en inglés que representan desde unas cuantas, hasta muchas instrucciones equivalentes de lenguaje ensamblador o lenguaje de máquina. algunos lenguajes de alto nivel son: BASIC, FORTRAN, Pascal, C, COBOL, etc.

LIFO (Last-In-First-Out) memoria,de lectura/escritura en donde el último dato en entrar es el primero en salir.

Localidad es el nombre dado a cada uno de estos registros de la memoria

LSI (Large Scale Integration) de gran escala de integración.

Memorias son circuitos capaces de almacenar grandes cantidades de información mediante un determinado número de registros que utilizan entradas y salidas comunes para el acceso a todos ellos.

Micro computadoras tienen como característica principal su tamaño reducido y el empacado del CPU, el cuál está contenido en un solo circuito integrado llamado microprocesador.

Microcontrolador es un circuito integrado que contiene: CPU, RAM, ROM, interfaz serie, interfaz paralelo, timer, y circuitos con un programa de interrupciones,

Microcontroladores son circuitos integrados LSI (Large Scale Integration) altamente desarrollados, cuyo diseño esta enfocado hacia actividades de control.

Minicomputadoras son computadoras que tienen un CPU consistente de varios circuitos integrados, lo cuál es necesario para obtener mayor velocidad y potencia de cómputo, además las minicomputadoras son multiusuario y ejecutan simultáneamente varios programas, lo cual es una ilusión resultante de un recurso de la CPU llamado "tiempo compartido".

Mnemónicos son abreviaturas o siglas de palabras (en inglés) que facilitan el recordar la función que realiza una instrucción y se utilizan para aproximar la lectura y escritura de programas al lenguaje habitual (en inglés).

Modelo del Programador es el nombre dado al conjunto de registros internos a que tiene acceso el programador. (Acumulador A, B y D, Registros Indice X e Y, Apuntador de Pila SP, Contador de Programa PC , Registro del Código de Condición CCR.

Modos de Direccionamiento son las diferentes formas en que se especifican el origen o destino de los datos que son manipulados por la CPU.

MSI (Medium Scale Intregration) de media escala de integración

Nibble es un conjunto de 4 bits, por lo tanto un byte tiene dos nibble.

PC (Program Counter) Contador de Programa es un registro de 16 bit que apunta siempre a la localidad de la siguiente instrucción que va a ser ejecutada.

PLCC (Plastic Leaded Chip Carrier) tipo de encapsulado de circuitos integrados

Pop es leer en la parte de arriba del Stack,

Programa en Lenguaje Ensamblador es el texto de una Programa Fuente.

Programa Fuente o Programa en Lenguaje Ensamblador es el texto donde los mnemónicos (ayudas de memorización) del fabricante son utilizados para representar las instrucciones del MCU.

Programa Monitor es un paquete de software utilizado para modificar y verificar que el software desarrollado por el usuario realiza la función esperada.

Programa Objeto es el nombre dado a un programa ya expresado en lenguaje de máquina.

Programación es la preparación de la lista de instrucciones que una computadora, microprocesador o microcontrolador debe ejecutar.

Programación Estructurada es una técnica de programación muy parecido a la estructura jerárquica de una corporación, donde se establece en forma descendente como se desea que el programa realice su trabajo.,

PROM: Memorias programadas por el usuario en el momento de diseñar el sistema. Solo pueden ser programadas una vez.

Puerto es un conjunto de terminales utilizado para entrada o salida de datos en un MCU.

PUERTO I/O es un conjunto de terminales utilizadas para entrada (Input) o salida (Output) de datos en un microcontrolador.

Pull es leer en la parte de arriba del Stack,

Push es escribir en la parte de arriba del Stack,

RAM (Random Access Memory) o memoria de acceso aleatorio es una memoria de lectura/escritura en donde se almacena en forma temporal los datos o algunos programas de naturaleza temporal.

RAM (Random Access Memory) son memorias de acceso aleatorio en las que es posible la escritura y lectura de datos.

Read o lectura es la operación de obtener el valor previamente almacenado en una localidad de memoria.

Registros Indice X e Y son de 16 bit, y son usados para el Modo de Direccionamiento Indexado.

RESET. señal usada como entrada para inicializar en un estado de arranque conocido

ROM (Read Only Memory) o Memoria de Solo Lectura habitualmente se utiliza para almacenar el programa en forma permanente. \

Señal digitalizada es una lista de números, que nos proporciona el valor de la señal en instantes de tiempo específicos,

Símbolo ⇦ indica que "es colocado en" o "transferido a".

Simbolo ☜ indica la siguiente instrucción que será ejecutada por la CPU.
símbolo #, indica direccionamiento inmediato.

Sistema de Desarrollo de Software es el conjunto de técnicas y herramientas que el programador utiliza en el proceso de desarrollo de software y consta de: Técnicas de Programación, Lenguajes de Programación, Editor de Texto, Programa Ensamblador, Programa Monitor.

Software es un término general utilizado para denominar a todos los programas, está constituido por una serie de instrucciones combinadas para realizar un trabajo importante.

Stack o Pila es un área seleccionada de memoria RAM, utilizado para almacenar en forma temporal datos o direcciones.

Stack Pointer (SP) o Apuntador de Pila es el registro utilizado para direccionar el Stack. El Stack Pointer o SP almacena (apunta hacia) la dirección de la última localidad vacía utilizable para almacenar datos.

Strobe es una señal de control.

Subrutinas son un bloque de instrucciones que son utilizadas repetidamente, y permiten de una manera sencilla, dividir el trabajo de un programa en secciones.

Talker es el nombre dado al programa de comunicación usado por el Programa Monitor PCbug11

Write o escritura es la operación de almacenar datos en memoria.

XIRQ o Interrupción No Enmascarable, terminal que proporciona un medio para requerir interrupciones no enmascarables

APENDICE. CONJUNTO DE INSTRUCCIONES DEL MC68HC11

Mnemonic	Operation	Boolean Espression	Addresing Mode for Operand	Machine Coding (hexadecimal)		Condition Codes							
				Opcode	Operand(s)	S	X	H	I	N	Z	V	C
ABA	Add Accumulators	A+B⇨A	INH	1B	dd mm	-	-	⇳	-	⇳	⇳	⇳	⇳
ABX	Add B to X	IX+00:B⇨IX	INH	3A		-	-	-	-	-	-	-	-
ABY	Add B to Y	IY+00:B⇨IY	INH	183A		-	-	-	-	-	-	-	-
ADCA	Add with Carry to A	A+M+C⇨A	A IMM	89	ii	-	-	⇳	-	⇳	⇳	⇳	⇳
			A DIR	99	dd								
			A EXT	B9	hh ll								
			A IND,X	A9	ff								
			A IND,Y	18 A9	ff								
ADCB	Add with Carry to B	B+M+C⇨B	B IMM	C9	ii	-	-	⇳	-	⇳	⇳	⇳	⇳
			B DIR	D9	dd								
			BEXT	F9	hh ll								
			B IND,X	E9	ff								
			B IND,Y	18 E9	ff								
ADDA	Add Memory to A	A+M⇨A	A IMM	8B	ii	-	-	⇳	-	⇳	⇳	⇳	⇳
			A DIR	9B	dd								
			A EXT	BB	hh ll								
			A IND,X	AB	ff								
			A IND,Y	18 AB	ff								
ADDB	Add Memory to B	B+M⇨B	B IMM	CB	ii	-	-	⇳	-	⇳	⇳	⇳	⇳
			B DIR	DB	dd								
			B EXT	FB	hh ll								
			B IND,X	EB	ff								
			B IND,Y	18 EB	ff								
ADDD	Add 16-bit to D	D+M:M+1⇨D	IMM	C3	jj kk	-	-	-	-	⇳	⇳	⇳	⇳
			DIR	D3	dd								
			EXT	F3	hh ll								
			IND,X	E3	ff								
			IND,Y	18 E3	ff								
ANDA	AND A with Memory	A•M⇨A	A IMM	84	ii	-	-	-	-	⇳	⇳	0	-
			A DIR	94	dd								
			A EXT	B4	hh ll								
			A IND,X	A4	ff								
			A IND,Y	18A4	ff								

CONJUNTO DE ISTRUCCIONES continuación...

Mnemonic	Operation	Boolean Expression	Addresing Mode for Operand	Machine Coding (hexadecimal)		Condition Codes							
				Opcode	Operand(s)	S	X	H	I	N	Z	V	C
ANDB	AND B with Memory	B•M⇒B	B IMM	C4	ii	-	-	-	-	⇕	⇕	0	-
			B DIR	D4	dd								
			B EXT	F4	hh ll								
			B IND,X	E4	ff								
			B IND,Y	18E4	ff								
ASL	Arithmetic Shift Left		EXT	78	hh ll	-	-	-	-	⇕	⇕	⇕	⇕
			IND,X	68	ff								
			IND,Y	18 68	ff								
ASLA			A INH	48									
ASLB			B INH	58									
ASLD	Arithmetic Shift Left Double		INH	05		-	-	-	-	⇕	⇕	⇕	⇕
ASR	Arithmetic Shift Rigth		EXT	77	hh ll	-	-	-	-	⇕	⇕	⇕	⇕
			IND,X	67	ff								
			IND,Y	18 67	ff								
ASRA	Arithmetic Shift Rigth		A INH	47									
ASRB	Arithmetic Shift Rigth		B INH	57									
BCC	Branch if Carry Clear	?C=0	REL	24	rr	-	-	-	-	-	-	-	-
BCLR	Clear Bit(s)	M•(mm)⇒M	DIR	15	dd mm	-	-	-	-	⇕	⇕	0	-
			IND,X	1D	ff mm								
			IND,Y	18 1D	ff mm								
BCS	Branch if Carry Set	?C=1	REL	25	rr	-	-	-	-	-	-	-	-
BEQ	Branch if = Zero	?Z=1	REL	27	rr	-	-	-	-	-	-	-	-
BGE	Branch if ≥ = Zero	?N⊕V=0	REL	2C	rr	-	-	-	-	-	-	-	-
BGT	Branch > Zero	?Z+(N⊕V)=0	REL	2E	rr	-	-	-	-	-	-	-	-
BHI	Branch if Higher	?C+Z=0	REL	22	rr	-	-	-	-	-	-	-	-
BHS	Branch if Higher or Same	?C=0	REL	24	rr	-	-	-	-	-	-	-	-
BITA	Bit(s) Test A with Memory	A•M	A IMM	85	ii	-	-	-	-	⇕	⇕	0	-
			A DIR	95	dd								
			A EXT	B5	hh ll								
			A IND,X	A5	ff								
			A IND,Y	18A5	ff								

CONJUNTO DE INSTRUCCIONES continuación..

Mnemonic	Operation	Boolean Expression	Addresing Mode for Operand	Machine Coding (hexadecimal)		Condition Codes							
				Opcode	Operand(s)	S	X	H	I	N	Z	V	C
BITB	Bit(s) Test B with Memory	B•M	B IMM B DIR B EXT B IND,X B IND,Y	C5 D5 F5 E5 18 E5	ii dd hh ll ff ff	-	-	-	-	⇕	⇕	0	-
BLE	Branch if ≤ Zero	?Z+(N⊕V)=1	REL	2F	rr	-	-	-	-	-	-	-	-
BLO	Branch if Lower	?C=1	REL	25	rr	-	-	-	-	-	-	-	-
BLS	Branch if Lower or Same	?C+Z=1	REL	23	rr	-	-	-	-	-	-	-	-
BLT	Branch if < Zero	?N⊕V=1	REL	2D	rr	-	-	-	-	-	-	-	-
BMI	Branch if Minus	?N=1	REL	2B	rr	-	-	-	-	-	-	-	-
BNE	Branch if Not = Zero	?Z=0	REL	26	rr	-	-	-	-	-	-	-	-
BPL	Branch if Plus	?N=0	REL	2A	rr	-	-	-	-	-	-	-	-
BRA	Branch Always	?1=1	REL	20	rr	-	-	-	-	-	-	-	-
BRCLR	Branch if Bit(s) Clear	?M•mm=0	DIR IND,X IND,Y	13 1F 18 1F	dd mm rr ff mm rr ff mm rr	-	-	-	-	-	-	-	-
BRN	Branch Never	?1=0	REL	21	rr	-	-	-	-	-	-	-	-
BRSET	Branch if Bit(s) Set	?(M)•mm=0	DIR IND,X IND,Y	12 1E 18 1E	dd mm rr ff mm rr ff mm rr	-	-	-	-	-	-	-	-
BSET	Set Bit(s)	M+mm⇨M	DIR IND,X IND,Y	14 1C 18 1C	dd mm ff mm ff mm	-	-	-	-	⇕	⇕	0	-
BSR	Branch to Subroutine		REL	8D	rr	-	-	-	-	-	-	-	-
BVC	Branch if Overflow Clear	?V=0	REL	28	rr	-	-	-	-	-	-	-	-
BVS	Branch if Overflow Set	?V=1	REL	29	rr	-	-	-	-	-	-	-	-
CBA	Compare A to B	A-B	INH	11		-	-	-	-	⇕	⇕	⇕	⇕
CLC	Clear Carry Bit	0⇨C	INH	0C		-	-	-	-	-	-	-	0
CLC	Clear Carry Bit	0⇨C	INH	0C		-	-	-	-	-	-	-	0
CLI	Clear Interrupt Mask	0⇨I	INH	0E		-	-	-	0	-	-	-	-
CLR	Clear Memory Byte	0⇨M	EXT IND,X IND,Y	7F 6F 18 6F	hh ll ff ff	-	-	-	-	0	1	0	0

CONJUNTO DE INSTRUCCIONES continuación...

Mnemonic	Operation	Boolean Expression	Addresing Mode for Operand	Machine Coding (hexadecimal) Opcode	Operand(s)	Condition Codes S	X	H	I	N	Z	V	C
CLRA		0⇨A	A INH	4F		-	-	-	-	0	1	0	0
CLRB		0⇨B	B INH	5F		-	-	-	-	0	1	0	0
CLV		0⇨V	INH	0A		-	-	-	-	-	-	0	-
CMPA	Compare A to Memory	A-M	A IMM	81	ii	-	-	-	-	⇕	⇕	⇕	⇕
			A DIR	91	dd								
			A EXT	B1	hh ll								
			A IND,X	A1	ff								
			A IND,Y	18 A1	ff								
CMPB	Compare B to Memory	B-M	B IMM	C1	ii	-	-	-	-	⇕	⇕	⇕	⇕
			B DIR	D1	dd								
			B EXT	F1	hh ll								
			B IND,X	E1	ff								
			B IND,Y	18 E1	ff								
COM	1's Complement Memory Byte	$FF-M⇨M	EXT	73	hh ll	-	-	-	-	⇕	⇕	0	1
			IND,X	63	ff								
			IND,Y	18 63	ff								
COMA	1's Complement A	$FF-A⇨A	A INH	43		-	-	-	-	⇕	⇕	0	1
COMB	1,s Complement B	$FF-B⇨B	B INH	53		-	-	-	-	⇕	⇕	0	1
CPD	Compare D to Memory	D-M:M+1	IMM	1A 83	jj kk	-	-	-	-	⇕	⇕	⇕	⇕
			DIR	1A 93	dd								
			EXT	1A B3	hh ll								
			IND,X	1A A3	ff								
			IND,Y	CD A3	ff								
CPD	Compare D to Memory	D-M:M+1	IMM	1A 83	jj kk	-	-	-	-	⇕	⇕	⇕	⇕
			DIR	1A 93	dd								
			EXT	1A B3	hh ll								
			IND,X	1A A3	ff								
			IND,Y	CD A3	ff								
CPX	Compare X to Memory	IX-M:M+1	IMM	8C	jj kk	-	-	-	-	⇕	⇕	⇕	⇕
			DIR	9C	dd								
			EXT	BC	hh ll								
			IND,X	AC	ff								
			IND,Y	CD AC	ff								

CONJUNTO DE INSTRUCCIONES continuación...

Mnemonic	Operation	Boolean Expression	Addresing Mode for Operand	Machine Coding (hexadecimal)		Condition Codes							
				Opcode	Operand(s)	S	X	H	I	N	Z	V	C
CPY	Compare Y to Memory	IY-M:M+1	IMM	18 8C	jj kk	-	-	-	-	⇕	⇕	⇕	⇕
			DIR	18 9C	dd								
			EXT	18 BC	hh ll								
			IND,X	1A AC	ff								
			IND,Y	18 AC	ff								
DAA	Decimal Adjust A	Adjust sum	INH	19		-	-	-	-	⇕	⇕	⇕	⇕
DEC	Decrement Memory Byte	M-1⇨M	EXT	7A	hh ll	-	-	-	-	⇕	⇕	⇕	-
			IND,X	6A	ff								
			IND,Y	18 6A	ff								
DECA	Decrement Accumulator A	A-1⇨A	A INH	4A		-	-	-	-	⇕	⇕	⇕	-
DECB	Decrement Accumulator B	B-1⇨B	B INH	5A		-	-	-	-	⇕	⇕	⇕	-
DES	Decrement Stack Pointer	SP-1⇨SP	INH	34		-	-	-	-	-	-	-	-
LDD	Load Double Accumulator D	M⇨A,M+1⇨B	IMM	CC	jj kk	-	-	-	-	⇕	⇕	0	-
			DIR	DC	dd								
			EXT	FC	hh ll								
			IND,X	EC	ff								
			IND,Y	18 EC	ff								
LDS	Load Stack Pointer	M:M+1⇨SP	IMM	8E	jj kk	-	-	-	-	⇕	⇕	0	-
			DIR	9E	dd								
			EXT	BE	hh ll								
			IND,X	AE	ff								
			IND,Y	18 AE	ff								
LDX	Load Index Register X	M:M+1⇨IX	IMM	CE	jj kk	-	-	-	-	⇕	⇕	0	-
			DIR	DE	dd								
			EXT	FE	hh ll								
			IND,X	EE	ff								
			IND,Y	CD EE	ff								
LDY	Load Index Register Y	M:M+1⇨IY	IMM	18 CE	jj kk	-	-	-	-	⇕	⇕	0	-
			DIR	18 DE	dd								
			EXT	18 FE	hh ll								
			IND,X	1A EE	ff								
			IND,Y	18 EE	ff								

CONJUNTO DE INSTRUCCIONES continuación...

Mnemonic	Operation	Boolean Expression	Addresing Mode for Operand	Machine Coding (hexadecimal)		Condition Codes							
				Opcode	Operand(s)	S	X	H	I	N	Z	V	C
LSL	Logical Shift Left		EXT	78	hh ll	-	-	-	-	⇕	⇕	⇕	⇕
			IND,X	68	ff								
			IND,Y	18 68	ff								
			A INH	48									
			B INH	58									
LSLD	Logical Shift Left Double		INH	05		-	-	-	-	⇕	⇕	⇕	⇕
LSR			EXT	74	hh ll	-	-	-	-	0	⇕	⇕	⇕
			IND,X	64	ff								
			IND,Y	18 64	ff								
LSRA			A INH	44									
LSRB			B INH	54									
LSRD	Logical Shift Left Double		INH	04		-	-	-	-	0	⇕	⇕	⇕
MUL	Multiply 8 by 8	AxB⇨D	INH	3D		-	-	-	-	-	-	-	⇕
NEG	2's Complement Memory Byte	0-M⇨M	EXT	70	hh ll	-	-	-	-	⇕	⇕	⇕	⇕
			IND,X	60	ff								
			IND,Y	18 60	ff								
NEGA	2's Complement A	0-A⇨A	A INH	40		-	-	-	-	⇕	⇕	⇕	⇕
NEGB	2,s Complement B	0-B⇨B	B INH	50		-	-	-	-	⇕	⇕	⇕	⇕
NOP	No Operation	No Operation	INH	01		-	-	-	-	-	-	-	-
ORAA	OR Accumulator A (inclusive)	A+M⇨A	A IMM	8A	ii	-	-	-	-	⇕	⇕	0	-
			A DIR	9A	dd								
			A EXT	BA	hh ll								
			A IND,X	AA	ff								
			A IND,Y	18 AA	ff								
ORAB	OR Accumulator B (Inclusive)	B+M⇨B	B IMM	CA	ii	-	-	-	-	⇕	⇕	0	-
			B DIR	DA	dd								
			B EXT	FA	hh ll								
			B IND,X	EA	ff								
			B IND,Y	18 EA	ff								
PSHA	Push A onto Stack	A⇨Stk,SP=SP-1	A INH	36		-	-	-	-	-	-	-	-
PSHB	Push B onto Stack	B⇨Stk,SP=SP-1	B INH	37		-	-	-	-	-	-	-	-
PSHX	Push X onto Stack (Lo First)	IX⇨Stk,SP=SP-2	INH	3C		-	-	-	-	-	-	-	-

CONJUNTO DE INSTRUCCIONES continuación...

Mnemonic	Operation	Boolean Expression	Addresing Mode for Operand	Machine Coding (hexadecimal)		Condition Codes							
				Opcode	Operand(s)	S	X	H	I	N	Z	V	C
PSHY	Push Y onto Stack (Lo First)	IY⇨Stk,SP=SP-2	INH	18 3C		-	-	-	-	-	-	-	-
PULA	Pull A from Stack	SP=SP+1,A⇦Stk	A INH	32		-	-	-	-	-	-	-	-
PULB	Pull B from Stack	SP=SP+1,B⇦Stk	B INH	33		-	-	-	-	-	-	-	-
PULX	Pull X from Stack (Hi First)	SP=SP+2,IX⇦Stk	INH	38		-	-	-	-	-	-	-	-
PULY	Pull Y from Stack (Hi First)	SP=SP+2,IY⇦Stk	INH	18 38		-	-	-	-	-	-	-	-
ROL	Rotate Left		EXT	79	hh ll	-	-	-	-	⇳	⇳	⇳	⇳
			IND,X	69	ff								
			IND,Y	18 69	ff								
ROLA			A INH	49									
ROLB			B INH	59									
ROR	Rotate Right		EXT	76	hh ll	-	-	-	-	⇳	⇳	⇳	⇳
			IND,X	66	ff								
			IND,Y	18 66	ff								
RORA			A INH	46									
RORB			B INH	56									
RTI	Return from Interrup		INH	3B		⇳	⇳	⇳	⇳	⇳	⇳	⇳	⇳
RTS	Return from Subroutine		INH	39		-	-	-	-	-	-	-	-
SBA	Subtract B from A	A-B⇨A	INH	10		-	-	-	-	⇳	⇳	⇳	⇳
SBCA	Subtract with Carry From A	A-M-C⇨A	A IMM	82	ii	-	-	-	-	⇳	⇳	⇳	⇳
			A DIR	92	dd								
			A EXT	B2	hh ll								
			A IND,X	A2	ff								
			A IND,Y	18 A2	ff								
SBCB	Subtract with Carry From B	B-M-C⇨B	B IMM	C2	ii	-	-	-	-	⇳	⇳	⇳	⇳
			B DIR	D2	dd								
			B EXT	F2	hh ll								
			B IND,X	E2	ff								
			B IND,Y	18 E2	ff								
SEC	Set Carry	1⇨C	INH	0D		-	-	-	-	-	-	-	1
SEI	Set Interrupt Mask	1⇨I	INH	0F		-	-	-	1	-	-	-	-
SEV	Set Overflow Flag	1⇨V	INH	0B		-	-	-	-	-	-	1	-
STAA	Store Accumulator A	A⇦M	A DIR	97	dd	-	-	-	-	⇳	⇳	0	-
			A EXT	B7	hh ll								
			A IND,X	A7	ff								
			A IND,Y	18 A7	ff								

CONJUNTO DE INSTRUCCIONES continuación...

Mnemonic	Operation	Boolean Expression	Addresing Mode for Operand	Machine Coding (hexadecimal)		Condition Codes							
				Opcode	Operand(s)	S	X	H	I	N	Z	V	C
STAB	Store Accumulator B	B⇨M	B DIR	D7	dd	-	-	-	-	⇕	⇕	0	-
			B EXT	F7	hh ll								
			B IND,X	E7	ff								
			B IND,Y	18 E7	ff								
STD	Store Accumulator D	A⇨M,B⇨M+1	DIR	DD	dd	-	-	-	-	⇕	⇕	0	-
			EXT	FD	hh ll								
			IND,X	ED	ff								
			IND,Y	18 ED	ff								
STOP	Stop Internal Clocks		INH	CF		-	-	-	-	-	-	-	-
STS	Store Stack Pointer	SP⇨M:M+1	DIR	9F	dd	-	-	-	-	⇕	⇕	0	-
			EXT	BF	hh ll								
			IND,X	AF	ff								
			IND,Y	18 AF	ff								
STX	Store Index Register X	IX⇨M:M+1	DIR	DF	dd	-	-	-	-	⇕	⇕	0	-
			EXT	FF	hh ll								
			IND,X	EF	ff								
			IND,Y	CD EF	ff								
STY	Store Index Register Y	IY⇨M:M+1	DIR	18 DF	dd	-	-	-	-	⇕	⇕	0	-
			EXT	18 FF	hh ll								
			IND,X	1A EF	ff								
			IND,Y	18 EF	ff								
SUBA	Subtract Memory from A	A-M⇨A	A IMM	80	ii	-	-	-	-	⇕	⇕	⇕	⇕
			A DIR	90	dd								
			A EXT	B0	hh ll								
			A IND,X	A0	ff								
			A IND,Y	18 A0	ff								
SUBB	Subtract Memory from B	B-M⇨B	B IMM	C0	ii	-	-	-	-	⇕	⇕	⇕	⇕
			B DIR	D0	dd								
			B EXT	F0	hh ll								
			B IND,X	E0	ff								
			B IND,Y	18 E0	ff								

CONJUNTO DE INSTRUCCIONES continuación...

Mnemonic	Operation	Boolean Expression	Addresing Mode for Operand	Machine Coding (hexadecimal)		Condition Codes							
				Opcode	Operand(s)	S	X	H	I	N	Z	V	C
SUBD	Subtract Memory from D	D-M:M+1⇨D	IMM DIR EXT IND,X IND,Y	83 93 B3 A3 18 A3	jj kk dd hh ll ff ff	-	-	-	-	⇕	⇕	⇕	⇕
SWI	Software Interrupt		INH	3F		-	-	-	1	-	-	-	-
TAB	Transfer A to B	A⇨B	INH	16		-	-	-	-	⇕	⇕	0	-
TAP	Transfer A to CCR	A⇨CCR	INH	06		⇕	⇕	⇕	⇕	⇕	⇕	⇕	⇕
TBA	Transfer B to A	B⇨A	INH	17		-	-	-	-	⇕	⇕	0	-
TEST	TEST (Only in Test Modes)	Address bus counts	INH	00		-	-	-	-	-	-	-	-
TPA	Transfer CCR to A	CCR⇨A	INH	07		-	-	-	-	-	-	-	-
TST	Test for Zero or Minus	M-0	EXT IND,X IND,Y	7D 6D 18 6D	hh ll ff ff	-	-	-	-	⇕	⇕	0	0
TSTA		A-0	A INH	4D		-	-	-	-	⇕	⇕	0	0
TSTB		B-0	B INH	5D		-	-	-	-	⇕	⇕	0	0
TSX	Transfer Stack Pointer to X	SP+1⇨IX	INH	30		-	-	-	-	-	-	-	-
TSY	Transfer Stack Pointer to Y	SP+1⇨IY	INH	18 30		-	-	-	-	-	-	-	-
TXS	Transfer X to Stack Pointer	IX-1⇨SP	INH	35		-	-	-	-	-	-	-	-
TYS	Transfer Y tp Stack Pointer	IY-1⇨SP	INH	18 35		-	-	-	-	-	-	-	-
WAI	Wait for Interrupt	Stack Regs&Wait	INH	3E		-	-	-	-	-	-	-	-
XGDX	Exchange D with X	IX⇨D,D⇨X	INH	8F		-	-	-	-	-	-	-	-
XGDY	Exchange D with Y	IY⇨D,D⇨IY	INH	18 8F		-	-	-	-	-	-	-	-

VIII.4. OTROS ASPECTOS.

Como este trabajo está basado en el uso de información y equipo de Motorola, traté de evitar conflictos legales con respecto a propiedad intelectual y otros relacionados, para lo cual se enviaron cartas solicitando permiso de su uso.

Como hasta la fecha no he recibido contestación, se sugiere que los posibles usuarios de estos apuntes revisen ese aspecto, e incluyo a continuación copia de las cartas enviadas.

TELEFAX

Monterrey, N.L., Abril 22 de 1994

MOTOROLA
Fax: (5) 282 2864

At'n: Ing. Arturo Muñoz

De acuerdo con nuestra conversación telefónica de hoy, le agradeceré su ayuda para conseguir de Motorola la autorización para el uso de la información de su M68HC11 Micro Controller Manual en la elaboración de mi tésis titulada "CURSO DE MICRO CONTROLADORES " para la obtención del Grado de Maestría en Ciencias en la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autonóma de Nuevo León.

Aradezco de antemano su atención y espero su respuesta.

Atentamente

René Briones

Ing. René Briones L.

Fax: (8) 339 2721
Tel: (8) 337 8333
Dirección:: Calle 7 No. 2133
Col. La Quinta
Cd. Guadalupe N.L:

TELEFAX

Monterrey, N.L., México; March 23, 1994

MOTOROLA
Fax: (512) 505 8811

Dear Sirs:

I will appreciate you help me to know the right people on Motorola Corporation in order to get authorization to copyrights some information from your M68HC11 Micro controller Manuals and use it to elaborate my thesis for Master Degree "CURSO DE MICROCONTROLADORES" from Universidad Autónoma de Nuevo León, Facultad de Ingeniería Mecánica y Eléctrica.

Thanks a lot

René Briones L.

Ing. René Briones L.
Phone & Fax: 01152 (8) 339 2721
Address: Calle 7 No. 2133
Col. La Quinta
Cd. Guadalupe N.L.

T.. REPORT

Apr.22 1994 4:49PM

RECEIVING FACSIMILE	MODE	PAGES	START TIME	USAGE TIME	RESULT
MONTERREY DE MEXI	TX	01	Apr.22 4:47PM	00'24	OK

TX REPORT

Mar.23 1994 11:31AM

RECEIVING FACSIMILE	MODE	PAGES	START TIME	USAGE TIME	RESULT
512 3328811	TX	01	Mar.23 11:31AM	00'33	OK