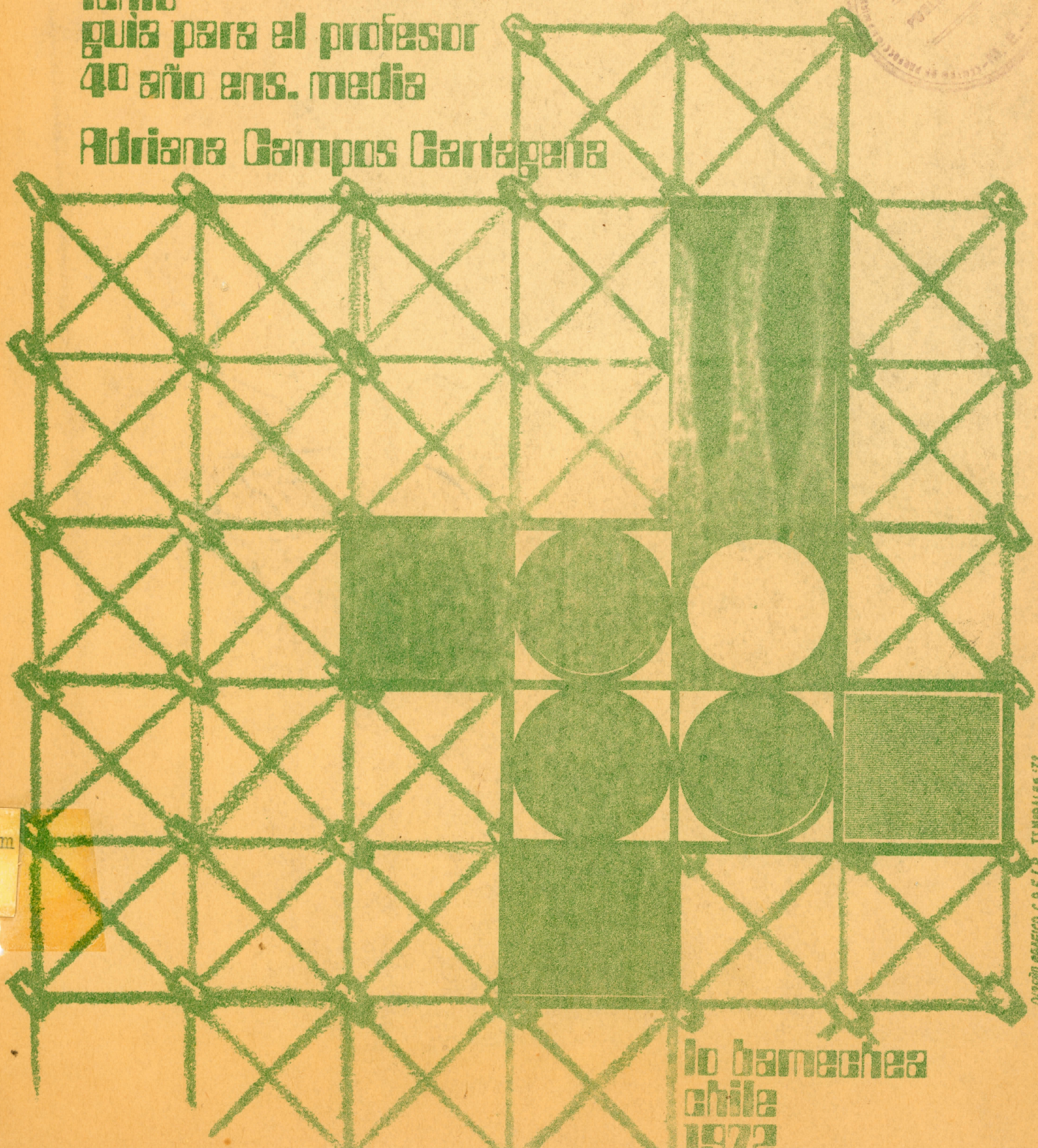
introducción a la computación electrónica

texto
guía para el profesor
4º año ens. media

Adriana Campos Cartagena



DISEÑO GRAFICO C.P.E.I.P. J.I.NIÑALES '72

lo barnechea
chile
1972

M I N I S T E R I O D E E D U C A C I O N
CENTRO DE PERFECCIONAMIENTO, EXPERIMENTACION
E INVESTIGACIONES PEDAGOGICAS

DEPARTAMENTO DE MATEMATICA

PROF. ADRIANA CAMPOS CARTAGENA

9172.

introducción a la computación electrónica

texto guía para el profesor
4º año ens. media

P R E S E N T A C I O N

La importancia de las Técnicas de Computación electrónica son suficientemente valoradas en el medio ambiente actual.- No cabe por lo tanto exaltar sus méritos, ni hacer énfasis en la utilidad que prestan al progreso en general y en especial a la investigación educacional y científica .

La Educación no puede estar ajena a los avances de la Tecnología y de las Ciencias y nos parece lógico que su proyección hacia un futuro próximo le haga planificar sobre nuevas técnicas de aprendizaje, que no tienen otro objetivo que el de mantener gente preparada para estar siempre en las posiciones de vanguardia, y para formar en último término al hombre del futuro el que impulsará la industrialización de nuestro medio y velará por la producción y la independencia efectiva de nuestro país .

La Reforma Educacional pretendió cumplir con estos objetivos y todas las áreas de la Educación recibieron su influencia rejuvenecedora. Es evidente que a una reforma deben seguir otros impulsos de reforma puesto que la posición reformista es como la vida misma .

La presentación de este nuevo texto guía es para nosotros de gran valor, y estimamos en lo que vale, el esfuerzo de la profesora Adriana Campos Cartagena, quién está contribuyendo positivamente a dar este impulso renovador a nuestra enseñanza. La existencia de textos de Computación es limitada en nuestro ambiente y la mayor parte de ellos son tratados en idioma inglés y están orientados en forma técnica. El texto guía que ahora presentamos, está orientado directamente a la enseñanza de esta Técnica y pretende dar las sugerencias metodológicas para que los profesores inicien cuanto antes su docencia; en la creencia que para todos será beneficioso. No esperemos estar totalmente preparados para empezar. No tengamos la mentalidad perfeccionista de hacer las cosas tan bien hechas que pasamos lo mejor de nuestra vida buscando modelos acabados. Nuestros países subdesarrollados se tienen que modelar en los cambios y durante los cambios; de otro modo no iremos al ritmo de la velocidad moderna. Los países que hoy son capaces

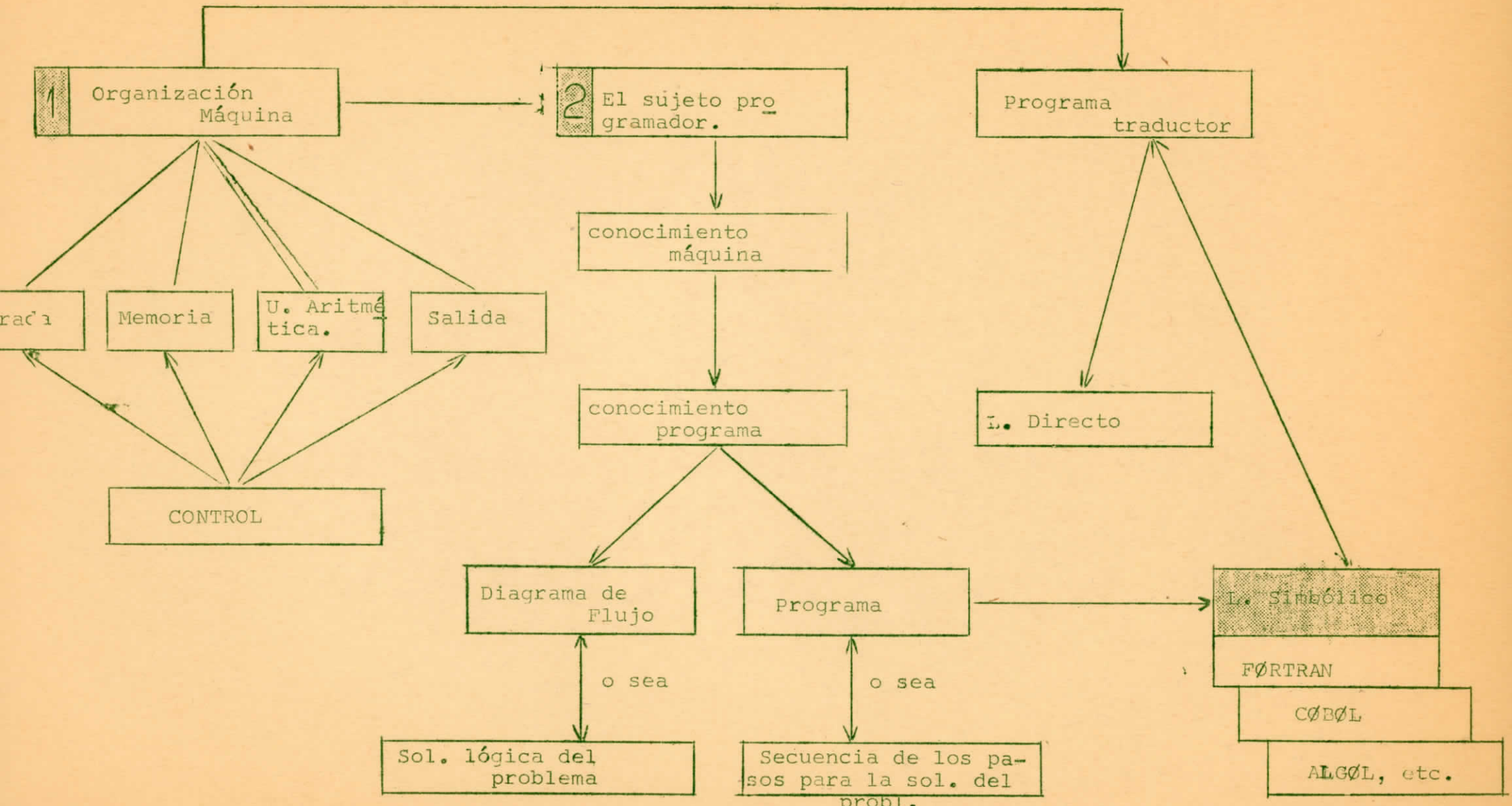
de construir sus propias máquinas electrónicas, un día no las tuvieron, pero empezaron por construir malas máquinas y las perfeccionaron luego. Los países de América que también han logrado construir sus máquinas empezaron por informarse bien, aprender a usarlas y luego fueron capaces de construirlas; y mañana, las tendrán mejores. Creo que en la mentalidad actual de los colegas esto es un lugar común y no hay que enfatizarlo más.

Ojalá que este texto sirva de ayuda para impulsar esta enseñanza y que los estimados colegas nos hagan llegar cualquier sugerencia en referencia al texto; lo que consideramos en beneficio de la autora y de las labores de este Centro de Perfeccionamiento .

MARIO LEYTON SOTO
Director .

Lo Barnechea , Agosto de 1972 .-

ESQUEMA DE TRABAJO



EXPLICACION DEL ESQUEMA

El esquema de trabajo es un buen indicador de los pasos a seguir en la enseñanza del ramo por una parte y si este esquema no se sigue tiene por lo menos la ventaja de permitir de un vistazo tener un panorama de los aspectos esenciales de esta enseñanza.

1) Organización de la máquina :

Entrada
Memoria
U. Aritmética
Salida, todas regidas por la Unidad de Control que permite la pasada de una sección a otra durante la ejecución de un programa .

Es parte de la estructura de la máquina el Programa traductor o Compilador que ella tiene, y que le permite trabajar con distintos lenguajes simbólicos, como son FORTRAN, COBOL u otros , traduciéndolos a Lenguaje Directo o de Máquina, que tiene mayor relación con los dispositivos electrónicos de la máquina y el sistema de codificación usado .

2) El sujeto programador, tiene necesidad de conocer por lo menos en forma teórica, la estructura y organización de la máquina . Pero principalmente tiene necesidad de:

a) dominio absoluto de los aspectos matemáticos que entran en el problema.
b) conocimiento de la técnica de diagrama de flujo que permite hacer una descripción de procedimientos para la solución del problema, atendiendo a la inclusión de todas sus alternativas.
c) conocimiento del lenguaje simbólico que permite la codificación del diagrama de flujo,por un vaciamiento paso a paso de los procedimientos y que es el objetivo principal de estos apuntes.

Naturalmente para alcanzar este objetivo hay que recorrer un camino relativamente largo, y existe niveles de profundidad de estas materias. Me ha parecido que tomando lo estrictamente necesario

para la comprensión de la solución del problema desde su entrada en la computadora hasta la impresión de la solución del mismo, permite que el profesor comprenda la utilidad de esta materia para la aplicación de la matematica y el gran beneficio que alcanzará nuestra educación al enriquecer la cultura del educando con una técnica que en la vida común ya va resultando ineludible .

La profundización de las materias por parte de cada profesor será un fruto de la claridad con que haya captado lo esencial de esta técnica y la motivación personal por llegar a un grado de dominio de los temas que aquí se esbozaron .

De acuerdo con el esquema revisado existen dos aspectos que se deben abordar y que pretendo esclarecer: 1) el conocimiento de la máquina y 2) el uso de la máquina a través de sencillas técnicas de programación .

Interesa el conocimiento de la máquina no con un espíritu de tecnismo para dominar la estructura interna de la misma. Esto corresponde a quienes poseen los conocimientos de electrónica y están capacitados para construir y perfeccionar los circuitos y sus posibles combinaciones en orden a su operacionalidad . Me parece que el profesor necesita un conocimiento global de la máquina; un reconocimiento de los aspectos utilitarios de ellas; para aprovecharlos didácticamente y para saber dirigir el interés de sus alumnos por estos aspectos y quizá más tarde orientarlos a estudios más profundos .

LA MAQUINA.

Exteriormente al entrar a un Centro de Computación se encuentra el observador con muebles; armarios metálicos provistos de muchos botones y luces verdes y rojas, discos que giran, pliegos de papel que se deslizan de una máquina a otra etc.

Es interesante reconocer las partes principales de un equipo computador . Debe existir una Unidad de Entrada (lectora), una Unidad de Salida (impresora), la Unidad de Aritmética o procesamiento y la Memoria fundamentalmente.

La Memoria, es la unidad más importante y también llamada Unidad Central . La comprensión, de la función que corresponde a ésta unidad es clave para entender el proceso de la programación aún en los niveles más elementales.
La Memoria es un centro priviligiado.
La Memoria es un lugar de almacenaje al que se puede recurrir constantemente para retirar material que anteriormente se depositó o para guardarlo nuevamente después de reelaborado .

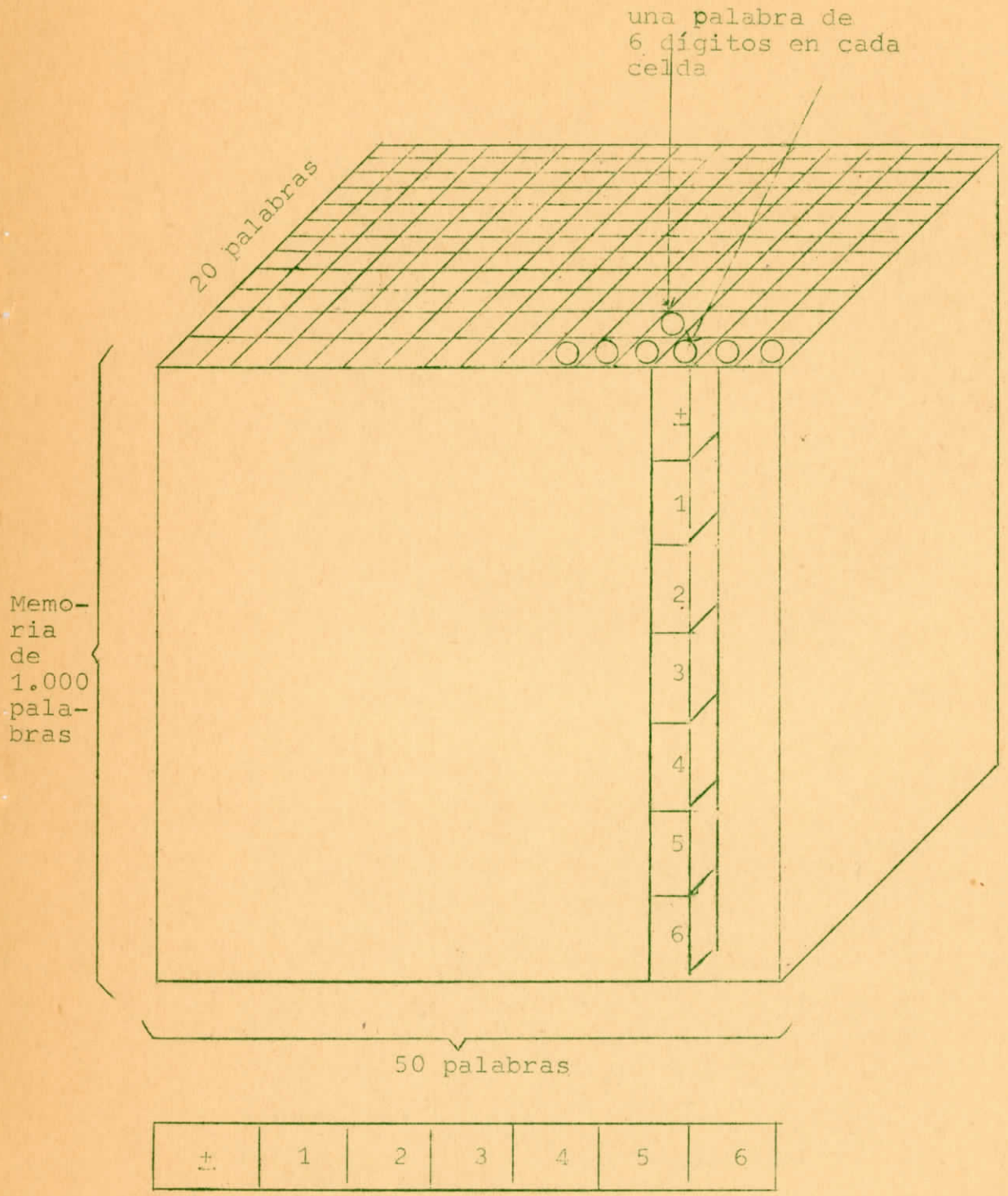
En Memoria se guardan los datos de un problema. por ejemplo y se van retirando en cierta secuencia, a medida que se necesite utilizarlos . Y en ella se pueden guardar los resultados parciales o el resultado total que corresponde a la solución del problema.

En Memoria es posible guardar programas que corresponden a aspectos variables de otro programa y que pueden ser utilizados tantas veces como se requiera para la solución de problemas más complejos .

Como se encuentra almacenado el material en memoria ?
Que disposición material tiene la memoria ?
Qué es la memoria ?

Aún a riesgo de perder todo sentido poético, diré que la memoria es un mueble. En este mueble con multitud de casilleros vitalizados en forma eléctrica, llegan y salen palabras codificadas, pudiendo retenerse encasillada abundante cantidad de información, de ahí su denominación antropomorfica de Memoria.

Ver dibujo página siguiente.


una palabra de
6 dígitos en cada
celda
20 palabras
Memo-
ria
de
1.000
pala-
bras
±
1
2
3
4
5
6
50 palabras

±	1	2	3	4	5	6

Me detendré brevemente para examinar la forma en que la Memoria puede guardar información. Siendo un aparato electrónico se vió primeramente la necesidad de utilizar en ella un sistema binario de numeración. Esto permitía usar las dos posibilidades físicas de luz y no luz asociadas a los dos únicos símbolos numéricos 0 y 1 del sistema binario, al que se puede traducir cualquier número del sistema decimal o décuplo. De este modo el problema numérico quedó solucionado y convencionalmente la mayoría de los constructores de máquinas electrónicas aceptaron el sistema binario. El segundo paso dirigido a registrar eléctrónicamente palabras del idioma hablado (inglés) se hizo codificando el abecedario .

Para tratar de seguir en forma elemental el trabajo interno de la máquina y para comprender mejor lo que ocurre en Memoria, es necesario tener una idea clara de esta aplicación del sistema binario y la forma usual de codificación . Es por todos conocidos el valor posicional de los dígitos en el sistema numérico decimal.

Sistema déclupo o decimal (posibilidad de escritura con 10 símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

10^3	10^2	10^1	10^0
4	5	3	0
2	0	0	8

Ejemplo

$4 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 0 \cdot 10^0 =$

$4.000 + 500 + 30 + 0 = 4.530$

Ejemplo

$2 \times 10^3 + 0 \cdot 10^2 + 0 \cdot 10^1 + 8 \cdot 10^0 =$

$2000 + 8 = 2.008$

Ejercicio de aritmética en página 18 y siguiente.

Sistema binario (posibilidad de escritura con 2 símbolos 0,1)

2^3	2^2	2^1	2^0
			1
		1	0
		1	1
	1	0	0

Ej. $1 \times 2^0 = 1 \times 1 = 1$

Ej. $1 \times 2^1 + 0 \times 2^0 =$

$2 + 0 = 2$

Ej. $1 \times 2^1 + 1 \times 2^0 =$

$2 + 1 = 3$

Ej. $1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 =$

$4 + 0 + 0 = 4$

De modo que se ha obtenido la tabla de valores que sigue (para los dígitos) .

decimal		binario
0	=	0
1	=	1
2	=	10
3	=	11
4	=	100
5	=	101
6	=	110
7	=	111
8	=	1000
9	=	1001

Para dar más homogeneidad a la escritura en binarios asignemos cuatro bits (un bit es siempre 1 ó 0) para expresar un dígito .

0	=	0000
1	=	0001
2	=	0010
3	=	0011
4	=	0100
5	=	0101
6	=	0110
7	=	0111
8	=	1000
9	=	1001

Números no dígitos es decir superiores a 9 se codifican según estos valores .

Aún cuando el auténtico valor de 13 en base 2 es 1101

Ejemplo

1	3
0001	0011

13 se codifica como 0001 0011 .

Lo que ocurre es que para números grandes se hace cada vez más difícil el cálculo y para evitarlos se codifica.

El acuerdo para codificar en binario cualquier número entero (de varios dígitos), es lo que se ha designado como Codificación Binaria Decimal (CBD) que tiende a evitar cálculos para codificar números de un sistema a otro.

Si se trata de codificar el número 42 en binario se usará el sistema de Codificación Binaria Decimal, dando el valor de cada dígito separadamente, es decir:

4	2
0100	0010

de este modo 42 queda codificado en binario usando CBD, y sin necesidad de cálculo alguno.

Otros ejemplos:

62 ------> 0110 0010

1325 ------> 0001 0011 0010 0101
(1 3 2 5)

Inversamente 1001 1000 0011 1000 0001 0101 corresponde a una codificación binaria del número: 9 8 3 8 1 5

Esto no quiere decir que la interpretación decimal en cuanto al valor posicional de cada 1 y cada 0, del número binario corresponde a 983815. Evidentemente esto no ocurre. El número 983815 está codificado (CBD) como 1001 1000 0011 1000 0001 0101 (5) y descodificando - de CBD como: 983815

Veamos cual es la utilidad de estos convenios. La utilidad se concreta en que es posible codificar también en binario, el alfabeto, disponiendo así de números y letras que pueden interpretarse en una máquina como 1 ó 0; como luz o no luz, o con otras expresiones electrónicas.

Codificación del alfabeto.

El abecedario se divide en tres grupos de letras en distinto nivel o zona.

A B C D E F G H I } zona 11
1 2 3 4 5 6 7 8 9 }

J K L M N O P Q R } zona 10
1 2 3 4 5 6 7 8 9 }

S T U V W X Y Z } zona 01
2 3 4 5 6 7 8 9 }

A cada letra se hace corresponder un número dígito así la A corresponde 1
P corresponde 7
S corresponde 2

(en forma convencional se omitió 1 en el último grupo y también la letra Ñ en el grupo anterior .)

A cada grupo se le asigna una codificación de ZONA por ejemplo.

al 1er. grupo (A - I) zona 11
al 2º grupo (J - R) zona 10
al 3er. grupo (S - Z) zona 01

(IBM llama ZONA 12 al 1º
ZONA 11 al 2º
ZONA 0 al 3º).

De este modo podríamos codificar cualquier frase , por ejemplo :

E	L		H	O	M	B	R	E		N	A	C	I	O
11	10		11	10	10	11	10	11		10	11	11	11	10
5	3		8	6	4	2	9	5		5	1	3	9	6

P	A	R	A		S	E	R		F	E	L	I	Z
10	11	10	11		01	11	10		11	11	10	11	10
7	1	9	1		2	5	9		6	5	3	9	9

Ahora este primer intento de código tiene una mezcla de binario y decimal. Podemos mantener los binarios que corresponden a las zonas o grupos numéricos y codificar los dígitos en CBD. De este modo un nombre como MARIA lo codificaríamos así :

M	A	R	I	A	
1	1	1	1	1	zona
0	1	0	1	1	
=	=	=	=	=	
0	0	1	1	0	
1	0	0	0	0	CBD
0	0	0	0	0	
0	1	1	1	1	

M	A	R	I	A
10	11	10	11	11
4	1	9	9	1

Otra forma sería escribirlo en forma horizontal

	M		A		R		I		A	
Z		Z		Z		Z		Z		
1 0	0 1 0 0	1 1	0 0 0 1	1 0	1 0 0 1	1 1	1 0 0 1	1 1	0 0 0 1	

1 byte tiene 6 bits.

La palabra María se expresa con 6 bytes de 6 bits o sea con 36 bits entendiéndose por un bit, la menor cantidad de información o sea 1 ó 0 .

El número 3247 se expresa con 16 bites es decir no hay que tomar en cuenta zonas de modo que se tiene :

3	2	4	7
0	0	0	0
0	0	1	1
1	1	0	1
1	0	0	1

} CBD

Se usan bytes de 4 bits. En **total** 4 bytes.

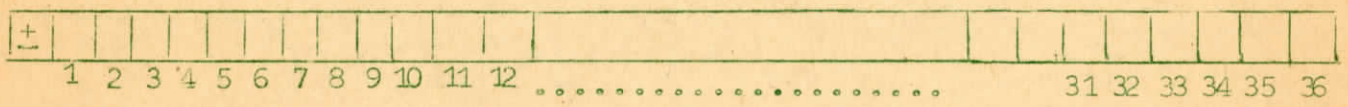
En forma horizontal

3			2			4			7						
0	0	1	1	0	0	1	0	0	1	0	0	0	1	1	1

16 bits.

Para obtener cierta homogeneidad de expresión se usan siempre palabras alfaméricas o alfanuméricas es decir nombres que pueden contener letras o números . Y se fija la longitud de palabra teniendo en cuenta los dígitos,por ejemplo el programa de 4º de Enseñanza Media se refiere a un **computador hipotético** con Memoria de 1000 palabras de 6 dígitos decimales. Esto quiere decir que para guardar todas las posibilidades (palabra alfabética palabra numérica tomando en cuenta el signo del número en cuestión deberá tomarse una extensión de 37 bits, ya que si en el lugar de los dígitos hubiera letras, se requeriría **para cada** letra 1 byte = 6 bites. En total la máxima extensión para 6 letras es de 36 bites y un lugar reservado para el signo de la palabra : son 37 bites.

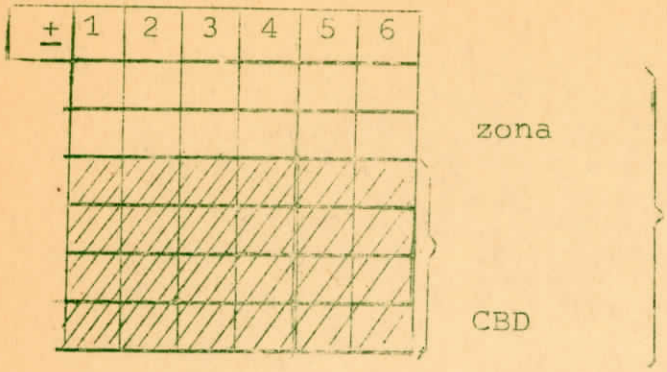
Ejemplo :



37 bites.

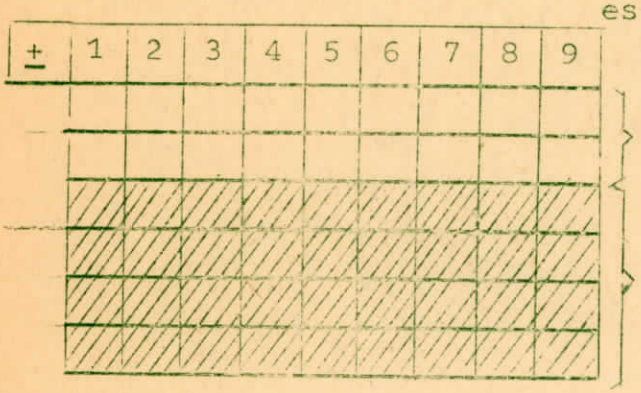
No se está tomando en cuenta la codificación del signo 0 para + ; 1 para - .

Expresado en otra forma



Prácticamente el signo se codifica como 0 para (+) y 1 para (-); por lo tanto esta explicación es de tipo didáctico.
De Hecho la palabra ocuparía 6x(6 + 1) = 42 bits.

Si en otra máquina requirieran palabras con 9 dígitos serían necesarios 55 bites 9 x 6 + 1 . "Considerando la observación anterior esta palabra ocuparía 60 bits. "



Actividades sugeridas

1) Revisión de la información que tienen los alumnos sobre sistema binario .
2) Trabajo en grupos con el objetivo de que los alumnos tengan dominio del sistema binario .
3) Solución de problemas sencillos con la aritmética binaria.
4) Algún trabajo de información sobre el uso de otros sistemas numéricos en máquinas computadoras de generación anterior o bien más moderna .
5) Trabajo en grupo para codificar un mensaje .
6) Cambio de mensajes. Trabajo en grupo para descodificar los mensajes.
7) Sugerencias de códigos inventados por los alumnos.

Con el objeto de tener mayor comprensión de lo que ocurre en Memoria vamos a imaginarla momentaneamente como en tablero en el que calzan exactamente cierto número de regletas todas de 37 bits es decir todas con posibilidad de contener palabras

Tratándose de un computador de 1000 palabras de 6 dígitos, Tendríamos que imaginar 1000 regletas como las descritas en el párrafo anterior.

Inmediatamente surge la necesidad de dar cierto órden en Memoria para recibir las regletas. Ellas, las regletas serán todas de contenidos variables por lo tanto podrán ir de una colocación a otra como las personas que van a una platea numerada.

El tablero que está representando a nuestra Memoria deberá llevar una numeración adecuada de 1 a 1000 . Para dar mayor homogeneidad se elegirá la numeración que va de 000 a 999 (tres casilleros para dirección)).

Esquema de la Memoria vacía

Dirección	signo	37 bits						
0 0 0								
0 0 1								
0 0 2								
0 0 3								
0 0 4								
0 0 5								
0 0 6								
9 9 6								
9 9 7								
9 9 8								
9 9 9								

Aunque sabemos que cada dígito esta codificado en sistema binario podemos exteriormente considerar los números en su expresión decimal. Es decir si hago colocar una regleta como
1001000100010100 en la dirección 224 de Memoria puedo trabajar
9 1 1 4

exteriormente con el conocimiento de que el contenido de la regleta en 224 de Memoria es el número 9114 del tal modo que gráficamente este número y los números 14 , (-128) y (-1972) y 900307 considerados respectivamente en la dirección de memoria 500 - 501 - 502 - 503 podrían ubicarse de este modo .

Dirección	Signo	palabra alfamérica longitud 6 dígitos .								
2	2	4	0			9	1	1	4	
5	0	0	0					1	4	
5	0	1	1				1	2	8	
5	0	2	1			1	9	7	2	
5	0	3	0	9	0	0	3	0	7	

Convencionalmente para el signo más se codifica 0 y para el signo (-) se codifica 1

Con estos conocimientos elementales podemos pensar en resolver el problema de cómo colocar palabras en Memoria y cómo sacarlas de allí .

No existiendo todavía un lenguaje para entenderse con Memoria directamente, sería conveniente pensar en órdenes como las siguientes que ahora ya tendrían sentido : "coloque en Memoria en la dirección 335 este número + 30111" o bien "coloque en Memoria en la dirección 999 la palabra María " --- "Saque de Memoria el contenido de la dirección 503 "

2) Actividades

1) Fabricar un tablero numerado al estilo del esquema descrito .
2) Que cada alumno confecciones 5 regletas en las que pueda variar números o letras .
3) Dar órdenes de "entradas" y "salidas"
4) Inventar órdenes para operaciones aritméticas .

Relación entre las unidades de máquina.

Es muy útil guardar y acumular información; también lo es clasificarla y disponerla de tal modo que esta información clasificada pùeda extraerse rápidamente evitando la pérdida de un tiempo precioso y que esta información permanezca al servicio de muchos para ser utilizada en cualquier momento y cuantas veces se desee. Esta es sólo una de las muchas funciones de la Memoria y de la máquina computadora en general .

Desde el punto de vista operacional en Matemática es altamente beneficiosa la existencia de máquinas electrónicas de cálculo y es obvio que el beneficio sería mayor si se le pudiera indicar a una máquina la ejecución de ciertos cálculos sin tener que inducirlos directamente.

Esto es posible en una computadora . Para realizar esta función existen mecanismos de "control" que hacen llegar la "entrada de información a "Memoria" .

La entrada puede contener datos u órdenes .

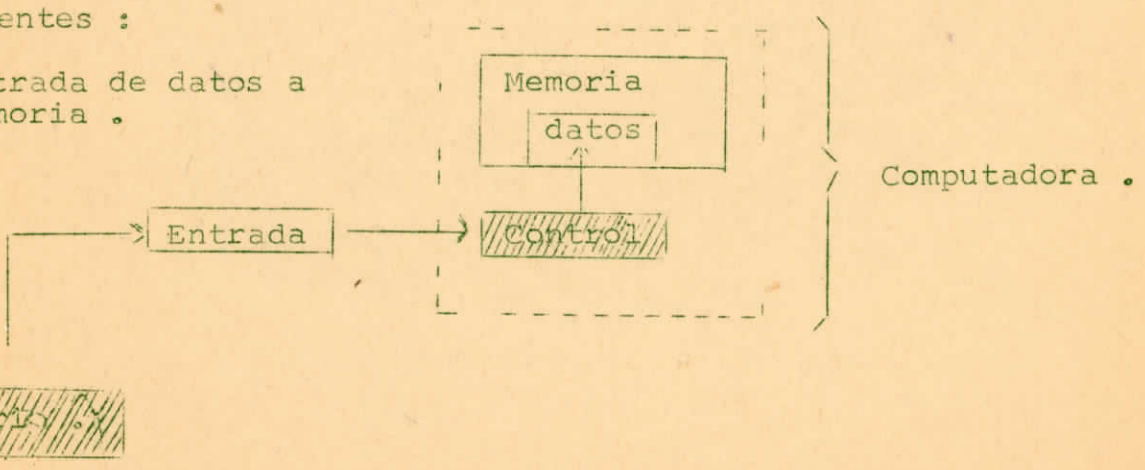
Los datos quedan en Memoria .

Las órdenes revisadas por "control" demandan ciertos datos de "Memoria" para que se procesen en la "unidad aritmética". Este paso puede repetirse varias veces de acuerdo a las órdenes y la complejidad de los cálculos.

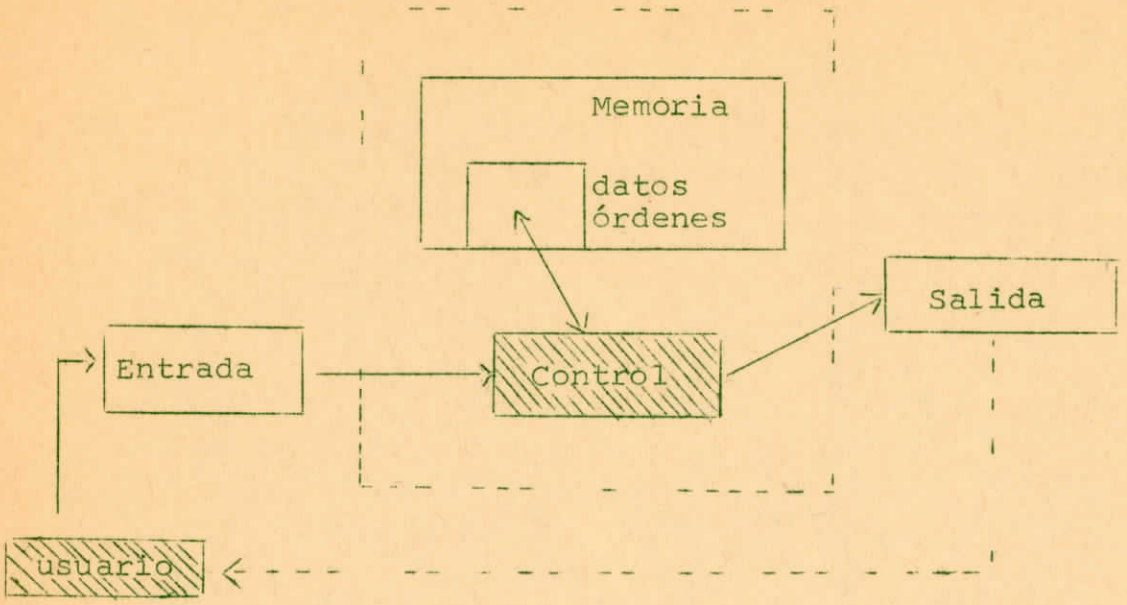
La información procesada vuelve a "control" para que este de la "salida" y se impriman las soluciones en forma correcta .

Una visión panorámica de este mecanismo lo dan los esquemas siguientes :

1. Entrada de datos a Memoria .



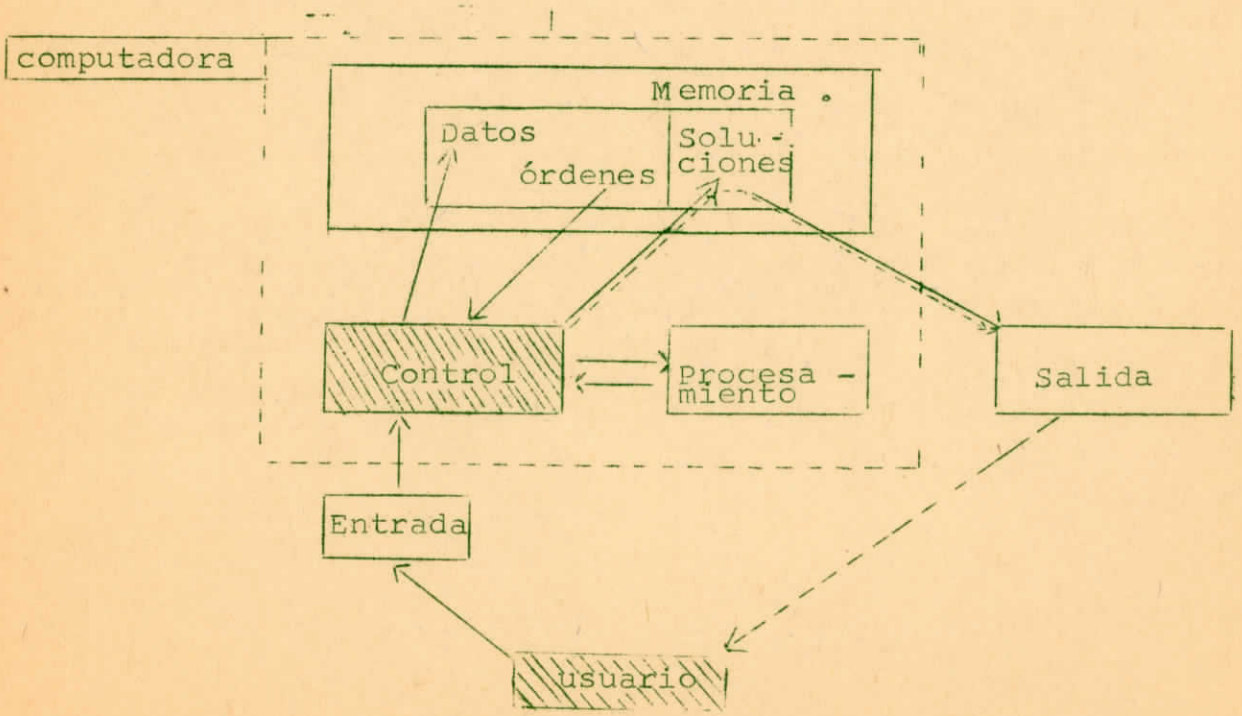
2. Almacenamiento de información y algunos usos de ella.



La Unidad de Memoria y la Unidad de Control constituyen partes de la computadora .

La Unidad de Entrada y la Unidad de Salida son implementos , son canales de comunicación .

3. - - Entrada de información
- procesamiento de datos
- salida de soluciones .



Es necesario intuir como un diálogo entre dos entidades el usuario o sujeto que usa la computadora y la máquina misma representada por el control o entidad mecánica .

Si se observa los esquemas, se verá que Entrada y Salida son dispositivos externos a la computadora y expresamente ésta ha sido mi intensión al diseñarlos; que se tenga una visión clara de que la esencia de la máquina y los procesos vitales ocurren entre la Unidad de Memoria y la Unidad Aritmética o de Procesamiento , regidos por una Unidad de Control . Mientras que la alimentación del proceso y la extracción de los resultados tiene relación estricta con el usuario que utiliza las Unidades de Entrada y Salida como canales de comunicación .

Unidad de Entrada.

Concebida ya como un elemento adicional a la máquina es fácil entender que existan diversas formas de entrada.

1) tarjetas perforadas
2) cintas de papel perforado
3) cinta magnética
4) discos magnéticos
5) máquina de escribir

En la actualidad la forma más difundida de efectuar la entrada es a través de las tarjetas perforadas.

¿Cómo recibe la computadora las tarjetas que harán la entrada de información? A través de una lectora de tarjetas , que es un dispositivo electrónico adicional a la Unidad central. Cuando se trata de discos , cintas etc. la lectora será especializada para este medio es decir lectora de discos, lectora de cinta, etc. En general la computadora tiene un equipo completo de lectoras clasificadas. En esta guía se hará referencia solamente a dispositivos a través de tarjetas perforadas.

Unidad de Salida.

Concebida como elemento adicional a la computadora que recoge la información o soluciones que se extraen de la Unidad Central y que permiten que el usuario haga uso de ella . En general

existen varias formas de salida

1) tarjetas perforadas
2) impresora
3) cintas magnéticas
4) discos
5) máquina de escribir
6) plotter o pantalla para gráficos

Las forma más **difundida** es a través de una impresora que en hojas especiales entrega la información o los resultados del procesamiento .

Unidad Aritmética o de Procesamiento

La unidad aritmética está constituida por dispositivos electrónicos que permiten obtemer los resultados aritméticos de las cuatro operaciones, aún cuando en principio la máquina solamente efectúa adiciones y multiplicaciones .

Para ilustrar esta afirmación daré un ejemplo para las operaciones en binario .

Se necesita tener dos conceptos , el de "over float" y el "complemento binario".

1) "Over float" si la máquina tiene una capacidad para 5 dígitos y al sumar se forma un numeral de 6 dígitos, existe rebalse que algunas máquinas acusan con luz roja .

2) Complemento binario

Tal como sucede en el sistema décuplo: que si se da el numeral 315 su complemento para 999 es 684 y para 9999 es 9684, en los binarios el complemento se determina para $2^n - 1$, es decir el complemento de 1101 para $1111 = 2^4 - 1$ será 0010 y para $11111 = 2^5 - 1$ será 10010 ya que

$$\begin{array}{r} 1101 \\ +\ 0010 \\ \hline 1111 \end{array} \qquad \text{y} \qquad \begin{array}{r} 1101 \\ +\ 10010 \\ \hline 11111 \end{array}$$

Adición de binarios: tenemos la siguiente tabla :

+	0	1
0	0	1
1	1	0 →

pero éste valor es 10 de modo que hay una reserva de 1

1) Adición

```
 1111      reservas
 11101     --------->   29
  1011     ------->     11   +
 ------                ----
101000     --------->   40
```

comprobación en el sistema décuplo .

2) Sustracción

```
          1110
        - 111
          ||
          \/
la máquina suma    1110
el complemento   + 1000
del número       1 0110
                   1
el 1 de overfloat ---
se suma >          111
```

Solución

Comprobación en sistema décuplo .

```
1110   -----   14
 111   -----  - 7
                ---
```

----------> 7

3) Multiplicación.

.	0	1
0	0	0
1	0	1

Comprobación

```
  10111   x 101          23 . 5
  10111                  115
 10111
-------
1110011
```

también pudo hacerse

```
  10111                1
  10111 }
  10111 }  +           10   (2 veces
 -------
 101110 }
  10111 }  +           11   (3 veces)
 -------
```

```
 1000101 }+      100  4 veces)
   10111 }
 1011100 }
   10111 }+      101 ( 5 veces)
 1110011         solución
```

o bien

```
 10111          23 . 5
 10111
 10111
 10111
 10111
1110011         115
```

4) División : Se hace por sustracción sucesiva, ya se explicó como sustraer (overfloat y complemento)

101000 : 1000 a 101000 se le suma el complemento de 1000= 1000 = 110111 (con esto se está restando)

```
  101000 }
  110111 }      +  1 vez = 1
1 011111 }
       1 }
  100000 }
  110111 }
1 010111 }      +  2º vez = 10
       1 }
  011000
   10111        +  3a. vez = 11
1 01111
       1
   10000
   10111        +  4a. vez = 100
1 00111
       1
    1000
-   1000        +  5a. vez = 101
    0000
```

101000 : 1000 = 101

ACTIVIDADES

Sugerencia . Esta parte de trabajo con operatoria de binarios no es esencialmente necesaria. Sin embargo la mayor comprensión de la máquina requiere tener frescos estos conocimientos, especialmente para los colegas que darán estas clases. Respecto a los alumnos doy una guía de material binario para ejercitar este aspecto . El profesor verá si conviene ejercitar con los alumnos esta operatoria en relación con el nivel del curso y con las inquietudes y las preguntas que los alumnos hagan al respecto.

A) Binarios a decimales.

1110_2	10001_2	11000_2
11111_2	1001001_2	1111001_2
1010101_2	010101_2	1000001_2
1100011_2	10111101_2	110011_2
1111111_2	1000000_2	10001111_2

B) Contar en binario . ejemplos . Observación: En este ejercicio se trata de hacer siempre uniones binarias.

x x x x x x x x = 1001

a) x x x x x x x x x
x x x x x x

b) x x x x x x x x x x x x x x x

c) x x x x x x
x x

d) x x x x
x x x x x x
x x x x

e) x x
x x x
x x x x
x x x x x
x x x

f) x x x x x x
x x x x x
x x x x
x x x
x

Ejemplo : x x x x x = 101
2^2 2^0

C) Decimales a binarios

42	108	24
165	91	36
72	234	587
21	19	54

D) Escribir los siguientes decimales en binarios usando CBD .

42	108	24
165	91	36
72	234	587
21	19	54
500	1003	997

E) CBD a decimales

10010011 =

0110000101110101 =

100101110110 =

100110011001 =

100000110100 =

100110000111 =

Ejercicios 2

Operaciones con números binarios .

Adición

A)

1001	11111	1000
100	111	10001

100110	110101	111111
100110	101010	1011011

B) Adición

111	100001	11101
111	100001	11101
		11101

B) Adición

111	100001	11101
111	100001	11101
		11101

C) 10101
10101
10101
10101
10101

10101 x 101

Compruebe ambos resultados usando la equivalencia decimal .

D) Efectuar las siguientes multiplicaciones y comprobar con números decimales.

Ej. : 111 • 101
111
111
100011

comprobación

7 • 5 = 35

100011 = 35

a) 11001 x 1101

b) 101 x 11100

c) 1001111 x 110010

d) 110011 x 100

e) 110011 x 10

f) 110011 x 10000

E) - Cómo se multiplica por potencias de 2 ?
- Qué propiedades conocidas se cumplen en los números binarios ?.
- existe isomorfismo entre decimales y binarios ¿porqué ?

EJERCICIOS

A) Determinar el complemento de un número binario . Ejemplos

10011 ——————→ 01100

1) 10111
11110
00111
01010
01000
11001
101010
110001

B) Efectuar las siguientes sustracciones.

1100	11101	1011110
110	1001	1111

C)

100011	101011
- 1110	-100101
11100111	10011111
- 100010	- 111111

D) Dividir .

110000 : 1100

101101 : 1001

1001110 : 110

Resolver por sustracciones sucesivas, comprobando con números decimales.

PROGRAMA ALMACENADO

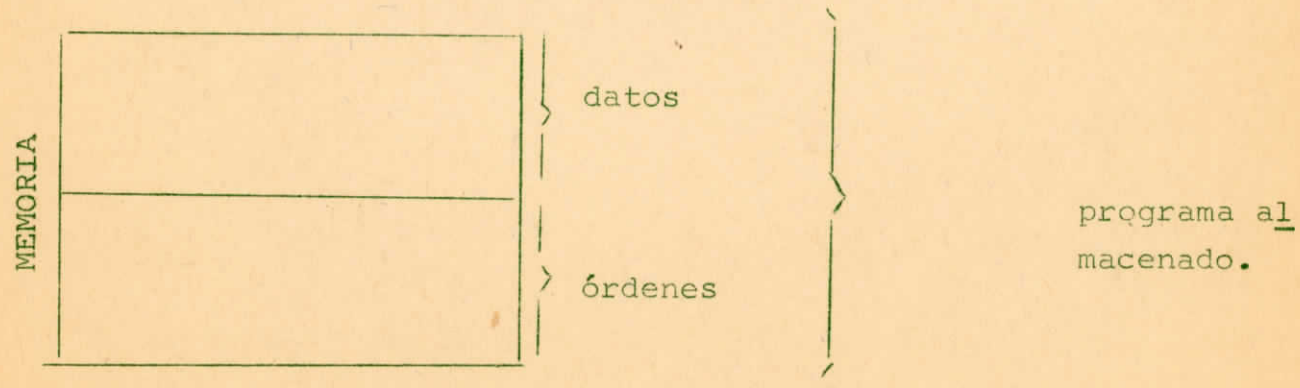
Uno de los conceptos más esenciales en el estudio de la computadora electrónica y en la técnica de la programación y en último término, para hacer posible el manejo de la máquina, es el de programa almacenado.

El programa almacenado consiste en entrar a la Memoria los DATOS y las ORDENES en forma secuencial de modo que la Unidad de Control no tenga ningún tropiezo para dirigir, a velocidad electrónica, la secuencia de pasos; hasta culminar en la solución e impresión correcta del problema .

Durante mucho tiempo se programaba en forma externa; es decir, la computadora recibía las instrucciones en forma exterior. Ejecutaba la instrucción, devolvía una solución y esperaba por la próxima instrucción que llegaba a través de medios externos; ya fuera tarjetas perforadas o cintas. Este sistema era sumamente lento y limitaba la capacidad de la computadora, aumentando el trabajo del programador que estaba literalmente atado a la maquina .

El programa almacenado fue invención de John Von Neumán - Matemático de este siglo de condiciones extraordinarias . El investigó y llegó a lograr la automatización total de la computadora , almacenando el programa (instrucciones y datos en memoria) y haciéndola dirigir su propio programa casi, sin intervención humana.

El programa, es decir el conjunto de DATOS y ORDENES debe estar elaborado de tal modo que la máquina lo "entienda"; es decir, que los dispositivos electrónicos se accionen en forma correcta, por lo tanto la programación directa se hace en sistema binario y se habla de un Lenguaje de Máquina .



Las palabras que entran en Memoria podrán ser alfaméricas (números y letras) unas servirán para almacenar datos y otras para almacenar órdenes.

En qué difiere un dato de una órden ?

Las palabras constan de 10 casilleros , tres casilleros para la ubicación o dirección en Memoria y luego 7 casilleros para datos u órdenes .

a) si se trafa de un dato, un casillero para el signo 0 ó 1 y 6 para los dígitos que componen el número (ver ejemplo a,b)

b) si se trata de una órden : los dós primeros casilleros llevan 00, dos casilleros llevan codificada la órden y los tres últimos, para la dirección donde se halla el contenido en el que recae la órden. (ver ejemplo c) . Como caso especial de órden para la lectora y solo para explicar un lenguaje de Máquina en una Máquina hipotética inventada para fines didácticos, en lugar de 00 antes de la órden se designa un 9 en el primer casillero para alertar a Control de que se trata de una órden especial para la "lectora" así:

			9	0	READ	2	2	2

"Lea los números que viene a continuación y éntrelos en Memoria a partir de la dirección 222

Nº de ubicación

signo

a)

3	3	3	1	5	7	4	2	0	3

palabra dato

dato: el Número −574.203

b)

3	3	4	0				5	2	8

palabra dato

datos: Número + 528

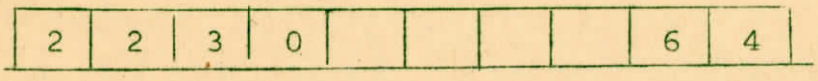
orden

c

3	3	5	0	0	5	T	2	2	3

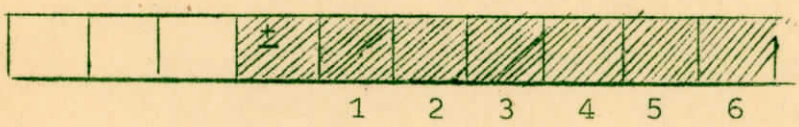
palabra orden

orden: <u>almacene</u> (STORE) en Memoria lo que hay en el acumulador en este momento en el lugar 223 de Memoria .
Si en el "acumulador" (Ac) se tiene el número 64 este número pasa al lugar 223 de Memoria.



+ 64 y ahí queda almacenado

En la Unidad de Aritmética existe una palabra libre que registra los resultados parciales y finales de las operaciones. Es el (Ac) Acumulador que ocupa una palabra con el mismo número de dígitos .



Máquina Computadora hipotética .
Lenguaje de Máquina .

Como actúa la computadora electrónica? La mejor forma de comprenderlo es ir siguiendo los pasos del procesamiento (las operaciones que hace la máquina) desde la Entrada de un problema, hasta su Salida o solución .

Esto es posible con una máquina simplificada o hipotética. Existe gran variedad de órdenes "inventadas" para visualizar el Lenguaje de Máquina o sea la forma de manejarla .

Instrucciones de Hollindale y Tootill .
para Entrada con cinta magnética) .

01 Sumar C(n) a C(A)
02 Restar C(n) a C(A)
03 Multiplicar C(A) por C(n)
04 Dividir C(A) por C(n)
05 Copiar C(n) en A
06 Imprimir C(n)
07 Copiar C(A) en (n)
08 Cinta a (n)
09 Saltar a (n) si C(A) es negativo
10 Parar .

Algunas instrucciones de Maly y Hilweil.

ADD	Sumar	FAD	Sumar en flotante
SUB	Restar	FSD	Restar en flotante
MPY	Multiplicar	FMP	Multipl." "
DVP	Dividir	FDP	Dividir " "
RND	Redondear	FRN	Redondear "

STO Almacenar
LDQ Cargar el Acumulador
TRA Transferir
CAS Comparar el Acumuladro con Almacenamiento
HAL Detenerse

Instrucciones para Lenguaje de Máquina.

Dr. Ralph T. Heimer
State College. P.A. University USA.
(Publicación 1970) .

LDA	101	Carge el acumulador
STO	102	Guarde el contenido del Ac.
ADD	103	Sume el Acumulador
SUB	104	RESTE DEL "
MPY	105	Multiplique el Acumulador
DIV	106	Divida el "
WWD	107	Imprima una palabra
RCD	108	Lea una tarjeta
BRU	109	Salte incondicionalmente
BRC	110	Salte si, condicionalmente
LR1	111	Coloque en registro 1 lo almacenado en Memomoria
LR2	112	Coloque en registro 2 lo guardado en M.
TR1	113	Compruebe si es 0 el contenido del registro 1
TR2	114	y salte si lo es (para registro 2 lo mismo)
SHA	115	Deslice a la izquierda (decimales)
SHR	116	Deslice a la derecha (decimales)
HLT	117	Deténgase .

Instrucciones del Hull / Day .
Usa directamente el código

+ 0000010	AAA	Loal accumulator
+ 0000011	AAA	Store accumulator
+ 0000020	AAA	Add to accumulator
+ 0000021	AAA	Subtract from accumulator
		etc .

Siempre las órdenes son similares: Entrada e Impresión Aritmética, Saltos condicionados e incondicionados, Detenerse, etc.

Ordenes simplificadas

(Adaptación Dr. Jaime Michelow)

Código Numérico	Código verbal	Significado de la órden
01	CAD	traiga al Acumulador
02	AD	sume
03	SU	reste
04	M	Multiplique
05	DIV	divida
06	ADA	sume el valor absoluto
07	90READ	lea. "lectora"
08	ST	almacene
09	W	Imprima
10	CU	salte incondicionalmente
11	CNZ	salte condicionalmente
12	STØP	deténgase
13	90STØP	deténgase "lectora"
14	FAD	sume en Floating point
15	FSU	reste " " "
16	FM	multiplique en Floating pojnt
17	FDIV	divida " " "
18	FADA	sume valor absoluto en Floating point .

Observación.

Seguiremos en esta guía las órdenes usadas por el Dr. Michelow ya que él, es el autor de esta Unidad programática y con la intención de aprovechar los modelos que él ha utilizado a través de los cursos impartidos por el C.P.E.I.P.

Nos ceñiremos a la interpretación que hace de las órdenes y a su forma de uso en el Lenguaje de Máquina.

La Máquina hipotética que aquí describiremos tendrá, según expresión del Programa, una Memoria de 1000 palabras .

La estructura de esta Computadora didáctica, es precisamente la que ya se ha explicado. Su Lenguaje es directo es decir uno muy simple y similar a los anteriores. (Ver la lista de órdenes que se usará).

Lo importante es la correcta interpretación de las órdenes. Por lo tanto las Explicaremos enseguida .

Orden de Entrada : Orden READ con codificación 07 en la lista de órdenes (el número que antecede a la expresión) .

La orden completa de entrada es :

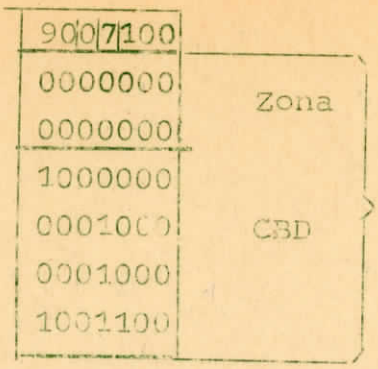
90READ101	9007101

En ambos casos la órden es: "Lea y coloque en Memoria a partir de 101" . (Otros lenguajes dicen simplemente : "Carge la Memoria ".

Qué se coloca en Memoria? Ya lo hemos dicho: los DATOS, para que las INSTRUCCIONES u ORDENES que se entran luego, puedan actuar sobre ellos .

Ejemplo : 7 datos en 101
102
103
104
105
106

y luego las respectivas órden de suma

90 READ 100 = 

Al exterior .

en el interior de la máquina en binario .

Para no complicar los ejercicios usaré la palabra y no el código al resolver los problemas .

Revisando las órdenes:

01) CAD 204 c(abc) ----> Ac
- "lo que está en 204 tráigalo al Acumulador "
- "el contenido de 204 " " " "

02) AD 108 c (Ac) + c(108) ----> Ac .
- "a lo que está en el Ac súmele lo que está en 108"
- "al contenido del Ac súmele el contenido de 108 "

03) SU 101 - "al contenido del Ac sustráigale el contenido del 101"

04) M 112 - "el contenido de Ac multiplíquelos por el contenido de 112"

05) DIV 319 - "el contenido del Ac divídalo por el contenido de

06) ADA 333 - al contenido del Ac súmele el valor absoluto del contenido de 333

Estas órdenes trabajan con números enteros y decimales. Se puede observar que siempre se opera sobre el contenido del Ac y el resultado de la operación queda en el Ac . Es por esto que en la lista de operaciones, después de la operación se indica con una flecha el acumulador.

Ejemplo :

05) |D|IV|a|b|c| quiere decir c(Ac) ⁄. c(abc) -----> Ac

(a b c representan tres numerales cualesquiera).

"el contenido del Ac divídalo por el contenido de a b c y el cuociente queda en Ac."

Orden de entrada para tres datos: (ejemplo)

Se tiene A,B,C,más adelante se quiere operar con ellos. Es necesario entrarlos a Memoria .

"Entre en Memoria a partir de la dirección de Memoria 100 los siguientes datos A = -14 , B = 52, C = 345 " .

Observación: A cada palabra (7 casilleros) se le agregan tres a la izquierda para dar el número de órden o secuencia, que coincide para los datos con la dirección en Memoria .

			signo						
			9	0	READ		1	0	0
1	0	0	1	0	0	0	0	1	4
1	0	1	0	0	0	0	0	5	2
1	0	2	0	0	0	0	3	4	5

Una vez entendido el mecanismo de entrada podrá programarse una operatoria sencilla :

Ejemplo
a) A + B + C
b) (A + B) - C
* c) (A x B) + (C x D)
d) A + (B x C)
e) A + (B/C) - B

Los colegas podrán elaborar otros ejercicios a modo de los aquí sugeridos .

Veamos el desarrollo del programa para el ej. c) A = 18, B = -21
C = 13 , D = 73

(A x B) - (C x D)

```
      90  READ100
100               18      A=18 en 100
1011              21      B=-21 en 101
1020              13      C=13 en 102
1030              73      D =73 en 103
1040000000000             Lugar de reserva
→105      CAD 102         Traiga al Ac lo que está en 102  C= 13
106       M   103         Multiplique por lo que hay en 103  D=73
107       ST  104         Guarde el prod que está en Ac en 104
108       CAD 100         Traiga 18 al Ac; o sea A = 18
109       M   101         Multiplique 18x(-21) o sea AB en Ac
110       SU  104         A lo que está en Ac= AB quítele CD que está en 104
111       W               Imprima AB - CD = - 1327
112       STØP            Deténgase! deje de operar
        9 STØP            Deje de leer.
```

Solución = - 1327 .

——→ La flecha indica cuando se empieza la operatoria.

Observación: Hay dos STØP una para la máquina y otro para la lectora (codif. 9 sólo para lectora) .

El Ac (Acumulador) es como un pizarrón en donde se escriben los datos, donde se opera y donde quedan los resultados.

En el programa, en los lugares interiores donde no hay nada escrito, antes de los números y antes de las órdenes deben ir Ceros. Ejemplo 00CAD102

Por qué no se multiplicó primeramente A · B ?

Esta alternativa se examinó previamente con respecto a la secuencia de solución del problema: Si se calcula A · B hay que guardarlo en un lugar de reserva y al efectuar C · D hay que guardarlo en otro lugar de reserva, puesto que la sustracción no es conmutativa y no se puede efectuar C · D - A · B . Esto obliga a perder dos pasos: el primero almacenar, el segundo llamar nuevamente al acumulador A · B para sustraerle C D . Para examinar estos aspectos que son como la lógica de la solución de un problema, me referiré más adelante a una técnica especial en programación : el diagrama de flujo. Pero por ser estos problemas relativamente sencillos no usaré el diagrama, hasta explicar el segundo aspecto del esquema de trabajo "el papel del sujeto programador" y los "Lenguajes simbólicos" .

Algo más, antes de continuar examinando la estructura y el poder de otras órdenes.

En el programa que revisamos cada una de las 15 líneas que representan la Entrada, Procesamiento y Salida, deben ir en tarjetas perforadas. O sea para 15 líneas, se necesitan 15 tarjetas perforadas.

Las órdenes que allí aparecen deben ir codificadas por la tipiadora

o sea	90	READ	----> 07	lectora
		CAD	----> 01	
		M	----> 04	
		ST	----> 08	
		W	----> 09	
		STØP	----> 12	
	90	STOP	----> 13	lectora

Ejemplo órden para la lectora (una tarjeta)

9013000

La órden : 09) | W | 0 | 0 | 0 | es ligeramente diferente. A la órden IMPRIMA (W = WRITE) siguen ceros, o sea no se indica ningún lugar en Memoria. El significado práctico es que la máquina "imprime lo que está en el Acumulador, o sea el último resultado de la operatoria. (Revisar estas órdenes en el programa desarrollado pág. 39) y también las siguientes .

Las órdenes 10) y 11) son ambas de Transferencia de Control es decir: obligan a romper una secuencia como: 103 - 104 - 105 - 106 y saltar a un órden más adelante, 110 por ejemplo, evitando cumplir las instrucciones intermedias o bien saltando a un órden anterior, para repetir parte de la secuencia .

a)

10) | C | U | 3 | 4 | 2 | Esta es una órden que contiene un salto incondicional . Suponiendo que el programa estuviera corriendo en la secuencia Nº 315 al encontrar esta órden: "salte incondicionalmente a la secuencia 342" , no realizaría las instrucciones desde 316 - 341 y empezaría a operar inmediatamente en 342 .-

b)

10) | C | U | 3 | 0 | 9 | Esta órden indica, suponiendo siempre que se está en la secuencia 315 que incondicionalmente se debe regresar a 309 y ahí volver a avanzar 310- 311- 312- 313- 314 hasta 315 . En este caso especial volvería a 309 y repetirá indefinidamente este proceso . En Programación esta vuelta atrás y repetición se denomina LØØP (Más adelante se verá)

11) | C | NZ | 1 | 0 | 9 | Es una órden de salto condicional . Representa una alternativa la traducción de la órden en inglés "Change if not zero" indica que el salto está condicionado a que em el Ac el contenido sea o no sea 0 .

a) Ejemplo

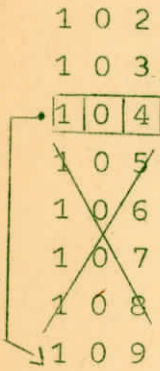
1 0 2					
1 0 3	SU	1	0	0	
1 0 4	CNZ	1	0	9	

1 0 5
1 0 6
1 0 7

En el Ac el contenido es 47
En 100 el contenido también es 47
En el Acumulador 47 - 47 = 0 (La existencia de 0 en el Ac anula la orden de salto)

No hay salto se sigue en la secuencia 105- 106- 107- 108- 109- 110 etc.

b) Ejemplo



Ac = 47
En 101 el contenido es 20
Ac = 47 - 20 = 27

Control transfiere la continuación del programa a la secuencia 109, porque el último contenido del Ac ≠ 0 .
Se salta de 104 a 109 .

ACTIVIDADES.

Conviene ejercitar las órdenes. Que los alumnos adquieran dominio de ellas traduciéndolas correctamente en situaciones aisladas .

Que los alumnos confeccionen cartulinas con las órdenes del problema resuelto y lo dramaticen como se indica a continuación .

Un alumno puede hacer de receptor o Entrada (su rol es recibir datos y órdenes)

Otro alumno Lector Lee los datos en voz alta. Quien hace de Control los ubica en Memoria (Tablero que la representa) .

Los alumnos se reparten las siguientes órdenes y las van entregando a Entrada y este a Lector. Mientras que Control indica a Unidad Aritmética (otro alumno) que opera en el pizarrón (Acumulador) y el Control va y viene de Memoria proporcionando los datos o guardándolos.

Un alumno representa la Salida, va al pizarrón apunta el resultado final y lo entrega al profesor. En este momento Lectora lee su última órden : FIN DEL PROBLEMA .

Faltaría explicar una última órden ADA. Quiere decir "Add alsolute" Sume el valor absoluto .

3	2	1	0	0	A	D	A	1	0	2

"Sume el valor absoluto del contenido de 102"
Si en 102 se encuentra guardado un 4, al realizarse la órden quedará
$4 + |4| = 4 + 4 = 8$

Si en 102 se encontrara guardado un -5, al realizarse la órden quedaría
$-5 + |-5| = -5 + 5 = 0$

Esta órden se combina con "Cambie si no es 0" (C N Z)
y sirve para reconocer si el número que está guardado en 102 es positivo o negativo. Si es positivo luego de la órden ADA queda un valor $\neq 0$.

Si el número es negativo luego de la órden queda un valor = 0

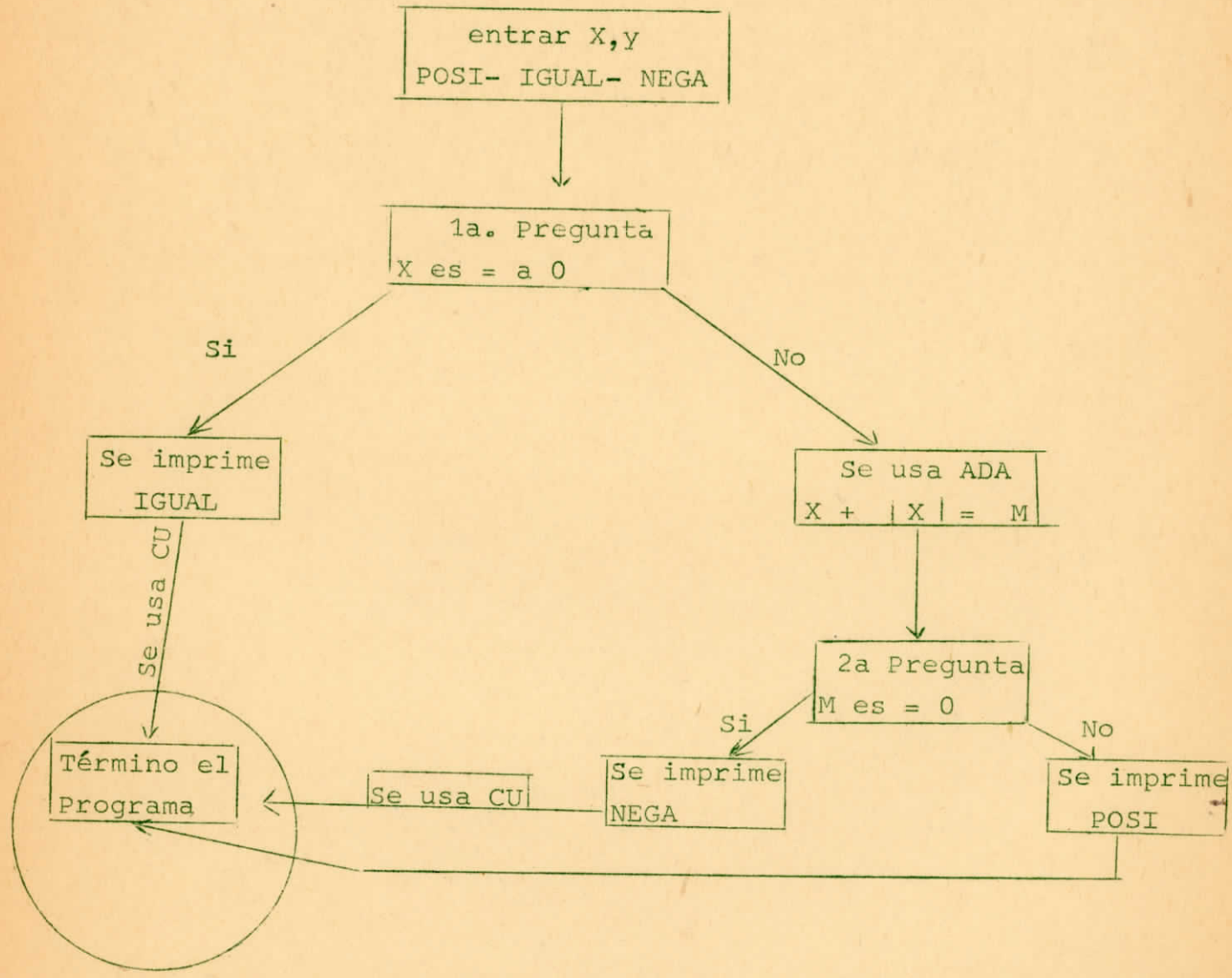
Vamos a ejemplarizar la utilidad de esta órden. Solucionando un 2º problema.

Problema : Que la máquina reconozca si un número es positivo o negativo. Ejemplo el número (- 55)

Preámbulo: Se entrará a Memoria tres palabras significativas
"Posi" para > 0
"Nega" para < 0
"Igual" para = 0

En este programa se ejercitan las órdenes ADA-CNZ y CU .

El programa no está elaborado para un número especial como (-55) Sino para cualquier entero X. Es por esto que se deben incluir todas las alternativas. Los pasos seguidos en general son según el diagrama:



Es de gran importancia la ejercitación y la comprensión del lenguaje de Máquina por lo tanto se darán algunos problemas resueltos a continuación para ser estudiados .

Como actividad para los alumnos. se les pedirá la programación de problemas triviales. a, b, d, e del ejemplo anterior .

Ejemplo de actividades :

a) $\frac{A \cdot B}{C}$

b) $\frac{A + B}{A - B}$

c) $\frac{(A + B) - (A - B)}{C - D}$

d) $Y = \frac{C \cdot t \cdot \%}{360}$

Fórmula del interés simple .

e) $y = u^2 + v^2$

f) $n = 2 \pi r$

g) Sup. $\triangle = \frac{b\ h}{2}$

Problemas resueltos.

1) Sumar los primeros 60 números pares en $\mathbb{N}$.

2 + 4 + 6 + 8, ---------- + 120

$\sum_{N=2}^{120} N = S$

Programa

		90	READ	100
	100		2	N
	101		122	
	102		2	
	103		0	S
→	104		CAD 100	2
	105		AD 103	2 + 0 = 2
	106		ST 103	S = 2
	107		CAD 100	2
	108		AD 102	2 + 2 = 4
	109		ST 100	N = 4
	110		SU 101	4 - 122 = - 118
	111		CNZ 104	No es 0, va a 104 !
	112		CAD 103	S = Suma de los 60 primeros Naturales
	113		W 000	imprime $\sum_{N=2}^{120} N = S$
	114		STOP	
		90	STOP	

La interpretación del Programa es la siguiente. Con la órden 90READ100 se entra en Memoria los datos :

en 100 ------> 2 (primer número par)

en 101 -------> 122 (Este valor sirve para comparar, ya que el último par que debe sumarse es 120) .

en 102 -----> 2 Este número sirve para incrementar cada vez en 2 e ir formando así el sucesor de cada par hasta llegar a 120.

en 103 -----> 0 Aquí estarán acumuladas las sumas

0 + 2, la primera.

2 + 4, la segunda

2 + 4 + 6, la tercera, etc . hasta $\sum_{N=2}^{120} N$

Vamos a "correr" imaginariamente el Programa en la secuencia 104 empieza la operatoria o procesamiento de datos:

104	se trae al Acumulador N= 2
105	se le suma + 0
106	se guarda en S = 2
107	se trae el Acumularor N= 2
108	se le agrega + 2
109	se guarda en N = 4
110	4 permanece en el Acumulador y se le sustrae 122
111	2- 122 = 0 ? ---- No! ,¡se vuelve a 104 !

Segunda vuelta al Programa

104	se trae al Ac. N = 4
105	se el suma S = 2
106	se guarda en S = 6
107	se trae al Ac. N = 4
108	se le suma 2
109	se guarda en N = 6
110	6 permanece en el Acumulador y se le sustrae 122
111	6 - 122 = - 116 ¡No es 0! Se vuelve a 104 .

Tercera vuelta

104	se trae alAc N = 6
105	se le suma S = 6
106	se guarda en S = 12

107 se trae al Ac = N = 6
108 se le suma 2
109 se guarda en N = 8
110 8 permanece en el Acumuladro y se le sustrae 122
111 8 - 122 = - 114 ¡No es 0! Se vuelve a 104 .

Se repetirá estos pasos hasta que se cierra el ciclos de repetición, en la secuencia 112 . Veamos el cierre :

104
105
106
107 se trae al Acumulador N = 120 .
108 se le suma 2
109 se guarda en N = 122
110 122 permanece en el Acumulador y se le sustrae 122
111 122 - 122 = 0
112 se trae al Acumuladro S = $\sum_{N=2}^{120}$ N = (sumatoria final)
113 se imprime la sumatoria final ,
114 se detiene el proceso .

Observación El incremento de N se hace hasta N + 2 pero este último sumando ya no se procesa y de este modo se consigue la sumatoria con 60 sumandos que era la exigencia del problema.

2) Transfromación de temperaturas Farenheit a Celsius entre $1^{\circ}F$ y $212^{\circ}F$

Fórmula $\frac{5}{9}(N-32)$

Programa

	90READ	100	
100		5	
101		9	
102		32	
103		213	
104		1	N
105		1	
106			casillero vacío para almacenar un cálculo parcial .
→107	CAD	100	5
108	DIV	101	5/9
109	ST	106	5/9 en 106
# 110	CAD	104	1
111	SU	102	1 - 32
112	MU	106	(1 - 32)5/9
113	W	000	se imprime 5/9(1-32)
114	CAD	104	1
115	AD	105	1 + 1
116	ST	104	2 en 104
117	SU	103	2 - 213 = - 211 ¿No es 0 ?
118	CNZ	110	Vuelve a 110 .
119	STØP		
	90STØP		

-- - - - - - - - - --

Interpretación del Programa

Al"correr" el Programa en la primera vuelta se empieza a procesar (efectuar cálculos) en la secuencia 107 y se forma la constante 5/9 colocándola en Memoria en el lugar destinado a guardar reservas , en este caso en 106 .

110 | se trae al Acumulador 1
111 | se le sustrae 32
112 | se multiplica (1 - 32) 5/9
113 | se imprime este valor: 1º F -------> -17,222°C.
114 | se trae N = 1 al Acumulador
115 | se le suma 1
116 | se guarda N = 2 en 104
117 | 2 permanece en el Acumulador y se le compara con 213
118 | 2 - 213 = - 211 ; distinto de 0 !
Se vuelve a 110 .

Segunda vuelta

110 | se trae al Ac 2
111 | se le sustrae 32
112 | se multiplica (2 - 32) 5/9
113 | se imprime este valor : 2° F ----> - 16,669°C.
114 | se trae N = 2 al Ac
115 | se le suma 1
116 | se guarda N = 3 en 104
117 | 3 permanece en Ac y se le compara con 213
118 | 3 - 213 = - 210 distinto de 0 .
Se vuelve a 110 .

El ciclo se cierra cuando N que va tomando los valores 1, 2, 3, 4, 5 .... llega a N = 213 y al compararlo con 213 la diferencia es 0. El último valor impreso en este momento es (212-32) 5/9 o sea 212° F -----> 84° Celcius . La secuencia detiene el procesamiento, y se ha obtenido la siguiente tabla de valores.

Tabla solo de 1 - 45

1° F	- 17,222 °C	36° F	2,222 ° C.
2	- 16,669	37	2,778
3	- 16,111	38	3,333
4	- 15,556	39	3,889
5	- 15,000	40	4,444
6	- 14,444	41	5,000
7	- 13,889	42	5,556
8	- 13,333	43	6,111
9	- 12,778	44	6,667
10	- 12,222	45	7,222
11	- 11,667		
12	- 11,111		
13	- 10,556		
14	- 10,000		
15	- 9,444		
16	- 8,889		
17	- 8,333		
18	- 7,778		
19z	- 7,222		
20	- 6,667		
21	- 6,111		
22	- 5,556		
23	- 5,000		
24	- 4,444		
25	- 3,889		
26	- 3,333		
27	- 2,778		
28	- 2,222		
29	- 1,667		
30	- 1,111		
31	- 0,556		
32	0,000		
33	0,556		
34	1,111		
35	1,667		

Actividades :

Programar el siguiente problema : hacer una tabla con los valores del espacio recorrido por un objeto en caída libre .

Fórmula $\frac{1}{2} g t^2$

$\frac{1}{2}$ es constante

g = 9.8 cte.
t varía de 1 a 15 segundos

Sugerencia:

Dar la entrada de los datos en Memoria. Supongamos que el Programa comienza en el lugar 200 de Memoria. A partir de 200 se ubican los datos .

```
    90READ200
200       0 5        constante
201       9.8        aceleración de gravedad g. (cOnstante)
202         1        t variando
203         1        necesario para incrementar hasta 15
204        16        número de comparación
205    reservas
```

```
→206
207
 .
 .
 .
 .                   ¿Cómo se continúa la Programación )
```

Decimales.

Las Computadoras eléctrónicas en general tienen dos formas para trabajar los decimales:

El Modo Fixed point que permite fijar los decimales con que se desea trabajar.

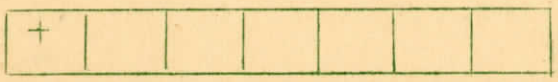
El Modo Floating point que deja abierta la posibilidad de trabajar con decimales que superan la posibilidad de los dígitos que la máquina tiene para notación común (6- 10- 12- 24 dígitos) .

1) Fixed point .

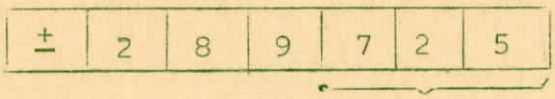
Máquinas que tienen posibilidad de palabras de 10 dígitos decimales (0 - 9), pueden fijarse para cálculos con 1 decimal 2, 3, hasta 9 decimales. (Hewlett Packard 9100 B).

El uso de los decimales limita los dígitos reservados para la parte entera del número. Así, si se quiere trabajar con 7 decimales, la parte entera no podrá tener más de 3 dígitos; 8 decimales exigen una parte entera de dos dígitos, etc.

Ejemplo aplicado a la capacidad de nuestra máquina hipotética con posibilidad de 6 dígitos .

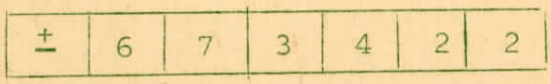


a) si se quiere trabajar con 3 decimales quedan 3 lugares para la parte entera



289.725

b) si se quiere trabajar con 5 decimales queda libre solamente 1 lugar para la parte entera .



6.73422

2) Floating point

El uso de este modo libera totalmente la posibilidad física de la máquina. Es decir aún cuando ella posea capacidad de (6 - 10 - 12 - 24 dígitos) la notación permite superarlos.

El modo floating point corresponde a lo que se denomina notación científica, que facilita de hecho la expresión de números muy grandes o muy pequeños .

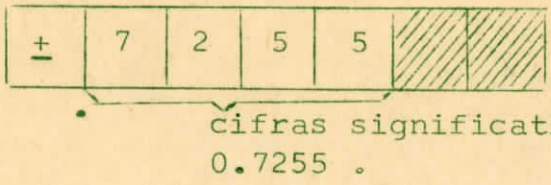
Ejemplo

$0.123 \quad 10^{18} = 123.000.000000.000.000$

$0.4235 \cdot 10^{-6} = 0.0000004235$

$0.8 \cdot 10^{-13} = 0.00000000000008$

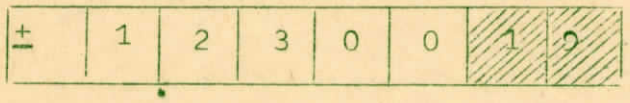
Existen convenios algunas máquinas llevan el punto decimal de tal manera quê este antecede las cifras significativas



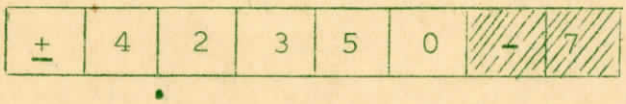
cifras significativas.

0.7255 .

Otras llevan el punto decimal después del primer casillero libre dejando posibilidad para colocar en él, la primera cifra significativa .



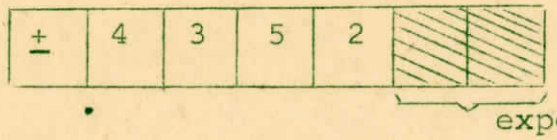
$1.23 \cdot 10^{9}$



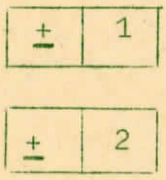
$4.325 \; 10^{-7}$

La notación floating point requiere de dos casilleros para indicar el exponente de la potencia de 10 que determina el decimal. Por convenio se pueden ocupar los dos primeros después del signo o las dos últimas.

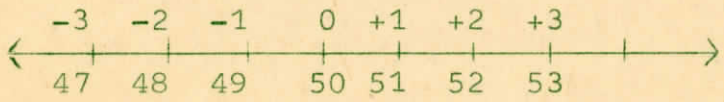
Para nuestra máquina hipotética usaremos la última alternativa y el punto decimal a la izquierda de las cifras significativas .



Los dos lugares destinados al exponente pueden cubrir exponentes entre 0 y 9 si se deja un lugar de los dos, para indicar el signo. Es decir quedarían reducidos a designar un dígito.



Con el fin de cubrir exponentes entre 00 y 99, ocupando los dos lugares se ha convenido en designar como positivas de 50 a 99, y como negativos a partir de 50 hasta 0 .

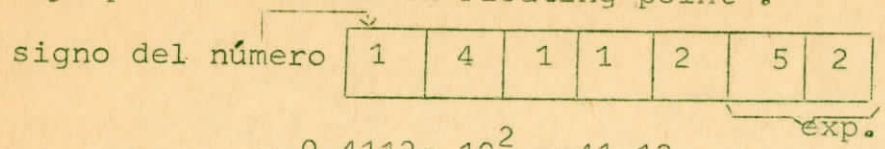


Ejemplo: al exponente + 14 corresponde | 6 | 4 |
al exponente - 9 corresponde | 4 | 1 |

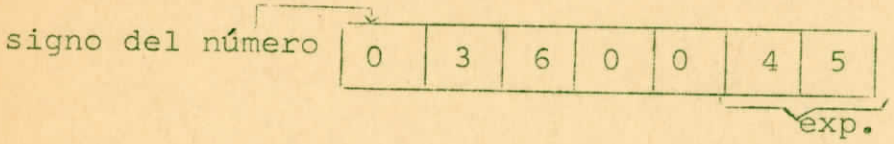
Otra acepción de lo que es el modo Fixed point, más antigua, se refiere a trabajar los decimales como si fueran enteros y el programador deberá cuidar de colocar los decimales correspondientes después de una seudo multiplicación o división. Situación por demás engorrosa.

Esto lleva a identificar en muchos textos de Programación el modo Fixed point con los Enteros y el modo Floating point con los Reales.

Ejemplo de núneros en Floating point .



$- 0.4112 \cdot 10^{2} = -41.12$



$+ 0.36 \cdot 10^{-5} = 0.0000036$

Ejercicio de programación simple usando modo Floating point .

Sumar A + B + C , Siendo A = 38, 7
B = 0,014
C = 1379

38,7 entra como | 0 | 3 | 8 | 7 | 0 | 5 | 2 |

0,014 " " | 0 | 1 | 4 | 0 | 0 | 4 | 9 |

1379 " " | 0 | 1 | 3 | 7 | 9 | 5 | 4 |

```
    90CAD100
100 0387052    A
101 0140049    B
102 0137954    C
193 00CAD100       A en el Ac o sea          38,7
104 00FAD101       Súmele en Floating B       0,014
105 00FAD101       Súmele  "     "    C     1379,000
106 00 W 000       Imprimo A+B+C =          1417,714
107 00STØP000
    90STØP000
```

Se puede observar que como los números están usándose con punto variable (Floating point) las órdenes de operación llevan una F para indicarle a la máquina el "modo" de aritmética que se está usando y que ella debe mantener el resultado en Floating point .

0	1	4	1	8	5	4	= 1418,00

porque aproximó las 714 milésimas 1417.714 al entero superior.

Si el entero resultante hubiera sido menor, habría alcanzado a dar dos enteros por ejemplo y dos decimales o bien los cuatro decimales si hubiera sido una adición de decimales pequeños.

Actividades sugeridas.

Los alumnos deben llegar a la comprensión del modo Floating point y para ello contiene ejercitar bastante, la codificación Floating y la descodificación. Ejemplo

1	0035	76	= $-0,0035 \cdot 10^{26}$ = -35000000000000000000000000

0	0015	53	= $+0,0015 \cdot 10^{3}$ = 1,5

0	0843	48	= $+0,0843 \cdot 10^{-2}$ = 0,000843

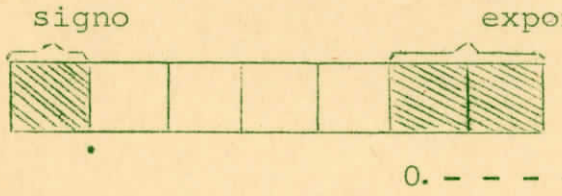
Ejercitación .

Se sugiere constantes usadas por el alumno o por lo menos conocidas en sus clases de ciencias .

Escribir correctamente según el modo Floating point las siguientes constantes.

1) constante de gravitacióm	6.67	$\cdot 10^{-8}$
2) carga eléctrica	4.803	$\cdot 10^{-10}$
3) masa del electrón	9.107	$\cdot 10^{-28}$
4) masa del atomo de hidrógeno	1.673	$\cdot 10^{-24}$
5) número de Avogadro	6.023	$\cdot 10^{23}$
6) constante de Planck	6.624	$\cdot 10^{-27}$
7) velocidad de la luz en el vacío	2.99776	$\cdot 10^{8}$

No olvidar el convenio dado para Floating point



Antes de entrar al Lenguaje Simbólico, es conveniente destacar que existen en la actualidad en nuestro ambiente, computadoras de tamaño y efectos reducidos que prestan gran utilidad en el área científica, tales como la **Hewlett** Packard - Modelo 9100 B que se encuentra en el C.P.E.I.P. , en la Universidad Técnica; en el Centro Nuclear, y en otros institutos de Santiago y la Olivetti 101, que permiten desarrollar Programación bastante elaboradas en Lenguaje Directo o Lenguaje de Máquina.

Esta primera parte de nociones de Computación podría visualizarse totalmente con la ayuda de una o viarias sesiones en que los alumnos, tuvieran acceso a su manipulación o vieran su manejo y utilización .

2 El sujeto programador .

Es el sujeto que va a dirigir la máquina; el que quiere resolver cierto tipo de problemas sencillos en nuestro caso .

Este sujeto debe poseer básicamente :

1) un conocimiento de la máquina y de como ella actúa ; vale decir:

a) Organización de la máquina

- Dónde ocurre la Entrada- cómo se hace-
- En qué consiste y cuál es la función de Memoria
- Qué hace la U. de Aritmética
- Cómo se realiza la Salida
- Cuál es la función de Control
- * - Cuál es la función del Programa traductor o Compilador de la máquina.

Este último punto se estudiará más adelante en relación con el Lenguaje de Máquina y los Lenguajes Simbólicos .

2) Debe conocer bien la naturaleza y los conceptos matemáticos del problema que pretende resolver.

3) El Diagrama de flujo o sea una técnica que permite analizar gráficamente los pasos del problema en relación a su solución completa.

4) El lenguaje de Programación.
Que será el Fortran en su forma elemental .

Los requisitos aquí señalados como 1 y 2 se supondran ya logradas y me referiré a continuación a los puntos 3 y 4 .

PREAMBULO

Al iniciar esta etapa de la explicación del aprendizaje de la Programación, me asaltó la siguiente duda. Existen dos técnicas que se ayudan mutuamente en la Programación: La Técnica de traducción del problema al Lenguaje Simbólico FORTRAN y la Técnica del Diagrama de Flujo que estudia graficamente los algoritmos del problema, dando de él una visión general muy valiosa.

Cuál de estas dos técnicas conviene explicar primero ?

La Diagrmación en Flujo, o Fluxograma ayuda a programar paso a paso, porque es un estudio lógico de las alternativas que ofrece el problema. Sin embargo, me parece que su aprendizaje requiere aplicación inmediata; de otro modo no es posible la apreciación total de su utilidad .

Esta propiedad me ha inclinado a pensar en el Diagrama de Flujo como una técnica a la que es posible sacar sus mejores frutos en Programación, después de "saber programar". Pienso por otra parte que el aprendizaje del Lenguaje FORTRAN tiene una gran cantidad de elementos que hay que considerar y que su aprendizaje total, requiere una elaboración compleja aunque se trate de sus niveles elementales. Y que debe tenerse ideas muy precisas de como funciona el FORTRAN al programar, antes de necesitar de la ayuda del Fluxograma para concretar la rogramación .

Es posible por otra parte que la Técnica de Diagrama ya sea conocida por los alumnos, a través de otros aprendizajes. En todo caso la secuencia que seguí es la siguiente: Lenguaje Fortran; y una vez que ya sea posible utilizarlo, hacerlo más funcional con la utilización de los Diagramas de Flujo .

LENGUAJE FORTRAN

Fortran es un Lenguaje simbólico que fue elaborado para trabajar con problemas y fórmulas. Por eso su nombre FORTRAN, es una sigla que quiere decir ("FORmula TRANslation") traducción de fórmulas.

El lenguaje de base del FORTRAN es el inglés y se han mantenido las expresiones inglesas como READ, STØP, GØ TØ, etc., por razones evidentes de la gran producción de máquinas computadoras que tiene U.S.A. y la gran difusión del idioma inglés en los niveles ténicos y cientistas.

Existen otros Lenguajes simbólicos: COBOL = (COMON BUSINESS ORDINER LANGUAGE) Lenguaje común de ordenadores comerciales .

ALGOL = Lenguaje general de computadoras aritméticas y van apareciendo otras variantes y simplificaciones de ellos : BASIC , APL, etc.

CARACTERISTICAS GENERALES

a) FORTRAN es un lenguaje simple

b) Las órdenes en FORTRAN son globales, es decir una órden como (A.B + C) : D se ejecuta completamente con una sola orden . Mientras que en Lenguaje de máquina se subdividiría la órden en otras cuatro:
   1. "traiga A al acumulador"
   2. "multiplique por B "
   3. "súmele C"
   4. "divida todo por D" .

c) Tiene una dirección simbólica .
   Quiere decir que en Memoria guarda, Variables o nombres alfaméricos que se "asignan" a las Constantes numéricas .

d) La aritmética también simbólica tiene dos formas de actuar: al "modo de enteros" y al "modo de reales" (con decimales). Y no está permitido mezclar modos; es decir trabajar en forma mezclada con enteros y decimales.

FORTRAN es un lenguaje con el cual se intenta dirigir la máquina computadora y por lo tanto tiene recursos muy similares a los estudiados en Lenguaje Directo o de Máquina. Deben existir ciertas órdenes para:

1) Entrada a Memoria de Datos
2) Procesamiento y resolver alternativas del procesamiento
3) Salida o impresión de resultados.

ARITMETICA EN FORTRAN.

1) Usa casi todos los recursos algebraicos conocidos .
2) Los símbolos de operatoria usual varían ligeramente, así tenemos :

   Para adición el símbolo es +
   Para sustracción el simbolo es -
   Para multiplicación el símbolo es # (asterisco)
   Para división el símbolo es /
   Para potenciación el símbolo es ## (2 asteriscos) .

3) Existe jerarquía en las operaciones .
   ## tiene la primera jerarquía para operar
   # y / tienen segunda jerarquía para operar
   + y - tienen tercera jerarquía para operar

4) Cuando hay operaciones que tienen igualdad de jerarquía, se hacen las operaciones de izquierda a derecha en el mismo órden en que se lee n.

5) Se usa paréntesis para alterar las jerarquías. Es decir si existen varias multiplicación y sumas, pero entre paréntesis hay una sustracción és... se ejecuta primero

6) Dentro del paréntesis si hay varias operaciones, también rige la jerarquía .
7) La aritmética no acepta la "coma" decimal siempre se usa "punto" .
8) No se aceptan dos signos de operatoria seguidos; a exepción de ## para potenciar.
9) La escritura es en una sola línea, es decir $\frac{A}{B}$ que va en dos líneas o $4^{\#\#2}$, no se aceptan . Lo correcto es escribir A/B ; 4 ## 2
10) Las expresiones algebraicas se escriben unicamente con mayúsculas.

ACTIVIDADES.

Ejercicios de aritmética FORTRAN.

A) Sobre jerarquía de las operaciones.
B) Escritura correcta de fórmulas comunes en Lenguaje FORTRAN.

A.- Jerarquía de operaciones.

a) 4 - 9 - 8 + 3 - 6 # 5 / 2
(1º y 2a jerarquía: deben efectuarse primero)

1º

2a.

30 / 2

"

15

4 - 9 - 8 + 3 - 15 = - 25

órden de izquierda a derecha

b) Ejercicio en que los paréntesis alteran el órden . Si no hubiera paréntesis debería empezarse por la multiplicación .

((14+5) - (7 - 2)) # 8 = (19 - 5) # 8
1º 2º

3º = 14 # 8

4º = 112

c) El paréntesis no está alterando el órden

$$\underset{1º}{(12 - 8)} \;\underset{2º}{/}\; 2 \;\underset{3º}{+}\; 79 =$$

4 / 2

2 + 79 = 81

d) El paréntesis altera el órden (El alumno deberá indicar el órden de la jerarquía en los siguientes ejercicios.)

12 / 2 / 2 # (6 - 3) = 9

e) (2 + 3 # 5 - (9/3)) - 32 = - 18

B.- Escritura de fórmulas comunes en FORTRAN

	arit. común		FORTRAN
1)	$\frac{ab}{c}$	=	A # B / C
2)	$\frac{a}{b \cdot c}$	=	A / (B # C)
3)	$k + \frac{l}{m+n}$	=	K + L/(M + N)
4)	$\frac{Ax + By}{\frac{c}{d} + e}$	=	(A # X + B # Y)/(C/D + E)

	arit. común		FORTRAN
5)	$2,718\ 281^{x}$	=	2.718 281 ## X

6) $\dfrac{a\ (b - c)}{b + \dfrac{c}{a}}$ = (A#(B-C)) / (B + (C/A))

Es conveniente el ejercicio inverso. Dar la escritura correcta en FORTRAN y traducirla a la operatoria común que debe ejecutar la máquina.

La escritura en el Lenguaje FORTRAN tiene una gran importancia porque un detalle, puede ser motivo de error y de impedir la solución de un problema. Este es uno de los aspectos negativos del Lenguaje Simbólico. Muchas veces un problema de gran elaboración "no corre" , es decir no sigue siendo resuelto, porque una letra, un espacio, una coma, aparecen mal codificados y la máquina no los acepta. Las reglas de escritura deben ser, pues, respetadas.

En FORTRAN es muy importante el correcto uso de la escritura de las Asignaciones de Constantes , y de Variables .

Las Constantes son numéricas,las Variables son nombres .

En FORTRAN las Constantes pueden ser

a) Constantes de enteros : 2; 8; 100; etc.
b) Constantes de reales : 3.14 ; 2.718 ; 0.5 etc.

Las Variables son nombres o palabras afmaméricos y también se clasifican como

a) Variables de enteros : Las que empiezan con las 7 letras mayúsculas I, J, K, L, M, N.

Ejemplo IMPAX
MIN14
LOLY88

palabras de 6 bits máximos

L	O	L	Y	8	8

b) Variables de reales: Las que empiezan con las otras 22 letras del abecedario.

Ejemplo AMORXX RIMPAX

PEPE

R	I	M	P	A	X

Observaciones respecto a variables: No deben tener más de 6 signos alfaméricos (números o letras) .

No pueden llevar ningún otro símbolo (sólo alfamérica)

Ejemplo:	Nombres Aceptables	Nombres No Aceptables
	RADIO5	CONT;4 SU-PRA
	PHILCO	LOLY! OVNI + 3
	INES	

En la Programación FORTRAN por causa de la existencia de fórmulas, continuamente existen combinaciones de letras y números.

Debe existir una cuidadosa relación entre constante entera y Variable entera y también entre constante real y Variable real.

Es decir debe entenderse esta coincidencia, en el sentido de"usar el mismo modo" o en el séntido de la contraindicación: "no usar modos mezclados".

Ejemplo a) si a la constante = 3,14 se le diera un nombre como PI = 3,14 o PINOMB = 3,14 la Asignación sería correcta porque PI empieza con P por lo tanto es un nombre de reales y 3,14 es un real.

b) si a la Constante entera 12 se le asigna un nombre de real (que empiece con cualquier letra que no sean las 7 de enteros) ROLEX = 12 , SIGMA = 12, etc. se estaría usando modos mezclados que la máquina comunmente rechaza .

I J K L M N se usan para nombres enteros , las otras letras del ABC para reales.

Indique que nombres son Reales, cuáles Enteros y cuáles no codificables (y por qué) .

ADA2 =
ADA,2 =
MA3TIL =
05J283 =
PI =
IPONA
JAPAN3 =
L;STUS =
4SIGMA =
OPI =
LMN =
PERATO =
BOTE88 =
AFOME=
FO-ME =
CAL =
+ . - =
8BETA2 =
CECIL3 =
ENRI4Q =
JAIMEO3=
UTILIE =
JUAN23 =
ITAL =
M =
8UPI =
B =

Asignaciones .

Cuando a un número entero (o real) le hacemos corresponder un nombre de entero (o real) esta es una Asignación.

El sentido que tal expresión tiene, es guardar el valor en memoria bajo el nombre señalado .

Ejemplo : EVALOR <— 2.718281
real real

"El número real 2.718281 guárdelo en Memoria, asignándole el nombre real EVALOR" .

La flecha indica que la Asignación se hace de derecha a izquierda (El signo= puede reemplazar la flecha <—)ya que el signo igual no tiene el sentido de igualdad sino el de una Asignación en Memoria .

Es por esto que Existen expresiones muy notables que se usan en computación como: $\boxed{N = N + 1}$.

Esta expresión no tiene sentido aritmético sino de Asignación o, una órden .

El significado de esta expresión es el siguiente: N es un nombre o variable de enteros . Debe existir una asignación anterior (N = 0 por ejemplo) y la órden que tratamos de explicar indica: Tome N = 0 de Memoria, súmele 1 y asígnelo a N nuevamente .

$$N = 0 + 1$$

$$\boxed{N = 1}$$

Si se ejecuta la órden $N = N + 1$, nuevamente tenemos

$$\boxed{N = N + 1}$$

$$N = 1 + 1$$

$$\boxed{N = 2}$$

En una Asignación el nombre de la izquierda debe ser uno solo, a la derecha es posible que exista una variable y que estén indicadas varias operaciones ejemplo :

REMI= 3.# A + P: Al valor asignado a A se le multiplica por 3 y se le suma el valor asignado a P. A todo esto se le asigna el nombre REMI.

Ejemplo de asignaciones especiales

a) A = - A está indicando que se debe cambiar de signo .

b) CAP= I Está indicando un cambio de modo: "Tome el entero I y transfórmelo en real asignándolo al nombre real: CAP.

Otra observación en cuanto a la aritmética.

La división entre enteros es muy inexacta por cuanto acepta solamente el cuociente por defecto ej .

4 : 3 = 1

6 : 4 = 1

7 : 4 = 1

15 : 4 = 3 etc.

por lo tanto debe preferirse el modo real al efectuarse cuocientes.

Ejercicios

En las operaciones siguientes clasifique el modo, y efectúe el cálculo asignándole el modo correctamente.

Ej.	modo	
7#3-8/3	entero	I = 19
7.#3.-8./3.	real	T = 18.4

	modo real o entero	resultado con la asignación correspondiente.
a) 4#10/8 =		
b) 4#19/5 =		
c) (3+7)/6 =		
d) 8.0#3.0/9.0 =		
e) 1./3.+1./3. =		
f) 2.#6.+1. =		
#g) (4.3+2.2)# #2 =		
h) 3#(10/4) =		
i) 5#(9##2) =		
j) (15#3)/12 =		
k) 2/3 =		

En el ejercicio q hay un ejemplo de modos mezclados y es el único caso que se acepta: "elevar a exponente entero" . En la elevación a potencia de base real o entera se sugiere usar siempre exponente entero para evitar un cálculo exponencial que haría recargar el trabajo de la computadora .-

Ejemplo :

Diferencia al usar un exponente entero o un exponente real en la elevación a potencia .

a) 3.92^{5} En este caso es posible iterar el producto
3.92 x 3.92 x 3.92 x 3.92 x 3.92 = | 925.61 |

b) $3.92^{6.8}$ En este caso hay que recurrir al cálculo logarítmico (logaritmo natural) .

1) $LN3.92^{6.8} = 6.8LN3.92$

2) $e^{6.8LN3.92}=$ | 10822.94 |

ØRDENES FORTRAN .

El programa de esta Unidad no tiene la ambición de dar el Lenguaje FORTRAN con todo su desarrollo y amplitud. Muy por el contrario, pretende entregar un FORTRAN simple, estrictamente operable para el nivel de 4º año de Media, sobre la solución de problemas que los alumnos sean capaces de desarrollar .

Esto es posible tomando las órdenes absolutamente necesarias para operar . Tales son :

I Ordenes de Entrada o Imput

READ

FØRMAT

II Procesamiento que incluye la Aritmética y el cálculo de Funciones.

Aritmética

+	Adición
-	Sustracción
#	Multiplicación
/	División
# #	Potenciación

Funciones : SQRT
SIN
CØS
ATAN
EXP
ALØG
ABS

III Transferencia de Control

IF (N) < , = , >	Salto condicionado a tres alternativas menos que 0 (<) igual a 0 (=) mayor que 0 (>)
GØ TØ	Salto incondicional

IV Salida o Output
WRITE
FØRM\T

V Fin de Programa
STØP
END

Aunque podría haber explicado las órdenes en la misma secuencia en que aquí se han agrupado, voy a preferir dar primaramente las de Transferencia de Control, que aún cuando tienen semejanza al CU y al CNZ viste en Lenguaje de Máquina merecen un tratamiento especial; ellas son IF y GØ TØ. Vimos con anterioridad el concepto de Programa Almacenado, que permite la Programación secuencial de órdenes y datos para lograr que ellos actúen efectuando un procesamiento automático. Sin embargo la misma agilidad del mecanismo obliga a saltar secuencias para evitar repeticiones o el revés a repetir un mismo procesamiento gran número de veces.

Las instrucciones que logran esta variante en la secuencia general son IF para "saltos condicionales". GØ TØ para salto incondicional .

Declaración IF. Esta órden permite discriminar entre tres alternativas : menor que 0
igual a 0
mayor que 0

En otras palabras frente a cierto valor (una asignación) o una diferencia entre dos asignaciones pregunta como es esta diferencia con respecto a 0; y la respuesta "condiciona" su salto que rompe la secuencia regular .

IF va seguido de un paréntesis, IF ( ) ; en este paréntesis está el número cuestionado o la diferencia cuestionada.

IF (Z)
IF (P - Q)
Esto quiere decir Si Z) es .....
Si (P - Q) es .....

y a continuación de la declaración deben ir tres números cualesquiera en órden o sin él separados por dos comas solamente. Algunas veces estos números se repiten. Ellos indican si el número cues -

tionado es < , = ó >

Ejemplo

	Significado de los números
1) IF (Z) 3,2,7	Si Z < 0 salte a la secuencia 3 y ahí cumpla órdenes Si Z = 0 salte a la secuencia 2 y ahí cumpla órdenes. Si Z > 0 salte a la secuencia 7 y ahí cumpla órdenes.
2) IF (z) 8,9,11	8) Si (Z) es menor que 0, control debe continuar en la secuencia que se encabeza con 8; (ahí recibe orden de cambiar el signo)
8 Z =-Z	
9 WRITE (Z)	9) Si (Z) = 0, Control salta a la secuencia que se encabeza con 9 (ahí imprime Z).
11 Z = Z - 3	11) Si (Z) es mayor que 0 Control salta a la secuencia 11; y resta 3 a Z y lo asigna todo a Z .
3) IF (P-Q) 3,3,4	Si (P-Q) < 0 } vaya a 3 a cumplir órSi (P-Q) = 0 } denes Si (P-Q) > 0 vaya a 4
4) IF(R) 5,6,5	
Dos alternativas	Si R< 0 } Salte a la secuencia 5 y Si R >0 } ahí cumpla órdenes: "imprimir , detenerse" .
R ≠ 0	
R = 0	Si R = 0 Salte a la secuencia 6 . y ahí cumpla órden: "incremente R en 1 y asígnelo a R".
5 WRITE (R) STØP	{ Imprima y Deténgase
6 R = R + 1	Sumele 1 a R y guárdelo en R .

Declaración GØ TØ

Es una órden de salto incondicional a determinada secuencia del programa. Es una órden muy simple GØ TØ tiene una variedad de usos .

a) En combinación con IF

. Ejemplo

```
    IF (X) 6,3,6,
 6  GØ TØ 20
 3  GØ TØ 15
    |
    |
    v
15
```

6)(Si X ≠ 0) "vaya a" la secuencia 20.

3)(Si X = 0) "vaya a" la secuencia 15 .

b) Para evitar un STØP (detención en medio de programa)

Ejemplo: GØ TØ 9 en mitad del Programa .

Implica que Control transfiere la órden a la secuencia que lleva número 9 . En 9 se encuentra la órden de STØP o sea término de programa,pero para esa alternativa solamente; el programa continúa desarrollándose para otras alternativas.

c) En general GØ TØ es una órden que da mucha agilidad al programa y que permite cortar una secuencia, retroceder para efectuar cálculos repetidos, avanzar evitando otras secuencias que no conviene realizar.

Estas dos órdenes son facilmente asimilables con la práctica y a través del Diagrama de Flujo se entenderá mejor su eficacia .

Ordenes de Procesamiento .

Son de una simplicidad extrema y obedecen a la operatoria común .

Si en una secuencia se encuentra A + B, tal operación se realiza y el valor de la adición se asigna a un nombre C .

C = A + B sería la órden correcta.

Del mismo modo se procede en las diferentes operaciones o en la combinación de ellas.

M = (I ## K)/36 # L/2 Una vez efectuado completamente este cálculo, la computadora asigna su valor a M y puede continuar nuevos procesamientos en que intervenga M.

Cálculo de Funciones.

Las Computadoras electrónicas tiene ciertos pequeños programas ya desarrollados para las distintas funciones. Así la órden Y = SQRT(A), implica $\sqrt{A}$ que se asigna a Y. Se debe tener por lo tanto una asignación previa para A. Ejemplo: A = 17.64, la órden Y = SQRT(A) implica que la computadora entrega la asignación Y = 4.2

De manera similar se efectúa el procesamiento de valores trigonométricos y logaritmicos .

Es importante destacar que al hablar de secuencia no se trata sin embargo de una secuencia numerada como ocurría en el Lenguaje de Máquina en que cada órden estaba precedida de un número
100
101
102
103, etc.

Las órdenes en Lenguaje simbólico no están numeradas secuencialmente, sino que el programador da una secuencia escrita de órdenes; y entre ellas dá números elegidos al azar para aquellas secuencias que lo necesiten. Si se usa un GØ TØ para volver a un paso anterior bien determinado, el programador da a ese paso un número, por ejemplo 15, y cuando desea regresar a esa secuencia indicará GØ TØ 15 .

Sin embargo existen dos declaraciones que siempre van precedidas de número, son las declaraciones de especificación que acompañan a la Salida y Entrada de datos, los FORMATOS .

La órden STØP , es una órden cuyo significado es evidente : "Deténgase " .

Si en el programa es necesario enviar a fin de programa con un GØ TØ. En ese caso la declaración STØP debe llevar número Este número es arbitrario, supongamos que sea 65. La órden que inicia el proceso de detención de la computadora es GØ TØ 65 donde se encuentra la órden de STØP y se detiene la máquina . No obstante cuando STØP está en secuencia normal del programa no lleva ninguna numeración .

La órden END indica "fin de programa" y siempre sigue a la órden STØP.

Antes de revisar las órdenes de Entrada y de Salida que exigen Formato, veremos algunas declaraciones que son compuestas y que no están determinadas en el Lenguaje elemental

Sumas parciales acumuladas.

Para efectuar este procesamiento se parte de una Suma acumulada 0 y se establece la siguiente declaración: S=S + P

Esta no es una igualdad, es una asignación en que P es variable .

S a la izquierda del igual es la última suma

S a la derecha es la suma anterior que se suma a la variable .

Ejemplaricemos: Se quiere sumar los números naturales a partir de 4.

S = 4 + 5 + 6 + 7

P = 4 S = 0 de partida $\boxed{S = S + P}$

Veamos S = S + P

P = 4	S = 0 + 4 ——→ S = 4	la primera suma
P = 5	S = 4 + 5 ——→ S = 9	la segunda suma
P = 6	S = 9 + 6 ——→ S = 15	la tercera suma
P = 7	S = 15 + 7 ——→ S = 22	la cuarta suma
		etc.

En el procesamiento siempre se recurre a fórmulas de asignación como éstas.

Otros ejemplos :

N = N + 1

K = K • L (Con K inicial = 1) .-

Entradas y Salidas en FORTRAN .

La Entrada o input se refiere a la alimentación de la computadora mediante información externa que llega a la LECTORA.

Esta lectora recibe tarjetas perforadas.

La tarjeta.

La tarjeta es rectangular con la punta superior izquierda cortada.

Esta tarjeta permite una codificación de las palabras y los números décuplos.

Tiene 80 columnas nemeradas de izquierda a derecha y en cada columna de arriba abajo los números de 0 a 9 .

En la parte superior el ebecedario con 26 letras. No se usa la Ñ ni letras dobles o compuestas. El abecedario ocupa 51 columnas y esta dividido en tres zonas de la A - I (zona 12(; J - R (zona 11)de la S - Z (zona 0).

ABCDEFGHI JKLMNOPQR STUVWXYZ

zona 12

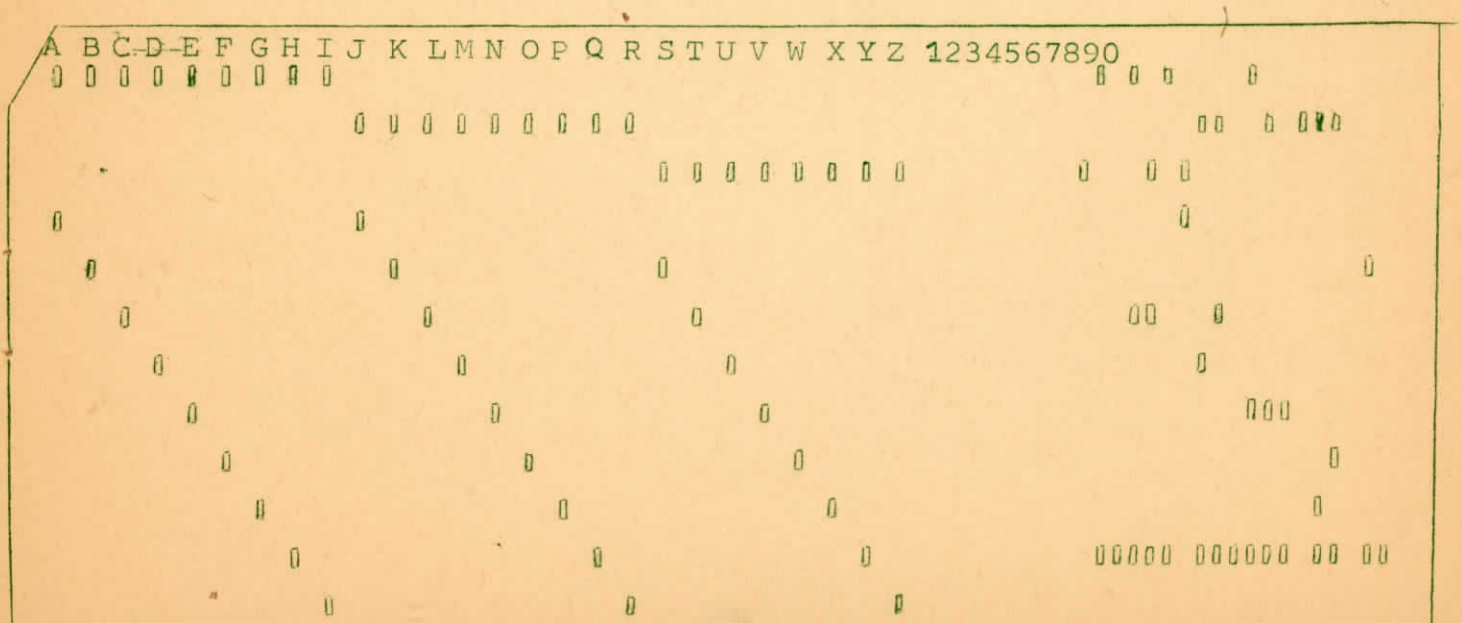
zona 11

zona 0

Como vimos en la pag 6, las zonas son utilizadas para codificar cada letra y formar palabras, así a la letra A le corresponde zona 12 y la codificación del dígito 1 o sea

A
12
1

De la columna 53 a 63 están ubicados los dígitos 1 al 9 , y el 0; y de 63 a 79 un grupo de símbolos + =.,</*$"<>"_¬;



HOJA PROGRAMACION FORTRAN

Programa

Programador Fecha

Página de

Identificación

C PARA COMENTARIOS

Instruc. Nº | Cont. | INSTRUCCION FORTRAN

1 5 6 7 10 15 20 25 30 35 40 45 50 55 60 65 70

Hoja de Programación FØRTRAN .

Tiene al igual que la tarjeta 80 columnas ya que a cada línea de programación corresponde al contenido de una tarjeta .(La hoja que damos tiene sólo 72 columnas) .-

La columna del 1 al 5 sirven para escribir el número de la declaración cuando necesite llevarlo y el número correspondiente a Formato en la Entrada y Salida de datos.

La columna 6 numera la continuación de una línea indicando en ella 1, 2, 3, la secuencia de la información en diferentes tarjetas.

En la columna 7 se empieza a escribir el programa y en cada línea es posible escribir hasta la columna 72.

Las columnas 73 al 80 se ocupan para que el programador escriba información especial o numere el órden de tarjetas.

Comentarios: Son antecedentes para el Programa.

El programador debe iniciar el programa, indicando el contenido, dando su identificación personal y añadiendo algunos otros comentarios todo esto a partir de la columna 7. Aún cuando cualquier comentario debe estar presedido por una C en las columnas entre 1 y 5 .

Entrada de datos .

Supongamos que en cierto problema se va a operar con tres números reales.

Los datos deben entrar a Memoria como reales y por lo tanto deben estar asignados a nombres de reales, por lo tanto una buena declaración para entrada de tres reales sería :

READ ( ) ALFA, BETA, GAMA

Esta declaración es incompleta.

En el paréntesis debe indicarse un número que designa con qué lectora se lee; y separado con una coma otro número que especifica con que Formato entran los números reales es decir (cuántos decimales.)

READ (1,27) ALFA, BETA, GAMA

Esta declaración es todavía incompleta. Pero ya dice bastante. Su sentido sería: "Lea, con (1) lectora de tarjetas los tres números reales asignados a ALFA, BETA, GAMA; los leerá con el RORMATO que se indica en la secuencia Nº 27 .

Es evidente que falta el formato.

a) La declaración correcta (para entrar a la computadora tres números reales a través de una lectora de tarjetas es ;

```
1 2 3 4 5 6 7
C       PRØGRAMA  JUAN SOTO
C       ARITMETICA
C
        READ(1,27)ALFA,BETA,GAMA
    27  FØRMAT(F8.3,F8.2,F10.3)
```

Analicemos el formato : FØRMAT ( )

FØRMAT va presedido del número 27 en la columnas 4 y 5 (o sea en las columnas para indicar comentarios o números de declaraciones).

Sigue a la declaración FØRMAT un paréntesis dentro del cual se indican tres números reales :

F 8.3 significa : número real(esto lo indica la presencia de F) escrito en 8 columnas con tres decimales en las últimas tres columnas de las 8 totales.

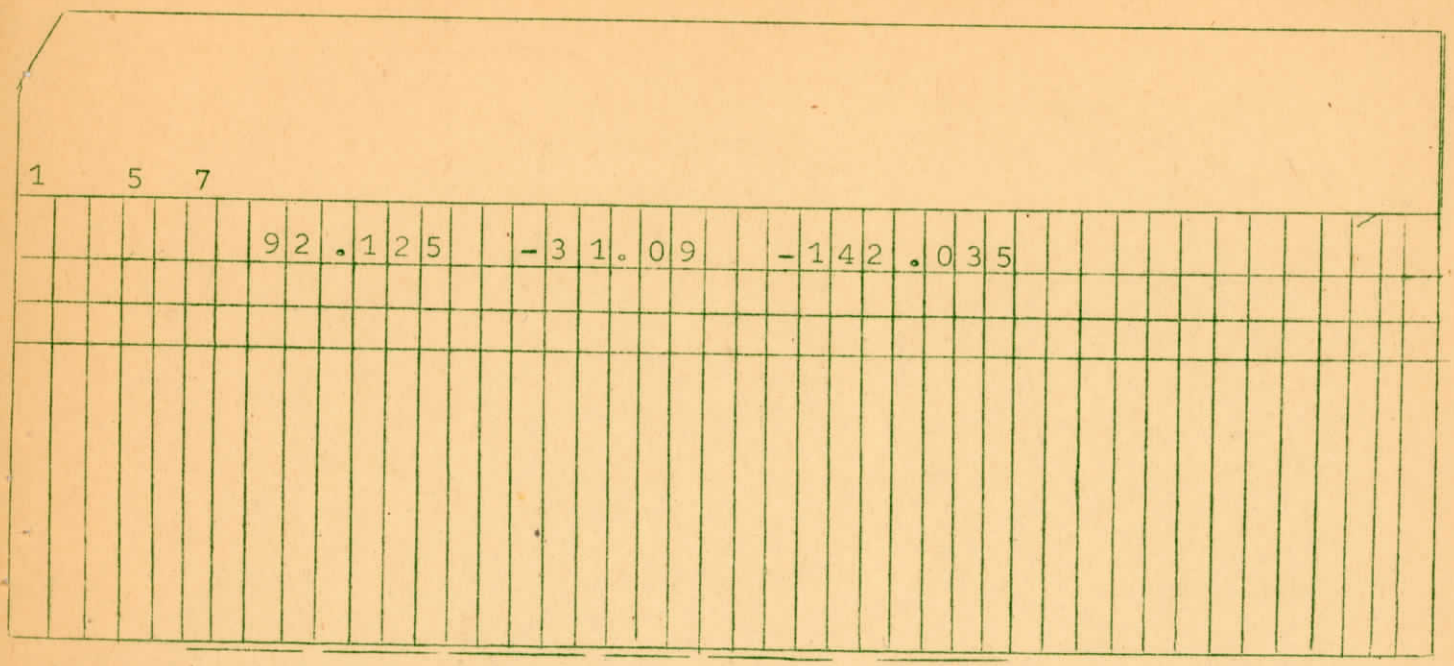
F 8.2 significa : número real escrito en 8 columnas estando las decimales en las 2 últimas.

F 10.3 significa : número real escrito en 10 columnas estando los tres decimales que tiene en las tres últimas columnas.

La comprensión de esta entrada se aclara más si concretamos el ejemplo, dando los valores de los tres números reales que se eben entrar.

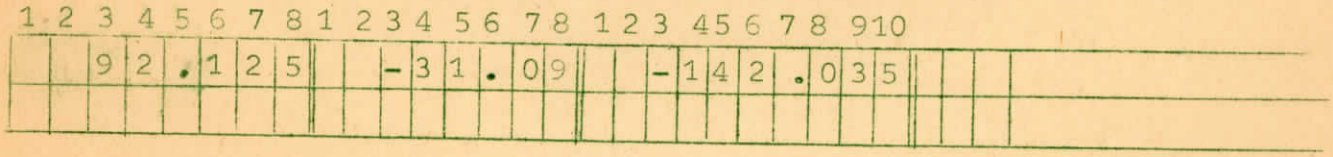
Ejemplo 92.125 , -31,09 , - 142.035

Estos tres números pueden entrarse en tres tarjetas perforadas o bien los tres en una misma tarjeta. Estamos dando el formato en el último caso.



(Este gráfico solo tiene valor de explicación, el cuadriculado no corresponde completamente al de la tarjeta.)

Generalmente en el extremo inferior de la hoja de programación se incluyen los datos, en esta misma forma .



b) Otro formato para el mismo problema .

```
    READ(1,50)A,B,C,
50  FØRMAT(F8.3/F8.2/F10.3)
```

Los slash (/) en el FØRMAT indican el uso de otra tarjeta por lo tanto la lectura y la entrada se haría en este caso en tres tarjetas distintas una para el dato A otra para B y otra para C .

Respecto al número del FØRMAT es completamente libre; la elección la hace el programador. Su sentido es que la máquina ubique inmediatamente la tarjeta de FORMATO cualquiera que sea este número .

Observaciones:

1) FORMAT siempre lleva número de declaración .
2) Los números de declaraciones no deben repetirse nunca.
3) Una declaración lleva número sólo si es necesario .

Compare las dos entradas y observe que en el caso de querer separar los datos en distintas tarjetas basta colocar líneas oblicuas o slash (/) entre las especificaciones del campo para los distintos datos.

c Todavía una tercera posibilidad para la misma entrada.

```
    READ(1,27)A,B,C
27  FØRMAT(2F8.3,F10.3)
```

tomando el mismo campo 8 columnas y los mismo decimales para los datos A y B los dos primeros y otras especificación distinta para C .

d) Por último pudieron entrarse todos los datos con 8 columnas y 3 decimales, o sea :

```
    READ(1,27)A,B,C
27  FØRMAT(3F8.3)
```

Una situación diferente de entrada hace variar el formato si los datos son números enteros y también la asignación de nombres . Ejemplo : datos 1786 , 42 , - 36002

La máquina deberá ser avisada de que operará con números enteros y para ello los números deben asignarse a nombres de enteros .

A cada dato en el READ se asigna un nombre de enteros como JØTA, MIN, LACA y el formato reconoce los nombres enteros con I. La entrada correcta podría ser .

a)

```
      READ(1,18)JOTA,MIN,LACA
18    FØRMAT(I6,I4,I8 )
```

en una misma tarjeta los tres datos con campos JØTA de 6 columnas MIN de 4 columnas , LACA de 8 columnas.

I6 significa un número entero(esto indica la I)escrita en 6 columnas.
I4 significa un número entero con 4 columnas
I8 significa un número entero en 8 columnas.

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 1| 7| 8| 6|  |  | 4| 2|  |  | -| 3| 6| 0| 0| 2|  |  |  |  |
```

b)
```
      READ(1,24)JØTA,MIN,LACA
   24 FØRMAT(I6/I4/I8)
```
entrada en tres tarjetas diferentes (cada dato en una tarjeta aparte).

c)
```
      READ(1,49)JØTA,MIN,LACA
   49 FØRMAT(3I8)
```
Entrada de tres enteros con tres campos de 8 columnas ; los tres en una misma tarjeta.

Al hacer una entrada de datos algunos pueden ser enteros y otros reales, no es necesario hacer una separación en la entrada, pero la declaración debe estar correctamente especificada.

Ejemplo : Entrar los siguientes datos :

π , 294 , $\sqrt{2}$, 68.20 , 129

PI = 3.1415
R = 1.4142
A = 68.20

PI, R , A nombres de reales

K = 294
L = 129

K.L nombres de enteros .

Entrada :
```
      READ(1,68)PI,R,A,K,L
   68 FØRMAT(2F8.4,F8.2,2I5)
```

Salida de información (datos o soluciones)

La salida o output es semejante a la entrada esta formada de dos partes : WRITE Y FØRMAT

```
WRITE(        )
```

La declaración WRITE va seguido de un paréntesis que lleva dos números separados por una coma (,) el primero indica con que se imprime; existiendo, impresora, cinta magnética, disco, etc., La impresora que es la más usada tiene una numeración varia-

ble según el equipo. Nosotros usaremos el número 3, El segundo número es arbitrario y designa el FORMATO

Ej.1) WRITE(3,55)SIGMA

Imprima un real , SIGMA,con la impresora (3) y con el formato que se le indica en 55 .

Ej.2) WRITE(3,91) X .

Imprima con la impresora el valor de X, en la forma que se le indica en la declaración 91 .

Dando una situación concreta para los dos ejemplos:

Supongamos que se ha resuelto un cálculo de promedio y el resultado es un número real 18.6, con la asignación SIGMA = 18.6 . Este valor se desea imprimir

a) WRITE(3,22)SIGMA
22 FØRMAT(1H1,15X,F6.2)

En el FØRMAT de salida hay dos codificaciones nuevas en el paréntesis: 1H1 quiere decir "empiece a imprimir en página limpia"

15x que quiere decir, al imprimir salte 15 espacios a partir de la izquierda (margen) .

F6.2 (ya no es una expresión nueva)
quiere decir "imprima el real en 6 columnas con dos decimales en las dos últimas columnas.

b) SIGMA = 18.6 NX = 20

WRITE(3,58)NX , SIGMA
58 FORMAT(1Hb,10X,I6,F6.2)

Toda la declaración de salida de este ejemplo tiene la siguiente traducción :

"Imprima dos valores: uno con contenido entero (NX) y el otro real (SIGMA) según se el indica en el formato con número 58" .

La especificación del formato 58 es la siguiente: "Empiece a imprimir en página nueva (1Hb o 1H1 es lo mismo), dejando un margen de 10 casilleros; a continuación el primer resultado escrito en 6 columnas es entero y a continuación 6 columnas para el real dejando las dos últimas para sus decimales.

Otro ejemplo :

```
      WRITE (3,19) L,K,A,B,C,D
19    FØRMAT(10X,2I8,4F10.3)
```

Imprima dos enteros y cuatro reales sin tomar página nueva, dejando un margen de 10 espacios .

```
      WRITE(3,16) L,K,A,B,C,D
16    FØRMAT(1H1, 20X, 2I8/20X,4F10.3)
```

Imprime en páginas nuevas con margen 20 espacios 2 enteros .

El slash (/) en "salida" indica otra línea por lo tanto, salta a otra línea la impresora deja un margen de 20 espacios e imprime 4 reales .

Impresión de títulos .

Cuándo se quiere imprimir una frase , algunas máquinas eceptan la impresión si la frase se escribe entre comas.

Otras exigen el uso de la forma Hollerith ; consiste en indicar el número de espacios que comprende la frase seguida de una H .

Ejemplo a)

```
      WRITE (3,28) Z
28    FØRMAT (1H1, 'LA SUMATORIA ES', F7.3 )
```

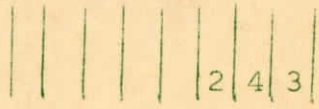
b)

```
      WRITE(3,28)Z
28    FØRMAT (1H1, 15HLA SUMATORIA ES , F7.3)
```

Información

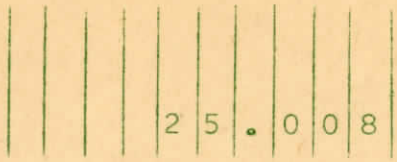
I 8 significa: "ubicar un entero en 8 columnas"

Ejemplo: 243



F 10.3 significa: "ubicar un real en 10 columnas con tres decimales " .

Ejemplo: 25.008



Ejercicios:

1) Codificar en una hoja de programación

3 enteros	I6,	I15,	I2,		
5 enteros	I8,	I4,	I7,	I9,	I3
7 reales	F9.2	F8.1	F6.3	F15.3	F5.0
	F7.2	F10.3			
4 reales	F12.4	F15.0	F12.3	F9.4	

2) Escribir correctamente una declaración de impresión para 5 datos, 3 de ellos enteros. Se desea que la impresión sea en página nueva y esté titulada en la siguiente forma :

DATOS REALES DATOS ENTEROS

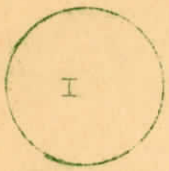
Observación :

Las declaraciones en FØRTRAN no permite espacios libres porque en la Hoja de Programación, cada letra, cada número, cada signo ortográfico o aritmético, va en un casillero aparte.

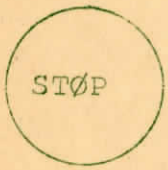
Al escribir estos apuntes no ha sido posible siempre mantener esta regla ya que falta el cuadriculado .

Diagrama de Flujo .

Signos gráficos convencionales .



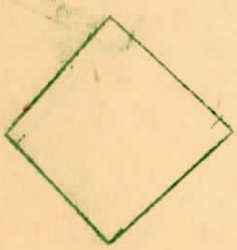
inicio del diagrama



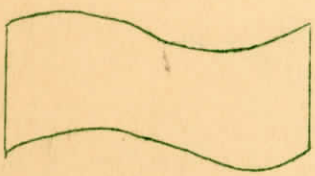
fin del diagrama

lectura de datos

operatorias o asignaciones.



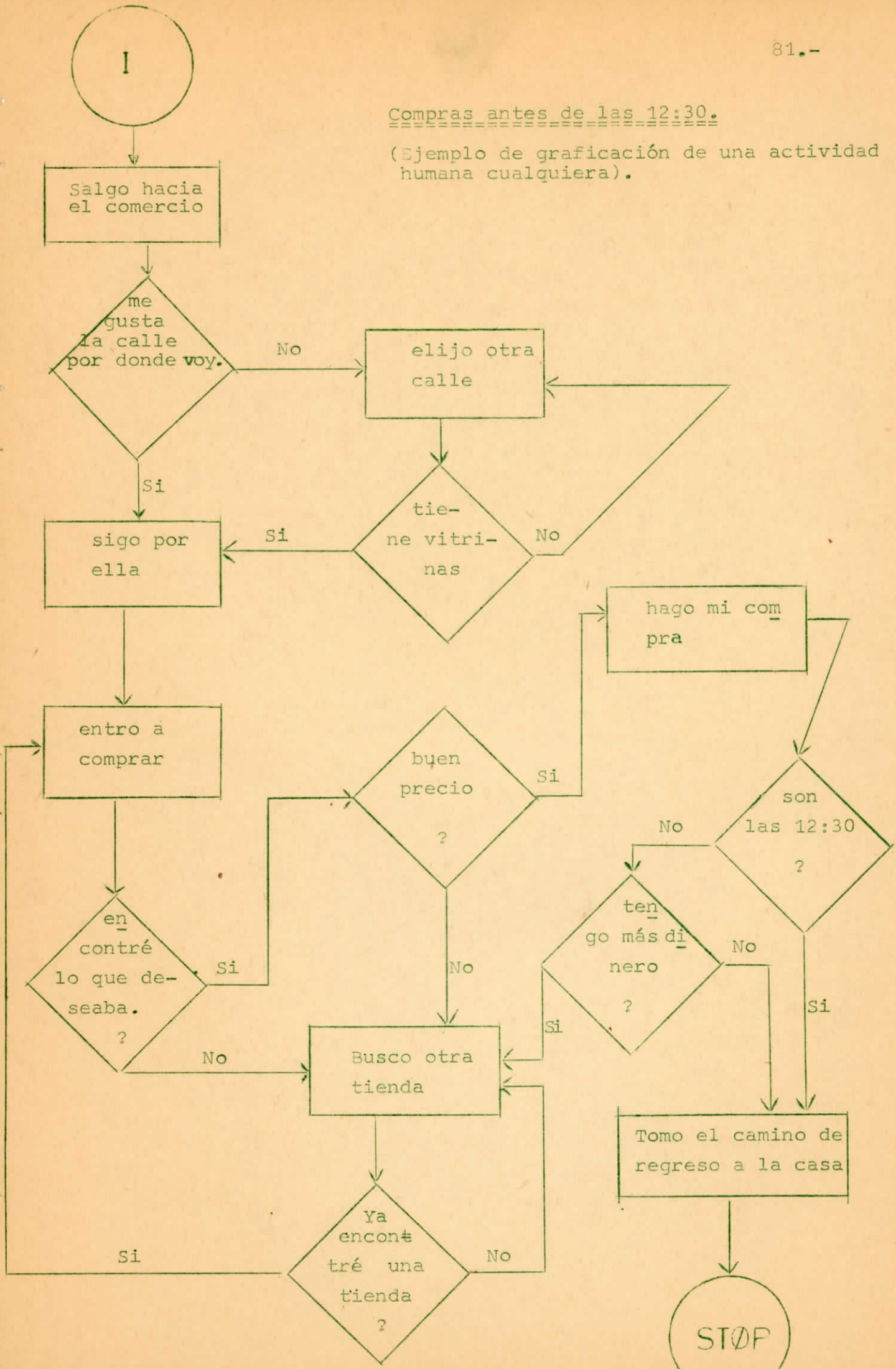
pregunta para solucionar alternativas.



impresión de valores.

Compras antes de las 12:30.

(Ejemplo de graficación de una actividad humana cualquiera).



Como observamos en el diagrama, todas las actividades humanas requieren de una serie de pasos para llevarse a cabo. Y en este proceso se presentan situaciones que obligan a cambios de actitudes. El ser humano usa su capacidad, su criterio para tomar decisiones y resuelve de acuerdo a las circunstancias y conveniencias personales o colectivas, las alternativas contingentes, tratando de no perder los objetivos que se fijó al iniciar la obra .

Un problema es una situación que ofrece alternativas antes de llegar a su solución. Como una máquina no es capaz de resolver frente a una situación nueva, el hombre que dirige a la máquina debe prever estas alternativas y entregar una secuencia completa de pasos que incluyen lo que hay que hacer en cada situación esto es lo que persigue un Diagrama de Flujo .

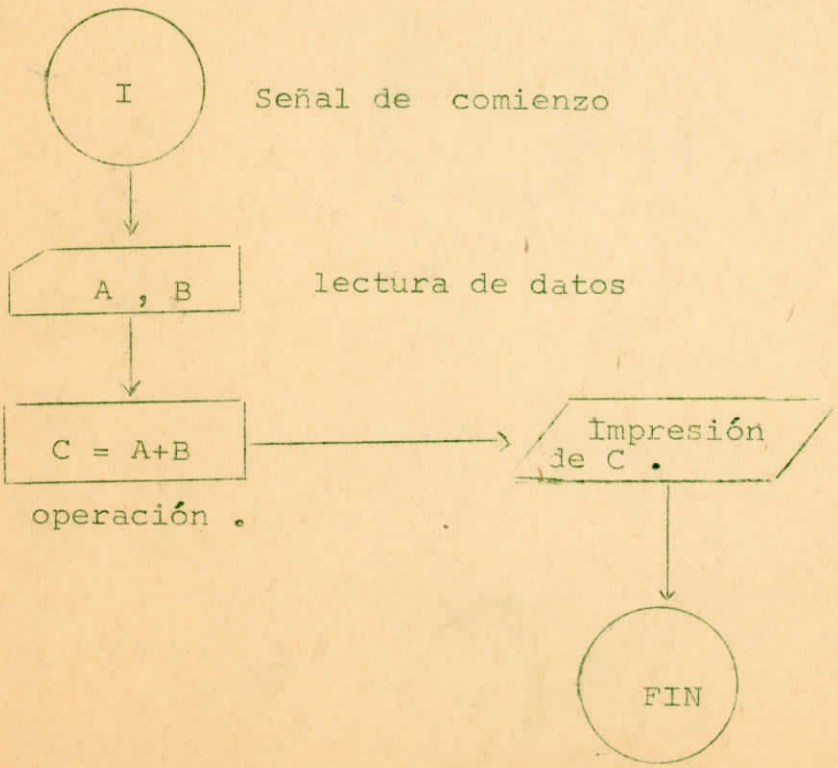
Ejemplo: Solución de un problema elemental .

El problema elemental podría ser la adición de dos datos. A,B. (reales) .

Secuencia .

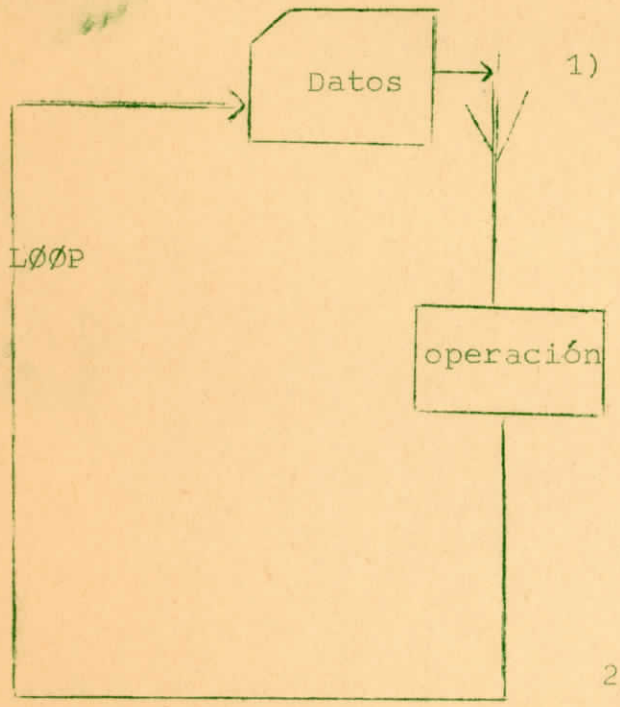
1.- Se leen los datos (entrada) A, B
2.- Se suman los datos A + B
3.- Se imprime el resultado C = A + B

Un esquema trivial de la secuencia.



Una observación interesante es la siguiente ; una operación tan simple como sumar dos datos; no es propia para usar una computadora, pero sí lo es si se necesita hacer 1000 sumas, ya que la computadora ahora, en este caso, al hacerlo electronicamente ahorra tiempo humano y energía humana .

Para que la computadora ejecute una operación repetida como la descrita es decir computar la adición de 1000 sumas de dos datos cada una, es necesario darle las instrucciones para que haga la primera adición y para que realice una repetición de la operación cambiando de datos estos constituye un Ciclo o LØØP .



1) Se ejecuta una operación

LØØP

2) se regresa a buscar nuevos datos para repetir la operación .

Aclaración del Concepto de Diagrama de Flujo .

La necesidad de resolver el problema de la adición de 2.000 datos para organizarlos en 1.000 sumas cuyo resultado se necesita conocer, nos llevará a hacer un diseño lógico del problema, tomando todas las alternativas .

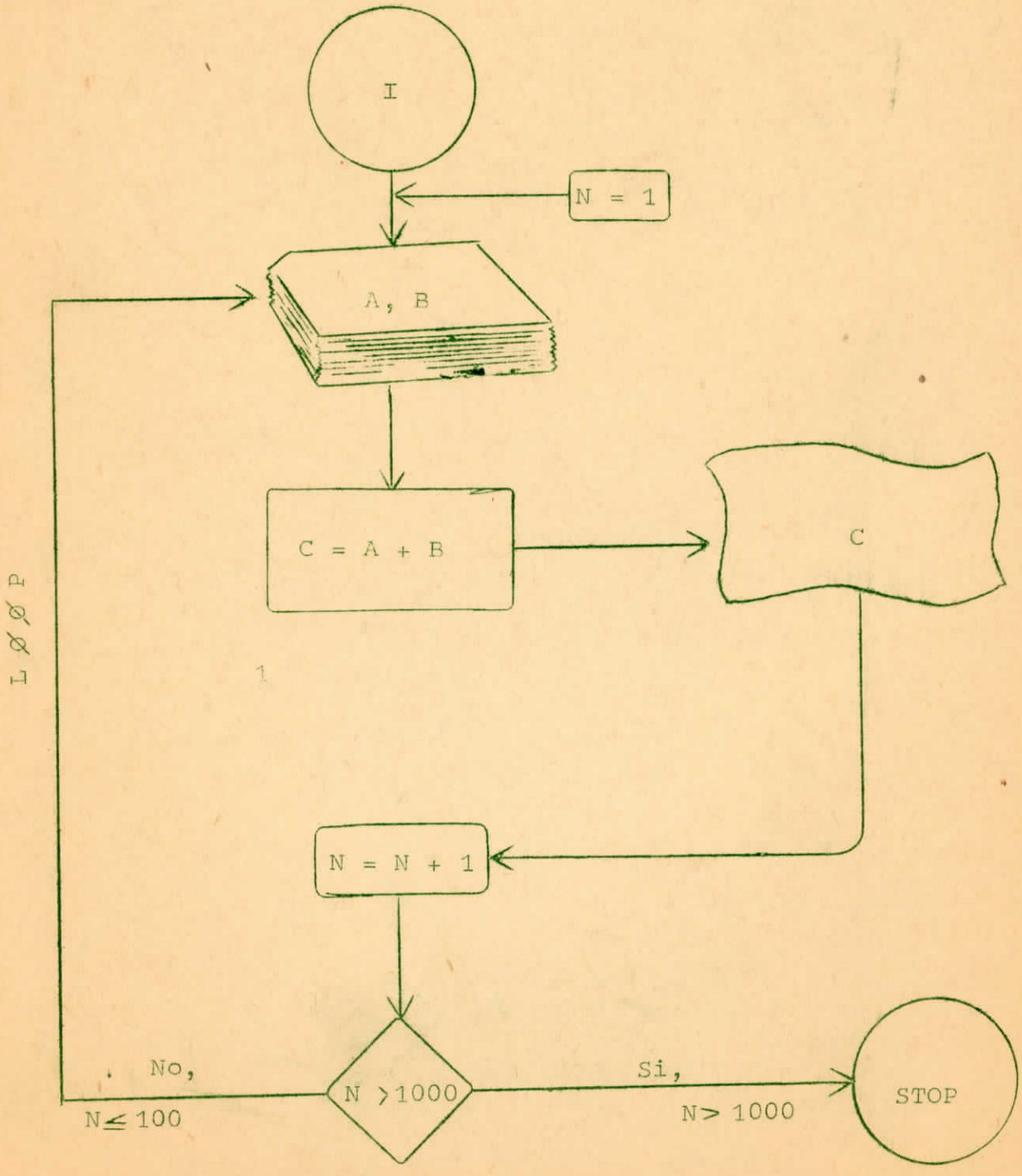
Esto es una primera aproximación al concepto de Diagrama de Flujo .

Podríamos ya decir, que el Diagrama de Flujo usa ciertos símbolos gráficos para hacer una descripción de procedimientos cuando se realiza algo; y si tenemos conciencia de que estas instrucciones las vamos a dar para una máquina "que no tiene la posibilidad de preguntar", tenemos que describir el procedimiento paso a paso desde el inicio al fin del proceso, evitando cualquier ambigüedad en la descripción y cubriendo todas las alternativas que puedan presentarse al realizarlo. Esta es la definición de un algoritmo. Entonces el gráfico, el dibujo de un algoritmo es lo que se llama una Diagrama de Flujo.

En la página siguiente veremos el Diagrama de Flujo que corresponde al problema de repetir 1000 sumas variando los datos . Se incluye un "contador" N que indicará cuando se realizan 1000 vueltas en el LØØP .

Diagrama de flujo para resolver el problema planteado anteriormente.

Datos 1000 tarjetas, con 2 datos cada una



Explicación del Diagrama

En el esquema aparece una asignación ajena al problema N = 1, este elemento actúa como contador para la máquina. En cada vuelta va variando: N=2, N=3, etc, y conjuntamente con la alternativa motivada al preguntar si N > 1000 ? promueve él LØØP por una parte y permite salir de el terminando el problema .

Revisión del proceso .

1.- La computadora recibe una órden de asignación N = 1 .
2. La lectora debe leer los datos A y B de la primera tarjeta.
3. Suma los valores de A y B .
4.- Imprime el resultado de la operación C = A + B
5.- Efectúa un incremento de N. Es decir al valor de N = 1 le incrementa en 1 y la nueva asignación es N = 2 .

La posibilidad de preguntarse a sí misma que tiene la computadora es aritmética; y en tres alternativas
N - 1000 será el valor cuestionado en cada vuelta y
puede ser menor que 0 y la diferencia es negativa ;
puede ser igual a 0 y la diferencia es 0;
puede ser mayor que 0 y la diferencia es positiva.

En el caso concreto 2 - 1000 = - 998 (es negativa) .

Pero si la máquina incrementa N en cada vuelta del LØØP hasta llegar a 1000 en este caso 1000- 1000=0 y en un nuevo incremento 1001 - 1000 = 1 positivo .

Veamos que ocurre: Mientras el número N sea menor o igual a 1000, Control envía a la lectura de una nueva tarjeta; suma los datos que viene en ella, los imprime, se efectúa un nuevo incremento de N = N + 1, se compara nuevamente con 1000 y se va por otra tarjeta con nuevos datos .

Este LØØP se repite hasta que N sea 1001 o sea se repite 1000 veces, y cuando tal ocurre se sale del LØØP porque al ser 1001 > 1000 Control termina el problema en STØP END .

Ahora se ha resuelto la lógica del problema es decir se aplica teóricamente un algoritmo a través de una planificación "gráfica" que es el Diagrama de Flujo .

La máquina deberá seguir estos pasos, me digo a mi misma. Pero debo añadir: "siempre que la máquina entienda el plan que le he señalado para cada paso y alternativa del problema ".

Bien la máquina me entenderá si ella está programada con lenguaje FØRTRAN, por ejemplo, y yo le escribo un programa ceñido al Diagrama de Flujo y también en lenguaje FØRTRAN.

Es absolutamente verdadero lo que estoy afirmando? No, sólo encierra una verdad relativa, puesto que la máquina lo que entiende realmente es un lenguaje electrónico de paso de corriente o interrupción del circuito, es decir, su idioma es binario y es el Lenguaje de Máquina .

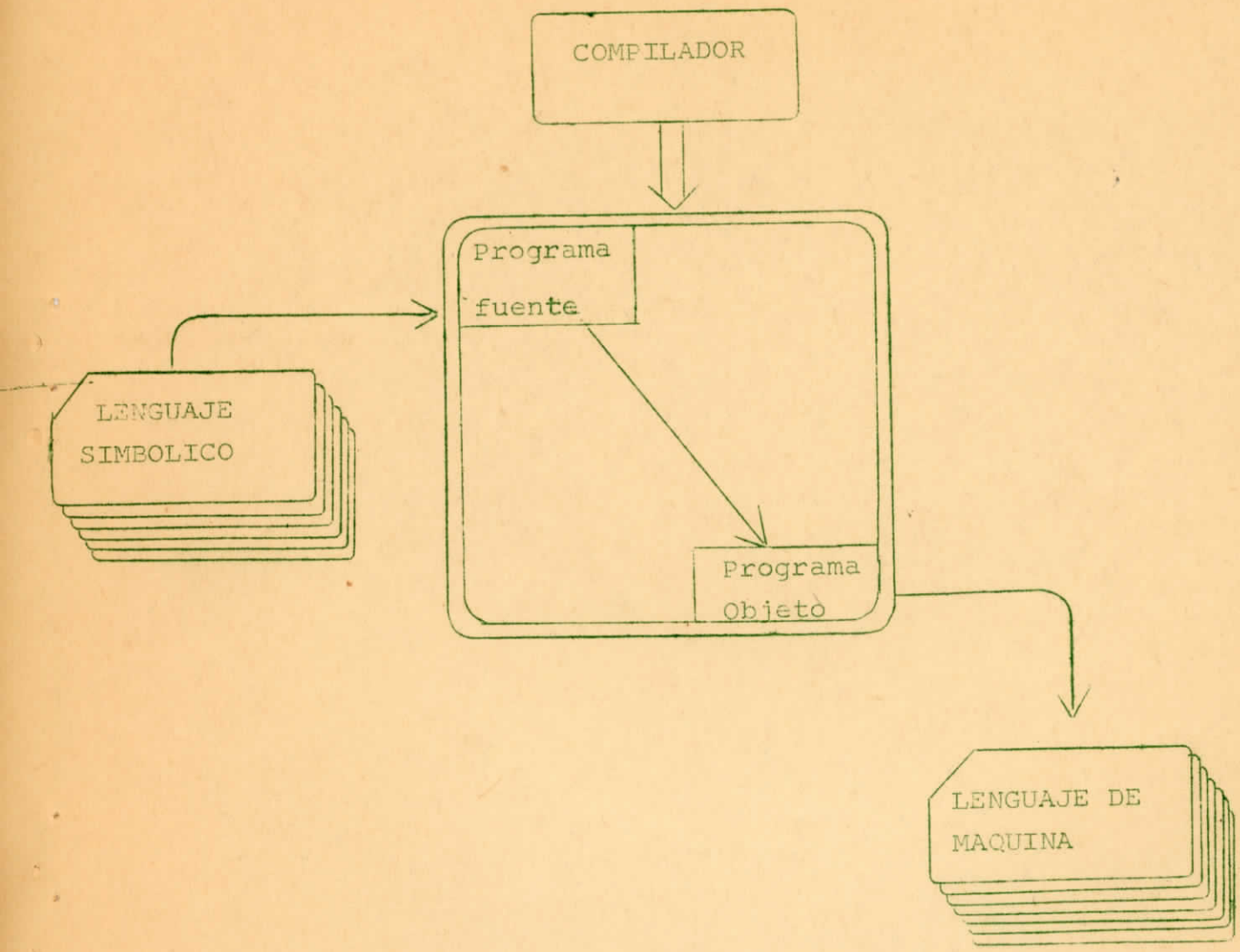
Compilador y Programas.

Toda máquina Computadora programada con Lenguajes Simbólicos (FØRTRAN-ALGOL-CØBØL-,etc.) tiene un programa traductor o compilador; de modo que en la computadora el programa simbólico pasa primero por el compilador, y éste lo traduce a un Lenguaje de Máquina, posible de ser interpretado por la máquina electrónicamente. Y es éste el programa que la computadora realiza.

El Programa: el fajo de tarjetas que entran en la Lectora en Lenguaje Simbólico recibe el nombre de programa fuente . Internamente, Control, lo hace pasar al Compilador; quién lo transforma en un programa capaz de accionar los terminales electrónicos de la computadora (Lenguaje de Máquina) . Este programa traducido recibe el nombre de programa objeto; y es el que dirige toda la operatoria.

Actualmente las computadoras vienen programadas para varios Lenguajes Simbólicos es decir tienen más de un compilador o un Compilador múltiple .

Esquema del paso por el Compilador .

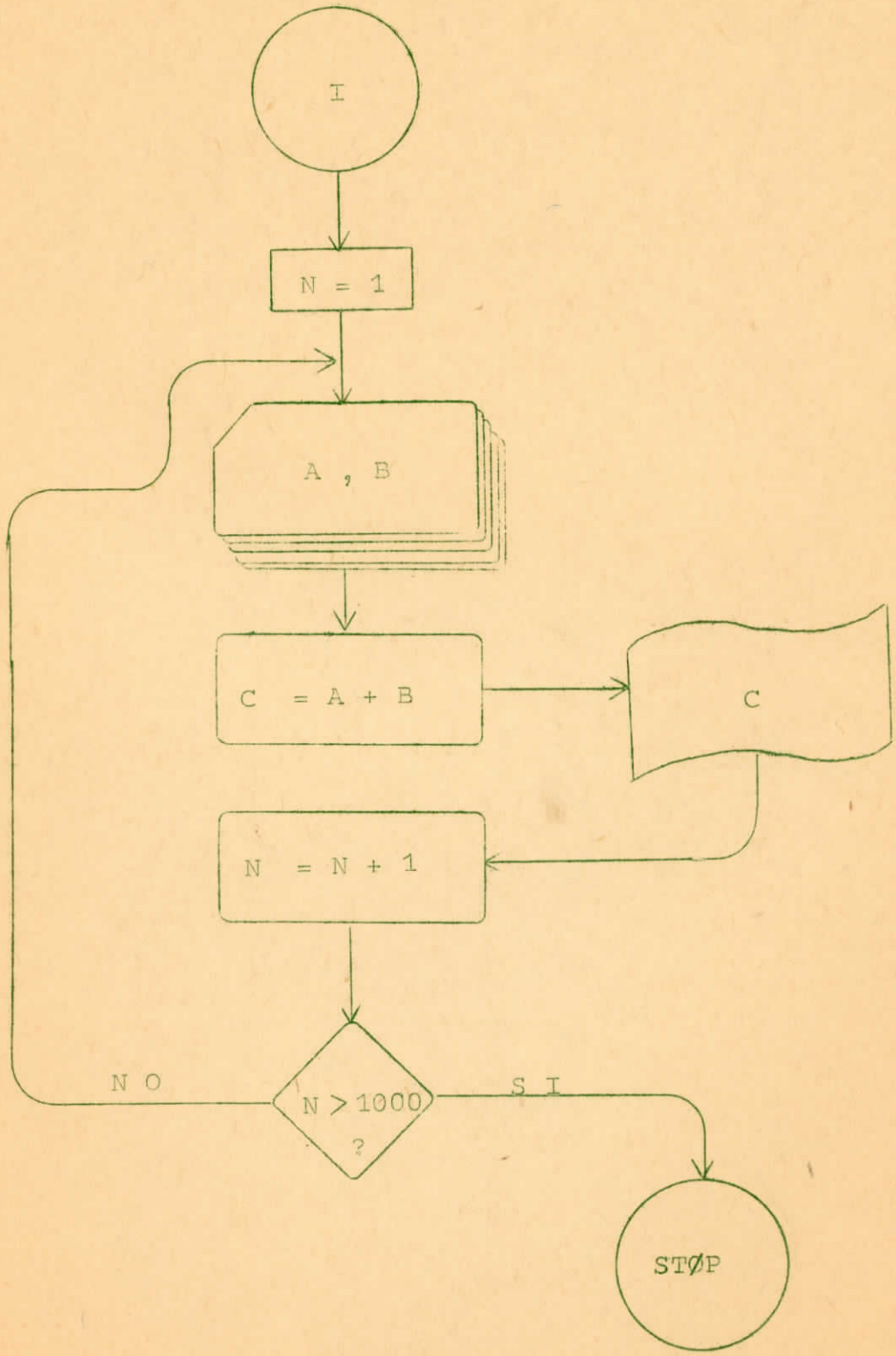


Todo esto ocurre en el interior de la máquina y no es directamente observable .

Es fácil comprender ahora, por qué la máquina se detiene muchas veces por algo que nos parece un detalle (un guión en el lugar de una coma, por ej.) . El compilador no reconoce estos signos (los guiones para hacer separaciones) como algunos de los símbolos que su programa traductor maneja y devuelve la tarjeta con "error", indicando que no es "codificable tal signo".

Programación .

Teniendo a la vista el Diagrama de Flujo se programa en FØRTRAN .



Problema:

Dados 2000 datos en tarjetas perforadas cada una con dos datos . Computar las 1000 sumas .

Programa :

```
      C
      C     PROGRAMA PARA SUMAR 2000 DATOS
      C     LOS DATOS SON NUMEROS REALES
            N = 1
        25  READ(1,20)A,B
        20  FORMAT(2F10.4)
            C=A+B
            WRITE(3,30)C
LØØP    30  FØRMAT(1H1,10X,F10,4)
            N=N+1
            IF(N-1000)8,8,12
         8  GØ TØ 25
        12  STØP
            END
```

Observación: Esta solución es teórica. La entrada de datos es muy poco práctica porque requeriría tipiar (perforar) mil tarjetas con datos.

Si en cada tarjeta se entran 10 datos en campos de F7.2 , la perforación se reduciría a 200 tarjetas lo que sería más práctico . Pero todavía inoficioso .

En el programa estudiado anteriormente se puede hacer algunas modificaciones prácticas tales modificaciones debieran hacerse primeramente en el Diagrama de Flujo .

Como actividad sugerida, los alumnos puede indicar las modificaciones que deseen y solucionar la Diagramación y la Programación correspondiente.

Problema: Se quiere confeccionar una lista de las raíces cuadradas de los números naturales a partir de 2. (Se desea estos valores con 5 decimales)

Explicitación del problema.

1) La computadora debiera ser programada para extraer

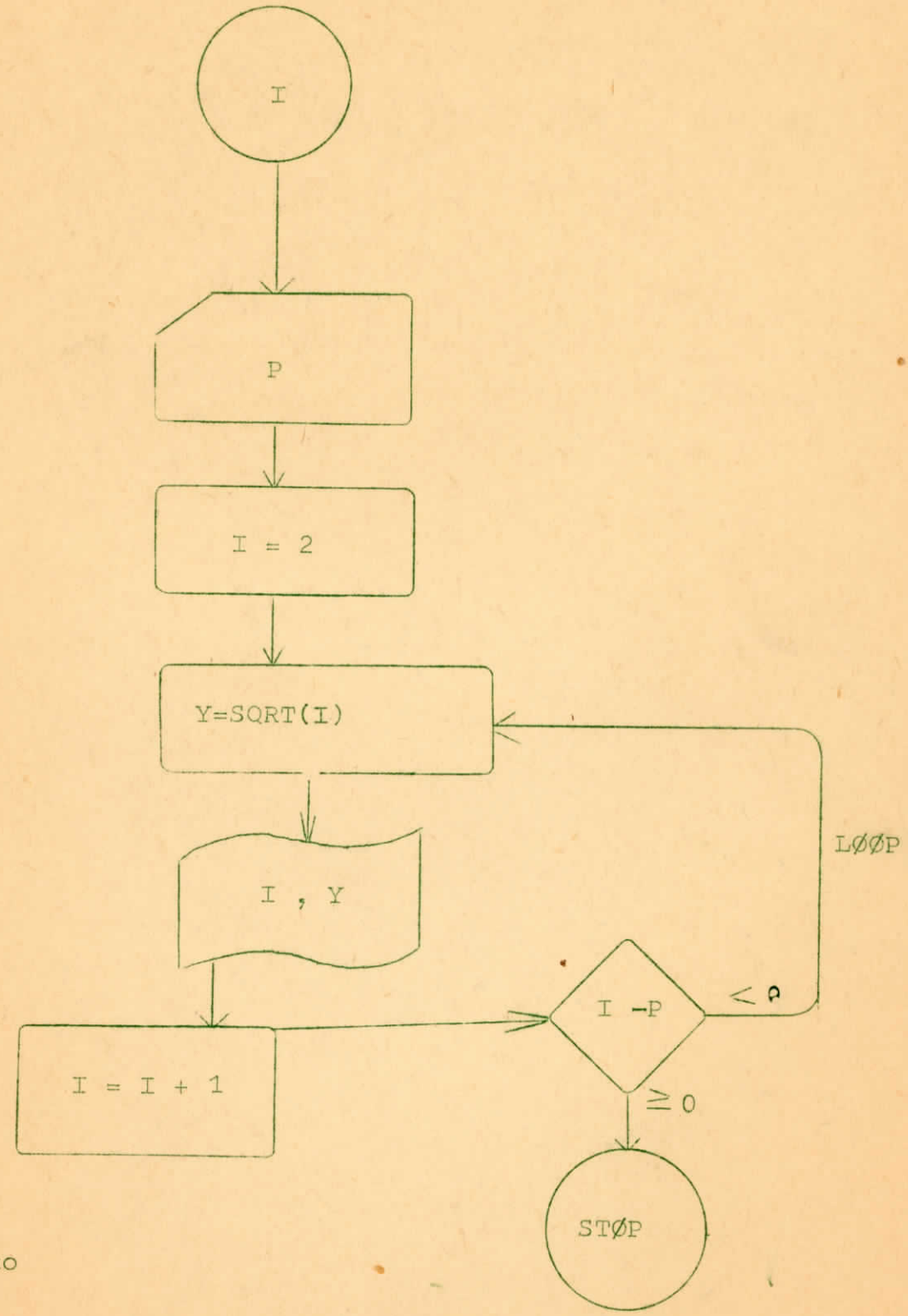
$\sqrt{2}$, luego
$\sqrt{3}$
$\sqrt{4}$
$\sqrt{5}$
etc hasta llegar a

$\sqrt{100}$

2) La operación indicada usando FORTRAN será
   Y = SQRT 2)
   Y = SQRT(3) , etc.

3) Tenemos que usar un incremento para repetir la extracción de raíz, variando desde 2 hasta 100

Diagrama de flujo correspondiente.



P , único dato
P = 101 .

Programa

Observación 1: La computadora necesita alimentarse con un único dato P = 101.

Observación 2: La computadora calcula $\sqrt{2}$ usando para hacerlo la asignación I = 2
La asignación se incrementa I = 3 la computadora entra a un LØØP del que saldrá al ser I = 101 pero en este caso (I - P) = 0 y el problema termina

Programa :

```
C     CALCULØ DE RAICES CUADRADAS
C     PRØGRAMA N N.
C
      READ(1,99)P
   99 FØRMAT(F10.4)
      I=2
   13 Y=SQRT(I)
      WRITE(3,88)I,Y
   88 FØRMAT(1H1,9X,'ENTERØS',I8,3X,'REALES'F10.4)
      I=I+1
      IF(I-P)4,7 7
    4 GØ TØ
    7 STØP
      END
```

Dato

		1	0	1	.	0	0	0	0

Con los elementos de FØRTRAN que ya poseemos se puede resolver gran variedad de problemas , daré algunos modelos .

Programa

Problema : Dado un número entero positivo, calcular N! (factorial de N)

$$N! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot \ldots\ldots \cdot N$$

El estudio del problema consiste en establecer la forma de "generar" este producto y en guardar los productos parciales. Lo mismo ocurrió cuando se trataba de obtener una sumatoria en Lenguaje de Máquina (recuérdese que la sumatoria se inicia con 0 el elemento neutro de la adición)

Tratándose de un producto el elemento esencial es el 1: elemento neutro de la multiplicación .

El primer producto es $1 \cdot 1 = 1$

El segundo producto es $1 \cdot 2 = 2$

El tercero producto es $1 \cdot 2 \cdot 3 = 6$ etc.

Al elemento neutro multiplicativo con que se inicia el producto: 1 se le asigna el nombre de enteros NFAC . La fórmula NFAC = NFAC * I es una asignación generadora del factorial.

I = 1 es una asignación para incrementos .

I = I + 1 es una asignación que genera los naturales generalmente usada como un contador .

Nota :

1) S = 0
2) S = S + P

Asignaciones para acumular sumas.

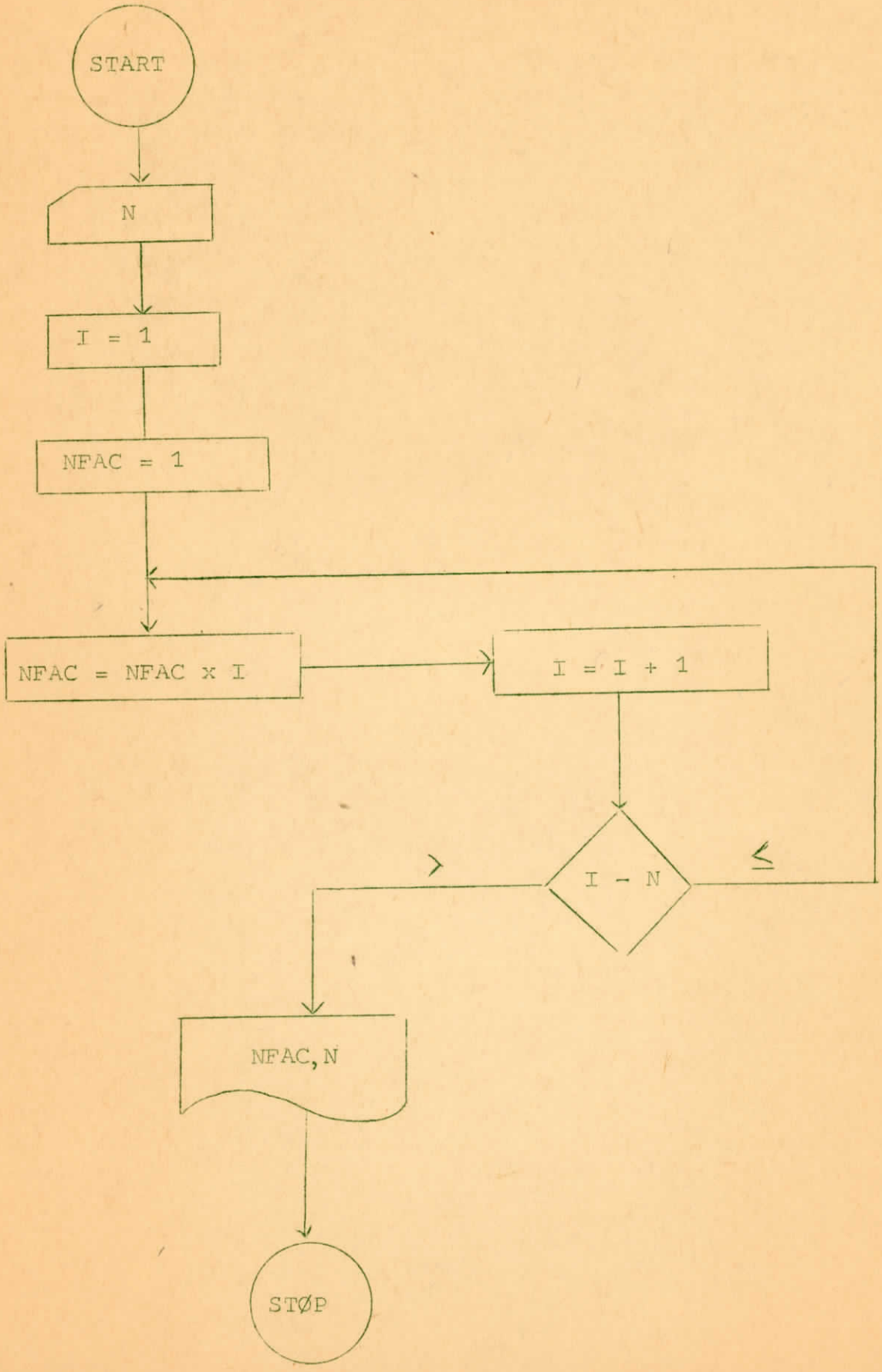
3) N = 1
4) N = N . L

Asignaciones para acumular productos .

5) I = 1
6) I = I + 1

Asignación para incrementar.

Diagrama de flujo del problema.



Programa en FØRTRAN

```
C                    PRØGRAMA MØDELØ Nº2
C                    COMENTARIOS
C                    PROGRAMA NN
                     READ(1,20)N
                20   FØRMAT(I8)
                     I=1
                     NFAC=1
                 8   NFAC=NFAC#I
                     I=I+1
                     IF=(I-N)3,3,4
                 3   GØ TØ 8
                 4   WRITE(3,30)N,NFAC
                30   FØRMAT(1H1,10X,I8,'FACTORIAL':I40)
                     STØP
                     END
```

Observación: La letra C que corresponde a Comentarios se coloca en cualquier casillero de la "Hoja de Programación" entre las columnas 1 y 5 ,

En estos mismo casilleros generalmente se colocan las columnas 4 y 5 los números de declaraciones que se citan en el Programa 8,3,4 y también 20 y 30 que son números de formato obligados El Programa propiamente tal , las declaraciones FORTRAN se escriben a partir de la columna 7. Y la columna 6 queda libre a menos que la línea de declaración FORTRAN sea tan larga que necesite ocupar 2 o 3 líneas en ese caso al escribir una segunda línea de continuación se escribe 1 en la columna 6 y 2 si se continúa en una tercera .

Existen algunas declaraciones FØRTRAN que tienen gran poder, por la utilidad que prestan en el cálculo; una de ellas es la declaración DØ; y aún cuando no está incluída en el programa de IV Año me parece que es muy sencilla de captar y que permite agilizar la programación. Daré solamente una breve información de lo que es y alguno problemas donde se ejemplarice su aplicación .

La declaración DØ permite concebir una iteración de la operación que implica el problema y dar en una sola orden todos los elementos que entran en la iteración y que conforman el Ciclo o LØØP. Es decir, se incluye en la misma órden o declaración el intervalo de variación de la variable .

Ejemplo de una Declaración DØ .

```
DØ30I = 2,55
```

El número 30 que sigue al DØ es como su número de identificación. Se deben repetir en ciclo las operaciones que siguen al DØ. Cuando este ciclo se repite, se sigue el programa en el número del DØ y allí hay un CØNTINUE , que mantiene la iteración. Acaba el ciclo sobrepasando la expresión CONTINUE .

```
Es decir DØ30I=2,55
                    }  Esta parte se repite
30 CØNTINUE
   STØP             Aquí ocurre la salida
   END              del ciclo.
```

Los números 2,55 después del igual tienen relación con I y expresan que I tomará inicialmente el valor 2 y su último valor, en la última vuelta del LØØP, debe ser 55. Con esto se evita la declaración I = 2, y luego I = I + 1 . Es decir el LØØP o ciclo se realiza mientras I varíe entre 2 y 55 .

Existen otras formas de uso de la declaración DØ: por ejemplo, si se quiere realizar un LØØP variando el incremento de 5 en 5, se incluyen tres valores después del = y el valor último, corresponde al incremento. ejemplo DØ30I= 4,104,5 .

La interpretación de esta declaración es que I varía entre 4 y 104 y el incremento es de 5 cada vez. Es decir I = 4 luego I = 9 , I = 14, I = 19, I = 24 etc. hasta I = 104 .

Otras veces el DØ lleva una variable de enteros DØ45J = 3,M . En este caso la variación de J es a partir de 3 en una unidad hasta terminar en M. Pero la máquina debe recibir en su información el valor de M por una asignación o por entrada en READ.

Ejemplo: Antes del DØ, debe existir M = 28 o bien READ(2,10) M .

Ver el programa para la solución de un problema resuelto anteriormente, usando ahora el DØ en esta forma.

"Dar una lista de las raíces cuadradas a partir de 2 hasta 100 ".

```
      READ(1,99)N
   99 FØRMAT(I10)
      DØ30I=2,N
      Y=SQRT(I)
      WRITE(3,20)I,Y
   20 FØRMAT(1H1,10X,I8,5X,F10.4)
   30 CØNTINUE
      STØP
      END
```

Problema

Calcular el número de diagonales de un polígono de N lados, a partir del cuadrilátero .

```
LDIAG=I*(I-3)/2
```

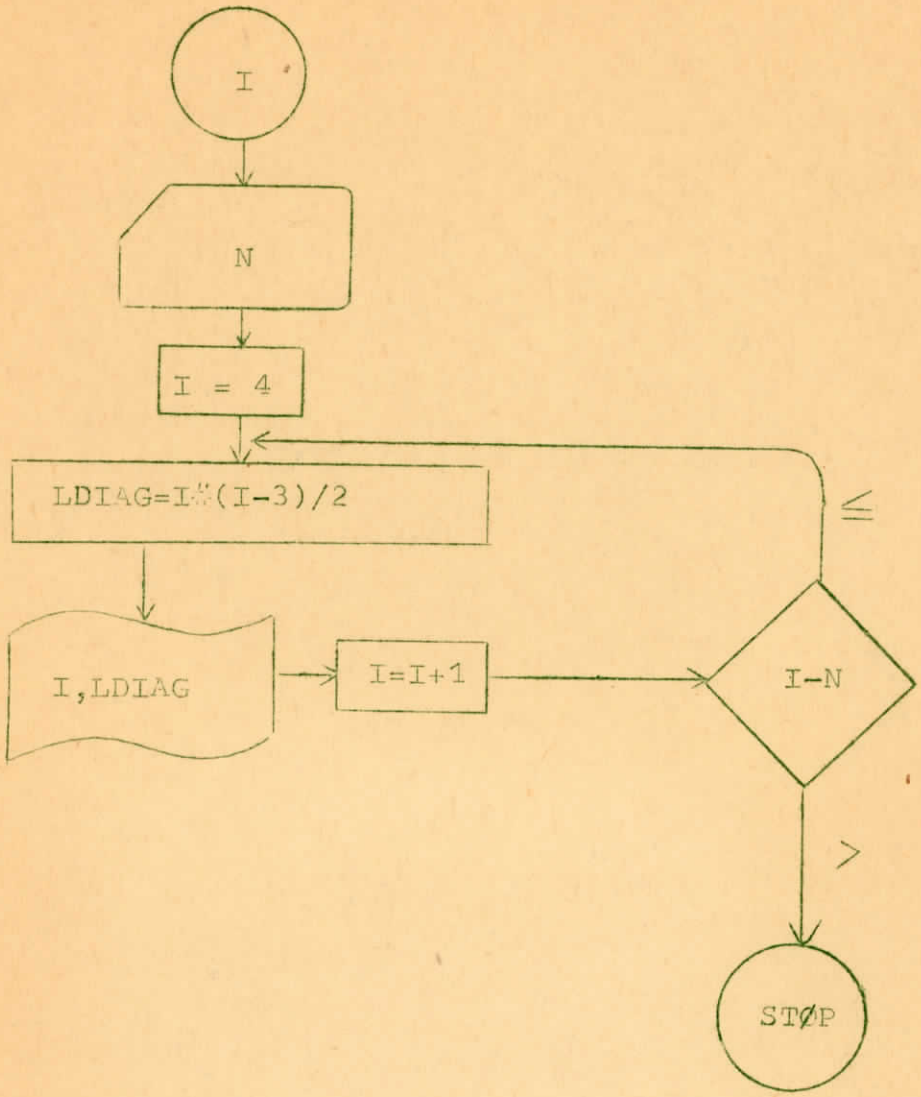
Este problema puede solucionarse con DØ o sin él .

Conviene observar las dos soluciones que se darán a continuación; y el Diagrama de Flujo que corresponde a la solución completa.

Supongamos como dato N = 20 (polígono de 20 lados) .

Problema :

Calcular el número de diagonales de un polígono de N lados , a partir del cuadrilátero



Primera solución .

```
      READ(1,10)N
10    FØRMAT(I8)
      I=4
13    LDIAG=I#(I-3)/2
      WRITE(3,20)I,LDIAG
20    FØRMAT(1H1,10X,2I8)
      I=I+1
      IF(I-N)3,3,4
3     GØ TØ 13
4     STØP
      END
```

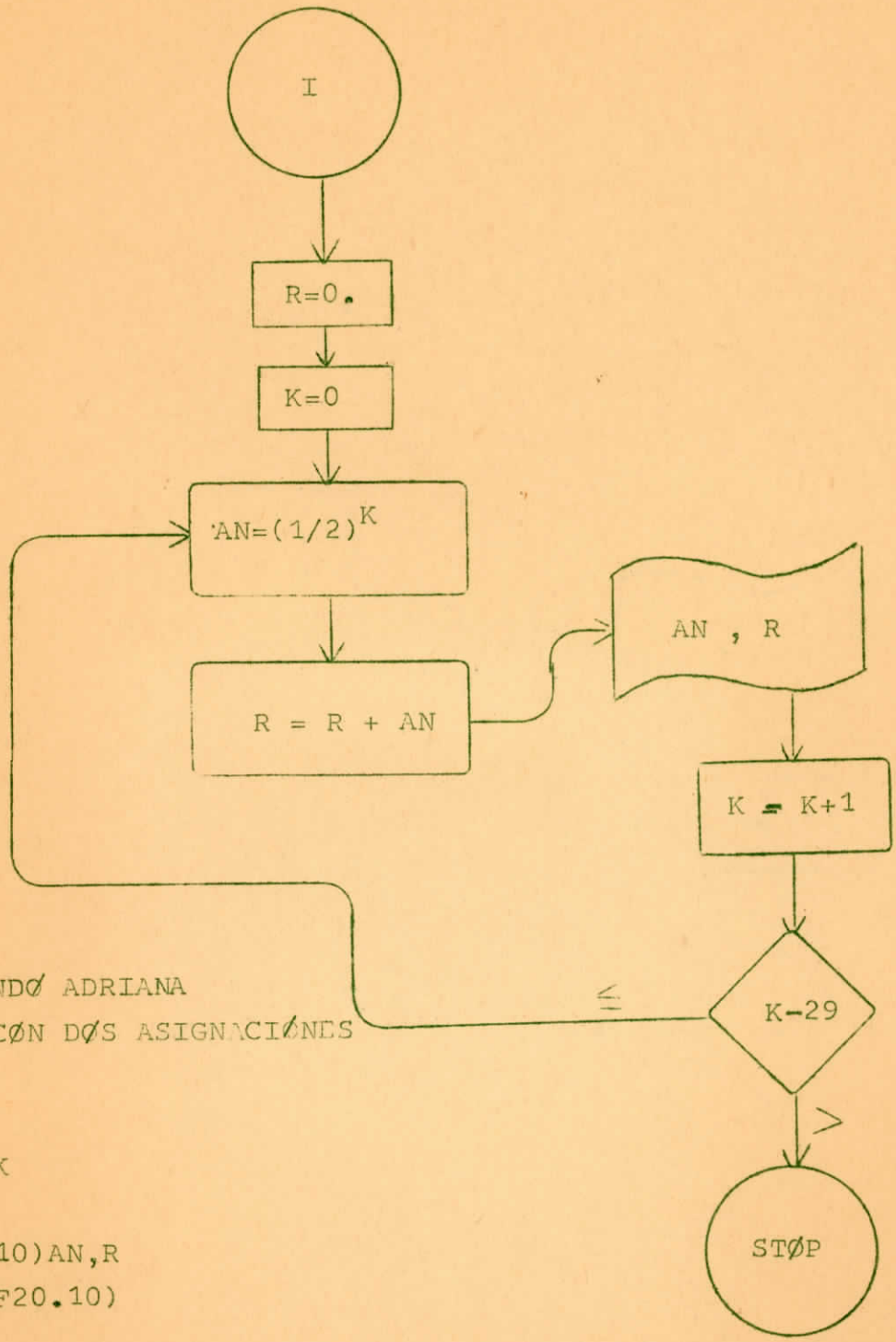
Segunda solución .

```
C     PRØGRAMA ADRIANA
C     SØLUCIØN USANDØ DØ
C     ..................
      DØ30I=4,20
      LDIAG=I#(I-3)/2
      WRITE 3,20)I,LDIAG
20    FØRMAT(1H1,10X,I8,3X,I8)
30    CØNTINUE
      STØP
      END
```

Problema:

Dada la serie $1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots\ldots\ldots$, comprobar que la suma de los términos se aproxima a 2 (como límite), calculando 30 sumatorias.

C




```
C   PRØGRAMANDØ ADRIANA
C   ENTRADA CØN DØS ASIGNACIØNES
    R=0.
    K=0
13  AN=0.5##K
    R=R+AN
    WRITE(1,10)AN,R
10  FØRMAT(2F20.10)
    K=K+1
    IF(K-29)7,7,11
 7  GØ TØ 13
11  STØP
    END
```

Sugerencias finales

Tal vez una de las mejores sugerencias metodológicas para el trabajo con los alumnos en esta Etapa del Aprendizaje es el trabajo, en grupos de 3 a 4 alumnos, con alguna fórmula que los enfrente al desafío de desarrollar una Programación adecuada para el cálculo . Hacer "correr" teoricamente el Programa elaborado es decir, efecturar la tarea que ejecuta el programador de correr varias vueltas del Programa, en el Diagrama de Flujo; variando los valores y confirmando que las soluciones son correctas. Estos programas así tratados pueden ser tipiados en cualquier Centro deComputación y "corridas" en forma particular por los alumnos que generalmente tienen contactos valiosos en IBM o en las Universidades. Esto, mientras no exista un Convenio oficial entre el Ministerio de Educación y las Universidades y Centro de Computación existentes en el país, para ceder horas de Computación a la experimentación educacional Y esto sucederá cuando los colegas pongan en marcha la enseñanza de esta Unidad; y la demanda de horas de computación a los distintos Centros sea una realidad que lo exija.

Lo interesante es que se comience con la Enseñanza de la Computación; que esta Unidad de Enseñanza que los Programas oficiales han considerado y que dan a Chile una situación privilegiada en la vanguardia de América Latina sean parte del curriculum y que nuestro colegas comprendan, que pueden ofrecer a sus alumnos grandes posibilidades a través de este aprendizaje, aún en los aspectos vocacionales; ya que la Orientación, no puede desconocer un campo ocupacional tan amplio, en el que ya se desarrollan multitud de carreras cortas al serivicio de la Técnica y de la Ciencias. Por otra parte , las Universidades desde este año, incluyen en sus programas educacionales para la formación de Profesores, algunos créditos de Computación Electrónica. Por lo tanto estas Técnicas Modernas de Procesamiento han dejado de ser patrimonio de ingenieros , y de investigadores de alto nivel, para ser elementos culturales que es preciso prever y planificar con tiempo; antes que la demanda no permita una enseñanza serena.

I N D I C E G E N E R A L