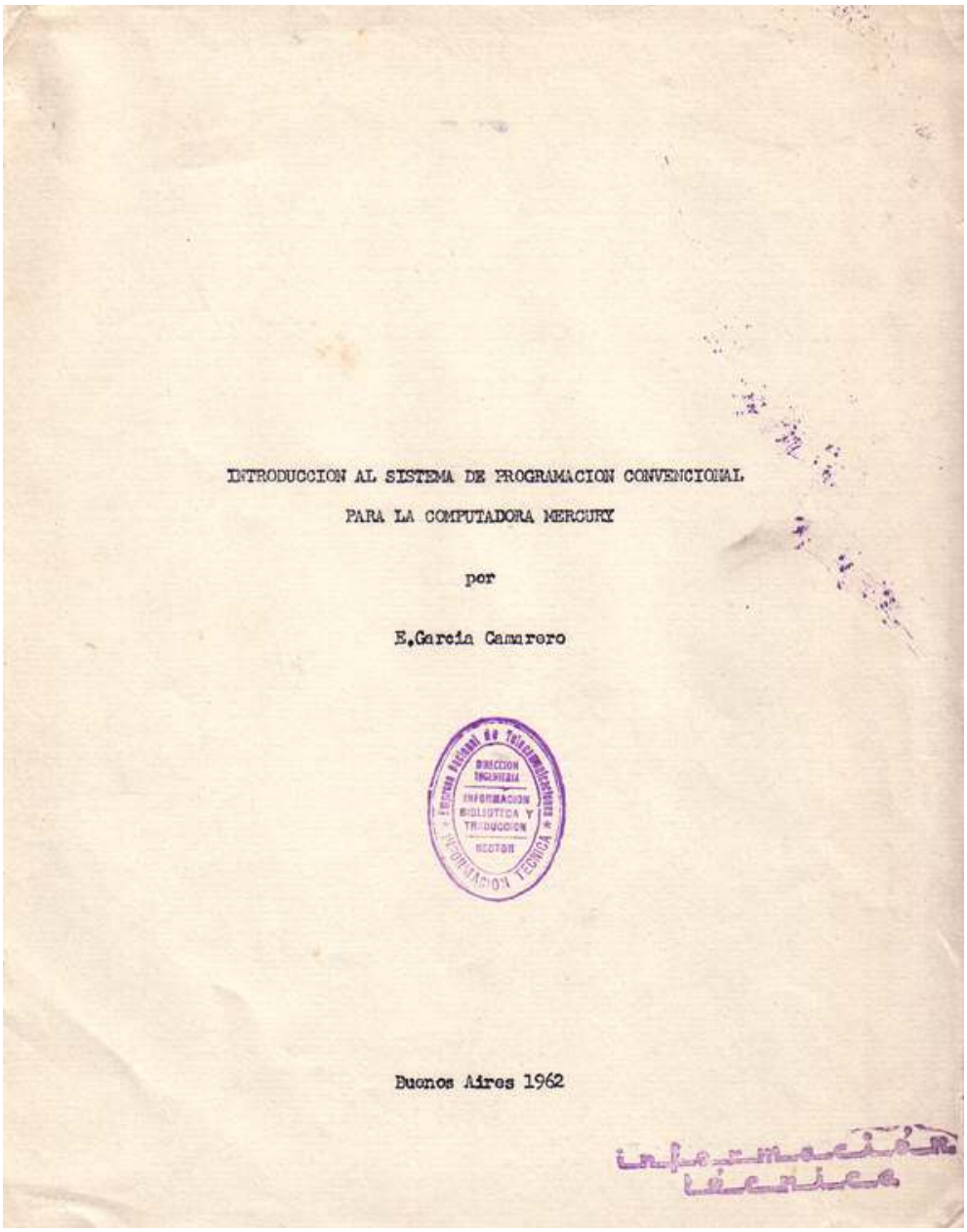


Portada de un ejemplar distribuido como «Información técnica» por la Dirección de Ingeniería de la Empresa Nacional de Telecomunicaciones de Buenos Aires [para información, biblioteca y traducción].

INTRODUCCION AL SISTEMA DE PROGRAMACION CONVENCIONAL

PARA LA COMPUTADORA MERCURY

por

E. García Camarero

Buenos Aires 1962

1. Generalidades.

El sistema convencional de programación simplificada para la computadora Mercury tiene los siguientes objetivos principales:

a) Sustituir la expresión binaria de toda la información interna, en expresión decimal para la información externa (programa datos con lo que se consigue mejor nemotécnica y reducción en el número de dígitos que involucra cada unidad de información.

b) Disponer de direcciones flotantes y relativas, y de marcas para identificar instrucciones, resolviendo así la rigidez del uso único de direcciones fijas.

c) Disponer de directivas que facilitan la organización de un programa.

Todo esto se consigue en la ejecución de un programa traductor llamado PIG 2, que vierte nuestro programa escrito en convencional al lenguaje binario absoluto de la máquina.

La presencia de los anteriores objetivos no limita en nada la potencia del lenguaje absoluto y lo hace mas flexible.

2. La máquina Mercury: Las partes esenciales de la Computadora Mercury son:

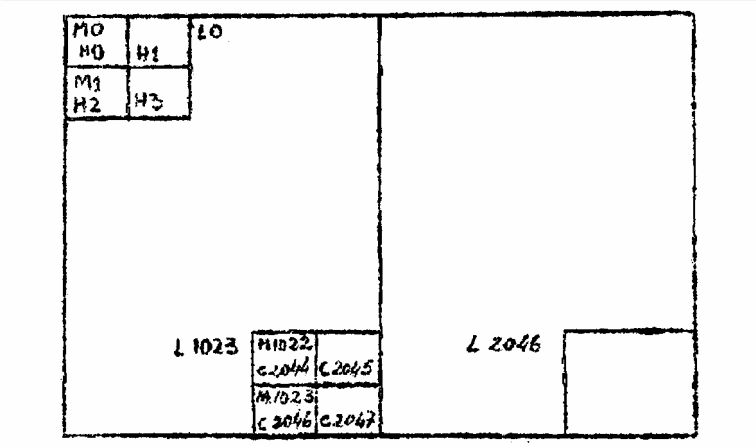
1 Memoria.
2 Órganos de entrada y salida.
3 Unidad Aritmética,
4 Órgano de Control.
5 Consola.

2.1 La memoria: Es el órgano de la máquina en el que se almacena la información, tanto numérica (datos) como operativa (instrucciones). Como las unidades de información de cada tipo son de longitud distinta tenemos distintos tamaños para las casillas en donde esas informaciones se deben almacenar, ya sea por unidades o por bloques de unidades. Para el primer caso usamos una memoria de acceso al azar y de gran velocidad de acceso que llamamos memoria rápida, y para el segundo (almacenamiento por bloques) utilizamos otra memoria de acceso semi al azar, que llamamos memoria grande.

La memoria rápida : La memoria rápida esta constituida por núcleos magnéticos de ferrita. Cada núcleo según su estado magnético representa un uno o un cero cifras base del sistema binario, llamadas brevemente bit (binary digit). El número total de bits de la memoria rápida es de 40960. El acceso a la información lo hacemos no bit por bit sino por grupos de bits. A estos grupos o compartimentos donde se almacenan las unidades de información los llamaremos registros. Estos registros los llamaremos según su longitud registros largos (con capacidad para 40 bits) registros medios (20 bits), registros cortos (10 bits) y los indicaremos abreviadamente mediante las letras L, M y H respectivamente. A cada registro se asocian dos números, uno su contenido y otro su dirección (o domicilio) que nos indica su ubicación.

En la memoria rápida podemos identificar 1024 largos numerados de 0 a 2046; l024 registros medios numerados de 0 a 1023, que ocupan la misma zona que los primeros 512 registros largos y 2048 registros cortos que ocupan esta misma zona. Es decir que en la primera mitad de la memoria rápida se pueden almacenar y retirar unidades de información de longitud de 40, 20 o 10 bits de longitud.

La memoria rápida la podemos considerar dividida en 32 partes iguales llamadas páginas numeradas de 0 a 31.



Las dieciséis primeras páginas (P.0 - P.15) pueden, cada una, contener 32 registros largos, 64 registros medios o 128 registros cortos; las ultimas páginas (P.16 - P.31) contienen solo registros largos.

La memoria rápida está conectada con todos los órganos de la computadora.

La memoria grande. La memoria grande esta constituida por dos tambores magnéticos. En ellos la información se o almacena por bloques, cada bloque tiene la misma capacidad de información que una página de la memoria rápida; a estos bloques los llamaremos sectores. El número total de sectores que contiene la memoria grande es de 512 (*) numerados de 0 a 511. Cada 32 diremos que forman una zona.

La memoria grande esta únicamente conectada con la memoria rápida, las transferencias entre ambas deberán hacerse por bloques de información pasando de un sector a una página o viceversa.

2.2 Órganos de entrada y salida. Tanto la información numérica (datos) como la información operativa (instrucciones) deberán ser introducidas en la máquina antes de que comience el proceso de cálculo. Los órganos de entrada y salida están diseñados según sea el soporte material externo sobre el que se fija la información, en el caso de la Mercury el soporte de entrada y salida de la información es cinta perforada de papel de cinco canales, y los órganos de entrada y salida son respectivamente un lector fotoeléctrico y un perforador electromecánico de cinta de papel.

Tanto para la edición del programa y los datos como para la interpretación de los resultados hay un equipo electromecánico exterior (fuera de línea) similar al usado por los servicios de teletipo.

(*) Nos referimos al ejemplar instalado en el Instituto de Cálculo de la Universidad de Buenos Aires.

2.3 La unidad aritmética. Es el órgano donde se ejecutan las operaciones aritméticas de suma, diferencia y producto, y las operaciones lógicas de conjugación y no equivalente.

A la unidad aritmética la podemos considerar constituida por:

– El acumulador.
– Los registros B.

- El acumulador. Es un registro especial de 40 bits donde se ejecutan las operaciones aritméticas con números largos (son números largos los contenidos en registros largos). Como los números largos están almacenados en la forma llamada de punto flotante, los 40 bits están distribuidos en 30 bits para la mantisa de dicho número y 10 para el exponente. Los indicaremos brevemente por las letras m y e. Todas las operaciones aritméticas hechas en el acumulador se ejecutan con la técnica llamada flotante.

- Los registros B. Son siete registros, de 10 bits, numerados de 1 a 7 (El registro B_0, inexistente, actúa como si permanentemente tuviera almacenando el número cero). Al registro B_7, que tiene algunas propiedades, se lo llama SAC o acumulador corto. Existen además otros dos registros de 2 bits llamados B-test y S-test respectivamente y en los que únicamente se almacena el signo de un número

Además de poder ejecutarse en los registres B las operaciones aritméticas y lógicas a que antes aludíamos tienen una función especial muy importante llamada B-modificación, que mas adelante veremos con detalle.

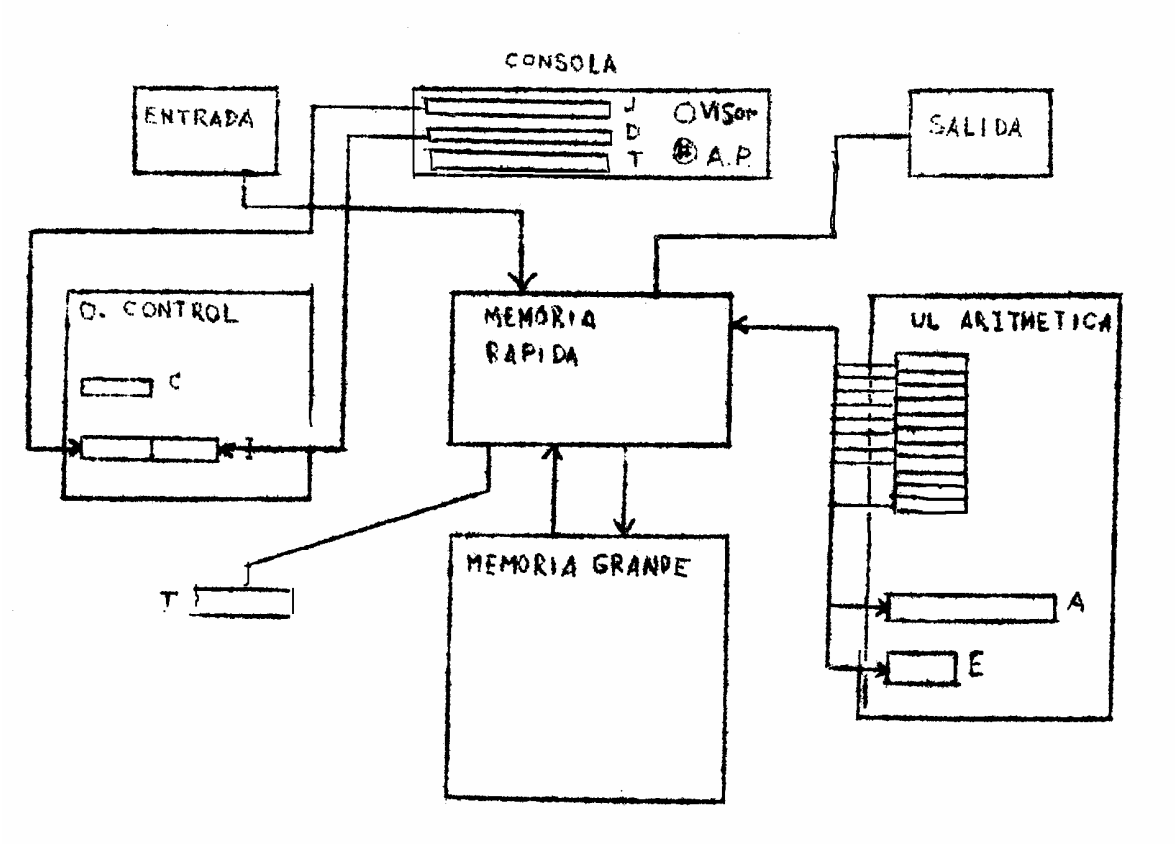
2.4 Órgano de control. Mediante este órgano se decodifican y se ordena la ejecución de las instrucciones que componen nuestro programa. El órgano de control esta compuesto por dos registros uno de 10 bits en donde se almacena la dirección de la instrucción que se quiere ejecutar, y otro de 20 bits en donde se almacena dicha instrucción durante la decodificación. Al primer registro lo llamaremos C y al segundo I Desde este órgano se envían las señales necesarias a los demás de la computadora para que ejecuten las operaciones especificas indicadas en las instrucciones.

2.5 La Consola. Para la comunicación directa entro el hombre y la computadora, es decir, sin necesidad de utilizar los órganos de entrada y salida, la computadora se sirve de la consola o tablero de mando.

La comunicación manual del operador con la computadora se hace a través de tres grupos de 10 llaves, cada uno de los cuales se puede considerar como un registro de 10 bits. El primero, que llamaremos J, está directamente comunicado con la primera parte del registro I del órgano de control. El segundo, que llamaremos D, esta comunicado con la segunda parte del registro I del órgano de control. El tercero, que llamaremos F, lo está directamente con la memoria rápida.

La comunicación directa de la computadora se hace a través de un altoparlante que emite, bajo control del programa, una señal sonora y de dos visores en los que aparece visiblemente el contenido de los registros B1 - B7, A, E, I, C ya mencionados. (*)

(*) Del registro I aparecen solo los 10 bits más significativos; además existen otras dos posibilidades llamadas Display A, E.



3. Codificación de la información en Mercury. La información en Mercury puede ser de dos tipos: operativa (instrucciones) y numérica (datos, resultados) la primera viene dada por unidades de 20 bits y la segunda por unidades de 10 o 40 bits.

3.1 Instrucciones. Los 20 bits que constituyen una instrucción están distribuidos de la siguiente forma:

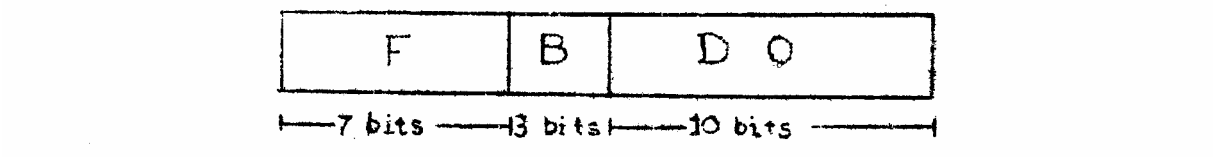
– Los siete bits primeros que llamamos parte de función y designaremos por la letra F los utilizamos para identificar el tipo de operación que ejecutará la instrucción al ser decodificada. Con estos siete bits podemos identificar hasta 128 instrucciones de las cuales no todas están designadas en Mercury.

– Los tres bits siguientes identifican el registro B sobre el que se opera, o bajo el que se ejecuta la B - modificación. Con estos tres bits se pueden identificar hasta ocho registros B (0 - 7) aunque en Mercury solo existen siete (1 - 7). Como ya indicamos el B0 se supone almacenando siempre cero.

– Los diez bits últimos pueden ser interpretados de dos formas

a. – Como la dirección del operando a que se refiere la función. Este operando puede ser un numero o una función.
b. – Como el operando mismo, considerado este como un numero entero.

La forma de las instrucciones quedan representadas en la figura siguiente.



3.2 Los números. La representación numérica puede tomar dos formas según quo se utilicen por ella grupos de 10 o 40 bits, llamando a los primeros números cortos y a los segundos números largos.

Los números cortos se consideran como enteros con signo y los pesos relativos vienen dados por

$$-2^9 \quad 2^8 \quad 2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$$

que representarían un número

$$-2^9\, b_9 + 2^8\, b_8 + 2^7\, b_7 + 2^6\, b_6 + 2^5\, b_5 + 2^4\, b_4 + 2^3\, b_3 + 2^2\, b_3 + 2^1\, b_1 + 2^0\, b_0$$

Por tanto los números positivos comenzaran con 0, es decir $b_9 = 0$, y los números negativos con 1 ($b_9 = 1$).

El número positivo mayor será

$$01111\ 11111 \quad = \quad 511$$

el número negativo de mayor módulo será

$$10000\ 00000 \quad = \quad -512$$

luego los números cortos estarán incluidos en el intervalo;

$$(-512, 511)$$

Ejemplos:

$$\begin{aligned} -1 &= 11111 \quad 11111 \\ 1 &= 00000 \quad 00001 \\ -2 &= 11111 \quad 11110 \end{aligned}$$

Fenómenos de overflow. Se llaman fenómenos de overflow a la aparición en una operación aritmética cantidades que sobrepasan la capacidad del registro que deberá recibir el resultado Esto tiene por efecto perder la par te mas significativa de dicho resulta de hacerlo cambiar de signo, por ejemplo:

256 = 01000 00000
256 = 01000 00000
---
512 ≠ l0000 00000 = −522

−512 = 10000 00000
−512 = 10000 00000

−1024 ≠ 1 00000 00000 = 0

– Los números largos. Los números de 40 bits utilizan la forma de punto flotante para su expresión, es decir:

$$(x, y) = x \cdot 2^{y}$$

donde a x llamamos mantisa, y a y exponente.

– De los cuarenta bits de un número largo 30 se usan para la mantisa con los siguientes pesos

$$-2^{0} \quad 2^{-1} \quad 2^{-2} \quad 2^{-3} \ldots . \quad 2^{-29}$$

que representan los números en la forma

$$-2^{0} \; b_0 + 2^{-1} \; b_1 + 2^{-2} \; b_2 + 2^{-3} \, b_3 + \ldots + 2^{-29} \, b_{29}$$

las mantisa con b_0= 0 son números positivos y b_0 = 1 dan números negativos, por ejemplo:

01000 00000 00000 00000 00000 00000 = 1/2

11000 00000 00000 00000 00000 00000 = −1/2

Nota: La representación binaria de un número viene dada por la sumatoria

$$\sum_{n,m} b_i \, 2^{i}$$

m indica el número de bits de la parte entera, y n el de bits fraccionarios.

La mantisa positiva mayor será:

$$01111\ 11111\ 11111\ 11111\ 11111\ 11111 = 1 - 2^{-29}$$

y la mantisa negativa de mayor valor absoluto será

$$10000\ 00000\ 00000\ 00000\ 00000\ 00000 = -1$$

luego el intervalo de variabilidad de las mantisas es:

$$-1 \leq x \leq 1$$

– Para el exponente de un número largo se usan 10 bits, que representarán un entero con signo siguiendo al convenio dado en el párrafo anterior. Es decir que el intervalo de variabilidad de los exponentes es

$$-512 \leq y \leq 511$$

Intervalo normal. Como la representación de un número en forma flotante no es única, es preciso para lograrlo fijar a la mantisa en un intervalo de terminado y modificar el exponente lo que sea preciso. El intervalo elegido con el objeto de aprovechar el máximo de información es:

$$-1 \leq x < -1/2$$
$$1/2 \leq x < 1$$

lo que se consigue exigiendo a los números positivos que

$$b_1 = 1$$

y los números negativos que

$$b_1 = 0$$

a este intervalo se llama normal o standard, a los números incluidos en este intervalo números en forma normal o standard y a la operación que reduce un número a su forma normal normalización o standardización.

En el exponente de un número expresado en la forma de punto flotante se presenta el peligro de overflow y para detectarlo fácilmente se exige que el exponente comience con dos ceros si es positivo y con dos unos si son negativos, lo que reduce el intervalo de variabilidad de los exponentes a

$$(-256\ ,\ 255)$$

y el intervalo de variabilidad de los números largos será

$$(2^{-256}\ ,\ 2^{555})$$

que corresponde hablando en términos decimales aproximadamente a:

$(10^{-70}, 10^{70})$

4.El sistema convencional. La unidad operativa elemental es la instrucción; cada instrucción tiene una parte de función a la que se asigna un decimal entre 0 y 99; una parte B identificada por un entero decimal entro 0 y 7 que indica el registro B sobre el que se opera; y una parte de dirección, que puede tomar diversas formas (direcciones fijas, flotantes, relativas, paramétricas) que indica el registro de memoria sobre el que opera la función.

Un conjunto de instrucciones con un fin específico forman otra unidad operativa que llamamos rutina. En el sistema convencional tenemos tres tipos de rutinas: v - rutinas, x - rutinas, y las q - rutinas.

Un conjunto de rutinas forman un capítulo, y uno o varios capítulos forman un programa.

Previo a la ejecución de un programa escrito en lenguaje o convencional, el programa traductor PIG 2, vierte cada una de sus instrucciones al lenguaje binario interno de la máquina. Las instrucciones correspondientes a una rutina quedan almacenadas en la memoria rápida. Cuando se completa la traducción de todas las rutinas correspondientes a un capitulo se almacenan en la memoria grande y se comienza la traducción del capítulo siguiente o se pasa al proceso de ejecución.

Para la ejecución de la instrucción, cuya dirección esta registrada en el registro C del órgano de control, se la transfiere al registro I del mismo y se decodifica siguiendo los pasos siguientes:

- analiza la porte de función
- según el análisis anterior interpreta la parte de dirección del operando como la dirección de un registro corto, medio o largo, o como un entero,
- En las funciones llamadas B - modificables, se suma a la parte de dirección el contenido del registro B indicado en la instrucción. A esta operación se llama B–modificación.
- Se suma 1 al registro C, para proceder al análisis de la instrucción siguiente salvo en las instrucciones de salto en las que la parte de dirección del operando pasan al registro C.

En lo que sigue emplearemos las siguientes notaciones:

- registro B, a veces se particu1arizara con un subíndice a un B-registro determinado.
- S registro B_7
- Bt registro B - test.
- St registro S - test.
- A acumulador.
- E exponente del acumulador
- C registro C del órgano de control
- H dirección de un registro corto de memoria
- L dirección de un registro largo de memoria
- n número entero
- ti un carácter de cinco bits de la cinta de entrada
- to un carácter de cinco bits de la cinta de salida.
- T registro T

Los anteriores símbolos marcados con un acento indican el estado del registro correspondiente después de ejecutada la instrucción.

4.1 Direcciones fijas. Al número de 10 bits que se asocia a cada registro de memoria rápida, para su identificación, se le llama dirección absoluta de un registro. Estos 10 bits representan números binarios enteros con los siguientes pesos.

$$2^9 \quad 2^8 \quad 2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$$

por lo tanto la dirección mas pequeña es

00000 00000 = 0

y la mayor

11111 11111 = 1023

y podemos considerar que el registro siguiente al 1023 es el registro 0 ya que

```
 11111 11111
           +1
------------------
1 00000 00000
```

perdiéndose el dígito mas significativo por overflow. Análogamente si a la dirección 0 le restamos un uno nos dará una posición determinada, por ejemplo

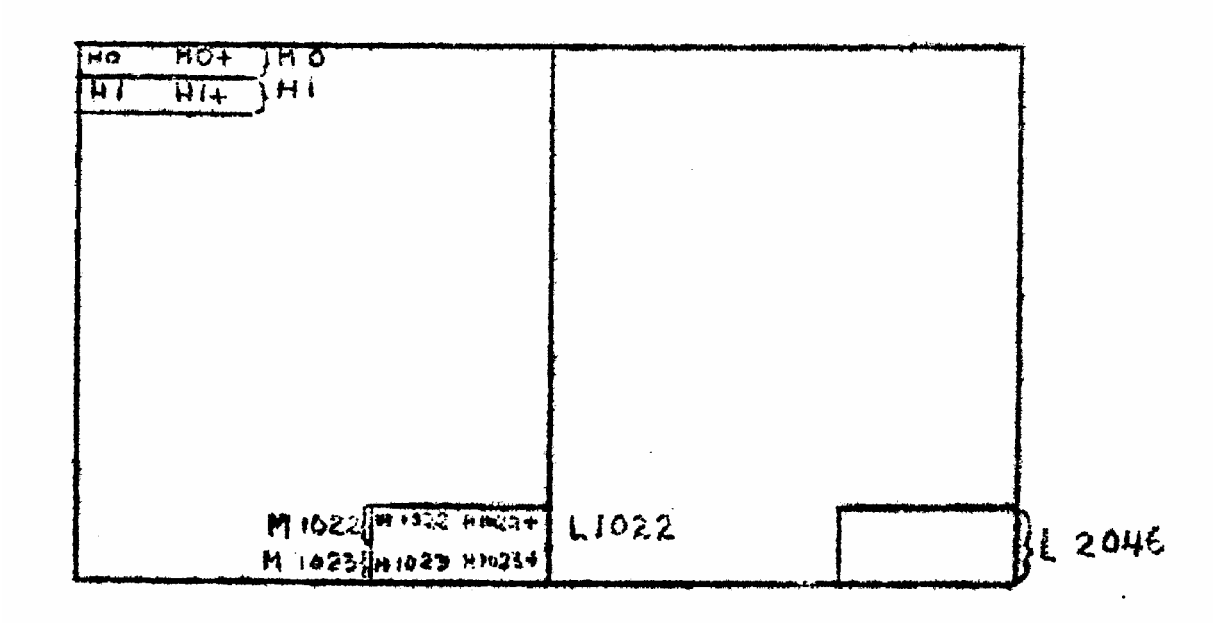
```
 00000 00000
           −1
------------------
 11111 11111
```

lo que nos dice que el registro anterior al 0 es el 1023, por ello es lo mismo considerar la dirección −1 que 1023, −2 que 1022, ... , −1023 que 1.

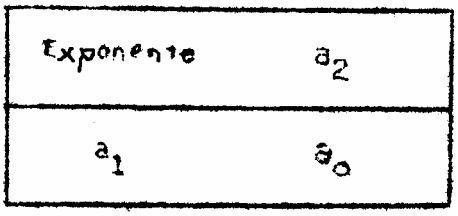
Direcciones fijas. En el sistema convencional, expresamos las direcciones mediante números decimales, que el programa traductor PIG 2 vierte en direcciones binarias o absolutas. Indica direcciones de registros largos mediante les números pares 0, 2, 4... 2046; los registros medios mediante la sucesión 0, l,. . . ., 1023 los registros cortos mediante la sucesión 0, 0+, 1 , 1+, . . . , 1023 , 1023+. Teniendo en cuenta lo dicho anteriormente los intervalos de variabilidad de las direcciones son:

```
−2046   L   2046
−1023   M   1023
−1023   H   1023+
```

El siguiente esquema indica la posición relativa de los registros.



Los registros largos almacenan números en forma de punto binario flotante, El primero de los cuatro registros cortos de que esta constituido contiene el exponente de 10 bit, y la mantisa ocupa los tres registros siguientes, como indica. el esquema.



siendo a_0 la parte mas significativa y a_2 la menos significativa de la mantisa.

Para comodidad se pueden agrupar los registros de memoria en páginas. Una página tiene 32 registros largos, 64 registros medios y 128 registros cortos. La expresión de las direcciones se puede también dar por un par de números enteros separados por un punto: el primero indica la página y el segundo el registro dentro de la página.

Ejemplos:

0.0	indica	L_0	M_0	H_0
0.1	indica		M_1	H_1
l.2+	indica			H_{66+}
31.62	indica	L_{2046}		

4.2 Funciones. Los siete primeros bits de una instrucción indican la función u operación elemental que ejecutara la máquina al interpretarla. Hay funciones según sobre el registro especial (B-registro, Acumulador,...) que operen o según la clase de operación que efectúen. Nosotros las dividiremos como sigue:

– Registros B.
  aritméticas
  lógicas
  salto

– El acumulador
  Sobre el acumulador
  sobre el exponente
  salto

– Entrada y salida. Transferencias entre memorias
– Funciones especiales.

Los registros B. Como parte de la unidad aritmética ya indicamos la existencia de siete registros cortos de 10 bits en donde se pueden ejecutar operaciones aritméticas y lógicas con números enteros. El registro B7 goza de la propiedad particular de que en el se puede aplicar la B - modificación. Al registro B7 lo llamaremos acumulador corto o brevemente sac.

Hay otros dos registros Bt y St de dos únicos bits en el que se almacena automáticamente el signo del contenido del último registro B utilizado; o bajo instrucción, el signo de la diferencia entre un registro B y registro corto de memoria o un entero, o la diferencia entre sac y un registro corto o un entero, respectivamente. La utilidad de estos registros es el control o recuento en los ciclos de operaciones, que veremos mas justificados cuando hablemos de las operaciones de salto.

Los registros B están directamente comunicados con la primera mitad de la memoria rápida con la que se puede hacer transferencias en ambos sentidos y con el exponente del acumulador; pero no se pueden comunicar directamente los registros B entre si y necesitan para una transferencia entre ellos el paso intermedio o por la memoria.

Funciones aritméticas en registros B

00b	$B'_b = H$	$Bt' = sig\ H$
01b	$H' = B_b$	
02b	$B'_b = B_b + H$	$Bt' = sig\ (B_b + H)$
03b	$B'_b = B_b - H$	$Bt = sig\ (B_b - H)$
04b	$B'_b = ½B_b - H$	$Bt' = sig\ (½B_b - H)$
10b	$B'_b = n$	$Bt' = sig\ n$
(*) 31b	$B'_b = S + n$	$Bt' = sig\ (S + n)$
12b	$B'_b = B_b + n$	$Bt' = sig\ (B_b + n)$
13b	$B'_b = B_b$	$Bt' = sig\ (B_b - n)$
14b	$B'_b = ½\ B_b - n$	$Bt' = sig\ (½B_b - n)$

(*)Incluimos aquí esta instrucción por no ser B-modificable.

Ejemplos:

– Sumar los exponentes de los números (e_1 , m_1) y (e_2 , m_2) almacenados en los registres largos 2.0 y 2.2, poner el resultado en H 2.63. Comenzar la secuencia de instrucciones en el registro medio 1.0

l.0	002	2.0	$B'_2 = e_1$
1.1	022	2.2	$B'_2 = e_1 + e_2$
l.2	012	2.63	$e_1 + e_2 \rightarrow 2.63$

– Restar 15 al entero m almacenado en la mitad derecha de M.178, comenzando la secuencia de instrucciones en la pag.8

8.0	007	178+	S' = m
8.1	137	15	S' = m−15
8.2	017	178+	m' = m−15

Como en el grupo anterior de instrucciones, necesitamos identificar en que particular registro B queremos operar y para ello usamos el carácter, b, estas instrucciones no son B- modificables. Para tener esta operación con números cortos, existe un grupo especial funciones B modificables que operan siempre sobre S (B7) que son modificables por todos los registros B inclusive el B7. La B- modificación opera de la siguiente manera, según se considere la parte de dirección del operando como una dirección de memoria H o como un entero n:

H'= H + ½ B
n' = n + B

El de funciones B - modificables sobre S, es idéntico al grupo de instrucciones dado anteriormente con la salvedad de que comienzan por 2 y 3 en lugar de 0 y 1. Exponemos la tabla a continuación:

20	S' = H	St' = sig H
21	H' = S	
22	S' = S + H	St' = sig (S + H)
23	S' = S − H	St' = sig (S − H)
24	S' = ½S − H	St' = sig (½S − H)
30	S'= n	St'= sig n
32	S'= S + n	St'= sig (S + n)
33	S'= S − n	St'= sig (S−n)
34	S'= ½ S − n	St'= sig (½S − n)

Ejemplo:

– Poner en S la suma de los contenidos de todos los registros B.

300	0
321	0
322	0
323	0
324	0
325	0
326	0

Lo esencial en las funciones 04, l4, 24 y 34 es encontrar la mitad del contenido de un registro B o, lo que es lo mismo, realizar un desplazamiento hacia la derecha de una posición. Para duplicar el contenido de un registro B, se puede utilizar la operación de suma, y en particular se utiliza el registro S, la instrucción 327 0.

Funciones lógicas. Con números cortos y sobre registros B podemos efectuar las operaciones lógicas conjunción (&) y no equivalente (#) dadas por las siguientes tablas:

&	0	1
0	0	0
1	0	1

#	0	1
0	0	1
1	1	0

Ejemplos

```
    10011 10100
&   11111 00000
-----------------------
    10011 00000

    10011 10100
#   11111 00000
-----------------------
    01100 10100
```

Nótese que en el primer caso los cinco primeros bits quedan invariantes, y en el segundo quedan sin cambiar los últimos cinco bits. Estas operaciones las ejecuta la máquina con el siguiente grupo de funciones:

No B-modificables:

05 b	B'b = Bb & H	Bt' = sig (Bb & H)
06 b	B'b = Bb # H	Bt' = sig (Bb # H)
15 b	B'b = Bb & n	Bt' = sig (Bb & n)
16 b	B'b = Bb # n	Bt' = sig (Bb # n)

B - modificables

25 b	S' = S & H	St' = sig (S & H)
26 b	S' = S # H	St' = sig (S # H)
35 b	S' = S & n	St' = sig (S & n)
36 b	S' = S # n	St' = sig (S # n)

La conjunción sirve para extraer de una palabra corta el valor de uno o varios bits determinados, ya que como en la aplicación de la conjunción aparece uno solo si los bits correspondientes son iguales a uno y cero en los demás casos, bastará aplicar a la palabra dada otra compuesta por unos en las posiciones en que queremos extraer el valor de los bits, y cero en los restantes. A estas palabras las llamaremos máscaras.

Ejemplos:
- Es par o impar el número almacenado en H 4.0+ ?
Bastara paro ello saber si su último bit es cero o uno es decir aplicarle la mascara 00000 0000l, para lo cual bastan las instrucciones

```
001  4.0+
151  1
```

y en B1 habrá 0 si es par y 1 si es impar.

– Sustituir el contenido de H 4.0 por su mínimo resto modulo 128

```
001 4.0
151 127
010 4.0
```

– Llevar las cifras 6º, 7º y 8º de B6 a la parte menos significativa de Sac.

```
146 0      B'6      = ½ B6
146 0      B'6      = ½ B6        desplaza dos lugares a la derecha
156 7      B'6      = B6&7        7 = 00000 00111
016 4.0    (H4.0)'  = B6
007 4.0    S'       = H 4.0
```

La operación de no equivalente hace que el resultado sea uno en los lugares en que los bits correspondientes de los operando son distintos y cero si estos son iguales. Las siguientes propiedades son inmediatas

$$u \# u = 0$$
$$u \# 0 = u$$
$$u \# (u \# v) = v$$
$$v \# (u \# v) = u$$
$$u \# (-1) = -(u + 1)$$

Ejemplo:

Hallar el complemento a 2 del contenido n del registro corto H 4.0.

Para ello es suficiente cambiar los 0 por 1 y los 1 por 0, cosa que logramos mediante las instrucciones.

```
01 1    4.0     B'1 = n
16 1    −1      B'1 = n # −1     −1 = 11111 11111
00 1    4.0
```

Funciones de B - salto. Llamamos así a las funciones de salto condicionado al contenido de los registros Bt y St. Estas funciones se utilizan para controlar los ciclos de operaciones y las rupturas de secuencia. Las funciones son las siguientes:

no B - modificables

```
08    si Bt ≠ 0        c'= n
09    si Bt ≥ 0        c'= n
18    si Bt ≠ 0        c'= n   y   B' = B+1   Bt = sig (B + 1)
```

B - modificables

28	si $S \neq 0$	c' = n
29	si $S \geq 0$	c' = n
38	si $St \neq 0$	c' = n y S'=S + 1 St= sig (S + 1)

A veces necesitamos usar previamente a las anteriores, las siguientes instrucciones de comparación

No B-modificables

07	Bt' = B − H
17	Bt' = B − n

B - modificables

27	St' = S − H
37	St' = S − n

Ejemplos:

– Dados los números n_i (i 0(1)124) almacenados en los registros H 4.0 a H 4.62, formar la sumatoria

m=\sum n_i1

y almacenar m en 5.0. Comenzar la secuencia de instrucciones en la página 1.

1.0	20 0	4.0	S' = n0		
1 1	0 1	− 123	B1 = −123		
1.2	22 1	4.62	S' = S + ni	i=1(1)124	
1.3	18 1	1.2	si $Bt \neq 0$	C' = 1.2	B1 = B1 + 1
1.4	21 0	5.0	$\Sigma\, n_i \rightarrow 5.0$		

– Si n_1 y n_2 son enteros positivos almacenados en H 2.0 y H 2.0+ y tales que verifiquen

$$0 \leq n1 \;\; n2 \leq 1024$$

calcular su producto y colocarlo en H 1.0. Comenzar la rutina en la pagina 3.

3.0	106	0
3.1	036	2.0
3.2	200	2.0
3.3	270	2.0+
3.4	290	3.8
3.5	106	0
3.6	036	2.0
3.8	210	3.9+
3.7	200	2.0+

3.9	337	0
3.10	220	3. 9+
3.11	186	3.10
3.12	2.0	1.0

Funciones en el acumulador. El acumulador trabaja con números largos en forma de punto flotante, que representaremos

$$X = m\ 2^e = (m, e)$$

Las operaciones aritméticas se realizaran automáticamente según el siguiente esquema.

Suma: Dados los sumandos

$$X_1 = (m_1, e_1)$$
$$X_2 = (m_2, e_2)$$

con m^1 y m_2 en forma normal tendremos

si $e_1 \geq e_2$

$$X_3 = (m_3, e_3) = X_1 + X_2 = (m_1 + m'_2, e_1) \quad \text{con} \quad m'_2 = (m_2, e_2 - e_1)$$

si $e_2 \geq e_1$

$$X_3 = (m_3, e_3) = X_1 + X_2 = (m'_1 + m_2, e_1) \qquad \text{con} \quad m'_1 = (m_1, e_1 - e_2)$$

donde en general m^3 no será de forma normal.

Producto: Dados X_1 y X_2 con la misma condición de antes, tendremos

$$X_3 = (m_3, e_3) = X_1X_2 = (m_1.m_2, e_1 + e_2)$$

en donde m_3 no estará en general sobre el intervalo normal.

Errores. Dado que el acumulador tiene un número fijo y finito de dígitos, después de cada operación aritmética, se produce un error llamado de truncación al no poder considerar las cifras que no tienen cabida en el acumulador. El error de truncación es siempre positivo lo que hace que estos errores se acumulen en un proceso largo de cálculo pudiendo afectar considerablemente los resultados esperados. Para evitar esta acumulación, se introduce la operación llamada redondeo, que consiste en hacer que el error de truncación tome valores positivos y negativos el mismo número de veces en promedio, para de esta forma compensar la acumulación. La forma más habitual de hacer el redondeo, hablando en términos de sistema decimal, es sumar 5 a la primera cifra despreciada. La forma utilizada en Mercury, en sistema binario, es colocar siempre un 1 como la cifra menos significativa considerada. El error cometido al usar la operación de redondeo se llama error de redondeo.

A continuación damos el grupo de operaciones aritméticas que se ejecutan en el acumulador:

40	A' = L	
41	L' = A	
42	A' = A + L	con redondeo y
43	A' = A − L	normalización
44	A' = A + 1	con normalización y
45	A' = A − L	sin redondeo.
46	A' = A + L	sin normalización y
47	A' = A − L	sin redondeo.
50	A' = A L	con normalización y
51	A' = −A L	con redondeo
52	A' = A L	con normalización, y
53	A' = −A L	sin redondeo
54	A' = (AL) −	indicamos por (A L) − la parte menos
55	A' = (−AL) −	significativa del producto A L.

Ejemplo

– Dados los números largos a, b, c en los registros 1.0, 1.2, 1.4 respectivamente, calcular el valor del polinomio

$$a^2b^2 + b^2c^2 + c^2a^2$$

y almacenarlo en 1.4. Comenzar a escribir la secuencia de instrucciones en la página 8 de la memoria rápida

8.0	400	1.0	
8.1	500	1.0	
8.2	410	32	$a^2 \rightarrow 32$
8.3	400	1.2	
8.4	500	l.2	
8.5	410	34	
8.6	420	32	$A' = (a^2 + b^2)$
8.7	500	1.4	$A' = (a^2 + b^2)\ c$
8.8	500	1.4	$A' = (a^2 + b^2)\ c^2$
8.9	410	1.4	
8.10	400	32	$A' = a^2$
8.11	500	34	$A' = a^2b^2$
8.12	420	l.4	$A' = a^2b^2 + a^2c^2 + b^2c^2$
8.13	410	1.4	

Para ver en la forma en que opera el acumulador de Mercury en las operaciones aritméticas, pongamos algunos ejemplos en binario.

– Sumar $X_1 = 0.5$ y $X_2 = 0.75$. Su expresión en punto binario flotante seria:

X_1 =01000 00000 00000 00000 00000 00000, 00000 00000
X_2 =01100 00000 00000 00000 00000 00000, 00000 00000

$(X_1 + X_2)_R$ = 10100 00000 00000 00000 00000 00001, 00000 00000
$(X_1 + X_2)_S$ = 01010 00000 00000 00000 00000 00000, 00000 00001
$(X_1 + X_2)_{SR}$ = 01010 00000 00000 00000 00000 00001, 00000 00001

- Sumar X_1 =7 y X_2 =l.5.

Como no están en el intervalo normal tendríamos previamente que normalizarlos, teniendo en decimal:

X_1 = 0.875 , 3
X_2 = 0.75 , 1

o en binario:

X_1 =01110 00000 00000 00000 00000 00000, 00000 00011
X_2 =01100 00000 00000 00000 00000 00000, 00000 00001

$(X_1 + X_2)_R$ = 10001 00000 00000 00000 00000 00001, 00000 00011
$(X_1 + X_2)_R$ = 01000 10000 00000 00000 00000 00000, 00000 00100
$(X_1 + X_2)_{SR}$ = 01000 10000 00000 00000 00000 00001, 00000 00100

- Multiplicar X_1 =0.75 por X_2 =0,625.

X_1 = 01100 00000 00000 00000 00000 00000, 00000 00000
X_2 = 01010 00000 00000 00000 00000 00000, 00000 00000

$(X_1.X_2)_R$ = 00111 10000 00000 00000 00000 00001, 00000 0000
$(X_1\ X_2)_{SR}$ = 01111 00000 00000 00000 00000 00010, 11110 11111

Operaciones aritméticas con el exponente del acumulador. El exponente del acumulador esta directamente comunicado con los registros B, lo que permite que podemos hacer operaciones en el acumulador sin modificar la mantisa. El grupo de funciones con el exponente es formado análogamente, a las funciones que operan sobre los registros B, como se indica a continuación:

70 b	Bb' = E + n	Bt' sig (E + n)
71 b	E' = Bb	
72 b	Bb' = Bb + (E + n)	Bt' = sig Bb'
73 b	Bb' = Bb − (E + n)	Bt' = sig Bb'
74 b	Bb' = 1/2 Bb − (E + n)	Bt' = sig Bb'
75 b	Bb' = Bb \& (E − n)	Bt' = sig Bb'
76 b	Bb' = Bb # (E + n)	Bt' = sig Bb'
77 b		Bt' = sig (Bb − (E + n))
78 b	$A' = A + 0.2^n$	

Ejemplos:

– Dividir por 2^9 el acumulador

```
731 0
131 9
711 0
```

– Doblar el exponente del acumulador

```
707 0
327 0
717 0
```

Funciones de salto condicionado al acumulador e incondicional. Son las siguientes

48	c' = n	si desplazamiento ≥ 31
49	c' = n	si A ≥ 0
59	c' = n	

Ejemplos

– Reemplazar a por a supuesto almacenado en 2.0, y -l en 4.0,

```
1.0  400  2.1
1.1  490  1.3
1.2  500  4.0
1.3  410  2.0
```

– Dados a y b en los registros 2.0 y 2.2 colocar el menor en 4.0

```
3.0  400  2.0
3.1  430  2.2
3.2  490  3,5
3.3  400  2.0
3.4  590  3.6
3.5  400  2.2
3.6  410  4.01
```

Funciones de entrada, salida y transferencia entre memorias

A las funciones que permiten el acceso de la información externa a la memoria de la computadora las llamamos funciones de entrada. Dos son las formas de entrada: mediante el lector fotoeléctrico que lee caracteres de cinco bits perforados en una cinta de papel o mediante el registro F de consola en el que se puede escribir manualmente palabras de 10 bits. El primer método se utiliza para el ingreso de gran cantidad de información y el segundo para la entrada de alguna palabra clave.

Las expresiones de estas funciones son:

60	$H' = t_i$
61	H' = F

La función 60 coloca los cinco bits del carácter perforado que se encuentra debajo del lector fotoeléctrico en la parte menos significativa del registro H, pone en cero los cinco bits restantes, y hace avanzar la cinta un carácter, La función 61 coloca los 10 bits del registro F en el registro H.

De la misma forma para el egreso de información de la memoria como el único medio de salida es la perforación de un carácter de cinco bits sobre cinta de papel, tendremos las instrucciones según que los cinco bits que queremos perforar estén sobre la parte de dirección de la instrucción, o estén almacenados en un registro H para esos casos las funciones serían:

62 $t'_0 = n$
63 $t = H$

perforando los cinco bits menos significativos de n o de H.

Podemos considerar también como salida de información a la operación comandada por la función

64 (Display)' = L

con lo que se proyecta en el visor de consola, en las posiciones Display A y E, durante 120 μs el número largo almacenado en L. Como la imagen proyectada solo 120 μs no es visible, para conseguir la visibilidad será preciso ejecutar reiteradamente por un tiempo mayor dicha operación mediante un ciclo.

La transferencia de información entre las memorias se hace por bloques, equivalentes a una página de memoria rápida o a un sector de memoria grande. Para ejecutar una transferencia necesitamos, una instrucción previa que nos indique que sector es el que va intervenir, lo que se hace mediante la función

67 b n T' = n

que tiene por efecto almacenar en el registro T el entero contenido en la parte dirección operando.

Una vez conocido el sector, es preciso indicar la página y el sentido de la transferencia esto se logra mediante las dos siguientes funciones

68 b n Transferir el contenido del sector indicado en T a la página n.
69 b n Transferir al contenido de la página n al sector indicado en T.

Funciones especiales: Con la función

99

se para la secuencia de cálculo, la que se puede reiniciar únicamente operando normalmente en consola. Para los efectos de esta instrucción es indiferente el dígito B y la D. O. que aparezcan. A veces para controlar

nuestras paradas en el visor de consola, es conveniente incluir un número determinado como digito B.

La función

57

llamada función hueca, tiene por misión dejar pasar 60 μs sin hacer ninguna operación. Su utilidad es rellenar casillas impares que a veces no se utilizan en nuestra secuencia o impedir que se interrumpa nuestro proceso.

La función

58

hace emitir al altoparlante un sonido de 60 μs de duración, su finalidad es de control externo Como un sonido de 60 μs es inaudible es preciso ejecutar un ciclo en el que se incluya la función 58.

4.3 Los números. Como parte de una rutina se pueden incluir números en forma explicita (*), escritos directamente en base decimal, guardando algunas normas para su interpretación y conversión a binario por el traductor PIG 2. Los números serán de dos formas.

– números fraccionarios
– números enteros

Números fraccionarios. Se podrán escribir en forma de punto Lijo de la siguiente forma:

<signo> <parte entera><punto><parte fraccionaria>

o en punto flotante en la forma:

<signo><parte entera> <punto> <parte fraccionaria> <coma> <signo><exponente>

Se podrán omitir las partes innecesarias como la <parte entera> si carece de ella, el <punto> si carece de parte fraccionaria, pero nunca del <signo> de la Mantisa ya que este indica al PIG 2 que deberá almacenar el número que sigue, después de haberlo convertido a binario, en el próximo registro par de memoria e incluir una instrucción 570 0, si fuera necesario (**).

Ejemplos:

+14,42732
−3
+171,12
−1.427, −36
+.021

Estos números deberán estar comprendidos o el intervalo comprendido entre 2^−256 y 2^255, equivalente aproximadamente a 10^−70 y 10^70. La precisión de la conversión de números fraccionarios puede estar afectada de errores equivalentes a tres o cuatro veces del valor del digito binario menos significativo.

(*) Veremos mas adelante al tratar las rutinas Q como leer números no contenidos en el programa. Estos números deberán ser perforados separadamente sobre una cinta llamada cinta de datos, y so 1eeran durante el proceso de ejecución mientras que los indicados en esto parágrafo, son interpretados durante el proceso de traducción.

(**) Como vimos los números 1aros (40 bits) deben ir siempre ubicadas en registro par.

Números enteros. Para representar números enteros comprendidos en el rango (−512, 511) o (0, 1023) equivalentes a números de 10 bits bastara simplemente escribir el número en forma decimal precedido de uno de los siguientes signos: =, >, ≠. En el primor caso almacenará el equivalente binario del número decimal indicado, y en las otras el doble o la mitad respectivamente de dicho número

Ejemplos: Son equivalentes las siguientes expresiones.

= 4	>2	≠ 8	a 00000 00100
= −2	= 1022		
≠ 4	≠ 2044		a 1111111110
> −1	> 511		
= −1	= 1023		
≠ −2	≠ 2046		a 11111 11111
> 0+	>511+		

Para incluir una lista de números cortos, esta se podía representar colocando los números en columna, o separándolos por una coma (,).

Ejemplo.

Escribir la tabla de los números primos inferiores a 50.

= 1
= 3
= 5
= 7
= 11
= 13
= 17
= 19
= 23
= 29
= 31
= 37
= 41
= 43
= 47

o bien:

=1, = 3, = 5, = 7
=11, = l3, =17, =19
=23, =29
=31, =37
=41, =43, =47

En ambos casos, si la lista siguiera a una instrucción almacenada en el registros 12.3 los números primos dados anteriormente se alojarían en los registros cortos 12.4, 12.4+, 12.5, 12.5+, . . . , 12.11, 12.11+.

Si el número de enteros de la lista es impar, en el registro siguiente al último se almacenara automáticamente un 0.

5. – Las rutinas. Direcciones simbólicas

En el apartado anterior vimos las unidades elementales de información (instrucciones, números) en la forma en que son considerados por el sistema de programación convencional, a continuación veremos como tratar bloques de instrucciones y números.

Definimos como rutina a un conjunto de instrucciones identificables, y en el que rige un criterio para identificar simbólicamente cualquiera de sus instrucciones (*).
Generalmente una rutina tiene une finalidad operativa especifica.

Según que el punto de regreso, después de ejecutada una rutina, sea constante o este determinada por las condiciones de entrada a esa rutina se llamara abierta o cerrada respectivamente.

El sistema convencional considera tres tipos de rutinas, que llamamos:

v – rutinas
x – rutinas
Q – rutina

5.1 Las v-rutinas. Direcciones flotantes. Las v-rutinas, se las identifica encabezándolas con al letra R seguida de un número comprendido entre 1 y 999. El final de una rutina v lo determina el comienzo de la rutina v siguiente (**).

(*) Es decir que no es preciso conocer las direcciones absolutas do los registros en que las instrucciones o números son almacenados.

(**) El final de la última rutina v viene fijado por la directiva E, cuya función ya veremos mas adelante.

Ejemplo:

```
R  1
---------
---------
---------
R  5
---------
---------
---------
R  3
---------
---------
---------
E v 11
```

Las instrucciones, o los números de una rutina – v se los puede identificar mediante una marca. Esta marca consistirá en un paréntesis abierto [( ] seguido de un entero comprendido entre 1 y 100.

Ejemplos:

```
------------
411      64        (5
------------
------------
------------
102      0         (10
------------
------------
------------
=17(3, =193
------------
------------
------------
+ 1,42314         (7
------------
------------
------------
```

Utilizando marcas, no necesitamos conocer la dirección absoluta o dirección fija de la instrucción o número a que hacemos referencia en una instrucción mediante su parte de dirección, parte que sustituiremos por una v minúscula seguida del número

de la marca a que nos referimos.

Ejemplos:

590	v1	saltar a la instrucción marcada (1
180	v10	saltar a la instrucción marcada (10
400	v7	poner en el Ac el número marcado (7
101	v3	poner en B1 el número marcado (3

A las instrucciones o números de una misma rutina no se los puede asociar una misma marca. Como el número de instrucciones de una rutina puede ser mayor que el número de marcas distintas posibles, y para comodidad de podernos referir a alguna instrucción no marcada, apoyándonos en esta marca mediante direcciones simbólicas que llamaremos direcciones mixtas; para ello sustituiremos la parte de dirección de la instrucción por dos enteros separados por una v minúscula. El primer entero (que puede estar afectado del signo menos) llamado parte fija, indica cuantas posiciones hacia adelante o hacia atrás (signo menos) esta situado respecto de la marca indicada por el segundo entero llamado parte flotante.

Ejemplos:

1̃. Dada la rutina siguiente

1.0	300	1	(11
1.1	210	7.7	
1.2	580	0	(5
1.3	074	7.1+	

son equivalentes como parte de dirección las siguientes expresiones

Dirección fija	Dirección flotante	Dirección mixta
1.0	v 11	−2 v 5
1.1		1 v 11
1.1		−1 v 5
1.2	v 5	2 v 11
1.3		3 v 11
1.3		1 v 5

2. – Dada la siguiente tabla de números

```
=1 (1   ,      =3
=5      ,      =7
=11     ,      =13
=17     ,      =19 (8
=23     ,      =29
=31     ,      =37
=41     ,      =43
=47     ,      =49(16
```

La dirección del número 11, puede estar referida como 2v1 o −2+v8, la del 43 como 3v8 o −1v15 y la del 31 como 1+v8 o −3+v15.

3. – Si los números son largos nos referimos de la siguiente forma

```
+ 1.34178
− 2.73215     (1
+ 7.3241, −2
+ 1.4596
```

con −2v1 indicamos al número 1.34178
con v1 al número − 2.73215
con 2v1 al número 0.073241 y
con 4v1 al número 1.4596

5.2 Las x-rutinas. Direcciones relativas.

La identificación de las rutinas x se hace encabezándolas con la letra R seguida de un número mayor o igual que mil, este número puede tener parte entera y parte fraccionaria. Para la ejecución de la rutina x esta debe estar siempre incluida en una rutina v.

Ejemplo

```
R 1
---------------
R   1020.1
---------------
620     0
620     30
620     13
620     30
015     0+
```

La finalidad de las rutinas x, es facilitar la formación y utilización de la biblioteca de programas (*). Es importante que las direcciones de una rutina de biblioteca no sean fijas, ya que estas rutinas se utilizaran en cualquier punto de un programa.

Para facilitar esto, las rutinas x utilizan las direcciones simbólicas llamadas direcciones relativas, es decir que la dirección de cualquier instrucción esta referida a la primera instrucción de la rutina, a la que nos referimos como dirección x, las sucesiva instrucciones diremos que están ubicadas en las direcciones 1x, 2x, 3x,..., si nos referimos a los registros de la derecha las direcciones respectivas serian 1+x, 2+x, 3+x, ...

Ejemplo

Dirección Relativa	R 1050.1	
x	300	−89
1x	620	0
2x	380	1x
3x	014	41+x
4x	016	39+x
5x	300	−9
6x	620	0
7x	380	6x

.
.
.

5.3 Las q-rutinas

El programa traductor PIG 2, contiene, como parte de el, una pequeña biblioteca con rutinas de uso mas frecuente (lectura y conversión de números decimales a binario, ídem para la impresión inversa, raíz cuadrada, funciones transcendentes elementales, etc. (**))

Estas rutinas se llaman rutinas q o quickies, y para designarlas en el programa se emplea la letra Q seguida de un entero comprendido entre 1-19. El programa traductor incluye el bloque de instrucciones correspondientes a dicho quickie en el programa cuando es requerido.

Ejemplo:

```
010     8.3+
Q1
210     8.0
```

que seria equivalente a la siguiente rutina.

(*) Llamaré así a la colección de subprogramas numerado y acompañado de su modo de empleo, en particular una colección sistemática se sub-programas probadas destinadas a un modelo particular de computadora y puesta a la disposición de los usuarios.

(**) Ver apéndice 1 el detalle de los distintos quickies.

```
010     8.3+

707     −259
300     1
090     0
210     2
410     32
410     34
230     32
210     34
300     749
230     33+
490     v1
330     475
210     35+     (1      Q1
300     −1
510     34
430     2       (2
500     34
410     34
510     32
380     v2
010     2
450     2
500     34
420     34

210     8.0
```

Esta forma de llamar (*) al quickie corresponde a considerarle como una rutina abierta, y tiene el inconveniente que si se necesita usar varias veces el mismo quickie, es preciso incluir todas sus instrucciones cada vez, con el consiguiente aumento de longitud de nuestro programa. Para salvar este inconveniente existe otra forma de llamar a los quickies como rutinas cerradas, que consiste en considerarlas marcadas con marcas comprendidas entre 101 y 119 (Q1 a Q19), y hacer referencia a estas marcas mediante las direcciones simbólicas v101 a v119 después de haber almacenado en BI la dirección donde hay que saltar a la salida. De esta forma aunque se hagan varias referencias en un mismo capitulo a un determinado quickie este se incluirá una sola vez al final del capitulo, agregando a las instrucciones del quickie la instrucción 591 0 (salvo el Q9 que incluye 591 3).

(*) Incluir, hacer formar parte.

Si hubiere referencias a distintos quickie estos se incluirán al final en el orden en que aparecen las referencias.

Ejemplo:

```
010  8.3+
101  v101
210  8.0  (5
```

6. – Programas - Directivas

Se define en general como programa, al conjunto de instrucciones, de expresiones y eventualmente de cualquier otro dato, reunidos con el objeto de ordenar la marcha de una computadora.

En el sistema convencional un programa esta compuesto por capítulos y los capítulos por rutinas.

Para la organización del proceso de traducción de un programa, existen unidades de información llamadas directivas (*). Las directivas se ejecutan durante el proceso de traducción, pero no se almacenan en el programa traducido (a veces su ejecución consiste en la inclusión de algunas instrucciones, o de una rutina entera como en el caso de las Q).

6.1 Capítulos

Como generalmente un programa no cabe íntegramente en la memoria rápida, es preciso fraccionarlo en partes que quepan en ella, a estas partes llamamos Capítulos. Los Capítulos los almacenamos permanentemente en la memoria grande, llevando uno por vez a la memoria rápida para su ejecución.

El comienzo de un Capitulo se indica mediante la directiva CAPITULO (o simplemente C) seguido de un número comprendido entre 1 y 100, llamado número de Capitulo. Cada Capitulo debe tener un número diferente.

El final de un Capitulo esta indicado por el comienzo de otro Capitulo o, si se trata del último, por la directiva E.

Cuando se comienza la traducción de un Capitulo, aquella se empieza a almacenar (si no hay indicación de lo contrario) en el primer registro de la pagina 1. Cuando se termina de traducir un Capitulo, todo el Capitulo traducido se almacena en memoria a partir del primer registro del sector siguiente al último utilizado para el almacenamiento del Capitulo anterior. Si fuera el primer Capitulo traducido se almacenaría (si no hay indicación de lo contrario) a partir del sector 129.

(*) Ya hemos visto las directivas R y Q.

Toda instrucción de un capitulo debe de estar incluida en alguna de las v-rutinas del capitulo.

Ejemplo.

Puede ser el comienzo de un capitulo la siguiente secuencia:

```
C1              CAPITULO 1
R1              RUTINA 1
996   0         996     0
620   30        620     30
620   13        620     13
620   13        620     13
620   7.0+      620     7.0+
```

6.2 Entrar

Para comenzar la ejecución de un programa, es decir para indicar la separación del proceso de traducción y el de ejecución existe la directiva E o ENTRAR seguida de la dirección del punto en que queremos comenzar la ejecución.

Ejemplo: La directiva

```
E v1 /3
```

hace transferir a la memoria rápida el capitulo en el que esta la Rutina 3, y comienza a ejecutar en la instrucción marcada con 1.

6.3 Salto entre capítulos.

Como saltar de una instrucción de un capitulo a otra de un capitulo diferente, involucra la transferencia del segundo capitulo a la memoria rápida, este salto no se puede realizar mediante una instrucción de salto. Para ello se dispone de tres directivas de salto entre capítulos: across, down, up.

Si el salto es a una rutina abierta utilizamos la directiva ACROSS o A, seguida de la dirección del punto a donde queremos saltar.

Ejemplo: En la siguiente secuencia

```
330   1
210   7.1+
A     v/3
+0.333333    (1
+0.166667    (2
```

la directiva A v/3 tiene por efecto traer a la memoria rápida el capitulo que incluye a la rutina 3 y comenzar a ejecutar en su primera instrucción.

Si el salto es a una rutina cerrada incluida en otro capitulo, esto se hará mediante la directiva DOWN o D seguida de la dirección del punto de entrada de la rutina a que vamos a saltar; la ultima instrucción de esta rutina deberá ser seguida por la directiva UP o U, que ordena un salto a la instrucción siguiente al DOWN de que proviene. El capitulo en donde aparece el UP se llama subcapítulo. Un subcapítulo puede tener a su vez otro subcapítulo, que será un sub-sub-capítulo o subcapítulo de orden dos. Es posible la existencia de subcapítulos hasta de orden 9.

Ejemplos:

```
A  5−0/1
D  x1v1/2
D  n1v2/15
```

6.4 Ubicación especial en las memorias.

Vimos que mientras no se indicara lo contrario los capítulos se almacenaban en memoria rápida a partir de la pagina 1, y en la grande a partir del sector 129, o en sectores consecutivos a partir del último utilizado por el capitulo anterior. Igualmente cada instrucción o número se almacena en la memoria rápida en el registro siguiente a la instrucción o número anterior.

Pero si por la naturaleza especial de nuestro programa deseamos modificar esta ubicación habitual, tenemos la facilidad de utilizar las siguientes directivas: F, P, S, L, M.

F

Utilizamos la directiva F (Firstsector) seguida de un entero **α** para indicar al programa traductor (PIG2) que queremos almacenar nuestro programa traducido a partir del sector **α**.

Ejemplo:

```
F 320
C 1
R 1
993 0
----------
----------
```

lo que implica que nuestro programa en lugar de almacenarse a partir del sector 129, lo hace a partir del 321 (*)

(*) Reserva el sector 320 para almacenar el titulo como veremos mas adelante

P

Si se quiere comenzar un capitulo en pagina distinta de la 1, utilizamos la directiva P o PAGINA seguida del número de la pagina en que queremos comenzar. Es conveniente utilizar esta directiva cuando el capitulo a que se aplica es de mucho uso, evitando que cada vez que se ejecute sea preciso hacer transferencias entre las memorias, pudiendo utilizar simplemente instrucciones de salto.

```
F 320
C 1    P 6
R 1
993          0
--------
--------
--------
```

con estas indicaciones almacena el capitulo 1 en la pagina 6, de la memoria rápida, pero su ubicación en la memoria grande no será alterada, correspondiendo en el ejemplo con el sector 321.

Si queremos fijar extensión máxima al capitulo, completamos la directiva P haciendo seguir al numero de primera pagina del capitulo el numero de la ultima pagina, separando ambos números por un guión.

Ejemplo:

```
F 320
C 1    P 6-12
R 1
993 0
--------
--------
--------
```

lo que evita invadir la pagina 13, que podría estar destinada por nuestro programa a otra finalidad (lugar de trabajo, rutina especial, ...)

S

Si se precisa que un determinado capitulo se almacene en un sector distinto al que le correspondería normalmente, (es decir al siguiente del último sector utilizado por el capitulo precedente) se utilizará la directiva S, o SECTOR, seguida del número de sector que deseamos, escritos en la misma línea que la directiva C, y antes o después, indistintamente, que la directiva P.

Ejemplo:

```
F 320
C 1    P 6-12   S 400
R 1
993 0
-----------
-----------
-----------
```

L

La directiva L, o LINEA, tiene por finalidad almacenar la secuencia de instrucciones que siguen a partir de la dirección que se escriba a continuación de la L. Esta dirección puede ser fija o flotante. La directiva LINEA puede ser intercalada en cualquier punto de un capitulo.

Ejemplo:

```
F      320
C 1   P6-12   S 400
L 6.0
R 1
993   0
-----------
-----------
-----------
```

o bien

```
F 320
C 1   P6-12   S 400
R 1
L 6.0
993 0
-----------
-----------
-----------
```

Como control de la ubicación definitiva de los capítulos y rutinas, tenemos la facilidad de la directiva * (asterisco), la que colocada al comienzo del programa, en una nueva línea, (antes que cualquier otra información) tiene por efecto imprimir el número del primer sector y pagina en donde se almacena cada capitulo, precedidos por la letra C y el número del capitulo y la dirección de la primera instrucción de cada rutina, precedida por la letra R y el número de rutina.

Ejemplo:

Un programa que comenzase:

```
*
F 320 C 1   P6-12   S 400
R 1
-----------
-----------
-----------
R 2
-----------
-----------
-----------
```

imprimiria durante la entrada:

```
C 1
S 400    P6
R 1          6.0
R 2          8.6
```

M

Para poder corregir alguna instrucción, o conjunto de instrucciones, o incluir alguna constante o instrucción, sin necesidad de modificar la cinta del programa, existe la directiva M (M - corrección). La letra M seguida de una dirección cruzada (análoga a las direcciones que aparecen en las directivas A y D), tiene por efecto traer a la memoria rápida el capitulo donde esta incluida la rutina que queremos modificar y a partir de la dirección indicada incluir las instrucciones que siguen hasta que aparece otra M - corrección, o la directiva E.

Ejemplo:

```
F 320
C 1    P6-12   S 400
R 1
990          0
-----------
-----------
-----------
M v/1
993          0
E v/1
```

sustituye la instrucción 990 0 por la 993 0.

6.5 Directivas especiales.

Por último, trataremos de algunas directiva, que sin ser esenciales para la redacción de un programa, nos permiten algunas facilidades adicionales para la operación o interpretación de los resultados; estas directivas son: T, N, ?, W.

T

Es conveniente para identificar un programa, asignarlo a su comienzo un nombre; esto se consigue en el sistema Convencional, mediante la directiva T o TITULO, que tiene por efecto almacenar en memoria grande (en el sector 128 en los casos que no se utiliza la directiva F, y en el número que sigue a dicha letra en caso contrario)

y perforar sobre la cinta de resultados, todos los caracteres comprendidos entre el primer FS posterior a la directiva T, y dos o mas FS consecutivos, siempre que el total de caracteres no exceda de 244. Si se necesita un número de caracteres mayor, se podrían emplear varias directivas T, aunque en este caso solo los caracteres correspondientes a la ultima T serian almacenados, si bien todos perforados durante la entrada. La directiva T deberá ponerse siempre delante de la primera directiva C que contenga el programa.

Ejemplo:

```
F 320
T CALCULO DE LA ORBITA DE UN SATELITE
POR
JUAN PEREZ
12. 11. 62 FS FS  (*)
C 1    P6-12   S 400
R 1
L 6.0
993   0
----------
----------
----------
```

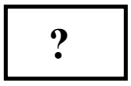
N

La directiva NOMBRE o N, tiene solo efecto si se utiliza la directiva *, y su misión es perforar en la cinta de resultados durante la entrada los caracteres que siguen hasta el primer CR. Las rutinas de Biblioteca están encabezadas para su identificación por un titulo bajo la directiva N.

W

Para interrumpir temporalmente la ejecución del programa traductor, se utiliza la directiva W (WAIT), la que detiene el proceso de traducción y emite un sonido intermitente. Para continuar basta con bajar la llave 9 del clavijero inferior de la consola. Es útil colocar una W antes de la directiva E, para permitir la utilización de la M-corrección, si fuera preciso, sin modificar la cinta del programa.

(*) Como los caracteres FS no aparecen registrados en el pliego de la teleimpresora es conveniente verificar que se encuentren perforadas en la cinta. Si esto no ocurriera interpretaría como titulo el comienzo del programa.



Durante la puesta a punto de un programa, puede ser útil imprimir en diversos instantes el contenido de alguno de los registros B o del Acumulador. Para ello se emplea la directiva ? que seguida del número 1, 2, 3, ... imprime B1, B2, B3, ...del número 9 imprime el acumulador como cuatro enteros cortos, y seguido de 10 imprime el contenido de todos los registros B y del acumulador.

Ejemplo:

```
022     6.0+
?2
012     8.0+
```

imprimiría el contenido de B2.