18

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

Simulación de Microprocesadores

T E S I S

QUE PARA OBTENER EL TITULO DE:
INGENIERO MECANICO ELECTRICISTA

P R E S E N T A N:
ARTURO HERNANDEZ PEÑA
HUMBERTO SANCHEZ CRUZ
ENRIQUE VIZCARRA VALDEZ

DIRECTOR DE TESIS:
M. EN C. JUAN ANTONIO NAVARRO MARTINEZ

Premio Nacional de Ciencia y Tecnología
BANAMEX

Cuautitlán Edo. de Méx. 1984

SIMULACION
DE
MICROPROCESADORES

C O N T E N I D O

I N T R O D U C C I O N:

El ábaco es, sin lugar a dudas, la primera calculadora: se utilizó, desde el primer milenio A. de C. en varias culturas; los españoles, al desembarcar en nuestras costas, se encontraron con un conjunto de varillas paralelas unidas a una pieza de madera en la que se podían ensartar cuentas: el ábaco azteca. Los ábacos se usan actualmente en muchas partes del mundo, pero destaca el ábaco japonés moderno llamado SOROBAN, con el que pueden hacerse operaciones a una velocidad impresionante. En Japón se le da tanta importancia que existe un instituto de investigación del ábaco.

Ahora nos ocuparemos de esas cajas negras que no trabajan solas: las computadoras, aunque no hay que olvidar la observación de A. M. Turing. "Una computadora es, esencialmente un dispositivo que permite recibir, almacenar, manipular y comunicar la información". Las dos grandes partes que la constituyen son el equipo (en inglés HARDWARE) y el conjunto de programas (en inglés SOFTWARE).

Dentro del equipo, la parte más importante de la computadora es la unidad central de procesamiento (CPU en inglés, de las siglas de Central Processing Unity), que se encarga de realizar las operaciones aritméticas como sumar, restar, multiplicar y dividir, de las funciones lógicas y de supervisar la operación correcta de todo el equipo en dos partes: la Unidad de Control y la Unidad Aritmética y Lógica. En las computadoras grandes la CPU ocupa un volumen conside-

rable; hoy, sin embargo, con la aparición de las microcomputadoras, toda la CPU cabe en un solo CHIP (trocito) de 25 milímetros cuadrados y, debido a su tamaño tan reducido, se le llama microprocesador. En un microprocesador de estas dimensiones caben miles de componentes electrónicos que se encargan de realizar las operaciones lógicas y aritméticas de la microcomputadora.

Otra parte fundamental de la computadora es la memoria, en donde se almacena toda la información. La memoria de una computadora la constituye el conjunto de células elementales. En cada una de estas se necesita conocer la dirección la cual se hace con un contador y un decodificador, Este último es un circuito que establece una relación unívoca entre las líneas de salida y los números binarios codificados en la entrada. El contador sería equivalente a un cuentakilómetros en sistema binario y externo a la memoria. La palabra u octeto pasara por un registro de I/O (entrada/salida) . La capacidad de memoria se mide en kilobytes ó kilooctetos y actualmente los microprocesadores tienen una memoria de 16 a 64 kilobytes e inclusive de 1 megabyte.

La memoria de las computadoras puede ser de varios tipos: la que puede ser leída o escrita, llamada memoria viva o, en inglés, Random Access Memory (RAM); la que solo puede ser leída, llamada memoria muerta o, en inglés, Read - Only Memory (ROM); otras memorias se pueden borrar y programar únicamente leyendo (EPROM y PROM) . Estas dos últimas forman parte de la memoria muerta o ROM .

Tanto la memoria RAM como la ROM es fabrican en un solo elemento de dimensiones muy pequeñas y de técnicas complejas: el chip, y constituyen la memoria RAM y el chip de memoria ROM. Ambas conforman la memoria principal de la com-

putadora. Sin embargo, puede existir una memoria secundaria o auxiliar para aumentar la capacidad de ésta, ya sea usando un cassette y una grabadora o un disco flexible, floppy disk o diskette, con un operador de disco en el que una cabeza magnética lo recorre radialmente mientras gira, unido a un controlador de disco que, a su vez, está unido a la memoria principal. Los discos flexibles resultan más convenientes que los cassettes; la información queda grabada magnéticamente en ellos, y aunque su capacidad (entre 125 y 500 kilobytes) es generalmente mayor que la memoria principal, tiene la desventaja que la información se recupera lentamente y en grandes bloques.

Otra parte esencial de la computadora son los puertos de entrada/salida; en estos chips las señales se convierten de digitales en analógicas o viceversa. La conversión digital-analógica se utiliza para convertir el lenguaje de la máquina en señales analógicas, para ver en una pantalla de televisión las instrucciones que sigue la máquina, los resultados o datos de un programa, para usar graficadoras, tabletas o impresoras, y la analógica-digital se emplea para convertir la salida de los transductores analógicos de la computadora en señales digitales que pueda manejar el microprocesador y la memoria.

El equipo periférico de una microcomputadora lo forman en primer lugar, un tubo de rayos catódicos, llamado monitor, que puede ser una pantalla de televisión común y corriente, donde se pueden ver 24 líneas de texto de 80 caracteres cada una. Las impresoras, ya sean térmicas o de matriz de puntos, con las que se pueden imprimir, por ejemplo, 200 caracteres por segundo. Los modem, que permiten comunicarse vía la línea de teléfono con otras computadoras que se encuentran muy lejos o con el usuario. El equipo para los dis-

cos flexibles, hechos de plastico mylar y recubiertos de material magnético, y las unidades que lo manejan y controlan, los graficadores y tabletas graficadoras, que convierten la información digital a puntos son periféricos importantes de una microcomputadora.

La otra parte de la computadora la forman el conjunto de programas. Entre ellos el más importante es el llamado sistema operativo, constituido por los programas que aumentan la eficiencia de la computadora, eliminando tiempos de espera, errores que puede cometer el usuario y dando prioridades. Los programas son las instrucciones que se le proporcionan a la computadora en una secuencia lógica que sea capaz de realizar.

El lenguaje de la máquina es binario, pero como este lenguaje resulta complicado para programar, se hicieron necesarios otros lenguajes más faciles de manejar; primero aparecieron los lenguajes ensambladores que combinaban el lenguaje binario de la computadora a números, simbolos y palabras. Finalmente aparecieron los lenguajes denominados de alto nivel, como son : el FORTRAN (FORmula TRANslation), BASIC (Beginer's All Purpose Symbolic Instruccion Code), ALGOL (ALgorithmic Language), COBOL (COmmon Business Language), PL/I (Programing Language), APL (A. Programing Language), PASCAL, C. FORTH, LISP, entre otros.

Las dos grandes clases de programas son el intérprete y el compilador. Cuando se escribe un programa, el lenguaje traducido se guarda en una sucesión de comandos de alto nivel. Cuando el programa se corre, un segundo programa (el intérprete) traduce cada comando y lo pone en una sucesión de instrucciones adecuadas al lenguaje de máquina, que lo lleva a cabo rápidamente. Con un programa compilador toda la

traducción se termina antes de que la ejecución empiece. Así, con un programa intérprete se cuenta con la ventaja de que el resultado de cada operación del programa se pueda conocer paso a paso, mientras que con un programa compilador, aunque no puede verse paso a paso el programa, corre mucho más rápido.

La electrónica tiene actualmente un desarrollo muy rápido, por lo que provoca que los equipos para desarrollo se conviertan en obsoletos en un corto plazo, por ejemplo: a principios de la década de los 70's no se diseñaba equipo en base a los microprocesadores, cosa que a mediados de dicha década ya era una realidad, aunque para muy pocos microprocesadores, pero actualmente los equipos se estan diseñando en base a muchos otros microprocesadores. Mientras no se hacian diseños con estos circuitos, el problema no era tan crítico, bastaba con que los profesionales actualizaran constantemente sus conocimientos; pero con la aparición de los microprocesadores el problema se acrecentó pues además de tener que actualizar los conocimientos de los profesionales se tienen que adquirir computadoras las cuales sirven para un solo microprocesador y estas en un corto tiempo se convierten en obsoletas debido a las innovaciones.

Por lo anteriormente expuesto, podemos ver la importancia de que las computadoras que se tienen se adapten a la evolución de la electrónica incluyéndoles las innovaciones que surgen al pasar el tiempo.

En materia de computadoras, una forma de adaptarlas a la evolución de la electrónica, es mediante programas simuladores, los cuales al ser ejecutados darán la sensación al operario de que se esta manejando otro tipo de máquina, todo esto con el fin de poder desarrollar en el futuro proyectos

basados en otros microprocesadores sin tener la necesidad de una máquina diferente.

En la elaboracion de un programa simulador es muy importante la estructuración del mismo. Esto es, primero se hace un planteamiento general del problema, pasando a una etapa intermedia y finalmente a la etapa de código. Este tipo de estructura y la finalidad de cada etapa se explican en el capítulo tercero.

Además la estructura del programa llevado a cabo puede ser muy útil para el desarrollo de programas posteriores sobre simulación de microprocesadores. Teniendo la facilidad de actualizar los programas simuladores originales de acuerdo a las nuevas versiones de los mismos.

Con esta tesis se intenta introducir al lector en la simulación de lenguajes de programación, ya que como se explicó anteriormente el método empleado podrá servir para la simulación de cualquier otro lenguaje de microprocesador, siendo el objetivo principal hacer un estudio completo de la programación del microprocesador MC6809 .

Debido a la dependencia tecnológica que se tiene con los Estados Unidos es muy común utilizar términos y siglas en el idioma Inglés, por lo que unicamente al principio de esta tesis aparecerán dichos términos y siglas con su traducción al Español, pero en lo sucesivo serán expresados unicamente en el idioma Inglés, lo cual constituye una buena razón para estudiar la tesis desde el principio. Además de que la estructura de esta tesis permite comprender todos los aspectos para la programación del microprocesador MC6809 con un mínimo de conocimientos tanto de electrónica como de computación.

CAPITULO I

SIMULACION Y LENGUAJES

1.1 SIMULACION

El proceso de simulación involucra hacer creer a la máquina que se esta operando de acuerdo a las reglas establecidas por su descripción. Este proceso va acompañado necesariamente de la escencia de una idea.

La menos común pero más efectiva simulación se hace con la idea de una computadora. La computadora hará exactamente lo que se le indique y por lo tanto estará comprobando la completa exactitud de las descripciones y definiciones.

Si únicamente son comprobadas las definiciones lógicas, el estudio de los circuitos eléctricos se separa del problema de diseño. La computadora simula un algoritmo de estado de máquina para copiar la ejecución de cada estado en

la descripción. La computadora procesará la descripción de - estado de acuerdo a las apropiadas definiciones en la forma más conveniente para producir el resultado correcto en el -- próximo período estabi

La simulación en la computadora requiere cambiar la descripción de la máquina y sus definiciones dentro de una - forma aceptable para la computadora. La computadora acepta u na cadena de carácteres. El cambio es logrado por la traslación de estatutos definidos, conexiones de diagramas a blo-- ques, ecuaciones, tablas, mapas y las cartas ASM dentro de - un lenguaje llamado el LENGUAJE SIMULADOR. El lenguaje simulador es construido para hacer esta transición fácil y repre sentar todas las descripciones como cadena de caracteres.

El lenguaje simulador es alimentado en una computado ra, la cual es programada para leer y organizar la informa-- ción en un listado de códigos los cuales representan la in-- terpretación de instrucciones y definiciones. Este listado - es entonces interpretado una vez más por otro programa, llamado INTERPRETE, el cual provee el medio eficaz para ejecu-- tar la información en el listado y actualiza los elementos - de memoria para cada nuevo estado del que está simulando a - la máquina. El interprete tiene factores los cuales proveen los medios para la evaluación de la ejecución del simulador de la máquina hasta las salidas hacia una impresora o un mos trador óptico (display). El diseñador podrá alterar el tipo

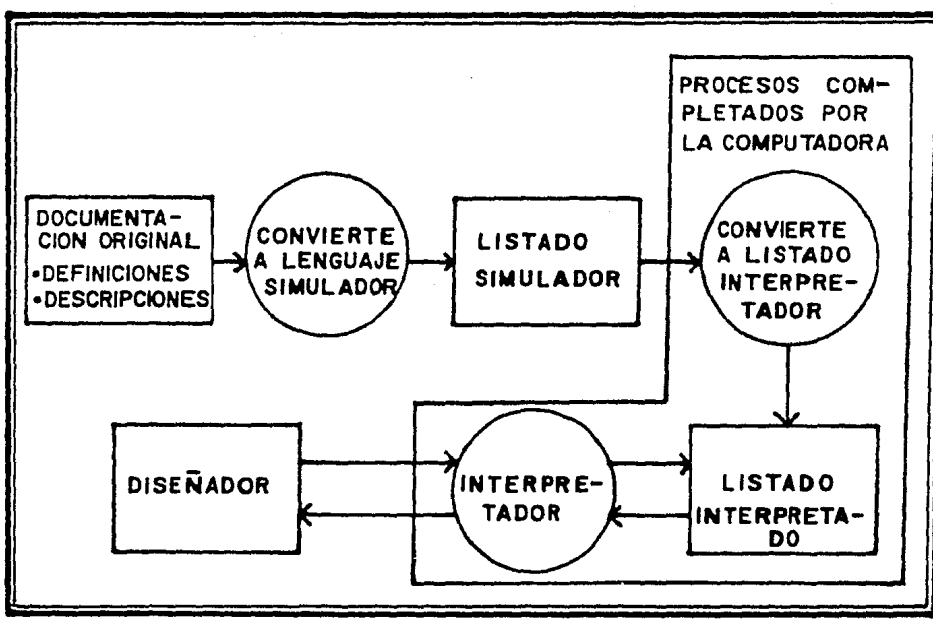


Fig. I.I El proceso de simulación por computadora.

de información que desee ver durante el curso de la simulación. La figura 1.1 describe los procesos involucrados al usar una computadora para simular un estado de máquina. En este diagrama los círculos representan un proceso y los rectángulos representan un resultado.

El interpretador provee acceso al lenguaje simulador, en las siguientes formas:

1).- Elementos seleccionados de memoria pueden ser revisados en cada estado de tiempo. Estos elementos son seleccionados por los comandos PRINT.

2).- La máquina que se simula puede estar diseñada para ejecutar un estado en un tiempo a correr hasta que alguna condición pre-establecida sea satisfecha. El parado de cada estado es llamado ALTO ABSOLUTO. Cuando el parado se realiza únicamente cuando se satisface una condición es llamado ALTO CONDICIONAL. Los altos absolutos y condicionales pueden ser actualizados para cada nivel de una máquina interpretada. Esta característica permite comprobar las salidas contra las salidas deseadas para cada estado de tiempo.

3).- Las entradas pueden ser seleccionadas por el diseñador para probar trayectorias alternadas basadas en las entradas. Esta característica permite probar las respuestas de interfases.

Con la ayuda de estas tres características, la máqui

na simulada puede ser corrida y lógicamente comprobada contra las respuestas deseadas.

La simulación realiza cambios fáciles para el comprobado. El cambio es hecho en el estado del simulador y la computadora hace los cambios apropiados en el listado de interpretación. El interpretador permite entonces accesos convenientes para comprobar la validez de un cambio.

Usando un simple simulador. Para dar una idea de lo que envuelve y usa un simulador, a continuación consideraremos algunas de las capacidades del simulador de Motorola para su microcomputadora 6800. El programa simulador es primeramente llamado con un comando RUN MPSSIM. El empezará por preguntar el nombre del archivo del código objeto del programa a ser simulado, a lo cual nosotros deberemos responder — MYFILE. Si nosotros queremos que el simulador ejecute 20 — (hexadecimal) instrucciones, empezando en la dirección 0100 y desplegando una línea de rastreo (TRACE LINE) de todos los registros de la CPU después de la ejecución de cada instrucción, nosotros pondremos:

SR P0100.T20

lo cual da dos comandos al simulador, separados por un período. El comando SR (juego de registros, set registers) es usado para poner el contenido de cualquiera de los registros de la CPU (Unidad Central de Procesamiento) a cualquier valor deseado. En este caso nosotros estamos poniendo únicamente P,

el contador de programa, a la dirección 0100. El comando T - (rastreo, trace) dice, las siguientes 20 (hexadecimal) ins-- trucciones a ejecutar, incluyendo una línea de rastreo (trace line) después de cada una. Después de ejecutar estas instrucciones el simulador esperará nuestro siguiento comando, el cual puede ser:

DM 0031,8

para mostrar memoria. El contenido de las ocho localidades - consecutivas empezando en la dirección 0031, serán mostradas. En seguida podemos poner el siguiente comando:

SM 0031,0,0,0,0,0,0,99,0.SR P0100.T20

Con lo cual repetiremos la misma rutina ejecutada anteriormente, pero con diferente inicialización. Al contrario de - empezar con todos los ceros en RAM (memoria de acceso aleatorio, random acces memory) este tiempo de localización --- 0037 es puesto a 99 por el comando SM (pone memoria, set memory).

1.2 TIPOS DE LENGUAJES DE PROGRAMACION

En una clasificación muy primaria los lenguajes de programación pueden separarse en 4 clases:

- Lenguajes máquina (números binarios).
- Lenguajes simbólicos directos (escritos en mnemónicos, correspondencia uno a uno entre instrucción en mnemónico y número binario ENSAMBLADOR (ASSEMBLER).
- Lenguajes de alto nivel funcionales o algoritmos (escritos con mnemónicos, cada instrucción se convierte en un conjunto de instrucciones máquina. — FORTRAN, ALGOL, PL/1).
- Lenguajes de alto nivel, conversacionales o dialógicos (de función parecida a la anterior pero cambia en que son interactivas la ejecución y la creación o modificación de instrucciones, BASIC).

El primer ingrediente de la lógica programada (software), el más elemental y a veces el único con el que tiene que enfrentarse el diseñador de un sistema que incluye un microprocesador, es el lenguaje de programación. Un microprocesador realiza las acciones que les especifica su programa. El programa está formado por una secuencia de instrucciones. Una instrucción es un conjunto de bits que tienen un significado para la unidad de control del microprocesador. Este sentido puede ser el desencadenamiento de microprogramas (micro

procesadores microprogramables) o la ejecución de acciones - sobre registros o puertos a través de un circuito cableado.

1.2.1.- Lenguaje de máquina. El conjunto de instrucciones válidas para un microprocesador es lo que se denomina lenguaje de máquina o abreviadamente lenguaje máquina. Programar en lenguaje máquina supone por lo tanto escribir secuencias de números en binario (instrucciones) que son directamente decodificables por los circuitos de la unidad de control e interpretables por los microprogramas de la memoria - de control.

Veamos la dificultades que se plantean al programar directamente en lenguaje máquina.

1.- Los códigos de operación son difíciles de recordar en binario. Cuando se ha estado trabajando - con más de un microprocesador hay que usar una - tabla de equivalencias. La codificación se hace pues lenta y penosa por los números en binario.

2.- Las direcciones de los operandos de las instrucciones son también difíciles de recordar como números binarios. Muchas instrucciones contienen - direcciones relativas a distintos punteros. Estas direcciones relativas son el gran enemigo de las correcciones en que hay que insertar o eliminar instrucciones. Estas operaciones modifican -

las distancias entre instrucciones y acostumbra a ser difícil corregir todas las instrucciones afectadas.

3.- Se plantea el problema de cargar el programa en memoria. Si no se va hacer en RAM directamente, como es de suponer, habrá que trasladar una página llena de unos y ceros a algún medio físico, sea cual sea éste (interruptores, cinta de papel, etc.), el paso será lento y sujeto a numerosos errores mecánicos.

4.- Teniendo en cuenta que muy probablemente el programa no funcionará a la primera, resultará sin duda difícil seguir las ejecuciones de prueba, a través de direcciones en binario.

5.- El lenguaje de máquina es el que produce mayor grado de incompatibilidad entre programas. Un programa escrito en lenguaje máquina sólo puede ser trasladado a otro microprocesador igual al primero. El problema de incompatibilidad no se plantea quizás al equipo que diseña por primera vez un sistema que incluye un microprocesador, pero se presentará seguramente al querer cambiar de modelo de procesador para aprovechar los avances de la tecnología, por razones de precio o necesidades de diseño.

6.- Un programa en lenguaje máquina está tan densa--

mente codificado que es imposible de entender, - tanto por su propio autor al cabo de un cierto - tiempo de haberlo escrito como por los posibles interesados en adaptarlo para un sistema parecido.

7.- La incompatibilidad y la dificultad de conocer - su funcionamiento hacen punto menos que imposible la creación de una biblioteca de programas.

La programación en lenguaje máquina puede sistematizarse y mejorarse con una metodología adecuada. El análisis previo, la confección de diagramas de flujo, el escribir previamente el programa en algún lenguaje simbólico, el confeccionar tablas de símbolos y el empleo de sistemas octal o -- hexadecimal pueden constituir una innegable ayuda.

La automatización de estas ayudas a la programación se concreta en la utilización de los lenguajes ensambladores.

1.2.2.- El lenguaje ensamblador. Al empezar a describir lenguajes distintos del lenguaje máquina hay que notar - que, sea cual sea el empleado habrá que hacer algún tipo de proceso sobre éste para conseguir una versión en lenguaje máquina, único ejecutable por el microprocesador.

Conviene tener en cuenta que con el nombre de ensamblador se conocen dos cosas muy distintas. Se llama ensamblador a un lenguaje simbólico en que se pueden escribir progra

mas para un microprocesador. Recibe el mismo nombre el pro-- grama traductor encargado de convertir (ensamblar) los pro-- gramas escritos en lenguaje simbólico en programas objeto en lenguaje máquina. El ensamblador proporciona tres grandes a- yudas al programador. Le permite utilizar símbolos (mnemóni- cos) para designar operaciones y nombres para designar direc ciones y especificar datos (constantes) en otras formas que binario puro.

Cada ensamblador tiene su lista prefijada de símbolos para - las instrucciones. Esta lista puede ser fija o expandible. - De hecho puede escribirse un ensamblador generalizado en que los nombres de los códigos de operación sean definidos por - el usuario dentro de una longitud razonable (3 a 5 caracte- res).

Mucho más importante si cabe es el manejo de direc-- ciones. Cuando se utiliza lenguaje ensamblador se puede asig nar un nombre a una dirección utilizando éste nombre como e- tiqueta. El programa ensamblador hace equivalentes los nom-- bres con las direcciones. El nombre es libremente inventado por el programador y sólo está limitado en longitud y en lo que se refiere a su primer caracter.

Para el programador esto representa poder denominar a las di recciones por un nombre relacionado con el significado de su contenido. La legibilidad del programa aumenta considerable- mente y el ensamblador pasa a manejar automáticamente todas

las direcciones relativas. La inserción o eliminación de una instrucción no representa ya problemas por cuanto el ensamblador pasa a manejar todas las direcciones y recalcula sistemáticamente todos los desplazamientos (diferencias de dirección) del programa. En el manejo de los direccionamientos (absoluto, relativo, indirecto, inmediato), el ensamblador también ayuda al programador por cuanto suministra una serie de símbolos especiales:

*, .,, X,

que los representa en el programa fuente.

La tercer ayuda es la especificación de constantes. No se debería aceptar un ensamblador que no proporcionase la posibilidad de introducir directamente datos de distintos tipos como decimal, octal, hexadecimal, caracter (ASCII, BCD), y coma flotante, además de proporcionar la posibilidad de algún tipo de aritmética (suma, resta y multiplicación como mínimo) sobre las direcciones. Esta tercera ayuda es valiosísima cuando un programa contiene gran cantidad de constantes. Su conversión manual es pesada, lenta y sujeta a errores.

Además de estas tres ayudas directas a la codificación, la utilización de un ensamblador reporta otras ventajas.

En primer lugar, el ensamblador permite la incorporación de comentarios al texto mismo del programa. Los comentarios son una gran ayuda a la documentación de los programas. Pueden ser frases cortas explicativas o la versión en algún lenguaje de alto nivel del mismo programa, conservando en la

medida de lo posible el paralelismo entre instrucciones de ensamblador y sentencias o frases de alto nivel. Este tipo de documentación es muy útil a la hora de hacer nuevas versiones, ya sea en lenguajes de alto nivel o en otro lenguaje ensamblador.

Muy importante de cara a la producción de los programas es la carga, o sea la operación de llevar al programa ya traducido a memoria para ser ejecutado o probado. El ensamblador deja el programa traducido ya directamente en memoria (RAM) o produce alguna salida sobre algún medio físico (cinta de papel, cassette, floppy). Paralelamente produce un listado del programa con los comentarios, la traducción de las instrucciones y una tabla de equivalencias entre nombres y direcciones del programa.

Con la ayuda de un buen programa ensamblador, un programador con experiencia puede realizar programas de varios miles de instrucciones con un aprovechamiento casi óptimo de memoria y de la potencia del microprocesador. Programando en ensamblador, el programador debe manejar, a la vez, los problemas que emanan del producto que está desarrollando y los concernientes a la estructura y características del microprocesador.

1.2.3.- Extensiones del lenguaje ensamblador. El ensamblador reubicable y el macroensamblador son extensiones naturales del ensamblador. Desde el punto de vista del len--

guaje fuente no son muy distintos a éste.

El ensamblador reubicable produce un código que no - es código objeto puro (lenguaje máquina ejecutable). Necesita un último proceso, realizado por un programa llamado cargador reubicable, que permite colocar el programa en - cualquier lugar de la memoria y no en una posición fija, determinada al escribir el programa fuente. Esto permite constituir una biblioteca de rutinas (matemáticas, de conversión de código, de entrada-salida) y cargar junto con el programa principal las necesarias para su funcionamiento.

Hasta ahora sólo hemos hablado de traducciones que - se efectúan línea a línea. Así una instrucción en lenguaje - ensamblador es traducida a una instrucción de código máquina. Los programas macroensambladores permiten dar un nombre a un conjunto de varias instrucciones (macroinstrucción).

Después de la definición, al aparecer una de estas - macroinstrucciones en el programa, el macroensamblador inserta el conjunto de instrucciones definido, expandiendo una línea del programa fuente a varias del programa objeto. El resultado es algo parecido a una subrutina pero sin los tiempos extras de llamada y retorno.

1.2.4.- Lenguajes de alto nivel. La utilización de - un lenguaje de alto nivel reduce los costos de programación, incrementa la fiabilidad de la lógica programada (software) producida y simplifica el mantenimiento y documentación de -

los programas si los comparamos con la utilización de lenguajes de bajo nivel (máquina o ensamblador). Como contrapartida, la utilización de lenguajes de alto nivel supone la utilización de volúmenes de memoria que son desde un 10 a un 100 por ciento mayores que los que necesitaría un programa equivalente en ensamblador.

La distancia que separa los lenguajes de alto nivel del ensamblador es mucho mayor que la que separa a éste del lenguaje máquina. De hecho, al pasar el programa en lenguaje de alto nivel pasamos a manejar no ya nuestro microprocesador, con su estructura de registros, acumuladores, pilas (stacks) y puertos sino un microprocesador de estructura distinta, concebido no para ser realizado físicamente, sino para adaptarse a la solución de los problemas planteados.
Las instrucciones contienen directamente expresiones aritméticas y lógicas y los datos pueden estructurarse. Se necesitan programas traductores bastante complejos llamados compiladores, para generar programas en código máquina a partir de sentencias en lenguajes de alto nivel.

Vamos a describir a continuación tres ventajas innegables de la utilización de los lenguajes de alto nivel.

1.- Fiabilidad de los programas. Los programas escritos en algún lenguaje de alto nivel son muy compactos.

Se entiende mucho más fácilmente lo que hace cada sentencia o grupo de sentencias.

2.- Rapidez de puesta a punto. La velocidad de codificación para un programador viene a ser de unas 10 instrucciones por día (contando tiempos de preparación, depuración, etc.). Esta velocidad es independiente del lenguaje. Como un mismo programa escrito en un lenguaje de alto nivel puede tener 10 veces menos líneas que uno en ensamblador, el aumento de velocidad es considerable.

3.- Los lenguajes de alto nivel tienen una ventaja aplastante sobre los ensambladores: su vida media. Mientras el lenguaje ensamblador cambia para cada nueva arquitectura de microprocesador, el lenguaje de alto nivel es independiente de estos cambios. Desarrollando el compilador adecuado se pueden tener todos los programas escritos en este lenguaje, adaptados a cualquier nuevo microprocesador. La independencia de los programas respecto al microprocesador utilizado para una aplicación no es sólo algo deseable, sino que es una necesidad si uno no quiere verse atrapado por el desarrollo del software. Actualmente, el desarrollo de nueva programación es mucho más costosa y lenta que la adopción de un nuevo microprocesador, cuando de la adopción de este microprocesador puede depender la permanencia en el mercado.

Si hay que partir de cero a cada cambio, el resultado puede ser desastroso. Si por el contrario hemos adoptado un lenguaje de programación de alto nivel y estándar podremos aprovechar no sólo la propia experiencia sino la de otros grupos que ya se enfrentaron con el problema que nos preocupa.

De lo que se ha dicho no debe inferirse la inutilidad del lenguaje ensamblador. Un microprocesador puede no contar con compiladores. Un lenguaje de alto nivel puede no proporcionar acceso a ciertas características deseadas del microprocesador. En estos casos hay que utilizar forzosamente el lenguaje ensamblador. La no utilización del compilador puede venir dictada también por medidas económicas. El problema es el exceso de memoria que para un programa dado ocupa el código generado por el compilador.

CAPITULO 2

ESTUDIO DEL MC6809

2.1 CARACTERISTICAS DE SOFTWARE

Los modos de direccionamiento de cualquier microprocesador proporcionan la capacidad para direccionar eficientemente la mamoria para obtener datos e instrucciones. El ---- MC6809 tiene un juego versátil de modos de direccionamiento los que le permiten funcionar usando modernas técnicas de -- programación.

Los modos de direccionamiento e instrucciones de el MC6809 son muy compatibles con el MC6800, los modos de direccionamiento viejos han sido conservados y muchos nuevos han sido adicionados.

Un registro directo de página ha sido incluido el — cual permite una página directa de 256 bytes en cualquier lugar del espacio de direcciones de 64K. El registro directo de página es usado para mantener el byte más significativo de la dirección usada en el direccionamiento directo y disminuye el tiempo requerido para el calculo de la dirección.

El direccionamiento de salto relativo para cualquier parte de la memoria (-32768 a +32767) está disponible.

El direccionamiento de contador de programa relativo está también disponible para acceso de datos asi como las — instrucciones de salto.

Los modos de direccionamiento de indexado han sido expandidos para incluir:

Desplazamientos constantes de 0, 5, 8 y 16 bits.
Desplazamientos de acumulador de 8 y 16 bits.
Auto incremento/decremento (operación de stack).

En suma, más modos de direccionamiento de indexado pueden tener un nivel adicional de indirección sumada.

Cualquiera de los registros puede ser metido o sacado a o desde el stack con una sola instrucción.

Una instrucción de multiplicación está incluida, la cual multiplica números binarios en los acumuladores A y B y pone el resultado (binario) en el acumulador D de 16 bits. Esta instrucción de multiplicación binaria también permite - la multiplicación de presición múltiple en complemento 2.

2.2 MODELO DE PROGRAMACION

El modelo de programado (Fig. 2.1) para este procesador contiene 5 registros de 16 bits y 4 de 8 bits que estan disponibles para el programador.

2.2.1.- Registros de Indice (X, Y). Los registros de índice son usados durante el modo de direccionamiento de indexado. La información de dirección en un registro índice es usada en el cálculo de una dirección efectiva. Esta direc--- ción puede ser usada para apuntar directamente a un dato o - puede ser modificada por una constante opcional o desplaza-- miento de registro para producir la dirección efectiva.

2.2.2.- Registros apuntadores de stack (U, S). Dos - registros apuntadores de stack estan disponibles en este procesador. Ellos son: un registro apuntador de stack de usua-- rio (U) controlado exclusivamente por el programador, y un - registro apuntador de stack hardware (S) el cual es usado au

tomáticamente por el procesador durante llamadas a subrutinas e interrupciones, pero, puede también ser usado por el programador. Ambos apuntadores de stack siempre apuntan la parte más alta del stack.

Estos registros tienen las mismas capacidades del modo de direccionamiento de indexado como los registros de índice, y también soportan instrucciones para meter y sacar información del stack. Los cuatro registros indexables (X, Y, U, S) son referidos como registros apuntadores.

2.2.3.- Contador de programa (PC). El registro de contador de programa es usado por este procesador para almacenar la dirección de la próxima instrucción a ser ejecutada. Este puede también ser usado como un registro índice en ciertos modos de direccionamiento.

2.2.4.- Registros acumuladores (A, B, D). Los registros acumuladores (A, B) son registros de 8 bits de propósito general usados para cálculos matemáticos y manipulación de datos.

Ciertas instrucciones concatenan estos registros dentro de un acumulador de 16 bits con el registro A ubicado como el byte más significativo. Cuando son concatenados, este registro es referido como acumulador D.

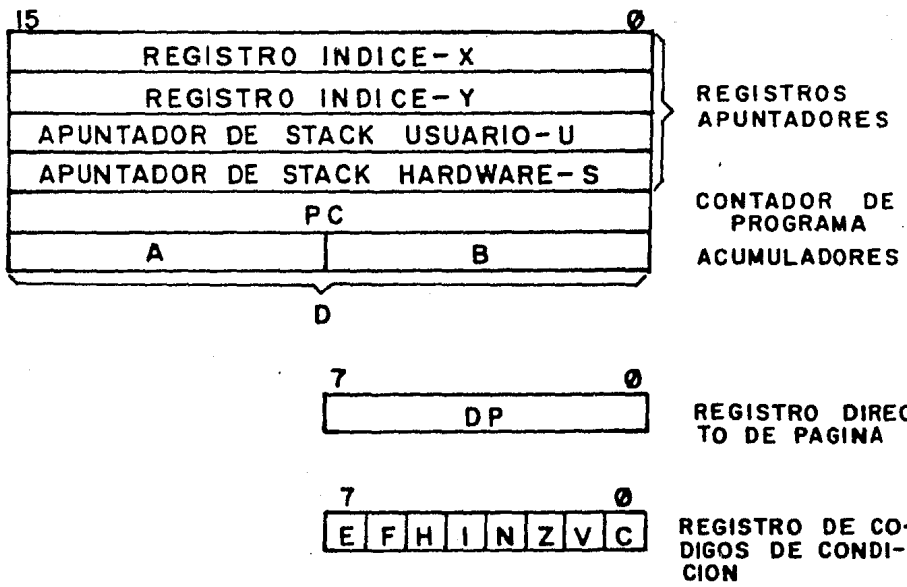


Fig. 2.1 Modelo de programación.

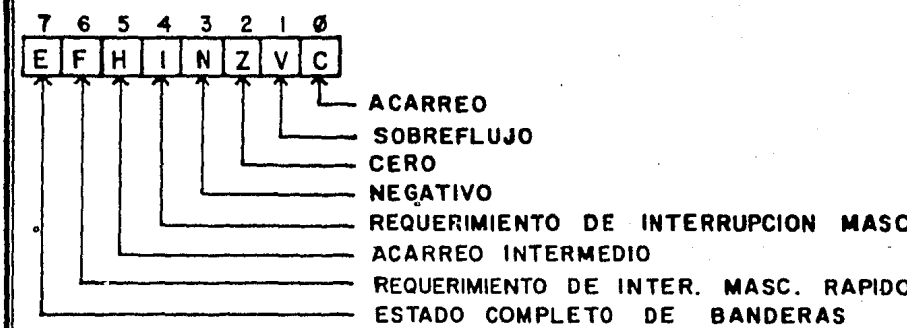


Fig. 2.2 Registro de códigos de condición.

2.2.5.- Registro directo de página (DP). Este registro de 8 bits contiene el byte más significativo de la dirección a ser utilizada en el modo de direccionamiento directo. El contenido de este registro es concatenado con el byte que sigue al código de operación en el modo de direccionamiento directo para formar la dirección efectiva de 16 bits.

El contenido del registro directo de página aparece como los bits A15 hasta A8 de la dirección. Este registro es automáticamente borrado por un reset hardware para asegurar la compatibilidad con el MC6800.

2.2.6.- Registro de códigos de condición (CC). Este registro contiene los códigos de condición y las interrupciones mascarables como se muestra en la Fig. 2.2.

- Bits de códigos de condición. Cinco bits en el registro de código de condición son usados para indicar los resultados de la instrucción que manipula datos. Ellos son: Acarreo intermedio (H), Negativo (N), Cero (Z), Sobreflujo (V), y Acarreo (C).
  El efecto que cada instrucción tiene sobre estos bits será explicado más detalladamente en el capítulo 4.

  a). Carry intermedio (H), bit 5. Este bit es usado para indicar que un carry fué generado desde

el tercer bit en la unidad aritmética lógica como un resultado de una suma de 8 bits.
Este bit no es definido en todas las instrucciones de resta. La instrucción de ajuste decimal (DAAC) usa el estado de este bit para resolver la operación de ajuste.

b). Negativo (N), bit 3. Este bit contiene el valor del bit más significativo del resultado de la operación de datos previa.

c). Cero (Z), bit 2. Este bit es usado para indicar que el resultado de la operación previa fué cero.

d). Overflow (V), bit 1. Este bit es usado para indicar que la operación previa causó un sobreflujo aritmético en complemento 2.

e). Carry (C), bit Ø. Este bit es usado para indicar que un acarreo o un pedido fué generado desde el bit 7 en la unidad aritmética lógica como un resultado de una operación matemática de 8 bits.

- Bits de interrupciones mascarables e indicador de estado completo metido al stack hardware. Dos bits (I y F) son usados como bits mascarables para las

entradas de requerimiento de interrupción y requerimiento de interrupción rápida. Cuando cualquiera o ambos de estos bits estan puestos, su entrada asociada no será reconocida.

Un bit (E) es usado para indicar cuantos registros (todos, o sólo el contador de programa y el de códigos de condición) fueron metidos durante la pasada interrupción.

a). Requerimiento de interrupción mascarable rápido (F), bit 6. Este bit es usado para mascarar (deshabilitar) cualquier línea de requerimiento de interrupción rápido ($\overline{FIRQ}$). Este bit es puesto automáticamente por un reset hardware o después de un reconocimiento de otra interrupción. La ejecución de ciertas instrucciones tales como SWI también impedirán el reconocimiento de una entrada ($\overline{FIRQ}$).

b). Requerimiento de interrupción mascarable (I), bit 4. Este bit es usado para mascarar (deshabilitar) cualquier entrada de requerimiento de interrupción ($\overline{IRQ}$). Este bit se pone automáticamente por un reset hardware o después del reconocimiento de otra interrupción. La ejecución de ciertas --

instrucciones tales como SWI también impedirán el reconocimiento de una entrada $\overline{IRQ}$.

c). Estado completo almacenado (E), bit 7. Este bit es usado para indicar cuantos registros fueron metidos al stack. Cuando está puesto, todos los registros fueron metidos durante la última operación.

Cuando está limpio, sólo el registro de contador de programa y el de códigos de condición fueron metidos durante la última interrupción.

El estado del bit E en el registro de códigos de condición metido es usado por la instrucción de regreso de interrupción (RTI) para determinar el número de registros a ser sacados del stack hardware.

2.3 MODOS DE DIRECCIONAMIENTO

Los modos de direccionamiento disponibles en el M6809 son: Inherente, Inmediato, Extendido, Directo, Indexado (con varios desplazamientos y auto incremento/decremento) y Saltos relativos. Algunos de estos modos de direccionamiento requieren un byte adicional después del código de operación para proporcionar interpretación adicional del direccionamiento. Este byte es llamado byte posterior "postbyte".

Los siguientes párrafos proporcionan una descripción de cada modo de direccionamiento. En estas descripciones el término dirección efectiva es usado para indicar la dirección en memoria desde la cual el argumento para una instrucción es traído o almacenado, o desde la cual el procesamiento de instrucciones va a proceder.

2.3.1.- Inherente. La información necesaria para ejecutar la instrucción está contenida en el código de operación. Algunas operaciones especificando únicamente los registros índices o los acumuladores, y no otros argumentos, están también incluidas en este modo de direccionamiento.

Ejemplo: MUL

2.3.2.- Inmediato. El operando está contenido en uno o dos bytes inmediatos al código de operación. Este modo de direccionamiento es usado para proporcionar valores de datos constantes que no son cambiados durante la ejecución del programa.

Los operandos de 8 o 16 bits son usados dependiendo del tamaño del argumento especificado en el código de operación.

```
Ejemplo: LDA #CR          LDB #%1110000
         LDB #7           LDX #$8004
         LDA #$F0
```

Otra forma del modo de direccionamiento inmediato usa un postbyte para determinar los registros a ser manipulados.
las instrucciones de intercambio (EXG) y transferencia (TFR) usan el postbyte como se muestra en la Fig. 2.3 (A). Las instrucciones para meter y sacar información del stack usan el postbyte para designar los registros a ser metidos o sacados como se muestra en la Fig. 2.3 (B).

2.3.3.- Extendido. La dirección efectiva del argumento está contenida en dos bytes que siguen al código de operación. Las instrucciones que usan el modo de direccionamiento extendido pueden referenciar argumentos en cualquier parte del espacio de direccionamiento de 64K. El direccionamiento extendido generalmente no es usado en programas de posición independiente porque estos proporcionan una dirección absoluta.

Ejemplo: LDA>CAT

2.3.4.- Directo. La dirección efectiva es desarrollada por la concatenación de el contenido de el registro directo de página con el byte inmediato al código de operación. El registro directo de página contiene el byte más significativo de la dirección. Esto permite accesar a 256 localidades dentro de cada una de las 256 páginas. Por lo tanto, el rango de direccionamiento completo está disponible para su ac-

b7	b6	b5	b4	b3	b2	b1	b0
FUENTE (R1)				DESTINO (R2)			

Código*	Registro	Código*	Registro
0000	D (A:B)	0101	Contador de programa
0001	Indice X	1000	Acumulador A
0010	Indice Y	1001	Acumulador B
0011	Apuntador de Stack U	1010	Código de Condición
0100	Apuntador de Stack S	1011	Directo de Página

* Todas las demas combinaciones de bits no estan definidas.

(A) Postbyte para las instrucciones de intercambio (EXG) o transferencia (TFR).

b7	b6	b5	b4	b3	b2	b1	b0
PC	S/U	Y	X	DP	B	A	CC

PC = Contador de Programa
S/U = Apuntador de Stack Hardware/Usuario
Y = Registro Indice Y
X = Registro Indice X
DP = Registro Directo de Página
B = Acumulador B
A = Acumulador A
CC = Registro de Códigos de Condición

(B) Postbyte para las instrucciones push (PSH) o pull (PUL).

Fig. 2.3 Postbyte usado para las instrucciones EXG/TFR, PSH/PUL.

ceso con una sóla instrucción de dos bytes.

Ejemplo: LDA>CAT

2.3.5.- Indexado. En este modo de direccionamiento, uno de los registros apuntadores (X, Y, U o S), y algunas veces el contador de programa (PC) es usado en el cálculo de la dirección efectiva del operando de la instrucción. Los tipos básicos (y sus variantes) de direccionamiento de indexado se muestran en la Tabla 2.1 con su configuración de postbyte usada.

a).- Desplazamiento constante de registro. El contenido del registro designado en el postbyte es sumado a un valor de desplazamiento en complemento 2 para formar la dirección efectiva de el operando de la instrucción. El contenido del registro designado no es afectado por esta suma. Los tamaños de desplazamientos disponibles son:

sin desplazamiento- El registro designado contiene la dirección efectiva.

5 bit- -16 a +15

8 bit- -128 a +127

16 bit- -32768 a +32767

El valor de desplazamiento de 5 bits está contenido en el postbyte. Los valores de los desplazamientos de 8 y 16 bits estan contenidos en el byte o bytes inmediatos al postbyte.

Mode Type	Variation	Direct	Indirect
Constant Offset from Register (twos Complement Offset)	No Offset	1RR00100	1RR10100
	5-Bit Offset	0RRnnnnn	Defaults to 8-bit
	8-Bit Offset	1RR01100	1RR11000
	16-Bit Offset	1RR01001	1RR11001
Accumulator Offset from Register (twos Complement Offset)	A Accumulator Offset	1RR00110	1RR10110
	B Accumulator Offset	1RR00101	1RR10101
	D Accumulator Offset	1RR01011	1RR11011
Auto Increment/Decrement from Register	Increment by 1	1RR00000	Not Allowed
	Increment by 2	1RR00001	1RR10001
	Decrement by 1	1RR00010	Not Allowed
	Decrement by 2	1RR00011	1RR10011
Constant Offset from Program Counter	8-Bit Offset	1XX01100	1XX11100
	16-Bit Offset	1XX01101	1XX11101
Extended Indirect	16-Bit Address	--------	10011111

Tabla 2.1 Postbyte usado para los modelos de direccionamiento de indexado.

Ejemplos: LDA ,X LDY -64000,U
LDB 0,Y LDA 17,PC
LDX 64,000,S LDA There,PCR

b),- Desplazamiento de acumulador de registro. El -- contenido del registro índice o apuntador designado en el -- postbyte es sumado temporalmente a un valor de desplazamiento de complemento 2 contenido en un acumulador (A, B o D) -- también designado en el postbyte. Ni el contenido del registro designado ni el del acumulador son afectados por esta suma.

Ejemplos: LDA A,X LDA D,U
LDA B,Y

c).- Autoincremento/decremento de registro. Este modo de direccionamiento trabaja en una manera de postincremento o predecremento. La cuenta de incremento o decremento, una o dos posiciones, es designada en el postbyte.

En el modo de autoincremento, el contenido de la dirección efectiva está contenida en el registro apuntador, designado en el postbyte, y entonces el registro apuntador es automáticamente incrementado; asi, el registro apuntador es posincrementado.

En el modo de autodecremento, el registro apuntador, designado en el postbyte, es automáticamente decrementado -- primero y entonces el contenido de la nueva dirección es usada; así, el registro apuntador es predecrementado.

Ejemplos:	Autoincremento		Autodecremento	
	LDA ,X+	LDY ,X++	LDA ,-X	LDY ,--X
	LDA ,Y+	LDX ,Y++	LDA ,-Y	LDX ,--Y
	LDA ,S+	LDX ,U++	LDA ,-S	LDX ,--U
	LDA ,U+	LDX ,S++	LDA ,-U	LDX ,--S

d).- Indirección. Cuando se usa la indirección, la dirección efectiva del modo de direccionamiento base es usada para traer dos bytes los cuales contienen la dirección efectiva final del operando. Este puede ser usado con todos los modos de direccionamiento de indexado y el modo de direccionamiento de contador de programa relativo.

e).- Extendido indirecto. La dirección efectiva del argumento está localizada en la dirección especificada por los dos bytes siguientes al postbyte. El postbyte es usado para indicar indirección.

Ejemplo: LDA [$F000]

f).- Contador de programa relativo. El contador de programa también puede ser usado como un apuntador con un desplazamiento en complemento 2 constante de 8 o 16 bits. El valor del desplazamiento es sumado al contador de programa para desarrollar una dirección efectiva. Parte del postbyte es usado para indicar si el desplazamiento es de 8 o 16 bits.

2.3.6.- Salto relativo. Este modo de direccionamiento es usado cuando son deseados saltos desde la localidad de la instrucción corriente hasta alguna otra localidad relativa a el contador de programa corriente. Si la prueba de condición de la instrucción de salto es verdadera, entonces la dirección efectiva es calculada (contador de programa más -- desplazamiento en complemento 2) y el salto se hace. Si la - prueba de la condición es falsa, el procesador continua con la próxima instrucción en la línea. Nótese que el contador - de programa está siempre apuntando a la próxima instrucción cuando el desplazamiento es sumado. El modo de direcciona--- miento de salto relativo es siempre usado en programas de posición independiente para todas las transferencias de con--- trol.

Para los saltos cortos, el byte que sigue al código de operación de la instrucción de salto es tratado como desplazamiento en complemento 2 de 8 bits para ser usado para - calcular la dirección efectiva de la próxima instrucción si el salto se hace. Este es llamado un salto relativo corto y el rango está limitado a +127 o -128 bytes desde el código - de operación siguiente.

Para saltos largos, los dos bytes después del código de operación son usados para calcular la dirección efectiva. Este es llamado un salto relativo largo y el rango es +32767 o -32768 bytes desde el código de operación siguiente o el - espacio de direcciones lleno de 64K de memoria que el proce-

sador puede direccionar en un tiempo.

Ejemplos: Salto Corto BRA POLE Salto Largo LBRA CAT

2.4 INTERRUPCIONES SOFTWARE (SWI, SWI2, SWI3)

Las interrupciones software causan que el procesador vaya hasta una secuencia de requerimiento normal de interrupción para guardar el estado de máquina completo aún cuando la fuente interrumpida sea el procesador mismo. Estas interrupciones son comunmente usadas para expulgar programas (eliminar errores en programas) y para llamadas a un sistema operativo.

2.5 INSTRUCCIONES

Las tablas 2.2 a la 2.6 contienen las instrucciones y sus variantes del MC6809 agrupadas dentro de 5 categorias.

Tabla 2.2 Instrucciones de 8 bits de acumulador y memoria.

Instruction	Description
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumulator or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A × B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memroy
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

NOTE: A, B, CC, or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Tabla 2.3 Instrucciones de 16 bits de acumulador y memoria.

Instruction	Description
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U, or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U, or PC
TFR R, D	Transfer X, Y, S, U, or PC to D

NOTE: D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

Tabla 2.4 Instrucciones indica/apuntador de stack.

Instruction	Description
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, X, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U, or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S, or PC from hardware stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U, or PC to D, X, Y, S, U, or PC
ABX	Add B accumulator to X (unsigned)

Tabla 2.5 Instrucciones de salto.

Instruction	Description
SIMPLE BRANCHES	
BEQ, LBEC	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
SIGNED BRANCHES	
BGT, LBGT	Branch if greater (signed)
BVS, LBVS	Branch if invalid twos complement result
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BLE, LBLE	Branch if less than or equal (signed)
BVC, LBVC	Branch if valid twos complement result
BLT, LBLT	Branch if less than (signed)
UNSIGNED BRANCHES	
BHI, LBHI	Branch if higher (unsigned)
BCC, LBCC	Branch if higher or same (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BLS, LBLS	Branch if lower or same (unsigned)
BCS, LBCS	Branch if lower (unsigned)
BLO, LBLO	Branch if lower (unsigned)
OTHER BRANCHES	
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

Tabla 2.6 Instrucciones de miscelanea.

Instruction	Description
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

2.6 PROGRAMACION

El microprocesador permite utilizar modernas técnicas de programación tales como independencia de posición, programación modular y programación reentrante/recursiva. Una breve descripción de estos métodos es dada en los siguientes párrafos.

2.6.1.- Independencia de posición. Se dice que un programa se encuentra en "independencia de posición" si éste correrá correctamente cuando el mismo código de máquina es puesto arbitrariamente en memoria. Tal programa es útil en muchas diferentes configuraciones hardware, y podrá ser copiado de un disco a una RAM cuando el sistema operativo ve primero un requerimiento para usar un sistema de utilidad. Los programas con independencia de posición núnca usan direccionamiento absoluto (extendido o directo): en cambio utilizan inherente, inmediato, registros, indexado y modos relativos. Tampoco deberá de haber saltos (absolutos) o instrucciones de salto a subrutinas ni direcciones absolutas. Un programa con independencia de posición es casi siempre preferido a un programa con posición dependiente (aunque el código de posición independiente es generalmente de 5 a 10% más lento que el código normal).

2.6.2.- Programación modular. Un modulo es un elemento del programa el cual puede ser desconectado fácilmente del resto del programa para ser reusado en un nuevo desarrollo o para ser remplazado. Un modulo es generalmente una subrutina (aunque una subrutina no es necesariamente un modulo); frecuentemente el programador aisla registros que son alterados internamente en el modulo empujando (instrucción PSH) estos registros dentro del stack, y sacándolos (instrucción POP) de él antes del regreso (instrucciones de regreso). Aislando los registros que cambian en el modulo llamado, en ese único modulo, permite al código en el programa llamado ser más fácilmente analizado además de que se puede asumir que todos los registros (excepto aquellos específicamente usados para transferir parámetros) no serán alterados por cada modulo llamado. Esto deja a los registros del procesador libres de cada nivel para cuentas de "loop", comparación de direcciones, etc..

1).- Almacen local. Un método eficaz para ubicar almacen "local" es requerido tanto para programas de posición independiente como para programas modulares. El almacen local o temporal es usado para detener los valores únicamente durante la ejecución de un modulo (o modulos llamados) y son puestos en libertad al regreso. Una forma para ubicar almacen local es decrementar el apuntador del stack hardware con el número de bytes necesarios. Por lo tanto las interrupcio-

nes dejarán esta área intacta y podrá ser re-ubicada a la salida del modulo. Un modulo necesitará casi siempre más almacen temporal que los registros de la unidad micro-procesadora.

2).- Almacen global. Aún en desarrollos modulares existe, quizás, la necesidad de valores "globales" los cuales son accesibles para muchos modulos dentro de un sistema dado. Estos proveen un medio conveniente para almacenar valores de una invocación a otra invocación en una misma rutina. El almacen global puede ser creado como almacen local en un mismo nivel, y usando un registro apuntador (generalmente U) para apuntar a esta área. Este registro será pasado sin cambio en todas las subrutinas, y puede ser usado como índice dentro del área global.

2.6.3.- Reentrada/recursión. Muchos programas eventualmente involucran la ejecución en un desarrollo de manejo de interrupción. Si el manejo de las interrupciones es complejo, ellos podrán llamar la misma rutina la cual será entonces interrumpida. Por lo tanto para proteger los programas presentes contra determinadas obsolencias, todos los programas deberán ser escritos para ser reentrantes. Una rutina reentrante ubica diferente almacen variable local en cada entrada. De este modo, una entrada tardía no destruirá el proceso asociado con una entrada temprana.

La misma técnica que fué realizada para permitir "re entrada" también permite "recursión". Una rutina recursiva - es definida como una rutina que se llama a si misma. Una rutina recursiva podrá ser escrita para simplificar la solu--- ción de determinados tipos de problemas, específicamente a-- quellos que tienen una estructura de datos cuyos elementos - pueden ser ellos mismos una estructura. Por ejemplo, una e-- cuación con paréntesis representa un caso donde la expresión en paréntesis podrá ser considerada como un valor el cual es operado fuera del resto de la ecuación. Un programador podrá optar por escribir una expresión evaluadora pasando por la - expresión entre paréntesis (la cual podrá también contener - expresiones entre paréntesis) en la llamada y recibir el valor regresado por la expresión en paréntesis.

2.7 FACULTADES DEL MC6809

Los siguientes párrafos explican brevemente como el MC6809 es usado con las técnicas de programación mencionadas anteriormente.

2.7.1.- Construcción de modulos. Un modulo puede ser definido como una parte misma lógica y discreta de un programa extenso. Un modulo construido apropiadamente aceptará entradas bien definidas, transporte hasta un juego de procesos,

y producirá una salida determinada. El uso de parámetros, almacen local y global es tratado en los siguientes párrafos. Donde los registros serán usados dentro del modulo (esencialmente en forma de almacen local), la primer cosa que se hace generalmente a la entrada de un modulo es empujarlos (salvarlos) dentro del stack. Esto puede hacerse con una instrucción (por ejemplo; PSHS Y,X,B,A). Después el cuerpo del modulo será ejecutado, los registros salvados son colectados, y un regreso de subrutina será realizado, al mismo tiempo que se saca el contador de programa del stack (por ejemplo; PULS A,B,X,Y,PC).

1).- Parámetros. Los parámetros pueden ser pasados a o desde otros modulos en registros, en caso de que estos (registros) provean almacen suficiente para el paso de parámetros, o en el stack. Si los parámetros son pasados por medio del stack, ellos serán puestos ahí antes del llamado del modulo de nivel inferior. El modulo llamado será entonces escrito para usar almacen local dentro del stack como sea necesario (por ejemplo; ADDA offset,S). Nótese que el desplazamiento (offset) requerido consiste del número de bytes empujados, más dos de la dirección de regreso guardada en el stack, más el dato de desplazamiento al tiempo de la llamada. Este valor puede ser calculado de una forma manual, dibujando la estructura del stack, representando el modulo de entrada y asignando mnemónicos convenientes a estos desplazamien-

tos por medio del ensamblador. Los parámetros regresados son reubicados en la rutina que los envió. Si más parámetros son regresados de los que se enviaron al stack, un espacio para su regreso es ubicado por la rutina llamada antes de la llamada actual (si cuatro bytes adicionales serán regresados, - el llamador ejecutará LEAS -4,S para adquirir almacen adicional).

2).- Almacen local. El espacio para almacen local es adquirido del stack mientras la presente rutina se está ejecutando y se regresa al stack antes de la salida. El acto de empujar los registros que serán usados esencialmente en los cálculos posteriores, salva aquellos registros en almacen local temporal. Almacen local adicional puede ser fácilmente - adquirido del stack, por ejemplo; ejecutando LEAS -2048,S se adquiere un área de buffer que va de 0,S a 2047,S. Un byte - en esta área puede ser accesado directamente por una instrucción que tenga un modo de direccionamiento de índice. Al final de la rutina, el área adquirida para almacen local es regresada (ejemplo; LEAS 2048,S) antes de la última instruc--- ción PULL (sacar del stack). Para programas limpiadores, el almacen local deberá ser ubicado en la entrada del modulo y soltado a la salida del modulo.

3).- Almacen global. El área requerida para almacen global es también más efectivamente adquirida del stack, pro

bablemente por la rutina de más alto nivel en el paquete estándar. Aunque esto es un almacen local en la rutina de más alto nivel, esto viene a ser "global" al posicionar un registro para apuntar a este almacen, (algunas veces referido como una marca del stack) entonces se establece la convección de que todos los modulos pasan por el mismo valor apuntador cuando son llamados los modulos de nivel inferior. En la práctica, esto es conveniente para dejar este registro marca del stack inalterado en todos los modulos, especialmente si los accesos globales son comunes. La rutina de más alto nivel del paquete estándar ejecutará la siguiente secuencia sobre la entrada (para inicializar el área global):

```
PSHS  U      marca de nivel superior.
TFR   S,U    nueva marca del stack.
LEAS  -17,U  ubicación de almacen global.
```

Nótese que el registro U define ahora 17 bytes de ubicación local (permanente) que es global (los cuales son -1,U hasta -17,U) tal como otros globales externos (2,U y precipitados) los cuales han sido pasados por el stack por la rutina que llama al paquete estándar. Cualquier global puede ser accesado por cualquier modulo usando exactamente el mismo valor offset en cualquier nivel (ejemplo; ROL RAT,U; donde RAT EQU -11 ha sido definido). Además, los valores guardados en el stack antes de llamar al paquete estándar pueden incluir apuntadores de datos o de periféricos de entrada-salida.

Cualquier operación de indexado podrá realizarse con indexado indirecto de estos apuntadores, lo cual significa, por ejemplo, que el modulo no necesita conocer nada acerca - de la configuración actual hardware, excepto que (entrada reciente) el apuntador a un registro de entrada/salida haya sido ubicado en una localidad dada del stack.

2.7.2.- Código de posición independiente. Código en posición independiente significa que el mismo código de lenguaje de máquina puede ser ubicado en cualquier parte de memoria y a pesar de eso funcionar correctamente.

El MC6809 tiene un modo de salto largo relativo (desplazamiento de 16 bits) igual a los saltos del MC6800 (instrucciones BRANCH), más un direccionamiento con contador de programa relativo. El direccionamiento con contador de programa relativo usa el contador de programa (PC) como un registro indexable, lo cual permite que todas las instrucciones que referencían memoria también referencien datos relativos al contador de programa. El MC6809 también cuenta con -- instrucciones de cargado de dirección efectiva (LEA, load effective address) que permiten al usuario apuntar a un dato en una ROM (memoria de sólo lectura, read only memory) en una forma de independencia de posición.

Una regla importante para generar código con independencia - en la posición es: NUNCA USAR DIRECCIONAMIENTO ABSOLUTO.

El direccionamiento con contador de programa relativo en el

MC6809 es una forma de direccionamiento de índice que usa el contador de programa como el registro base para una operación de indexado para offset constante.

Sin embargo, el ensamblador del M6809 trata el campo de dirección del PCR (contador de programa relativo) diferente del usado en otras instrucciones de indexado. En direccionamiento con PCR, el tiempo de ensamblado del valor de ubicación es restado del valor (constante) del offset con PCR. La distancia resultante al símbolo deseado es el valor puesto dentro del código objeto de lenguaje de máquina. Durante la ejecución, el procesador suma el valor del tiempo de corrido del PC a la distancia para dar una dirección absoluta en posición independiente.

La forma de direccionamiento de indexado con PCR puede ser utilizada para apuntar a cualquier localidad relativa al programa descuidando la posición en memoria. La forma PCR de direccionamiento indexado permite el acceso a tablas dentro del espacio del programa en una forma de posición independiente usando la instrucción de cargado de dirección efectiva (LEA).

En un programa que esté completamente en independencia de posición, algunas localidades absolutas son comunmente requeridas, particularmente para entradas/salidas. Si las localidades de los dispositivos de entradas/salidas son puestas en el stack (como globales) por una pequeña rutina de organización antes de que el paquete estándar sea llamado, to-

dos los modulos internos podrán ser sus entradas/salidas a - este apuntador (ejemplo; STA (ACIAD,U)), permitiendo si se - desea, que el hardware sea fácilmente cambiado. Unicamente - la simple, pequeña y obvia rutina de organización necesitará ser reescrita para cada configuración hardware diferente.

Los valores globales, permanentes y temporales necesitan ser fácilmente disponibles en la forma de independencia de posición. Usar el stack para este tipo de datos, donde el dato es metido al stack y será directamente accesible. Guardando en el stack la dirección absoluta de los dispositivos de entradas/salidas antes del llamado de cualquier paquete software estándar, el paquete podrá usar las direcciones en el stack para entradas/salidas en cualquier sistema.

Las instrucciones LEA permiten el acceso a tablas, - datos o valores inmediatos en el texto del programa en una - forma de posición independiente como se muestra en el siguiente ejemplo:

```
        LEAX    : MSG1,PCR
        LBSR    . PDATA
MSG1    FCC       /PRINT THIS;/
```

Esto significa que deseamos apuntar un mensaje a ser impreso del cuerpo del programa. Escribiendo "MSG1,PCR" señalamos a el ensamblador que calcule la distancia entre la dirección actual (dirección de la instrucción LBSR) y MSG1. Este resultado es insertado como una constante dentro de la —

instrucción LEA el cual será indexado del valor del contador de programa al tiempo de ejecución, Ahora no importa donde es ubicado el código, donde sea ejecutado el desplazamiento calculado del contador de programa será apuntado el MSG1. Este código tiene independencia en la posición.

Es común utilizar espacios en el stack hardware (SP) para almacen temporal. El espacio es hecho para variables temporales de 0,S hasta TEMP -1,S (donde TEMP es un valor, ejemplo 2000) por un decremento del apuntador del stack igual a la longuitud requerida de almacenado. Usaremos:

LEAS -TEMP,S

No únicamente se simplifica el código en posición independiente sino que está estructurado y ayuda en reentradas y recursiones.

2.7.3.- Programas reentrantes. Un programa que puede ser ejecutado por varios usuarios diferentes compartiendo la misma copia de éste en memoria es llamado reentrante. Esto es importante para sistemas que manejan interrupciones. Este método ahorra un espacio considerable de memoria, especialmente en las rutinas con interrupciones grandes. Los stacks son requeridos en los programas reentrantes, y el M6809 puede soportar cuatro stacks, usando los registros índices X y Y como apuntadores de stack.

Los stacks son mecanismos simples y convenientes para generar programás reentrantes.

Las subrutinas que usan stacks para pasar parámetros y resultados pueden ser construidas para ser reentrantes. -- Los accesos al stack usan el modo de direccionamiento de indexado para rapidez y eficiencia en la ejecución. El direccionamiento de estos stacks es rápido.

El código puro, o codigo que no se modifica a si mismo, es obligatorio para producir código reentrante. Ninguna información interna dentro del código esta sujeta a modificaciones. El código reentrante núnca tiene almacen temporal interno, es fácil de expulgar, puede ser puesto en ROM y podrá ser interrumpible.

2.7.4.- Programas recursivos. Un programa recursivo es aquel que puede llamarse a si mismo. Ellos son muy útiles para analizar mecanismos y determinadas funciones aritméticas tales como computos factoriales. Tal como en la programación reentrante, los stacks son muy útiles para esta técnica.

2.7.5.- Vueltas (loops). La estructuración usual de los loops (por ejemplo; REPEAT ... UNTIL,WHILE...DO,FOR..., etc.) es disponible en lenguaje ensamblador exactamente en la misma forma en que un lenguaje compilador de alto nivel puede trasladar la construcción para ejecución sobre la máquina deseada. Usando un loop FOR...NEXT como un ejemplo, es posible empujar (push) la cuenta del loop, incrementar el va

lor, y el valor final ponerlo en el stack como variables locales a este loop. En cada paso a través del loop, el registro utilizado es salvado, la cuenta del loop actualizada, el incremento sumado, y el resultado comparado con el valor final. Basado en esta comparación, el contador del loop puede ser actualizado, el registro usado recobrado y las variables del loop des-localizadas. Macros medianos pueden hacer la -- forma original para loops triviales, en el lenguaje ensamblador. Tales macros pueden reducir errores resultantes del uso de instrucciones múltiples simplificando el armado de una estructura de control estándar.

2.7.6.- Programación del stack. Muchas aplicaciones del microprocesador requieren almacenar datos como partes -- contiguas de información en memoria. El dato puede ser temporal, esto es, sujeto a cambio o puede ser permanente. Los datos temporales serán comunmente guardados en RAM. Los datos permanentes serán comunmente guardados en ROM.

Esto es importante para permitir tanto al programa - principal como a las subrutinas el acceso a este bloque de - datos, especialmente si los argumentos estan siendo pasados del programa principal a las subrutinas y viceversa.

1).- Operación de los stacks del M6809. Los apuntadores de stack son los marcadores que apuntan a el stack y su contenido interno. Aunque los cuatro registros de índice pue

den ser usados de stack, el S (apuntador de stack hardware) y el U (apuntador de stack de usuario) son generalmente preferidos porque las instrucciones push y pull administran a estos registros. Ambos son registros indexables de 16 bits. El procesador utiliza el registro S automáticamente durante las interrupciones y llamadas de subrutinas. El registro U es libre para cualquier propósito necesario. Este no es afectado por las interrupciones o llamadas a subrutinas llevadas a cabo por la lógica alambrada (hardware).

Un uso del modo de direccionamiento indirecto utiliza los apuntadores de stack para dejar direcciones de datos pasados a una subrutina en el stack como argumentos a una subrutina. La subrutina podrá ahora referenciar los datos con una instrucción. Cada operación push o pull del stack en un programa usa un postbyte el cual especifica que registro o juego de registros serán metidos o sacados del stack. En realidad con el largo número de instrucciones que se usan autoincremento y autodecremento, el M6809 puede competir con una verdadera arquitectura de stack de computadora.

Usando el apuntador de stack S o U, el orden en el cual los registros son metidos o sacados es mostrado en la Fig. 2.4. Nótese que metemos en el stack decrementando las localidades de memoria. El contador de programa es metido primero. Entonces el apuntador de stack decrementado y el otro apuntador de stack (U o S, según sea el caso) es metido en el stack. El decremento y almacenamiento continua hasta que todos los registros indicados por el postbyte son meti—

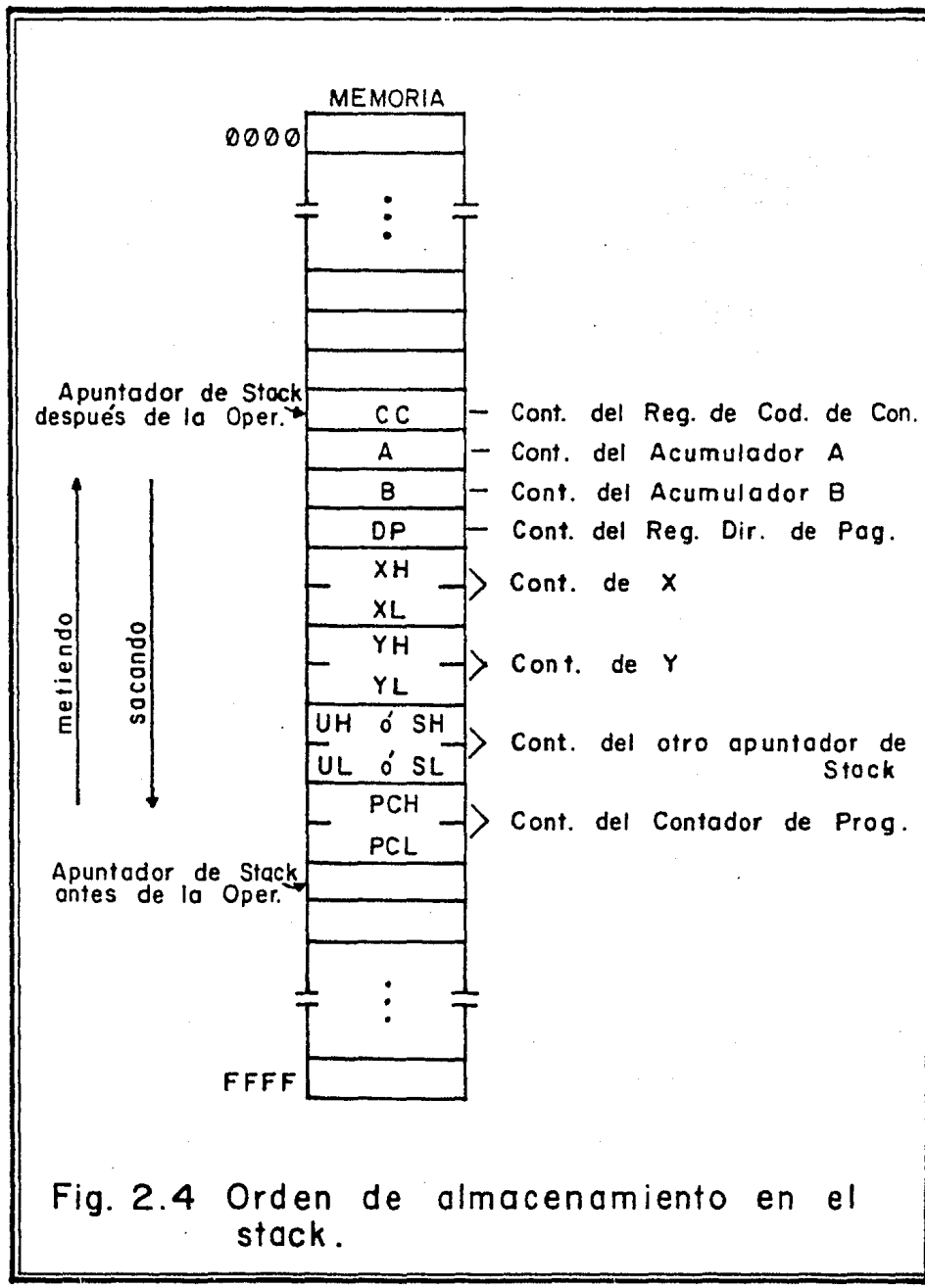


Fig. 2.4 Orden de almacenamiento en el stack.

dos en el stack. El apuntador de stack apuntará a lo alto -- del stack después de la operación push.

El orden de apilamiento es especificado por el procesador. El orden de apilamiento es idéntico al orden utilizado para todas las interrupciones hardware y software. El mismo orden es usado si un subjuego de registros es metido al - stack.

Sin los stacks, los bloques estructurados de lenguajes de alto nivel más modernos serían muy molestos para llevarlos a cabo. El eslabonamiento de subrutinas es muy importante en la generación de lenguajes de alto nivel. En el párrafo 2.6.2 se describe como usar una marca de apuntador de stack para esta importante tare

Una buena programación impone el uso del stack hardware para almacen temporal. Para reservar espacio, decrementaremos el stack por la cantidad de almacen requerido con la instrucción LEAS -TEMP,S . Esta instrucción guarda un espacio para variables temporales de 0,S hasta TEMP - 1,S .

2).- Ligado de subrutinas. En las rutinas de nivel - superior, las variables globales son algunas veces consideradas a ser locales. En consecuencia, el almacen global es colocado en este punto, pero el acceso a esas mismas variables requiere valores de desplazamiento diferentes dependiendo de la profundidad de la subrutina. Porque la profundidad de la subrutina cambia dinamicamente, la longuitud no puede ser co

nocida de antemano. Este problema es solucionado asignando - un apuntador (el registro U será utilizado en la siguiente - descripción, pero X o Y podrán también ser utilizados) para marcar una localidad en el stack hardware usando una instrucción TFR S,U (pasar contenido de S a U). Si el programador - hace esto inmediatamente antes de colocar almacen global, entonces todas las variables serán disponibles con un desplazamiento constante negativo de la localidad de esta marca del stack. Si el stack es marcado después de que las variables - globales son ubicadas, entonces dichas variables serán disponibles con un desplazamiento constante positivo de registro U. El registro U es entonces llamado el apuntador marca del stack. Llamadas del apuntador del stack hardware pueden ser modificadas por interrupciones hardware. Por tal motivo, es fatal utilizar datos referidos mediante un desplazamiento negativo con respecto al apuntador del stack hardware S.

3).- Stacks software. Si más de dos stacks son necesarios, el modo de direccionamiento de autoincremento y autodecremento pueden ser usados para generar apuntadores de --- stack software adicionales.

Los registros índices X, Y y U son absolutamente útiles en loops para propósitos de incrementos y decrementos. - El apuntador es utilizado para buscar tablas y también para mover datos de un área de memoria a otra (mover bloques). Esta característica de autoincremento y autodecremento es dis-

ponible en los modos de direccionamiento de indexado del — M6809 para facilitar tales operaciones.

En autoincremento, el valor contenido por el registro índice (X o Y, U o S) es usado como la dirección efectiva y entonces el registro es incrementado (posincrementado). Y en autodecremento, el registro índice es primero decrementado y entonces usado para obtener la dirección efectiva (— predecrementado). Posincremento o predecremento es siempre - realizado en este modo de direccionamiento. Esto es equivalente en operación al push y pull de un stack. Esta equivalencia permite a los registros X y Y ser usados como apuntadores de stack software. El modo de direccionamiento indexado puede también llevarse a cabo en un nivel extra de posindirección. Esta característica soporta parámetros y operaciones de apuntador.

2.7.7.- Tiempo real de programación. El tiempo real de programación requiere de un cuidado especial. Algunas veces un periférico o una tarea determinada demandan una respuesta inmediata de el procesador, otras veces pueden esperar. Más aplicaciones en tiempo real son demandadas en términos de la respuesta del procesador.

Una solución común es utilizar las capacidades de interrupción del procesador en la solución de problemas de — tiempo real. Las interrupciones significan exactamente que; ellas requieren un rompimiento en la secuencia corriente de

los eventos para solucionar un requerimiento de servicio asíncrono. El diseñador del sistema deberá considerar todas las variaciones de las condiciones que serán encontradas por el sistema incluyendo interacciones software con interrupciones. Como un resultado, los problemas debido al diseño software son más comunes en el código para llevar a cabo las interrupciones para programación en tiempo real que en otras situaciones. Tiempo fuera en software, interrupciones hardware e interrupciones de control de programa son comunmente usadas en la solución de problemas con programación en tiempo real.

2.8 DOCUMENTACION DEL PROGRAMA

El sentido común dictamina que un programa bien documentado es obligatorio. Comunmente es necesario explicar cada grupo de instrucciones desde su uso que no es siempre obvio visto en el código. Los programas ligados y las instrucciones de salto (branch) necesitan ser aclarados totalmente. Considerando los siguientes puntos cuando escribimos comentarios: actualización, precisión, integridad, brevedad y claridad.

La documentación precisa hace que usted y otras personas mantengan y adapten programas modernizados y/o los usen como complementos de otros programas.

A continuación se propone un patrón de documentación

de un programa:

A).- Cada subrutina deberá tener un bloque colector asociado que contenga a lo menos los siguientes elementos:

1). Una especificación total para esta subrutina (incluyendo estructuras de datos asociados) tal que el remplazo de código pueda ser generado partiendo de esta única descripción.

2). Todo uso de memoria posible deberá ser definida, incluyendo:

a) Toda RAM necesaria de almacen temporal (local) usado durante la ejecución de esta subrutina o llamado de subrutinas.

b) Toda RAM necesaria para almacen permanente (usada para transferir valores de una ejecución de una subrutina a futuras ejecuciones).

c) Toda RAM accesada como almacen global (usada para transferir valores de o a subrutinas de nivel superior).

d) Todas las condiciones posibles de estados de salida, si estas van a ser usadas para el llamado de rutinas que prueban los acontecimientos internos de la subrutina.

B).- El código interno para cada subrutina deberá tener suficientes comentarios de la línea asociada para ayudar en el entendimiento del código.

C).- En independencia de posición el código no podrá modificarse a si mismo.

D).- Cada subrutina que incluya un "loop" deberá ser documentada por separada con un diagrama de flujo o un seudo-algoritmo de lenguaje de alto nivel.

E).- Cualquier modulo o subrutina deberá ser ejecutable empezando en la primer localidad y terminando en la última.

CAPITULO 3

PROGRAMA SIMULADOR

3.1 PROGRAMA SIMULADOR DEL MC6809

Definición de programa:

Este programa tiene como finalidad simular la ejecución de un programa escrito en código del microprocesador -- MC6809 empleando una microcomputadora Cromenco con Mp Z-80.

Para simular la ejecución se tendrán dos opciones:

Una en la que se ejecuta el programa simulado desde la dirección de inicio hasta la dirección indicada, pudiéndo sele indicar hasta 3 direcciones de fin, terminando al llegar a la primera de ellas.

La segunda opción consiste en la ejecución a pasos - del programa simulado (instrucción por instrucción) permi---

tiendo así el expulgado del programa.

3.1.1.- Datos de entrada:

a). El código del programa por simular que estará alojado en la memoria, mismo que será apuntado por un registro simulado (contador de programa), que se iniciará automáticamente con un valor que se definirá posteriormente pero que se podrá alterar por el usuario.

Este programa simulará en memoria todos los registros del MC6809, permitiendo al usuario examinarlos y modificarlos en cualquier momento además se podrá examinar y modificar todas las localidades de memoria que se destinen para área de usuario.

b). El nombre del archivo en el disco que contiene el programa, mismo que se indicará al solicitar la ejecución del programa.

c). Un archivo conteniendo el código por ejecutar mismo que se supondrá a partir de la dirección ØØØØH y se alojará en memoria por el programa simulador, mapeándolo en la zona destinada como área del usuario. Además se considerará de una longuitud determinada por el número de caracteres leidos del disco hasta encontrar un byte cuyo ASCII corresponda al "control Z".

d). Durante la ejecución de la simulación se aceptaran de la consola comandos que le indiquen las siguientes funciones :

1). Ejecución del programa completo .
2). Ejecución del programa a pasos .
3). Despliegue de registros .
4). Modificación de registros .
5). Despliegue de memoria .
6). Modificación de memoria .
7). Salida de simulación .
8). Indicación de error del usuario .

mismos que se indicaran según la gramática mostrada en la figura 3.1 . La explicación de cada una de las etiquetas indicadas por esta figura se detallan en la figura 3.2 .

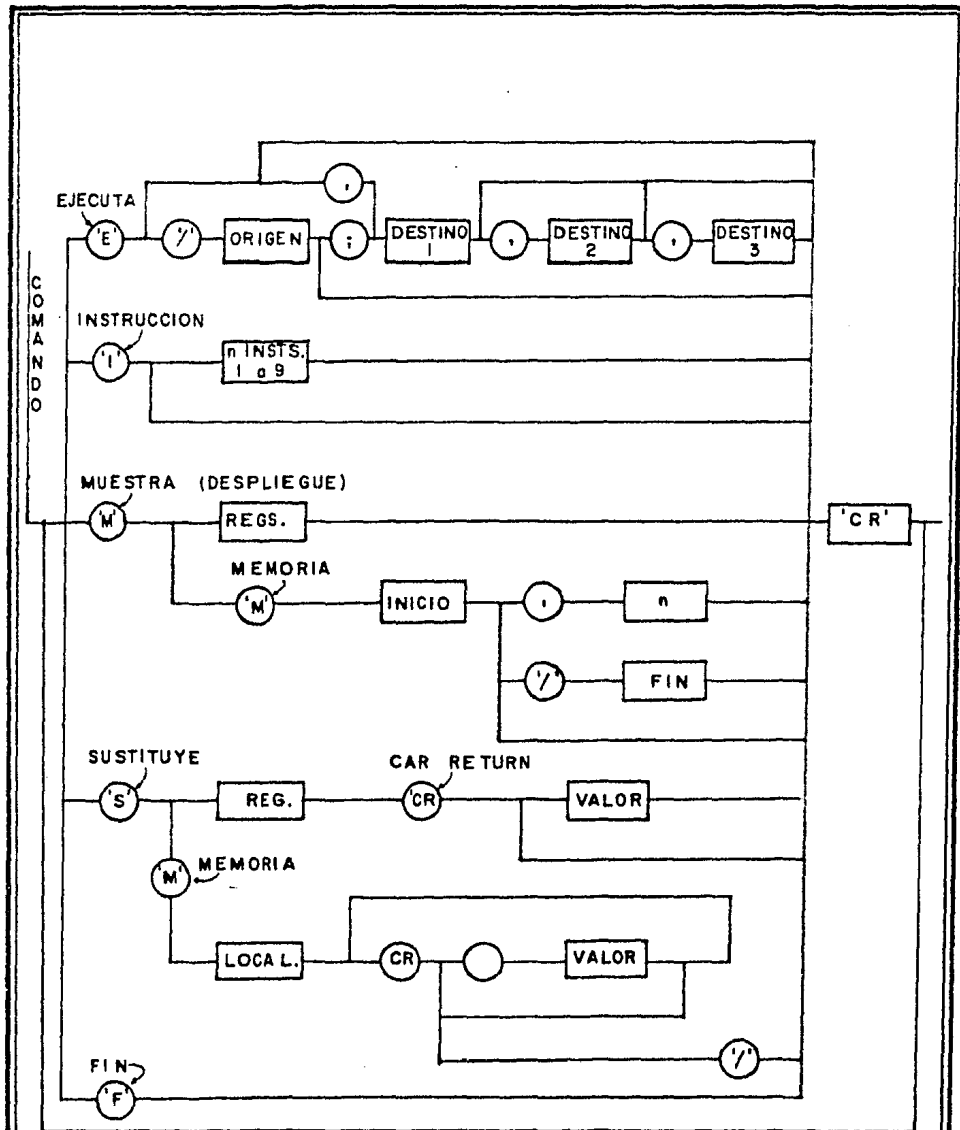


Fig. 3.1 Diagrama de análisis de comandos.

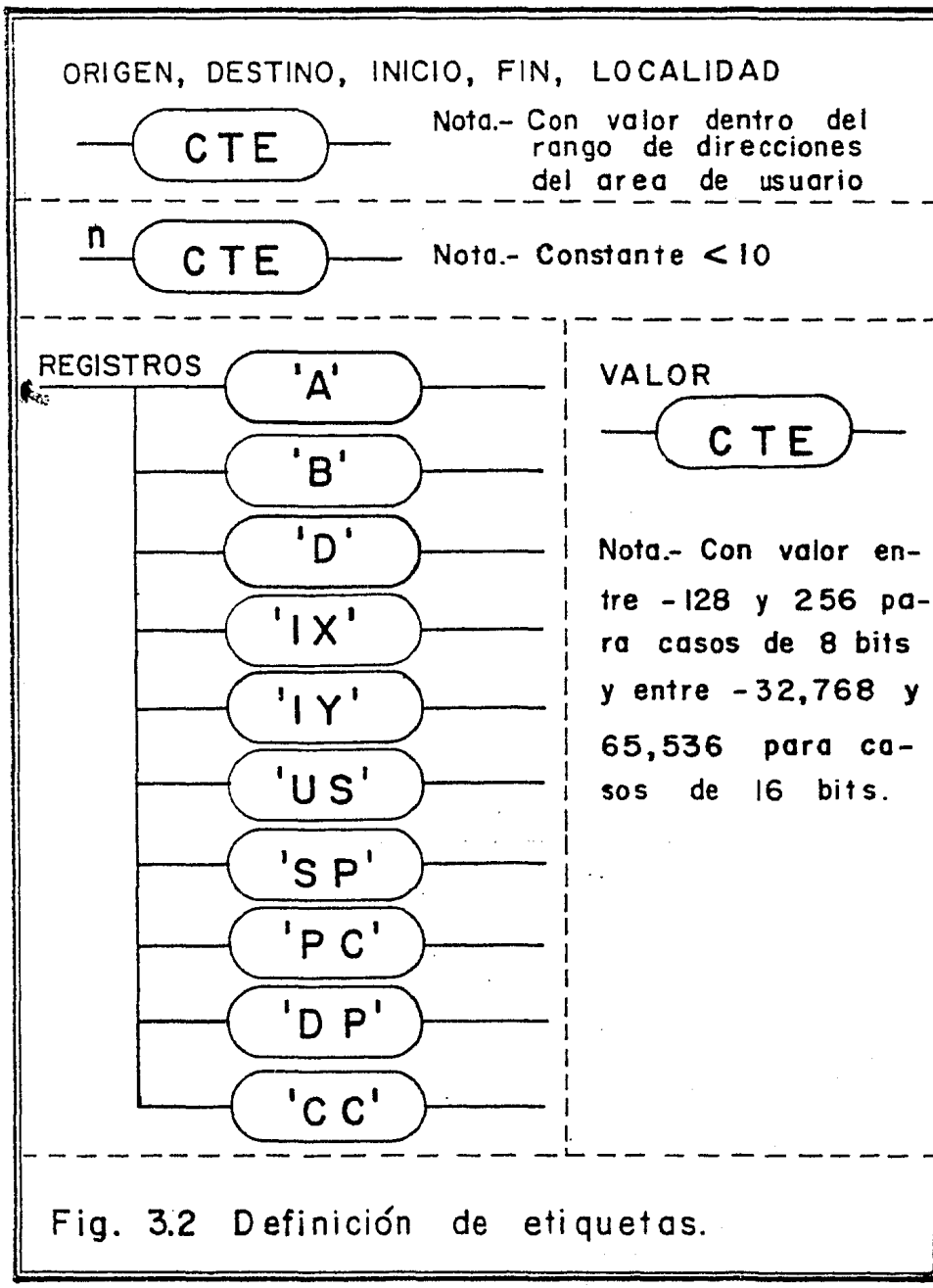


Fig. 3.2 Definición de etiquetas.

3.1.2.- Datos de salida:

1). Afectará los registros y las localidades del área de usuario según se indicará en cada instrucción.

Si se pretende usar una localidad fuera del área de usuario indicará error con el mensaje "fuera de memoria".

2). Igual al caso No. 1, pero además indicará el contenido de todos los registros y banderas después de cada instrucción.

3). Mostrará el contenido del registro solicitado, indicando el nombre y contenido, en el caso de que exista el registro, de lo contrario indicará "registro desconocido".

4). Igual al caso No. 3, pero después de mostrar el contenido esperará el nuevo valor.

5). Mostrará una tabla con tres columnas:

- La dirección.
- El dato en la dirección (cuando se pretenda leer contenidos de localidades no existentes, indicara error con el mensaje ("fuera de memo--- ria").
- El ASCII del contenido.

6). Mostrará la dirección, el valor del dato, el --- ASCII en caso de que éste exista y esperará un nuevo valor.

7). Cederá el control al CDOS (Sistema Operativo de Disco Cromenco).

8). Además indicara error de comando cuando se violen las gráficas, con el mensaje "ERROR EN COMANDO".

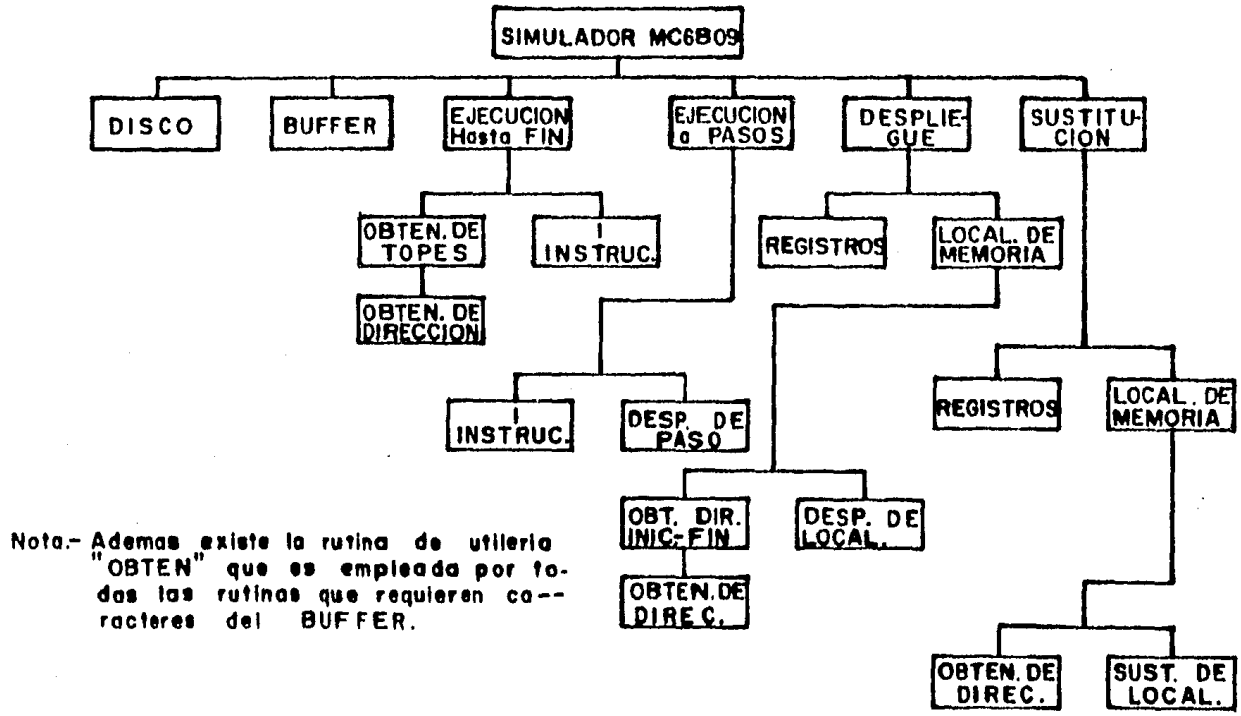
3.2 EXPLICACION DE LA ESTRUCTURA DEL PROGRAMA

Debido a lo extenso y complejo del tema a tratar, el programa simulador del MC6809 fué tratado en tres etapas y además en subrutinas, todo esto con el fin de que el lector vaya adentrándose de una forma simple y sencilla en las características de programación de este procesador.

Al ser dividido el programa en subrutinas el análisis del problema se simplifica ya que cada subrutina se encargara del estudio de problemas con características similares entre ellos mismos, por tal motivo el programa simulador quedó dividido en los grupos de rutinas que se muestran en las figuras 3.3 y 3.4 .

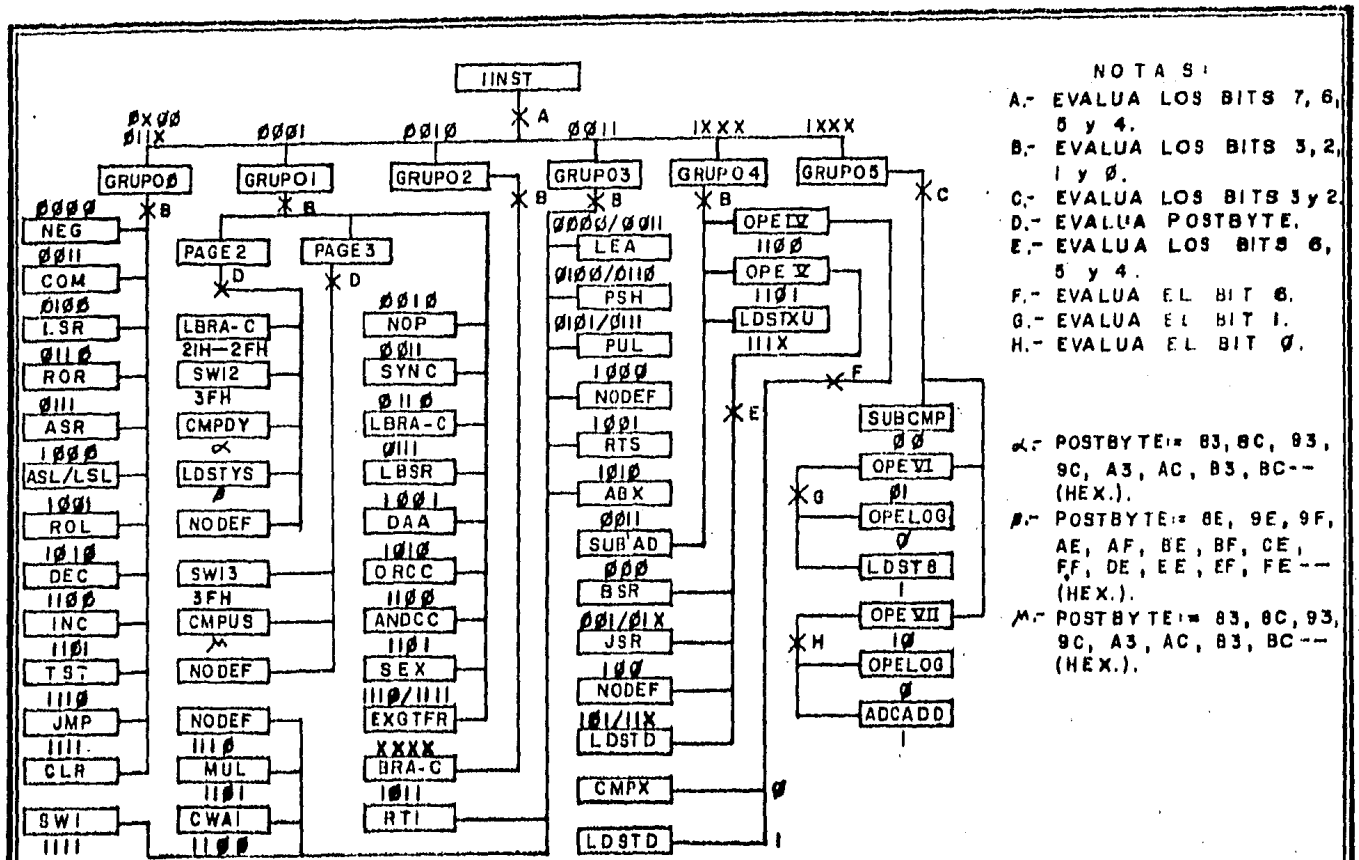
Cada una de las subrutinas consta de las siguientes 3 etapas:

Definición , Pseudo-codigo , Código .



Nota.- Ademas existe la rutina de utileria "OBTEN" que es empleada por todas las rutinas que requieren caracteres del BUFFER.

Fig. 3.3 Estructura del programa simulador del MC6809.



NOTAS:

A.- EVALUA LOS BITS 7, 6, 5 y 4.
B.- EVALUA LOS BITS 3, 2, 1 y Ø.
C.- EVALUA LOS BITS 3 y 2.
D.- EVALUA POSTBYTE.
E.- EVALUA LOS BITS 6, 5 y 4.
F.- EVALUA EL BIT 6.
G.- EVALUA EL BIT 1.
H.- EVALUA EL BIT Ø.

∝.- POSTBYTE= 83, 8C, 93, 9C, A3, AC, B3, BC-- (HEX.).

β.- POSTBYTE= 8E, 9E, 9F, AE, AF, BE, BF, CE, FF, DE, EE, EF, FE-- (HEX.).

M.- POSTBYTE= 83, 8C, 93, 9C, A3, AC, B3, BC-- (HEX.).

Fig. 3.4 Estructura de la rutina IINST.

a).- Definición. En esta etapa se realizó en forma - de enunciado las cualidades y características - de cada subrutina. Esta etapa comprende los siguientes puntos:

1) Nombre de la rutina.
2) Quien la realizó.
3). Cuando (fecha)
4) Cual es el fin de la rutina (objetivo).
5) Qué datos le dan cuando la llaman.
6) Qué datos proporciona cuando termina.
7) Cómo hace su función (procedimiento).
8) Qué datos proporciona a las rutinas que -- llama.
9) Qué datos le proporcionan las rutinas que llama.

b).- Pseudocodigo. Basándose en la etapa de definición para cada subrutina se trató el tema pero en una forma más técnica. Esto es, lo que se indica en la definición se planteó de una forma - más especifica detallando paso a paso la estructura de la subrutina, de tal forma que el lector pueda comprender más facilmente el desarrollo de la etapa de código siguiente.

c).- Código. Comprende la solución del objetivo plan

teado en la etapa de definición. El desarrollo de esta etapa se basa únicamente en la de pseudocodigo ya que está contiene la información necesaria. El código es ya el lenguaje simulador del procesador de MOTOROLA MC6809, llevado a cabo con el procesador de Zilog Z-80.

CAPITULO 4

DEFINICION

```
D E F I N I C I O N :

; PROGRAMA : SIMULADOR [illegible].

; PROGRAMA PRINCIPAL [PPAL].

; AUTOR : [illegible]                    FECHA : [illegible].

; OBJETIVO : ADMINISTRACION DE LAS RUTINAS QUE EJECUTAN -
;          LOS DIFERENTES COMANDOS ASI COMO EL ESTABLECIMIEN-
;          TO DE LAS CONDICIONES INICIALES.

; DATOS DE ENTRADA :
;          - NOMBRE DEL PROGRAMA A EJECUTAR.

; DATOS DE SALIDA :
;          - MENSAJES DE ERROR.

; PROCEDIMIENTO : INICIALIZA LOS REGISTROS SIMULADOS ME -
;          DIANTE LA RUTINA "DISCO" Y EL NOMBRE DEL PROGRAMA
;          LEE DEL DISCO EL PROGRAMA POR EJECUTAR, POSTERIOR-
;          MENTE MEDIANTE LA RUTINA "BUFFER" OBTIENE EL NOM -
;          BRE DEL COMANDO, SI ESTE ESTA DEFINIDO SOLICITA A
;          LA RUTINA CORRESPONDIENTE SU EJECUCION Y FINALMEN-
;          TE SE EJECUTA EL PROGRAMA A PARTIR DE LA OBTENCION
;          DEL COMANDO, SALVO QUE EL COMANDO SOLICITADO SEA -
;          EL DE FIN Y EN TAL CASO SE TERMINA DE EJECUTAR EL
;          PROGRAMA, CEDIENDO EL CONTROL AL "CDOS".

; DATOS DE ENTRADA A LA RUTINA "DISCO". :
;          - NOMBRE DEL ARCHIVO (VARIABLE BUFF).
```

```
; DATOS DE SALIDA A LA RUTINA "DISCO" :
;          - BUFFER DE PROGRAMA (CON CODIGO 6809).
;            (VARIABLE USUARIO).

; DATOS DE ENTRADA A LA RUTINA "BUFFER" :
;          - NINGUNO.

; DATOS DE SALIDA A LA RUTINA "BUFFER" :
;          - BUFFER CON COMANDO (VARIABLE BUFF).

; DATOS DE ENTRADA A LAS RUTINAS "EJECUCION","DESPLIEGUE",
; Y "SUSTITUCION" :
;          - REGISTROS SIMULADOS (VARIABLE:ACCA,ACCB,RIX,...)
;          - AREA DEL USUARIO (VARIABLE USUARIO).
;          - BUFFER (VARIABLE BUFF).

; DATOS DE SALIDA A LAS RUTINAS "EJECUCION","DESPLIEGUE",
; Y "SUSTITUCION" :
;          - REGISTROS SIMULADOS (VARIABLE:ACCA,ACCB,RIX,...)
;          - AREA DEL USUARIO (VARIABLE USUARIO).

; DATOS DE ENTRADA A LA RUTINA "ERROR" :
;          - NINGUNO.

; DATOS DE SALIDA A LA RUTINA "ERROR" :
;          - INDICACION EN EL VIDEO "ERROR EN COMANDO".

; DATOS DE ENTRADA A LA RUTINA "OBTEN" :
;          - BUFFER (VARIABLE BUFF).

; DATOS DE SALIDA A LA RUTINA  "OBTEN" :
;          - CARACTER (EN REGISTRO "A").

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
; DISCO [_DISCO_].

; ELABORO : A.E.H.                    FECHA : 7-V-82.

; OBJETIVO : LEER DEL DISCO EL PROGRAMA POR EJECUTAR.

; DATOS DE ENTRADA :
;       - NOMBRE DEL PROGRAMA.
;       - ARCHIVO EN EL DISCO.

; DATOS DE SALIDA :
;       - BUFFER DEL PROGRAMA CON CODIGO 6809.

; PROCEDIMIENTO.: PARTIENDO DE LA AYUDA QUE NOS DA EL SIS-
;       TEMA OPERATIVO DE DISCO (CDOS), SE BAJA DEL DISCO
;       EL PROGRAMA POR EJECUTAR Y LO LOCALIZA A PARTIR DE
;       LA DIRECCION 0000 DEL AREA DEL USUARIO.

;////////////////////////////////////////////////////////////

; BUFFER [_BUFFER_].

; ELABORO : A.E.H.                    FECHA : 7-V-82.

; OBJETIVO : OBTENER EL NOMBRE DEL COMANDO POR EJECUTAR.

; DATOS DE ENTRADA :
;       - CADENA DE DATOS DE 1 A 20 CARACTERES PROVENIEN-
;       . TES DE LA CONSOLA.

; DATOS DE SALIDA :
;       - BUFFER CON COMANDO.
```

```
; PROCEDIMIENTO : CON AYUDA DEL "CDOS" SE LEE UNA CADENA -
;       DE DATOS DE 1 A 20 CARACTERES PROVENIENTES DE LA -
;       CONSOLA, DANDOSE POR TERMINADA LA CADENA DE DATOS
;       POR UN 'CR' O BIEN CUANDO SE LLEGA A LOS 20 CARAC-
;       TERES.

; DATOS DE ENTRADA A "CDOS" :
;       - NINGUNO.

; DATOS DE SALIDA DE "CDOS" :
;       - CARACTER EN EL REGISTRO "A"

;////////////////////////////////////////////////////////////

; EJECUCION COMPLETA [EJECOM].

; ELABORO : A.E.H.                    FECHA : 11-V-82.

; OBJETIVO : EJECUCION COMPLETA DEL PROGRAMA.

; DATOS DE ENTRADA :
;       - BUFFER.
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS ALTERADOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : CON AYUDA DE LA RUTINA "OBTOP", SE OB —
;       TIENEN LOS TOPES DEL PROGRAMA (INICIO-FIN) Y EN SE
;       GUIDA CON LA AYUDA DE LA RUTINA "IINST" SE EJECU —
```

```
;       PARAR LAS INSTRUCCIONES CONTENIDAS EN ESTA AREA.

; DATOS DE ENTRADA A LA RUTINA "OBTOP" :
;       - BUFFER.

; DATOS DE SALIDA A LA RUTINA "OBTOP" :
;       - DIRECCION DE INICIO.
;       - DIRECCION DE PRIMER TOPE.
;       - DIRECCION DE SEGUNDO TOPE.
;       - DIRECCION DE TERCER TOPE.

; DATOS DE ENTRADA A LA RUTINA "IINST"
;       - DIRECCION DE INICIO.
;       - DIRECCION DE PRIMER TOPE.
;       - DIRECCION DE SEGUNDO TOPE.
;       - DIRECCION DE TERCER TOPE.
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA A LA RUTINA "IINST" :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

;////////////////////////////////////////////////////////////

; OBTENCION DE TOPES [OBTOP].

; ELABORO : A.S.H.                    FECHA : 14-V-82.

; OBJETIVO : OBTENER LAS DIRECCIONES DE INICIO Y FIN.

; DATOS DE ENTRADA :
```

```
;         - BUFFER.

; DATOS DE SALIDA :
;         - BUFFER.
;         - DIRECCION DE INICIO.
;         - DIRECCION DE PRIMER TOPE.
;         - DIRECCION DE SEGUNDO TOPE.
;         - DIRECCION DE TERCER TOPE.

; PROCEDIMIENTO : CON AYUDA DE LA RUTINA "OBTDIR" SE OB —
;         TIENE UNA DIRECCION. ESTA DIRECCION PUEDE SER DE -
;         INICIO O DE FIN.

; DATOS DE ENTRADA A LA RUTINA "ERROR" :
;         - NINGUNO.

; DATOS DE SALIDA A LA RUTINA "ERROR" :
;         - INDICACION EN EL VIDEO("ERROR EN COMANDO").

; DATOS DE ENTRADA A LA RUTINA "OBTEN" :
;         - BUFFER.

; DATOS DE SALIDA A LA RUTINA "OBTEN" :
;         - CARACTER.

; DATOS DE ENTRADA A LA RUTINA "OBTDIR" :
;         - BUFFER.

; DATOS DE SALIDA A LA RUTINA "OBTDIR" :
;         - DIRECCION.

;////////////////////////////////////////////////////////////
```

```
; OBTENCION DE DIRECCION [OBTDIR].

; ELABORO : A.E.M.                    FECHA : 14-V-82.

; OBJETIVO : OBTENER UNA DIRECCION.

; DATOS DE ENTRADA :
;       - BUFFER.

; DATOS DE SALIDA :
;       - DIRECCION.

; PROCEDIMIENTO : ESTA RUTINA OBTENDRA CUATRO CARACTERES
;       (EN ASCII) DEL BUFFER Y A PARTIR DE ELLOS GENERA-
;       RA UN NUMERO QUE SERA LA DIRECCION.

; DATOS DE ENTRADA A LA RUTINA "OBTEN" :
;       - BUFFER.

; DATOS DE SALIDA DE LA RUTINA "OBTEN" :
;       - CARACTER.

;////////////////////////////////////////////////////////////

; UNA INSTRUCCION [INST].

; ELABORO : A.E.M.                    FECHA : 14-V-82.

; OBJETIVO : ADMINISTRAR EL CODIGO DEL PROGRAMA Y VER LA
;       EJECUCION DE LA INSTRUCCION MEDIANTE GRUPOS.

; DATOS DE ENTRADA :
```

```
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : CON EL CONTENIDO DEL CONTADOR DE PROGRA-
;       MA SE OBTIENE EL CODIGO DE LA INSTRUCCION POR EJE-
;       CUTAR, INCREMENTA EL CONTADOR DE PROGRAMA Y FINAL-
;       MENTE CLASIFICA LA INSTRUCCION POR EJECUTAR EN 6
;       GRUPOS DE ACUERDO A LAS SIGUIENTES CARACTERISTICAS
;       DEL CODIGO DE LA INSTRUCCION POR EJECUTAR.
;       - GRUPO 0 : CUANDO LOS BITS 7,6,5,4. TOMEN VALOR -
;                   DE 0000 Y DIEN LOS BITS 7 Y 6 TOMEN EL
;                   VALOR DE 0 Y 1 .
;       - GRUPO 1 : CUANDO LOS BITS 7,6,5,4. TOMEN VALOR -
;                   DE 0001.
;       - GRUPO 2 : CUANDO LOS BITS 7,6,5,4. TOMEN VALOR -
;                   DE 0010.
;       - GRUPO 3 : CUANDO LOS BITS 7,6,5,4, TOMEN VALOR -
;                   DE 0011.
;       - GRUPO 4 : CUANDO LOS BITS 7,3,2. TOMEN VALOR DE
;                   111. O BIEN LOS BITS 7,3,2,1,0. TOMEN
;                   VALOR DE 10011.
;       - GRUPO 5 : CUANDO EL BIT 7 SEA 1 Y NO ESTE INCLU-
;                   IDO EN EL GRUPO 4.

; DATOS DE ENTRADA A LAS RUTINAS "GRUPO 0" HASTA "GRUPO 5":
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - CODIGO DE INSTRUCCION.
```

```
; DATOS DE SALIDA DE LAS RUTINAS "GRUPO 0" HASTA "GRUPO 5":
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.

;;/////////////////////////////////////////////////////////////

; GRUPO O [GRUPOØ].

; ELABORO L.A.E.R.                    FECHA : 15-VI-82.

; OBJETIVO : OBTENER LA DIRECCION DE LA LOCALIDAD DE MEMO-
;          RIA QUE AFECTARA LA INSTRUCCION A EJECUTAR Y VER -
;          LA EJECUCION DE LA INSTRUCCION.

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.
;          - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.

; PROCEDIMIENTO : MEDIANTE EL CODIGO DE LA INSTRUCCION OB-
;          TIENE LA REGLA PARA ADQUIRIR LA DIRECCION POR AFEC
;          TAR Y POSTERIORMENTE CLASIFICA EL CODIGO ENTRE LAS
;          DIFERENTES INSTRUCCIONES. LAS REGLAS PARA OBTENER
;          LA DIRECCION DE LA LOCALIDAD POR AFECTAR SON LAS -
;          SIGUIENTES :
;          DIRECTO : CUANDO LOS BITS 7,6,5,4. TOMEN VALOR DE
;                    0000.
;          ACUMULADOR "A" : CUANDO LOS BITS 7,6,5,4. TOMAN -
```

```
;                   VALOR DE 0100.
;       ACUMULADOR "B" : CUANDO LOS BITS 7,6,5,4. TOMEN -
;                   VALOR DE 0101.
;       INDEXADO : CUANDO LOS BITS 7,6,5,4. TOMEN VALOR DE
;                   0110.
;       EXTENDIDO : CUANDO LOS BITS 7,6,5,4. TOMEN VALOR -
;                   DE 0111.
;       LAS REGLAS PARA CLASIFICAR LA INSTRUCCION DEL GRU-
;       PO 0 DE ACUERDO A LOS VALORES QUE TOMEN LOS BITS 3
;       ,2,1,0. SON LAS SIGUIENTES :
;       BIT  3 2 1 0      MNEMONICO
;       -    0 0 0 0      NEG
;       -    0 0 0 1      + CODIGO NO DEFINIDO.
;       -    0 0 1 0      + CODIGO NO DEFINIDO.
;       -    0 0 1 1      COM
;       -    0 1 0 0      LSR
;       -    0 1 0 1      + CODIGO NO DEFINIDO.
;       -    0 1 1 0      ROR
;       -    0 1 1 1      ASR
;       -    1 0 0 0      ASL (LSL)
;       -    1 0 0 1      ROL
;       -    1 0 1 0      DEC
;       -    1 0 1 1      + CODIGO NO DEFINIDO.
;       -    1 1 0 0      INC
;       -    1 1 0 1      TST
;       -    1 1 1 0      JMP
;       -    1 1 1 1      CLR

; DATOS DE ENTRADA A LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR AFECTAR :
;       - REGISTROS SIMULADOS.
;       - 1 O 2 BYTES DE CODIGO.
```

```
; DATOS DE SALIDA DE LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR AFECTAR :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; DATOS DE ENTRADA A LAS RUTINAS QUE EJECUTAN LA INSTRUC-
; CION :
;       - PC ACTUALIZADO.
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - DIRECCION EFECTIVA.

; DATOS DE SALIDA DE LAS RUTINAS QUE EJECUTAN LA INSTRUC-
; CION :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

;////////////////////////////////////////////////////////////

; DIRECTO [DIRECT].

; ELABORO : A.E.H.                    FECHA : 16-VI-82.

; OBJETIVO : OBTENER LA DIRECCION POR AFECTAR.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
```

```
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE OBTIENE EL BYTE APUNTADO POR EL PC Y
;       DE SEGUIDA SE INCREMENTA EL PC.
;       SE OBTIENE EL CONTENIDO DEL REGISTRO DIRECTO DE -
;       PAGINA Y SE CONCATENA CON EL BYTE OBTENIDO DEL PC.
;       EL REGISTRO DIRECTO DE PAGINA CONTENDRA EL BYTE -
;       MAS SIGNIFICATIVO Y EL BYTE OBTENIDO DEL PC SERA -
;       EL MENOS SIGNIFICATIVO. FORMANDO ASI UNA DIRECCION
;       EFECTIVA DE 16 BITS.
;       ESTO PERMITE EL ACCESO A 256 LOCALIDADES DENTRO DE
;       CUALQUIERA DE LAS 256 PAGINAS.

;////////////////////////////////////////////////////////////

; ACUMULADOR "A" [ACCA].

; ELABORO : A.E.H.                    FECHA : 16-VI-82.

; OBJETIVO : OBTENER EL REGISTRO POR AFECTAR.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL ACUMULADOR "A" SE REFERENCIARA COMO -
;       LA DIRECCION DE UNA DETERMINADA LOCALIDAD DE ME-
;       MORIA.
```

```
;////////////////////////////////////////////////////////////

; ACUMULADOR "B" [_ACCB_].

; ELABORO : A.E.H.                    FECHA : 21-VI-82.

; OBJETIVO : OBTENER EL REGISTRO SIMULADO POR AFECTAR.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL ACUMULADOR "B" SE REFERENCIARA COMO -
;       LA DIRECCION DE UNA DETERMINADA LOCALIDAD DE MEMO-
;       RIA.

;////////////////////////////////////////////////////////////

; INDEXADO [_INDEX_].

; ELABORO : A.E.H.                    FECHA : 30-VI-82.

; OBJETIVO : OBTENER LA DIRECCION POR AFECTAR.

; DATOS DE ENTRADA :
;       - AREA DEL USUARIO.
;       - REGISTROS SIMULADOS.
```

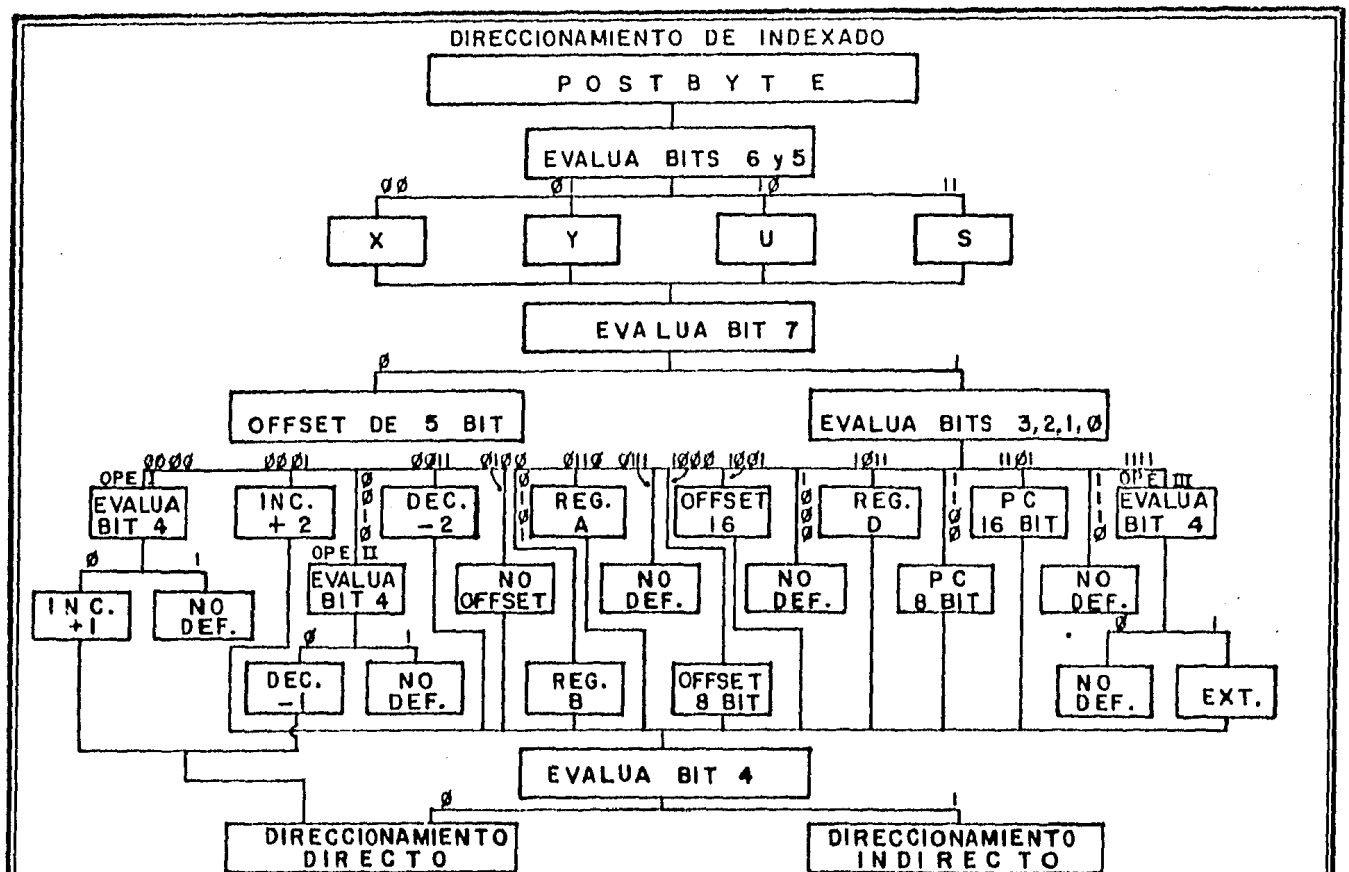


Fig. 4.1 Estructura del direccionamiento de indexado (postbyte).

```
; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE TOMA EL POSTBYTE Y SE INCREMENTA EL -
;       PC .
;       USO DE LOS REGISTROS (X,Y,U O S) Y ALGUNAS VECES -
;       EL PC ES USADO EN EL CALCULO DE LA DIRECCION EFEC-
;       TIVA DEL OPERANDO DE LA INSTRUCCION.
;       EL POSTBYTE NOS INDICARA CON QUE REGISTRO TRABAJA-
;       REMOS, ESTO ES :
;       INDICE X : CUANDO LOS BITS 6 Y 5 TOMEN EL VALOR DE
;                  00.
;       INDICE Y : CUANDO LOS BITS 6 Y 5 TOMEN EL VALOR DE
;                  01.
;       STACK U  : CUANDO LOS BITS 6 Y 5 TOMEN EL VALOR DE
;                  10.
;       STACK S  : CUANDO LOS BITS 6 Y 5 TOMEN EL VALOR DE
;                  11.
;      EN SEGUIDA EL BIT 7 (DEL POSTBYTE) NOS INDICARA :
;      OFFSET DE 5 BITS : CUANDO SU VALOR SEA DE 0.
;      EVALUACION DE LOS BITS 3,2,1 Y 0 . CUANDO EL VALOR
;      DE ESTE BIT 7 SEA 1. EN TAL CASO LAS OPERACIONES SE
;      CLASIFICARAN DE LA SIGUIENTE MANERA:
;       BIT  3 2 1 0     OPERACION
;       -    0 0 0 0     OPERACION I
;       -    0 0 0 1     INC + 2
;       -    0 0 1 0     OPERACION II
;       -    0 0 1 1     DEC - 2
;       -    0 1 0 0     NO OFFSET
;       -    0 1 0 1     REGISTRO "B"
;       -    0 1 1 0     REGISTRO "A"
```

```
;       -   0 1 1 1     + CODIGO NO DEFINIDO.
;       -   1 0 0 0     OFFSET DE 8 BITS
;       -   1 0 0 1     OFFSET DE 16 BITS
;.      -   1 0 1 0     + CODIGO NO DEFINIDO.
;       -   1 0 1 1     REGISTRO "D"
;       -   1 1 0 0     PC 8 BITS
;       -   1 1 0 1     PC 16 BITS
;       -   1 1 1 0     + CODIGO NO DEFINIDO.
;       -   1 1 1 1     OPERACION III
;       PARA EL CASO EN EL QUE LOS BITS 3,2,1,0. NO HAYAN
;       TOMADO LOS VALORES DE 0000, 0010, 1111.(OPERACIONES
;       I,II,III) Y QUE EL CODIGO ESTE DEFINIDO, EL BIT -
;       CUATRO DEL POSTBYTE NOS INDICARA :
;       DIRECCIONAMIENTO NO INDIRECTO : CUANDO SU VALOR -
;       SEA 0.
;       DIRECCIONAMIENTO INDIRECTO : CUANDO SU VALOR SEA 1

; DATOS DE ENTRADA A LAS RUTINAS DE "OPERACION" :
;       - CODIGO DEL POSTBYTE.
;       - VARIABLE REGISTRO.

; DATOS DE SALIDA DE LAS RUTINAS DE "OPERACION" :
;       - REGISTROS CALCULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; OFFSET DE 5 BITS [OFSET5].

; ELABORO : A.E.H.                    FECHA : 20-VII-82.
```

```
; OBJETIVO : REALIZAR UN CORRIMIENTO DE 5 BITS EN UN RE —
;          GISTRO INDICE O EN UN APUNTADOR DE STACK.

; DATOS DE ENTRADA :
;          - CODIGO DEL POSTBITE.
;          - VARIABLE REGISTRO.

; DATOS DE SALIDA :
;          - REGISTROS SIMULADOS.
;          - DIRECCION EFECTIVA.
;          - PC ACTUALIZADO.

; PROCEDIMIENTO : CUANDO EL BIT 7 DEL POSTBYTE TOME EL VA-
;          LOR DE 0, EL CONTENIDO DEL REGISTRO DESIGNADO EN
;          EL POSTBYTE (BIT 6 Y 5) SERA SUMADO A UN VALOR OFF
;          SET DE 5 BITS (BITS 4,3,2,1,0.; CONTENIDOS EN EL -
;          POSTBYTE) DE DOS COMPLEMENTO PARA FORMAR LA DIREC-
;          CION EFECTIVA DEL OPERANDO DE LA INSTRUCCION.

;////////////////////////////////////////////////////////////

; OPERACION I [~OPE1~].

; ELABORO : A.E.H.                    FECHA : 23-VII- 82.

; OBJETIVO : EVALUAR EL BIT 4 DEL POSTBYTE. Y EJECUTAR UNA
;          INSTRUCCION.

; DATOS DE ENTRADA :
;          - CODIGO DEL POSTBYTE.
;          - VARIABLE REGISTRO.
```

```
; DATOS DE SALIDA :
;       - REGISTROS CALCULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE EVALUARA EL BIT 4 DEL POSTBYTE DE LA
;       SIGUIENTE MANERA :
;       - INC + 1 : CUANDO SU VALOR SEA 0(CERO).
;       - + CODIGO NO DEFINIDO : CUANDO SU VALOR SEA 1(UNO)
;       EN ESTA RUTINA EL DIRECCIONAMIENTO SERA NO INMEDIA
;       TO.

;////////////////////////////////////////////////////////////

; OPERACION 11 [_OPE11_].

; ELABORO : A.E.H.                    FECHA : 23-VII-82.

; OBJETIVO : EVALUAR EL BIT 4 DEL POSTBYTE. Y EJECUTAR UNA
;       INSTRUCCION.

; DATOS DE ENTRADA :
;       - CODIGO DEL POSTBYTE.
;       - VARIABLE REGISTRO.

; DATOS DE SALIDA :
;       - REGISTROS CALCULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE EVALUARA EL BIT 4 DEL POSTBYTE DE LA
;       SIGUIENTE MANERA :
```

```
;       - DEC - 1 : CUANDO SU VALOR SEA 0.
;       - + CODIGO NO DEFINIDO : CUANDO SU VALOR SEA 1.
;       EN ESTA RUTINA EL DIRECCIONAMIENTO SERA NO INDI -
;       RECTO.

;////////////////////////////////////////////////////////////

; OPERACION III [ OPEIII ].

; ELABORO : A.S.H.                    FECHA : 23-VII-82.

; OBJETIVO : EVALUAR EL BIT 4 DEL POSTBYTE. Y EJECUTAR UNA
;       INSTRUCCION.

; DATOS DE ENTRADA :
;       - CODIGO DEL POSTBYTE.
;       - PC.

; DATOS DE SALIDA :
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE EVALUARA EL BIT 4 DEL POSTBYTE DE LA
;       SIGUIENTE MANERA :
;       - + CODIGO NO DEFINIDO : CUANDO SU VALOR SEA DE 0.
;       - ENTRADIDO "ENTER" : CUANDO SU VALOR SEA DE 1.

;////////////////////////////////////////////////////////////

; + CODIGO NO DEFINIDO [ NODEF ].
```

```
; ELABORO : A.E.H..                    FECHA : 26-VII-82.

; OBJETIVO : INDICAR QUE EL CODIGO UTILIZADO NO ESTA DEFI-
;       NIDO. Y LA DIRECCION DEL PC.

; DATOS DE ENTRADA :
;       - PC.
;       - TOPE 1, TOPE 2, Y TOPE 3 .

; DATOS DE SALIDA :
;       - INDICACION VISUAL " CODIGO NO DEFINIDO ".
;       - DIRECCION DEL PC.

; PROCEDIMIENTO : SE INDICARA EN LA TERMINAL DE VIDEO QUE
;       EL CODIGO UTILIZADO NO ESTA DEFINIDO. ESTO ES QUE
;       NO EXISTE LA INSTRUCCION. ADEMAS, INDICARA LA DI-
;       RECCION DE ESTE CODIGO.

; +++ ESTA  ES UNA RUTINA DE UTILERIA +++.

; DATOS DE ENTRADA A LA RUTINA QUE CONVIERTE UN VALOR BI-
; NARIO (DEL PC) EN SU HEXADECIMAL "ASCII" :
;       - VALOR BINARIO.

; DATOS DE SALIDA DE LA RUTINA "ASCII" :
;       - VALOR ASCII (HEXADECIMAL).

;////////////////////////////////////////////////////////////

; INCREMENTO AAS DOS [INC + 2].

; ELABORO : A .E.H.                    FECHA : 26-VII-82.
```

```
; OBJETIVO : ENCONTRAR LA DIRECCION EFECTIVA DEL OPERANDO
;          DE LA INSTRUCCION.

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.

; DATOS DE SALIDA :
;          - REGISTROS SIMULADOS.
;          - DIRECCION EFECTIVA.
;          - PC ACTUALIZADO.

; PROCEDIMIENTO : EL CONTENIDO DE UNA DIRECCION EFECTIVA -
;          CONTENIDA EN UN REGISTRO INDICADOR (X,Y,U O S) ES
;          USADO Y A CONTINUACION EL REGISTRO INDICADOR ES INCRE-
;          MENTADO. LA CANTIDAD DE INCREMENTO SERA DE DOS PO-
;          SICIONES (16 BITS).

;////////////////////////////////////////////////////////////////

; DECREMENTO MENOS DOS [-DEC - 2].

; AUTOR : A.E.H.                    FECHA : 27-VII-82.

; OBJETIVO : ENCONTRAR LA DIRECCION EFECTIVA DEL OPERANDO
;          DE INSTRUCCION.

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.

; DATOS DE SALIDA :
;          - REGISTROS SIMULADOS.
;          - DIRECCION EFECTIVA.
```

```
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL REGISTRO INDICADOR (X,Y,U O S) DESIG-
;       NADO EN EL POSTBYTE, SE PREINC AUTOMATICAMENTE DE
;       CREMENTADO DOS POSICIONES Y ENTONCES EL CONTENIDO
;       DE LA NUEVA DIRECCION ES USADO.

;////////////////////////////////////////////////////////////

; NO OFFSET [_SINCOR_].

; ELABORO : A.E.H.                    FECHA : 27-VII-82.

; OBJETIVO : ENCONTRAR LA DIRECCION EFECTIVA DEL OPERANDO
;       DE INSTRUCCION.

; DATOS DE ENTRADA :
;       - VARIABLE REGISTRO.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL CONTENIDO DEL REGISTRO DESIGNADO EN -
;       EL POSTBYTE ES SUMADO A EL VALOR OFFSET EN DOS COM
;       PLEMENTO PARA FORMAR LA DIRECCION EFECTIVA DEL DIR
;       NANDO DE LA INSTRUCCION. EL CONTENIDO DEL REGISTRO
;       DESIGNADO NO ES AFECTADO POR ESTA SUMA. EN ESTE CA
;       SO EL REGISTRO DESIGNADO CONTIENE LA DIRECCION E-
;       FECTIVA.

;////////////////////////////////////////////////////////////
```

```
; REGISTRO "B" [_REGB_].

; ELABORO : A.E.H.                    FECHA : 28-VII-82.

; OBJETIVO : ENCONTRAR LA DIRECCION EFECTIVA DEL OPERANDO
;          DE INSTRUCCION.

; DATOS DE ENTRADA :
;          - VARIABLE REGISTRO.
;          - ACUMULADOR "B".

; DATOS DE SALIDA :
;          - REGISTROS SIMULADOS.
;          - DIRECCION EFECTIVA.
;          - PC ACTUALIZADO.

; PROCEDIMIENTO : EL CONTENIDO DEL REGISTRO INDICE O APUN-
;          TADOR DESIGNADO EN EL POSTBYTE ES TEMPORALMENTE SU
;          MADO A UN VALOR OFFSET EN DOS COMPLEMENTO CONTENI-
;          DO EN EL ACUMULADOR "B" (8 BITS). NI EL REGISTRO -
;          DESIGNADO NI EL ACUMULADOR SON AFECTADOS POR ESTA -
;          RUTA.

;////////////////////////////////////////////////////////////

; REGISTRO "A" [_REGA_].

; ELABORO : A.E.H.                    FECHA : 29-VII-82.

; OBJETIVO : ENCONTRAR LA DIRECCION EFECTIVA DEL OPERANDO
;          DE INSTRUCCION.
```

```
; DATOS DE ENTRADA :
;       - REGISTRO SIMULADO (X,Y,U O S).
;       - ACUMULADOR "A".

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL CONTENIDO DEL REGISTRO INDICE O APUN-
;       TADOR DESIGNADO EN EL POSTBYTE, ES SUMADO TEMPORAL
;       MENTE A UN VALOR OFFSET EN DOS COMPLEMENTO CONTENI
;       DO EN EL ACUMULADOR "A" (8 BITS). NI EL REGISTRO -
;       DESIGNADO NI EL ACUMULADOR SON AFECTADOS POR ESTA
;       SUMA.

;////////////////////////////////////////////////////////////

; OFFSET DE 8 BITS [_OFST8_].

; ELABORO : A.E.H.                    FECHA : 29-VII-82.

; OBJETIVO : ENCONTRAR LA DIRECCION EFECTIVA DEL OPERANDO
;       DE INSTRUCCION.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS (X,Y,U O S).
;       - 1 BYTE QUE INDIQUE EL OFFSET.

; DATOS DE SALIDA  :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
```

```
;       OFFSET DE 16 BITS DEBERA ESTAR CONTENIDO EN DOS BY
;       TES INMEDIATAMENTE SEGUIDOS DEL POSTBYTE.

;///////////////////////////////////////////////////////////////

; REGISTRO "D" [REGD].

; ELABORO : A.E.H.                    FECHA : 29-VII-82.

; OBJETIVO : ENCONTRAR LA DIRECCION EFECTIVA DEL OPERANDO
;       DE INSTRUCCION.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS (X,Y,U O S).
;       - ACUMULADOR "D" (A:B).

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL CONTENIDO DEL REGISTRO INDICE O APUN-
;       TADOR DESIGNADO EN EL POSTBYTE ES TEMPORALMENTE SU
;       MADO A UN VALOR OFFSET EN DOS COMPLEMENTO CONTENI-
;       DO EN EL ACUMULADOR "D" (16BITS). NI EL REGISTRO -
;       DESIGNADO NI EL ACUMULADOR SON AFECTADOS POR ESTA
;       SUMA.

;///////////////////////////////////////////////////////////////

; PC OFFSET DE 8 BITS [PCOF8].
```

```
; ELABORO : A.B.H.                    FECHA : 29-VII-82.

; OBJETIVO : ENCONTRAR LA DIRECCION EFECTIVA DEL OPERANDO
;       DE INSTRUCCION.

; DATOS DE ENTRADA :
;       - PC.
;       - 1 BYTE DE OFFSET.

; DATOS DE SALIDA :
;       - PC ACTUALIZADO.
;       - DIRECCION EFECTIVA.

; PROCEDIMIENTO : EL CONTADOR DE PROGRAMA PUEDE TAMBIEN -
;       SER USADO COMO UN APUNTADOR CON UN OFFSET CONSTAN-
;       TE DE 8 O 16 BITS CON SIGNO. EN ESTE CASO EL VALOR
;       DE 8 BITS DE OFFSET ES SUMADO AL CONTADOR DE PRO -
;       GRAMA PARA DESARROLLAR UNA DIRECCION EFECTIVA. UNA
;       PARTE DEL POSTBYTE ES USADO PARA INDICAR QUE EL —
;       OFFSET(CORRIMIENTO) ES DE 8 BITS. EL VALOR OFFSET
;       DEBERA ESTAR CONTENIDO EN UN BYTE INMEDIATAMENTE -
;       SEGUIDO DEL POSTBYTE.

  ////////////////////////////////////////////////////////////

; PC OFFSET DE 16 BITS [_PCOF16_].

; ELABORO : A.B.H.                    FECHA : 29-VII-82.

; OBJETIVO : ENCONTRAR LA DIRECCION EFECTIVA DEL OPERANDO
;       DE INSTRUCCION.
```

```
; DATOS DE ENTRADA :
;       - PC.
;       - 2 BYTES DE OFFSET.

; DATOS DE SALIDA :
;       - PC ACTUALIZADO.
;       - DIRECCION EFECTIVA.

; PROCEDIMIENTO : LOS 2 BYTES DE OFFSET SON SUMADOS AL CON
;       TADOR DE PROGRAMA PARA DESARROLLAR UNA DIRECCION -
;       EFECTIVA. UNA PARTE DEL POSTBYTE ES USADA PARA IN-
;       DICAR QUE EL OFFSET ES DE 16 BITS.

;////////////////////////////////////////////////////////////

; EXTENDIDO [EXTEN].

; ELABORO : A.E.H.                    FECHA : 2-VIII-82.

; OBJETIVO : OBTENER LA DIRECCION POR AFECTAR.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE OBTIENEN DE LOS SIGUIENTES BYTES APUN
;       TADOS POR EL PC. LOS CUALES NOS INDICARAN LA DIREC
;       CION DE LA LOCALIDAD DE MEMORIA POR AFECTAR.
```

```
;////////////////////////////////////////////////////////////

; OPERACION [NEG].

; ELABORO : A.S.H.                    FECHA : 2-VIII-82.

; OBJETIVO : EJECUTAR UNA INSTRUCCION.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : SE REEMPLAZA EL OPERANDO CON SU DOS COM-
;       PLEMENTO.
;       EL BIT C (CARRY) REPRESENTA UN PEDIDO Y ES PUESTO
;       A LA INVERSA DEL CARRY BINARIO RESULTANTE.
;       M'<--- C <---M
;       LAS BANDERAS SON AFECTADAS DE LA MANERA SIGUIENTE:
;       H - NO DEFINIDO.
;       N - SE PONE SI EL RESULTADO ES NEGATIVO DE OTRA -
;           FORMA QUEDARA LIMPIO.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE OTRA FORMA
;           QUEDARA LIMPIO.
;       V - SE PONE SI EL OPERANDO ORIGINAL FUE OCHO.
;       C - SE PONE SI UN PEDIDO SE GENERARA. DE OTRA FOR-
;           MA QUEDARA LIMPIO.

;////////////////////////////////////////////////////////////
```

```
; COMPLEMENTO [ COM ].

; ELABORO : A.S.M.                    FECHA : 2-VIII-82.

; OBJETIVO : EJECUTAR UNA INSTRUCCION.

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.
;          - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.

; PROCEDIMIENTO : REEMPLAZA EL CONTENIDO DE LA LOCALIDAD -
;          DE MEMORIA M POR SU COMPLEMENTO LOGICO.
;          CUANDO SE OPERA EN VALORES ABSOLUTOS UNICAMENTE -
;          BEQ (BRANCH EQUAL) Y BNE (BRANCH NO EQUAL) PUEDEN
;          SER ESPERADOS PARA TENER PROPIAMENTE EN SEGUIDA -
;          UNA INSTRUCCION COM. CUANDO SE OPERA EN VALORES -
;          DE DOS COMPLEMENTO TODAS LAS INSTRUCCIONES BRANCH
;          CON SIGNO SON DISPONIBLES (BGT,BGE,BLE,BLT,BEQ,BNE)
;          M' <--- 0 + M̄
;          LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE MANERA:
;          H - NO AFECTADA.
;          N - SE PONE SI EL RESULTADO ES NEGATIVO. DE OTRA -
;               MANERA ESTARA LIMPIO.
;          Z - SE PONE SI EL RESULTADO ES CERO. DE OTRA FORMA
;               ESTARA LIMPIO.
;          V - SIEMPRE LIMPIO.
;          C - SIEMPRE SE PONE.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
; CORRIMIENTO LOGICO A LA DERECHA [ LSR ].

; ELABORO : A.E.[illegible]                 FECHA : 2-VIII-82.

; OBJETIVO : EJECUTAR UNA INSTRUCCION.

; DATOS DE ENTRADA :
;         - REGISTROS SIMULADOS.
;         - AREA DEL USUARIO.
;         - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;         - REGISTROS SIMULADOS.
;         - AREA DEL USUARIO.

; PROCEDIMIENTO : SE REALIZA UN CORRIMIENTO LOGICO A LA DE
;         RECHA EN EL CONTENIDO DE LA LOCALIDAD DE MEMORIA -
;         POR AFECTAR.
;         SE CORRE UN CERO DENTRO DEL BIT 7 Y EL BIT 0 ENTRA
;         EN EL BIT C (CARRY).
;         LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE MANERA:
;         N - NO AFECTADA.
;         H - SIEMPRE LIMPIO.
;         Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;             RIO ESTARA LIMPIO.
;         V - NO AFECTADO.
;         C - CARGADO CON EL BIT 0 DEL OPERANDO ORIGINAL.

;//////////////////////////////////////////////////////////


; ROTACION A LA DERECHA [ ROR ].
```

```
; ELABORO : A.E.H.                    FECHA : 2-VIII-82.

; OBJETIVO : EJECUTAR UNA INSTRUCCION.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : SE ROTAN TODOS LOS BITS DEL OPERANDO (EN
;       CALIDAD DE MEMORIA INDICADA) UN LUGAR A LA DERECHA
;       ATRAVES DEL BIT C (CARRY). POR LO TANTO ESTA ES —
;       UNA ROTACION DE 9 BITS.

;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE MANERA:
;       N - NO AFECTADA.
;       Z - SE PONE SI EL RESULTADO ES NEGATIVO. DE LO CON
;           TRARIO ESTARA LIMPIA.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO ESTARA LIMPIA.
;       V - NO AFECTADA.
;       C - CARGADO CON EL BIT 0 DEL OPERANDO PREVIO.

;////////////////////////////////////////////////////////////

; COMPLEMENTO ARITMETICO A LA DERECHA [~A S R~].

; ELABORO : A.E.H.                    FECHA : 3-VIII-82.
```

```
; OBJETIVO : EJECUTAR UNA INSTRUCCION.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; PROCESAMIENTO : SE CORREN TODOS LOS BITS DEL OPERANDO UN
;       LUGAR A LA DERECHA. EL BIT 7 ES CONSERVADO CONTEN
;       TE Y EL BIT CERO SE ALMACENA DENTRO DEL BIT C (CA
;       RRY).
;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE MANERA:
;       H - NO DEFINIDA.
;       N - SE PONE SI EL RESULTADO ES NEGATIVO. DE LO CON
;           TRARIO ESTARA LIMPIA.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO ESTARA LIMPIA.
;       V - NO AFECTADA.
;       C - CARGADO CON EL BIT 0 DEL OPERANDO ORIGINAL.

;/////////////////////////////////////////////////////////

; CORRIMIENTO ARITMETICO A LA IZQUIERDA   [ ASL ].
; CORRIMIENTO LOGICO A LA IZQUIERDA       [ LSL ].

; ELABORO : A.E.H.                  FECHA : 3-VIII-82.

; OBJETIVO : EJECUTAR UNA INSTRUCCION.
```

```
; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : SE CORREN TODOS LOS BITS DEL OPERANDO UN
;       LUGAR A LA IZQUIERDA. EL BIT 0 ES CARGADO CON UN -
;       CERO. EL BIT 7 ES CORRIDO AL BIT C (CARRY).
;       ESTAS INSTRUCCIONES (ASL Y LSL) SON UN DUPLICADO -
;       DE LENGUAJE MAQUINA.
;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;       H - NO DEFINIDA.
;       N - PUESTA SI EL RESULTADO ES NEGATIVO. DE LO CON-
;           TRARIO ESTARA LIMPIA.
;       Z - PUESTA SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO ESTARA LIMPIA.
;       V - CARGADO CON EL RESULTADO DE UNA OPERACION OR-
;           EXCLUSIVA ENTRE LOS BITS 6 Y 7 DEL OPERANDO O-
;           RIGINAL.
;       C - CARGADO CON EL BIT 7 DEL OPERANDO ORIGINAL.

;////////////////////////////////////////////////////////////

; ROTACION A LA IZQUIERDA [ROL].

; ELABORO : A.E.R.                    FECHA : 5-VIII-82.

; OBJETIVO : EJECUTAR UNA INSTRUCCION .
```

```
; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.
;          - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.

; PROCEDIMIENTO : SE ROTAN TODOS LOS BITS DEL OPERANDO UN
;          LUGAR A LA IZQUIERDA A TRAVES DEL BIT C (CARRY). -
;          SE TRATA ENTONCES DE UNA ROTACION DE 9 BITS.
;          LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;          H - NO AFECTADA.
;          N - SE PONE SI EL RESULTADO ES NEGATIVO. DE LO CON-
;               TRARIO ESTARA LIMPIA.
;          Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;               RIO ESTARA LIMPIA.
;          V - CARGADO CON EL RESULTADO DE UNA OR-EXCLUSIVA -
;               ENTRE LOS BITS 6 Y 7 DEL OPERANDO ORIGINAL.
;          C - CARGADO CON EL BIT 7 DEL OPERANDO ORIGINAL.

;///////////////////////////////////////////////////////////


; DECREMENTO [ DEC ].

; ELABORO : A.E.L.                    FECHA : 3-VIII-82.

; OBJETIVO : EJECUTAR UNA INSTRUCCION.

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.
```

```
;          - AREA DEL USUARIO.
;          - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;          - REGISTROS CIRCULANTES.
;          - AREA DEL USUARIO.

; PROCEDIMIENTO : SE SUBSTRAE UNA UNIDAD DEL OPERANDO. EL
;          BIT DE CARRY NO ES AFECTADO, POR LO TANTO ESTA IN-
;          TRUCCION PUEDE SER USADA COMO UN CONTADOR DE LOOP
;          EN CALCULOS DE PRECISION MULTIPLE. CUANDO SE OPERA
;          CON VALORES EN VALOR ABSOLUTO, UNICAMENTE LAS IN-
;          TRUCCIONES BRANCH DE BAE Y BEQ PUEDEN SER ESPERA-
;          DAS A SER CONSISTENTES. CUANDO SE OPERA CON VALO-
;          RES DE DOS COMPLEMENTO, TODAS LAS INSTRUCCIONES -
;          BRANCH "CON SIGNO" SON DISPONIBLES.
;          N'<---- N - 1
;          LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;          H - NO AFECTADA.
;          N - SE PONE SI EL RESULTADO ES NEGATIVO. DE LO CON-
;              TRARIO ESTARA LIMPIA.
;          Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;              RIO ESTARA LIMPIA.
;          V - SE PONE SI EL OPERANDO ORIGINAL FUE 80H. DE LO
;              CONTRARIO ESTARA LIMPIA.
;          C - NO AFECTADA.

;////////////////////////////////////////////////////////////////

; INCREMENTO [INC].

; AUTOR : A.B.H.                    FECHA : 4-VIII-82.
```

```
;  OBJETIVO : EJECUTAR LA INSTRUCCION.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : SUMA UNA UNIDAD AL OPERANDO. EL BIT DE -
;       CARRY NO SE AFECTADO, POR LO CUAL ESTA INSTRUCCION
;       PUEDE SER USADA COMO UN CONTADOR DE LOOP EN COMPU-
;       TOS DE PRECISION MULTIPLE. CUANDO SE ESTA OPERANDO
;       EN VALORES ASSOLUTOS, SOLO BEQ Y BNE (INSTRUCCIONES)
;       PUEDEN SER SUPERADAS A SER CONSISTENTES. CUANDO SE
;       OPERA CON VALORES DE DOS COMPLEMENTO, TODAS LAS IN-
;       STRUCCIONES BRANCH SON DISPONIBLES.
;       E' <---- E + 1
;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;       H - NO AFECTADA.
;       N - SE PONE SI EL RESULTADO ES NEGATIVO. DE LO CON-
;           TRARIO ESTARA LIMPIA.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO ESTARA LIMPIA.
;       V - SE PONE SI EL OPERANDO ORIGINAL FUE 7FH. DE LO
;           CONTRARIO ESTARA LIMPIA.
;       C - NO AFECTADA.

;////////////////////////////////////////////////////////////
```

```
; PRUEBA [TST].

; REALIZO : A.B.R.                    FECHA : 4-VIII-82.

; OBJETIVO : REALIZAR UNA INSTRUCCION.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : SE PONEN LAS BANDERAS A Y Z DE ACUERDO -
;       AL CONTENIDO DE LA LOCALIDAD DE MEMORIA "M" Y SE
;       LIMPIA EL BIT V (OVERFLOW). LA INSTRUCCION "TST"
;       PROVEE UNICAMENTE UN MINIMO DE INFORMACION CUANDO
;       SON PROBADOS VALORES DE VALOR ABSOLUTO. EN VALORES
;       MENORES QUE CERO QUE NO ESTEN EN VALOR ABSOLUTO —
;       DLO Y DL NO TIENEN UTILIDAD. MIENTRAS QUE BNI PUE
;       DE SER USADA DESPUES DE TST, ESTA PROVEE EXACTAMEN-
;       TE EL MISMO CONTROL QUE BNE, LO CUAL ES PREFERIBLE.
;       TODOS LOS BRANCH SON SIEMPRE SON DISPONIBLES.
;       "TEMP"<---- "M" - 0
;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;       A - NO AFECTADA .
;       N - SE PONE SI EL RESULTADO ES NEGATIVO. DE LO CON
;           TRARIO ESTARA LIMPIA.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO ESTARA LIMPIA.
;       V - SIEMPRE LIMPIADA.
;       C - NO AFECTADA.
```

```
;////////////////////////////////////////////////////////////

; SALTO [_JAP_].

; ELABORO : A.E.H.                    FECHA : 4-VIII-82.

; OBJETIVO : EJECUTAR UNA INSTRUCCION.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA  DEL USUARIO.
;       - DIRECCION EFECTIVA POR AFECTAR.
;       - CODIGO DE INSTRUCCION.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : SI LOS BITS 6 Y 5 SON 10, RESPECTIVAMEN-
;       TE. ENTONCES, INDICAR QUE EL CODIGO NO ESTA DEFINI-
;       DO. DE LO CONTRARIO :
;       EL CONTROL DEL PROGRAMA ES TRANSFERIDO A LA  DIREC
;       CION EFECTIVA .
;       PC' <---- EA (DIRECCION EFECTIVA).
;       LAS BANDERAS NO SON AFECTADAS.

;////////////////////////////////////////////////////////////

; LIMPIAR [_CLR_].

; ELABORO : A.E.H.                    FECHA : 4-VIII-82.
```

```
; OBJETIVO : EJECUTAR UNA INSTRUCCION.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - DIRECCION EFECTIVA POR AFECTAR.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

;  PROCEDIMIENTO : LA LOCALIDAD DE MEMORIA INDICADA ES CAR
;       GADA CON CCCC CCCC.
;       "TEMP" <---- "L"
;       "L" <------- CON
;       NOTAR QUE LA DIRECCION EFECTIVA ES LEIDA DURANTE -
;       ESTA OPERACION.
;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;       H - NO AFECTADA.
;       N - SIEMPRE LIMPIADA.
;       Z - SIEMPRE ES PUESTA .
;       V - SIEMPRE LIMPIADA.
;       C - SIEMPRE LIMPIADA.
;
;////////////////////////////////////////////////////////////

; GRUPO UNO (_GRUPO1_).

; ELABORO : A.E.H.                    FECHA : 4-VIII-82.

; OBJETIVO : MEDIANTE DOS PASOS OBTENER LA DIRECCION DE LA
;       LOCALIDAD DE MEMORIA Y EJECUTAR LA INSTRUCCION.
```

```
;  DATOS DE ENTRADA :
;        - REGISTROS SIMULADOS.
;        - AREA DEL USUARIO.
;        - CODIGO DE LA  INSTRUCCION.

; DATOS DE SALIDA :
;        - REGISTROS SIMULADOS.
;        - AREA DEL USUARIO.

; PROCEDIMIENTO : MEDIANTE LOS BITS 3,2,1 Y 0 DEL CODIGO -
;        DE LA INSTRUCCION SE CLASIFICAN LAS SIGUIENTES IN-
;        TRUCCIONES :
;        BIT  3 2 1 0      MNEMONICO
;        -    0 0 0 0      PARE
;        -    0 0 0 1      PARS
;        -    0 0 1 0      NOP
;        -    0 0 1 1      SYNC
;        -    0 1 0 0      + CODIGO NO DEFINIDO.
;        -    0 1 0 1      + CODIGO NO DEFINIDO.
;        -    0 1 1 0      LDA - C
;        -    0 1 1 1      LDA
;        -    1 0 0 0      + CODIGO NO DEFINIDO.
;        -    1 0 0 1      DAA
;        -    1 0 1 0      CMC
;        -    1 0 1 1      + CODIGO NO DEFINIDO.
;        -    1 1 0 0      ADC
;        -    1 1 0 1      SBI
;        -    1 1 1 0      EXITR
;        -    1 1 1 1      EXITR.

; DATOS DE ENTRADA A LAS RUTINAS QUE EJECUTAN LA INSTRUC-
; CION :
;        - REGISTROS SIMULADOS.
```

```
;       - AREA DEL USUARIO.

; DATOS DE SALIDA DE LAS RUTINAS QUE EJECUTAN LA INSTRUC—
; CION :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

;  PAGINA DOS [PAGE2].

; ELABORO : A.B.A.                    FECHA : 9-VIII-82.

; OBJETIVO : VERIFICAR QUE SE EJECUTE UNA  INSTRUCCION DE
;       LA RUTINA "PAGE2".

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE TOMA EL POSTBYTE DEL AREA DEL USUARIO
;       Y SE INCREMENTA EL PC. A CONTINUACION EL POSTBYTE ES -
;       EVALUADO EN LA SIGUIENTE MANERA :
;       - LBRA-C : CUANDO LOS BITS 7,6,5,4 TOMEN VALORES -
;                  DE  0 0 1 0.
;       - S.12   ; CUANDO EL VALOR DEL POSTBYTE SEA 3FH.
```

```
;       - CAPO : CUANDO LOS BITS 7 Y 6 TOMEN VALORES DE -
;                1 Y 0 RESPECTIVAMENTE Y ADEMAS EL BIT 3
;                SEA,IGUAL BIT 2 ,AL BIT 1 NEGADO E IGUAL
;                AL BIT 0 NEGADO.
;       - LARTE :CUANDO EL BIT 7 SEA 1. Y LOS BITS 3,2 Y
;                1 TOMEN LOS VALORES DE 111.
;       - CODIGO NO DEFINIDO : TODOS LOS CODIGOS QUE NO -
;                              QUEDARON DEFINIDOS EN LOS -
;                              CASOS ANTERIORES.

; DATOS DE ENTRADA A LAS RUTINAS QUE EJECUTAN LA INSTRUC-
; CION :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - POSITIB.

; DATOS DE SALIDA DE LAS RUTINAS QUE EJECUTAN LA INSTRUC -
; CION :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; ////////////////////////////////////////////////////////

; SALTO LARGO CONDICIONAL [_LDRA-C_].

; AUTORO : M.E.R.                    FECHA : 11-VIII-82.

; OBJETIVO : REALIZAR UN SALTO EN EL CONTADOR DE PROGRAMA
;       EN CASO DE QUE LA CONDICION BUSCADA SE CUMPLA.

; DATOS DE ENTRADA :
```

```
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - POSTBYTE.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO

; PROCEDIMIENTO : SE OBTIENE EL OPERANDO DEL AREA DEL USUA
;       RIO Y PC = PC + 1. SE GUARDA EL OPERANDO DE LA INS
;       TRUCCION EN UNA LOCALIDAD DE ALMACEN TEMPORAL (16-
;       BITS) Y SE ACTUALIZA EL CONTADOR DE PROGRAMA.
;       "TEMP" <--- "X" : "X" + 1.
;       SI EL CODIGO DEL POSTBYTE ES 10H ESTE ES CAMBIADO
;       A 20H.
;       EN SEGUIDA SE LE ASIGNA A LA VARIABLE CONDICION UN
;       VALOR SEGUN LA SIGUIENTE CLASIFICACION DE BITS ---
;       (POSTBYTE) :
;       BIT   3 2 1     CONDICION
;       -     0 0 0     0
;       -     0 0 1     C OR Z
;       -     0 1 0     C (CARRY)
;       -     0 1 1     Z (CERO)
;       -     1 0 0     V (OVERFLOW)
;       -     1 0 1     N (NEGATIVO)
;       -     1 1 0     N OR EXC V
;       -     1 1 1     (N OR EXC V) OR Z
;       SI EL BIT 0 DEL CODIGO DEL POSTBYTE ES CERO, LA VA
;       RIABLE CONDICION SERA NEGADA.
;       SI LA CONDICION SE CUMPLE (VARIABLE CONDICION=1) -
;       LA LOCALIDAD DE ALMACEN TEMPORAL ES SUMADA AL CON-
;       TADOR DE PROGRAMA, REALIZANDO ASI UN SALTO DE 16 -
```

```
;         BITS EN EL PC.
;         SI LA CONDICION NO SE CUMPLE EL PC NO ES AFECTADO,
;         ESTO ES, EL CONTENIDO DE LA LOCALIDAD DE ALMACEN —
;         TEMPORAL NO SERA SUMADO AL PC Y SE PROCEDERA A EJE
;         CUTAR LA PROXIMA INSTRUCCION.
;         LOS REGISTROS, LOCALIDADES DE MEMORIA Y BANDERAS NO
;         SERAN AFECTADAS.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


; INTERRUPCION SOFTWARE #2 [SWI2].

; ELABORO : A.D.M.                  FECHA : 12-VIII-82.

; OBJETIVO : HACER UNA INTERRUPCION Y SALVAR EL CONTENIDO
;         DE TODOS LOS REGISTROS DEL PROCESADOR.

; DATOS DE ENTRADA :
;         - REGISTROS SIMULADOS.
;         - AREA DEL USUARIO.

; DATOS DE SALIDA :
;         - REGISTRO TEMPORAL.

; PROCEDIMIENTO : SE LE ASIGNA LA DIRECCION DE FFF4 AL RE-
;         GISTRO TEMPORAL :
;         "TEMP" <---- FFF4.
;         INSTRUCCION DISPONIBLE AL USUARIO FINAL Y NO DEBE-
;         RA SER USADA EN PAQUETES SOFTWARE.
;         ESTA INTERRUPCION NO DESHABILITA LAS INTERRUPCIO-
;         NES NORMAL (IRQ) Y RAPIDA (FIRQ). A CONTINUACION -
;         SE PASA A LA RUTINA "SW".
```

```
; DATOS DE ENTRADA A LA RUTINA "SW" :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.
;          - REGISTRO TEMPORAL.

; DATOS DE SALIDA DE LA RUTINA "SW" :
;          - LA BANDERA "I" ES PUESTA.
;          - PC EN LA DIRECCION DEL VECTOR DE INTERRUPCIONES
;            #2 ((FFF4):(FFF5)).

;////////////////////////////////////////////////////////////


; INTERRUPCION SOFTWARE [SWI].

; ELABORO : A.E.H.                    FECHA : 12-VIII-82.

; OBJETIVO : SALVAR EL CONTENIDO DE TODOS LOS REGISTROS  -
;          DEL PROCESADOR Y POR LO TANTO PONER LA BANDERA "E"
;          (ESTADO COMPLETO SALVADO).

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.
;          - REGISTRO TEMPORAL.

; DATOS DE SALIDA :
;          - BANDERAS AFECTADAS (E,I O/Y F)
;          - PC EN LA DIRECCION DEL VECTOR DE INTERRUPCION.

; PROCEDIMIENTO : SE PONE LA BANDERA "E" Y EN SEGUIDA SE
;          ALMACENAN LOS REGISTROS EN EL STACK HARDWARE EN
;          EL SIGUIENTE ORDEN :
```

```
;       SP' <--- SP-1, (SP) <--- PCL
;       SP' <--- SP-1, (SP) <--- PCH
;       SP' <--- SP-1, (SP) <--- USL
;       SP' <--- SP-1, (SP) <--- USH
;       SP' <--- SP-1, (SP) <--- IYL
;       SP' <--- SP-1, (SP) <--- IYH
;       SP' <--- SP-1, (SP) <--- IXL
;       SP' <--- SP-1, (SP) <--- IXH
;       SP' <--- SP-1, (SP) <--- DPR
;       SP' <--- SP-1, (SP) <--- ACCB
;       SP' <--- SP-1, (SP) <--- ACCA
;       SP' <--- SP-1, (SP) <--- CCR
;       EN SEGUIDA SE LE SUMA UN UNO AL BYTE DE DIRECCION
;       (REGISTRO TEMPORAL) Y SE ALMACENA EN EL PC.
;       PCL <--- (REGISTRO TEMPORAL + 1)
;       FORMANDO ASI LA DIRECCION DEL VECTOR DE INTERRUP-
;       CION.
;       PCH : PCL

;///////////////////////////////////////////////////////////////

; COMPARACION CON REGISTRO "D" O "Y" [_CMPDY_].

; ELABORO : A.B.H.                    FECHA : 12-VIII-82.

; OBJETIVO : DECIDIR SI EL REGISTRO POR USAR SERA EL ACU -
;       MULADOR "D" O EL REGISTRO DE INDICE "Y".

; DATOS DE ENTRADA :
;       - POSTBYTE.
;       - REGISTROS SIMULADOS.
```

```
; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS POR UTILIZAR.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL BIT 5 DEL POSTBYTE ES CLASIFICADO DE
;       LA SIGUIENTE MANERA :
;       ACUMULADOR "D" : SI SU VALOR ES 0.
;       INDICE "Y" : SI SU VALOR ES 1.
;       EN SEGUIDA SE PASA A LA RUTINA DE COMPARA EN 16 -
;       BITS "CMP16".

; DATOS DE ENTRADA A LA RUTINA "CMP16" :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - POSTBYTE.

; DATOS DE SALIDA DE LA RUTINA "CMP16" :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; COMPARA EN 16-BITS [CMP16].

; ELABORO : A.E.H.                    FECHA : 12-VIII-82.

; OBJETIVO : VERIFICAR QUE SE EJECUTE UNA INSTRUCCION DE -
;       COMPARACION CON EL MODO DE DIRECCIONAMIENTO INDICA
;       DO POR EL BYTE DE CODIGO.

;  DATOS DE ENTRADA :
```

```
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - 1 BYTE DE CODIGO(POSTBYTE PARA D,X,Y U S ; O CO-
;         DIGO DE INSTRUCCION PARA "X").

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : MEDIANTE EL BYTE PROPORCIONADO (DATO DE
;       ENTRADA) SE OBTIENE LA REGLA PARA SABER QUE DIREC-
;       CION SE UTILIZARA Y POSTERIORMENTE SE HACE LA COM-
;       PARACION ENTRE EL REGISTRO Y LA LOCALIDAD DE MEMO-
;       RIA PROPORCIONADA.
;       LAS REGLAS PARA OBTENER LA DIRECCION DE LA LOCALI-
;       DAD DE MEMORIA POR AFECTAR SON LAS SIGUIENTES :
;       INMEDIATO DE 16 BITS : CUANDO LOS BITS 5 Y 4 TOMEN
;       (IMM16)                VALORES DE 00.
;       DIRECTO : CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE
;       (DIRECT)               01.
;       INDEXADO : CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE
;       (INDEX)                10.
;       EXTENDIDO : CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE
;       (EXTEN.)               11.
;       UNA VEZ OBTENIDA LA LOCALIDAD DE MEMORIA DE 16 BITS
;       POR AFECTAR SE TENDRA LO SIGUIENTE :
;       "TEMP" <--- "R" - "M" : "M" + 1.
;       SE COMPARA EL CONTENIDO DE LAS LOCALIDADES DE MEMO
;       RIA CONCATENADAS (16 BITS) CON EL CONTENIDO DEL RE
;       GISTRO ESPECIFICADO Y LAS BANDERAS SON AFECTADAS -
;       DE LA SIGUIENTE FORMA :
;       H - NO AFECTADA.
```

```
;       N - SE PONE SI EL RESULTADO ES NEGATIVO, DE LO CON
;           TRARIO BORRADA.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO BORRADA .
;       V - SE PONE SI UN SOBRE FLUJO FUE GENERADO. DE LO
;           CONTRARIO BORRADA.
;       C - SE PONE SI UN PRESTAMO FUE GENERADO. DE LO CON-
;           TRARIO BORRADA .
;       EL CONTENIDO DE MEMORIA AL REGISTRO ESPECI -
;       FICADO SON MODIFICADOS, A MENOS QUE SEAN BORRADAS -
;       LAS INSTRUCCIONES DE AUTOINCREMENTO O AUTODECREMEN
;       TO (INDEXADO).

; DATOS DE ENTRADA A LAS RUTINAS QUE EVALUAN  LA DIRECCION
; POR AFECTAR :
;       - REGISTROS SIMULADOS.
;       - 1 O 2 BYTES DE CODIGO.

; DATOS DE SALIDA DE LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR AFECTAR :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA POR AFECTAR.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; DIRECCIONAMIENTO INMEDIATO DE 16 BITS [IMM16].

; ELABORO : A.B.H.                    FECHA : 12-VIII-82.

; OBJETIVO : OBTENER LA DIRECCION POR AFECTAR.
```

```
; DATOS DE ENTRADA :
;       - REGISTROS SALVADOS.

; DATOS DE SALIDA :
;       - REGISTROS SALVADOS.
;       - DIRECCION EFECTIVA (PC).
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL OPERANDO POR UTILIZAR ESTA CONTENIDO
;       EN 2 BYTES (DATO DE ENTRADA) INMEDIATAMENTE SEGUI-
;       DOS DEL CODIGO DE OPERACION.
;       ESTE MODO DE DIRECCIONAMIENTO ES USADO PARA PROVE-
;       ER VALORES DE DATOS QUE NO CAMBIEN DURANTE LA EJE-
;       CUCION DEL PROGRAMA.
;       EL SIGNO #, INDICARA QUE EL DIRECCIONAMIENTO ES —
;       INMEDIATO.

;////////////////////////////////////////////////////////////

; CARGADO O ALMACENADO CON REGISTRO "Y" O "S" [INDEX].

; AUTOR : A.B.B.                    FECHA : 12-VIII-82.

; OBJETIVO : DECIDIR QUE REGISTRO SERA AFECTADO POR LA IN-
;       TRUCCION DE CARGADO O ALMACENADO.

; DATOS DE ENTRADA :
;       - POSTBYTE.
;       - REGISTROS SALVADOS.

; DATOS DE SALIDA :
;       - REGISTROS SALVADOS POR AFECTAR.
```

```
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL BIT 6 DEL POSTBYTE ES CLASIFICADO DE
;       LA SIGUIENTE FORMA  :
;       INDICE "Y" : SI SU VALOR ES 0.
;       ETAQ "U" : SI SU VALOR ES 1.
;       EL SIGUIENTE SE PASA A LA RUTINA DE CARGADO O ALMA -
;       CENADO DE 16 BITS "LDST16".

; DATOS DE ENTRADA A LA RUTINA "LDST16".:
;       - VARIABLE REGISTRO.
;       - AREA DEL USUARIO.
;       - POSTBYTE.

; DATOS DE SALIDA DE LA RUTINA "LDST16" :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; CARGADO O ALMACENADO DE 16 BITS [ LDST16 ].

; ELABORO : A.E.H.                    FECHA : 12-VIII-82.

; OBJETIVO : VERIFICAR QUE SE EJECUTE UNA INSTRUCCION DE
;       CARGADO O ALMACENADO CON EL MODO DE DIRECCIONA --
;       MIENTO INDICADO POR UN BYTE DE CODIGO.

; DATOS DE ENTRADA :
;       - VARIABLE REGISTRO.
;       - AREA DEL USUARIO.
```

```
;       - 1 BYTE DE CODIGO.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : CON EL BYTE DE DATO DE ENTRADA SE OBTIE-
;       NE LA REGLA PARA SABER QUE DIRECCION SE UTILIZARA.
;       LA CLASIFICACION SERA LA SIGUIENTE :
;       INMEDIATO DE 16 BITS "IMM16" :
;       CUANDO LOS BITS 5 Y 4 TOMEN  VALORES DE ØØ.
;       DIRECTO "DIRECT" :
;       CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE Ø1.
;       INDEXADO "INDEX" :
;       CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE 1Ø.
;       EXTENDIDO "EXTEN" :
;       CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE 11.
;       UNA VEZ OBTENIDA LA LOCALIDAD DE MEMORIA DE 16 BITS
;       POR AFECTAR SE TENDRA LO SIGUIENTE :
;       R' <--- "M" : "M" + 1 : SI EL BIT Ø DEL PRIMER BYTE
                                DE CODIGO ES CERO(CARGADO).
;       DE LO CONTRARIO SE INTERCALEIA LA DIRECCION, ESTO
;       ES :
;       "M" : "M" + 1 <--- R (ALMACENADO) Y SERAN EVALUADOS
;                               LOS BITS 5 Y 4, SI TOMAN -
;                               VALORES DE ØØ; SE INDICARA
;                               QUE EL CODIGO NO ESTA DE-
;                               FINIDO Y SE DARA POR TERMI
;                               NADA LA RUTINA.
;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;       H - NO AFECTADA.
```

```
;          N - ES PUESTA SI EL DATO TRANSFERIDO ES NEGATIVO. DE
;               OTRA FORMA BORRADA.
;          Z - ES PUESTA SI EL DATO TRANSFERIDO ES CERO. DE LO
;               CONTRARIO BORRADA.
;          V - SIEMPRE LIMPIADA.
;          C - NO AFECTADA.

; DATOS DE ENTRADA A LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR AFECTAR :
;          - REGISTROS SIMULADOS.
;          - 1 O 2 BYTES DE CODIGO.

; DATOS DE SALIDA DE LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR AFECTAR :
;          - REGISTROS SIMULADOS.
;          - DIRECCION EFECTIVA POR AFECTAR.
;          - PC ACTUALIZADO.

;///////////////////////////////////////////////////////////////


; PAGINA 3 [PAGE3].

; ELABORO : A.S.R.                      FECHA : 16-VIII-82.

; OBJETIVO : VERIFICAR QUE SE EJECUTE UNA INSTRUCCION DE
;          LA RUTINA "PAGE3".

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.

; DATOS DE SALIDA :
```

```
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE TOMA EL POSTBYTE DEL AREA DEL USUARIO
;       Y SE INCREMENTA EL PC.
;       EN SEGUIDA EL POSTBYTE ES EVALUADO DE LA SIGUIENTE
;       FORMA :
;       - SWI3 : CUANDO SU VALOR SEA 3FH.
;       - CAPUS : CUANDO LOS BITS 7 Y 6 TOMEN VALORES DE -
;                 1 Y 0 RESPECTIVAMENTE Y ADEMAS, EL BIT 3
;                 SEA IGUAL AL BIT 2, AL BIT 1 NEGADO E —
;                 IGUAL AL BIT 0 NEGADO.
;       - CODIGO NO DEFINIDO : TODOS LOS CODIGOS QUE NO —
;                 QUEDARON INCLUIDOS EN LOS CASOS ANTERIO-
;                 RES. . .

; DATOS DE ENTRADA A LAS RUTINAS QUE EJECUTAN LA INSTRUC—
; CION :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA DE LAS RUTINAS QUE EJECUTAN LA INSTRUC—
; CION :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

;////////////////////////////////////////////////////////////

; INTERRUPCION SOFTWARE #3 [SWI3].

; ELABORO : A.E.H.                    FECHA : 16-VIII-82.
```

```
; OBJETIVO : HACER UNA INTERRUPCION Y SALVAR EL CONTENIDO
;          DE TODOS LOS REGISTROS DEL PROCESADOR.

; DATOS DE ENTRADA :
;          - REGISTROS CALCULADOS.
;          - AREA DEL USUARIO.

; DATOS DE SALIDA :
;          - REGISTRO TEMPORAL.

; PROCEDIMIENTO : SE LE ASIGNA LA DIRECCION DE FFF2 AL RE-
;          GISTRO TEMPORAL.
;          "TEMP" <--- FFF2
;          ESTA INTERRUPCION NO DESHABILITA LAS INTERRUPCIO-
;          NES NORMAL (IRQ) Y RAPIDA (FIRQ).
;          EL SEGUIDA SE PASA A LA RUTINA "SR".

; DATOS DE ENTRADA A LA RUTINA "SR" :
;          - REGISTROS CALCULADOS.
;          - AREA DEL USUARIO.
;          - REGISTRO TEMPORAL.

; DATOS DE SALIDA DE LA RUTINA "SR" :
;          - LA BANDERA "E" ES PUESTA.
;          - PC EN LA DIRECCION DEL VECTOR DE INTERRUPCION #3
;            ((FFF2) : (FFF3)).

;////////////////////////////////////////////////////////////

; COMPARACION CON EL REGISTRO "U" O "S" [CAPUS_7.

; ELABORO : A.B.M.                    FECHA : 16-VIII-82.
```

```
; OBJETIVO : DECIDIR SI EL REGISTRO POR USAR ES EL STACK -
;          "U" O EL STACK "S".

; DATOS DE ENTRADA :
;          - POSTBYTE.

; DATOS DE SALIDA :
;          - REGISTROS SIMULADOS POR APILAR.
;          - PC ACTUALIZADO.

; PROCEDIMIENTO : EL BIT 6 DEL POSTBYTE ES EVALUADO DE LA
;          SIGUIENTE FORMA :
;          STACK "U" : SI SU VALOR ES 0.
;          STACK "S" : SI SU VALOR ES 1.
;          EN SEGUIDA SE PASA A LA RUTINA DE "CAP16".

; DATOS DE ENTRADA A LA RUTINA "CAP16" :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.
;          - POSTBYTE.

; DATOS DE SALIDA DE LA RUTINA "CAP16" :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.
;          - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; LO OPERACION [NOPC].

; ELABORO : A.E.H.                    FECHA : 10-VIII-82.
```

```
; OBJETIVO : INCREMENTAR EL CONTADOR DE PROGRAMA. O NINGU-
;            NO.

; DATOS DE ENTRADA :
;         - NINGUNO.

; DATOS DE SALIDA :
;         - PC ACTUALIZADO.

; PROCEDIMIENTO : EL PC ES INCREMENTADO AL OBTENER EL CO -
;         DIGO DE INSTRUCCION.
;         PC' <--- PC + 1.
;         NINGUN OTRO REGISTRO, NI LOCALIDAD DE MEMORIA, NI
;         BANDERAS SON AFECTADOS.

;//////////////////////////////////////////////////////////


; SINCRONIZACION PARA EVENTO EXTERNO [SYNC].

; ELABORO : A.E.M.                   FECHA : 18-VIII-88.

; OBJETIVO : PARAR EL PROCESADO DE INSTRUCCIONES Y ESPERAR
;         A QUE SE PIQUE UNA TECLA.

; DATOS DE ENTRADA :
;         - NINGUNO.

; DATOS DE SALIDA :
;         - REGISTROS SIMULADOS.
;         - AREA DEL USUARIO.

; PROCEDIMIENTO : ESTA INSTRUCCION ES USADA PARA PROVEER -
```

```
;       SINCRONIZACION SOFTWARE CON UN PROCESO EXTERNO ---
;       HARDWARE.
;       CUANDO UNA INSTRUCCION "SYNC" ES EJECUTADA, EL PRO-
;       CESADOR ENTRA EN UN ESTADO DE SINCRONIZACION, DESDE
;       DE PROCESAR INSTRUCCIONES Y ESPERA A QUE SE FIJE
;       UNA TECLA. CUANDO UNA TECLA SE FIJADA EL ESTADO DE
;       SINCRONIZACION ES DEJADO Y EL PROCESADO DE INS--
;       TRUCCIONES CONTINUA.
;       LAS BANDERAS NO SON AFECTADAS.

;////////////////////////////////////////////////////////////

; SALTO LARGO A SUBRUTINA [_LBOR_].

; ELABORO : A.B.B.                    FECHA : 17-VIII-88.

; OBJETIVO : REALIZAR UN SALTO DE 16 BITS EN EL CONTADOR -
;       DE PROGRAMA.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : LOS 2 BYTES DE OFFSET SON GUARDADOS EN -
;       UN REGISTRO TEMPORAL (BYTES OBTENIDOS DEL AREA DEL
;       USUARIO).
;       "TEMP" <--- "L" (16 BITS).
```

```
;       EL CONTADOR DE PROGRAMA  ES METIDO EN EL STACK :
;       SP' <--- SP-1, (SP) <--- PCL
;       SP' <--- SP-1, (SP) <--- PCH
;       ENTONCES EL CONTADOR DE PROGRAMA ES CARGADO CON EL
;       OFFSET REQUERIDO :
;       PC' <--- PC + "TEMP"
;       LAS BANDERAS NO SON AFECTADAS.

;//////////////////////////////////////////////////////////////

; AJUSTE DECIMAL EN LA SUMA [DAA].

; ELABORO : A.E.R.                    FECHA : 17-VIII-82.

; OBJETIVO : REALIZAR UN AJUSTE DECIMAL DESPUES DE HABER -
;       REALIZADO UNA INSTRUCCION DE SUMA.

; DATOS DE ENTRADA :
;       - ACUMULADOR "A".
;       - REGISTRO DE CODIGO DE CONDICION.

; DATOS DE SALIDA :
;       - ACUMULADOR "A".
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE LE SUMA UN BYTE DE FACTOR DE CORREC—
;       CION AL ACUMULADOR "A".
;       ACCA' <--- ACCA + CF(MSN) : CF(LSN).
;       DONDE CF ES UN FACTOR DE CORRECCION. EL CF ES DE—
;       TERMINADO POR SEPARADO PARA CADA PEDAZO BCD Y PUE-
;       DE SER 6 Ó 0, ESTO ES :
;       CF(LSN) = 6 SI Y SOLO SI. 1) H=1, O 2) LSN > 9
```

```
;          CF(MSN) = 6 SI Y SOLO SI. 1) C=1. O 2) MSN > 9. O
;                                   3) MSN > 8. Y LSN > 9.
;          LA SECUENCIA DE INSTRUCCION SUMA DE UN SOLO BYTE -
;          EN EL ACUMULADOR "A" SEGUIDA DE UNA INSTRUCCION  -
;          "DAA" RESULTA EN UNA SUMA EN BCD CON UN BIT APRO -
;          PIADO DE CARRY. AMBOS VALORES DEBEN ESTAR EN FOR
;          MA BCD(CADA PEDAZO TAL QUE : 0 ≤ PEDAZO BCD ≤ 9)
;          SUMAS DE PRECISION MULTIPLE, SUMARAN EL CARRY GE—
;          NERADO POR ESTE AJUSTE DECIMAL EN LA SUMA DENTRO -
;          DEL PROXIMO DIGITO MAS ALTO DURANTE LA OPERACION -
;          SUMA (ADCA) INMEDIATAMENTE ANTES DEL PROXIMO "DAA"
;          LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;          H - NO AFECTADA.
;          N - ES PUESTA SI EL RESULTADO ES NEGATIVO. DE LO CON
;                TRARIO BORRADA.
;          Z - ES PUESTA SI EL RESULTADO ES CERO. BORRADA DE LO
;                CONTRARIO.
;          V - NO DEFINIDA.
;          C - ES PUESTA SI UN CARRY ES GENERADO O SI EL BIT DE
;                CARRY FUE PUESTO ANTES DE LA OPERACION. BORRA-
;                DA DE LO CONTRARIO.

;/////////////////////////////////////////////////////////////


; OPERACION OR LOGICA ENTRE EL DATO INMEDIATO Y EL "CCR"
; [ ORCC ].

; ELABORO : A.S.R.                     FECHA : 8-X-82.

; OBJETIVO : AFECTAR EL REGISTRO DE CODIGO DE CONDICION  -
;           (BANDERAS ; E,F,H,I,N,Z,V,C ).
```

```
; DATOS DE ENTRADA :
;       - REGISTRO DE CODIGO DE CONDICION.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA :
;       - REGISTRO DE CODIGO DE CONDICION.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE OBTIENE DEL AREA DEL USUARIO UN BYTE
;       DE CODIGO Y SE INCREMENTA EL PC.
;       SE REALIZA UNA OPERACION OR LOGICA ENTRE EL CONTE-
;       NIDO DEL REGISTRO DE CODIGO DE CONDICION (CCR) Y -
;       EL BYTE.
;       CCR' <--- CCR OR BYTE.
;       ESTA INSTRUCCION PUEDE SER USADA PARA PODER INTE -
;       RRUPCIONES ENMASCARABLES (DESHABILITAR INTERRUPCIO-
;       NES) O CUALQUIER OTRO BIT O BITS.
;       LAS BANDERAS SON AFECTADAS DE ACUERDO AL RESULTADO
;       DE LA OPERACION OR.

;////////////////////////////////////////////////////////////

; OPERACION AND LOGICA ENTRE EL DATO INMEDIATO Y EL "CCR"
; [ANDCC_].

; AUTOR : A.B.A.                    FECHA : 8-4-82.

; OBJETIVO : AFECTAR EL REGISTRO DE CODIGO DE CONDICION -
;       (BANDERAS ; E,F,H,I,N,Z,V,C).

; DATOS DE ENTRADA :
;       - REGISTRO DE CODIGO DE CONDICION.
```

```
;       - AREA DEL USUARIO.

; DATOS DE SALIDA :
;       - REGISTRO DE CODIGO DE CONDICION.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE OBTIENE DEL AREA DEL USUARIO (PC) UN
;       BYTE DE CODIGO Y SE INCREMENTA EL PC.
;       EN SEGUIDA SE REALIZA UNA OPERACION AND LOGICA EN-
;       TRE EL CONTENIDO DEL REGISTRO DE CODIGO DE CONDI -
;       CION (CCR) Y EL BYTE.
;       CCR' <--- CCR AND BYTE.
;       ESTA INSTRUCCION PUEDE SER USADA PARA PONER LAS CON-
;       DICIONES ENMASCARADAS (DESHABILITAR INTERRUPCIO—
;       NES) O CUALQUIER OTRO BIT O BITS.
;       LAS BANDERAS SON AFECTADAS DE ACUERDO AL RESULTADO
;       DE LA OPERACION AND.

;////////////////////////////////////////////////////////////////

; EXTENSION DE SIGNO [ SSX ].

; ELABORO : A.E.H.                    FECHA : 20-VIII-82.

; OBJETIVO : TRANSFORMAR UN REGISTRO DE 8 BITS EN UN REGIS-
;       TRO DE 16 BITS.

; DATOS DE ENTRADA :
;       - REGISTROS SIMBOLICOS.

; DATOS DE SALIDA :
;       - ACUMULADOR "D".
```

```
;       - REGISTRO DE CODIGO DE CONDICION.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SI EL BIT 7 DEL ACUMULADOR "B" ESTA ----
;       PUESTO. ENTONCES ; ACCA' <--- FFH.
;       DE LO CONTRARIO  ; ACCA' <--- 00H.
;       ESTA INSTRUCCION TRANSFORMA UN VALOR DE 8 BITS EN
;       SU COMPLEMENTO DEL ACUMULADOR "B" SI UN VALOR DE
;       16 BITS EN SU COMPLEMENTO EN EL ACUMULADOR "D".
;       LAS BANDERAS SON AFECTADAS COMO SIGUE :
;       X - NO AFECTADA.
;       N - SE PONE SI EL RESULTADO ES NEGATIVO. BORRADA -
;           DE OTRA FORMA.
;       Z - SE PONE SI EL RESULTADO ES CERO. BORRADA DE --
;           OTRA FORMA.
;       V - NO AFECTADA.
;       C - NO AFECTADA.

;////////////////////////////////////////////////////////////

; INTERCAMBIO O TRANSFERENCIA DE REGISTROS [_EXGTFR_].

; ELABORO : A.E.H.                    FECHA : 20-VIII-82.

; OBJETIVO : INTERCAMBIAR O TRANSFERIR UN DATO ENTRE DOS
;       REGISTROS DESIGNADOS.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - CODIGO DE INSTRUCCION.

; DATOS DE SALIDA :
```

```
;       - REGISTROS SIMULADOS.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE OBTIENE EL POSTBYTE Y SE INCREMENTA -
;       EL PC. LOS BITS 7,6,5, Y 4 DEL POSTBYTE DEFINEN UN
;       REGISTRO, MIENTRAS QUE LOS BITS 3,2,1,Y Ø DEFINEN-
;       A OTRO REGISTRO TAL COMO SIGUE :
;       - Ø Ø Ø Ø = A : B
;       - Ø Ø Ø 1 = X
;       - Ø Ø 1 Ø = Y
;       - Ø Ø 1 1 = US
;       - Ø 1 Ø Ø = SP
;       - Ø 1 Ø 1 = PC
;       - Ø 1 1 Ø = + CODIGO NO DEFINIDO
;       - Ø 1 1 1 = + CODIGO NO DEFINIDO
;       - 1 Ø Ø Ø = A
;       - 1 Ø Ø 1 = B
;       - 1 Ø 1 Ø = CCR
;       - 1 Ø 1 1 = DPR
;       - 1 1 Ø Ø = + CODIGO NO DEFINIDO
;       - 1 1 Ø 1 = + CODIGO NO DEFINIDO
;       - 1 1 1 Ø = + CODIGO NO DEFINIDO
;       - 1 1 1 1 = + CODIGO NO DEFINIDO
;       UNICAMENTE ESTA PERMITIDO TRANSFERIR O INTERCAMBIAR
;       REGISTROS DE UN MISMO TAMAÑO (8 BITS CON 8 BITS O-
;       16 BITS CON 16 BITS).
;       UNA VEZ OBTENIDOS LOS REGISTROS POR UTILIZAR, EL-
;       BIT Ø DEL CODIGO DE LA INSTRUCCION NOS INDICARA :
;       INTERCAMBIO (R1 <--- R2) : SI SU VALOR ES Ø.
;       TRANSFERENCIA (R1 <--- R2) : SI SU VALOR ES 1.
;       LAS BANDERAS NO SON AFECTADAS, A MENOS QUE EL "CCR"
;       HALLA SIDO CARGADO CON EL CONTENIDO DE OTRO REGIS-
;       TRO.
```

```
;////////////////////////////////////////////////////////////

; GRUPO DOS [GRUPO2].

; REALIZADO : A.E.N.                    FECHA : 20-VIII-82.

; OBJETIVO : REALIZAR UN SALTO DE 8 BITS EN EL CONTADOR DE
;          PROGRAMA, EN CASO DE QUE LA CONDICION BUSCADA SE -
;          CUMPLA.

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.
;          - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.
;          - PC ACTUALIZADO.

; PROCEDIMIENTO : SE OBTIENE EL OPERANDO APUNTADO POR EL -
;          PC. SE GUARDA EL OPERANDO DE LA INSTRUCCION (1BYTE
;          DE OFFSET) EN UNA LOCALIDAD DE ALMACEN TEMPORAL Y
;          SE ACTUALIZA EL CONTADOR DE PROGRAMA :
;          "TEMP" <--- "L"
;          EN SEGUIDA SE LE ASIGNA A LA VARIABLE-CONDICION UN
;          VALOR SEGUN LA SIGUIENTE CLASIFICACION DE LOS BITS
;          DEL CODIGO DE LA INSTRUCCION :
;          BIT   3 4 5        CONDICION
;          -     0 0 0        0 (CERO)
;          -     0 0 1        C OR Z
;          -     0 1 0        C (CARRY)
```

```
;       -       0 1 1       Z(CERO)
;       -       1 0 0       V(OVERFLOW)
;       -       1 0 1       N(NEGATIVO)
;       -       1 1 0       A XOR V
;       -       1 1 1       (A XOR V) OR Z
;       SI EL BIT 0 DEL CODIGO DE INSTRUCCION ES CERO, LA
;       VARIABLE CONDICION SERA FALSA.
;       SI LA CONDICION SE CUMPLE (VARIABLE CONDICION = 1)
;       EL CONTENIDO DE LA LOCALIDAD DE ALMACEN TEMPORAL -
;       ES SUMADO AL CONTADOR DE PROGRAMA, REALIZANDO ASI
;       UN SALTO DE 8 BITS EN EL PC.
;       PC' <--- PC + "TEMP"
;       SI LA CONDICION NO SE CUMPLE, EL PC NO SE AFECTARA
;       ESTO ES, EL CONTENIDO DE LA LOCALIDAD DE ALMACEN -
;       TEMPORAL NO SERA SUMADA AL PC Y SE PROCEDERA A EJE-
;       CUTAR LA PROXIMA INSTRUCCION.
;       LOS REGISTROS, LOCALIDADES DE MEMORIA Y BANDERAS -
;       NO SERAN AFECTADAS.

;////////////////////////////////////////////////////////////

; GRUPO TRES [GRUPO3].

; ELABORO : A.B.H.                    FECHA : 20-VIII-82.

; OBJETIVO : EJECUTAR UNA INSTRUCCION DEL GRUPO TRES.

; DATOS DE ENTRADA :
;       - REGISTROS CARGADOS.
;       - CODIGO DE LA INSTRUCCION.
;       - ORDEN DEL USUARIO.
```

```
; DATOS DE SALIDA :
;       - REGISTROS ESTABLECIDOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : EL CODIGO DE LA INSTRUCCION NOS INDICARA
;       LA INSTRUCCION POR EJECUTAR, SEGUN LA SIGUIENTE —
;       CLASIFICACION :
;       BIT     3 2 1 0     INSTRUCCION
;       -       0 0 0 0     LEA
;       -       0 0 0 1     LBA
;       -       0 0 1 0     LEX
;       -       0 0 1 1     LEA
;       -       0 1 0 0     PEA
;       -       0 1 0 1     PEL
;       -       0 1 1 0     PEX
;       -       0 1 1 1     PEL
;       -       1 0 0 0     + CODIGO NO DEFINIDO
;       -       1 0 0 1     ATS
;       -       1 0 1 0     ESX
;       -       1 0 1 1     API
;       -       1 1 0 0     CMAI
;       -       1 1 0 1     EOL
;       -       1 1 1 0     + CODIGO NO DEFINIDO
;       -       1 1 1 1     SMI

; DATOS DE ENTRADA A LAS RUTINAS QUE EJECUTAN LA INSTRUC -
; CION :
;       - REGISTROS ESTABLECIDOS.
;       - AREA DEL USUARIO.
;       - CODIGO DE INSTRUCCION.

; DATOS DE SALIDA DE LAS RUTINAS QUE EJECUTAN LA INSTRUC -
; CION :
```

```
;           - REGISTROS SIMULADOS.
;           - AREA DEL USUARIO.
;           - PC ACTUALIZADO.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; CARGADO DE DIRECCION EFECTIVA [ LEA ].

; ELABORO : A.E.R.                       FECHA : 23-VIII-82.

; OBJETIVO : FORMAR UN APUNTADOR Y CARGARLO DENTRO DE CUAL
;            QUIER REGISTRO INDICE O STACK POINTER.

; DATOS DE ENTRADA :
;           - REGISTROS SIMULADOS (X,Y,U, O  S).
;           - AREA DEL USUARIO.
;           - CODIGO DE INSTRUCCION.

; DATOS DE SALIDA :
;           - REGISTROS SIMULADOS.
;           - AREA DEL USUARIO.
;           - PC ACTUALIZADO.

; PROCEDIMIENTO : SE CLASIFICA EL CODIGO DE LA INSTRUCCION
;           PARA DECIDIR EN QUE REGISTRO SERA CARGADO EL APUN-
;           TADOR EVALUANDO LOS BITS 1 Y 0. Y PC:=PC + 1 :
;           INDICE X : SI SUS VALORES SON 00.
;           INDICE Y : SI SUS VALORES SON 01.
;           STACK S  : SI SUS VALORES SON 10.
;           STACK U  : SI SUS VALORES SON 11.
;           DESPUES DE HABER DECIDIDO EN QUE REGISTRO SE VA A
;           CARGAR LA DIRECCION DE LA LOCALIDAD DE MEMORIA. -
```

```
;       SE LLAMA A LA RUTINA DE LECTURA "INDEX" .
;       LA RUTINA "INDEX" RECIBIRA LA DIRECCION DE LA LOCA-
;       LIDAD DE MEMORIA O REGISTRO POR UTILIZAR.
;       LA DIRECCION EFECTIVA SE CARGARA EN EL REGISTRO :
;       X' ---- "DE"
;       SI, SE CALCULA LA DIRECCION EFECTIVA DEL MODO -
;       DE DIRECCIONAMIENTO DE INDICE Y SE COLOCA LA DIREC
;       CION EN UN REGISTRO INTERNO (X,Y, U O S).
;       DEBIDO AL CASO EN EL CUAL LAS DIRECCIONES EFECTI-
;       VAS SON CALCULADAS INTERNAMENTE, LAS INSTRUCCIO-
;       NES LDAX,A++ Y LDAX,A+.O SOLAS DEN Y NO (RESPECTIVA-
;       MENTE) A EL REGISTRO "A" ; PERO EL CALCU DEBA EL
;       REGISTRO "A" SIN CAMBIO ALGUNO, POR TANTO SE A-
;       PLICA A LOS REGISTROS Y,U, Y S. PARA LOS RESULTA-
;       DOS ESPERADOS, USAR LAS INSTRUCCIONES LDAX 2,X .Y
;       LDAX 1,A.
;       LDAX Y LDAY PUEDEN SER USADOS COMO CONTADORES. LAS
;       CANTIDADES SON AFECTADAS COMO SE INDICA A CONTINUA-
;       CION :
;       A - NO AFECTADA.
;       B - NO AFECTADA.
;       Z - LDAX, LDAY : SE PONE SI EL RESULTADO ES CERO.
;           DE LO CONTRARIO BORRADA.
;       V - NO AFECTADA.
;       C - NO AFECTADA.

; DATOS DE ENTRADA A LA RUTINA "INDEX" :
;       - REGISTROS SIMULADOS (X O Y O U O S)
;       - AREA DEL USUARIO.

; DATOS DE SALIDA DE LA RUTINA "INDEX" :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA.
```

```
;        - PC ACTUALIZADO.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; EMPUJA REGISTROS DENTRO DEL STACK [PSH].

; ELABORO : A.B.M.                    FECHA : 2[illegible]-VIII-8[illegible].

; OBJETIVO : ALMACENAR REGISTROS EN EL STACK.

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.
;          - CODIGO DE LA INSTRUCCION.
;          - AREA DEL USUARIO.

; DATOS DE SALIDA :
;          - REGISTRO STACK.
;          - PC ACTUALIZADO.

; PROCEDIMIENTO : EL BIT UNO DEL CODIGO DE INSTRUCCION NOS
;          INDICARA EN QUE STACK SERA ALMACENADO CADA UNO DE
;          LOS REGISTROS. ANTES DE INCREMENTAR PC.
;          STACK HARDWARE "S" : SI SU VALOR ES CERO.
;          STACK DE USUARIO "U" : SI SU VALOR ES UNO.
;          EN SEGUIDA EL POSTBYTE NOS INDICARA LOS REGISTROS
;          QUE SE DESEAN ALMACENAR Y PC : = PC + 1. ;
;          SI EL BIT 7 ES PUESTO, ENTONCES :
;          S' <--- S-1, (S) <--- PCL
;          S' <--- S-1, (S) <--- PCH
;          SI EL BIT 6 ES PUESTO, ENTONCES :
;          S' <--- S-1, (S) <--- USL O SPL
;          S' <--- S-1, (S) <--- USH O SPH
```

```
;       SI EL BIT 5 ES PUESTO, ENTONCES :
;       S' <--- S-1, (S) <--- IYL
;       S' <--- S-1, (S) <--- IYH
;       SI EL BIT 4 ES PUESTO, ENTONCES :
;       S' <--- S-1, (S) <--- IXL
;       S' <--- S-1, (S) <--- IXH
;       SI EL BIT 3 ES PUESTO, ENTONCES :
;       S' <--- S-1, (S) <--- DPH
;       SI EL BIT 2 ES PUESTO, ENTONCES :
;       S' <--- S-1, (S) <--- ACCU
;       SI EL BIT 1 ES PUESTO, ENTONCES :
;       S' <--- S-1, (S) <--- ACCA
;       SI EL BIT 0 ES PUESTO, ENTONCES :
;       S' <--- S-1, (S) <--- DPL
; NOTA :AL SER PUESTO EL BIT 6 DEL REGISTRO, SE TOMARA EN
;       EN CUENTA LO SIGUIENTE :
;       SI EL BIT 1 DEL CODIGO DE INSTRUCCION POS CERO EL
;       REGISTRO POR ALMACENAR SERA EL STACK POINTER (SP)
;       DE LO CONTRARIO EL REGISTRO POR ALMACENAR SERA EL
;       STACK DEL USUARIO (US).
;       TODOS, ALGUNO O NINGUNO DE LOS REGISTROS DEL PROCE
;       SADOR PUEDEN SER APILADOS AL STACK. (CON EXCEPCION
;       DEL STACK USADO PARA ALMACENAR ESTOS REGISTROS).
;       LAS BANDERAS NO SON AFECTADAS.
;       +++ S = STACK +++.

;////////////////////////////////////////////////////////////////

; SACAR REGISTROS DEL STACK [_PUL_].

; ELABORO : A.E.H.                    FECHA : 24-VIII-82.
```

```
; OBJETIVO : GRABAR REGISTROS DEL STACK.

; DATOS DE ENTRADA :
;       - REGISTRO STACK (0 O 1).
;       - CODIGO DE INSTRUCCION.
;       - BASE DEL USUARIO.

; DATOS DE SALIDA :
;       - REGISTROS GRABADOS.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL BIT 1 DEL CODIGO DE INSTRUCCION NOS -
;       INDICARA SI YA HAYA SIDO GRABADOS LOS REGISTROS:
;       PARA BANDERA "0" : SI SU VALOR ES CERO.
;       PARA DEL USUARIO "0" : SI SU VALOR ES UNO.
;       SE OBTIENE EL POSTBYTE DEL PC Y PC : = PC + 1 .
;       SE OBTIENE EL POSTBYTE NOS INDICARA LOS REGISTROS
;       QUE SE DEBEN GRABAR Y PC : = PC + 1 ;
;       SI EL BIT 0 DE POSBYTE, ENTONCES :
;       SOM' <--- (S), S' <--- S + 1
;       SI EL BIT 1 DE POSBYTE, ENTONCES :
;       ACCA' <--- (S), S' <--- S + 1
;       SI EL BIT 2 DE POSBYTE, ENTONCES :
;       ACCB' <--- (S), S' <---- S + 1
;       SI EL BIT 3 DE POSBYTE, ENTONCES :
;       DPR' <--- (S), S' <--- S + 1
;       SI EL BIT 4 DE POSBYTE, ENTONCES :
;       IXH' <--- (S), S' <--- S + 1
;       IXL' <--- (S), S' <--- S + 1
;       SI EL BIT 5 DE POSBYTE, ENTONCES :
;       IYH' <--- (S), S' <--- S + 1
;       IYL' <--- (S), S' <--- S + 1
;       SI EL BIT 6 DE POSBYTE, ENTONCES :
```

```
;       USE' O SPH' <--- (S), S' <--- S + 1
;       USE' O SPL' <--- (S), S' <--- S + 1
;       SI EL BIT 7 DE POSTB, ENTONCES :
;       PCH' <--- (S), S' <--- S + 1
;       PCL' <--- (S), S' <--- S + 1
;       AL SER PUESTO EL BIT 6 DEL POSTBYTE, SE TOMARA EN
;       CUENTA LO SIGUIENTE :
;       SI EL BIT 1 DEL CODIGO DE INSTRUCCION FUE CERO, EL
;       REGISTRO PUNTERO DEL STACK USADO EL STACK HARDWARE
;       (SP) ; DE LO CONTRARIO EL REGISTRO PUNTERO USADO -
;       EL STACK DE USUARIO (US).
;       TODOS, ALGUNOS O NINGUNO DE LOS REGISTROS DEL PRO-
;       CESADOR PUEDEN SER SACADOS DEL STACK (CON EXCEP--
;       CION DEL STACK USADO PARA SACAR ESTOS REGISTROS).
;       LAS BANDERAS SERAN SOLAS VALIDAS DEL STACK ; DE
;       LO CONTRARIO NO SERAN AFECTADAS.

;////////////////////////////////////////////////////////////////

;RUTINA DE SUBRUTINA [_RTS_].

; REALIZO : A.A.S.                    FECHA : 24-VIII-82.

; OBJETIVO : SACAR DEL STACK HARDWARE EL CONTADOR DE PRO -
;       GRAMA.

; DATOS DE ENTRADA :
;       - STACK HARDWARE.

; DATOS DE SALIDA :
;       - CONTADOR DE PROGRAMA.
```

```
; PROCEDIMIENTO : EL CONTADOR DE PROGRAMA ES RECUPERADO DE
;       LA SUBRUTINA AL PROGRAMA LLAMADO. LA DIRECCION DE
;       RETORNO DE SALIDA DEL STACK.
;       PCH' <--- (SP), SP' <--- SP + 1
;       PCL' <--- (SP), SP' <--- SP + 1
;       LAS BANDERAS NO SON AFECTADAS.

;////////////////////////////////////////////////////////////

; SUMA ACUMULADOR "B" DENTRO DEL REGISTRO INDICE "X"
; [ADDA] .

; AUTOR : A.D.H.                    FECHA : 24-VIII-82.

; OBJETIVO : REALIZAR UNA SUMA ENTRE EL CONTENIDO DEL ACU-
;       MULADOR "B" Y EL REGISTRO INDICE "X".

; DATOS DE ENTRADA :
;       - REGISTROS CARGADOS.

; DATOS DE SALIDA :
;       - REGISTROS CARGADOS.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE SUMA EL "VALOR BINARIO" DEL ACUMULA-
;       DOR "B " DENTRO DEL REGISTRO INDICE "X".
;       IX' <--- IX + ACCB.
;       LAS BANDERAS NO SON AFECTADAS.
; NOTA :LA INSTRUCCION CARECE DE SIMILAR, PERO SI IDEN-
;       TICA A LA INSTRUCCION "ADA". LA PRIMERA TRATA A
;       "B" COMO UN NUMERO SIN SIGNO (DE DOS COMPLEMENTO),
;       EN TANTO QUE ADX LOS A "B" COMO UN OFFSET POSITIVO
```

```
;       ENTRE 0 Y 255.

;////////////////////////////////////////////////////////////////

; REGRESO DE INTERRUPCION [_RTI_].

; ELABORO : A.E.H.                    FECHA : 24-VII-82.

; OBJETIVO : RECUPERAR UNO O LAS REGISTROS DEL STACK HARD-
;       WARE.

; DATOS DE ENTRADA :
;       - STACK HARDWARE.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL REGISTRO DE CODIGO DE CONDICION ES LA
;       CABE DEL STACK HARDWARE.
;       CCR' <--- (SP), SP' <--- SP + 1
;       ENTONCES, SI EL BIT "E" ESTA PUESTO :
;       ACCA' <--- (SP), SP' <--- SP + 1
;       ACCB' <--- (SP), SP' <--- SP + 1
;       DPR' <--- (SP), SP' <--- SP + 1
;       IXH' <--- (SP), SP' <--- SP + 1
;       IXL' <--- (SP), SP' <--- SP + 1
;       IYH' <--- (SP), SP' <--- SP + 1
;       IYL' <--- (SP), SP' <--- SP + 1
;       USH' <--- (SP), SP' <--- SP + 1
;       USL' <--- (SP), SP' <--- SP + 1
;       PCH' <--- (SP), SP' <--- SP + 1
```

```
;       PCL' <--- (SP), SP' <--- SP + 1
;       DE LO CONTRARIO :
;       PCH' <--- (SP), SP' <--- SP + 1
;       PCL' <--- (SP), SP' <--- SP + 1
;       EL ESTADO DE MAQUINA SALVADO ES RECOBRADO DEL ---
;       STACK HARDWARE Y EL CONTROL ES REGRESADO AL PROGRA
;       MA INTERRUPIDO. SI EL BIT "E" (ESTADO COMPLETO —
;       SALVADO) ESTA LLENO, ESTO INDICA UNICAMENTE QUE -
;       UN SUBCONJUNTO DEL ESTADO DE MAQUINA FUE SALVADO (DIR
;       ECCION DE REGRESO Y CODIGO DE CONDICION) Y UNICA -
;       MENTE ESTE SUBCONJUNTO ES RECOBRADO.
;       LAS BANDERAS (CON) SON RECOBRADAS DEL STACK.

;////////////////////////////////////////////////////////////////

; SOBRE BANDERAS Y ESPERA UNA INTERRUPCION [ CWAI ].

; ELABORO : A.E.H.                    FECHA : 24-VIII-82.

; OBJETIVO : LIMPIAR BITS DEL "CCR" (REGISTRO DE CODIGO DE
;       CONDICION).

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.

; DATOS DE SALIDA :
;       - STACK HARDWARE.
;       - PC EN LA DIRECCION DEL VECTOR DE INTERRUPCION.

; PROCEDIMIENTO : ESTA INSTRUCCION REALIZARA UNA OPERACION
;       AND LOGICA ENTRE UN BYTE INMEDIATO (OPERANDO DE —
;       INSTRUCCION) Y EL REGISTRO DE CODIGO DE CONDICION,
```

```
;       ALMACENA EL ESTADO COMPLETO DE MAQUINA EN EL STACK
;       HARDWARE Y ESPERA A QUE SE PIQUE UNA TECLA.
;       SE OBTIENE UN BYTE (A,I) APUNTADO POR EL PC Y SE INCR
;       MENTA EL PC.
;       CCR' <---- CCR AND #I.
;       DONDE #I ES EL DATO INMEDIATO.
;       EN SEGUIDA SE PONE LA BANDERA "E" Y POR LO TANTO :
;       SP' <---- SP-1, (SP) <---- PCL
;       SP' <---- SP-1, (SP) <---- PCH
;       SP' <---- SP-1, (SP) <---- UUL
;       SP' <---- SP-1, (SP) <---- UUH
;       SP' <---- SP-1, (SP) <---- IYL
;       SP' <---- SP-1, (SP) <---- IYH
;       SP' <---- SP-1, (SP) <---- IXL
;       SP' <---- SP-1, (SP) <---- IXH
;       SP' <---- SP-1, (SP) <---- DPR
;       SP' <---- SP-1, (SP) <---- ACCB
;       SP' <---- SP-1, (SP) <---- ACCA
;       SP' <---- SP-1, (SP) <---- CCR
;       CUANDO EL PIQUEO DE UNA TECLA OCURRA, NINGUNA IN-
;       FORMACION DEL ESTADO DE MAQUINA NECESITARA SER SAL
;       VADO ANTES DE VECTORIZAR LA INTERRUPCION DE MANEJO
;       DE RUTINA.
;       LA INSTRUCCION RTI (REGRESO DE INTERRUPCION) REGRE
;       SARA AUTOMATICAMENTE EL ESTADO COMPLETO DE MAQUINA
;.      DESPUES DE HABER PROBADO EL BIT "E" DEL REGISTRO -
;       DE CODIGO DE CONDICION RECOBRADO.
;       LAS BANDERAS SON AFECTADAS DE ACUERDO A LA OPERA -
;       CION AND.

;////////////////////////////////////////////////////////////
```

```
; MULTIPLICACION [MUL].

; ELABORO : A.E.H.                    FECHA : 24-VIII-82.

; OBJETIVO : REALIZAR UNA MULTIPLICACION ENTRE DOS REGIS-
;       TROS DE 8 BITS.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE MULTIPLICAN LOS NUMEROS BINARIOS (SIN
;       SIGNO) DE LOS ACUMULADORES "A" Y "B" Y SE PONE EL
;       RESULTADO EN AMBOS ACUMULADORES ( EL ACUMULADOR -
;       "A" CONTENDRA EL BYTE MAS SIGNIFICATIVO DEL RESUL-
;       TADO ) .
;       ACCA' : ACCB' <--- ACCA X ACCB
;       LA MULTIPLICACION SIN SIGNO (BINARIA) PERMITE OPE-
;       RACIONES DE PRECISION MULTIPLE.
;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;       H - NO AFECTADA.
;       N - NO AFECTADA.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO BORRADA.
;       V - NO AFECTADA.
;       C - SE PONE SI EL BIT 7 DEL ACUMULADOR "B" (DE RE-
;           SULTADO) ES PUESTO. DE LO CONTRARIO LIMPIADA.
;       EL BIT DE CARRY (C) PERMITE REDONDEAR EL BYTE MAS
;       SIGNIFICATIVO CON LA SECUENCIA : MUL,ADCA # 0.

;////////////////////////////////////////////////////////////
```

```
; INTERRUPCION SOFTWARE # 1 [INT_1].

; ELABORO : A.B.M.                    FECHA : 25-VIII-62.

; OBJETIVO : HACER UNA INTERRUPCION Y SALVAR EL CONTENIDO
;          DE TODOS LOS REGISTROS DEL PROCESADOR.

; DATOS DE ENTRADA :
;          - REGISTROS SALVADOS.
;          - AREA DEL USUARIO.

; DATOS DE SALIDA :
;          - REGISTRO TEMPORAL.

; PROCEDIMIENTO : SE LE ASIGNA LA DIRECCION DE FFFA AL RE-
;          GISTRO TEMPORAL .
;          "TEMP" <--- FFFA.
;          LAS INTERRUPCIONES NORMAL (INQ) Y RAPIDA (FIQ) -
;          SON ENMASCARADAS (DESHABILITADAS).
;          EL CONTROL SE LLAMA A LA RUTINA "SW"

; DATOS DE ENTRADA A LA RUTINA "SW" :
;          - REGISTROS SALVADOS.
;          - AREA DEL USUARIO.
;          - REGISTRO TEMPORAL
;          - "I" Y "F" DE "CCR" LIMPIADAS.

; DATOS DE SALIDA DE LA RUTINA "SW" :
;          - LA BANDERA "E" ES PUESTA.
;          - PC EN LA DIRECCION DEL VECTOR DE INTERRUPCION -
;            # 1 . (( FFFA ) : ( FFFB )).

;////////////////////////////////////////////////////////////
```

```
; GRUPO CUATRO [GRUPO4_7.

; REALIZO : A.S.R.                    FECHA : 25-VIII-62.

; OBJETIVO : DECIDIR LA OPERACION POR REALIZAR.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - CODIGO DE INSTRUCCION.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : EL CODIGO DE LA INSTRUCCION ES CLASIFI -
;       CADO DE LA SIGUIENTE FORMA :
;       BIT     3 2 1 0     OPERACION
;       -       0 0 1 1     SUB'AD
;       -       1 1 0 0     OPEIV
;       -       1 1 0 1     OPEV
;       -       1 1 1 0     LDSTXU
;       -       1 1 1 1     LDLTXU

; DATOS DE ENTRADA A LAS RUTINAS QUE EJECUTAN LA OPERACION:
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - CODIGO DE INSTRUCCION.

; DATOS DE SALIDA DE LAS RUTINAS QUE EJECUTAN LA OPERACION:
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

;////////////////////////////////////////////////////////////
```

```
;  SUBSTRAER O SUMAR EN EL REGISTRO "D" [SUB'AD_].

; AUTOR : A.E.H.                    FECHA : 25-VIII-82.

; OBJETIVO : OBTENER LA DIRECCION DE LA LOCALIDAD DE MEMO-
;        RIA, DECIDIR SI SE REALIZARA UNA SUMA O UNA RESTA
;        EN EL ACUMULADOR "D" Y EJECUTARLA.

; DATOS DE ENTRADA :
;        - ACUMULADOR "D".
;        - AREA DEL USUARIO.
;        - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;        - ACUMULADOR "D".
;        - AREA DEL USUARIO.
;        - PC ACTUALIZADO.

; PROCEDIMIENTO : PARA OBTENER LA DIRECCION DE LA LOCALI—
;        DAD DE MEMORIA POR ACCESAR, EL CODIGO DE LA INS—
;        TRUCCION NOS INDICARA EL MODO DE DIRECCIONAMIENTO
;        POR UTILIZAR DE LA SIGUIENTE FORMA :
;        INMEDIATO DE 16 BITS "INM16" :
;        CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE 00.
;        DIRECTO "DIRECT" :
;        CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE 01.
;        INDEXADO "INDEX" :
;        CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE 10.
;        EXTENDIDO "EXTEN" :
;        CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE 11.
;        UNA VEZ OBTENIDA LA LOCALIDAD DE MEMORIA POR UTILI_
;        ZAR("M" : "M" + 1 = 16 BITS), SE EVALUARA EL BIT 6
;        DEL CODIGO DE LA INSTRUCCION DE LA SIGUIENTE FORMA:
```

```
;       SI SU VALOR ES CERO : SE SUBSTRAE EL VALOR EN LA -
;                             LOCALIDAD DE MEMORIA "E" : -
;                             "E"+1 DE ACUMULADOR "D".
;       D' <---- "D" - "E" : "E" + 1.
;       SI SU VALOR ES UNO : EL VALOR EN LA LOCALIDAD DE -
;                            MEMORIA "E":"E"+1 ES SUMADO -
;                            AL ACUMULADOR "D".
;       D' <---- "D" + "E" : "E" + 1.
;       UNA VEZ REALIZADA LA OPERACION (SUMA O RESTA) LAS
;       BANDERAS SERAN AFECTADAS DE LA SIGUIENTE FORMA :
;       H - NO AFECTADA.
;       N - SE PONE SI EL RESULTADO ES NEGATIVO. DE LO CON-
;           TRARIO BORRADA.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO BORRADA.
;       V - SE PONE SI UN  SOBRE FLUJO ES GENERADO. DE LO-
;           CONTRARIO BORRADA.
;       C - SE PONE SI UN "CARRY" (SUMA) O "BORROW" (RESTA)
;           ES GENERADO. DE LO CONTRARIO BORRADA.

; DATOS DE ENTRADA A LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR AFECTAR :
;       - REGISTROS SIMULADOS.
;       - 1, 2, O 3 BYTES.

; DATOS DE SALIDA DE LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR AFECTAR :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA POR AFECTAR.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////
```

```
; OPERACION = IV [OPRIV_].

; ELABORO : A.A.H.                    FECHA : 25-VIII-62.

; OBJETIVO : DECIDIR LA INSTRUCCION POR EJECUTAR.

; DATOS DE ENTRADA :
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;       - INSTRUCCION POR EJECUTAR.

; PROCEDIMIENTO : EL BIT 6 DEL CODIGO DE LA INSTRUCCION ES
;       EVALUADO DE LA SIGUIENTE FORMA :
;       CERO : SI SU VALOR ES CERO.
;       UNITO : SI SU VALOR ES UNO.

; DATOS DE ENTRADA A LAS RUTINAS DE INSTRUCCION :
;       - REGISTROS ESTABLECIDOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA DE LAS RUTINA DE INSTRUCCION :
;       - REGISTROS ESTABLECIDOS POR EFECTO.
;       - PC ACTUALIZADO.
;       - AREA DEL USUARIO.

;//////////////////////////////////////////////////////////////////

; COMPARACION CON REGISTRO INDICE "X" [CMPX_].

; ELABORO : A.H.H.                    FECHA : 25-VIII-62.
```

```
; OBJETIVO : LLAMAR A LA RUTINA COMPARACION DE 16 BITS
;        "CMP16".

; DATOS DE ENTRADA :
;        - REGISTRO INDICE "X".

; DATOS DE SALIDA :
;        - REGISTRO INDICE "X".
;        - PC ACTUALIZADO.

; PROCEDIMIENTO : SE TOMA EL REGISTRO INDICE "X" Y SE LLA-
;        MA A LA RUTINA DE COMPARACION DE 16 BITS "CMP16".

; DATOS DE ENTRADA A LA RUTINA "CMP16" :
;        - REGISTRO INDICE "X".
;        - AREA DEL USUARIO.
;        - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA DE LA RUTINA "CMP16" :
;        - REGISTROS SALVADOS.
;        - AREA DEL USUARIO.
;        - PC ACTALIZADO.

;////////////////////////////////////////////////////////////

; CARGADO O ALMACENADO DEL REGISTRO "D" [LDSTD].

; ELABORO : A.E.H.                    FECHA : 25-VIII-82.

; OBJETIVO : USAR EL ACUMULADOR "D" EN LA INSTRUCCION DE -
;        CARGADO O ALMACENADO EN 16 BITS.
```

```
; DATOS DE ENTRADA :
;       - ACUMULADOR "D".

; DATOS DE SALIDA :
;       - ACUMULADOR "D".
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE TOMA EL ACUMULADOR "D" Y SE LLAMA A
;       LA RUTINA DE CARGADO O ALMACENADO DE 16 BITS "LDST
;       16 ".

; DATOS DE ENTRADA A LA RUTINA "LDST16" .
;       - ACUMULADOR "D".
;       - AREA DEL USUARIO.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA DE LA RUTINA "LDST16" :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; OPERACION = V [OPEV].

; ELABORO : A.E.H.                    FECHA : 26-VIII-62.

; OBJETIVO : DECIDIR LA INSTRUCCION POR EJECUTAR.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
```

```
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL CODIGO DE LA INSTRUCCION ES EVALUADO
;       PARA SABER LA INSTRUCCION POR EJECUTAR :
;       BIT     6 5 4       INSTRUCCION
;       -       0 0 0       ADA
;       -       0 0 1       ADA
;       -       0 1 0       ADA
;       -       0 1 1       ADA
;       -       1 0 0       + CODIGO NO DEFINIDO.
;       -       1 0 1       LDATD
;       -       1 1 0       LDATD
;       -       1 1 1       LDATD

; DATOS DE ENTRADA A LAS RUTINAS QUE EJECUTAN LA INSTRUC-
; CION :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA DE LAS RUTINAS QUE EJECUTAN LA INSTRUC-
; CION :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////////
```

```
; SALTO CORTO A SUBRUTINA [_BSR_].

; ELABORO : A.B.H                    FECHA : 26-VII-82.

; OBJETIVO : REALIZAR UN SALTO DE 8 BITS EN EL CONTADOR DE
;       PROGRAMA.

; DATOS DE ENTRADA :
;       - REGISTROS SALVADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA :
;       - REGISTROS SALVADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE TOMA 1 BYTE DE OFFSET (POSTBYTE) DEL
;       AREA DE USUARIO Y SE GUARDA EN UN REGISTRO TEMPO-
;       RAL. "TEMP" <--- "M" (8 BITS).
;       EL CONTADOR DE PROGRAMA ES METIDO EN EL STACK :
;       SP'<--- SP-1, (SP) <--- PCL.
;       SP'<--- SP-1, (SP) <--- PCH.
;       EN SEGUIDA EL CONTADOR DE PROGRAMA ES CARGADO CON
;       EL OFFSET ANTERIOR :
;       PC' <--- PC + "TEMP".
;       LAS BANDERAS CC SON AFECTADAS.
;       UNA INSTRUCCION "RTS" (REGRESO DE SUBRUTINA) SE -
;       USADA PARA EL PROCESO INVERSO Y SERA LA ULTIMA INS
;       TRUCCION EJECUTADA EN UNA SUBRUTINA.

;////////////////////////////////////////////////////////////
```

```
; CALL A SUBRUTINA [_JSR_].

; ELABORO : A.S.H.                    FECHA : 26-VIII-82.

; OBJETIVO : REALIZAR UN  SALTO EN EL CONTADOR DE PROGRAMA

; DATOS DE ENTRADA :
;       - REGISTROS ESTABLECIDOS.
;       - BASE DEL USUARIO.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;       - STACK HARDWARE.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL CODIGO DE LA INSTRUCCION NOS INDICARA
;       EL MODO DE DIRECCIONAMIENTO POR UTILIZAR PARA OBTE
;       NER LA DIRECCION EFECTIVA POR UTILIZAR :
;       DIRECTO  (DIR) :
;       CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE 01.
;       INDEXADO (INDEX) :
;       CUANDO LOS BITS 5 Y 4 TOMEN LOS VALORES DE 10.
;       EXTENDIDO (EXTEN) :
;       CUANDO LOS BITS 5 Y 4 TOMEN LOS VALORES DE 11.
;       UNA VEZ OBTENIDA LA LOCALIDAD DE MEMORIA POR AFEC-
;       TAR DE 16 BITS, SE ALMACENA LA DIRECCION DE REGRE-
;       SO EN EL STACK HARDWARE :
;       SP' <--- SP-1, (SP) <--- PCL
;       SP' <--- SP-1, (SP) <--- PCH
;       Y EN SEGUIDA EL CONTADOR DE PROGRAMA ES TRANSFE —
;       RIDO A LA DIRECCION EFECTIVA.
;       PC' <--- EA.
;       UNA INSTRUCCION RTS DEBERA SER LA ULTIMA INSTRUC-
```

```
;       CION EJECUTADA DE LA SUBRUTINA .

; DATOS DE ENTRADA A LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR AFECTAR :
;       - REGISTROS SIMULADOS.
;       - 1 O 2 BYTES DE CODIGO.

; DATOS DE SALIDA DE LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR AFECTAR :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA POR AFECTAR.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; CARGADO O ALMACENADO CON REGISTRO "A" O "L" [LDSTX].

; AUTOR : A.E.H.                    FECHA : 26-VIII-82.

; OBJETIVO : DECIDIR QUE REGISTRO SERA AFECTADO POR LA INS
;       TRUCCION DE CARGADO O ALMACENADO.

; DATOS DE ENTRADA :
;       - CODIGO DE LA INSTRUCCION.
;       - REGISTROS SIMULADOS.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS POR AFECTAR.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL BIT 6 SE CLASIFICARA DE LA SIGUIENTE
;       FORMA :
```

```
;       INDICE "X" : SI SU VALOR ES CERO.
;       STACK "U" : SI SU VALOR ES UNO.
;       EN SEGUIDA SE LLAMA A LA RUTINA DE CARGADO O ALMA-
;       CENADO EN 16 BITS "LDST16".

; DATOS DE ENTRADA A LA RUTINA "LDST16" :
;       - REGISTROS SIMULADOS. (X O U).
;       - AREA DEL USUARIO.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA DE LA RUTINA "LDST16" :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; GRUPO CINCO [GRUPO5].

; ELABORO : A.E.H.                    FECHA : 27-VIII-82.

; OBJETIVO : OBTENER LA DIRECCION DE LA LOCALIDAD DE MEMO-
;       RIA Y EL ACUMULADOR "A" O "B" QUE AFECTARAN LA INS
;       TRUCCION POR EJECUTAR Y VER LA EJECUCION DE ESTA.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
```

```
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : MEDIANTE EL CODIGO DE LA INSTRUCCION SE
;       OBTIENE LA REGLA PARA ADQUIRIR LA DIRECCION POR A-
;       FECTAR, SE INDICA EL ACUMULADOR POR UTILIZAR Y POS
;       TERIORMENTE SE CLASIFICA EL CODIGO ENTRE LAS DIFE-
;       RENTES INSTRUCCIONES.
;       LAS REGLAS PARA OBTENER LA DIRECCION DE LA LOCALI-
;       DAD DE MEMORIA POR AFECTAR, SON LAS SIGUIENTES :
;       INMEDIATO DE 8 BITS "INME" :
;       CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE ØØ.
;       DIRECTO "DIRECT" :
;       CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE Ø1.
;       INDEXADO "INDEX" :
;       CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE 1Ø.
;       EXTENDIDO "EXTEN" :
;       CUANDO LOS BITS 5 Y 4 TOMEN VALORES DE 11.
;       EN SEGUIDA EL BIT 6 DEL CODIGO DE LA INSTRUCCION -
;       NOS INDICARA EL ACUMULADOR POR UTILIZAR, ESTO ES :
;       ACUMULADOR "A" : SI SU VALOR ES CERO.
;       ACUMULADOR "B" : SI SU VALOR ES UNO.
;       OBTENIENDO ASI LA LOCALIDAD DE MEMORIA Y EL ACUMU-
;       LADOR CON LOS CUALES SE EJECUTARAN LAS SIGUIENTES
;       OPERACIONES, SEGUN EL CODIGO DE LA INSTRUCCION :
;       BIT     3 2     OPERACION
;       -       Ø Ø     SUBCMP
;       -       Ø 1     OPEVI
;       -       1 Ø     OPEVII

; DATOS DE ENTRADA A LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR AFECTAR :
;       - REGISTROS SIMULADOS.
```

```
;       - 1 O 2 BYTES DE CODIGO.

; DATOS DE SALIDA DE LAS RUTINAS QUE EVALUAN LA DIRECCION
; POR EFECTUAR :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA POR EFECTUAR.
;       - PC ACTUALIZADO.

; DATOS DE ENTRADA A LAS RUTINAS QUE EJECUTAN LA OPERACION :
;       - REGISTROS SIMULADOS.
;       - DIRECCION EFECTIVA POR EFECTUAR.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA DE LAS RUTINAS QUE EJECUTAN LA OPERACION :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; DIRECCIONAMIENTO INMEDIATO DE 8 BITS [IMM8].

; ELABORO : A.E.H.                    FECHA : 27-VIII-82.

; OBJETIVO : OBTENER LA DIRECCION POR UTILIZAR.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.

; DATOS DE SALIDA :
;       - DIRECCION EFECTIVA (PC).
;       - PC ACTUALIZADO.
```

```
; PROCEDIMIENTO : SE OBTIENE UN BYTE DE CODIGO DEL PC EL -
;       CUAL SERA EL VALOR UTILIZADO EN UNA OPERACION DE -
;       TRANSFERENCIA.

;////////////////////////////////////////////////////////////

; SUBSTRAER O COMPARAR [_SUBCMP_].

; AUTOR : A.S.R                    FECHA : 27-VIII-82.

; OBJETIVO : DETERMINAR QUE INSTRUCCION SE EJECUTARA Y VER
;       SU EJECUCION .

; DATOS DE ENTRADA :
;       - ACUMULADOR POR UTILIZAR.
;       - MEMORIA DE 8 BITS POR UTILIZAR (1 BYTE).
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;       - REGISTROS ALMACENADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : SE HARA UNA RESTA ENTRE EL ACUMULADOR Y
;       LA MEMORIA :
;       "TEMP" <--- A - M
;       EL REGISTRO Y LA MEMORIA NO SERAN AFECTADOS. SI EL
;       BIT 0 DEL CODIGO DE LA INSTRUCCION ES CERO EL RE -
;       GISTRO TEMPORAL SE ALMACENARA EN EL ACUMULADOR (DA
;       TO DE ENTRADA) :
;       A' <--- "TEMP"
;       Y EL CONTENIDO DE LA LOCALIDAD DE MEMORIA SE BORRA -
```

```
;       AA .
;       SI EL BIT 0 DEL CODIGO DE LA INSTRUCCION FUE CERO
;       Y EL BIT 1 DE ESTE MISMO CODIGO DE LA INSTRUCCION
;       ES UNO EL BIT DE CARRY SERA UTILIZADO :
;       A' <--- A - C (CARRY).
;       LAS BANDERAS DEL REGISTRO DE CODIGO DE CONDICION
;       SON ALTERADAS DE LA SIGUIENTE MANERA :
;       H - NO DEFINIDA.
;       A - SE PONE SI EL RESULTADO ES NEGATIVO. DE LO CON-
;           TRARIO BORRADA.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO BORRADA.
;       V - SE PONE SI UN SOBRE FLUJO ES GENERADO. DE LO -
;           CONTRARIO BORRADA.
;       C - SE PONE SI UN PRESTAMO ES GENERADO. DE LO CONTRA
;           RIO BORRADA.

;////////////////////////////////////////////////////////////

; OPERACION = VI [OPCVI].

; ELABORO : A.S.H.                    FECHA : 27-VIII-62.

; OBJETIVO : DECIDIR LA INSTRUCCION POR EJECUTAR.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - LOCALIDAD DE MEMORIA.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;       - AREA DEL USUARIO.
```

```
;       - REGISTROS SIMULADOS.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL BIT 1 DEL CODIGO DE LA INSTRUCCION -
;       NOS INDICARA LA INSTRUCCION POR EJECUTAR DE LA SI-
;       GUIENTE FORMA :
;       OPELOG : SI SU VALOR ES CERO.
;       LDSTE  : SI SU VALOR ES UNO.
;       EN SEGUIDA SE LLAMA A LA RUTINA ELEGIDA.

; DATOS DE ENTRADA A LAS RUTINAS QUE EJECUTAN LA INSTRUC—
; CION  :
;       - REGISTROS SIMULADOS.
;       - LOCALIDAD DE MEMORIA.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA DE LAS RUTINAS QUE EJECUTAN LA INSTRUC—
; CION  :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; OPERACION LOGICA [OPELOG].

; ELABORO : A.E.h.                FECHA : 27-VIII-82.

; OBJETIVO : REALIZAR UNA OPERACION LOGICA (AND,OR O XOR)
;       ENTRE UN REGISTRO Y UNA LOCALIDAD DE MEMORIA.

; DATOS DE ENTRADA :
```

```
;       - ACUMULADOR.
;       - LOCALIDAD DE MEMORIA.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;       - RESULTADO CALCULADO.
;       - AREA DEL ACUMULADOR.

; PROCEDIMIENTO : SE REALIZA UNA OPERACION LOGICA ENTRE EL
;       CONTENIDO DE UN ACUMULADOR DE 8 BITS Y UNA LOCALI-
;       DAD DE MEMORIA DE 8 BITS.
;       PARA DECIDIR LA OPERACION LOGICA POR REALIZAR SE -
;       RECURRIRA AL CODIGO DE LA INSTRUCCION.
;       BIT     2 1     OPERACION
;       -       0 0     OR EXCLUSIVA
;       -       0 1     OR
;       -       1 0     AND
;       SI EL BIT 0 DEL CODIGO DE LA INSTRUCION ES UNO. -
;       EL RESULTADO DE LA OPERACION LOGICA ENTRE EL ACUMU
;       LADOR Y LA LOCALIDAD DE MEMORIA, SERA ALMACENADO -
;       EN UN REGISTRO TEMPORAL PARA NO AFECTAR EL CONTENI
;       DO DE AMBOS :
;       "TEMP" <--- "A" OPERACION LOGICA "M" .
;       DE LO CONTRARIO EL ACUMULADOR SERA AFECTADO Y LA -
;       MEMORIA NO.
;       A' <--- "A" OPERACION LOGICA "M".
;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;       N - NO AFECTADA.
;       S - SE PONE SI EL RESULTADO ES NEGATIVO. DE LO CO
;           TRARIO APAGADA.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO APAGADA.
;       V - BANDERA LIMPIADA.
```

```
;       C - NO AFECTADA.

;////////////////////////////////////////////////////////////

; CARGADO Y ALMACENADO DE 8 BITS [LDST8].

; AUTOR : A.B.M.                    FECHA : 27-VIII-82.

; OBJETIVO : VERIFICAR QUE SE EJECUTE UNA INSTRUCCION DE -
;       CARGADO O ALMACENADO DE UN ACUMULADOR DE 8 BITS —
;       CON UNA MEMORIA DE 8 BITS.

; DATOS DE ENTRADA :
;       - ACUMULADOR.
;       - MEMORIA.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;       - ACUMULADOR.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL BIT CERO NOS INDICARA SI SE DEBE REA-
;       LIZAR UNA INSTRUCCION DE CARGADO O ALMACENADO :
;       B' <--- "L" : SI SU VALOR ES CERO.
;       DE LO CONTRARIO (BIT 0=1) SE INTERCAMBIARAN LAS DI
;       RECCIONES, ESTO ES :
;       B' <--- "H" : LOS BITS 5 Y 4 DEL CODIGO DE LA INS-
;                     TRUCCION SERAN EVALUADOS ; SI TOMAN
;                     LOS VALORES DE 00 SE INDICARA QUE EL
;                     CODIGO NO ESTA DEFINIDO Y SE DARA —
;                     POR TERMINADA LA RUTINA.
;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
```

```
;       H - NO AFECTADA.
;       H - SE PONE SI EL DATO TRANSFERIDO ES NEGATIVO. EN
;           LO CONTRARIO BORRADA.
;       Z - SE PONE SI EL DATO TRANSFERIDO ES CERO. EN LO
;           CONTRARIO BORRADA.
;       V - SIEMPRE LIMPIADA.
;       C - NO AFECTADA.

;////////////////////////////////////////////////////////////////

; OPERACION # VII [OPEVII].

; ELABORO : A.M.R.                    FECHA : 30-VIII-62.

; OBJETIVO : DECIDIR LA INSTRUCCION POR EJECUTAR.

; DATOS DE ENTRADA :
;       - ACUMULADOR.
;       - LOCALIDAD DE MEMORIA.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; PROCEDIMIENTO : EL BIT 0 DEL CODIGO DE LA  INSTRUCCION -
;       NOS INDICARA LA INSTRUCCION POR EJECUTAR :
;       OPELOG : SI SU VALOR ES CERO.
;       ADCALD : SI SU VALOR ES UNO.
;       LA SUBRUTINA SE LLAMA A LA RUTINA ELEGIDA.
```

```
; DATOS DE ENTRADA A LAS RUTINAS QUE EJECUTAN LA INSTRUC -
; CION :
;       - ACUMULADOR .
;       - LOCALIDAD DE MEMORIA.
;       - CODIGO DE LA INSTRUCCION.

; DATOS DE SALIDA DE LAS RUTINAS QUE EJECUTAN LA INSTRUC -
; CION :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

;////////////////////////////////////////////////////////////

; SUMA DE 8 BITS CON O SIN CARRY [ADCADD].

; ELABORO : A.E.H.                    FECHA : 30-VIII-82.

; OBJETIVO : SUMAR EL CONTENIDO DE UNA LOCALIDAD DE MEMO -
;       RIA DE 8 BITS A UN ACUMULADOR.

; DATOS DE ENTRADA :
;       - ACUMULADOR.
;       - LOCALIDAD DE MEMORIA.
;       - CODIGO DE LA INSTRUCCION.

; PROCEDIMIENTO : SE SUMARAN LOS CONTENIDOS DEL ACUMULADOR
;       Y LA MEMORIA Y EL RESULTADO SERA CARGADO EN EL MIS
;       MO ACUMULADOR, ESTO ES :
;       A' <--- "A" + "E".
;       EN SEGUIDA EL BIT 1 DEL CODIGO DE LA INSTRUCION
        SE PRUEBA. SI ES CERO EL CONTENIDO DE LA BANDERA -
```

```
;       DE CARRY SERA SUMADO AL ACUMULADOR HACIEN UTILIZA-
;       DO, ESTO ES :
;       R' <--- "R" + 3 (CARRY).
;       DE LO CONTRARIO SE DARA POR TERMINADA ESTA RUTINA.
;       LAS BANDERAS SON AFECTADAS DE LA SIGUIENTE FORMA :
;       H - SE PONE SI UN CARRY INTERMEDIO SE GENERADO. DE
;           LO CONTRARIO BORRADA.
;       N - SE PONE SI EL RESULTADO ES NEGATIVO. DE LO CON
;           TRARIO BORRADA.
;       Z - SE PONE SI EL RESULTADO ES CERO. DE LO CONTRA-
;           RIO BORRADA.
;       V - SE PONE SI UN DESBORDAMIENTO ES GENERADO. DE LO -
;           CONTRARIO BORRADA.
;       C - SE PONE SI UN CARRY ES GENERADO. DE LO CONTRA-
;           RIO BORRADA.

;////////////////////////////////////////////////////////////

; SUBRUTINA A PASOS [SUBPRO_I].

; AUTORES : A.S.R.                    FECHA : 30-VIII-82.

; OBJETIVO : EJECUTAR A PASOS EL PROGRAMA SIMULADO.

; DATOS DE ENTRADA :
;       - BUFFER.
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
```

```
; PROCEDIMIENTO : CON AYUDA DE LA RUTINA DE UTILERIA "OB-
;       TEN" SE SABRA CUANTAS INSTRUCCIONES SE VAN A EJE-
;       CUTAR (INSTRUCCION POR INSTRUCCION) CON EL CONSE-
;       CUENTE DESPLIEGUE DE REGISTROS.

; DATOS DE ENTRADA A LA RUTINA DE "IINST" :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA DE LA RUTINA "IINST" :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.
;       - PC ACTUALIZADO.

; DATOS DE ENTRADA A LA RUTINA DE DESPLIEGUE DE PASOS "DES
; PAS" :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA DE LA RUTINA DE DESPLIEGUE DE PASOS "DES
; PAS" :
;       - REGISTROS SIMULADOS.
;       - LOCALIDADES DE MEMORIA.

;////////////////////////////////////////////////////////////

; DESPLIEGUE DE PASOS [_DESPAS_].

; ELABORO : A.E.H.                    FECHA : 31-VIII-62.

; OBJETIVO : MOSTRAR EL CONTENIDO DE LOS REGISTROS Y DIR—
;       RAS PUESTAS, DESPUES DE HABER EJECUTADO UN BLO -
```

```
;       QUE DE INSTRUCCIONES.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - BANDERAS HABILITADAS.

; PROCEDIMIENTO : DESPUES DE HABER EJECUTADO UN BLOQUE DE
;       INSTRUCCIONES (UNA INSTRUCCION O MAS) SE REALIZARA
;       UN DESPLIEGUE DEL CONTENIDO DE TODOS LOS REGISTROS
;       SE INDICARAN ADEMAS LAS BANDERAS QUE SE ENCUENTRAN
;       HABILITADAS. EL CONTENIDO DE CADA REGISTRO SE MOS-
;       TRARA EN SU REPRESENTACION ASCII (HEXADECIMAL).

; DATOS DE ENTRADA A LA RUTINA QUE CONVIERTE UN VALOR BINA
; RIO EN UN VALOR ASCII (HEXADECIMAL) "ASCII" :
;       - VALOR BINARIO DEL CONTENIDO DEL REGISTRO.

; DATOS DE SALIDA DE LA RUTINA "ASCII" :
;       - VALOR ASCII (HEXADECIMAL).

;////////////////////////////////////////////////////////////

; CONVERSION DE UN NUMERO BINARIO A SU REPRESENTACION ASC-
; II EN HEXADECIMAL [ASCII].

; ELABORO : A.E.H.                    FECHA : 2-XII-82.

; OBJETIVO : CONVERTIR UN NUMERO BINARIO A SU ASCII.
```

```
DATOS DE ENTRADA :
        - VALOR BINARIO.

DATOS DE SALIDA :
        - VALOR ASCII (HEXADECIMAL).

PROCEDIMIENTO : EL VALOR EN CODIGO ASCII SERA OBTENIDO -
:           EN DOS PARTES.
:           EL PRIMER VALOR SE OBTENDRA AL MULTIPLICAR EL VA—
:           LOR BINARIO POR 16 Y REALIZAR UNA (AND) OPERACION
:           LOGICA ENTRE ESTE VALOR Y OFH.
;           SI ESTE VALOR ES MENOR O IGUAL QUE 9, ENTONCES SE
;           LE SUMARA EL ASCII DEL NUMERO CERO.
;           DE LO CONTRARIO, SE LE SUMARA EL ASCII DE "A" Y SE
;           LE RESTARA EL #9.
;           EL SEGUNDO VALOR SE ENCONTRARA REALIZANDO UNA OPE—
;           RACION LOGICA (AND) ENTRE EL VALOR BINARIO Y EL VA
;           LOR OFH.

;//////////////////////////////////////////////////////////////


; DESPLIEGUE [DESP].

; ELABORO : A.S.E.                     FECHA : 31-VIII-82.

; OBJETIVO : DECIDIR SI SE MOSTRARA EL CONTENIDO DE LOS RE
;           GISTROS SIMULADOS O DE LAS LOCALIDADES DE MEMORIA.

; DATOS DE ENTRADA :
;           - BUFFER.
;           - REGISTROS SIMULADOS.
;           - AREA DEL USUARIO.
```

```
; DATOS DE SALIDA :
;       - REGISTROS SIMULADOS.
;       - AREA DEL USUARIO.

; PROCEDIMIENTO : CON AYUDA DE LA RUTINA DE UTILERIA "OB -
;       TEN" SE SABE SI SE TRATA DE UN DESPLIEGUE DE REGIS
;       TROS O DE LOCALIDADES DE MEMORIA. SI SE TRATA DE -
;       REGISTROS SE EJECUTA LA RUTINA "DESREG". EN CASO -
;       DE TRATARSE DE UN DESPLIEGUE DE LOCALIDADES DE ME-
;       MORIA, SE EJECUTARA LA RUTINA "INIFIN".

; DATOS DE ENTRADA A LA RUTINA "OBTEN" :
;       - BUFFER.

; DATOS DE SALIDA DE LA RUTINA "OBTEN" :
;       - CARACTER.

; DATOS DE ENTRADA A LA RUTINA "DESREG" :
;       - REGISTROS SIMULADOS.
;       - BUFFER.
;       - CARACTER.

; DATOS DE SALIDA DE LA RUTINA "DESREG" :
;       - REGISTROS SIMULADOS.

; DATOS DE ENTRADA A LA RUTINA "INIFIN" :
;       - BUFFER.

; DATOS DE SALIDA DE LA RUTINA "INIFIN" :
;       - DIRECCION DE INICIO.
;       - DIRECCION DE FIN.

;////////////////////////////////////////////////////////////
```

```
; OBTENCION DE DIRECCIONES DE INICIO A FIN [INIFIN].

; ELABORO : A.A.H.                    FECHA : 2-II-82.

; OBJETIVO : SABER LAS DIRECCIONES DE INICIO Y FIN DE LAS
;       LOCALIDADES POR ACCESAR.

; DATOS DE ENTRADA :
;       - BUFFER.

; DATOS DE SALIDA :
;       - DIRECCION DE INICIO.
;       - DIRECCION DE FIN.

; PROCEDIMIENTO : CON LA AYUDA DE LA RUTINA "OBTDIR" SE OB
;       TIENE UNA DIRECCION. ESTA DIRECCION PUEDE SER DE -
;       INICIO O DE FIN.

; DATOS DE ENTRADA A LA RUTINA "OBTDIR" :
;       - BUFFER.

; DATOS DE SALIDA DE LA RUTINA "OBTDIR" :
;       - DIRECCION DE UNA LOCALIDAD DE MEMORIA.

; DATOS DE ENTRADA A LA RUTINA "OBTBYT" :
;       - BUFFER.

; DATOS DE SALIDA DE LA RUTINA "OBTBYT" :
;       - CARACTER.

; DATOS DE ENTRADA A LA RUTINA PARA DESPLIEGUE DE MEMORIA
; "DESLOC" :
;       - DIRECCION DE INICIO.
```

```
;       - DIRECCION DE FIN.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA DE LA RUTINA "DESLOC" :
;       - CONTENIDO DE UNA ZONA DE MEMORIA.

;////////////////////////////////////////////////////////////

; VALOR O CARACTER [_NULO_].

; ELABORO : A.E.H.                    FECHA : 6-XII-82.

; OBJETIVO : INDICAR QUE NO EXISTE EL REGISTRO PEDIDO.

; DATOS DE ENTRADA :
;       - NINGUNO.

; DATOS DE SALIDA :
;       - INDICACION EN EL VIDEO.

; PROCEDIMIENTO : AL SER LLAMADA ESTA RUTINA SE INDICARA -
;       EN EL VIDEO QUE EL REGISTRO PEDIDO NO EXISTE EN E
;       L.MICROPROCESADOR . MC6809 CON EL SIGUIENTE MENSAJE :
;       " REGISTRO DESCONOCIDO ".

;////////////////////////////////////////////////////////////

; DESPLIEGUE DE REGISTROS [_DESREG_].

; ELABORO : A.E.H.                    FECHA : 6-XII-82.
```

```
; OBJETIVO : REALIZAR UN DESPLIEGUE DEL CONTENIDO DEL RE -
;          GISTRO SOLICITADO.

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.
;          - BUFFER.

; DATOS DE SALIDA :
;          - CONTENIDO DEL REGISTRO SOLICITADO.

; PROCEDIMIENTO : SE REALIZARA EL DESPLIEGUE DE LOS CONTE-
;          NIDOS DE UNO O TODOS LOS REGISTROS DEL LIBROFRECU-
;          SADOR LCU809, EN VALOR ASCII.
;          EN CASO DE NO EXISTIR EL REGISTRO SE INDICARA "RE-
;          GISTRO DESCONOCIDO" .

; DATOS DE ENTRADA A LA RUTINA "CHTEN" :
;          - BUFFER.

; DATOS DE SALIDA DE LA RUTINA "CHTEN" :
;          - CARACTER.

; DATOS DE ENTRADA A LA RUTINA "ASCII" :
;          - VALOR BINARIO DEL CONTENIDO DE UN REGISTRO.

; DATOS DE SALIDA DE LA RUTINA "ASCII" :
;          - VALOR ASCII (HEXADECIMAL).

; DATOS DE ENTRADA A LA RUTINA "DESPAS" :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.

; DATOS DE SALIDA DE LA RUTINA "DESPAS" :
```

```
;          - REGISTROS SIMULADOS.
;          - BANDERAS HABILITADAS.

; DATOS DE ENTRADA A LA RUTINA "NULO" .
;          - NINGUNO.

; DATOS DE SALIDA DE LA RUTINA "NULO" :
;          - INDICACION DE " REGISTRO DESCONOCIDO "

;/////////////////////////////////////////////////////////////////


; DESPLIEGUE DE LOCALIDAD [ DESLOC ].

; ELABORO : A.E.H.                    FECHA : 6-IX-82.

; OBJETIVO : MOSTRAR EL CONTENIDO DE UNA ZONA DE MEMORIA.

; DATOS DE ENTRADA :
;          - AREA DEL USUARIO.
;          - DIRECCION DE INICIO.
;          - DIRECCION DE FIN .

; DATOS DE SALIDA :
;          - CONTENIDO DE UNA ZONA DE MEMORIA.

; PROCEDIMIENTO : SE MOSTRARA EL CONTENIDO DE LAS ZONAS DE
;         MEMORIA CONTENIDAS POR LAS DIRECCIONES DE INICIO Y
;         FIN EN VALOR ASCII.

; DATOS DE ENTRADA A LA RUTINA "ASCII" :
;          - VALOR BINARIO.

; DATOS DE SALIDA DE LA RUTINA "ASCII" :
```

```
;          - VALOR ASCII (HEXADECIMAL).

;//////////////////////////////////////////////////////////

; SUSTITUCION [SUST].

; ELABORO : A.S.H.                    FECHA : 10-XII-82.

; OBJETIVO : DECIDIR SI SE REALIZARA UNA SUSTITUCION DE VA
;          LORES EN LOS REGISTROS O EN LAS LOCALIDADES DE ME-
;          MORIA.

; DATOS DE ENTRADA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.
;          - BUFFER.

; DATOS DE SALIDA :
;          - REGISTROS SIMULADOS.
;          - AREA DEL USUARIO.

; PROCEDIMIENTO : CON AYUDA DE LA RUTINA DE UTILERIA "OB-
;          TEN" SE SABRA SI SE DEBE REALIZAR UNA SUSTITUCION
;          EN LOS REGISTROS O EN LAS LOCALIDADES DE MEMORIA.
;          SI SE TRATA DE UNA SUSTITUCION EN LAS LOCALIDADES
;          DE MEMORIA SE LLAMARA A LA RUTINA "SUSLOC". EN CA-
;          SO DE TRATARSE DE UNA SUSTITUCION EN LOS REGISTROS
;          SE LLAMARA A LA RUTINA "SUSREG".

; DATOS DE ENTRADA A LA RUTINA "OBTEN" :
;          - BUFFER.
```

```
; DATOS DE SALIDA DE LA RUTINA "OBTEN" :
;       - CARACTER.

; DATOS DE ENTRADA A LA RUTINA "SUSLOC" :
;       - BUFFER.
;       - AREA DEL USUARIO.

; DATOS DE SALIDA DE LA RUTINA "SUSLOC" :
;       - LOCALIDAD DE MEMORIA CON EL VALOR DESEADO.

; DATOS DE ENTRADA A LA RUTINA "SUSREG" :
;       - REGISTROS SIMULADOS.
;       - BUFFER.

; DATOS DE SALIDA DE LA RUTINA "SUSREG" :
;       - REGISTROS SIMULADOS CON EL VALOR DESEADO.

;////////////////////////////////////////////////////////////

; SUSTITUCION EN LOCALIDAD DE MEMORIA [SUSLOC].

; ELABORO : A.E.H.                    FECHA : 10-XII-62.

; OBJETIVO : MODIFICAR EL CONTENIDO DE UNA DETERMINADA LO-
;       CALIDAD DE MEMORIA.

; DATOS DE ENTRADA :
;       - AREA DEL USUARIO.
;       - BUFFER.

; DATOS DE SALIDA :
;       - LOCALIDAD DE MEMORIA.
```

```
; PROCEDIMIENTO : CON AYUDA DE LA RUTINA "OBTDIR" SE SABRA
;       LA DIRECCION DE LA LOCALIDAD POR AFECTAR. EL SIGUI
;       DA SE REALIZARA UN DESPLIEGUE DE LA DIRECCION DE -
;       ESTA LOCALIDAD, SU VALOR Y SU ASCII; ESPERANDO EL
;       NUEVO VALOR DESEADO. SI SE OPRIME UN 'CR' SE LE SU
;       MARA UNA UNIDAD A LA DIRECCION DE INICIO, PUDIENDO
;       AFECTAR EL CONTENIDO DE ESTA NUEVA DIRECCION. CON
;       UN '/' SE PODRA SALIR ESTA RUTINA.
;       SI SE DA UN CARACTER ERRONEO SE INDICARA EN EL VI-
;       DEO.

; DATOS DE ENTRADA A LA RUTINA "OBTDIR" :
;       - BUFFER.

; DATOS DE SALIDA DE LA RUTINA "OBTDIR" :
;       - DIRECCION.

; DATOS DE ENTRADA A LA RUTINA "OBTER" :
;       - BUFFER.

; DATOS DE SALIDA DE LA RUTINA "OBTER" :
;       - CARACTER.

; DATOS DE ENTRADA A LA RUTINA "ASCII" :
;       - VALOR BINARIO.

; DATOS DE SALIDADE LA RUTINA "ASCII" :
;       - VALOR ASCII (HEXADECIMAL).

; DATOS DE ENTRADA A LA RUTEIA "ERROR" :
;       - NINGUNO.

; DATOS DE SALIDA DE LA RUTINA "ERROR" :
```

```
;       - INDICACION EN EL VIDEO.

;////////////////////////////////////////////////////////////

; ERROR EN COMANDO [_ERROR_].

; ELABORO : A.S.H.                    FECHA : 14-XII-82.

; OBJETIVO : INDICAR QUE EXISTE UN ERROR.

; DATOS DE ENTRADA :
;       - MINUTO.

; DATOS DE SALIDA :
;       - INDICACION EN EL VIDEO.

; PROCEDIMIENTO : AL SER LLAMADA ESTA RUTINA, SE INDICARA
;       EN EL VIDEO QUE EXISTE UN ERROR EN EL COMANDO CON
;       EL MENSAJE " ERROR EN COMANDO " .

;////////////////////////////////////////////////////////////


; SUBTITUCION EN REGISTRO [_SUBREG_].

; ELABORO : A.S.H.                    FECHA : 10-XII-82.

; OBJETIVO : MODIFICAR EL CONTENIDO DE UN REGISTRO.

; DATOS DE ENTRADA :
;       - REGISTROS SIMULADOS.
;       - BUFFER.
```

```
; DATOS DE SALIDA :
;       - REGISTROS MODIFICADOS.

; PROCEDIMIENTO : CON AYUDA DE LA RUTINA "OBTEN" SE SABRA
;       QUE REGISTRO SE QUIERE AFECTAR.
;       SE MOSTRARA EL REGISTRO POR AFECTAR, SU DATO Y SU
;       ASCII Y SE ESPERARA SU NUEVO VALOR.

; DATOS DE ENTRADA A LA RUTINA "OBTEN" :
;       - BUFFER.

; DATOS DE SALIDA DE LA RUTINA "OBTEN" :
;       - CARACTER.

; DATOS DE ENTRADA A LA RUTINA "ASCII" :
;       - VALOR BINARIO.

; DATOS DE SALIDA DE LA RUTINA "ASCII" :
;       - VALOR ASCII (HEXADECIMAL)

; DATOS DE ENTRADA A LA RUTINA "NULO" :
;       - NINGUNO.

; DATOS DE SALIDA DE LA RUTINA "NULO" :
;       - INDICACION EN EL VIDEO " REGISTRO DESCONOCIDO ".

; DATOS DE ENTRADA A LA RUTINA "ERROR" :
;       - NINGUNO.

; DATOS DE SALIDA DE LA RUTINA "ERROR" :
;       - INDICACION EN EL VIDEO "ERROR EN COMANDO ".

;////////////////////////////////////////////////////////////
```

```
; RUTINA DE UTILERIA "OBTEN" [_OBTEN_].

; ELABORO : A.E.H.                    FECHA : 15-IX-82.

; OBJETIVO : OBTENER UN CARACTER DEL BUFFER.

; DATOS DE ENTRADA :
;       - BUFFER.

; DATOS DE SALIDA :
;       - CARACTER.

; PROCEDIMIENTO : SE TOMA EL CARACTER INDICADO POR EL APUN
;       TADOR DE INICIO DEL BUFFER Y SE INCREMENTA ESTE A-
;       PUNTADOR.

;////////////////////////////////////////////////////////////
```

CAPITULO 5

PSEUDO-CODIGO
Y
CODIGO

```
;       PSEUDO-CODIGO                                   7/IX/82
;                                   (PPRIN)

; Inicio
;    1.- Ejecuta Inicializa registros
;    2.- Ejecuta Lee nombre de registros
;    3.- Ejecuta "DISCO"
;    4.- Fin := Falso
;    5.- Repite
;        A.- Ejecuta "ENTTER"
;        B.- Ejecuta "CONTEN"
;        C.- Clasifica comando entre
;            1.- "E" ejecuta "EJECOM"
;            2.- "I" ejecuta "EJEPAS"
;            3.- "D" ejecuta "DESP"
;            4.- "S" ejecuta "SUST"
;            5.- "F" Fin := Verdadero
;            6.- Otros ejecuta "ERROR"
;    6.- Hasta Fin := Verdadero
; Fin
;.....
;            CODIGO                                  17/XII/82
;                                   (PPRIN)

        FIN   :  DS 1                                ; 1.-
        ACCD  :  DS 2
        ACCB  :  DS 1
        ACCA  :  DS 1
        RIX   :  DB ØØ,ØØ
        RIY   :  DB ØØ,ØØ
        RSP   :  DB ØØ,ØØ
        US    :  DB ØØ,ØØ
        RPC   :  DB ØØ,ØØ
```

```
CC     :  DB ØØ
DP     :  DB ØØ
PPRIN  :  CALL    BUFFER          ; 2.-
          CALL    CURSEU          ; 3.-
          LD      A,Ø             ; 4.-
          LD      (FIN),A
PRIN5  :                          ; 5.-
          CALL    BUFFER          ; 5.A.-
          CALL    CURSEU          ; 5.B.-
          CP      A,'E'           ; 5.C.-
          JP      NZ,PRINC2
          CALL    EJECCU          ; 5.C.1.-
          JP      PRIN6
PRINC2:   CP      A,'I'
          JP      NZ,PRINC3
          CALL    INST            ; 5.C.2.-
          JP      PRIN6
PRINC3:   CP      A,'D'
          JP      NZ,PRINC4
          CALL    DESP            ; 5.C.3.-
          JP      PRIN6
PRINC4:   CP      A,'S'
          JP      NZ,PRINC5
          CALL    SUST            ; 5.C.4.-
          JP      PRIN6
PRINC5:   CP      A,'F'
          JP      NZ,PRINC6
          LD      A,ØØFH          ; 5.C.5.-
          LD      (FIN),A
          JP      PRIN6
PRINC6:   CALL    ERROR           ; 5.C.6.-
PRIN6  :  LD      A,(FIN)         ; 6.-
          CP      Ø
```

```
                JP      Z,PRIN5
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;       PSEUDO-CODIGO                           8/IX/82
;                               (DISCO)

; Inicio
;    1.- Es abierto el archivo correspondiente
;    2.- Mientras no es leido todo, ejecuta :
;        A.- Lee bloque
;        B.- Relocaliza bloque
; Fin

;.....................................................................

;           CODIGO                              17/I/83
;                               (DISCO)

        CDOS   : EQU 5
        ABRIR  : EQU 15
        LEERD  : EQU 2Ø
        CERRAR: EQU 16
        BUFSIS: EQU 8Ø
        ARCHI  :
        DISSEL:   DS 1
        NOMBRE:   DS 7
        EXT    :  DS 3
        ENT    :  DS 1
```

```
OTROS :  DS 2Ø
DISCO :  LD      A,Ø            ; 1.-
         LD      (DISSEL),A     ; Disco corriente (Ø)
         LD      (EXT),A
         LD      A,Ø2ØH
         LD      HL,NOMBRE
         LD      B,1Ø           ; Llenar con blancos
                                ; los espacios a la
                                ; derecha
REP1  :  LD      (HL),A
         INC     HL
         DJNZ,   REP1
         LD      IX,AP.INI
         LD      A,Ø3AH         ; Disco corriente (Ø)
         CP      A,(IX + 1)
         JP      NZ,OTRO
         LD      A,(IX)         ; Disco "A" (1)
         CP      A,Ø41H
         JP      NZ,OTRO1
         LD      A,1
         JP      OTRO2
OTRO1 :  CP      A,Ø42H         ; Disco "B" (2)
         JP      NZ,OTRO
         LD      A,2
OTRO2 :  LD      (DISSEL),A
         INC     IX
         INC     IX
OTRO  :  LD      HL,NOMBRE
         LD      B,8
         LD      C,Ø2EH         ; Nombre de archivo
LOOP  :  LD      A,(IX)
         CP      A,C
         JP      Z,SAL
         LD      (HL),A
```

```
        INC     IX
        INC     HL
        DJNZ,   LOOP
        JP      LEER
SAL   : LD      B,3             ; Extensión 3 Bytes
        LD      HL,EXT
        LD      C,ØØDH
LOOP1 : LD      A,(IX)
        CP      A,C
        JP      Z,LEER
        LD      (HL),A
        INC     IX
        INC     HL
        DJNZ,   LOOP1
LEER  : LD      C,ABRIR
        LD      DE,ARCHIVO
        CALL    CDOS
        CP      A,-1
        JP      Z,ERROR         ; Archivo no encontrar
        LD      DE,USUARI
LOOP2 : PUSH    DE              ; 2.-
        LD      C,LEERD         ; 2.A.-
        LD      DE,ARCHI
        CALL    CDOS
        POP     DE
        LD      HL,TIPSIS
        LD      BC,8ØH
        OR      A,A             ; Borra carry
        LD      HL,ULU          ; Si DE = ULU ,error
        SBC     HL,DE
        JP      C,ERROR
        SBC     HL,BC           ; Si BC=Ø -- Fin
        JP      C,ERROR
        LDIR
```

```
        CP      A,Ø
        JP      Z,LOOP2
        CP      A,2
        JP      Z,ERROR
        LD      C,CERRAR
        LD      DE,ARCHI        ; 2.B.-
        CALL    CDOS
        RET
```

::

PSEUDO-CODIGO 10/IX/82

(BUFFER)

```
Inicio
   1.- Apuntador Inicio = Inicio Buffer
   2.- Apuntador Fin = Inicio Buffer
   3.- Fin = Falso     (etiqueta  FIN4)
   4.- Repite
       A.- Lee caracter de consola
       B.- Si caracter = "DELET"
           1.- Entonces: Si Ap. Fin > Ap. Inicio, entonces:
               A.- Ap. Fin = Ap. Fin - 1
               B.- Borra caracter de consola
           2.- De lo contrario:
               A.- Buffer (Ap. Fin) : = caracter
               B.- Apuntador Fin : = Ap. Fin + 1
               C.- Si Ap. Fin > Tope Buffer
                   1.- Entonces Fin = Verdadero
               D.- Si caracter = "CR"
                   1.- Entonces Fin = Verdadero
   5.- Hasta Fin = Verdadero
Fin
```

.....

CODIGO 4/I/83

(BUFFER)

```
USUARI:  DS 5
LEEC   :  EQU 1
ESCC   :  EQU 2
AP.INI:  DS 2
AP.FIN:  DS 2
BUFF   :  DS 2Ø
FIN4   :  DS 1
BUFFER:  LD      DE,BUFF        ; 1.-
         LD      (AP.INI),DE
         LD      (AP.FIN),DE    ; 2.-
         LD      A,Ø            ; 3.-
         LD      (FIN4),A
BUFF4 :                         ; 4.-
         LD      C,LEEC         ; 4.A.- El caracter
         CALL    CDOS           ;       queda en el re
                                ;       gistro "A" del
                                ;       microprocesa.
         CP      A,Ø7FH         ; 4.B.-
         JP      NZ,BUFB.2
         LD      HL,(AP.FIN)    ; 4.B.1.-
         LD      DE,(AP.INI)
         OR      A,A            ; Borra carry
         SBC     HL,DE
         JP      Z,BUFF5
         LD      HL,(AP.FIN)    ; 4.B.1.A.-
         DEC     HL
         LD      (AP.FIN),HL
         LD      C,ESCC         ; 4.B.1.B.-
         LD      E,ØØ8H         ; espacio atras
         CALL    CDOS
```

```
                JP      BUFF5
BUFB.2:         LD      HL,(AP.FIN)     ; 4.B.2.-
                LD      (HL),A          ; 4.B.2.A.-
                INC     HL              ; 4.B.2.B.-
                LD      (AP.FIN),HL
                LD      DE,BUFF + 20    ; 4.B.2.C.-
                OR      A,A             ; Limpia carry
                SBC     HL,DE
                JP      NZ,BUF2.D
                LD      A,0FFH
                LD      (FIN4),A
BUF2.D:         CP      A,00DH          ; 4.B.2.D.-
                JP      NZ,BUFF5
                LD      A,0FFH
                LD      (FIN4),A
BUFF5 :         LD      A,(FIN4)        ; 5.-
                CP      0
                JP      Z,BUFF4
                RET
```

::

PSEUDO-CODIGO 10/IX/82

(EJECOM)

```
Inicio
   1.- Ejecuta "OBTOP"
   2.- Contador de Programa  := Inicio
   3.- Repite
       A.- Ejecuta "INST"
   4.- Hasta RPC = Tope 1  ó  RPC = Tope 2  ó  RPC = Tope 3
Fin
```

.....

```
;            CODIGO                              6/I/83
;                          (EJECOM)

        EJECOM:  CALL    OBTOP          ; 1.-
                 LD      HL,(INICIO)    ; 2.-
                 LD      (RPC),HL         .
        REP    :                        ;3.-
                 CALL    IINST          ;3.A.-
                 LD      DE,(RPC)       ; 4.-
                 LD      HL,(FIN1)
                 OR      A
                 SBC     HL,DE
                 RET     Z
                 LD      HL,(FIN2)
                 OR      A
                 SBC     HL,DE
                 RET     Z
                 LD      HL,(FIN3)
                 OR      A
                 SBC     HL,DE
                 RET     Z
                 JP      REP
```

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                          13/IX/82
;                          (OBTOP)

; Inicio
;    1.- Inicio = RPC
;    2.- FIN1 = Ultima localidad del usuario
;    3.- FIN2 = FIN1
;    4.- FIN3 = FIN1
```

```
;     5.- Obten caracter
;     6.- Si caracter = "/" , entonces:
;         A.- Obten dirección
;         B.- Inicio = Dirección
;         C.- Obten caracter
;         D.- Si caracter = ";" , entonces caracter = ","
;     7.- Clasifica caracter entre
;         A.- "," entonces ejecuta lo siguiente:
;             1.- Fin = Falso
;             2.- Contador = Ø
;             3.- Repite
;                 A.- Obten dirección
;                 B.- Fin (contador) = Dirección
;                 C.- Obten caracter
;                 D.- Contador = Contador + 2
;                 E.- Clasifica caracter entre:
;                     1.- "," si contador = 6 , entonces:
;                         A.- Ejecuta ERROR
;                         B.- Fin:= Verdadero
;                     2.- "CR" ; Fin := Verdadero
;                     3.- Otros ; Ejecuta lo siguiente:
;                         A.- ERROR
;                         B.- Fin := Verdadero
;             4.- Hasta Fin := Verdadero
;         B.- "CR" ; Salirse de rutina
;         C.- Otros ; Ejecuta ERROR
; Fin
;.....
;             CODIGO                          7/I/83
;                              (OBTCP)

        FIN1  :  DS 2
        FIN2  :  DS 2
```

```
FIN3   :  DS 2
INICIO:   DS 2
FINTOP:   DS 1
OBTOP  :  LD      HL,(RPC)      ; 1.-
          LD      (INICIO),HL
          LD      DE,ULN        ; 2.-
          LD      (FIN1),DE
          LD      (FIN2),DE     ; 3.-
          LD      (FIN3),DE     ; 4.-
          CALL    OBTEN         ; 5.-
          CP      A,Ø2EH        ; 6.-
          JP      NZ,PASO7
          CALL    OBTDIR        ; 6.A.-
          LD      HL,(DIR)      ; 6.B.-
          LD      (INICIO),HL
          CALL    OBTEN         ; 6.C.-
          CP      A,Ø3BH        ; 6.D.-
          JP      NZ,PASO7
          LD      A,Ø2CH
PASO7  :                        ; 7.-
          CP      A,Ø2CH        ; 7.A.-
          JP      NZ,PASO7B
          LD      A,Ø           ; 7.A.1.-
          LD      (FINTOP),A
          LD      BC,Ø          ; 7.A.2.-
REP3   :                        ; 7.A.3.-
          CALL    OBTDIR        ; 7.A.3.A.-
          LD      HL,(FIN1)     ; 7.A.3.B.-
          ADD     HL,BC
          LD      DE,(DIR)
          LD      (HL),E
          INC     (HL)
          LD      (HL),D
```

```
          CALL    OBTEN          ; 7.A.3.C.-
          INC     C              ; 7.A.3.D.-
          INC     C
          CP      A,Ø2CH         ; 7.A.3.E.1.-
          JP      NZ,PASOE2
          LD      A,C
          CP      A,6
          JP      NZ,REP3
          CALL    ERROR          ; 7.A.3.E.1.A.-
          LD      A,1            ; 7.A.3.E.1.B.-
          LD      (FINTOP),A
          JP      PASO4
PASOE2:   CP      A,ØØDH         ; 7.A.3.E.2.-
          JP      NZ,PASOE3
          LD      A,1
          LD      (FINTOP),A
          JP      PASO4
PASOE3:                          ; 7.A.3.E.3.-
          CALL    ERROR          ; 7.A.3.E.3.A.-
          LD      A,1            ; 7.A.3.E.3.B.-
          LD      (FINTOP),A
PASO4  :  LD      C,A            ; 7.A.4.-
          LD      A,(FINTOP)
          LD      B,A
          LD      A,C
          CP      A,B
          JP      NZ,REP3
          RET
PASO7B:   CP      A,ØDH          ; 7.B.-
          JP      NZ,ERROR       ; 7.C.-
          RET

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;       PSEUDO-CODIGO                          14/IX/82
;                              (OBTDIR)

; Inicio
;     1.- Fin = Falso
;     2.- Dirección = Ø
;     3.- Repite
;         A.- Obten caracter
;         B.- Si caracter = "Ø"----- "F"
;             1.- Entonces; Valor = Valor * 1ØH  +
;                                   ordinal (caracter)
;             2.- De lo contrario ; Fin = Verdadero
;     4.- Hasta Fin = Verdadero ó Valor > ØFFFH
; Fin
;.....................................................................
;             CODIGO                           10/I/83
;                              (OBTDIR)

        DIR    :  DS 2
        OBTDIR:   LD      A,Ø           ; 1.-
                  LD      (FIN),A
                  LD      HL,Ø          ; 2.-
        OBTD3  :                        ; 3.-
                  CALL    CBTEN         ; 3.A.-
                  CP      A,3ØH
                  JP      M,OD3B2
                  CP      A,3AH
                  JP      P,OBTD3A
                  SUB     A,3ØH
                  JP      OD3B1
        OBTD3A:   CP      A,41H
                  JP      M,OD3B2
                  CP      A,47H
```

```
          JP      P,OBT2
          SUB     A,37H
OD3B1 :   ADD     HL,HL        ; 3.B.1.-
          ADD     HL,HL
          ADD     HL,HL
          ADD     HL,HL
          LD      E,A
          LD      D,Ø
          ADD     HL,DE
          JP      OBTD4
OD3B2 :   LD      (DIR),HL     ; 3.B.2.-
          LD      A,ØFFH
          LD      (FIN),A
          RET
OBTD4 :   LD      A,H          ; 4.-
          AND     A,ØFØH
          JP      Z,OBTD3
          LD      (DIR),HL
          RET
```

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              14/IX/82
;                          (OBTEN)

; Inicio
;    1.- Si el apuntador de inicio <> ó < el Ap. de fin
;        A.- Entonces; Ejecuta lo siguiente:
;            1.- Caracter = Buffer (Ap. de Inicio)
;            2.- Ap. de Inicio = Ap. de Inicio + 1
;        B.- De lo contrario; Caracter = "?"
; Fin
;................................................................
```

```
;            CODIGO                          12/I/83
;                          (OBTEN)

       OBTEN :  LD      HL,(AP.INI)  ; 1.-
                LD      DE,(AP.FIN)
                OR      A
                SBC     HL,DE
                JP      NC,PASO1B
                LD      HL,(AP.INI)
                LD      A,(HL)       ; 1.A.-
                INC     HL
                LD      (AP.INI),HL
                RET
       PASO1B:  LD      A,'?'        1.B.-
                RET


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                          17/IX/82
;                          (IINST)

; Inicio
;    1.- Codigo de la instrucción = Memoria (RPC)
;    2.- RPC = RPC + 1
;    3.- NODEF = Falso
;    4.- Clasifica el codigo de la instrucción entre:
;        A.- ØØØØXXXX , Ø1XXXXXX : Ejecuta "GRUPOØ
;        B.- ØØØ1XXXX : Ejecuta "GRUPO1"
;        C.- ØØ1ØXXXX : Ejecuta "GRUPO2"
;        D.- ØØ11XXXX : Ejecuta "GRUPO3"
;        E.- 1XXX11XX , 1XXXØØ11 : Ejecuta "GRUPO4"
;        F.- Otros : Ejecuta "GRUPO5"
; Fin
;.....
```

CODIGO 4/III/83

(IINST)

```
CODINS:  DS 1
CNODEF:  DS 1
IINST :  LD      DE,(RPC)      ; 1.-
         LD      A,(DE)        ; Codigo en el reg. "A"
         LD      (CODINS),A    ; y en CODINS .
         INC     DE            ; 2.-
         LD      (RPC),DE
         LD      B,Ø           ; 3.-
         LD      (CNODEF),B
         BIT     7,A           ; 4.-
         JP      NZ,II4E       ; 4.A.-
         BIT     6,A
         JP      NZ,GRUPOØ
         BIT     5,A
         JP      NZ,II4C
         BIT     4,A
         JP      Z,GRUPOØ
         JP      GRUPO1        ; 4.B.-
II4C  :  BIT     4,A
         JP      Z,GRUPO2      ; 4.C.-
         JP      GRUPO3        ; 4.D.-
II4E  :  BIT     3,A           ; 4.E.- y 4.F.-
         JP      Z,II4F
         BIT     2,A
         JP      Z,GRUPO5
         JP      GRUPO4
II4F  :  BIT     2,A
         JP      NZ,GRUPO5
         BIT     1,A
         JP      Z,GRUPO5
         BIT     Ø,A
         JP      Z,GRUPO5
         JP      GRUPO4
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;       PSEUDO-CODIGO                                   20/IX/82
;                                   (GRUPCØ)

; Inicio
;    1.- Clasifica codigo de inst. entre (direccionamientos)
;        A.- ØØØØXXXX  : Ejecuta "DIRECT"
;        B.- Ø1ØØXXXX  : Ejecuta "ACA"
;        C.- Ø1Ø1XXXX  : Ejecuta "ACB"
;        D.- Ø11ØXXXX  : Ejecuta "INDEX"
;        E.- Ø111XXXX  : Ejecuta "EXTEN"
;    2.- Si UNODEF = FALSO , Entonces:
;        A.- Clasifica el Cod. de Inst. entre (instrucción)
;            1.- XXXXØØØØ : Ejecuta "NEG"
;            2.- XXXXØØ11 : Ejecuta "COM"
;            3.- XXXXØ1ØØ : Ejecuta "LSR"
;            4.- XXXXØ11Ø : Ejecuta "ROR"
;            5.- XXXXØ111 : Ejecuta "ASR"
;            6.- XXXX1ØØØ : Ejecuta "A(L)SL"
;            7.- XXXX1ØØ1 : Ejecuta "ROL"
;            8.- XXXX1Ø1Ø : Ejecuta "DEC"
;            9.- XXXX11ØØ : Ejecuta "INC"
;           10.- XXXX11Ø1 : Ejecuta "TST"
;           11.- XXXX111Ø : Ejecuta "JMP"
;           12.- XXXX1111 : Ejecuta "CLR"
;           13.- Otros : Ejecuta "NODEF"
; Fin
;.....
;            CODIGO                                 6/III/83
;                                   (GRUPCØ)

       GRUPCØ:  BIT     6,A          ; 1.-
                JP      NZ,GRUØ1B
                CALL    DIRECT       ; 1.A.-
                JP      GRUØ2
```

```
GRUØ1B:  BIT     5,A          ; 1.B.-
         JP      NZ,GRUØ1D
         BIT     4,A
         JP      NZ,GRUØ1C
         CALL    ACA
         JP      GRUØ2
GRUØ1C:  CALL    ACB .        ;;1.C.-
         JP      GRUØ2
GRUØ1D:  BIT     4,A          ; 1.D.-
         JP      NZ,GRUØ1E
         CALL    INDEX
         JP      GRUØ2
GRUØ1E:  CALL    EXTEN        ; 1.E.-
GRUØ2 :  LD      C,A          ; 2.-
         LD      B,Ø
         LD      A,(CNODEF)
         CP      A,B
         JP      NZ,GØ2A13
         LD      A,C          ; 2.A.-
         BIT     3,A          ; 2.A.1.-
         JP      NZ,GØ2A6
         BIT     2,A.
         JP      NZ,GØ2A3
         BIT     1,A
         JP      NZ,GØ2A2
         BIT     Ø,A
         JP      NZ,GØ2A13
         JP      NEG
GØ2A2 :  BIT     Ø,A          ; 2.A.2.-
         JP      Z,GØ2A13
         JP      COM
GØ2A3 :  BIT     1,A          ; 2.A.3.-
         JP      NZ,GØ2A4
         BIT     Ø,A
```

```
                JP      NZ,GØ2A13
                JP      LSR
        GØ2A4 : BIT     Ø,A             ; 2.A.4.-
                JP      Z,ROR
                JP      ASR             ; 2.A.5.-
        GØ2A6 : BIT     2,A             ; 2.A.6.-
                JP      NZ,GØ2A9
                BIT     1,A
                JP      NZ,GØ2A8
                BIT     Ø,A
                JP      Z,A(L)SL
                JP      ROL             ; 2.A.7.-
        GØ2A8 : BIT     Ø,A             ; 2.A.8.-
                JP      NZ,GØ2A13
                JP      DEC
        GØ2A9 : BIT     1,A             ; 2.A.9.-
                JP      NZ,GØ2A11
                BIT     Ø,A
                JP      Z,INC
                JP      TST             ; 2.A.10.-
        GØ2A11: BIT     Ø,A             ; 2.A.11.-
                JP      Z,JMP
                JP      CLR             ; 2.A.12.-
        GØ2A13: JP      NODEF           ; 2.A.13.-


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           20/IX/82
;                               (DIRECT)

; Inicio
;    1.- Dirección efective = Mem.(DP),Mem.(RPC)
;    2.- RPC := RPC + 1
; Fin
```

```
;           CODIGO                                5/III/83
;                              (DIRECT)

     DIREF  :  DS 2
     DIRECT:   LD      A,(DP)          ; 1.-
               LD      B,A
               LD      DE,(RPC)
               LD      A,(DE)
               LD      C,A
               LD      (DIREF),BC
               INC     DE
               LD      (RPC),DE
               RET

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                            20/IX/82
;                              (ACA)

; Inicio
;    1.- Dirección efectiva = Acumulador "A"
; Fin
;-------------------------------------------------------------
;           CODIGO                               5/III/83
;                              (ACA)

     ACA    :  LD      DE,ACCA         ; 1.-
               LD      (DIREF),DE
               RET

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;       PSEUDO-CODIGO                                   20/IX/82
;                                   (ACB)

; Inicio
;    1.- Dirección efectiva = Acumulador "B"
; Fin
;-----------------------------------------------------------------------
;              CODIGO                                   5/III/83
;                                   (ACB)

       ACB    :  LD       DE,ACCB       ; 1.-
                 LD       (DIREF),DE
                 RET


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; PSEUDO-CODIGO                                         21/IX/82
;                               (INDEX)

; Inicio
;    1.- Postbyte = Memoria (RPC)
;    2.- RPC := RPC + 1
;    3.- Clasifica el postbyte entre :
;        A.- XØØXXXXX : Registro := Indice "X"
;        B.- XØ1XXXXX : Registro := Indice "Y"
;        C.- X1ØXXXXX : Registro := Stack "U"
;        D.- X11XXXXX : Registro := Stack "S"
;    4.- Si postbyte = ØXXXXXXX
;        A.- Entonces:
;            1.- Ejecuta "OFSET5"
;        B.- De lo contrario:
;            1.- Clasifica postbyte entre:
;               A.- XXXXØØØØ : Ejecuta "OPEI"
```

```
                B.- XXXXØØØ1 : Ejecuta "INC+2"
                C.- XXXXØØ1Ø : Ejecuta "OPEII"
                D.- XXXXØØ11 : Ejecuta "DEC-2"
                E.- XXXXØ1ØØ : Ejecuta "SINCOR"
                F.- XXXXØ1Ø1 : Ejecuta "REGB"
                G.- XXXXØ11Ø : Ejecuta "REGA"
                H.- XXXX1ØØØ : Ejecuta "OFST8"
                I.- XXXX1ØØ1 : Ejecuta "OFST16"
                J.- XXXX1Ø11 : Ejecuta "REGD"
                K.- XXXX11ØØ : Ejecuta "PCOF8"
                L.- XXXX11Ø1 : Ejecuta "PCOF16"
                M.- XXXX1111 : Ejecuta "OPEIII"
                N.- Otros:  Ejecuta "NODEF"
            2.- Si postbyte = XXX1XXXX
                A.- Entonces:
                    1.- Dir. ef. := Memoria (Dir. ef.)
; Fin
;.....................................................................
;           CODIGO                              7/III/83
;                           (INDEX)

      REG.: DS 2
      INDEX : LD      DE,(RPC)      ; 1.-
              LD      A,(DE)        ; Postbyte en reg. "A"
              INC     DE            ; 2.-
              LD      (RPC),DE
              BIT     6,A           ; 3.-
              JP      NZ,IND3C
              BIT     5,A
              JP      NZ,IND3B
              LD      DE,RIX        ; 3.A.-
              LD      (REG.),DE
              JP      IND4
```

```
IND3B :  LD      DE,RIY          ; 3.B.-
         LD      (REG.),DE
         JP      IND4
IND3C :  BIT     5,A             ; 3.C.-
         JP      NZ,IND3D
         LD      DE,US
         LD      (REG.),DE
         JP      IND4
IND3D :  LD      DE,RSP          ; 3.D.-
         LD      (REG.),DE
IND4  :  BIT     7,A             ; 4.-
         JP      NZ,IND4B        ; 4.A.-
         JP      OFSET5          ; 4.A.1.-
IND4B :  PUSH    AF              ; 4.B.-
         BIT     3,A             ; 4.B.1.-
         JP      NZ,IX4B1H
         BIT     2,A
         JP      NZ,IX4B1E
         BIT     1,A
         JP      NZ,IX4B1C
         BIT     Ø,A
         JP      NZ,IX4B1B
         CALL    OPEI            ; 4.B.1.A.-
         JP      IX4B2
IX4B1B:  CALL    INC+2           ; 4.B.1.B.-
         JP      IX4B2
IX4B1C:  BIT     Ø,A             ; 4.B.1.C.-
         JP      NZ,IX4B1D
         CALL    OPEII
         JP      IX4B2
IX4B1D:  CALL    DEC-2           ; 4.B.1.D.-
         JP      IX4B2
IX4B1E:  BIT     1,A             ; 4.B.1.E.-
```

```
                JP      NZ,IX4B1G
                BIT     Ø,A
                JP      NZ,IX4B1F
                CALL    SINCOR
                JP      IX4B2
IX4B1F:         CALL    REGB            ; 4.B.1.F.-
                JP      IX4B2
IX4B1G:         BIT     Ø,A             ; 4.B.1.G.-
                JP      NZ,IX4B1N
                CALL    REGA
                JP      IX4B2
IX4B1H:         BIT     2,A             ; 4.B.1.H.-
                JP      IX4B1K
                BIT     1,A
                JP      NZ,IX4B1J
                BIT     Ø,A
                JP      NZ,IX4B1I
                CALL    OFST8
                JP      IX4B2
IX4B1I:         CALL    OFST16          ; 4.B.1.I.-
                JP      IX4B2
IX4B1J:         BIT     Ø,A             ; 4.B.1.J.-
                JP      Z,IX4B1N
                CALL    REGD
                JP      IX4B2
IX4B1K:         BIT     1,A             ; 4.B.1.K.-
                JP      NZ,IX4B1M
                BIT     Ø,A
                JP      NZ,IX4B1L
                CALL    PCOF8
                JP      IX4B2
IX4B1L:         CALL    PCOF16          ; 4.B.1.L.-
                JP      IX4B2
```

```
IX4B1M:  BIT     Ø,A            ; 4.B.1.M.-
         JP      Z,IX4B1N
         CALL    OPEIII
         JP      IX4B2
IX4B1N:  JP      NODEF          ; 4.B.1.N.-
IX4B2 :  LD      B,Ø1ØH         ; 4.B.2.-
         POP     AF
         AND     A,B
         JP      NZ,IX4B2A
         RET
IX4B2A:                         ; 4.B.2.A.-
         LD      DE,(DIREF)     ; 4.B.2.A.1.-
         LD      A,(DE)
         LD      C,A
         INC     DE
         LD      A,(DE)
         LD      B,A
         LD      (DIREF),BC
         RET
```

```
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              21/IX/82
;                              (OFSET5)

; Inicio
;    1.- Postbyte := Postbyte  and ØØØ11111
;    2.- Si postbyte = XXXØXXXX
;        A.- Entonces ;  Dirección ef. = Mem. (Registro) +
;                                        ( Ø,POSTBYTE )
;        B.- De lo contrario ; Dir. ef.= Mem. (Registro) +
;                                        ( ØFFH,POSTBYTE )
; Fin
;.....:.....
```

```
;            CODIGO                              7/III/83
;                          (OFSET5)

      OFSET5:  AND     A,Ø1FH          ; 1.- Postbyte en A
               BIT     4,A             ; 2.-
               JP      NZ,OF52B
               LD      DE,(REG.)       ; 2.A.-
               LD      B,Ø
               LD      C,A
               JP      OFST5
      OF52B :  LD      DE,(REG.)       ; 2.B.-
               LD      B,ØFFH
               OR      A,ØEØH
               LD      C,A
      OFST5 :  LD      A,(DE)
               LD      L,A
               INC     DE
               LD      A,(DE)
               LD      H,A
               ADD     HL,BC
               LD      (DIREF),DE
               RET
```

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              22/IX/82
;                          (OPEI)

; Inicio
;    1.- Si postbyte = XXXØXXXX , entonces:
;        A.- Dirección efectiva = Memoria (Registro)
;        B.- Memoria (Registro) = Memoria (Registro) + 1
;    2.- De lo contrario: Ejecuta rutina "NODEF"
; Fin
;.....
```

CODIGO 8/III/83

(OPEI)

```
OPEI  :  AND     A,ØlØH        ; 1.-
         JP      NZ,NODEF      ; 2.-
         LD      DE,(REG.)     ; 1.A.-
         LD      A,(DE)
         LD      C,A
         INC     DE
         LD      A,(DE)
         LD      B,A
         LD      (DIREF),BC
         INC     BC            ; 1.B.-
         LD      A,B
         LD      (DE),A
         DEC     DE
         LD      A,C
         LD      (DE),A
         RET
```

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              24/IX/82
;                              (OPEII)

; Inicio
;    1.- Si postbyte = XXXØXXXX , entonces:
;        A.- Memoria (Registro) = Memoria (Registro) - 1
;        B.- Dirección efectiva = Memoria (Registro)
;    2.- De lo contrario: Ejecuta rutina "NODEF"
; Fin
;..............................................................
```

```
;            CODIGO                              8/III/83
;                          (OPEII)

       OPEII : AND      A,Ø1ØH        ; 1.-
               JP       NZ,NODEF      ; 2.-
               LD       DE,(REG.)     ; 1.A.-
               LD       A,(DE)
               LD       L,A
               INC      DE
               LD       A,(DE)
               LD       H,A
               DEC      HL
               LD       A,H
               LD       (DE),A
               LD       A,L
               DEC      DE
               LD       (DE),A
               LD       (DIREF),HL    ; 1.B.-
               RET


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDOCODIGO                              27/IX/82
;                          (OPEIII)

; Inicio
;    1.- Si postbyte = XØØ1XXXX , entonces :
;        A.- Dirección ef. = Mem.(RPC + 1) , Mem.(RPC)
;        B.- RPC := RPC + 2
;    2.- De lo contrario:
;        A.- Ejecuta "NODEF"
; Fin
;.....................................................
```

```
;            CODIGO                              8/III/83
;                              (OPEIII)

       OPEIII:  AND      A,Ø1ØH          ; 1.-
                JP       Z,NODEF         ; 2.- y 2.B.-
                LD       DE,(RPC)        ; 1.A.-
                LD       A,(DE)
                LD       C,A
                INC      DE
                LD       A,(DE)
                LD       B,A
                LD       (DIREF),BC
                INC      BC              ; 1.B.-
                INC      BC
                LD       A,B
                LD       (DE),A
                DEC      DE
                LD       A,C
                LD       (DE),A
                RET


;;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                            16/XII/82
;                              (NODEF)

; Inicio
;     1.- TOPE1 := RPC
;     2.- Número de instrucciones := 1
;     3.- Mensaje ( 1:54) := '      '
;     4.- Mensaje ( 8:25) := "CODIGO NO DEFINIDO"
;     5.- Mensaje (36:46) := 'PC (HEX) :='
;     6.- Mensaje (48:54) := ASCII (mem.(RPC)) 'CR' 'LF' '$'
```

```
;     7.- Imprime mensaje (1:54)
; Fin
;.....................................................
;            CODIGO                          10/III/83
;                          (NODEF)

       CONODE:  DB 'CODIGO NO DEFINIDO'
       PCH   :  DB 'PC (HEX) :='
       NODEF :  LD      DE,RPC        ; 1.-
                LD      (FIN1),DE
                                       ; 2.-
                LD      A,Ø2ØH        ; 3.-
                LD      HL,BUFSIS
                LD      B,54
       NODEF3:  LD      (HL),A
                INC     HL
                DJNZ,   NODEF3
                LD      DE,BUFSIS + 7; 4.-
                LD      HL,CONODE
                LD      BC,18
                LDIR
                LD      DE,BUFSIS + 35  ; 5.-
                LD      HL,PCH
                LD      BC,11
                LDIR
                ASC16   BUFSIS + 47,RPC ; 6.-
                LD      DE,BUFSIS + 51
                LD      A,CR
                LD      (DE),A
                INC     DE
                LD      A,LF
                LD      (DE),A
                INC     DE
```

```
            LD      A,PESOS
            LD      (DE),A
            LD      C,IMPRIM      ; 7.-
            LD      DE,BUFSIS
            CALL    CDOS
            RET


;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;     PSEUDO-CODIGO                           30/IX/82
;                         (INC+2)

; Inicio
;    1.- Dirección efectiva = Memoria (Registro)
;    2.- Memoria (Registro) := Memoria (Registro) + 2
; Fin
;............................................................
;           CODIGO                            10/III/83
;                         (INC+2)

   INC+2 :  LD      DE,(REG.)     ; 1.-
            LD      A,(DE)
            LD      C,A
            INC     DE
            LD      A,(DE)
            LD      B,A
            LD      (DIREF),BC
            INC     BC            ; 2.-
            INC     BC
            LD      A,B
            LD      (DE),A
            DEC     DE
            LD      A,C
```

```
            LD      (DE),A
            RET


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;     PSEUDOCODIGO                              30/IX/82
;                         (DEC-2)

; Inicio
;    1.- Memoria (Registro) := Memoria (Registro) - 2
;    2.- Dirección efectiva = Memoria (Registro)
; Fin
;.............................................................
;           CODIGO                              10/III/83
;                         (DEC-2)

    DEC-2 : LD      DE,(REG.)     ; 1.-
            LD      A,(DE)
            LD      C,A
            INC     DE
            LD      A,(DE)
            LD      B,A
            DEC     BC
            DEC     BC
            LD      (DE),B
            DEC     DE
            LD      (DE),C
            LD      (DIREF),BC    ; 2.-
            RET


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;      PSEUDO-CODIGO                          30/IX/82
;                            (SINCOR)

; Inicio
;    1.- Dirección efectiva = Memoria (Registro)
; Fin
;.....................................................
;            CODIGO                          11/III/83
;                            (SINCOR)

       SINCOR:  LD      DE,(REG.)     ; 1.-
                LD      A,(DE)
                LD      C,A
                INC     DE
                LD      A,(DE)
                LD      B,A
                LD      (DIREF),BC
                RET


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                          30/IX/82
;                             (REGB)

; Inicio
;    1.- Si Memoria (acumulador B) = ØXXXXXXX
;        A.- Entonces: Dir. efec. = Memoria (Registro) +
;                                   ( Ø , Memoria (ACCB))
;        B.- De lo contrario: Dir. ef. = Memoria (Registro)
;                                   + ( ØFFH, Mem.(ACCB))
; Fin
;.....................................................
```

```
;           CODIGO                                    11/III/83
;                              (REGB)

REGB   :  LD      DE,(REG.)      ; 1.-
          LD      A,(DE)
          LD      L,A
          INC     DE
          LD      A,(DE)
          LD      H,A
          LD      B,ØFFH         ; 1.B.-
          LD      A,(ACCB)
          LD      C,A
          BIT     7,A
          JP      NZ,REGB1B
          LD      B,Ø            ; 1.A.-
REGB1B:   ADD     HL,BC
          LD      (DIREF),HL
          RET

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              30/IX/82
;                              (REGA)

; Inicio
;    1.- Si memoria (ACCA) = ØXXXXXXX
;        A.- Entonces: Dir. efec. = Memoria (Registro) +
;                                   ( Ø , Memoria (ACCA) )
;        B.- De lo contrario: Dir. ef. = Memoria (Registro)
;                                   + ( ØFFH, Mem.(ACCA) )
; Fin

;..................................................................
```

```
;           CODIGO                              11/III/83
;                              (REGA)

     REGA   :  LD       DE,(REG.)      ; 1.-
               LD       A,(DE)
               LD       L,A
               INC      DE
               LD       A,(DE)
               LD       H,A
               LD       B,ØFFH         ; 1.B.-
               LD       A,(ACCA)
               LD       C,A
               BIT      7,A
               JP       NZ,REGA1B
               LD       B,Ø            ; 1.A.-
     REGA1B:   ADD      HL,BC
               LD       (DIREF),HL
               RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              30/IX/82
;                           (OFST8)

; Inicio
;    1.- Byte = Memoria (RPC)
;    2.- RPC := RPC + 1
;    3.- Si byte = ØXXXXXXX
;        A.- Entonces: Dir. efec. = Memoria (Registro) +
;                                   ( Ø , Byte )
;        B.- De lo contrario: Dir. ef. = Memoria (Registro)
;                                   + ( ØFFH , Byte )
; Fin
;.............................................................
```

```
;            CODIGO                              12/III/83
;                          (OFST8)

       OFST8 :  LD      DE,(RPC)      ; 1.-
                LD      A,(DE)
                LD      B,ØFFH
                LD      C,A
                INC     DE            ; 2.-
                LD      (RPC),DE
                BIT     7,C           ; 3.-
                JP      NZ,OFST83
                LD      B,Ø
       OFST83:  LD      DE,(REG.)     ; 3.A.- y 3.B.-
                LD      A,(DE)
                LD      L,A
                INC     DE
                LD      A,(DE)
                LD      H,A
                ADD     HL,BC
                LD      (DIREF),HL
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                          30/IX/82
;                          (OFST16)

; Inicio
;    1.- Dir. efec. = Memoria (Registro)  ±
;                     ( Mem. (RPC + 1) , Mem. (RPC) )
;    2.- RPC := RPC + 2
; Fin
;.....................................................
```

```
;            CODIGO                                   16/III/83
;                              (OFST16)

      OFST16:  LD        DE,(REG.)       ; 1.-
               LD        A,(DE)
               LD        L,A
               INC       DE
               LD        A,(DE)
               LD        H,A
               LD        DE,(RPC)
               LD        A,(DE)
               LD        C,A
               INC       DE              ; RPC + 1
               LD        A,(DE)
               LD        B,A
               INC       DE              ; RPC + 1
               LD        (RPC),DE        ; 2.-
               ADD       HL,BC
               LD        (DIREF),HL
               RET

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                               30/IX/82
;                                 (REGD)

; Inicio
;    1.- Dirección efectiva = Memoria (Registro) ±
;                             Mem. (ACCA) , Mem. (ACCB)
; Fin
;............................................................
;            CODIGO                                   17/III/83
;                               (REGD)

      REGD  :  LD.       DE,(REG.)       1.-
```

```
                LD      A,(DE)
                LD      L,A
                INC     DE
                LD      A,(DE)
                LD      H,A
                LD      BC,ACCA
                LD      A,(BC)
                LD      D,A
                LD      BC,ACCB
                LD      A,(BC)
                LD      E,A
                ADD     HL,DE
                LD      (DIREF),HL
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           30/IX/82
;                           (PCOF8)

; Inicio
;    1.- Byte = Memoria (RPC)
;    2.- RPC := RPC + 1
;    3.- Si byte = ØXXXXXXX
;        A.- Entonces; Dirección efectiva = RPC - 2  +
;                                            ( Ø , Byte )
;        B.- De lo contrario; Dir. efec. = RPC - 2 +
;                                            ( ØFFH , BYTE )
; Fin
;.............................................................
;           CODIGO                            18/III/83
;                           (PCOF8)

       PCOF8 :  LD      HL,(RPC)      ; 1.-
                LD      A,(HL)
```

```
                INC     HL              ; 2.-
                LD      (RPC),HL
                LD      B,Ø
                LD      C,A
                BIT     7,A             ; 3.-
                JP      Z,PCOF8B
                LD      B,ØFFH
        PCOF8B: DEC     HL              ; 3.A.- y 3.B.-
                DEC     HL
                ADD     HL,BC
                LD      (DIREF),HL
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           30/IX/82
;                               (PCOF16)

; Inicio
;    1.- Dirección efectiva = RPC ± ( Memoria (RPC + 1),
;                                     Memoria ( RPC ) )
;    2.- RPC := RPC + 2
; Fin
;.......................................................
;            CODIGO                             18/III/83
;                               (PCOF16)

        PCOF16: LD      HL,(RPC)        ; 1.-
                LD      A,(HL)
                LD      C,A
                INC     HL
                LD      A,(HL)
                LD      B,A
                DEC     HL
```

```
                ADD     HL,BC
                LD      (DIREF),HL
                INC     HL              ; 2.-
                INC     HL
                LD      (RPC),HL
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           30/IX/82
;                               (EXTEN)

; Inicio
;    1.- Dir. efec. = Mem. (RPC + 1) , Mem. (RPC)
;    2.- RPC := RPC + 2
; Fin
;...............................................................
;             CODIGO                            19/III/83
;                               (EXTEN)

       EXTEN :  LD      DE,(RPC)       ; 1.-
                LD      A,(DE)
                LD      C,A
                INC     DE              ; RPC + 1
                LD      A,(DE)
                LD      B,A
                LD      (DIREF),BC
                INC     DE              ; RPC + 1
                LD      (RPC),DE        ; 2.-
                RET


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;       PSEUDO-CODIGO                                   1/X/82
;                               (NEG)

; Inicio
;     1.- Negar Memoria (Dirección efectiva)
;     2.- Banderas 6809 = Banderas Z-80 .
;           (H,N,Z,V y C)    (H,S,Z,V y C ; respectivamente)
; Fin
;.....
;             CODIGO                                    30/IX/83
;                               (NEG)

        NEG    :  LD      DE,(DIREF)    ; 1.-
                  LD      A,(DE)
                  NEG     A
                  LD      (DE),A
                  PUSH    AF            ; 2.-
                  LD      A,(CC)
                  AND     A,ØDØH
                  POP     BC
                  BIT     Ø,C
                  JP      Z,NEG1
                  SET     Ø,A
        NEG1   :  BIT     2,C
                  JP      Z,NEG2
                  SET     1,A
        NEG2   :  BIT     6,A
                  JP      Z,NEG3
                  SET     2,A
        NEG3   :  BIT     4,C
                  JP      Z,NEG4
                  SET     5,A
        NEG4   :  BIT     7,C
                  JP      Z,NEG5
```

```
                SET     3,A
        NEG5  : LD      (CC),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           1/X/82
;                               (COM)

; Inicio
;    1.- Complementar Memoria (Dirección efectiva)
;    2.- Bandera (C) := 1
;    3.- Bandera (B) := Ø
;    4.- Si Bit 7 de Memoria (Dir. efec.) := Ø
;        A.- Entonces; Bandera (N) := Ø
;        B.- De lo contrario; Bandera (N) := 1
;    5.- Si Memoria (Dir. efec.) := Ø
;        A.- Entonces; Bandera (Z) := 1
;        B.- De lo contrario; Bandera (Z) := Ø
; Fin
;...............................................................
;             CODIGO                            30/IX/83
;                               (COM)

        COM   : LD      DE,(DIREF)    ; 1.-
                LD      A,(DE)
                CPL
                LD      (DE),A
                LD      A,(CC)        ; 2.-
                SET     Ø,A
                RES     1,A           ; 3.-
                LD      (CC),A
                LD      A,(DE)        ; 4.-
                LD      B,A
                LD      A,(CC)
```

```
                BIT     7,B
                JP      Z,COM1
                SET     3,A
                JP      COM2
        COM1  : RES     3,A
        COM2  : LD      C,A             ; 5.-
                LD      A,B
                AND     A,ØFFH          ; 5.A.-
                JP      Z,COM3
                RES     2,C             ; 5.B.-
                JP      COM4
        COM3  : SET     2,C
        COM4  : LD      A,C
                LD      (CC),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   1/X/82
;                               (LSR)

; Inicio
;    1.- Corrimiento lógico a la derecha Mem.(Dir. efec)
;    2.- BANDERAS 6809 (Z,C) = BANDERAS Z-80 (Z,C)
;    3.- Bandera 6809 (N) := Ø
; Fin
;.....................................................
;            CODIGO                                 30/IX/83
;                               (LSR)

        LSR   : LD      DE,(DIREF)      ; 1.-
                LD      A,(DE)
                SRL     A
                LD      (DE),A          ; 2.-
                PUSH    AF
```

```
              LD      A,(CC)
              AND     A,ØFAH
              POP     BC
              BIT     Ø,C
              JP      Z,LSR1
              SET     Ø,A
LSR1     :    BIT     6,C
              JP      Z,LSR2
              SET     2,A
LSR2     :    RES     3,A          ; ?
              LD      (CC),A
              RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;        PSEUDO-CODIGO                          1/X/82
;                           (ROR)

; Inicio
;    1.- Rotación a la derecha de Mem.(Dir. efec.)
;    2.- Banderas 6809 (N,Z y C) = Banderas Z-80 (S,Z y C)
; Fin
;...............................................................
;             CODIGO                            30/IX/83
;                           (ROR)

ROR      :    LD      DE,(DIREF)   ; 1.-
              LD      A,(DE)
              RR      A
              LD      (DE),A
              PUSH    AF           ; 2.-
              LD      A,(CC)
              AND     A,ØF2H
              POP     BC
              BIT     Ø,C
```

```
                JP      Z,ROR1
                SET     Ø,A
        ROR1  : BIT     6,C
                JP      Z,ROR2
                SET     2,A
        ROR2  : BIT     7,C
                JP      Z,ROR3
                SET     3,A
        ROR3  : LD      (CC),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           1/X/82
;                               (ASR)

; Inicio
;    1.- Corrimiento aritmetico a la derecha de memoria
;        (Direcciôn efectiva)
;    2.- Banderas 6809 (N,Z y C) = Banderas Z-80 (S,Z y C)
; Fin
;.............................................................
;             CODIGO                            30/IX/83
;                               (ASR)

        ASR   : LD      DE,(DIREF)    ; 1.-
                LD      A,(DE)
                SRA     A
                LD      (DE),A
                PUSH    AF            ; 2.-
                LD      A,(CC)
                AND     A,ØF2H
                POP     BC
                BIT     Ø,C
                JP      Z,ASR1
```

```
                SET     Ø,A
        ASR1  : BIT     6,C
                JP      Z,ASR2
                SET     2,A
        ASR2  : BIT     7,C
                JP      Z,ASR3
                SET     3,A
        ASR3  : LD      (CC),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           1/X/82
;                           (A(L)SL)

; Inicio
;    1.- Si Bit 7 ⊕ Bit 6 de Mem.(Dir. efec.) = 1
;        A.- Bandera (V) := 1
;    2.- Corrimiento aritmetico (lógico) a la izquierda
;        de Memoria (Dirección efectiva)
;    3.- Banderas 6809 (N,Z y C) = Banderas Z-80 (S,Z y C)
; Fin
;.............................................................
;             CODIGO                            30/IX/83
;                           (A(L)SL)

        A(L)SL: LD      DE,(DIREF)    ; 1.-
                LD      A,(DE)
                LD      B,A
                AND     A,8ØH
                LD      C,A
                LD      A,B
                AND     A,4ØH
                SLA     A
                XOR     A,C
```

```
                JP      Z,ASL1
                LD      A,(CC)          ; 1.A.-
                SET     1,A
                LD      (CC),A
        ASL1  : LD      A,(DE)          ; 2.-
                SLA     A
                LD      (DE),A
                PUSH    AF              ; 3.-
                LD      A,(CC)
                AND     A,ØF2H
                POP     BC
                BIT     Ø,C
                JP      Z,ASL2
                SET     Ø,A
        ASL2  : BIT     6,C
                JP      Z,ASL3
                SET     2,A
        ASL3  : BIT     7,C
                JP      Z,ASL4
                SET     3,A
        ASL4  : LD      (CC),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           1/X/82
;                               (ROL)

; Inicio
;    1.- Si Bit 7 ⊕ Bit 6 de Mem. (Dir. efec.) = 1
;        A.- Bandera (V) := 1
;    2.- Rotación a la izquierda de Mem. (Dir. efec.)
;    3.- Banderas 6809 (N,Z y C) = Banderas Z-80 (S,Z y C)
; Fin
;.....................................................
```

```
;              CODIGO                              30/IX/83
;                                (ROL)

        ROL   :  LD      DE,(DIREF)    ; 1.-
                 LD      A,(DE)
                 LD      B,A
                 AND     A,8ØH
                 LD      C,A
                 LD      A,B
                 SLA     A
                 XOR     A,C
                 JP      Z,ROL1
                 LD      A,(CC)        ; 1.A.-
                 SET     1,A
                 LD      (CC),A
        ROL1  :  LD      A,(DE)        ; 2.-
                 RL      A
                 LD      (DE),A
                 PUSH    AF            ; 3.-
                 LD      A,(CC)
                 AND     A,ØF2H
                 POP     BC
                 BIT     Ø,C
                 JP      Z,ROL2
                 SET     Ø,A
        ROL2  :  BIT     6,C
                 JP      Z,ROL3
                 SET     2,A
        ROL3  :  BIT     7,C
                 JP      Z,ROL4
                 SET     3,A
        ROL4  :  LD      (CC),A
                 RET
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::.
```

```
;       PSEUDO-CODIGO                                   1/X/82
;                               (DEC)

; Inicio
;    1.- Decremento (-1) de Memoria (Dir. efec.)
;    2.- Banderas 6809 (N,Z y V) = Banderas Z-80 (S,Z y V)
; Fin
;..........................................................
;            CODIGO                                     30/IX/83
;                               (DEC)

        DEC    :  LD      DE,(DIREF)    ; 1.-
                  LD      A,(DE)
                  DEC     A
                  LD      (DE),A
                  PUSH    AF            ; 2.-
                  LD      A,(CC)
                  AND     A,ØF1H
                  POP     BC
                  BIT     2,C
                  JP      Z,DEC1
                  SET     1,A
        DEC1   :  BIT     6,C
                  JP      Z,DEC2
                  SET     2,A
        DEC2   :  BIT     7,C
                  JP      Z,DEC3
                  SET     3,A
        DEC3   :  LD      (CC),A
                  RET

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;       PSEUDO-CODIGO                          1/X/82
;                              (INC)

; Inicio
;     1.- Incremento (+1) de Memoria (Dir. efec.)
;     2.- Banderas 6809 (N,Z y V) = Banderas Z80 (S,Z y V)
; Fin
;.....................................................
;             CODIGO                           30/IX/83
;                              (INC)

        INC    :  LD      DE,(DIREF)    ; 1.-
                  LD      A,(DE)
                  INC     A
                  LD      (DE),A
                  PUSH    AF            ; 2.-
                  LD      A,(CC)
                  AND     A,ØF1H
                  POP     BC
                  BIT     2,C
                  JP      Z,INC1
                  SET     1,A
        INC1   :  BIT     6,C
                  JP      Z,INC2
                  SET     2,A
        INC2   :  BIT     7,C
                  JP      Z,INC3
                  SET     3,A
        INC3   :  LD      (CC),A
                  RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;      PSEUDO-CODIGO                              1/X/82
;                              (TST)

; Inicio
;    1.- Bandera (V) := Ø
;    2.- Si Bit 7 de Memoria (Dir. efec.) = 1
;        A.- Entonces: Bandera (N) := 1
;    3.- Si Memoria (Dirección efectiva) = Ø
;        A.- Entonces: Bandera (Z) := 1
; Fin
;.....................................................
;            CODIGO                              30/IX/83
;                              (TST)

       TST   :  LD      A,(CC)        ; 1.-
                RES     1,A
                LD      DE,(DIREF)    ; 2.-
                LD      B,A
                LD      A,(DE)
                BIT     7,A
                JP      Z,TST1
                LD      A,B           ; 2.A.-
                SET     3,A
                LD      B,A
       TST1  :  LD      A,(DE)
                AND     A,ØFFH
                JP      NZ,TST2
                LD      A,B           ; 3.A.-
                SET     2,A
       TST2  :  LD      (CC),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;      PSEUDO-CODIGO                              1/X/82
;                              (JMP)

; Inicio
;    1.- Si codigo de instrucción = X1ØXXXXX
;        A,- Entonces: Ejecuta "NODEF"
;    2.- RPC := Memoria (Dirección efectiva)
; Fin
;.....................................................
;            CODIGO                               30/IX/83
:                              (JMP)

       JMP   :  LD      A,(CODINS)    ; 1.-
                BIT     6,A
                JP      Z,JMP1
                BIT     5,A
                JP      Z,NODEF
       JMP1  :  LD      DE,(DIREF)
                LD      (RPC),DE
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              1/X/82
;                              (CLR)

; Inicio
;    1.- Memoria (Dirección efectiva) := ØØH
;    2.- Banderas 6809 : N = V = C = Ø
;    3.- Bandera 6809 (Z) := 1
; Fin
;.....................................................
;            CODIGO                               30/IX/83
;                              (CLR)
```

```
      CLR   :  LD      DE,(DIREF)    ; 1.-
               LD      A,Ø
               LD      (DE),A
               LD      A,(CC)        ; 2.-
               AND     A,ØF4H
               SET     2,A           ; 2.A.-
               LD      (CC),A
               RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                                  5/X/82
;                           (GRUPO1)

; Inicio
;    1.- Clasifica el código de la instrucción entre:
;        A.- XXXXØØØØ : Ejecuta "PAGE2"
;        B.- XXXXØØØ1 : Ejecuta "PAGE3"
;        C.- XXXXØØ1Ø : Ejecuta "NOPC"
;        D.- XXXXØØ11 : Ejecuta "SYNC"
;        E.- XXXXØ11Ø : Ejecuta "LBRA-C"
;        F.- XXXXØ111 : Ejecuta "LBSR"
;        G.- XXXX1ØØ1 : Ejecuta "DAAC"
;        H.- XXXX1Ø1Ø : Ejecuta "ORCC"
;        I.- XXXX11ØØ : Ejecuta "ANDCC"
;        J.- XXXX11Ø1 : Ejecuta "SEX"
;        K.- XXXX111X : Ejecuta "EXGTFR"
;        L.- Otros : Ejecuta "NODEF"
; Fin
;.....................................................
;           CODIGO                                  1/X/83
;                           (GRUPO1)

:      GRUPO1:  BIT     3,A           ; 1.-
                JP      NZ,G11G
```

```
                BIT     2,A
                JP      NZ,G11E
                BIT     1,A
                JP      NZ,G11C
                BIT     Ø,A
                JP      Z,PAGE2         ; 1.A.-
                JP      PAGE3           ; 1.B.-
G11C    :       BIT     Ø,A             ; 1.C.-
                JP      Z,NOPC
                JP      SYNC            ; 1.D.-
G11E    :       BIT     1,A             ; 1.E.-
                JP      Z,NODEF         ; 1.L.-
                BIT     Ø,A
                JP      Z,LBRA-C
                JP      LBSR            ; 1.F.-
G11G    :       BIT     2,A             ; 1.G.-
                JP      NZ,G11I
                BIT     1,A
                JP      NZ,G11H
                BIT     Ø,A
                JP      NZ,DAAC
                JP      NODEF           ; 1.L.-
G11H    :       BIT     Ø,A             ; 1.H.-
                JP      Z,ORCC
                JP      NODEF           ; 1.L.-
G11I    :       BIT     1,A             ; 1.I.-
                JP      NZ,EXGTFR       ; 1.K.-
                BIT     Ø,A
                JP      Z,ANDCC
                JP      SEX             ; 1.J.-

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;      PSEUDO-CODIGO                              5/X/82
;                              (PAGE2)

; Inicio
;    1.- Postbyte = Memoria (RPC)
;    2.- RPC := RPC + 1
;    3.- Si postbyte = ØØ1ØØØØØ
;        A.- Entonces: Ejecuta "NODEF"
;        B.- De lo contrario: Clasifica postbyte entre:
;            1.- ØØ1Ø XXXX : Ejecuta "LBRA-C"
;            2.- ØØ11 1111 : Ejecuta "SWI2"
;            3.- 1ØXX 11ØØ : Ejecuta "CMPDY"
;            4.- 1ØXX ØØ11 : Ejecuta "CMPDY"
;            5.- 1XXX 111X : Ejecuta "LDSTYS"
;            6.- Otros : Ejecuta "NODEF"
; Fin
;.....................................................
;            CODIGO                              1/X/83
;                              (PAGE2)

      PAGE2 :  LD      DE,(RPC)      ; 1.-
               LD      A,(DE)
               INC     DE            ; 2.-
               LD      (RPC),DE
               LD      (CODINS),A    ; 3.-
               LD      B,A
               XOR     A,2ØH
               JP      Z,NODEF       ; 3.A.-
               LD      A,(CODINS)
               BIT     7,B           ; 3.B.-
               JP      NZ,PA23B3
               BIT     6,B
               JP      NZ,PA23B6
               BIT     5,B
```

```
                JP      Z,PA23B6
                BIT     4,B
                JP      Z,LBRA-C        ; 3.B.1.-
                XOR     A,3FH           ; 3.B.2.-
                JP      NZ,PA23B6
                JP      SWI2
PA23B3:         BIT     6,B             ; 3.B.3.-
                JP      NZ,PA23B5
                BIT     3,B
                JP      Z,PA23B4
                BIT     2,B
                JP      Z,PA23B5
                BIT     1,B
                JP      NZ,LDSTYS
                BIT     Ø,B
                JP      NZ,PA23B4
                JP      CMPDY
PA23B4:         BIT     2,B             ; 3.B.4.-
                JP      NZ,PA23B5
                BIT     1,B
                JP      Z,PA23B5
                BIT     Ø,B
                JP      Z,PA23B5
                JP      CMPDY
PA23B5:         BIT     3,B             ; 3.B.5.-
                JP      Z,PA23B6
                BIT     2,B
                JP      Z,PA23B6
                BIT     1,B
                JP      Z,PA23B6
                JP      LDSTYS
PA23B6:         JP      NODEF           ; 3.B.6.-

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;       PSEUDO-CODIGO                                   6/X/82
;                                   (LBRA-C)

; Inicio
;     1.- Reg. temporal = Mem.(RPC + 1) , Mem.(RPC)
;     2.- RPC := RPC + 2
;     3.- Si código de instrucción = 16H
;         A.- Entonces; postbyte := 2ØH
;     4.- Clasifica postbyte para asignar condición:
;         A.- XXXX ØØØX : Condición := Ø
;         B.- XXXX ØØ1X : Condición := C or Z
;         C.- XXXX Ø1ØX : Condición := C (CARRY)
;         D.- XXXX Ø11X : Condición := Z (CERO)
;         E.- XXXX 1ØØX : Condición := V (SOBREFLUJO)
;         F.- XXXX 1Ø1X : Condición := N (NEGATIVO)
;         G.- XXXX 11ØX : Condición := N or exc V
;         H.- XXXX 111X : Condición := (N or exc V) or Z
;     5.- Si Bit Ø del postbyte = Ø
;         A.- Entonces: Condición := Condición negada
;     6.- Si condición = 1 ; Entonces:
;         A.- RPC := RPC + Reg. temp. - 3
;         B.- Si postbyte <> 2ØH
;             1.- Entonces:  RPC := RPC - 1
; Fin
;,.....................................................
;             CODIGO                              3/X/83
;                                   (LBRA-C)

        REGTEM:  DS 2
        CONDIC:  DS 1
        LBRA-C:  LD      DE,(RPC)      ; 1.-
                 LD      A,(DE)
                 LD      C,A
                 INC     DE
```

```
        LD      A,(DE)
        LD      B,A
        LD      (REGTEM),BC
        INC     DE              ; 2.-
        LD      (RPC),DE
        LD      A,(CODINS)      ; 3.-
        LD      B,A
        XOR     A,16H
        JP      NZ,LBRA4
        LD      B,20H           ; 3.A.-
LBRA4 : LD      A,(CC)          ; 4.-
        BIT     3,B
        JP      NZ,LBRA4E       ; 4.A.-
        BIT     2,B
        JP      NZ,LBRA4C
        BIT     1,B
        JP      NZ,LBRA4B
        LD      A,0
        LD      (CONDIC),A
        JP      LBRA5
LBRA4B: BIT     0,A             ; 4.B.-
        JP      NZ,LBRA1
        BIT     2,A
        JP      NZ,LBRA1
LBRA2 : LD      A,0
        LD      (CONDIC),A
        JP      LBRA5
LBRA1 : LD      A,1
        LD      (CONDIC),A
        JP      LBRA5
LBRA4C: BIT     1,B
        JP      NZ,LBRA4I
        BIT     0,A
```

```
            JP      NZ,LBRA1
            JP      LBRA2
LBRA4D:     BIT     2,A             ; 4.D.-
            JP      NZ,LBRA1
            JP      LBRA2
LBRA4E:     BIT     2,B             ; 4.E.-
            JP      NZ,LBRA4G
            BIT     1,B
            JP      NZ,LBRA4F
            BIT     1,A
            JP      NZ,LBRA1
            JP      LBRA2
LBRA4F:     BIT     3,A             ; 4.F.-
            JP      NZ,LBRA1
            JP      LBRA2
LBRA4G:     BIT     1,B             ; 4.G.-
            JP      NZ,LBRA4H
            BIT     3,A
            JP      NZ,LBRA3
            BIT     1,A
            JP      NZ,LBRA1
            JP      LBRA2
LBRA3 :     BIT     1,A
            JP      NZ,LBRA2
            JP      LBRA1
LBRA4H:     BIT     3,A             ; 4.H.-
            JP      NZ,LBRAB
            BIT     1,A
            JP      NZ,LBRAC
LBRAE :     LD      C,Ø
            JP      LBRAD
LBRAB :     BIT     1,A
            JP      NZ,LBRAE
```

```
LBRAC :  LD    C,1
LBRAD :  BIT   Ø,C
         JP    NZ,LBRA1
         BIT   2,A
         JP    NZ,LBRA1
         JP    LBRA2
LBRA5 :  LD    A,(CODINS)     ; 5.-
         LD    B,A
         LD    A,(CONDIC)
         BIT   Ø,B
         JP    NZ,LBRA6
         CPL   A              ; 5.A.-
LBRA6 :  BIT   Ø,A            ; 6.-
         JP    NZ,LBRA6A
         RET
LBRA6A:  LD    DE,(RPC)       ; 6.A.-
         DEC   DE
         DEC   DE
         DEC   DE
         LD    HL,(REGTEM)
         ADD   HL,DE
         LD    (RPC),HL
         LD    A,2ØH          ; 6.B.-
         XOR   A,B
         JP    NZ,LBRA6B
         RET
LBRA6B:  DEC   HL             ; 6.B.1.-
         LD    (RPC),HL
         RET

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;       PSEUDO-CODIGO                                   6/X/82
;                                 (SWI2)

; Inicio
;    1.- Registro temporal := ØFFF4H
;    2.- Ejecuta "SW"
; Fin
;..........................................................
;             CODIGO                                    5/X/83
;                                 (SWI2)

        SWI2   :  LD        DE,ØFFF4H        ; 1.-
                  LD        (REGINT),DE
                  JP        SW               ; 2.-

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   6/X/82
;                                  (SW)

; Inicio
;    1.- Bandera (E) := 1
;    2.- Ejecuta lo siguiente:
;        A.- SP' := SP - 1 , (SP) = RPCL
;        B.- SP' := SP - 1 , (SP) = RPCH
;        C.- SP' := SP - 1 , (SP) = USL
;        D.- SP' := SP - 1 , (SP) = USH
;        E.- SP' := SP - 1 , (SP) = RIYL
;        F.- SP' := SP - 1 , (SP) = RIYH
;        G.- SP' := SP - 1 , (SP) = RIXL
;        H.- SP' := SP - 1 , (SP) = RIXH
;        I.- SP' := SP - 1 , (SP) = DP
;        J.- SP' := SP - 1 , (SP) = ACCB
;        K.- SP' := SP - 1 , (SP) = ACCA
;        L.- SP' := SP - 1 , (SP) = CC
```

```
;     3.- RPC := Memoria (Registro temporal)
; Fin
;.....................................................
;              CODIGO                          5/X/83
;                              (SW)

       SW     :  LD      A,(CC)          ; 1.-
                 SET     7,A
                 LD      (CC),A
                 LD      DE,RSP          ; 2.-
                 DEC     DE              ; 2.A.-
                 LD      HL,DE
                 LD      (RSP),DE
                 LD      DE,RPC
                 LD      A,(DE)
                 LD      BC,DE
                 LD      DE,(RSP)
                 LD      DE,A
                 DEC     HL              ; 2.B.-
                 INC     BC
                 LD      (RSP),HL
                 LD      A,(BC)
                 LD      (HL),A
                 DEC     HL              ; 2.C.-
                 LD      (RSP),HL
                 LD      DE,US
                 LD      A,(DE)
                 LD      BC,DE
                 LD      DE,(RSP)
                 LD      (DE),A
                 DEC     HL              ; 2.D.-
                 LD      (RSP),HL
                 INC     BC
```

```
LD      A,(BC)
LD      (RSP),A
DEC     HL              ; 2.E.-
LD      (RSP),HL
LD      DE,RIY
LD      A,(DE)
LD      BC,DE
LD      DE,(RSP)
LD      (DE),A
DEC     HL              ; 2.F.-
INC     BC
LD      A,(BC)
LD      (RSP),A
DEC     HL              ; 2.G.-
LD      (RSP),HL
LD      DE,RIX
LD      A,(DE)
LD      BC,DE
LD      DE,(RSP)
LD      (DE),A
DEC     HL              ; 2.H.-
INC     BC
LD      A,(BC)
LD      (RSP),A
DEC     HL              ; 2.I.-
LD      (RSP),HL
LD      DE,DP
LD      A,(DE)
LD      DE,(RSP)
LD      (DE),A
DEC     HL              ; 2.J.-
LD      (RSP),HL
LD      DE,ACCB
```

```
                LD      A,(DE)
                LD      DE,(RSP)
                LD      (DE),A
                DEC     HL              ; 2.K.-
                LD      (RSP),HL
                LD      DE,ACCA
                LD      A,(DE)
                LD      DE,(RSP)
                LD      (DE),A
                DEC     HL              ; 2.L.-
                LD      (RSP),HL
                LD      DE,CC
                LD      A,(DE)
                LD      DE,(RSP)
                LD      (DE),A
                LD      DE,(REGINT)     ; 3.-
                LD      (RPC),DE
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           6/X/82
;                               (CMPDY)

; Inicio
;    1.- Si el Bit 3 del postbyte = Ø
;        A.- Entonces; Registro = Acumulador "D"
;        B.- De lo contrario; Registro = Indice "Y"
;    2.- Ejecuta "CMP16"
; Fin
;.....................................................
;            CODIGO                             5/X/83
;                               (CMPDY)

       RTRO  :  DS 2
```

```
CMPDY :  LD      A,(CODINS)    ; 1.-
         BIT     3,A
         JP      NZ,CMPDYB
         LD      A,(ACCA)      ; 1.A.-
         LD      B,A
         LD      A,(ACCB)
         LD      C,A
         LD      (RTRO),BC
         JP      CMP16         ; 2.-
CMPDYB:  LD      BC,(RIY)      ; 1.B.-
         LD      (RTRO),BC
         JP      CMP16
```

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              6/X/82
;                              (CMP16)

; Inicio
;    1.- Si código de instrucción = 1ØH  ó  11H
;        A.- Entonces: Código de inst. = Postbyte
;    2.- Clasifica Cod. de inst. entre (Direccionamientos):
;        A.- XXØØ XXXX  : Inmediato de 16 Bits
;        B.- XXØ1 XXXX  : Directo
;        C.- XX1Ø XXXX  : Indexado
;        D.- XX11 XXXX  : Extendido
;    3.- Registro temporal := Registro  -  Mem. (Dir. ef. +1),
;                                 Memoria (Dir. efec.)
;    4.- Si Bit 15 de registro temporal = 1
;        A.- Bandera (N) := 1
;    5.- Si registro temporal = Ø
;        A.- Bandera (Z) := 1
;    6.- Si Bit 15 or exc Bit 14 de reg. temp. = 1
;        A.- Bandera (V) := 1
```

```
;    7.- Si Bit 15 de Registro = 1  y  Bit 15 de registro
;        temporal = Ø
;        A.- Bandera (C) := 1
; Fin
;.....................................................
;            CODIGO                          5/X/83
;                              (CMP16)

       CMP16 :  LD      DE,(RPC)      ; Obtención del 1er.
                DEC     DE            ; Byte de código de
                DEC     DE            ; Instrucción
                LD      A,(DE)        ; 1.-
                LD      B,A
                BIT     Ø,A
                JP      NZ,CMP161
                XOR     A,1ØH
                JP      Z,CMP16A
       CMP16W:  LD      A,B
                JP      CMP162
       CMP161:  XOR     A,11H
                JP      NZ,CMP16W
       CMP16A:  LD      A,(CODINS)    ; 1.A.-
       CMP162:  BIT     5,A           ; 2.-
                JP      NZ,CMP62C
                BIT     4,A
                JP      Z,CMPW        ; 2.A.-
                CALL    DIRECT        ; 2.B.-
                JP      CMP163
       CMP62C:  BIT     4,A           ; 2.C.-
                JP      NZ,CMPZ       ; 2.D.-
                INC     DE
                LD      (RPC),DE
                CALL    INDEX
                JP      CMP163
```

```
CMPW  :  CALL    INM16
         JP      CMP163
CMPZ  :  CALL    EXTEN
CMP163:  LD      BC,(DIREF)     ; 3.-
         LD      HL,(RTRO)
         OR      A              ; Borra carry
         SBC     HL,BC          ; HL = Reg. Temp.
         BIT     7,H            ; 4.-
         JP      Z,CMP165
         LD      A,(CC)
         SET     3,A            ; 4.A.-
         LD      (CC),A
CMP165:  LD      BC,Ø           ; 5.-
         OR      A
         ADC     HL,BC
         JP      NZ,CMP166
         LD      A,(CC)
         SET     2,A            ; 5.A.-
         LD      (CC),A
CMP166:  LD      A,H            ; 6.-
         SLA     A
         LD      B,A
         AND     A,8ØH
         LD      A,H
         AND     A,8ØH
         XOR     A,B
         JP      Z,CMP167
         LD      A,(CC)
         SET     1,A            ; 6.A.-
         LD      (CC),A
CMP167:  LD      HL,(RTRO)      ; 7.-
         BIT     7,H
         JP      NZ,PRUEBA
```

```
                RET
        PRUEBA: LD      BC,(DIREF)
                BIT     7,B
                JP      NZ,FINAL
                LD      A,(CC)
                SET     Ø,A             ; 7.A.-
        FINAL : LD      (CC),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   7/X/82 .
;                               (INM16)

; Inicio
;    1.- Dirección Efectiva = RPC
;    2.- RPC := RPC + 2
; Fin
;.....................................................
;             CODIGO                                    6/X/83
;                               (INM16)

        INM16 : LD      DE,(RPC)        ; 1.-
                LD      A,(DE)
                INC     DE              ; RPC := RPC + 1
                LD      C,A
                LD      A,(DE)
                LD      B,A
                INC     DE              ; 2.-
                LD      (RPC),DE
                LD      (DIREF),BC
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;       PSEUDO-CODIGO                                   7/X/82
;                             (LDSTYS)

; Inicio
;    1.- Si el Bit 3 del postbyte = Ø
;        A.- Entonces: Registro = Indice "Y"
;        B.- De lo contrario: Registro = Stack "S"
;    2.- Ejecuta "LDST16"
; Fin
;.........................................................
;            CODIGO                                     6/X/83
;                            (LDSTYS)

        LDRTRO:  DS 2
        LDSTYS:  LD      A,(CODINS)      ; 1.-
                 BIT     3,A
                 JP      NZ,LDSTYA
                 LD      BC,RIY          ;.1.A.-
                 LD      (LDRTRO),BC
                 JP      LDST16          ; 2.-
        LDSTYA:  LD      BC,RSP          ; 1.B.-
                 LD      (LDRTRO),BC
                 JP      LDST16          ; 2.-

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   7/X/82
;                            (LDST16)

; Inicio
;    1.- Si el código de instrucción = 1ØH ó 11H
;        A.- Entonces: Código de inst. := Postbyte
;    2.- Clasifica el cod. de inst. entre (Direccionamiento)
;        A.- XXØØ XXXX : Inmediato de 16 Bits
;        B.--.XXØ1 XXXX : Directo
```

```
;         C.- XX1Ø XXXX : Indexado
;         D.- XX11 XXXX : Extendido
;     3.- Si el código de inst. < > XXØØ XXX1
;         A.- Entonces:
;             1.- Si el código de inst. = XXXX XXXØ
;                 A.- Entonces; Memoria (Registro)  :=
;                                Mem.(Dir. ef. + 1), Mem.(D.E.)
;                 B.- De lo contrario; Memoria (Dirección ef.
;                                + 1), Mem. (Dir. efec.) :=
;                                Memoria (Registro)
;             2.- Banderas (V,N y Z) := Ø
;             3.- Si Bit 15 de dato transferido = 1
;                 A.- Bandera (N) := 1
;             4.- Si dato transferido = Ø
;                 A.- Bandera (Z) := 1
;         B.- De lo contrario; Ejecuta "NODEF"
; Fin
;.....,.....
;             CODIGO                          7/X/83
;                          (LDST16)

        POST  :  DS 1
        LDST16:  LD      DE,(RPC)      ; 1.-
                 DEC     DE
                 DEC     DE
                 LD      A,(DE)        ; Obtención del 1er.
                 LD      B,A           ; Byte de código de
                 BIT     Ø,A           ; instrucción
                 JP      NZ,LDS161
                 XOR     A,1ØH
                 JP      Z,LDS16A
        LDS16W:  LD      A,B
                 JP      LDS162
        LDS161:  XOR     A,11H
```

```
            JP      NZ,LDS16W
LDS16A:     LD      A,(CODINS)      ; 1.A.-
LDS162:     BIT     5,A             ; 2.-
            JP      NZ,LDS62C
            BIT     4,A
            JP      Z,LDSTW         ; 2.A.-
            CALL    DIRECT          ; 2.B.-
            JP      LDS163
LDS62C:     BIT     4,A             ; 2.C.-
            JP      NZ,LDSTZ        ; 2.D.-
            INC     DE
            LD      (RPC),DE
            CALL    INDEX
            JP      LDS163
LDSTZ :     CALL    EXTEN
            JP      LDS163
LDSTW :     CALL    INM16
LDS163:     LD      A,(POST)        ; 3.-
            BIT     5,A
            JP      NZ,LDS164
            BIT     4,A
            JP      NZ,LDS164
            BIT     Ø,A
            JP      NZ,NODEF        ; 3.B.-
LDS164:     BIT     Ø,A             ; 3.A.1.-
            JP      NZ,LDS165
            LD      DE,(DIREF)      ; 3.A.1.A.-
            LD      (RTRO),DE
            JP      LDS3A2
LDS165:     LD      DE,(LDRTRO)     ; 3.A.1.B.-
            LD      (DIREF),DE
LDS3A2:     LD      A,(CC)          ; 3.A.2.-
            AND     A,ØF1H
```

```
                LD      BC,(LDRTRO)
                BIT     7,B
                JP      Z,LDS3A3
                SET     3,A
LDS3A3:         LD      HL,ØØH
                ADD     HL,BC
                JP      NZ,LDS3A4
                SET     2,A
LDS3A4:         LD      (CC),A
                RET

;NOTA: Registro = Etiqueta "LDRTRO"
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
;      PSEUDO-CODIGO                                7/X/82
;                              (PAGE3)

; Inicio
;    1.- Posbyte : = Memoria (PC)
;    2.- PC : = PC + 1
;    3.- Clasifica postbyte entre :
;        A.- 1 1 1 1 1 1 1 1 : SWI3
;        B.- 1 Ø X X Ø Ø 1 1 : CMPUS
;        C.- 1 Ø X X 1 1 Ø Ø : CMPUS
;        D.- OTROS           : NODEF
; Fin
;.....................................................
;                CODIGO                          6/X/83
;                                  (PAGE3)

      PAGE3 :  LD      DE,(RPC)      ; 1.-
               LD      A,(DE)
               INC     DE            ; 2.-
               LD      (RPC),DE
               LD      B,A           ; 3.-
               BIT     7,B
               JP      Z,PAG33A
               BIT     6,B           ; 3.B.-
               JP      NZ,PAG33D
               BIT     3,B
               JP      NZ,PAG33C
               BIT     2,B
               JP      NZ,PAG33D
               BIT     1,B
               JP      Z,PAG33D
               BIT     Ø,B
               JP      Z,PAG33D
               JP      CMPUS
```

```
PAG33A:  BIT     6,B          ; 3.A.-
         JP      NZ, PAG33D
         XOR     A,3FH
         JP      NZ,PAG33D
         JP      SWI3
PAG33C:  BIT     2,B          ; 3.C.-
         JP      Z,PAG33D
         BIT     1,B
         JP      NZ,PAG33D
         BIT     Ø,B
         JP      NZ,PAG33D
         JP      CMPUS
PAG33D:  JP      NODEF        ; 3.D.-

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              7/X/82
;                        (SWI3)

; Inicio
;    1.- Registro Temp. : = ØFFF2H
;    2.- Ejecuta (SW)
; Fin
;..............................................................
;            CODIGO                               6/X/83
;                              (SWI3)

       REGINT;  DS 2
       SWI3  :  LD      DE,0FFF2H    ; 1.-
                LD      (REGINT),DE
                JP      SW           ; 2.-

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              7/X/82
;                              (CMPUS)
```

```
; Inicio
;    1.- Si bit 3 : = Ø (Postbyte)
;        A.- Entonces; Registro : = Stack "U"
;        B.- De lo contrario; Registro : = Stack "S"
;    2.- Ejecuta (CMP16)
; Fin
;.....................................................
;             CODIGO                          6/X/83
;                          (CMPUS)

       CMPUS :  LD      A,(CODINS)    ; 1.-
                BIT     3,A
                JP      NZ,CMPUSB
                LD      BC,(US)       ; 1.A.-
                LD      (RTRO),BC
                JP      CMP16         ; 2.-
       CMPUSB:  LD      BC,(RSP)      ; 1.B.-
                LD      (RTRO),BC
                JP      CMP16         ; 2.-

;:::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                          7/X/82
;                              (NOP)

; Inicio
;    1.- Ejecuta "NOP" (Z-80)
; Fin
;.....................................................
;             CODIGO                          6/X/83
;                              (NOP)

       NOPC  :  NOP                   ; 1.-
                RET
```

```
;       PSEUDO-CODIGO                                    7/X/82
;                              (SYNC)

; Inicio
;    1.- Fin : = Falso
;    2.- Repite
;        A.- Si tecla oprimida
;            1.- Entonces; Fin : = Verdadero
;    3.- Hasta Fin : = Verdadero
; Fin
;..............................................................
;               CODIGO                                   31/X/83
;                              (SYNC)

        FIN5   :  DS 1
        SYNC   :  LD      A,Ø            ; 1.-
                  LD      (FIN5),A
        SYNC1  :  LD      C,ØBH          ; 2.-
                  CALL    CDOS
                  BIT     Ø,A            ; 2.A.-
                  JP      Z,SYNC1
                  LD      (FIN5),A       ; 2.A.1.-
                  RET                    ; 3.-

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                    8/X/82
;                              (LBSR)

; Inicio
;    1.- SP' : = SP - 1, (SP) : = PCL
;    2.- SP' : = SP - 1, (SP) : = PCH
;    3.- PC : = PC + (Mem. (PC + 1), Mem. (PC))--1
; Fin
;..............................................................
```

```
;               CODIGO                                  7/X/83
;                                   (LBSR)

       LBSR  :  LD      DE,(RSP)       ; 1.-
                DEC     DE
                LD      HL,DE
                LD      (RSP),DE
                LD      DE,RPC
                LD      A,(DE)
                LD      BC,DE
                LD      DE,(RSP)
                LD      (DE),A
                DEC     HL             ; 2.-
                INC     BC
                LD      (RSP),HL
                LD      A,(BC)
                LD      (HL),A
                LD      DE,(RPC)       ; 3.-
                LD      A,(DE)
                LD      L,A
                INC     DE
                LD      A,(DE)
                LD      H,A
                DEC     DE
                DEC     DE
                ADD     HL,DE
                LD      (RPC),HL
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           8/X/82
                                    (DAAC)
;
; Inicio
```

```
;     1.- Ejecuta "DAA" (Z-80)
;     2.- Bandera (N, Z, y C) del 6809 : = Banderas (S, Z y C)
          del Z-80
; Fin
;............................................................
;            CODIGO                               7/X/83
;                             (DAAC)

      DAAC  :  LD      A,(ACCA)        ; 1.-
               DAA
               LD      (ACCA),A
               PUSH    AF              ; 2.-
               LD      A,(CC)
               AND     A,ØF2H
               POP     BC
               BIT     Ø,C
               JP      Z,DAA1
               SET     Ø,A
      DAA1  :  BIT     6,C
               JP      Z,DAA2
               SET     2,A
      DAA2  :  BIT     7,C
               JP      Z,DAA3
               SET     3,A
      DAA3  :  LD      (CC),A
               RET

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              8/X/82
;                             (ORCC)

; Inicio
;     1.- Byte : = Memoria (PC)
;     2.- PC : = PC + 1
```

```
;     3.- Cy : = Byte (Ø) or Cy
;     4.- V : = Byte (1) or V
;     5.- Z : = Byte (2) or Z
;     6.- N : = Byte (3) or N
;     7.- I ; = Byte (4) or I
;     8.- H : = Byte (5) or H
;     9.- F : = Byte (6) or F
;    10.- E : = Byte (7) or E
; Fin
;.....................................................
;              CODIGO                          7/X/83
;                              (ORCC)

       ORCC  :  LD      DE,(RPC)      ; 1.-
                LD      A,(DE)
                INC     DE            ; 2.-
                LD      (RPC),DE
                LD      B,A
                LD      A,(CC)        ; 3.-
                BIT     Ø,A
                JP      NZ,ORCC8
                BIT     Ø,B
                JP      NZ,ORCC8
                JP      ORCC9
       ORCC8:   SET     Ø,A
       ORCC9 :  BIT     1,A           ; 4.-
                JP      NZ,ORCC1Ø
                BIT     1,B
                JP      NZ,ORCC1Ø
                JP      ORCC11
       ORCC1Ø:  SET     1,A
       ORCC11:  BIT     2,A           ; 5.-
                JP      NZ,ORCC12
                BIT     2,B
```

```
        JP      NZ,ORCC12
        JP      ORCC13
ORCC12: SET     2,A
ORCC13: BIT     3,A             ; 6.-
        JP      NZ,ORCC14
        BIT     3,B
        JP      NZ,ORCC14
        JP      ORCC15
ORCC14: SET     3,A
ORCC15: BIT     4,A             ; 7.-
        JP      NZ,ORCC16
        BIT     4,B
        JP      NZ,ORCC16
        JP      ORCC17
ORCC16: SET     4,A
ORCC17: BIT     5,A             ; 8.-
        JP      NZ,ORCC18
        BIT     5,B
        JP      NZ,ORCC18
        JP      ORCC19
ORCC18: SET     5,A
ORCC19: BIT     6,A             ; 9.-
        JP      NZ,ORCC2Ø
        BIT     6,B
        JP      NZ,ORCC2Ø
        JP      ORCC21
ORCC2Ø: SET     6,A
ORCC21: BIT     7,A             ; 1Ø.-
        JP      NZ,ORCC22
        BIT     7,B
        JP      NZ,ORCC22
        JP      ORCC23
ORCC22: SET     7,A
ORCC23: LD      (CC),A          ; 11.-
```

```
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           8/X/82
;                               (ANDCC)

; Inicio
;     1.- Byte : = Memoria (PC)
;     2.- PC = PC + 1
;     3.- Cy : = Byte (Ø) and Cy
;     4.- V : = Byte (1) and V
;     5.- Z : = Byte (2) and Z
;     6.- N : = Byte (3) and N
;     7.- I : = Byte (4) and I
;     8.- H : = Byte (5) and H
;     9.- F : = Byte (6) and F
;    10.- E : = Byte (7) and E
; Fin
;.............................................................
;               CODIGO                          7/X/83
;                               (ANDCC)

        ANDCC :  LD      DE,(RPC)      ; 1.-
                 LD      A,(DE)
                 INC     DE            ; 2.-
                 LD      (RPC),DE
                 LD      B,A
                 LD      A,(CC)        ; 3.-
                 BIT     Ø,A
                 JP      Z,ANDCC3
                 BIT     Ø,B
                 JP      Z,ANDCC3
                 SET     Ø,A
                 JP      ANDCC4
```

```
ANDCC3:  RES     Ø,A
ANDCC4:  BIT     1,A            ; 4.-
         JP      Z,ANDC4A
         BIT     1,B
         JP      Z,ANDC4A
         SET     1,A
         JP      ANDC5
ANDC4A:  RES     1,A
ANDC5 :  BIT     2,A            ; 5.-
         JP      Z,ANDC5A
         BIT     2,B
         JP      Z,ANDC5A
         SET     2,A
         JP      ANDC6
ANDC5A:  RES     2,A
ANDC6 :  BIT     3,A            ; 6.-
         JP      Z,ANDC6A
         BIT     3,B
         JP      Z,ANDC6A
         SET     3,A
         JP      ANDC7
ANDC6A:  RES     3,A
ANDC7 :  BIT     4,A            ; 7.-
         JP      Z,ANDC7A
         BIT     4,B
         JP      Z,ANDC7A
         SET     4,A
         JP      ANDC8
ANDC7A:  RES     4,A
ANDC8 :  BIT     5,A            ; 8.-
         JP      Z,ANDC8A
         BIT     5,B
         JP      Z,ANDC8A
         SET     5,A
```

```
                JP      ANDC9
        ANDC8A: RES     5,A
        ANDC9 : BIT     6,A             ; 9.-
                JP      Z,ANDC9A
                BIT     6,B
                JP      Z,ANDC9A
                SET     6,A
                JP      ANDC1Ø
        ANDC9A: RES     6,A
        ANDC1Ø: BIT     7,A             ; 10.-
                JP      Z,AND1ØA
                BIT     7,B
                JP      Z,AND1ØA
                SET     7,A
                JP      AND11
        AND1ØA: RES     7,A
        AND11 : LD      (CC),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           8/X/82
;                               (SEX)

; Inicio
;    1.- Si bit 7 del Acumulador "B" : = 1
;        A.- Entonces; Acumulador "A" : = ØFFH
;        B.- De lo contrario; Acumulador "A" : = ØØH
;    2.- Acumulador "D" : = ACCA : ACCB
;    3.- Si bit 15 de ACCD : = 1
;        A.- Entonces; Bandera (N) : = 1
;    4.- Si Acumulador "D" : = Ø
;        A.- Entonces; Bandera (Z) : = 1
; Fin
;.....................................................
```

```
;                CODIGO                                  7/X/83
;                                  (SEX)

        SEX   :  LD      A,(ACCB)       ; 1.-
                 LD      B,A
                 BIT     7,B
                 JP      Z,SEX1
                 LD      A,ØFFH         ; 1.A.-
                 JP      SEXW
        SEX1  :  LD      A,Ø            ; 1.B.-
        SEXW  :  LD      (ACCA),A
                 LD      D,A            ; 2.-
                 LD      A,(ACCB)
                 LD      E,A
                 LD      (ACCD),DE
                 BIT     7,D            ; 3.-
                 JP      Z,SEX2
                 LD      A,(CC)         ; 3.A.-
                 SET     3,A
                 LD      (CC),A
        SEX2  :  LD      HL,Ø           ; 4.-
                 ADD     HL,DE
                 JP      Z,SEX3
                 SET     2,A            ; 4.A.-
                 LD      (CC),A
        SEX3  :  RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                              8/X/82
;                                (EXGTFR)

; Inicio
;    1.- Postbyte : = Memoria (PC)
;    2.- PC : = PC + 1
```

```
;    3.- Clasifica Postbyte entre (Reg. # 1 : = ) :
;        A.- Ø Ø Ø Ø X X X X : ACCD
;        B.- Ø Ø Ø 1 X X X X : X
;        C.- Ø Ø 1 Ø X X X X : Y
;        D.- Ø Ø 1 1 X X X X : US
;        E.- Ø 1 Ø Ø X X X X : SP
;        F.- Ø 1 Ø 1 X X X X : PC
;        G.- 1 Ø Ø Ø X X X X : A
;        H.- 1 Ø Ø 1 X X X X : B
;        I.- 1 Ø 1 Ø X X X X : CC
;        J.- 1 Ø 1 1 X X X X : DP
;        K.- OTROS           : Ejecuta (NODEF)
;    4.- Clasifica Postbyte entre (Reg. # 2 : = ) :
;        A.- X X X X Ø Ø Ø Ø : ACCD
;        B.- X X X X Ø Ø Ø Ø : X
;        C.- X X X X Ø Ø 1 Ø : Y
;        D.- X X X X Ø Ø 1 1 : US
;        E.- X X X X Ø 1 Ø Ø : SP
;        F.- X X X X Ø 1 Ø 1 : PC
;        G.- X X X X 1 Ø Ø Ø : A
;        H.- X X X X 1 Ø Ø 1 : B
;        I.- X X X X 1 Ø 1 Ø : CC
;        J.- X X X X 1 Ø 1 1 : DP
;        K.- OTROS           : Ejecuta (NODEF)
;    5.- Si tamaño de Reg. # 1 : = tamaño de Reg. # 2
;        A.- Entonces:
;            1.- Reg. Temp. # 1 : = Registro # 1
;            2.- Reg. Temp. # 2 : = Registro # 2
;        B.- De lo contrario indica ERROR
;    6.- Si bit Ø del código de instrucción : = Ø
;        A.- Entonces:
;            1.- Reg. # 2 : = Reg. Temp. # 1
;            2.- Reg. # 1 : = Reg. Temp. # 2
;        B.- De lo contrario:
```

```
;            1.- Reg. # 2 : = Reg. Temp. # 1
; Fin
;.....................................................
;             CODIGO                          8/X/83
:                          (EXGTFR)

       OCHO   :  DS 1
       EXGTFR:   LD      DE,(RPC)      ; 1.-
                 LD      A,(DE)
                 INC     DE            ; 2.-
                 LD      (RPC),DE
                 BIT     7,A           ; 3.-
                 JP      NZ,EXG3G
                 BIT     6,A
                 JP      NZ,EXG3E
                 BIT     5,A
                 JP      NZ,EXG3C
                 BIT     4,A
                 JP      NZ,EXG3B
                 LD      DE,ACCD       ; 3.A.-
                 JP      EXG4
       EXG3B  :  LD      DE,RIX        ; 3.B.-
                 JP      EXG4
       EXG3C  :  BIT     4,A           ; 3.C.-
                 JP      NZ,EXG3D
                 LD      DE,RIY
                 JP      EXG4
       EXG3D  :  LD      DE,US         ; 3.D.-
                 JP      EXG4
       EXG3E  :  BIT     5,A           ; 3.E.-
                 JP      NZ,EXG3K
                 BIT     4,A
                 JP      NZ,EXG3F
                 LD      DE,(RSP)
```

```
            JP      EXG4
EXG3F :     LD      DE,RPC          ; 3.F.-
            JP      EXG4
EXG3G :     BIT     6,A             ; 3.G.-
            JP      NZ,EXG3K
            BIT     5,A
            JP      NZ,EXG3I
            BIT     4,A
            JP      NZ,EXG3H
            LD      DE,ACCA
            JP      EXG4
EXG3H :     LD      DE,ACCB         ; 3.H.-
            JP      EXG4
EXG3I :     BIT     4,A             ; 3.I.-
            JP      NZ,EXG3J
            LD      DE,CC
            JP      EXG4
EXG3J :     LD      DE,DP           ; 3.J.-
            JP      EXG4
EXG3K :     JP      NODEF           ; 3.K.-
EXG4  :     BIT     3,A             ; 4.-
            JP      NZ,EXG4G
            BIT     2,A
            JP      NZ,EXG4E
            BIT     1,A
            JP      NZ,EXG4C
            BIT     Ø,A
            JP      NZ,EXG4B
            LD      HL,ACCD         ; 4.A.-
            JP      EXG5
EXG4B :     LD      HL,RIX          ; 4.B.-
            JP      EXG5
EXG4C :     BIT     Ø,A             ; 4.C.-
            JP      NZ,EXG4D
```

```
            LD      HL,RIY
            JP      EXG5
EXG4D :     LD      HL,US           ; 4.D.-
            JP      EXG5
EXG4E :     BIT     1,A             ; 4.E.-
            JP      NZ,EXG4K
            BIT     Ø,A
            JP      NZ,EXG4F
            LD      HL,(RSP)
            JP      EXG5
EXG4F :     LD      HL,RPC          ; 4.F.-
            JP      EXG5
EXG4G :     BIT     2,A             ; 4.G.-
            JP      NZ,EXG4K
            BIT     1,A
            JP      NZ,EXG4I
            BIT     Ø,A
            JP      NZ,EXG4H
            LD      HL,ACCA
            JP      EXG5
EXG4H :     LD      HL,ACCB         ; 4.H.-
            JP      EXG5
EXG4I :     BIT     Ø,A             ; 4.I.-
            JP      NZ,EXG4J
            LD      HL,CC
            JP      EXG5
EXG4J :     LD      HL,DP           ; 4.J.-
            JP      EXG5
EXG4K :     JP      NODEF           ; 4.K.-
EXG5  :     BIT     7,A             ; 5.-
            JP      NZ,EXG55
            BIT     3,A
            JP      Z,EXG5AA
            JP      ERROR           ; 5.B.-
```

```
EXG55 :  BIT     3,A
         JP      NZ,EXG5AB
         JP      ERROR          ; 5.B.-
EXG5AA:  LD      A,ØFFH
         LD      (OCHO),A
         JP      EXG5A1
EXG5AB:  LD      A,Ø
         LD      (OCHO),A
EXG5A1:  LD      BC,(RPC)       ; 6.-
         DEC     BC
         DEC     BC
         LD      A,(BC)
         BIT     Ø,A
         JP      NZ,EXG6B       ; 6.A.-
EXG6A4:  LD      A,(HL)
         LD      B,A
         LD      A,(DE)
         LD      (HL),A
         LD      A,B
         LD      (DE),A
         LD      A,(OCHO)
         JP      NZ,EXG6A3
         RET
EXG6A3:  INC     DE
         INC     HL
         LD      A,Ø
         LD      (OCHO),A
         JP      EXG6A4
EXG6B :  LD      A,(DE)         ; 6.B.-
         LD      (HL),A
         LD      A,(OCHO)
         JP      NZ,EXG6B1
         RET
EXG6B1:  INC     DE
```

```
                INC     HL
                LD      A,(DE)
                LD      (HL),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           8/X/82
;                               (GRUPO2)

; Inicio
;     1.- Registro Temporal : = Memoria (PC)
;     2.- PC : = PC + 1
;     3.- Clasifica código de instrucción para asignar Condi-
          ción:
;         A.- X X X X Ø Ø Ø X : Condición : = Ø
;         B.- X X X X Ø Ø 1 X : Condición : = C or Z
;         C.- X X X X Ø 1 Ø X : Condición : = C (Carry)
;         D.- X X X X Ø 1 1 X : Condición : = Z (Cero)
;         E.- X X X X 1 Ø Ø X : Condición : = V (Overflow)
;         F.- X X X X 1 Ø 1 X : Condición : = N (Negativo)
;         G.- X X X X 1 1 Ø X : Condición : = N or exc. V
;         H.- X X X X 1 1 1 X : Condición : = (N or exc. V)
              or Z
;     4.- Si bit Ø del código de instrucción : = Ø
;         A.- Entonces; Condición : = Condición negada
;     5.- Si Condición : = 1
;         A.- Entonces; PC : = PC + Reg. Temp. - 2
; Fin
;.....................................................
;             CODIGO                            10/X/83
                                (GRUPO2)

       REGTE2:  DS 1
       GRUPO2:  LD      DE,(RPC)      ; 1.-
```

```
            LD      A,(DE)
            LD      (REGTE2),A
            INC     DE              ; 2.-
            LD      (RPC),DE
            LD      A,(CODINS)      ; 3.-
            LD      B,A
            LD      A,(CC)
            BIT     3,B
            JP      NZ,GRU23E
            BIT     2,B
            JP      NZ,GRU23C
            BIT     1,B
            JP      NZ,GRU23B
            LD      A,Ø             ; 3.A.-
            LD      (CONDIC),A
            JP      GRU24
GRU23B;     BIT     Ø,A             ; 3.B.-
            JP      NZ,GRU21
            BIT     2,A
            JP      NZ,GRU21
GRU22 :     LD      A,Ø
            LD      (CONDIC),A
            JP      GRU24
GRU21 :     LD      A,1
            LD      (CONDIC),A
            JP      GRU24
GRU23C:     BIT     1,B             ; 3.C.-
            JP      NZ,GRU23D
            BIT     Ø,A
            JP      NZ,GRU21
            JP      GRU22
GRU23D:     BIT     2,A             ; 3.D.-
            JP      NZ,GRU21
            JP      GRU22
```

```
GRU23E:  BIT     2,B          ; 3.E.-
         JP      NZ,GRU23G
         BIT     1,B
         JP      NZ,GRU23F
         BIT     1,A
         JP      NZ,GRU21
         JP      GRU22
GRU23F:  BIT     3,A          ; 3.F.-
         JP      NZ,GRU21
         JP      GRU22
GRU23G:  BIT     1,B          ; 3.G.-
         JP      NZ,GRU23H
         BIT     3,A
         JP      NZ,GRU23
         BIT     1,A
         JP      NZ,GRU21
         JP      GRU22
GRU23 :  BIT     1,A
         JP      NZ,GRU22
         JP      GRU21
GRU23H:  BIT     3,A          ; 3.H.-
         JP      NZ,GRU2B
         BIT     1,A
         JP      NZ,GRU2C
GRU2E :  LD      C,Ø
         JP      GRU2D
GRU2B :  BIT     1,A
         JP      NZ,GRU2E
GRU2C :  LD      C,1
GRU2D :  BIT     Ø,C
         JP      NZ,GRU21
         BIT     2,A
         JP      NZ,GRU21
         JP      GRU22
```

```
GRU24 :  BIT     Ø,B          ; 4.-
         JP      NZ,GRU25
         CPL     A
GRU25 :  BIT     Ø,A          ; 5.-
         JP      NZ,GRU25A
         RET
GRU25A:  LD      DE,(RPC)
         DEC     DE
         DEC     DE
         LD      HL,(REGTE2)
         ADD     HL,DE
         LD      (RPC),HL
         RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;     PSEUDO-CODIGO                          8/X/82
;                        (GRUP03)

; Inicio
;    1.- Clasifica código de instrucción entre (instruccio-
         nes):
;        A.- X X X X Ø Ø X X : LEA
;        B.- X X X X Ø 1 X Ø : PSH
;        C.- X X X X Ø 1 X 1 : PUL
;        D.- X X X X 1 Ø Ø 1 : RTS
;        E.- X X X X 1 Ø 1 Ø : ABX
;        F.- X X X X 1 Ø 1 1 : RTI
;        G.- X X X X 1 1 Ø Ø : CWAI
;        H.- X X X X 1 1 Ø 1 : MUL
;        I.- X X X X 1 1 1 1 : SWI
;        J.- OTROS            : NODEF
; Fin
;.............................................................
;            CODIGO                          10/X/83
```

(GRUPO3)

```
GRUPO3:  BIT     3,A          ; 1.-
         JP      NZ,GRUP3D
         BIT     2,A
         JP      NZ,GRP3BC
         JP      LEA          ; 1.A.-
GRP3BC:  BIT     Ø,A
         JP      Z,PSH        ; 1.B.-
         JP      PUL          ; 1.C.-
GRUP3D:  BIT     2,A
         JP      NZ,GP3GHI
         BIT     1,A
         JP      NZ,GRP3EF
         BIT     Ø,A
         JP      Z,NODEF      ; 1.J.-
         JP      RTS          ; 1.D.-
GRP3EF:  BIT     Ø,A
         JP      Z,ABX        ; 1.E.-
         JP      RTI          ; 1.F.-
GP3GHI:  BIT     1,A
         JP      NZ,GRUP3I
         BIT     Ø,A
         JP      Z,CWAI       ; 1.G.-
         JP      MUL
GRUP3I:  BIT     Ø,A
         JP      NZ,SWI       ; 1.I.-
         JP      NODEF
```

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                              8/X/82
;                              (LEA)

; Inicio
```

```
;    1.- Clasifica Código de inst. entre (Registro):
;        A.- X X X X X X Ø Ø : Registro : = Indice "X"
;        B.- X X X X X X Ø 1 : Registro : = Indice "Y"
;        C.- X X X X X X 1 Ø : Registro : = Stack "S"
;        D.- X X X X X X 1 1 : Registro : = Stack "U"
;    2.- Ejecuta Rutina (INDEX)
;    3.- Registro : = Dirección Efectiva
;    4.- Si bit 1 y bit Ø de Código de Inst. : 1Ø o 11
;        A.- Entonces; Banderas no afectadas
;        B.- De lo contrario:
;            1.- Si Registro : = Ø
;                A.- Entonces; Bandera (Z) : = 1
; Fin
;.....................................................
;            CODIGO                          10/X/83
;                              (LEA)

      RETRO :  DS 2
      LEA   :  BIT     1,A          ; 1.-
               JP      NZ,LEA1C
               BIT     Ø,A
               JP      NZ,LEA1B
               LD      DE,RIX       ; 1.A.-
               LD      (RETRO),DE
               JP      LEA2
      LEA1B :  LD      DE,RIY       ; 1.B.-
               LD      (RETRO),DE
               JP      LEA2
      LEA1C :  BIT     Ø,A
               JP      NZ,LEA1D
               LD      DE,RSP       ; 1.C.-
               LD      (RETRO),DE
               JP      LEA2
      LEA1D :  LD      DE,US        ; 1.D.-
```

```
                LD      (RETRO),DE
        LEA2  : CALL    INDEX           ; 2.-
                LD      DE,DIREF        ; 3.-
                LD      A,(DE)
                LD      C,A
                INC     DE
                LD      A,(DE)
                LD      B,A
                LD      DE,(RETRO)
                LD      A,C
                LD      (DE),A
                INC     DE
                LD      A,B
                LD      (DE),A
                LD      A,(CODINS)      ; 4.-
                BIT     1,A
                JP      NZ,LEA4A        ; 4.A.-
                LD      HL,Ø
                ADD     HL,DE
                JP      NZ,LEA4A
                LD      A,(CC)          ; 4.B.1.A.-
                SET     2,A
                LD      (CC),A
        LEA4A : RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           11/X/82
;                               (PSH)

; Inicio
;    1.- Si bit 1 de Código de Inst. : = Ø
;        A.- Entonces; Registro : = Stack "S"
;        B.- De lo contrario; Registro : = Stack "U"
;    2.- Postbyte : = Memoria (PC)
```

```
;    3.- PC : = PC + 1
;    4.- Clasifica Postbyte entre (Registro por salvar):
;        A.- 1 X X X X X X X : Reg. : = Reg. - 1; Mem. (REG)
;            : = PCL,  Reg. : = Reg. - 1; Mem. (REG) : = PCH
;        B.- X 1 X X X X X X : Reg. : = Reg. - 1; Mem. (REG)
;            : = SL,  Reg. : = Reg. - 1; Mem. (REG) : = SH
;        C.- X X 1 X X X X X : Reg. : = Reg. - 1; Mem. (REG)
;            : = IYL,  Reg. : = Reg. - 1; Mem. (REG) : = IYH
;        D.- X X X 1 X X X X : Reg. : = Reg. - 1; Mem. (REG)
;            : = IXL,  Reg. : = Reg. - 1; Mem. (REG) : = IXH
;        E.- X X X X 1 X X X : Reg. : = Reg. - 1; Mem. (REG)
;            : = DP
;        F.- X X X X X 1 X X : Reg. : = Reg. - 1; Mem. (REG)
;            : = ACCB
;        G.- X X X X X X 1 X : Reg. : = Reg. - 1; Mem. (REG)
;            : = ACCA
;        H.- X X X X X X X 1 : Reg. : = Reg. - 1; Mem. (REG)
;            : = CC
;    5.- Actualiza Stack (U o S)
;        Nota.- SL = Parte baja del Stack (U o S)
;               SH = Parte alta del Stack (U o S)
; Fin
;.....................................................
;            CODIGO                    14/X/83
;                        (PSH)

       PSH   :  BIT     1,A          ; 1.-
                JP      NZ,PSH1B
                LD      DE,(RSP)     ; 1.A.-
                LD      L,Ø
                JP      PSH2
       PSH1B :  LD      DE,(US)      ; 1.B.-
                LD      L,ØFFH
       PSH2  :  LD      BC,(RPC)     ; 2.-
```

```
                LD      A,(BC)
                INC     BC              ; 3.-
                LD      (RPC),BC
                LD      H,A             ; Salva Postbyte
                BIT     7,H             ; 4.-
                JP      Z,PSH4B
                DEC     DE              ; 4.A.-
                LD      A,C
                LD      (DE),A
                DEC     DE
                LD      A,B
                LD      (DE),A
PSH4B :         BIT     6,H             ; 4.B.-
                JP      Z,PSH4C
                DEC     DE
                BIT     Ø,L
                JP      NZ,PSH4B2
                LD      BC,(US)
PSH4B1:         LD      A,C
                LD      (DE),A
                DEC     DE
                LD      A,B
                LD      (DE),A
                JP      PSH4C
PSH4B2:         LD      BC,(RSP)
                JP      PSH4B1
PSH4C :         BIT     5,H             ; 4.C.-
                JP      Z,PSH4D
                DEC     DE
                LD      BC,(RIY)
                LD      A,C
                LD      (DE),A
                DEC     DE
                LD      A,B
```

```
            LD      (DE),A
PSH4D :     BIT     4,H             ; 4.D.-
            JP      Z,PSH4E
            DEC     DE
            LD      BC,(RIX)
            LD      A,C
            LD      (DE),A
            DEC     DE
            LD      A,B
            LD      (DE),A
PSH4E :     BIT     3,H             ; 4.E.-
            JP      Z,PSH4F
            DEC     DE
            LD      BC,DP
            LD      A,(BC)
            LD      (DE),A
PSH4F :     BIT     2,H             ; 4.F.-
            JP      Z,PSH4G
            DEC     DE
            LD      BC,ACCB
            LD      A,(BC)
            LD      (DE),A
PSH4G :     BIT     1,H             ; 4.G.-
            JP      Z,PSH4H
            DEC     DE
            LD      BC,ACCA
            LD      A,(BC)
            LD      (DE),A
PSH4H :     BIT     Ø,H             ; 4.H.-
            JP      Z,PSH5
            DEC     DE
            LD      BC,CC
            LD      A,(BC)
            LD      (DE),A
```

```
PSH5  :  BIT     ¢,L          ; 5.-
         JP      NZ,PSH5A
         LD      (RSP),DE
         RET
PSH5A :  LD      (US),DE
         RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              11/X/82
;                            (PUL)

; Inicio
;    1.- Si bit 1 de Código de Inst. : = ∅
;        A.- Entonces; Registro : = Stack "S"
;        B.- De lo contrario; Registro : = Stack "U"
;    2.- Postbyte : = Memoria (PC)
;    3.- PC : = PC + 1
;    4.- Clasifica Postbyte entre (Registro por sacar):
;        A.- X X X X X X X 1 : CC : = Mem. (REG), Registro :
;            = REG. + 1
;        B.- X X X X X X 1 X : ACCA : = Mem. (REG), Registro
;            : = REG. + 1
;        C.- X X X X X 1 X X : ACCB : = Mem. (REG), Registro
;            : = REG. + 1
;        D.- X X X X 1 X X X : DP : = Mem. (REG), Registro :
;            = REG. + 1
;        E.- X X X 1 X X X X : IXH : = Mem. (REG), Registro
;            : = REG. + 1,  IXL : = Mem. (REG), Registro : =
;            REG. + 1
;        F.- X X 1 X X X X X : IYH : = Mem. (REG), Registro
;            : = REG. + 1,  IYL : = Mem. (REG), Registro : =
;            REG. + 1
;        G.- X 1 X X X X X X : SH : Mem. (REG), Registro : =
;            REG. + 1,  SL : = Mem. (REG), Registro : = REG.
```

```
;            + 1
;        H.- 1 X X X X X X : PCH : = Mem. (REG), Registro :
;            = REG. + 1,  PCL : = Mem. (REG), Registro : =
;            REG. + 1
;    5.- Actualiza Stack (U o S)
;        Nota.- SH = Parte alta del Stack (U o S)
;               SL = Parte baja del Stack (U o S)
; Fin
;.....................................................
;             CODIGO                          14/X/83
;                          (PUL)

      PUL   : BIT     1,A          ; 1.-
              JP      NZ,PUL1B
              LD      DE,(RSP)     ; 1.A.-
              LD      L,Ø
              JP      PUL2
      PUL1B : LD      DE,(US)      ; 1.B.-
              LD      L,ØFFH
      PUL2  : LD      BC,(RPC)     ; 2.-
              LD      A,(BC)
              INC     BC
              LD      (RPC),BC     ; 3.-
              LD      H,A
              BIT     Ø,H          ; 4.-
              JP      Z,PUL4B
      PUL4A : LD      A,(DE)       ; 4.A.-
              LD      (CC),A
              INC     DE
      PUL4B : BIT     1,H          ; 4.B.-
              JP      Z,PUL4C
              LD      A,(DE)
              LD      (ACCA),A
              INC     DE
```

```
PUL4C :  BIT     2,H             ; 4.C.-
         JP      Z,PUL4D
         LD      A,(DE)
         LD      (ACCB),A
         INC     DE
PUL4D :  BIT     3,H             ; 4.D.-
         JP      Z,PUL4E
         LD      A,(DE)
         LD      (DP),A
         INC     DE
PUL4E :  BIT     4,H             ; 4.E.-
         JP      Z,PUL4F
         LD      A,(DE)
         LD      B,A
         INC     DE
         LD      A,(DE)
         LD      C,A
         LD      (RIX),BC
         INC     DE
PUL4F :  BIT     5,H             ; 4.F.-
         JP      Z,PUL4G
         LD      A,(DE)
         LD      B,A
         INC     DE
         LD      A,(DE)
         LD      C,A
         LD      (RIY),BC
         INC     DE
PUL4G :  BIT     6,H             ; 4.G.-
         JP      Z,PUL4H
         LD      A,(DE)
         LD      B,A
         INC     DE
         LD      A,(DE)
```

```
                LD      C,A
                BIT     Ø,L
                JP      NZ,PUL4G1
                LD      (RSP),BC
                JP      PUL4H
        PUL4G1: LD      (US),BC
        PUL4H : BIT     7,H             ; 4.H.-
                JP      Z,PUL5
                LD      A,(DE)
                LD      B,A
                INC     DE
                LD      A,(DE)
                LD      C,A
                LD      (RPC),BC
        PUL5  : BIT     Ø,L
                JP      NZ,PUL5A
                LD      (RSP),DE
                RET
        PUL5A ; LD      (US),DE
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           11/X/82
;                               (RTS)

; Inicio
;    1.- PCH : = (SP), SP' : = SP + 1
;    2.- PCL ; = (SP), SP' : = SP + 1
;        Nota.- SP = Apuntador de Stack Hardware
; Fin
;.....................................................................
;           CODIGO                              14/X/83
;                               (RTS)
```

```
RTS   :  LD      DE,(RSP)     ; 1.-
         LD      A,(DE)
         LD      C,A
         INC     DE
         LD      A,(DE)
         LD      B,A
         INC     DE
         LD      (RSP),DE     ; 2.-
         LD      (RPC),BC
         RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                          13/X/82
;                              (ABX)

; Inicio
;    1.- IX' : = IX + ACCB
: Fin
;        Nota.- Suma binaria
;.....................................................
;             CODIGO                          31/X/83
;                              (ABX)

     ABX   :  LD      HL,(RIX)      ; 1.-
              LD      BC,ACCB
              LD      A,(BC)
              LD      C,A
              LD      B,Ø
              ADD     HL,BC
              LD      (RIX),HL
              RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                          13/X/82
```

```
;                                   (RTI)

; Inicio
;    1.- CC' : = (SP), SP' : = SP + 1
;    2.- Si bit 7 (E) de Memoria (CC) : = 1
;        A.- Entonces:
;            1.- ACCA : = (SP), SP' : = SP + 1
;            2.- ACCB : = (SP), SP' : = SP + 1
;            3.- DP : = (SP), SP' : = SP + 1
;            4.- IXH : = (SP), SP' : = SP + 1
;            5.- IXL : = (SP), SP' : = SP + 1
;            6.- IYH : = (SP), SP' : = SP + 1
;            7.- IYL : = (SP), SP' : = SP + 1
;            8.- USH : = (SP), SP' : = SP + 1
;            9.- USL : = (SP), SP' : = SP + 1
;            10.- PCH : = (SP), SP' : = SP + 1
;            11.- PCL : = (SP), SP' : = SP + 1
;        B.- De lo contrario:
;            1.- PCH : = (SP), SP' : = SP + 1
;            2.- PCL : = (SP), SP' : = SP + 1
; Fin
;                Nota.- Banderas recobradas del Stack "S"
;.....................................................
;              CODIGO                          17/X/83
;                                   (RTI)

       RTI   :  LD      DE,(RSP)      ; 1.-
                LD      A,(DE)
                LD      (CC),A
                INC     DE
                BIT     7,A           ; 2.-
                JP      Z,RTI2B
                LD      A,(DE)        ; 2.A.1.-
                LD      (ACCA),A
```

```
                INC     DE
                LD      A,(DE)          ; 2.A.2.-
                LD      (ACCB),A
                INC     DE
                LD      A,(DE)          ; 2.A.3.-
                LD      (DP),A
                INC     DE
                LD      A,(DE)          ; 2.A.4.-
                LD      B,A
                INC     DE
                LD      A,(DE)          ; 2.A.5.-
                LD      C,A
                LD      (RIX),BC
                INC     DE
                LD      A,(DE)          ; 2.A.6,-
                LD      B,A
                INC     DE
                LD      A,(DE)          ; 2.A.7.-
                LD      C,A
                LD      (RIY),BC
                INC     DE
                LD      A,(DE)          ; 2.A.8.-
                LD      B,A
                INC     DE
                LD      A,(DE)          ; 2.A.9.-
                LD      C,A
                LD      (US),BC
                INC     DE
RTI2B :         LD      A,(DE)          ; 2.A.10.-
                LD      B,A
                INC     DE
                LD      A,(DE)          ; 2.A.11.-
                LD      C,A
                LD      (RPC),BC
```

```
                INC     DE
                LD      (RSP),DE
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           14/X/82
;                               (CWAI)

; Inicio
;     1.- Fin : = Falso
;     2.- Byte : = Memoria (PC)
;     3.- PC : = PC + 1
;     4.- CC' : = CC and Byte
;     5.- Bandera (E) : = 1
;     6.- SP' : = SP - 1, (SP) : = PCL
;     7.- SP' : = SP - 1, (SP) : = PCH
;     8.- SP' : = SP - 1, (SP) : = USL
;     9.- SP' : = SP - 1, (SP) : = USH
;     10.- SP' : = SP - 1, (SP) : = IYL
;     11.- SP' : = SP - 1, (SP) : = IYH
;     12.- SP' : = SP - 1, (SP) : = IXL
;     13.- SP' : = SP - 1, (SP) : = IXH
;     14.- SP' : = SP - 1, (SP) : = DP
;     15.- SP' : = SP - 1, (SP) : = ACCA
;     16.- SP' : = SP - 1, (SP) : = ACCB
;     17.- SP' : = SP - 1, (SP) : = CC
;     18.- Repite o llama Rutina SYNC
; Fin
;           Nota.- Banderas afectadas de acuerdo a la opera---
;                  ción and.
;.............................................................
;               CODIGO                          17/X/83
;                               (CWAI)
```

```
CWAI  :  LD      A,Ø            ; 1.-
         LD      (FIN5),A
         LD      DE,(RPC)       ; 2.-
         LD      A,(DE)
         INC     DE             ; 3.-
         LD      (RPC),DE
         LD      B,A            ; 4.-
         LD      A,(CC)
         AND     A,B
         SET     7,A            ; 5.-
         LD      (CC),A
         LD      DE,(RSP)       ; 6.-
         DEC     DE
         LD      BC,(RPC)
         LD      (DE),C
         DEC     DE
         LD      (DE),B         ; 7.-
         DEC     DE
         LD      BC,(US)        ; 8.-
         LD      (DE),C
         DEC     DE
         LD      (DE),B         ; 9.-
         DEC     DE
         LD      BC,(RIY)       ; 10.-
         LD      (DE),C
         DEC     DE
         LD      (DE),B         ; 11.-
         DEC     DE
         LD      BC,(RIX)       ; 12.-
         LD      (DE),C
         DEC     DE
         LD      (DE),B         ; 13.-
         DEC     DE
         LD      A,(DP)         ; 14.-
```

```
                LD      (DE),A
                DEC     DE
                LD      A,(ACCA)        ; 15.-
                LD      (DE),A
                DEC     DE
                LD      A,(ACCB)        ; 16.-
                LD      (DE),A
                DEC     DE
                LD      A,(CC)          ; 17.-
                LD      (DE),A
                LD      (RSP),DE
                CALL    SYNC            ; 18.-
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   14/X/82
;                               (MUL)

; Inicio
;     1.- Banderas (C y Z) : = Ø
;     2.- M (7 : Ø) ; = (ACCA)
;     3.- A (7 : Ø) : = (ACCB)
;     4.- A (15 : 8) : = Ø
;     5.- R (15 : Ø) : = Ø
;     6.- C ; = 8
;     7.- Repite
;         A.- M (7 : Ø) : = M (7 : Ø) / 2
;         B.- Si existe residuo en división anterior; Entonces
;             1.- R (15 : Ø) : = R (15 : Ø) + A (15 : Ø)
;         C.- A (15 : Ø) : = A (15 : Ø) * 2
;         D.- C : = C - 1
;     8.- Hasta C ; = Ø
;     9.- (ACCA) : = R (15 : 8)
;     10.- (ACCB) : = R (7 : Ø)
```

```
;    11.- Si bit 7 de ACCB : = 1; Entonces:
;         A.- Bandera (C) : = 1
;    12.- Si ACCA y ACCB : = Ø, Entonces:
;         A.- Bandera (Z) : = 1
; Fin.
;    Nota.- A : = Registro Temporal
;.....
;              CODIGO                          17/X/83
;                                (MUL)

       MUL    : LD      A,(CC)        ; 1.-
                AND     A,ØFAH
                LD      (CC),A
                LD      A,(ACCA)      ; 2.-
                LD      C,A
                LD      A,(ACCB)      ; 3.-
                LD      B,Ø           ; 4.-
                LD      HL,Ø          ; 5.-
                LD      D,Ø8H         ; 6.-
       MUL7A  : SRL     A             ; 7.-
                JP      NC,MUL7C
                ADD     HL,BC         ; 7.B.-
       MUL7C  : SLA     C             ; 7.C.-
                RL      B
                DEC     D             ; 7.D.-
                JP      NZ,MUL7A      ; 8.-
                LD      A,L           ; 9.-
                LD      (ACCB),A
                LD      B,A
                LD      A,H           ; 10.-
                LD      (ACCA),A
                BIT     7,B           ; 11.-
                JP      Z,MUL12
                LD      A,(CC)
```

```
                SET     Ø,A
        MUL12 : LD      DE,Ø            ; 12.-
                ADD     HL,DE
                JP      NZ,MUL13
                SET     2,A
        MUL13 : LD      (CC),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           14/X/82
;                               (SWI)

; Inicio
;     1.- Registro Temporal : = ØFFFAH
;     2.- Banderas (I) y (F) : = Ø
;     3.- Ejecuta (SW)
; Fin
;.....................................................................
;               CODIGO                          18/X/83
;                               (SWI)

        SWI   : LD      DE,ØFFFAH       ; 1.-
                LD      (REGINT),DE
                LD      A,(CC)          ; 2.-
                RES     6,A
                RES     4,A
                LD      (CC),A
                JP      SW              ; 3.-

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           14/X/82
;                               (GRUPO4)

; Inicio
```

```
;    1.- Clasifica Código de Inst. entre (Operación):
;        A.- X X X X Ø Ø 1 1 : SUB'AD
;        B.- X X X X 1 1 Ø Ø : OPEIV
;        C.- X X X X 1 1 Ø 1 : OPEV
;        D.- X X X X 1 1 1 X : LDSTXU
;        E.- OTROS           : NODEF
; Fin
;.....................................................
;            CODIGO                          18/X/83
;                          (GRUPO4)

       GRUPO4:  BIT     3,A          ; 1.-
                JP      NZ,GP4BCD
                BIT     2,A
                JP      NZ,NODEF     ; 1.E.-
                BIT     1,A
                JP      Z,NODEF      ; 1.E.-
                BIT     Ø,A
                JP      NZ,SUB'AD    ; 1.A.-
                JP      NODEF        ; 1.E.-
       GP4BCD:  BIT     2,A
                JP      Z,NODEF      ; 1.E.-
                BIT     1,A
                JP      NZ,LDSTXU    ; 1.D.-
                BIT     Ø,A
                JP      Z,OPEIV      ; 1.B.-
                JP      OPEV         ; 1.C.-

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                        14/X/82
;                          (SUB'AD)

; Inicio
;    1.- Clasifica Código de Inst. entre (Direccionamientos):
```

```
;         A.- X X Ø Ø X X X X : INMEDIATO de 16 bits
;         B.- X X Ø 1 X X X X : DIRECTO
;         C.- X X 1 Ø X X X X : INDEXADO
;         D.- X X 1 1 X X X X : EXTENDIDO
;     2.- Reg. Temp. : = ACCD
;     3.- Si bit 6 de Código de Inst. : = Ø
;         A.- Entonces; ACCD : = ACCD - Mem. (DIR. EFEC. + 1),
;             Mem. (DIR. EFEC.)
;         B.- De lo contrario; ACCD : = ACCD + Mem. (DIR. E--
;             FEC. + 1), Mem. (DIR. EFEC.)
;     4.- Banderas (N, Z, V y C) : = Ø
;     5.- Si bit 15 de ACCD : = 1
;         A.- Entonces; Bandera (N) : = 1
;     6.- Si ACCD : = Ø
;         A.- Entonces; Bandera (Z) : = 1
;     7.- Si (bit 15 or exc. bit 14) de ACCD : = 1
;         A.- Entonces; Bandera (V) : = 1
;     8.- Si Bandera de C de Z-80 : = 1, Entonces:
;         A.- Bandera (C) del 6809 : = 1
; Fin
;.....................................................
;              CODIGO                          18/X/83
;                          (SUB'AD)

       SUB'AD:  BIT     5,A          ; 1.-
                JP      NZ,SUB1C
                BIT     4,A
                JP      NZ,SUB1B
                CALL    INM16        ; 1.A.-
                JP      SUB?
       SUB1B :  CALL    DIRECTO      ; 1.B.-
                JP      SUB?
       SUB1C :  BIT     4,A
                JP      NZ,SUB1D
```

```
            CALL    INDEX           ; 1.C.-
            JP      SUB2
SUB1D :     CALL    EXTEN           ; 1.D.-
SUB2  :     LD      H,A             ; 2.-
            LD      BC,(DIREF)
            LD      A,(BC)
            LD      E,A
            INC     BC
            LD      A,(BC)
            LD      D,A
            BIT     6,H             ; 3.-
            JP      NZ,SUB3B
            LD      HL,(ACCD)
            LD      C,Ø
            OR      A
            SBC     HL,DE           ; 3.A.-
            JP      NC,SUB3AA
            LD      C,ØFFH
SUB3AA:     LD      (ACCD),HL
            JP      SUB4
SUB3B :     LD      HL,(ACCD)       ; 3.B.-
            ADD     HL,DE
            LD      C,Ø
            JP      NC,SUB3BB
            LD      C,ØFFH
SUB3BB:     LD      (ACCD),HL
SUB4  :     LD      A,(CC)          ; 4.-
            AND     A,ØFØH
            BIT     7,H             ; 5.-
            JP      Z,SUB6
            SET     3,A             ; 5.A.-
SUB6  :     LD      BC,Ø            ; 6.-
            ADD     HL,BC
            JP      NZ,SUB7
```

```
                SET     2,A             ; 6.A.-
                LD      (CC),A
        SUB7  : LD      A,H             ; 7.-
                AND     A,8ØH
                LD      B,A
                LD      A,H
                AND     A,4ØH
                SLA     A
                XOR     A,B
                JP      Z,SUB8
                LD      A,(CC)
                SET     1,A             ; 7.A.-
                LD      (CC),A
        SUB8  : BIT     Ø,C             ; 8.-
                JP      Z,SUB8A
                LD      A,(CC)
                SET     Ø,A             ; 8.A.-
                LD      (CC),A
        SUB8A : RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           14/X/82
;                               (OPEIV)

; Inicio
;    1.- Si bit 6 de Código de Inst. : = Ø
;        A.- Entonces; Ejecuta (CMPX)
;        B.- De lo contrario; Ejecuta (LDSTD)
; Fin
;.....................................................
;               CODIGO                          18/X/83
;                               (OPEIV)

        OPEIV : BIT     6,A             ; 1.-
```

```
                JP      Z,CMPX          ; 1.A.-
                JP      LDSTD           ; 1.B.-

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   14/X/82
;                               (CMPX)

; Inicio
;    1.- Registro : = Indice "X"
;    2.- Ejecuta (CMP16)
; Fin
;.....................................................................
;               CODIGO                                  18/X/83
;                               (CMPX)

       CMPX  :  LD      DE,(RIX)        ; 1.-
                LD      (RTRO),DE
                JP      CMP16           ; 2.-

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   14/X/82
;                               (LDSTD)

; Inicio
;    1.- Registro : = Acumulador "D"
;    2.- Ejecuta (LDST16)
; Fin
;.....................................................................
;               CODIGO                                  19/X/83
;                               (LDSTD)

       LDSTD :  LD      DE,ACCD         ; 1.-
                LD      (LDRTRO),DE
                JP      LDST16          ; 2.-
```

```
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                              15/X/82
;                              (OPEV)

; Inicio
;    1.- Clasifica Código de Inst. entre (instrucciones):
;        A.- X Ø Ø Ø X X X X : BSR
;        B.- X Ø Ø 1 X X X X : JSR
;        C.- X Ø 1 X X X X X : JSR
;        D.- X 1 Ø 1 X X X X : LDSTD
;        E.- X 1 1 X X X X X : LDSTD
;        F.- OTROS           : NODEF
; Fin
;...............................................................
;              CODIGO                               19/X/83
;                              (OPEV)

       OPEV  :  BIT     6,A          ; 1.-
                JP      NZ,OPEV1D
                BIT     5,A
                JP      NZ,JSR       ; 1.C.-
                BIT     4,A
                JP      NZ,JSR       ; 1.B.-
                JP      BSR          ; 1.A.-
       OPEV1D:  BIT     5,A
                JP      NZ,LDSTD     ; 1.E.-
                BIT     4,A
                JP      NZ,LDSTD     ; 1.D.-
                JP      NODEF        ; 1.F.-

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                              15/X/82
:                              (BSR)
```

```
; Inicio
;    1.- Reg. Temp. : = Memoria (PC)
;    2.- PC : = PC + 1
;    3.- SP' : = SP - 1, (SP) : = PCL
;    4.- SP' : = SP - 1, (SP) : = PCH
;    5.- PC' : = PC + Reg. Temp. - 1
; Fin
;    Nota._ SP = Apuntador del Stack Hardware
;.....................................................
;             CODIGO                          19/X/83
;                          (BSR)

      BSR   :  LD      DE,(RPC)      ; 1.-
               LD      A,(DE)
               LD      H,A
               INC     DE            ; 2.-
               LD      (RPC),DE
               LD      BC,(RSP)
               DEC     BC
               LD      A,E
               LD      (BC),A        ; 3.-
               DEC     BC
               LD      A,D
               LD      (BC),A        ; 4.-
               LD      (RSP),BC
               DEC     DE            ; 5.-
               LD      L,H
               LD      H,Ø
               ADD     HL,DE
               LD      (RPC),HL
               RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                          15/X/82
```

```
;                                   (JSR)

; Inicio
;    1.- Clasifica Código de Inst. entre (direccionamientos):
;        A.- X X Ø 1 X X X X : DIRECTO
;        B.- X X 1 Ø X X X X : INDEXADO
;        C.- X X 1 1 X X X X : EXTENDIDO
     2.- SP' : = SP - 1, (SP) : = PCL
     3.- SP' : = SP - 1, (SP) : = PCH
;    4.- PC : = Mem. (DIR. EFEC. + 1), Mem. (DIR. EFEC.)
; Fin
;.....................................................................
;              CODIGO                                19/X/83
;                                   (JSR)

       JSR    :  BIT     5,A          ; 1.-
                 JP      NZ,JSR1B
                 CALL    DIRECTO      ; 1.A.-
                 JP      JSR2
       JSR1B  :  BIT     4,A          ; 1.B.-
                 JP      NZ,JSR1C
                 CALL    INDEX
                 JP      JSR2
       JSR1C  :  CALL    EXTEN        ; 1.C.-
       JSR2   :  LD      DE,(RSP)     ; 2.-
                 DEC     DE
                 LD      BC,(RPC)
                 LD      A,C
                 LD      (DE),A
                 DEC     DE
                 LD      A,B
                 LD      (DE),A
                 LD      (RSP),DE
                 LD      DE,(DIREF)   ; 4.-
```

```
                LD      A,(DE)
                LD      C,A
                INC     DE
                LD      A,(DE)
                LD      B,A
                LD      (RPC),BC
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   15/X/82
;                               (LDSTXU)

; Inicio
;    1.- Si bit 6 de Código de Inst. : = Ø
;        A.- Entonces; Registro : = Indice "X"
;        B.- De lo contrario; Registro : = Stack "U"
;    2.- Ejecuta (LDST16)
; Fin
;.....................................................
;               CODIGO                          19/X/83
;                               (LDSTXU)

        LDSTXU: BIT     6,A             ; 1.-
                JP      NZ,LDSX16
                LD      DE,RIX          ; 1.A.-
                LD      (LDRTRO),DE
                JP      LDST16
        LDSX16: LD      DE,US           ; 1.B.-
                LD      (LDRTRO),DE
                JP      LDST16

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   15/X/82
;                               (GRUPO5)
```

```
; Inicio
;    1.- Clasifica Código de Inst. entre (direccionamientos)
;        A.- X X Ø Ø X X X X : INMEDIATO de 8 bits
;        B.- X X Ø 1 X X X X : DIRECTO
;        C.- X X 1 Ø X X X X : INDEXADO
;        D.- X X 1 1 X X X X : EXTENDIDO
;    2.- Si bit 6 de Código de Inst. : = Ø
;        A.- Entonces; Registro : = Acumulador "A"
;        B.- De lo contrario; Registro : = Acumulador "B"
;    3.- Clasifica Código de Inst. entre (operaciones):
;        A.- X X X X Ø Ø X X : SUBCMP
;        B.- X X X X Ø 1 X X : OPEVI
;        C.- X X X X 1 Ø X X : OPEVII
; Fin
;.....................................................
;            CODIGO                        19/X/83
;                          (GRUPO5)

      REG5  :  DS 2
      GRUPO5:  BIT     5,A          ; 1.-
               JP      NZ,GRU51C
               BIT     4,A
               JP      NZ,GRU51B
               CALL    INM8         ; 1.A.-
               JP      GRU52
      GRU51B:  CALL    DIRECTO      ; 1.B.-
               JP      GRU52
      GRU51C:  BIT     4,A          ; 1.C.-
               JP      NZ,GRU51D
               CALL    INDEX
               JP      GRU52
      GRU51D:  CALL    EXTEN        ; 1.D.-
      GRU52 :  LD      B,A          ; 2.-
               BIT     6,A
```

```
                JP      NZ,GRU52B
                LD      DE,ACCA         ; 2.A.-
                LD      (REG5),DE
                JP      GRU53
        GRU52B: LD      DE,ACCB         ; 2.B.-
                LD      (REG5),DE
        GRU53 : LD      A,B             ; 3.-
                BIT     3,A
                JP      NZ,OPEVII       ; 3.C.-
                BIT     2,A
                JP      NZ,OPEVI        ; 3.B.-
                JP      SUBCMP          ; 3.A.-

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       PSEUDO-CODIGO                                   15/X/82
;                               (INM8)

; Inicio
;     1.- Dirección Efectiva : = PC
;     2.- PC : = PC + 1
; Fin
;.....................................................................
;               CODIGO                                  19/X/83
;                               (INM8)

        INM8  : LD      DE,(RPC)        ; 1.-
                LD      (DIREF),DE
                INC     DE              ; 2.-
                LD      (RPC),DE
                RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       PSEUDO-CODIGO                                   15/X/82
;                               (SUBCMP)
```

```
; Inicio
;    1.- Reg. Temp. : = Registro - Memoria (DIR. EFEC.)
;    2.- Si Código de Inst. : = X X X X X X 1 Ø
;        A.- Reg. Temp. : = Reg. Temp. - C (Carry)
;    3.- Banderas N, Z, V y C : = Ø
;    4.- Si bit 7 de Reg. Temp. : = 1
;        A.- Bandera (N) : = 1
;    5.- Si Reg. Temp. : = Ø
;        A.- Bandera (Z) : = 1
;    6.- Si (bit 7 or exc. bit 6) de Reg. Temp. : = 1
;        A.- Bandera (V) : = 1
;    7.- Si bit 8 de Registro : = 1 y bit 8 de Reg. Temp. :
;        = 1
;        A.- Bandera (C) : = 1
;    8.- Si bit Ø de Código de Inst. : = Ø
;        A.- Registro : = Registro Temporal
;        B.- Memoria (DIR. EFEC.) : = Ø
; Fin
;        Nota.- La operación se realiza en 16 bits (aunque -
;               es de 8 bits) para evitar problemas con el -
;               manejo de la bandera de Carry.
;.....................................................
;            CODIGO                          20/X/83
;                          (SUBCMP)

      SUBCMP:  PUSH    AF            ; Salva Carry
               LD      DE,(DIREF)    ; 1.-
               LD      A,(DE)
               LD      E,A
               LD      D,Ø
               LD      HL,(REG5)
               LD      A,(HL)
               LD      L,A
               LD      H,Ø
```

```
            OR      A
            SBC     HL,DE
            LD      A,(CODINS)      ; 2.-
            BIT     1,A
            JP      Z,SBCP3
            BIT     Ø,A
            JP      NZ,SBCP3
            POP     BC
            BIT     Ø,C
            JP      Z,SBCP3
            DEC     HL              ; 2.A.-
SBCP3 :     LD      A,(CC)          ; 3.-
            AND     A,ØFØH
            LD      C,A
            BIT     7,L             ; 4.-
            JP      Z,SBCP5
            SET     3,C             ; 4.A.-
SBCP5 :     LD      DE,Ø            ; 5.-
            ADD     HL,DE
            JP      NZ,SBCP6
            SET     2,C             ; 5.A.-
SBCP6 :     BIT     7,L             ; 6.-
            JP      Z,SBCP6A
            BIT     6,L
            JP      NZ,SBCP7
            SET     1,C             ; 6.A.-
            JP      SBCP7
SBCP6A:     BIT     6,L
            JP      Z,SBCP7
            SET     1,C             ; 6.A.-
SBCP7 :     LD      DE,(REG5)       ; 7.-
            LD      A,(DE)
            JP      NC,SBCP8
            BIT     Ø,H
```

```
                JP      NZ,SBCP8
                SET     Ø,C             ; 7.A.-
        SBCP8 : LD      A,C             ; 8.-
                LD      (CC),A
                LD      A,(CODINS)
                BIT     Ø,A
                JP      NZ,SBCP9
                LD      DE,(REG5)       ; 8.A.-
                LD      A,L
                LD      (DE),A
                LD      DE,(DIREF)      ; 8.B.-
                LD      A,Ø
                LD      (DE),A
        SBCP9 : RET
;    Nota.- Reg. Temp. = Registro par HL
;           Registro = Etiqueta REG5

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                                  15/X/82
;                          (OPEVI)

; Inicio
;    1.- Si bit 1 de Código de Inst. : = Ø
;        A.- Entonces; Ejecuta (OPELOG)
;        B.- De lo contrario; Ejecuta (LDST8)
; Fin
;.........................................................
;             CODIGO                                   20/X/83
;                           (OPEVI)

        OPEVI : BIT     1,A             ; 1.-
                JP      Z,OPELOG        ; 1.A.-
                JP      LDST8           ; 1.B.-
```

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   15/X/82
;                                (OPELOG)

; Inicio
;     1.- Clasifica Código de Inst. entre (Operación Lógica):
;         A.- X X X X X Ø Ø X : Reg. Temp. ; = Reg. or exc.
;                                Mem. (DIR. EFEC.)
;         B.- X X X X X Ø 1 X : Reg. Temp. : = Reg or Mem. —
;                                (DIR. EFEC.)
;         C.- X X X X X 1 Ø X : Reg. Temp. : = Reg. and Mem.
;                                (DIR. EFEC.)
;     2.- Si bit Ø de Código de Inst. : = Ø
;         A.- Registro ; = Reg. Temp.
;     3.- Banderas (N, Z y V) : = Ø
;     4.- Si bit 7 de Reg. Temp. : = 1
;         A.- Bandera (N) : = 1
;     5.- Si Reg. Temp. : = Ø
;         A.- Bandera (Z) ; = 1
; Fin
;.....................................................................
;              CODIGO                                   20/X/83
;                                (OPELOG)

        OPELOG:  LD      H,A            ; 1.-
                 LD      DE,(REG5)
                 LD      A,(DE)
                 LD      B,A
                 LD      DE,(DIREF)
                 LD      A,(DE)
                 LD      C,A
                 BIT     2,H
                 JP      NZ,OPEL1C
                 BIT     1,H
```

```
                JP      NZ,OPEL1B
                LD      A,B             ; 1.A.-
                XOR     A,C             ; Resultado en Reg. A
                JP      OPEL2
OPEL1B:         LD      A,B             ; 1.B.-
                OR      A,C             ; Resultado en Reg. A
                JP      OPEL2
OPEL1C:         LD      A,B             ; 1.C.-
                AND     A,C             ; Resultado en Reg. A
OPEL2 :         LD      B,A             ; 2.-
                BIT     Ø,H
                JP      NZ,OPEL3
                LD      DE,(REG5)       ; 2.A.-
                LD      (DE),A
OPEL3 :         LD      A,(CC)          ; 3.-
                AND     A,ØF1H
                BIT     7,B             ; 4.-
                JP      Z,OPEL5
                SET     3,A             ; 4.A.-
OPEL5 :         LD      H,A             ; 5.-
                LD      A,B
                LD      B,Ø
                AND     A,B
                JP      NZ,OPEL5B
                SET     2,H             ; 5.A.-
OPEL5B:         LD      A,H
                LD      (CC),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           15/X/82
;                           (LDST8)

; Inicio.
```

```
;    1.- Si bit Ø de Código de Inst. : = Ø
;        A.- Entonces; Registro : = Memoria (DIR. EFEC.)
;        B.- De lo contrario:
;            1.- Si Código de Inst. : = X X Ø Ø X X X X
;                A.- Entonces; Ejecuta (NODEF)
;                B.- De lo contrario; Mem. (DIR. EFEC.) : =
;                    Registro
;    2.- Banderas (N, Z y C) : = Ø
;    3.- Si bit 7 de dato transferido : = 1
;        A.- Bandera (N) : = 1
;    4.- Si dato transferido : = Ø
;        A.- Bandera (Z) : = 1
; Fin
;.....................................................
;              CODIGO                        20/X/83
;                          (LDST8)

      LDST8 :  LD      H,A          ; 1.-
               LD      DE,(REG5)
               LD      BC,(DIREF)
               BIT     Ø,H
               JP      NZ,LDST8B
               LD      A,(BC)       ; 1.A.-
               LD      (DE),A
               JP      LDST82
      LDST8B:  BIT     5,H          ; 1.B.-
               JP      NZ,LDS81B
               BIT     4,H
               JP      NZ,LDS81B
               JP      NODEF        ; 1.B.1.A.-
      LDS81B:  LD      A,(DE)       ; 1.B.1.B.-
               LD      (BC),A
      LDST82:  LD      L,A          ; 2.-
               LD      A,(CC)
```

```
                AND     A,ØF1H
                BIT     7,L             ; 3.-
                JP      Z,LDST84
                SET     3,A             ; 3.A.-
                LD      (CC),A
      LDST84:   LD      H,Ø             ; 4.-
                LD      A,L
                ADD     A,H
                JP      NZ,LDST85
                LD      A,(CC)
                SET     2,A             ; 4.A.-
                LD      (CC),A
      LDST85:   RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           15/X/82
;                               (OPEVII)

; Inicio
;    1.- Si bit Ø de Código de Inst. : = Ø
;        A.- Entonces; Ejecuta (OPELOG)
;        B.- De lo contrario; Ejecuta (ADCADD)
; Fin
;.......................................................................
;               CODIGO                          20/X/83
;                               (OPEVII)

      OPEVII:   BIT     Ø,A             ; 1.-
                JP      Z,OPELOG        ; 1.A.-
                JP      ADCADD          ; 1.B.-

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           15/X/82
;                               (ADCADD)
```

```
; Inicio
;    1.- Registro : = Registro + Memoria (DIR. EFEC.)
;    2.- Si bit 1 de Código de Inst. : = Ø
;        A.- Registro : = Registro + C (Carry)
;    3.- Banderas (H, N, Z, V y C) : = Ø
;    4.- Si bit 4 de Registro : = 1
;        A.- Bandera (H) : = 1
;    5.- Si bit 7 de Registro ; = 1
;        A.- Bandera (N) : = 1
;    6.- Si Registro : = Ø
;        A.- Bandera (Z) : = 1
;    7.- Si (bit 7 or exc. bit 6) de Registro : = 1
;        A.- Bandera (V) : = 1
;    8.- Si bit 8 de Registro : = 1; Entonces
;        A.- Bandera (C) : = 1
; Fin
;    Nota.- La operación se realizará en 16 bits (aunque es
;           de 8 bits) para evitar problemas con el manejo -
;           de la bandera de Carry.
;.....................................................
;            CODIGO                          21/X/83
;                          (ADCADD)

       ADCADD:  PUSH    AF              ; Salva Carry
                LD      DE,(DIREF)      ; 1.-
                LD      A,(DE)
                LD      E,A
                LD      D,Ø
                LD      HL,(REG5)
                LD      A,(HL)
                LD      L,A
                LD      H,Ø
                ADD     HL,DE
                LD      A,(CODINS)      ; 2.-
```

```
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                  27/X/82
;                              (EJEPAS)

; Inicio
;    1.- Fin = Falso
;    2.- Ejecuta Rutina (OBTEN)
;    3.- Si Ø< = Caracter< = 9
;        A.- Entonces;
;            1.- n : = Caracter
;            2.- Mientras n< ó >Ø
;                A.- Ejecuta Rutina (IINST)
;                B.- Ejecuta Rutina (DESPAS)
;                C.- n = n - 1
; Fin
;.....................................................
;             CODIGO                                13/I/82
;                              (EJEPAS)

       EJEPAS:                          ; 1.-
                CALL    OBTEN           ; 2.-
                CP      A,3ØH           ; 3.-
                JP      M,PASO
                CP      A,3AH
                JP      P,PASO
                SUB     A,3ØH
       PASOA1:                          ; 3.A.1.-
                CP      A,Ø             ; 3.A.2.-
                JP      Z,PASO
                PUSH    AF
                CALL    IINST           ; 3.A.2.A.-
                CALL    DESPAS          ; 3.A.2.B.-
```

```
        BIT     1,A
        JP      NZ,ADC3
        POP     BC
        BIT     Ø,C
        JP      Z,ADC3
        INC     HL              ; 2.A.-
ADC3  : LD      A,L
        LD      DE,(REG5)
        LD      (DE),A
        LD      A,(CC)          ; 3.-
        AND     A,ØDØH
        BIT     4,L             ; 4.-
        JP      Z,ADC5
        SET     5,A             ; 4.A.-
ADC5  : BIT     7,L             ; 5.-
        JP      Z,ADC6
        SET     3,A
ADC6  : LD      DE,Ø            ; 6.-
        ADD     HL,DE
        JP      NZ,ADC7
        SET     2,A             ; 6.A.-
ADC7  : BIT     7,L             ; 7.-
        JP      Z,ADC7A
        BIT     6,L
        JP      NZ,ADC8
        SET     1,A             ; 7.A.-
        JP      ADC8
ADC7A : BIT     6,L
        JP      Z,ADC8
        SET     1,A             ; 7.A.-
ADC8  : BIT     Ø,H             ; 8.-
        JP      Z,ADC8A
        SET     Ø,A             ; 8.A.-
ADC8A : LD      (CC),A
```

```
                POP     AF
                DEC     A               ; 3.A.2.C.-
                JP      PASOA1
        PASO    ;       RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                       .   2/XII/82
;                               (DESPAS)

; Inicio
;    1.- Mensaje (1 : 9) : = 'b b A C C A b b b'
;    2.- Mensaje (10 : 18) : = 'b b A C C B b b b'
;    3.- Mensaje (19 : 25) : = 'b b D P b b b'
;    4.- Mensaje (26 : 32) : = 'b b C C b b b'
;    5.- Mensaje (33 : 42) : = 'b b b I X b b b b b'
;    6.- Mensaje (43 : 51) : = 'b b I Y b b b b b'
;    7.- Mensaje (56 : 60) : = 'b b U S b b b b b'
;    8.- Mensaje (61 : 69) : = 'b b S P b b b b b'
;    9.- Mensaje (70 : 78) : = 'b b P C b b b b b'
;   10.- Mensaje (79 : 81) : = ' CR '  ' LF '  ' $ '
;   11.- Mensaje (8 : 9) : = ASCII (Contenido del Reg. "A")
;   12.- Mensaje (17 : 18) : = ASCII (Cont. del Reg. "B")
;   13.- Mensaje (24 : 25) : = ASCII (Cont. del Reg. "DP")
;   14.- Mensaje (31 : 32) : = ASCII (Cont. del Reg. "CC")
;   15.- Mensaje (39 : 42) : = ASCII (Cont. del Reg. "IX")
;   16.- Mensaje (48 : 51) : = ASCII (Cont. del Reg. "IY")
;   17.- Mensaje (57 : 60) : = ASCII (Cont. del Reg. "US")
;   18.- Mensaje (66 : 69) : = ASCII (Cont. del Reg. "SP")
;   19.- Mensaje (75 : 78) : = ASCII (Cont. del Reg. "PC")
;   20.- Imprimir Mensaje (1 : 80)
;   21.- Mensaje (1 : 30) : = ' '
;   22.- Mensaje (3 : 10) : = 'BANDERAS'
;   23.- Mensaje (28 : 30) : = 'CR' 'LF' '$'
;   24.- Si Bandera de Estado Completo Salvado : = 1; Enton-
```

```
;          ces:
;          A.- Mensaje (13) ; = 'E'
;    25.- Si Bandera de Int. Rápida ; = 1; Entonces:
;          A.- Mensaje (15) : = 'F'
;    26.- Si Bandera de Carry Intermedio : = 1; Entonces:
;          A.- Mensaje (17) : = 'H'
;    27.- Si Bandera de Interrupción : = 1; Entonces:
;          A.- Mensaje (19) : = 'I'
;    28.- Si Bandera de Signo : = 1; Entonces:
;          A.- Mensaje (21) : = 'N'
;    29.- Si Bandera de Cero : = 1; Entonces:
;          A.- Mensaje (23) : = 'Z'
;    30.- Si Bandera de Overflow : = 1; Entonces:
;          A.- Mensaje (25) : = 'V'
;    31.- Si Bandera de Carry : = 1; Entonces:
;          A.- Mensaje (27) : = 'C'
;    32.- Imprimir Mensaje (1 : 30)
; Fin
;          Nota.- ' ' : = ASCII de un espacio.
;.....................................................
;               CODIGO                        22/I/82
;                              (DESPAS)

        CARRY :  DB 'C'
        CERO  :  DB 'Z'
        NEGAT :  DB 'N'
        SOBREF:  DB 'V'
        INT   :  DB 'I'
        INTF  :  DB 'F'
        MEDIO :  DB 'H'
        ENTERO:  DB 'E'
        MENSAJ:  DB 'bbACCAbbb'        ; 1.-
                 DB 'bbACCBbbb'        ; 2.-
                 DB 'bbDPbbb'          ; 3.-
```

```
                DB 'bbCCbbb'            ; 4.-
                DB 'bbbIXbbbbb'         ; 5.-
                DB 'bbIYbbbbb'          ; 6.-
                DB 'bbUSbbbbb'          ; 7.-
                DB 'bbSPbbbbb'          ; 8.-
                DB 'bbPCbbbbb'          ; 9.-
CR      :       EQU ØDH                 ; 10.-
IMPRIM:         EQU Ø9H
LF      :       EQU ØAH
FLAG    :       DB 'bbBANDERAS'
PESOS   :       EQU '$'
ASC8    :       MACRO   #DEST,#REG
                LD      HL,#DEST
                LD      A,(#REG)
                CALL    ASCII
                MEND
ASC16   :       MACRO   #DEST1,REG1
                LD      HL,#DEST1
                LD      A,(#REG1+1)
                CALL    ASCII
                LD      A,(#REG1)
                CALL    ASCII
                MEND
DESPAS:         LD      DE,BUFSIS
                LD      HL,MENSAJ
                LD      BC,8Ø
                LDIR
                ASC8    BUFSIS+7,ACCA   ; 11.-
                ASC8    BUFSIS+16,ACCB  ; 12.-
                ASC8    BUFSIS+23,DP    ; 13.-
                ASC8    BUFSIS+30,CC    ; 14.-
                ASC16   BUFSIS+38,RIX   ; 15.-
                ASC16   BUFSIS+47,RIY   ; 16.-
                ASC16   BUFSIS+56,US    ; 17.-
```

```
                ASC16   BUFSIS+65,RSP   ; 18.-
                ASC16   BUFSIS+74,RPC   ; 19.-
                LD      DE,BUFSIS+78
                LD      A,CR
                LD      (DE),A
                INC     DE
                LD      DE,BUFSIS+79
                LD      A,LF
                LD      (DE),A
                INC     DE
                LD      DE,BUFSIS+8Ø
                LD      A,PESOS
                LD      (DE),A
                INC     DE
                LD      C,IMPRIM        ; 20.-
                LD      DE,BUFSIS
                CALL    CDOS
                LD      A,Ø2ØH          ; 21.-
                LD      HL,FLAG
                LD      B,3Ø
DP21    :       LD      (HL),A
                INC     HL
                DJNZ    DP21
                LD      DE,BUFSIS       ; 22.-
                LD      HL,FLAG
                LD      BC,1Ø
                LDIR
                LD      DE,BUFSIS+27    ; 23.-
                LD      A,CR
                LD      (DE),A
                INC     DE
                LD      DE,BUFSIS+28
                LD      A,LF
                LD      (DE),A
```

```
        INC     DE
        LD      DE,BUFSIS+29
        LD      A,PESOS
        LD      (DE),A
        INC     DE
        LD      A,CC            ; 24.-
        BIT     7,A
        JP      Z,DP25
        LD      DE,BUFSIS+12
        LD      A,ENTERO
        LD      (DE),A
        INC     DE
DP25 :  BIT     6,A             ; 25.-
        JP      Z,DP26
        LD      DE,BUFSIS+14
        LD      A,INTF
        LD      (DE),A
        INC     DE
DP26 :  BIT     5,A             ; 26.-
        JP      Z,DP27
        LD      DE,BUFSIS+16
        LD      A,MEDIO
        LD      (DE),A
        INC     DE
DP27 :  BIT     4,A             ; 27.-
        JP      Z,DP28
        LD      DE,BUFSIS+18
        LD      A,INT
        LD      (DE),A
        INC     DE
DP28 :  BIT     3,A             ; 28.-
        JP      Z,DP29
        LD      DE,BUFSIS+2Ø
        LD      A,NEGAT
```

```
                LD      (DE),A
                INC     DE
        DP29  : BIT     2,A             ; 29.-
                JP      Z,DP3Ø
                LD      DE,BUFSIS+22
                LD      A,CERO
                LD      (DE),A
                INC     DE
        DP3Ø  : BIT     1,A             ; 30.-
                JP      Z,DP31
                LD      DE,BUFSIS+24
                LD      A,SOBREF
                LD      (DE),A
                INC     DE
        DP31  : BIT     Ø,A             ; 31.-
                JP      Z,DP32
                LD      DE,BUFSIS+26
                LD      A,CARRY
                LD      (DE),A
                INC     DE
        DP32  : LD      C,IMPRIM        ; 32.-
                LD      DE,BUFSIS
                CALL    CDOS
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           2/XII/82
;                               (ASCII)

; Inicio
;     1.- Valor ASCII # 1 : = (V binario/16) ∧ ØFH
;     2.- Si valor ASCII # 1 < ó = 9; Entonces:
;         A.- Valor ASCII # 1 : = Valor ASCII # 1 + 'Ø'
;         B.- De lo contrario:
```

```
;           1.- Valor ASCII # 1 : = Valor ASCII # 1 + 'A' -
;               1Ø
;    3.- Valor ASCII # 2 : = (V. binario)∧ØFH
;    4.- Si valor ASCII # 2<6 = 9
;        A.- Entonces; Valor ASCII # 2 : = Valor ASCII # 2 +
;            'Ø'
;        B.- De lo contrario;
;            1.- Valor ASCII # 2 : = Valor ASCII # 2 + 'A' -
;                1Ø
;    5.- Valor ASCII = Valor ASCII # 1, Valor ASCII # 2
; Fin
;    Nota.- 'Ø' : = ASCII del número Ø.
;.....................................................
;            CODIGO                          2/XII/82
;                          (ASCII)

      ASCII :  LD      B,A
               SRL     A              ; 1.- 1er. valor
               SRL     A
               SRL     A
               SRL     A
               AND     A,ØF
               CP      A,1Ø           ; 2.-
               JP      P,MAYOR
               ADD     A,'Ø'          ; 2.A.-
               JP      P,COMUN
      MAYOR :                         ; 2.B.-
               ADD     A,'A'-1Ø       ; 2.B.1.-
      COMUN :  LD      (HL),A         ; Carga 1er. valor en HL
               INC     HL
               LD      A,B            ; 2o. valor
               AND     A,ØF           ; 3.-
               CP      A,1Ø           ; 4.-
               JP      P,MAYOR1
```

```
                ADD     A,'Ø'           ; 4.A.-
                JP      COMUN1
        MAYOR1:                         ; 4.B.-
                ADD     A,'A'-1Ø        ; 4.B.1.-
        COMUN1: LD      (HL),A          ; 5.-
                INC     HL
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   3/XII/82
;                               (DESP)

; Inicio
;     1.- Ejecuta Rutina (OBTEN)
;     2.- Si caracter : = M
;         A.- Entonces; Ejecuta Rutina (INIFIN)
;         B.- De lo contrario; Ejecuta Rutina (DESREG)
; Fin
;.....................................................
;               CODIGO                                  2/II/83
;                               (DESP)

        DESP  : CALL    OBTEN           ; 1.-
                CP      A,'M'           ; 2.-
                JP      NZ,DESP2B
                CALL    INIFIN          ; 2.A.-
                RET
        DESP2B: CALL    DESREG          ; 2.B.-
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   6/XII/82
;                               (INIFIN)
```

```
; Inicio
;    1.- Ejecuta Rutina (OBTDIR)
;    2.- Inicio : = Dirección
;    3.- Ejecuta Rutina (OBTEN)
;    4.- Clasifica caracter entre:
;        A.-  "," ; Entonces:
;            1.- Ejecuta Rutina (OBTEN)
;            2.- Si Ø ∠o = caracter ∠ o = 9 ;
;                A.- Entonces:
;                    1.- Fin : = Inicio + caracter
;                    2.- Ejecuta Rutina (DESLOC)
;                B.- De lo contrario:
;                    1.- Ejecuta Rutina (NODEF)
;        B.- "1" ; Entonces:
;            1.- Ejecuta Rutina (OBTDIR)
;            2.- Fin : = Dirección
;            3.- Ejecuta Rutina (DESLOC)
;        C.- "CR" ; Entonces:
;            1.- Fin : = Inicio + 32Ø
;            2.- Ejecuta Rutina (DESLOC)
;        D.- Otros; Ejecuta Rutina (NODEF)
; Fin
;.....................................................
;            CODIGO                        7/II/83
;                          (INIFIN)

INIFIN:  CALL    OBTDIR        ; 1.-
         LD      BC,(DIR)      ; 2.-
         LD      (INICIO),BC
         CALL    OBTEN         ; 3.-
         CP      A,','         ; 4.A.-
         JP      NZ,INI4B
         CALL    OBTEN         ; 4.A.1.-
         CP      A,3ØH         ; 4.A.2.-
```

```
                JP      M,INI4A2
                CP      A,3AH
                JP      P,INI4A2
                LD      HL,(INICIO)
                SUB     A,3ØH
                OR      A
                LD      C,A             ; 4.A.2.A.1.-
                LD      B,Ø
                ADD     HL,BC
                LD      (FIN1),HL
                CALL    DESLOC          ; 4.A.2.A.2.-
                RET
INI4A2:         CALL    NODEF           ; 4.A.2.B.-
                RET
INI4B :         CP      A,'1'           ; 4.B.-
                JP      NZ,INI4C
                CALL    OBTDIR          ; 4.B.1.-
                LD      HL,(DIR)        ; Fin en Reg. HL
                LD      (FIN1),HL       ; 4.B.2.-
                CALL    DESLOC          ; 4.B.3.-
                RET
INI4C :         CP      A,'CR'          ; 4.C.-
                JP      NZ,INI4D
                LD      HL,(INICIO)
                LD      BC,32Ø          ; 4.C.1.-
                OR      A
                ADD     HL,BC
                LD      (FIN1),HL
                CALL    DESLOC          ; 4.C.2.-
                RET
INI4D :                                 ; 4.D.-
                CALL    NODEF
                RET
```

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                          8/XII/82
;                              (DESLOC)

; Inicio
;     1.- Mensaje (1 : 13) : = ' b b b b b D I R. b b b b '
;     2.- Mensaje (14 : 17) : = ' + Ø b b '
;     3.- Mensaje (18 : 21) : = ' + 1 b b '
;     4.- Mensaje (22 : 25) : = ' + 2 b b '
;     5.- Mensaje (26 : 29) : = ' + 3 b b '
;     6.- Mensaje (30 : 33) : = ' + 4 b b '
;     7.- Mensaje (34 : 37) : = ' + 5 b b '
;     8.- Mensaje (38 : 41) : = ' + 6 b b '
;     9.- Mensaje (42 : 45) : = ' + 7 b b '
;    10.- Mensaje (46 : 49) : = ' + 8 b b '
;    11.- Mensaje (50 : 53) : = ' + 9 b b '
;    12.- Mensaje (54 : 57) : = ' + A b b '
;    13.- Mensaje (58 : 61) : = ' + B b b '
;    14.- Mensaje (62 : 65) : = ' + C b b '
;    15.- Mensaje (66 : 69) : = ' + D b b '
;    16.- Mensaje (70 : 73) : = ' + E b b '
;    17.- Mensaje (74 : 77) : = ' + F b b '
;    18.- Mensaje (78 : 80) : = 'CR' 'LF' '$'
;    19.- Imprime mensaje (1 : 80)
;    20.- Repite
;         A.- Mensaje (1 : 80) : = ' '
;         B.- X : = BCD menos significativo de la dirección
;             de inicio
;         C.- Valor : = Dirección de inicio ∧FFFØH
;         D.- Mensaje (6 : 9) : = Valor (hexadecimal)
;         E.- Repite
;             1.- Mensaje ((14 + 4X) : (15 + 4X)) : ASCII ---
;                 (contenido de la dirección de inicio)
;             2.- Dir. de inicio : = Dir. de inicio + 1
```

```
;            3.- X : = X + 1
;        F.- Hasta X : = 16 o Dir. de inicio : = Dir. de Fin
;            + 1
;        G.- Mensaje (78) : = 'CR'
;        H.- Mensaje (79) : = 'LF'
;        I.- Mensaje (80) : = '$'
;        J.- Imprime mensaje (1 : 80)
;   21.- Hasta Dir de inicio : = Dir. de Fin + 1
; Fin
;.....................................................
;              CODIGO                        7/II/83
;                          (DESLOC)

      DIRE  :  DB 'bbbbbDIR.bbbb'      ; 1.-
               DB '+Øbb'               ; 2.-
               DB '+1bb'               ; 3.-
               DB '+2bb'               ; 4.-
               DB '+3bb'               ; 5.-
               DB '+4bb'               ; 6.-
               DB '+5bb'               ; 7.-
               DB '+6bb'               ; 8.-
               DB '+7bb'               ; 9.-
               DB '+8bb'               ; 10.-
               DB '+9bb'               ; 11.-
               DB '+Abb'               ; 12.-
               DB '+Bbb'               ; 13.-
               DB '+Cbb'               ; 14.-
               DB '+Dbb'               ; 15.-
               DB '+Ebb'               ; 16.-
               DB '+Fbb'               ; 17.-
               DB 'CR'                 ; 18.-
               DB 'LF'
               DB '$'
```

```
INICIO:  DS 2
VALORX:  DS 2
DESFIN:  DB Ø
DESLOC:  LD      DE,BUFSIS
         LD      HL,DIRE
         LD      BC,8Ø
         LDIR
         LD      C,IMPRIM
         LD      DE,BUFSIS
         CALL    CDOS
DESL2Ø:                          ; 20.-
         LD      A,Ø2ØH          ; 20.A.-
         LD      HL,BUFSIS
         LD      B,8Ø
LOOP5 :  LD      (HL),A
         INC     HL
         DJNZ    LOOP5
         LD      HL,(INICIO)     ; 20.B.-
         LD      A,L
         AND     A,ØØFH
         LD      F,A
         LD      A,Ø
         LD      (VALORX),AF
         LD      A,L             ; 20.C.-
         AND     A,ØFØH
         LD      L,A
         LD      (INICIO),HL     ; inicio = XXXØ
         ASC16   BUFSIS+5,INICIO ; 20.D.-
LOOP6 :  LD      BC,(VALORX)     ; X en Reg. C
         LD      HL,ØØ           ; 20.E.-
         ADD     HL,BC           ; 20.E.1.-
         ADD     HL,BC
         ADD     HL,BC
         ADD     HL,BC
```

```
LD      BC,ØØEH
ADD     HL,BC
LD      BC,BUFSIS
ADD     HL,BC
LD      DE,(INICIO)
LD      A,(DE)
CALL    ASCII
INC     DE
LD      HL,(INICIO)       ; 20.E.2.-
INC     (HL)
LD      (INICIO),HL
LD      BC,(VALORX)       ; 20.E.3.-
INC     BC
LD      (VALORX),BC
LD      DE,(FIN1)         ; 20.F.-
INC     DE
OR      A                 ; borra carry
SBC     HL,DE
JP      P,FINALI
LD      DE,(FIN2)
INC     DE
LD      HL,(INICIO)
OR      A
SBC     HL,DE
JP      P,FINALI
LD      DE,(FIN3)
INC     DE
LD      HL,(INICIO)
OR      A
SBC     HL,DE
JP      P,FINALI
OR      A
LD      A,Ø1ØH
LD      BC,(VALORX)
```

```
                CP      A,C
                JP      Z,DES2ØG
                JP      LOOP6
        FINALI: LD      A,ØFFH
                LD      (DESFIN),A
        DES2ØG: LD      DE,BUFSIS+77    ; 20.G.-
                LD      A,CR
                LD      (DE),A
                INC     DE              ; 20.H.-
                LD      A,LF
                LD      (DE),A
                INC     DE              ; 20.I.-
                LD      A,PESOS
                LD      (DE),A
                INC     DE
                LD      C,IMPRIM        ; 20.J.-
                LD      DE,BUFSIS
                CALL    CDOS
                LD      B,ØFFH
                LD      A,(DESFIN)
                CP      A,B
                JP      NZ,DESL2Ø
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           7/XII/82
;                           (DESREG)

; Inicio
;     1.- Mensaje (1 : 16) : = ' '
;     2.- Ejecuta Rutina (OBTEN)
;     3.- Clasifica caracter entre:
;         A.- "A" : Mensaje (9 : 10) ; = ASCII (cont. en Reg.
;             "A")
```

B.- "B" : Mensaje (9 : 10) : = ASCII (cont. de Reg. "B")

C.- "I" ; Entonces:

1.- Ejecuta Rutina (OBTEN)

2.- Clasifica caracter entre:

A.- "X" : Mensaje (9 : 12) : = ASCII (cont. de Reg. IX)

B.- "Y" : Mensaje (9 : 12) : = ASCII (cont. de Reg. IY)

C.- Otros : Ejecuta Rutina (NULO)

D.- "U" ; Entonces:

1.- Ejecuta Rutina (OBTEN)

2.- Si caracter : = "S"

A.- Entonces; Mensaje (9 : 12) : = ASCII -- (cont. Reg. US)

B.- De lo contrario Ejecuta Rutina (NULO)

E.- "S" ; Entonces:

1.- Ejecuta Rutina (OBTEN)

2.- Si caracter : = "P"

A.- Entonces; Mensaje (9 : 12) : = ASCII -- (cont. Reg. SP)

B.- De lo contrario; Ejecuta Rutina (NULO)

F.- "P" ; Entonces:

1.- Ejecuta Rutina (OBTEN)

2.- Si caracter : = "C"

A.- Entonces; Mensaje (9 : 12) : = ASCII -- (cont. Reg. PC)

B.- De lo contrario; Ejecuta Rutina (NULO)

G.- "D" ; Entonces:

1.- Ejecuta Rutina (OBTEN)

2.- Clasifica caracter entre:

A.- "P" : Mensaje (9 : 12) : = ASCII (cont. Reg. DP)

B.- "CR" : Entonces:

```
;                       1.- Mensaje (9 : 12) : = ASCII (cont. -
;                           Reg. A)
;                       2.- Mete caracter ASCII (cont. Reg. B)
;                   C.- Otros : Ejecuta Rutina (NULO)
;           H.- "C" ; Entonces:
;               1.- Ejecuta Rutina (OBTEN)
;               2.- Si caracter : = "C"
;                   A.- Entonces: Mensaje (9 : 12) : = ASCII --
;                       (cont. Reg. CC)
;                   B.- De lo contrario: Ejecuta Rutina (NULO)
;           I.- "R" ; Ejecuta Rutina (DESPAS)
;           J.- Otros : Ejecuta Rutina (NULO)
;       4.- Mensaje (14) : = 'CR'
;       5.- Mensaje (15) : = 'LF'
;       6.- Mensaje (16) : = '$'
;       7.- Imprime mensaje (1 : 16)
; Fin
;.....................................................
;               CODIGO                          10/II/83
;                               (DESREG)

        DESREG:  LD      A,Ø2ØH          ; 1.-
                 LD      HL,BUF
                 LD      B,16
        DESR1 :  LD      (HL),A
                 INC     HL
                 DJNZ    DESR1
                 CALL    OBTEN           ; 2.-
                 CP      A,'A'           ; 3.-
                 JP      NZ,DESR3B
                 LD      DE,BUF
                 LD      HL,ACUMA
                 LD      BC,7
                 LDIR
```

```
                ASC8    BUF+8,ACCA      ; 3.A.-
                JP      DESR4
DESR3B:         CP      A,'B'           ; 3.B.-
                JP      NZ,DESR3C
                LD      DE,BUF
                LD      HL,ACUMB
                LD      BC,7
                LDIR
                ASC8    BUF+8,ACCB
                JP      DESR4
DESR3C:         CP      A,'I'           ; 3.C.-
                JP      NZ,DESR3D
                CALL    OBTEN           ; 3.C.1.-
                CP      A,'X'           ; 3.C.2.-
                JP      NZ,DR3C2B
                LD      DE,BUF+1
                LD      HL,INDX
                LD      BC,6
                LDIR
                ASC16   BUF+8,RIX       ; 3.C.2.A.-
                JP      DESR4
DR3C2B:         CP      A,'Y'           ; 3.C.2.B.-
                JP      NZ,NULO         ; 3.C.2.C.-
                LD      DE,BUF+1
                LD      HL,INDY
                LD      BC,6
                LDIR
                ASC16   BUF+8,RIY
                JP      DESR4
DESR3D:         CP      A,'U'           ; 3.D.-
                JP      NZ,DESR3E
                CALL    OBTEN           ; 3.D.1.-
                CP      A,'S'           ; 3.D.2.-
                JP      Z,DR3D2B        ; 3.D.2.A.
```

```
            JP      NULO            ; 3.D.2.B.-
DR3D2B:     LD      DE,BUF+1
            LD      HL,STACKU
            LD      BC,6
            LDIR
            ASC16   BUF+8,US
            JP      DESR4
DESR3E:     CP      A,'S'           ; 3.E.-
            JP      NZ,DESR3F
            CALL    OBTEN           ; 3.E.1.-
            CP      A,'P'           ; 3.E.2.-
            JP      Z,DR3E2B        ; 3.E.2.A.-
            JP      NULO            ; 3.E.2.B.-
            LD      DE,BUF+1
            LD      HL,STACKS
            LD      BC,6
            LDIR
DR3E2B:
            ASC16   BUF+8,RSP
            JP      DESR4
DESR3F:     CP      A,'P'           ; 3.F.-
            JP      NZ,DESR3G
            CALL    OBTEN           ; 3.F.1.-
            CP      A,'C'           ; 3.F.2.-
            JP      Z,DR3F2B        ; 3.F.2.A.-
            JP      NULO            ; 3.F.2.B.-
            LD      DE,BUF+1
            LD      HL,CONPRO
            LD      BC,6
            LDIR
DR3F2B:
            ASC16   BUF+8,RPC
            JP      DESR4
DESR3G:     CP      A,'D'           ; 3.G.-
```

```
                JP      NZ,DESR3H
                CALL    OBTEN           ; 3.G.1.-
                CP      A,'P'           ; 3.G.2.-
                JP      NZ,DR3G2B
                LD      DE,BUF+1
                LD      HL,DIRPAG
                LD      BC,6
                LDIR
                ASC16   BUF+8,DP        ; 3.G.2.A.-
                JP      DESR4
DR3G2B:         CP      A,'CR'          ; 3.G.2.B.-
                JP      NZ,NULO         ; 3.G.2.C.-
                LD      DE,BUF
                LD      HL,ACUMD
                LD      BC,7
                LDIR
                ASC8    BUF+8,ACCA      ; 3.G.2.B.1.-
                ASC8    BUF+1Ø,ACCB
                LD      A,'CR'          ; 3.G.2.B.2.-
                JP      DESR4
DESR3H:         CP      A,'C'           ; 3.H.-
                JP      NZ,DESR3I
                CALL    OBTEN           ; 3.H.1.-
                CP      A,'C'           ; 3.H.2.-
                JP      Z,DR3H2B        ; 3.H.2.A.-
                JP      NULO            ; 3.H.2.B.-
                LD      DE,BUF+1
                LD      HL,CODCON
                LD      BC,6
                LDIR
DR3H2B:
                ASC8    BUF+8,CC
                JP      DESR4
DESR3I:         CP      A,'R'           ; 3.I.-
```

```
                JP      NZ,NULO         ; 3.-
                JP      DESPAS
        DESR4 : LD      DE,BUF+13       ; 4.-
                LD      A,CR
                LD      (DE),A
                INC     DE              ; 5.-
                LD      A,LF
                LD      (DE),A
                INC     DE              ; 6.-
                LD      A,PESOS
                LD      (DE),A
                LD      C,IMPRIM        ; 7.-
                LD      DE,BUF
                CALL    CDOS
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           7/XII/82
;                           (NULO)

; Inicio
;    1.- Mensaje (1 : 30) : = ' '
;    2.- Mensaje (5 : 25) : = 'REGISTRO DESCONOCIDO'
;    3.- Mensaje (26 : 28) : = 'CR' 'LF' '$'
;    4.- Imprime mensaje (1 : 30)
; Fin
;    Nota.-  ' ' : = Espacio en blanco.
;.....................................................
;               CODIGO                          10/II/83
;                           (NULO)

        DESCON: DB 'REGISTRObbDESCONOCIDO'
        NULO  : LD      A,Ø20H          ; 1.-
                LD      HL,BUFSIS
```

```
                LD      B,3Ø
        NUL01 : LD      (HL),A
                INC     HL
                DJNZ    NUL01
                LD      DE,BUFSIS+4     ; 2.-
                LD      HL,DESCON
                LD      BC,21
                LDIR
                LD      DE,BUFSIS+25    ; 3.-
                LD      A,CR
                LD      (DE),A
                INC     DE
                LD      A,LF
                LD      (DE),A
                INC     DE
                LD      A,PESOS
                LD      (DE),A
                LD      C,IMPRIM        ; 4.-
                LD      DE,BUFSIS
                CALL    CDOS
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           8/XII/82
;                           (SUST)

; Inicio
;     1.- Ejecuta Rutina (OBTEN)
;     2.- Si caracter : = "M"
;         A.- Entonces; Ejecuta Rutina (SUSLOC)
;         B.- De lo contrario; Ejecuta Rutina (SUSREG)
; Fin
;.....................................................
```

```
;              CODIGO                              12/II/83
;                              (SUST)

       SUST   :  CALL     OBTEN            ; 1.-
                 CP       A,'M'            ; 2.-
                 JP       NZ,SUST2B
                 CALL     SUSLOC           ; 2.A.-
                 RET
       SUST2B:   CALL     SUSREG           ; 2.B.-
                 RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      PSEUDO-CODIGO                              14/XII/82
;                              (SUSLOC)

; Inicio
;    1.- Mensaje (1 : 76) : = ' '
;    2.- Mensaje (6 : 17) : = 'Dirección ='
;    3.- Mensaje (23) : = 'H'
;    4.- Mensaje (29 : 40) : = 'Contenido ='
;    5.- Mensaje (45) : = 'H'
;    6.- Mensaje (51 : 72) : = 'Nuevo contenido (H) ='
;    7.- Mensaje (73) ; = '$'
;    8.- Ejecuta Rutina (OBTDIR)
;    9.- Inicio : = Dirección
;   10.- Fin : = Falso
;   11.- Repite
;        A.- Mensaje (19 : 22) : = Valor (inicio)
;        B.- Mensaje (42 : 43) : = ASCII (cont. de inicio)
;        C.- Imprime Mensaje (1 : 73)
;        D.- Ejecuta Rutina (BUFFER)
;        E.- Ejecuta Rutina (OBTEN)
;        F.- Clasifica caracter entre:
```

```
;            1.- 'Ø' hasta 'F' ; Entonces:
;                A.- Valor : = Ordinal de caracter
;                B.- Ejecuta Rutina (OBTEN)
;                C.- Clasifica caracter entre:
;                    1.- 'Ø' hasta 'F' ; Entonces:
;                        A.- Valor : = Valor * 1ØH + Ord. de
;                            caracter
;                        B.- Memoria (INICIO) : = Valor
;                        C.- Inicio = Inicio + 1
;                    2.- 'CR' ; Entonces:
;                        A.- Memoria (Inicio) : = Valor
;                        B.- Inicio : = Inicio + 1
;                    3.- '1' ; Entonces:
;                        A.- Memoria (Inicio) : = Valor
;                        B.- Fin : = Verdadero
;                    4.- Otros : Ejecuta Rutina (ERROR)
;            2.- 'CR' ; Inicio : = Inicio + 1
;            3.- '1' ; Fin : = Verdadero
;            4.- Otros ; Ejecuta Rutina (ERROR)
;  12.- Hasta Fin : = Verdadero
; Fin
;.....................................................
;             CODIGO                          14/II/83
;                          (SUSLOC)

      MENSA :  DB 'bbbbbDIRECCIONb=b'
               'bbbbbH'
               'bbbbbCONTENIDOb=b'
               'bbbbH'
               'bbbbbNUEVO CONTENIDO (H)b=b'
               '$'
     SUSLOC:  LD      DE,BUFSIS        ; 1.- hasta 7.-
              LD      HL,MENSA
              LD      BC,73
```

```
            LDIR
            CALL    OBTDIR          ; 8.-
            LD      BC,(DIR)        ; 9.-
            LD      (INICIO),BC
            LD      A,Ø             ; 10.-
            LD      (FIN),A
SUSL11:                             ; 11.-
            ASC16   BUFSIS+18,INICIO ; 11.A.-
            LD      HL,BUFSIS+41    ; 11.B.-
            LD      DE,(INICIO)
            LD      A,(DE)
            CALL    ASCII
            LD      DE,BUFSIS       ; 11.C.-
            LD      C,IMPRIM
            CALL    CDOS
            CALL    BUFFER          ; 11.D.-
            CALL    OBTEN           ; 11.E.-
            CP      A,3ØH           ; 11.F.1.-
            JP      M,SUSLF2
            CP      A,3AH
            JP      P,SUSLF1
            SUB     A,3ØH
            JP      SLF1A
SUSLF1:     CP      A,41H
            JP      M,ERROR
            CP      A,47H
            JP      P,ERROR
            SUB     A,37H
SLF1A :     LD      B,A             ; 11.F.1.A.-
            CALL    OBTEN           ; 11.F.1.B.-
            CP      A,3ØH           ; 11.F.1.C.-
            JP      M,SLF1C2
            CP      A,3AH
            JP      P,SLF1C
```

```
        SUB     A,3ØH
        LD      C,A
        JP      SL1C1A
SLF1C : CP      A,41H
        JP      M,ERROR
        CP      A,47H
        JP      P,ERROR
        SUB     A,37H
        LD      C,A
SL1C1A:                         ; 11.F.1.C.1.A.-
        LD      A,B
        ADD     A,A
        ADD     A,A
        ADD     A,A
        ADD     A,A
        ADD     A,C
        LD      DE,(INICIO)     ; 11.F.1.C.1.B.-
        LD      (DE),A
        INC     DE              ; 11.F.1.C.1.C.-
        LD      (INICIO),DE
        JP      SUSL11
SLF1C2: CP      A,'CR'          ; 11.F.1.C.2.-
        JP      NZ,SLF1C3
        LD      A,B             ; 11.F.1.C.2.A.-
        LD      (DE),INICIØ
        LD      (DE),A
        INC     DE              ; 11.F.1.C.2.B.-
        LD      (INICIO),DE
        JP      SUSL11
SLF1C3: CP      A,'/'           ; 11.F.1.C.3.-
        JP      NZ,ERROR        ; 11.F.1.C.4.-
        LD      A,B             ; 11.F.1.C.3.A.-
        LD      DE,(INICIO)
        LD      (DE),A
```

```
                JP      SUSL12          ; 11.F.1.C.3.-
SUSLF2:         CP      A,'CR'          ; 11.F.2.-
                JP      NZ,SUSLF3
                LD      DE,(INICIO)
                INC     DE
                LD      (INICIO),DE
                JP      SUSL11
SUSLF3:         CP      A,'/'           ; 11.F.3.-
                JP      NZ,ERROR        ; 11.F.4.-
SUSL12:         LD      A,ØFF           ; 12.-
                LD      (FIN),A
                RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                           14/XII/82
;                           (ERROR)

; Inicio
;    1.- Mensaje (1 : 25) ; = ' '
;    2.- Mensaje (5 : 22) : = 'ERROR EN COMANDO'
;    3.- Mensaje (23) : = 'CR'
;    4.- Mensaje (24) : = 'LF'
;    5.- Mensaje (25) : = '$'
;    6.- Imprime Mensaje (1 : 25)
; Fin
;.....................................................................
;               CODIGO                          16/II/83
;                           (ERROR)

        ERRCOM:  DB 'ERROR EN COMANDO'
        ERROR :  LD     A,Ø2ØH          ; 1.-
                 LD     HL,BUFSIS
                 LD     B,25
        ERROR1:  LD     (HL),A
```

```
                INC     HL
                DJNZ    ERROR1
                LD      DE,BUFSIS+4
                LD      HL,ERRCOM       ; 2.-
                LD      BC,16
                LDIR
                LD      DE,BUFSIS+22    ; 3.-
                LD      A,CR
                LD      (DE),A
                INC     DE              ; 4.-
                LD      A,LF
                LD      (DE),A
                INC     DE              ; 5.-
                LD      A,PESOS
                LD      (DE),A
                LD      DE,BUFSIS       ; 6.-
                LD      C,IMPRIM
                CALL    CDOS
                RET

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;       PSEUDO-CODIGO                                   15/XII/82
;                               (SUSREG)

; Inicio
;    1.- Mensaje (1 : 48) : = ' '
;    2.- ERROR : = Falso
;    3.- Ejecuta Rutina (OBTEN)
;    4.- Clasifica caracter entre:
;        A.- 'A' ; Entonces:
;            1.- Registro : = Acumulador "A"
;            2.- Mensaje (9 : 15) : = 'ACCA : = '
;            3.- Mensaje (18 : 19) : = ASCII (Memoria (ACCA))
;        B.- 'B' ; Entonces:
```

```
;           1.- Registro : = Acumulador "B"
;           2.- Mensaje (9 : 15) : = 'ACCB : ='
;           3.- Mensaje (18 : 19) : = ASCII (Memoria (ACCB))
;       C.- 'I' ; Entonces:
;           1.- Ejecuta Rutina (OBTEN)
;           2.- Clasifica caracter entre:
;               A.- 'X' ; Entonces:
;                   1.- Registro : = Indice "X"
;                   2.- Mensaje (10 : 15) : = 'IX  : ='
;                   3.- Mensaje (17 : 20) : = ASCII (Memo--
;                       ria (IX))
;               B.- 'Y' ; Entonces:
;                   1.- Registro : = Indice "Y"
;                   2.- Mensaje (10 : 15) : = 'IY  : ='
;                   3.- Mensaje (17 : 20) : ASCII (Memoria
;                       (IY))
;               C.- OTROS; Entonces:
;                   1.- Ejecuta Rutina (NULO)
;                   2.- ERROR : = Verdadero
;       D.- 'U'; Entonces:
;           1.- Ejecuta Rutina (OBTEN)
;           2.- Si caracter : = 'S'
;               A.- Entonces:
;                   1.- Registro : = Stack del Usuario
;                   2.- Mensaje (10 : 15) : = 'US  : ='
;                   3.- Mensaje (17 : 20) : = ASCII (Memo--
;                       ria (US))
;               B.- De lo contrario:
;                   1.- Ejecuta Rutina (NULO)
;                   2.- ERROR : = Verdadero
;       E.- 'S'; Entonces:
;           1.- Ejecuta Rutina (OBTEN)
;           2.- Si caracter : = 'P'
;               A.- Entonces:
```

```
;                       1.- Registro : = Stack Hardware
;                       2.- Mensaje (10 : 15) : = 'SP  : ='
;                       3.- Mensaje (17 : 20) : = ASCII (Memo-
;                           ria (SP))
;                   B.- De lo contrario:
;                       1.- Ejecuta Rutina (NULO)
;                       2.- ERROR : = Verdadero
;           F.- 'P'; Entonces:
;               1.- Ejecuta Rutina (OBTEN)
;               2.- Si caracter : = 'C'
;                   A.- Entonces:
;                       1.- Registro : = "PC"
;                       2.- Mensaje (10 : 15) : = 'PC  : ='
;                       3.- Mensaje (17 : 20) : = ASCII (Memo--
;                           ria (PC))
;                   B.- De lo contrario:
;                       1.- Ejecuta Rutina (NULO)
;                       2.- ERROR : = Verdadero
;           G.- 'D'; Entonces:
;               1.- Ejecuta Rutina (OBTEN)
;               2.- Clasifica caracter entre:
;                   A.- 'P'; Entonces:
;                       1.- Registro : = "DP"
;                       2.- Mensaje (10 : 15) : = 'DP  : ='
;                       3.- Mensaje (18 : 19) : = ASCII (Memo--
;                           ria (DP))
;                   B.- 'CR'; Entonces:
;                       1.- Mete caracter
;                       2.- Registro : = Acumulador "D"
;                       3.- Mensaje (9 : 15) : = 'ACCD  : ='
;                       4.- Mensaje (17 : 20) : = ASCII (Memo--
;                           ria (ACCD))
;                   C.- OTROS; Entonces:
;                       1.- Ejecuta Rutina (NULO)
```

```
;                       2.- ERROR : = Verdadero
;           H.- 'C'; Entonces:
;               1.- Ejecuta Rutina (OBTEN)
;               2.- Si caracter : = 'C'
;                   A.- Entonces:
;                       1.- Registro : = 'CC'
;                       2.- Mensaje (10 : 15) : = 'CC  : ='
;                       3.- Mensaje (18 : 19) : = ASCII (Memo--
;                           ria (CC))
;
;                   B.- De lo contrario:
;                       1.- Ejecuta Rutina (NULO)
;                       2.- ERROR : = Verdadero
;           I.- OTROS; Entonces:
;               1.- Ejecuta Rutina (NULO)
;               2.- ERROR : = Verdadero
;       5.- Si ERROR : = Falso
;           A.- Entonces:
;               1.- Mensaje (29 : 48) : = 'NUEVO VALOR (HEX) :
;                   = $'
;               2.- Imprime Mensaje (1 : 48)
;               3.- Ejecuta Rutina (BUFFER)
;               4.- Ejecuta Rutina (OBTEN)
;               5.- Clasifica caracter entre:
;                   A.- 'Ø' hasta 'F'; Entonces:
;                       1.- Valor : = Ordinal de caracter
;                       2.- Ejecuta Rutina (OBTEN)
;                       3.- Clasifica caracter entre:
;                           A.- 'Ø' hasta 'F'; Entonces:
;                               1.- Valor : = Valor * 1ØH + Ord.
;                                   de caracter
;                               2.- Memoria (REGISTRO) : = Va--
;                                   lor
;                           B.- 'CR'; Memoria (REGISTRO) : = Va
;                               lor
```

```
;                    C.- OTROS; Ejecuta Rutina (ERROR)
;            B.- 'CR'; Entonces no afectar el Registro
;            C.- OTROS; Ejecuta Rutina (ERROR)
; Fin
;.....................................................
;             CODIGO                        28/II/83
;                          (SUSREG)

REGIS :  DS 2
ERRO  :  DS 1
SUSRV :  DS 1
ACUMA :  DB 'ACCAb:='
ACUMB :  DB 'ACCBb:='
ACUMD :  DB 'ACCDb:='
INDX  :  DB 'IXbb:='
INDY  :  DB 'IYbb:='
STACKU:  DB 'USbb:='
STACKS:  DB 'SPbb:='
DIRPAG:  DB 'DPbb:='
CODCON:  DB 'CCbb:='
CONPRO:  DB 'PCbb:='
NUEVAL:  DB 'NUEVObVALORb(HEX)b:=$'
SUSREG:  LD      A,Ø2ØH          ; 1.-
         LD      HL,ØØ
         PUSH    HL
         LD      HL,BUFSIS
         LD      B,48
SUSR1 :  LD      (HL),A
         INC     HL
         DJNZ    SUSR1
         LD      A,Ø             ; 2.-
         LD      (ERRO),A
         CALL    OBTEN           ; 3.-
         CP      A,'A'           ; 4.-
```

```
        JP      NZ,SUSR4B       ; 4.A.-
        LD      BC,ACCA         ; 4.A.1.-
        LD      (REGIS),BC
        LD      A,Ø             ; Tamaño valor = 2
        LD      (SUSRV),A
        LD      DE,BUFSIS+8     ; 4.A.2.-
        LD      HL,ACUMA
        LD      BC,7
        LDIR
        ASC8    BUFSIS+17,ACCA  ; 4.A.3.-
        JP      SUSR5
SUSR4B: CP      A,'B'           ; 4.B.-
        JP      NZ,SUSR4C
        LD      BC,ACCB         ; 4.B.1.-
        LD      (REGIS),BC
        LD      DE,BUFSIS+8     ; 4.B.2.-
        LD      HL,ACUMB
        LD      BC,7
        LDIR
        ASC8    BUFSIS+17,ACCB  ; 4.B.3.-
        JP      SUSR5
SUSR4C: CP      A,'I'           ; 4.C.-
        JP      NZ,SUSR4D
        CALL    OBTEN           ; 4.C.1.-
        CP      A,'X'           ; 4.C.2.-
        JP      NZ,SUSC2B       ; 4.C.2.A.-
        LD      BC,RIY          ; 4.C.2.A.1.-
        LD      (REGIS),BC
        LD      A,ØFFH          ; identifica tamaño
        LD      (SUSRV),A       ; de Registro
        LD      DE,BUFSIS+9     ; 4.C.2.A.2.-
        LD      HL,INDX
        LD      BC,6
        LDIR
```

```
                ASC16   BUFSIS+16,RIY   ; 4.C.2.B.3.-
                JP      SUSR5
SUSR4D:         CP      A,'U'           ; 4.D.-
                JP      NZ,SUSR4E
                CALL    OBTEN           ; 4.D.1.-
                CP      A,'S'           ; 4.D.2.-
                JP      NZ,SUSNUL       ; 4.D.2.A.-
                LD      BC,US           ; 4.D.2.A.1.-
                LD      (REGIS),BC
                LD      A,ØFFH
                LD      (SUSRV),A
                LD      DE,BUFSIS+9     ; 4.D.2.A.2.-
                LD      HL,STACKU
                LD      BC,6
                LDIR
                ASC16   BUFSIS+16,US    ; 4.D.2.A.3.-
                JP      SUSR5
SUSR4E:         CP      A,'S'           ; 4.E.-
                JP      NZ,SUSR4F
                CALL    OBTEN           ; 4.E.1.-
                CP      A,'P'           ; 4.E.2.-
                JP      NZ,SUSNUL       ; 4.E.2.A.-
                LD      BC,RSP          ; 4.E.2.A.1.-
                LD      (REGIS),BC
                LD      A,ØFFH
                LD      (SUSRV),A
                LD      DE,BUFSIS+9     ; 4.E.2.A.2.-
                LD      HL,STACKS
                LD      BC,6
                LDIR
                ASC16   BUFSIS+16,RSP   ; 4.E.2.A.3.-
                JP      SUSR5
SUSR4F:         CP      A,'P'           ; 4.F.-
```

```
        JP      NZ,SUSR4G
        CALL    OBTEN           ; 4.F.1.-
        CP      A,'C'           ; 4.F.2.-
        JP      NZ,SUSNUL       ; 4.F.2.A.-
        LD      BC,RPC          ; 4.F.2.A.1.
        LD      (REGIS),BC
        LD      A,ØFFH
        LD      (SUSRV),A
        LD      DE,BUFSIS+9
        LD      HL,CONPRO
        LD      BC,6
        LDIR
        ASC16   BUFSIS+17,RPC   ; 4.F.2.A.3.-
        JP      SUSR5
SUSR4G: CP      A,'D'           ; 4.G.-
        JP      NZ,SUSR4H
        CALL    OBTEN           ; 4.G.1.-
        CP      A,'P'           ; 4.G.2.-
        JP      NZ,SUSG2B       ; 4.G.2.A.-
        LD      BC,DP
        LD      (REGIS),BC
        LD      DE,BUFSIS+9     ; 4.G.2.A.1.-
        LD      HL,DIRPAG
        LD      BC,6
        LDIR
        ASC8    BUFSIS+17,DP    ; 4.G.2.A.3.-
        JP      SUSR5
SUSG2B: CP      A,'CR'          ; 4.G.2.B.-
        JP      NZ,SUSNUL
        LD      BC,(APINIC)     ; 4.G.2.B.1.-
        DEC     BC
        LD      (APINIC),BC
        LD      A,CR
        LD      (DE),A
```

```
        LD      BC,ACCD         ; 4.G.2.B.2.-
        LD      (REGIS),BC
        LD      A,ØFFH
        LD      (SUSRV),A
        LD      DE,BUFSIS+8     ; 4.G.2.B.3.-
        LD      HL,ACUMD
        LD      BC,7
        LDIR
        ASC8    BUFSIS+16,ACCA  ; 4.G.2.B.4.-
        ASC8    BUFSIS+18,ACCB
        JP      SUSR5
SUSR4H: CP      A,'C'           ; 4.H.-
        JP      NZ,SUSNUL       ; 4.I.-
        CALL    OBTEN           ; 4.H.1.-
        CP      A,'C'           ; 4.H.2.-
        JP      NZ,SUSNUL       ; 4.H.2.A.-
        LD      BC,CC           ; 4.H.2.A.1.-
        LD      (REGIS),BC
        LD      DE,BUFSIS+9     ; 4.H.2.A.2.-
        LD      HL,CODCON
        LD      BC,6
        LDIR
        ASC8    BUFSIS+17,CC    ; 4.H.2.A.3.-
        JP      SUSR5
SUSNUL:                         ; 4.(C....H).2.B.-
        CALL    NULO            ; 4.(C....H).2.B.1.-
        LD      A,ØFFH          ; 4.(C....H).2.B.2.-
        LD      (ERRO),A
SUSR5 : LD      A,(ERRO)        ; 5.-
        LD      E,A
        LD      A,ØFFH
        CP      A,B
        JP      NZ,SUSR5A
        RET
```

```
SUSR5A:                                 ; 5.A.-
        LD      DE,BUFSIS+28            ; 5.A.1.-
        LD      HL,NUEVAL
        LD      BC,21
        LDIR
        LD      C,IMPRIM                ; 5.A.2.-
        LD      DE,BUFSIS·
        CALL    CDOS
        CALL    BUFFER                  ; 5.A.3.-
        CALL    OBTEN                   ; 5.A.4.-
SUS5A5: CP      A,3ØH                   ; 5.A.5.-
        JP      M,SUSR5B
        CP      A,3AH
        JP      P,SUS5BØ
        SUB     A,3ØH
        JP      SUS55A
SUS5BØ: CP      A,41H
        JP      M,ERROR
        CP      A,47H
        JP      P,ERROR
        SUB     A,37H
SUS55A: LD      E,A
        LD      D,Ø
        POP     HL
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL                   ; 5.A.5.A.1.-
        PUSH    HL
        CALL    OBTEN
        CP      A,'CR'
        JP      NZ,SUS5A5
        POP     HL
        LD      A,(SUSRV)
```

```
         CP      A,Ø
         JP      Z,SR2BCD
         LD      A,H
         LD      DE,(REGIS+1)
         LD      (DE),A
SR2BCD:  LD      A,L
         DEC     DE
         LD      (DE),A
         RET
SUSR5B:  CP      A,'CR'
         JP      NZ,ERROR
         RET

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

C O N C L U S I O N E S

Al resolver un problema es recomendable, primeramente, hacer un análisis y en seguida planear sus posibles soluciones. En nuestro caso de simular la programación de un microprocesador MC6809, la etapa de análisis consistió en conocer las características de funcionamiento del microprocesador, tales como:

a) Modos de direccionamiento.

b) Instrucciones.

c) Registros.

d) Capacidades.

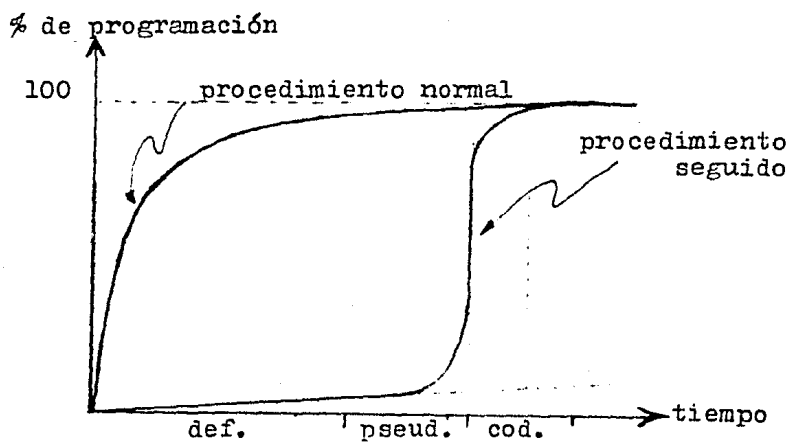
e) Facultades.

f) Banderas.

La etapa de planeación se realizó agrupando el pro-- grama simulador en rutinas y subrutinas. Todas las instruc-- ciones del microprocesador fueron agrupadas en seis rutinas. También se hicieron rutinas de utilería encargadas de ejecu- tar los requerimientos del usuario, tales como, mostrar con- tenido de registros o localidades de memoria, cambiar el con tenido de registros o localidades de memoria, etc..

La tercera y última etapa es en sí el programa con - código del microprocesador Z-80 que simula la programación - del microprocesador MC6809.

Este metodo hace más fácil pero más "largo" el cami- no para solucionar nuestro problema, ya que las primeras dos etapas (definición y pseudocódigo) son puramente comentarios, pero creemos que es necesario incluirlas para ir despejando poco a poco las dudas que pudiésen surgir en la etapa de có- digo. El paso de pseudocódigo a código es ya muy rápido tal como lo indica la siguiente gráfica:



Este proceso presenta una gran facilidad para su estudio a los lectores que esten poco familiarizados con la programación de estos dos procesadores, ya que irán adentrándose en el programa de una forma simple y sencilla, leyendo primeramente la definición, en seguida la etapa intermedia (pseudocódigo) y al llegar a la tercera y última etapa el lector tendrá ya un amplio conocimiento de lo que el código pretende hacer (note que tiene además comentarios a la derecha de las instrucciones, los cuales representan los incisos correspondientes en el pseudocódigo, proporcionándose así una valiosa ayuda para descifrar lo que una o varias instrucciones pretenden hacer.

La ventaja principal que se obtiene al operar correctamente el programa es obviamente el resultado de la simulación, "el poder trabajar con un microprocesador que físicamente no se tiene" sino que es simulado por una secuencia lógica de instrucciones de otro procesador, en nuestro caso el Z-80.

El haber realizado esta tesis incrementa grandemente nuestros conocimientos sobre:

a) Manejo de computadoras.

b) Lenguaje de programación del Z-80.

c) Lenguaje de programación del MC6809.

d) Análisis de problemas.

Un apéndice C fué incluido en este trabajo para mostrar al lector algunos ejemplos sobre programación del micro procesador que se simuló (MC6809).

Uno de los instrumentos más poderosos ideado por el hombre para la solución de problemas de cualquier índole es sin duda la computación electrónica. Cuando hay que contestar a la pregunta "¿ Para que puedo yo usar una computadora ?", resulta muy dificil hacerlo, pues si se responde "para lo que usted quiera", el que recibe la respuesta no quedará satisfecho, pues piensa que se le está contestando para salir del paso. Sin embargo, la respuesta es correcta, ya que la limitación de sus aplicaciones reside sólamente en el ingenio del usuario.

La simulación encuentra aplicación en todas las ciencias y particularmente en las ramas de la Ingeniería como son: Computación, Construcción, Aviación, Diseño de automoviles, etc..

B I B L I O G R A F I A .

1.- Motorola: "MC6809-MC6809E 8-BIT MICROPROCESSOR PROGRAMMING", Original issue: March 1, 1981.

2.- Cromemco: "DAZZLER Z-80 MICROCOMPUTER DEVICES TECHNICAL MANUAL", Cromemco, Inc., October 1978.

3.- Cromemco: "Z-80 MACRO ASSEMBLER INSTRUCTION MANUAL", - Cromemco, Inc., October 1978.

4.- Revista: "INFORMACION CIENTIFICA Y TECNOLOGICA", Conacy t, Octubre de 1983, vol. 5, Núm. 65.

5.- Serie, Mundo electrónico: "MICROPROCESADORES Y MICROCOMPUTADORES", Barcombo, Boixareu Editores, 2a. - Edición, 1978.

6.- A. Clare, Cristopher : "DESIGNING LOGIC SYSTEMS USING STATE MACHINES", McGraw Hills Book Company, 1973.

7.- E. Pectman, John: "MICROCOMPUTER BASED DESIGN".

APENDICE A

TABLAS DE INSTRUCCIONES DEL MC6809

Las Tablas A.1, A.2 y A.3 contienen una compilación de datos que ayudarán en la programación del M6809.

La Tabla A.4 es el mapa de código de operación para el M6809. El número(s) para cada instrucción indica el número de ciclos de máquina requeridos para ejecutar dicha instrucción. Cuando el número contiene un "1" (ejemplo 4+1), este indica que el modo de direccionamiento de indexado esta siendo usado y que un número adicional de ciclos de máquina pueden ser requeridos. La Tabla A.5 determina los ciclos de máquina adicionales a ser sumados.

Algunas instrucciones en el mapa de código de operación tienen dos números, el segundo dentro de un parentesis. Indica que la instrucción involucra un salto (branch). El número entre parentesis se aplica si el salto fué hacho.

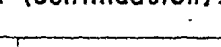
La notación de "página 2, página3" en la columna uno indican que todas las instrucciones de página 2 estan precedidas por un código de operación 10 hexadecimal y todas las instrucciones de página 3 estan precedidas por un código de operación 11 hexadecimal.

Instruction	Forms	Addressing Mode Relative OP	~	#	Description	5 H	3 N	2 Z	1 V	0 C
BLS	BLS	23	3	2	Branch Lower or Same	•	•	•	•	•
	LBLS	10 23	5(6)	4	Long Branch Lower or Same	•	•	•	•	•
BLT	BLT	2D	3	2	Branch < Zero	•	•	•	•	•
	LBLT	10 2D	5(6)	4	Long Branch < Zero	•	•	•	•	•
BMI	BMI	2B	3	2	Branch Minus	•	•	•	•	•
	LBMI	10 2B	5(6)	4	Long Branch Minus	•	•	•	•	•
BNE	BNE	26	3	2	Branch Z ≠ 0	•	•	•	•	•
	LBNE	10 26	5(6)	4	Long Branch Z ≠ 0	•	•	•	•	•
BPL	BPL	2A	3	2	Branch Plus	•	•	•	•	•
	LBPL	10 2A	5(6)	4	Long Branch Plus	•	•	•	•	•
BRA	BRA	20	3	2	Branch Always	•	•	•	•	•
	LBRA	16	5	3	Long Branch Always	•	•	•	•	•
BRN	BRN	21	3	2	Branch Never	•	•	•	•	•
	LBRN	10 21	5	4	Long Branch Never	•	•	•	•	•
BSR	BSR	8D	7	2	Branch to Subroutine	•	•	•	•	•
	LBSR	17	9	3	Long Branch to Subroutine	•	•	•	•	•
BVC	BVC	28	3	2	Branch V = 0	•	•	•	•	•
	LBVC	10 28	5(6)	4	Long Branch V = 0	•	•	•	•	•
BVS	BVS	29	3	2	Branch V = 1	•	•	•	•	•
	LBVS	10 29	5(6)	4	Long Branch V = 1	•	•	•	•	•

Instruction	Forms	Addressing Mode Relative OP	~	#	Description	5 H	3 N	2 Z	1 V	0 C
BCC	BCC	24	3	2	Branch C = 0	•	•	•	•	•
	LBCC	10 24	5(6)	4	Long Branch C = 0	•	•	•	•	•
BCS	BCS	25	3	2	Branch C = 1	•	•	•	•	•
	LBCS	10 25	5(6)	4	Long Branch C = 1	•	•	•	•	•
BEQ	BEQ	27	3	2	Branch Z = 0	•	•	•	•	•
	LBEQ	10 27	5(6)	4	Long Branch Z = 0	•	•	•	•	•
BGE	BGE	2C	3	2	Branch ≥ Zero	•	•	•	•	•
	LBGE	10 2C	5(6)	4	Long Branch ≥ Zero	•	•	•	•	•
BGT	BGT	2E	3	2	Branch > Zero	•	•	•	•	•
	LBGT	10 2E	5(6)	4	Long Branch > Zero	•	•	•	•	•
BHI	BHI	22	3	2	Branch Higher	•	•	•	•	•
	LBHI	10 22	5(6)	4	Long Branch Higher	•	•	•	•	•
BHS	BHS	24	3	2	Branch Higher or Same	•	•	•	•	•
	LBHS	10 24	5(6)	4	Long Branch Higher or Same	•	•	•	•	•
BLE	BLE	2F	3	2	Branch ≤ Zero	•	•	•	•	•
	LBLE	10 2F	5(6)	4	Long Branch ≤ Zero	•	•	•	•	•
BLO	BLO	25	3	2	Branch lower	•	•	•	•	•
	LBLO	10 25	5(6)	4	Long Branch Lower	•	•	•	•	•

Tabla A.1 Ayuda de programación (saltos).

Tabla A.2 Ayuda de programación (continuación).

Instruction	Forms	Addressing Modes: Immediate Op	~	#	Direct Op	~	#	Indexed Op	~	#	Extended Op	~	#	Inherent Op	~	#	Description	5 H	3 N	2 Z	1 V	0 C
ABX														3A	3	1	B + X → X (Unsigned)	•	•	•	•	•
ADC	ADCA	99	2	2	99	4	2	A9	4+	2+	B9	5	3				A + M + C → A	↕	↕	↕	↕	↕
	ADCB	C9	2	2	D9	4	2	E9	4+	2+	F9	5	3				B + M + C → B	↕	↕	↕	↕	↕
ADD	ADDA	8B	2	2	9B	4	2	AB	4+	2+	BB	5	3				A + M → A	↕	↕	↕	↕	↕
	ADDB	CB	2	2	DB	4	2	EB	4+	2+	FB	5	3				B + M → B	↕	↕	↕	↕	↕
	ADDD	C3	4	3	D3	6	2	E3	6+	2+	F3	7	3				D + M:M+1 → D	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	4	2	A4	4+	2+	B4	5	3				A ∧ M → A	•	↕	↕	0	•
	ANDB	C4	2	2	D4	4	2	E4	4+	2+	F4	5	3				B ∧ M → B	•	↕	↕	0	•
	ANDCC	1C	3	2													CC ∧ IMM → CC					?
ASL	ASLA													48	2	1		↕	↕	↕	↕	↕
	ASLB													58	2	1		↕	↕	↕	↕	↕
	ASL				08	6	2	68	6+	2+	78	7	3					↕	↕	↕	↕	↕
ASR	ASRB													47	2	1		↕	↕	↕	•	↕
	ASR													57	2	1		↕	↕	↕	•	↕
	ASR				07	6	2	67	6+	2+	77	7	3					↕	↕	↕	•	↕
BIT	BITA	85	2	2	95	4	2	A5	4+	2+	B5	5	3				Bit Test A (M ∧ A)	•	↕	↕	0	•
	BITB	C5	2	2	D5	4	2	E5	4+	2+	F5	5	3				Bit Test B (M ∧ B)	•	↕	↕	0	•
CLR	CLRA													4F	2	1	0 → A	•	0	1	0	0
	CLRB													5F	2	1	0 → B	•	0	1	0	0
	CLR				0F	6	2	6F	6+	2+	7F	7	3				0 → M	•	0	1	0	0
CMP	CMPA	81	2	2	91	4	2	A1	4+	2+	B1	5	3				Compare M from A	↕	↕	↕	↕	↕
	CMPB	C1	2	2	D1	4	2	E1	4+	2+	F1	5	3				Compare M from B	↕	↕	↕	↕	↕
	CMPD	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M:M+1 from D	•	↕	↕	↕	↕
		93			93			A3			B3											
	CMPS	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M:M+1 from S	•	↕	↕	↕	↕
		8C			9C			AC			BC											
	CMPU	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M:M+1 from U	•	↕	↕	↕	↕
		83			93			A3			B3											
	CMPX	8C	4	3	9C	6	2	AC	6+	2+	BC	7	3				Compare M:M+1 from X	•	↕	↕	↕	↕
	CMPY	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M:M+1 from Y	•	↕	↕	↕	↕
		8C			9C			AC			BC											
COM	COMA													43	2	1	$\overline{A}$ → A	•	↕	↕	0	1
	COMB													53	2	1	$\overline{B}$ → B	•	↕	↕	0	1
	COM				03	6	2	63	6+	2+	73	7	3				$\overline{M}$ → M	•	↕	↕	0	1
CWAI		3C	≥20	2													CC ∧ IMM → CC Wait for Interrupt					?
DAA														19	2	1	Decimal Adjust A	•	↕	↕	0	↕
DEC	DECA													4A	2	1	A − 1 → A	•	↕	↕	↕	•
	DECB													5A	2	1	B − 1 → B	•	↕	↕	↕	•
	DEC				0A	6	2	6A	6+	2+	7A	7	3				M − 1 → M	•	↕	↕	↕	•
EOR	EORA	88	2	2	98	4	2	A8	4+	2+	B8	5	3				A ⊻ M → A	•	↕	↕	0	•
	EORB	C8	2	2	D8	4	2	E8	4+	2+	F8	5	3				B ⊻ M → B	•	↕	↕	0	•
EXG	R1, R2	1E	8	2													R1 ↔ R2[2]	•	•	•	•	•
INC	INCA													4C	2	1	A + 1 → A	•	↕	↕	↕	•
	INCB													5C	2	1	B + 1 → B	•	↕	↕	↕	•
	INC				0C	6	2	6C	6+	2+	7C	7	3				M + 1 → M	•	↕	↕	↕	•
JMP					0E	3	2	6E	3+	2+	7E	4	3				EA[3] → PC	•	•	•	•	•
JSR					9D	7	2	AD	7+	2+	BD	8	3				Jump to Subroutine	•	•	•	•	•
LD	LDA	86	2	2	96	4	2	A6	4+	2+	B6	5	3				M → A	•	↕	↕	0	•
	LDB	C6	2	2	D6	4	2	E6	4+	2+	F6	5	3				M → B	•	↕	↕	0	•
	LDD	CC	3	3	DC	5	2	EC	5+	2+	FC	6	3				M:M+1 → D	•	↕	↕	0	•
	LDS	10	4	4	10	6	3	10	6+	3+	10	7	4				M:M+1 → S	•	↕	↕	0	•
		CE			DE			EE			FE											
	LDU	CE	3	3	DE	5	2	EE	5+	2+	FE	6	3				M:M+1 → U	•	↕	↕	0	•
	LDX	8E	3	3	9E	5	2	AE	5+	2+	BE	6	3				M:M+1 → X	•	↕	↕	0	•
	LDY	10	4	4	10	6	3	10	6+	3+	10	7	4				M:M+1 → Y	•	↕	↕	0	•
		8E			9E			AE			BE											
LEA	LEAS							32	4+	2+							EA[3] → S	•	•	•	•	•
	LEAU							33	4+	2+							EA[3] → U	•	•	•	•	•
	LEAX							30	4+	2+							EA[3] → X	•	•	↕	•	•
	LEAY							31	4+	2+							EA[3] → Y	•	•	↕	•	•

Legend

OP	Operation Code (Hexadecimal)	$\overline{M}$	Complement of M
~	Number of MPU Cycles	→	Transfer Into
#	Number of Program Bytes	H	Half-carry (from bit 3)
+	Arithmetic Plus	N	Negative (sign bit)
−	Arithmetic Minus	Z	Zero (Reset)
*	Multiply	V	Overflow, 2's complement
		C	Carry from ALU

↕	Test and set if true, cleared otherwise
•	Not Affected
CC	Condition Code Register
:	Concatenation
V	Logical or
∧	Logical and
⊻	Logical Exclusive or

Tabla A.3 Ayuda de programación (continuación).

Instruction	Forms	Addressing Modes: Immediate Op	~	#	Direct Op	~	#	Indexed[1] Op	~	#	Extended Op	~	#	Inherent Op	~	#	Description	5 H	3 N	2 Z	1 V	0 C
LSL	LSLA													48	2	1	A, B, M — C b7 b0	•	↕	↕	↕	↕
	LSLB													58	2	1		•	↕	↕	↕	↕
	LSL				08	6	2	68	6+	2+	78	7	3					•	↕	↕	↕	↕
LSR	LSRA													44	2	1	A, B, M — 0 b7 b0 C	•	0	↕	•	↕
	LSRB													54	2	1		•	0	↕	•	↕
	LSR				04	6	2	64	6+	2+	74		3					•	0	↕	•	↕
MUL														3D	11	1	A × B — D (unsigned)	•	•	↕	•	↕
NEG	NEGA													40	2	1	A + 1 — A	↕	↕	↕	↕	↕
	NEGB													50	2	1	B + 1 — B	↕	↕	↕	↕	↕
	NEG				00	6	2	60	6+	2+	70	7	3				M + 1 — M	↕	↕	↕	↕	↕
NOP														12	2	1	No Operation	•	•	•	•	•
OR	ORA	8A	2	2	9A	4	2	AA	4+	2+	BA	5	3				A v M — A	•	↕	↕	0	•
	ORB	CA	2	2	DA	4	2	EA	4+	2+	FA	5	3				B v M — B	•	↕	↕	0	•
	ORCC	1A	3	2													CC v IMM — CC	↕	↕	↕	↕	↕
PSH	PSHS	34	5+[4]	2													Push Registers on S Stack	•	•	•	•	•
	PSHU	36	5+[4]	2													Push Registers on U Stack	•	•	•	•	•
PUL	PULS	35	5+[4]	2													Pull Registers from S Stack	•	•	•	•	•
	PULU	37	5+[4]	2													Pull Registers from U Stack	•	•	•	•	•
ROL	ROLA													49	2	1	A, B, M — C b7 b0 C	•	↕	↕	↕	↕
	ROLB													59	2	1		•	↕	↕	↕	↕
	ROL				09	6	2	69	6+	2+	79	7	3					•	↕	↕	↕	↕
ROR	RORA													46	2	1	A, B, M — C b7 b0 C	•	↕	↕	•	↕
	RORB													56	2	1		•	↕	↕	•	↕
	ROR				06	6	2	66	6+	2+	76	7	3					•	↕	↕	•	↕
RTI														3B	6, 15	1	Return From Interrupt	↕	↕	↕	↕	↕
RTS														39	5	1	Return from Subroutine	•	•	•	•	•
SBC	SBCA	82	2	2	92	4	2	A2	4+	2+	B2	5	3				A — M — C — A	↕	↕	↕	↕	↕
	SBCB	C2	2	2	D2	4	2	E2	4+	2+	F2	5	3				B — M — C — B	↕	↕	↕	↕	↕
SEX														1D	2	1	Sign Extend B into A	•	↕	↕		•
ST	STA				97	4	2	A7	4+	2+	B7	5	3				A — M	•	↕	↕	0	•
	STB				D7	4	2	E7	4+	2+	F7	5	3				B — M	•	↕	↕	0	•
	STD				DD	5	2	ED	5+	2+	FD	6	3				D — M M + 1	•	↕	↕	0	•
	STS				10 DF	6	3	10 EF	6+	3+	10 FF	7	4				S — M M + 1	•	↕	↕	0	•
	STU				DF	5	2	EF	5+	2+	FF	6	3				U — M M + 1	•	↕	↕	0	•
	STX				9F	5	2	AF	5+	2+	BF	6	3				X — M M + 1	•	↕	↕	0	•
	STY				10 9F	6	3	10 AF	6+	3+	10 BF	7	4				Y — M M + 1	•	↕	↕	0	•
SUB	SUBA	80	2	2	90	4	2	A0	4+	2+	B0	5	3				A — M — A	↕	↕	↕	↕	↕
	SUBB	C0	2	2	D0	4	2	E0	4+	2+	F0	5	3				B — M — B	↕	↕	↕	↕	↕
	SUBD	83	4	3	93	6	2	A3	6+	2+	B3	7	3				D — M M + 1 — D	•	↕	↕	↕	↕
SWI	SWI[6]													3F	19	1	Software Interrupt 1	•	•	•	•	•
	SWI2[6]													10 3F	20	2	Software Interrupt 2	•	•	•	•	•
	SWI3[6]													11 3F	20	1	Software Interrupt 3	•	•	•	•	•
SYNC														13	≥ 4	1	Synchronize to Interrupt	•	•	•	•	•
TFR	R1, R2	1F	6	2													R1 — R2[7]	•	•	•	•	•
TST	TSTA													4D	2	1	Test A	•	↕	↕	0	•
	TSTB													5D	2	1	Test B	•	↕	↕	0	•
	TST				0D	6	2	6D	6+	2+	7D	7	3				Test M	•	↕	↕	0	•

Notes:

1. This column gives a base cycle and byte count. To obtain total count, add the values obtained from the INDEXED ADDRESSING MODE table in Appendix F.
2. R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
   The 8 bit registers are: A, B, CC, DP
   The 16 bit registers are: X, Y, U, S, D, PC
3. EA is the effective address
4. The PSH and PUL instructions require 5 cycles plus 1 cycle for each byte pushed or pulled.
5. 5(6) means: 5 cycles if branch not taken, 6 cycles if taken (Branch instructions)
6. SWI sets I and F bits. SWI2 and SWI3 do not affect I and F
7. Conditions Codes set as a direct result of the instruction.
8. Value of half-carry flag is undefined.
9. Special Case — Carry set if b7 is SET

Least Significant Four Bits	Most-Significant Four Bits: DIR 0000	0001	REL 0010	0011	ACCA 0100	ACCB 0101	IND 0110	EXT 0111	IMM 1000	DIR 1001	IND 1010	EXT 1011	IMM 1100	DIR 1101	IND 1110	EXT 1111	Least Significant Four Bits
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000 0	6 NEG	PAGE2	3 BRA	4+1 LEAX	2	2	6+1 NEG	7	2	4	4+1 SUBA	5	2	4	4+1 SUBB	5	0
0001 1	——	PAGE3	3 BRN/ 5 LBRN	4+1 LEAY	——				2	4	4+1 CMPA	5	2	4	4+1 CMPB	5	1
0010 2	——	2 NOP	3 BHI/ 5(6) LBHI	4+1 LEAS	——				2	4	4+1 SBCA	5	2	4	4+1 SBCB	5	2
0011 3	6 COM	2 SYNC	3 BLS/ 5(6) LBLS	4+1 LEAU	2	2	6+1 COM	7	4,6,6+1,7 SUBD	5,7,7+1,8 CMPD	5,7,7+1,8 CMPU		4	6	6+1 ADDD	7	3
0100 4	6 LSR	——	3 BHS 5(6) (BCC)	5+1/by PSHS	2	2	6+1 LSR	7	2	4	4+1 ANDA	5	2	4	4+1 ANDB	5	4
0101 5	——	——	3 BLO 5(6) (BCS)	5+1/by PULS	——				2	4	4+1 BITA	5	2	4	4+1 BITB	5	5
0110 6	6 ROR	5 LBRA	3 BNE/ 5(6) LBNE	5+1/by PSHU	2	2	6+1 ROR	7	2	4	4+1 LDA	5	2	4	4+1 LDB	5	6
0111 7	6 ASR	9 LBSR	3 BEQ/ 5(6) LBEQ	5+1/by PULU	2	2	6+1 ASR	7	——	4	4+1 STA	5	——	4	4+1 STB	5	7
1000 8	6 ASL (LSL)	——	3 BVC/ 5(6) LBVC	——	2	2	6+1 ASL (LSL)	7	2	4	4+1 EORA	5	2	4	4+1 EORB	5	8
1001 9	6 ROL	2 DAA	3 BVS/ 5(6) LBVS	5 RTS	2	2	6+1 ROL	7	2	4	4+1 ADCA	5	2	4	4+1 ADCB	5	9
1010 A	6 DEC	3 ORCC	3 BPL/ 5(6) LBPL	3 ABX	2	2	6+1 DEC	7	2	4	4+1 ORA	5	2	4	4+1 ORB	5	A
1011 B	——	——	3 BMI/ 5(6) LBMI	6/15 RTI	——				2	4	4+1 ADDA	5	2	4	4+1 ADDB	5	B
1100 C	6 INC	3 ANDCC	3 BGE/ 5(6) LBGE	20 CWAI	2	2	6+1 INC	7	4,6,6+1,7 CMPX	5,7,7+1,8 CMPY	5,7,7+1,8 CMPS		3	5	5+1 LDD	6	C
1101 D	6 TST	2 SEX	3 BLT/ 5(6) LBLT	11 MUL	2	2	6+1 TST	7	7 BSR	7	7+1 JSR	8	——	5	5+1 STD	6	D
1110 E	3 JMP	8 EXG	3 BGT/ 5(6) LBGT	——	——		3+1 JMP	4	3,5,5+1,6 LDX		4,6,6+1,7 LDY		3,5,5+1,6 LDU		4,6,6+1,7 LDS		E
1111 F	6 CLR	7 TFR	3 BLE/ 5(6) LBLE	19/20/20 SWI/2/3	2	2	6+1 CLR	7	——	5,5+1,6 STX	6,6+1,7 STY		——	5,5+1,6 STU	6,6+1,7 STS		F

Tabla A.4 Mapa de códigos de operación.

Type	Forms	Non Indirect				Indirect			
		Assembler Form	Postbyte OP Code	~	#	Assembler Form	Postbyte OP Code	~	#
Constant Offset From R (twos complement offset)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (twos complement offset)	A — Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B — Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D — Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R+	1RR00000	2	0	not allowed			
	Increment By 2	,R++	1RR00001	3	0	[,R++]	1RR10001	6	0
	Decrement By 1	,-R	1RR00010	2	0	not allowed			
	Decrement By 2	,--R	1RR00011	3	0	[,--R]	1RR10011	6	0
Constant Offset From PC (twos complement offset)	8 Bit Offset	n, PCR	1XX01100	1	1	[n, PCR]	1XX11100	4	1
	16 Bit Offset	n, PCR	1XX01101	5	2	[n, PCR]	1XX11101	8	2
Extended Indirect	16 Bit Address	—	—	—	—	[n]	10011111	5	2

R = X, Y, U or S X = 00 Y = 01
X = Don't Care U = 10 S = 11

~ and # Indicate the number of additional cycles and bytes for the particular variation

Tabla A.5 Datos para el modo de direccionamiento de indexado

APENDICE B

EL MICROPROCESADOR Z-80

La familia de componentes MOSTEK Z-80 es un resultado del avance de las microcomputadoras. Estos componentes -- pueden ser configurados con cualquier tipo de memorias semiconductoras estandard para generar sistemas computacionales con un rango de capacidades extremadamente largo.

El Z-80 es totalmente compatible en software con el Mp 8080A.

El juego de componentes del Z-80 es superior tanto - en hardware como en software comparado con cualquier otro -- sistema microcomputarizado de 8 bits existente en el mercado.

La CPU es el corazón del sistema. Su función es obtener instrucciones de la memoria y realizar las operaciones - deseadas.

La CPU Z-80 cuenta con 18 registros de 8 bits y 4 registros de 16 bits. Todos los registros del Z-80 son llevados a cabo usando RAM'S estáticas.

Los registros incluyen dos juegos de seis registros de propósito general que pueden ser utilizados individualmente como registros de 8 bits o en pares como registros de 16. También cuenta con dos juegos de acumuladores y registros de banderas. Todo lo anteriormente dicho se ilustra en la Figu-

ra B.1.

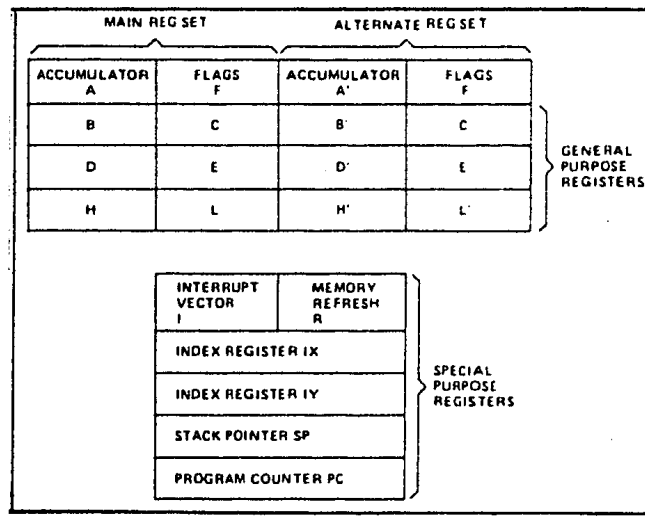


Fig. B.1 Configuración de registros en la CPU Z-80.

La CPU Z-80 puede ejecutar 158 tipos diferentes de - instrucciones incluyendo las 78 de la CPU 8080A.

Las instrucciones pueden ser separadas en 11 grandes grupos, primeramente se ilustran estos grupos de una forma - breve y concisa en 15 tablas con las cuales el lector podrá determinar todas las combinaciones disponibles de una determinada instrucción, y en seguida se muestran estos mismos 11 grupos pero con una información más detallada sobre las consecuencias que suceden al ejecutar una determinada instrucción.

Tabla B.1 Grupo de cargado de 8 bits .

DESTINATION			SOURCE																	
			IMPLIED		REGISTER							REG INDIRECT			INDEXED		EXT ADDR	IMME
			I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX + d)	(IY + d)	(nn)	n
	REGISTER	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
		B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
		C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
		D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
		E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
		H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
		L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
	REG INDIRECT	(HL)			77	70	71	72	73	74	75							36 n
		(BC)			02													
		(DE)			12													
	INDEXED	(IX+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
		(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
	EXT ADDR	(nn)			32 n n													
	IMPLIED	I			ED 47													
		R			ED 4F													

Tabla B.2 Grupo de 16 bits "LD" "PUSH" y "POP".

DESTINATION REGISTER	SOURCE REGISTER AF	BC	DE	HL	SP	IX	IY	IMM. EXT. nn	EXT. ADDR. (nn)	REG INDIR (SP)
AF										F1
BC								01 n n	ED 4B n n	C1
DE								11 n n	ED 5B n n	D1
HL								21 n n	2A n n	E1
SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
IX								DD 21 n n	DD 2A n n	DD E1
IY								FD 21 n n	FD 2A n n	FD E1
EXT. ADDR. (nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH INSTRUCTIONS → REG. IND. (SP)	F5	C5	D5	E5		DD E5	FD E5			

NOTE: The Push & Pop instructions adjust the SP after every execution

POP INSTRUCTIONS ↑

Tabla B.3 Grupo de intercambios "EX" y "EXX".

		IMPLIED ADDRESSING AF	BC, DE & HL	HL	IX	IY
IMPLIED	AF	08				
	BC, DE & HL		D9			
	DE			EB		
REG INDIR.	(SP)			E3	DD E3	FD E3

Tabla B.4 Grupo de transferencia de bloques.

DESTINATION			SOURCE: REG INDIR	SOURCE: (HL)	
REG INDIR	(DE)		ED A0	'LDI' – Load (DE)←(HL) Inc HL & DE, Dec BC	
			ED B0	'LDIR' – Load (DE)←(HL) Inc HL & DE, Dec BC Repeat until BC = 0	
			ED A8	'LDD' – Load (DE)←(HL) Dec HL & DE Dec BC	
			ED B8	'LDDR' – Load (DE)←(HL) Dec HL & DE Dec BC Repeat until BC = 0	

Reg HL points to source
Reg DE points to destination
Reg BC is byte counter

Tabla B.5 Grupo de búsqueda de bloques.

SEARCH LOCATION: REG. INDIR	SEARCH LOCATION: (HL)	
ED A1	'CPI' Inc HL, Dec BC	
ED B1	'CPIR', Inc HL, Dec BC repeat until BC = 0 or find match	
ED A9	'CPD' Dec HL & BC	
ED B9	'CPDR' Dec HL & BC Repeat until BC = 0 or find match	

HL points to location in memory to be compared with accumulator contents
BC is byte counter

Tabla B.6 Grupo de aritmética y lógica de 8 bits.

	SOURCE										
	REGISTER ADDRESSING							REG INDIR	INDEXED		IMMED
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

Tabla B.7 Grupo de operaciones de propósito general.

Decimal Adjust Acc. 'DAA'	27
Complement Acc. 'CPL'	2F
Negate Acc. 'NEG' (2's complement)	ED 44
Complement Carry Flag. 'CCF'	3F
Set Carry Flag. 'SCF'	37

Tabla B.8 Grupo de aritmética de 16 bits.

			SOURCE					
			BC	DE	HL	SP	IX	IY
DESTINATION	'ADD'	HL	09	19	29	39		
		IX	DD 09	DD 19		DD 39	DD 29	
		IY	FD 09	FD 19		FD 39		FD 29
	ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
	SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
	INCREMENT 'INC'		03	13	23	33	DD 23	FD 23
	DECREMENT 'DEC'		0B	1B	2B	3B	DD 2B	FD 2B

Tabla B.9 Grupo de rotaciones y corrimientos.

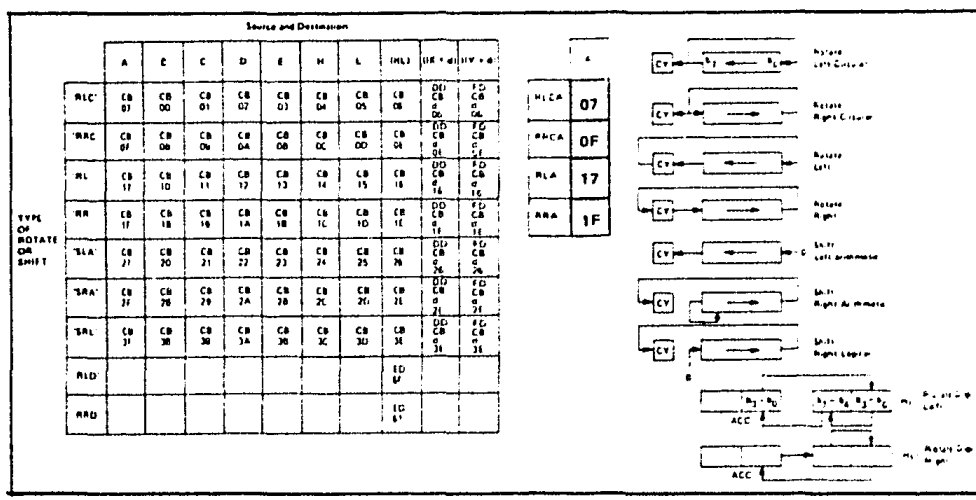


Tabla B.10 Grupo de manipulación de bits.

		REGISTER ADDRESSING							REG. INDIR.	INDEXED	
	BIT	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
TEST BIT	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RESET BIT RES	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET BIT SET	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB F4	CB E5	CB Eh	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d F4	FD CB d FE

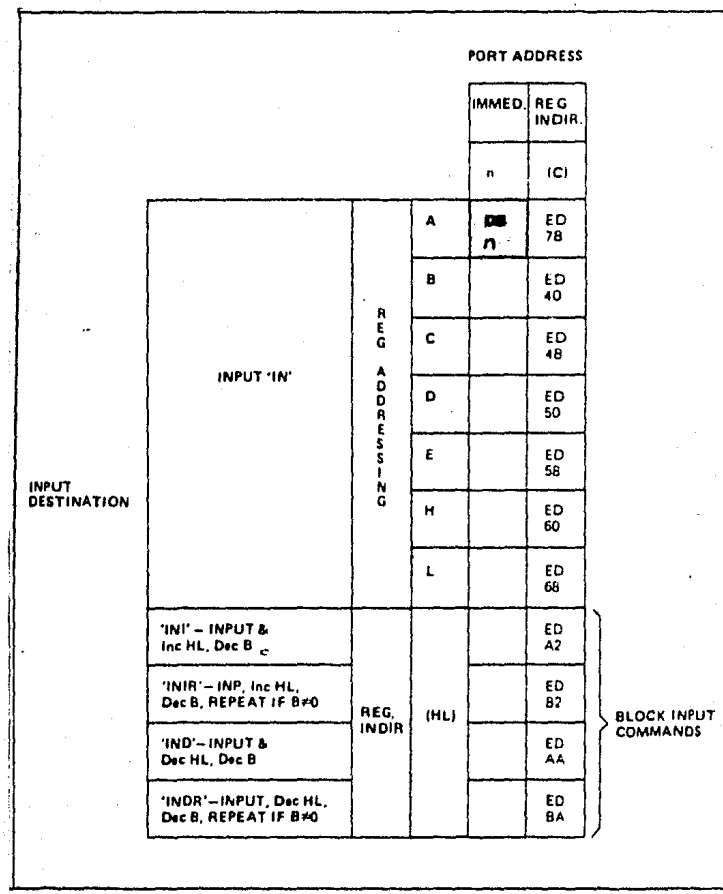
Tabla B.11 Grupo de saltos, llamadas y regresos.

			CONDITION									
			UN COND	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG	SIGN POS	B≠0
JUMP 'JP'	IMMED. EXT.	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JR'	RELATIVE	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP 'JP'	REG. INDIR.	(HL)	E9									
JUMP 'JP'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
'CALL'	IMMED. EXT.	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DECREMENT B, JUMP IF NON ZERO 'DJNZ'	RELATIVE	PC+e										10 e-2
RETURN 'RET'	REGISTER INDIR.	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETURN FROM INT 'RETI'	REG. INDIR.	(SP) (SP+1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	REG. INDIR.	(SP) (SP+1)	ED 45									

Tabla B.12 Grupo de restablecido.

		OP CODE	
CALL ADDRESS	0000_H	C7	'RST 0'
	0008_H	CF	'RST 8'
	0010_H	D7	'RST 16'
	0018_H	DF	'RST 24'
	0020_H	E7	'RST 32'
	0028_H	EF	'RST 40'
	0030_H	F7	'RST 48'
	0038_H	FF	'RST 56'

Tabla B.13 Grupo de entradas.



INPUT DESTINATION				PORT ADDRESS: IMMED. n	PORT ADDRESS: REG INDIR. (C)
	INPUT 'IN'	REG ADDRESSING	A	n	ED 78
			B		ED 40
			C		ED 48
			D		ED 50
			E		ED 58
			H		ED 60
			L		ED 68
	'INI' – INPUT & Inc HL, Dec B	REG. INDIR	(HL)		ED A2
	'INIR' – INP, Inc HL, Dec B, REPEAT IF B≠0				ED B2
	'IND' – INPUT & Dec HL, Dec B				ED AA
	'INDR' – INPUT, Dec HL, Dec B, REPEAT IF B≠0				ED BA

BLOCK INPUT COMMANDS

Tabla B.14 Grupo de salidas.

			SOURCE							
			REGISTER							REG. IND.
			A	B	C	D	E	H	L	(HL)
'OUT'	IMMED	n	D3 n							
	REG IND	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69	
'OUTI' – OUTPUT Inc HL, Dec B	REG IND	(C)								ED A3
'OTIR' – OUTPUT, Inc HL, Dec B, REPEAT IF B≠0	REG IND	(C)								ED B3
'OUTD' – OUTPUT Dec HL & B	REG IND	(C)								ED AB
'OTDR' – OUTPUT, Dec HL & B, REPEAT IF B≠0	REG. IND.	(C)								ED BB

BLOCK OUTPUT COMMANDS

PORT DESTINATION ADDRESS

Tabla B.15 Grupo de control de la CPU e instrucciones de miscelanea.

'NOP'	00	
'HALT'	76	
DISABLE INT 'DI'	F3	
ENABLE INT 'EI'	FB	
SET INT MODE 0 'IM0'	ED 46	8080A MODE
SET INT MODE 1 'IM1'	ED 56	CALL TO LOCATION 0038_H
SET INT MODE 2 'IM2'	ED 5E	INDIRECT CALL USING REGISTER I AND 8 BITS FROM INTERRUPTING DEVICE AS A POINTER

La siguiente sección muestra un sumario del juego de instrucciones del Z-80. Las instrucciones tienen un arreglo lógico dentro de los grupos, tal como se muestra en las tablas.

Cada tabla indica el código de operación del mnemónico en lenguaje ensamblador, el código de operación efectivo, la operación simbólica, el contenido del registro de banderas después de la ejecución de cada instrucción, el número de bytes requeridos para cada instrucción, el número de ciclos de memoria y el número total de estados T (períodos de reloj externo) requeridos para la búsqueda y ejecución de cada instrucción. Se ha tenido la precaución de que cada tabla de una explicación total de las instrucciones que contiene, no siendo necesario consultar otra tabla o texto.

Tabla B.16 Sumario de operaciones de banderas.

Instruction	D7 S	Z		H		P/V	N	D0 C	Comments
ADD A,s; ADC A,s	↕	↕	X	↕	X	V	0	↕	8 bit add or add with carry
SUB s; SBC A,s; CP s; NEG	↕	↕	X	↕	X	V	1	↕	8 bit subtract, subtract with carry, compare and negate accumulator
AND s	↕	↕	X	1	X	P	0	0	Logical operations
OR s; XOR s	↕	↕	X	0	X	P	0	0	
INC s	↕	↕	X	↕	X	V	0	•	8 bit increment
DEC s	↕	↕	X	↕	X	V	1	•	8 bit decrement
ADD DD, SS	•	•	X	X	X	•	0	↕	16 bit add
ADC HL, SS	↕	↕	X	X	X	V	0	↕	16-bit add with carry
SBC HL, SS	↕	↕	X	X	X	V	1	↕	16-bit subtract with carry
RLA, RLCA, RRA, RRCA	•	•	X	0	X	•	0	↕	Rotate accumulator
RL s, RLC s, RR s, RRC s, SLA s, SRA s, SRL s	↕	↕	X	0	X	P	0	↕	Rotate and shift locations
RLD, RRD	↕	↕	X	0	X	P	0	•	Rotate digit left and right
DAA	↕	↕	X	↕	X	P	•	↕	Decimal adjust accumulator
CPL	•	•	X	1	X	•	1	•	Complement accumulator
SCF	•	•	X	0	X	•	0	1	Set carry
CCF	•	•	X	X	X	•	0	↕	Complement carry
IN r,(C)	↕	↕	X	0	X	P	0	•	Input register indirect
INI, INIR, IND, INDR, OUTI, OTIR, OUTD, OTDR	X	↕	X	X	X	X	1	•	Block input and output Z = 0 if B ≠ 0 otherwise Z = 1
LDI, LDD	X	X	X	0	X	↕	0	•	Block transfer instructions
LDIR, LDDR	X	X	X	0	X	0	0	•	P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI, CPIR, CPD, CPDR	X	↕	X	X	X	↕	1	•	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	↕	↕	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	X	↕	X	↕	X	X	0	•	The state of bit b of location s is copied into the Z flag

The following notation is used in this table:

SYMBOL	OPERATION
C	Carry/link flag. C=1 if the operation produced a carry from the MSB of the operand or result.
Z	Zero flag. Z=1 if the result of the operation is zero.
S	Sign flag. S=1 if the MSB of the result is one.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V=1 if the result of the operation is even, P/V=0 if result is odd. If P/V holds overflow, P/V=1 if the result of the operation produced an overflow.
H	Half-carry flag. H=1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.
N	Add/Subtract flag. N=1 if the previous operation was a subtract.
	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care".
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range <0, 255>
nn	16-bit value in range <0, 65535>

Tabla B.17 Cargado de 8 bits.

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r, s	r ← s	•	•	X	•	X	•	•	•	01 r s		1	1	4	r, s Reg
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110 ← n →		2	2	7	000 B 001 C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010 D
LD r, (IX+d)	r ← (IX+d)	•	•	X	•	X	•	•	•	11 011 101 01 r 110 ← d →	DD	3	5	19	011 E 100 H 101 L
LD r, (IY+d)	r ← (IY+d)	•	•	X	•	X	•	•	•	11 111 101 01 r 110 ← d →	FD	3	5	19	111 A
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7	
LD (IX+d), r	(IX+d) ← r	•	•	X	•	X	•	•	•	11 011 101 01 110 r ← d →	DD	3	5	19	
LD (IY+d), r	(IY+d) ← r	•	•	X	•	X	•	•	•	11 111 101 01 110 r ← d →	FD	3	5	19	
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110 ← n →	36	2	3	10	
LD (IX+d), n	(IX+d) ← n	•	•	X	•	X	•	•	•	11 011 101 00 110 110 ← d → ← n →	DD 36	4	5	19	
LD (IY+d), n	(IY+d) ← n	•	•	X	•	X	•	•	•	11 111 101 00 110 110 ← d → ← n →	FD 36	4	5	19	
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010 ← n → ← n →	3A	3	4	13	
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010 ← n → ← n →	32	3	4	13	
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101 01 010 111	ED 57	2	2	9	
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101 01 011 111	ED 5F	2	2	9	
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101 01 000 111	ED 47	2	2	9	
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101 01 001 111	ED 4F	2	2	9	

Notes: r, s means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation

Tabla B.18 Cargado de 16 bits.

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments	
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00	dd0	001		3	3	10	dd	Pair
										←	n	→					00	BC
										←	n	→					01	DE
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	11	011	101	DD	4	4	14	10	HL
										00	100	001	21				11	SP
										←	n	→						
										←	n	→						
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	11	111	101	FD	4	4	14		
										00	100	001	21					
										←	n	→						
										←	n	→						
LD HL, (nn)	H ← (nn+1)	•	•	X	•	X	•	•	•	00	101	010	2A	3	5	16		
	L ← (nn)									←	n	→						
										←	n	→						
LD dd, (nn)	dd$_H$ ← (nn+1)	•	•	X	•	X	•	•	•	11	101	101	ED	4	6	20		
	dd$_L$ ← (nn)									01	dd1	011						
										←	n	→						
										←	n	→						
LD IX, (nn)	IX$_H$ ← (nn+1)	•	•	X	•	X	•	•	•	11	011	101	DD	4	6	20		
	IX$_L$ ← (nn)									00	101	010	2A					
										←	n	→						
										←	n	→						
LD IY, (nn)	IY$_H$ ← (nn+1)	•	•	X	•	X	•	•	•	11	111	101	FD	4	6	20		
	IY$_L$ ← (nn)									00	101	010	2A					
										←	n	→						
										←	n	→						
LD (nn), HL	(nn+1) ← H	•	•	X	•	X	•	•	•	00	100	010	22	3	5	16		
	(nn) ← L									←	n	→						
										←	n	→						
LD (nn), dd	(nn+1) ← dd$_H$	•	•	X	•	X	•	•	•	11	101	101	ED	4	6	20		
	(nn) ← dd$_L$									01	dd0	011						
										←	n	→						
										←	n	→						
LD (nn), IX	(nn+1) ← IX$_H$	•	•	X	•	X	•	•	•	11	011	101	DD	4	6	20		
	(nn) ← IX$_L$									00	100	010	22					
										←	n	→						
										←	n	→						
LD (nn), IY	(nn+1) ← IY$_H$	•	•	X	•	X	•	•	•	11	111	101	FD	4	6	20		
	(nn) ← IY$_L$									00	100	010	22					
										←	n	→						
										←	n	→						
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	11	111	001	F9	1	1	6		
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	11	011	101	DD	2	2	10		
										11	111	001	F9					
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	11	111	101	FD	2	2	10		
										11	111	001	F9				qq	Pair
PUSH qq	(SP-2) ← qq$_L$	•	•	X	•	X	•	•	•	11	qq0	101		1	3	11	00	BC
	(SP-1) ← qq$_H$																01	DE
PUSH IX	(SP-2) ← IX$_L$	•	•	X	•	X	•	•	•	11	011	101	DD	2	4	15	10	HL
	(SP-1) ← IX$_H$									11	100	101	E5				11	AF
PUSH IY	(SP-2) ← IY$_L$	•	•	X	•	X	•	•	•	11	111	101	FD	2	4	15		
	(SP-1) ← IY$_H$									11	100	101	E5					
POP qq	qq$_H$ ← (SP+1)	•	•	X	•	X	•	•	•	11	qq0	001		1	3	10		
	qq$_L$ ← (SP)																	
POP IX	IX$_H$ ← (SP+1)	•	•	X	•	X	•	•	•	11	011	101	DD	2	4	14		
	IX$_L$ ← (SP)									11	100	001	E1					
POP IY	IY$_H$ ← (SP+1)	•	•	X	•	X	•	•	•	11	111	101	FD	2	4	14		
	IY$_L$ ← (SP)									11	100	001	E1					

Notes: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
(PAIR)$_H$, (PAIR)$_L$ refer to high order and low order eight bits of the register pair respectively
e.g. BC$_L$ = C, AF$_H$ = A

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ flag is affected according to the result of the operation

Tabla B.19 Grupo de intercambio, transferencia y búsqueda de bloques.

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE↔HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF'	AF↔AF'	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	(BC↔BC' DE↔DE' HL↔HL')	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H↔(SP+1) L↔(SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX (SP), IX	(IX$_H$)↔(SP+1) (IX$_L$)↔(SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	(IY$_H$)↔(SP+1) (IY$_L$)↔(SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE)←(HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	•	•	X	0	X	[1]	0	•	11 101 101 10 100 000	ED A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE)←(HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
LDD	(DE)←(HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	•	•	X	0	X	[1]	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE)←(HL) DE ← DE-1 HL ← HL-1 BC ← BC-1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A − (HL) HL ← HL+1 BC ← BC-1	[2]	[2]	X	[1]	X	[1]	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A − (HL) HL ← HL+1 BC ← BC-1 Repeat until A = (HL) or BC = 0	[2]	[2]	X	[1]	X	[1]	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A − (HL) HL ← HL-1 BC ← BC-1	[2]	[2]	X	[1]	X	[1]	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A − (HL) HL ← HL-1 BC ← BC-1 Repeat until A = (HL) or BC = 0	[2]	[2]	X	[1]	X	[1]	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

Notes: [1] P/V flag is 0 if the result of BC-1 = 0, otherwise P/V = 1
[2] Z flag is 1 if A = (HL), otherwise Z = 0.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Tabla B.20 Grupo de aritmética y lógica de 8 bits.

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A,r	A ← A+r	↕	↕	X	↕	X	V	0	↕	10 000 r		1	1	4	r Reg.
ADD A,n	A ← A+n	↕	↕	X	↕	X	V	0	↕	11 000 110		2	2	7	000 B
										← n →					001 C
															010 D
ADD A,(HL)	A ← A+(HL)	↕	↕	X	↕	X	V	0	↕	10 000 110		1	2	7	011 E
ADD A,(IX+d)	A ← A+(IX+d)	↕	↕	X	↕	X	V	0	↕	11 011 101	DD	3	5	19	100 H
										10 000 110					101 L
										← d →					111 A
ADD A,(IY+d)	A ← A+(IY+d)	↕	↕	X	↕	X	V	0	↕	11 111 101	FD	3	5	19	
										10 000 110					
										← d →					
ADC A,s	A ← A+s+CY	↕	↕	X	↕	X	V	0	↕	001					s is any of r, n, (HL), (IX+d), (IY+d) as shown for ADD instruction. The indicated bits replace the 000 in the ADD set above.
SUB s	A ← A − s	↕	↕	X	↕	X	V	1	↕	010					
SBC A,s	A ← A − s − CY	↕	↕	X	↕	X	V	1	↕	011					
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0	100					
OR s	A ← A ∨ s	↕	↕	X	0	X	P	0	0	110					
XOR s	A ← A ⊕ s	↕	↕	X	0	X	P	0	0	101					
CP s	A − s	↕	↕	X	↕	X	V	1	↕	111					
INC r	r ← r+1	↕	↕	X	↕	X	V	0	•	00 r 100		1	1	4	
INC (HL)	(HL) ← (HL)+1	↕	↕	X	↕	X	V	0	•	00 110 100		1	3	11	
INC (IX+d)	(IX+d) ← (IX+d)+1	↕	↕	X	↕	X	V	0	•	11 011 101	DD	3	6	23	
										00 110 100					
										← d →					
INC (IY+d)	(IY+d) ← (IY+d)+1	↕	↕	X	↕	X	V	0	•	11 111 101	FD	3	6	23	
										00 110 100					
										← d →					
DEC s	s ← s − 1	↕	↕	X	↕	X	V	1	•	101					s is any of r, (HL), (IX+d), (IY+d) as shown for INC. DEC same format and states as INC. Replace 100 with 101 in OP Code.

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation. Similarly, the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow. P = 1 means parity of the result is even, P = 0 means parity of the result is odd.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ↕ = flag is affected according to the result of the operation.

Tabla B.21 Grupos de aritmética, propósito general y control del CPU.

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	Decimal adjust accumulator
CPL	$A - \overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Complement accumulator (One's complement)
NEG	$A - \overline{A} + 1$	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Negate acc. (two's complement)
CCF	$CY - \overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag
SCF	CY - 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Set carry flag
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI*	IFF - 0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI*	IFF - 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation

*Interrupts are not sampled at the end of EI or DI

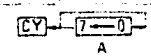
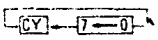
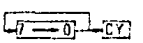
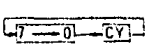
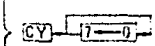
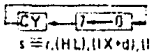
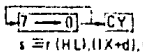
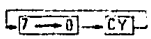
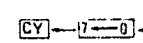
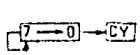
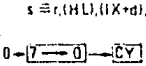
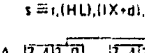
Tabla B.22 Grupo de aritmética de 16 bits.

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments	
ADD HL, ss	HL ← HL+ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss	Reg.
															00	BC
ADC HL, ss	HL ← HL+ss+CY	↕	↕	X	X	X	V	0	↕	11 101 101	ED	2	4	15	01	DE
										01 ss1 010					10	HL
															11	SP
SBC HL, ss	HL ← HL-ss-CY	↕	↕	X	X	X	V	1	↕	11 101 101	ED	2	4	15		
										01 ss0 010						
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101	DD	2	4	15	pp	Reg.
										00 pp1 001					00	BC
															01	DE
															10	IX
															11	SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101	FD	2	4	15	rr	Reg.
										00 rr1 001					00	BC
															01	DE
															10	IY
															11	SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6		
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101	DD	2	2	10		
										00 100 011	23					
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101	FD	2	2	10		
										00 100 011	23					
DEC ss	ss ← ss - 1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6		
DEC IX	IX ← IX - 1	•	•	X	•	X	•	•	•	11 011 101	DD	2	2	10		
										00 101 011	2B					
DEC IY	IY ← IY - 1	•	•	X	•	X	•	•	•	11 111 101	FD	2	2	10		
										00 101 011	2B					

Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Tabla B.23 Grupo de rotaciones y corrimientos.

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Op Code 7 6 5 4 3 2 1 0	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		•	•	X	0	X	•	0	0 0 0 0 0 1 1 1	07	1	1	4	Rotate left circular accumulator
RLA		•	•	X	0	X	•	0	0 0 0 1 0 1 1 1	17	1	1	4	Rotate left accumulator
RRCA		•	•	X	0	X	•	0	0 0 0 0 1 1 1 1	0F	1	1	4	Rotate right circular accumulator
RRA		•	•	X	0	X	•	0	0 0 0 1 1 1 1 1	1F	1	1	4	Rotate right accumulator
RLC r		‡	‡	X	0	X	P	0	1 1 0 0 1 0 1 1 0 0 0 0 0 r	CB	2	2	8	Rotate left circular register
RLC (HL)		‡	‡	X	0	X	P	0	1 1 0 0 1 0 1 1 0 0 0 0 0 1 1 0	CB	2	4	15	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX+d)	 r,(HL),(IX+d),(IY+d)	‡	‡	X	0	X	P	0	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 - d - 0 0 0 0 0 1 1 0	DD CB	4	6	23	
RLC (IY+d)		‡	‡	X	0	X	P	0	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 - d - 0 0 0 0 0 1 1 0	FD CB	4	6	23	
RL s	 s ≡ r,(HL),(IX+d),(IY+d)	‡	‡	X	0	X	P	0	010					Instruction format and states are as shown for RLC s. To form new Op Code replace 000 of RLC s with shown code
RRC s	 s ≡ r,(HL),(IX+d),(IY+d)	‡	‡	X	0	X	P	0	001					
RR s	 s ≡ r,(HL),(IX+d),(IY+d)	‡	‡	X	0	X	P	0	011					
SLA s	 s ≡ r,(HL),(IX+d),(IY+d)	‡	‡	X	0	X	P	0	100					
SRA s	 s ≡ r,(HL),(IX+d),(IY+d)	‡	‡	X	0	X	P	0	101					
SRL s	 s ≡ r,(HL),(IX+d),(IY+d)	‡	‡	X	0	X	P	0	111					
RLD		‡	‡	X	0	X	P	0	• 1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 1	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected
RRD		‡	‡	X	0	X	P	0	• 1 1 1 0 1 1 0 1 0 1 1 0 0 1 1 1	ED 67	2	5	18	

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
‡ = flag is affected according to the result of the operation.

Tabla B.24 Grupo de prueba, pone y borra bit.

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States
BIT b, r	$Z = \overline{r_b}$	X	‡	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	8
BIT b, (HL)	$Z = \overline{(HL)_b}$	X	‡	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12
BIT b, (IX+d)	$Z = \overline{(IX+d)_b}$	X	‡	X	1	X	X	0	•	11 011 101 11 001 011 - d - 01 b 110	DD CB	4	5	20
BIT b, (IY+d)	$Z = \overline{(IY+d)_b}$	X	‡	X	1	X	X	0	•	11 111 101 11 001 011 - d - 01 b 110	FD CB	4	5	20
SET b, r	$r_b = 1$	•	•	X	•	X	•	•	•	11 001 011 11 b r	CB	2	2	8
SET b, (HL)	$(HL)_b = 1$	•	•	X	•	X	•	•	•	11 001 011 11 b 110	CB	2	4	15
SET b, (IX+d)	$(IX+d)_b = 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 - d - 11 b 110	DD CB	4	6	23
SET b, (IY+d)	$(IY+d)_b = 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 - d - 11 b 110	FD CB	4	6	23
RES b, s	$s_b = 0$ s ≡ r, (HL), (IX+d), (IY+d)	•	•	X	•	X	•	•	•	10				

Comments

r	Reg
000	B
001	C
010	D
011	E
100	H
101	L
111	A

b	Bit Tested
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

To form new Op-Code replace 11 of SET b, s with 10. Flags and time states for SET instruction

Notes: The notation s_b indicates bit b (0 to 7) of location s.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ‡ = flag is affected according to the result of the operation.

Tabla B. 25 Grupo de saltos.

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 ← n → ← n →	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 ← n → ← n →		3	3	10	cc Condition 000 NZ non zero 001 Z zero 010 NC non carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000 ← e-2 →	18	2	3	12	
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 ← e-2 →	38	2	2	7	If condition not met
	If C = 1, PC ← PC+e											2	3	12	If condition is met
JR NC, e	If C = 1, continue	•	•	X	•	X	•	•	•	00 110 000 ← e-2 →	30	2	2	7	If condition not met
	If C = 0, PC ← PC+e											2	3	12	If condition is met
JR Z, e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 ← e-2 →	28	2	2	7	If condition not met
	If Z = 1, PC ← PC+e											2	3	12	If condition is met
JR NZ, e	If Z = 1, continue	•	•	X	•	X	•	•	•	00 100 000 ← e-2 →	20	2	2	7	If condition not met
	If Z = 0, PC ← PC+e											2	3	12	If condition is met
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	B ← B-1 If B = 0, continue	•	•	X	•	X	•	•	•	00 010 000 ← e-2 →	10	2	2	8	If B ≠ 0
	If B ≠ 0, PC ← PC+e											2	3	13	If B ≠ 0

Notes: e represents the extension in the relative addressing mode.

e is a signed two's complement number in the range <126, 129>

e-2 in the op-code provides an effective address of pc+e as PC is incremented by 2 prior to the addition of e

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown, ↕ = flag is affected according to the result of the operation

Tabla B.26 Grupo de llamadas y regresos.

Mnemonic	Symbolic Operation	S	Z	X	H	X	P/V	N	C	Op Code 76	543	210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	(SP-1) ← PC_H (SP-2) ← PC_L PC ← nn	•	•	X	•	X	•	•	•	11 - -	001 n n	101 - -	CD	3	5	17	
CALL cc, nn	If condition cc is false	•	•	X	•	X	•	•	•	11 -	cc n	100 -		3	3	10	If cc is false
	continue, otherwise same as CALL nn									-	n	-		3	5	17	If cc is true
RET	PC_L ← (SP) PC_H ← (SP+1)	•	•	X	•	X	•	•	•	11	001	001	C9	1	3	10	
RET cc	If condition cc is false	•	•	X	•	X	•	•	•	11	cc	000		1	1	5	If cc is false
	continue, otherwise same as RET													1	3	11	If cc is true
RETI	Return from interrupt	•	•	X	•	X	•	•	•	11 01	101 001	101 101	ED 4D	2	4	14	
RETN[1]	Return from non maskable interrupt	•	•	X	•	X	•	•	•	11 01	101 000	101 101	ED 45	2	4	14	
RST p	(SP-1) ← PC_H (SP-2) ← PC_L PC_H ← 0 PC_L ← p	•	•	X	•	X	•	•	•	11	t	111		1	3	11	

cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

[1] RETN loads IFF_2 → IFF_1

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Tabla B.27 Grupo de entradas y salidas.

Mnemonic	Symbolic Operation	Flags S	Z	H	P/V	N	C	Op Code 76 543 210	Hex	No of Bytes	No of M Cycles	No of T States	Comments
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011	DB	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕	X	P	X	0	•		11 101 101 01 r 000	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	X	1 ↕	X	X	X	X	1	•	11 101 101 10 100 010	ED A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 010	ED B2	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	X	1 ↕	X	X	X	X	1	•	11 101 101 10 101 010	ED AA	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 010	ED BA	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011	D3	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 01 r 001	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	X	1 ↕	X	X	X	X	1	•	11 101 101 10 100 011	ED A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 011	ED B3	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	X	1 ↕	X	X	X	X	1	•	11 101 101 10 101 011	ED AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTDR	(C) ← (HL) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 011	ED BB	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

Notes: 1: If the result of B - 1 is zero the Z flag is set, otherwise it is reset

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation

APENDICE C

EJEMPLOS DE PROGRAMACION DEL MC6809

Este apendice tiene como finalidad, mostrar al lector el formato de algunas de las instrucciones del MC6809 — que pudieran resultar más complicadas.

Los programas indican la forma que deben llevar los mnemónicos, aunque debemos aclarar que tal como se muestran estos programas, no sería posible trabajar con ellos en la microcomputadora Cromenco, con la cual se simuló la programación del MC6809, ya que es necesario contar con el ensamblador de este microprocesador (MC6809). Esto es, el programa simulador diseñado necesita del ensamblador para poder traducir las instrucciones indicadas en mnemónicos.

```
;          D E F I N I C I O N

; MULTIPLICACION        ( MULTIP )

; Elaboró : A. E. H.                    Fecha : 21/XI/83

; Objetivo :
;      Realizar una multiplicación entre dos números, los -
;      cuales tendrán 16 bits de mantisa y exponente de 8 -
;      bits .

; Datos de entrada :
;      - Byte menos significativo de la mantisa del multipli
;        cando; etiqueta "MULTI1B" .
;      - Byte más significativo de la mantisa del multipli--
;        cando; etiqueta "MULTI1A" .
;      - Exponente del multiplicando; etiqueta "EXP1"
;      - Byte menos significativo de la mantisa del multipli
;        cador; etiqueta "MULTI2B" .
;      - Byte más significativo de la mantisa del multiplica
;        dor; etiqueta "MULTI2A" .
;      - Exponente del multiplicador ; etiqueta "EXP2" .

; Datos de salida :
;      - Byte de resultado de exponente; etiqueta "EXPNTE".
;      - Byte más significativo del resultado; etiqueta ---
;        "RESUL4" .
;      - 2o. Byte más significativo del resultado; etiqueta
;         "RESUL3" .
;      - 3er. byte más significativo del resultado; etiqueta
;        "RESUL2" .
;      - Byte menos significativo del resultado;  etiqueta
;        "RESUL1" .
```

```
; Procedimiento :
;       Se suman los exponentes y el resultado se guarda en -
;       una etiqueta llamada "EXPNTE". Se multiplican las man
;       tisas, trabajadas en bytes, y el resultado se guarda_
;       en las etiquetas "RESUL4", "RESUL3", "RESUL2"   y --
;       "RESUL1"  .

;////////////////////////////////////////////////////////////
;       PSEUDO-CODIGO                   21/XI/83
;                       ( MULTIP )

; Inicio
;    1.- Inicializar etiquetas
;    2.- Sumar los exponentes  ( "EXP1" + "EXP2" ) y cargar_
;        el resultado en "EXPNTE" .
;    3.- ACCA := MULTI1B
;    4.- ACCB := MULTI2B
;    5.- Ejecuta  MUL
;    6.- RESUL1 := ACCB
;    7.- PROD1B := ACCA
;    8.- ACCA := MULTI1A
;    9.- ACCB := MULTI2B
;   10.- Ejecuta  MUL
;   11.- PROD2B := ACCB
;   12.- PROD2C := ACCA
;   13.- ACCA := PROD1B
;   14.- ACCA := ACCA + PROD2B
;   15.- RP1B := ACCA
;   16.- ACCA := PROD2C
;   17.- ACCA := ACCA + Carry
;   18.- RP1C := ACCA
;   19.- Si carry = Ø
;        A.- Entonces; salta a paso 21
```

```
;    20.- Registro indice "X"  := 1
;    21.- RPCAR ;= registro indice "X"
;    22.- ACCA :="MULTI1B"
;    23.- ACCB := MULTI2A
;    24.- Ejecuta MUL
;    25.- RP2A := ACCB
;    26.- PROD1B := ACCA
;    27.- ACCA := MULTI1A
;    28.- ACCB := MULTI2A
;    29.- Ejecuta MUL
;    30.- PROD2B := ACCB
;    31.- PROD2C := ACCA
;    32.- ACCA := PROD1B
;    33.- ACCA := ACCA + PROD2B
;    34.- RP2B := ACCA
;    35.- ACCA := PROD2C
;    36.- ACCA := ACCA + Carry
;    37.- RP2C := ACCA
;    38.- ACCB := RP1B
;    39.- ACCB := ACCB + RP2A
;    40.- RESUL2 := ACCB
;    41.- ACCB := RP1C
;    42.- ACCB ;= ACCB + RP2B + Carry
;    43.- RESUL3 := ACCB
;    44.- ACCB := RP2C
;    45.- ACCB := ACCB + Carry
;    46.- ACCA := Ø
;    47.- ACCD := ACCD + RPCAR
;    48.- RESUL4 := ACCB

; Fin

;////////////////////////////////////////////////////////////////
```

```
;                CODIGO                          21/XI/83
;                                  MULTIP )

      EXP1   :  DB                        ; 8 bits indicados; 1.-
      EXP2   :  DB                        ; por el usuario.
      EXPNTE:   DS 1                        .
     MULTI1A:   DB                        ; 16   bits
     MULTI1B:   DB                        ; indicados  (USUARIO)
     MULTI2A:   DB                        ; 16 bits indicados
     MULTI2B:   DB                        ; por el usuario
     PROD2B :   DS 1
     PROD2C :   DS 1
     RP1B   :   DS 1
     RP1C   :   DS 1
     RP2A   :   DS 1
     RP2B   :   DS 1
     RP2C   :   DS 1
     RESUL1 :   DS 1
     RESUL2 :   DS 1
     RESUL3 :   DS 1
     RESUL4 :   DS 1
     RPCAR  :   DS 2
     MULTIP :   LDA     EXP1             ; 2.-
                ADDA    EXP2
                STA     EXPNTE
                LDA     MULTI1B          ; 3.-
                LDB     MULTI2B          ; 4.-
                MUL                      ; 5.-
                STB     RESUL1           ; 6.-
                STA     PROD1B           ; 7.-
                LDA     MULTI1A          ; 8.-
                LDB     MULTI2B          ; 9.-
                MUL                      ; 10.-
                STB     PROD2B           ; 11.-
```

```
            STA     PROD2C      ; 12.-
            LDX     # Ø        ; indicador de carry
            LDA     PROD1B      ; 13.-
            ADDA    PROD2B      ; 14.-
            STA     RP1B        ; 15.-
            LDA     PROD2C      ; 16.-
            ADCA    # Ø        ; 17.-
            STA     RP1C        ; 18.-
            BCC     MULTIP1     ; 19.-
            LDX     # 1         ; 20.-
MULTIP1:    STX     RPCAR       ; 21.-
            LDA     MULTI1B     ; 22.-
            LDB     MULTI2A
            MUL
            STB     RP2A
            STA     PROD1B
            LDA     MULTI1A
            LDB     MULTI2A
            MUL                 ; 29.-
            STB     PROD2B
            STA     PROD2C
            LDA     PROD1B
            ADDA    PROD2B      ; 33.-
            STA     RP2B
            LDA     PROD2C
            ADCA    # Ø        ; 36.-
            STA     RP2C
            LDB     RP1B
            ADDB    RP2A
            STB     RESUL2      ; 40.-
            LDB     RP1C
            ADCB    RP2B        ; 42.-
            STB     RESUL3
            LDB     RP2C
```

```
ADCB    # Ø           ; 45.-
LDA     # Ø
ADDD    RPCAR
STB     RESUL4        ; 48.-
RTS
```

```
;////////////////////////////////////////////////////////////
;////////////////////////////////////////////////////////////

;                    D E F I N I C I O N

; DIRECCIONAMIENTO DE DATOS        ( EJEMP )

; Elaboró :  A. E. H.                    Fecha : 20/I/84

; Objetivo :
;     Obtener datos con modos de direccionamiento
;     diferentes para despues sumarlos.

; Datos de entrada :
;     - Contenido de los registros "DP", "RIX","RIY", "US" .
;     - Direcciones que van concatenadas con el registro DP.
;       ( etiquetas; DIR1, DIR2 y DIR3 )
;     - Valores de los desplazamientos de 5 y 8 bits.
;       ( etiquetas;  n, DESP5Y  ; respectivamente ).

; Datos de salida :
;     - Resultado de la suma de los datos apuntados
;       por las direcciones obtenidas.

; Procedimiento :
;     Con el modo de direccionamiento directo se obtienen  -
```

```
;     los tres primeros datos ( DP:DIR1 , DP:DIR2 , DP:DIR3)
;     y se suman.
;     Con el modo de direccionamiento de indexado se realiza
;     un desplazamiento de 5 bits ( etiqueta "n" ) en el re-
;     gistro "RIX" , y un desplazamiento de 8 bits ( etique-
;     ta "DESP5Y" ) en el registro "RIY", obteniendose así -
;     las direcciones efectivas de otros dos datos, los cua-
;     les son adicionados a la suma anterior.
;     Por último se realiza un decremento de 2 ( direcciona-
;     miento indirecto ) en el contenido del stack del usua-
;     rio ( US ) este valor ( 16 bits ) es la dirección efec
;     tiva del dato que nos interesa adicionar a la suma an-
;     terior. La suma de todos los datos obtenidos quedará -
;     en la etiqueta "RES" .

;////////////////////////////////////////////////////////////

;     PSEUDO-CODIGO                    20/I/84
;                      ( EJEMP )

; Inicio
;    1.-  Cargar registros ( DP, RIX, RIY, US )
;    2.-  Cargar valores ( 8 bits ) en las etiquetas DIR1 ,
;         DIR2, DIR3 .
;    3.-  Cargar valores de los desplazamientos de 5 y 8 bits
;         ( "n"  y  "DESP5Y" )
;    4.-  Sumar los datos apuntados  por DIR1, DIR2, DIR3 con
;         direccionamiento directo  .
;    5.-  Tomar en cuenta la bandera de carry .
;    6.-  Guardar resultado en etiqueta "RES" .
;    7.-  Realizar un desplazamiento de 5 bits en complemento
;         de dos ( +  ó  - ) en la dirección apuntada por RIX,
;         El dato apuntado por esta nueva dirección sumarlo a
;         "RES"  .
```

```
;    8.- Realizar un desplazamiento de 8 bits en complemento
;        de dos ( + ó - ) en la dirección apuntada por RIY,
;        El dato apuntado por esta nueva dirección sumarlo a
;        "RES".
;    9.- Realizar un decremento de dos al valor del registro
;        US ( direccionamiento de indexado ). El dato obteni
;        do es la dirección efectiva ( direccionamiento indi
;        recto ) del nuevo dato que es sumado a "RES"

; Fin

;////////////////////////////////////////////////////////////

;            CODIGO                    20/I/84
;                    ( EJEMP )

      PARCIAL:  DS  1
      RES     :  DS  2
      DIR1    :  EQU                   ; 2.-
      DIR2    :  EQU
      DIR3    :  EQU
      n       :  EQU                   ; 3.-
      DESP5Y  :  EQU
      PARTE2  :  DS  1
      EJEMP   :  LDA     DIR1          ; 4.-
                 ADDA    DIR2
                 ADCA    DIR3          ; 5.-
                 LDB     # Ø           ; dir. inmediato
                 EXG     A,B
                 STD     RES           ; 6.-
                 BCC     SALTO
                 LDA     #% 1          ; Valor inm. binario
                 STD     RES
      SALTO   :  LDB     n,X           ; 7.-
```

```
                LDA     DESP5Y          ; 8.-
                STA     PARCIAL
                ADDB    PARCIAL
                LDA     ( , --U )       ; 9.-
                STA     PARTE2
                LDA     #$Ø             ; inm. hexadecimal
                ADCB    PARTE2
                BCC     SALTO2
                LDA     #% 1
SALTO2 :        ADDD    RES
                STD     RES
                RTS
```

;//
;//
;//

I N D I C E A L F A B E T I C O
D E R U T I N A S

LLaTD	102	308	308
LLaTAL	106	312	312
LLaTTA	128	261	261
LEA	146	266	267
LoR	108	255	255
MUL	156	301	302
MBG	106	253	253
MODBF	96	221	222
MOPO	134	267	267
MULO	193	345	345
OBTDLA	84	205	205
OBTEA	191	206	207
OBTOP	82	201	202
OPbBT6	93	217	218
OPbBT8	101	227	228
OPbBT16	102	228	229
OPBI	94	218	219
OPBII	95	219	220
OPBIII	96	220	221
OPBIV	101	307	307
OPBV	103	309	309
OPBVI	171	317	317
OPBVII	175	321	321
OPBLOG	172	318	318
ORCC	138	270	271
PAGB2	119	247	247
PAGB3	131	265	265
PGOF8	103	250	251
PGOF16	104	251	251

::