Universidad Nacional Autónoma de México

FACULTAD DE INGENIERIA

SIMULACION DE UNA CALCULADORA CON SISTEMA DE BASE EN UNA MICROCOMPUTADORA

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A:

MANUEL RODRIGUEZ PEREZ

MEXICO, D. F. 1982

I N D I C E

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

Los objetivos de esta Tésis son los siguientes:

1.- Conocer la arquitectura y organización básica de -- una calculadora electrónica a través de su simula-- ción en un programa de computadora.

2.- Desarrollar un sistema que pueda manejar números en base sesenta.

3.- Presentar un enfoque didáctico para el entendimien- to tanto del aspecto Software como de Hardware.

Estos tres objetivos se presentan en mayor detalle a -- continuación:

El diseño de sistemas o circuítos electrónicos puede -- constar de las siguientes cuatro etapas:

a) Simular el sistema en una computadora utilizando un- super lenguaje (p.e. Basic).

b) Convertir este programa a uno en lenguaje de máquina (Ensamblador) utilizando un interpretador.

c) Cargar en un dispositivo de memoria (RAM, PLA, otros) el programa (Firmware).

d) Construir un circuito dedicado específicamente para- realizar las funciones del programa (Hardware).

En algunos casos esta última etapa no se realiza y dependerá si el volúmen de ventas esperado es mayor de 200,000 unidades, en caso contrario será más costeable llegar hasta la etapa anterior en donde se maneja el sistema por medio de Firmware.

Conviene hacer notar que cuando el sistema a diseñar es muy complicado, se procede a la fabricación directa de éste utilizando simuladores, un ejemplo de esto es el computador AHR desarrollado por el IIMAS.

En esta Tésis se hará la simulación del sistema con un super lenguaje (BASIC), tratando a través de esto conocer la arquitectura y organización básica de una calculadora eletrónica.

Para explicar el segundo objetivo es necesario tomar en cuenta que en algunas operaciones de tipo comercial es indispensable hacer cálculos considerando el tiempo. La mayoría de las calculadoras electrónicas no están diseñadas para manejar números en base sesenta o en algunas otras es necesario hacer conversiones en cada operación. En esta Tésis se trata de simular una calculadora que dependiendo de la posición de un interruptor selector maneje números decimales o números de base sesenta.

El tercer objetivo se explica al considerar que el programa de simulación pudo ser más corto, sin embargo,

este se extiende ya que cada conjunto de instrucciones- trata de simular un circuito electrónico, así como la - forma de realizar las operaciones se hacen tomando en - cuenta el flujo normal que seguiría en una calculadora- electrónica. Por lo anterior, existe una analogía directa entre algunos conjuntos de instrucciones con ciertos circuitos electrónicos, lo que permite lograr complementarse el entendimiento desde los puntos de vista de - - Software y de Hardware.

Lo siguiente es un esbozo de lo que trata cada uno de - los siguientes capítulos.

El segundo capítulo trata con la historia y desarrollo- de las calculadoras, se ve desde el ábaco, pasando por- dispositivos mecánicos hasta llegar a ver algo de computadoras.

En el tercer capítulo se presenta una introducción a -- los sistemas numéricos. También se muestra la arquitectura y organización básica de una calculadora electrónica, así como el código BCD que es el indicado para usarse en una calculadora electrónica por haber gran información de entrada y salida.

En el capítulo cuarto se muestra el desarrollo a través de mapas de Karnaugh de las funciones lógicas y de control, así como también se presentan y explican los algoritmos empleados tanto para las operaciones con números decimales como con números de base sesenta.

Posteriormente se presenta en el capítulo quinto el listado del programa, así como una explicación tanto de -- las variables usadas en este como del programa en general. El programa fué simulado en una microcomputadora- Apple II utilizando el super lenguaje Basic (Apple soft) y se encuentra grabado en diskette.

El capítulo sexto explica la configuración física que - tendría la calculadora mostrando los registros y circuitos empleados así como las funciones que harían, también se indica en que parte del programa se encuentra simulado.

Se finaliza presentando las conclusiones en el capítulo séptimo.

C A P I T U L O II

PANORAMA HISTORICO DEL DESARROLLO

DE LAS CALCULADORAS.

PANORAMA HISTORICO DEL DESARROLLO DE LAS CALCULADORAS.

El deseo de contar con un dispositivo para efectuar operaciones aritméticas surgió cuando al hombre se le presentó la necesidad de poder contar el número de bienes que poseía y efectuar intercambios comerciales.

El primer instrumento que apareció para resolver esta necesidad fué el ábaco, mismo que apareció hace cerca de mil años (ver figura 2.1)

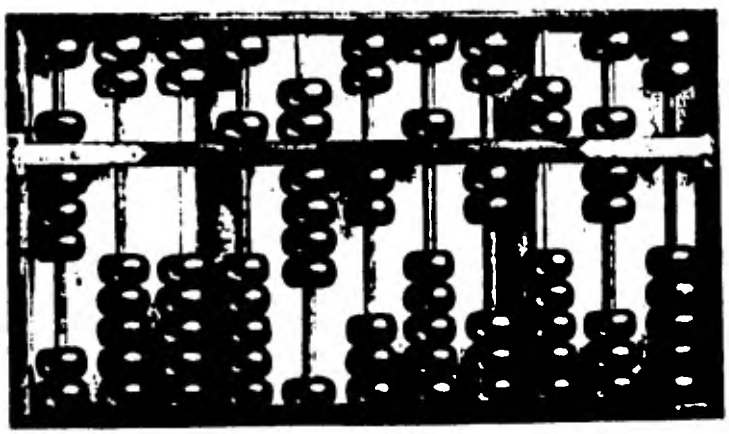


Figura 2.1

Surgieron varias versiones y su uso se extendió a todo el mundo civilizado. El ábaco consiste de una serie de varillas paralelas con cuentas cuyas posiciones, numeradas de derecha a izquierda, representan las unidades, decenas, centenas y así sucesivamente. Las cuentas tienen valor unicamente cuando se colocan junto al extremo

contrario. Cuando todas las cuentas de una varilla se encuentran agrupadas, en el extremo contrario se tiene que hacer un reagrupamiento, moviendo todas estas cuentas a su posición original y poniendo una cuenta de la siguiente posición en el extremo opuesto.

El siguiente avance en este campo lo logró Blaise Pascal, en 1642. Pascal contaba con 19 años, cuando trabajaba con su padre ayudandole en el cálculo de los impuestos de la familia. Este cálculo involucraba columnas muy grandes de sumas lo cual era un trabajo muy tedioso y con grandes fasilidades de error. Para simplificar esta labor diseño una máquina calculadora que era operada por una serie de discos, acoplados con engranes, que tenían los números del cero al nueve con su circunferencia (ver figura 2.2).

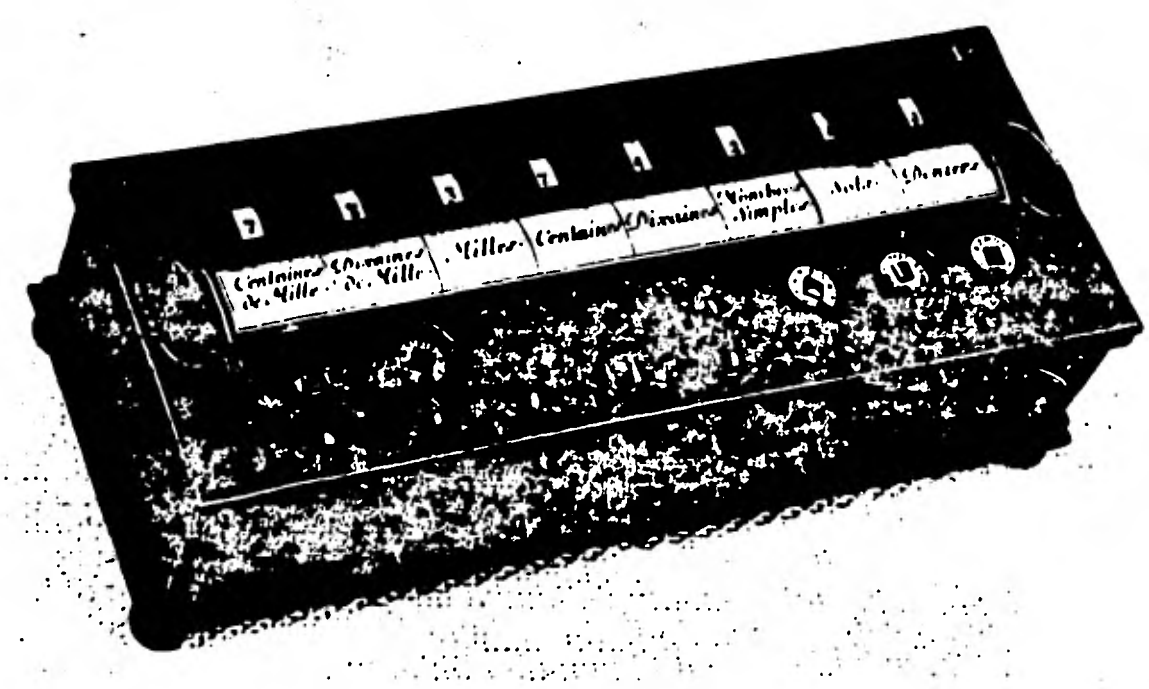


Figura 2.2

El mecanismo de engranes hacía que avanzara un dígito - en la rueda inmediata cuando la rueda contigua menos -- significativa completara una vuelta, el resultado aparecía en unos indicadores montados arriba de los discos.- La máquina de Pascal solo servía para sumar.

En 1671 Gottfried Wilhelm Leibinitz, empezó a trabajar- en una máquina que también pudiese multiplicar y divi-- dir (ver figura 2.3).

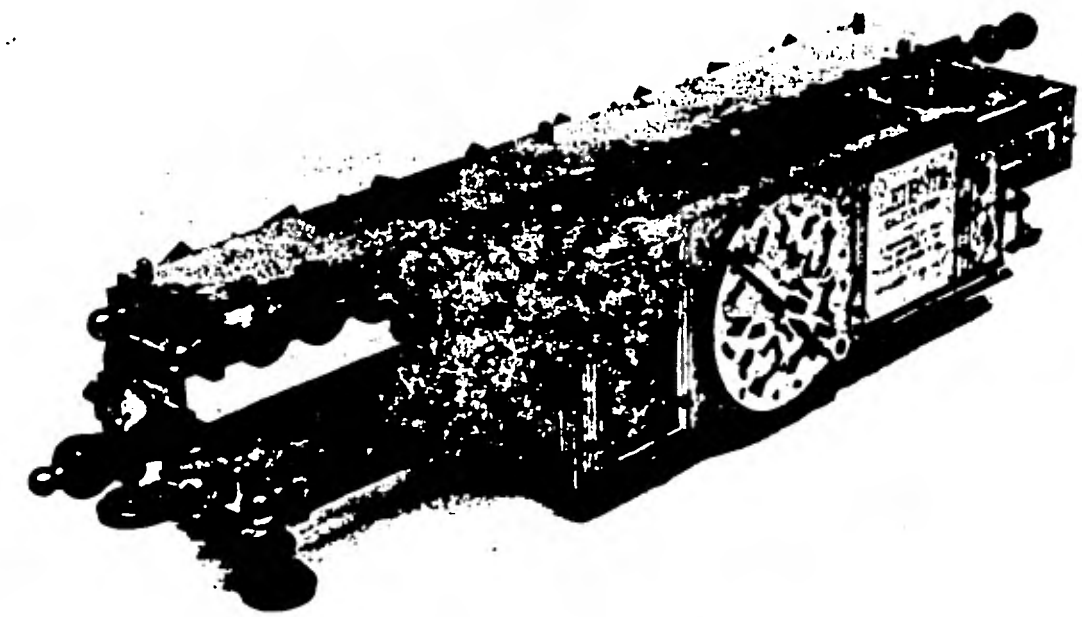


Figura 2.3

Leibinitz completo su modelo en 1694.

Debido al atraso de la tecnología en ese tiempo, tanto- la máquina de Pascal como la de Leibinitz no eran muy - precisas y no pudieron evolucionar.

Un gran avance tuvo lugar en la historia del procesa- - miento de datos, cuando en 1801 Joseph Marie Jacquard -

desarrolló una máquina controlada por tarjetas perforadas (ver figura 2.4).

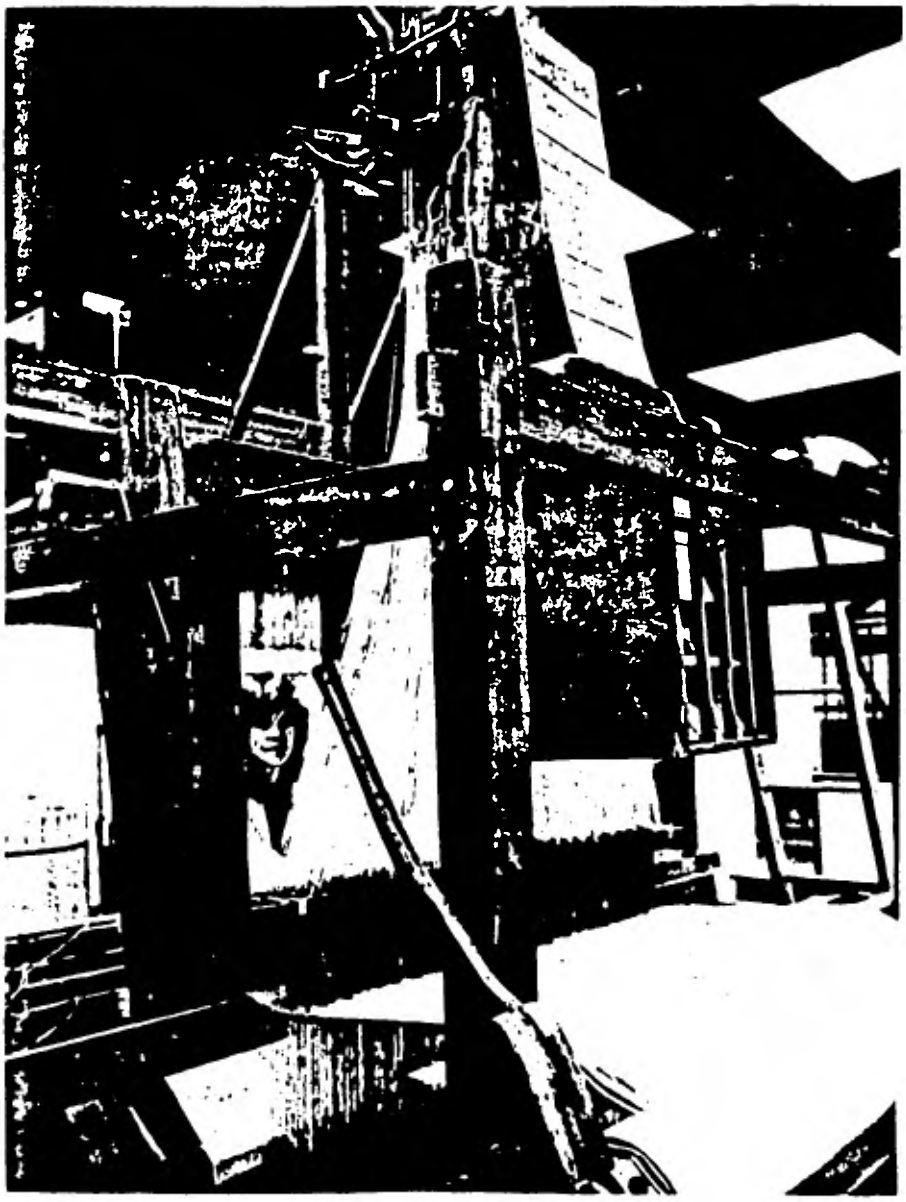


Figura 2.4

Su aplicación fué en la industria textil permitiendo hacer tejidos con patrones tan complicados como se quisiera. Cada tarjeta perforada representaba una hilera del tejido. Esta máquina continúa usandose hasta nuestros días.

En 1822 el inglés Charles Babbage, empezó a trabajar en lo que llamó la máquina de diferencias la cual podía -- producir tablas logarítmicas de seis cifras (ver figura 2.5).

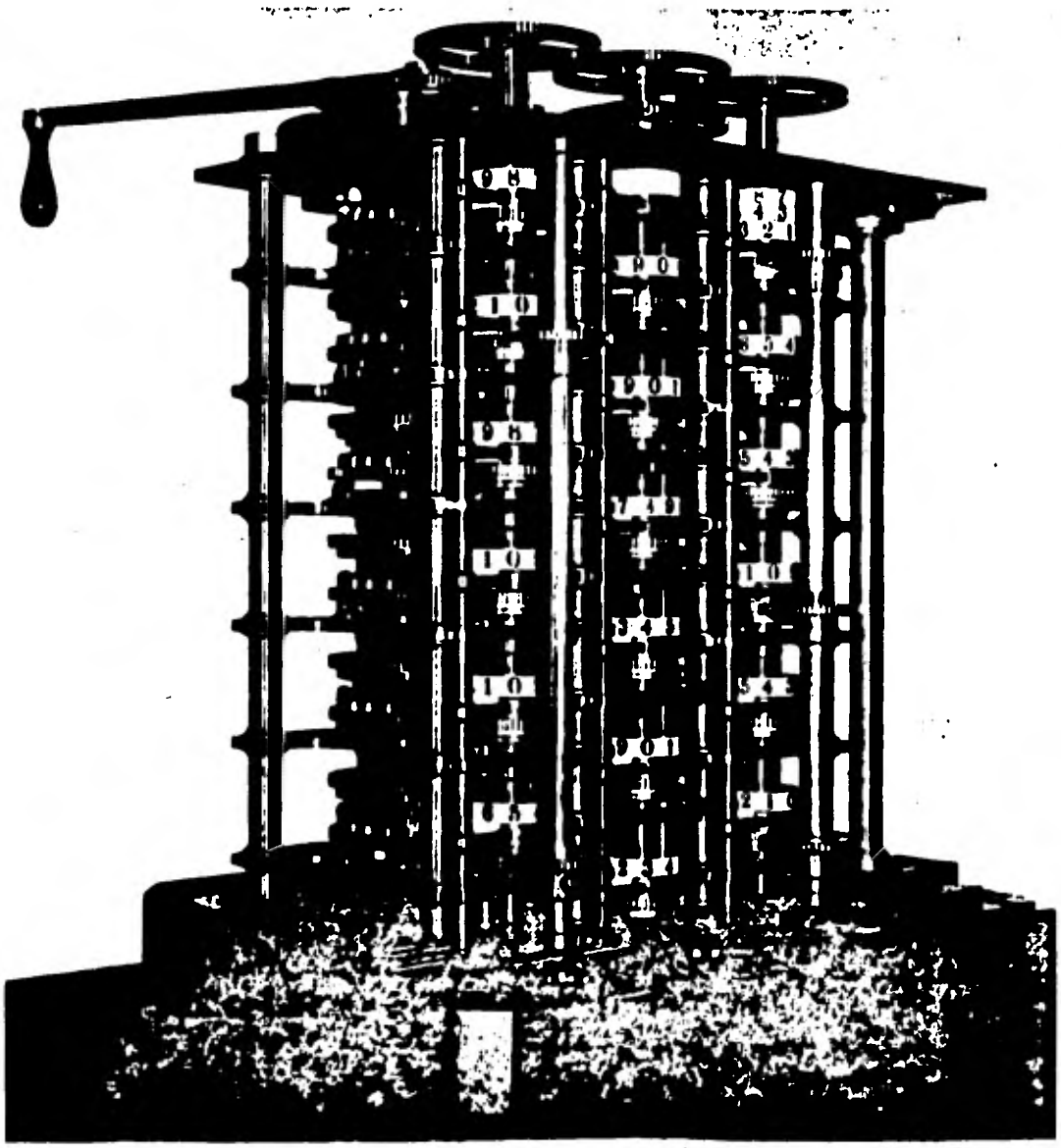


Figura 2.5

Babbage logró completar un modelo a escala de esta máquina y comenzó a trabajar en otra máquina: La máquina analítica. Esta ultima incorporaba muchas ideas que se usan en las computadoras modernas. La máquina analítica se basaba en el principio de las tarjetas perforadas para almacenar las instrucciones y permitía obtener listados en su propia unidad impresora. Sin embargo, al igual que como sus antecesores, Babbage también estaba adelantado a la tecnología de su época por lo que su máquina analítica no pudo ser implementada.

En 1890 la Oficina de Censos de Estados Unidos de America se dió cuenta de que, si procesaba manualmente los datos obtenidos de las encuentas, los resultados sólo estarían disponibles hasta cuando se efectuara el siguiente censo; esto haría que fueran inútiles los resultados obtenidos por la tardanza en el procesamiento dedatos. Fué entonces cuando el doctor Herman Hollerithque trabajaba como estadistico en la Oficina de Censosde E.U.A., diseño un sistema mecánico de clasificación(ver figura 2.6).

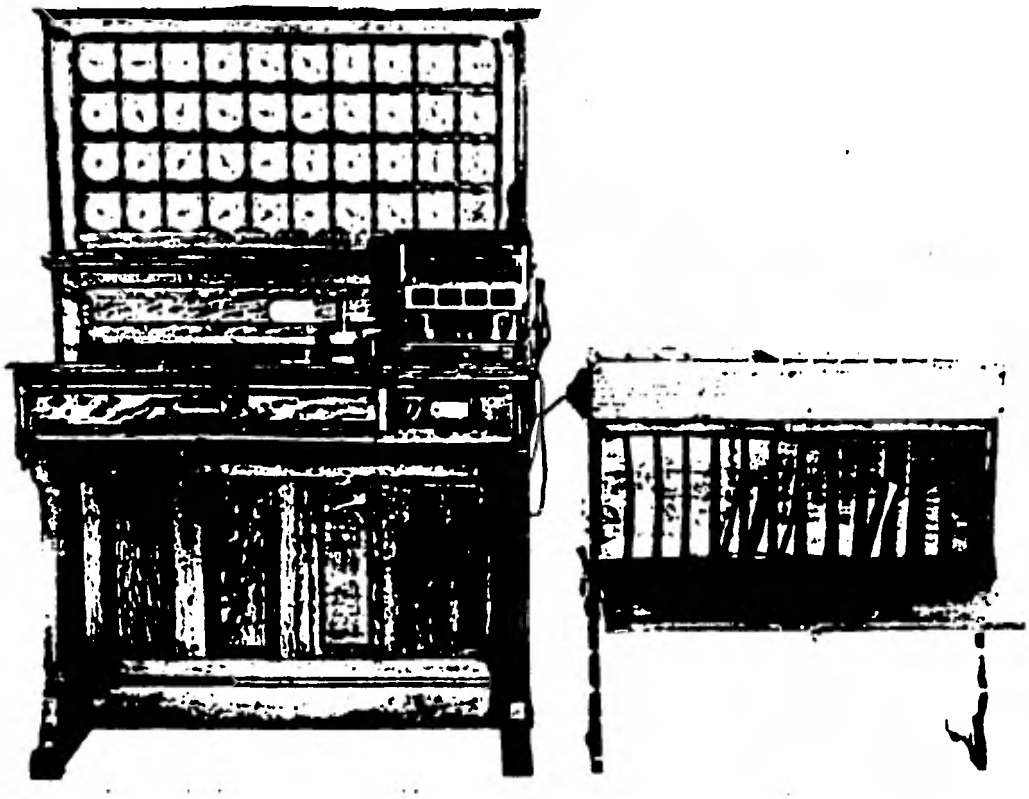


Figura 2.6

En él, se perforaban tarjetas para indicar los diferentes datos (nombre, sexo, dirección, etc.). Los datos que contenían las tarjetas eran contadas eléctricamente. Además esta máquina permitía clasificar semi-automaticamente las tarjetas.

En 1937 el Dr. Howard Aiken, empezó a desarrollar una máquina de computo automática, basandose en las ideas de Babbage, Jacguard y Hollerith. A esta máquina se le llamó Mark I y pesaba 5 toneladas, contando con 78 máquinas de sumas y cálculo e interconectada con 800 Km de alambre. El Mark I era una máquina electromecánica-

contando con relevadores e interruptores y siendo por - lo tanto lenta. A partir de este momento se lograron - grandes avances en el campo de las calculadores electro mecánicas. (ver figura 2.7).

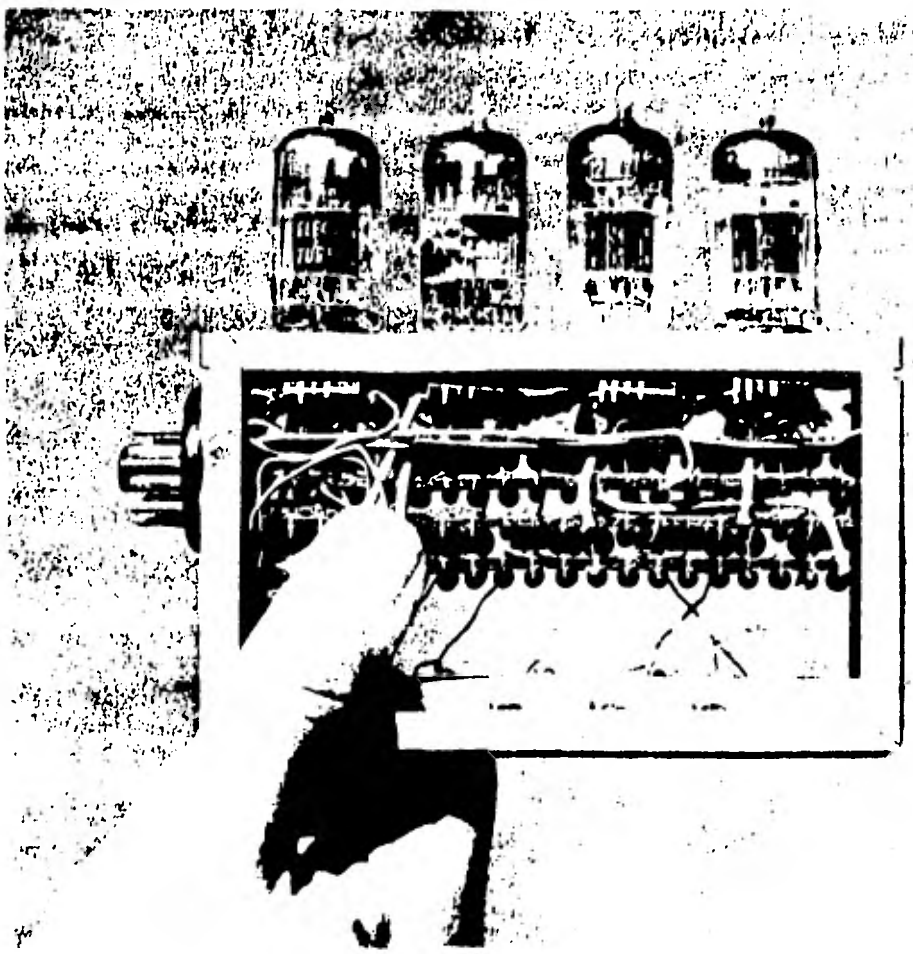


Figura 2.7

Con el desarrollo de la tecnología en el campo de la -- electrónica, hicieron su aparición las calculadoras - - electrónicas. Podemos decir que con el desarrollo del- transistor en 1952 (ver figura 2.8).

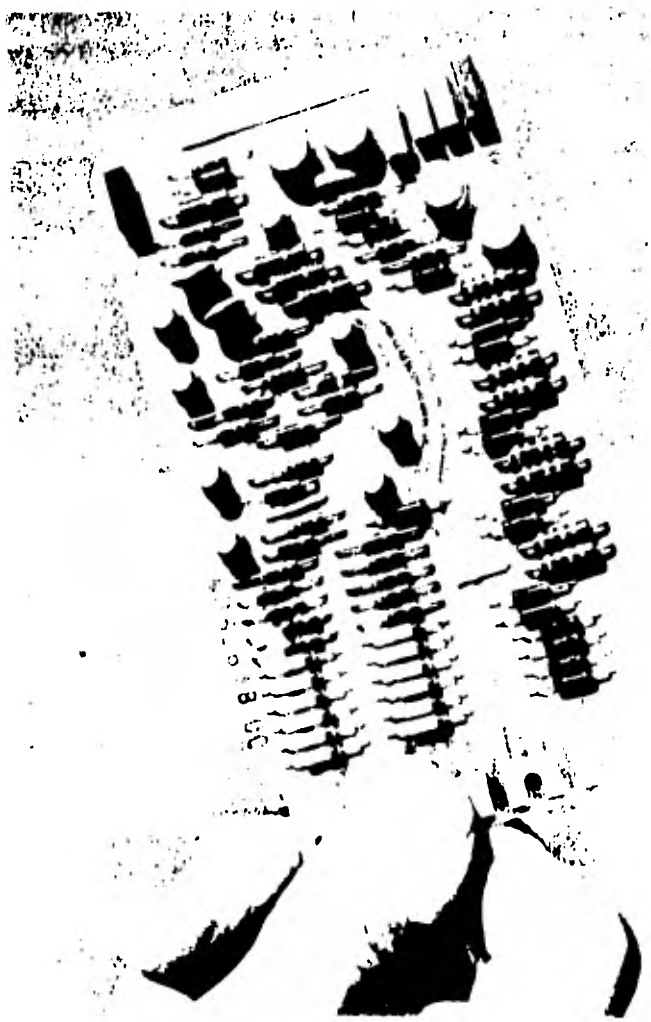
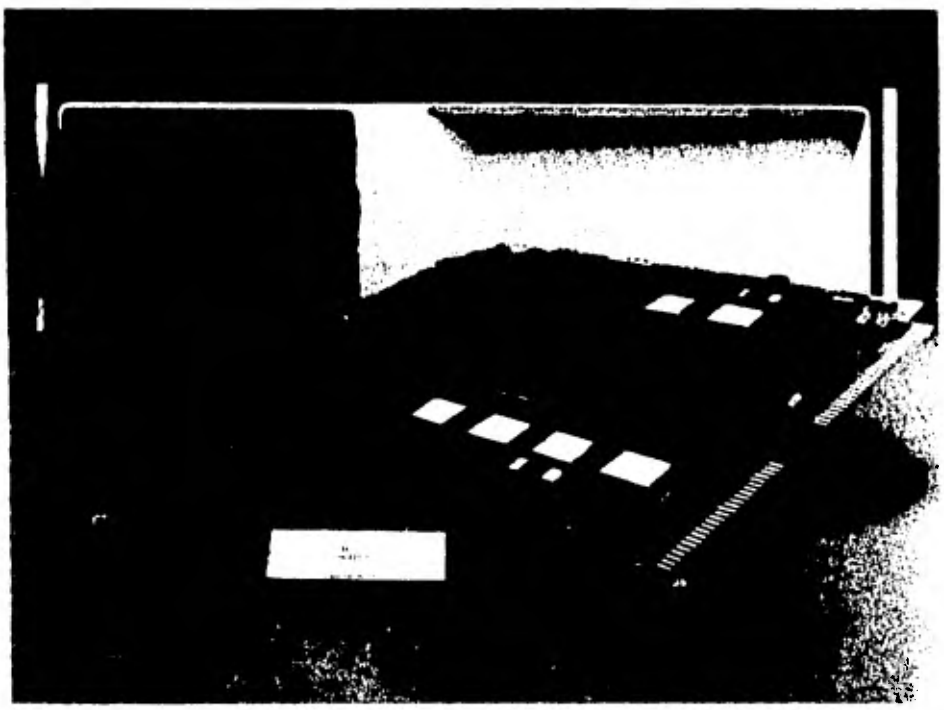


Figura 2.8

Comienzan las claculadoras electrónicas, aunque su precio era bastante alto y ocupaban demasiado espacio. El invento del circuito integrado en 1960 por Kelby (norteamericano) permitió reducir el tamaño y consumo de estas, así como abatir los costos, permitiendo la aparición de las calculadoras portátiles (ver figura 2.9).



El circuito integrado permitió también construir calculadoras electrónicas más poderosas, incluyendo ahora las calculadoras portátiles programables. En términos generales.

se puede considerar que en la historia de la humanidad-han existido 4 tipos de calculadoras:

1).- Manuales - ábaco.

2).- Mecánicas - a base de levas, varillas y otros elementos mecánicos.

3).- Electromecánicas - igual que las mecánicas pero -- además circuitos eléctronicos.

4).- Electrónicas - principalmente compuestas por cir-- cuitos integrados.

Los circuitos integrados con alto grado de integración, cuentan con la mayoría de las funciones que realizaban-las primeras computadoras electrónicas, dicho circuito-es el microprocesador. Con este último, los costos han disminuido y la flexibilidad de aplicaciones y usos ha-aumentado.

Las siguientes graficas nos indican la relación entre - costo y capacidad de memoria según ha evolucionado la - tecnología (ver figura 2.10).

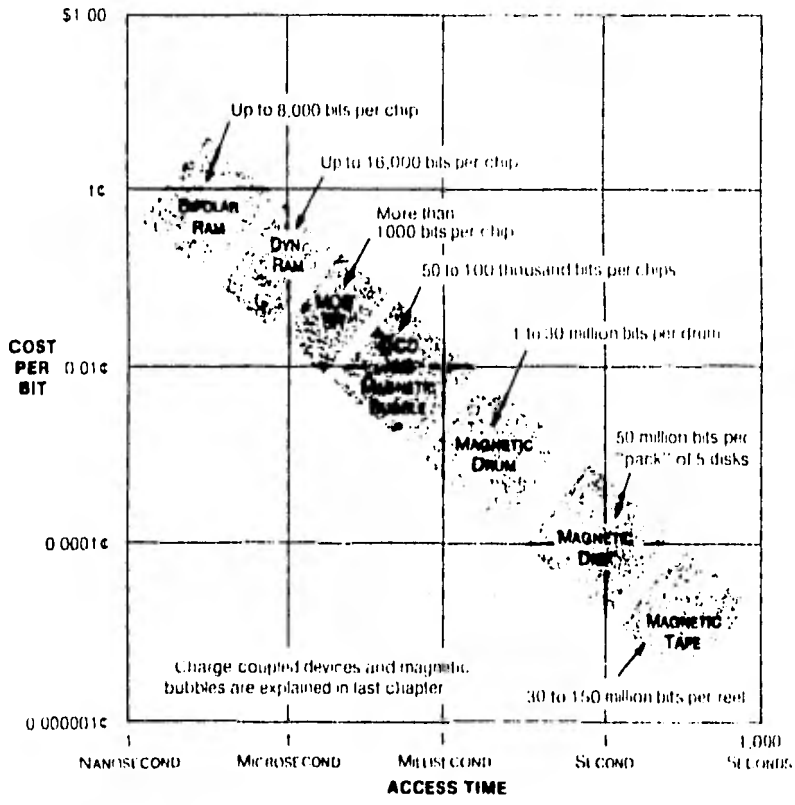


Figura 2.10

Esta tecnología será la que consideraré para implemen-- tar el proyecto motivo de este trabajo.

C A P I T U L O III

ARQUITECTURA DE UNA CALCULADORA

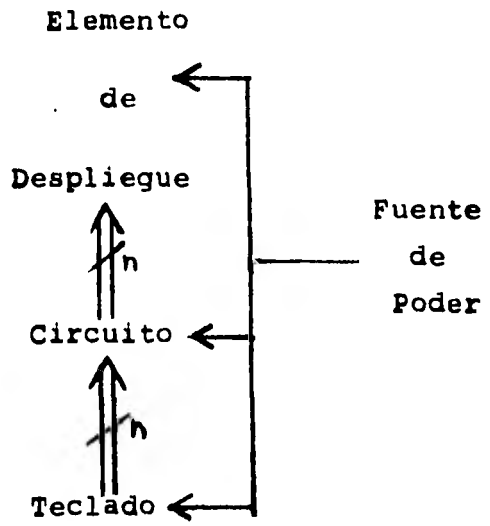
Y

SISTEMAS DE NUMERACION

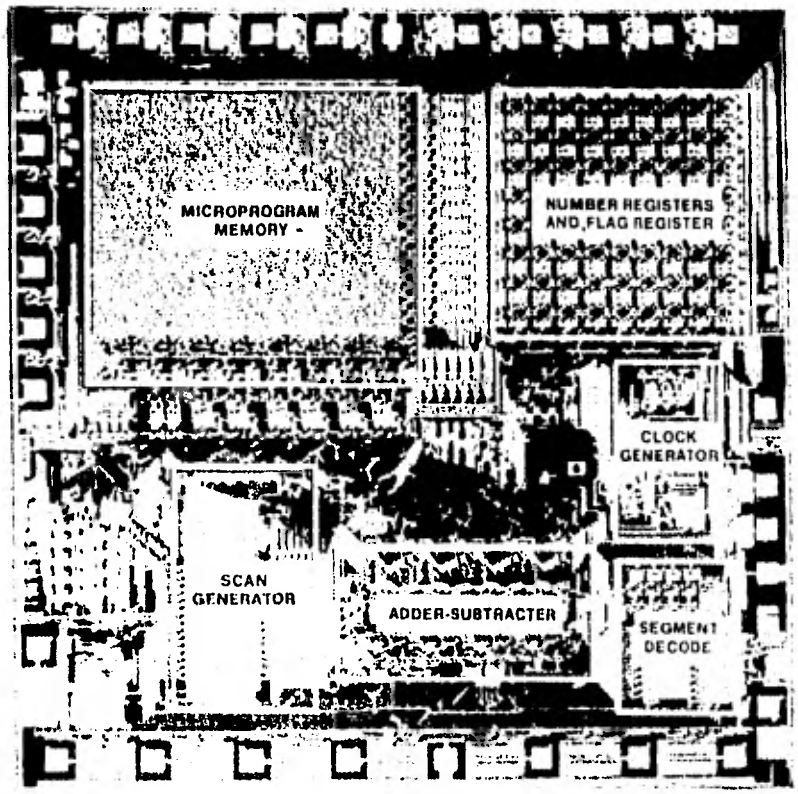
ARQUITECTURA DE UNA CALCULADORA

En este capítulo se describe la estructura de una calculadora electrónica; primero desde un punto de vista funcional y posteriormente estructural, para concluir con un análisis de sus componentes.

En términos generales podemos decir que una calculadora está formada por: Un elemento de despliegue, un teclado, un circuito y la fuente de poder. Esta última en la mayoría de las calculadoras electrónicas portátiles inteligentes es una pila. La representación gráfica sería:



y su apariencia física (ver figura 3.2).



Las líneas dobles, asociadas con un cierto número (en - este caso representado por n) nos indican los dígitos - de información que se están manejando simultáneamente,- es decir nos indica la longitud de la palabra en el sis tema de numeración empleado. En general se tiene que - la longitud de la palabra puede variar de calculadora - a calculadora y está será mayor en los casos en que se- requiera de mayor precisión.

Existen dos sistemas muy empleados para el teclado, --- Uno está compuesto por un conjunto de interruptores que son barridos y analizados varias veces por segundo. Cada interruptor representa una tecla y se encuentran - - arreglados matricialmente (ver figura 3.3)

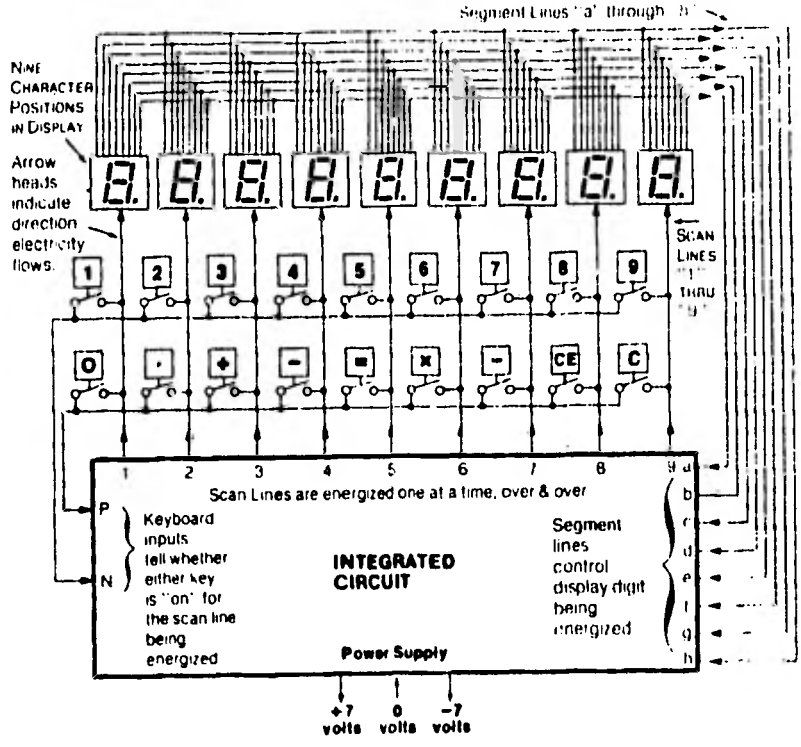


Figura 3.3

Si se tiene un arreglo de nueve columnas y dos renglo-- nes, como en la figura anterior, el circuito generador- de barrido barrerá todas y cada una de las nueve columnas y en el momento que detecte que una tecla (interruptor) ha sido accionada, generará un pulso en el renglón correspondiente a esa tecla. Nosotros podemos identificar la tecla que fué accionada con la siguiente información:

1) ¿Qué columna se estaba barriendo en ese instante?

2) ¿Qué renglón generá el pulso correspondiente?.

El sistema descrito presenta problemas de rapidez y requiere de una gran sincronización entre los circuitos.

Un sistema que elimina los problemas anteriores es el-- que está basado en interrupciones, es decir la unidad - de procesamiento estará realizando sus actividades normales y cada vez que esté disponible información de entrada, se encenderá una bandera y al detectar la unidad- de control ésta condición, hará una busqueda en la ma-- tríz de interruptores para determinar la tecla accionada.

Consideremos el medio de despliegue.

Podemos representar cualquier dígito decimal por medio- de 7 segmentos y un octavo para el punto decimal, tal - como se muestra en la figura 3.4 .

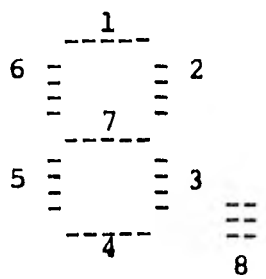


Figura 3.4

Cada segmento está formado por un diodo emisor de luz - (LED)

Para el elemento de despliegue contamos con dos circuitos; el de barrido (tal como el del teclado) y el decodificador de segmentos (ver figura 3.5).

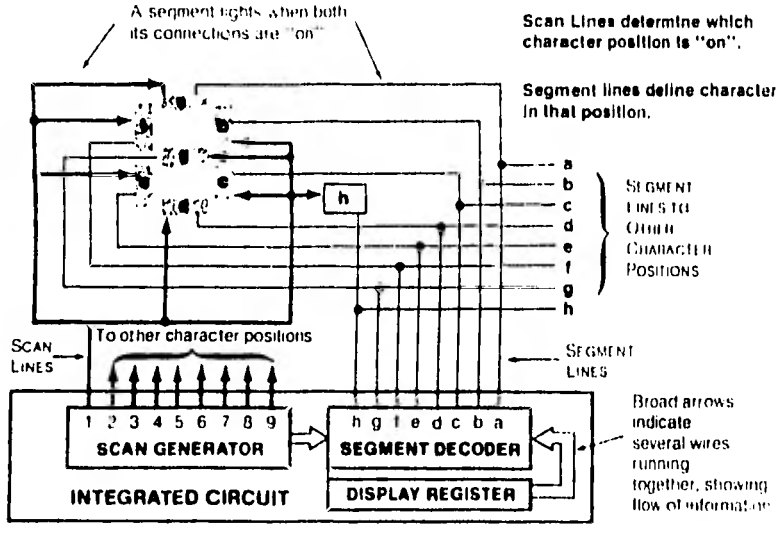


Figura 3.5

El primero tiene 'x' número de salidas llamadas líneas-de barrido, cada una conectada a un caracter. El decodificador de segmentos nos determina los segmentos quedeben encenderse para formar el número deseado. Solo - mencionaremos que ésta información le es transmitida a- este último circuito desde el registro de despliegue. - Debe existir una sincronización entre el circuito de -- barrido y el decodificador, de tal manera que cuando el primero active un caracter el decodificador contenga la información correspondiente a ese caracter. Las funciones anteriores pueden realizarse a base de multiplexo-- res y decodificadores. Un circuito típico es el SN74LS 48 (decodificador de BCD a 7 segmentos), cuya tabla defunciones y diagrama funcional se presentan en las figuras: 3.6 y 3.7 .

'48, 'LS48
FUNCTION TABLE

DECIMAL OR FUNCTION	INPUTS						BI/RBO†	OUTPUTS							NOTE
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	H	H	H	H	H	H	L	1
1	H	X	L	L	L	H	H	L	H	H	L	L	L	L	
2	H	X	L	L	H	L	H	H	H	L	H	H	L	H	
3	H	X	L	L	H	H	H	H	H	H	H	L	L	H	
4	H	X	L	H	L	L	H	L	H	H	L	L	H	H	
5	H	X	L	H	L	H	H	H	L	H	H	L	H	H	
6	H	X	L	H	H	L	H	L	L	H	H	H	H	H	
7	H	X	L	H	H	H	H	H	H	H	L	L	L	L	
8	H	X	H	L	L	L	H	H	H	H	H	H	H	H	
9	H	X	H	L	L	H	H	H	H	H	L	L	H	H	
10	H	X	H	L	H	L	H	L	L	L	H	H	L	H	
11	H	X	H	L	H	H	H	L	L	H	H	L	L	H	
12	H	X	H	H	L	L	H	L	H	L	L	L	H	H	
13	H	X	H	H	L	H	H	H	L	L	H	L	H	H	
14	H	X	H	H	H	L	H	L	L	L	H	H	H	H	
15	H	X	H	H	H	H	H	L	L	L	L	L	L	L	
BI	X	X	X	X	X	X	L	L	L	L	L	L	L	L	2
RBI	H	L	L	L	L	L	L	L	L	L	L	L	L	L	3
LT	L	X	X	X	X	X	H	H	H	H	H	H	H	H	4

H = high level, L = low level, X = irrelevant

NOTES: 1. The blanking input (BI) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple-blanking input (RBI) must be open or high, if blanking of a decimal zero is not desired.

2. When a low logic level is applied directly to the blanking input (BI), all segment outputs are low regardless of the level of any other input.

3. When ripple-blanking input (RBI) and inputs A, B, C, and D are at a low level with the lamp-test input high, all segment outputs go low and the ripple-blanking output (RBO) goes to a low level (response condition).

4. When the blanking input/ripple-blanking output (BI/RBO) is open or held high and a low is applied to the lamp-test input, all segment outputs are high.

†BI/RBO is wire-AND logic serving as blanking input (BI) and/or ripple-blanking output (RBO).

'49, 'LS49
FUNCTION TABLE

DECIMAL OR FUNCTION	INPUTS					OUTPUTS							NOTE
	D	C	B	A	BI	a	b	c	d	e	f	g	
0	L	L	L	L	H	H	H	H	H	H	H	L	1
1	L	L	L	H	H	L	H	H	L	L	L	L	
2	L	L	H	L	H	H	H	L	H	H	L	H	
3	L	L	H	H	H	H	H	H	H	L	L	H	
4	L	H	L	L	H	L	H	H	L	L	H	H	
5	L	H	L	H	H	H	L	H	H	L	H	H	
6	L	H	H	L	H	L	L	H	H	H	H	H	
7	L	H	H	H	H	H	H	H	L	L	L	L	
8	H	L	L	L	H	H	H	H	H	H	H	H	
9	H	L	L	H	H	H	H	H	L	L	H	H	
10	H	L	H	L	H	L	L	L	H	H	L	H	
11	H	L	H	H	H	L	L	H	H	L	L	H	
12	H	H	L	L	H	L	H	L	L	L	H	H	
13	H	H	L	H	H	H	L	L	H	L	H	H	
14	H	H	H	L	H	L	L	L	H	H	H	H	
15	H	H	H	H	H	L	L	L	L	L	L	L	
BI	X	X	X	X	L	L	L	L	L	L	L	L	2

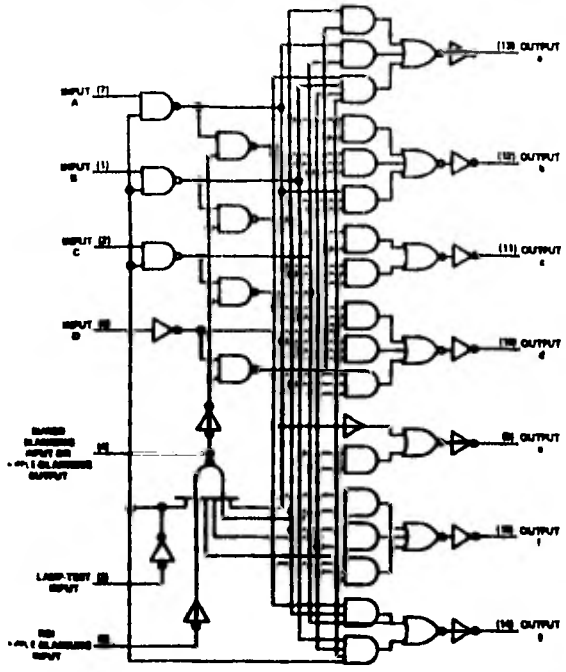
H = high level, L = low level, X = irrelevant

NOTES: 1. The blanking input (BI) must be open or held at a high logic level when output functions 0 through 15 are desired.

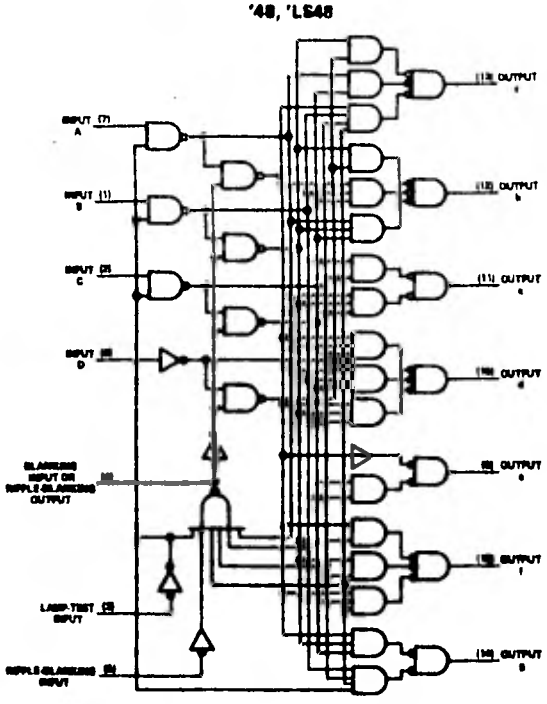
2. When a low logic level is applied directly to the blanking input (BI), all segment outputs are low regardless of the level of any other input.

functional block diagrams

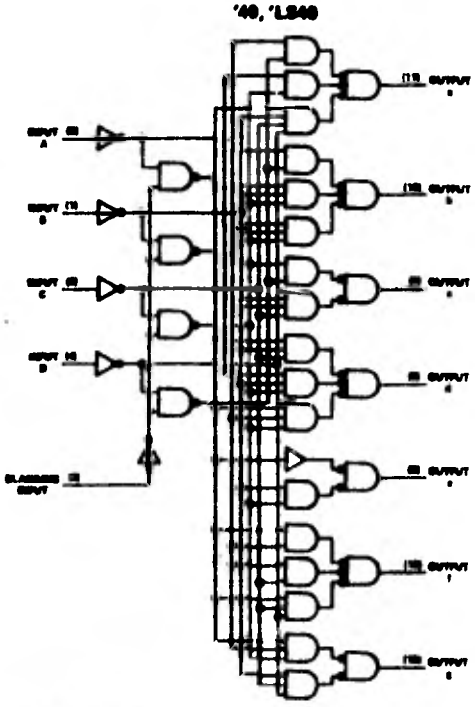
'46A, '47A, 'L46, 'L47, 'LS47



'48, 'LS48



'49, 'LS49



Este circuito cuenta con cuatro entradas para el código BCD y 7 salidas que van a los caracteres de despliegue. Además se cuenta con otras tres entradas para probar -- los elementos de despliegue (en este caso Vleds) y bo-- rrar ceros o suprimirlos.

La parte modular de la calculadora es lo que hemos llamado el circuito. Este es en realidad, un sistema inteligente compuesto por un conjunto de circuitos que realizan diferentes funciones.

El circuito puede estar compuesto en una sola pastilla, tal como se mostró en la figura 3.2, o puede haber un - circuito para cada función e inclusive pueden existir - en elementos discretos (transitores, diodos, etc.).

Independientemente de como está implementado el circuito, éste deberá contar con las funciones que se muestra en la figura 3.8 .

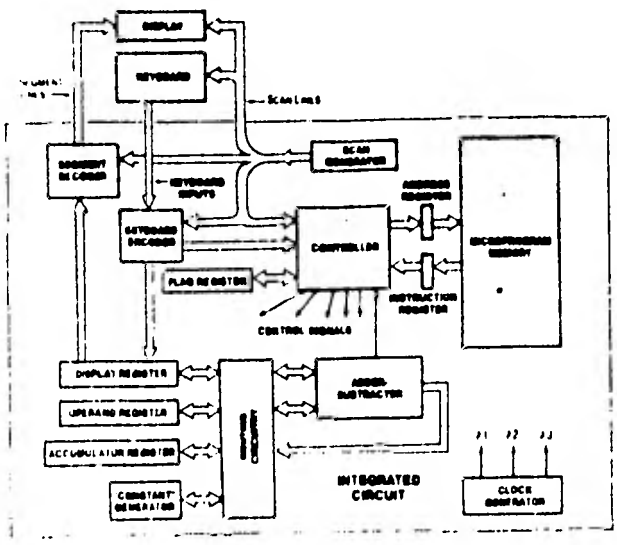


Figura 3.8

Analicemos cada uno de los elementos mostrados en la figura anterior.

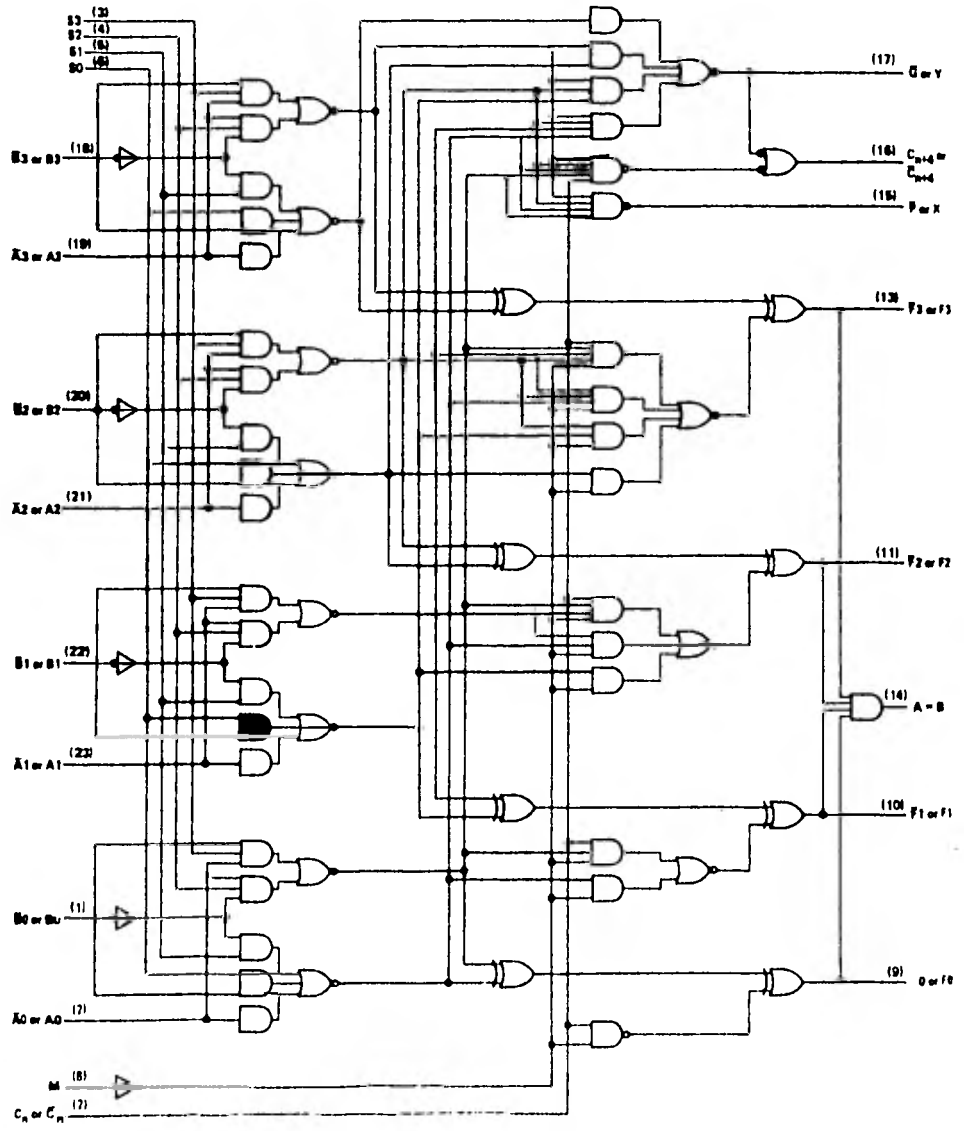
Una unidad de control se encarga de iniciar secuencias de micro-operaciones. Una micro-operación es una operación elemental que puede ser realizada en un pulso de reloj. Una unidad de control puede implementarse por Hardware o por una memoria de control. En este último caso la unidad de control interpretará las micro-instrucciones almacenadas en la memoria del micro-programa y en base a estas nos activa los diferentes elementos en determinados instantes de tiempo.

La memoria para el micro-programa, contiene todas las micro-instrucciones necesarias para indicarle al sistema como hacer las operaciones y algunas otras funciones. El control y la memoria se comunican a través del registro de direcciones y del registro de instrucciones. La unidad de control, a través del contador del programa, o de una dirección indicada en una transferencia, cargará el registro de dirección. La micro-instrucción especificada por la dirección anterior, le será transmitida a la unidad de control a través del registro de instrucciones. El tiempo que toma ejecutar una micro-instrucción está definido como un ciclo de instrucción.

El reloj o generador de pulsos nos sirve para sincronizar todas las actividades en el sistema. Este circuito debe proporcionar las señales necesarias para el sistema.

La unidad aritmética lógica nos permite realizar todas-las operaciones (inclusive multiplicaciones y divisio--nes a base de repeticiones) y también ciertas funciones lógicas como: Comparaciones, corrimientos a la izquier da y a la derecha, rotación, etc. El diagrama funcio--nal de una unidad aritmética lógica típica se muestra-en la figura 3.9

functional block diagram



Este circuito permite realizar 16 operaciones aritméticas binarias con dos palabras de 4 bits. Las funciones a realizar se determinan con las cuatro líneas de selección. Este circuito permite trabajar con carry look -- ahead, además de contener las salidas correspondientes- para el carry.

El acumulador es uno de los registros más importantes.- Básicamente este es un registro de trabajo que nos irá- acumulando o guardando los resultados de las operacio-- nes. Hay algunos acumuladores que cuentan con caracte- rísticas muy especiales que se seleccionan de acuerdo - a las necesidades del mismo, algunos otros se encuen- - tran integrados dentro de la unidad aritmética lógica.

El registro de despliegue contiene la información que - se mostrará en el elemento de despliegue. Este regis-- tro puede ser del tipo común, formado únicamente por -- compuertas lógicas y flip flops.

El registro de operación contiene el código de la operación que la calculadora deberá realizar.

El registro de cociente y multiplicación se emplea para las operaciones de multiplicación y división. Un cir-- cuito típico de registro que podría usarse para los casos anteriores se muestra en la figura 3.11.

y la tabla de funciones en la figura 3.10.

signal designations (continued)

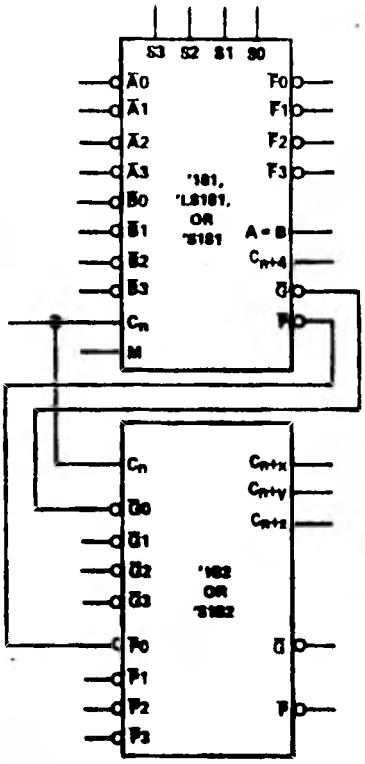


FIGURE 1
(Use with Table 1)

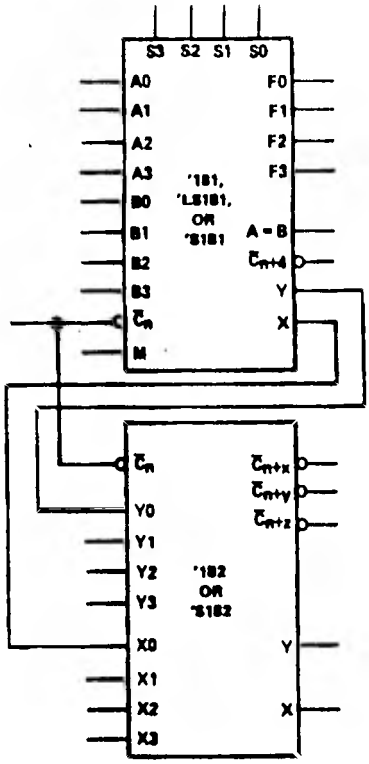


FIGURE 2
(Use with Table 2)

TABLE 1

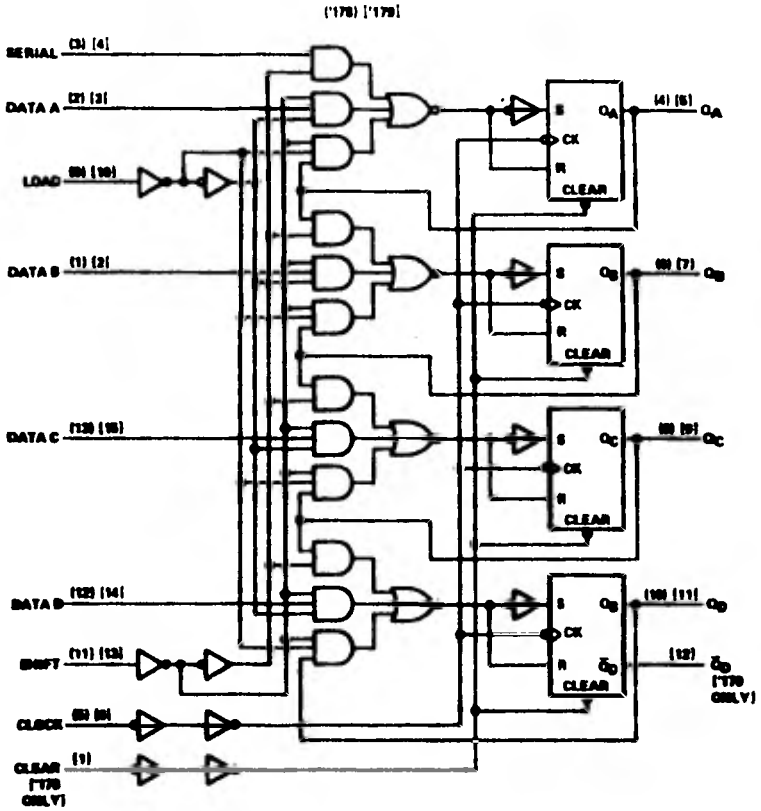
SELECTION				M = H	ACTIVE-LOW DATA M = L: ARITHMETIC OPERATIONS	
S3	S2	S1	S0	LOGIC FUNCTIONS	C_n = L (no carry)	C_n = H (with carry)
L	L	L	L	$F = \overline{A}$	F = A MINUS 1	F = A
L	L	L	H	$F = \overline{AB}$	F = AB MINUS 1	F = AB
L	L	H	L	$F = \overline{A} + B$	F = $A\overline{B}$ MINUS 1	F = $A\overline{B}$
L	L	H	H	F = 1	F = MINUS 1 (2's COMPL)	F = ZERO
L	H	L	L	$F = \overline{A + B}$	F = A PLUS (A + $\overline{B}$)	F = A PLUS (A + $\overline{B}$) PLUS 1
L	H	L	H	$F = \overline{B}$	F = AB PLUS (A + $\overline{B}$)	F = AB PLUS (A + $\overline{B}$) PLUS 1
L	H	H	L	$F = A \oplus B$	F = A MINUS B MINUS 1	F = A MINUS B
L	H	H	H	$F = A + \overline{B}$	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
H	L	L	L	$F = \overline{A}B$	F = A PLUS (A + B)	F = A PLUS (A + B) PLUS 1
H	L	L	H	$F = A \odot B$	F = A PLUS B	F = A PLUS B PLUS 1
H	L	H	L	F = B	F = $A\overline{B}$ PLUS (A + B)	F = $A\overline{B}$ PLUS (A + B) PLUS 1
H	L	H	H	F = A + B	F = (A + B)	F = (A + B) PLUS 1
H	H	L	L	F = 0	F = A PLUS A*	F = A PLUS A PLUS 1
H	H	L	H	F = $A\overline{B}$	F = AB PLUS A	F = AB PLUS A PLUS 1
H	H	H	L	F = AB	F = $A\overline{B}$ PLUS A	F = $A\overline{B}$ PLUS A PLUS 1
H	H	H	H	F = A	F = A	F = A PLUS 1

TABLE 2

SELECTION				M = H	ACTIVE-HIGH DATA M = L: ARITHMETIC OPERATIONS	
S3	S2	S1	S0	LOGIC FUNCTIONS	$\overline{C}_n$ = H (no carry)	$\overline{C}_n$ = L (with carry)
L	L	L	L	$F = \overline{A}$	F = A	F = A PLUS 1
L	L	L	H	$F = \overline{A + B}$	F = A + B	F = (A + $\overline{B}$) PLUS 1
L	L	H	L	$F = \overline{A}B$	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
L	L	H	H	F = 0	F = MINUS 1 (2's COMPL)	F = ZERO
L	H	L	L	$F = \overline{AB}$	F = A PLUS $A\overline{B}$	F = A PLUS $A\overline{B}$ PLUS 1
L	H	L	H	$F = \overline{B}$	F = (A + B) PLUS $A\overline{B}$	F = (A + B) PLUS $A\overline{B}$ PLUS 1
L	H	H	L	$F = A \oplus B$	F = A MINUS B MINUS 1	F = A MINUS B
L	H	H	H	$F = A\overline{B}$	F = $A\overline{B}$ MINUS 1	F = $A\overline{B}$
H	L	L	L	$F = \overline{A} + B$	F = A PLUS $A\overline{B}$	F = A PLUS $A\overline{B}$ PLUS 1
H	L	L	H	$F = A \odot B$	F = A PLUS B	F = A PLUS B PLUS 1
H	L	H	L	F = B	F = (A + $\overline{B}$) PLUS $A\overline{B}$	F = (A + $\overline{B}$) PLUS $A\overline{B}$ PLUS 1
H	L	H	H	F = AB	F = $A\overline{B}$ MINUS 1	F = $A\overline{B}$
H	H	L	L	F = 1	F = A PLUS A*	F = A PLUS A PLUS 1
H	H	L	H	F = A + $\overline{B}$	F = (A + B) PLUS A	F = (A + B) PLUS A PLUS 1
H	H	H	L	F = A + B	F = (A + $\overline{B}$) PLUS A	F = (A + $\overline{B}$) PLUS A PLUS 1
H	H	H	H	F = A	F = A MINUS 1	F = A

*Each bit is shifted to the next more significant position.

Este registro permite corrimiento hacia la derecha o -- izquierda, cargar o retener cierta información, además- como cuenta con lógica de tri-state permite conectarse- directamente al bus. Su tabla de funciones se muestra- en la figura 3.12 .



'170, '179†
FUNCTION TABLE

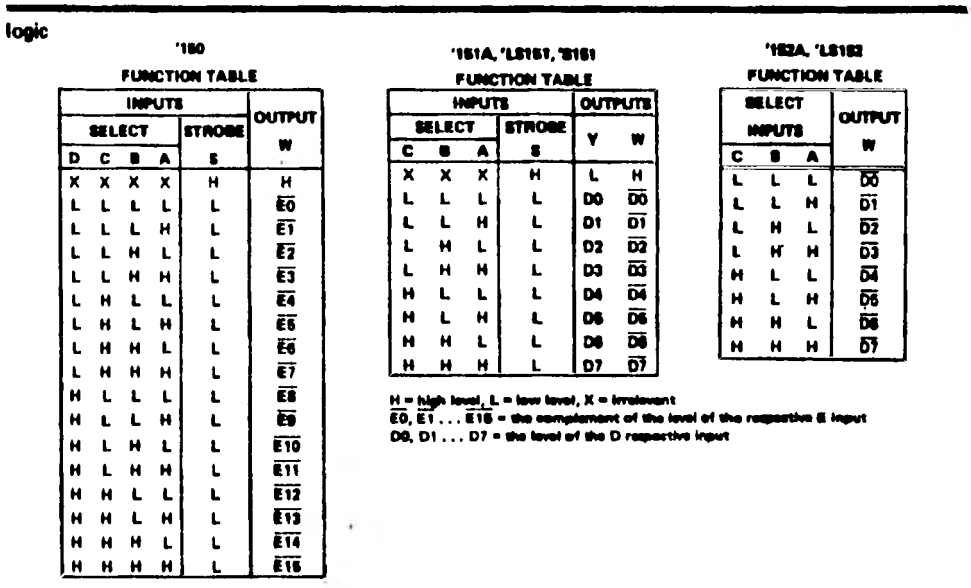
INPUTS									OUTPUTS				
CLEAR†	SHIFT	LOAD	CLOCK	SERIAL	PARALLEL A	B	C	D	Q_A	Q_B	Q_C	Q_D	$\overline{Q}_D$†
L	X	X	X	X	X	X	X	X	L	L	L	L	H
H	X	X	H	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}	$\overline{Q}_{D0}$
H	L	L	↓	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}	$\overline{Q}_{D0}$
H	L	H	↓	X	a	b	c	d	a	b	c	d	$\overline{d}$
H	H	X	↓	H	X	X	X	X	H	Q_{An}	Q_{Bn}	Q_{Cn}	$\overline{Q}_{Cn}$
H	H	X	↓	L	X	X	X	X	L	Q_{An}	Q_{Bn}	Q_{Cn}	$\overline{Q}_{Cn}$

† The columns for clear, $\overline{Q}_D$, and the top line of the table apply for the '179 only.

También existe una serie de registros de bandera, que - se utilizan para indicar ciertas condiciones como saturación en sumas, multiplicaciones y divisiones, indicación de que una tecla ha sido accionada, terminación -- de una operación y otras, dependiendo de las necesidades y propósitos del sistema.

Para lograr la comunicación entre los diferentes circuitos y registros puede ser necesario el uso de multiplexores. Este circuito nos permite seleccionar el destino de cierta información. Un circuito típico de un multiplexores se muestra en la figura 3.13.

Y su tabla de funciones en la figura 3.14.



logic

'150

FUNCTION TABLE

INPUTS					OUTPUT
SELECT				STROBE	
D	C	B	A	S	W
X	X	X	X	H	H
L	L	L	L	L	$\overline{E0}$
L	L	L	H	L	$\overline{E1}$
L	L	H	L	L	$\overline{E2}$
L	L	H	H	L	$\overline{E3}$
L	H	L	L	L	$\overline{E4}$
L	H	L	H	L	$\overline{E5}$
L	H	H	L	L	$\overline{E6}$
L	H	H	H	L	$\overline{E7}$
H	L	L	L	L	$\overline{E8}$
H	L	L	H	L	$\overline{E9}$
H	L	H	L	L	$\overline{E10}$
H	L	H	H	L	$\overline{E11}$
H	H	L	L	L	$\overline{E12}$
H	H	L	H	L	$\overline{E13}$
H	H	H	L	L	$\overline{E14}$
H	H	H	H	L	$\overline{E15}$

'151A, 'LS151, 'S151

FUNCTION TABLE

INPUTS				OUTPUTS	
SELECT			STROBE		
C	B	A	S	Y	W
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

H = high level, L = low level, X = irrelevant
$\overline{E0}$, $\overline{E1}$ . . . $\overline{E15}$ = the complement of the level of the respective E input
D0, D1 . . . D7 = the level of the D respective input

'152A, 'LS152

FUNCTION TABLE

SELECT INPUTS			OUTPUT
C	B	A	W
L	L	L	$\overline{D0}$
L	L	H	$\overline{D1}$
L	H	L	$\overline{D2}$
L	H	H	$\overline{D3}$
H	L	L	$\overline{D4}$
H	L	H	$\overline{D5}$
H	H	L	$\overline{D6}$
H	H	H	$\overline{D7}$

El circuito decodificador de teclas generará el código correspondiente a la tecla que haya sido oprimida. Tanto el decodificador de segmentos como el generador de barrido ya han sido descritos y estan relacionados con el elemento de despliegue y el teclado.

Finalmente se cuenta con una fuente de suministro de energía conocida como de alimentación o de poder. Esta podrá estar compuesta por un circuito de suministro o por una pila y en ocasiones un cargador para recargar estas. En la figura 3.15

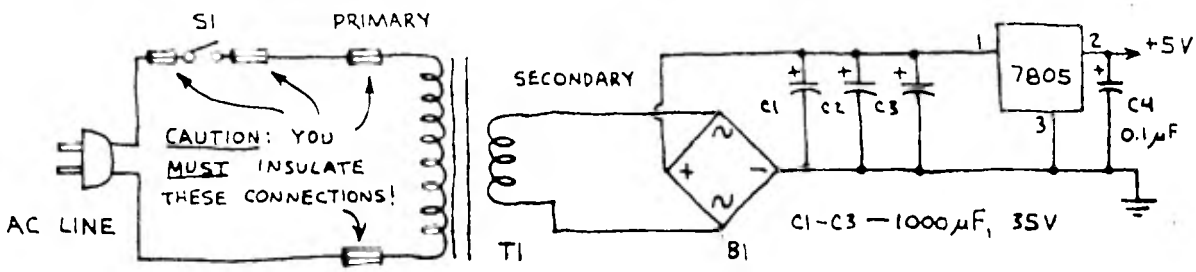

S1
PRIMARY
CAUTION: YOU
MUST INSULATE
THESE CONNECTIONS!
AC LINE
SECONDARY
T1
B1
C1
C2
C3
7805
+5V
C4
0.1μF
C1-C3 — 1000μF, 35V

Figura 3.15

se muestra un circuito de suministro. El transformador permite reducir el voltaje, posteriormente los diodos - lo rectifican y el regulador lo mantiene constante.

Refiriendonos a la figura 3.8 mostraremos el flujo de - información dentro de una calculadora.

Al oprimir una tecla el generador de barrido la detecta y el codificador del teclado la convierte a un código.- Esta información pasa a la unidad de control que a su - vez se comunica con la memoria del micro-programa. Suponiendo que la tecla oprimida sea un número, entonces- la unidad de control emitirá una señal para que permita encender un registro de bandera que indicará que hay -- información presente. A su vez ésta información será - desplegada en el elemento de despliegue. Al oprimir -- una tecla de operación, por ejemplo suma (+), sucederá- lo mismo, solo cambiando en el sentido de que la bandera prendida será otra y que el elemento de despliegue contendrá lo mismo que tenía anteriormente. La memoria -- del micro-programa le dice al controlador que es necesario esperar otro dato. Al oprimir otra tecla de un 'x' número, sucederá lo mismo que con el primer número. Al apretarse la tecla de igual (=) provocará que la memo-- ria del micro-programa le indique al controlador que -- efectúe la suma, esta será llevada a cabo en la unidad- aritmética/lógica. El resultado será mostrado en la -- unidad de despliegue. En el ejemplo anterior, se hace-

uso del registro del acumulador para contener los números y del registro de operador, así como del registro - de despliegue para la información que se esté mostrando en ese momento. Si se hubiera hecho una multiplicación o división, se hubiera usado el registro de cociente y- en caso necesario los registros de bandera para indicar saturación.

SISTEMA DE NUMERACION

Un sistema de numeración es una representación mediante un conjunto de símbolos de valores numéricos. En un sistema de base M se emplean M dígitos (0 a M-1) para representar los valores numéricos. En el sistema de base 60, se tiene del 0 al 59. Como se tiene en los sistemas electrónicos información de tipo binaria, el sistema binario (base 2) o aquellas cuya base es potencia de dos son los utilizados. En los sistemas digitales a cada dígito binario se le conoce como bit, en donde un conjunto de 8 bits es equivalente a un Byte. Dependiendo del sistema en particular, un Byte puede ser equivalente a una palabra, en donde palabra se define como un conjunto de bits ampleados para representar un valor numérico. La siguiente fórmula es empleada para convertir un número de base M a su correspondiente en el sistema decimal:

$$A_n A_{n-1} \cdots A_0 A_{-1} A_{-2} \cdots A_{-n})_M = \sum_{i=0}^{n} A_i M^i + \sum_{i=-1}^{-n} \qquad ---(1)$$

Para poder convertir un número en base 10 a base M es necesario expresar primero el número decimal en su parte fraccionaria, es decir:

$$A_{10} = A_{e_{10}} + A_{f_{10}} \qquad ----- (2)$$

La parte entera en base M se obtiene dividiendo $A_{e_{10}}$ entre M sucesivamente. Cada uno de los residuos resultantes corresponderá a los dígitos de la parte entera A_{e_m} - donde el ler. residuo es el dígito menos significativo- y así sucesivamente. Matemáticamente equivale a lo siguiente:

M	$A_{e_{10}}$	Residuos
M	N_1	A_0
M	N_2	A_1
⋮		
M	N_n	A_{n-1}
	0	$A_n = N_n$

$$A_{e_m} = A_n\ A_{n-1} \cdots A_0$$

$$A_{e_{10}} = M \times N_1 + A_0$$

$$A_{e_{10}} = M(M \times N_2 + A_1) + A_0$$

$$A_{e_{10}} = M\overline{M(M \times N_3 + A_2) + A_1} + A_0$$

$$A_{e_{10}} = M^3 N_3 + M^2 A_2 + M A_1 + M^0 A_0$$

$$A_{e_{10}} = \sum_{i=0}^{n} M^i A_i$$

La conversión de la parte fraccionaria ($A_{f_{10}}$ a A_{f_m}) se logra multiplicando sucesivamente $A_{f_{10}}$ por M. En cada multiplicación solo se emplea la parte fraccionaria resultante del paso anterior y las partes enteras son los dígitos correspondientes a A_{f_m} La primera parte entera que se obtiene es la más significativa. Matemáticamente se tiene:

A_{f10}
x M
A_{-1} . R_1
x M
A_{-2} . R_2
x M
A_{-3} . R_3
x M
⋮
.

$$A_{f_m} = .A_{-1}\ A_{-2}\ A_{-3} \cdots$$

$$R_{n-1} = A_n M^{-1}$$

$$R_2 = A_{-3} M^{-1} + A_{-4} M^{-2} + \cdots + A_{-n} M^{-(n-2)}$$

$$R_1 = A_{-2} M^{-1} + A_{-3} M^{-2} + \cdots + A_{-n} M^{-(n-1)}$$

$$A_{f10} = A_{-1} M^{-1} + A_{-2} M^{-2} + \cdots + A_{-n} M^{-n}$$

$$M\ A_{f10} = A_{-1}+A_{-2}M^{-1}+A_{-3}M^{-2}....+A_{-n}M^{-(n-1)}$$

$$M\ A_{f10} = A_{-1}+R_1 = A_{fm}+R_1'$$

Para poder realizar operaciones con números binarios o-códigos y representar los números negativos existen diversas operaciones básicas y representaciones. Dentro-de las operaciones basicas, se tiene:

El complemento M-1 de N_m, que se obtiene sustrayendo ca da dígito de N del número M-1.

El complemento M de N_m, se obtiene agregando uno al com plemento M-1 de N_m.

Por lo que se refiere a la representación de números -- signados, se tiene que estos estan compuestos por un -- signo (0 para +, 1 para -) y la magnitud (parte entera- y fraccionaria). Se pueden expresar como de punto fijo, es decir parte entera y parte fraccionaria separadas -- por un punto, o de punto flotante, con el punto a la -- izquierda seguido de la mantisa y el exponente. Exis-- ten 3 representaciones para los números negativos:

a) Magnitud signada.
b) M-1 complemento.
c) M complemento.

La representación con magnitud y signo es como sigue:

bn+1	.	bn bn-1.....b_0
signo		magnitud

Su desventaja es que la representación del cero no es - única. Para la representación b), se obtendrá el com-- plemento M-1 del número (incluyendo el bit del signo) si el número es negativo. La desventaja de esta represen- tación es la de tener que checar magnitud y signo de -- los operandos antes de efectuar las operaciones. Para- la representación c), se obtendrá el complemento M del- número (incluyendo bit de signo) en caso de ser negati- vo. Las ventajas que presenta esta representación es - de que las operaciones con números de diferentes signo- quedan en M complemento y la representación del cero es única.

En la representación de números también se pueden utili zar códigos. Este se define como una representación de los dígitos o caracteres (información) mediante otro ti po de símbolos. Cuando la información de entrada y sa- lida es grande, como sucede con una calculadora, es con veniente utilizar el codigo BCD (dígitos decimales codi ficados en binario). Cada dígito binario se representa por 4 bits, tal como se muestra en la tabla 3.1.

Dígito decimal	Código BCD	Dígito decimal	CódigoBCD
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Tabla 3.1

Una ventaja de este sistema es el de mininizar errores- por redondeo, pero, sin embargo, no es fácil obtener el 9 complemento. Debido a las características dichas - - anteriormente, este sería el código empleado.

C A P I T U L O IV

DESARROLLO DE ALGORITMOS Y FUNCIONES.

En este capítulo se presentan los algoritmos empleados para efectuar las operaciones, así como el desarrollo, a través de mapa de Karnaugh de algunas funciones lógicas.

Primero se presenta el algoritmo para la suma. Hay que considerar que se trabajará con números en código BCD.

Luego se presenta el algoritmo para la resta, el cuál está basado en el de la suma, pero contando con una serie de instrucciones adicionales para efectuar el complemento a diez del número negativo.

El algoritmo para la multiplicación utiliza iterativamente el de la suma, así como también decrementa un registro, como se verá más adelante.

En la división, al igual que en la multiplicación se utiliza el algoritmo de la suma y el de la resta también, ya que se usa el método de restar y restablecer en caso de tener un residuo parcial negativo.

Finalmente se presentan los algoritmos para la suma y la resta de números en base sesenta. Estos están basados en los algoritmos para la suma y resta de números decimales pero con algunas condiciones adicionales.

ALGORITMO PARA LA SUMA

Supongamos dos números de diez digitos máximo cada uno, siendo estos:

$$A = A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1$$

y

$$B = B_{10}B_9B_8B_7B_6B_5B_4B_3B_2B_1$$

y que cada uno de sus dígitos están representados en código BCD, por lo tanto cada dígito estará compuesto de 4 bits. Ahora, sean:

$$a_8a_4a_2a_1 \qquad y \qquad b_8b_4b_2b_1$$

La representación de un dígito cualquiera del primer y segundo número respectivamente. Tomemos un bit cualquiera de cada uno de los dos números y consideremos un posible carry. Ahora asignemos todas las posibles combinaciones a estos y obtendremos los resultados mostrados en la tabla siguiente.

a_i	b_i	Cent	s_i	Csal
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabla IV.1

donde:

Cent = carry de la etapa anterior

Csal = carry a la etapa siguiente

s_i = suma de a_i , b_i y Cent

Haciendo un mapa de Karnaugh a partir de la tabla anterior para encontrar la función lógica para s_i, tenemos:

Cent \ a_i b_i	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Figura IV.1 Mapa de Karnaugh para S

Obteniendo:

$$Si = \overline{Cent}\,\bar{a}ibi+\overline{Cent}\,ai\bar{b}i+Cent\,\bar{a}i\bar{b}i+Cent\,aibi \quad ---(1)$$

Como Si vale uno cuando un número impar de entradas vale uno, la función para Si es impar y representa el OR-exclusivo, o sea:

$$Si = Cent \oplus ai \oplus bi \quad ----------(1')$$

Ahora hagamos un mapa de Karnaugh a partir de la misma tabla IV.1, para encontrar la función para el carry de salida Csal. Esta se muestra en la figura IV.2

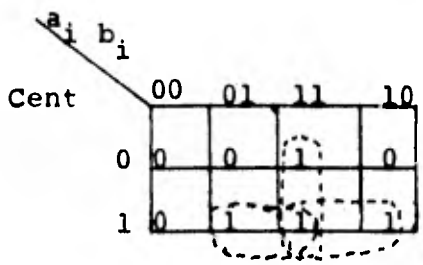


figura IV.2 Mapa de Karnaugh para Csal

Obteniendo:

$$Csal = a_i b_i + Cent\ a_i + Cent\ b_i \text{ - - - - -(2)}$$

Estas dos funciones son necesarias para poder realizar-sumas a nivel de bits.

Regresemos a los dos números que teníamos originalmente A y B. Cada dígito de estos números puede alcanzar el-valor máximo de 9. Por lo tanto el valor de la suma no podrá exceder de 9+9+1=19, siendo el uno, un posible -- carry considerado de la etapa anterior.

Suma Binaria					Suma BCD					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
1	1	1	0	0	1	0	0	1	0	12
1	1	1	0	1	1	0	0	1	1	13
1	1	1	1	0	1	0	1	0	0	14
1	1	1	1	1	1	0	1	0	1	15

Tabla IV.2

Si sumamos estos dos dígitos utilizando las funciones - (1) y (2) anteriores, obtendremos todos los posibles resultados mostrados a la izquierda (debajo de suma binaria) de la tabla IV.2 Sin embargo sabemos que el núme-

ro máximo representable en código BCD es el 9 (1001), - por lo tanto los números mayores a este son caracteres- inválidos. Aquí se determinará una función para conver tir estos números a su representación correcta en códi- go BCD, que es la columna mostrada bajo el titulo de su ma BCD de la tabla anterior. Formemos el mapa de Kar-- naugh para encontrar dicha función de corrección.

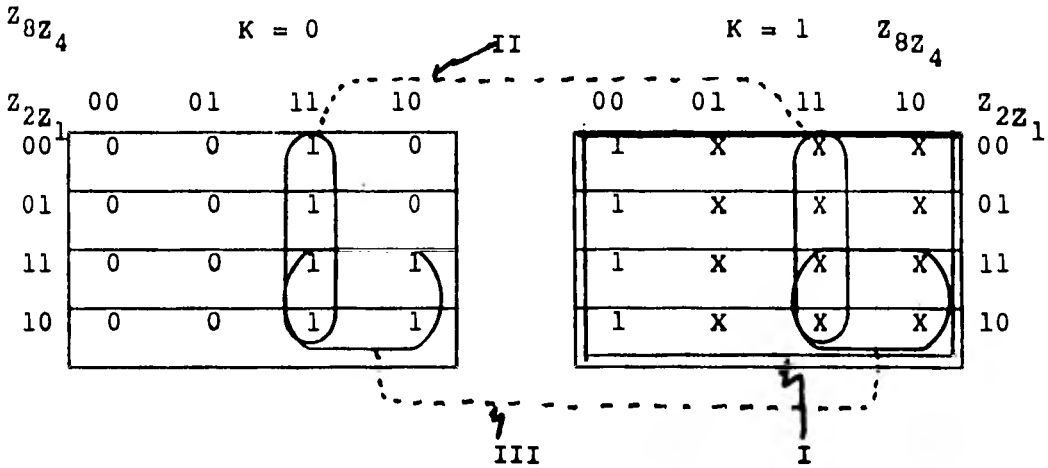


Figura IV.3 Mapa de Karnaugh para encontrar la función- de corrección de donde se obtiene la siguiente función:

$$C = K + Z_8Z_4 + Z_8Z_2$$

y usando la terminología anterior queda:

$$Csal = Cent + A_{i,8}A_{i,4} + A_{i,8}A_{i,2} \quad \text{----------(3)}$$

Regresando otra vez a nuestros números A y B estableceremos que habrá un bit extra As y Bs que serán usados - para guardar información sobre el signo. Si estos con-

tienen un cero el número será positivo y si contienen - un uno será negativo. De tal manera que una suma se -- puede representar de la siguiente forma:

$$Ss\ S \leftarrow As\ A + Bs\ B$$

El siguiente paso es por lo tanto encontrar la función- para poder determinar el signo del resultado (Ss). En- la siguiente tabla se muestra el valor que deberá tener Ss dependiendo de la magnitud y signo de los números -- A y B. Conviene señalar que las operaciones de resta - serán explicadas con mayor detalle en el siguiente inci so.

f (.)	Condición	Resultado	Csal	Ss
A+B	A B ó B A	A+B	X	0
$A+\bar{B}+1$	A B	A-B	1	0
$A+\bar{B}+1$	B A	10'(B-A)	0	1
$B+\bar{A}+1$	B A	B-A	1	0
$B+\bar{A}+1$	A B	10'(A-B)	0	1
$\bar{A}+1+\bar{B}+1$	A B ó B A	A+B	X	1

Tabla IV.3

Donde:

$\bar{A}+1$ = Complemento a diez de A

$\bar{B}+1$ = Complemento a diez de B

10'(B-A)= Diez complemento de B-A

X = No importa el valor que tome.

En el primer renglón el signo del resultado Ss deberá ser positivo, por estarse sumando dos números positivos. En el segundo y tercer renglón el valor de Ss dependerá de cual de los dos números (A ó B) sea mayor. Esta información la podemos conocer a partir de Csal, tal como se explica a continuación.

La suma de A+B'+1 es equivalente a $A+(10^n-B)=10^n+(A-B)$. Cuando el carry de salida Csal es igual a uno, significa que un uno está disponible en la posición n+1 del resultado y por lo tanto representa el valor de 10^n. - - Ahora si $A \geq B$ entonces (A-B) debe ser un número positivo y $10^n+(A-B) \geq 10^n$. Si quitamos el carry de salida -- (10^n) entonces nos queda (A-B). Ahora si $A < B$, entonces (A-B) debe ser un número negativo. Por lo tanto -- $10^n+(A-B) < 10^n$ y Csal deberá ser cero. El resultado de esta operación puede ser escrito como $10^n-(B-A)$, donde B-A es un número positivo. Esto es equivalente al diez complemento de (B-A). Para obtener la magnitud de la diferencia, debemos complementar otra vez para obtener $10^n-(10^n+A-B) = B-A$, siendo el resultado negativo.

Los renglones cuatro y cinco son similares al segundo y tercero y el último al primer renglón. A partir de la tabla IV.3 podemos escribir la tabla de verdad siguiente:

As	Bs	Csal	Ss
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Tabla IV.4

Y el mapa de Karnaugh es:

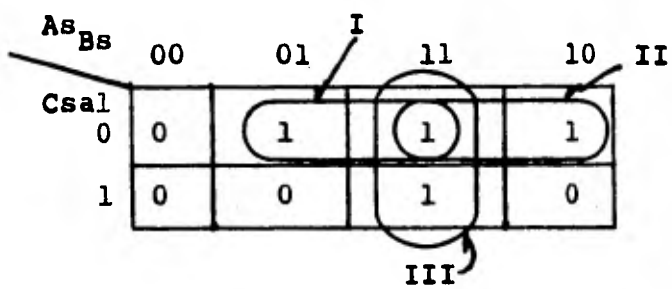


Figura IV.4 Mapa de Karnaugh para Ss

Obteniendose la siguiente función:

$$Ss = \bar{C}sal\ Bs + AsBs + \bar{C}sal\ As \quad \text{---------(4)}$$

Solo nos falta encontrar la función que nos detectará - el caso en que exista un overflow, la cual es explicada enseguida.

Consideremos el siguiente circuito, usando para sumar- dos números.

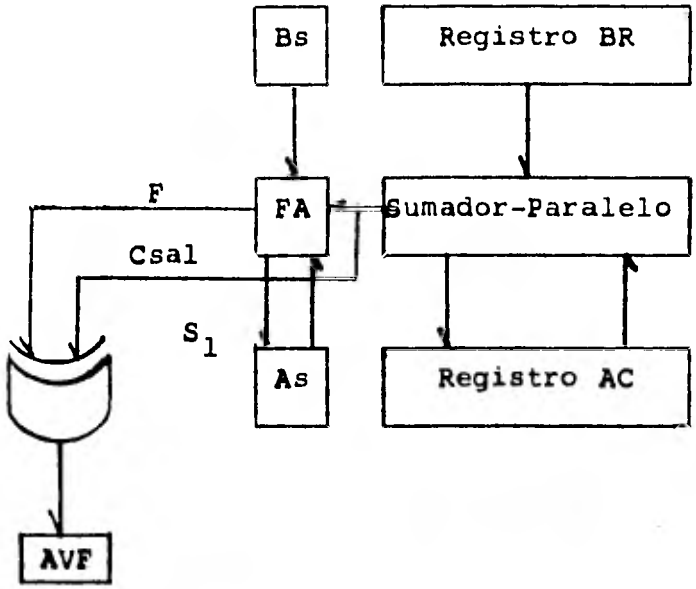


Figura IV.5 Circuito para sumar.

Donde: F = AsBs+As Csal+Bs Csal ----------(5)

En los registros AC y BR se almacenan los sumandos, a - la vez que el resultado se transfiere al registro AC. - En este circuito se muestra un sumador completo (FA) -- empleado para obtener la suma de los bits de signo As y Bs. También se muestra un Flip Flop llamado AvF el - - cual valdrá uno en caso de existir overflow y cero en - caso de no haberlo.

Cuando los números a sumar son de diferente signo no po drá existir overflow ya que el resultado será menor en- magnitud que cualquiera de los dos.

Por lo tanto ,solo se podrá presentar overflow cuando se trate con números del mismo signo, habiendo dos posibles casos. El primero que es cuando los dos números son positivos, tendremos que As y Bs serán cero y por lo tanto S_1 también tiene que valer cero. Si el valor de Csal, que es el carry de salida de la última etapa, vale cero, entonces S_1 también valdrá cero como es requerido. Pero si existe un overflow en la suma de las magnitudes entonces Csal valdrá uno y S_1 también. El valor de F valdrá cero, lo anterior se muestra en los dos primeros renglones de la tabla IV.5

As	Bs	Csal	F	S_1	AVF
0	0	0	0	0	0
0	0	1	0	1	1
1	1	1	1	1	0
1	1	0	1	0	1

Tabla IV.5 Detección de Overflow.

Para el segundo caso que es cuando los dos números son negativos, tenemos que As y Bs valen uno. El valor de S_1 deberá ser uno por obtenerse un resultado negativo. Si Csal vale uno el sumador-completo (FA) producirá un uno en S_1 como es requerido y F también valdrá uno. Sin embargo si Csal vale cero, se tendrá un cero en S_1 y un uno en F. Lo anterior solo ocurriría si existe un overflow durante la adición de dos números negativos ya que un cero para números negativos corresponde a un uno con números positivos. Por lo anterior se concluye que

habrá overflow cuando exista un cambio en el signo del- resultado.

Para encontrar la función requerida para detectar esto, observemos que E y F representan la función exclusiva-- OR y nos indican un overflow, por lo tanto:

$$AVF = Csal \oplus F$$

ó

$$AVF = \bar{C}sal\ F + Csal\ \bar{F} \quad \text{---------- (6)}$$

Ahora si ya contamos con todas las funciones necesarias para realizar una suma de números en código BCD. Sin - embargo, antes de presentar el diagrama de flujo se mencionarán tres métodos empleados para realizar una suma, estos van relacionados directamente con los diferentes- circuitos que hay en el mercado.

El primer método llamado de adición decimal en paralelo, se tiene "n" sumadores de tipo BCD, uno por cada dígito a sumar. Este es el método más rápido pero también el- más costoso, Figura IV.6

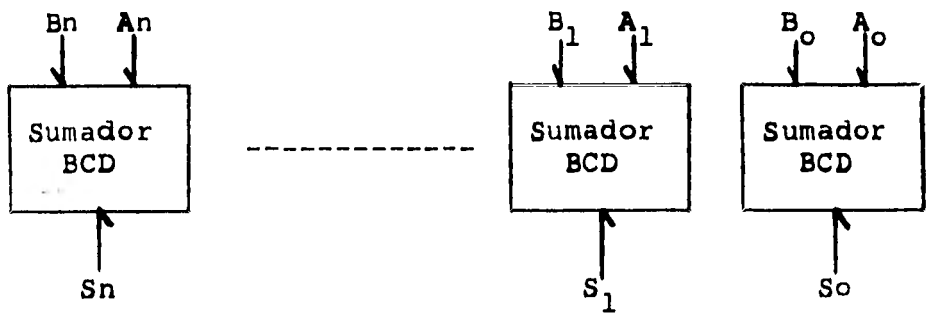


Figura IV.6 Adición en Paralelo.

Donde :

$Ai = a_8\ a_4\ a_2\ a_1$

$Bi = b_8\ b_4\ b_2\ b_1$

$Si = s_8\ s_4\ s_2\ s_1$

Nota: En el segundo método se suman los bits en paralelo y los dígitos en serie. Aquí solo se emplea un sumador de tipo BCD y los dígitos se van alimentando uno -- tras del otro por medio de corrimientos, ver figura IV.7.

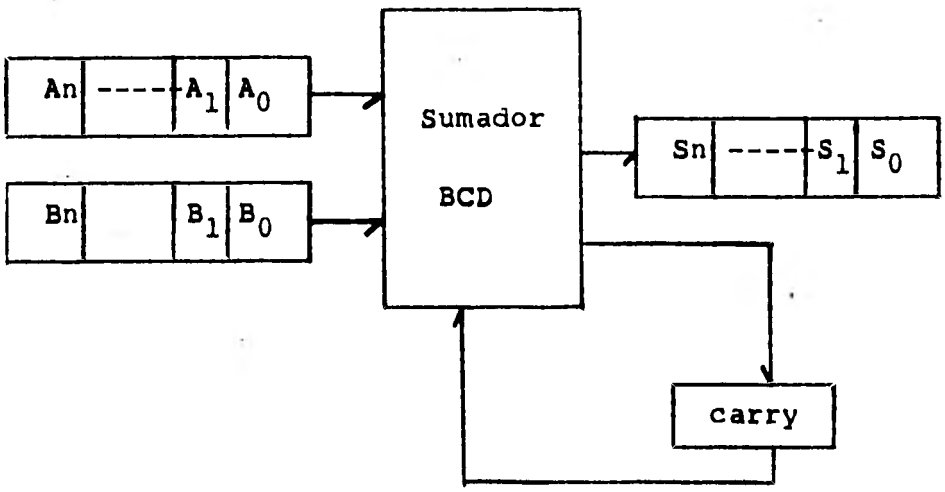


Figura IV.7 Suma de bits en paralelo y dígitos en serie.

Nota: Un sumador de tipo BCD es el que integra ya la corrección.

Este método proporciona un valor medio en rapidez y costo y es el que se emplea normalmente en las calculadoras. Por lo tanto, este será el que tomemos como modelo en nuestro diseño.

En el tercer método toda la suma se hace en forma com-- pletamente seriada usando un sumador completo (FA), por lo tanto hay que preveer un circuito para la corrección. Este se muestra en la figura IV.8

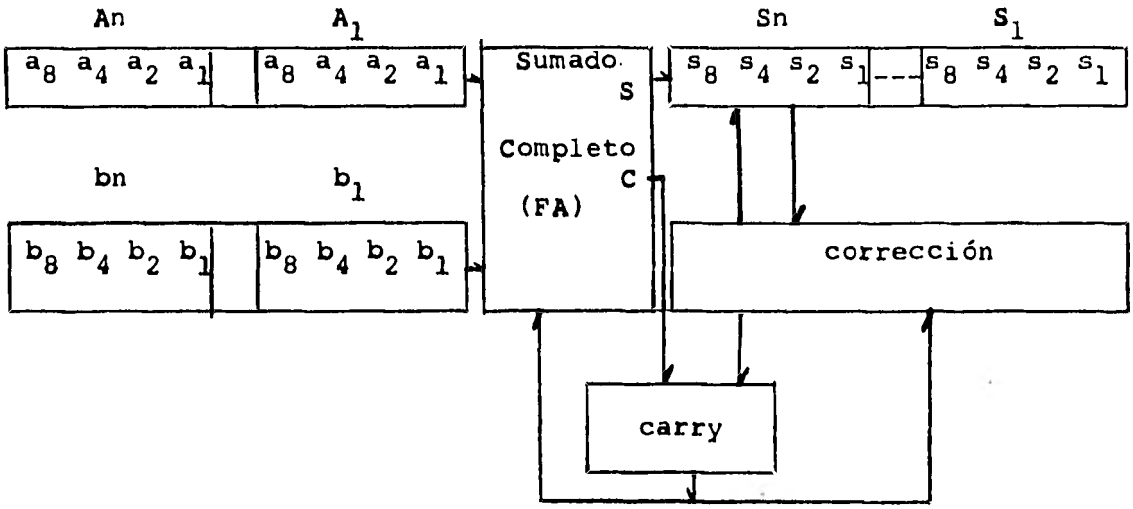
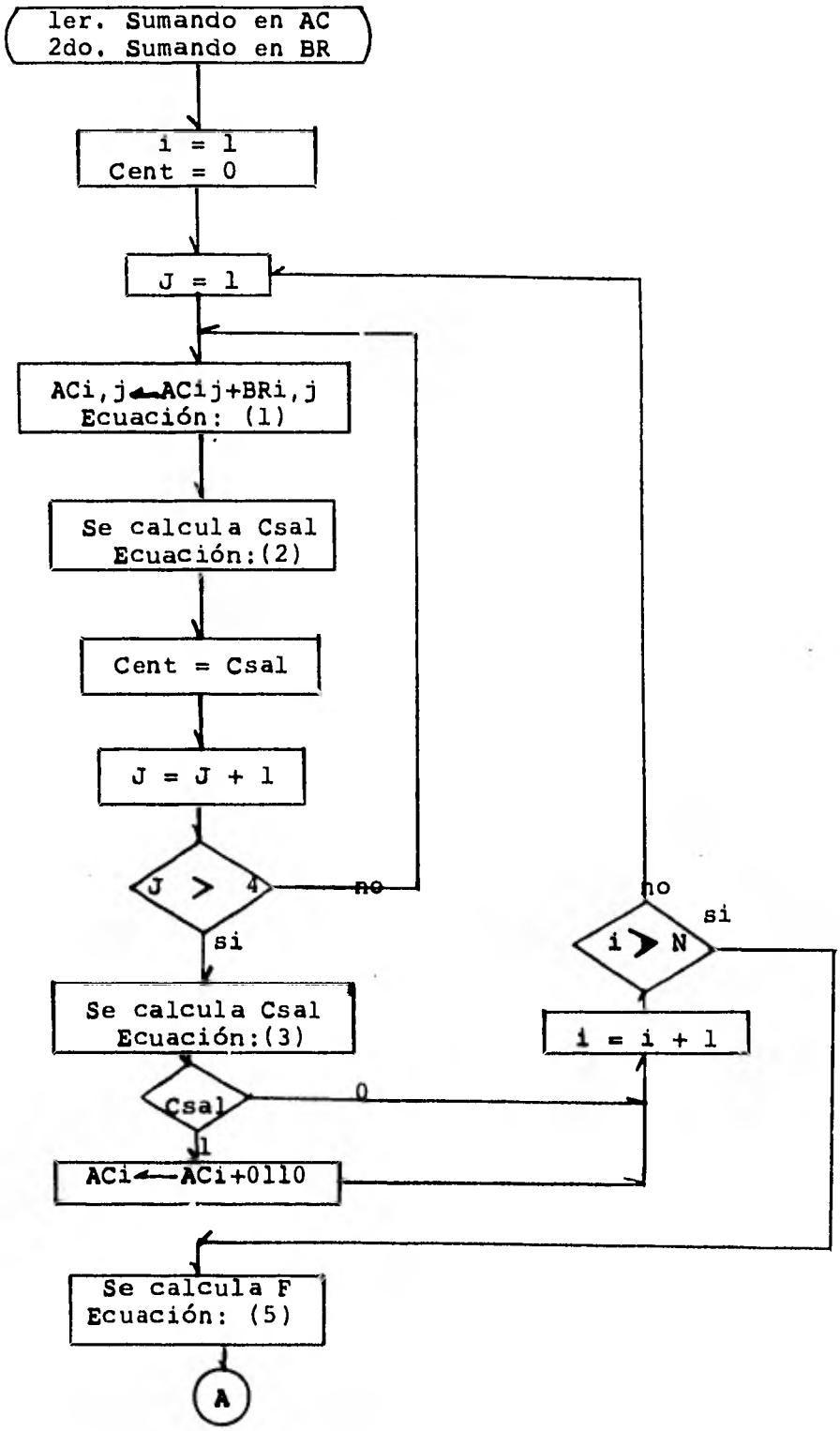


Figura IV.8 Suma completamente en serie..

Este tercer método es bastante barato pero muy lento.

A continuación se presenta el algoritmo para efectuar - una suma con números en código BCD.



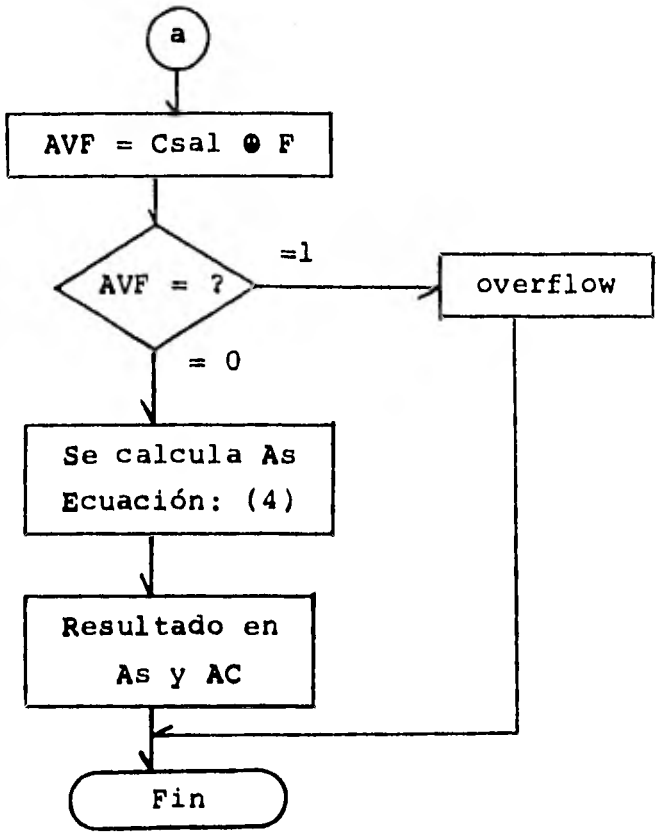


Figura IV.9 Algoritmo para la suma.

Inicialmente se guarda en AC y BR los sumandos, luego - se procede a sumar un dígito de cada número usando las- ecs. desarrollados. Al complementar esto, se calcula -- Csal y dependiendo de este se hará o no corrección a la suma obtenida de los dos dígitos. El valor de N es una constante y representa al número máximo de dígitos que- puede contener un número. En nuestro caso valdrá diez. Después de haber completado la suma de los dos números- se verifica que no haya overflow. En caso de no exis-- tirlo se calcula el signo del resultado As. Finalmente el resultado queda en As y AC.

ALGORITMO PARA LA RESTA

La operación de resta está basada en el algoritmo para-la suma, ya que todos los principios para la operación-de suma rigen también en la resta. Sin embargo es necesario establecer algunas condiciones y funciones adicionales.

La manera en que aquí se efectuará una suma será obte--niendo el complemento a diez del ó de los números negativos y luego sumarlo al otro.

Para obtener el complemento a diez, primero se debe obtener el complemento a nueve y hacer Cent igual a uno - con el primer par de dígitos. Existen dos métodos de - obtener el complemento a nueve.

En el primer método se debe complementar todos los bits de un dígito y luego sumarle diez (1010), descartando - el carry de salida. Esto se hace para todos los dígi--tos del número a complementar a nueve. La demostración de lo anterior es como sigue. Sea N un número, al complementar los bits de uno de sus dígitos equivale a restar quince (1111) menos el dígito y sumarle luego diez- equivale a 15-N+10 = 9-N+16, pero como 16 equivale a un carry de salida y dijimos anteriormente que este lo descartamos, obtenemos: 9-N, como es deseado.

En el segundo método primero se le suma seis (0110) al-dígito y luego se complementa cada bit. Siendo lo anterior equivalente a: 15-(N+6)=9-N, como es requerido.

A continuación se desarrollarán las funciones necesa---rias para obtener directamente los valores de los bits-para obtener el complemento a nueve. En la tabla IV.6-se muestra los números del 0 al 9 y sus correspondien--tes complementos a nueve.

Número	Complemento a nueve
0000	1001
0001	1000
0010	0111
0011	0110
0100	0101
0101	0100
0110	0011
0111	0010
1000	0010
1001	0000

Tabla IV.6

Complemento a nueve de los números cero a nueve.

Formemos el mapa de Karnaugh para encontrar la función-de complemento a nueve de A_1, o sea X_1.

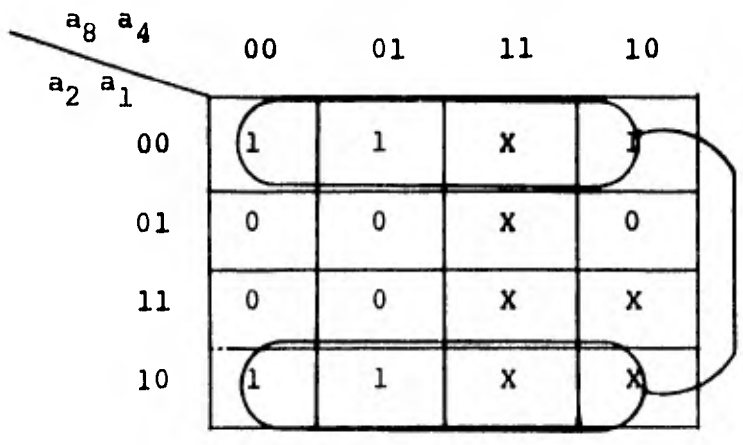


Figura IV.10 Mapa de Karnaugh para X_1

de donde se obtiene:

$$X_1 = \bar{a}_1 \quad \text{---------- (7)}$$

En las figuras IV.11, IV.12 y IV.13 se encuentran los - mapas de Karnaugh para X_2 X_4 y X_8 respectivamente, de -- donde se obtienen las siguientes ecuaciones.

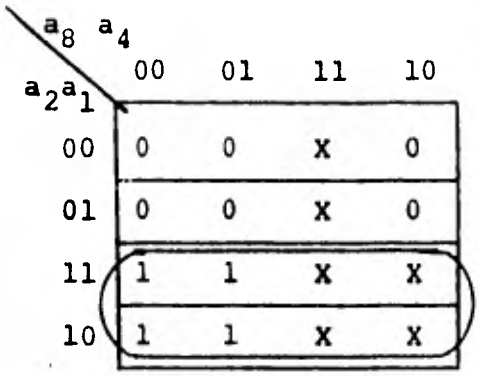


Figura IV.11 Mapa de Karnaugh par X_2

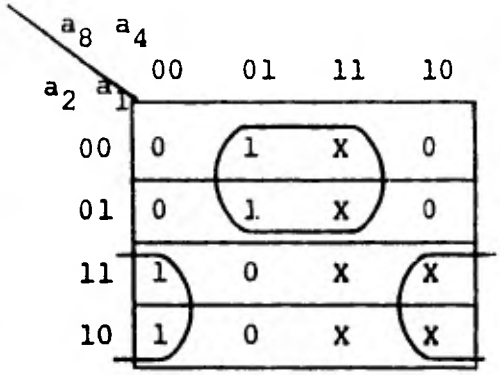


Figura IV.12 Mapa de Karnaugh papa X_4

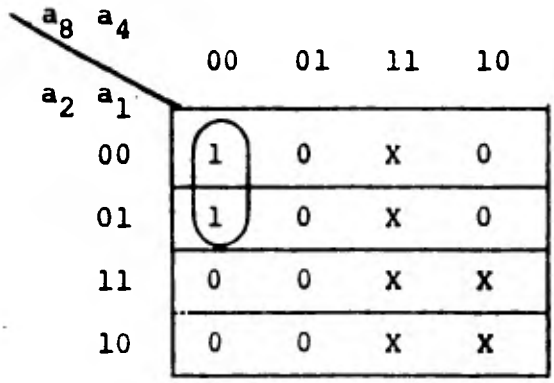


Figura IV.13 Mapa de Karnaugh para X_8.

$$X_2 = a_2 \quad \text{---------- (8)}$$

$$X_4 = \bar{a}_2\, a_4 + a_2\, \bar{a}_4 \quad \text{---------- (9)}$$

$$X_8 = \bar{a}_2\, \bar{a}_4\, \bar{a}_8 \quad \text{---------- (10)}$$

Ahora debemos establecer cuando será necesario comple-- mentar. Para esto es necesario explicar un poco acerca de la forma en que trabajará la simulación de la calculadora.

Inicialmente se teclea un número, el cual será almacenado en AC, posteriormente se oprime alguna tecla que represente una función (+,-,*,/), dependiendo de la que haya sido, se le dará un valor de 1 a 4 a una bandera.- Si la tecla oprimida es menos entonces el valor de la - bandera será dos. Posteriormente se teclea el segundo- número, en caso de resta o suma, este número será almacenado en el registro BR. Finalmente se oprime la tecla de igual.

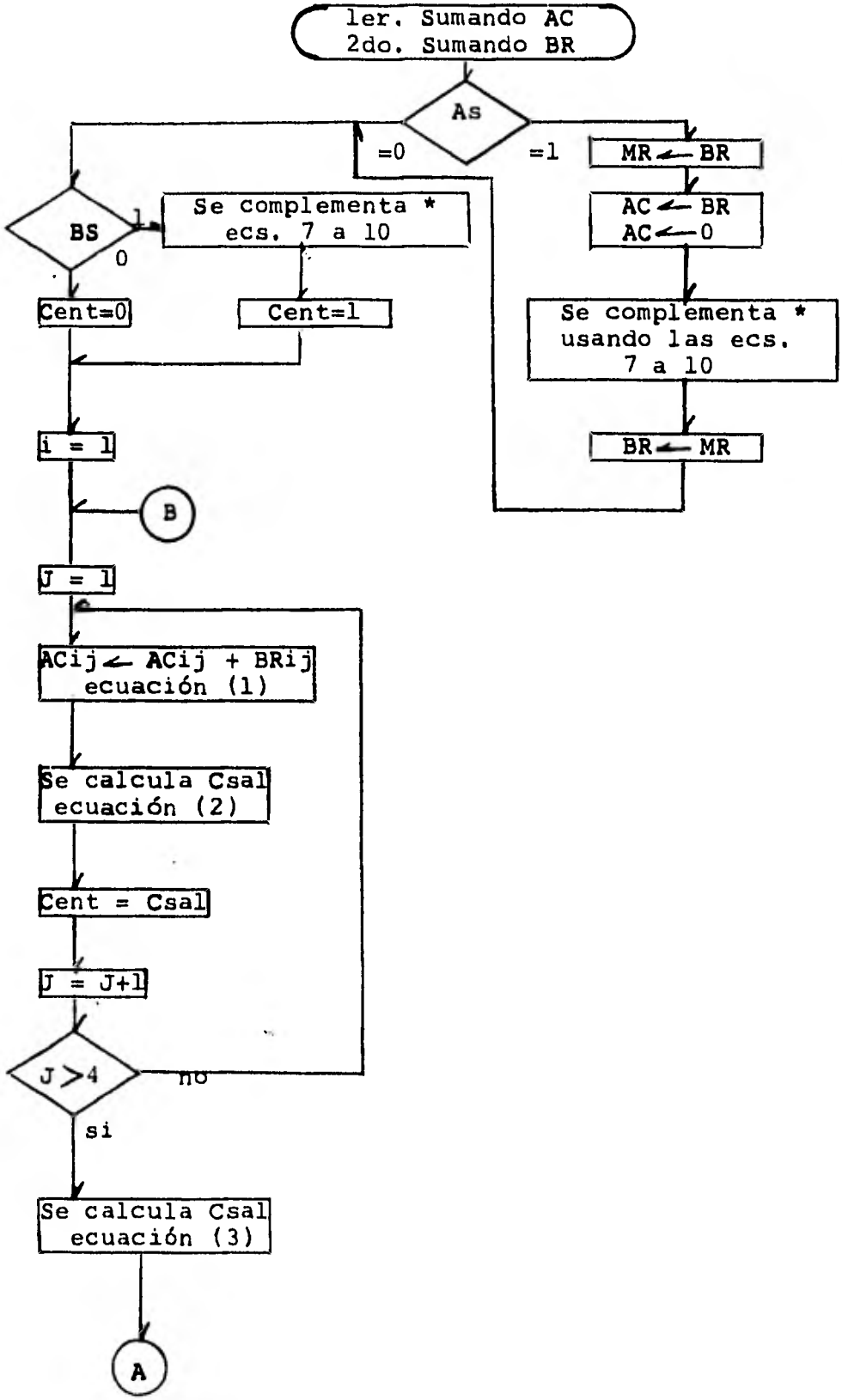
Por lo anterior se ve que existen tres casos en los que es necesario complementar. Primero cuando el valor de- As es uno, segundo cuando la bandera vale dos o Bs vale uno y tercero cuando el resultado es negativo, ya que - este se presentará en complemento a dos y será necesa-- rio volver a complementar para obtener la representación correcta de la magnitud.

La manera en que se obtenga el complemento para el primer caso dependerá del circuito que se esté simulando.- En este pundo diremos que utilizaremos un sumador de tipo BCD con un elemento adicional de complemento a nueve, como será explicado con mayor detalle en el capítulo -- sexto. Debido a la condición de este circuito, es nece

sario pasar a través de sumador BCD para poder comple-- mentar un número. Por lo anterior hay que hacer movi-- mientos entre registros y limpiar otros. Así, para el - primer caso mencionado anteriormente será necesario --- guardar el valor de BR en un registro de memoria, o sea hacer MR←BR, luego transferir el valor del número o -- complementar a BR, o sea: BR←AC, a la vez que se ten-- drá que limpiar el registro de AC, AC←0. Después de - estos movimientos se pasará por el sumador BCD y su ele mento de complemento a nueve. Finalmente se volverá a- reintegrar el valor original de BR, es decir BR←MR.

Para el tercer caso se hará BR←AC, luego AC←0 pasando lo luego por el complementador. El resultado quedará - en AC como es requerido.

En el siguiente diagrama de flujo se presentan estas -- condiciones, así como las funciones anteriores.



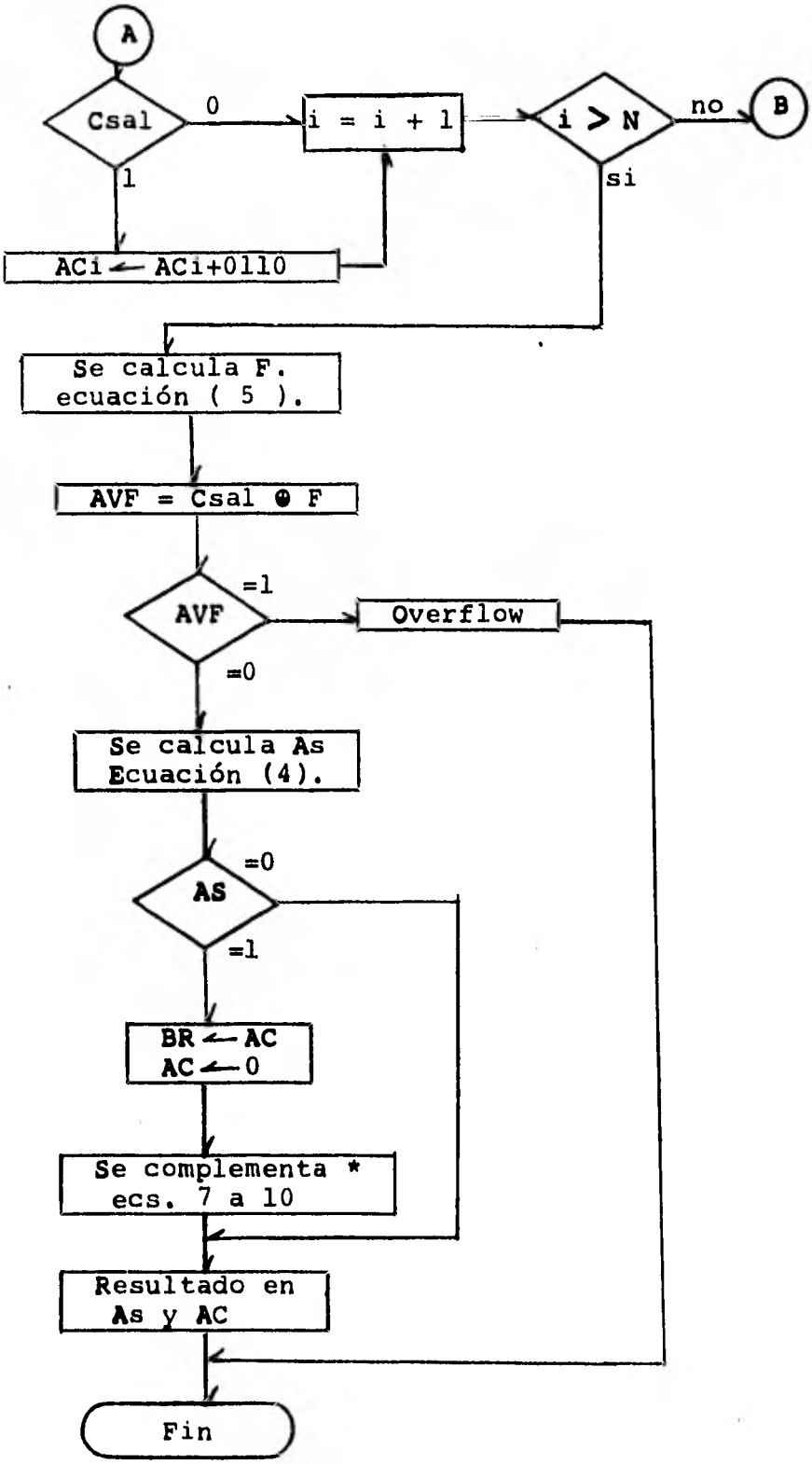


Figura IV.14 Algoritmo para la resta.

Para no sobre llenar este algoritmo, la lógica para los cuadros marcados con un asterisco se muestran en la figura IV.15. En caso de ser nagativo el resultado, el valor que tomará As sera de uno.

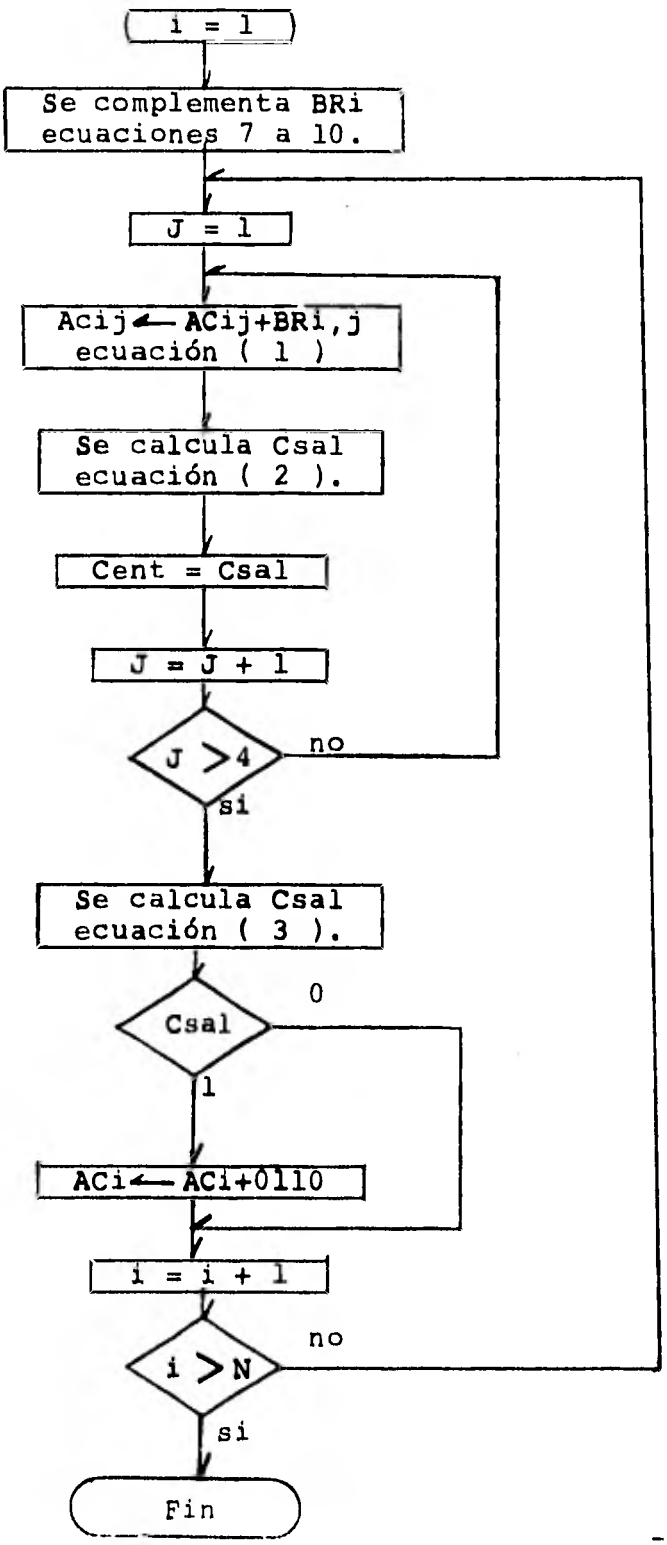


Figura IV.15 Diagrama de flujo para los cuadros marcados con asterisco.

ALGORITMO PARA LA MULTIPLICACION

Una operación de multiplicación se puede efectuar por - medio de un proceso que combine sumas sucesivas y corri mientos. Veamos el siguiente ejemplo para aclarar esto. Tomemos dos números decimales cualesquiera y multipli-- quémoslos.

```
  23      multiplicando
x426      multiplicador
```

Para hacer esto tomemos el dígito menos significativo - del multiplicador (6). El hecho de multiplicar 6 por - 23 es equivalente a sumar seis veces el valor de 23, es decir:

23+23+23+23+23+23=138 producto parcial.

luego colocamos este producto parcial (138) debajo del- 426:

```
  23
x426
 138
```

Tomando ahora el segundo dígito del multiplicador (2),- vemos que necesitamos sumar 23 dos veces, es decir:

23+23=46

Para asignarle la posición correcta a este segundo producto parcial, tenemos que recorrerlo un lugar a la - - izquierda del 138, es decir:

```
  23
x426
 138
 46
```

analogamente para el tercer dígito del multiplicador -- (4), tenemos:

```
  23          23+23+23+23=92
x426
 138
 46
92
9798
```

y al sumar los tres productos parciales, obtenemos el - resultado mostrado.

El proceso que se acaba de presentar es el usado normal mente cuando se efectúa la operación de multiplicación- con un lápiz y papel. Al desarrollar el algoritmo para la simulación de esta operación en una computadora, nos basaremos en el proceso anterior pero con algunas modi- ficaciones. La primera es que los productos parciales- se sumarán inmediatamente y no hasta el final, ya que - de esta última manera se necesitaría un registro por ca da producto parcial, siendo que de otra forma solo se - requieren dos registros. Así 138 y 460 se sumarían in- mediatamente y no hasta el final dando 598 y luego este sería sumado a 9200 dando 9798.

La segunda modificación es la de recorrer el producto - parcial hacia la derecha y no hacia la izquierda. Al - hacer esto las posiciones relativas de los productos --

parciales se conservan y no cambia el resultado. Así - en el ejemplo anterior, al sumar 46 más 138 se recorrerá 138 a la derecha y no 46 a la izquierda, siendo los- dos movimientos equivalentes.

Para desarrollar el algoritmo partiremos del hecho que- tendremos 3 registros AC, BR y QR cada uno con un Flip- Flop extra para indicar el signo, AS, BS y QS, respecti- vamente. Inicialmente AC valdrá cero, BR contendrá el- multiplicando y QR, el multiplicador. El primer paso - es tomar el dígito menos significativo (4 bits) del mul- tiplicador, si este es diferente de cero, se sumarán BR y AC. Luego se le restará uno al dígito de QR que se - está tratando, si todavía no alcanza el valor de cero - se volverá a sumar BR a AC y lo anterior se repetirá -- continuamente hasta que el dígito alcance el valor de - cero. En este momento se hará un corrimiento hacia la- derecha, es decir se hará:

dshr AC,QR

Luego se toma el siguiente dígito menos significativo - de QR y se repite el proceso anterior, hasta agotar to- dos los dígitos. En este punto conviene hacer notar -- que al estarse sumando dos productos parciales interme- dios puede originarse un overflow, por lo tanto tendre- mos otro registro E que constará de 4 bits para preveer lo anterior.

E dependerá del carry de salida. Como las entradas para calcular el valor de E son este mismo y Csal, se podrá obtener E con un medio sumador, usando las siguientes ecuaciones.

$E = e_8\ e_4\ e_2\ e_1$

$e_i = e_i \oplus Csal = \bar{e}_i\ Csal + e_i\ \bar{C}sal$ -----(11)

$Cent = e_i\ Csal$ ----------(12)

donde:

Csal es el carry generado de último par de dígitos de AC y BR.

Por lo anterior se tendrá que el corrimiento que se hará cuando el dígito que se esté manejando sea cero, será:

dshr E,AC,QR

Ahora para poder restarle uno a este dígito se le sumaquince (1111), tal como se demuestra enseguida. El complemento a nueve de uno es 1000 y el complemento a diez es 1001, sin embargo al sumar este valor a cualquier dígito obtendremos un número mayor que nueve (1001) y por lo tanto se le tendrá que sumar seis (0110) para poderlo representar correctamente en código BCD. Integrando estos dos valores tenemos:

1001 + 0110 = 1111

como se quería demostrar.

A continuación se presenta el diagrama del flujo del algoritmo para la multiplicación.

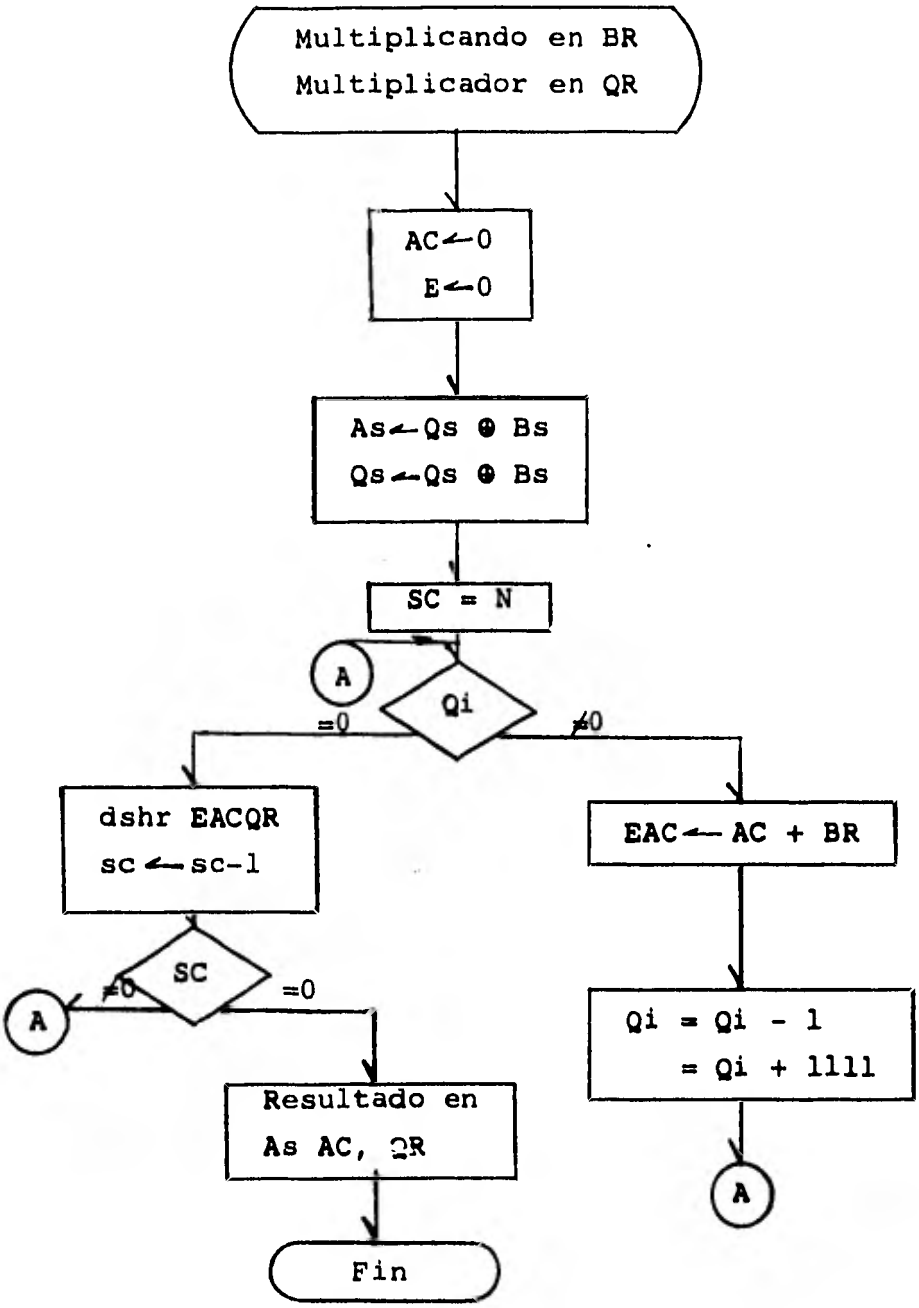


Figura IV.16 Algoritmo para la multiplicación.

Primero se cargan los registros BR y QR. Luego se limpian AC y E. Posteriormente se calcula el signo de As que es igual al de Qs por contener los dos el resultado, ya que este es de precisión doble. El cálculo --

del signo se hace usando la función OR exclusiva. El - valor de N representa el número máximo de dígitos que - podrá contener un número. En el siguiente paso se analiza Qi, si este es diferente de cero se suma AC y BR - obteniendo EAC, esto se hace usando el algoritmo para - la suma desarrollado anteriormente y también las ecuaciones 11 y 12. Luego se le resta uno a Qi y se repite el proceso. Si Qi vale cero, se hace un corrimiento -- hacia la derecha de EACQR y se decrementa el contador - SC, al haberse repetido N veces este proceso, SC valdrá cero y el resultado estará en As AC y QR.

ALGORITMO PARA LA DIVISION

La operación de división puede realizarse "a mano" por-- el método de reposición. Un ejemplo de esto se presenta a continuación.

```
                21          cociente
divisor  6  ) 131           dividendo
              -6
              ---
               7            1er. residuo parcial
              -6
              ---
               1
              -6
              ---
              -5
              +6
              ---
               11           se baja el siguiente dígito (uno)
              - 6           se corre el 6 un lugar a la derecha
              ---
                5
               -6
              ---
               -1
               +6
              ---
               +5           residuo final
```

Figura IV.17 División por el método de reposición.

Tomemos el trece y dividámoslo entre seis, para hacer - esto, restemos seis de trece tantas veces como sea necesario hasta obtener un número negativo. Contemos el número de veces que restamos 6 de 13 hasta antes de obtener el residuo parcial negativo y coloquemos este número como primer dígito del cociente. Ahora, no debemos- tener residuos parciales negativos por lo tanto lo tenemos que "reponer" sumandole seis a este número negativo. El siguiente paso es el de colocar el dígito que sigue- del 13 y ponerlo al lado del residuo parcial último (1). Seguidamente se recorre un lugar el 6 a la derecha y se repite el proceso de substracción y reposición hasta -- agotar todos los dígitos del dividendo.

Este método es el que nosotros tomaremos como modelo -- para nuestra simulación, pero sin embargo hay que hacer le una modificación. En el ejemplo anterior nosotros - recorrimos el divisor (6) hacia la derecha, el cambio - requerido es que sea el residuo parcial el que se recorra y a la izquierda. El efecto es el mismo ya que las posiciones relativas de estos se conservan.

De lo anterior es fácil ver que la operación de divi- - sión está basada en sumas y restas, por lo que solo falta determinar la función para el signo y el overflow.

Por lo que se refiere al signo del cociente, se tiene - que este será positivo si el divisor y el dividendo son del mismo signo y que será negativo en caso de que el - divisor y el dividendo tengan signos diferentes. La -- función OR exclusiva cumple completamente con lo ante-- rior y será la utilizada para determinar el signo del - cociente.

La posibilidad de overflow en la división se origina -- del hecho de que el divisor generalmente ocupa un registro de N bits, mientras que el dividendo se hace de precisión doble y ocupa dos registros de N bits.

Entonces existirá overflow si los primeros N-bits del - dividendo son mayores que los del divisor, ya que se tendrá de N+1 bits en el cociente, información que no podrá ser manejada por los registros al haberlos supuesto

de N bits. Esta información la podremos obtener a partir de la primera vez que se resta el divisor del dividendo. Si el carry de salida vale uno, entonces el dividendo será mayor que el divisor y tendremos al final un cociente compuesto por N+1 bits, existiendo por lo tanto overflow.

A continuación se presenta el algorítmo para la opera-- ción de división.

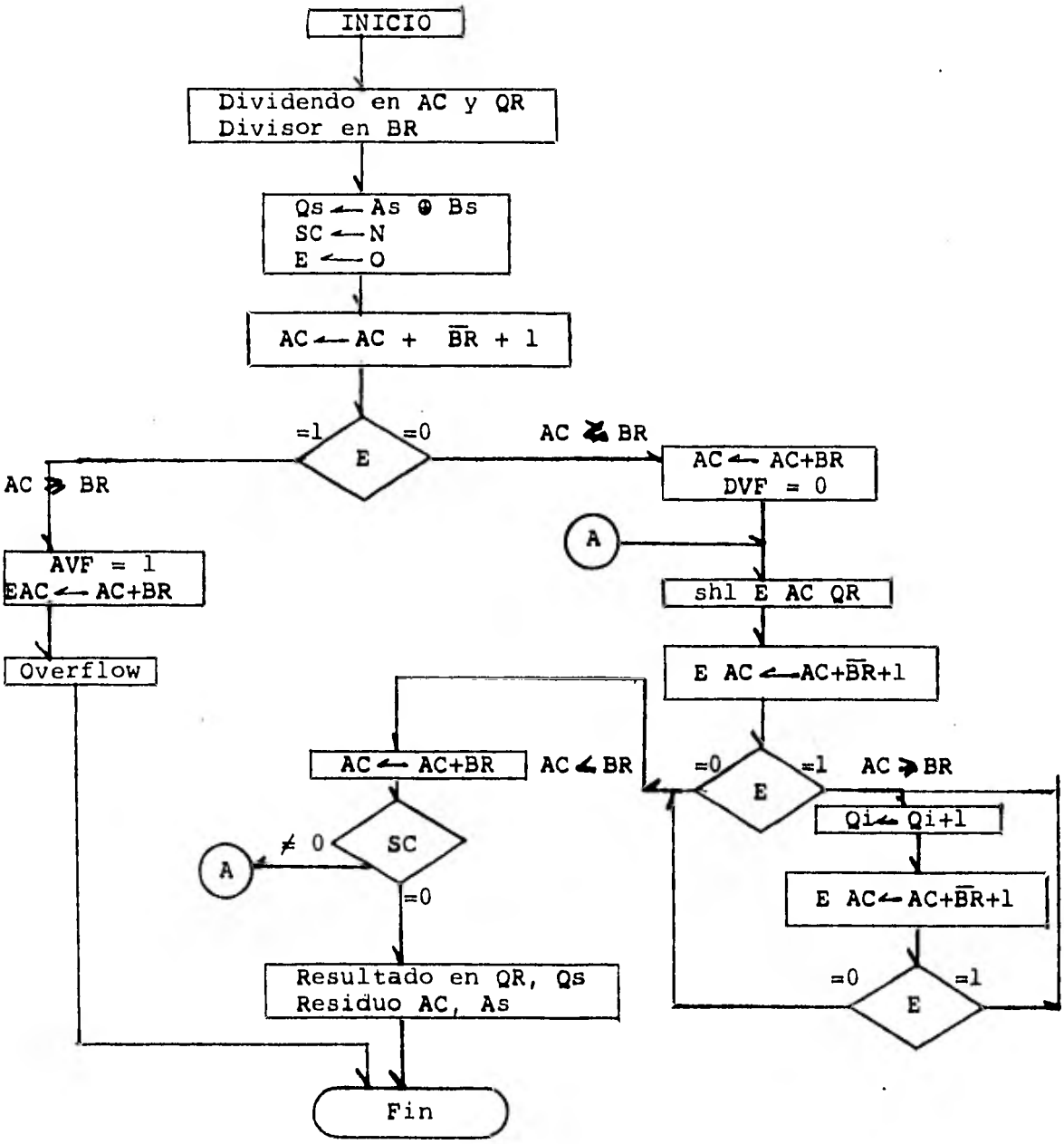


Figura IV.18 Algorítmo para la división.

Inicialmente se guarda el dividendo en los registros AC y BR y el divisor en el registro BR. Luego se limpia - el registro E y se hace SC igual a N donde N es el número de dígitos del dividendo. También se calcula el signo del cociente. Posteriormente se resta BR del divi-- dendo, si E vale uno entonces existe overflow y se termina el programa. Si E vale cero, entonces se repone - el valor del dividendo y se hace DVF igual a cero. Luego se hace un corrimiento hacia la izquierda de todo - el dividendo y se le resta el divisor si E vale cero, - entonces se vuelve a reponer el valor del dividendo y - se pregunta si ya se emplearon todos los dígitos del dividendo, en caso afirmativo la operación ha terminado y el resultado se encuentra en QR y Qs. En caso de que - aún queden algunos dígitos, se repite el proceso. En - Ql se va llevando la cuenta del número de veces en que- se ha restado el divisor del dividendo para el conjunto de bits que se estén manejando en ese momento.

ALGORITMO PARA LA SUMA CON NUMEROS DE BASE SESENTA

Este algorítmo esta basado en el de la suma con números decimales. De hecho es el mismo pero con la adición de varias funciones para hacer conversiones. La primera - conversión que es necesario hacer, es cuando los núme-- ros de entrada estan formados por centésimas y no por - minutos. Es decir,los dos dígitos a la derecha del punto decimal serán considerados como los minutos y si estos dígitos alcanzan un valor mayor de sesenta, se tendrán que convertir a un número entre 0 y 59 a la vez -- que se incrementará el valor de las horas, que están representadas por los dígitos a la izquierda del punto. - Por lo anterior sabemos que debemos hacer la conversión, cuando el primer dígito a la derecha del punto sea igual o mayor que seis. Tomando como base lo anterior podemos hacer el mapa de Karnaugh para obtener la función - de conversión. La figura IV.19 presenta este mapa.

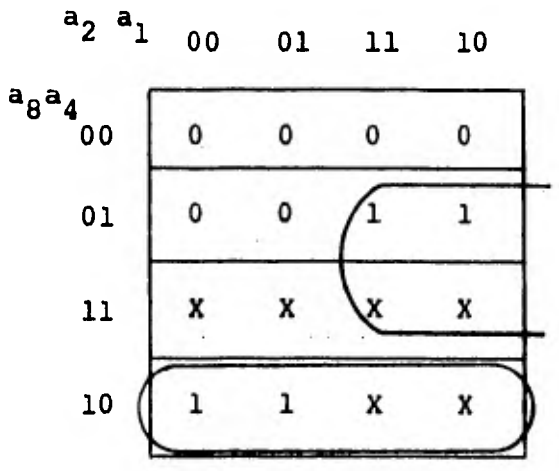


Figura IV.19 Mapa de Karnaugh para obtener la función de conversión.

y la función de conversión que se obtiene es:

$$F = a_8\bar{a}_4 + a_4a_2 \quad \text{---------- (13)}$$

Ahora ya sabemos cuando convertir, pero falta saber cómo. Esto es fácil, con solo sumar .40 se logra la corrección. Para ver esto más claro supongamos que tenemos sesenta y cinco minutos es decir 0.65, si le sumamos o.40 tendremos una hora y cinco minutos, 1.05, como se quería.

La otra conversión que es necesario efectuar, es con el resultado. Para esto existen dos casos. En la primera que es cuando se genera un carry de salida, habrá que sumar 0.40 al igual que antes. Para ejemplificar esto supongamos que sumamos 0.55 más 0.53, el resultado será de 1.08, valiendo el carry de salida uno después de haber sumado los dos cincos, siendo que el resultado debería ser 1.48. Si sumamos 0.40 a 1.08 obtendremos este resultado. El segundo caso se presenta cuando los minutos del resultado son mayor que sesenta, en este caso también será necesario sumar 0.40. Sin embargo como en el programa para hacer esta última conversión solo se utiliza un Full adder y no todo el sumador BCD, se le suma 1010 que es la suma de 0.4 (0100) y de 6 (0110), donde el seis es debido a la corrección que es necesario hacer cuando se obtienen códigos BCD mayores de nueve, tal como se explicó en el desarrollo del algoritmo para la suma. El hecho de haber utilizado un sumador completo y no el sumador de tipo BCD, fué por cues-

tiones de rapidez en el computo, tal como se verá en -- los siguientes capítulos. El diagrama de flujo para sumar números en base sesenta se presenta en la figura -- IV.20.

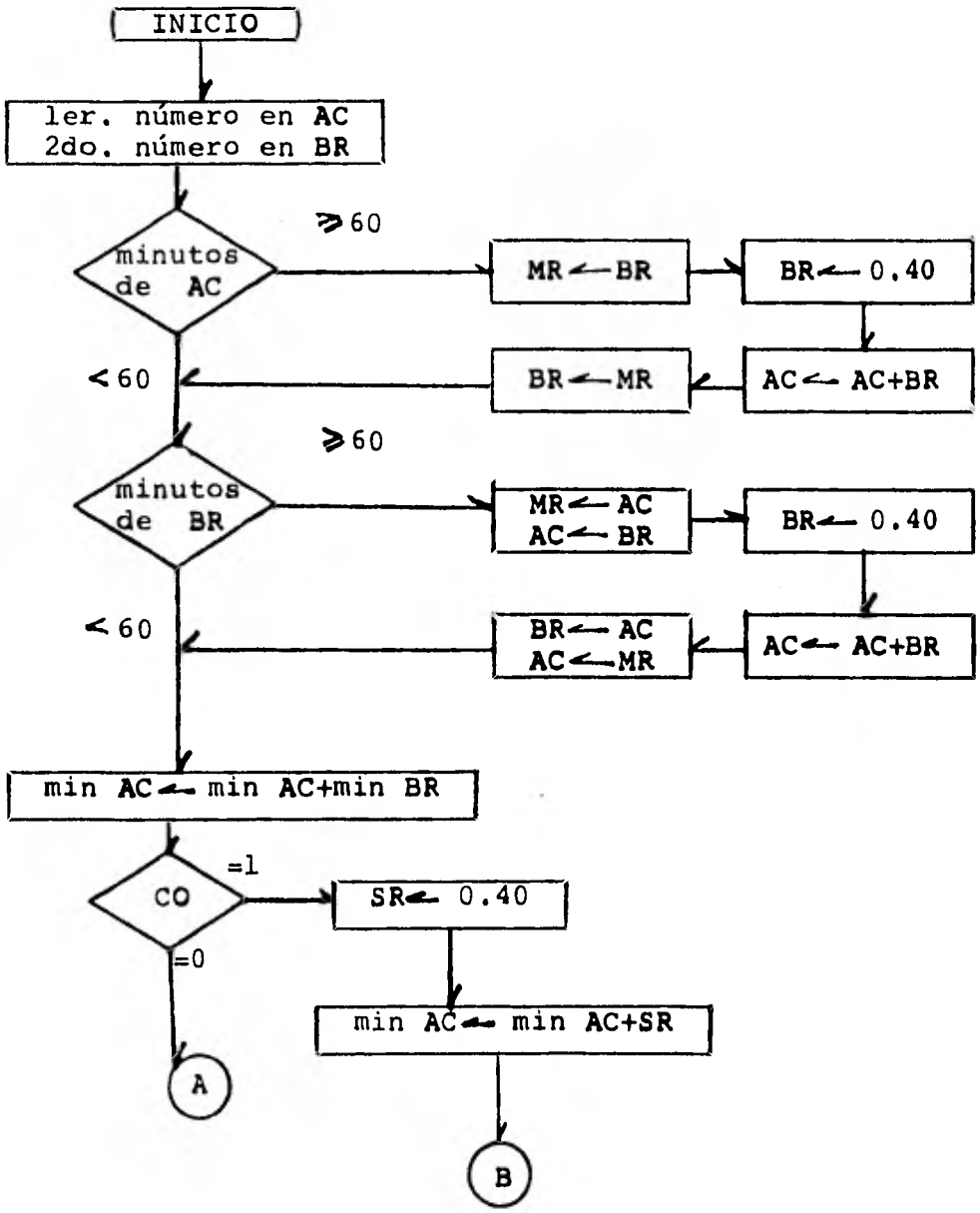
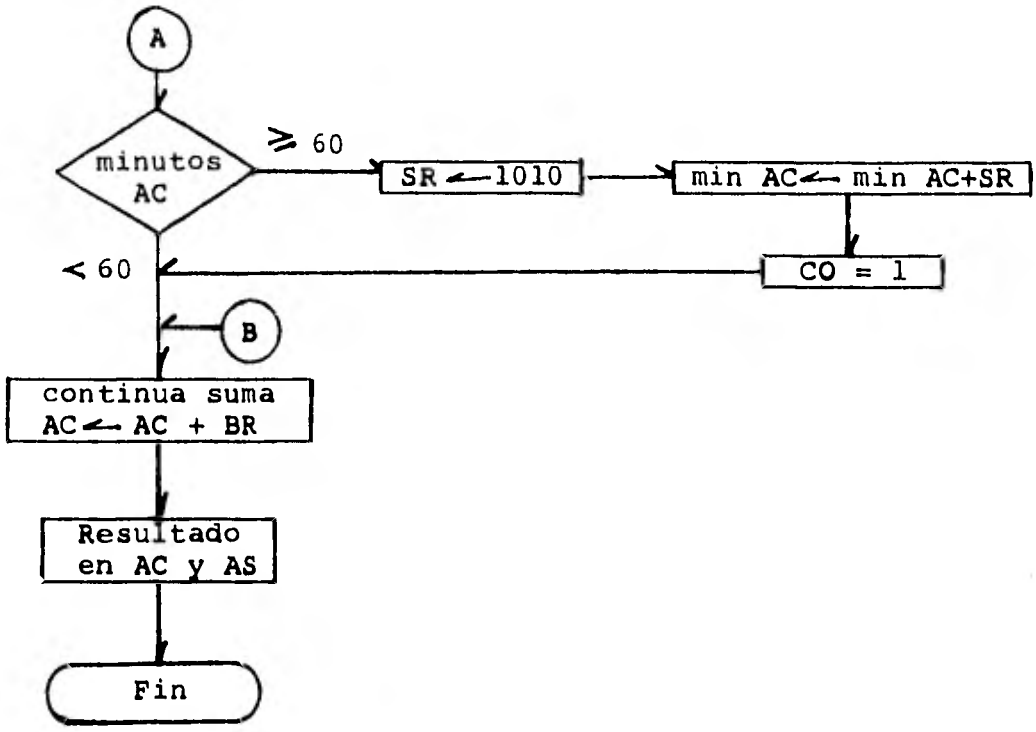


Figura IV.20 Diagrama de flujo para la suma con números de base sesenta.



Primero se checan si los números de entrada son mayores o iguales que sesenta, en caso afirmativo se les hace la conversión. Todos los movimientos mostrados entre registros es para poder usar el sumador tipo BCD. El registro MR, es un registro de memoria que sirve para guardar temporalmente el valor o los datos de otro registro. Luego se procede a sumar los minutos, en caso de que el carry de salida valga uno o de que la suma haya dado un número mayor de 59 se hacen las correcciones mencionadas antes.

El registro SR sirve para guardar constantes o valores para hacer conversiones y en algunos otros casos correcciones. Luego se procede a completar la suma de la manera acostumbrada. El resultado estará en AC y As y será un número de base sesenta.

ALGORITMO PARA LA RESTA CON NUMEROS DE BASE SESENTA.

Este algorítmo está basado en el de la resta con números decimales. Sin embargo, al igual que en el inciso anterior es necesario hacer ciertas conversiones. La primera, que es para convertir los números de entrada, es identica a la de la suma, presentada en el inciso anterior. La que si posee diferencias, es la conversión que es necesario hacer al resultado. Al estar restando los minutos de dos números se deberá checar que los minutos del minuendo sean mayor que los del sustraendo. Esto lo podemos determinar fácilmente como ya sabemos a partir del carry de salida de esta etapa. Si el carry de salida vale cero, entonces los minutos del sustraendo serán más grandes que los del minuendo y será necesario, hacerle una conversión a los minutos del resultado. Esta conversión se logra con restar 0.40 a los minutos del resultado, o sea sumarle seis (0110) que es el complemento a diez de cuatro. Sin embargo, como es necesario hacer la corrección para códigos invalidos o sea mayores que (1001) nueve, se le debe sumar 0110, así el valor final a sumar para hacer la conversión es 0110+0110 = 1100.

El diagrama de flujo para la resta con números de base sesenta, figura IV.21, es casi igual a su similar para la suma, sólo diferenciandose en la última conversión.

El significado de las variables es el mismo y el proceso seguido es idéntico.

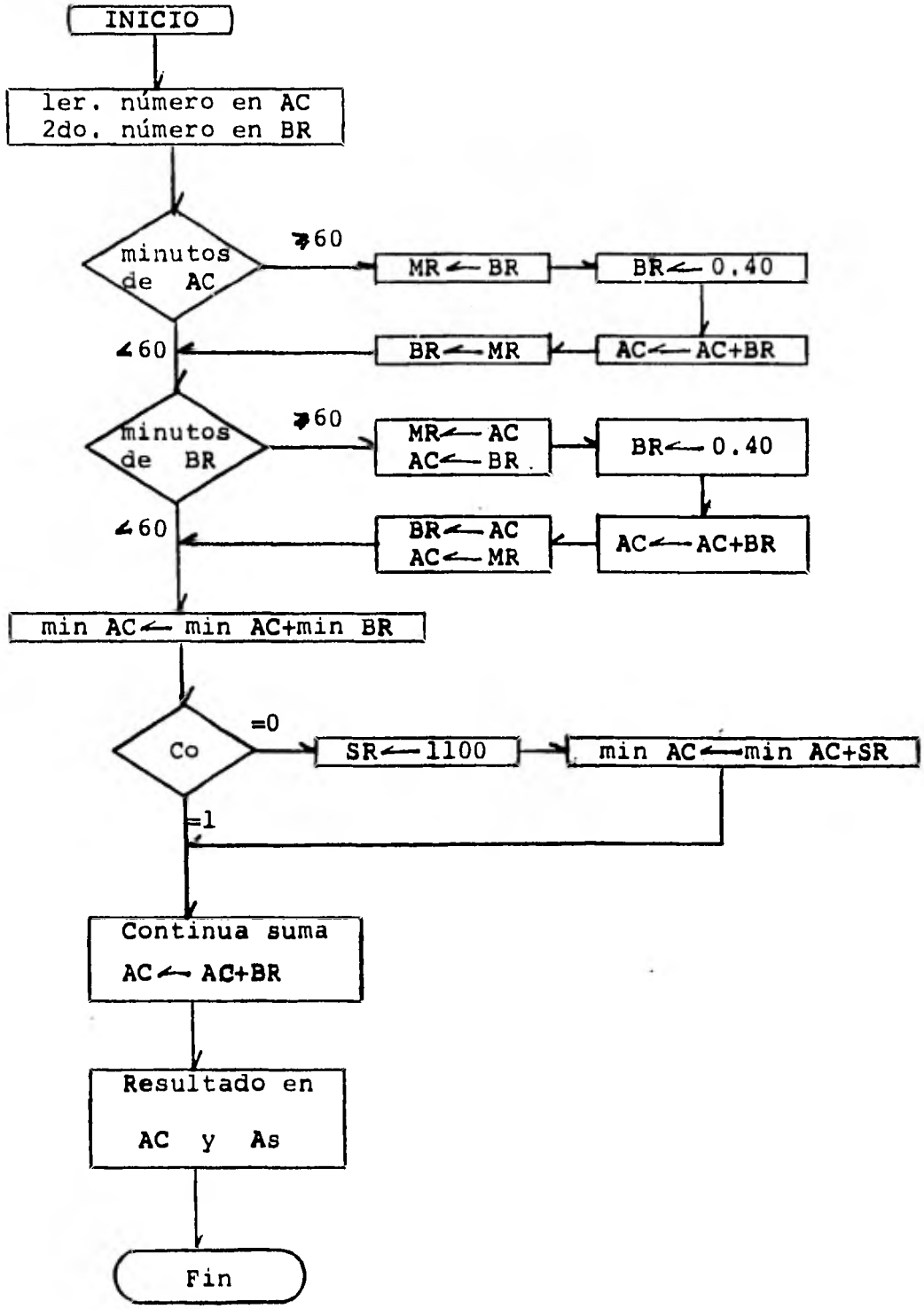


Figura IV.21 Diagrama de flujo para la resta con números en base sesenta.

C A P I T U L O V

SIMULACION POR MEDIO DE UN PROGRAMA DE COMPUTADORA.

En este capítulo se presenta el programa de computadora hecho para la simulación de la calculadora. En este se encuentran implementados los algorítmos del capítulo -- anterior. Este capítulo está ordenado de la siguiente- manera; primero se mencionan algunas características generales del programa. Seguidamente se presentan las variables empleadas y se describe el significado de estas. Luego se presenta el diagrama de flujo del programa y - se explica este. Finalmente se muestra el listado del- programa.

El programa fué desarrollado y corrido en una micro-computadora Apple II de 48 K de memoria en lenguaje BASIC- (Apple soft). Para mayor protección el programa se - - encuentra grabado en un disco floppy de 5 1/4 pulgadas- de diámetro y en casette. La impresión del listado se- hizo en una impresora de 80 caracteres por línea - - - (Epson 80).

En la tabla V.1 se presentan las variables empleadas -- acompañadas de una breve explicación de estas.

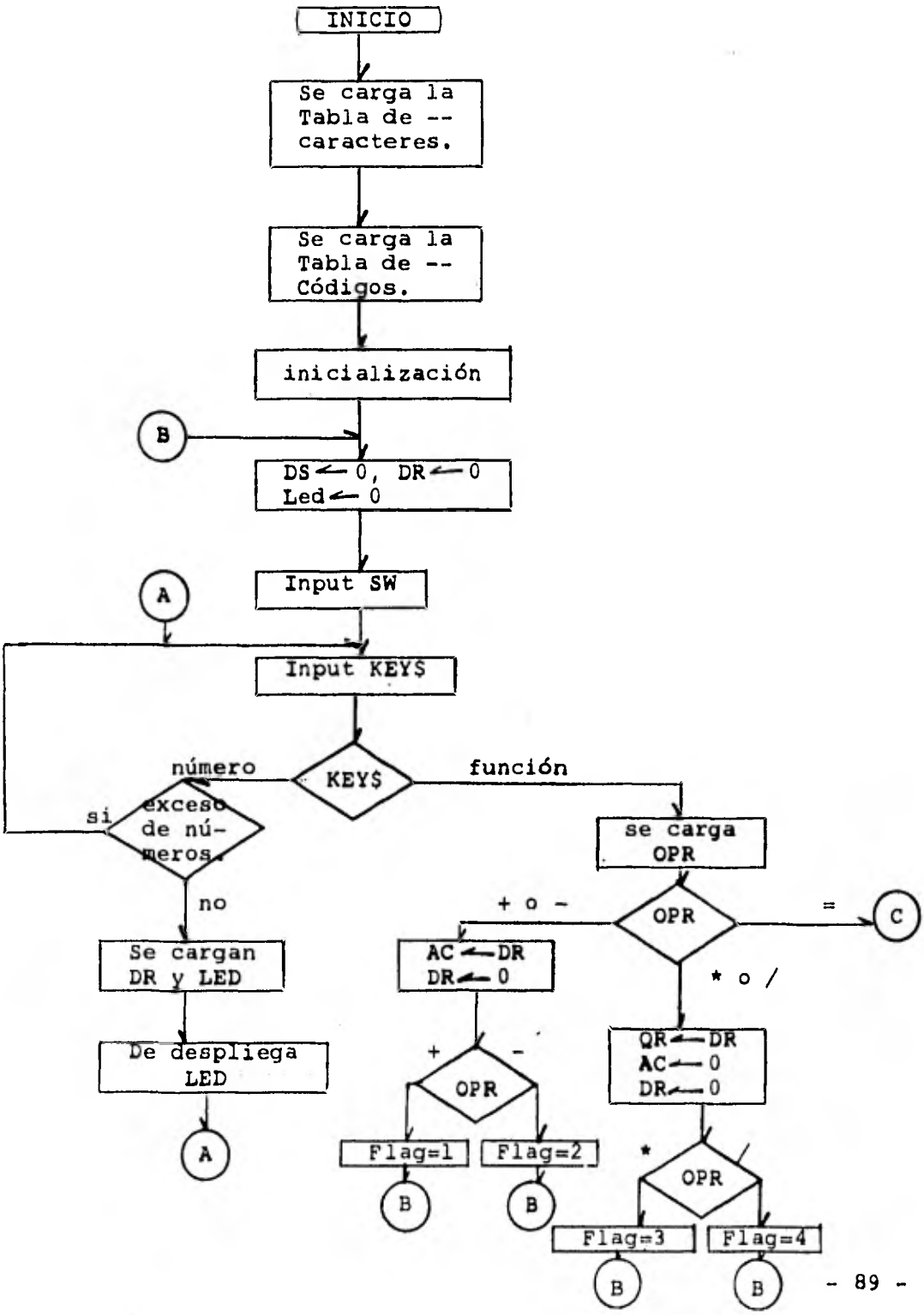
Variable	Nombre	Explicación
AC (10,4)	Acumulador	Va guardando la acumulación de los resultados. Puede- guardar 10 caracteres re-- presentados en codigo BCD- (4 bits).
BR (10,4)	Registro Buffer	Guarda información de en-- trada. Se usa en conjunto con AC como entrada al su- mador tipo BCD.

Variable	Nombre	Explicación
QR (10,4)	Registro Cociente	Es usado para efectuar las operaciones de multiplicación y división.
DR (10,4)	Reg. Despliegue	Contiene la representación en código BCD del número en el despliegue.
MR (10,4)	Registro memoria	Guarda temporalmente la información de otro registro.
CODE (16,4)	Tabla Código	Esta tabla contiene los códigos BCD de todos los caracteres y de las funciones.
CHAR$ (16)	Tabla Caracteres	En esta tabla se guardan todos los caracteres de los números y funciones.
LED (10)	Despliegue	Es la variable que se despliega, mostrando los caracteres del número.
SR (4)	Reg. Almacenar	Contiene constantes que se sumarán a otros registros, para hacer correcciones o conversiones.
E (4)	Registro Extra	Es usando en múltiplicaciones y divisiones para manejar posibles overflow intermedios.
CI	Carry Entrada	Representa al carry de entrada, puede valer cero o uno.
CO	Carry Salida	Representa al carry de salida, puede valer cero o uno.
OPR (4)	Reg. Operación	Guarda el código de la operación a efectuar.
AS	Signo AC	Representa el signo de AC
BS	Signo BR	Representa el signo de BR
QS	Signo QR	Representa el signo de QR
DS	Signo DR	Representa el signo de DR
SW	Switch 60	Cuando vale uno indica que se manejaran números en base 60.

Variables	Nombre	Explicación
Flag	Bandera	Tomará un valor de 1 a 4 dependiendo de la operación a realizar.
AVF	Overflow Adición	Si vale uno, entonces -- hay overflow en la adi-- ción.
DVF	Overflow División	Si vale uno, entonces -- hay overflow en la división.
Key$	Tecla	Pide información del teclado.
KBI	Teclado/Inihibidor	Inhibe cualquier información del teclado, hasta- que la entrada anterior- haya sido procesada.
PD	Punto decimal	Si vale uno, entonces se ha oprimido la tecla del punto.
CE	Limpia entrada	Limpia la última entrada.
C	Limpia	Limpia todos los regis-- tros.

Tabla V.1

En la siguiente figura se presenta el diagrama de flujo del programa.



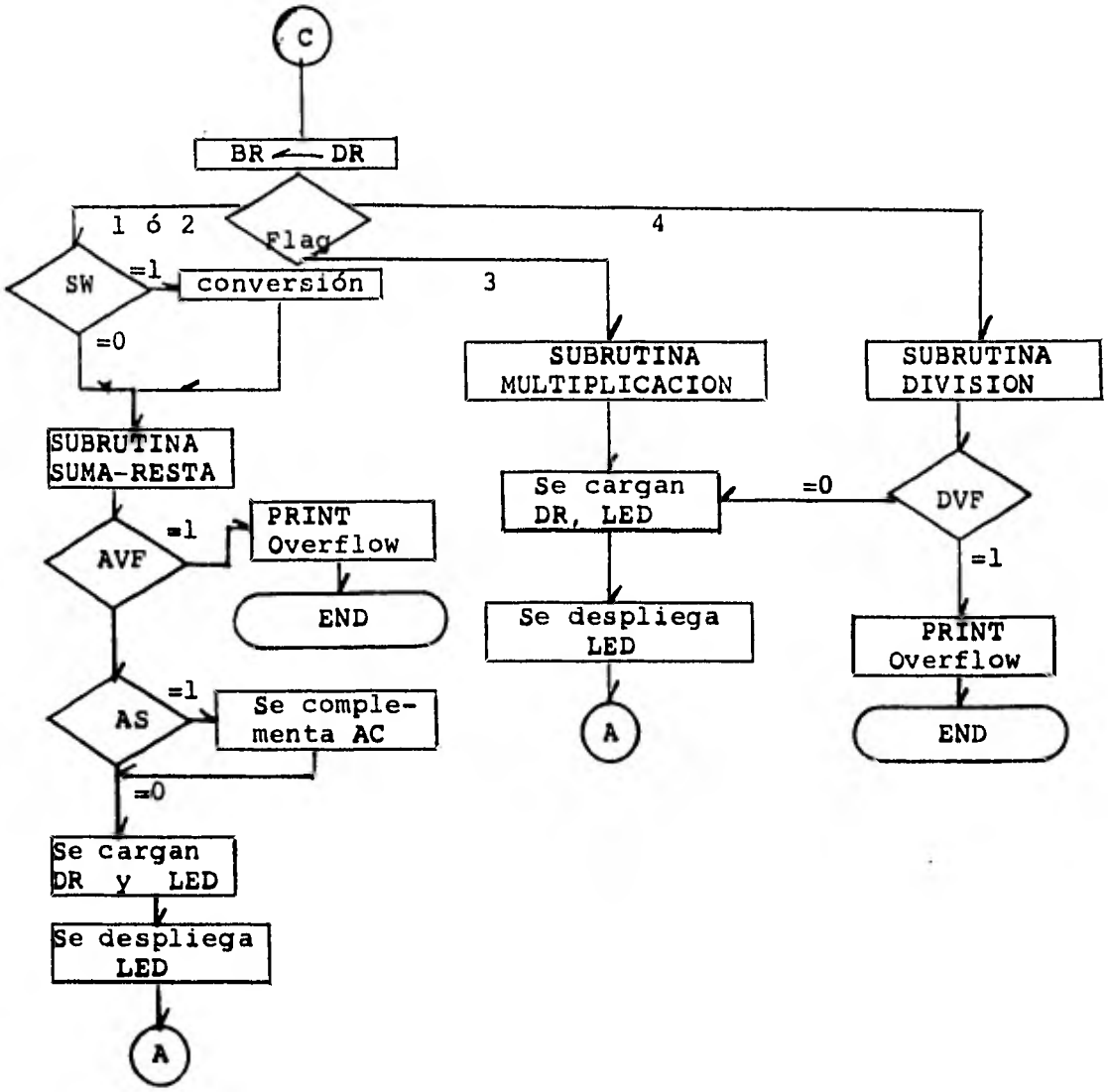


Figura V.1 Diagrama de flujo del Programa.

Primeramente se cargan dos tablas, la de los caracteres y la de los códigos. Después se inicializan algunas variables y registros. Luego se pide el valor de SW y el de KEY$. Enseguida se analiza si la última tecla oprimida, o sea el valor de KEY$ fué un número o una función. En caso de haber sido un número, se ve si no se ha excedido la capacidad del despliegue que es de diez números. En caso de no haber exceso se carga DR con los códigos correspondientes y luego se carga LED con los caracteres a desplegar, lo cual se hace en el siguiente paso.

Si la tecla oprimida fué una tecla indicando una función entonces se carga el registro de operaciones con el código de esa operación. Enseguida se pregunta por la naturaleza de esta, si fué una suma o una resta se carga el acumulador con lo que hay en el registro de despliegue y luego se limpia este último. Aquí también se le asigna un valor a la bandera, siendo uno para la suma y dos para una resta. En caso de que la tecla oprimida representa a una multiplicación o una división, se carga el registro de cociente con lo que contenga el registro de despliegue, seguidamente se limpia este último registro y el del acumulador. En las siguientes instrucciones se le asigna el valor de tres a la bandera si se trata de una multiplicación y el valor de cuatro para una división. En caso de que la tecla oprimida representa el signo de igual, se carga el registro del buffer con lo que contenga el registro de despliegue y se analiza el valor de la bandera. En caso de valer uno o --

dos, quiere decir que se trata de una suma o una resta- y lo primero que se hace es ver que valor tiene la va-- riable SW, si vale uno, entonces nos indica que se trabajará con números de base sesenta y se pasa a una su-- brutina para hacer las conversiones. Luego se llama a- la subrutina para sumar-restar y al regresar de esta -- calcula el bit de overflow, si este vale uno quiere decir que hubo overflow, en caso contrario se calcula el- bit de signo del resultado y si este llegase a valer -- uno, se complementa el acumulador. Finalmente se carga el registro de despliegue y se despliega la información después de esto se regresa a recibir otra tecla.

Cuando la bandera vale tres, entonces se llama a la subrutina de multiplicación. Cuando se ha terminado de - efectuar esta operación, se cargan los registros necesarios y se despliega la información. Posteriormente semanda el control al inicio para recibir otra información de entrada.

Cuando la bandera vale cuatro, sucede algo similar. La unica diferencia es que aquí se llama a la subrutina de la división y al regresar de esta se calcula el bit de- overflow.

En el programa se presentan primero las subrutinas.

Esto es con el fin de hacer más rápido la ejecución del programa.

```
LIST
```

```
10  GOTO 2080
20  REM
30  REM  SUBRUTINA PARA
40  REM  SUMAR/RESTAR
50  REM
60 CI = 0
70 LR = 0
80  IF FLAG = 2 THEN CI = 1
90 CI = F1
100  FOR L = 1 TO 10
110  IF FLAG = 1 GOTO 190
120  REM
130  REM  SI FLAG=2, SE
140  REM  COMPLEMENTA BR
150  REM
160 BR(L,4) =  NOT BR(L,4)
170 BR(L,2) =  NOT BR(L,2) AND BR
    (L,3) OR BR(L,2) AND  NOT BR
    (L,3)
180 BR(L,1) =  NOT BR(L,1) AND  NOT
    BR(L,2) AND  NOT BR(L,3)
190  FOR M = 4 TO 1 STEP  - 1
200 CR = AC(L,M) AND BR(L,M) OR A
    C(L,M) AND CI OR BR(L,M) AND
    CI
210 AC(L,M) =  NOT AC(L,M) AND  NOT
    BR(L,M) AND CI OR  NOT AC(L,
    M) AND BR(L,M) AND  NOT CI OR
    AC(L,M) AND  NOT BR(L,M) AND
     NOT CI OR AC(L,M) AND BR(L,
```

```
      M) AND CI
  220 CI = CR
  230 NEXT M
  240 REM
  250 REM  SE CALCULA EL CARRY-OUT
  260 REM
  270 CO = CI OR AC(L,1) AND AC(L,2
      ) OR AC(L,1) AND AC(L,3)
  280 REM
  290 REM  SE HACE CORRECCION A
  300 REM  LA SUMA, SI ES NECESARI
      O
  310 REM   AC > 1001 => AC=AC +
      0110
  320 REM
  330 CI = 0
  340 GOSUB 560
  350 SR(2) = CO
  360 SR(3) = CO
  370 GOSUB 460
  380 IF SW = 1 GOTO 5920
  390 CI = CO
  400 NEXT L
  410 RETURN
  420 REM
  430 REM SUBRUTINA
  440 REM  ACI <-- ACI + SR
  450 REM
  460 FOR M1 = 4 TO 1 STEP  - 1
  470 CR = AC(L,M1) AND SR(M1) OR A
      C(L,M1) AND CI OR SR(M1) AND
      CI
```

```
480 AC(L,M1) = NOT AC(L,M1) AND
     NOT SR(M1) AND CI OR  NOT A
     C(L,M1) AND SR(M1) AND  NOT
     CI OR AC(L,M1) AND  NOT SR(M
     1) AND  NOT CI OR AC(L,M1) AND
     SR(M1) AND CI
490 CI = CR
500  NEXT M1
510  RETURN
520  REM
530  REM  SUBRUTINA
540  REM  SR <-- 0
550  REM
560  FOR J = 1 TO 4
570 SR(J) = 0
580  NEXT J
590  RETURN
600  REM
610  REM  SUBRUTINA
620  REM  LED <-- 0
630  REM  DR <-- 0
640  REM  FB <-- 0
650  REM
660  FOR L = 1 TO 10
670 LED(L) = 0
680  FOR M = 1 TO 4
690 DR(L,M) = 0
700  NEXT M
710  NEXT L
720 DB = 0
730 FB = 0
740  RETURN
750  REM
```

```
760  REM  SUBRUTINA
770  REM  AC <-- QR
780  REM
790  FOR L = 1 TO 10
800  FOR M = 1 TO 4
810 AC(L,M) = QR(L,M)
820  NEXT M
830  NEXT L
840  RETURN
850  REM
860  REM  DSHL E->AC->QR
870  REM
880  FOR M = 1 TO 4
890 E(M) = AC(10,M)
900  NEXT M
910  FOR L = 9 TO 1 STEP  - 1
920  FOR M = 1 TO 4
930 AC(L + 1,M) = AC(L,M)
940  NEXT M
950  NEXT L
960  FOR M = 1 TO 4
970 AC(1,M) = QR(10,M)
980  NEXT M
990  FOR L = 9 TO 1 STEP  - 1
1000  FOR M = 1 TO 4
1010 QR(L + 1,M) = QR(L,M)
1020  NEXT M
1030  NEXT L
1040  FOR M = 1 TO 4
1050 QR(1,M) = 0
1060  NEXT M
1070  RETURN
1080  REM
```

```
1090  REM  SUBRUTINA
1100  REM  BR <-- AC
1110  REM  AC <--0
1120  REM
1130  FOR L = 1 TO 10
1140  FOR M = 1 TO 4
1150 BR(L,M) = AC(L,M)
1160 AC(L,M) = 0
1170  NEXT M
1180  NEXT L
1190  RETURN
1200  REM
1210  REM  ESTA SUBRUTINA
1220  REM  CARGA LOS RE-
1230  REM  GISTROS DE
1240  REM  DESPLIEGUE
1250  REM
1260  FOR L = 1 TO 10
1270  FOR J = 1 TO 16
1280  FOR M = 1 TO 4
1290  IF DR(L,M) < > CODE(J,M) GOTO
      1330
1300  NEXT M
1310 LED(L) =  VAL ( MID$ (CHAR$,
      J,1))
1320  GOTO 1360
1330  NEXT J
1340  PRINT "ERROR EN SUB/DISPLAY
1350  GOTO 2400
1360  NEXT L
1370  IF AS = 1 THEN  PRINT "-";
1380  RETURN
```

```
1390  REM
1400  REM  SUBRUTINA DE
1410  REM  DESPLIEGUE
1420  REM
1430  FOR L = 10 TO 1 STEP  - 1
1440  IF L = 2 THEN  PRINT ".";
1450  PRINT LED(L);
1460  NEXT L
1470  PRINT " "
1480  RETURN
1490  REM
1500  REM  SUBRUTINA PARA HACER :


1510  REM  AC <-- DR
1520  REM
1530  FOR L = 1 TO 10
1540  FOR M = 1 TO 4
1550 AC(L,M) = DR(L,M)
1560  NEXT M
1570  NEXT L
1580  RETURN
1590  REM
1600  REM  SUBRUTINA
1610  REM  BR <-- MR
1620  REM
1630  FOR L4 = 1 TO 10
1640  FOR M4 = 1 TO 4
1650 BR(L4,M4) = MR(L4,M4)
1660  NEXT M4
1670  NEXT L4
1680  RETURN
1690  REM
1700  REM  SUBRUTINA
```

```
1710  REM  MR <-- BR
1720  REM
1730  FOR L4 = 1 TO 10
1740  FOR M4 = 1 TO 4
1750 MR(L4,M4) = BR(L4,M4)
1760  NEXT M4
1770  NEXT L4
1780  RETURN
1790  REM
1800  REM  SUBRUTINA
1810  REM  DR <-- AC
1820  REM
1830  FOR L = 1 TO 10
1840  FOR J = 1 TO 4
1850 DR(L,J) = AC(L,J)
1860  NEXT J
1870  NEXT L
1880  RETURN
1890  REM
1900  REM  SUBRUTINA
1910  REM  NUMEROS BASE SESENTA
1920  REM
1930  FOR L = 1 TO 10
1940  FOR M = 1 TO 4
1950 BR(L,M) = 0
1960  NEXT M
1970  NEXT L
1980  REM  MR<--BR BR<--0
1990 BR(2,2) = 1
2000  REM  BR <- 0,40
2010 FT = FLAG
2020 SW = 0
2030 FLAG = 1
```

```
2040  GOSUB 60
2050 FLAG = FT
2060 SW = 1
2070  RETURN
2080  DIM CODE(16,4),CHAR$(16)
2090  DIM DR(10,4),LED(14)
2100  DIM QR(10,4),SR(4)
2110  DIM AC(10,4),BR(10,4),OPR(4
      )
2120  DIM MR(10,4),E(4)
2130  DATA  "0123456789+-*/=."
2140  REM
2150  REM  SE LEEN LOS CARACTERES
2160  REM
2170  READ CHAR$
2180  DATA  0,0,0,0,  0,0,0,1,
            0,0,1,0,  0,0,1,1
2190  DATA  0,1,0,0,  0,1,0,1,
            0,1,1,0,  0,1,1,1
2200  DATA  1,0,0,0,  1,0,0,1,
            1,0,1,0,  1,0,1,1
2210  DATA  1,1,0,0,  1,1,0,1,
            1,1,1,0,  1,1,1,1
2220  REM
2230  REM  SE LEEN LOS CODIGOS
2240  REM
2250  FOR L = 1 TO 16
2260  FOR M = 1 TO 4
2270  READ CODE(L,M)
2280  NEXT M
2290  NEXT L
2300  GET SW
```

```
2310  REM                                                        1
2320  REM  SE INICIALIZAN LAS VAR
     IABLES
2330  REM
2340 C = 0
2350 AS = 0
2360 BS = 0
2370 QS = 0
2380 CE = 0
2390  GOSUB 660
2400 FLAG = 0
2410 ID = 2
2420 KD = 0
2430 PD = 0
2440 KBI = 0
2450  REM
2460  REM  EN ESPERA DE INFOR-
2470  REM  MACION DE ENTRADA
2480  REM
2490  GET KEY$
2500 KBI = 1
2510  REM
2520  REM  SE DETECTA O IDENTIFIC
     A
2530  REM  LA TECLA OPRIMIDA
2540  REM
2550  FOR J = 1 TO 16
2560  IF KEY$ < > MID$ (CHAR$,J
     ,1) THEN  GOTO 2600
2570  LET F = J
2580  IF J = 16 THEN PD = 1
2590  GOTO 2670
2600  NEXT J
```

```
2610  GOTO 2440
2620  REM
2630  REM  SE CHECA SI ES :
2640  REM  FUNCION ---3170
2650  REM  NUMERO ---2760
2660  REM
2670  IF F = 16 GOTO 3100
2680  IF CODE(F,1) = 0 GOTO 2760
2690  IF CODE(F,2) = 1 GOTO 3170
2700  IF CODE(F,3) = 1 GOTO 3170
2710  REM
2720  REM  VERIFICA QUE NO
2730  REM  HAYA EXCESO DE
2740  REM  NUMEROS DE ENTRADA
2750  REM
2760  IF FB = 1 GOTO 2790
2770  GOSUB 660
2780 FB = 1
2790  IF ID = 1 GOTO 2440
2800  IF PD = 1 GOTO 2840
2810  IF ID = 10 GOTO 2440
2820 ID = ID + 1
2830  GOTO 2900
2840 ID = 2 - KD
2850 KD = 1
2860  GOTO 3020
2870  REM
2880  REM  SE CARGA DR Y LED
2890  REM
2900  FOR L = ID TO 3 STEP - 1
2910 J = L - 1
2920 LED(L) = LED(J)
2930  FOR M = 1 TO 4
```

```
2940 DR(L,M) = DR(J,M)
2950  NEXT M
2960  NEXT L
2970 LED(3) =  VAL (KEY$)
2980  FOR M = 1 TO 4
2990 DR(3,M) = CODE(F,M)
3000  NEXT M
3010  GOTO 3100
3020 LED(ID) =  VAL (KEY$)
3030  FOR M = 1 TO 4
3040 DR(ID,M) = CODE(F,M)
3050  NEXT M
3060  REM
3070  REM  SE LLAMA A LA SUBRUTIN
      A
3080  REM  DE DESPLIEGUE
3090  REM
3100  GOSUB 1430
3110  GOTO 2440
3120  REM
3130  REM  CARGA EL REGISTRO
3140  REM  DE INSTRUCCIONES
3150  REM  OPR <-- INT
3160  REM
3170  FOR M = 1 TO 4
3180 OPR(M) = CODE(F,M)
3190  NEXT M
3200 FB = 0
3210  IF OPR(2) = 0 GOTO 3300
3220  IF OPR(3) = 0 GOTO 3500
3230  GOTO 3680
3240  REM
3250  REM  SE CARGAN LOS REGISTRO
```

```
      S
3260  REM  PARA EFECTUAR UNA
3270  REM  SUMA O RESTA :
3280  REM  AC <- DR , DR <- 0
3290  REM
3300  IF FLAG = 1 OR FLAG = 2 GOTO
      3680
3310 FLAG = 1
3320 AS = DS
3330  FOR L = 1 TO 10
3340  FOR M = 1 TO 4
3350 AC(L,M) = DR(L,M)
3360 DR(L,M) = 0
3370  NEXT M
3380  NEXT L
3390 DS = 0
3400  IF OPR(4) = 1 THEN FLAG = 2

3410  IF OPR(4) = 1 THEN DS = 1
3420  GOTO 2410
3430  REM
3440  REM  SE CARGAN LOS REGISTRO
      S
3450  REM  PARA EFECTUAR UNA
3460  REM  MULTIPLICACION O
3470  REM  DIVISION : QR <- DR
3480  REM  AC <- 0 , DR <- 0
3490  REM
3500 FLAG = 3
3510 AS = DS
3520 QS = AS
3530  FOR L = 1 TO 10
3540  FOR M = 1 TO 4
```

```
3550 QR(L,M) = DR(L,M)
3560 AC(L,M) = 0
3570 DR(L,M) = 0
3580  NEXT M
3590  NEXT L
3600  IF OPR(4) = 1 THEN FLAG = 4
3610  GOTO 2410
3620  REM
3630  REM  SE OPRIMIO LA TECLA
3640  REM  DE : "="
3650  REM  SE CARGA EL REGISTRO #
3660  REM  BR <- DR
3670  REM
3680  IF FLAG = 0 GOTO 2410
3690  FOR L = 1 TO 10
3700  FOR M = 1 TO 4
3710 BR(L,M) = DR(L,M)
3720  NEXT M
3730  NEXT L
3740 BS = DS
3750  IF FLAG = 2 THEN BS = 1
3760  ON FLAG GOTO 3840,3840,4250
      ,4950
3770  REM
3780  REM  ALGORITMO PARA SUMAR/R
      ESTAR
3790  REM  NUMEROS EN CODIGO BCD
3800  REM  REPRESENTADOS EN
3810  REM  DIEZ-COMPLEMENTO
3820  REM  AC <- AC + BR
3830  REM
```

```
3840  IF SW = 1 GOTO 5680
3850  IF AB = 0 GOTO 3930
3860  GOSUB 1730
3870 FR = FLAG
3880 FLAG = 2
3890  GOSUB 1130
3900  GOSUB 60
3910 FLAG = FR
3920  GOSUB 1630
3930  GOSUB 60
3940  REM
3950  REM  SE CHECA SI HAY
3960  REM  OVERFLOW
3970  REM
3980 F = AS AND BS OR AS AND CI OR
     BS AND CI
3990 AVF = F AND  NOT CI OR  NOT
     F AND CI
4000  IF AVF = 1 GOTO 5620
4010 AS =  NOT CI AND AS OR AS AND
     BS OR  NOT CI AND BS
4020  REM
4030  REM  SI AS=1, SE COM
4040  REM  PLEMENTA AC
4050  REM
4060  IF AS = 0 GOTO 4200
4070 FR = FLAG
4080 FLAG = 2
4090  GOSUB 1130
4100  GOSUB 60
4110 FLAG = FR
4120 AS = 1
4130  REM
```

```
4140  REM  SE LLAMA A LA
4150  REM  SUBRUTINA PARA
4160  REM  CARGAR LOS RE-
4170  REM  GISTROS DE DES-
4180  REM  PLIEGUE
4190  REM
4200  GOSUB 1830
4210  GOSUB 1260
4220  GOSUB 1430
4230 DS = AS
4240  GOTO 2400
4250  REM
4260  REM  OPERACION DE MULTIPLIC
      ACION
4270  REM  RESULTADO EN : AC Y QR

4280  REM
4290  REM  SE DETERMINA EL SIGNO
4300  REM
4310 AS = QS AND  NOT BS OR BS AND
      NOT QS
4320  FOR M = 1 TO 4
4330 E(M) = 0
4340  NEXT M
4350  FOR L1 = 1 TO 12
4360  IF L1 = 11 OR L1 = 12 GOTO
      4430
4370  FOR M = 1 TO 4
4380  IF QR(1,M) <  > 0 THEN  GOTO
      4610
4390  NEXT M
4400  REM
4410  REM  DSHR EACQR
```

```
4420  REM
4430  FOR L = 1 TO 9
4440  FOR M = 1 TO 4
4450 QR(L,M) = QR(L + 1,M)
4460  NEXT M
4470  NEXT L
4480  FOR M = 1 TO 4
4490 QR(10,M) = AC(1,M)
4500  NEXT M
4510  FOR L = 1 TO 9
4520  FOR M = 1 TO 4
4530 AC(L,M) = AC(L + 1,M)
4540  NEXT M
4550  NEXT L
4560  FOR M = 1 TO 4
4570 AC(10,M) = E(M)
4580 E(M) = 0
4590  NEXT M
4600  GOTO 4840
4610 FLAG = 1
4620  GOSUB 560
4630  GOSUB 60
4640  REM
4650  REM  SE OBTIENE E A PARTIR
4660  REM  DEL CARRY OUT E=E+CO
4670  REM
4680  FOR M = 4 TO 1 STEP  - 1
4690 G = E(M)
4700 E(M) = G AND  NOT CO OR  NOT
     G AND CO
4710 CO = CO AND G
4720  NEXT M
4730  REM
```

```
4740  REM  QN =QN - 1
4750  REM
4760 CI = 0
4770  FOR M = 4 TO 1 STEP  - 1
4780 SR(M) = 1
4790 G = QR(1,M)
4800 QR(1,M) = G AND  NOT SR(M) AND
      NOT CI OR  NOT G AND SR(M) AND
      NOT CI OR  NOT G AND  NOT S
     R(M) AND CI OR G AND SR(M) AND
      CI
4810 CI = G AND SR(M) OR G AND CI
      OR SR(M) AND CI
4820  NEXT M
4830  GOTO 4370
4840  NEXT L1
4850 DS = AS
4860  REM
4870  REM  SE LLAMA A LAS SUBRUTI
      NAS
4880  REM  DE DESPLIEGUE
4890  REM
4900  GOSUB 790
4910  GOSUB 1830
4920  GOSUB 1260
4930  GOSUB 1430
4940  GOTO 2400
4950  REM
4960  REM  ALGORITMO PARA
4970  REM  LA DIVISION
4980  REM  COCIENTE EN QR
4990  REM  RESIDUO EN AC
5000  REM
```

```
5010  REM  SE DETERMINA
5020  REM  EL SIGNO
5030  REM
5040  FOR L = 1 TO 10
5050  FOR M = 1 TO 4
5060 AC(L,M) = 0
5070  NEXT M,L
5080 QS = AS AND  NOT BS OR  NOT
      AS AND BS
5090  FOR M = 1 TO 4
5100 E(M) = 0
5110  NEXT M
5120  REM
5130  REM  SE CHECA OVERFLOW
5140  REM
5150 F1 = 1
5160 DC = AC(10,1) OR AC(10,2) OR
      AC(10,3) OR AC(10,4)
5170 FLAG = 2
5180  GOSUB 60
5190 DVF = DC AND CO
5200  IF DVF = 1 THEN  GOTO 5620
5210 F1 = 0
5220  GOSUB 60
5230 F1 = 1
5240  GOSUB 880
5250 FLAG = 2
5260 DVF = 0
5270  FOR L1 = 1 TO 12
5280  IF L1 = 1 THEN  GOTO 5300
5290  GOSUB 880
5300  GOSUB 60
5310 FLAG = 1
```

```
5320  IF CO = 1 GOTO 5390
5330 FLAG = 2
5340 F1 = 0
5350  GOSUB 60
5360 F1 = 1
5370 FLAG = 2
5380  GOTO 5320
5390  GOSUB 560
5400 SR(4) = 1
5410 CI = 0
5420  FOR M = 4 TO 1 STEP  - 1
5430 G = QR(1,M)
5440 QR(1,M) =  NOT G AND  NOT SR
     (M) AND CI OR  NOT G AND SR(
     M) AND  NOT CI OR G AND  NOT
     SR(M) AND  NOT CI OR G AND S
     R(M) AND CI
5450 CI = G AND SR(M) OR G AND CI
     OR SR(M) AND CI
5460  NEXT M
5470 FLAG = 1
5480  GOSUB 560
5490  GOSUB 60
5500  IF CO = 1 GOTO 5390
5510  GOTO 5330
5520  NEXT L1
5530  REM
5540  REM  SE LLAMA A LAS SUBRUTI
      NAS
5550  REM  DE DESPLIEGUE
5560  REM
5570  GOSUB 790
5580  GOSUB 1830
```

```
5590  GOSUB 1260
5600  GOSUB 1430
5610  GOTO 2400
5620  PRINT "OVF": STOP
5630  REM
5640  REM  SUBRUTINA
5650  REM  CONVERSION PARA NUMERO
      S
5660  REM  DE BASE SESENTA
5670  REM
5680  IF AC(2,2) AND AC(2,3) OR A
      C(2,1) AND  NOT AC(2,2) GOTO
      5710
5690  GOTO 5740
5700  REM  FUE MAYOR QUE 60
5710  GOSUB 1730
5720  GOSUB 1930
5730  GOSUB 1630
5740  IF BR(2,2) AND BR(2,3) OR B
      R(2,1) AND  NOT BR(2,2) GOTO
      5760
5750  GOTO 3890
5760  REM  FUE MAYOR QUE 60
5770  FOR Z = 1 TO 10
5780  FOR V = 1 TO 4
5790  MR(Z,V) = AC(Z,V)
5800  AC(Z,V) = BR(Z,V)
5810  NEXT V
5820  NEXT Z
5830  REM  BR<--AC  AC<--BR
5840  GOSUB 1930
5850  FOR Z = 1 TO 10
5860  FOR V = 1 TO 4
```

```
5870 BR(Z,V) = AC(Z,V)                                          1
5880 AC(Z,V) = MR(Z,V)
5890  NEXT V
5900  NEXT Z
5910  GOTO 3850
5920  IF L < > 2 GOTO 390
5930  IF FLAG = 1 GOTO 6000
5940  IF FLAG = 2 GOTO 6190
5950  GOTO 390
5960  REM
5970  REM  CORRECION PARA SUMA DE
5980  REM  NUMEROS DE BASE BEBENT
      A
5990  REM
6000  IF CO = 1 GOTO 6040
6010 F = AC(2,2) AND AC(2,3) OR A
      C(2,1) AND  NOT AC(2,2)
6020  IF F = 1 GOTO 6090
6030  GOTO 390
6040  GOSUB 560
6050 BR(2) = 1
6060 CI = 0
6070  GOSUB 460
6080  GOTO 390
6090  GOSUB 560
6100 BR(1) = 1
6110 BR(3) = 1
6120  GOSUB 460
6130 CO = 1
6140  GOTO 390
6150  REM
6160  REM  CORRECION PARA RESTA D
```

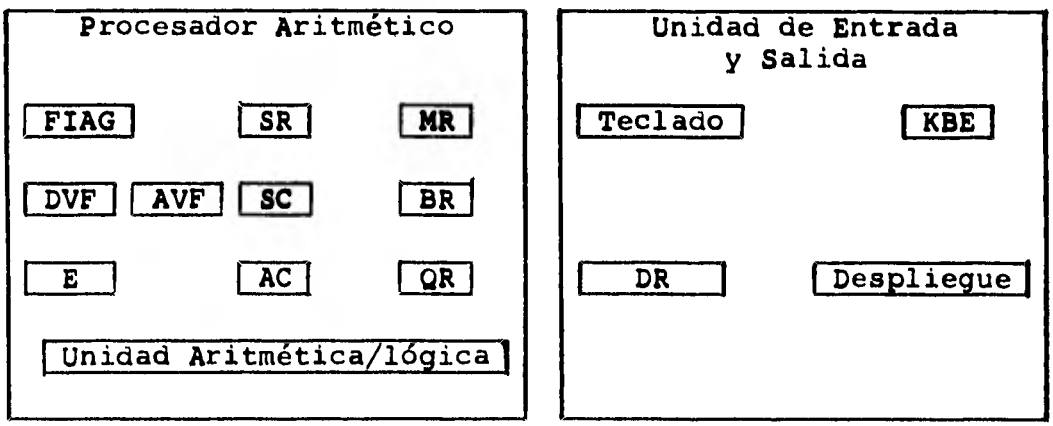
```
      E                                                              1
6170  REM  NUMEROS DE BASE SESENT
      A
6180  REM
6190  IF CO = 0 GOTO 6210
6200  GOTO 390
6210  GOSUB 560
6220 SR(1) = 1
6230 SR(2) = 1
6240  GOSUB 460
6250  GOTO 390

JPR#0
```

C A P I T U L O VI

ANALOGIA CON CIRCUITOS ELECTRONICOS.

En este capítulo se presenta en forma general la configuración de la calculadora electrónica.

En la figura VI.1 se presenta esta configuración y está dividida en diferentes unidades. En la unidad del procesador aritmético están los circuitos y registros necesarios para efectuar las operaciones. En la unidad de control se muestran circuitos y registros necesarios para controlar las micro-operaciones de la calculadora electrónica. En la unidad de entrada y salida se muestran los circuitos y registros empleados para recibir la información de entrada así como para desplegarla y también comprende al teclado. Finalmente la última unidad es la de la fuente de poder.



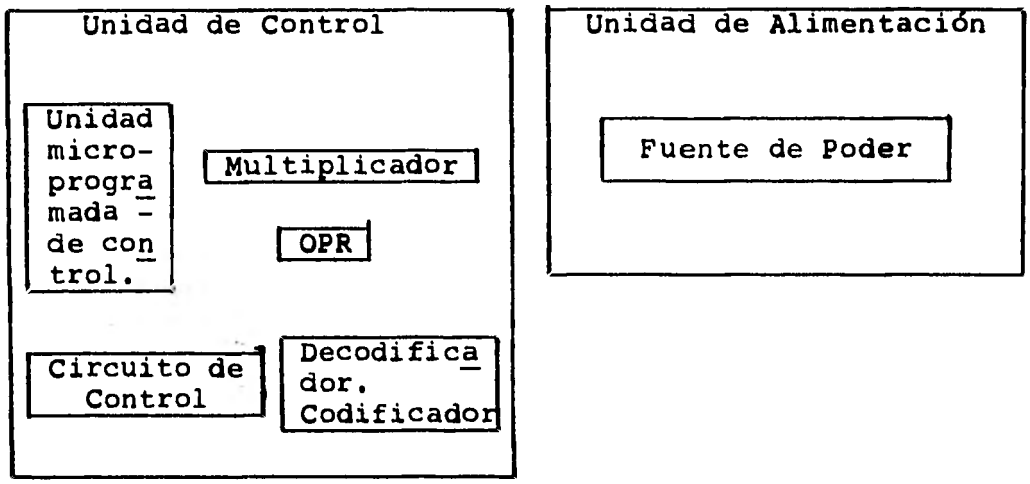


Figura VI.1 Configuración de la calculadora Electrónica.

Empecemos analizando la unidad del procesador aritmético. Vemos que el primer registro que tenemos es el de la bandera (flag), como este registro puede tomar los valores de 0 a 4, necesitamos que este formado por 3 bits, tal como se muestra en la figura VI.2.

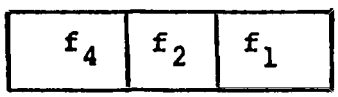


Figura VI.2 Registro de bandera.

Este registro podrá estar formado por flip-flop, tipo D y con una entrada para poder limpiarlos todos sus bits (clear).

El registro de almacenaje SR, solo deberá poder guardar un número o valor representado por cuatro bits (código BCD), por lo tanto este tendrá la configuración mostrada en la figura VI.3

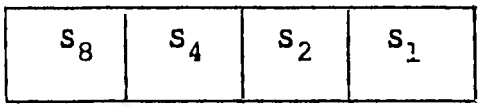


Figura VI.3 Registro de almacenamiento de un caracter.

Este registro también deberá estar formado por flip- -- flop, tipo D y con una entrada para limpiarlo.

Los registros para indicar overflow, ya sea en la suma- o en la división, solo podrán tomar los valores de cero o uno. Por lo tanto con un flip-flop, Tipo D será sufi ciente, figura VI.4.

Figura VI.4 Configuración para los flip-flop de overflow.

El registro SC, es un contador que nos indica el número máximo de números que se utilizarán para efectuar una - operación y el valor máximo que puede tomar es diez, -- por lo tanto tendrá una configuración similar al registro de almacenamiento SR y deberá ser de ese mismo tipo y con las mismas características.

Los registros BR y MR tienen una estructura idéntica y- solo difieren en el uso que se les da. El registro del buffer guardará un número que será utilizado junto con- el del AC para efectuar una operación. Mientras que el registro de memoria MR guardará temporalmente la información de otro registro. Estos dos registros podrán --

guardar un máximo de 10 caracteres representados en código BCD, por lo tanto su estructura es como la mostrada en la figura VI.5.

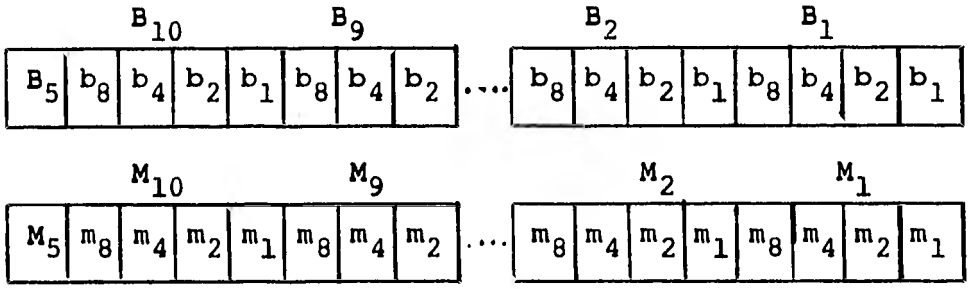


Figura VI.5 Configuración de los registros BR y MR.

Estos registros deberán estar compuestos por flip-flops tipo D y con una entrada para limpiar a todos estos - - (Clear). También deberán contar un bit extra para guardar la información sobre el signo.

Los registros del acumulador AC y del cociente QR son - del mismo tamaño y del mismo tipo pero deberán junto -- con el registro E poder hacer corrimientos tanto hacia- la derecha como hacia la izquierda. El registro E es - idéntico al registro SR. Lo anterior se muestra en la- figura VI.6.

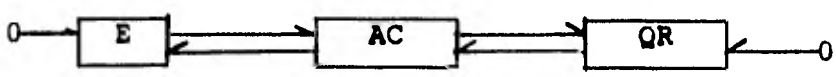


Figura VI.6 Registro E, AC y QR. Deberán poder reali-- zar corrimientos decimales tanto a la derecha como a la izquierda.

Finalmente dentro de la unidad del procesador aritmético tenemos la unidad aritmética lógica. Esta deberá -- ser capaz de obtener el complemento a diez de un número cuando este sea negativo y de poder hacer correcciones- cuando el resultado esté representado por número en un- código BCD invalido. Lo anterior se muestra en la figura VI.7.

Todo lo que está mostrado en la figura anterior es lo - que llamaremos el sumador tipo BCD con unidades de complementación y de corrección.

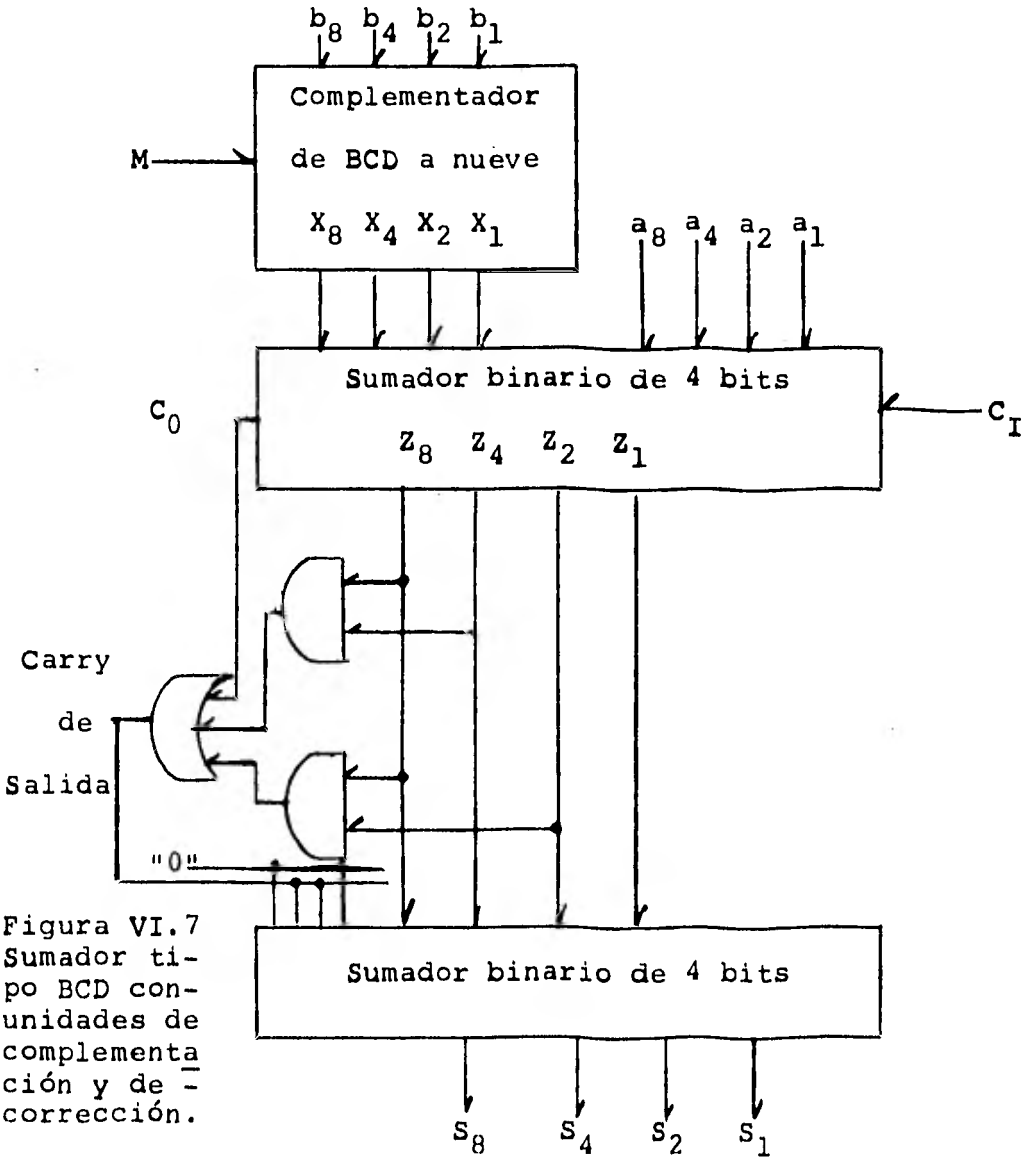


Figura VI.7 Sumador tipo BCD con- unidades de complementación y de - corrección.

Cuando M vale uno entonces se hará uso del circuito complementador, en caso contrario se dejará pasar lo mismo que haya a la entrada de este circuito. Si el carry de salida vale uno, entonces se sumará 0110 al resultado - para obtener una representación valida en código BCD.

Como la suma se hará por el método de dígito en serie - y bits en paralelo, solo será necesario uno de estas -- unidades. Ver figura VI.8.

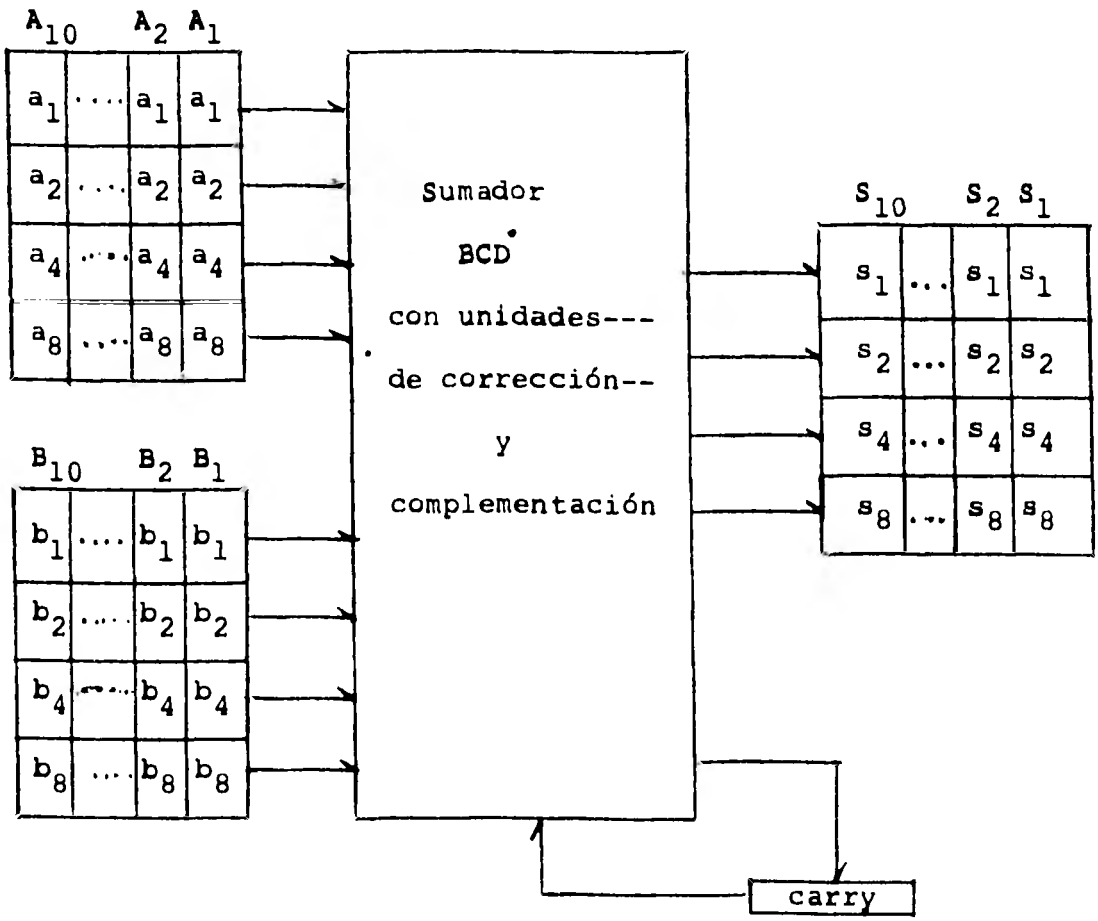


Figura VI.8 Suma de dígitos en serie y de bits en paralelo.

Ahora pasemos a analizar la unidad de control. Aquí tendremos una unidad de control micro-programada, es decir esta será una memoria ROM que contendrá las micro-instrucciones a realizarse, así como los registros necesarios para el funcionamiento de esta.

El registro OPR será idéntico al registro de bandera es decir contará con 3 bits de longitud y sera usado para guardar el código de la operación a realizar.

En la unidad de control se contará con un multiplexor que será usado para pasar información de un registro a otro, así como también para seleccionar los registros adecuados que se vayan a usar. Un circuito multiplexor capaz de seleccionar hasta ocho registros será aceptable.

El circuito del codificador/decodificador será usado para decodificar la información de entrada y para codificar la información de salida.

Finalmente tendrémos algunos circuitos de control que serán usados según las necesidades de la unidad micro-programada de control.

Dentro de la unidad de entrada y salida tenemos el teclado. Este deberá estar compuesto por los números 0 a 9 así como todas las teclas de las operaciones y un switch para indicarnos cuando se quiera trabajar con números de base sesenta, figura VI.9.

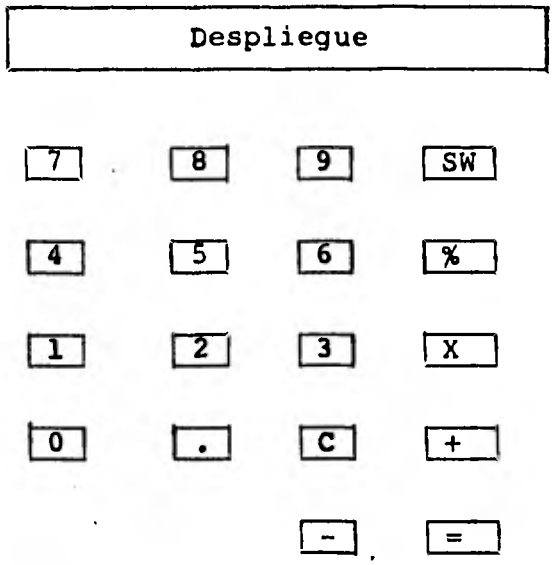


Figura VI.9 configuración del teclado.

El despliegue deberá ser capáz de desplegar hasta diez- caracteres y el punto decimal.

El registro de despliegue DR guardará los códigos de los caráctes que se estan desplegando o que se van a mostrar. Su estructura es idéntica a los registros MR y BR.

El registro KBE valdrá uno para inhibir la información - de entrada hasta que se haya procesado la última informa ción de entrada. Por lo anterior estará compuesto por - un flip-flop tipo D tal como DVF y AVF.

La última unidad es la de alimentación y se deberá selec cionar de tal manera que tenga la capacidad necesaria y- que provea los niveles de voltaje adecuado. Físicamente esta puede constar de un banco de pilas o de una fuente- de poder similar a la mostrada en el capítulo III.

C A P I T U L O VII

C O N C L U S I O N E S

B I B L I O G R A F I A.

C O N C L U S I O N E S

Se puede decir de los resultados obtenidos, que el programa se pudo haber hecho más corto y que tardara menos en ejecutarse, pero sin embargo, al hacer esto se hubiera perdido el sentido de esta tésis, que es el de simular los circuitos necesarios, aunque esto involucre algunas instrucciones y rutinas adicionales. Al haber -- empleado este último enfoque se obtuvo una ventaja desde el punto de vista didáctico y que fué el de comprender como trabaja cada circuito y el sistema completo en sí. Una de las formas en que se podría hacer más efi-- ciente y rápido este programa sería el elaborarlo en -- lenguaje de ensamblador. Pudiéndose tomar el programa- en Basic y pasarlo por un interpretador o elaborarlo -- completo, tomando como modelo el programa de simulación.

Por otra parte, se puede decir que el programa permite - una mejor comprensión del hardware a través del uso del software lograndose una visión más general de todo el - sistema.

Por lo anterior se puede pensar en este tipo de simulaciones en dos casos. En el primero, es cuando se desee diseñar un sistema complejo, y se puede utilizar este - enfoque como una etapa inicial en el diseño. El segundo, es el empleo de este tipo de simulaciones para fi-- nes didácticos.

Por otra parte, otro de los objetivos era el simular -- una calculadora que pudiese manejar números en base sesenta, lo cual se logró. El hecho de poder manejar estos números, nos permite la facilidad de manejar operaciones principalmente comerciales. Es decir, se puedepensar en el caso de que una compañía sea necesario contabilizar el número de horas que trabajo tal persona. - Con este programa de simulación, se puede llevar la contabilidad de este tiempo sin tener que hacer conversiones.

Se concluye que el siguiente paso es el de la construcción de una calculadora basada en el programa desarro-- llado y que además permitiese calcular el sueldo correspondiente de una persona, al poder multiplicar su sueldo por el número de horas que hubiese trabajado.

BIBLIOGRAFIA

1.- Computer System Architecture. Morris Mano Prentice-Hill, 1976. New Jersey.

2.- A Heterarchical Multi-Microporcessor LISP Machine - Adolfo Guzmán. IIMAS, 1981 México.

3.- The Computer, en every day machine. Enid Squire. Addison Wesley. 2da. Edición, Canada.

4.- Understanding Digital Electronics. Gene Mc Whorter. Radio Shack, 1978 Texas.

5.- Teoría de Computación y Diseño lógico. Hill Peterson. Editorial limusa 1978 México.

6.- The Apple soft tutorial. Apple Computer INC, 1979.

7.- The Dos Manual. Apple Computer INC, 1980.

8.- The TTL Data Book For Design Enginners, Dallas Texas: Texas Instruments INC. 1973.

9.- Small Computer Hand book, Maynad, Mass: Digital Equipment Corp, 1973.

10.- Introduction to Programming Maynard, Mass Digital Equipment Corp, 1973.