ESCUELA SUPERIOR POLITECNICA DEL LITORAL

DEPARTAMENTO DE INGENIERIA ELECTRICA

" Ensamblador Cruzado para Programación de Microprocesadores 8080 y 8085 a Usarse en el Laboratorio de Microprocesadores de la ESPOL "

TESIS DE GRADO

PREVIA A LA OBTENCION DEL TITULO DE

INGENIERO EN ELECTRICIDAD
ESPECIALIDAD: ELECTRONICA

POR

MARCO ESTUARDO LOPEZ ORTUÑO

GUAYAQUIL - ECUADOR

1980

"ENSAMBLADOR CRUZADO PARA PROGRAMACION DE MICROPROCESADORES 8080 Y 8085 A USARSE EN EL LABORATORIO DE MICROPROCESADORES DE LA ESPOL"

DIRECTOR DE TESIS

ING. PEDRO CARLO PAREDES

AUTOR

MARCO E. LOPEZ ORTUNO

AGRADECIMIENTO

Agradezco a la comunidad Politécnica, a mis maestros, a mis compañeros, amigos y más personas que con su ayuda y consejo han colaborado en la realización de este trabajo. Agradezco de manera especial al Ing. Sergio Flores, mentalizador del tema y original Director de tesis y ex-director del Dpto. de Ing. Eléctrica; al Ing. Pedro Carló, Director de Tesis y ex-subdirector del Dpto. de Ing. Eléctrica; y al Ing. Juan del Pozo, Ex-Director del Dpto. de Ing. Eléctrica por sus valiosas sugerencias.

También de manera especial, agradezco la comprensión, ayuda, consejo y cariño de mi querida Esposa y mis queridos hijos.

MARCO LOPEZ ORTUÑO

DEDICATORIA

Con mucho cariño:

A mi querida esposa: DIANA ELOISA

y a mis queridos hijos:

MARCO GIOVANNI

XAVIER ESTUARDO

MARIO FRANCISCO

RAUL RENATO

DIANITA ISABEL

PROLOGO

La avanzada tecnología de los tiempos actuales de la humanidad no hubiera sido posible sin el concurso de la ciencia más moderna y apasionante, desarrollada por el espíritu de investigación que incansablemente anima a los seres humanos pensantes:

LA ELECTRONICA MODERNA

Esta ciencia, que dió sus primeros pasos como una determinante ayuda a las comunicaciones se identificó con ésta, a tal punto que se confundía electrónica con comunicaciones. Sin embargo, su verdadera identificación, oculta en sus primeras aplicaciones, fue el sueño dorado de la humanidad:

LA AUTOMATIZACION

Se puede demostrar la afirmación de que el estudio de los fenómenos electrónicos siempre estuvo asociada a la automatización con un ejemplo sencillo: Un receptor de radio no es sino un dispositivo gobernado por telecomando que

una vez en funcionamiento, debidamente sintonizado y ajustado su nivel de volumen de sonido, reproduce la voz o la música que se origina en algún lugar distante. Esra reproducción del sonido es AUTOMATICA en el transductor llamado parlante que obra gobernado o controlado electrónicamente.

Ejemplo más complicado es un receptor de TV en donde la automatización adquiere mayor intensidad en comunicaciones comerciales.

Y así, podría dedicarse un volumen entero a la demostración de que ELECTRONICA es sinónimo de AUTOMATIZACION desde su nacimiento.

Asi se explica el hecho de que el desarrollo de la automatización se efectua paralelo al desarrollo de la electrónica, más aún el increíble grado de perfección, rapidez y precisión que ha alcanzado la automatización solo ha sido posible mediante los logros tecnológicos de la Electrónica moderna.

Y es justamente el ultra rápido desarrollo de esta tecnología el que nos ha maravillado con diseños cada vez más eficientes, más pequeños, mas baratos, mas comple

tos, etc. Con lo que el fascinante mundo de la automatización y su máxima expresión: EL ORDENADOR ELECTRONICO, comunmente llamado computadora, maravilla de entendidos y profanos, científicos y analfabetos, ha alcanzado nivcles increíbles de sofisticada versatilidad de ilimitadas aplicaciones con la aparición de el "MICROPROCESADOR" que es un pequeño, pequeñísimo componente microeléctrónico que cada vez integra mas funciones a tal punto que es posible que en un futuro no muy lejano tengamos una computadora completa en un pequeño circuito integrado.

En este trabajo se analizarán las funciones de los microprocesadores 8080/8085 de INTEL para entregar programas en código objeto de sus instrucciones de máquina.

En esta obra se sintetiza muchos años de estudio en la Escuela Superior Politécnica del Litoral, durante los cuales la labor de conjunto del sistema compuesto de profesores, instructores y estudiantes, se ha puesto de manifiesto, para demostrar que nuestra Institución es la mejor del País.

INDICE GENERAL

Pág.

Pág.

Pág.

DECLARACION EXPRESA:

DECLARO QUE: Hechos, ideas y doctrinas expuestos en esta tesis son de responsabilidad exclusiva de su autor y que el patrimonio intelectual de la misma corresponde a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL.

(Reglamento de Exámenes y Títulos Profesionales de la ESPOL).

RESUMEN

Los microcomputadores, como los que están en uso en el Laboratorio de Microcomputadores de nuestra Escuela, solo disponen de facilidades para programarlos en código hexadecimal del lenguaje de máquina de su microprocesador; para prácticas preliminares y con programas extremadamente cortos, esto es suficiente, pero para experiencias más útiles se desperdicia mucho tiempo con tal modo de trabajo; de manera que el Ensamblador cruzado para programación de microprocesadores 8080 y 8085 servirá de ayuda y ahorro de tiempo, ya que permite programar en lenguaje Ensamblador, que lo acepta, y entrega el mismo programa, en el código hexadecimal, listo para usarse en el microcomputador.

Con tal propósito se analizaron los siguientes aspectos:

- Organización de computadoras - sistemas operativos
- Microprocesadores - Microcomputadores
- Microprocesador 8080 - Microprocesador 8085
- Memorias
- Interfases

- Compiladores - traductores - Interpretadores - Simuladores.
- Programación de sistemas
- Lenguaje ensamblador 8080/8085
- Programación - uso y manejo del lenguaje Ensamblador 8080/8085.
- Lenguaje de diagramas de flujo

Luego esa información se tradujo en la elaboración del Ensamblador cruzado 8080/8085 como programa principal al que asiste un programa secundario elaborado con el nombre de EDITOR.

LISTA DE FIGURAS

FIG.

LISTA DE TABLAS

CAPITULO 1

INTRODUCCION

GENERALIDADES

Cualquier movimiento u operación conocida se puede efectuar y controlar automáticamente si se dispone de métodos y dispositivos adecuados.

El control automático más eficiente, rápido y preciso, elaborado por el hombre, se ejecuta desde una computadora que obedeciendo un programa establecido, regula el funcionamiento de movimientos mecánicos, ejecuta operaciones matemáticas, establece condiciones adecuadas de funcionamiento de acuerdo a informaciones recibidas del sistema controlado, etc. etc.

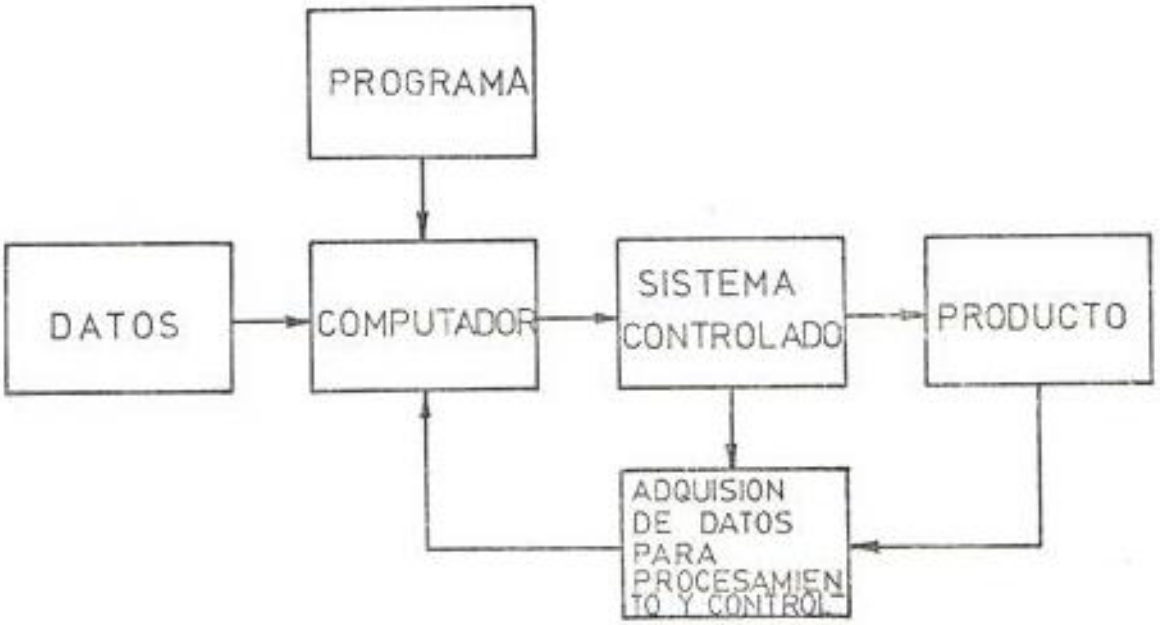


FIG. 1.1 UN SISTEMA GENERAL

1.1. EL SISTEMA CONTROLADO

Puede ser cualquier cosa: una operación matemática, un terminal o una máquina, del funcionamiento de la cual depende el producto para la que fue diseñada.

1.1.1. LOS DATOS

Los datos que recibe y procesa la computadora pueden tener diversa procedencia:

a) Pueden estar suministrados por programa.
b) Pueden introducirse a través de un dispositivo en línea.

c) Pueden ser producto de la operación interna de la computadora.

d) O lo más importante: pueden producirse y suministrarse desde el mismo sistema que se controla o del producto que se está - obteniendo.

1.1.2. EL PROGRAMA

El programa en su descripción más sencilla consiste en una lista de instrucciones que la máquina (computadora) debe obedecer para gobernar el sistema bajo su control. La extensión y complejidad de un programa depende de varios factores:

a) El sistema que controla o el producto que se debe obtener.

b) El lenguaje de programación utilizado; y

c) Las facilidades con que el sistema computador cuenta.

1.1.3. LA MEMORIA

La memoria es la parte del sistema en don-

de se almacenan los programas y los datos que la computadora utiliza para su operación y procesamiento y los datos que producidos por esa operación y procesamiento se deben almacenar para uso posterior.

1.1.4. LA COMPUTADORA

La computadora está compuesta de 5 elementos básicos esenciales (1).

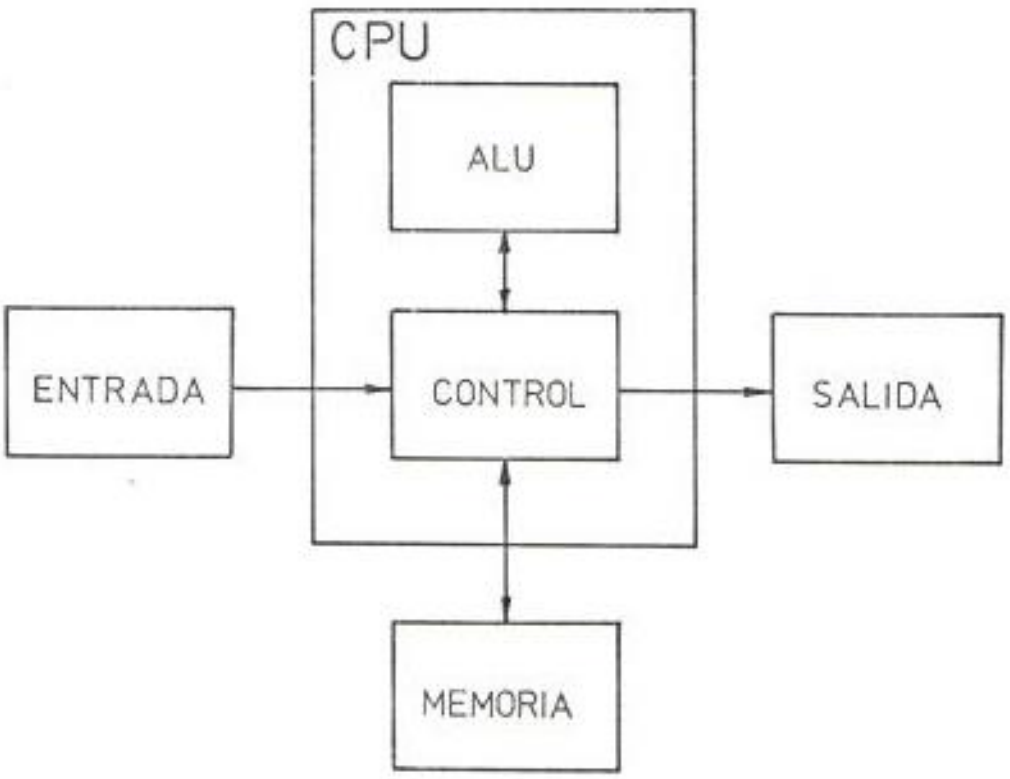


FIG. 1.2 ELEMENTOS DE UNA COMPUTADORA

dos de los cuales se agrupan con el nombre de unidad de procesamiento central CPU; los elementos de entrada y salida que conectan la computadora con el resto del sistema; y la memoria es el elemento de almacenamiento que ya está mencionado.

1.2. LA UNIDAD DE PROCESAMIENTO CENTRAL

Esta sección de la computadora es el centro y eje director del sistema, el "cerebro" cuya función - es ejecutar programas almacenados en la memoria - principal, buscando sus instrucciones, examinándo las, y ejecutándolas una después de otra(2). Está constituída por las dos partes principales mencionadas, mas registros adicionales:

a. La unidad de lógica aritmética ALU;
b. La unidad de control; y
c. Registros

En realidad una computadora no es sino una gran colección de registros arreglados de tal manera que ejecuten una función predeterminada en el instante preciso, y la CPU no escapa a esta re-

gla, sin embargo, independientes de los registros propios de la unidad ALU y de la unidad de control y de acuerdo con la filosofía del diseño de cada CPU encontramos otros registros que complementan o ayudan en el trabajo de la CPU.

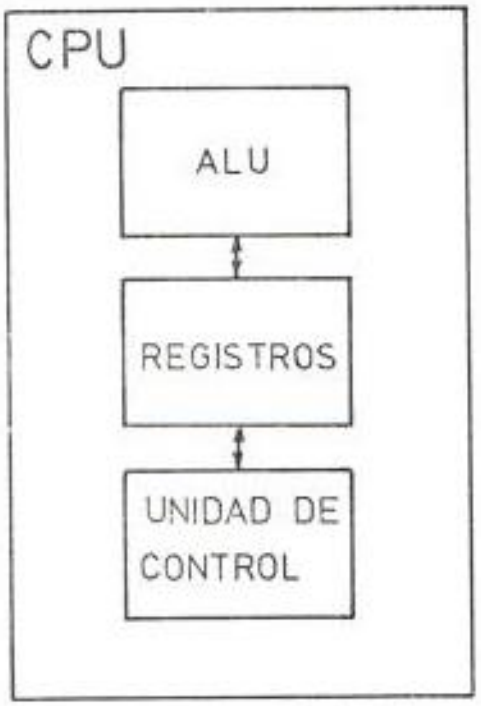


FIG.1.3 UNIDAD DE CONTROL DE PROCESAMIENTO CENTRAL

1.2.1. LA UNIDAD DE LOGICA ARITMETICA

La ALU a menudo está asociada con un solo registro llamado ACUMULADOR, pero la operación típica de esta unidad es sobre el contenido de pares de registros, uno de los cuales generalmente es el Acumulador que contiene uno de los operandos y que -

almacena el resultado de la operación efectuada. El tamaño del Acumulador es el tamaño de "palabra" del computador (1). Las operaciones que efectúa la unidad ALU son de varias clases:

a. Aritméticas: suma, resta, multiplicación, división, etc.

b. Lógicas: AND, OR, XOR, NOT, etc.

c. Desplazamiento.

1.2.2. LA UNIDAD DE CONTROL

La unidad de control es la sección del computador que enlaza todos los diversos registros juntos sobre el tiempo para ejecutar - algún conjunto requerido de tareas.

1.2.3. INSTRUCCIONES DE MAQUINA

Debido a que todas las posibles transferencias de registros que pueden tener lugar, están bajo el control de esta sección central del computador, es necesario alguna -

manera de especificar que clase de transferencia se efectuará, estas especificaciones se denominan instrucciones, y la lista de todas las posibles instrucciones que el computador puede aceptar es llamado el "CONJUNTO DE INSTRUCCIONES" o "INSTRUCTION SET"(1).

1.2.4. LOS REGISTROS

La unidad CPU también contiene una pequeña memoria de alta velocidad usada para almacenar resultados temporales y cierta información de control.

Esta memoria consiste de un número de registros, cada uno de los cuales tiene una cierta función. El registro mas importante es el contador de programa (PC), que señala la próxima instrucción a ser ejecutada. También hay un registro de instrucciones (IR), el cual retiene la instrucción que actualmente se está ejecutando. La mayoría de los computadores tienen también otros registros, algunos de los cuales están disponibles para almacenar resultados intermedios.

1.3. HARDWARE

Programas elaborados en lenguaje de máquina (instrucciones de máquina suministrados a esta mediante Switches), de un computador pueden ser ejecutados directamente por los circuitos electrónicos del computador, sin la necesidad de que intervengan interpretadores o traductores.

Estos circuitos electrónicos, junto con la memoria y dispositivos de entrada/salida forman lo que se denomina el "HARDWARE" del computador.

1.3.1. El Hardware consiste de objetos tangibles:

circuitos integrados, paneles de circuitos impresos, cables, fuentes de alimentación, memorias, lectoras, impresoras, terminales, etc. en lugar de ideas abstractas o instrucciones. Cómo estos componentes son construídos y cómo ellos trabajan pertenece al dominio de la ingeniería eléctrica y electrónica.

1.4. SOFTWARE

El software en contraste, consiste en algoritmos o instrucciones detalladas que dicen como hacer algo, y sus representaciones de computadora denominados programas.

1.4.1. PROGRAMAS

Los programas se pueden presentar en tarjetas perforadas, cintas magnéticas, película fotográfica, diskettes, y otros medios, pero la esencia del software es el conjunto de instrucciones que integran los programas y no el medio físico en los que ellos están - grabados.(2).

1.4.2. EQUIVALENCIA DE SOFTWARE y HARDWARE

El Hardware y el Software son lógicamente equivalentes y cualquier operación efectuada por el software puede ser construída directamente en el Hardware y a la inversa, - cualquier instrucción ejecutada por el Hardware también puede ser simulada por el software.

1.4.3. HARDWARE VS. SOFTWARE

En las primeras computadoras la diferencia entre software y hardware era muy clara y definida, ya que los circuitos del Hardware solo eran capaces de ejecutar muy pocas instrucciones simples y cada cosa mas tenía que ser explícitamente programada en el Hardware.

Con el advenimiento de la microprogramación y de computadores multinivel, la tendencia contraria se hace aparente y desaparecen las rígidas reglas de lo que debe estar en el Hardware y lo que debe estar en el software, de tal manera que en la actualidad el límite entre Hardware y Software es arbitrario y está cambiando constantemente, lo que ahora es software mañana es Hardware y viceversa.

También los límites entre los niveles de lenguaje son fluídos y un lenguaje de alto nivel se elabora o representa a través de un

lenguaje de bajo nivel, o un programa en lenguaje de bajo nivel puede ser traducido, interpretado o compilado mediante un lenguaje de alto nivel (2), ésta última - cualidad es la que precisamente vamos a utilizar en el trabajo motivo de esta obra y que se explicará en detalle mas adelante.

De manera que desde el punto de vista del programador, cómo se implementa actualmente una instrucción carece de importancia, excepto quizás por su velocidad y extensión.

Todo lo anterior nos ayudará a comprender como el trabajo de un computador puede efectuarse en otro computador o cómo un computador nos puede servir de intermedio para comunicarnos con otro computador que - tal es el caso del presente trabajo cuya descripción general se presentará en el próximo capítulo. Referencias (1) y (2).

Referencias: (1) Microcomputer Dessign
(2) Structured Computer organization

CAPITULO 2

INFORMACION NECESARIA

La ESPOL cuenta en la actualidad con un laboratorio de microprocesadores en el que se utilizan microcomputadoras basadas en el microprocesador 8080 de INTEL, y en un futuro inmediato tendremos también microcomputadoras basadas en el microprocesador 8085 de INTEL; mas adelante se ampliará el laboratorio con otros tipos de microprocesadores en tal forma que se lo pueda utilizar con fines de entrenamiento, estudio, e investigación y desarrollo; actividades todas de mucha importancia tecnológica para los profesionales técnicos modernos de cualquier rama, que pronto se verán envueltos en un mar de sistemas automáticos controlados a través de microprocesadores, y en particular para los técnicos que como nosotros pertenecientes a este muy importante y prestigioso campo de la electricidad y la electrónica, responsable y director de ese desarrollo, que tenemos contacto directo con estos diseños y dispositivos.

De manera que este laboratorio pronto será el más importante y sofisticado de los laboratorios de la ESPOL (y del país) por lo que me siento muy complacido de partici

par en su iniciación y con mucho placer mi trabajo se constituirá en un grano de arena que ayude en su desarrollo.

En lo que sigue, se describe brevemente la información que se utilizó para la ejecución del objetivo propuesto.

2.1. SISTEMA OPERATIVO HT/11

Como instrumento principal de trabajo, se utilizó el sistema HT/11, cuyo Sistema Operativo lo podemos dividir en dos partes:

a. Hardware
b. Software

2.1.1. EL HARDWARE

El Hardware del sistema operativo del HT/11 está constituído principalmente por la Unidad de Procesamiento Central asociada a la Memoria disponible de 16k bytes, y periféricos tales como: Terminal de Pantalla, Floppy Disk Drive, Impresor, etc., que se utilizó para la elaboración del ensamblador cruzado

8080/8085, se describirán en el siguiente capítulo.

2.1.2. EL SOFTWARE

En cuanto al Software del Sistema Operativo del HT/11 tenemos, en primer término el Monitor residente, al que agregamos el monitor - del teclado, para tener acceso y facilidad de utilización de otros programas de alto nivel que se usarán para la elaboración del Ensamblador cruzado y que al igual que el Hardware se utilizarán en el uso normal del Ensamblador.

El principal programa de alto nivel utilizado es el de Lenguaje "BASIC". Luego tenemos el "PIP", el "EDIT", el "FORMAT", que junto con los programas del sistema y el de la fecha: "DATE", son útiles para nuestro trabajo.

Información detallada de los elementos mencionados, se puede obtener directamente de los - manuales respectivos, sin embargo, se hará una brevísima descripción del uso de tres de

ellos.

2.1.2.1. FORMAT. Este programa sirve y se - utiliza para preparar las condiciones de utilización de los diskettes, operación que se denomina "FORMATEADO".

2.1.2.2. PIP. Una vez formateado, un Diskette, se puede usar para recibir programas que estén almacenados en otro disco o dispositivo, mediante el PIP que significa Intercambiador de programas entre periféricos.

Realmente, este programa es más útil para la manipulación de los archivos, como por ejemplo borrarlos, reacomodarlos, recuperarlos, etc.

Aunque más frecuentemente se lo usa para obtener el directorio actual de un Diskette. Siempre es útil tener un listado del directorio actualiza-

do de un disco, especialmente cuando se hacen operaciones de recuperación de información, que accidentalmente se puede haber dañado, o para tener conocimiento de los archivos existentes, que espacio ocupan, en que posición se encuentran y cuanto espacio hay disponible en el disco.

2.1.2.3. EDIT. El "EDIT", es un programa que el HT/11 dispone para crear o modificar programas, es útil en cuanto tiene mucha flexibilidad para el manejo de la información almacenada en los archivos y en ocasiones sus cualidades son aprovechables - para ahorrar tiempo y trabajo.

Se lo usó con poca frecuencia, ya que el BASIC dispone de su propio Editor, y se usará con menos frecuencia todavía, ya que se ha elaborado un "EDITOR" específicamente

para crear y manipular archivos escritos en lenguaje ensamblador del 8080/8085.

2.2. SISTEMA OPERATIVO DE MICROCOMPUTADOR (SDK/85)

Un microcomputador no es sino un computador que como CPU tiene un microprocesador, dispone de memorias de semiconductor y en general sus componentes se caracterizan por ser circuitos integrados.

Como antes, el Hardware de un microcomputador se describirá en el siguiente capítulo.

2.2.1. EL SOFTWARE

En un microcomputador el Software está constituído principalmente por un programa MONITOR almacenado en un ROM, que permite la interacción del operador con el computador a través de un teclado generalmente hexadecimal y un exhibidor (Display) de direcciones y datos.

2.2.2. POSIBILIDADES DE AMPLIACION Y MEJORA

La anterior descripción es el Sistema Operativo Básico que comunmente se encuentra en los microcomputadores, sin embargo muchos de ellos ya contemplan en su sistema otros periféricos, como el monitor del SDK/85 que tiene en su monitor una sección que permite, mediante un sencillo circuito adicional, conectarse a un terminal de pantalla y su teclado correspondiente. Existen sistemas basados en el 8080 y también en el Z80 o el 6800 que se pueden conectar con un interfase adecuado a un impresor de línea, etc.

2.2.3. APLICACIONES

Como hemos expresado, el sistema básico, puede ser ampliado y aun más, las posibilidades de usos y aplicaciones, específicas y generales de un microcomputador son ilimitadas, sólo se necesita un Sistema Operativo adecuado en cada caso.

2.3. PROGRAMADORES DE EPROM/PROM, INTERFASES

La clase de memoria que se usa en los sistemas microcomputadores, como se dijo, es de semiconductor, que en el caso de los EPROM/PROM y ROM, se conocen con el nombre de memorias no volátiles, porque no se pierde la información, cuando la alimentación es removida.

2.3.1. ROM.

El ROM, es una memoria no volátil de semiconductor, de acceso aleatorio que se ha programado en la fábrica y en la que sólo se puede efectuar la operación de Lectura, como su nombre lo indica: Memoria de lectura solamente (Read only memory) generalmente contiene el programa MONITOR del sistema.

2.3.2. PROM

La memoria PROM o Memoria Programable de lectura solamente (Programable read only memory) puede ser programada ex-fábrica, es

decir por el usuario para programas de aplicación fija específica, pero una vez efectuada la operación de programación no es posible cambiar absolutamente en nada la información almacenada, de manera que una sola equivocación puede malograr el circuito integrado completo.

2.3.3. EPROM

La memoria EPROM, a diferencia de las anteriores puede ser reprogramada después de efectuar un proceso de BORRADO, razón por la que se llama: Memoria Borrable, Programable de lectura solamente (erasable, programable read only memory); el proceso de borrado, generalmente consiste en exponer el circuito integrado de 10 a 20 minutos a la luz ultravioleta a través de la ventanilla dispuesta para el efecto.

2.3.4. PROGRAMADORES DE EPROM/PROM

Como nos hemos percatado, para muchos propósitos de aplicación definida de los microprocesadores o de los microcomputadores,

nos conviene tener almacenados programas o datos en forma permanente, es decir no volátil, para lo cual podemos proveernos de circuitos integrados de Memorias Programables, pero eso no es suficiente, necesitamos métodos y dispositivos adecuados para almacenar programas en esas memorias ya que debemos tomar en cuenta que además de conocer la dirección que se va a ocupar y el dato que se va a depositar, necesitamos un medio físico para efectuar tales operaciones, que disponga de Alimentación, temporización, señalización y conmutación adecuada.

Existen fabricantes y proveedores que los suministran con estas y otras facilidades, hay muchos programadores de PROM que se pueden adquirir, pero esta no es la idea, como lo veremos en el capítulo de PROYECTO (Capítulo 6), porque en un laboratorio no es importante lo que ya está hecho, sinó - lo que se puede hacer.

2.3.5. INTERFASES

Con el mismo criterio que se estableció en los párrafos anteriores debemos mencionar las posibilidades de interconectar el Programador de PROMS. a un microcomputador, a un terminal, o a otro dispositivo que pueda efectuar transferencia de datos, y esto necesariamente necesitará un interfase para cada caso, nos remitiremos al capítulo 6 nuevamente.

2.4. METODOS DE PROGRAMACION DE SISTEMAS

El método que se use para programar un sistema, depende del sistema y de las facilidades con que este sistema cuente.

Sin embargo, cualquiera que sea el sistema computador, hay varios métodos, denominados niveles que son comunes en su origen y estructura, aun cuando difieran en la forma.

2.4.1. NIVEL 1

El nivel inferior o nivel mas bajo (nivel 1)

es aquel en el que los programas son ejecutados directamente por los circuitos electrónicos, se utilizaba para programar los primitivos computadores.

2.4.2. NIVEL 2

El segundo nivel de programación que en la actualidad es el nivel 1, es el de lenguaje de instrucciones de máquina, que en realidad son interpretados por el microprograma que gobierna los circuitos electrónicos.

2.4.3. LENGUAJE ENSAMBLADOR

El método ensamblador consiste en usar nombres que identifican las instrucciones de máquina, que son fáciles de recordar y en consecuencia, de usar, este método ya necesita de un interpretador o traductor para convertir esos nombres o códigos, en las instrucciones correspondientes.

2.4.4. LENGUAJES DE ALTO NIVEL

Se pueden usar lenguajes que se denominan

de alto nivel, por obvios motivos, en sistemas que cuenten con lo que se conoce como: traductores, interpretadores o compiladores, y que a partir de sentencias que, generalmente en Inglés, especifican que operaciones efectuar, por lo que se comprenderá su utilidad y facilidad de aprendizaje y aplicación. Ejemplos de estos lenguajes son: BASIC, FORTRAN, PLI, APL, PASCAL, COBOL, RPG, etc.

2.5. METODOS DE TRANSFERENCIA DE PROGRAMAS

Se ha establecido ya, que sea para uso temporal o para uso permanente, necesitamos almacenar programas que determinen qué operaciones van a efectuar los computadores y dispositivos asociados.

También se ha definido que uno de nuestros propósitos es tener almacenados programas permanentemente en PROMS, y que necesitamos medios para efectuar ese almacenamiento. Y lo que es principal, que hay varios niveles de programación.

Asociado a esto último encontramos que si pensamos

en la programación de un PROM, necesitamos saber que métodos utilizar para efectuar la transferencia de los programas elaborados, y nos encontramos con que podemos utilizar tres métodos:

a. Manual
b. Semimanual
c. Semiautomático
d. Automático

2.5.1. METODO MANUAL

Solo se puede utilizar con el nivel mas bajo de programación, es decir, utilizar interruptores de encendido-apagado para seleccionar direcciones de memoria y depositar datos, en lenguaje binario o de Maquina.

2.5.2. METODO SEMIMANUAL

Para el método semimanual, utilizamos una variante del anterior, es decir que seguimos utilizando interruptores de encendido-apagado para la transferencia de un programa, tanto en lo que se refiere a dirección

como al dato a depositarse en esa posición, también seguimos utilizando el lenguaje de Máquina, pero ahora en lugar de binario, lo podemos expresar en hexadecimal, para lo - que en realidad utilizamos un traductor que convierte el código hexadecimal en binario.

2.5.3. METODO SEMIAUTOMATICO

Lo que he denominado un método SEMIAUTOMATICO es el que utilizando el Ensamblador cruzado 8080/8085 que para el efecto se ha elaborado, permite escribir programas en un lenguaje mas alto como el Ensamblador, y luego de la TRADUCCION AUTOMATICA A EXPRESION HEXADECIMAL, utilizar el método Semimanual, ver capítulo 6.

2.5.4. METODO AUTOMATICO

Podemos diseñar, como se expresará en el capítulo 6, sistemas que eliminen la operación manual de tal forma que AUTOMATICAMENTE y DIRECTAMENTE se efectue la transferencia de los programas escritos en lenguaje Ensamblador y compilados por el En-

samblador cruzado 8080/8085 al programador, a través de interfases adecuados.

2.6. LENGUAJE DE MAQUINA 8080/8085 INSTRUCCIONES

Anteriormente nos hemos referido a lo que se denomina lenguaje de Máquina y a las Instrucciones, pero de carácter general. Ahora se va a especificar el material con el que se va a trabajar, y ese es, el lenguaje que específicamente utilizan los microprocesadores 8008-8080-8085.

2.6.1. El primero de los microprocesadores mencionados, es antecesor de los otros dos y fue diseñado para que trabaje con 48 Instrucciones.

2.6.2. Luego, el 8080 mejorando las características del primero e incluyendo algunas funciones en el mismo circuito integrado y para - compatibilidad con su software, usa las mismas 48 Instrucciones de Máquina a las que se agregan 30 Instrucciones que lo hacen más versátil y útil.

2.6.3. Mas tarde, un nuevo diseño mejora las características del 8080 e incluye otras funciones lógicas en un sólo circuito integrado y así mismo para que sea compatible con el software de su antecesor utiliza sus 78 Instrucciones de Máquina a las que solamente se agregan dos: SIM y RIM, con las que el microprocesador cumple múltiples funciones de máscaras de interrupción, lectura y salida de datos, ver "MANUAL DE PROGRAMACION DE LENGUAJE ENSAMBLADOR 8080/8085".

2.6.4. EL PROCESAMIENTO DE LAS INSTRUCCIONES

De modo que nuestra materia prima es precisamente ese conjunto de 80 Instrucciones de Máquina del que se habló previamente, pero que están expresadas en Código MNEMONICO o Ensamblador y que se usan para elaborar programas en dicho lenguaje, que luego deben traducirse a lenguaje de Máquina en Código objeto y es justamente allí donde interviene el Ensamblador Cruzado para efectuar automáticamente la traducción de

aquel programa escrito en lenguaje Ensamblador y entregar su código objeto, listo para usarse en el microcomputador o con un programador de PROMS.

CAPITULO 3
EQUIPO NECESARIO

Este capítulo trata específicamente de los equipos y materiales físicos que se utilizaron y se van a utilizar en el trabajo del ENSAMBLADOR CRUZADO 8080/8085, desde luego su tratamiento es relativamente breve y se sugiere al lector que para cualquier ampliación, consulte los manuales correspondientes. Su conjunto se visualiza en la figura 3.1.

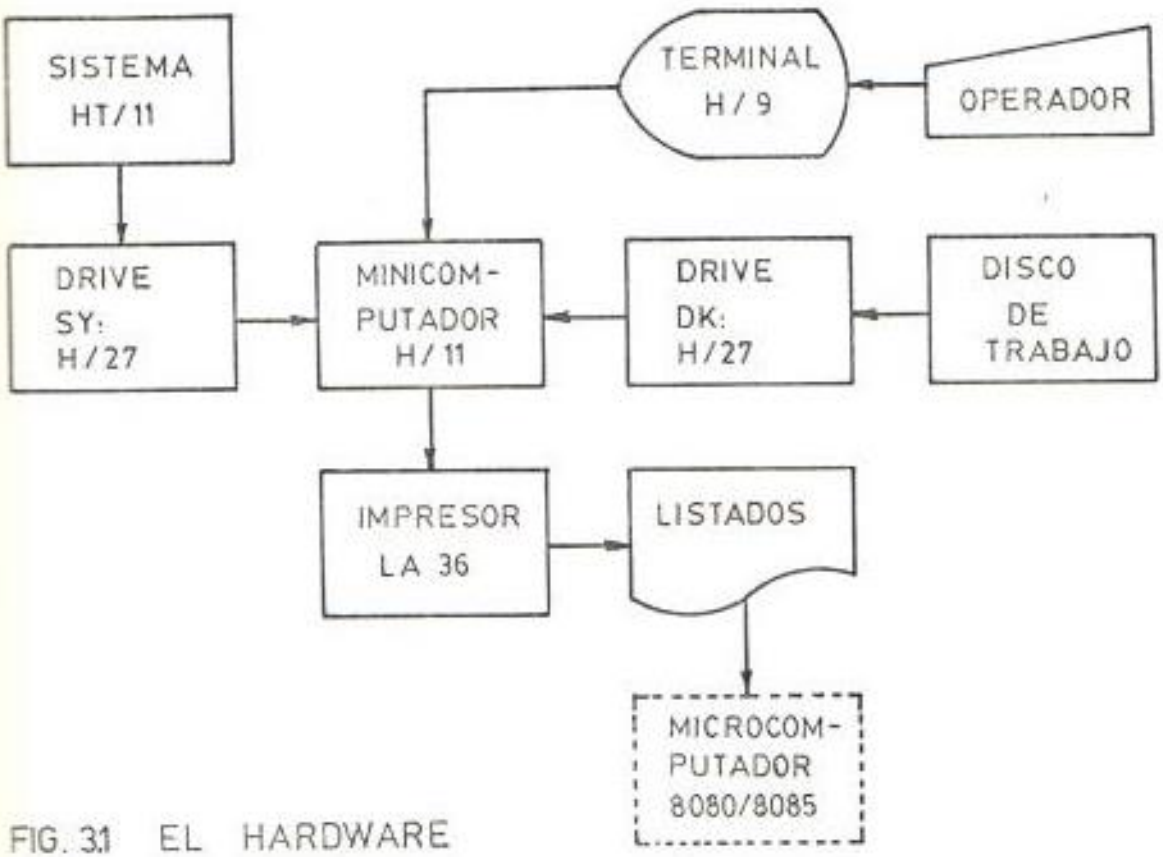


FIG. 3.1 EL HARDWARE

3.1. MINICOMPUTADOR H/11

Tal como vemos en el gráfico precedente, el Minicomputador H/11 es el instrumento central del sistema físico de trabajo, vemos que a través de el pasa toda la información y es el dispositivo que procesa esa información.

Se denomina minicomputador debido a su corta longitud de palabra (16 bits), pero es ampliamente usado en aplicaciones tales como comunicaciones, - control de procesos industriales, monitoreo y colección de datos en experimentos científicos, educación, etc.

3.1.1. ESTRUCTURA DEL H/11

El minicomputador consiste de una CPU, una memoria principal expandible a 32 K bytes asociada, e interfases de entrada/salida serie y paralelo mediante las cuales se incorporan a él los restantes periféricos con los que se interconecta para recibir, procesar, y entregar información.

3.2. MICROMPUTADOR BASADO EN MICROPROCESADOR 8080 u 8085

El objetivo final es la aplicación del Ensamblador Cruzado, en el uso de programas elaborados para ser corridos en microcomputadores basados sea en el microprocesador 8080 o en el microprocesador 8085. Lo tenemos en el gráfico precedente encerrado con líneas de segmentos.

3.2.1. ESTRUCTURA BASICA DE MICROCOMPUTADORES

Los microcomputadores tienen exactamente todos los elementos básicos de una computadora moderna, su principal característica, que los diferencia es que están diseñados y construídos con el uso y ventajas de la MICROELECTRONICA.

Del Software ya hablamos en el capítulo 2.

3.2.2. EL HARDWARE BASICO DE MICROMPUTADORES

El cuanto al Hardware, todos los microcomputadores, con ligeras variantes, tienen la misma estructura básica y sus componentes -

principales son:

a. El microprocesador
b. Reloj
c. Lógica de control
d. Bus alimentadores (drivers)
e. Decodificador de memoria
f. Memoria RAM
g. Memoria ROM conteniendo el monitor
h. Decodificador de entrada/salida
i. Separadores (Buffer)
j. Puertas de entrada/salida
k. Controlador de teclado/display
l. Teclado
m. Mostrador (displays)
n. Acceso para la alimentación de energía; y
o. Espacios para expansión y experimentación.

3.2.3. EL MICROCOMPUTADOR EN EL LABORATORIO

Los elementos de Hardware y software mencionados, son suficientes para efectuar experimentos sencillos y cortos propios de laboratorio, pero no lo son para investigación y desarrollo en donde no estamos tan interesa-

dos en las reacciones elementales del micro computador a los estímulos externos, como - en la utilidad de estas reacciones en aplicaciones prácticas, científicas, industriales, etc.

Ya en el campo de la investigación y desarrollo necesitamos otra clase de ayuda, que nos permita obviar las dificultades de lo elemental, entre las que en primer plano está el nivel de lenguaje utilizado, luego la escasa capacidad de memoria y la ausencia - de relaciones con dispositivos externos.

3.2.4. EL ENSAMBLADOR CRUZADO Y EL MICROCOMPUTADOR

En esta obra, se trata principalmente de solucionar el problema del lenguaje y eso es lo que hace el Ensamblador cruzado, al que ya nos hemos referido.

3.2.5. PROYECTOS

Mas adelante, capítulo 6, y a manera de sugerencia se elaborará un proyecto de ampliación de capacidad de memoria y relaciones - con dispositivos externos.

3.3. TERMINAL

El Terminal de Pantalla H9, puede mostrar información que viene desde un computador digital o información que un operador introduce desde un teclado. Su pantalla de 12" de dimensión diagonal, puede mostrar 960 caracteres a la vez distribuídos en 12 filas de 80 caracteres cada una.

3.3.1. EL TECLADO DEL TERMINAL

Su teclado con 67 teclas ASCII permite componer y editar directamente en pantalla, comunicarse con un computador digital en lenguajes de bajo o alto nivel y otras características importantes que son de mucha utilidad para el trabajo propuesto:

a. Creación del Ensamblador Cruzado 8080/8085 en lenguaje BASIC.
b. Creación del EDITOR para archivar y manipular programas elaborados en lenguaje - Ensamblador 8080/8085; y
c. Uso del Ensamblador cruzado para obtener

el programa objeto a partir de programa fuente en lenguaje Ensamblador 8080/8085.

3.4. IMPRESOR

El Impresor LA36 en nuestro sistema, lo utilizamos para obtener toda clase de listados impresos:

a. Programas;
b. Datos; y
c. En nuestra obra, para obtener el listado del programa objeto que entrega el computador 8080/8085 y que luego se utilizará en el microcomputador.

3.5. FLOPPY DISK DRIVE

El Dual-drive Floppy H27, es un importante medio de almacenamiento de programas y datos con una capacidad total de 512K bytes en los dos diskettes, lo que nos da suficiente espacio de almacenamiento para la mayoría de las aplicaciones de propósito general. El microcontrolador basado en el microprocesor Z80 nos permite un tiempo de acceso promedio rápido de menos de 6 ms. con lo que vir-

tualmente podemos llegar inmediatamente a todos los archivos y programas.

3.5.1. DRIVE Ø

El Drive Ø que en nuestro sistema se denomina SY: es el que aloja el Diskette llamado - del SISTEMA, porque es el que contiene el - Software del Sistema Operativo del H/11. En este mismo Diskette y ya como parte del sistema operativo del Ensamblador cruzado que - serán utilizados a través del BASIC estan los programas: "CMP851 BAS", "CMP852.BAS" y "EDITOR.BAS". Así mismo los siguientes archivos de datos: "INSTRS.DAT" y "CLINST.DAT". Que finalmente constituyen el nuevo sistema para trabajar con el Ensamblador Cruzado 8080/8085.

3.5.2. DRIVE 1

El Drive 1 que en nuestro sistema se denomina DK: aloja el Diskette de trabajo del usuario (Scratch Disc) en el cual se van a archivar todos los programas elaborados en lenguaje Ensamblador 8080/8085, a través del Editor,

y un pequeño programa transitorio llamado "LINEA.BAS" que se utiliza para editar datos en línea cuando se está haciendo uso del "EDITOR.BAS" y del "CMP851.BAS", como se describirá posteriormente.

3.6. DISKETTES

El Diskette o Floppy Disk es una pieza circular de plástico flexible MYLAR cubierto en una o en ambas superficies con óxido magnético y encapsulado en una envoltura plástica. Un corte en la envoltura permite el acceso de la cabeza a la superficie del disco. La envoltura con el disco es insertado en un Drive como una unidad y el disco rota dentro de la envoltura.

3.6.1. DISKETTE IBM

El Diskette IBM tiene un solo hueco índice localizado fuera del centro para identificar el principio de una pista o carril magnético medios electrónicos se utilizan para dividir cada pista en 26 sectores de 128 bytes cada uno, hay 77 pistas de las que la

primera se reserva el sistema para el direc
torio con lo que quedan disponibles 480 blo
ques de 4 sectores cada uno en 76 pistas,
para almacenamiento de archivos y programas
en una capacidad formateada de 245.760 bytes
(1'966.080 bits).

CAPITULO 4

ELABORACION DE LOS PROGRAMAS

4.0. LENGUAJE DE PROGRAMACION BASIC

Aquí no se va a describir en que consiste el lenguaje de programación BASIC, porque el usuario del Ensamblador Cruzado no lo va a utilizar sino indirectamente, y no necesitará conocer este lenguaje.

Se describirá más bien de que modo se han aprovechado las facilidades que el sistema HT/11 provee en su lenguaje de programación BASIC, en la elaboración de los programas de COMPILACION Y EDICION.

Para cualquier información respecto del BASIC el lector puede recurrir al "Manual del Usuario BASIC del HT/11".

4.0.1. FUNCIONES OPCIONALES

El Basic del HT/11 dispone de una serie de funciones opcionales, la mayoría de las cuales son muy útiles para el trabajo de los programas que se han elaborado en BASIC y que son:

a. Ensamblador Cruzado 8080/8085 I Parte

b. Ensamblador Cruzado 8080/8085 II Parte

c. Editor

d. Referencia cruzada

Se utilizan las siguientes funciones:

TABLA I. Funciones opcionales BASIC a usarse.

1	INT	8	POS
2	TAB	9	CHR$
3	BIN	10	DAT$
4	OCT	11	SEG$
5	VAL	12	STR$
6	LEN	13	TRM$
7	ASC		

Las funciones que no se utilizan y que a fin de obtener mayor espacio de memoria se pueden excluír son:

TABLA II. Funciones opcionales BASIC que se pueden eliminar

1	RND
2	ABS
3	SGN

4.0.2. VARIABLES

Se utilizan variables numéricas y de caracteres, con subíndices y sin subíndices.

Variables numéricas sencillas, se usan con fines de conteo, como el contador de localización o el contador del número de errores.

Variables de caracteres sencillos, se usan para la identificación de caracteres o de símbolos, o para imprimir mensajes de error.

Variables con subíndice, numéricas o de caracteres, se utilizan para establecer tablas de símbolos o de datos.

4.0.3. OPERACIONES

A pesar de que casi todo el trabajo de compilación o de edición se lo efectúa con caracteres, se usan las funciones aritméticas con variables numéricas, relacionales con ambas clases de variables, y de concatenación con variables de caracteres.

4.0.4. OPERACIONES LOGICAS

Esta clase de operaciones no están contempladas en el BASIC del HT/11 de manera que se elaboraron adecuados algoritmos con los que se efectúan operaciones lógicas tales como: NOT, AND, OR, XOR, de desplazamiento, o de módulo, etc. que son necesarios para usar el lenguaje Ensamblador 8080/8085.

4.0.5. OPERACIONES EN MODO INMEDIATO

Se ha usado la facilidad de operar en modo inmediato para la creación de programas, archivos y con propósitos de depuración de los mismos.

4.0.6. CREACION Y ELABORACION DE ARCHIVOS

Se pueden crear dos clases de archivos: secuenciales y virtuales.

Cada uno tiene ventajas y desventajas.

4.0.7. ARCHIVOS SECUENCIALES

Son aquellos que aceptan y entregan información en orden estrictamente secuencial, cada unidad

no se la puede crear ni utilizar por partes, sino en orden secuencial.

Tiene la ventaja de que el espacio de almacenamiento que utiliza, es el estrictamente necesario.

Se usa esta clase de archivo en el Sistema Ensamblador Cruzado para almacenamiento de datos y como se verá mas adelante como un medio muy útil de editar en línea al tiempo de corrida, segmentos de programa que definen las condiciones de trabajo del sistema. Compilador o Editor.

4.0.8. ARCHIVOS VIRTUALES

Son aquellos que permiten el acceso aleatorio a los datos archivados; eso no es exactamente cierto al momento de creación del archivo, pero el programa "EDITOR" elaborado para crear los archivos utiliza un algoritmo especialmente diseñado para obtene el efecto de crear un archivo virtual comenzando en cualquier punto y dejando líneas vacías - así mismo en cualquier punto del programa sin

que el operador deba tener cuidados especiales.

Por lo demás, modificaciones, listados o usos, pueden hacerse de cualquier línea de un archivo previamente creado, sin cuidado, desde luego a través del uso del programa "EDITOR" y de esa manera podemos afirmar que actúa como una memoria de acceso aleatorio.

Este método tiene dos desventajas principales:

a. El cuidado que se debe tener para crear y usar el archivo, ya que se deben conocer - con exactitud las características propias de un archivo virtual cualquiera, en particular, para usarlo sin riesgo de obtener resultados extraños e incomprensibles; y

b. El espacio de almacenamiento desperdiciado, en razón de que todas las líneas de un archivo de estos, tienen igual longitud, se use o no, el espacio reservado.

Esta clase de archivo se utiliza para la creación de archivos de programas en lenguaje EN-

samblador 8080/8085; fundamentalmente porque se harán frecuentes modificaciones de secciones o de líneas, con propósitos de mejoramiento, ampliación o depuración, y entonces sólo será necesario llegar y operar en esa determinada sección, lo que no es posible en los archivos secuenciales en los que para cualquier modificación es necesario crear nuevamente el archivo completo con las molestias consecuentes de ese proceso.

4.1. LA ELABORACION DE LOS PROGRAMAS DE COMPILACION Y EDITOR

Estos programas que hacen nuestro "NUEVO SISTEMA" están elaborados en Lenguaje BASIC del HT/11, fueron creados, editados y depurados en el modo inmediato que es el propio "Editor" de este BASIC, haciendo uso adecuado de las particularidades descritas en este capítulo, las proposiciones normales y características del BASIC.

En vista de la extensión del programa de COMPILACION, éste tuvo que ser dividido en dos partes llamadas "CMP851" y "CMP852" respectivamente que se editan en

línea al tiempo de corrida usando la proposición "OVERLAY" que llama al "CMP852", eliminando al mismo tiempo toda la parte del "CMP851" que ya no es útil.

La muy útil proposición "OVERLAY" también ha servido en los programas de COMPILACION y EDITOR para insertar en línea al tiempo de corrida, líneas con especificaciones que no pueden definirse con variables como por ejemplo, las proposiciones de dimensión DIM, o el dimensionamiento de las proposiciones OPEN de abrir archivos, virtuales o secuenciales. Con lo que hemos obtenido un sistema de programación dinámico.

El archivo autocreado por el COMPILADOR y el EDITOR se llama "LINEA.BAS", y es el que se elabora al tiempo de corrida insertando los datos que luego se autoeditan en el programa que lo usa.

Las restantes características y el uso o manejo de los programas: "CMP851", "CMP852" y "EDITOR", junto con las precauciones e interpretación de errores se especifican claramente en el "ENSAMBLADOR CRUZADO, MANUAL DE OPERACION", que suministra aparte, en el Apéndice D.

4.1.1. PROGRAMA I: COMPILADOR CRUZADO EN LENGUAJE BASIC PARA ENSAMBLADOR 8080/8085 (PRIMERA PARTE)

Debido a la dificultad que representaba elaborar programas en el lenguaje de máquina o binario, alguien ideó representar cada operación que el computador podía ejecutar, con un nombre que fuera fácil de recordarlo y asociarlo con la operación; y se dió origen al llamado código de operación y que por su facilidad de recordarlo se denominó MNEMONICO, de esa manera se hizo más fácil la tarea de elaborar y comprender los programas que se escribían, y además había menos propensión a equivocarse, claro que para poder utilizar ese programa se necesita tener la tabla que en binario representa la propia instrucción de máquina, esa operación todavía había que hacerla manualmente hasta que se pudo contar con traductores especiales que tomando el código de operación, entregue a la máquina la instrucción propia.

Ese es el proceso denominado Ensamblador y el código por ello se llama Ensamblador. Cada máquina y en nuestro caso, cada microprocesador

tiene su propio conjunto de códigos de operación o "INSTRUCTION SET" pero en todos los casos el proceso es el mismo, y ahora, en lugar de trabajar con grupos de dígitos binarios que son difíciles de leer y entender, tenemos una secuencia de operaciones fáciles de leer y que van a ser ejecutadas por la máquina.

Los computadores no trabajan sólo con instrucciones, también necesitan datos, generalmente direcciones usualmente dadas como números, aunque se pueden establecer símbolos que representen esos números, a los que se hace referencia por el nombre, de esta manera, el simple ensamblador se extiende para permitir al programador introducir sus propios símbolos.

El ensamblador entonces, maneja dos clases de símbolos: los MNEMONICOS o códigos de operación que son símbolos predefinidos en tablas estáticas que representan las instrucciones de máquina; y las direcciones simbólicas que son nombres arbitrarios que se establecen en tablas dinámicas en las que el programador representa direcciones de memoria.

4.1.1.1. Pseudo operadores

Los pseudo operadores o directivas ensambladoras son facilidades que un ensamblador ofrece al programador, que no representan instrucciones a ser generadas en el programa objeto.

Estas pueden ser entendidas como órdenes que el programador emite al ensamblador sea para generar datos, controlar direcciones, manipular caracteres, reservar áreas de memoria, etc.

En base a lo anterior clasificamos nuestro material de trabajo en tres grupos de elementos:

a. Códigos de operación (MNEMONICOS);
b. Directivas Ensambladoras; y
c. Símbolos creados

4.1.1.2. Códigos de Operación (INSTRUCCION SET)

El Ensamblador 8080/8085 consiste de 80 instrucciones de las cuales dos: SIM y RIM sólo pertenecen al 8085, los restantes 78 son comunes al 8080 y al 8085, además 48 de estas son las instrucciones del microprocesador 8008.

La compilación como se describirá mas adelante, se efectúa en dos pasos, con esa finalidad el conjunto de instrucciones se ha clasificado de dos maneras, una para cada paso del ensamblaje.

Para el primer paso se ha clasificado en tres grupos:

a. Instrucciones de 1 byte
b. Instrucciones de 2 bytes; y
c. Instrucciones de 3 bytes.

4.1.1.3. Tablas de símbolos

En el primer paso se definen las tablas de símbolos, que se utilizarán en el segundo paso y son:

a. De identificadores LABEL
b. De nombres EQU; y
c. De nombres SET

4.1.1.4. Directivas Ensambladoras

Siete son las Directivas Ensambladoras que se procesan en el Compilador y estas son:

Tabla III. Directivas Ensambladoras

a. NAME
b. DB
c. DW
d. ORG
e. DS
f. STKLN
g. END

4.1.1.5. Mensajes

En la primera parte se emiten dos clases de mensajes, los de solicitud de datos, en la pantalla, y de error en el impresor.

El listado completo de esta primera parte se encuentra en el Apéndice "A".

4.1.2. PROGRAMA II: COMPILADOR CRUZADO (Segunda parte)

Para la segunda parte se explora nuevamente el programa - fuente desde el comienzo, pero esta vez las Instrucciones se han clasificado en 7 grupos:

a. 1 Instrucción de dos registros sencillos y
   1 Instrucción de un registro sencillo y un byte de datos.

b. 27 instrucciones sin Operando

c. 10 Instrucciones de registro sencillo;

d. 8 Instrucciones de Registro doble

e. 10 instrucciones de 1 byte de Datos

f. 33 instrucciones de 2 bytes de datos; y

g. 1 instrucción RST

4.1.2.1. Tablas de símbolos

Se exploran las tablas de símbolos cuando se busca un valor referenciado con un símbolo, que puede ser una dirección o un dato.

4.1.2.2. Los Operandos

Para procesar los operandos es que se estableció la anterior clasificación de instrucciones, esta es la parte que caracteriza al programa y define su codificación. Es la parte mas difícil de tratar y para ello se utilizan como se explicará mas adelante, muchas subrutinas.

4.1.2.3. Salida Impresa

Se establecen mecanismos para que salgan al impresor las siguientes informaciones:

a. Programa fuente

b. Programa objeto

c. Errores

d. Cantidad de errores

4.1.3. PROGRAMA III: REFERENCIA CRUZADA, INCLUIDO EN LA SEGUNDA PARTE DEL COMPILADOR CRUZADO

Como se expresa, este programa está incluído en la segunda parte del Ensamblador Cruzado y solo se corre condicionado a la ausencia de errores en la compilación, caso contrario no se usa, si trabaja emite listados de símbolos con las líneas de referencia respectivas.

El listado completo de los programas II y III se encuentra en el Apéndice "B".

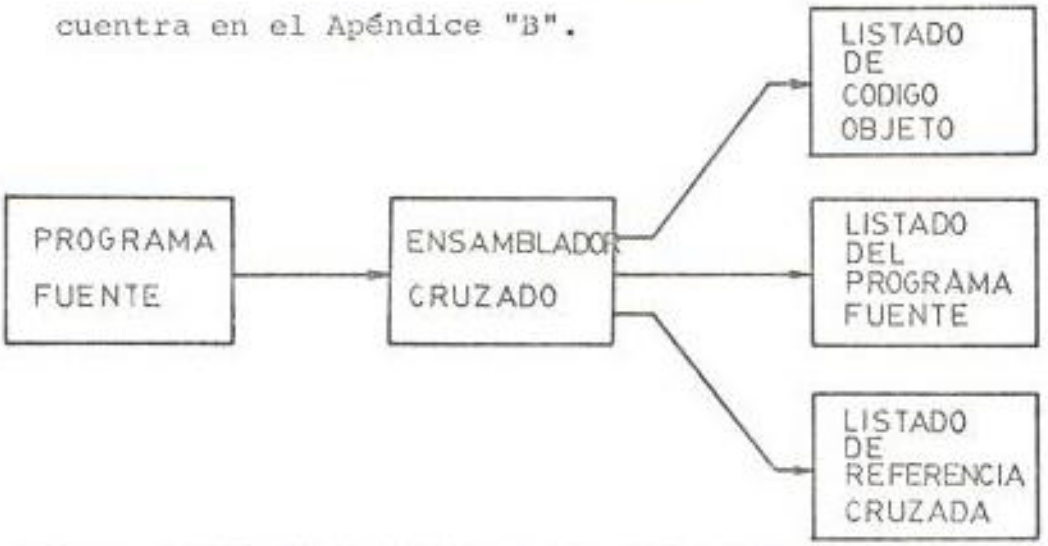


FIG. 4.1 ENTRADA Y SALIDAS DEL ENSAMBLADOR CRUZADO

4.2. PROGRAMA IV: EDITOR

Para cumplir con las necesidades de el programa ENSAMBLADOR CRUZADO, se ha elaborado un programa EDITOR que permite crear, modificar o simplemente emitir listados de programas que se han escrito en lenguaje Ensamblador 8080/8085 usando el método de archivos virtuales.

Listado completo de este programa se encuentra en el Apéndice "C".

4.3. LENGUAJE CONVERSACIONAL Y MENSAJES DE ERROR EN TODOS LOS PROGRAMAS

En la elaboración de los programas Ensamblador Cruzado y EDITOR se ha utilizado un método de "PREGUNTAS / RESPUESTAS" que a través del terminal de pantalla permiten la comunicación directa entre el operador o usuario y el programa, para la definición de datos o condiciones de trabajo del sistema. El idioma utilizado es el Castellano, y las preguntas, sencillas de comprender.

Así mismo y determinado por las condiciones de fallo detectado por los programas, se han establecido méto-

dos para detectar fallas y emitir de inmediato el mensaje de error correspondiente.

CAPITULO 5
ELABORACION DE INFORMACION RELATIVA

5.1. GENERALIDADES

Como ya hemos expresado, la entrada a un Ensamblador, es un programa escrito en lenguaje ensamblador, la salida es un programa objeto mas información que sirve al cargador para prepararlo para la ejecución en la máquina.

Además, hemos expresado que se necesitan tablas y datos para que el Programa ensamblador pueda cumplir su función.

FIG. 5.1. EL SISTEMA

De esa manera: nuestros sistema central O, necesita y solicita los datos de 1 y 2 para producir 3 (Gráfico 5.1).

5.1.1. DESCRIPCION DEL SISTEMA

5.1.1.1. Datos de programa Fuente

Son líneas de lenguaje ensamblador 8080/8085 que se encuentran almacenadas en el Diskette de trabajo del usuario. Están constituídas por cuatro campos claramente diferenciados:

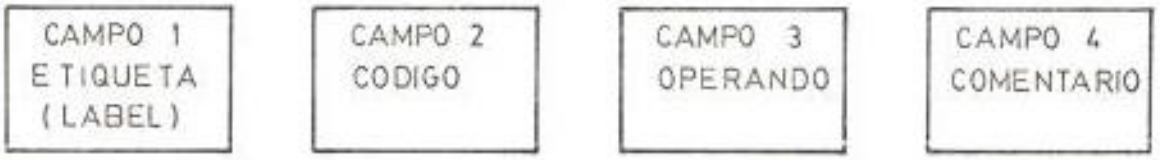


FIG. 5.2 CAMPOS DE LINEA DE PROGRAMA FUENTE

Campo 1. El campo 1 debe usarse para un símbolo llamado nombre sólo en el caso en el que el campo 2 tenga un "EQU" o un "SET" los "MACROS" no se procesan en este Ensamblador Cruzado.

El label es opcional y es un símbolo que debe estar terminado por ":", se usa como dirección simbólica con cualquier código de operación y con "NAME", "DB", "DW. "ORG", "DS", "STKLN", o "END".

Campo 2. En el campo 2, se debe encontrar un Código de operación o una Directiva, limitados por espacios en blanco, cualquier otro contenido es error.

Campo 3. No existe para los códigos que en el capítulo anterior se mencionó, están agrupados con el nombre de "Instrucciones sin Operando" y que son 27. Para las restantes instrucciones y las Directivas mencionadas es obligatorio, excepto para la Directiva "END" en la que es opcional.

El contenido de este campo es el de más difícil procesamiento porque puede presentarse en una amplia variedad de formas, representaciones o expresiones y definir de 0 a 2 Bytes extras de memoria según el caso, excepto con la directiva "DB", con la que puede tener hasta 127 bytes extras de memoria.

Campo 4. Este es un campo completamente opcional, debe comenzar con ";" y el Ensamblador Cruzado no lo toma en cuenta, mas que para eliminarlo del proceso.

5.1.1.2. Tablas de datos

El Ensamblador dispone de varias tablas de símbolos o datos que se han elaborado para procesar el campo 2, dos de estas tablas, una de símbolos y otra numérica, se han archivado en el Diskette del sistema desde donde el programa principal los llama cada vez que las necesita, son archivos secuenciales. Otras tres tablas se generan antes del primer paso al tiempo de Ensamblaje, en memoria.

5.1.1.3. Tablas de Símbolos del Usuario

Tres son las tablas de símbolos que se generan en memoria en el primer paso, que luego se utilizan en el segundo paso y mas tarde para la Referencia Cruzada. Estas tablas son:

a. Tabla de símbolos LABEL
b. Tabla de símbolos NOMBRE - EQU
c. Tabla de símbolos NOMBRE - SET

Otros símbolos: Además, se genera el nombre del

programa si lo tiene, caso contrario el Ensamblador asume el nombre "MODULO". Como otros símbolos podemos considerar los mensajes de error y los encabezamientos de los listados, etc.

5.1.1.4. Estructura básica del ensamblador cruzado-ϕ-

Básicamente el Programa Ensablador está compuesto de cinco secciones principales que se concatenan y complementan entre sí para procesar los datos anteriormente mencionados y dar como resultado el listado impreso de los programas fuente y objeto del Emsablador 8080/8085 y si no hay errores, la Referencia Cruzada.

Estas secciones son:

I. Subrutinas de análisis y procesamiento de datos y caracteres. Emiten señalización de errores.

II. Definición de Datos e informaciones y preparación de condiciones de trabajo del Ensamblador Cruzado 8080/8085.

III. Primer paso de compilación.- Elaboración de tablas de símbolos y asignación de valores o direcciones según el caso, a cada símbolo. Además, inicializa las tablas de Referencia Cruzada.

IV. Segundo paso de compilación.- Procesamiento de datos, asignación de localización, codificación, elabora las tablas de Referencia Cruzada, emite mensajes de error, los contabiliza, Emite líneas de programa ensamblador en código objeto hexadecimal y en lenguaje fuente.

V. Si no hay errores, emite tablas de - símbolos del usuario con la referencia de líneas en las que aparecen dichos símbolos.

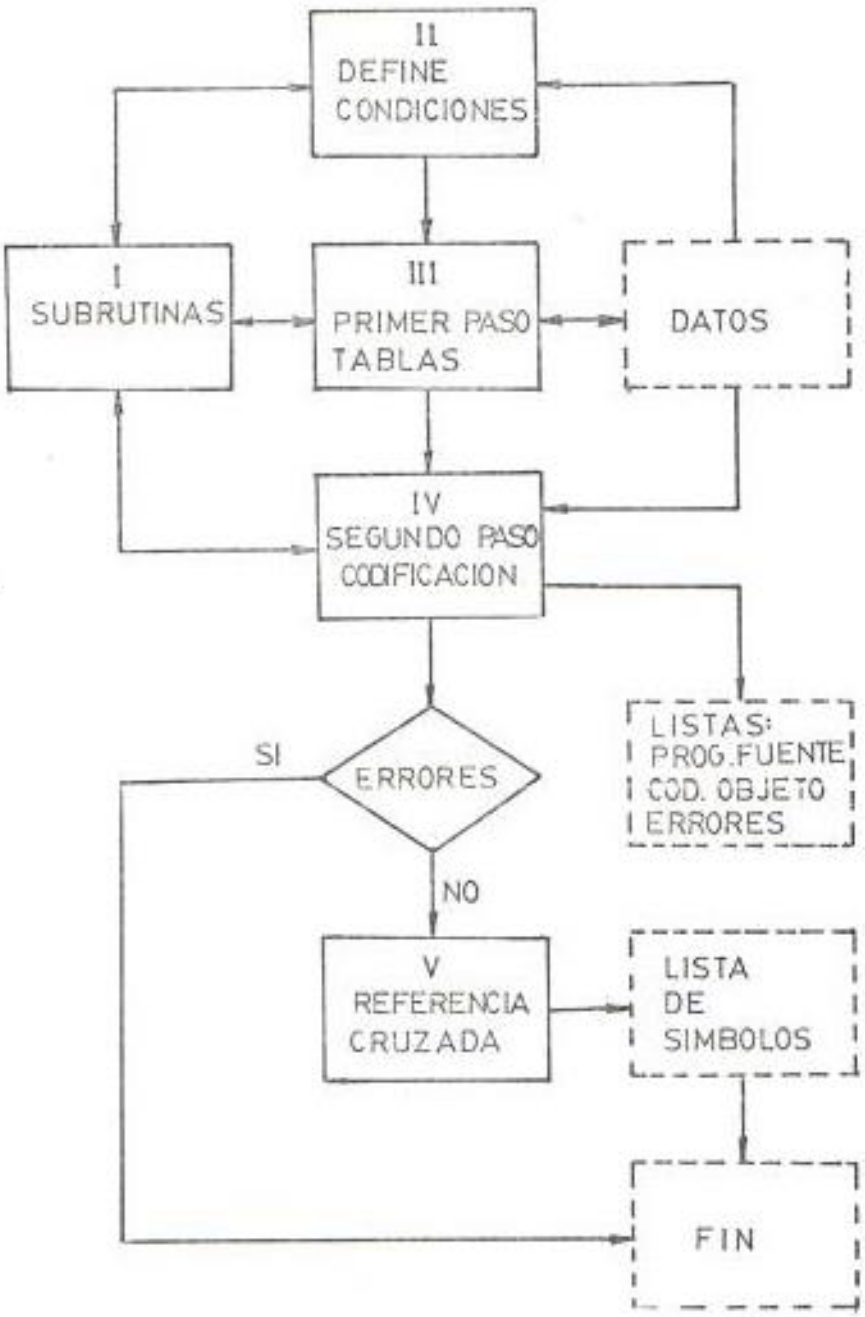


FIG. 5.3 ENSAMBLADOR CRUZADO 80/85

En el diagrama de bloques del gráfico 5.3 encontramos todas las secciones de que está compuesto el Ensamblador cruzado 8080/8085 y que están distribuídas así:

5.1.1.5. Programa "CMP851"

Este es el programa que podemos llamar PRINCIPAL y está inicialmente conformado como vemos en el gráfico 5.4.

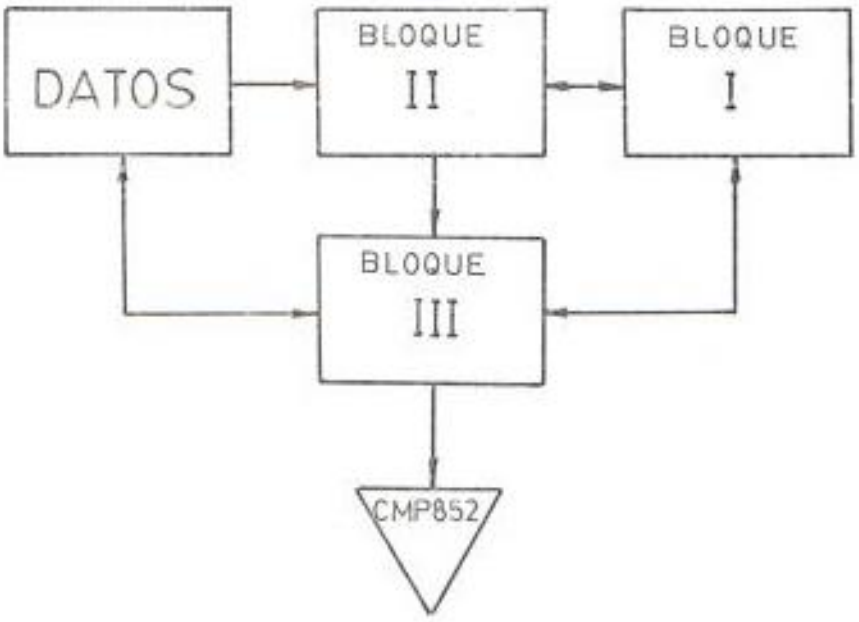


FIG. 5.4 ESTRUCTURA INICIAL - PRIMER PASO - CMP851

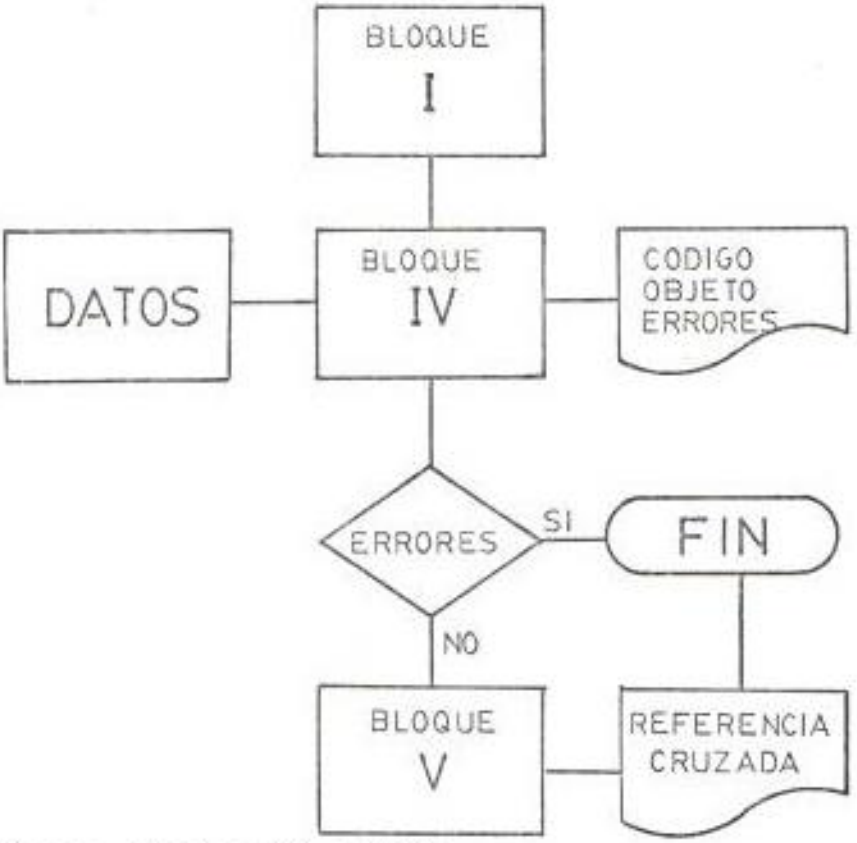


FIG. 55 SEGUNDO PASO

Luego del OVERLAY observamos en el gráfico 5.5 que los bloques IV y V han pasado a reemplazar a los bloques II y III que ya no son necesarios.

Se puede visualizar ahora que el programa fuente en lenguaje ensamblador se explora de principio a fin, en cada uno de los dos pasos de compilación.

5.1.2. DESCRIPCION DE LOS PROGRAMAS

5.1.2.1. El bloque de subrutinas (I)

A fin de comprender mejor el mecanismo de los demás bloques que conforman el Ensamblador se ha colocado y se va a explicar en primer lugar el bloque de las subrutinas, ya que éstas son parte importante del trabajo del sistema completo.

Las subrutinas se han establecido para ejecutar operaciones que se repiten continuamente, o hacen el mismo trabajo para distintas partes del programa principal, estas son las siguientes:

SUB 1. Verifica que los símbolos cumplen con las especificaciones establecidas por el Ensamblador 8080/8085.

SUB 2. Verifica los caracteres ASCII y devuelve el código respectivo.

SUB 3. Explora la tabla de símbolos LABEL e informa si existe o no determinado símbolo.

SUB 4. Explora la tabla de símbolos nombre de EQU e informa si existe o no determinado símbolo.

SUB 5. Explora la tabla de símbolos nombre de SET e informa si existe o nó determinado símbolo.

SUB 6. Verifica si hay coma en el operando y si hay, aisla las partes separadas por la coma.

SUB 7. Convierte cantidad decimal en Binaria.

SUB 8. Convierte cantidad decimal en hexadecimal.

SUB 9. Elimina blancos delanteros.

SUB 10. Aisla dos segmentos de caracteres separados por el primer blanco que encuentra.

SUB 11. Procesa expresiones encerradas entre paréntesis.

SUB 12. Elabora una tabla de segmentos de caracteres separados por blancos.

SUB 13. Elabora una tabla de segmentos de caracteres unidos por operadores aritméticos.

SUB 14. Ejecuta operaciones aritméticas o lógicas.

SUB 15. Explora la tabla de símbolos LABEL para tomar el valor que corresponde a un LABEL.

SUB 16. Explora la tabla de símbolos nombre de EQU para tomar el valor que le corresponde.

SUB 17. Explora la tabla de símbolos nombre de SET para tomar el valor que le corresponde.

SUB 18. Verifica si una cadena de caracteres representa un decimal y devuelve su valor.

SUB 19. Verifica si el último caracter es una "D" y luego si la cadena de caracteres anterior a la "D" representa un número decimal devuelve su valor.

SUB 20. Verifica si el último caracter es una "H" si es así, verifica si la cadena de caracteres anterior a la "H" representa una cantidad hexadecimal y devuelve su valor.

SUB 21. Verifica si el último caracter es una "O" o una "Q", si es así, verifica si la cadena de caracteres anterior representa una cantidad octal y devuelve su valor.

SUB 22. Verifica si el último caracter es una "B", si es así, verifica si la cadena anterior representa una cantidad Binaria y devuelve su valor.

SUB 23. Imprime: localización, contenido y programa fuente.

SUB 24. Imprime: localización y contenido.

SUB 25. Imprime: Valor y línea fuente.

SUB 26. Verifica si dato que especifica dimensionamiento representa un decimal.

SUB 27. Imprime: mensaje de error.

La elaboración del contenido de este bloque fué la mas laboriosa, larga y tediosa de todas, pero es el bloque de mayor importancia y el trabajo se justifica.

5.1.2.2. Bloque de condicionamiento (II)

En esta sección se inicializan y se definen las características del trabajo que va a ejecutar el Ensamblador, para lo cual el sistema solicita los datos correspondientes emitiendo preguntas a través del terminal, las que deberán contestarse adecuadamente; por ejemplo, debemos suministrar al sistema: El nombre del programa que elaborado en lenguaje Ensamblador 8080/8085 se encuentra archivado en el Diskette de trabajo del Usuario y cual es el número de su última línea.

Debemos indicarle cuantas direcciones simbólicas (LABEL) tiene ese programa, para que se reserve espacio de memoria a fin de almacenar las varias tablas que con estos símbolos y datos relativos se van a establecer en el transcurso del proceso de Compilación.

Debemos indicarle con el mismo propósito, cuantos símbolos representarán los nombres de asignaciones "EQU" y también los de asignación "SET".

Con esas informaciones, el programa se autodimensiona y abre adecuadamente el archivo virtual que contiene el programa cuya información se va a procesar.

De la lectura de datos suministrados dentro del programa elabora tres tablas de símbolos:

1. Tabla de registros de microprocesador en la que la variable R(J) representa cada uno de los ocho registros en su código ASCII decimal:

TABLA IV. Registros Simbólicos

R(0) = B	R(1) = C
R(2) = D	R(3) = E
R(4) = H	R(5) = L
R(6) = M	R(7) = A

2. Tabla de operadores aritméticos, lógicos, de desplazamiento, etc. en la que la variable O$(J) representa su símbolo en el orden de precedencia:

TABLA V. Operadores Simbólicos

O$(0) = "*"	O$(4) = "SHR"	O$(5) = "AND"
O$(1) = "/"	O$(5) = "+"	O$(6) = "OR"
O$(2) = "MOD"	O$(6) = "-"	O$(7) = "XOR"
O$(3) = "SHL"	O$(7) = "NOT"	

3. Tabla de Directivas ensambladoras que acepta este Ensamblador Cruzado, cuyos símbolos están representados por la variable D$(J).

TABLA VI. Directivas Simbólicas

D$(1) = "DB"	D$(4) = "DS"
D$(2) = "DW"	D$(5) = "STKLN"
D$(3) = "ORG"	D$(6) = "END"

Y finalmente se inicializa la variable:

N$ = "MODULO"

Luego abre la salida al impresor de línea e imprime parte del encabezamiento, con el nombre del archivo y la fecha.

A continuación se define el intervalo que del programa ensamblador se va a procesar, puede definirse desde cero hasta el final o comenzando en cualquier número de proposición, hasta cualquier otro número posterior, siempre entre el cero y el último, que ya se definió al comienzo.

Finalmente se solicita el número que señalará la primera localización o posterior posición de memoria, el que debe suministrarse en hexadecimal y seguido de la letra "H", caso contrario o si el valor excede de 65.535, se emite un mensaje de error y se pide el dato otra vez. Si no se pone nada, es decir se presiona 'RETURN', el sistema asume valor ϕ.

Acto seguido se inicia el primer paso

de Ensamblaje del que se hablará en el siguiente capítulo.

En la sección 5.1.4 encontramos el diagrama de flujo de esta sección.

5.1.2.3. El primer paso (Bloque III)

Antes de describir los algoritmos que ejecuta el bloque III echemos un vistazo al material con el que va a trabajar y cual va a ser el producto de su trabajo.

a. DATOS

1. Programa escrito en lenguaje Ensamblador 8080/8085, almacenado en archivo virtual, en Diskette de trabajo del Usuario, abierto en el paso dado por el bloque II y del que el bloque actual llama las líneas una por una.

2. Tabla de símbolos de códigos de operación del Lenguaje Ensamblador 8080/8085 almacenados en un

archivo secuencial en el Diskette del sistema, desde donde el bloque actual lo utiliza para el reconocimiento (Parsing) del contenido del campo del código de operación. Esta tabla se llama "INSTRS".

3. Tabla de Instrucciones de Máquina en el equivalente decimal del código binario del lenguaje máquina archivado en secuencia adecuada para las necesidades del trabajo de los pasos 1 y 2 del COMPILADOR. Está almacenada en un archivo secuencial con el nombre "CLINST" en el Diskette del Sistema, desde donde el bloque actual lo llama para explorar (SCAN) la clase de instrucción que se está procesando.

4. La tabla de Directivas ensambladoras establecida en memoria por el bloque II, desde donde se llaman para efectuar el reconocimiento (PARSING) del contenido del campo

de Código de Operación cuando no se ha encontrado en éste un código de operación.

5. Tabla de código de error.

6. Las tres tablas de símbolos del Usuario que al tiempo que se van generando y almacenando en memoria, se convierten en datos para que en el análisis de las siguientes líneas del Programa fuente - sean exploradas (SCANNED) en busca de símbolos repetidos, caso en el cual se emite un mensaje de error. Solo los nombres de la Directiva "SET" pueden aparecer repetidamente.

b. PROGRAMAS DE ENSAMBLADOR CRUZADO

1. Sección Bloque III, objeto de esta Sección.

2. Sección Bloque I, Subrutinas descrita en la Sección 5.1.2.

c. RESULTADOS

1. Tabla de símbolos de direcciones simbólicas (LABEL:), compuesta de:

   I. Símbolo
   II. Línea en la que aparece
   III. Dirección que le corresponde.

2. Tabla de símbolos de nombres de asignaciones "EQU", compuesta de:

   I. Símbolo
   II. Línea en la que aparece
   III. Equivalente decimal de la asignación.

3. Tabla de símbolos de nombres de asignaciones "SET", compuesta de:

   I. Símbolo
   II. Línea en la que aparece o se repite.

4. Nombre del programa Ensamblador, si cumple con las especificaciones que se encuentran en el manual, si no, se emite mensaje de error.

5. Después de mantener la pista del contador de localización, lo vuelve a inicializar, cuando

termina el primer paso, antes de cambiar al bloque IV, es decir, antes de iniciar el segundo paso.

6. Se emiten mensajes de error y se contabilizan si existen.

7. Se imprime el encabezamiento de las columnas de listado de programas: objeto y fuente.

5.1.2.4. Descripción del paso 1.

Según la descripción precedente y el gráfico 5.6, podemos comprender fácilmente que el trabajo que deben efectuar los algoritmos que conforman el - bloque III del Primer paso de compilación es fundamentalmente la exploración (SCAN) del programa fuente, en busca de los símbolos que se hayan especificado en el campo 1 de las líneas del programa fuente, para la elaboración de las tablas respectivas, que luego serán utilizadas como datos en el segundo paso.

5.1.2.5. Línea vacía

Lo primero que se hace es verificar si la línea llamada tiene algún contenido, si no, se llama la

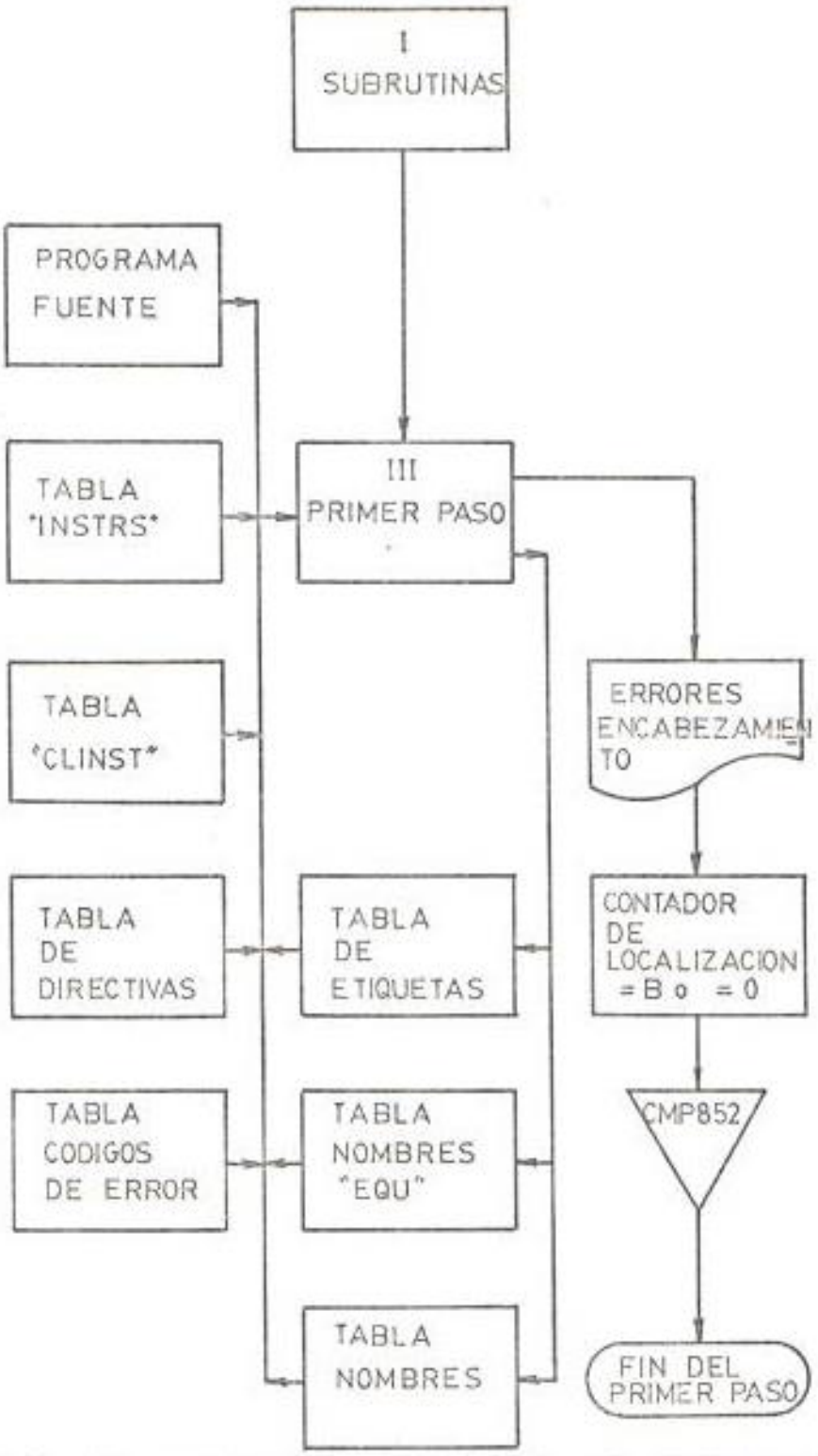
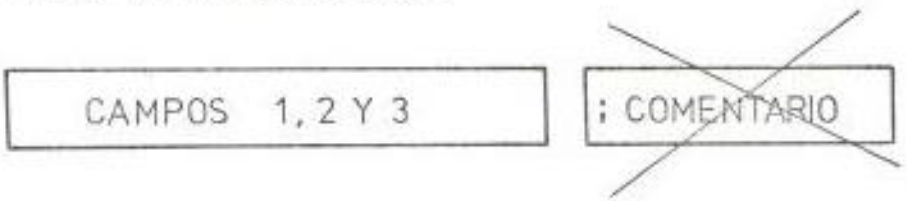


FIG. 5.6 RELACIONES EXTERNAS DEL PRIMER PASO

siguiente línea.

Campo de comentario o campo 4. A continuación se determina si tiene comentario, para lo que se busca la existencia del separador-identificador que le corresponde iniciar el campo 4: ";" y si existe se elimina la parte de la línea que está después de esta clave quedando para examinarse si existen o no y las características de los primeros campos, desde luego una línea puede estar constituída solo de comentarios en cuyo caso se elimina toda, o puede no tener este campo en cuyo caso se conserva toda.



5.1.2.6. Campo de Símbolos de identificación o campo 1.

Luego se determina si existe el campo 1 y si este corresponde a una dirección simbólica, LABEL o a un nombre simbólico de EQU o SET. Para lo que se procede como sigue:

I. TABLA DE SIMBOLOS "LABEL"

a. Se explora la línea buscando el separador

que identifica un LABEL: ":" y si existe, se aisla la sección anterior y se elimina el separador, si no existe, se procede a las operaciones que se explicarán más adelante.

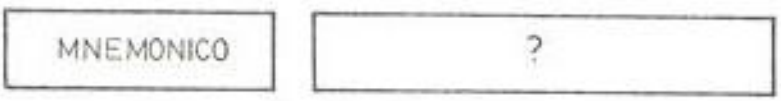
b. Si existe ETIQUETA (LABEL), se procede a verificar que cumpla las especificaciones para ser aceptado como símbolo, y si es nuevo, si no, se emiten mensajes de error.

c. Se le asigna el valor que tiene el contador de localización en ese punto, y se tabulan, luego se actualiza el contador de localización como se explicará.

SIMBOLO = CONTADOR DE LOCALIZACION

II. CONTADOR DE LOCALIZACION

Si existe etiqueta, se aisla el primer campo que se encuentra en lo que queda de la línea.



En el segmento aislado forzozamente debe encontrarse un código de operación o una direc

tiva que no sea ni "EQU" ni "SET". Si no es así, se emite mensaje de error.

Del contenido de este segmento depende la existencia y contenido de lo que resta de la línea y lo que es mas importante en este paso: el valor que tendrá el contador de localización para la siguiente línea, es decir, esta operación sirve principalmente para actualizar el contador de localización, y para lo que vamos a describir a continuación.

III. TABLAS DE NOMBRES SIMBOLICOS DE EQU y SET

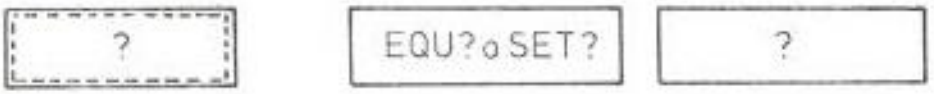
Si, y solo si, no existe un LABEL, verificado ya en I es posible la siguiente condición: como antes aislamos el primero de los campos que se encuentra en la información que tenemos para analizar.

?	?

pero ahora no sabemos que podemos encontrar en esta sección aislada y se procede primero como en el caso II solo que no emite mensaje de error todavía, para lo demás se pro

cede exactamente como en el caso II.

Si no hubo resultado positivo del trabajo anterior, entonces se reserva adecuadamente ésta sección y de lo que resta de la línea se aisla su primer campo.



Ahora sólo podemos encontrar una de las directivas "EQU" o "SET", caso contrario, se emite mensaje de error.

Si no es error, entonces se verifica la validez del símbolo como ya se ha explicado, y se lo acepta como nombre.

A continuación, se procesa la información que debe estar presente en el resto de la línea, si no es válida por algún motivo, se emite mensaje de error, caso contrario se asigna el valor decimal equivalente al código de Máquina binario, al nombre simbólico.

= valor del contenido residual de la línea.

Luego de lo cual se los incluye en la tabla correspondiente.

El contador de localización no se altera.

5.1.2.7. Directivas "NAME" y "END"

Estas, si bien están incluídas en las consideraciones anteriores, tienen particularidades que las hacen ser tratadas especialmente.

"NAME". Esta directiva para poder ser aceptada como válida, debe preceder a la primera instrucción de máquina, caso contrario, se emite mensaje de error y el programa Ensamblador asume el nombre de "MODULO".

No altera el contador de localización, salvo que tenga dirección simbólica. En lo demás, se procesa como ya se ha explicado.

"END". Esta directiva termina el proceso, sea del primero o del segundo paso, aún cuando existieran más líneas después de la línea en la que aparece "END". En lo demás, se procede como está explicado.

Otra manera de terminarse el trabajo es llegando al límite del intervalo definido en el Bloque II.

5.1.2.8. Finalización del primer paso

Luego de efectuadas las operaciones descritas, y concluído el intervalo o el programa, según sea el caso, se efectuan tres operaciones finales:

1. Se reinicializa el contador de localización, en preparación para el 2ª paso.

2. Se imprimen los encabezamientos del listado que se generará en el 2ª paso; y

3. Se llama la siguiente parte del programa ENSAMBLADOR 8080/8085: "CMP852", operación automática mediante la proposición OVERLAY.

Y estamos listos para el 2ª paso que se describirá en el siguiente capítulo.

5.1.2.9. El Segundo paso (Bloque IV)

Como en el capítulo anterior, antes de describir los algoritmos que ejecuta el bloque IV echaremos un vistazo a sus relaciones externas, es decir, al material con el que va a trabajar y a lo que va a producir que en realidad, será el producto final.

a. DATOS

Los datos que este paso utiliza son los mismos que se utilizaron en el primer paso exactamente, aunque en un grupo de ellos hay una variación, y se utiliza una tabla adicional, para recordarlos y explicar lo que cambia se los va a enumerar.

1. Programa fuente Ensamblador 8080/8085.
2. Tabla de símbolos de Códigos de operación
3. Tabla de Instrucciones de Máquina
4. Tabla de Directivas
5. Las tres tablas de símbolos que se crearon en el primer paso que ahora no cambian en el contenido de los símbolos.
6. La nueva tabla fija que ahora se utiliza

es la que elaborada por los algoritmos de iniciación (Bloque II) se encuentra en la memoria con las representaciones de los registros de microprocesador 8080/8085.

b. PROGRAMA DE ENSAMBLADOR CRUZADO

1. Sección Bloque IV, objeto de este capítulo.

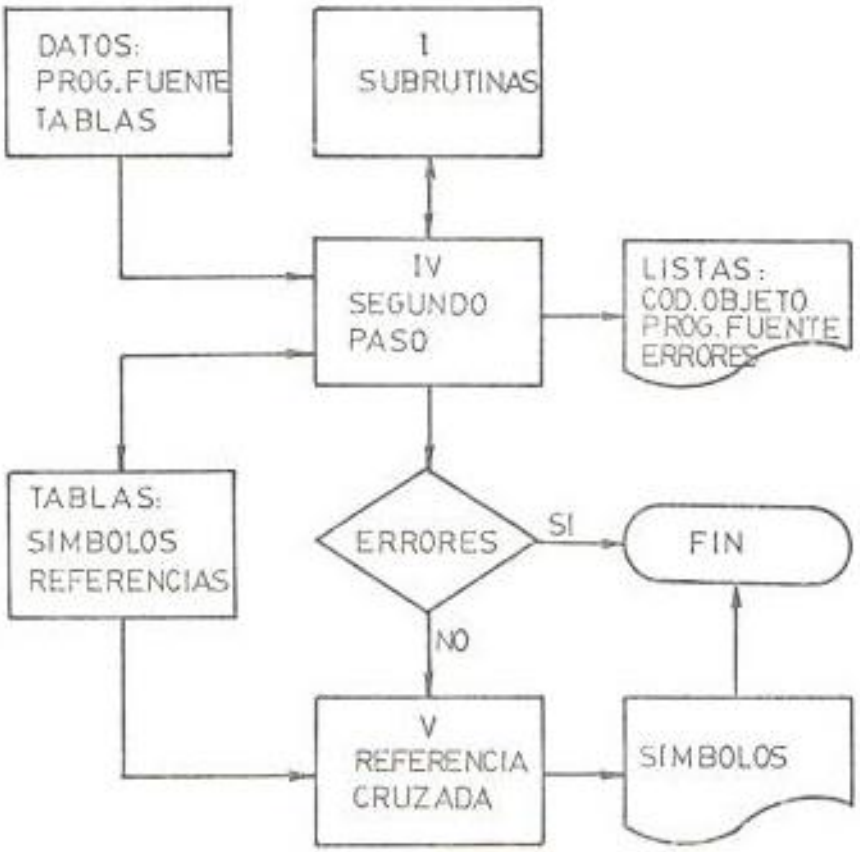


FIG. 5.7 RELACIONES EXTERNAS DEL SEGUNDO PASO

2. SECCION BLOQUE I. Subrutinas, descritas en la Sección 5.1.2.1.

c. RESULTADOS

1. Tabla de referencias de direcciones simbólicas (LABEL).

2. Tabla de referencias de nombres de asignaciones "EQU".

3. Tabla de referencias de nombres de asignaciones "SET".

4. Tabla de equivalentes decimales de las asignaciones set, que se va generando o cambiando en este paso.

5. Ensamblaje de las "instrucciones y datos procesados".

6. Localización para cada Byte del resultado obtenido en 5.

7. Emisión impresa en columnas y en hexadecimal de localización, contenido de esa localización, el número de secuencia de la línea de programa fuente y la línea de programa fuente.

8. Emisión de mensajes de error impresos; y

9. Emisión del número de errores.

d. SALIDA

La salida o finalización del trabajo, depende del número de errores ya que solamente si no hay errores sigue con el siguiente paso, que es la referencia cruzada, caso contrario no ejecuta este paso y señala fin.

5.1.2.10. Descripción del paso 2

En este paso vamos a explorar nuevamente el programa fuente desde el principio, pero con otras finalidades y para otros procesamientos. En este paso no estamos interesados para nada en el primer campo, como no sea para mantener el "track" del contador del localización, como veremos en la descripción que sigue.

Como antes, las líneas de programa fuente son llamadas de una en una en la secuencia ordenada por su número de línea.

5.1.2.11. Lineas vacías

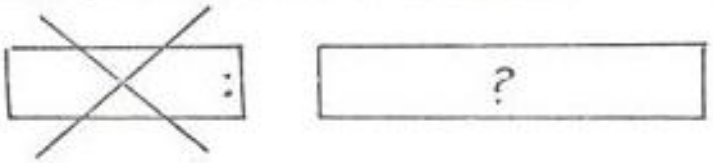
Igual que antes las líneas vacías son descartadas de inmediato.

5.1.2.12. Líneas con comentarios.

Así mismo, las líneas que solo son de comentario se descartan y si existe cualquier contenido antes del comentario se elimina la sección del comentario y se analiza la sección que ha quedado.

5.1.2.13. Los campos de LABEL

Se reconoce la línea que tiene un símbolo LABEL, y se descarta este campo.



Los campos de nombre se tratan posteriormente.

Si no tiene LABEL el caso es el mismo que una línea con este campo eliminado, y es con este material restante que ahora vamos a trabajar para su proceso.

5.1.2.14. Exploración de código de operación

Se aisla el primer campo del dato que disponemos y exploramos la tabla de símbolos de códigos de operación para determinar si es o no

un código de operación, si es así se lo traduce a su código de máquina equivalente en decimal, del binario, y se lo pasa a la segunda - exploración, que se efectúa mas adelante.

5.1.2.15. Exploración de directivas Ensambladoras

Si no es código de operación, pasamos a explorar

la tabla de símbolos de Directivas para determinar si es una de ellas, si es así, se pasa a otra sección del programa que procesa el dato residual.

5.1.2.16. Directivas EQU y SET

Si hasta aquí no se ha identificado el contenido del campo procesado, se lo elimina y del dato residual se aisla el primer campo, el cual se explora para reconocer si es EQU o SET, en cuyo caso se pasa a otra sección que procesa el dato residual, caso contrario se emite el mensaje de error de "CODIGO DE OPERACION" en

esa línea.

5.1.2.17. Las variantes

Hemos encontrado que en las dos últimas exploraciones, el contenido del campo que se analiza queda definido completamente, lo que no ocurre en la primera y eso se debe a que la mayoría de los códigos de operación o instrucciones de máquina tienen variaciones que cambian su propia estructura, lo que depende principalmente de lo que he definido como clase de instrucción.

5.1.2.18. Clases de instrucciones de máquina

Para cumplir con los propósitos del ensamblaje, he clasificado las instrucciones de máquina o códigos de operación, en 7 grupos o clases:

CLASE 1. 2 instrucciones: una de dos registros sencillos y una de un registro sencillo y un byte de datos.

CLASE 2. 27 instrucciones sin operando.

CLASE 3. 10 instrucciones de registro sencillo.

CLASE 4. 8 instrucciones de registro doble.

CLASE 5. 10 instrucciones de un byte de datos.

CLASE 6. 22 instrucciones de los bytes de datos.

CLASE 7. 1 instrucción RST.

Que suman 80 instrucciones: "INSTRUCTION SET".

5.1.2.19. El nombre del programa fuente

Como ya se dijo antes, el nombre del programa debe aparecer antes de la primera instrucción de máquina, así que antes de proceder a la codificación de las instrucciones, se verifica si existe tal nombre en cuyo caso se imprime en el listado, si no, se asume e imprime el nombre "MODULO" y ya estamos listos para procesar las Instrucciones de Máquina.

5.1.2.20. Codificación de las instrucciones de máquina

Como está dicho, la codificación de una Instrucción de Máquina, depende de su clase, así que una vez que hemos determinado que lo que el campo de códi-

go de operación contiene es una Instrucción de Máquina, el próximo paso es identificar a que clase pertenece para según eso procesar el residuo de la línea, si existe, adecuadamente y editar en línea los valores adicionales que definirán completamente la INSTRUCCION DE MAQUINA.

5.1.2.21. Exploración de clase de instrucción

A fin de cumplir lo anteriormente expresado y con el dato de instrucción de máquina que nos envía el primer explorador, se explora la tabla de clase de instrucción a fin de determinar a que grupo pertenece con el objeto de aplicarle el procedimiento diseñado para cada caso y finalmente lograr la codificación, esto nos da las siguientes alternativas:

1. INSTRUCCION CLASE 1. Son solo dos estas instrucciones: MVI y MOV.

   Verificamos que lo primero que aparece en campo residual es un registro sencillo, si es afirmativo se edita la Instrucción agregando el código del registro correspondien-

te, caso contrario se emite mensaje de error y sale.

Si lo anterior fue correcto, aislamos la última parte y si la instrucción es MOV repetimos el último procedimiento, pero si es MV1 se define un byte dato extra con su correspondiente localización. Igualmente si algo no está bien, se emite mensaje de error y sale.

2. INSTRUCCION CLASE 2. Son las 27 instrucciones más sencillas de operar, pues se codifican directamente, si el campo del operando no contiene nada, caso contrario se emite mensaje de error y sale.

3. INSTRUCCIONES CLASE 3. Esta clase de instrucción también es relativamente sencilla, pues únicamente tenemos que verificar que en el campo del operando sólo existe un registro sencillo, si es afirmativo, el código de Máquina del registro correspondiente, se edita en la clase de instrucción con lo que la codificación está completa. Como antes, si algo no está bien, se emite mensaje de error.

4. INSTRUCCIONES CLASE 4. Con estas instrucciones el procedimiento es algo parecido al de las instrucciones clase tres, solo que se busca un registro doble y se edita su propio código con lo que la instrucción está completa, excepto en el caso de "LXI" con la que además tenemos que aislar y procesar un dato que ocupara dos Bytes extras de memoria. Como sabemos, si algo no está bien se emite mensaje de error.

5. INSTRUCCIONES CLASE 5. En estas 10 instrucciones debemos procesar el contenido del campo del operando, que debe ser un dato que sólo ocupe un Byte extra de memoria. El código de Máquina de la instrucción, no se altera. También se detectan errores.

6. INSTRUCCIONES CLASE 6. Son 22 en las que tampoco se modifica el código de Máquina de la instrucción, y se procesa el contenido del - campo del operando a fin de obtener un dato que solo deberá ocupar 2 bytes extras de memoria. Se detectan errores.

7. INSTRUCCIONES CLASE 7. Realmente es una sola instrucción: RST. Pero esta instrucción tiene ocho variantes, lo que debe especificarse en el campo del operando que se procesa y cuya identificación de código de máquina se edita en la Instrucción. Si algo no está bien, se emite mensaje de error.

Y de esta manera, hemos procesado todas las instrucciones de Máquina, luego de la cual se las expresa en hexadecimal, lo mismo que la localización y sale al impresor junto con la línea - fuente original en el caso de ser instrucción de un solo Byte, para bytes adicionales solo salen: la localización y el código objeto en hexadecimal al impresor. Luego de lo cual se llama la siguiente línea y así, hasta el fin del intervalo especificado. Mas adelante explicaremos como finaliza el proceso.

5.1.2.22. Procesamiento de Directivas Ensambladoras

De la misma manera que se operó con las Instrucciones de Máquina. Una vez que se identifica una directiva como contenido del campo de código

de operación, se procede al análisis y procesamiento del contenido del campo del operando que es obligatorio, excepto para la directiva "END" en la que es optativo.

La primera diferencia en el proceso es que una directiva por sí sola no genera un código de máquina por lo tanto no se le asigna ningún valor.

Una directiva mas bien establece condiciones de funcionamiento o define datos que no se pueden definir con las Instrucciones, es decir, se puede tomar como órdenes a cumplirse en el proceso cuyos datos se encuentran en el campo del operando.

En el este paso se da cumplimiento a estas especificaciones mediante algoritmos adecuados y solo los datos se codifican en lenguaje de máquina que luego en hexadecimal se imprimen con la localización que le corresponde y junto a la línea fuente original en el primer renglón de localización.

También, si el caso lo requiere se emiten mensajes de error.

5.1.2.23. Procesamiento de asignaciones EQU

Cuando encontramos una Directiva EQU, explora mos su tabla (elaborada en el primer paso), en busca del valor que tiene asociado al símbolo y en código hexadecimal se emite al impresor junto con la línea fuente original.

5.1.2.24. Procesamiento de asignaciones SET

En el caso de la directiva SET, la tabla elaborada en el primer paso no contiene ningun dato asociado al símbolo que le corresponde, sólo tiene el símbolo y la referencia.

El contenido del campo del operando se proce sa en el segundo paso y se agrega a la tabla asociándolo al símbolo correspondiente debido a que esta directiva está hecha para que pueda cambiar su valor en el transcurso del programa.

Luego de tabularla para referencia futura, sa le en hexadecimal junto con la línea fuente.

5.1.2.25. Los errores

Los errores se emiten al impresor inmediatamente que se detectan, y se contabilizan.

Sólo en el caso en que no se hayan detectado errores en el proceso de Ensamblaje se procede al siguiente paso, contenido en el boque V, cuyo proceso se describirá en el siguiente capítulo.

5.1.2.26. El campo del operando

Todo lo anterior se ha descrito dando la impresión de que el procesamiento del contenido del campo del operando es sencillo, lamentablemente no lo es, y al contrario es el campo mas difícil de tratarlo, especialmente cuando su contenido se define como una expresión que representa un dato, de manera que una intrincada red de subrutinas contenidas en el bloque I se encarga de su procesamiento cada vez que éste aparece.

5.1.2.27. Finalización del segundo paso

El segundo paso puede finalizar de dos maneras:

a. Cuando alcanza el límite del intervalo definido al comienzo, en el paso preparativo o de condicionamiento definido en el bloque II.

b. Cuando se encuentra la directiva "END", que señala el fin del programa fuente.

Luego de lo cual verifica que no hay errores para pasar al bloque V, caso contrario, señala fin de Compilación, imprimiendo "FIN".

5.1.2.28. Referencia Cruzada (Bloque V)

Esta sección del Ensamblador Cruzado 8080/8085 emite un listado de los símbolos definidos por el usuario en su Programa Fuente, elaborado en lenguaje Ensamblador.

Además, y de ahí, su nombre, incluye asociado a cada símbolo los números de línea de Programa fuente en la que se define, y los números de línea de programa fuente en la que se referencian los datos asociados al símbolo.

Se usan para tal efecto los siguientes algoritmos:

5.1.2.29. Símbolos LABEL

Se explora la tabla de símbolos LABEL en la que estos símbolos están almacenados en el orden en el que aparecen en el Programa Fuente; se los toma secuencialmente y luego, asociado a cada símbolo existe una tabla que contiene los números de las líneas en las que ese determinado símbolo aparece, estos números también están almacenados en el orden en que el símbolo aparece o es referenciado.

Y así cada símbolo y sus números de línea de referencia, salen al impresor y se imprimen en una línea.

Luego se toma el siguiente símbolo y la operación, se repite.

5.1.2.30. Símbolos nombre de EQU y SET

El procedimiento usado para la impresión de los símbolos LABEL, se usa también para los símbolos de Nombres de EQU, y lo mismo, a continuación se procede con los símbolos de Nombres de SET.

Antes de imprimir las tablas se imprime el encabezamiento que indica que se trata de los símbolos definidos por el usuario.

Al término de la impresión, se imprime "FIN", que declara que se ha finalizado el trabajo.

5.1.2.31. El Editor

El Ensamblador cruzado 8080/8085 que se ha descrito en las secciones anteriores trabaja con programas que elaborados en lenguaje Ensamblador de microprocesador 8080/8085 deben estar almacenados en un Diskette denominado del Usuario, desde donde el COMPILADOR lo llama para el procesamiento de sus líneas.

Como ya se ha explicado, se trata de un archivo VIRTUAL, creado usando el lenguaje BASIC del sistema HT/11 y deben cumplirse ciertas características para no tener sorpresas al usar esos archivos, en vista de lo cual, y a fin de facilitar la labor del Usuario se ha elaborado un programa EDITOR, cuyo uso se

describirá en el "Manual de Operación del Ensamblador Cruzado 8080/8085". (Apéndice D).

Las características propias de los archivos virtuales se pueden consultar en el "Manual Basic del Usuario del HT/11".

Aquí se hará la descripción del Programa EDITOR que estará almacenado en el Diskette del sistema y desde donde se llamará para su uso.

5.1.2.32. Estructura del EDITOR

El Editor está constituído por seis partes que son:

a. Sección Preparatoria
b. Operación Ø. Para crear los archivos
c. Operación 1. Para modificar archivos
d. Operación 2. Para obtener listado del programa en pantalla.
e. Operación 3. Para obtener listado impreso del programa; y
f. Sección de subrutinas

5.1.2.33. Sección Preparatoria

Se ha elaborado al comienzo del programa un sistema que además de permitir la selección de la operación que se desea efectuar se defina el nombre y las características del programa que en lenguaje Ensamblador se va a manipular.

Aquí, al igual que en el Ensamblador Cruzado, se usa la técnica del Overlay para editar en línea, al tiempo de corrida, los datos que no pueden ser definidos mediante variables.

Para comenzar, después de explicar brevemente como proceder, se imprime en pantalla la lista de las operaciones que se pueden efectuar con este programa, de la que selecciona la que se desea efectuar, o salir del programa.

Si no se elige salir del programa, se define el nombre del archivo virtual que contiene, o contendrá, el programa Ensamblador 8080/8085.

A continuación, se define la dimensión del ar-

chivo, que luego será editada en línea y se ramifica a la operación elegida.

5.1.2.34. Operación Ø

La operación cero se ha diseñado para crear adecuadamente el archivo virtual y las líneas que lo van a conformar.

La creación consiste en reservar espacio para la cantidad de líneas determinada con anterioridad y limpiarlo, última operación que consiste en poner blancos en cada una de las líneas.

Como se comprenderá, si se ha usado el nombre de un archivo existente, la operación cero elimina todo su contenido.

Cierra el archivo y automáticamente ramifica a la operación 1, en la que la modificación de las líneas consiste en cambiar los blancos por una proposición en Ensamblador 8080/8085.

5.1.2.35. Operación 1

Con esta operación, se puede modificar cualquier

línea de un archivo existente.

Se emite un mensaje que indica que la longitud máxima de una línea puede ser de 32 caracteres, mas el número de la línea seguido de un blanco.

Si se desea, se puede tener un listado de cualquier segmento del programa, todo el programa, una sola línea, desde el principio hasta una línea determinada, o desde una línea dada hasta el final.

Para obtener el listado, basta comenzar la línea con la letra L en lugar del número, seguida del intervalo separado por un segmento, sólo un número, un número precedido por un segmento o un número seguido por un segmento, el programa toma las decisiones adecuadas e imprime en pantalla lo solicitado.

Para cambiar el contenido de una línea, basta contestar con el nuevo contenido, precedido del número.

Luego, para salir o terminar la operación, es

suficiente poner; FIN, con lo que se cierra el archivo y se vuelve al principio.

5.1.2.36. Operación 2

La operación dos emite una lista en pantalla del intervalo definido, del programa, chequea el intervalo para que no se exceda los límites del programa, y para cuando se define una línea inicial mayor que la final, caso en el que solo se imprime la línea mayor.

Luego se pregunta si se desea otro intervalo del mismo programa y si es afirmativo se repite esta operación, caso contrario, se cierra el archivo y se vuelve al principio.

5.1.2.37. Operación 3

Esta operación, abre la salida al impresor, imprime un encabezamiento y la fecha.

Solicita el intervalo, e imprime las líneas válidas contenidas en el archivo, dentro del intervalo.

Luego imprime "FIN" y sale, cerrando todos los archivos, al principio.

5.1.2.38. Operación 4

Con esta operación se termina de correr el programa y se imprime el mensaje "READY".

5.1.3. MODIFICACIONES QUE SE PUEDEN EFECTUAR Y COMO EFECTUARLAS

Dos son las modificaciones que en el programa Ensamblador cruzado se recomendaría hacer en el caso que se necesite y se pueda.

a) Ampliación de la capacidad de referencia simbólica; y
b) Creación de un archivo secuencial para almacenar el código objeto obtenido.

5.1.3.1. Ampliación de capacidad de referencia simbólica

Esta modificación, de desear hacerla, está condicionada a la ampliación de la capacidad de la la Memoria Principal del H/11 ya que tal como

está diseñado el programa, la capacidad de referencia máxima para los símbolos LABEL es de 6, esto es, solo se puede referir un mismo símbolo 6 veces.

Para los símbolos EQU y SET, la capacidad es de 5.

Se logra esta modificación simplemente cambiando en las líneas 1140 a 1144 el dato respectivo en las variables de doble dimensión IØ, E2, S2, a la nueva capacidad deseada.

5.1.3.2. Creación de un archivo secuencial para almacenar el Código objeto obtenido

Esta modificación solo sería útil en dos casos:

a. Se disponga de simulador de Microcomputador; y
b. Se disponga de un sistema automático de transferencia de programa a microcomputador o a un PROGRAMADOR DE PROMS.

En cualquiera de los dos casos, se podrían establecer los cambios adecuados para que al mismo

tiempo que se de salida del código objeto al impresor también se de salida al archivo secuencial.

También se debe especificar que este archivo sólo será válido en el caso de no existir errores, caso contrario, se lo debe eliminar.

En el caso de no existir errores, ya estamos en condiciones de transferirlo al sistema adecuado, además, lo mantendríamos almacenado en Lenguaje Máquina.

Actualmente ese archivo no se justifica porque unicamente vamos a utilizar, el Listado.

5.1.4. GRAFICOS Y DIAGRAMAS DE FLUJO

En las páginas siguientes se van a desarrollar los gráficos que representan los diagramas de flujo de los cuatro programas elaborados y que fueron descritos en la sección 5.1.2.

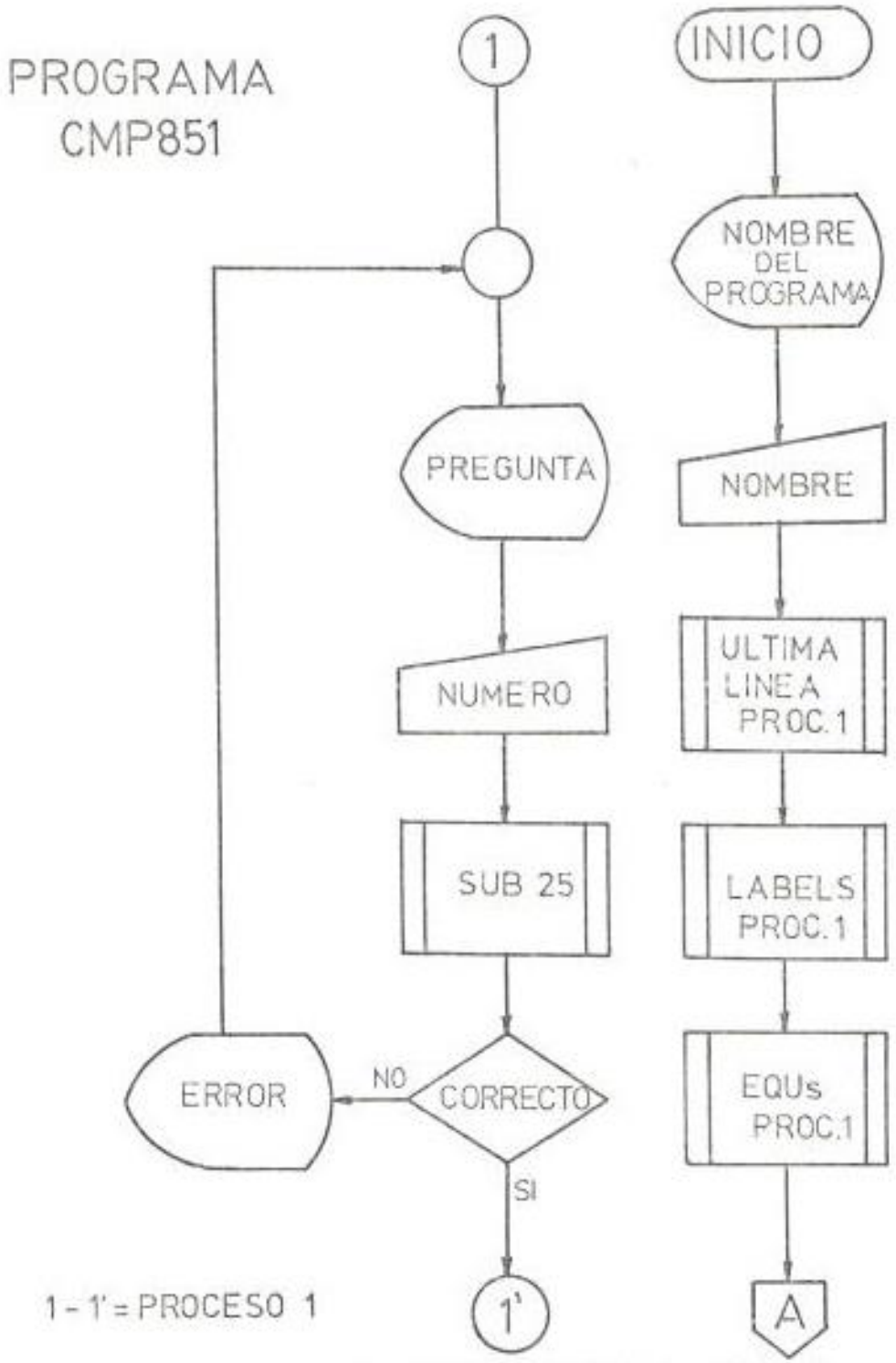


FIG 5141 BLOQUE II - PREPARACION - 1ra. PARTE

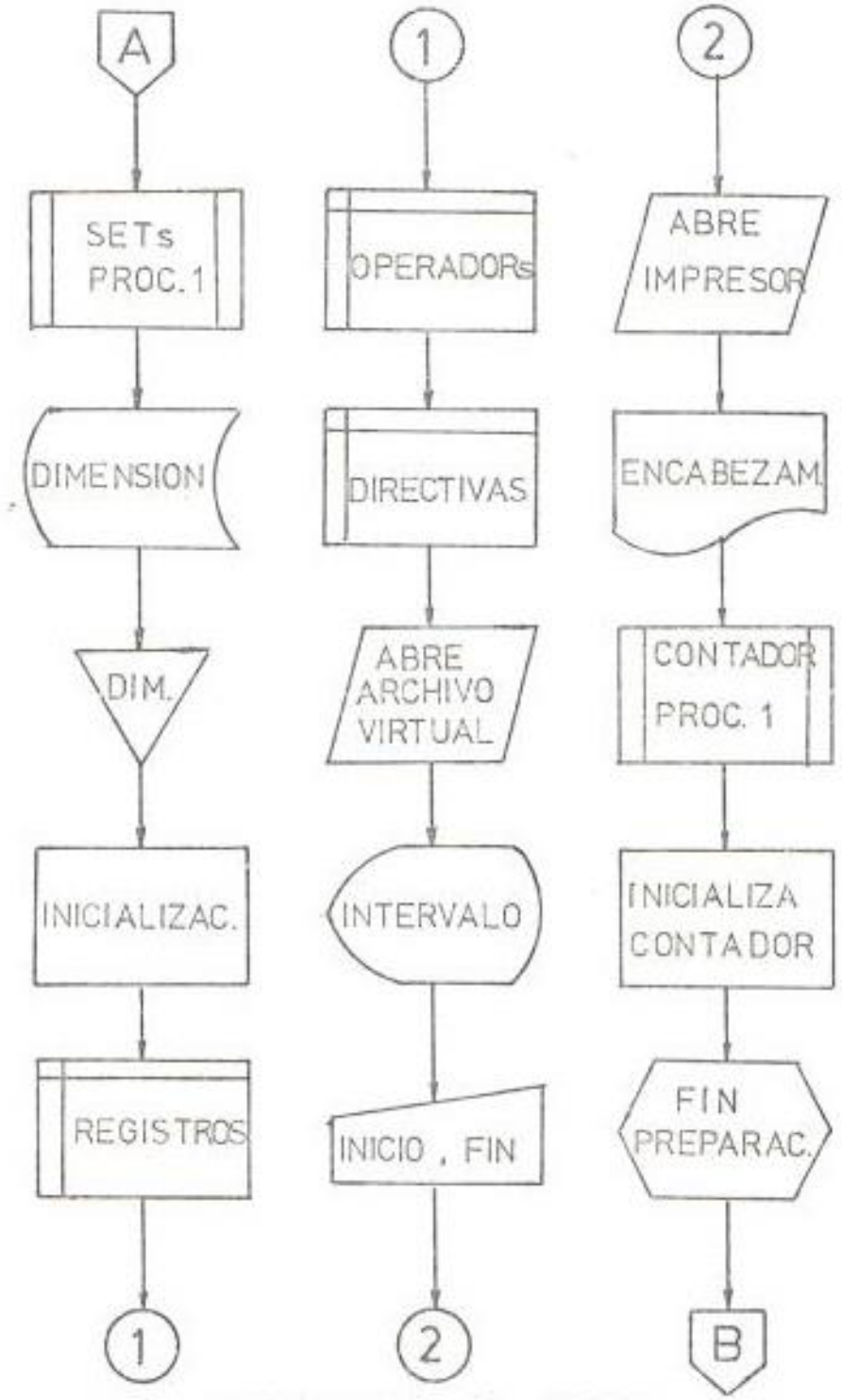


FIG 5.1.4.2 PREPARACION - 2da. PARTE

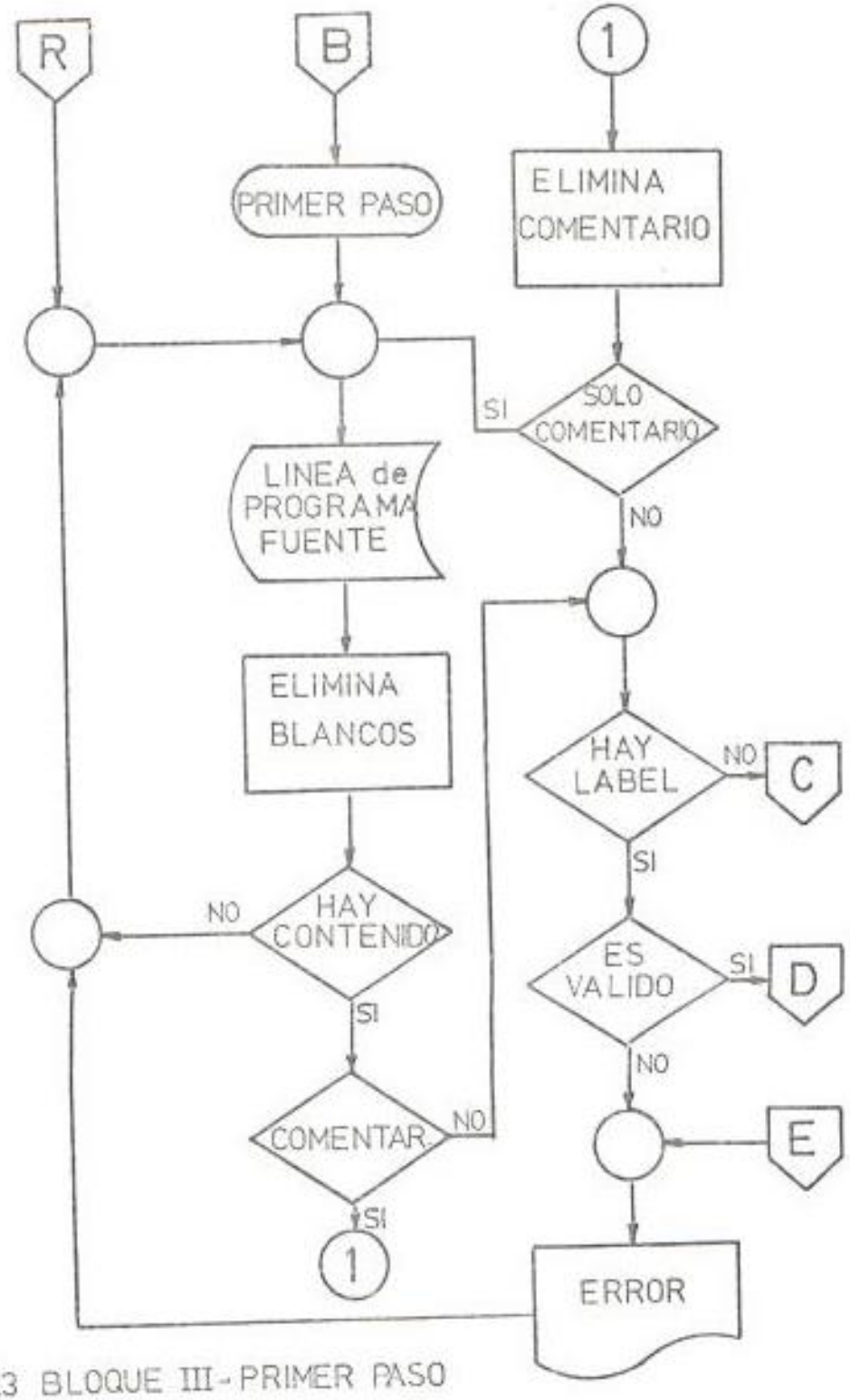


FIG. 51.4.3 BLOQUE III - PRIMER PASO

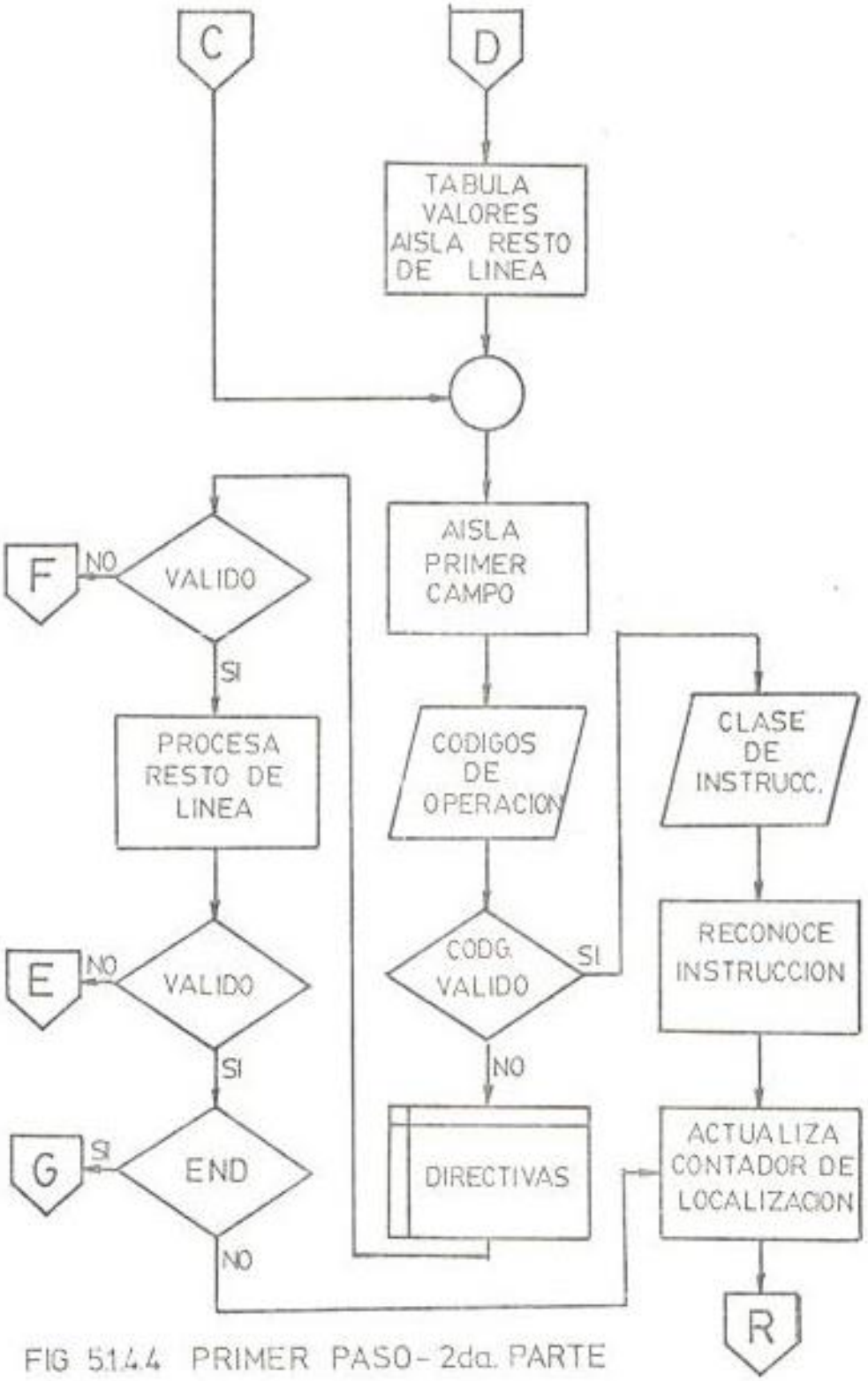


FIG 5.1.4.4 PRIMER PASO - 2da. PARTE

FIG. 5.1.4.5 PRIMER PASO - FINAL

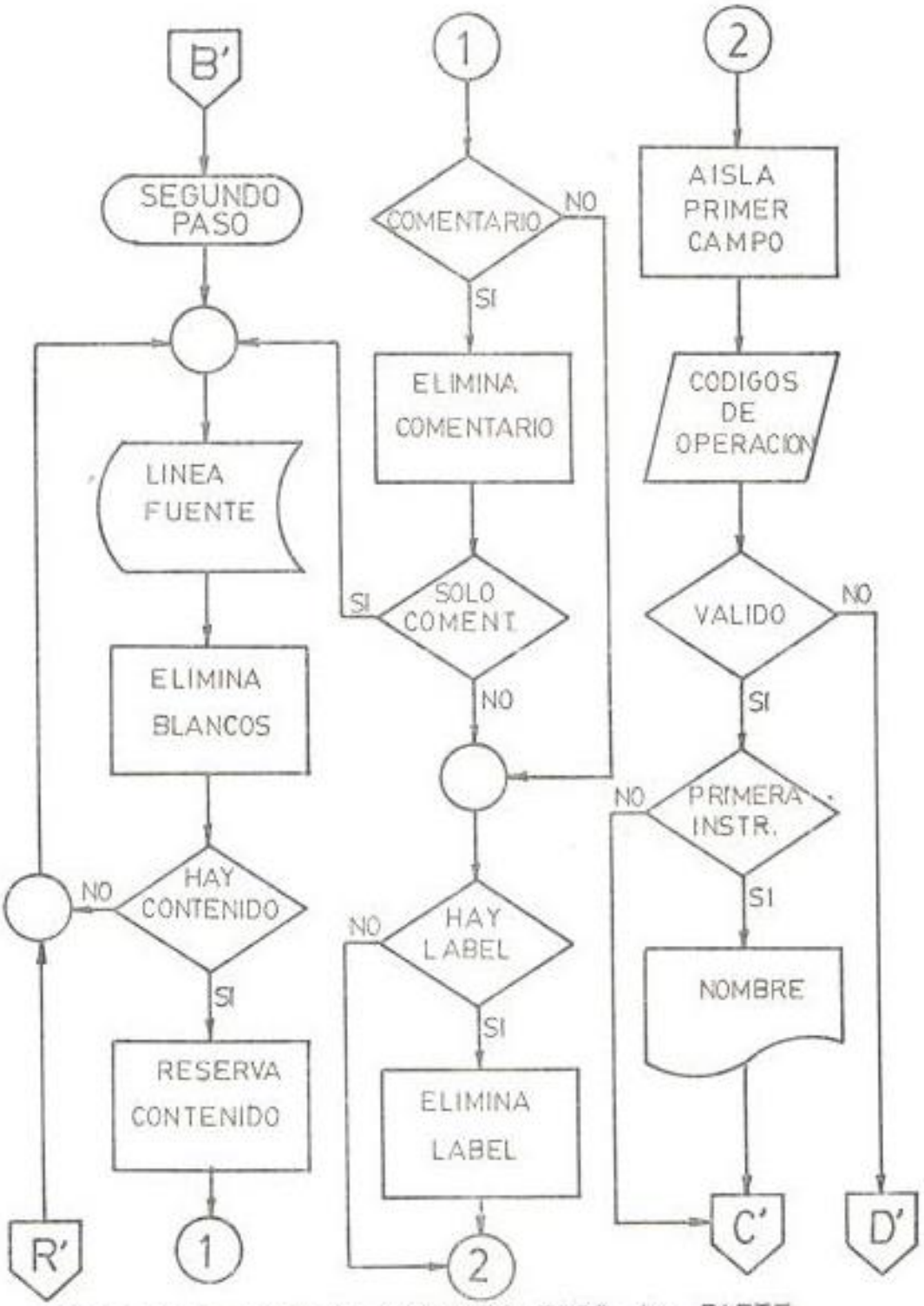


FIG. 5.1.4.6 BLOQUE IV- SEGUNDO PASO- 1ra. PARTE

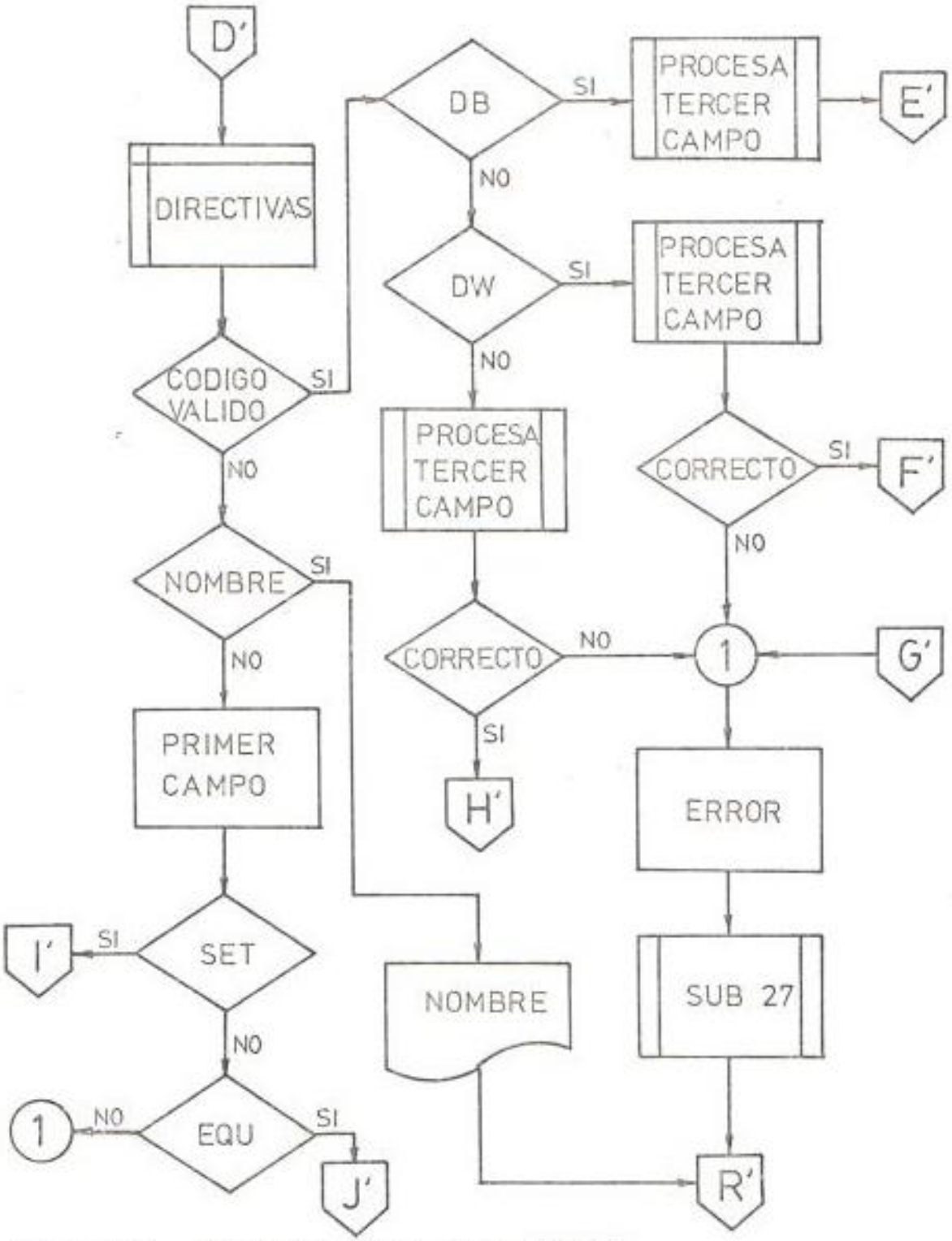


FIG. 5.1.4.7 SEGUNDO PASO - 2da. PARTE

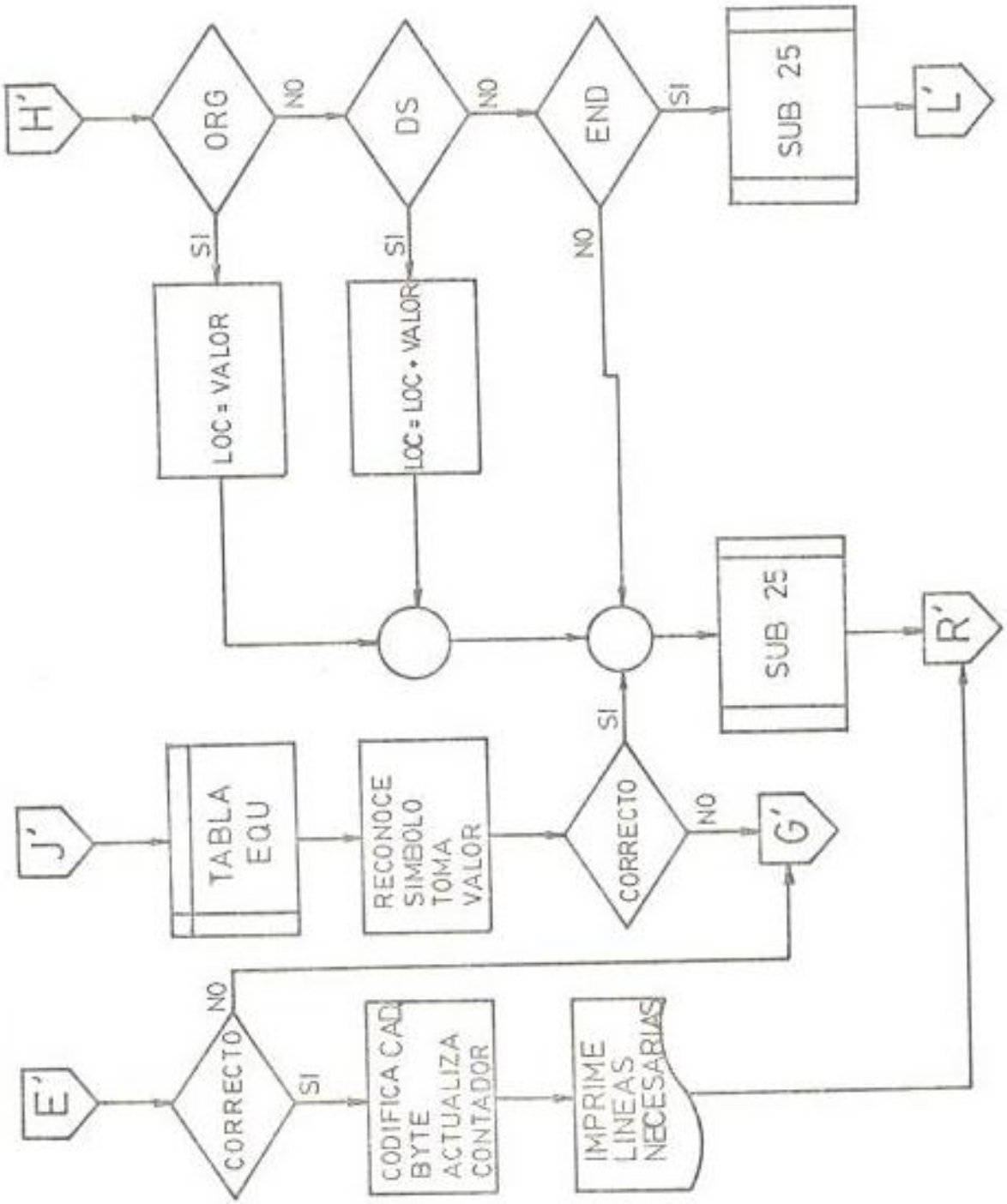


FIG. 5.14.8 SEGUNDO PASO - 3ra. PARTE

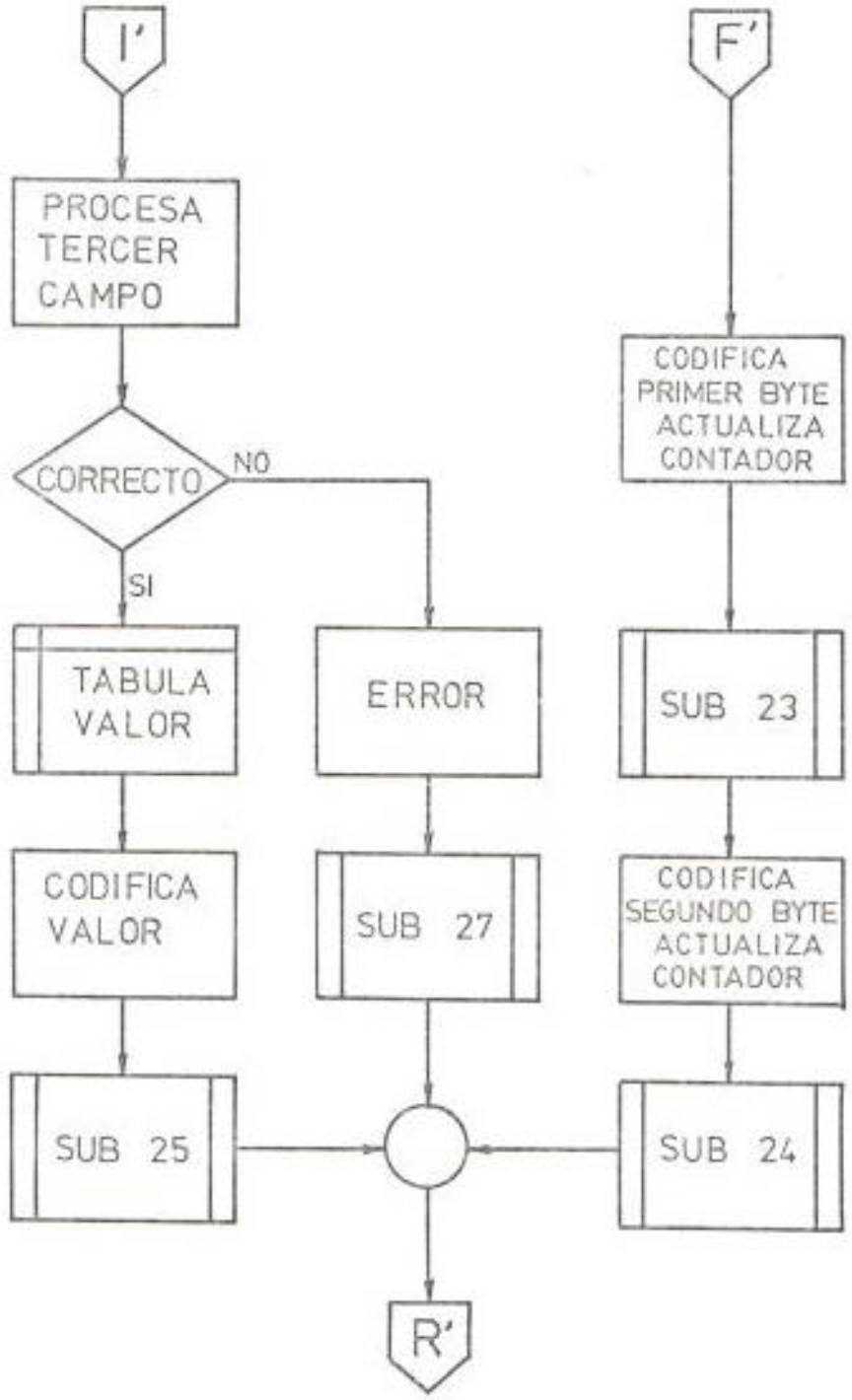


FIG. 5.1.4.9 SEGUNDO PASO - 4ta. PARTE

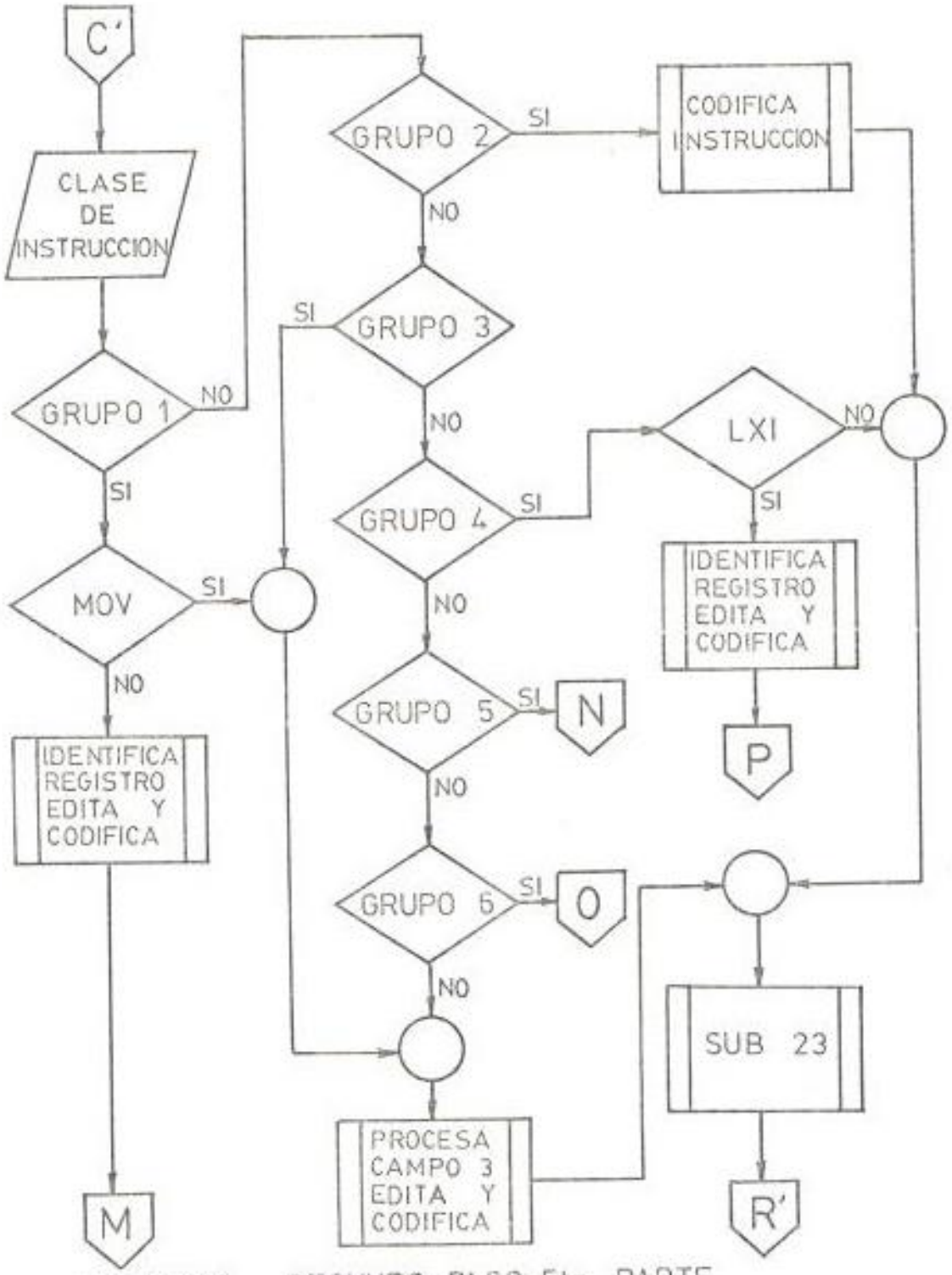


FIG. 5.1.4.10 SEGUNDO PASO-5ta. PARTE

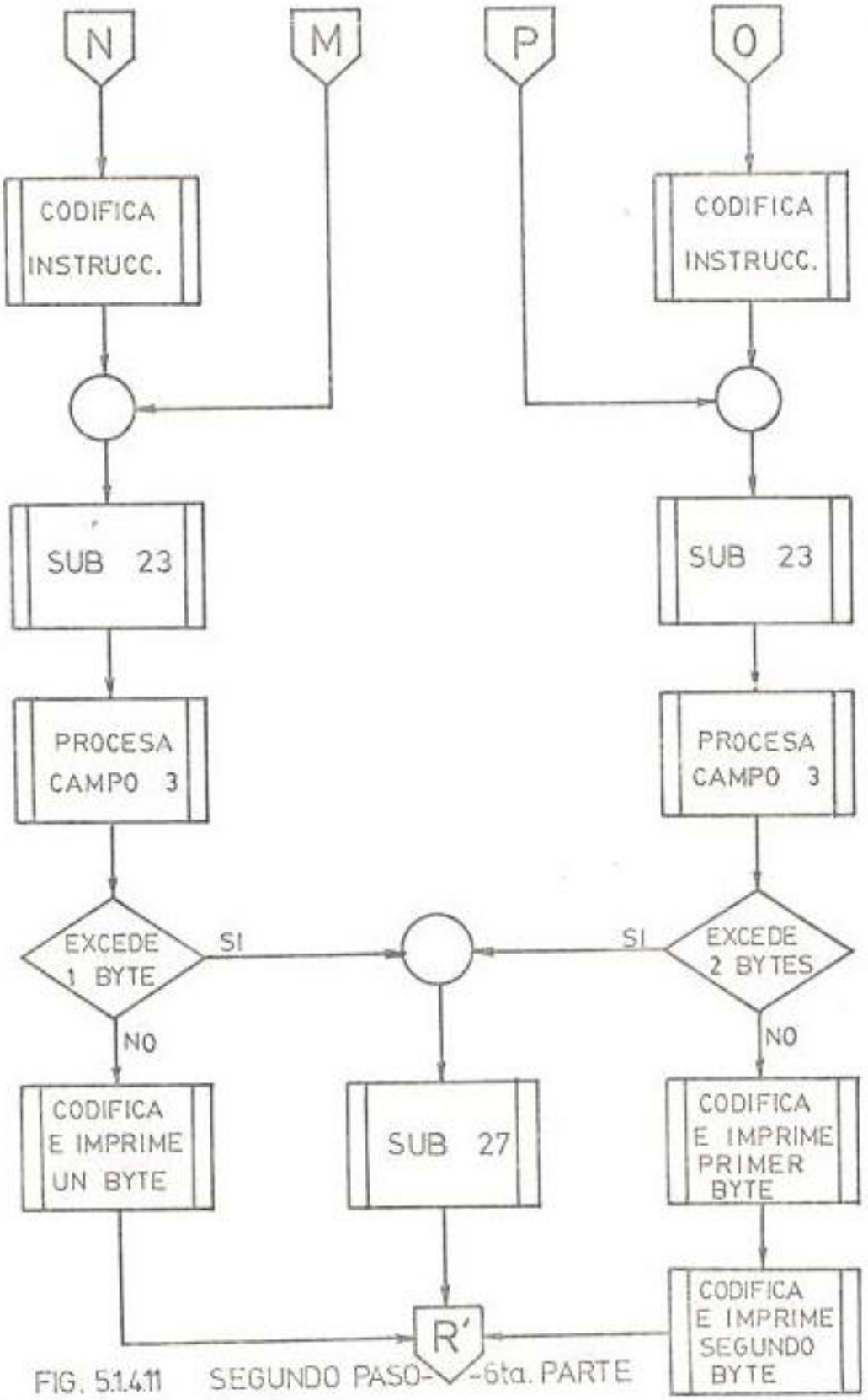


FIG. 5.1.4.11 SEGUNDO PASO- -6ta. PARTE

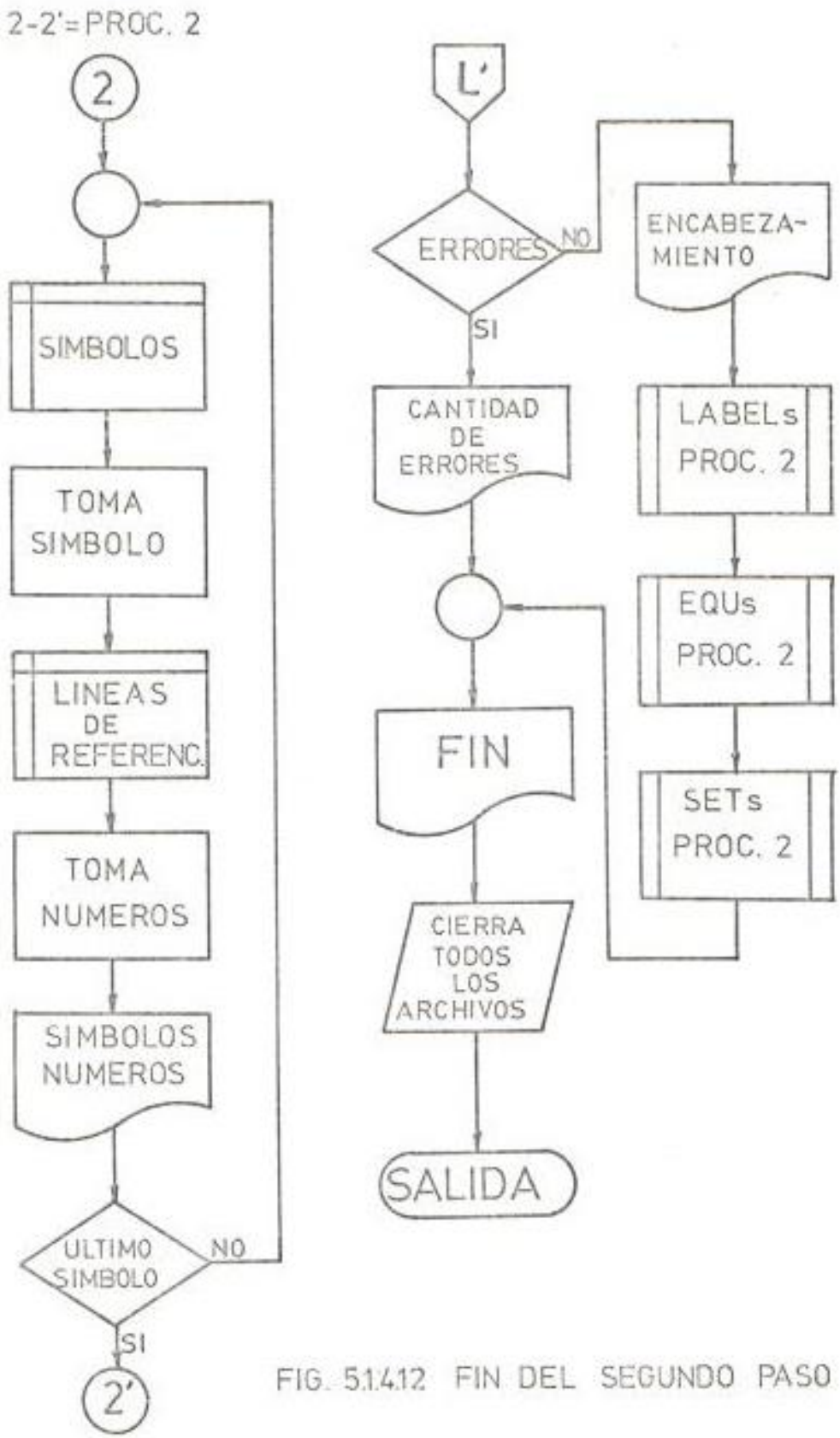


FIG. 5.1.4.12 FIN DEL SEGUNDO PASO

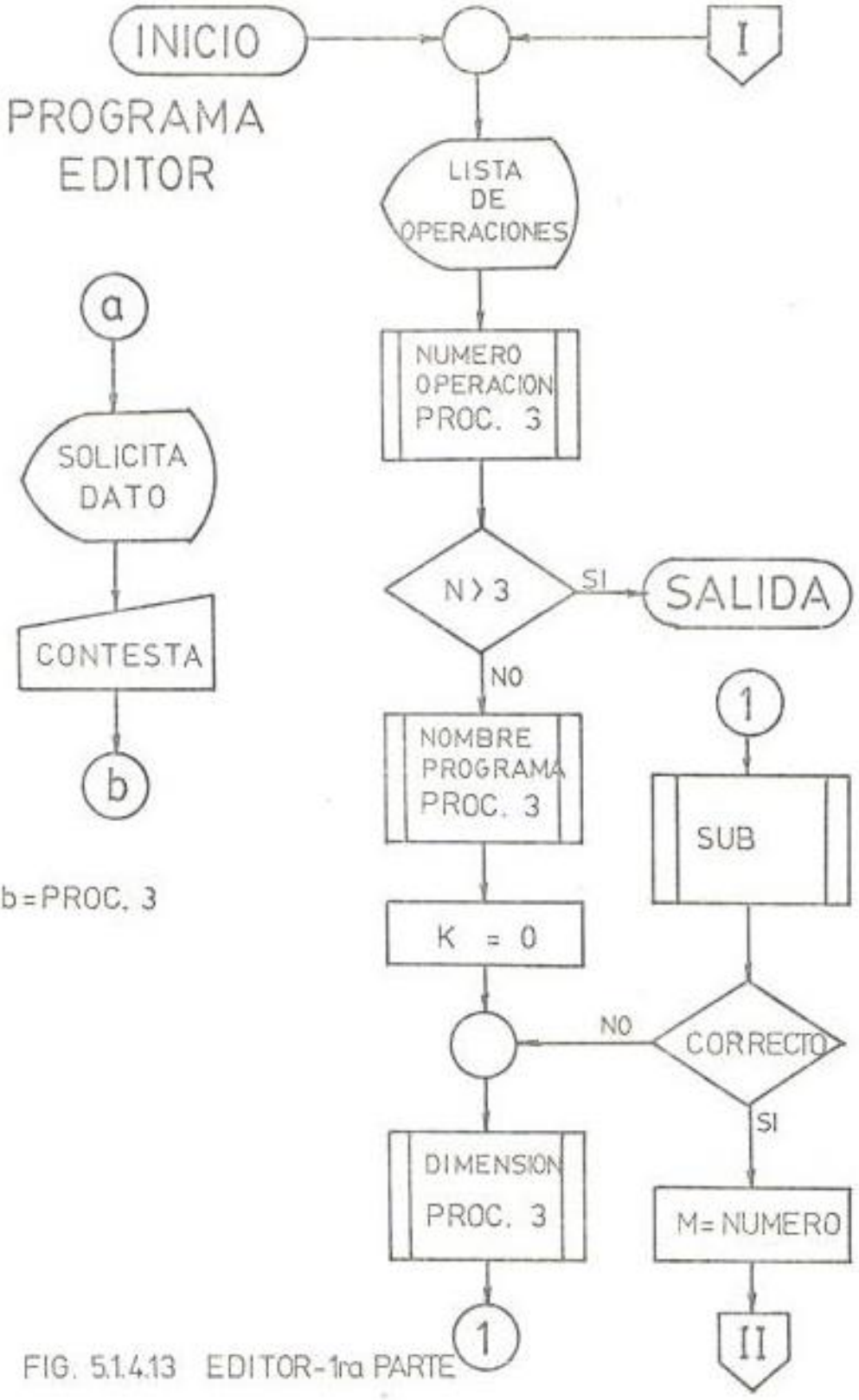


FIG. 5.1.4.13 EDITOR-1ra PARTE

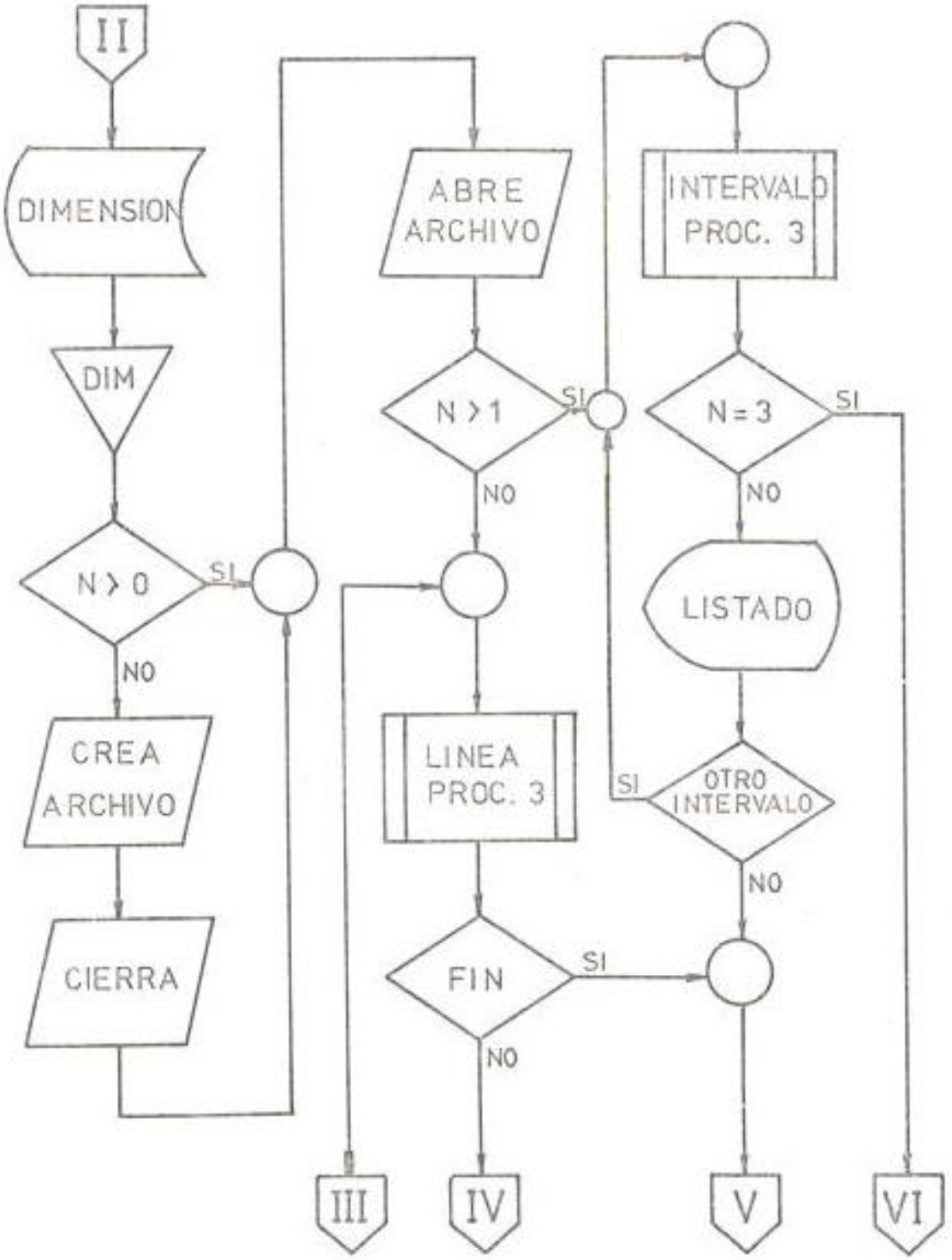


FIG. 5.1.4.14 EDITOR - 2da. PARTE

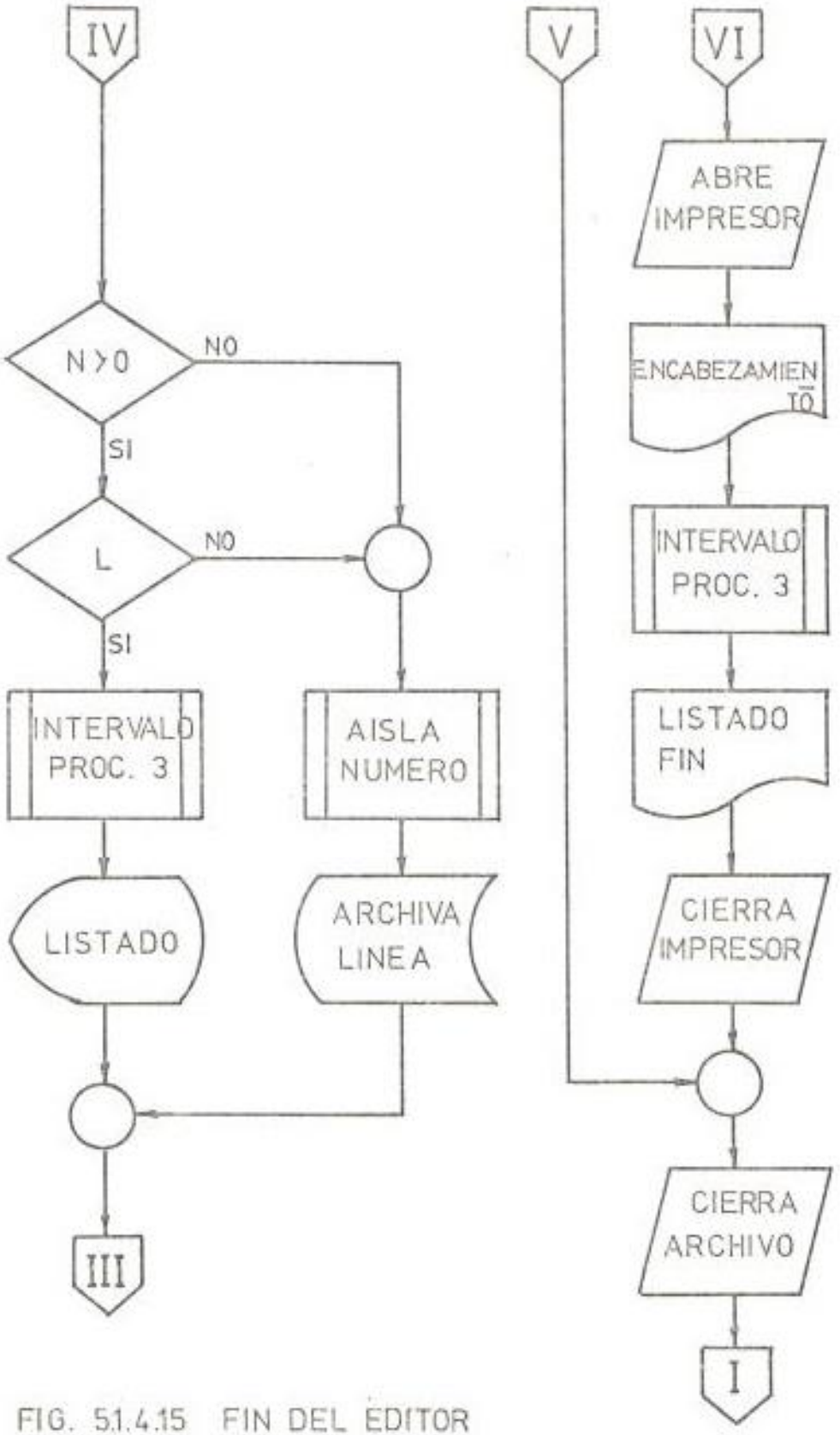


FIG. 5.1.4.15 FIN DEL EDITOR

5.2. MANUAL DE OPERACION

Con la finalidad de ilustrar y facilitar el uso de los programas de ENSAMBLAJE y EDICION, se ha elaborado un "MANUAL DE OPERACION DEL ENSAMBLADOR CRUZADO 8080/8085".

En este manual, se dan explicaciones detalladas del manejo y operación, apropiados del nuevo sistema operativo, consistente en el sistema del HT/11, EL EDITOR y el ENSAMBLADOR CRUZADO. se ha incluído en el apéndice D.

CAPITULO 6

PROYECTO DE CONSTRUCCION DE EQUIPO

Tal como se ha establecido con anterioridad, hay muchas cosas importantes e interesantes que con relación a los microprocesadores se pueden hacer a nivel de laboratorio o de investigación y desarrollo.

Sin embargo, lo más importante es que se mantenga el interés en los aspectos creativos de la Ingeniería para lograr algún cambio tecnológico útil.

En lo que sigue se van a sugerir algunos proyectos que pueden llevarse a la práctica con un poco de buena voluntad, un poco de tiempo, un poco de dinero y también mucha dedicación y estudio, que en suma es lo que realmente sirve.

6.1. PROGRAMADOR MANUAL DE PROMS

Este es el mas sencillo de los proyectos, solo requerirá de unos cuantos componentes:

a. Un PROM

b. Varios Switches

c. Un sistema de señal de tiempo;

d. Una fuente de alimentación adecuada; y

e. Un registro separador (Buffer-Latch)

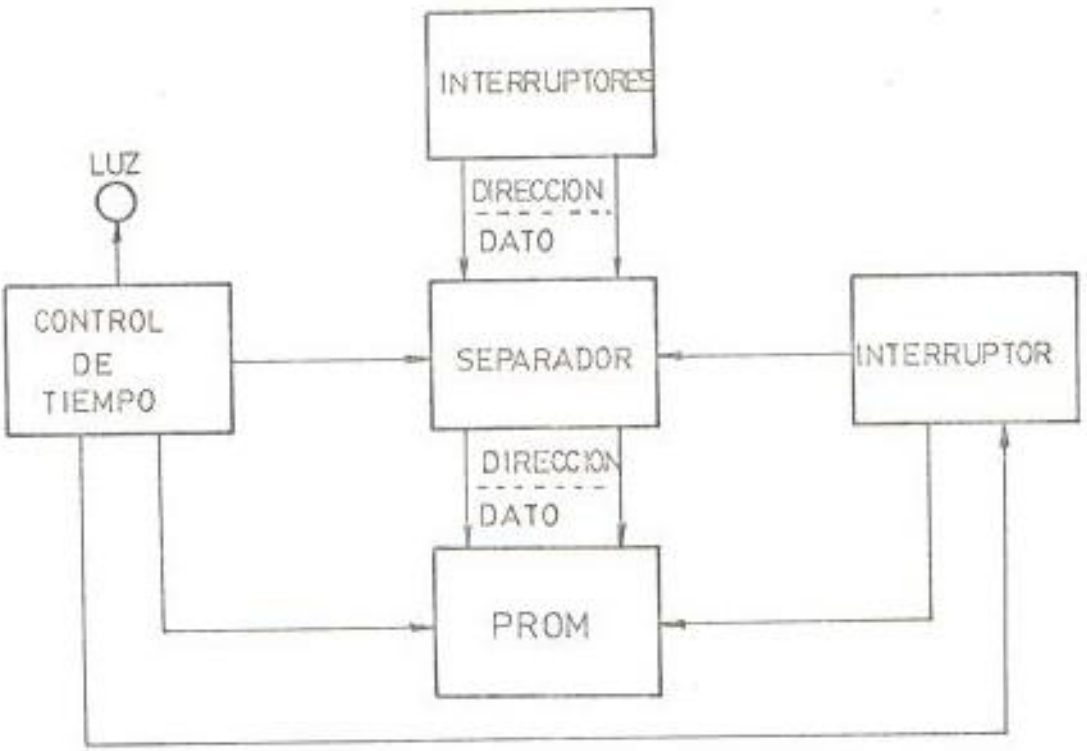


FIG. 6.1. PROGRAMADOR DE PROMS MANUAL

6.2. AUTOMATIZACION DEL PROGRAMADOR DE PROMS

La automatización del programador sugiere disponer de algún sistema de intercomunicación entre la fuente del programa y el chip PROM, para de esta manera transferir mas facilmente los datos a direcciones de PROMS.

En nuestro caso, sólo podríamos disponer de dos fuentes de programación:

a. El propio microcomputador; y
b. El sistema H/11

de los dos casos nos vamos a ocupar en los siguientes párrafos, sin embargo podemos definir al sistema de intercomunicación que hemos mencionado antes como un interfase.

6.3. INTERFASE H/11 PROGRAMADOR DE PROMS

Como queda dicho, para programar un PROM en forma completamente automática usando para ello el sistema H/11 es necesario un interfase cuyas características principales tendrían que ser: la coordinación temporal, mecanismo de interrupción y señales de control, entre el H/11 y el Programador, a través del Interfase; además, de las direcciones y los datos.

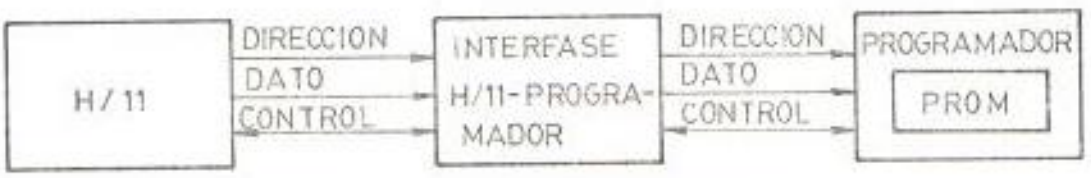


FIG. 6.2 INTERFASE H/11 PROGRAMADOR

6.4. INTERFASE MICROCOMPUTADOR-PROGRAMADOR DE PROMS

Las características de interconexión para este sistema son las mismas, aunque el interfase será radicalmente diferente porque el microcomputador tiene condiciones diferentes de funcionamiento a las del H/11.

FIG.6.3 INTERFASE MICROCOMPUTADOR PROGRAMADOR

6.5. SISTEMA INTERCONECTADO H/11-MICROCOMPUTADOR- PROGRAMADOR DE PROMS

Se puede automatizar aun más todo el sistema, interconectando a través de interfases adecuados el H/11, el Microcomputador y el programador.

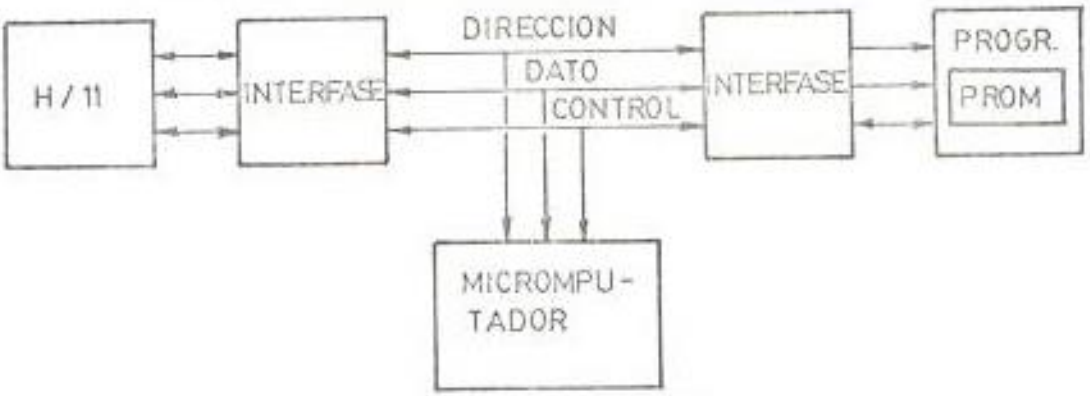


FIG. 6.4 SISTEMA INTERCONECTADO

CAPITULO 7

COMPROBACION DEL ENSAMBLADOR CRUZADO

Para verificar el trabajo del Ensamblador cruzado se utilizaron las 40 líneas que luego aparecen bajo el título de "LINEAS DE PRUEBA".

El objeto de esas 40 líneas fue el enfoque del Ensamblaje de las características mas importantes del Ensamblador 8080/8085.

Debido a ello, se encuentran algo mezcladas, directivas ensambladoras con segmentos de programas.

Están incluídas todas las directivas ensambladoras, direcciones simbólicas, e instrucciones de uno, dos y tres bytes.

Se ha probado el procesamiento del operando: con registros, símbolos, números, letras, expresiones aritméticas y lógicas, paréntesis, etc.

LINEAS DE PRUEBA

```
ENSAMBLADOR 8080/8085: PROG04
        GUAYAQUIL - ECUADOR
        ESPOL - 06-AUG-80

                        PROGRAMA

LOC.    CONT.   NUM.

                0       ESPOL   EQU     6CFH
                1       GUAY    SET     5670
                2       ECUAD   EQU     7650
                3       PROG    SET     101B
                4       PR001   EQU     ESPOL
                5       PR002   EQU     GUAY
                6       PROG    SET     ECUAD
                7               STKLN   100
                8               NAME    PR8085

                        PR8085  EQU     H+ESPOL
                9       START:  LXI     H,ESPOL
0036    21                      DS      72
0037    CF                      DB      570
0038    06              GIBIT:  RAR     ECUAD
0039    1F                      JC
003A    DA              
003B    F5              
003C    01                      DW      3E84H
003D    84              
003E    3E              
                15              ORG     34560
                16      MULT:   MVI     B,0
0E2E    06              
0E2F    00              
                17              MVI     E,9
0E30    1E              
0E31    09              
0E32    79              MULTO:  MOV     A,C
0E33    1F                      RAR
0E34    4F                      MOV     C,A
0E35    1D                      DCR     E
                21
```

```
0736  CA          22                  JZ    DONE
0737  43
0738  07
0739  78          23                  MOV   A,B
073A  D2          24                  JNC   MULT1
073B  3E
073C  07
073D  82          25                  ADD   D
073E  1F          26        MULT1:    RAR
073F  47          27                  MOV   B,A
0740  C3          28                  JMP   MULT0
0741  32
0742  07
0743  76          29        DONE:     HLT
                  30                  DB    'PROG''85'
0744  50
0745  52
0746  38
0747  30
0748  27
0749  38
074A  35
                  31                  DW    'AB'
074B  41
074C  42
                  32                  DW    'C'
074D  43
074E  00
074F  DA          33                  JC    (ESPOL+PROG)
0750  00
0751  00
0752  23          34                  INX   H
0753  C3          35                  JMP   ESPOL AND PROG
0754  C5
0755  00
```

```
0754    E8      36              XCHG
0757    E9      37              PCHL
0758    FE      38              CPI     DONE MOD 8
0759    03
0039                    39      END     GTRIT

NUMERO DE ERRORES:  0

SIMBOLOS USADOS:

START           ?
GTRIT           10      39
MULT            16
MULTO           18      28
MULT1           26      24
DONE            29      22      38
ESPOL           0       4       9       35
ECUAD           2       6       8       11
PROG1           4
PROG2           5
GUAY            1       5
PROG            3       6       35

FIN
```

7.1. ALMACENAMIENTO EN MEMORIA DE MICROCOMPUTADOR DE PROGRAMAS OBJETO OBTENIDOS CON EL COMPILADOR CRUZADO

Dentro de las 40 líneas mencionadas se encuentra desde la número 16 hasta la número 29 el segmento de programa que para multiplicar dos números está incluído en la página 6-9 del 8080/8085 Assembly Language Programming Manual. El programa objeto de este ejemplo, fue almacenado en RAM de la microcomputadora, ICS del Laboratorio de Microprocesadores.

7.2. APLICACION DE LABORATORIO

Una vez almacenado el segmento en la microcomputadora se efectuó la multiplicación.

CONCLUSIONES, SUGERENCIAS Y RECOMENDACIONES

Se pueden definir las siguientes conclusiones:

1. Cada vez es mayor la importancia de los microprocesadores en el control de procesos y procesamiento de información.

2. Es necesario tomar en cuenta lo anterior en las planificaciones académicas de la ESPOL.

3. Se requiere tener mas equipo en el Laboratorio de Microprocesadores.

En vista de lo anterior, se pueden establecer las siguientes sugerencias y recomendaciones:

1. Efectuar los análisis e investigaciones necesarias para definir métodos académicos de enseñanza y entrenamiento en el área de la microcomputación.

2. Instalar un laboratorio de microprocesadores de acuerdo a los análisis efectuados.

3. Dotar a este laboratorio de todas las facilidades

necesarias al buen entrenamiento.

4. Crear y mantener el interés de los estudiantes en el área de la microcomputación.

APENDICES

```
10 REM   ESPOL - GUAYAQUIL
12 REM   MARCO LOPEZ ORTUNO
1100 REM  ENSAMBLADOR CRUZADO 8080/8085
1102 REM  I PARTE - 28 DE JULIO DE 1980
1104 PRINT ,"NOMBRE DEL PROGRAMA: "\INPUT A$
1106 PRINT ,"ULTIMA LINEA DE ESTE PROGRAMA: "\INPUT K$
1108 I$=K$\GOSUB 1600
1110 IF E0=0 THEN 1114 \GOSUB 1620
1112 GO TO 1106
1114 PRINT ,"NUMERO DE IDENTIFICADORES (LABEL): "\INPUT L$
1116 I$=L$\GOSUB 1600
1118 IF E0=0 THEN 1122 \GOSUB 1620
1120 GO TO 1114
1122 PRINT ,"NUMERO DE ASIGNACIONES -EQU-: "\INPUT E$
1124 I$=E$\GOSUB 1600
1126 IF E0=0 THEN 1130 \GOSUB 1620
1128 GO TO 1122
1130 PRINT ,"NUMERO DE ASIGNACIONES -SET-: "\INPUT S$
1132 I$=S$\GOSUB 1600
1134 IF E0=0 THEN 1138 \GOSUB 1620
1135 GO TO 1130
1138 OPEN "LINEA.BAS" FOR OUTPUT(1) AS FILE #1
1140 PRINT #1:"1152 DIM I0$(%L%%"),I0(%L%%",5),I2(%L%%")"
1142 PRINT #1:"1154 DIM E1$(%E%%"),E2(%E%%",4),E3(%E%%")"
1144 PRINT #1:"1156 DIM S1$(%S%%"),S2(%S%%",4),S3(%S%%")"
1146 PRINT #1:"1158 DIM I3(%L%%"),E5(%E%%"),S5(%S%%")"
1148 PRINT #1:"1174 OPEN A$ AS FILE VF1%(%K%%")=32"
1150 CLOSE #1\OVERLAY "LINEA.BAS"
1152 DIM I0$(50),I0(50,5),I2(50)
1154 DIM E1$(15),E2(15,4),E3(15)
1156 DIM S1$(5),S2(5,4),S3(5)
1158 DIM I3(50),E5(15),S5(5)
1160 N1=0\N$="MODULO"\S1=0\S3=0\Z=0\B=0
1162 DATA 66,67,68,69,72,76,77,65
1164 FOR J=0 TO 7\READ R(J)\NEXT J
1166 DATA "%","/","MOD","SHL","SHR","+","-","NOT","AND","OR","XOR"
1168 FOR J=0 TO 10\READ O$(J)\NEXT J
1170 DATA "DB","DW","ORG","DS","STKLN","END"
```

```
1172 FOR J=1 TO 6\READ D$(J)\NEXT J
1174 OPEN A$ AS FILE VF1%(40)=32
1176 PRINT :"INTERVALO: INICIO,FIN "\INPUT J,K
1178 OPEN "LP:" FOR OUTPUT AS FILE #1
1180 PRINT #1:,"ENSAMBLADOR 8080/8085: "\A$
1182 PRINT #1:TAB(20)"GUAYAQUIL - ECUADOR"
1184 PRINT #1:TAB(20)"ESPOL - "\DAT$\PRINT #1
1186 PRINT :"DIRECION INICIAL: "\INPUT A$
1188 IF A$=" " THEN 1188 \T=LEN(A$)\GOSUB 2370
1190 IF E0=1 THEN 1192 \IF D0<65535 THEN 1194
1192 PRINT :"ERROR EN LA DIRECCION"\GO TO 1186
1194 B=D0
1196 S=B
1198 FOR I=J TO K\A$=TRM$(VF1(I))
1200 IF LEN(A$)=0 THEN 1560 \D=POS(A$,"$",1)
1202 IF D=0 THEN 1206 \A$=SEG$(A$,1,D-1)
1204 A$=TRM$(A$)\IF LEN(A$)=0 THEN 1560
1206 T=LEN(A$)\GOSUB 2200
1208 D=POS(A$,":",1)\IF D=0 THEN 1226
1210 I$=SEG$(A$,1,D-1)\GOSUB 2000
1212 IF E0=1 THEN 1520 \GOSUB 2050
1214 IF E0=0 THEN 1524 \GOSUB 2058
1216 IF E0=0 THEN 1524 \GOSUB 2066
1218 IF E0=0 THEN 1524 \IO$(I0)=I$\I0(I0,0)=I
1220 I3(I0)=S\I3(I0)=I\I0=I0+1
1222 A$=SEG$(A$,D+1,T)
1224 GOSUB 2200
1226 GOSUB 2212
1228 OPEN "SY:INSTRS.DAT" AS FILE #2
1230 FOR V=0 TO 254
1232 INPUT #2:I$\IF C$=I$ THEN 1248
1234 NEXT V
1236 CLOSE #2
1238 FOR V=1 TO 6
1240 IF C$=D$(V) THEN 1290
1242 NEXT V
1244 IF C$="NAME" THEN 1360
1246 GO TO 1378
1248 CLOSE #2
1250 OPEN "SY:CLINST.DAT" AS FILE #2
1252 FOR U=0 TO 44
1254 INPUT #2:N\IF N=V THEN 1276
1256 NEXT U
1258 FOR U=45 TO 54
1260 INPUT #2:N
1262 IF N=V THEN 1280
1264 NEXT U
1266 FOR U=55 TO 76
```

```
1268 INPUT #2,N
1270 IF N=0 THEN 1284
1272 NEXT U
1274 IF V=6 THEN 1280
1276 S=S+1\IF V=1 THEN 1280
1278 GO TO 1286
1280 S=S+2
1282 GO TO 1286
1284 S=S+3
1286 CLOSE #2
1288 GO TO 1560
1290 ON V GO TO 1292 ,1326 ,1344 ,1352 ,1560 ,1580
1292 FOR U=0 TO 7
1294 U=POS(A$,"*",1)
1296 IF U=0 THEN 1316
1298 I$=SEG$(A$,1,U-1)
1300 IF SEG$(I$,1,1)="'" THEN 1306
1302 S=S+T
1304 GO TO 1308
1306 S=S+LEN(I$)-2
1308 A$=SEG$(A$,U+1,T)
1310 T=LEN(A$)
1312 NEXT U
1314 GO TO 1536
1316 IF SEG$(A$,1,1)="'" THEN 1322
1318 S=S+1
1320 GO TO 1560
1322 S=S+T-2
1324 GO TO 1560
1326 FOR U=1 TO 8
1328 U=POS(A$,"*",1)
1330 IF U=0 THEN 1340
1332 A$=SEG$(A$,U+1,T)
1334 T=LEN(A$)
1336 NEXT U
1338 GO TO 1536
1340 S=S+2*U
1342 GO TO 1560
1344 GOSUB 2234
1346 IF E0=1 THEN 1536
1348 S=00
1350 GO TO 1560
1352 GOSUB 2234
1354 IF E0=1 THEN 1536
1356 S=S+00
1358 GO TO 1560
1360 IF S>B THEN 1532
1364 I$=A$
```

```
1366 GOSUB 2000
1368 IF E0=1 THEN 1520
1370 M$=I$
1374 GO TO 1560
1378 I$=C$
1380 GOSUB 2212
1382 IF C$="SET" THEN 1386
1384 IF C$<>"EQU" THEN 1532
1386 GOSUB 2000
1388 IF E0=1 THEN 1520
1390 GOSUB 2050
1392 IF E0=0 THEN 1524
1394 GOSUB 2058
1396 IF E0=0 THEN 1524
1398 GOSUB 2066
1400 IF C$="SET" THEN 1422
1402 IF E0=0 THEN 1524
1404 E1$(E1)=I$
1406 I$=A$
1408 GOSUB 2234
1410 IF E0=1 THEN 1536
1412 E5(E1)=00
1414 E3(E1)=1
1416 F2(E1,0)=I
1418 E1=E1+1
1420 GO TO 1560
1422 IF E0=0 THEN 1436
1424 S1$(S1)=I$\I$=A$
1426 GOSUB 2234
1428 IF E0=1 THEN 1536
1430 S5(S1)=00\S3(S1)=1
1432 S2(S1,0)=I\S1=S1+1
1434 GO TO 1560
1436 I$=A$
1438 GOSUB 2234
1440 IF E0=1 THEN 1536
1442 S5(P)=00\S4=S3(P)
1444 S2(P,S4)=I\S3(P)=S3(P)+1
1445 GO TO 1560
1520 E0$="SIMBOLO"
1522 GO TO 1538
1524 E0$="SIMBOLO REPETIDO"
1526 GO TO 1538
1528 E0$="SIMBOLO NO DEFINIDO"
1530 GO TO 1538
1532 E0$="CODIGO DE OPERACION"
1534 GO TO 1538
1536 E0$="OPERANDO"
```

```
1538 PRINT #1,E0$;" EN LINEA ";I
1540 Z=Z+1
1550 NEXT I
1580 S=R
1584 PRINT #1,"LOC.","CONT.","NUM.","PROGRAMA"
1586 PRINT #1
1590 OVERLAY "SY:CHPSS2.BAS"
1592 GO TO 1104
1600 FOR P=1 TO LEN(I$)
1602 FOR O=48 TO 57
1604 IF ASC(SEG$(I$,P,P))=O THEN 1610
1606 NEXT O
1608 GO TO 2750
1610 NEXT P
1612 GO TO 2746
1620 PRINT ,"ERROR EN EL NUMERO"
1622 RETURN
2000 IF LEN(I$)>6 THEN 2750
2002 FOR P=1 TO LEN(I$)
2004 N=ASC(SEG$(I$,P,P))
2006 FOR O=65 TO 90
2008 IF N=O THEN 2028
2010 NEXT O
2012 IF N=63 THEN 2026
2014 IF P>1 THEN 2020
2016 IF N=64 THEN 2028
2018 GO TO 2750
2020 FOR O=48 TO 57
2022 IF N=O THEN 2028
2024 NEXT O
2026 GO TO 2750
2028 NEXT P
2030 GO TO 2746
2050 FOR P=0 TO I0-1
2052 IF I0$(P)=I$ THEN 2746
2054 NEXT P
2056 GO TO 2750
2058 FOR P=0 TO E1-1
2060 IF E1$(P)=I$ THEN 2746
2062 NEXT P
2064 GO TO 2750
2066 FOR P=0 TO S1-1
2068 IF S1$(P)=I$ THEN 2746
2070 NEXT P
2072 GO TO 2750
2080 O=POS(A$,",",1)
2082 IF O=0 THEN 2750
2084 O$=SEG$(A$,1,O-1)
```

```
2086 D$=TRM$(D$)
2088 A$=SEG$(A$,D+1,T)
2090 GOSUB 2200
2092 GO TO 2746
2100 O$=" "
2102 FOR P=15 TO 0 STEP -1
2104 N=INT(OO/2^P)
2106 OO=OO-N*2^P
2108 O$=O$$STR$(N)
2110 NEXT P
2111 O$=SEG$(O$,2,17)
2112 RETURN
2120 D=A
2122 B=1
2124 GOSUB 2132
2126 HO$=H$
2128 O=S
2130 B=3
2132 H$=" "
2134 FOR P=8 TO 0 STEP -1
2136 C=INT(D/16^P)
2138 D=D-C*16^P
2140 IF C>9 THEN 2146
2142 C=C+48
2144 GO TO 2148
2146 C=C+55
2148 H$=H$&CHR$(C)
2150 NEXT P
2152 RETURN
2160 GOSUB 2120
2162 PRINT #1,H$,HO$,I,J$
2164 GO TO 2170
2166 GOSUB 2120
2168 PRINT #1,H$,HO$
2170 S=S+1
2172 RETURN
2200 FOR P=1 TO LEN(A$)
2202 IF SEG$(A$,P,P)<>" " THEN 2206
2204 NEXT P
2206 A$=SEG$(A$,P,LEN(A$))
2208 T=LEN(A$)
2210 RETURN
2212 FOR P=1 TO T
2214 IF SEG$(A$,P,P)=" " THEN 2224
2216 NEXT P
2218 C$=A$
2220 A$=" "
2222 RETURN
```

```
2224 C$=SEG$(A$,1,P-1)
2226 A$=SEG$(A$,P,T)
2228 GOSUB 2200
2230 RETURN
2234 B$=A$\H=0
2236 FOR U=0 TO 3
2238 U=POS(A$,"(",1)
2240 IF U=0 THEN 2250
2242 A$=SEG$(A$,U+1,T)
2244 T=LEN(A$)\H=H+U
2246 NEXT U
2248 GO TO 2750
2250 U=POS(A$,")",1)
2252 IF U=0 THEN 2274
2254 IF U=0 THEN 2750
2256 F$=SEG$(A$,U+1,T)
2258 A$=SEG$(A$,1,U-1)
2260 T=LEN(A$)
2262 D$=SEG$(B$,1,H-1)
2263 GOSUB 2276
2264 IF E0=1 THEN 2518
2265 IF U=1 THEN 2518
2266 A$=D$+STR$(00)+F$
2268 T=LEN(A$)
2270 GO TO 2234
2274 IF U>0 THEN 2750
2276 FOR L=0 TO 6
2278 IF POS(A$," ",1)=0 THEN 2288
2280 GOSUB 2212
2282 O0$(L)=C$
2284 NEXT L
2286 GO TO 2750
2288 IF L>0 THEN 2332
2294 FOR L=0 TO 4 STEP 2
2296 FOR Q=1 TO T
2298 C$=SEG$(A$,Q,Q)
2300 IF C$="*" THEN 2314
2302 IF C$="/" THEN 2314
2304 IF C$="+" THEN 2314
2306 IF C$="-" THEN 2314
2308 NEXT Q
2310 IF L=0 THEN 2512
2312 GO TO 2332
2314 IF Q>1 THEN 2316
2315 O0$(L)=C$\GO TO 2318
2316 O0$(L)=SEG$(A$,1,Q-1)
2317 O0$(L+1)=C$
2318 A$=SEG$(A$,Q+1,T)
```

```
2320 IF LEN(A$)=0 THEN 2750
2322 T=LEN(A$)
2324 NEXT L
2326 OO$(L+2)=A$
2328 L=L+2
2330 GO TO 2334
2332 OO$(L)=A$
2334 FOR O=0 TO 10
2336 FOR M=0 TO L-1
2338 IF OO$(M)=O$(O) THEN 2346
2340 NEXT M
2342 NEXT O
2344 GO TO 2750
2346 A$=OO$(M+1)
2348 GOSUB 2520
2350 IF E0=1 THEN 2518
2352 IF M>0 THEN 2380
2354 IF O=5 THEN 2362
2356 IF O=6 THEN 2366
2358 IF O=7 THEN 2456
2360 GO TO 2750
2362 OO=O+OO
2364 GO TO 2368
2366 OO=O-OO
2368 IF L<2 THEN 2514
2369 OO$(O)=STR$(OO)
2370 FOR W=1 TO L-1
2372 OO$(W)=OO$(W+1)
2374 NEXT W
2376 L=L-1
2378 GO TO 2334
2380 IF O=7 THEN 2750
2382 O1=OO
2384 A$=OO$(M-1)
2386 GOSUB 2520
2388 IF E0=1 THEN 2518
2390 IF O<3 THEN 2408
2392 IF O<5 THEN 2430
2394 IF O=5 THEN 2404
2396 IF O>6 THEN 2430
2398 OO=OO-O1
2400 GO TO 2502
2404 OO=OO+O1
2406 GO TO 2502
2408 IF O>0 THEN 2414
2410 OO=OO*O1
2412 GO TO 2502
2414 IF O1=0 THEN 2750
```

```
2416 IF D>1 THEN 2422
2418 DD=DD/D1
2420 GO TO 2502
2422 D=INT(DD/D1)
2424 DD=DD-D1*D
2426 GO TO 2502
2430 GOSUB 2100
2432 IF D>3 THEN 2442
2434 FOR W=1 TO D1
2436 D$=SEG$(D$,2,16)&"0"
2438 NEXT W
2440 DD TO 2448
2442 IF D>4 THEN 2452
2444 FOR W=1 TO D1
2446 D$="0"&SEG$(D$,1,15)
2447 NEXT W
2448 DD=BIN(D$)
2450 GO TO 2502
2452 DD$=D$
2454 DD=D1
2456 GOSUB 2100
2458 D1$=" "
2460 FOR W=1 TO 16
2462 C$=SEG$(D$,W,W)
2464 IF D=7 THEN 2482
2466 C0$=SEG$(DD$,W,W)
2468 ON D-7 GO TO 2470 ,2476 ,2480
2470 IF C$="0" THEN 2488
2472 IF C0$="0" THEN 2488
2474 GO TO 2492
2476 IF C0$="1" THEN 2492
2478 GO TO 2486
2480 IF C0$="0" THEN 2486
2482 IF C$="0" THEN 2492
2484 GO TO 2488
2486 IF C$="1" THEN 2492
2488 C$="0"
2490 GO TO 2494
2492 C$="1"
2494 D1$=D1$&C$
2496 NEXT W
2498 DD=BIN(SEG$(D1$,2,17))
2500 IF D=7 THEN 2368
2502 IF L<3 THEN 2514
2503 DD$(M-1)=STR$(DD)
2504 IF L>4 THEN 2508
2505 FOR W=M-L-2
2506 DD$(N)=DD$(M+2)
```

```
2507 NEXT W
2508 L=L-2
2510 GO TO 2334
2512 GOSUB 2520
2513 IF E0=1 THEN 2518
2514 IF 00<-256 THEN 2750
2515 IF 00>65535 THEN 2750
2516 02=INT(00/256)
2517 01=00-02*256
2518 RETURN
2520 IF T>6 THEN 2566
2522 IF A$="$" THEN 2672
2524 GOSUB 2050
2526 IF E0=1 THEN 2538
2528 00=I2(P)
2530 I4=I3(P)
2532 I0(P,I4)=I
2534 I3(P)=I3(P)+1
2536 GO TO 2746
2538 GOSUB 2058
2540 IF E0=1 THEN 2552
2542 00=E5(P)
2544 E4=E3(P)
2546 Z2(P,E4)=I
2548 E3(P)=E3(P)+1
2550 GO TO 2746
2552 GOSUB 2066
2554 IF E0=1 THEN 2566
2556 00=S5(P)
2558 S4=S3(P)
2560 S2(P,S4)=I
2562 S3(P)=S3(P)+1
2564 GO TO 2746
2566 GOSUB 2586
2568 IF E0=0 THEN 2748
2570 C$=SEG$(A$,T,T)
2572 IF C$="D" THEN 2584
2574 IF C$="H" THEN 2602
2576 IF C$="O" THEN 2636
2578 IF C$="Q" THEN 2636
2580 IF C$="B" THEN 2654
2582 GO TO 2750
2584 T=T-1
2586 FOR W=1 TO T
2588 FOR P=49 TO 57
2590 IF ASC(SEG$(A$,W,W))=P THEN 2596
2592 NEXT P
2594 GO TO 2750
```

```
2596 NEXT W
2598 D0=VAL(SEG$(A$,1,T))
2600 GO TO 2746
2602 IF T>5 THEN 2750
2604 D0=0
2606 FOR W=1 TO T-1
2608 G=T-1-W
2610 FOR P=48 TO 57
2612 IF ASC(SEG$(A$,W,W))=P THEN 2628
2614 NEXT P
2616 FOR P=65 TO 70
2618 IF ASC(SEG$(A$,W,W))=P THEN 2624
2620 NEXT P
2622 GO TO 2750
2624 P=P-55
2626 GO TO 2630
2628 P=P-48
2630 D0=D0+P*16^G
2632 NEXT W
2634 GO TO 2746
2636 IF T>7 THEN 2750
2638 FOR W=1 TO T-1
2640 FOR P=48 TO 55
2642 IF ASC(SEG$(A$,W,W))=P THEN 2648
2644 NEXT P
2646 GO TO 2750
2648 NEXT W
2650 D0=OCT(SEG$(A$,1,T-1))
2652 GO TO 2746
2654 IF T>17 THEN 2750
2656 FOR W=1 TO T-1
2658 D=ASC(SEG$(A$,W,W))
2660 IF D=48 THEN 2666
2662 IF D=49 THEN 2666
2664 GO TO 2750
2666 NEXT W
2668 D0=BIN(SEG$(A$,1,T-1))
2670 GO TO 2746
2672 D0=S
2746 E0=0
2748 RETURN
2750 E0=1
2752 RETURN
3000 END
```

APENDICE B

```
10 REM     ESPOL - GUAYAQUIL
12 REM     MARCO LOPEZ ORTUNO
1102 REM  II PARTE - 23 DE JULIO DE 1980
1104 FOR I=J TO K
1106 A$=TRM$(VF1(I))
1108 IF LEN(A$)=0 THEN 1560
1110 J$=A$
1112 O=POS(A$,";",1)
1114 IF O=0 THEN 1122
1116 A$=SEG$(A$,1,O-1)
1118 A$=TRM$(A$)
1120 IF LEN(A$)=0 THEN 1542
1122 T=LEN(A$)
1124 O=POS(A$,"I",1)
1126 IF O=0 THEN 1130
1128 A$=SEG$(A$,O+1,T)
1130 GOSUB 2200
1132 GOSUB 2212
1134 OPEN "SY:INSTRS.DAT" AS FILE #2
1136 FOR V=0 TO 254
1138 INPUT #2:I$
1140 IF C$=I$ THEN 1160
1142 NEXT V
1144 CLOSE #2
1146 FOR V=0 TO 6
1148 IF C$=D$(V) THEN 1393
1150 NEXT V
1152 IF C$="NAME" THEN 1542
1154 GOSUB 2212
1156 IF C$="SET" THEN 1368
1158 IF C$="EQU" THEN 1388
1159 GO TO 1532
1160 CLOSE #2
1162 IF N1>0 THEN 1168
1164 PRINT #1,,,,,N$
1166 N1=LEN(N$)
1168 IF V=199 THEN 1356
1170 OPEN "SY:CLINST.DAT" AS FILE #2
1172 FOR U=0 TO 26
1174 INPUT #2:N
1176 IF N=V THEN 1242
1178 NEXT U
1180 FOR U=27 TO 36
1182 INPUT #2:N
1184 IF N=V THEN 1246
1186 NEXT U
1188 FOR U=37 TO 44
```

```
1190 INPUT #2;N
1192 IF N=V THEN 1262
1194 NEXT U
1196 FOR U=45 TO 54
1198 INPUT #2;N
1200 IF N=V THEN 1306
1202 NEXT U
1204 FOR U=55 TO 76
1206 INPUT #2;N
1208 IF N=V THEN 1326
1210 NEXT U
1212 CLOSE #2
1214 GOSUB 2080
1216 IF E0=1 THEN 1536
1218 FOR U=0 TO 7
1220 IF ASC(O$)=R(U) THEN 1226
1222 NEXT U
1224 GO TO 1536
1226 A=V+U*8
1228 IF V=6 THEN 1310
1230 FOR U=0 TO 7
1232 IF ASC(A$)=R(U) THEN 1238
1234 NEXT U
1236 GO TO 1536
1238 A=A+U
1240 GO TO 1554
1242 CLOSE #2
1244 GO TO 1288
1246 CLOSE #2
1248 FOR U=0 TO 7
1250 IF ASC(A$)=R(U) THEN 1256
1252 NEXT U
1254 GO TO 1536
1256 IF V<=5 THEN 1364
1258 A=V+U
1260 GO TO 1554
1262 CLOSE #2
1263 IF V>1 THEN 1267
1264 GOSUB 2080
1265 IF E0=1 THEN 1536
1266 GO TO 1268
1267 O$=A$
1268 IF O$="B" THEN 1288
1270 IF O$="D" THEN 1292
1272 IF V=2 THEN 1536
1274 IF V=10 THEN 1536
1276 IF O$="H" THEN 1296
1278 IF V>=193 THEN 1284
```

```
1280 IF G$="SP" THEN 1300
1282 GO TO 1536
1284 IF G$="PSW" THEN 1300
1286 GO TO 1536
1288 A=V
1290 GO TO 1302
1292 A=V+16
1294 GO TO 1302
1296 A=V+32
1298 GO TO 1302
1300 A=V+48
1302 IF V=1 THEN 1330
1304 GO TO 1554
1306 CLOSE #2
1308 A=V
1310 GOSUB 2160
1312 IF SEG$(A$,1,1)="'" THEN 1322
1314 GOSUB 2234
1316 IF E0=1 THEN 1536
1318 IF D0>255 THEN 1536
1319 A=D0
1320 GO TO 1558
1322 IF T>3 THEN 1536
1324 GO TO 1350
1326 CLOSE #2
1328 A=V
1330 GOSUB 2160
1332 IF SEG$(A$,1,1)="'" THEN 1348
1334 GOSUB 2234
1336 IF E0=1 THEN 1536
1338 IF D0<0 THEN 1536
1340 A=01
1342 GOSUB 2166
1344 A=02
1346 GO TO 1558
1348 IF T>4 THEN 1536
1350 GOSUB 2000
1352 IF E0=1 THEN 1536
1354 GO TO 1560
1356 FOR U=0 TO 7
1358 IF VAL(A$)=U THEN 1364
1360 NEXT U
1362 GO TO 1560
1364 A=V+U*8
1366 GO TO 1554
1368 FOR V=0 TO S1-1
1370 FOR U=0 TO S3(V)-1
1372 IF I=S2(V,U) THEN 1380
```

```
1374 NEXT U
1376 NEXT V
1378 GO TO 1528
1380 GOSUB 2234
1382 IF E0=1 THEN 1536
1384 B5(V)=00
1386 GO TO 1496
1388 FOR V=0 TO E1-1
1390 IF I=E2(V,0) THEN 1396
1392 NEXT V
1394 GO TO 1528
1396 00=E5(V)
1397 GO TO 1496
1398 IF V>2 THEN 1480
1400 FOR U=0 TO 7
1402 O=POS(A$,"*",1)
1404 IF O=0 THEN 1418
1406 O1$(U)=SEG$(A$,1,O-1)
1408 O1$=TRN$(O1$(U))
1410 A$=SEG$(A$,O+1,T)
1412 GOSUB 2200
1414 NEXT U
1416 GO TO 1536
1418 O1$(U)=A$
1420 IF V=2 THEN 1450
1422 FOR V=0 TO U
1424 A$=O1$(V)
1426 IF SEG$(A$,1,1)="'" THEN 1440
1428 GOSUB 2234
1430 IF E0=1 THEN 1536
1432 IF 00>255 THEN 1536
1434 A=00
1436 GOSUB 2160
1438 GO TO 1446
1440 IF T>130 THEN 1536
1441 PRINT #1,V,I,J$
1442 GOSUB 2000
1444 IF E0=1 THEN 1536
1446 NEXT V
1448 GO TO 1560
1450 FOR V=0 TO U
1452 A$=O1$(V)
1454 IF SEG$(A$,1,1)="'" THEN 1470
1456 GOSUB 2234
1458 IF E0=1 THEN 1536
1460 A=01
1462 GOSUB 2160
1464 A=02
```

```
1466 GOSUB 2166
1468 GO TO 1476
1470 IF T%=4 THEN 1535
1471 PRINT #1,,,I,J$
1472 GOSUB 2000
1473 IF E0=1 THEN 1536
1474 IF LEN(A$)=2 THEN 1476
1475 A=0\GOSUB 2166
1476 NEXT V
1478 GO TO 1560
1480 GOSUB 2234
1482 IF E0=1 THEN 1536
1484 IF D0<0 THEN 1536
1486 ON V-2 GO TO 1488 ,1492 ,1496 ,1500
1488 S=D0
1490 GO TO 1496
1492 S=S+D0
1496 D=D0
1498 GO TO 1546
1500 D=D0
1502 GOSUB 2130
1504 PRINT #1,H$,,I,J$
1506 GO TO 1580
1542 PRINT #1,,,I,J$
1544 GO TO 1560
1546 GOSUB 2130
1548 PRINT #1,H$,,I,J$
1552 GO TO 1560
1554 GOSUB 2160
1556 GO TO 1560
1558 GOSUB 2166
1580 CLOSE VF1
1581 PRINT #1
1582 PRINT #1,,"NUMERO DE ERRORES: ",Z
1584 IF Z=0 THEN 1600
1586 GO TO 1645
1600 PRINT #1:
1602 PRINT #1:,"SIMBOLOS USADOS:"
1603 PRINT #1
1604 FOR I=0 TO I0-1
1606 PRINT #1:,I0$(I),
1608 FOR J=0 TO I3(I)-1
1610 PRINT #1:I0(I,J),
1612 NEXT J
1614 PRINT #1
1616 NEXT I
1618 FOR I=0 TO E1-1
1620 PRINT #1:,E1$(I),
```

```
1622 FOR J=0 TO E3(I)-1
1624 PRINT #1:E2(I,J),
1626 NEXT J
1628 PRINT #1
1630 NEXT I
1632 FOR I=0 TO S1-1
1634 PRINT #1:,S1$(I),
1636 FOR J=0 TO S3(I)-1
1638 PRINT #1:S2(I,J),
1640 NEXT J
1642 PRINT #1
1644 NEXT I
1645 PRINT #1
1646 PRINT #1:,"FIN"
1648 CLOSE #1
1650 GO TO 3000
2000 IF SEG$(A$,T,T)<>"/" THEN 2750
2002 A$=SEG$(A$,2,T-1)
2004 T=LEN(A$)
2006 O=POS(A$,"/",1)
2008 IF O=0 THEN 2016
2010 IF SEG$(A$,O+1,O+1)<>"/" THEN 2750
2012 C$=SEG$(A$,1,O-1)
2014 A$=C$+SEG$(A$,O+1,T)
2016 FOR V=1 TO LEN(A$)
2018 FOR O=0 TO 127
2020 IF ASC(SEG$(A$,V,V))=O THEN 2026
2022 NEXT O
2024 GO TO 2750
2026 A=O
2028 GOSUB 2166
2030 NEXT V
2032 GO TO 2746
2052 IF I0$(P)=A$ THEN 2746
2060 IF E1$(P)=A$ THEN 2746
2068 IF S1$(P)=A$ THEN 2746
3000 END
```

APENDICE C

```
100 REM ********************************************
110 REM *                                          *
120 REM *  EDITOR DE PROGRAMAS ESCRITOS EN LENGUAJE *
130 REM *          ENSAMBLADOR 8080/8085           *
124 REM *  PARA SER USADOS CON EL ENSAMBLADOR CRUZADO *
128 REM *                 "CMP85"                  *
132 REM *  ESTOS PROGRAMAS FUERON ELABORADOS POR:   *
136 REM *          MARCO LOPEZ ORTUÑO               *
140 REM *                                           *
141 REM *  ESCUELA SUPERIOR POLITECNICA DEL LITORAL *
142 REM *          GUAYAQUIL - ECUADOR              *
148 REM *          28 DE JULIO DE 1980              *
152 REM *                                           *
156 REM ********************************************
160 REM
200 PRINT "EDITOR DE PROGRAMAS ESCRITOS EN LENGUAJE ENSAMBLADOR 8080/8085"
210 PRINT "SELECCIONE LA OPERACION DESEADA"
216 PRINT "IMPRIMIENDO EL NUMERO CORRESPONDIENTE"
230 PRINT
240 PRINT "0  CREAR ARCHIVO E INTRODUCIR PROGRAMA"
250 PRINT "1  MODIFICAR PROGRAMA EXISTENTE"
260 PRINT "2  LISTADO EN PANTALLA"
270 PRINT "3  LISTADO IMPRESO"
280 PRINT "4  FIN"
290 PRINT
300 PRINT "OPERACION: ";
310 INPUT N
315 IF N>3 THEN 200
320 PRINT "NOMBRE DEL PROGRAMA: ";
330 INPUT A$
340 K=0
```

```
350 PRINT "NUMERO DE LA ULTIMA LINEA DEL PROGRAMA "
360 PRINT "INCLUYENDO LINEAS VACIAS: ";
370 INPUT N%
372 GOSUB 1300
374 IF E=1 THEN 350
376 N=L
380 OPEN "LINEA.BAS" FOR OUTPUT(1) AS FILE #1
390 PRINT #1,"420 OPEN A$ FOR OUTPUT AS FILE VF1$(""&M$&"")=32"
395 PRINT #1,"470 OPEN A$ AS FILE VF1$(""&N$&"")=32"
400 CLOSE #1
410 OVERLAY "LINEA.BAS"
415 IF N>0 THEN 470
430 FOR I=0 TO L
440 VF1(I)=" "
450 NEXT I
460 CLOSE VF1
480 IF N>1 THEN 820
490 PRINT "MAXIMO 32 CARACTERES POR LINEA, EXCLUIDO EL NUMERO"
500 PRINT "\INPUT A$
505 IF A$="FIN" THEN 970
510 IF VERIFY(A$+"1234567890.")<>L" THEN 700
520 A$=RIGHT$(A$,LEN(A$)-1)
522 IF LEN(TRIM$(A$))>0 THEN 530
524 I=N-M
526 GO TO 470
530 GOSUB 1200
540 O=POS(A$,"-",1)
550 IF O>0 THEN 560
552 M$=A$
554 GOSUB 1300
```

```
556 PRINT #L,VF1(L)
558 GO TO 500
560 M1=SEG$(A$,1,0-1)
562 IF LEN(TRM$(N$))>0 THEN 570
564 L=0
566 GO TO 580
570 GOSUB 1300
580 J=L
590 A$=SEG$(A$,M1,LEN(A$))
600 IF LEN(TRM$(A$))>0 THEN 630
610 K=M
620 GO TO 670
630 GOSUB 1200
640 N$=A$
650 GOSUB 1300
660 K=L
670 GOSUB 1500
680 GO TO 500
700 GOSUB 1200
710 FOR I=1 TO LEN(A$)
720 IF SEG$(A$,1,I)=" " THEN 740
730 NEXT I
740 N$=SEG$(A$,1,I)
750 GOSUB 1300
760 IF E=1 THEN 1500
770 A$=SEG$(A$,I,LEN(A$))
780 GOSUB 1200
784 IF LEN(A$)<32 THEN 790
786 A1=SEG$(A$,1,32)
788 GO TO 800
790 A$=A$+" "
800 VF1(L)=A$
810 IF L>M THEN 970
815 GO TO 500
820 PRINT :"LINEAS: PRIMERA,ULTIMA  ";
830 INPUT J,K
840 IF K<=M THEN 860
850 K=M
860 IF J<K THEN 870
865 K=J
870 IF N=3 THEN 900
880 GOSUB 1500
882 PRINT :"NUEVO INTERVALO  (SI/NO)  ";
884 INPUT A$
886 IF A$="SI" THEN 820
890 GO TO 970
```

```
900 OPEN "LPT:" FOR OUTPUT AS FILE #1
910 PRINT #1,"ENSAMBLADOR  BOSO/BOSS"
912 PRINT #1:TAB(20)"GUAYAQUIL - ECUADOR"
914 PRINT #1:TAB(20)"ESPOL - "DAT$
916 PRINT #1
918 PRINT #1,"NUM.","SEC.","PROGRAMA"
920 PRINT #1
922 N=0
930 FOR I=J TO K
934 IF VF1(I)="" THEN 940
936 PRINT #1,N,I,VF1(I)
938 N=N+1
940 NEXT I
950 PRINT #1
955 PRINT #1,,,"FIN"
960 CLOSE #1
970 CLOSE VF1
980 GO TO 230
1200 FOR P=1 TO LEN(A$)
1210 IF SEG$(A$,P,P)<>" " THEN 1230
1220 NEXT P
1230 A$=SEG$(A$,P,LEN(A$))
1240 RETURN
1300 N$=TRN$(N$)
1310 IF LEN(N$)>5 THEN 1370
1320 FOR P=1 TO LEN(N$)
1330 C=ASC(SEG$(N$,P,P))
1340 FOR Q=48 TO 57
1350 IF C=Q THEN 1400
1360 NEXT Q
1370 PRINT ,"ERROR EN EL NUMERO  "
1380 E=1
1390 RETURN
1400 NEXT P
1410 L=VAL(N$)
1420 E=0
1430 RETURN
1500 FOR I=J TO K
1510 A$=TRN$(VF1(I))
1520 IF LEN(A$)=0 THEN 1540
1530 PRINT ,I,A$
1540 NEXT I
1550 RETURN
2000 END
```

APENDICE D

MANUAL DE OPERACION DEL ENSAMBLADOR CRUZADO 8080/8085

INTRODUCCION

Con la finalidad de facilitar la labor de programación, para los microcomputadores, que usen los microprocesadores: 8008-8080 u 8085 que usan el sistema de Introducción de datos y selección de dirección de memoria en lenguaje hexadecimal de las Instrucciones de Máquina de los mencionados microprocesadores, se ha elaborado un sistema que traduce el código Ensamblador al lenguaje hexadecimal necesario.

En este manual se va a explicar el uso y manipulación adecuado de los programas que intervienen en este sistema.

CAPITULO I
EL SISTEMA

Se utiliza como base el sistema operativo del H/11, al que se agregan dos programas, cuya finalidad especifica es la manipulación de Programas que se escriben en Lenguaje Ensamblador 8080/8085, para entregarlo en forma de Programa Objeto.

LOS PROGRAMAS ELABORADOS.-Dos son los programas que se han elaborado:

a. Ensamblador Cruzado 8080/8085 - "CMP851.BAS"-
- "CMP852.BAS"-

b. Editor - "EDITOR.BAS"-

LOS PROGRAMAS QUE SE VAN A USAR.-Para que se pueda hacer uso de los programas antes mencionados, necesitamos el Compilador BASIC del HT/11. Para la manipulación de los archivos usaremos el PIP del HT/11, y para el acondicionamiento o preparación de Diskettes nuevos, usaremos el FORMAT del HT/11.

Las instrucciones del uso de los programas del HT/11 los encontrarán en el manual de su Sistema Operativo, y en el manual del BASIC correspondiente, sin embargo las condiciones necesarias, serán explicadas aquí.

EL TRABAJO A EFECTUARSE - Lo primero que necesitaremos será disponer del programa escrito en Lenguaje Ensamblador 8080/8085.

Este programa se va a almacenar en el Diskette denominado del Usuario,o deTrabajo, usando el programa EDITOR.

A continuación utilizaremos el "ENSAMBLADOR CRUZADO 8080/8085",'que tomando los datos previamente almacenados en el Diskette del Usuario, los procesa para entregar un listado de: Programa fuente, Programa objeto, Referencia cruzada, y Errores, si los hay.

CORRIENDO LOS PROGRAMAS - Como se explica al principio de éste capítulo, para usar cualquiera de nuestros dos programas, debemos previamente tener disponible el Compilador BASIC, luego llamamos al programa a usarse con el comando: OLD "SY:(Nombre de Programa a usarse) "<CR>.

Esperamos la respuesta del computador:

READY

Y comenzamos el trabajo del programa con :

RUN <CR> ó RUNNH <CR>

Luego de lo cual, ya estamos dentro del sistema que se está usando. Mas adelante se explica cómo se los maneja.

USO DEL COMANDO CLEAR.- Es recomendable usar el comando:

CLEAR <CR>

antes de correr un programa, ya que este comando limpia el área del usuario en la memoria, caso contrario, es posible que el HT/11 emita mensajes de error aparentemente sin sentido.

Si en cualquier momento del trabajo de un programa, el sistema emite un mensaje de error seguido de:

READY

Se debe recurrir al manual del BASIC para interpretar su significado, corregir su ERROR, si hay error, y proceder como se ha indicado en líneas anteriores.

USO DEL COMANDO SCR.- Es recomendable usar el comando

SCR <CR>

antes de llamar a la memoria un nuevo programa BASIC,

es decir, previo al uso del comando:

OLD "SY: (Nombre del programa)"

CAPITULO II

EL EDITOR

Una vez que se ha iniciado el trabajo del EDITOR, tal como se explicó en el capítulo I, el computador imprime en la pantalla, un mensaje, seguido de una lista de cuatro operaciones y una pregunta:

Ø. Crear archivo e introducir programa
1. Modificar programa existente
2. Listado en pantalla
3. Listado Impreso
4. Fin

OPERACION ?

Se contesta, introduciendo únicamente el número de la operación que se desea efectuar, luego de lo cual, el computador imprime:

NOMBRE DEL PROGRAMA?

Pregunta que se debe contestar introduciendo un nombre de archivo que sea aceptado por el HT/11, si es nuevo, caso contrario con el nombre correcto de un archivo existente. A continuación, el computador pregunta:

NUMERO DE LA ULTIMA LINEA DEL PROGRAMA INCLUYENDO LINEAS VACIAS?

Que debe ser contestada con un número decimal, caso contrario, se emite mensaje de error y se solicita nuevamente el dato.

Si se va a crear un archivo, este número significa la cantidad de líneas que podría tener el nuevo archivo, y eso incluye las líneas vacías que se podría dejar para futuros intercalamientos de datos, y en los restantes - casos representa el número de la última línea de un archivo existente, es decir, define la extensión del archivo.

OPERACION Ø. Esta operación, se elige para crear un nuevo archivo; si por error, se ha puesto un nombre que pertenece a un archivo ya existente, se elimina el contenido de este archivo.

A continuación, la operación es exactamente igual a la operación 1.

OPERACION 1. Dentro de la operación 1 se pueden efectuar 3 acciones:

a. Obtener un listado de contenido de líneas;
b. Introducir nuevo contenido de línea; y

c. Salir del modo de operación.

Se emite un mensaje, indicando, que la máxima cantidad de caracteres, para una línea, excluídos los caracteres del número de la línea, es de 32, y se solicita el dato.

LISTADO. Si se quiere verificar el contenido de una, o mas líneas, antes de almacenar nuevos datos, la contestación a la última pregunta debe comenzar con la letra L seguida del intervalo que puede presentarse - de varias formas:

? La

? La-b

? L - b

? L a -

En donde a y b son dos valores numéricos separados por un segmento, que están limitando las líneas a imprimirse en pantalla.

Para la ausencia de a (-b) se emite listado desde la línea Ø hasta la línea b.

Para la ausencia de b (a-) se emite listado desde la -

línea a hasta la última línea del archivo.

Una sola variable sin segmento, imprime sólo la línea especificada.

En cualquier caso, las partes pueden o no, estar separadas por blancos.

NUEVO CONTENIDO. La contestación para depositar nuevo contenido de una línea, debe comenzar con un número - decimal que esté dentro del límite del archivo, seguido de al menosUN blanco.

A continuación la línea de programa a ser ensamblada con un máximo de 32 caracteres:

Ej. ? 17 DATOS: MV1A, 36H ; COMENTARIO

...do	Espacio	Máximo 32 caracteres

SALIDA. Contestando:

? FIN

Se regresa al listado de Operaciones inicial.

OPERACION 2. La operación dos, sólo permite obtener listados en pantalla, para lo que pregunta el intervalo así:

LINEAS: PRIMERA, ULTIMA?

a lo que se contesta con los dos números separados por una coma, así:

LINEAS: PRIMERA, ULTIMA? a, b

a y b son los límites en números decimales.

Después de presentar el listado, se pregunta si se desea un nuevo intervalo:

NUEVO INTERVALO (SI/NO)?

Si se contesta SI, automáticamente se repite la operación. Caso contrario, se retorna al listado inicial de Operaciones del programa Editor.

OPERACION 3. La operación 3 es similar a la operación 2, pero esta vez, el listado sale al impresor de línea bajo un encabezamiento, y no pregunta por otro intervalo, sino que automáticamente termina la operación y regresa al listado inicial de Operaciones del programa Editor.

OPERACION 4. Esta operación se elige para salir del EDITOR. El sistema contesta:

READY

CAPITULO III

EL ENSAMBLADOR CRUZADO 8080/8085

Una vez que se ha iniciado el trabajo del CMP851, tal como se explicó en el capítulo I, el computador pregunta:

NOMBRE DEL PROGRAMA?

Debe contestarse con el nombre exacto de un archivo Ensamblador 8080/8085 existente. Luego se pide:

ULTIMA LINEA DE ESTE PROGRAMA?

Igual que en el EDITOR (Capítulo II), la contestación define la extensión o límite del archivo, se introduce en decimal, la misma cantidad que se usó para crear el archivo.

Las siguientes 3 preguntas:

NUMERO DE IDENTIFICADORES (LABEL)"?
NUMERO DE ASIGNACIONES -EQU- "?
NUMERO DE ASIGNACIONES -SET- "?

Debe contestarse con la cantidad que de cada uno de esos elementos tiene el programa ENSAMBLADOR 8080/8085 que va a compilarse.

CANTIDAD DE SIMBOLOS. Al momento, debido a la limitación impuesta por la capacidad actual de memoria del H/11 sólo se pueden definir un total de 70 símbolos que el usuario los puede distribuír entre símbolos - identificadores o de asignación.

En cualquier caso si una clase de símbolo no se va a usar, deberá ponerse cero, caso contrario el sistema, igual que en el caso de salto a la siguiente pregunta por falla del H/11, éste emitirá un mensaje de error y:

READY

Para volver a correr el programa, revise el capítulo I de este manual.

Si hasta aquí se ha procedido correctamente, la siguiente pregunta es:

INTERVALO: INICIO, FIN?

Debe contestarse con dos números decimales separados por una coma. Estos valores corresponden a los números de la primera y última línea respectivamente del programa o segmento de programa a ensamblarse.

A continuación se pregunta:

DIRECCION INICIAL?

Debe contestarse con un valor hexadecimal, terminado con "H", caso contrario, se emitirá un mensaje de error y pedirá el número nuevamente.

Si no se pone nada, y simplemente se presiona "RETURN", el Ensamblador asume dirección cero.

Puede ocurrir que por falla del Terminal el ensamblaje se inicie antes de que el operador haya puesto una dirección inicial deseada diferente de cero, en este caso el Ensamblador ha asumido dirección cero y comienza a trabajar, condición notable porque el Disc Drive se activa. El operador puede detener la compilación con CONTROL/Z o dejar que continúe el trabajo.

Luego, hasta que el programa termina imprimiendo la palabra FIN en el listado, no se requieren mas operaciones.

Para correr nuevamente el programa hay que repetir el procedimiento (ver capítulo I) y las condiciones de este capítulo:

SCR <CR >

READY

OLD "SY: CMP851" <CR>

READY

CLEAR <CR>

READY

RUN o RUNNH <CR >

MENSAJE PTB. Si se presenta el caso de que se interrumpe el ensamblaje con la emisión del mensaje PTB, significa que hay exceso de símbolos y hay que buscar otra combinación de cantidad menor de símbolos.

Como siempre, para reiniciar el ensamblaje se deben dar los pasos desde el principio.

MENSAJES DE ERROR IMPRESOS. Los mensajes de error que se imprimen en el listado, como resultado del trabajo - del Ensamblador, se refieren al programa en Lenguaje Ensamblador 8080/8085 que se está procesando.

BIBLIOGRAFIA

1. BARDEN, W. How to Program Microcomputers, Sams, Indianapolis, 1977, 256 p.

2. DONOVAN, J. Systems Programming, Mc Graw Hill, USA. 1972 488 p.

3. FORSYTHE, A. Programación BASIC, Limusa, México, 1978, 140 p.

4. GRIES, D. Construcción de Compiladores, Paraninfo, Madrid 1975, 560 p.

5. GASS, S. Guia ilustrada para la programación lineal, Continental, México, 1972, 190 p.

6. HILBURN, J. Microcomputers/Microprocessors: Hardware, software, and Applications, Prentice-Hall, N.J., 1976, 372 p.

7. HOPPL, H. Prosa 300 para automatización de procesos, Dossat, Madrid, 1971, 208 p.

8. HUNTER, R. Automated Process Control Systems: Conceptos and Hardware, Prentice-Hall, New Jersey, 1978, 390 p.

9. KEENAN, T. Lenguajes de Diagramas de flujo, Limusa, Méexico, 1977, 588 p.

10. KLINGMAN, E. Microprocessor Systems Design, Prentice - Hall, New Jersey, 1977, 480 p.

11. LEVINE, M. Digital Theory and Practice using integrated circuits, Prentice-Hall, New Jersey, 1978, 400 p.

12. OGDIN, C. Microcomputer Dessign, Prentice-Hall, N.J., 1978, 190 p.

13. PETER, R. Introductory experiments in digital electronics and 8080A Microcomputer programming and interfasing, Sams, Indianapolis, 1978, 400 p.

14. RASKHODOFF, N. Guía del dibujante proyectista en Electrónica, Gustavo Gili, Barcelona, 1977, 632 p.

15. TANENBAUM, A. Structured Computer organization. Prentice Hall, New Jersey, 1976, 443 p.

MANUALES Y REVISTAS

1. Sistema PDP 11/03 Basic Manual del Usuario, Guayaquil, 1979.

2. Sistema PDP 11/03 Manual de operación, Guayaquil, 1979.

3. SDK-85 User's Manual, INTEL, California, 1978

4. MCS-85 User's Manual, INTEL, California, 1978

5. Mundo Electrónico (45, 46, 52, 68, 79, 84, 97); Boixareu, Barcelona, 1975-1980.