0099

01149

102

FACULTAD DE INGENIERIA
DIVISION DE ESTUDIOS SUPERIORES
SECCION DE ELECTRONICA.

TESIS QUE PRESENTA EL

ING. MANUEL PADILLA NORIEGA.

PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERIA

CREDITOS POR TESIS. 6 (seis).

JURADO:

Dr. José Albarrán Núñez

M. en C. Alejandro Guarda Auras

M. en C. Angel Kuri M.

M. en C. Luis Marcial Hernández

M. en I. César Chávez Zapata.

JEFE DE LA SECCION

M. en C. Pedro Joselevitch

C.U., México, D.F.
Febrero de 1979

INDICE

INTRODUCCION.

Actualmente las microcomputadoras no tienen incluídas unidades aritméticas capaces de realizar multiplicaciones o divisiones mediante una sola instrucción. Una operación de este tipo se efectúa mediante un algoritmo, el cual obtiene la multiplicación mediante sumas repetidas. Utilizando otro algoritmo, se obtiene la división. Estos algoritmos son iterativos y, por lo tanto, requieren de un gran número de ciclos de máquina para efectuar una operación, lo cual hace que estas operaciones se realicen en forma lenta.

En el procesamiento de señales en tiempo real, los métodos analógicos se han mantenido alejados de las técnicas digitales, a pesar de los avances logrados en éstas. El uso de multiplicadores rápidos han permitido acelerar operaciones complejas que anteriormente sólo podían ser realizadas mediante grandes computadoras. Las funciones de autocorrelación y transformada rápida de Fourier usadas en filtros digitales son algunos ejemplos en los cuales es necesario el uso de multiplicadores rápidos. Los campos de Telefonía TV y sonido han sido altamente beneficiados por el uso de estos multiplicadores. Para ilustrar lo anterior mencionaremos algunos ejemplos. Para determinar los límites del ancho de banda en el procesamiento de señales digitales, la primera regla dice que la razón de muestreo debe ser al menos dos veces las componentes de frecuencia más grande de la forma de onda analizada. Por ejemplo, en el procesamiento de señales de voz, la componente de

frecuencia más grande es aproximadamente 4 KHz, por lo tanto se requiere la razón de muestreo es 8 KHz. De lo anterior sabemos que una muestra debe ser tomada al menos cada 125 microsegundos.

En el procesamiento de la señal de voz la velocidad de llegada depende de la operación que va a ser realizada. Para computar la función de autocorrelación para 256 muestras de una señal de voz, cada muestra debe ser multiplicada por otras 256 muestras. Por lo tanto, en un periodo de 125μs se deben realizar 256 multiplicaciones y por lo tanto cada multiplicación debe efectuarse en menos de 500 ns.

El número de bits de resolución en procesamiento de señales es muy importante para el tiempo de conversión análogo digitales y durante las computaciones. Cuando una señal analógica es convertida a una palabra digital, la relación señal-ruido es crítica. Una pobre relación puede provocar que se recorte la forma de onda, y al ser convertida introduce componentes de frecuencias fantasmas. Por ejemplo, si una forma de onda en un punto tiene un nivel de 160 milivolts y cada nivel de cuantización es 10 mV, 4-bits de resolución es suficiente para la conversión a-d. Si la resolución del sistema es extendido a 8 bits, se logra una confiabilidad mayor en el manejo de la señal.

Implementación en tiempo real de análisis y síntesis de señales de voz digitalizadas envuelve un gran número de multipli-

caciones y por lo tanto requiere una computación de alta velocidad. Puesto que el modelo de señal de voz esta variando en el tiempo, las computaciones son usualmente llevadas en bloques con una duración de 20 a 30 μs. Las muestras de la señal de voz son obtenidas a razón de 8 000 a 10000 muestras por segundo, con una longitud típica de 256 muestras en cada bloque.

Mientras un bloque de muestras es tomado, computaciones complicadas tales como auto correlación y datos de coeficientes de filtros digitales deben ser llevados a cabo. Un sistema es típicamente limitado en su capacidad por las velocidades de multiplicación disponibles, ya que a menudo millones de multiplicaciones por segundo deben ser realizadas.

Una multiplicación de 16 x 16 en una microcomputadora con un ciclo de máquina de 2μs y longitud de palabra de 8 bits, utilizando un algoritmo de sumas y corrimientos, se lleva a cabo en un tiempo de 1,000μs aproximadamente.[1]

Este tipo de operaciones pueden ser realizadas por "hardware" disminuyendo considerablemente el tiempo para efectuar las operaciones de multiplicación y división.

Existen diferentes métodos por "hardware" para efectuar estas operaciones. En la selección de alguno de ellos debe considerarse la velocidad y el costo.

Debe notarse que el ciclo de máquina de las microcomputadoras existentes en el mercado tienen un rango entre 2μs a 0,2μs, por lo cual una unidad que efectúe las operaciones en ese intervalo de tiempo es adecuado.

La mayor parte de los algoritmos de multiplicación son diseñados para ejecución en "software", las técnicas de "hardware" son generalmente adaptaciones de estas técnicas de software.

En el desarrollo de este trabajo se hizo la selección de un algoritmo para cada operación (multiplicación y división) tal que:

1) Ambas operaciones puedan usar el mismo "hardware"
2) El tiempo de ejecución de las operaciones sea menor que los tiempos de ejecución de un algoritmo en una microcomputadora con un costo mínimo.

1 TECNICAS DE MULTIPLICACION

1.1 Algoritmo de Booth

Las computadoras ejecutan las multiplicaciones mediante sumas repetidas y el tiempo requerido depende del número de sumas requeridas.

Un bit igual a cero en el multiplicador conduce a sumar un cero al producto parcial. Debido a que un corrimiento es una operación más rápida que una suma, el tiempo puede reducirse, haciendo corrimientos por cada cero o conjunto de ceros hallados en el multiplicador.

Ciertas propiedades del sistema de numeración binario, combinado con la complementación para permitir la sustracción, pueden ser usados para reducir el número de adiciones.

Un algoritmo usado en "software" es el algoritmo de Booth cuyo diagrama de flujo se halla en la Figura 1, al inicio del algoritmo el contenido del acumulador es cero. El algoritmo consiste en sensar dos bits consecutivos del multiplicador a la vez, si la secuencia de bits es 0,1 suma el multiplicando al contenido del acumulador (AC + X) y en seguida realiza un corrimiento del acumulador a la derecha, si la secuencia es 1,0 resta el multiplicando al contenido del multiplicador (AC -X) y en seguida realiza un corrimiento a la derecha, si la secuencia de bits son iguales únicamente se realiza un corrimiento del acumulador a la derecha.

La primera secuencia de bits que se analiza consta del bit menos significativo y un cero que se agrega para hacer la comparación.

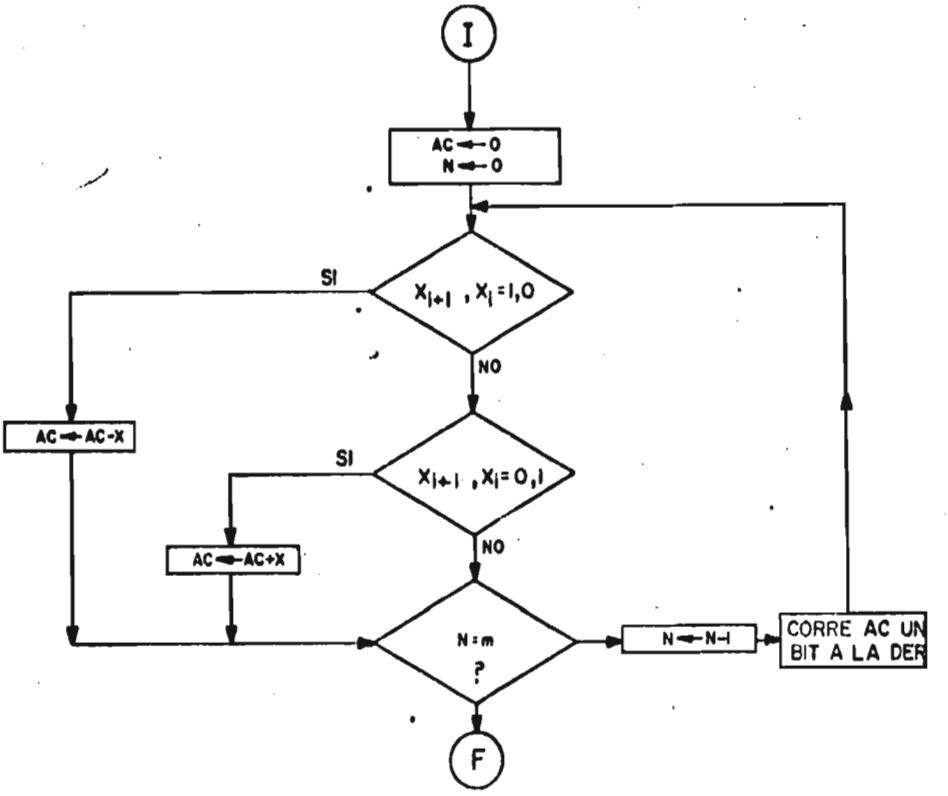


FIG.1 ALGORITMO BOOTH.

Este algoritmo es rápido cuando no hay cambios en los bits del multiplicador y efectúa únicamente corrimientos. Sin embargo, para bits alternados el algoritmo se hace lento. En la figura 2 se da un ejemplo utilizando este algoritmo.

X MULTIPLICANDO		Y MULTIPLICADOR	SECUENCIA DE BITS
1 0 0 1		0 1 1 0 : 0	
	0 0 0 0 .	A C = 0	
1er Paso	0 0 0 0 . 0	CORRIMIENTO	0,0
2° Paso	0 1 1 1 . 0	RESTA (AC-X)	1,0
	1 0 1 1 . 1 0	CORRIMIENTO	
3er Paso	1 1 0 1 . 1 1 0	CORRIMIENTO	1,1
4° Paso	0 1 1 0 . 1 1 0	SUMA (AC+X)	0,1
	0 0 1 1 . 0 1 1 0	CORRIMIENTO	
RESULTADO		1 1 0 1 1 0 = 54	

Fig. 2. Multiplicación usando el Algoritmo de Booth

El método descrito requiere un corrimiento variable y esto no permite predecir el número exacto de ciclos requerido para ejecutar una multiplicación.

1.2 MULTIPLICACION CON SUMAS Y CORRIMIENTOS

El principio básico usado en la multiplicación con sumas y corrimientos es la acumulación de productos parciales por adiciones repetidas del multiplicando bajo el control de los bits del multiplicador.

Se hace el análisis de cada uno de los bits del multiplicador, los cuales indican si se va a realizar una suma y, a continuación, se lleva a cabo un corrimiento.

La implementación de un multiplicador de sumas y corrimientos en hardware puede realizarse con algunos registros de corrimiento y sumadores (Fig. 3). Su funcionamiento es el siguiente.

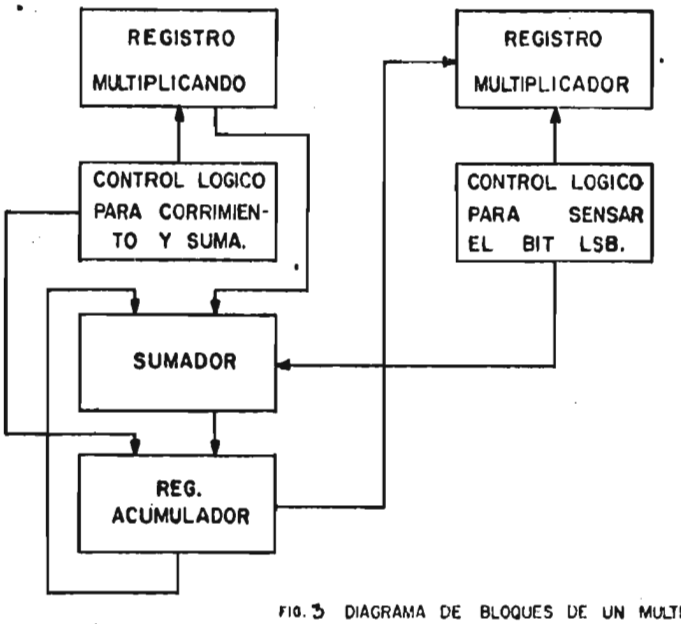


FIG. 3 DIAGRAMA DE BLOQUES DE UN MULTIPLICADOR

El control lógico sensa el bit menos significativo y determina si el multiplicando debe ser sumado con el registro acumulador, el cual almacena los productos parciales. El control lógico para sumas y corrimientos debe enviar una señal para efectuar la suma del registro acumulador y del registro multiplicando y también debe efectuar un corrimiento del acumulador a la derecha.

Para una multiplicación de 16 x 16 necesitamos 16 sumas y 16 corrimientos; si consideramos que la suma y el corrimiento pueden efectuarse dentro de un mismo intervalo de tiempo, se requieren 16 ciclos para completar una multiplicación y si consideramos cada ciclo de 100 ns, se requiere 1.6µs para una multiplicación y en el "software" se requiere 1000µs [1].

1.3 MULTIPLICADORES USANDO TABLAS ALMACENADAS EN ROM

La técnica de multiplicación utilizando ROMs consiste del almacenamiento de datos en tablas, los cuales representan los productos parciales en cada una de las localidades de la memoria.

Un producto parcial puede ser obtenido direccionando la localidad de memoria correspondiente.

Para una multiplicación de 4 bits x 4 bits, existen 256 resultados posibles, entonces la tabla debe contener 256 palabras de 8 bits cada una. Esta tabla puede almacenarse en un ROM de 256 x 8.

Para ejemplificar esta técnica, veremos el diseño de un multiplicador de 16 x 4 bits, (Fig. 4). Los bits del multiplicando y del multiplicador se dividen en grupos de 4 bits. El proceso requiere del almacenamiento de los productos parciales obtenidos de cada uno de los ROM. Los cuatro bits menos significativos del multiplicador son multiplicados por los cuatro bits menos significativos del multiplicando obteniendo el primer producto parcial. El siguiente producto parcial se obtiene multiplicando los siguientes cuatro bits del multiplicador.

Este producto parcial debe quedar corrido cuatro bits con respecto al primero para poder realizar la suma. De manera análoga, se obtienen los productos parciales restantes.

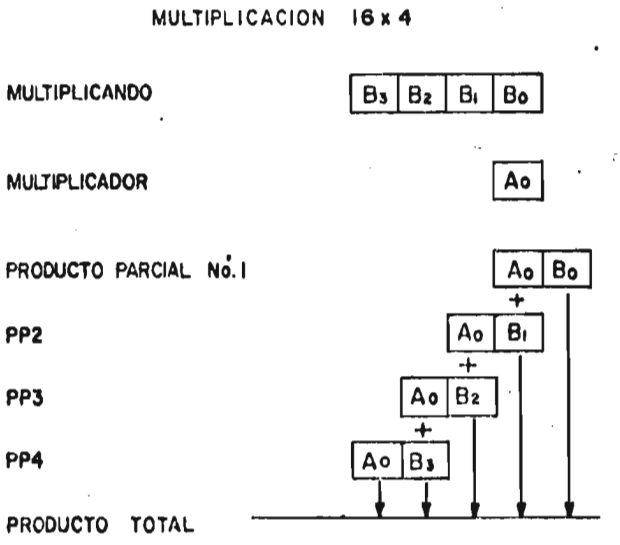


Fig. 4. Acumulación de Productos Parciales.

Una vez obtenidos los productos parciales, estos son sumados de acuerdo a la Fig. 5. El tiempo que requiere cada operación depende del tiempo de acceso a los ROM y rapidez de los sumadores.

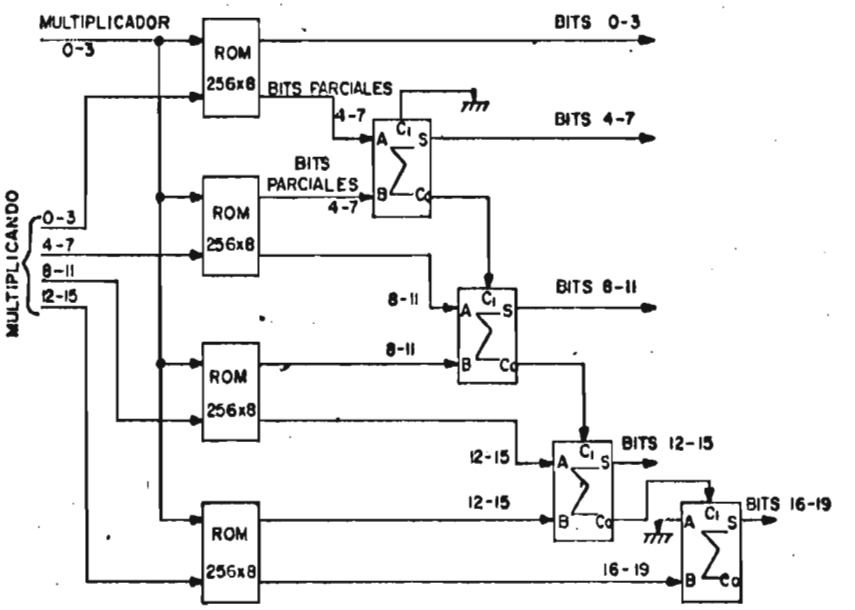


Fig. 5 Multiplicador usando tablas de ROM

1.4 MULTIPLICACON USANDO EL ARBOL DE WALLACE.

Esta técnica se implementa mediante multiplicadores binarios de 4 x 4 bits, los cuales se encuentran integrados en una sola pastilla y son usados en combinación con sumadores denominados árboles de Wallace. Por medio de esta técnica se pueden obtener multiplicaciones en tiempos del orden de nano-

segundos,

Los circuitos mencionados son:

74S274 Multiplicador de 4 x 4 bits.

74S275 Sumador "Arbol de Wallace" con 7 entradas.

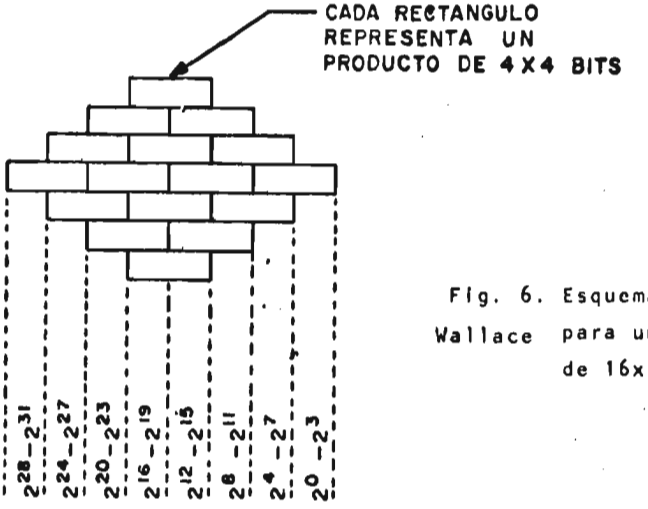


Fig. 6. Esquema del árbol de Wallace para una Multiplicación de 16x16

Los S274 son multiplicadores de 4 x 4 bits implementados mediante lógica combinacional, los cuales están diseñados para reducir el retardo para realizar multiplicaciones binarias en paralelo. El S275 es un bloque sumador que efectúa la suma de hasta siete bits de una sola columna y puede aceptar hasta dos acarreos de la columna anterior.

En la Figura 6 se muestra la configuración de un árbol de Wallace para un multiplicador de 16 x 16 bits.

En general la multiplicación en paralelo requiere de

la obtención simultánea de todos los productos parciales y la suma de los mismos también en forma simultánea para obtener el resultado final.

El "Hardware" para las técnicas de multiplicación en paralelo son bastante costosas, ya que incluye un gran número de compuertas combinacionales.

Por ejemplo, usando los bloques TI 74S274 y 74S275, mencionados anteriormente, una multiplicación en paralelo de 16 x 16 que se realiza en 120 ns se puede hacer con 47 circuitos integrados. Similarmente, el bloque multiplicador AMD 25S05 de 4 x 2 bits realiza una multiplicación de 16 x 16 en 150 ns y requiere de 32 circuitos integrados. Existen en el mercado un multiplicador TRW de 16 x 16 bits en un solo circuito integrado cuyo tiempo de operación es de 300 ns. |1|

1.5 CONCLUSIONES

[1] TABLA 1

TECNICAS DE MULTIPLICACION PARA 16 x 16 BITS.

METODO EN HARDWARE	COSTO	TIEMPO
Suma y corrimiento	$ 15.00	3,2 µs
Tabla almacenada en ROM	120.00	0,8 µs
Arreglo iterativo (AMD)	320,00	150 ns
Arbol de Wallace	550.00	120 ns
Un solo chip (TRW)	250.00	300 ns

METODO EN SOFTWARE	N° DE BYTES DE ALMACENAMIENTO	TIEMPO
Algoritmo de Booth		
En un Micro-pace	90	1,100 µs
Algoritmo Booth		
En un Micro-6800	110	600 µs
Suma y corrimiento en un micropace	20	1,000 µs
Micro programado en un Imp-16.	4	150 µs

Habiendo analizado las distintas técnicas de multiplicación en hardware podemos concluir que la técnica mas adecuada para nuestro diseño es la de sumas y corrimientos ya que el tiempo que requiere para realizar una operación es similar al ciclo de máquina de la microcomputadora y además su costo es mucho menor que el costo de las técnicas de multiplicación en paralelo.

|1| Ver bibliografía.

2. TECNICAS DE DIVISION

2.1 DIVISION MEDIANTE SUMAS Y CORRIMIENTOS

La técnica básica de la división consiste en comparar sucesivamente el divisor con el dividendo o residuo parcial, en cada comparación si el divisor es mayor, el bit resultante del cociente es 0, si el divisor es menor que el residuo parcial el bit del cociente es 1 y restamos el divisor del residuo parcial, en cada caso se hace un corrimiento a la derecha del divisor.

La comparación entre las dos cantidades en una microcomputadora se realiza restando una de la otra y anotando el signo del resultado, entonces debemos realizar una sustracción en cada ciclo.

La técnica de división mediante restas y corrimientos lo analizaremos a través de un ejemplo y veremos que este método se puede dividir en dos:

1. Método de restauración.
2. Método de No restauración.

La sustracción entre el residuo parcial y el divisor puede resultar una cantidad positiva o negativa. En el método de restauración, la cantidad negativa significa que el divisor sustraído se restaura al residuo parcial y a continuación se realiza la siguiente sustracción con el divisor corrido una posición a la derecha. En el segundo método el divisor no se restaura pero en el siguiente paso el divisor que ya se ha corrido un bit a la derecha se suma al residuo parcial que se obtuvo en el paso anterior, el bit del cociente deberá actualizarse debido al cambio de signo,

a continuación se muestra un ejemplo para cada caso.

Ejemplo: División

divisor (D) = 0.1011 dividendo (N) = 0.1111+001

a) Método de Restauración

	Residuo R	Cociente Q
Dividento	+ .1111001	
-D	- .1011	1.000
Primer Residuo	+ .0100001	1.100
-D/2	- .01011	
	-0.0001011	
+D/2	+0.01011	1.000
Segundo Residuo	+ .0100001	
-D/4	- .001011	1.010
Tercer Residuo	+ .0001011	
-D/8	- .0001011	1.011
	0	

b) Método de No Restauración

Dividendo	+ .1111001	
-D	- .1011	1.000
Primer Residuo	+ .0100001	
-D/2	- .01011	1.100
Segundo Residuo	- .0001011	
-D/4	+ .001011	1.010
Tercer Residuo	+ .0001011	
-D/8	- .0001011	1.011
	0	

La ejecución en "hardware" de este método requiere de al menos una sustracción y un corrimiento en cada bit resultante del cociente, requiriendo un tiempo mayor que el tiempo utilizado para realizar una multiplicación; el "hardware" para su realización es más costoso que el de la multiplicación.

2.2 DIVISION USANDO UN ALGORITMO DE CONVERGENCIA CUADRATICA. 4,5,6

El método está basado en un algoritmo de convergencia cua-

drática, por lo tanto el número de iteraciones es disminuído considerablemente.

El divisor y el dividendo son considerados en notación punto flotante normalizado, esto significa que el punto binario está a la izquierda de la posición más significativa y esta posición deberá contener un 1.

En cada iteración un factor R_k, multiplica al numerador y denominador de tal forma que el denominador resultante converge cuadráticamente a 1 y el numerador resultante converge cuadráticamente al cociente deseado.

$$\frac{N}{D} \times \frac{R}{R} \times \frac{R_1}{R_1} \times \frac{R_2}{R_2} \times \ldots\ldots \frac{R_n}{R_n}$$

donde

$N \cdot R \cdot R_1 \ldots R_n \rightarrow$ Cociente

N = Numerador = dividendo

D = denominador = divisor y

$D \quad R \quad R_1 \quad R_2 \ldots R_n \rightarrow 1$.

La selección del factor R_k es la parte esencial del procedimiento, este factor es una aproximación del inverso del divisor.

El divisor puede ser expresado como

$$D = 1-x$$

donde $x \leq \frac{1}{2}$ puesto que D es una cantidad fraccionaria en punto flotante normalizado.

Ahora si el factor R es puesto igual a 1+x y el denominador es multiplicado por R.

$$D_1 = DR = (1+x)\ (1-x) = 1-x^2$$

donde $X^2 \leq \frac{1}{4}$, puesto que $x \leq \frac{1}{2}$

El nuevo denominador tiene la forma 0.11xxxx

De igual forma haciendo $R_1 = 1+x^2$

$$D_2 = R_1\ D_1 = (1-x^2)\ (1+x^2) = (1-x^4)$$

$$D_2 = 0.1111\ xxxx$$

donde $x^4 \leq \frac{1}{10}$ puesto $x \leq \frac{1}{2}$

Si continuamos este proceso el resultado será:

$$D_k = 0.111111\ \ldots\ldots\ 111 \rightarrow 1$$

Ahora es importante notar que el multiplicador para cada iteración es el complemento a dos del denominador.

$$R_{k+1} = 2-D_k = 2-(1-x_n) = 1+x_n$$

Entonces el multiplicador para la iteración K es formado tomando el complemento a dos del resultado de la iteración (k-1) y

$$Q = N\ R\ R_1\ R_2\ \ldots\ R_n \rightarrow \text{Resultado}$$

Una ventaja de este método es que utiliza el "hardware" desarrollado en la unidad de multiplicación y sólo debe agregarse un ROM en donde se almacena la tabla de inversos.

3. DISEÑO DE LA UNIDAD ARITMETICA

3.1 CARACTERISTICAS DE LA UNIDAD ARITMETICA.

La unidad aritmética que implementaremos usa la notación punto flotante y acepta operandos con una longitud de palabra de 32 bits. Cada uno de los operandos contiene 24 bits de mantisa, 6 bits de exponentes, 1 bit de signo para la mantisa y 1 bit de signo para el exponente. El punto binario en el producto resultante está considerado a la izquierda de la posición mas significativa de la mantisa.

La unidad aritmética está interconectada a un sistema SDK-80, donde se realizan todo tipo de pruebas. Existe la posibilidad de ser adaptada a otros sistemas.

El diseño de la unidad aritmética tiene una parte "hardware" que es común a las operaciones de multiplicación y división por lo tanto en este capítulo se describe el diseño de la unidad de multiplicación y posteriormente describiremos el algoritmo de división, el cual utiliza el hardware diseñado.

3.2 UNIDAD DE MULTIPLICACION.

El método de multiplicación que se emplea en esta unidad es el de sumas y corrimientos. La razón por la cual se eligió este método es debido a su bajo costo en relación con otros métodos, y además, el tiempo en que realiza una multiplicación es comparable con el ciclo de máquina de la microcomputadora que será utilizada.

La unidad de multiplicación está integrada por los siguientes módulos:

1) Módulo de Control.
2) Módulo Multiplicación de Mantisas
3) Módulo suma de Exponentes.
4) Módulo Interfase con la Microcomputadora.

3.2.1 DESCRIPCION DE LA UNIDAD DE MULTIPLICACION.

La Figura N° 7 muestra el diagrama de bloques de la unidad de multiplicación cuyo funcionamiento es el siguiente:

La microcomputadora envía en primer lugar, a través de sus puertos de salida, los exponentes hacia el módulo suma de exponentes, esta operación se ejecuta y se almacena en espera de los resultados del módulo multiplicación de mantisas, esto permite actualizar el exponente resultante.

Después del envío de los exponentes, la micro envía a través de sus puertos de salida, el multiplicador y el multiplicando hacia el módulo multiplicación de mantisas, una vez que los operandos se hallan presentes la micro envía un pulso

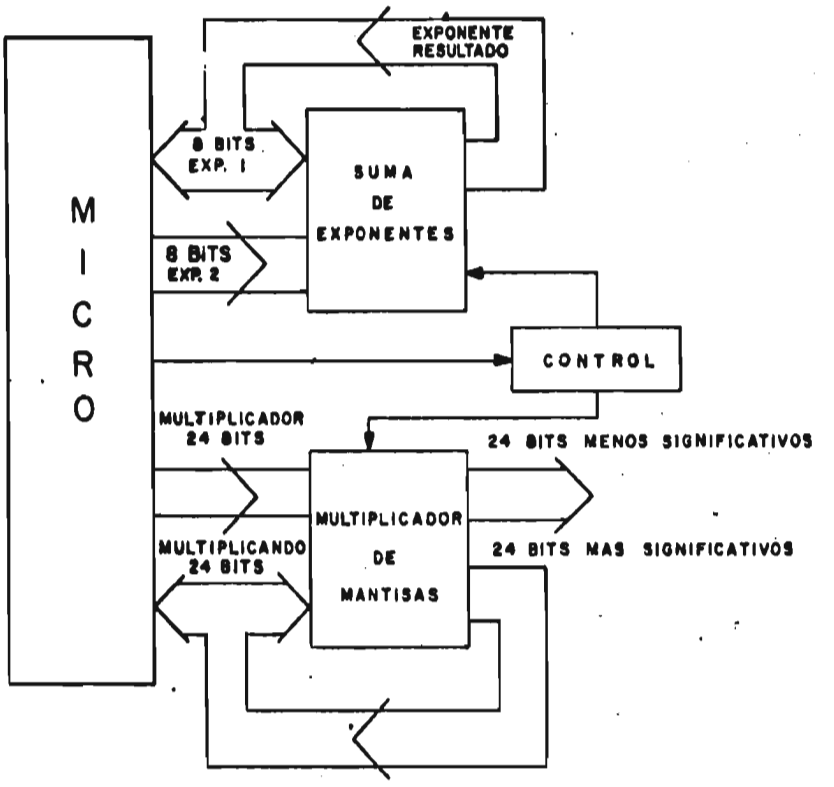


FIGURA 7 DIAGRAMA DE BLOQUES DEL MULTIPLICADOR

a la unidad de control, el cual inicia el proceso de multiplicación. Una vez concluída la multiplicación de mantisas la unidad de control realiza un corrimiento de los bits de tal forma que el primer bit significativo quede a continuación del punto binario, el número de corrimientos permite actualizar el exponente final, al término de esto los resultados son leídos por la micro.

3.2.2 DISEÑO LOGICO DEL MODULO DE CONTROL.

El módulo de control aparece en la Figura 9 y su diagrama de tiempos se encuentra en la Figura 10 El módulo de control, funciona con un reloj de 9MHz tomado del circuito generador de sincronía 8224 del SDK-80, y el pulso de inicio es generado mediante la conjunción de las señales I/OW, A0 y A1 provenientes de la micro.

Las señales generadas por el control son las siguientes:

SEÑAL	DEFINICION
INICIO 1	habilita la carga del contador
FIN MULTIPLICACION MANTISAS	indica el fin de la multiplicación de mantisas.
CUENTA MAX 1	deshabilita el contador
RELOJ 0	permite al módulo multiplicación de mantisas efectuar las sumas y corrimientos.
CORRE IZQ.	permite al módulo Mod. multiplicación de mantisas efectuar los corrimientos a la izquierda.
$\overline{Q46}$ $\overline{Q47}$	determina el fin del corrimiento a la izquierda en la división.
$\overline{Q47}$	determina el fin del corrimiento a la izquierda en la multiplicación.
RELOJ 1	permite al módulo multiplicación de mantisas efectuar los corrimientos a la izqquierda.
RELOJ 1 EXP.	permite al módulo suma de exponentes actualizar el exponente resultante.
PRODUCTO =0	Bandera que indica producto =0

SET	señal que indica el proceso de división
RESET 1	señal que indica el proceso de multiplicación
SIGNO E	señal que nos indica el signo del exponente para la división (signo contrario al de la multiplicación)
RESET 2	señal que nos deshabilita el signo de la división.

3.2.2.1 DISEÑO DE SUBSISTEMAS

CORRE IZQ. 1.- Para obtener esta señal debemos tener en cuenta las señales; fin multiplicación de mantisas, Q46, Q47 y SET. Esta señal permite el inicio y la terminación de los corrimientos a la izquierda, por lo tanto en la multiplicación el corrimiento se termina cuando el bit 47 es igual a uno y la señal SET=0, mientras la combinación del bit Q46 .Q47=1 determina el fin de la división, la razón por la cual Q46.Q47=1 determina el fin del corrimiento a la izquierda, se explicará en el "software" de la división. En ambos casos la señal fin de multiplicación de mantisas debe ser un "1" lógico.

Tabla de verdad

	DIV	Q46	Q47	CORRE IZQ-1
	0	0	0	0
	0	0	1	1
MULT	0	1	0	0
	0	1	1	1
	1	0	0	0
DIV.	1	0	1	0
	1	1	0	0
	1	1	1	1

DIV \ $\overline{Q46}\,\overline{Q47}$	00	01	11	10
0	0	1	1	0
1	0	0	1	0

Diagrama de Veitch Karnaugh

$$CORRE\ IZQ\ I = \overline{Q46}.\overline{Q47} + \overline{DIV}.\overline{Q47}$$

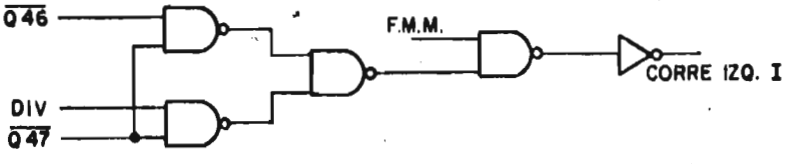


Fig. 8 Subsistema lógico del Modulo de Control.

3.2.2.2 FUNCIONAMIENTO DEL MODULO DE CONTROL.

Para iniciar una multiplicación se requiere que los operandos X e Y estén presentes a la entrada del módulo de multiplicación de mantisas, posterior a esto la micro envía un nivel alto (INICIO) a la entrada del FF1 para iniciar el proceso de multiplicación.

En la primera subida de la señal RELOJ el FF1 cambia de estado, quedando $\overline{Q}_1$ en nivel alto y habilitando durante 25 periodos de RELOJ la señal RELOJ D, la cual va hacia el módulo de multiplicación de mantisas. Además, $\overline{Q}_1$ permite que la señal INICIO 1 cambie a nivel bajo habilitando la carga del contador 1, el cual en la siguiente subida del reloj almacena un 01000, el contador 1 está formado por un contador 74161 y un FF 7476, los cuales integran un contador de 5 bits. La señal INICIO 1 se dirige además al módulo de multiplicación de mantisas.

Una vez que el contador ha sido cargado, la señal INICIO 1 regresa a su estado alto debido a la salida Q_D del contador 1, con lo cual el contador queda en el modo de cuenta. En la siguiente subida de reloj el contador inicia su cuenta ascendente y al llegar al estado 11111 la señal CUENTA MAXI permite que el FF1 cambie con la siguiente subida de reloj. Este cambio indica el fin de la multiplicación de mantisas y deshabilita la señal RELOJ D, además habilita el proceso de corrimiento a la izquierda. El bit más significativo del producto de mantisas (Q47) determina si se va a realizar un corrimiento para la multiplicación y Q46,Q47 para el caso de la di-

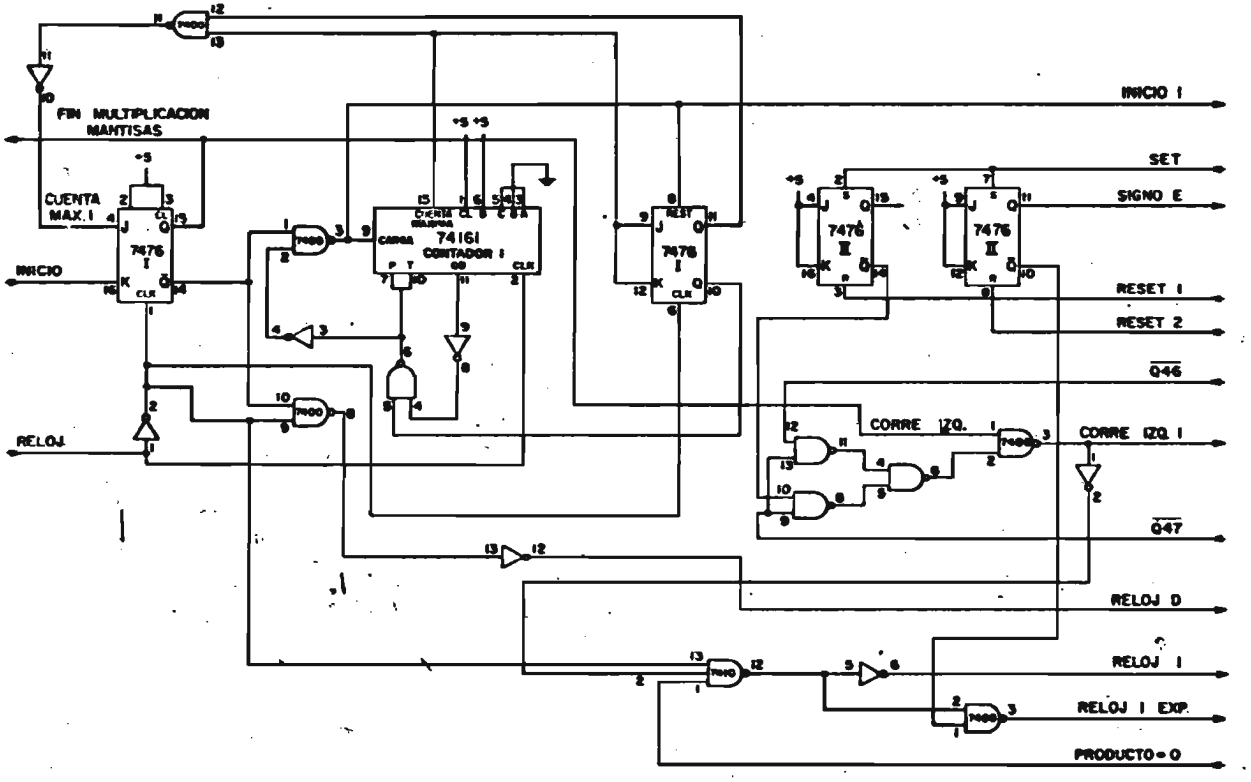


FIG. 9 MODULO DE CONTROL

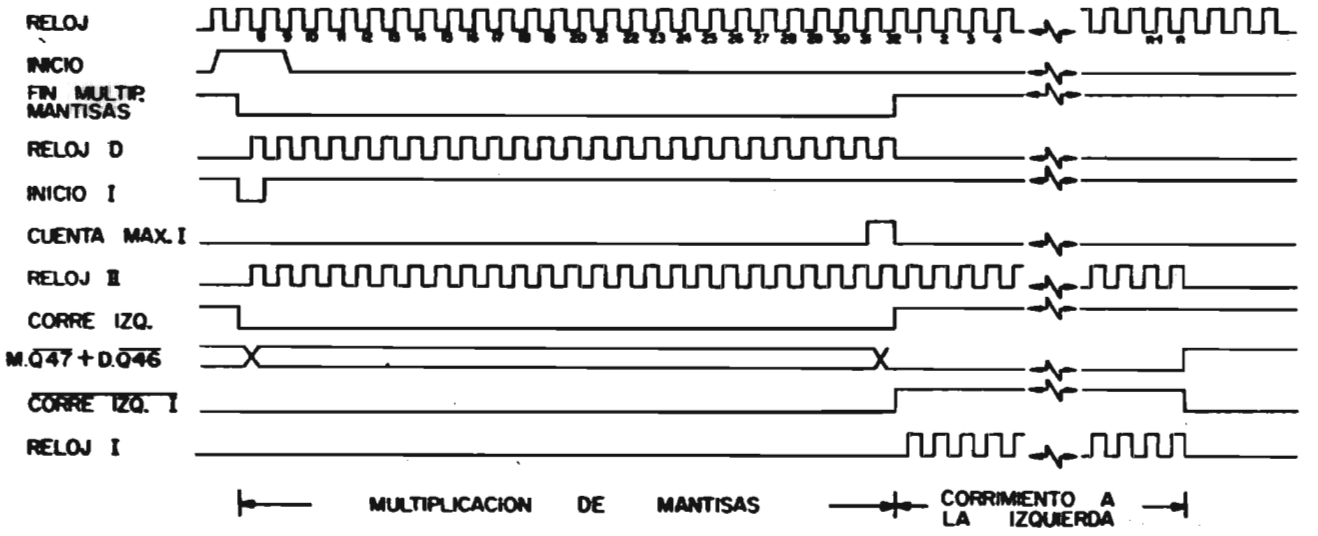


FIG. 10

DIAGRAMA DE TIEMPOS DEL MODULO DE CONTROL.

visión. Mientras Q47 o $\overline{Q46}.\overline{Q47}$ sean "0" lógico las señales RELOJ 1 y corre IZQ 1 que van al módulo de multiplicación de mantisas quedan habilitadas. Cuando el producto de una multiplicación es igual a cero la señal producto =0 deshabilita a la señal RELOJ 1.

3.2.3 DISEÑO LOGICO DEL MODULO MULTIPLICACION DE MANTISAS

La figura N° 11 muestra el multiplicador de mantisas; Este módulo realiza dos funciones:

1) Multiplicación de mantisas.
2) Corrimiento a la izquierda del producto final.

3.2.3.1 DISEÑO DE SUBSISTEMAS.

El FF7 tiene dos funciones que son:

a) En la multiplicación de mantisas el acarreo proveniente del sumador 7483 N° 6 es almacenado en el FF7, para esto es necesario que CORRE IZQ sea igual cero y además se esté efectuando un corrimiento a la izquierda (QH=0)

b) En el corrimiento a la izquierda se debe cargar la información proveniente de Q46 en el FF7, para esto es necesario que CORRE IZQ sea igual a uno y Q46=1

Por lo tanto la tabla de estados es:

	CORRE IZQ	Q46	Q_H	ACARREO	FF7
	0	0	0	0	0
	0	0	0	1	1
MULT.MANT	0	0	1	0	0
	0	0	1	1	0
	0	1	0	0	0
	0	1	0	1	x
	0	1	1	0	0
	0	1	1	1	0
	1	0	0	0	0
	1	0	0	1	0
CORR. IZQ	1	0	1	0	0
	1	0	1	1	0
	1	1	0	0	1
	1	1	0	1	1
	1	1	1	0	1
	1	1	1	1	1

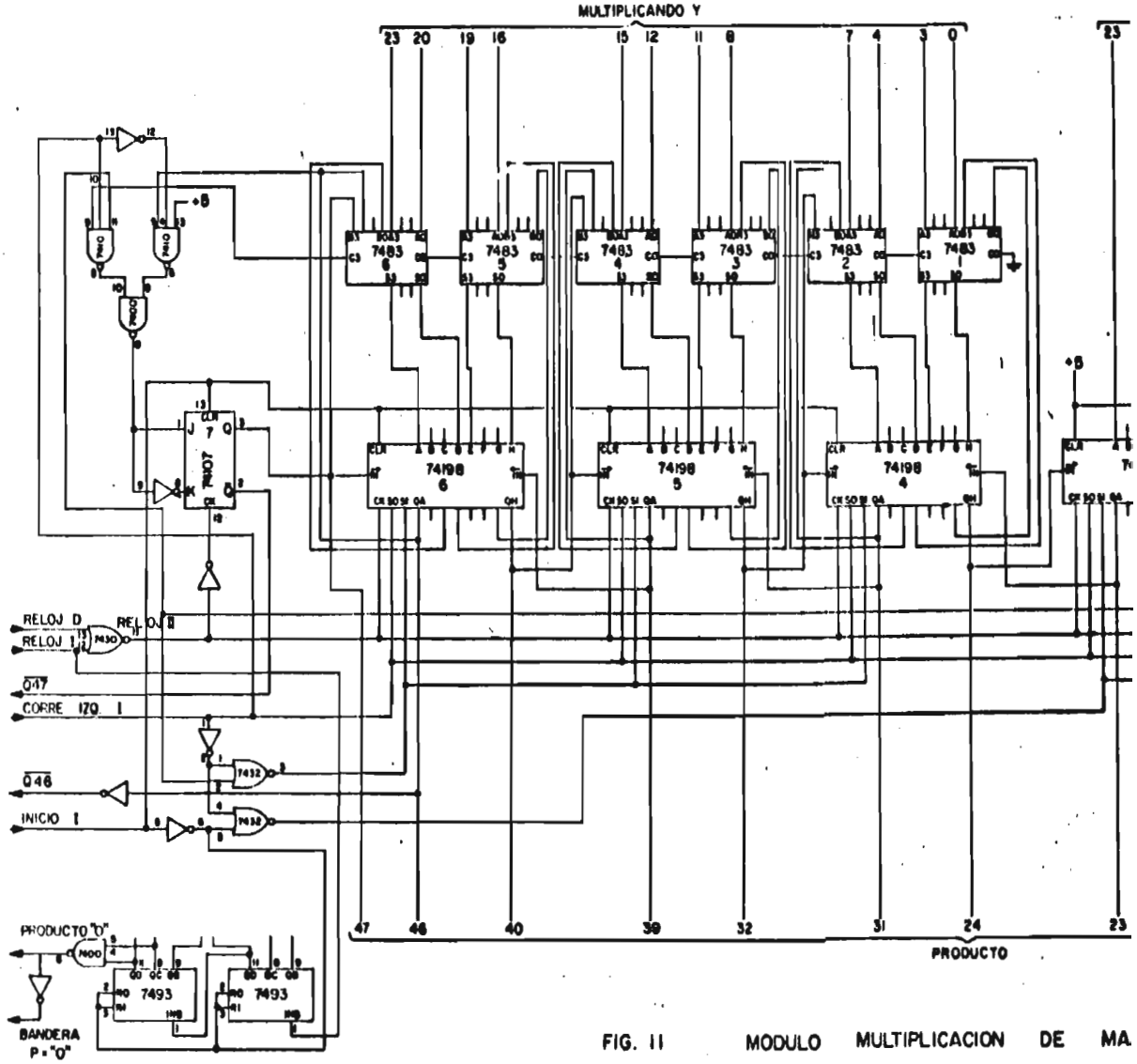


FIG. 11 MODULO MULTIPLICACION DE MA

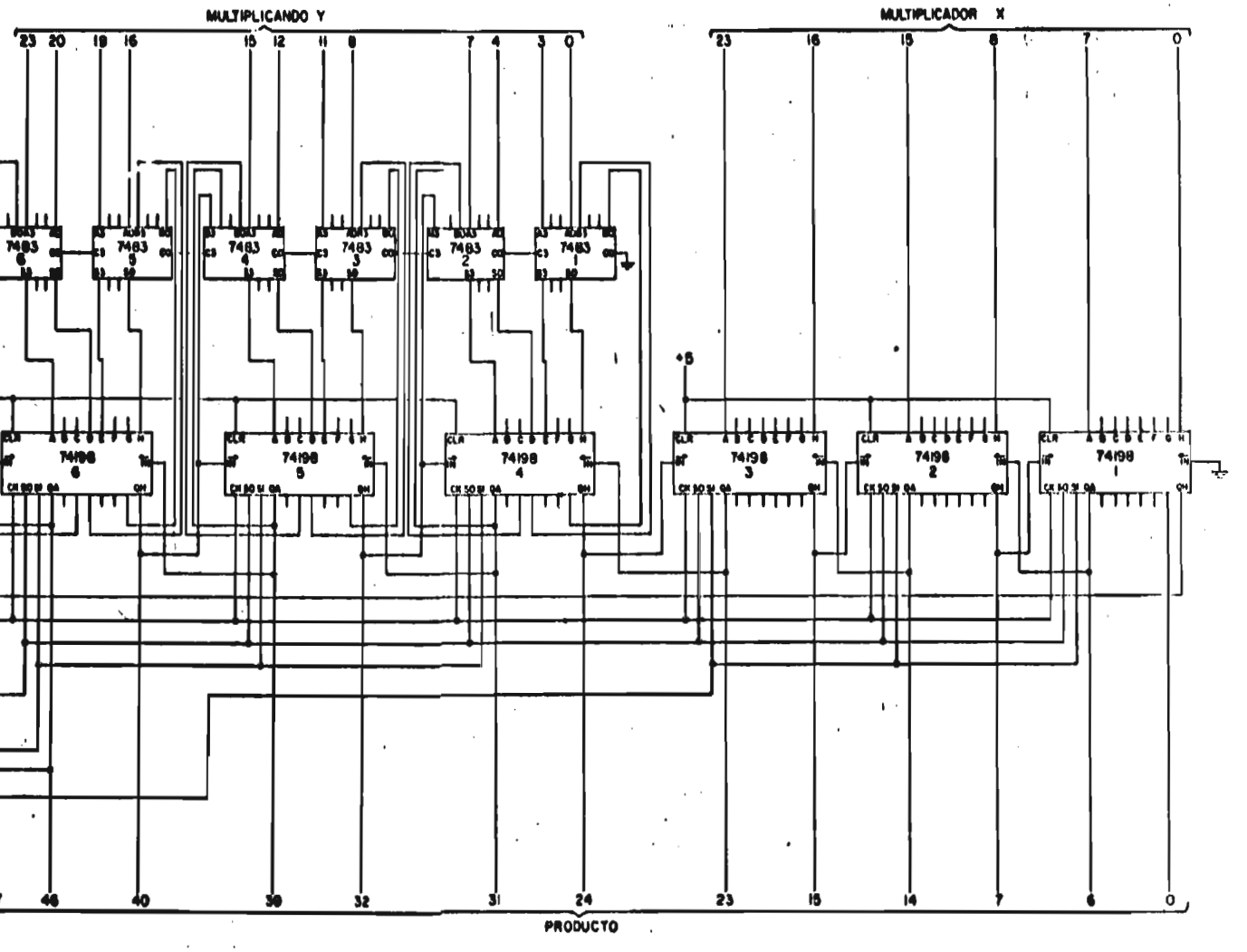


FIG. 11 MODULO MULTIPLICACION DE MANTISAS.

CORR.IZQ. \ QH ACARREO	00	01	11	10
Q46 00	0	1	0	0
01	0	x	0	0
11	1	1	1	1
10	0	0	0	0

FF7=CORR.IZQ.Q46 $\overline{\text{CORR.IZQ}}$ $\overline{\text{QH}}$ ACARREO

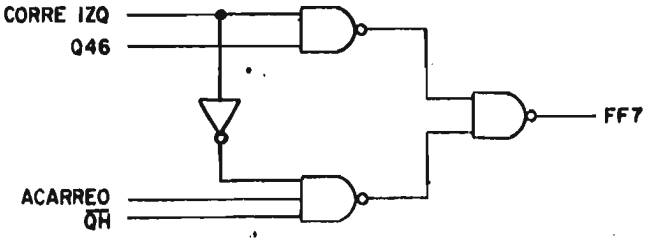


Fig. 12 Subsistema lógico del Modulo Mult. de Mantisas

REGISTROS DE CORRIMIENTOS 74198

Los registros de corrimiento 74198 deben realizar las funciones mencionadas en los incisos 1 y 2, dichos registros son seis, de los cuales los primeros tres (1 2 y 3) se utilizan en el procesamiento de multiplicador y los tres restantes en el procesamiento del multiplicando.

Control de los registros de corrimiento 74198-1 2 y 3

Para el inciso 1, los registros de corrimiento 74198-1, 2 y 3 del multiplicador almacenan en paralelo la información proveniente de la micro, para lograr esto, INICIO I=1, a continuación se realizan los corrimientos a la derecha, para lo cual es necesario que INICIO I=0, en ambos casos CORRE IZQ=0

Para el inciso 2, los registros 1 2 y 3 realizan corrimientos a la izquierda por lo tanto CORRE IZQ=1 e INICIO I=0.

CONTROL DE LOS REGISTROS DE CORRIMIENTO 74198-4 5 y 6

Para el inciso 1 los registros de corrimiento 4 5 y 6 almacenan la información proveniente de los sumadores 7483 y a continuación realizan un corrimiento a la derecha, esto lo determina Q_H, $S_i Q_H=0$ se realiza un corrimiento a la derecha, $S_i Q_H=1$, se realiza una carga en paralelo, en ambos casos CORRE IZQ=0.

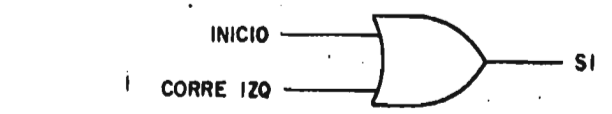
Para el inciso 2 se realizan corrimientos a la izquierda siendo necesario que CORRE IZQ=1.

Tabla de estados para 74198-1, 2 y 3

INICIO 1	CORRE IZQ 1	S_1	S_0
0	0	0	0
0	1	1	0
1	0	1	1
1	1	x	0

CORRE IZQ 1 \ INICIO	0	1
0	0	1
1	1	x

S_1 = INICIO + CORRE IZQ.



CORRE IZQ \ INICIO	0	1
0	1	1
1	0	0

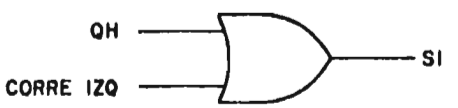
$$S_0 = \overline{\text{CORRE IZQ}}$$

Tabla de estados para el 74198 4, 5 y 6

Q_H	CORRE IZQ	S_1	S_0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

CORRE IZQ \ QH	0	1
0	0	1
1	1	1

$$S_1 = QH + \text{CORRE IZQ}.$$



CORRE IZQ \ QH	0	1
0	1	1
1	0	0

$$S_0 = \overline{\text{CORRE IZQ}}.$$

Fig. 13 Subsistema lógico del Módulo Mult. de Mantisas.

3.2.3.2 FUNCIONAMIENTO DEL MODULO MULTIPLICACION DE MANTISAS

El proceso de multiplicación comienza con un pulso bajo en la señal INICIO 1, la cual habilita la carga del multiplicador en los registros 1, 2 y 3, y al mismo tiempo borra los registros 4,5,6 y 7. RELOJ 11 es la señal de reloj para todos los registros.

A continuación se inicia el algoritmo de sumas y corrimientos, en cada periodo de RELOJ 11 el contenido de los registros 1, 2 y 3 es corrido a la derecha, y se analiza el estado del bit menos significativo QH_1; S_i QH_1 =0, el contenido de los registros 4,5,6 y 7 es corrido a la derecha, si QH_1=1, el multiplicando es sumado al producto parcial anterior; éste se encuentra en los registros 4,5,6 y 7, el nuevo producto parcial es cargado en los mismos registros, teniendo un corrimiento implícito a la derecha debido al alambrado del sumador.

Después de 25 periodos de reloj todos los bits del multiplicador han sido analizados y el proceso de multiplicación de mantisas concluye.

3.2.3.3 FUNCIONAMIENTO DEL CORRIMIENTO A LA IZQUIERDA DEL PRODUCTO FINAL

Este proceso consiste en colocar el primer bit significativo a continuación del punto decimal para lo cual el bit Q47 se envía a la unidad de control, mientras Q47 sea igual a "0" CORRE IZQ 1 y RELOJ 1 están habilitados permitiendo el corrimiento del producto, esto es en el proceso de multiplicación. Ahora bien, para la división mientras Q46.Q47 sea

igual acero CORRE IZQ 1 y RELOJ 1 están habilitados permitiendo el corrimiento del producto. El número de pulsos de RELOJ 1 son contados en el contador 2; esto equivale al número de corrimientos a la izquierda. Dicho contador está formador por dos contadores asíncronos 7493 integrando un contador de 6 bits.

El corrimiento a la izquierda puede concluir de tres formas:

a) Cuando Q47 sea igual a 1 para la multiplicación
b) Cuando el contador 2 llega al estado 11000, esto indica que el producto es igual a cero.
c) Cuando $\overline{Q46} \cdot \overline{Q47}=1$ para la división.

3.2.4 DISEÑO LOGICO DEL MODULO SUMA DE EXPONENTES

Los exponentes de cada operando están formados por palabras de 8 bits, los bits 0-5 representan la magnitud del exponente, el bit 6 el signo del exponente y el bit 7 el signo de la mantisa. Estas palabras son enviadas por la micro a través de dos puertos de salida.

Para realizar el diseño de este módulo lo dividimos en cuatro secciones:

1) Suma de exponentes
2) Actualización del resultado de la suma
3) Obtención de signos
4) Banderas de sobreflujo

3.2.4.1 DESCRIPCION DEL SUBSISTEMA SUMA DE EXPONENTES.

Para el diseño de este sistema vamos a considerar únicamente el caso de la multiplicación, ya que para el caso de la división el signo resultante SER=SE1-SE2, por lo cual invertimos el signo de SE2 y realizamos las operaciones de multiplicación y división con el mismo diseño.

Este cambio de signo se hace con la señal SIGNO E proveniente de la unidad de control, S_i SIGNO E=0 se tiene el proceso de multiplicación, si SIGNO E=1 se tiene el proceso de división.

Los bits de los exponentes se almacenan en los circuitos 7475 y 1 y 2, estos controlan la operación realizada por los circuitos 74381 de acuerdo a la siguiente tabla:

SE1	SE2	
0	0	A+B
0	1	A-B
1	0	B-A
1	1	A+B

En donde SE1 representa el signo del exponente 1

SE2 representa el signo del exponente 2

A representa la magnitud del exponente 1

B representa la magnitud del exponente 2

El resultado de estas operaciones se obtiene en complemento a dos y es enviado a los contadores 74191; el bit de acarreo de los sumadores (F6) se almacena en el registro 7475 N° 3 y la denominaremos CYS, Este bit representa el signo de la suma cuando los signos de los exponentes son distintos y se presentan dos casos :

a) CYS=0 el signo de la suma resultante es positivo

b) CYS=1 el signo de la suma resultante es negativo

Cuando los signos de los exponentes son iguales representa sobreflujo y se tienen dos casos

a) CYS=0 no hay sobreflujo

b) CYS=1 si hay sobreflujo

3.2.4.2 DESCRIPCION DEL SUBSISTEMA. ACTUALIZACION DEL RESULTADO DE LA SUMA.

El propósito de esta sección es actualizar el resultado de la suma de exponentes en función del número de corrimientos a la izquierda que se efectúan en el resultado del producto de mantisas. Esta actualización se realiza en el contador programable 74191. Al inicio de esta operación, el contador tiene

el valor de la suma de exponentes, a este valor se le resta algebraicamente el número de corrimientos mencionados anteriormente, excepto en el caso en que los exponente son ambos negativos. En este caso se le agrega el numero de corrimientos, ya que ambos signos son negativos.

La siguiente tabla muestra el modo de funcionamiento del contador 74191

SE1	SE2	MODO
0	0	Cuenta decreciente
0	1	cuenta decreciente
1	0	cuenta decreciente
1	1	cuenta creciente

Una vez terminado el conteo a la salida del circuito 74191, se tiene el resultado de la suma de exponentes, el cual está en complemento a dos; este resultado se envía a las entradas del circuito 74181 el cual entrega el resultado final de la suma de exponentes en magnitud signada.

3.2.4.3 DISEÑO DE SUBSISTEMAS

SIGNO DEL EXPONENTE. - El signo del exponente resultante SER depende de los signos de los exponentes SE1, SE2 y además del acarreo del contador Q_a del circuito 74191 que será denominado CYC.

Para obtener el signo resultante analizemos los siguientes casos.

I) Cuando ambos signos son positivos, el signo resultante depende del valor del acarreo del contador, si éste es 0 significa que el signo es positivo y no hubo un cambio de signo, si es 1 significa que el signo es negativo y se efectúa un cambio de signo debido al decremento en el contador.

II) Cuando los signos son distintos:

a) Si el acarreo del sumador es cero, el signo resultante depende del acarreo del contador.

b) Si el acarreo del sumador es uno, el signo resultante es negativo independientemente del valor del acarreo del contador

III) Cuando ambos signos son negativos el signo del exponente resultante es negativo

Con esto obtenemos la siguiente tabla de estados.

SE2	SE1	CYS	CYC	SER
0	0	0	0	0
0	0	0	1	1
0	0	1	0	x
0	0	1	1	x
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	x
1	1	1	1	x

SE1 SE2 \ CYS CYC	00	01	11	10
00	0	1	x	x
01	0	1	1	1
11	1	1	x	x
10	0	1	1	1

SER=CYC+CYS+SE1*SE2

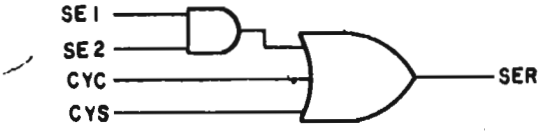


Fig. 14 Subsistema lógico del Módulo Suma de Exponentes.

SIGNO DE LA MANTISA.

Los bits que representan el signo de la mantisa SM1 SM2 son enviados a una compuerta OR exclusiva cuya función es obtener el signo de la mantisa resultante, el cual se almacena en el registro 7475 N° 4.

BANDERAS DE SOBREFLUJO.

Se presentan dos casos de sobreflujo

a) Sobreflujo por sumador

b) Sobreflujo por contador

El primer caso se presenta cuando los signos de los exponentes son iguales y la suma de exponentes sobrepasa el tamaño de la palabra con lo cual el acarreo del sumador es uno.

La tabla de estados es la siguiente

SE1	SE2	CYS	SF
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tabla de Velth Karnaugh

SE1 \ SE2 CYS	00	01	11	10
0	0	1	0	0
1	0	0	1	0

La ecuación que define SF

$$SF = \overline{SE1}\ \overline{SE2}\ CYS + SE1\ SE2, CYS,$$

$$= \overline{SE1\ \ SE2\ \ CYS + SE1\ \ \overline{SE2}\ CYS}$$

$$= SE1\ \ SE2 + SE1\ SE2\ \ CYS$$

$$= SE1\ \ SE2 + SE1\ SE2\ \ CYS$$

$$= SE1\ \ SE2\ \ SE1\ SE2 + CYS$$

$$= (SE1+SE2).\ (SE1+SE2)+CYS$$

$$= SE1\ SE2 + SE2\ SE1\ CYS$$

$$= SE1\ SE2 + SE2\ SE1\ CYS$$

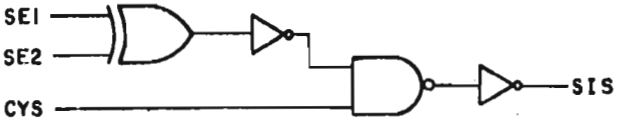


Fig. 15 Subsistema lógico del Modulo Suma de Exponentes.

El segundo caso occure cuando se cumple cualquiera de las siguientes condiciones:

I) Ambos signos son negativos, no existe acarreo en el sumador y el conteo del contador sobrepasa el tamaño de la palabra con lo cual el acarreo del contador es uno debido a que el valor inicial del contador esta en magnitud signada.

II) Los signos de los exponentes son diferentes; el resultado de la suma es negativo por lo cual el bit de acarreo es uno y el conteo del contador sobrepasa el tamaño de la palabra por lo tanto el acarreo del contador es cero debido a que el valor inicial del contador está en complemento a dos.

La tabla de estados es la siguiente:

SE1	SE2	CYS	CYC	SF
0	0	0	0	0
0	0	0	1	1
0	0	1	0	x
0	0	1	1	x
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

neg.

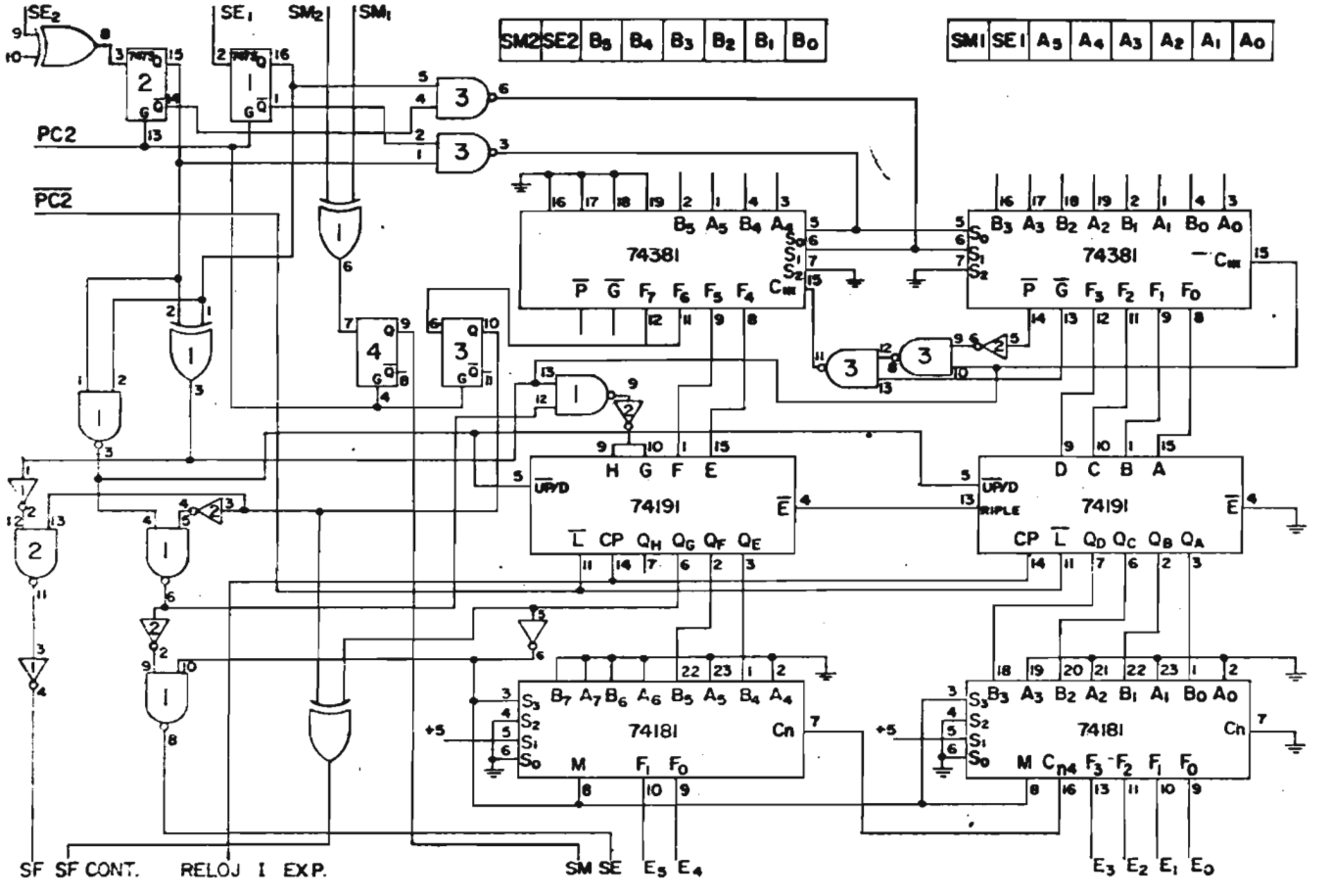


FIG. 16 MODULO SUMA DE EXPONENTES.

SE1 SE2 \ CIS CYC	00	01	11	10
00	0	1	x	x
01	0	1	0	1
11	0	1	x	x
10	0	1	0	1

$$SF\ CONT = \overline{CYS}.CYC + SCYS \cdot \overline{CYC}$$

Fig. 17 Subsistema lógico del Módulo Suma de Exponentes.

3.2.5 MODULO DE INTERFAZ CON LA MICROCOMPUTADORA.

La unidad aritmética se encuentra interconectada con la microcomputadora SDK-80, por lo tanto el diseño de la interfaz lo haremos utilizando las señales de control de dicho sistema. El diagrama de bloques de la interfaz se encuentra en la figura 18.

La transferencia de información de la microcomputadora hacia la unidad aritmética se realiza a través de dos circuitos periféricos programables 8255, que denominaremos periféricos No. 1 y 2; éstos son programados en modo cero, por lo tanto, en cada periférico se tienen tres puertos de ocho bits cada uno, los cuales denominaremos PA, PB y PC.

El regreso de información de la unidad aritmética se realiza a través de circuitos de 3 estados, los cuales se forman en grupos de ocho líneas para su selección, la salida de estos circuitos se conecta al bus de datos "DI"

3.2.5.1 SEÑALES DE CONTROL

Existe un conjunto de señales en el sistema SDK-80 que seleccionan los periféricos No. 1 y 2, estas señales son:

$A_0A_1A_3A_4A_5$ líneas de dirección de la micro
IO/W señal de control para escritura en puertos
IO/R señal de control para lectura en puertos

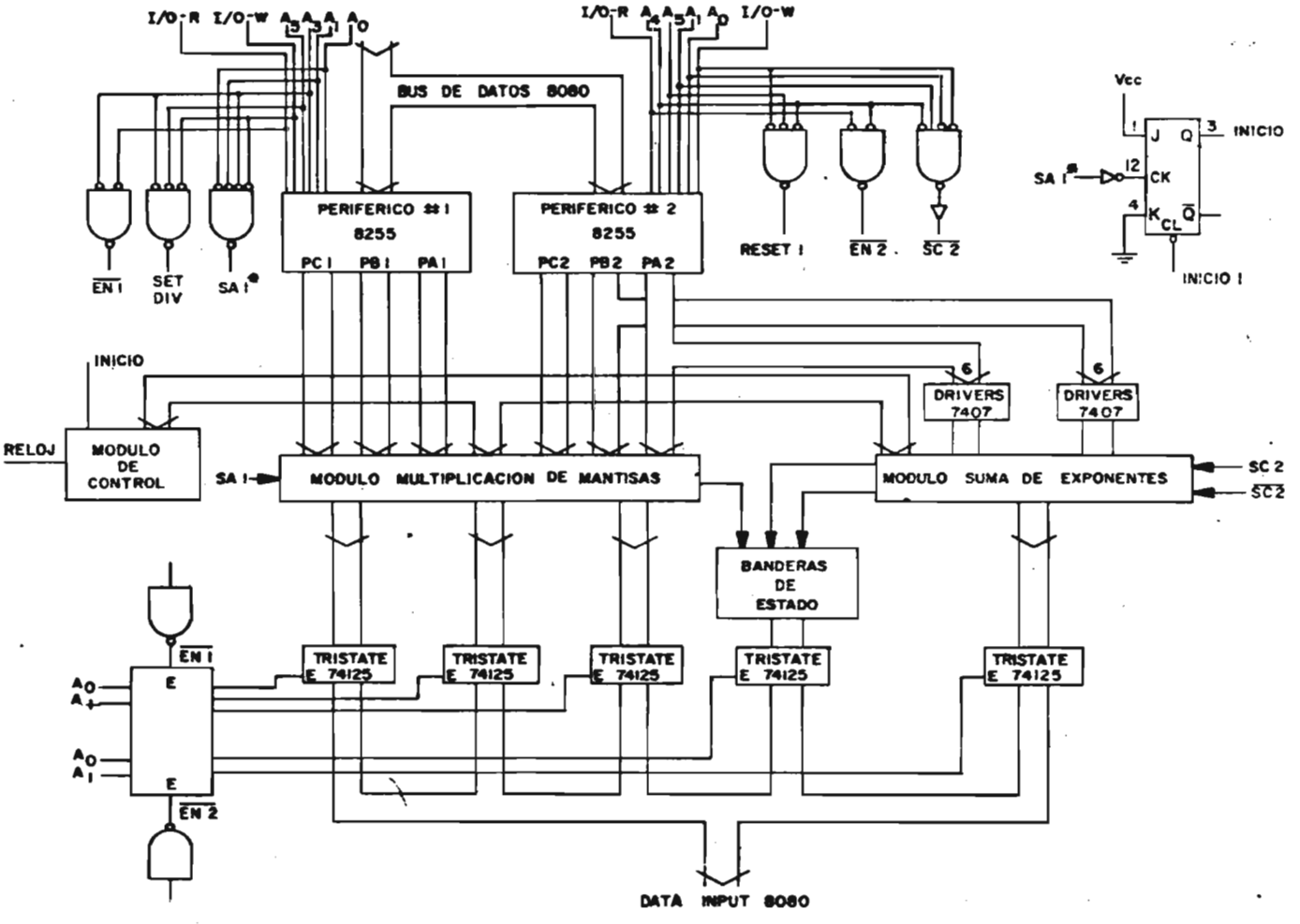


FIG. 18 MODULO INTERFAZ CON LA MICROCOMPUTADORA

3.2.5.2 MODO DE SELECCION DE PERIFERICOS.

A3	indica la selección del periférico No. 1
A4	indica la selección del periférico No. 2
A_1A_0	indica cual de los puertos será seleccionado
00	puerto A
01	puerto B
10	puerto C
11	programación de los puertos
10/W	indica la escritura en el puerto seleccionado
10/R	indica la lectura del puerto seleccionado

A partir de estas señales se generan las señales de control para la transferencia de información entre los periféricos y la unidad de multiplicación.

Es importante notar que las señales de control están en sincronía con la transferencia de información.

A continuación se da una lista de las señales de control de la interfaz.

a) $SA1 = \overline{A_0}\ \overline{A_1}\ \overline{A_3}\ \overline{IOW}$

Esta señal es un control para el módulo multiplicación de matisas, e indica el inicio de una multiplicación.

b) $SC2 = \overline{A_0}\ \overline{A_1}\ \overline{A_4}\ \overline{IO/W}$

Esta señal es un control para el módulo suma de exponentes permitiendo el almacenamiento de los bits de signo, a continuación se inicia la suma de exponentes.

c) EN2 = $\overline{A_4}\ \overline{IO/W}$

Esta señal habilita el selector de líneas que selecciona los circuitos de tercer estado para la lectura de las banderas de estado y suma de exponenentes.

d) EN1 = $\overline{A_3}\ \overline{IO/R}$

Esta señal habilita el selector de líneas que selecciona los circuitos de tercer estado para la lectura del producto de mantisas.

e) SET DIV = $\overline{A_3}\ \overline{A_5}\ \overline{I/OW}$

Esta señal indica al control que se va a efectuar una división.

f) RESET 1 = $\overline{A_4}\ \overline{A_5}\ \overline{I/OW}$

Esta señal indica al control el inicio de la operación de multiplicación.

3.2.5.3 SUBRUTINAS DE COMUNICACION MICRO-UNIDAD ARITMETICA.

Estas subrutinas se encargan de realizar la transferencia de información entre la microcomputadora y la unidad aritmética, además despliega los resultados de la operación realizada

En la Figura 20 se encuentra el diagrama de flujo para la multiplicación, y en la Figura 21 se tiene el diagrama de flujo para la división.

4 ALTERNATIVAS DE DISEÑO

4.1 Modificación de la longitud de la palabra

Se puede modificar la longitud de la palabra utilizada en la unidad aritmética; si se desea una palabra de 16 bits en la mantisa y 8 bits de exponente, se deben realizar algunos cambios en el "Hardware" y el "Software". Sabemos que en el modulo multiplicación de mantisas se tienen 3 registros 74198 para el multiplicador y 3 para el multiplicando, y debido a que ahora se tiene una palabra de 16 bits estos registros se reducen a 2 para el multiplicador y 2 para el multiplicando, así mismo se tienen 6 sumadores 7483 de 4 bits, los cuales se reducen a 4.

El módulo de control requiere un cambio en el alambrado del contador, este contaba hasta 24, ahora se debe alambrar para que cuente hasta 16.

El módulo suma de exponentes permanece igual ya que se desea manejar el mismo número de bits en el exponente, en caso de querer disminuir el número de bits de 6 a 4 bits, unicamente se reduce el número de registros sumadores para lograr el manejo de 4 bits.

El módulo interfaz permanece igual, ya que el circuito 8255 tiene 3 puertos de 8 bits cada uno y necesitamos 2 circuitos 8255 para el manejo de los 32 bits de los operendos, sin

embargo estos dos circuitos estan subutilizados.

En el software, debemos enviar los últimos ocho bits del último operando al mismo puerto que se enviaron en el caso de la multiplicación de 24x24 bits, ya que este puerto genera, en conjunción con otras señales de control la señal de inicio para la multiplicación.

El método descrito anteriormente se usa en forma general si se desea cambiar la longitud de la palabra.

4.2. CAMBIO DE LA LONGITUD DE PALABRA POR SOFTWARE

Si se desean realizar cambios en la longitud de palabra por software unicamente, se debe diseñar un circuito que realize los cambios en el hardware mencionados anteriormente, asf por ejemplo se debe enviar una señal S1 por programa para el conteo hasta 16 ó se envía otra señal S2 para contar hasta 24.

El contador esta formado por el circuito 74161 y un FF 7476, las entradas del contador almacenan un 01000 con el pulso de inicio, ahora se desea que almacene un 10000, para que al llegar a 11111 se indique el fin de la multiplicación de mantisas. Por lo tanto el circuito que realiza el cambio es:

S1	S2	D	FF
0	0	0	0
0	1	1	0
1	0	0	1
1	1	x	x

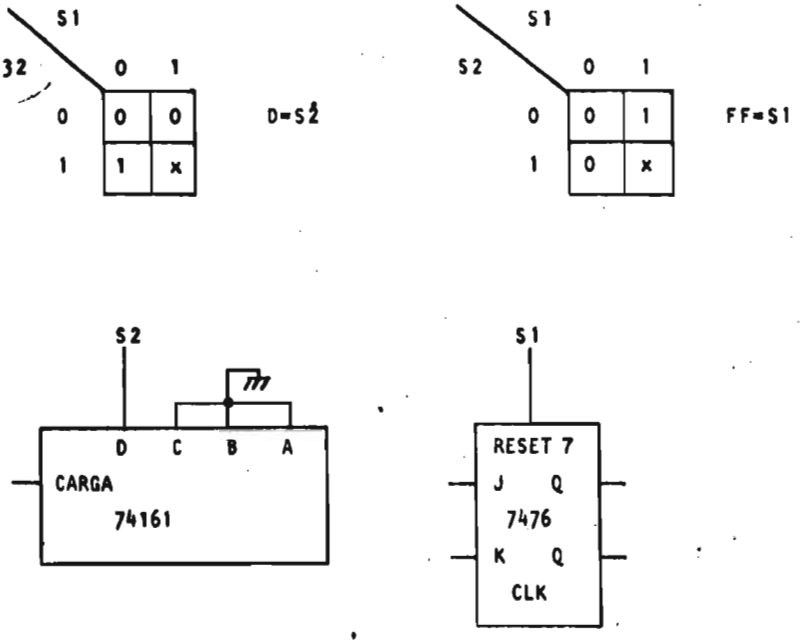


Fig. 19 Modificaciones al Modulo de Control.

Estas señales de control S1, S2 nos pueden servir para hacer el cambio en los registros de corrimiento 74198 y en los registros sumadores, por lo tanto el diseño de la Unidad Aritmética total requiere agregar unos cuantos circuitos integrados para tener la posibilidad de realizar multiplicaciones con diferentes longitudes de palabra.

4.2 REDUCCION DEL TIEMPO

Podemos reducir el tiempo de la multiplicación utilizando un acceso directo a memoria (DMA). Para lograr esto es necesario diseñar un circuito que tome el control del bus de datos, del bus de direcciones y control de interrupciones, esto aumenta el costo del hardware pero disminuye el tiempo de 180μs. a 5μs más el tiempo del acceso a memoria en lectura y escritura, que es del orden de algunos microsegundos. Por lo tanto si es conveniente agregar el circuito DMA para reducir el tiempo de la multiplicación.

4.3. UNIDAD ARITMETICA HIBRIDA

Es posible realizar una multiplicación de 24x24 utilizando el diseño de una Unidad aritmética de 8x8 bits y un programa que se encarge de hacer la multiplicación mediante la técnica de la multiplicación usando tablas de ROM.

Este método tiene la ventaja de disminuir el número de circuitos integrados sin embargo, el costo de la unidad aritmética se reduce aproximadamente a la mitad, ya que el modulo de control, el módulo suma de exponentes y el módulo interfaz sufren ligeros cambios.

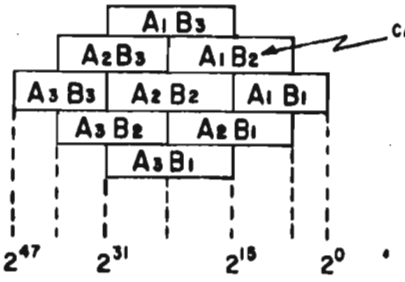


Fig. 20 Multiplicación de 24x24

El resultado de cada multiplicación parcial de 8x8 es de 16 bits (A_1B_1, A_1B_2, A_1B_3 etc.) por lo tanto el almacenamiento en memoria requiere de 2 ciclos de lectura. De la figura observamos que debemos realizar 2 sumas de 8+8 para obtener el segundo producto parcial, 4 sumas para obtener el 3er. producto parcial, 4 sumas para el 4°. P.P., 2 sumas para el 5° P.P. y 2 para el 6° P.P, que en total son 14 sumas de 8+8 bits.

La escritura de un factor de 8 bits requiere de 3 instrucciones (INX, MOV, OUT), que a su vez contienen 22 periodos de reloj y el tiempo consumido es de 22x0.5µs =11µs. Para la lectura de cada uno de los productos parciales de 16 bits se consume el doble del tiempo anterior, es decir, 22µs. El número de escrituras es igual al número de factores que son seis y por lo tanto el tiempo total de escritura es de 11x6=66µs. El número de lecturas son 9 y el tiempo total de la lectura es de 22x 9= 198 . El número de sumas son 14 y cada una consume 3.5µs por lo tanto 14x3.5 = 49µs. Además dentro de la secuencia de las sumas hay instrucciones tales como guardar en memoria los

resultados parciales, incrementar registros, almacenar el acarreo etc, lo cual para el peor de los casos consumiría cuatro veces el tiempo anterior, es decir 200μs. La escritura y lectura de exponentes consume 4x11μs=44μs. Por lo tanto el tiempo total que consume una multiplicación de 32x32 utilizando la Unidad Aritmetica Híbrida es de 49+200+198+66+44=551μs.

Para tener un punto de referencia, calculamos el tiempo consumido en la lectura y escritura de una multiplicación de 32x32 utilizando el programa dado en el apéndice, y se requieren 175μs. Cabe mencionar que, en el "hardware" de la U. A. diseñada se consumen únicamente 5μs utilizando un reloj de 10 MHz. Sin embargo, este tiempo no se considera ya que la microcomputadora no detecta el funcionamiento del "hardware".

De lo anterior podemos concluir que la U. A. híbrida consume aproximadamente 4 veces el tiempo de la U. A. diseñada. Por lo tanto si es conveniente implementar la posibilidad híbrida.

CONCLUSIONES

Las pruebas efectuadas a la Unidad Aritmetica (U. A.) consistieron en realizar multiplicaciones y divisiones con cambios en la magnitud de la mantisa y el exponente asi como también cambios en los signos de la mantisa y de los exponentes, logrando resultados correctos con una aproxima-- ción de 2^{24}=16777216, es decir ocho cifras decimales, lo cual para cualquier cálculo cientfico es bueno.

El tiempo de una multiplicación de 24x24 con un exponente de 6 bits, utilizando un reloj de 10 Mhz es de 180μs, es importante notar que la U. A. ejecuta la multiplicación en 5μs y el conjunto de instrucciones del sistema SDK-80 utilizado en el programa consume 175μs.

Para el caso de la división el tiempo aumenta cuatro veces, es decir una división de 24x24, utilizando el algoritmo de convergencia cuadrática se ejecuta en 750μs, sabemos que el tiempo utilizado en "software" para una multiplicación de 16x16 es de 1,100μs (tabla 1) y para una multiplicación de 24x24 el tiempo aumenta de acuerdo al número de bytes en que se descompone el operando, para este caso se descompone en dos bytes y el tiempo de ejecución es 2^2=4x1,100 s=4,400μs, por lo tanto la U. A. ejecuta la multiplicación 25 veces mas rapido.

El costo de la U.A. considerando únicamente las componentes utilizadas, incluyendo el ROM 2708 en donde se encuentra almacenado el programa y dos periféricos programables 8255, es de 100 dólares*, sabemos que un multiplicador de 16x16 integrado en un solo "chip", cuesta 200 dólares (tabla 1), el tiempo de ejecución es de 250ns, sin embargo este "chip" no contiene módulo suma de exponentes, ni módulo interfaz, por lo tanto el costo de la U.A. diseñada es adecuada por las ventajas que posee.

La U.A. fue integrada en dos tarjetas de circuito impreso de 14x25 cm y el consumo de potencia es de 6W por tarjeta.

* Ver tabla 2, Apéndice

BIBLIOGRAFIA.

1 PARASURAMAN, Bala. "Hardware multiplication Techniques for microprocessor systems". Computer Design, 1977, 16:75-88

2 TEXAS INSTRUMENTS, INC. Suplement to the TTL data book for design engineers. 1974

3 HILL & PETERSON. Digital Systems. Wille, 1973

4 ANDERSON, S. F. "The IBM system/360 model 91: floating-point, execution unit". IBM Journal, 1967:41-53.

5 LING, H. "High speed division for binary computers". Spring Joint Computer Conference. 1971:373-378

6 Mc SORLEY, O.L. "High-speed arithmetic in binary computers". Proceedings of the IRE, 1961, 49.

7 FERRARI, Domenico. "A division method using a parallel multiplier". IEEE Transactions on Electronic Computers, 1967, 16:224-226

8 WALLACE, C.S. "A suggestion for a fast multiplier". IEEE Transactions on Electronic Computers, 1964, 13,7

9 WASER, Shlomo. "State of the art in high speed arithmetic integrated circuits". Computer Design, 1978, 17:67-75

10 MAJITHIA, J.C. & KITAI, R. "An iterative array for multiplication on signed binary numbers". IEEE Transactions on Electronic Computers. 1971, 20

11 Mc DONALD, T. G. & GUHA, R.K. "The two's complement quasi serial multiplier. IEEE Transactions on Computers, 1975, 124:1233-1235

12 HO, Irving T. & CHI CHEN, Tien. "Multiple addition by residue threshold functions and their representation by logic array". IEEE Transactions on Computers, 1973, 22, 8:762-767

APENDICE

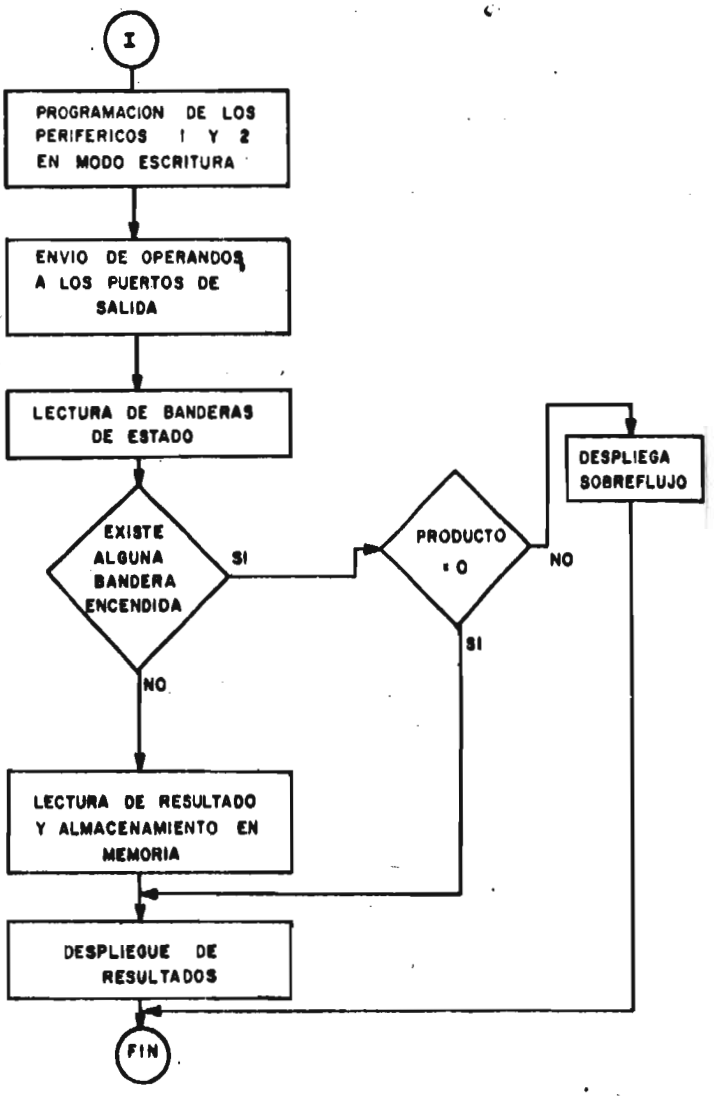


FIG. Nº 21 DIAGRAMA DE FLUJO DE LA SUBRUTINA DE MULTIPLICACION

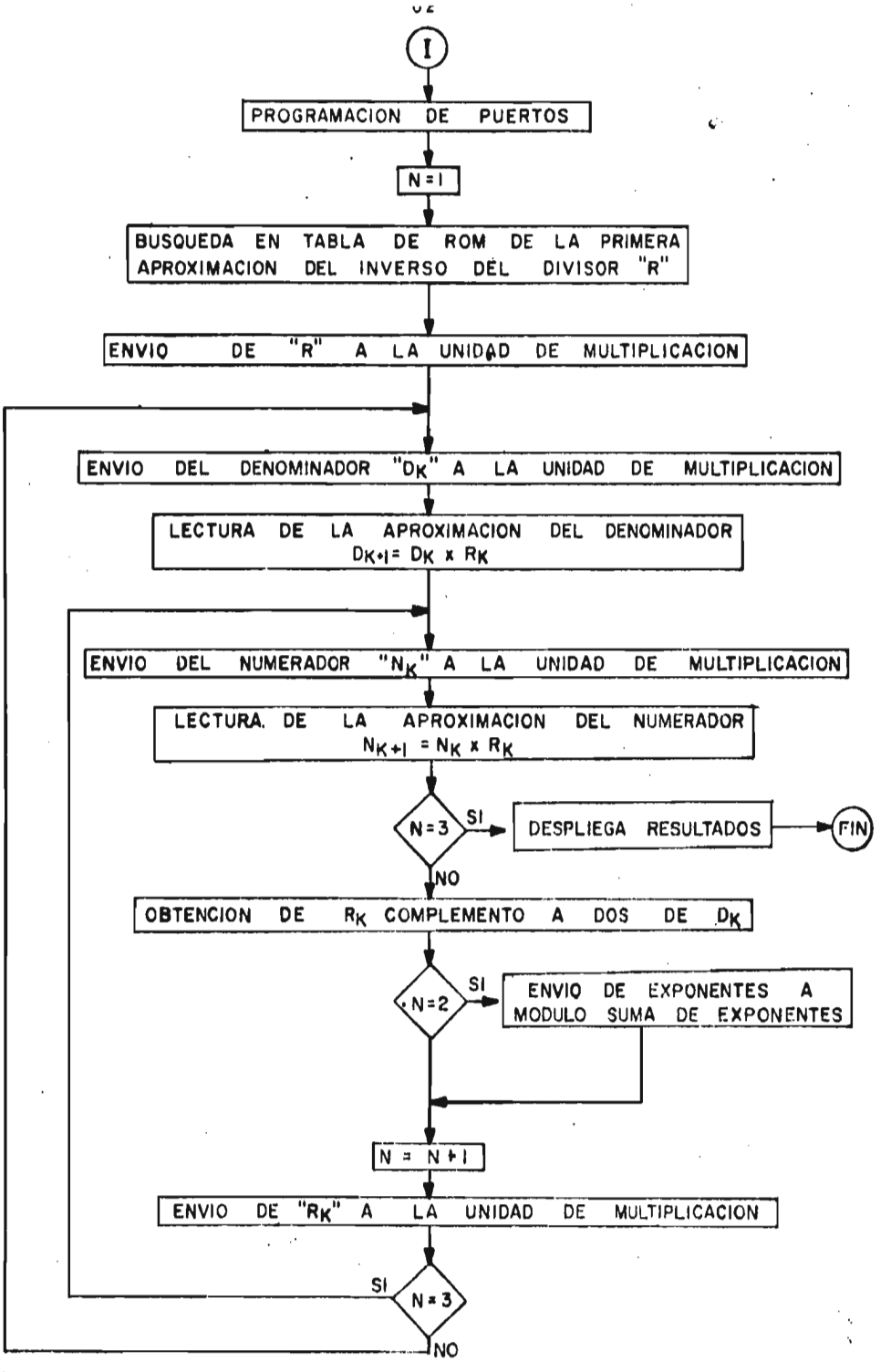

I
PROGRAMACION DE PUERTOS
N=1
BUSQUEDA EN TABLA DE ROM DE LA PRIMERA APROXIMACION DEL INVERSO DEL DIVISOR "R"
ENVIO DE "R" A LA UNIDAD DE MULTIPLICACION
ENVIO DEL DENOMINADOR "DK" A LA UNIDAD DE MULTIPLICACION
LECTURA DE LA APROXIMACION DEL DENOMINADOR DK+1 = DK x RK
ENVIO DEL NUMERADOR "NK" A LA UNIDAD DE MULTIPLICACION
LECTURA DE LA APROXIMACION DEL NUMERADOR NK+1 = NK x RK
N=3
SI
DESPLIEGA RESULTADOS
FIN
NO
OBTENCION DE RK COMPLEMENTO A DOS DE DK
N=2
SI
ENVIO DE EXPONENTES A MODULO SUMA DE EXPONENTES
N = N+1
ENVIO DE "RK" A LA UNIDAD DE MULTIPLICACION
SI
N=3
NO

TABLA 2

LISTA DE COMPONENTES

CLAVE	CANTIDAD	PRECIO UNITARIO	PRECIO TOTAL
74198	6	$ 2.00*	$ 12.00
7483	10	1.00	10.00
7476	2	0.40	0.80
7400	5	0.16	0.80
7404	5	0.20	1.00
74157	6	1.00	6.00
74381	5	1.50	7.50
7486	1	0.33	0.33
74161	1	1.10	1.10
74107	1	0.40	0.40
7493	2	0.40	0.80
74191	2	1.30	2.60
7432	2	0.33	0.66
9602	1	6.00	6.00
7410	4	0.20	0.80
7440	2	0.20	0.40
74125	10	0.60	6.00
8255	2	11.10	22.20
8708	1	15.40	15.40
			$ 94.79

* Precio en dolares, actualizado al 1º de febrero de 1979.

```
LOC  OBJ          SEQ          SOURCE STATEMENT

                    0 ;       ************************************************
                    1 ;
                    2 ;     SUBRUTINA PARA LA TRANSFERENCIA DE
                    3 ;     INFORMACION MICRO-UNIDAD ARITMETICA
                    4 ;     Y DESPLIEGUE DE RESULTADOS
                    5 ;
                    6 ;
                    7 ;          ************************
                    8 ;
                    9 ;  ALGORITMO DE MULTIPLICACION
                   10 ;
                   11 ;   ************************************
                   12
                   13 ;  ----------ESCRITURA DE OPERANDOS----------
                   14 ;
                   15 ;
0400               16 ORG 400H
000D               17 CR EQU 0DH
03FA               18 CO EQU 3FAH
000A               19 LF EQU 0AH
0400 DBE6          20 IN 0E6H;SET UNIDAD DE MULTIPLICACION
0402 3E80          21 MVI A,80H;PALABRA CONTROL MODO '0' ESCRITURA
0404 D3F7          22 OUT 0F7H;PROGRAMACION PERIFERICO # 1
0406 D3CF          23 OUT 0CFH;PROGRAMACION PERIFERICO # 2.
0408 210013        24 LXI H,1300H;CARGA DIRECCION DE OPERANDOS.
040B 7E            25 MOV A,M
040C D3EC          26 OUT 0ECH;EXPONENTE A HACIA PA2.
040E 23            27 INX H
040F 7E            28 MOV A,M
0410 D3ED          29 OUT 0EDH;EXPONENTE B HACIA PB2.
0412 23            30 INX H
0413 7E            31 MOV A,M
0414 D3EE          32 OUT 0EEH;MULTIPLICADOR MSB HACIA PC2.
0416 23            33 INX H
0417 7E            34 MOV A,M
0418 D3ED          35 OUT 0EDH;MULTIPLICADOR ISB HACIA PB2.
041A 23            36 INX H
041B 7E            37 MOV A,M
041C D3EC          38 OUT 0ECH;MULTIPLICADOR LSB HACIA PA2
041E 23            39 INX H
041F 7E            40 MOV A,M
0420 D3F6          41 OUT 0F6H;MULTIPLICANDO MSB HACIA PC1.
0422 23            42 INX H
0423 7E            43 MOV A,M
0424 D3F5          44 OUT 0F5H;MULTIPLICANDO ISB HACIA PB1.
0426 23            45 INX H
0427 7E            46 MOV A,M
0428 D3F4          47 OUT 0F4H;MULTIPLICANDO LSB HACIA PA1.
                   48 ;
                   49 ;     ************************************
                   50 ;  ---------- LECTURA DE RESULTADOS ----------
                   51 ;
                   52 ;
042A 23            53 INX H
042B 00            54 NOP;PALABRA CONTROL MODO '0' LECTURA.
042C 00            55 NOP;PROGRAMACION PERIFERICO # 1
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

0430 00             59 NOP
0431 DBEE           60 IN 0EEH;LECTURA BANDERAS DE PC2
0433 77             61 MOV M,A
0434 C3A104         62 JMP M1;SALTO  PRUEBA BANDERAS
0437 23             63 M0: INX H
0438 DBEC           64 IN 0ECH;LECTURA EXPONENTE DE PA2
043A 77             65 MOV M,A
043B 23             66 INX H
043C DBF6           67 IN 0F6H;LECTURA MANTISA MSB DE PC1
043E 77             68 MOV M,A
043F 23             69 INX H
0440 DBF5           70 IN 0F5H;LECTURA MANTISA ISB DE PB1
0442 77             71 MOV M,A
0443 23             72 INX H
0444 DBF4           73 IN 0F4H;LECTURA MANTISA LSB DE PA1
0446 77             74 MOV M,A
                    75 ;
                    76 ;     ****************************
                    77 ;  ---------- DESPLIEGUE DE RESULTADOS----------
                    78 ;
0447 0D             79 TIT: DB CR,LF,'EX MANTISA',CR,LF;CARGA TITULO
0448 0A
0449 4558204D
044D 414E5449
0451 5341
0453 0D
0454 0A
0455 214704         80 LXI H,TIT
0458 060E           81 MVI B,0EH
045A 4E             82 ESC: MOV C,M
045B CDFA03         83 CALL CO
045E 23             84 INX H
045F 05             85 DCR B
0460 C25A04         86 JNZ ESC
0463 210913         87 LXI H,1309H;APUNTA A RESULTADOS
0466 1600           88 MVI D,0H;INICIALIZA D
0468 3E00           89 M9: MVI A,0
046A BA             90 CMP D;INDICA PARTE DEL BYTE A CODIFICAR
046B CA7604         91 JZ M4
046E 15             92 DCR D
046F 7E             93 MOV A,M
0470 E60F           94 ANI 0FH;ENMASCARA MITAD MENOS SIGNIFICATIVA
0472 23             95 INX H;APUNTA SIGUIENTE CARACTER
0473 C38904         96 JMP M5
0476 3E0A           97 M4: MVI A,0AH;SALTA  SI DESPLEGO RESULTADO EX
0478 BD             98 CMP L
0479 C28104         99 JNZ M6
047C 0E20          100 MVI C,20H
047E CDFA03        101 CALL CO;DESPLIEGA ESPACIO
0481 14            102 M6: INR D
0482 7E            103 MOV A,M
0483 E6F0          104 ANI 0F0H;ENMASCARA MITAD MAS SIGNIFICATIVA
0485 0F            105 RRC;CORRE HACIA LA PARTE MENOS SIGNIFICATIVA
0486 0F            106 RRC
0487 0F            107 RRC
0488 0F            108 RRC
0489 FE0A          109 M5: CPI 0AH;VERIFICA SI CARACTER ES NUMERO O LETRA
048B D29304        110 JNC M7
048E C630          111 ADI 30H;SUMA PARA OBTENER CODIGO ASCII
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

0495 4F            114 MB: MOV C,A
0496 CDFA03        115 CALL C0;DESPLIEGA CARACTER
0499 3E0D          116 MVI A,0DH
049B BD            117 CMP L;VERIFICA FIN DESPLIEGUE DE RESULTADO
049C C26804        118 JNZ M9
049F CF            119 RST 1;FIN
                   120 ;
                   121 ;      **********************
                   122 ;  ---VERIFICA BANDERAS DE ESTADO-------
                   123 ;
04A0 00            124 NOP
04A1 E607          125 M1: ANI 07H;ENMASCARA BANDERAS DE ESTADO
04A3 FE00          126 CPI 0H;PREGUNTA SI ALGUNA ESTA ENCENDIDA
04A5 CA3704        127 JZ M0;REGRESA A LECTURA DE RESULTADOS
04A8 0600          128 MVI B,0;INICIALIZA CONTADOR
04AA 1F            129 M11: RAR;CORRE BANDERA A CARRY
04AB DAB204        130 JC M10;SALTA SI CY=1
04AE 04            131 INR B
04AF C3AA04        132 JMP M11;SALTA A VERIFICAR SIGUIENTE BANDERA
04B2 78            133 M10: MOV A,B
04B3 FE00          134 CPI 0H
04B5 CAC304        135 JZ M12;SALTA A PRODUCTO POR CERO
04B8 0E53          136 MVI C,53H;DESPLIEGA "S"
04BA CDFA03        137 CALL C0
04BD 0E46          138 MVI C,46H
04BF CDFA03        139 CALL C0
04C2 CF            140 RST 1;FIN
04C3 1E00          141 M12: MVI E,0
04C5 23            142 M14: INX H
04C6 3E0D          143 MVI A,0DH
04C8 BD            144 CMP L
04C9 CA4704        145 JZ TIT;SALTO DESPLIEGUE DE RESULTADOS
04CC 73            146 MOV M,E; CARGA CEROS EN LOCALIDADES DE RESULTADO
04CD C3C504        147 JMP M14
                   148 ;    ************************************************
                   149 ;
                   150 ;                    ALGORITMO DE DIVISION
                   151 ;
                   152 ;      ************************************************
04D0 0601          153 MVI B,1
04D2 3E80          154 MVI A,80H;     PROGRAMACION DE PUERTOS
                   155
04D4 D3D7          156 OUT 0D7H;     SET DIVISION
04D6 D3EF          157 OUT  0EFH
04D8 210213        158 LXI H,1302H;   BUSQUEDA EN TABLA DE ROM
04DB 7E            159 MOV A,M;     Y ENVIO DE PRIMERA APROXIMACION
04DC 5F            160 MOV  E,A;      DEL INVERSO DEL DIVISOR
04DD 1607          161 MVI D,07H
04DF 1A            162 LDAX D
04E0 D3EE          163 OUT 0EEH
04E2 7E            164 Y: MOV A,M;      ENVIO DEL DIVISOR 'DX'
04E3 D3F6          165 OUT 0F6H
04E5 23            166 INX H
04E6 7E            167 MOV A,M
04E7 D3F5          168 OUT 0F5H
04E9 23            169 INX H
04EA 7E            170 MOV A,M
04EB D3F4          171  OUT 0F4H
04ED 210D13        172 LXI H,130DH
```

```
LOC  OBJ          SEQ          SOURCE STATEMENT

04F2 00            175 NOP
04F3 00            176 NOP
04F4 DBF6          177 IN 0F6H;      LECTURA DE 'DK'
04F6 77            178 MOV M,A
04F7 23            179 INX H
04F8 DBF5          180 IN 0F5H
04FA 77            181 MOV M,A
04FB 23            182 INX H
04FC DBF4          183 IN 0F4H
04FE 77            184 MOV M,A
04FF 3E01          185 MVI A,1
0501 B8            186 CMP B
0502 C29305        187 JNZ R
0505 210513        188 LXI H,1305H
0508 7E            189 V:MOV A,M;      ENVIO DE NUMERADOR 'NK'
0509 D3F6          190 OUT 0F6H
050B 23            191 INX H
050C 7E            192 MOV A,M
050D D3F5          193 OUT 0F5H
050F 23            194 INX H
0510 7E            195 MOV A,M
0511 D3F4          196 OUT 0F4H
0513 210A13        197 LXI H,130AH
0516 00            198 NOP
0517 00            199 NOP
0518 00            200 NOP
0519 00            201 NOP
051A DBF6          202 IN 0F6H;      LECTURA DE 'NK'
051C 77            203 MOV M,A
051D 3E01          204 MVI A,1
051F B8            205 CMP B
0520 C23A05        206 JNZ M2
0523 7E            207 MOV A,M ;   DETECTA SI N>D,
0524 07            208 RLC;     SI SE CUMPLE, SE LE SUMA
0525 D23A05        209 JNC M2;UNO AL EXPONENTE DEL NUMERADOR
0528 210113        210 LXI H,1301H
052B 7E            211 MOV A,M
052C 07            212 RLC
052D 07            213 RLC
052E DA3705        214 JC M3
0531 7E            215 MOV A,M
0532 3C            216 INR A
0533 77            217 MOV M,A
0534 C33A05        218 JMP M2
0537 7E            219 M3: MOV A,M
0538 3D            220 DCR A
0539 77            221 MOV M,A
053A 210B13        222 M2: LXI H,130BH
053D DBF5          223 IN 0F5H
053F 77            224 MOV M,A
0540 23            225 INX H
0541 DBF4          226 IN 0F4H
0543 77            227 MOV M,A
0544 3E03          228 MVI A,3
0546 B8            229 CMP B
0547 CA9905        230 JZ X
054A 210F13        231 LXI H,130FH; , COMPLEMENTO A D[illegible] DE 'DK'
054D 7E            232 MOV A,M;    (NUEVO INVERSO)='NK'
054E 37            233 STC
```

```
LOC  OBJ          SEQ         SOURCE STATEMENT

0551 C601          236 ADI 1
0553 111213        237 LXI D,1312H
0556 12            238 STAX D
0557 2B            239 DCX H
0558 7E            240 MOV A,M
0559 2F            241 CMA
055A CE00          242 ACI 0
055C 1B            243 DCX D
055D 12            244 STAX D
055E 2B            245 DCX H
055F 7E            246 MOV A,M
0560 2F            247 CMA
0561 CE00          248 ACI 0
0563 1B            249 DCX D
0564 12            250 STAX D
0565 3E02          251 MVI A,2
0567 B8            252 CMP B
0568 CA8605        253 JZ U
056B 04            254 W: INR B
056C 211013        255 LXI H,1310H;     ENVIO DEL COMPLEMENTO "RK"
056F 7E            256 MOV A,M
0570 D3EE          257 OUT 0EEH
0572 23            258 INX H
0573 7E            259 MOV A,M
0574 D3ED          260 OUT 0EDH
0576 23            261 INX H
0577 7E            262 MOV A,M
0578 D3EC          263 OUT 0ECH
057A 3E03          264 MVI A,3
057C B8            265 CMP B
057D CA9305        266 JZ R
0580 210D13        267 LXI H,130DH
0583 C3E204        268 JMP Y
0586 210013        269 U: LXI H,1300H;      ENVIO DE EXPONENTES
0589 7E            270 MOV A,M;       RESET DIVISION
058A D3CD          271 OUT 0CDH
058C 23            272 INX H
058D 7E            273 MOV A,M
058E D3EC          274 OUT 0ECH
0590 C36B05        275 JMP W
0593 210A13        276 R:LXI H,130AH
0596 C30805        277 JMP V
0599 210813        278 X:LXI H,1308H;•     LECTURA DE RESULTADO
059C 00            279 NOP
059D 00            280 NOP
059E 00            281 NOP
059F 00            282 NOP
05A0 DBEE          283 IN 0EEH
05A2 77            284 MOV M,A
05A3 23            285 INX H
05A4 DBEC          286 IN 0ECH
05A6 77            287 MOV M,A
05A7 C34704        288 JMP TIT;      SALTO A DESPLIEGUE DE RESULTADOS
                   289 END


PUBLIC SYMBOLS
```