Conceptos Básicos.

A continuación se presentan una serie de conceptos y definiciones que es necesario conocer para comprender mejor el resto de los temas que se presentan en este texto.

Bit: Se considera a un digito binario con 2 posibles valores solamente (0 y 1), también es la cantidad mínima de información que puede almacenarse en una memoria electrónica. Eléctricamente se consideran 2 estados, encendido y apagado; En electrónica, considerando lógica TTL un bit puede tomar los valores de 0 = 0Volts y 1 = +5Volts

Byte: Es un arreglo o conjunto de 8 bits, se considera al byte como la unidad básica de almacenamiento de información. El valor o "significado" de cada bit dentro del arreglo depende del tipo de información que se este representando ya que cada bit puede tener un valor independiente o formar parte de un subconjunto de 3 ó mas bits donde su valor depende de la posición dentro del arreglo.

Nibble: (cuarteto) Es un conjunto ó arreglo de 4 bits. Se dice que un byte esta formado por un nibble bajo (los primeros 4 bits menos significativos) y un nibble alto (los 4 bits mas significativos)

Registro: Arreglo de flip-flops que sirve para almacenar datos o información binaria temporalmente y tiene un tamaño de palabra determinado. Los registros son una pequeña memoria de propósito especial y de alta velocidad utilizados para almacenar resultados intermedios y cierta información de control. Por ejemplo en todos los microprocesadores se tiene el registro llamado contador de programa y es el que apunta a la próxima instrucción a ejecutarse otro ejemplo son los registros acumuladores.

Localidad: Cuando se hace un arreglo ó conjunto de registros para formar una memoria, a cada registro se le da el nombre de localidad, la cual tiene una capacidad de almacenamiento (en bits) un contenido y una dirección, en suma es el espacio físico de una memoria donde se guarda un dato. También se le llama celda.

Dirección: Las memorias constan de cierto número de celdas, cada una de las cuales puede almacenar una porción de información. Cada celda tiene un número asociado denominado dirección, que la diferencía de las demás y por medio del cual los programas o el usuario, pueden referirse a ella.

Contenido: En este caso nos referimos al dato ó conjunto de bits almacenados en una localidad de memoria determinada

Memoria: Unidad de almacenamiento de datos. Conjunto de localidades o registro dentro de una misma unidad. Cada celda tiene una dirección asociada y la capacidad total de la memoria esta determinada por el numero de localidades que la conforman y la cantidad de bits que almacena cada localidad. Los sistemas digitales tienen capacidad de almacenar fácilmente grandes volúmenes de información digital por periodos de tiempo cortos y largos a través de dispositivos de memoria. Existen varios tipos y clasificaciones de las

memorias, la mas común es de memorias volátiles y no volátiles. La RAM es un tipo de memoria volátil mientras que un CDROM se considera un tipo de memoria no volátil (figura 1).

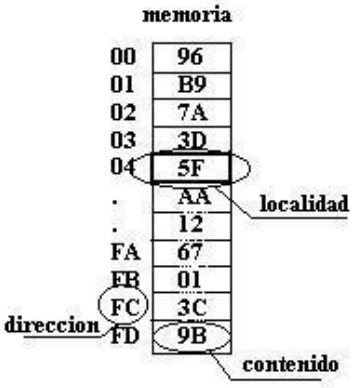


Figura 1

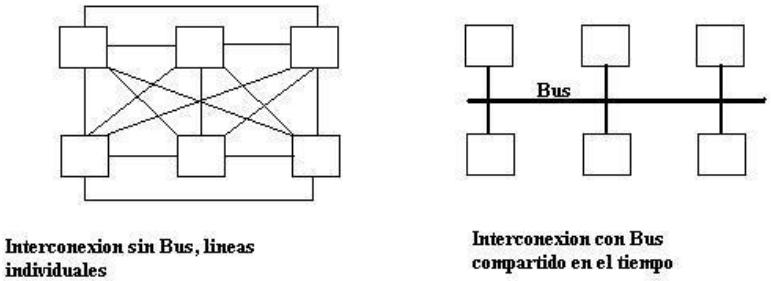


figura 2

Bus: Circuito o grupo de conductores eléctricos que constituyen un trayecto de comunicación entre dos o más dispositivos. Es circuitería distribuida a través de la máquina con el fin de mandar y recibir señales de control, direcciones de memoria y datos de un lugar a otro. En otras palabras es lo que interconecta todos los componentes de la máquina. Los buses son "recursos compartidos en el tiempo", es decir solo se puede estar transfiriendo información de una fuente a un destino a la vez. Otro ejemplo de "recursos compartidos en el tiempo" son las calles de una ciudad que interconectan a todas las casas y edificios entre si o las butacas de un salon de clase que son utilizadas durante un tiempo por un grupo de alumnos mientras que a la siguiente hora es utilizada por otro grupo diferente (figura 2)

Evolución de las computadoras.

Primera Generación de Computadoras

El Mark I fue desarrollado en 1944 por Aiken para facilitar los trabajos de cálculo. Fue un equipo que hacia los cálculos en forma mecánica, paralelamente había equipos de investigadores buscando diseñar equipos que hicieran los cálculos electrónicamente.
John V. Atansoff, un profesor de física graduado del estado de Iowa, inició la construcción de un computador electrónico pero a causa de la segunda guerra mundial no pudo terminarlo. Otro intento por construir un computador electrónico, no culminado a causa de la guerra, fue llevado a cabo por Atansoff quien inicialmente construyó un pequeño prototipo que fue de ayuda para iniciar la construcción del Atansoff-Berry Computer (ABC) que no fue terminado. Una característica destacable de este equipo era que utilizaba operaciones lógicas para realizar los cálculos, tenía capacidad para guardar datos como números binarios y usaba tarjetas perforadas como dispositivos de entrada/salida.

Durante la segunda Guerra mundial, los investigadores hicieron mayores avances para aligerar la carga de la realización de cálculos. Fue desarrollado el ENIAC (Electronic

Numerical Integrator and Calculator) que por sus dimensiones ocupaba un cuarto de treinta por cincuenta pies y pesaba 30 toneladas. Tenía 18000 tubos al vacío y podía realizar 5000 sumas por segundo. Trabajaba con tarjetas perforadas.

En 1947 fue construido el EDVAC (Electronic, Discrete Variable Automatic Computer) por Eckert and Mauchley que contaba con un programa almacenado electrónicamente. Este programa le permitía al computador alterar las operaciones dependiendo de los resultados obtenidos previamente.
En 1951, Eckert y Mauchley construyeron el UNIVAC para usarlo en la oficina de censos. Este equipo usaba cinta magnética para la entrada / salida de datos, fue el primero en ser fabricado para negocios y de él se comercializaron 46 unidades.
En 1953, IBM produjo su computador 701 y dos años más tarde el 752. Desde ese entonces la empresa ha seguido en desarrollo y expansión.

Segunda Generación de Computadoras

Esta generación está marcada por el invento del transistor en los Laboratorios Bell en 1947. En 1954 Texas Instrument lo mejoró utilizando silicio en su fabricación en lugar de germanio. Con el uso de los transistores se pudieron construir computadores más confiables y baratos. Como medio de almacenamiento de la información se utilizaron las cintas magnéticas. Al observar que los computadores no sólo servían para realizar cálculos se dividió en dos líneas la producción, unos para realizar cálculos y los otros para procesamiento de datos. Al final de esta genreración se empezó a trabajar en un equipo que realizara ambas labores al mismo tiempo.

Tercera Generación de computadoras

Esta generación está marcada por la creación de circuito integrado en 1958. Con este invento se ha extendido el uso de la computadoras en la actualidad. Al encontrar la forma de reducir el tamaño de los transistores para poner cientos de ellos en un pequeño chip de silicio los fabricantes de computadores pudieron construir equipos más pequeños. En 1964 IBM sacó la serie 360 que integraba las dos líneas de mercado y era compatible con cualquier otro de su familia. De aquí en adelante se buscó que los equipos pudieran suplir ambos requerimientos realizar cálculos y procesar información.
Al mismo tiempo se desarrollo el concepto de lenguaje de programación, inicialmente la programación era con tarjetas y cables. Pero al hacerse más complejos los computadores y el hacerlos funcionar, la comunicación entre el equipo y los usuarios se volvió más. En 1956 se desarrolló el FORTRAN (primer lenguaje de programación) y en 1959 el COBOL
Los lenguajes de programación le permitieron a los programadores escribir código con un nivel conceptual mayor, después un compilador traducía el código en lenguaje de máquina. Con los lenguajes se pudieron desarrollar los sistemas operativos.
Los primeros sistemas operativos eran monotarea, solo permitían desarrollar un proceso a la vez, y tenían muchos errores a causa de su complejidad. Fueron desarrollados sistemas

multitarea para permitir que las tareas fueran ejecutados continuamente y mientras alguna tarea esperaba una entrada pudieran ejecutarse otras.
En 1970, IBM puso una unidad de disquete en su computador 3740. Con el uso del disquete se incrementó la capacidad y velocidad de acceso a la información.

Cuarta generación de computadoras

Esta generación está marcada por la creación del microprocesador. Este unía todos los circuitos integrados que contenían a su vez transistores en un solo paquete. Los microprocesadores eran capaces de desarrollar todas las funciones de la unidad central de proceso.

El desarrollo del microprocesador permitió la creación de los computadores Personales (PC) que fue un concepto revolucionario y marcaría un cambio en la forma de trabajar e incluso de vivir para muchas personas.
El uso de los computadores se fue expandiendo inicialmente en el trabajo y luego en los hogares. Con desarrollo de programas orientados tanto a adultos como niños y la revolución en cuanto a desarrollo de hardware, el uso de los computadores se ha expandido y popularizado rápidamente.

Generación	**Año**	Características
1ª Generación	1947	Bulbos.
2ª Generación	1956	Transistores.
3ª Generación	1964	Circuitos integrados de lógica RTL y TTL, surgen los lenguajes FORTRAN y COBOL.
4ª Generación	1972	Circuitos integrados de mediana escala de integración (memorias). Aparece el concepto de buses y los primeros microprocesadores de propósito general.

Cronología de los Microprocesadores.

1642 .- Pascal inventa máquina de sumas y restas.
1671 .- Wilhem Gottfried Leibinitz inventa una máquina para hacer multiplicaciones y divisiones mediante sumas y restas repetidas.
1801 .- Jaquard diseña máquina tejedora a base de tarjetas perforadas.
1822 .- Charles Babage diseña una máquina analítica.
1940 .- Alan Touring inventa la máquina COLUSUS (basándose en tubos de vacío).
1946 .- Eckert y Mauchly construyen la ENAC.
1953 .- EDUAC, primera máquina totalmente programable.
1962 .- Aparecen los circuitos integrados en pequeña escala.
1965 .- PDP-8 Minicomputadora de 12 bits.
1971 .- Surgen las calculadoras de bolsillo, Microprocesador Intel 4004.

1972 .- Aparece el microprocesador de 8 bits 8008.
1973 .- Motorola desarrolla los microprocesadores 8080 y 6800.
1975 .- Z-80 (Zilog); 6501 y 6502 de Motorola.
1979 .- Surgen los primeros microcontroladores, incluyen memoria y puertos de entrada y salida en un solo dispositivo,.
1985 .- Memorias Semiconductoras de 1 Mega Byte.

Tendencias en la evolución de las computadoras.

La evolución de las computadoras nos permite observar cuales han sido las tendencias de mejora a través de las diferentes generaciones, y pueden resumirse en:

- Reducir consumo de energía.
- Reducir espacio.
- Integrar cada vez más funciones en un solo dispositivo.
- Lograr sistemas más versátiles
- Incrementar la velocidad de procesamiento.

ARQUITECTURAS BÁSICAS.

Existen una serie de arquitecturas, modelos o tendencias que han seguido los disenadores a lo largo del desarrollo de las computadoras y de las cuales algunas han prevalecido siendo significativas y determinantes sirviendo de guias basicas para aquellos que se proponen el diseno del hardware como del software de cotrol de los diferentes sistemas de computo, asi como para aquellos que disenan aplicaciones basados en los mismos. A continuacion se presentan algunas de las arquitecturas mas significativas que rigen el diseno de los diferentes sistemas computacionales.

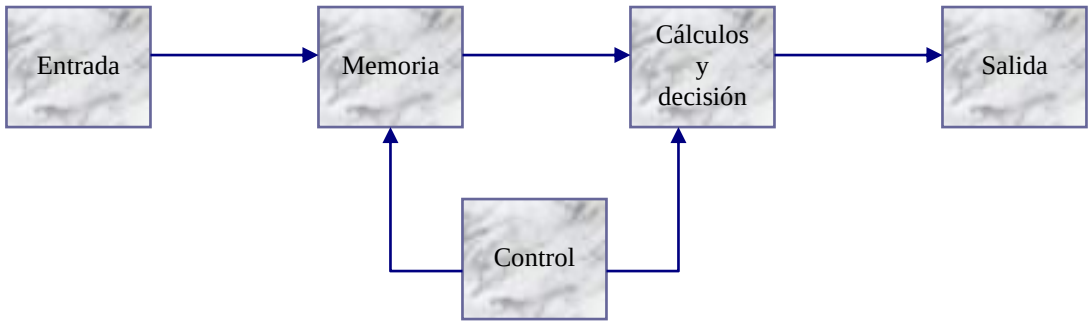


Diagrama a bloques de la máquina analítica de Babage.

Características:

- Totalmente mecánica.
- Entrada mediante tarjetas perforadas.
- La memoria recibe tanto la información a procesar como el programa.

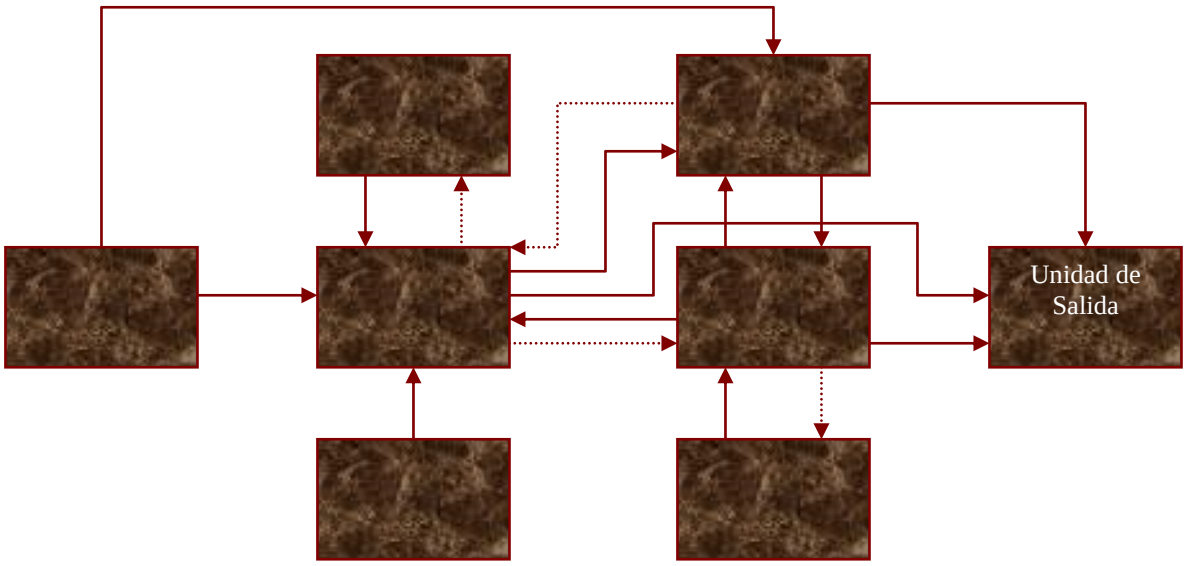


Diagrama a bloques de los primeros sistemas digitales
de proceso síncrono programable. MARK I y ENIAC.

Características:

- Máquina extremadamente programable.
- Además de los datos, también se tienen en memoria la secuencia de instrucciones que indican las micro operaciones que se realizarán sucesivamente.
- La Unidad de Control genera las señales adecuadas para que las unidades de Memoria y la Unidad Aritmético Lógica realicen las micro operaciones correspondientes.

La configuración inicial evolucionó almacenando en una misma memoria de acceso aleatorio datos e instrucciones del programa, con lo cual se reduce el número de terminales de la unidad de control. La Memoria de Datos (Activa) y la Memoria de Instrucciones (Pasiva) están físicamente separadas, pero se ligan a la Unidad de Control a través de las mismas conexiones.

Similarmente, la Unidad de Control, la Unidad Aritmético Lógica y los Registros de almacenamiento intermedio y temporal se agrupan en una sola Unidad Central de Procesamiento. Se llega entonces al esquema de **Von Newman**:

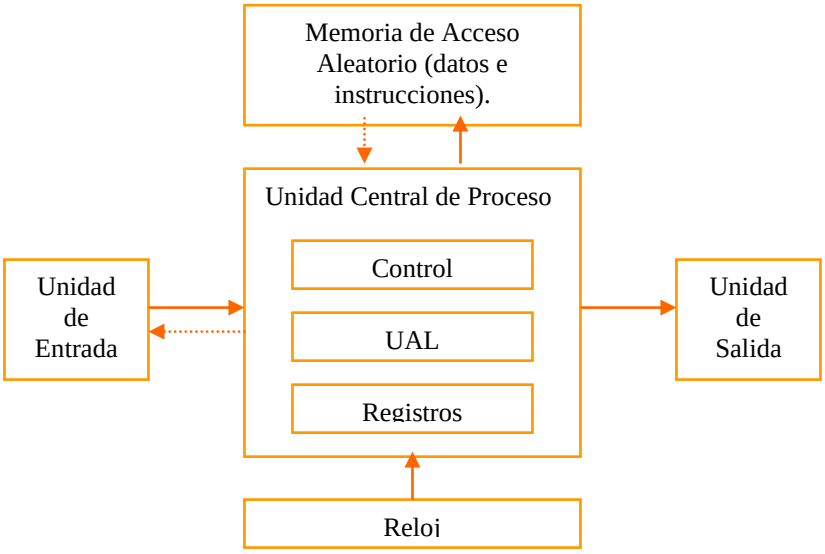
Arquitectura de Von Newmann:

Las características principales son: La unidad central de proceso integra en un solo dispositivo a la Unidad de Control, la Unidad Aritmético Lógica y los Registros de almacenamiento intermedio y temporal

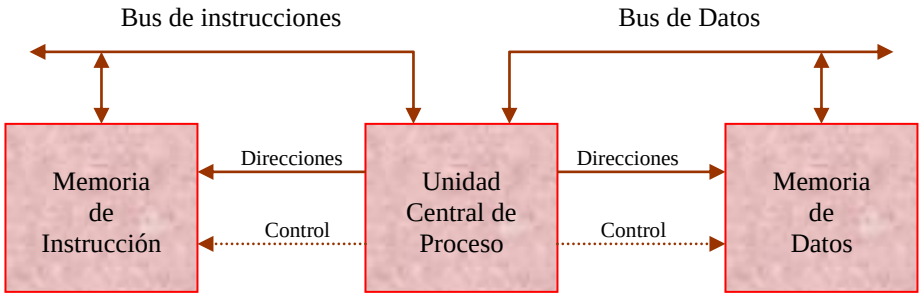
La memoria de programa y la memoria de datos están físicamente separadas pero eléctricamente unidas a través de un mismo bus de datos y bus de direcciones.

La Unidad de entrada puede ser un teclado, un mouse, tarjetas perforadas etc.

La Unidad de salida pueden ser displays, un monitor, una unidad de disco, etc.



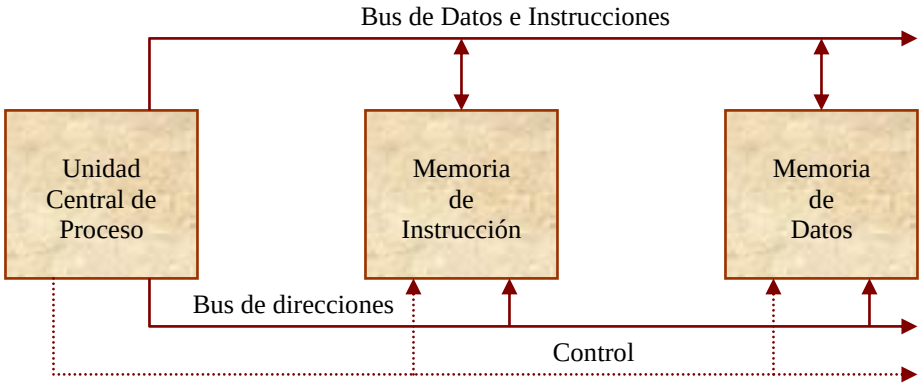
Máquina Harvard Clase I



Características:

- Módulos de memoria separados para Datos e Instrucciones.
- Buses separados e independientes para acceder cada memoria.
- Con ello se logra una velocidad de procesamiento mayor al acceder en un mismo ciclo a la instrucción y a los datos requeridos por la instrucción

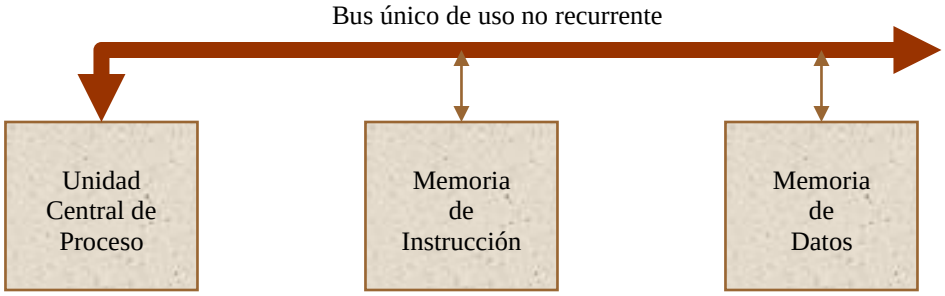
Máquina Harvard Clase II



Características:

- El Bus de infromación (datos e instrucciones) es común para accesar ámbos módulos de memoria.
- El Bus se comparte en el tiempo.

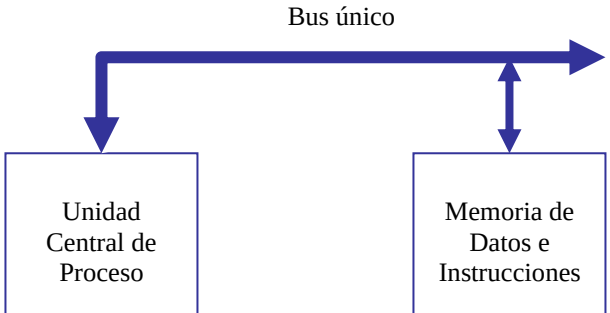
Máquina Harvard Clase III



Características:

- Bus único de datos, Instrucciones, Control y Direcciones.
- El Bus se comparte en el tiempo.

Máquina Princeton Estricta



Características:

- Datos e instrucciones comparten la misma Memoria.
- Emplea un solo Bus.
- Datos e instrucciones son formalmente idénticos, diferenciables únicamente por su unicación dentro de la memoria.

MICROPROCESADOR.

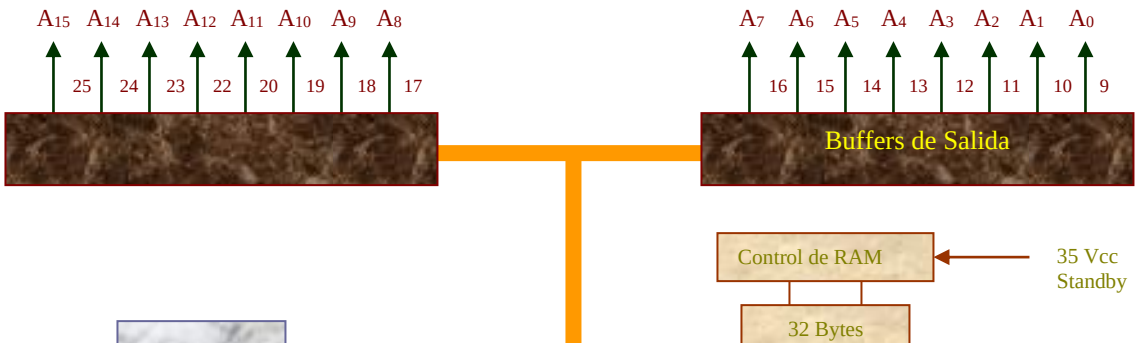
Dentro de un sistema de computo y en base a la evoluci[on de las computadoras, sabemos que la unidad central de procesamiento actualmente se conjunta en un circuito integrado llamado microprocesador, que es un dispositivo electronico de alta escala de integracion y a cuyo analisis y estudio se dedican los siguientes parrafos.

Definici[on de microprocesador.- Es un sistema que explora secuencialmente una información almacenada llamada programa, la cual interpreta y ejecuta. El Mp maneja los datos de acuerdo con su conjunto de instrucciones y coordina las operaciones de la computadora; es en realidad el organizador de las actividades. El Mp es el "cerebro" de la computadora. Su función es ejecutar los programas almacenados en la memoria central tomando sus instrucciones, examinándolas y luego ejecutándolas una tras otra. El Mp dirige el tránsito electrónico entre la memoria primaria y la unidad aritmético lógica y entre la unidad de control y las unidades de entrada y salida.

EL MICROPROCESADOR MC6802

En este texto se toma como base el estudio del microcontrolador MC6802. Se trata de un microprocesador de 8 bits, de motorola que opera a 8MHz y su funcionamiento se describira a lo largo del capitulo.

Diagrama a bloques del Microprocesador MC 6802



CICLO DE TRABAJO DEL MICROPROCESADOR

La función asignada a la Unidad Central de Procesamiento (CPU), es la de captar las instrucciones contenidas en la memoria y ejecutarlas a continuación, procediendo de esta manera instrucción por instrucción hasta completar el total del programa correspondiente. Este proceso lo efectúa de manera síncrona, o sea que se emplean pulsos procedentes de un reloj, que generalmente es un oscilador electrónico estabilizado a cristal, para dar la

cadencia de funcionamiento de forma tal que los eventos que deben tener lugar conforme a una cierta secuencia, ocurran de manera ordenada y sin ambigüedades.

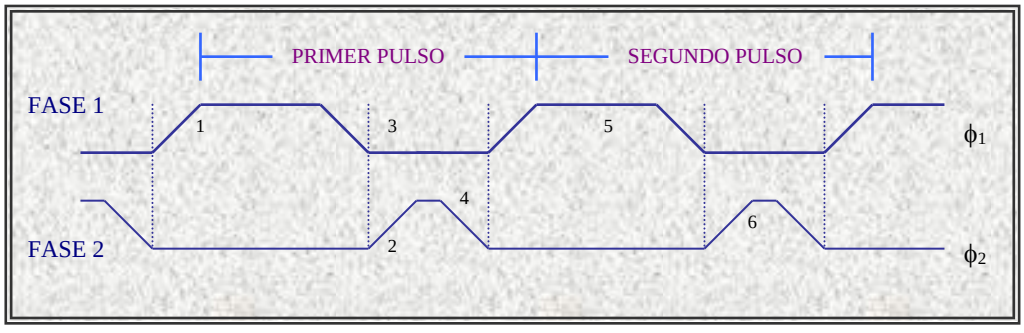
A cada pulso de reloj se le denomina “microciclo”. Para efectuar una operación usualmente se requieren varios pulsos de reloj. Así por ejemplo, si el microciclo es de 1 microsegundo y la operación se efectúa en dos microciclos, esta tardará 2 microsegundos. Una posible operación con esta característica es “complementar el contenido del acumulador B”, cuyo mnemónico es COM B y cuya representación hexadecimal en códico de máquina del microprocesador P6800 es 53, requiere dos pulsos de reloj para efectuarse y es una instrucción de un solo byte de longitud. Supongamos que dicha instrucción se encuentra en la dirección de memoria 01FF y que es la siguiente a efectuar por el microprocesador, los eventos que ocurren son los siguientes:

- En el primer pulso se envía al bus de direcciones el contenido del contador de programa, para seleccionar la celda de memoria que guarda la siguiente instrucción a realizar. El contenido de dicha celda se envía por el bus de datos y se guarda en el registro de instrucciones. Se incrementa el contador de programa.

- En el segundo pulso el CPU decodifica y ejecuta la instrucción.

Secuencia de lectura, decodificación y ejecución de una instrucción.

Para poder efectuar varias operaciones en secuencia en el transcurso de un mismo ciclo de reloj, se utilizan ondas desfasadas de igual frecuencia que el reloj. Así, el microprocesador P6802 requiere de dos fases de reloj para poder funcionar correctamente. En el ejemplo planteado se tiene lo siguiente:



Primer Pulso

1. El μP coloca el contenido del contador de programa (o sea una dirección) en el bus de direcciones.
2. El contenido de la memoria correspondiente a esa dirección queda disponible en el bus de datos.
3. Se ordena incrementar el contador de programa.

Segundo Pulso

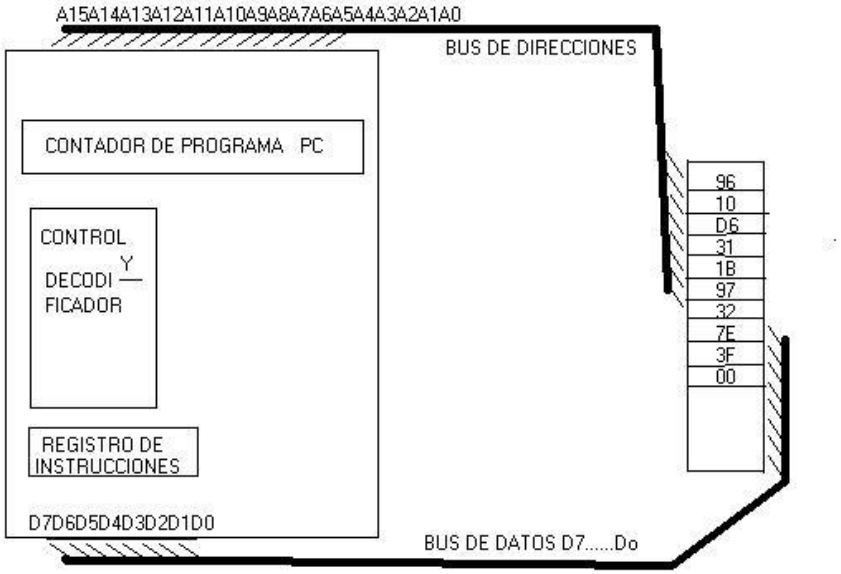
4. La información presente en el bus de datos queda guardada en el registro de instrucciones.
5. La CPU decodifica la instrucción.
6. La CPU ejecuta la instrucción.

El microprocesador ejecuta una instrucción completa en cada ciclo de trabajo, mismo que involucra dos estados:

a) Un estado de búsqueda en el cual genera los impulsos adecuados para leer la instrucción gurdada en memoria.
b) Un estado de ejecución, en el cual genera los impulsos necesarios para efectuar la instrucción.

Algunas instrucciones tienen un byte de longitud, otros dos bytes y varios más tres bytes, razón por la cual después de leer el primer byte de una instrucción, la CPU debe decodificarla y decidir si se trata de una instrucción de un byte, de dos o de tres bytes.

Si es una instrucción de un solo byte procederá a ejecutarla, si es de dos bytes incrementará automáticamente el contador de programa y procederá a captar el segundo byte a fin de tener la información completa de la instrucción y proceder.



ANALOGIA DEL MICROPROCESADOR

Es posible establecer una analogia entre un microprocesador y una orquesta, podemos decir que el director es la UCP que envia las señales de control al resto de los subsistemas (cuerdas, metales , percuciones, etc.) , el programa almacenado en la memoria de programa, es semejante a la partitura (almacenada en papel) que se ba ejecutando de manera secuencial, pero puede tener partes que se repitan, o brincos a otras que se omitan según sea el caso, asi mismo el reloj es el metronomo que establece el ritmo de ejecucion o cadencia de la pieza. El ejecutante incluye su propio decodificador de instrucción osea el ejecutante traduce cada nota en un movimiento o toque que provoca una salida parcial, (sonido) entonces el musico al igual que el microprocesador, lee en forma secuencial cada nota la cual interpreta y "ejecuta". Cada nota que lee la almacena en un registro temporal y eventualmente hay una nota o grupo de notas que ya ha leido anticipadamente mientras se encuentra ejecutando una nota en particular, en este caso estariamos hablando de una memoria interna que ba almacenando las instrucciones recien leidas y que estan en proceso de decodificacion mientras se esta ejecutando la anterior, en algunos casos podemos hablar incluso de una "memoria oculta" que almacena un grupo completo de notas para no interrumpir la ejecucion de la pieza mientras se da vuelta a la hoja de la partitura. Asi mismo incluye necesariamente un registro interno que indique o lleve el control de cual sera la siguiente nota a ejecutarse. Una unidad aritmetica sera necesaria para dividir en cada compaz el tiempo que debera darsele a cada nota según sea su valor es decir calcular la duracion de cada nota dependiendo si es corchea o semicorchea etc. No obstante tal cantidad de similitudes, una diferencia importante sera que el microcontinúa buscando instrucciones que leer mientras este energizado, de ahí la importancia de ciclar adecuadamente los programas en los microprocesadores.

FORMATOS DE INSTRUCCIONES

Cada μP en particular tiene un determinado conjunto de instrucciones que puede ejecutar. Para ello cada instrucción debe contener la siguiente información:

- La operación a realizar.
- La ubicación del operando o los operandos (Datos).
- La ubicación del resultado.
- La ubicación de la siguiente instrucción.

Previamente se mencionó la conveniencia de reducir al mínimo de longitud de las instrucciones usando para ello:

a) Un contador de programa,
b) haciendo que la dirección del resultado fuese la misma que la de uno de los operandos,
c) usar sus direcciones cortas de registros en vez de direcciones largas de memoria y

d) haciendo que las direcciones de origen de datos y de destino de resultados sean implícitas en vez de explícitas.

Todo ello se logra a cambio de perder flexibilidad, aumentar el número de instrucciones y alargar la longitud de los programas.

Usualmente las operaciones que realiza la Unidad Aritmético Lógica son:

- SUMAR
- SUMAR CON ACARREO
- RESTAR
- RESTAR CONSIDERANDO ARRASTRE
- COMPLEMENTAR A DOS
- INCREMENTAR
- DECREMENTAR
- AND LOGICA
- OR LOGICA
- OR EXCLUSIVA
- NEGACION
- CORRIMIENTO A LA IZQUIERDA
- CORRIMIENTO A LA DERECHA
- ROTACION

A las que se suman algunas otras instrucciones propias del μP. Existen en el mercado microprocesadores con menos de 50 instrucciones y otros tienen más de 1464 instrucciones en total. Si bien, muchas de estas instrucciones pueden considerarse repetidas, ya que puede tratarse de una misma operación referida a diferentes registros. Un microprocesador con un pequeño juego de instrucciones puede hacer lo mismo que otro con un repertorio mayor de instrucciones, pero generalmente requiere más pasos de programación que se traducen en mayor requerimiento de memoria y más tiempo para ejecutar un programa equivalente.

Ejemplo: Supongamos que se tiene un sistema con 32 instrucciones diferentes, 8 diferentes registros y 65,536 (2^{16} = 64 K bits) direcciones de memoria. Para citar una de las 32 instrucciones se requieren solo cinco bits; para diferenciar uno en particualr de los ocho diferentes registros se requieren3 bits; para ubicar una de las 65,536 distintas memorias se requieren 16 bits. En el formato de una dirección podríamos tener casos como los siguientes:

Operación a realizar	Dirección del primer operando y del resultado	Parte alta de la dirección segundo operando	Parte baja de la dirección del segundo operando.
SUMAR	ACUMULADOR A	E0	50
(5 bits)	(3 bits)	(8 bits)	(8 bits)

Operación $A + M_{E050} \rightarrow A$. La instrucción ocupa 3 bytes.

Operación a realizar	Dirección del primer operando y del resultado
NEGAR	ACUMULADOR B
(5 bits)	(3 bits)

Operación $00 - B \rightarrow B$.
La instrucción ocupa un byte.

Dado que la memoria está organizada en bytes y en ella debemos depositar el programa (listado de instrucciones) y los datos, conviene que cada instrucción ocupe el menor número posible de bytes. Conforme a ello, en el ejemplo de la instrucción "Complementar el acumulador A ($\bar{A} \rightarrow A$)" ocupa una posición de memoria y la instrucción " Sumar al acumulador el contenido de la memoria M_{4F27} ($A + M_{4F27} \rightarrow A$)" ocupa tres posiciones de memoria. En los dos casos las instrucciones están completamente especificadas sólo que la primera requiere 1 byte y la segunda requiere 3 bytes, ambas tienen en común que los primeros ocho bits indican la operación a realizar y los registros que intervienen.

Cuando hablamos del Código de Instrucciones de un microprocesador, diciendo por ejemplo que para el 6802 es de 197 diferentes instrucciones, para el 8080 es de 244 y para el Z80 de 256 distintas instrucciones, nos referimos a que en los ocho bits de cada instrucción queda indicada que la operación a efectuar y el registro utilizado. Cada instrucción se citará mediante un "MNEMÓNICO" conveniente para el programador y posterirormente se convertirá a "Código Máquina" leyéndolo de la tarjeta de programación. En dicha tarjeta, además del código correspondiente a la instrucción, encontramos el número de bytes que la instrucción requiere.

Volviendo al ejemplo previo, podemos diferenciar entre la primera instrucción de 1 byte de longitud y la segunda de 3 bytes diciendo que en el primer caso se trata de DIRECCIONAMIENTO INHERENTE, ya que en los ocho bits del código de la instrucción está contenida toda la información necesaria y en el segundo caso hablar de DIRECCIONAMIENTO EXTENDIDO, ya que además de los ocho bits del código de la instrucción se requieren de otros 16 bits correspondientes a la dirección real de la memoria a que hace referencia la instrucción. Esta froma de direccionamiento no requiere ningún cálculo y permite accesar cualquier posición de memoria de manera directa ya que la dirección es una parte de la instrucción.

Dado que muchos programas son relativamente cortos, menores de 256 bytes de memoria, al diseñar el 6800 se ideó una forma especial de direccionamiento denominado DIRECCIONAMIENTO DIRECTO o DIRECCIONAMIENTO EN PAGINA CERO, que puede accesar memorias cuyo rango esté entre 0000 a 00FF únicamente y en el cual, después del código de instrucción, se señalan los últimos dos dígitos de la dirección, formando una instrucción de solo 2 bytes de longitud. El μP al leer el código de instrucción correspondiente a una instrucción de este tipo, sabe que debe poner en ceros los primeros ocho bits de la dirección (A_8 a A_{15}) e interpretar el segundo byte de la instrucción como los restantes ocho bits de la dirección (A_0 a A_7).

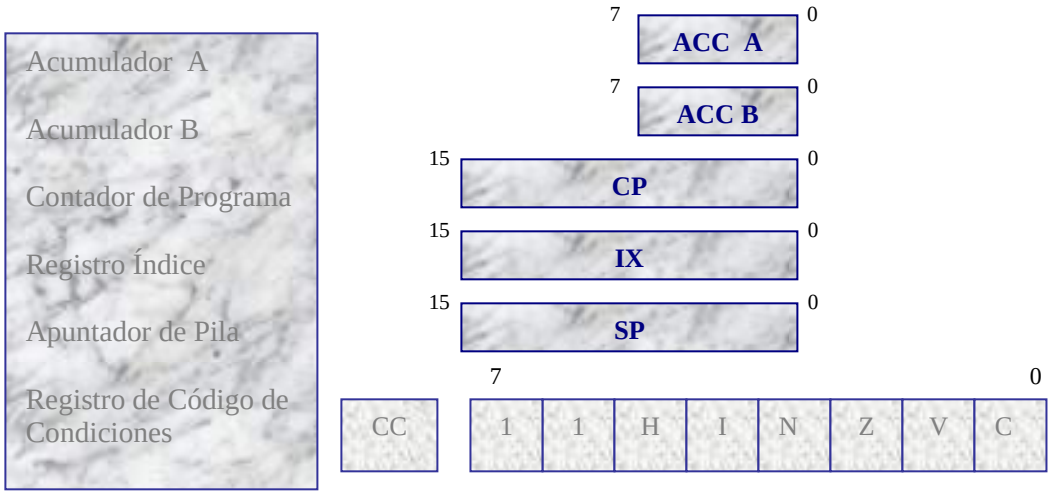
Operación a realizar	Dirección del primer operando y del resultado	Parte baja de la dirección del segundo operando
SUMAR EN PAG. CERO	Acumulador A	30
(5 bits)	(3 bits)	(8 bits)

Operación A + $M_{30} \rightarrow$ A

La instrucción ocupa 2 bytes

Nótese que el direccionamiento extendido y por ende el direccionamiento en página cero que es un subconjunto de éste, sólo puede accesar una sola dirección fija de memoria, careciendo de flexibilidad para procesar arreglos de datos.

"MODELO DE PROGRAMACIÓN" *REGISTROS INVOLUCRADOS EN LA PROGRAMACIÓN DEL MICROPROCESADOR MC6802*



El modelo de programacion se refiere a cuales son los registros internos disponibles para el programador, es decir, son los recursos internos del microcontrolador y es necesario conocerlos para desarrollar los programas de forma adecuada. En la figura anterior son mostrados los registros internos del microcontrolador 6802 y su función se describe a continuación.

ACUMULADOR A (acumulador ***B***).- Es un registro de 8 bits de proposito general, y para realizar operaciones aritmeticas o logicas el operando o uno de los operandos debe estar siempre en un acumulador ya que no se permiten las operaciones entre localidades de memoria. Es tambien en los acumuladores donde permanece el

resultado de una operación aritmetica ó logica, sustituyendo al del operando que se encontraba antes de ejecutar la operación. El proposito, tamaño y operación del acumulador B es igual.

CONTADOR DE PROGRAMA.- 

En este caso es un registro de 16 bits para poder direccionar a cualquier parte de la memoria que es posible direccionar con este microcontrolador (64kbytes). Su función es llevar la secuencia y orden de ejecucion de programa, su valor inicial debe indicar hacia la direccion donde se encuentre o inicie el codigo ejecutar. Su valor se modifica (incrementa) durante la decodificacion de la instrucción recien leída. La secuencia de ejecucion puede ser modificada como producto de un salto o una bifurcacion programada dentro del codigo por el programador.

REGISTRO INDICE IX.- 

Es un registro de 16 bits y se utiliza principalmente en la lectura de tablas de datos a fin de convertir instrucciones de 3 bytes en instrucciones de 2 bytes, tambien puede ser utilizado como un registro de almacenamiento temporal y algunas operaciones aritmeticas directas sobre este registro (como incremento o decremento) estan permitidas,

APUNTADOR DE PILA.- 

Cuando el microcontrolador tiene que brincar a otra parte de la memoria a ejecutar un subconjunto de instrucciones para regresar donde se quedo antes de desviarse, es necesario almacenar la localidad a la cual deba regresar. Tal es la función del apuntador de pila que se encarga de guardar la direccion de donde se desvio el microcontrolador, para regresarle el valor al apuntador de programa cuando deba continar con la secuencia principal. En caso de existir mas subrutinas anidadas, el apuntador de pila seguira almacenando la nueva direccion de la cual se derivo nuevamente.

REGISTRO DE CODIGO DE CONDICIONES.- Este registrode 8 bits tambien llamado de “banderas” tiene como finalidad indicar el estado que guarda la ALU despues de la operación (aritmetica o logica) realizada, y sirve de referencia sobre todo para las instrucciones de decisión, donde se pregunta por el estado de uno o varios bits en particular. También se almacena el bit de acarreo de una suma o la lleva de una resta. A continuación se muestra la ubicación y significado de las banderas presentes en el microprocesador de MC6802

Donde:

C = **Bandera de acarreo** .- Se póne a uno cuando el resultado de una adición supera el valor FF Hex. O cuando el sustraendo es mayor que el minuendo.

V = **Bandera de Sobreflujo**.- Indica que una suma o resta de numeros con signo ha sobrepasado la capacidad del procesador por ejemplo si sumamos +127 (7FH) +1(01H) = 128 (80H) pero 80H es en la notacion con signo el numero –128, es decir se ha excedido la capacidad de la ALU del microprocesador.

Z = **Bandera de cero**.- Cuando despues de cualquier operación (aritmetica ó logica) el resultado es cero

N = **Negativo, bandera de signo.-** Se pone en uno cuando el resultado de una operación con signo resultó negativo

I = **Interrupción.-** Poner este bit en uno significa que las interrupciones estan permitidas

H = **Medio acarreo**.- Se pone en 1 cuando existe un acarreo del nibble bajo, al nibble alto. Es util para las instrucciones de ajuste a decimal.

Los bits 7º y 8º No tienen una función específica y siempre estan en "1" logico.

CONCEPTO GENERAL DE LAS BANDERAS.- Al igual que en el automovilismo o en la marina, las banderas nos indican de manera Visual, una condición especifica, por ejemplo la bandera a cuadros indica al ganador de una carrera, la bandera amarilla indica que no es conveniente nadar en una playa, etc. En los microprocesadores y micricontroladores puede haber uno o mas registros de banderas que nos indican diferentes condiciones de un sistema o subsitema en particular. En este caso el registro de estado marca concretamente el estado de la ALU después de realizada una operación aritmetica o lógica (por ejemplo una suma); Si el resultado ha generado un acarreo, entonces esta es la bandera que se modificara inmediatamente. Cabe agregar que cada instrucción puede modificar una o varias banderas a la vez, e incluso algunas simplemente no las modifican. Asi mismo es importante considerar los valores de las banderas producto de una operación en particular antes de que sean modificados por la siguiente instrucción. También existen instrucciones para "inicializar" alguna bandera en particular, es decir ponerla a uno o a cero.

CONJUNTO DE INSTRUCCIONES.-

Todo microprocesador tiene su propio conjunto de instrucciones que forman el lenguaje mediante el cual, el diseñador le "indica" o programa la serie de operaciones o tareas que debe realizar. Es el repertorio de "palabras" que el microcontrolador "entiende" y mediante el cual podemos comunicarle la serie de operaciones a realizar. El conjunto de instrucciones del microcontrolador 6802 consiste en:

- Conjunto de 72 operaciones diferentes.
- Un total de 196 códigos máqina distintos.
- Operaciones de un solo byte (operando implícito).
- Operaciones de 2 a 3 bytes (operando explícito).

En realidad el microcontrolador solamente puede “interpretar” señales ó códigos binarios y es ello lo que se almacena o “puede” almacenarse en una memoria, dichos códigos binarios son representados también en su equivalente Hexadecimal por comodidad, sin embargo aún así resulta complicado memorizar la equivalencia de cada código hexadecimal asociada a cada instrucción, por lo que se ha desarrollado lo que se conoce como “lenguaje ensamblador” el cual consiste en una serie de abreviaturas también llamadas **MNEMONICOS** que son mas manejables y permiten recordar la instrucción a la que hacen alusion asi por ejemplo tenemos el mnemónico ADD que se deriva de la palabra “suma” en inglés (addition).
De esta forma tenemos que el conjunto de instrucciones de un microprocesador se constituye de un conjunto de mnemónicos que hacen referencia a la operación que representan; y es en este lenguaje que se desarrollan los programas que posteriormente son codificados a “lenguaje máquina”; es decir a los códigos binarios equivalentes que el micro puede interpretar y ejecutar. Mas adelante se muestra el conjunto completo de instrucciones (mnemónicos) para el 6802 (que es compatible con otros micros de motorola), asi mismo sus respectivos codigos maquina, las banderas de estado que modifican, el número de ciclos máquina que consumen y el numero de bytes que los componen.

NOTACION DE MICRO-OPERACIÓN

La notación de micro-operación es una forma de mostrar o representar las operaciones que un microprocesador realiza indicando los operandos ó registros involucrados, la operación a realizar y la ubicación del resultado. En el ejemplo se muestra la notación de micro-operación, el nemónicoy la descripción de la instrucción correspondiente

A ⟵ A + B **ABA** Suma el contenido del registro A con el contenido del Registro B y el resultado se almacena en el registro A

IX ⟵ IX + 1 **INX** Incrementa el contenido del registro Indice IX en una unidad.

A ⟵ (M10) LDAA 10 Carga el acumulador A con el contenido de la localidad 10

Con esta notación es posible identificar la operación a realizar por el porcesador en caso de que el mnemónico no sea muy específico.

MODOS DE DIRECCIONAMIENTO

Los modos de direccionamiento hacen referencia al lugar donde se encuentran los operandos que son requeridos por una instrucción en particular y de ellos depende el código máquina respectivo, de esta forma tenemos que una misma instrucción ó mnemónico puede

tener varios códigos máquina, dependiendo del modo de direccionamiento que se trate, por ello es importante conocerlos y diferenciarlos claramente para elegir el código máquina correcto.

Existen 7 modos de direccionamiento, los cuales están en relación directa con el tipo de instrucciones que realiza el microprocesador. Dichos modos son los siguientes:

a) Con operando implícito.- los operandos se encuentran contenidos "dentro" de alguno o algunos de los registros internos del microprocesador a su véz se clasifican en:

- De acumulador.
- Inherente o implícito.

b) Con operando explícito.- Al menos uno de los operandos esta "afuera" del microprocesador y son:

- Inmediato.
- Directo.
- Extendido.
- Indexado
- Relativo.

CON OPERANDO IMPLÍCITO

Direccionamiento de Acumulador.

Las instrucciones en las que sólo se utilizan uno de los dos acumuladores tiene este tipo de direccionamiento, por ejemplo:

"Decrementar Acumulador A"

Mnemónico	Código	Operación
DEC A	4C	ACC A ← ACC A - 1

En este caso, el operando se encuentra ya en un registro interno y concretamente se trata de uno de los acumuladores (acumulador A).

Modo de Direccionamiento Inherente.

Aquellas instrucciones que únicamente emplean los registros internos de la unidad de microprocesador (excepto acumuladores) tienen este modo de direccionamiento, tales como el registro de código de condiciones, apuntador de pila, registro índice etc.

"Incrementa el contenido del registro Indice X en una unidad"

Mnemónico	Código	Operación
INX	08	IX ← IX + 1

CON OPERANDO EXPLÍCITO

Modo de Direccionamiento Inmediato.

En el modo de direccionamiento inmediato, la instrucción está formada por 2 bytes, el primero es el código de operación y el segundo es el operando. Ejemplo:

" Sumar 4A al contenido del acumulador A"

Mnemónico	Código	Operación
ADD A #$ 4A	8B4A	ACC A ← ACC A + 4A

- El símbolo # indica el modo de direccionamiento inmediato.
- El símbolo $ indica el código hexadecimal.

En este caso el acumulador se carga "inmediatamente" con el numero 4AH No con el contenido de la localidad 4AH

Modo de direccionamiento directo.

En este modo de direccionamiento la instrucción está formada por 2 bytes, el primero contiene el código de operación y el segundo la dirección del operando. Esto permite accesar los primeros 256 bytes de memoria. Es decir de la localida 00 a la FF (Hex) como máximo (dirección de 1 Byte) Ejemplo:

"Cargar acumulador A con el contenido de la localidad 20"

Mnemónico	Código	Operación
LDA A $20	9620	ACC A ← (20)

En este caso, el acumulador A se carga con el Contenido de la Localidad de memoria numero 20H no con el valor de 20

Modo de direccionamiento extendido.

Una instrucción bajo este modo de direccionamiento está constituido por 3 bytes de los cuales el primero contiene el código de operación y los dos siguientes la dirección del operando. Los 8 bits más significativos de esta dirección (parte alta) están contenidos en el segundo byte. Esto permite acceder a cualquier localidad de memoria, desde la 0000H a la FFFFH Ejemplo:

"Cargar la dirección 201F con el contenido del acumulador A"

Mnemónico	Código	Operación
STA A $201F	97201F	(97201F) ← ACC

Modo de direccionamiento indexado.

En este modo de direccionamiento, la instrucción está formada por dos bytes, el primero contiene el código de operación y el segundo un número en complemento a 2 que

se suma al contenido del registro índice para formar la dirección del operando. Para ello es necesario cargar primeramente el registro Índice IX con la dirección inicial a partir de la cual se hará el desplazamiento indicado por el segundo byte de la instrucción:. Esto permite además el convertir una instrucción de 3 bytes en una de 2 bytes. Esta dirección se forma en un registro temporal y el registro índice no se modifica. Ejemplo:

" Suma el acumulador A más el siguiente operando el cual es el contenido de IX+10 "

Mnemónico	Código	Operación
ADD A $10,X	AB10	ACC A + (IX + 10_H)

Modo deDireccionamiento relativo.

Este modo de direccionamiento está asociado a las instrucciones de ruptura de secuencia condicionales o no (a excepción del JMP). El siguiente byte del código de operación contiene un número expresado en complemento a 2, el cual se suma con el contador de programa después de que este se ha incrementado para direccionar la siguiente instrucción. Esto permite efectuar saltos a los 126 bytes anteriores a la instrucción de ramificación, así como a los 129 posteriores.

La dirección "D" alcanzada a partir de la instrucción de salto localizada en "D1" con un desplazamiento llamado DESP, lo podemos expresar como: D = (D1 + 2) + DESP con desplazamiento $-128 \leq DESP \leq 127$

Ejemplo:

BNE ALFA 2607 Si Z ≠ 0 entonces PC ← PC + 02 + 07

Lenguaje ensamblador

Introducción

Los ensambladores son programas que procesan los enunciados del programa origen en lenguaje ensamblador y los traducen en archivos en lenguaje máquina que son ejecutados por un microprocesador o un microcontrolador.
Los ensambladores permiten que los programas origen se escriban y se editen en una computadora para generar un código ejecutable en otra computadora. El archivo en lenguaje objeto ejecutable resultante se carga y se ejecuta en el sistema destino.

Ventajas del Ensamblador

La primera razón para trabajar con ensamblador es que proporciona la oportunidad de conocer más a fondo la operación de su PC, lo que permite el desarrollo de software de una manera más consistente.

La segunda razón es el control total de la PC que se tiene con el uso del mismo.

Otra razón es que los programas de ensamblador son más rápidos, más compactos y tienen mayor capacidad que los creados en otros lenguajes.

Por último el ensamblador permite una optimización ideal en los programas tanto en su tamaño como en su ejecución.

ASMHC11

Es el ensamblador estándar 68hc11, desarrollado hace años por Motorola.
Asmhc11 toma el programa del lenguaje ensamblador (* asc) como entrada y genera diversos archivos incluyendo:
* s19 - la forma registrada de Motorola S del programa ejecutable
* lst - un listado de qué ocurrió durante el proceso de ensamble - (es decir el código automático generado para cada instrucción de ensamble).
Si tiene errores de sintaxis durante el proceso de ensamble, revise este archivo para descubrir qué errores ocurrieron. Una vez que localice los errores, vaya de nuevo a su archivo *asc a hacer las correcciones.

Programación en Lenguaje Ensamblador

Los programas escritos en lenguaje ensamblador consisten de una secuencia de enunciados origen.

Formato de los enunciados origen

Cada enunciado origen puede contener hasta cuatro campos: una etiqueta o " * " para una línea de comentario, una operación (ya sea el mnemónico de una instrucción o una directriz para el ensamblador), un operando y un comentario.

Campo de Etiquetas

El campo de etiquetas aparece como el primer campo dentro de un enunciado origen. El campo de etiquetas puede adoptar cualquiera de las siguientes formas:

1. Un asterisco (*) como el primer carácter en el campo de etiquetas indica que el resto del enunciado origen es un comentario. Los comentarios son ignorados por el ensamblador e impresos en el listado origen solamente como información de programación.
2. Un espacio de carácter en blanco (TAB o espacio) como primer carácter indica que el campo de etiquetas se encuentra vacío. La línea no tiene una etiqueta y no es un comentario.
3. Un símbolo como primer carácter indica que la línea tiene etiqueta. Estos símbolos son las letras mayúsculas y minúsculas (a - z), los dígitos (0 - 9) y caracteres especiales como punto (.), signo de pesos ($) y subrayado (_). Estos símbolos consisten de uno a quince caracteres, el primero de los cuales debe ser alfabético o un carácter especial punto o subrayado. Todos los caracteres son significantes y las mayúsculas y minúsculas son distintas.

Un símbolo puede aparecer solamente una vez en el campo de etiquetas.

Si un símbolo aparece más de una vez en el campo de etiquetas, toda referencia a dicho símbolo será marcada como error.

Con la excepción de algunas directrices a la etiqueta se le asigna el valor del contador de programa (PC) del primer byte de la instrucción o dato que se esté ensamblando.

El valor asignado a la etiqueta es absoluto.

De manera opcional, las etiquetas pueden ser terminadas con el símbolo de dos puntos (:).

Si dicho símbolo es utilizado, no formará parte de la etiqueta, simplemente servirá como separación entre la etiqueta y el resto del enunciado origen.

De esta forma, los siguientes fragmentos son equivalentes:

salta: deca salta deca bne salta bne salta

Una etiqueta puede aparecer por si sola en una línea. El ensamblador interpreta esto como "establece el valor de la etiqueta igual al valor actual del contador de programa (PC)".

Campo de Operaciones

El campo de operaciones aparece después del campo de etiquetas y debe de estar precedido por al menos un espacio en blanco.

El campo de operaciones debe de contener el mnemónico de un código de operación legal o una directriz del ensamblador.

En este campo, los caracteres en mayúsculas son convertidos en minúsculas antes de ser revisados como un mnemónico legal.

Debido a esto " nop ", " NOP " y " NoP " son reconocidos como el mismo mnemónico.

Los símbolos que aparecen en este campo pueden ser de uno de dos tipos.

Código de Operación.

Estos símbolos corresponden directamente a instrucciones de máquina.

El código de operación incluye a cualquier nombre e registro asociado con la instrucción.

Estos nombres de registros no deben de estar separados del código de operación por ningún espacio en blanco.

De esta forma, " clra " significa " limpia (poner en ceros) el acumulador (A) ", pero " clr a " significa " limpia la localidad de memoria identificada por la etiqueta a ".

Directriz.

Estos son códigos de operación especiales conocidos por el ensamblador, los cuales más bien controlan el proceso de ensamblado en vez de ser traducidos a instrucciones máquina.

Campo de Operandos

La interpretación del campo de operandos depende del contenido del campo de operaciones.

El campo de operandos, si se requiere, debe de seguir al campo de operaciones y debe de estar precedido por al menos un espacio en blanco.

El campo de operandos puede contener un símbolo, una expresión o una combinación de símbolos y expresiones separados por comas.

El campo de operandos de una instrucción máquina es utilizada para especificar el modo de direccionamiento de la instrucción, así como el operando de la instrucción.

Ejemplo de Programa en Lenguaje Ensamblador

El procedimientos es el siguiente

1.- Planteamiento del problema

2.- Diagrama de flujo utilizando notación de microoperación

3.- Transformar a mnemónicos según el microprocesador utilizado, tecleado en un editor de textos básico como el EDIT del MS.DOS o el NOTEPAD de Windows, respetando los espacios ó “campos” correspondientes

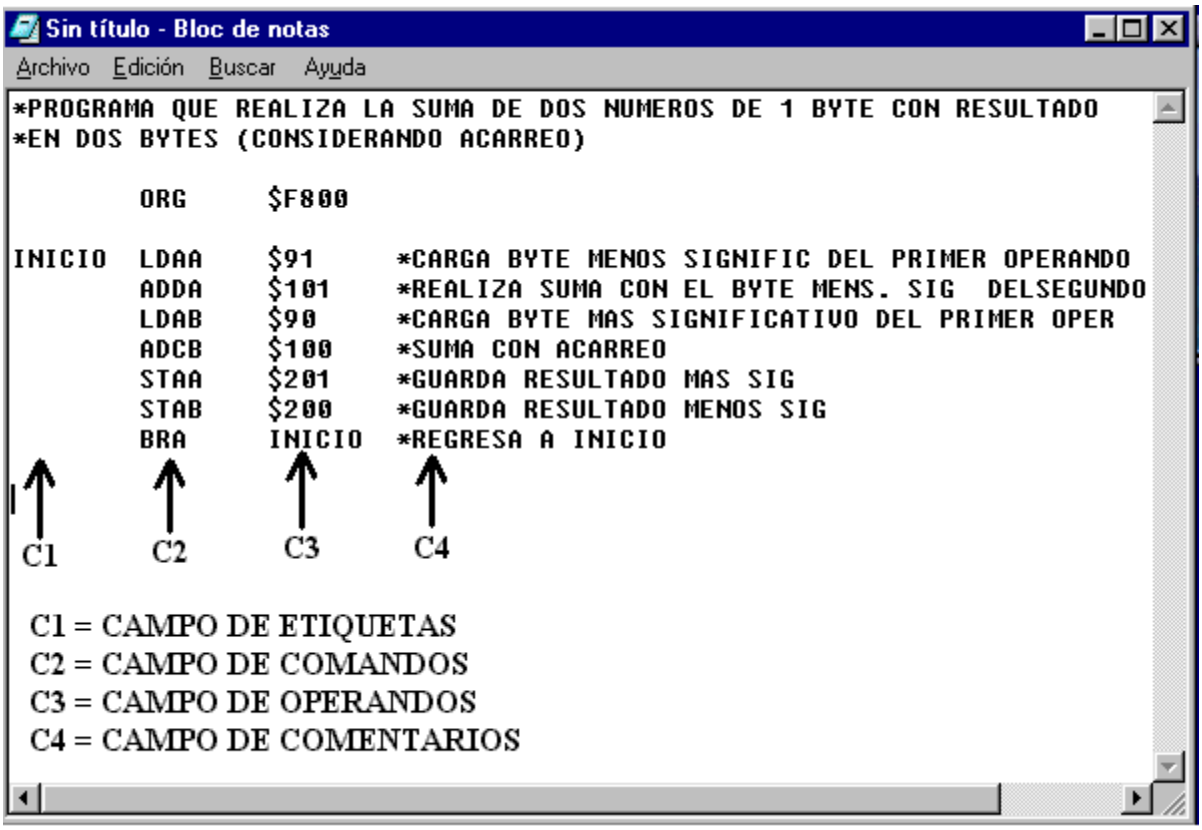
4.- Almacenar el archivo con extensión .ASM

5.- Ejecutar el programa ensamblador utilizado p. e. el ASMHC11

6.- En caso de existir errores, consultar el archivo .LST y corregirlos en el .ASM

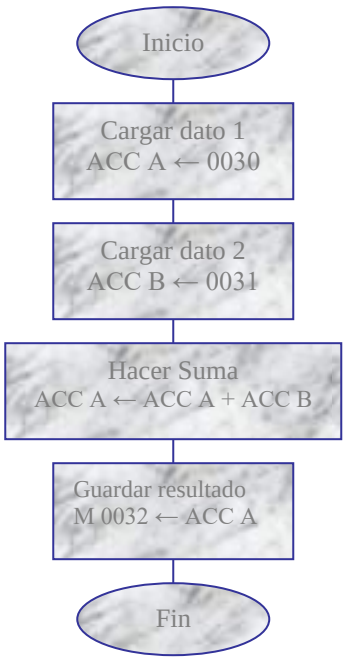
7.- Guardar el archivo objeto en la memoria de programa para su ejecución

A continuación se muestra un ejemplo con los campos correspondientes



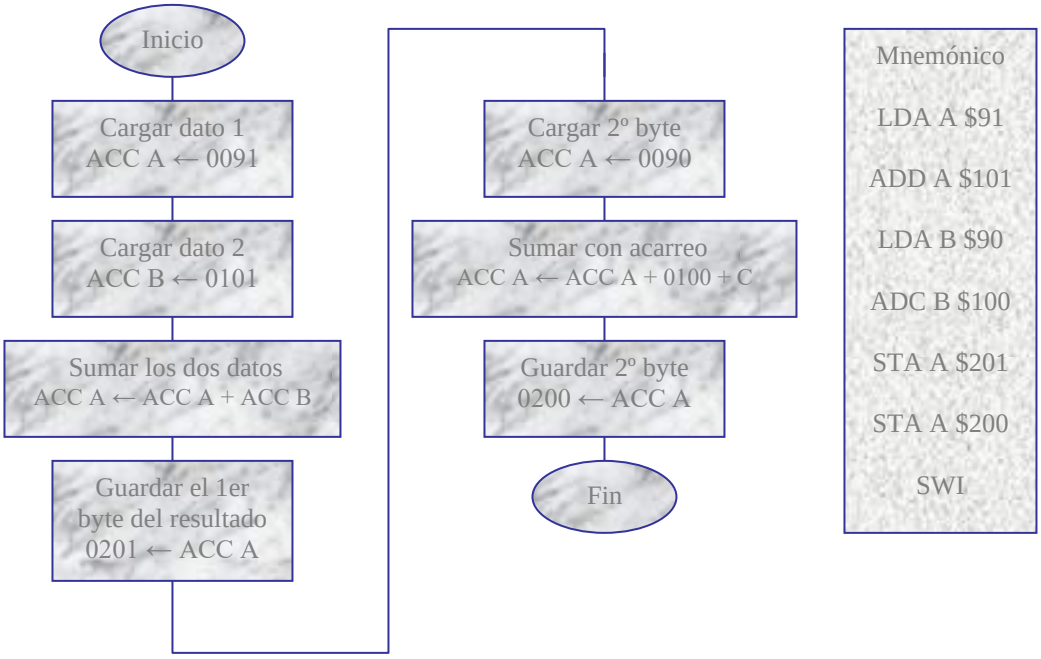
Ejercicios.

Programa: Se requiere efectuar la suma de dos números de un byte cada uno ubicados en las localidades 0030 y 0031, guardando el resultado en la localidad 0032.



Mnemónico	Código
LDA A $30	96 30
LDA B $31	D6 31
ABA	1B
STA A $32	97 32
SWI	7E 3F 00

Programa: se requiere efectuar las uma de dos números de 2 bytes cada uno ubicados en 0090, 0091 y 0100, 0101 guardando el resultado en 0200 y 0201.



EJMPLO1		M30	EJEMPLO2		M91	M90
	+	M31		+	M100	M101
		M32			M200	M201

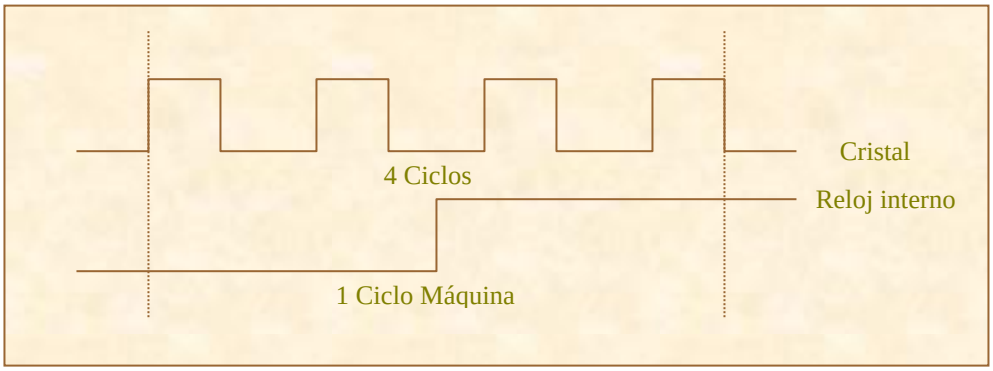
Subutinas de retardo.

Diseñar una rutina de retardo para 0.1 segundo. Tenemos el dato a considerar la frecuencia del microprocesador a:

3.579544 MHz.

0000 LDX CE 18 F7

0003	DEX	09
0004	NOP	01
0005	NOP	01
0006	NOP	01
0007	BNE FA	26 FA
0009	SWI	7E 3F 00



0.1 segundo x 89 x 488 – 15 =89473 / 14 = 6390
6390 → 18 f7 h

3.579544 MHz / 4 = 894.886 Hz = 1 ciclo / seg.

Para retraso de 4.00 MHz en el micro procesador 4.00 MHz → 4 = 1MHz

Temporización de un segundo.

0040	FF 00 7E	STX $007E	Guardar IX
0043	AE C7	LDX $#AEC7	Carga nuevo valor
0046	08	INX	Hacer tiempo
	09	DEX	
	09	DEX	
	01	NOP	
	01	NOP	
	26 F9	BNE (0046)	
	FE 00 7E	LDX 00 7E	Recupera IX
	08	INX	Hacer tiempo
	09	DEX	
	39	RTS	Regresa

Nombre de subrutina: SEG

Localización: 0040
Apilamiento: 2 bytes.

Temporización de un minuto.

0000	LDX $#003C	CE 003C
0003	JSR SEG	BD 0040
0006	DEX	09
0007	BNE $03	26 FD
000A	SWI	7E 3F 00

Tarjeta de programación del 6800

SUMAS

		Modos de Direccionamiento												Operación Aritmético / Lógica	Reg. Cod. Cond					
		Inmediato			Directo			Indexado			Extendido			Implícito				Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		refieren a los contenidos	H	I	N	Z	V	C
Sumar	ADD A	8B	2	2	9B	3	2	AB	5	2	BB	4	3					A + M → A	↕	•	↕	↕	↕	↕
	ADD B	CB	2	2	DB	3	2	EB	5	2	FB	4	3					B + M → B	↕	•	↕	↕	↕	↕
Sumar acumuladores	ABA													1B	2	1		A + B → A	↕	•	↕	↕	↕	↕
Suma con acarreo	ADC A	89	2	2	99	3	2	A9	5	2	B9	4	3					A + M + C → A	↕	•	↕	↕	↕	↕
	ADC B	C9	2	2	D9	3	2	E9	5	2	F9	4	3					B + M + C → B	↕	•	↕	↕	↕	↕

OPERACIÓN AND

		Modos de Direccionamiento												Operación Aritmético / Lógica	Reg. Cod. Cond					
		Inmediato			Directo			Indexado			Extendido			Implícito				Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		refieren a los contenidos	H	I	N	Z	V	C
Operación AND	AND A	84	2	2	94	3	2	A4	5	2	B4	4	3					A • M → A	•	•	↕	↕	R	•
	AND B	C4	2	2	D4	3	2	E4	5	2	F4	4	3					B • M → B	•	•	↕	↕	R	•
Operación de bits	BIT A	85	2	2	95	3	2	A5	5	2	B5	4	3					A • M	•	•	↕	↕	R	•
	BIT B	C5	2	2	D5	3	2	E5	5	2	F5	4	3					B • M	•	•	↕	↕	R	•

LIMPIAR Y COMPARACIÓN

		Modos de Direccionamiento												Operación Aritmético / Lógica	Reg. Cod. Cond					
		Inmediato			Directo			Indexado			Extendido			Implícito				Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		refieren a los contenidos	H	I	N	Z	V	C
Borrar (puesta a cero)	CLR							6F	7	2	7F	6	3					00 → M	•	•	R	S	R	R
	CLR A													4F	2	1		00 → A	•	•	R	S	R	R
	CLR B													5F	2	1		00 → B	•	•	R	S	R	R
Comparación	CMP A	81	2	2	91	3	2	A1	5	2	B1	4	3					A - M	•	•	↕	↕	↕	↕
	CMP B	C1	2	2	D1	3	2	E1	5	2	F1	4	3					B - M	•	•	↕	↕	↕	↕
Comparación de acumuladores	CBA													11	2	1		A - B	•	•	↕	↕	↕	↕

COMPLEMENTO

		Modos de Direccionamiento												Operación Aritmético / Lógica	Reg. Cod. Cond					
		Inmediato			Directo			Indexado			Extendido			Implícito				Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		refieren a los contenidos	H	I	N	Z	V	C
Complemento a 1	COM							63	7	2	73	6	3					M' → M	•	•	↕	↕	R	S
	COM A													43	2	1		A' → A	•	•	↕	↕	R	S
	COM B													53	2	1		B' → B	•	•	↕	↕	R	S
Complemento a 2 (negar)	NEG							60	7	2	70	6	3					00 - M → M	•	•	↕	↕	✱	✱

NEGACIÓN

		Modos de Direccionamiento												Operación Aritmético / Lógica	Reg. Cod. Cond					
		Inmediato			Directo			Indexado			Extendido			Implícito				Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		refieren a los contenidos	H	I	N	Z	V	C
Negación	NEG A													40	2	1		00 - A → A	•	•	↕	↕	✱	✱
	NEG B													50	2	1		00 - B → B	•	•	↕	↕	✱	✱
Ajustar decimal A	DAA													19	2	1		Conversor de suma binaria de caracteres BCD en Formato BCD	•	•	↕	↕	↕	✹

DECREMENTAR Y OR

		Modos de Direccionamiento												Operación Aritmético / Lógica	Reg. Cod. Cond					
		Inmediato			Directo			Indexado			Extendido			Implícito				Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		refieren a los contenidos	H	I	N	Z	V	C
Decrementar	DEC							6A	7	2	7A	6	3					M - 1 → M	•	•	↕	↕	4	•
	DEC A													4A	2	1		A - 1 → A	•	•	↕	↕	4	•
	DEC B													5A	2	1		B - 1 → B	•	•	↕	↕	4	•
OR exclusivo	EOR A	88	2	2	98	3	2	A8	5	2	B8	4	3					A ⊕ M → A	•	•	↕	↕	R	•
	EOR B	C8	2	2	D8	3	2	E8	5	2	F8	4	3					B ⊕ M → B	•	•	↕	↕	R	•

Tarjeta de programacióndel 6800

Modos de Direccionamiento — Operación Aritmético / Lógica — Reg. Cod. Cond

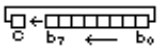
INCREMENTO Y CARGA		Inmediato			Directo			Indexado			Extendido			Implícito			Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	refieren a los contenidos	H	I	N	Z	V	C
Incrementar	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⌖	•
	INC A													4C	2	1	A + 1 → A	•	•	↕	↕	⌖	•
	INC B													5C	2	1	B + 1 → B	•	•	↕	↕	⌖	•
Cargar acumulador	LDA A	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDA B	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•

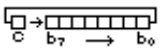
Modos de Direccionamiento — Operación Aritmético / Lógica — Reg. Cod. Cond

OR, INTRODUCIR Y EXTRAER		Inmediato			Directo			Indexado			Extendido			Implícito			Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	refieren a los contenidos	H	I	N	Z	V	C
OR inclusiva	ORA A	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORA B	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Introducir datos	PSH A													36	4	1	A → Msp.SP - 1 → SP	•	•	•	•	•	•
	PSH B													37	4	1	B → Msp.SP - 1 → SP	•	•	•	•	•	•
Extraer datos	PUL A													32	4	1	SP + 1 → SP.Msp → A	•	•	•	•	•	•
	PUL B													33	4	1	SP + 1 → SP.Msp → B	•	•	•	•	•	•

Modos de Direccionamiento — Operación Aritmético / Lógica — Reg. Cod. Cond

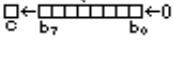
ROTACIÓN		Inmediato			Directo			Indexado			Extendido			Implícito			Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	refieren a los contenidos	H	I	N	Z	V	C
Rotación izquierda	ROL							69	7	2	79	7	3				M	•	•	↕	↕	⌖	↕
	ROL A													49	2	1	A	•	•	↕	↕	⌖	↕
	ROL B													59	2	1	B	•	•	↕	↕	⌖	↕
Rotación derecha	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⌖	↕
	ROR A													46	2	1	A	•	•	↕	↕	⌖	↕
	ROR B													56	2	1	B	•	•	↕	↕	⌖	↕

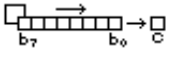


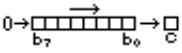


Modos de Direccionamiento — Operación Aritmético / Lógica — Reg. Cod. Cond

DESPLAZAMIENTO		Inmediato			Directo			Indexado			Extendido			Implícito			Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	refieren a los contenidos	H	I	N	Z	V	C
Desplazamiento	ASL							68	7	2	78	7	3				M	•	•	↕	↕	⌖	↕
aritmético a la izquierda	ASL A													48	2	1	A	•	•	↕	↕	⌖	↕
	ASL B													58	2	1	B	•	•	↕	↕	⌖	↕
Desplazamiento	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⌖	↕
aritmético a la derecha	ASR A													47	2	1	A	•	•	↕	↕	⌖	↕
	ASR B													57	2	1	B	•	•	↕	↕	⌖	↕
Desplazamiento	LSR							64	7	2	74	6	3				M	•	•	R	↕	⌖	↕
lógico a la derecha	LSR A													44	2	1	A	•	•	R	↕	⌖	↕
	LSR B													54	2	1	B	•	•	R	↕	⌖	↕







Modos de Direccionamiento — Operación Aritmético / Lógica — Reg. Cod. Cond

ALMACENAMIENTO Y RESTA		Inmediato			Directo			Indexado			Extendido			Implícito			Todas las etiquetas se	5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	refieren a los contenidos	H	I	N	Z	V	C
Almacenar acumulador	STA A				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STA B				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Resta	SUB A	80	2	2	90	3	2	A0	5	2	B0	4	3				A - M → A	•	•	↕	↕	↕	↕
	SUB B	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B - M → B	•	•	↕	↕	↕	↕
Restar acumuladores	SBA													10	2	1	A - B → A	•	•	↕	↕	↕	↕
Resta con acarreo	SBC A	82	2	2	92	3	2	A2	5	2	B2	4	3				A - M - C → A	•	•	↕	↕	↕	↕
	SBC B	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B - M - C → B	•	•	↕	↕	↕	↕

Tarjeta de programacióndel 6800

Modos de Direccionamiento

TRANSFERENCIA Y PRUEBA		Inmediato			Directo			Indexado			Extendido			Implícito			Operación Aritmético / Lógica	Reg. Cod. Cond					
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	Todas las etiquetas se refieren a los contenidos	5 H	4 I	3 N	2 Z	1 V	0 C
Transferir acumuladores	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Prueba (test) cero o menos	TST							6D	7	2	7D	6	3				M - 00	•	•	↕	↕	R	R
	TST A													4D	2	1	A - 00	•	•	↕	↕	R	R
	TST B													5D	2	1	B - 00	•	•	↕	↕	R	R

Modos de Direccionamiento

Instrucciones de manipulación de registro índice y pila		Inmediato			Directo			Indexado			Extendido			Implícito			Operación Aritmético / Lógica	Reg. Cod. Cond					
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		5 H	4 I	3 N	2 Z	1 V	0 C
Comparar registro índice	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				XH - M X → (M + 1)	•	•	•	↕	•	•
Decrementar reg. Índice	DEX													09	4	1	X - 1 → X	•	•	•	↕	•	•
Decrementar la pila	DES													34	4	1	SP - 1 → SP	•	•	•	•	•	•
Incrementar reg. Índice	INX													08	4	1	X + 1 → X	•	•	•	↕	•	•
Incrementar la pila	INS													31	4	1	SP + 1 → SP	•	•	•	•	•	•
Cargar registro Índice	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				M → X (M + 1) → X	•	•	•	↕	R	•
Cargar puntero de Pila	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				M → SP (M + 1) → SP	•	•	•	↕	R	•
Almacenar reg. Índice	STX				DF	5	2	EF	7	2	FF	6	3				X → M X → (M + 1)	•	•	•	↕	R	•
Almacenar puntero de pila	STS				9F	5	2	AF	7	2	BF	6	3				SP → M SP → (M + 1)	•	•	•	↕	R	•
Reg. Índice - puntero pila	TXS													35	4	1	X - 1 → SP	•	•	•	•	•	•
Puntro pila - reg. Índice	TSX													30	4	1	SP + 1 → X	•	•	•	•	•	•

Modos de Direccionamiento

Instrucciones de salto y bifurcación		Inmediato			Directo			Indexado			Extendido			Implícito			Operación Aritmético / Lógica	Reg. Cod. Cond					
Instrucción	**Mnemónico**	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		5 H	4 I	3 N	2 Z	1 V	0 C
Bifurcación incondicional	BRA	20	4	2													Ninguna	•	•	•	•	•	•
Bifurcar si no hay acarreo	BCC	24	4	2													C = 0	•	•	•	•	•	•
Bifurcar si hay acarreo	BCS	25	4	2													C = 1	•	•	•	•	•	•
Bifurcar si es igual a 0	BEQ	27	4	2													Z = 1	•	•	•	•	•	•
Bifurcar si es ? 0	BQE	2C	4	2													N ⊕ V = 0	•	•	•	•	•	•
Bifurcar si es > 0	BGT	2E	4	2													Z + (N ⊕ V) = 0	•	•	•	•	•	•
Bifurcar si es mayor	BHI	22	4	2													C + Z = 0	•	•	•	•	•	•
Bifurcar si es ? 0	BLE	2F	4	2													Z + (M ⊕ V) = 1	•	•	•	•	•	•
Bifurcar si es menor o igual	BLS	23	4	2													C + Z = 1	•	•	•	•	•	•
Bifurcar si es < 0	BLT	2D	4	2													N ⊕ V = 1	•	•	•	•	•	•
Bifurcar si es negativo	BMI	2B	4	2													N = 1	•	•	•	•	•	•
Bifurcar si es ? 0	BNE	26	4	2													Z = 0	•	•	•	•	•	•
Bifurcar si no hay sobreflujo	BVC	28	4	2													V = 0	•	•	•	•	•	•
Bifurcar si hay sobreflujo	BVS	29	4	2													V = 1	•	•	•	•	•	•
Bifurcar si es positivo	BPL	2A	4	2													N = 0	•	•	•	•	•	•
Bifurcar a subrutina	BSR	8D	8	2													Bifurca a subrutina	•	•	•	•	•	•
Salto	JMP				6E	4	2	7E	3	3							Salta a una localidad específica	•	•	•	•	•	•
Salto a subrutina	JSR				AD	8	2	BD	9	3							Salta a una subrutina del programa	•	•	•	•	•	•
No operación	NOP										01	2	1				Avavza solo el contador de programa	•	•	•	•	•	•
Retorno de interrupción	RTI										3B	10	1				Regresa después de interrupción	•	•	•	•	•	•
Retorno de subrutina	RTS										39	5	1				Regresa de una subrutina	•	•	•	•	•	•
Interrupción por software	SWI										3F	12	1				Interrupción de un programa	•	•	•	•	•	•
Esperar interrupción	WAI										3E	9	1				Espera una interrupción	•	•	•	•	•	•

Tarjeta de programación del 6800

Instrucciones de manipulación del Registro de código de condiciones

		Implícito				5	4	3	2	1	0
Instrucción	**Mnemónico**	OP	~	#	Operación Lógica	H	I	N	Z	V	C
Borrar acarreo	CLC	0C	2	1	$0 \rightarrow C$	•	•	•	•	•	R
Borrar máscara de interrupción	CLI	0E	2	1	$0 \rightarrow 1$	•	R	•	•	•	•
Borrar sobreflujo	CLV	0A	2	1	$0 \rightarrow V$	•	•	•	•	R	•
Poner a 1 el acarreo	SEC	0D	2	1	$1 \rightarrow C$	•	•	•	•	•	S
Poner a 1 máscara de interrupción	SEI	0F	2	1	$1 \rightarrow 1$	•	S	•	•	•	•
Poner a 1 sobreflujo	SEV	0B	2	1	$1 \rightarrow V$	•	•	•	•	S	•
Acumulador A - CCR (Transferencia)	TAP	06	2	1	$A \rightarrow CCR$	•	•	•	•	•	•
CCR - Acumulador A (Transferencia)	TPA	07	2	1	$CCR \rightarrow A$	•	•	•	•	•	•

LEYENDA:		Símbolos de Código de Condición
OP Código de operación (hexadecimal)	+ OR inclusiva lógica	H Semiacarreo del bit 3
~ Número de ciclos del CPU	⊕ OR exclusiva lógica	I Máscara de interrupción
# Número de bytes del programa	M' Complemento de M	N Negativo (bit de signo)
+ Suma aritmética	→ Transferencia a	Z Cero (byte)
- Resta aritmética	0 Bit = cero	V Sobreflujo (complemento a 2)
. Operación AND lógica	00 Byte = 00	C Acarreo desde bit 7
Msp Contenido de la posición de memoria apuntada por el puntero de pila		R Reset (puesta a 0)
		S Set (puesta a 1)
		↕ Test. Se pone a 1 si es cierto, a 0 en caso contrario
		• No afectado