# SISTEMA DE PROGRAMAÇÃO

# FORTRAN II

## E SUA UTILIZAÇÃO COM UM COMPUTADOR IBM 1620

FUNDAÇÃO CALOUSTE GULBENKIAN
INSTITUTO GULBENKIAN DE CIÊNCIA
CENTRO DE CÁLCULO CIENTÍFICO

# SISTEMA DE PROGRAMAÇÃO FORTRAN II

## E SUA UTILIZAÇÃO COM UM COMPUTADOR IBM 1620

*2.ª EDIÇÃO, REVISTA*

M. ODETE RODRIGUES CADETE
e
ANTÓNIO M. F. CADETE

OEIRAS
1971

## *PREFÁCIO*

*O pequeno manual que agora se reedita (1.ª edição em 1964) destina-se especialmente aos utilizadores do computador IBM 1620 do Centro de Cálculo Científico que desejem dispor da máquina com alguma eficiência sem se preocuparem com a aprendizagem da codificação básica.*

*Designa-se a linguagem exposta por FORTRAN II. Escrevendo expressões como «READ1, A», «Y = B + 1.», «PRINT 2, Z», etc., o leitor poderá, longe do computador electrónico, preparar as ordens necessárias para resolver o seu problema. Terá, no entanto, de seguir* sem a menor falta *um certo número de regras, que encontrará nas páginas deste trabalho.*

*Aproveita-se a ocasião para exemplificar a linguagem FORTRAN (abreviatura de Formula Translation) através de vários programas — cálculo de integrais definidos, sistemas de equações algébricas lineares, raízes de polinómios, etc. Cada exemplo é precedido de uma breve introdução matemática.*

*Os trabalhos de impressão foram orientados pela Dr.ª M. da Graça Lopo e a execução dos desenhos esteve a cargo de M. L. B. Guimarães. M. Helena Teixeira de Sousa dactilografou o manuscrito. A todos agradecemos a colaboração prestada.*

*Lisboa, Janeiro de 1970*

M. ODETE RODRIGUES CADETE

ANTÓNIO M. F. CADETE

# SUMÁRIO

## 1.ª PARTE — A LINGUAGEM FORTRAN II


CAP. 1 — **Introdução**

Págs.

1.1 O sistema FORTRAN . . . . . 15
1.2 As instruções FORTRAN . . . . . 17
1.3 Representação em ponto decimal flutuante e representação inteira de números . . 18
1.4 Operações nos dois sistemas de representação . . . . . 23

CAP. 2 — **Constantes e variáveis**

2.1 Constantes . . . . . 25
2.1.1 Constantes inteiras . . . . . 25
2.1.2 Constantes de «ponto flutuante» . . . . . 25
2.2 Variáveis . . . . . 26
2.2.1 Variáveis inteiras . . . . . 26
2.2.2 Variáveis de «ponto flutuante» . . . . . 26
2.2.3 Índices e variáveis com índices . . . . . 27
2.2.4 A intrução *DIMENSION* . . . . . 28

CAP. 3 — **Expressões. Funções de biblioteca. Instruções aritméticas**

3.1 Expressões . . . . . 29
3.2 Funções de biblioteca . . . . . 30
3.3 Instruções aritméticas . . . . . 31
3.4 Cálculo de potências . . . . . 32

CAP. 4 — **Instruções de controle**

4.1 Números de referência . . . . . 35
4.2 Instruções de salto . . . . . 35
4.2.1 *GO TO n* . . . . . 35
4.2.2 *GO TO* $(n_1, n_2, ..., n_m), i$ . . . . . 35
4.2.3 *IF* $(a)$ $n_1, n_2, n_3$ . . . . . 36
4.2.4 *IF* (*SENSE SWITCH i*) $n_1, n_2$ . . . . . 36
4.3 Instruções de paragem . . . . . 37
4.3.1 *PAUSE* ou *PAUSE n* . . . . . 37
4.3.2 *STOP* ou *STOP n* . . . . . 37
4.4 Ciclos de repetição. As instruções *DO* . . . . . 38
4.4.1 Forma geral . . . . . 38
4.4.2 Instruções *DO* dentro de instruções *DO* . . . . . 39

Págs.

4.4.3 Conservação dos valores do índice . . . . . 41
4.4.4 Restrições sobre cálculos no campo de um *DO* . . . . . 41
4.5 *CONTINUE* . . . . . 41
4.6 *END* . . . . . 41

CAP. 5 — **Instruções de entrada e saída de informação**

5.1 Instruções de entrada . . . . . 43
5.2 Instruções de saída . . . . . 44
5.3 Listas . . . . . 44
5.4 Instruções de formato . . . . . 46
5.4.1 Forma geral . . . . . 46
5.4.2 Especificações de campos numéricos . . . . . 46
*a)* A especificação *Iw* . . . . . 47
*a)* A especificação *Ew.d* . . . . . 48
*c)* A especificação *Fw.d* . . . . . 48
*d)* Transmissão de valores com mudança de tipo de representação . . . . . 48
5.4.3 Especificações de campos alfanuméricos . . . . . 50
*a)* A especificação *wH* . . . . . 50
*b)* A especificação *wX* . . . . . 51
*c)* A especificação *Aw* . . . . . 51
5.4.4 Associação de especificações de formato . . . . . 52
5.4.5 Utilização múltipla de especificações . . . . . 53
*a)* Grupos de especificações . . . . . 53
*b)* Repetição de especificações e grupos de especificações . . . . . 54
5.4.6 Uso do símbolo / nas instruções *FORMAT* . . . . . 54
5.5 Entrada de dados . . . . . 55

CAP. 6 — **Funções definidas por expressões** . . . . . 57

CAP. 7 — **Subprogramas**

7.1 Subprogramas do tipo *FUNCTION* . . . . . 59
7.2 Subprogramas do tipo *SUBROUTINE* . . . . . 61
7.3 Observações . . . . . 63

CAP. 8 — **As instruções EQUIVALENCE e COMMON**

8.1 A instrução *EQUIVALENCE* . . . . . 65
8.2 A instrução *COMMON* . . . . . 66
8.3 Outras considerações . . . . . 67
8.4 Aproveitamento de resultados de um programa em programas posteriores . . . 69
8.5 Segmentação de um programa . . . . . 70

CAP. 9 — **A tradução dos programas em linguagem FORTRAN. Execução dos programas traduzidos**


| | | Págs. |
|---|---|---|
| 9.1 | Preparação do problema. Codificação | 73 |
| 9.2 | Tradução | 77 |
| 9.3 | Execução do programa traduzido | 82 |
| 9.4 | Observações | 84 |


## 2.ª PARTE — EXEMPLOS DE APLICAÇÃO


| | | |
|---|---|---|
| 1. | Soma de séries alternadas | 85 |
| 2. | Transformação de Euler | 89 |
| 3. | Cálculo de integrais definidos pela regra de Simpson | 93 |
| 4. | Integração de equações diferenciais de 1.ª ordem pelo método de Runge-Kutta | 97 |
| 5. | Resolução de sistemas de equações algébricas lineares pelo método de Gauss. Inversão de matrizes | 101 |
| 6. | Inversão de matrizes pelo método de Jordan | 109 |
| 7. | Resolução de sistemas de equações lineares pelo método iterativo de Gauss-Seidel | 117 |
| 8. | Raiz característica de módulo máximo de uma matriz | 121 |
| 9. | Valores próprios de matrizes tridiagonais simétricas | 127 |
| 10. | Soma de séries de polinómios de Chebyshev pelo método de Clenshaw | 133 |
| 11. | Ajustamento de curvas por séries de polinómios ortogonais. Critério dos mínimos quadrados | 135 |
| 12. | A função Gama | 147 |
| 13. | Raízes de polinómios. Método de Bairstow | 151 |

BIBLIOGRAFIA 159

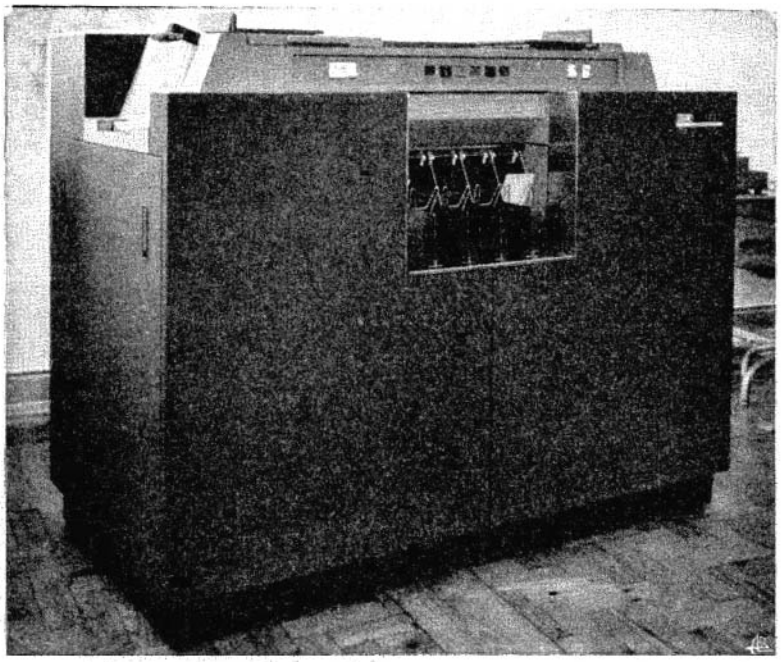


FIG. 2 — Unidade de perfuração e leitura de cartões

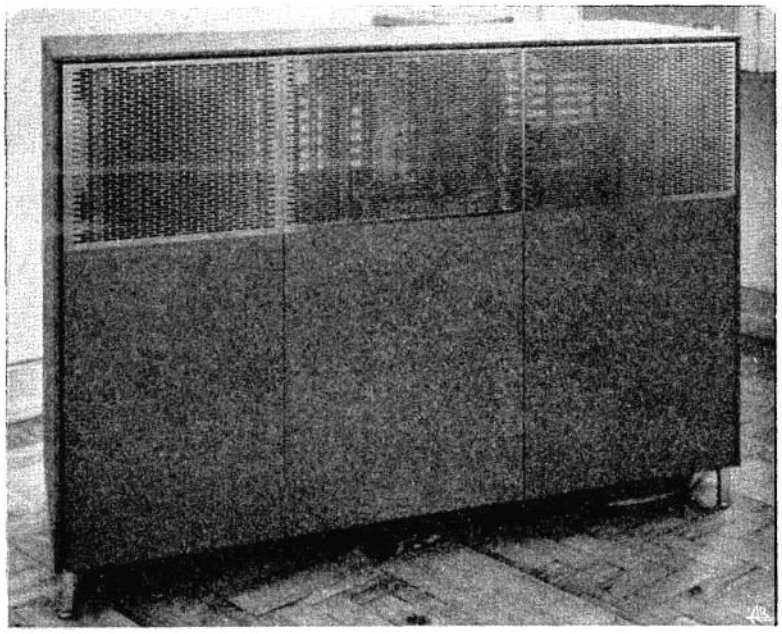


FIG. 3 — Memória adicional com 20 000 posições decimais

1.ª PARTE

# A LINGUAGEM FORTRAN II

## CAP. 1 — INTRODUÇÃO.

### 1.1 O Sistema FORTRAN

A linguagem de programação Fortran deve-se ao esforço de uma equipa americana dirigida por John Backus, que de 1954 a 1957 a concebeu e utilizou no computador IBM 704.

Um conjunto de ordens para resolver um problema, i. e., um programa, é uma série de símbolos materializados convenientemente em meios como cartões, fita de papel, etc., que transmitimos ao computador e que este regista, analisa e executa.

Cada computador tem os seus códigos de instrução próprios e só a eles é capaz de obedecer. As instruções em linguagem Fortran não são directamente aceitáveis por uma máquina. Fortran é uma linguagem artificial e torna-se necessário dispor de um programa TRADUTOR, em código da máquina, que resolva o problema preliminar: dado um «programa» na linguagem artificial Fortran, gerar as instruções aceitáveis pelo computador que executem a tarefa proposta inicialmente.

Informado o computador pelo TRADUTOR, podemos então fornecer-lhe um programa em Fortran, obtendo como resultado um programa correspondente no código da máquina. É este que dòravante se utiliza para resolver o problema.

Uma linguagem artificial (como Fortran, Algol, Cobol, PL/I, etc.) pode adquirir um certo grau de universalidade. Em princípio, um computador de qualquer marca pode receber a linguagem, desde que exista um tradutor adequado. Para cada computador particular, *linguagem* e *tradutor* formam um *sistema* solidário.

A versão original do Fortran tem sido progressivamente aumentada nas suas possibilidades. Designações como Fortran II e Fortran IV correspondem a esses desenvolvimentos.

O que vamos expor é o Sistema Fortran II para o IBM 1620. Comecemos por observar um programa.

C. C. C. **PROGRAMA "FORTRAN"**

PROGRAMA

PROGRAMADOR DATA PÁGINA 1

| COMENTÁRIO | CONT | INSTRUÇÃO FORTRAN | IDENTIFICAÇÃO |
|---|---|---|---|
| C | | CALCULO DO POLINOMIO P(X)=A(1)*X**N+....+A(N)*X+A(N+1), | 1 |
| C | | DE GRAU NAO SUPERIOR A 20 | 2 |
| C | | FORMULA DE RECORRENCIA B(K)=B(K-1)*X+A(K), B(1)=A(1), P(X)=B(N+1) | 3 |
| C | | DX=INCREMENTO DE X | 4 |
| C | | CHAVE 1 LIGADA-IMPRESSAO DOS RESULTADOS | 5 |
| C | | CHAVE 1 DESLIGADA-PERFURACAO DOS RESULTADOS | 6 |
| C | | | 7 |
| | | DIMENSION A(21) | 8 |
| | 5 | READ1,N | 9 |
| | 1 | FORMAT(I2) | 10 |
| | | IF(N-20)2,2,3 | 11 |
| | 3 | PRINT4 | 12 |
| | 4 | FORMAT(17HGRAU MAIOR QUE 20) | 13 |
| | | GO TO 5 | 14 |
| | 2 | N1=N+1 | 15 |
| | | READ6,(A(I),I=1,N1) | 16 |
| | 6 | FORMAT(E14.8) | 17 |
| | | READ7,X,DX,XL | 18 |
| | 7 | FORMAT(3E14.8) | 19 |
| | | IF(SENSE SWITCH1)8,9 | 20 |
| | 8 | PRINT10 | 21 |
| | 10 | FORMAT(7X,1HX,13X,4HP(X)) | 22 |
| | | GO TO 11 | 23 |
| | 9 | PUNCH10 | 24 |
| | 11 | P=A(1) | 25 |

C. C. C. **PROGRAMA "FORTRAN"**

PROGRAMA

PROGRAMADOR DATA PÁGINA 2

| COMENTÁRIO | CONT | INSTRUÇÃO FORTRAN | IDENTIFICAÇÃO |
|---|---|---|---|
| | | DO 12J=2,N1 | 26 |
| | 12 | P=P*X+A(J) | 27 |
| | | IF(SENSE SWITCH1)13,14 | 28 |
| | 13 | PRINT15,X,P | 29 |
| | | GO TO 16 | 30 |
| | 14 | PUNCH15,X,P | 31 |
| | 15 | FORMAT(E14.8,E16.8) | 32 |
| | 16 | X=X+DX | 33 |
| | | IF(X-XL)11,11,5 | 34 |
| | | END | 35 |

## 1.2 As instruções FORTRAN

As instruções Fortran, que o programador escreve primeiro numa folha de papel, são geralmente fornecidas ao computador em cartões. A cada símbolo correspondem certas perfurações, numa mesma coluna.

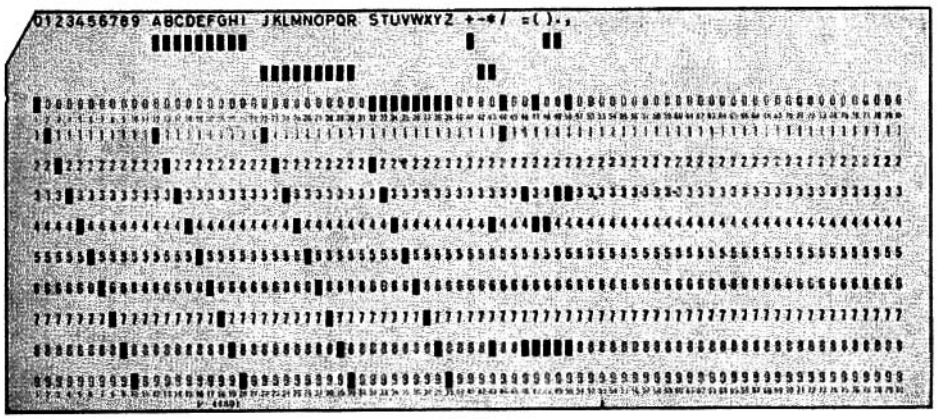



FIG. 4

Os símbolos mais utilizados em Fortran figuram no topo do cartão abaixo desenhado.

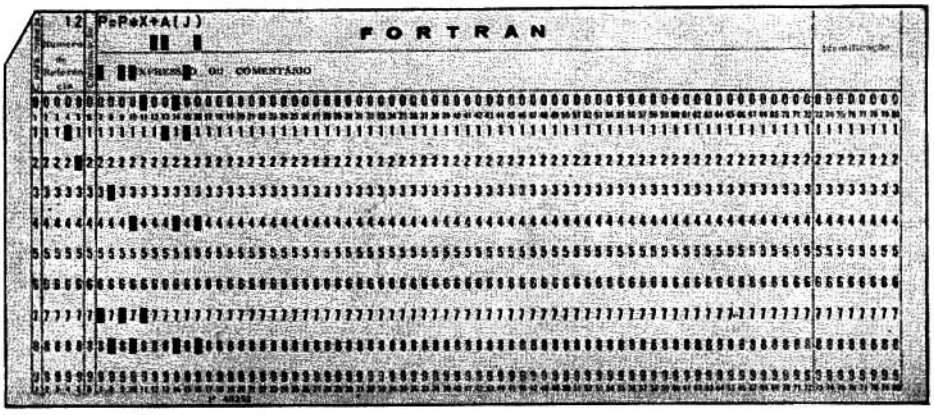



FIG. 5

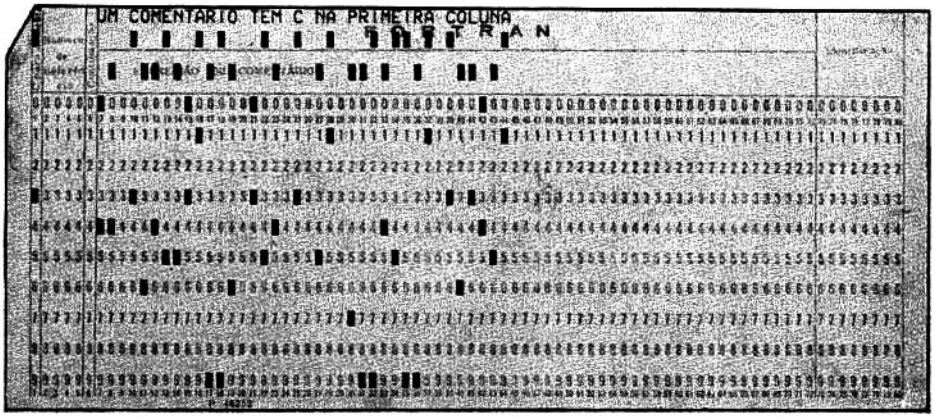



FIG. 6

Os símbolos que formam as instruções perfuram-se nas colunas 7 a 72 do cartão. Cada símbolo ou espaço em branco ocupa uma coluna. Se a instrução for demasiado extensa para um só cartão pode prosseguir noutros (cartões de continuação), que devem ter perfurado na coluna 6 um dígito diferente de 0. A sexta coluna do primeiro cartão de uma instrução pode ser deixada em branco ou levar um zero.

O número máximo de cartões permitido para uma instrução é cinco. As colunas 2 a 5 destinam-se aos «números de referência» (vidè 4.1). Podem introduzir-se comentários, sem efeito no programa. Os cartões onde são registados devem ter um C na primeira coluna.

## 1.3 Representação em «ponto decimal flutuante» e representação inteira de números

O par de números $(a;b)$ tal que $a\times 10^b = A\ (\neq 0)$, com $0.1 \leqslant |a| < 1$ (isto é, com o ponto decimal «flutuando» à esquerda do algarismo mais significativo de $a$) e $b$ inteiro, é a representação em ponto flutuante do número A. Zero tem tratamento especial.

No computador esta representação sofre as restrições seguintes:

1) $a$ tem $f$ e só $f$ dígitos, sendo $f$ um inteiro *que se fixa para cada programa.*

Os valores mínimo e máximo permitidos para $f$ são 2 e 28.

2) $-99 \leqslant b \leqslant +99$

Costuma chamar-se a *a mantissa* e a *b expoente.*

Exemplos

| A | f | a | b | Representação interna (*) |
|---|---|---|---|---|
| 36.973 | 8 | +.36973000 | +02 | $\bar{3}697300\bar{0}\bar{2}$ |
| −5.61 | 7 | −.5610000 | +01 | $\bar{5}61000\bar{0}\bar{1}$ |
| −0.007856396 | 6 | −.785639 | −02 | $\bar{7}8563\bar{9}\bar{0}\bar{2}$ |
| $3.4 \times 10^{100}$ | | número não representável em FORTRAN | | |
| 0. | 8 | +.00000000 | −99 | $\bar{0}000000\bar{9}\bar{9}$ |
| +3.56 | 3 | +.356 | +01 | $\bar{3}56\bar{0}1$ |

(*) NOTA — A memória do computador IBM 1620 existente no Centro de Cálculo Científico é constituída por 40 mil posições, com número de ordem variando de 0 a 39999, cada uma das quais está apta a registar um dígito decimal (ou certos símbolos especiais).

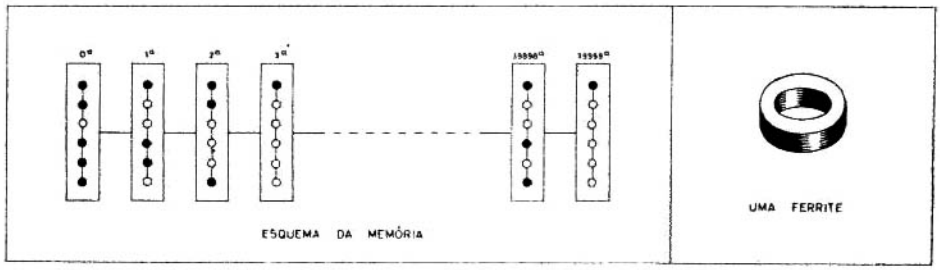



FIG. 7

Os números são registados em grupos de posições consecutivas. Cada posição é constituída por seis ferrites. Uma ferrite pode tomar um de dois estados magnéticos que designaremos por «1» e «0». Quatro das seis ferrites representam o dígito em forma binária. Para cada número, o algarismo mais à esquerda é assinalado com magnetização «1» na 5.ª ferrite; se o número for negativo, o mesmo sucede ao algarismo da direita. A 6.ª ferrite (de controle) toma o estado «1» ou «0» de modo a tornar ímpar o número de ferrites «1» de cada posição da memória.

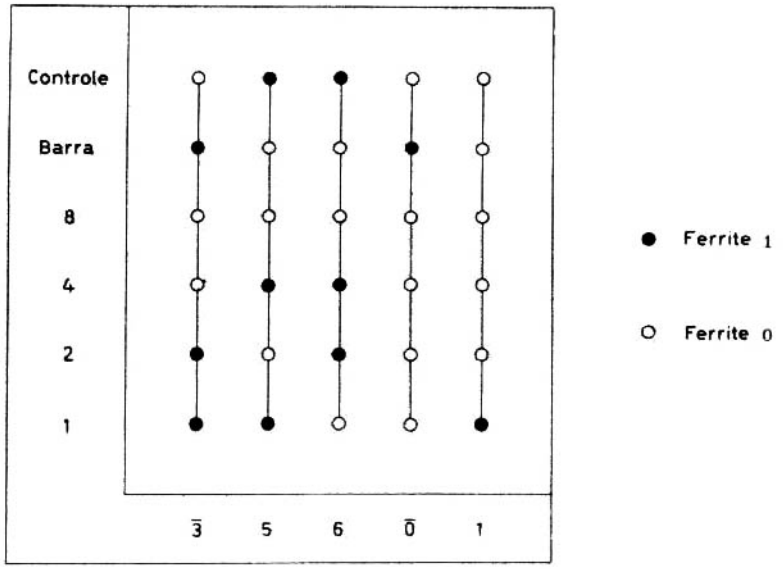



FIG. 8 — Representação interna de 3,56 (com $f = 3$).

A representação em ponto flutuante, fazendo corresponder a cada número um par de números, exige $f + 2$ posições de memória ($f$ para a mantissa e 2 para o expoente).

A representação por um único número é reservada apenas a números inteiros e designa-se por representação inteira.

Tal como para ponto flutuante, também se reserva em cada programa um número fixo de posições de memória para cada número de representação inteira.

Designaremos por $k$ este número. O valor de $k$ não pode ser superior a 10 nem inferior a 4.

Exemplos

| Número | k | Representação interna |
|---|---|---|
| 3107 | 4 | $\bar{3}107$ |
| −1963 | 6 | $\bar{0}01963\bar{3}$ |

Os valores de $f$ e $k$ a utilizar na execução do programa são fornecidos num cartão, que precede os cartões com instruções, perfurado do modo seguinte:

Coluna 1 — *
Colunas 2 e 3 — $f$
Colunas 4 e 5 — $k$

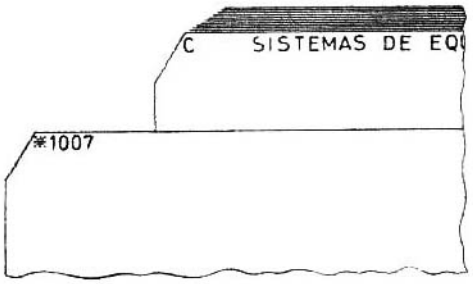



FIG. 9 — O cartão de precisão

Se não é fornecido um cartão com estas indicações a máquina opera com $f = 8$ e $k = 5$.

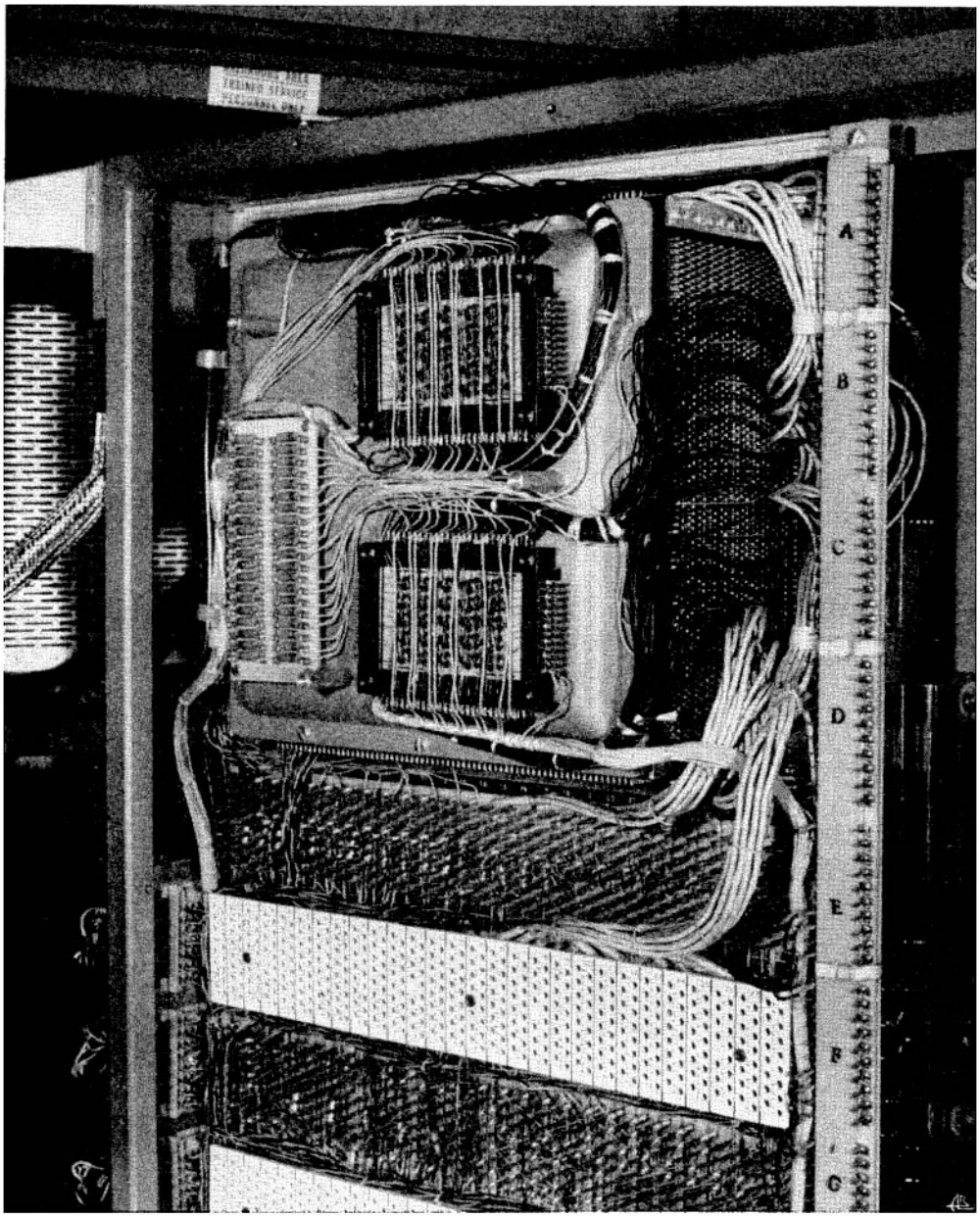



FIG. 10 — Um dos blocos de memória.

## 1.4 Operações nos dois sistemas de representação

Em adições, subtracções, multiplicações e divisões, os operandos não podem ser de representações diferentes e o resultado tem o tipo de representação dos operandos.

Com números de representação inteira processa-se aritmética de números inteiros e o resultado é registado com o mesmo número de dígitos dos operandos.

| k | Operandos | | Soma | Diferença | Produto | Cociente |
|---|---|---|---|---|---|---|
| 6 | $\bar{0}00003$ | $\bar{0}00002$ | $\bar{0}00005$ | $\bar{0}00001$ | $\bar{0}00006$ | $\bar{0}00001$ |
| 4 | $\bar{0}004$ | $\bar{0}006$ | $\bar{0}010$ | $\bar{0}00\bar{2}$ | $\bar{0}024$ | $\bar{0}000$ |
| 5 | $\bar{9}0986$ | $\bar{1}0235$ | $\bar{0}1221$ (*) | $\bar{8}0751$ | $\bar{4}1710$ (*) | $\bar{0}0008$ |

(*) Há truncatura da parte mais significativa se o resultado não cabe em $k$ posições. A soma e o produto no terceiro exemplo eram respectivamente 101221 e 931241710.

As operações com números em ponto flutuante executam-se da forma seguinte:

| f | Operandos | | Soma | Diferença | Produto | Cociente |
|---|---|---|---|---|---|---|
| 4 | $\bar{3}00\bar{0}\bar{0}1$ | $\bar{2}00\bar{0}\bar{0}1$ | $\bar{5}00\bar{0}\bar{0}1$ | $\bar{1}00\bar{0}\bar{0}1$ | $\bar{6}00\bar{0}\bar{0}1$ | $\bar{1}50\bar{0}\bar{0}1$ |
| 5 | $\bar{1}245\bar{6}\bar{0}1$ | $\bar{2}352\bar{3}\bar{0}2$ | $\bar{2}227\bar{8}\bar{0}2$ | $\bar{2}476\bar{8}\bar{0}2$ | $\bar{2}930\bar{0}\bar{0}2$ | $\bar{5}295\bar{2}\bar{0}\bar{1}$ |
| 5 | $\bar{1}005\bar{6}\bar{0}5$ | $\bar{9}987\bar{3}\bar{0}4$ | $\bar{2}004\bar{3}\bar{0}5$ | $\bar{6}900\bar{0}\bar{0}2$ | $\bar{1}004\bar{3}\bar{0}9$ | $\bar{1}006\bar{8}\bar{0}1$ |
| 6 | $\bar{9}9102\bar{3}\bar{0}1$ | $\bar{1}5430\bar{8}\bar{0}0$ | $\bar{9}7559\bar{3}\bar{0}1$ | $\bar{1}0064\bar{5}\bar{0}2$ | $\bar{1}5292\bar{2}\bar{0}1$ | $\bar{6}4223\bar{6}\bar{0}2$ |

A mantissa do resultado é registada com $f$ algarismos. Em adições e subtracções, a mantissa do operando de menor módulo é deslocada de acordo com a diferença dos expoentes, sendo truncados os algarismos que fiquem para a direita do último algarismo da mantissa do operando de módulo superior, antes de se efectuar a operação.

# Cap. 2 — CONSTANTES E VARIÁVEIS.

No capítulo anterior referimo-nos à representação de números no interior do computador. Veremos, agora, as formas externas da informação numérica.

## 2.1 Constantes

Ao elaborar um programa podemos escrever constantes de dois tipos, umas com ponto decimal outras sem ponto decimal. As primeiras terão representação de ponto flutuante e as segundas representação inteira.

### 2.1.1 — Constantes inteiras

São sucessões de 1 a $k$ dígitos precedidos ou não de sinal + ou — e sem ponto decimal.

Exemplos:

34 , +05 , —33 , —1 , —55380

O módulo máximo destas constantes é portanto $10^k - 1$.

### 2.1.2 — Constantes de «ponto flutuante»

Uma constante de ponto flutuante é qualquer número com ponto decimal precedido ou não de sinal e podendo ser seguido de «expoente». O «expoente» é constituído por um ou dois dígitos (com ou sem sinal) e separado da parte anterior pela letra $E$. A presença do expoente equivale a uma multiplicação por uma potência de base 10 e com esse expoente.

Exemplos:

$-3.5$ ; $.418$ ; $-6.002$ ; $-000.04$
$+4.6E3$ $(4.6\times10^{3})$ ; $-5.93E-03$ $(-5.93\times10^{-3})$
$10.E+17$ $(10^{18})$ ; $.697E+7$ $(0.697\times10^{7})$

Se a constante tem mais dígito do que os $f$ que a mantissa pode conter, serão considerados apenas os $f$ mais sign ficativos.

Assim, com $f=8$ a constante 123.456789 será representada internamente por $\bar{1}2345678\bar{0}3$ (sem arredondamento); para 0.000011223344556 o registo interno é $\bar{1}122334\bar{0}4$.

## 2.2 Variáveis

As variáveis são sucessões de letras e dígitos com que se localizam porções da memória.

Não podem ser formadas por mais de 6 caracteres, o primeiro dos quais deve ser uma letra.

Exemplos:

*A* , *A12* , *AB3C* , *CURSO*

Uma variável designa um local, que durante a execução do programa pode receber números diversos. Admitem-se também dois tipos de variáveis: inteiras e de ponto flutuante.

### 2.2.1 — Variáveis inteiras

A letra inicial destas variáveis deve ser *I*, *J*, *K*, *L*, *M* ou *N*. As $k$ posições de memória designadas por variáveis inteiras, só podem receber números de representação inteira.

Exemplos:

*IR* , *I11J* , *MAIOR* , *N* , *KAPA* , *J23456*

### 2.2.2 — Variáveis de «ponto flutuante»

A letra inicial deve ser *diferente* de *I*, *J*, *K*, *L*, *M* ou *N*. Estas variáveis designam $f+2$ posições de memória que só podem receber números de representação de ponto flutuante.

Exemplos:

*C* ; *X12* ; *ALFA* ; *REAL*

### 2.2.3 — Índices e variáveis com índices

Representamos vulgarmente por uma mesma letra acompanhada de índices os elementos de um vector ou de uma matriz. Anàlogamente, utilizam-se em Fortran variáveis seguidas de índices para designar elementos de conjuntos com uma, duas ou três dimensões.

Os índices são separados por vírgulas e envolvidos por parêntesis.

Exemplos:

$A(1)$ ; $B(3,5)$ ; $INT(1,2,3)$
$A(I)$ ; $C(I,J)$ ; $FLUT(L,M,N)$

$A(1)$ designa o primeiro elemento do conjunto monodimensional $A$.

$A(I)$ pode representar qualquer valor do mesmo conjunto, de acordo com o valor que tome a variável $I$.

Os índices só podem tomar valores positivos, e podem ter as formas a seguir apresentadas, onde $i$ designa uma variável inteira, $c$ e $c'$ são constantes inteiras sem sinal e $*$ é o sinal de multiplicação.

| Forma | Exemplos |
|---|---|
| $i$ | $J13$ |
| $c$ | 19 |
| $i+c$ | $K3+1$ |
| $i-c$ | $N-7$ |
| $c*i$ | $3*M$ |
| $c*i+c'$ | $2*I+1$ |
| $c*i-c'$ | $10*KLM-2$ |

São formas ilegais de índice

$I+J$ , $I-J$ , $I*3$, $+I$ , $-I$ , $-3*J$ , etc.

Os elementos dos conjuntos dimensionados terão internamente representação inteira se a letra inicial da variável for $I$, $J$, $K$, $L$, $M$ ou $N$; nos outros casos terão representação de ponto flutuante.

O valor máximo que um índice pode atingir deve figurar numa instrução especial (DIMENSION), que precede a primeira ocorrência da variável com esse índice (ver 2.2.4).

Assim $NK(I-1, J+1, 3*I)$ localiza elementos inteiros, enquanto que $FL(3*I+2,3)$ representa elementos de ponto flutuante.

# Cap. 3 — EXPRESSÕES. FUNÇÕES DE BIBLIOTECA. INSTRUÇÕES ARITMÉTICAS.

## 3.1 Expressões

As expressões Fortran são semelhantes às expressões matemáticas. Os símbolos usados para as diferentes operações são os seguintes:

| | |
|---|---|
| adição | + |
| subtracção | − |
| multiplicação | * |
| divisão | / |
| potenciação | ** |

Regras para formar expressões:

1) Numa expressão Fortran, os elementos constituintes (constantes, variáveis, etc.) devem ser do mesmo tipo de representação interna.

Constitui excepção a esta regra a possibilidade de uma expressão de aritmética de inteiros aparecer como índice ou expoente numa expressão com aritmética de ponto flutuante.

São válidas, por exemplo, as expressões

$A(I,J)$ ; $(A+B+C)/(D-C)$ ; $2./A*(3.*B-1.)$ ;
$3**5$ ; $I**5$ ; $I**I$ ; $3**I$ ; $(I+J)**(J+2*K+1)$ ;

(expressão inteira elevada a expressão inteira)

$2.**7$ ; $A**(-I)$ ; $(A1-X)**(I+J-K)$ ; $CD(1,I,J)**2$

(expressão real elevada a expressão inteira)

$2.**7.$ ; $A**0.5$ ; $A**E$ ; $(A+E)**(A-E)$

(expressão real elevada a expressão real)

mas não é legítimo escrever

$A+2$ ; $3.+I$ ; $2\times5.$

$I**R$ (expressão inteira elevada a expressão real)

2) Os símbolos de operação não podem apresentar-se em sucessão. Devem separar-se por parêntesis.

$\frac{A}{-B}$ deverá escrever-se $A/(-B)$ e não $A/-B$.

3) Quando, numa expressão, a ordem das operações não esteja completamente especificada por parêntesis, entende-se que é a seguinte:

1.ª — potenciação
2.ª — multiplicação ou divisão
3.ª — adição ou subtracção

Por exemplo

$(1. + A*B**(C**D))/(A + B)$ corresponde a $\frac{1+A\times B^{(C^D)}}{A+B}$

Numa sucessão de operações com a mesma prioridade respeita-se a ordem de escrita.

O tradutor não aceita contudo uma expressão do tipo $A**B**C$. Devemos precisar se desejamos $(A**B)**C$ ou $A**(B**C)$.

## 3.2 Funções de biblioteca

As funções seno, co-seno, arcotangente, exponencial, logaritmo, raiz quadrada e valor absoluto, podem ser fàcilmente utilizadas em Fortran. Para usar estas funções num programa, indica-se numa expressão o «nome» da função seguido do argumento envolvido por parêntesis.

| Nome | |
|---|---|
| LOGF ($x$) | logaritmo natural do módulo de $x$ (*) |
| EXPF ($x$) | exponencial de $x$ com base $e$ |
| COSF ($x$) | co-seno de $x$ ($x$ em radianos) |
| SINF ($x$) | seno de $x$ ($x$ em radianos) |
| ATANF ($x$) | arctg $x$; resultado em radianos, entre $-\frac{\pi}{2}$ e $+\frac{\pi}{2}$ |
| SQRTF ($x$) | raiz quadrada do módulo de $x$ (*) |
| ABSF ($x$) | módulo de $x$ |

(*) Observação: Se $x < 0$, o computador escreve uma mensagem de erro e prossegue os cálculos.

O argumento deve ser uma expressão de ponto flutuante excepto em *ABSF(x)* onde pode ser também uma expressão inteira. O resultado tem o mesmo tipo de representação do argumento.

Exemplos:

$$J + ABSF\,(L + 3)$$

$$SINF\,(A(2,I)) - (LOGF\,(2.*A))**2$$

$$EXPF\,(SQRTF\,(ABSF\,(4. - B)) - (ATANF\,(B**I))) .$$

Notar que o nome de uma função pode figurar no argumento de outra função.

## 3.3 Instruções aritméticas

Uma instrução aritmética transmite o valor de uma expressão a uma variável.

A forma geral destas instruções é:

«variável» = «expressão aritmética»

O sinal = tem aqui um papel de agente transmissor.

Exemplos:

| | |
|---|---|
| $Y = 3.$ | O número 3 é transmitido a $Y$ |
| $X = X+1.$ | Ao valor localizado por $X$ é adicionada uma unidade e o resultado colocado de novo em $X$ |
| $Z = A+ABSF(B(3))$ | O valor absoluto de $B(3)$ é somado a $A$ e transmitido a $Z$ |

Se a variável à esquerda do sinal = for inteira, e a expressão à direita for de ponto flutuante, o resultado será primeiramente calculado em ponto flutuante e depois truncado, passando a sua parte inteira para a variável da esquerda.

Se a variável da esquerda for de ponto flutuante e a expressão for inteira, esta será calculada em aritmética de inteiros e o resultado é passado para representação de ponto flutuante, para poder ser atribuído à variável do primeiro membro.

Exemplos:

| | |
|---|---|
| $I = A$ | Com $A = 35.1$ vem $I = 35$ |
| $A = I + 1$ | Com $I = 5$ temos um resultado equivalente a $A = 6$. |
| $K = B*C$ | Com $B = 1.2$ , $C = 3.9$ vem $K = 4$ |

## 3.4 Cálculo de potências

As potências cuja base e expoente são expressões de tipo real, como $BASE**E$, são calculadas, por meio de $EXPF\ (LOGF\ (BASE)*E)$. Se $BASE$ toma um valor negativo (e $E$ um valor não nulo) não tem significado calcular o logaritmo; o computador protesta, considera o módulo da base e prossegue.

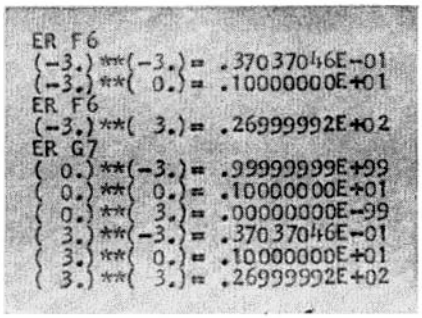
Para $BASE**0$. o resultado é sempre 1. (*); $0.**E$ dá-nos 0. se $E$ contém um número positivo.

As linhas seguintes, obtidas com a ajuda do próprio computador, esclarecem diversas situações de base inteira ou real associada a expoente inteiro, e base real com expoente real

```
ER G2
(-3)**(-3)=     0
(-3)**( 0)=     1
(-3)**( 3)=   -27
ER G1
( 0)**(-3)=99999
( 0)**( 0)=     1
( 0)**( 3)=     0
ER G2
( 3)**(-3)=     0
( 3)**( 0)=     1
( 3)**( 3)=    27
```

```
(-3.)**(-3)=-.37037036E-01
(-3.)**( 0)= .10000000E+01
(-3.)**( 3)=-.27000000E+02
ER G4
( 0.)**(-3)= .99999999E+99
( 0.)**( 0)= .10000000E+01
( 0.)**( 3)= .00000000E-99
( 3.)**(-3)= .37037036E-01
( 3.)**( 0)= .10000000E+01
( 3.)**( 3)= .27000000E+02
```

(*) O tradutor do Fortran Básico do IBM 1620 faz 0.**0. = 0.




onde aparecem mensagens de erro antes de cada caso anómalo.

Como dissemos em 3.1, não se admite base inteira e expoente real.

## 4.1 Números de referência

As instruções são normalmente obedecidas em sequência, pela ordem de escrita, mas é possível alterar essa ordem utilizando instruções de salto.

Às instruções chave para a alteração da estratégia associam-se então números entre 1 e 9999 que se chamam números de referência. Ocupam no cartão as colunas 2 a 5, com o algarismo das unidades na coluna 5.

Os números de referência são independentes da ordem de escrita das instruções e podem ser atribuídos a qualquer instrução, ainda que não sejam necessários.

## 4.2 Instruções de salto

### 4.2.1 *GO TO n* (salto incondicional)

Esta instrução indica que a próxima instrução a ser obedecida é a que tiver o número de referência $n$.

Exemplos:

*GO TO* 1
*GO TO* 7298

### 4.2.2 *GO TO* $(n_1, n_2, \ldots, n_m), i$ (salto calculado)

$n_1, n_2, \ldots, n_m$ são números de referência e $i$ é uma variável inteira sem índices. A próxima instrução a ser obedecida será a de referência $n_1$, $n_2$, ..., ou $n_m$, conforme $i$ tiver respectivamente os valores 1, 2, ..., ou $m$.

Exemplo:

*GO TO* (10, 10, 15, 8), $N$

Com $N = 3$ o programa continuará na instrução de referência 15. Um valor de $N$ menor que 1 ou maior que 4 produz um erro de efeitos pràticamente imprevisíveis.

### 4.2.3 *IF* (*a*) $n_1, n_2, n_3$

*a* é uma expressão aritmética e $n_1$, $n_2$, $n_3$ são números de referência.

Conforme for $a < 0$, $a = 0$ ou $a > 0$, assim o número de referência da instrução seguinte será $n_1$, $n_2$ ou $n_3$.

Exemplo:

*IF* (*A* (*I*,*J*)**B*(*K*) — *D*) 5,6,1

### 4.2.4 *IF* (*SENSE SWITCH i*) $n_1, n_2$

$n_1$ e $n_2$ são números de referência e *i* é um dos inteiros 1, 2, 3, 4, números de quatro chaves de comando (manual) do computador.

Se a chave *i* estiver ligada, o programa prossegue em $n_1$; se estiver desligada, continua em $n_2$.

Exemplo:

*IF* (*SENSE SWITCH* 2) 100,24

O programa continua na instrução com referência 100 se a chave 2 estiver ligada, e na instrução com o número 24 se estiver desligada.

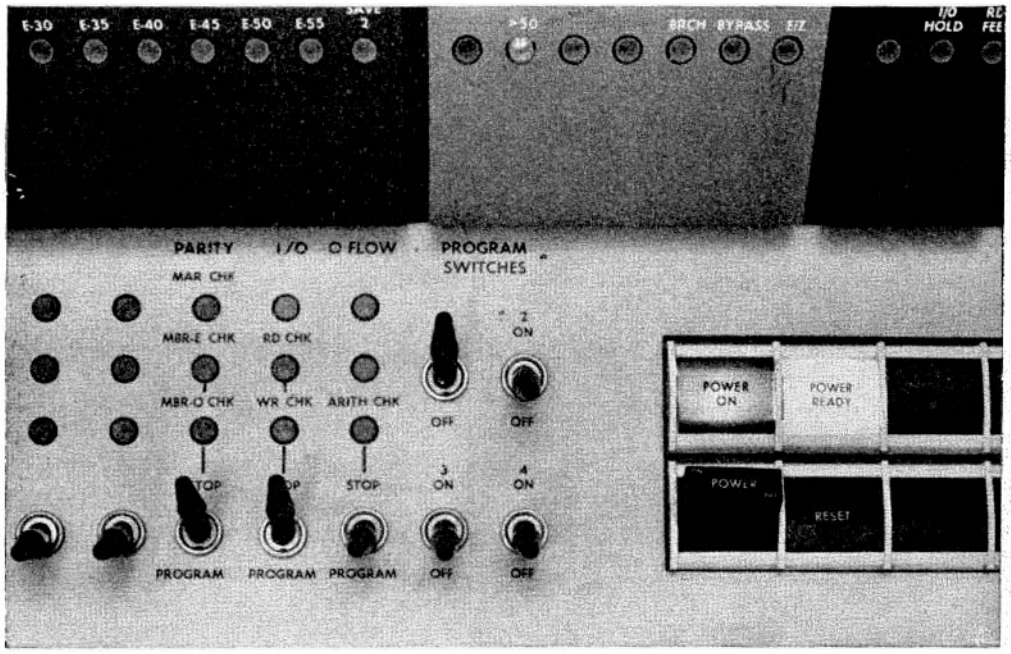



FIG. 11

## 4.3 Instruções de paragem

### 4.3.1 *PAUSE* ou *PAUSE n* (*)

Ao encontrar esta instrução a máquina pára, mas o programa pode prosseguir carregando na tecla de comando START. *n* é um inteiro com cinco dígitos, compreendido entre 00000 e 39999.

Exemplos:

*PAUSE* , *PAUSE* 08745 , *PAUSE* 00003.

### 4.3.2 *STOP ou STOP n* (*)

Ao atingir esta instrução a máquina de escrever imprime a palavra *STOP* e o computador pára sem poder prosseguir o cálculo.

Exemplos:

*STOP* , *STOP* 00022

(*) *NOTA* — Imediatamente após a paragem do computador devido a instruções *PAUSE n* ou *STOP n*, podemos observar o valor de *n* da forma seguinte:

1) Posicionar o botão selector do painel de comando em OR2;
2) Carregar na tecla *DISPLAY MAR*.
   Cada um dos cinco dítigos de *n* aparece numa das filas horizontais de luzes à esquerda do botão selector, em representação binária. A figura mostra *n* = 08745.

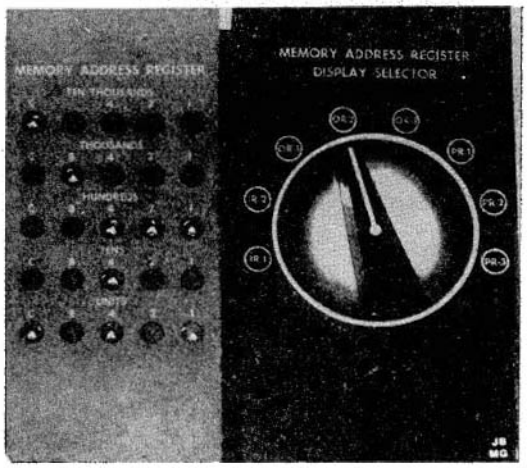



FIG. 12

## 4.4 Ciclos de repetição. As instruções *DO*

### 4.4.1 Forma geral

A forma geral das instruções *DO* é

$$DO\ n\ i = m_1\ ,\ m_2\ ,\ m_3$$

onde $n$ é um número de referência e $i$ uma variável inteira sem índices (chama-se-lhe *índice* da instrução *DO*) e $m_1$, $m_2$, $m_3$ são variáveis inteiras sem índices, tomando só valores positivos, ou constantes inteiras positivas sem sinal.

Quando $m_3$ é igual a 1 pode suprimir-se, escrevendo apenas

$$DO\ n\ i = m_1\ ,\ m_2$$

Exemplos:

$$DO\ 10\ J = 1, N$$
$$DO\ 150\ L1 = 10,\ 20,\ 2$$
$$DO\ 6\ KLM = K,\ L,\ M$$

A instrução *DO* é uma ordem para executar as instruções que a seguem, até à de referência $n$, inclusivè, com $i=m_1$, e repetir com $i=m_1+m_3$, $i=m_1+2m_3$, $i=m_1+3m_3$, etc., enquanto for $i\leqslant m_2$, prosseguindo depois na instrução que segue a instrução de referência $n$.

Se a instrução *DO* não menciona $m_3$, executa-se o ciclo de instruções com $i=m_1$, $i=m_1+1$, $i=m_1+2$ ... até $i=m_2$.

Durante cada ciclo, a variável $i$ pode utilizar-se como elemento de cálculo ou como índice de outra variável (desde que esteja nos limites da dimensão). Para exemplificar estas instruções, considere-se o seguinte fragmento de programa

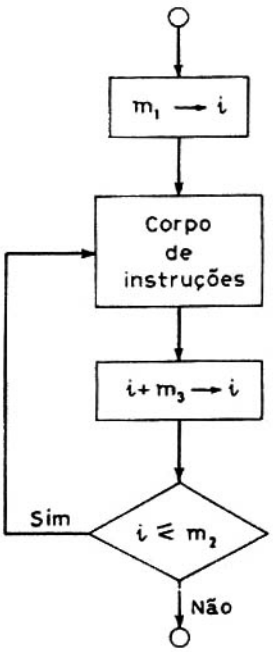
```
    DO 9K = 1,11,4
  9 A(K) = B(K) + C(K)
 10 ...
```

Neste exemplo o ciclo é constituído ùnicamente pela instrução 9. O programa repete-o com $K = 1$, 5 e 9 executando sucessivamente

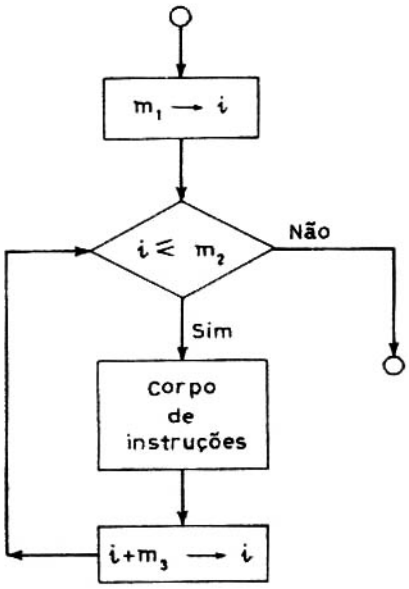
$$A(1) = B(1) + C(1)$$
$$A(5) = B(5) + C(5)$$
$$A(9) = B(9) + C(9)$$

Uma nova adição de 4 ao valor de $K$ excede 11. O programa liberta-se então do controle do *DO* e prossegue na instrução 10.

Fazemos notar que uma instrução como *DO* $1I = 2,1$ obrigaria a executar o ciclo com $I = 2$, embora fosse excedido logo de início o limite superior do índice. Esta característica do *DO* Fortran pode por vezes dar origem a contratempos.




Esquema de um *DO*

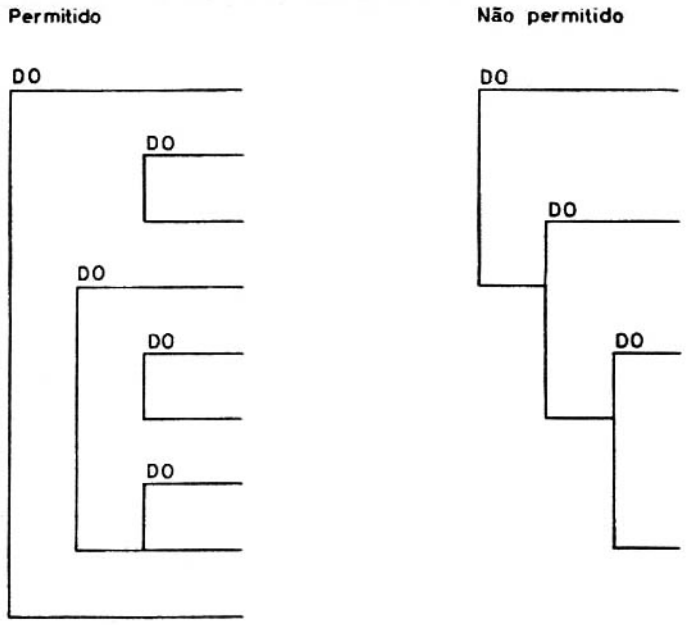



Esquema diferente do *DO* Fortran, onde não se executa o ciclo quando $m_1 > m_2$.

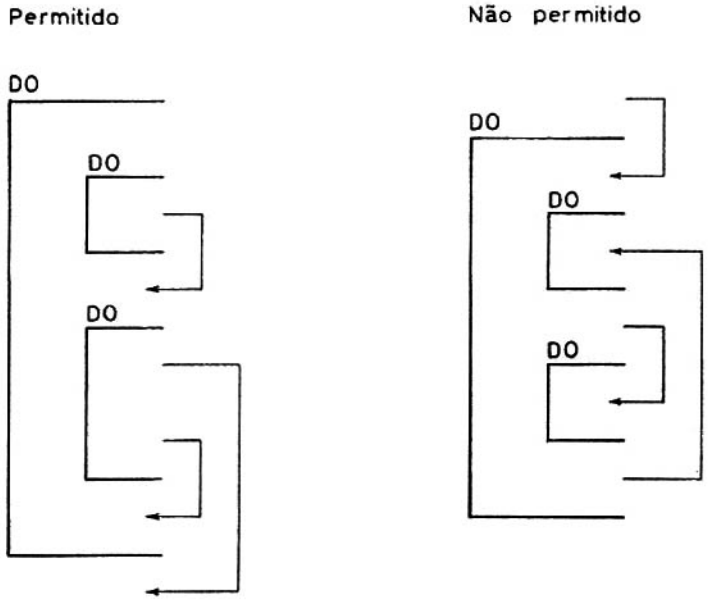
### 4.4.2 Instruções *DO* dentro de instruções *DO*

Podemos incluir uma ou mais instruções *DO* dentro do campo de acção doutra instrução *DO*. Há, no entanto, certas regras que é necessário respeitar:

1) Se o campo de uma instrução *DO* contiver outra instrução do mesmo tipo, todo o campo desta deve estar contido no campo da primeira. O esquema seguinte ilustra a regra.




2) Não se permite saltar de fora do campo de uma instrução *DO* para o interior desse campo, por meio de instruções *IF* ou *GO TO*, pois tais saltos não permitiriam o funcionamento normal do índice da instrução *DO*.



### 4.4.3 Conservação dos valores do índice

Quando o programa abandona o campo de um *DO*, depois de o índice ter tomado o maior valor que não excede $m_2$, dizemos que houve uma saída *normal*. Após uma saída normal, o valor do índice do *DO* não está definido e não pode ser usado sem ser definido de novo.

Se se efectua um salto do interior para o exterior do campo, o valor corrente do índice mantém-se, podendo continuar a ser utilizado.

### 4.4.4 Restrições sobre cálculos no campo de um *DO*

Permitem-se quase todos os tipos de instrução no campo de um *DO*. Exceptuam-se as instruções que produzam uma nova definição do índice ou de qualquer dos parâmetros $m_1$, $m_2$, $m_3$, que acompanham o índice. Por outras palavras, em cada ciclo *DO* o domínio de variação do índice deve estar bem determinado antes de ser executado.

A primeira instrução do campo de um *DO* não pode ser não executável, como *DIMENSION* e outras instruções estudadas adiante (*FORMAT*, *END*, *EQUIVALENCE* e *COMMON*).

A última instrução do campo não pode ser de salto.

## 4.5 *CONTINUE*

*CONTINUE* é uma instrução não operante.

Usa-se frequentemente como última instrução do campo de um *DO* para evitar que apareça uma instrução de salto em último lugar, ou simplesmente para criar uma instrução referenciada.

Exemplo:

```
    DO 1  I = 1,5
    IF  (A(I)) 1, 2, 1
1 CONTINUE
```

## 4.6 *END*

Esta instrução deve aparecer como última instrução de um programa. Serve apenas para informar o programa tradutor de que não há mais instruções desse programa para traduzir.

# Cap. 5 — INSTRUÇÕES DE ENTRADA E SAÍDA DE INFORMAÇÃO.

Para a resolução de qualquer problema é geralmente necessário fornecer à máquina, depois de registar o programa em memória, os valores particulares dos parâmetros (dados do programa) que nele figuram. Ao inverter uma matriz, por exemplo, é necessário ler a ordem e os elementos dessa matriz, se o programa, para ser geral, os não define.

Quer a entrada de dados quer a saída dos resultados é feita pela máquina de escrever ou por meio de cartões.

Dispõe-se de quatro instruções para efectuar estas transferências de informação — duas para a entrada e duas para a saída.

Cada uma contém o número de uma instrução de especificação (ver 5.4) (*FORMAT*), que pormenoriza a maneira como entram ou saem os números, e uma lista de elementos a serem transmitidos.

Exemplo:

$$PRINT\ 3,\ A(4),\ B,\ C(5,2),\ X,\ I$$

Ao executar esta instrução o programa imprimiria sucessivamente os valores de $A(4)$, $B$, $C(5,2)$, $X$ e $I$ de acordo com as especificações dadas pela instrução de referência 3.

Um cartão tem capacidade para fornecer (receber) informação que ocupe até 80 caracteres.

Uma linha de papel da máquina de escrever tem capacidade para 87 caracteres.

## 5.1 Instruções de entrada

*a)* Por cartões:
*READ n*, Lista

De acordo com a instrução *FORMAT* com referência *n*, lêem-se de um ou mais cartões, sucessivos números até se esgotar a lista. Esses números são lidos, convertidos na representação inteira ou de ponto flutuante, e colocados no local designado pela variável a que dizem respeito.

Exemplos:

*READ* 30, *A*, *B*, *C*(1,2)
*READ* 1, *RAIZ* 1

*b)* Por máquina de escrever:
*ACCEPT n*, Lista

Esta instrução chama a máquina de escrever, põe o carreto à esquerda e o computador espera que o operador escreva os dados. Os pormenores de escrita são regulados pela instrução *FORMAT* de referência *n*.

Exemplo:

*ACCEPT* 30, *A*, *B*, *D*(3)

## 5.2 Instruções de saída

*a)* Por cartões:
*PUNCH n*, Lista

Os cartões são perfurados obedecendo à instrução *FORMAT* de referência *n*.

Exemplos:

*PUNCH* 1, *A*, *X*, *Y*
*PUNCH* 45, *SQ*12

*b)* Por máquina de escrever:
*PRINT n*, Lista

Exemplo: *PRINT* 2, *DELTAX*

Nota: Pode usar-se a expressão *TYPE* em vez de *PRINT*.

## 5.3 Listas

Uma lista de variáveis pode apresentar um aspecto mais complexo do que o apresentado nos exemplos anteriores.

Consideremos, por exemplo, a instrução

*PRINT* 1, *A*, (*B*(*I*), *C*(*I*,3), *I*=1,2), ((*P*(*I*,*J*), *I*=2,10,4), *Q*(*J*,1), *J*=1,3)

Os elementos seriam transmitidos pela seguinte ordem:

$A$, $B(1)$, $C(1,3)$, $B(2)$, $C(2,3)$, $P(2,1)$, $P(6,1)$, $P(10,1)$, $Q(1,1)$,
$P(2,2)$, $P(6,2)$, $P(10,2)$, $Q(2,1)$, $P(2,3)$, $P(6,3)$, $P(10,3)$, $Q(3,1)$

Quer dizer, com a lista indicada, a instrução é equivalente (do ponto de vista de ordem de transmissão) ao seguinte conjunto de instruções:

```
      ...
      PRINT 2, A
      DO 3 I=1,2
    3 PRINT 4, B(I), C(I,3)
      DO 5 J=1,3
      DO 6 I=2,10,4
    6 PRINT 7, P(I,J)
    5 PRINT 8, Q(J, 1)
      ...
```

É portanto possível formar listas equivalentes à execução de ciclos de leituras, sendo permitidos ciclos interiores a outros.

Um elemento de uma lista só é considerado depois de o anterior ter sido completamente tratado.

Assim, é possível executar

$$READ\ 1,\ M,\ N,\ ((A(I,J),\ J=1,N)\ ,\ I=1,M)$$

onde $M$ e $N$, depois de lidos, actuam no controle da parte seguinte da lista.

Outros exemplos de listas (*):

$READ\ 3,\ (A(I),\ (C(I,J),\ J=1,N),\ I=1,M)$
$PUNCH\ 4,\ (I,(J,A(I,J),\ J=N1,N2),\ I=M1,M2)$
$PRINT\ 5,\ ((I,J,A(I,J),\ J=1,I),\ I=1,N)$

Podem ainda figurar numa lista elementos especiais, como em

$$READ\ 1,\ A,\ B,\ C,\ X\ (4),\ Y$$
$$PUNCH\ 2,\ A,\ B,\ C,\ X\ (4),\ Y$$

onde $A$, $B$ e $C$ são variáveis dimensionadas.

---

(*) Numa versão antiga do tradutor, o 1.º e 2.º exemplos (em que o primeiro elemento de lista não pertencia ao ciclo mais interior) não eram aceites.

Se, por exemplo, a instrução de dimensão for

*DIMENSION* $A(5,5)$, $B(3)$, $C(2,2)$, $X(6)$

teremos 25 casas reservadas para $A$, 3 para $B$, 4 para $C$ e 6 para $X$.

A instrução *READ* obrigaria os 25 primeiros valores lidos a ocuparem sucessivamente todo o espaço de $A$ (correspondendo a leitura por colunas — ver 2.2.3), os 3 seguintes o espaço de $B$, os 4 imediatos o espaço de $C$, e, finalmente leria valores para $X(4)$ e $Y$.

Anàlogamente para a instrução *PUNCH*.

Deste modo, a lista anterior seria equivalente a

$A(1, 1)$, $A(2, 1)$, ..., $A(5, 1)$, $A(1, 2)$, ..., $A(5, 2)$, ...,
$A(1, 5)$, ..., $A(5, 5)$, $B(1)$, $B(2)$, $B(3)$, $C(1, 1)$, $C(2, 1)$,
$C(1, 2)$, $C(2, 2)$, $X(4)$, $Y$

## 5.4 Instruções de formato

### 5.4.1 Forma geral

As instruções de entrada e saída contêm, além da lista de variáveis que localizam os valores a transmitir, o número de uma instrução que determina a forma como a informação vai ser transmitida — instrução *FORMAT*.

Uma instrução de formato é do tipo

*n FORMAT* (especificações)

onde $n$ é um número de referência.

Exemplo:

2 *FORMAT* (*I*2,*E*14.2,*F*10.4)

Vejamos agora as especificações possíveis.

### 5.4.2 Especificações de campos numéricos

Dispomos de três tipos de especificações de formato, que designamos por $E$, $F$ e $I$ (*), para pôr em diálogo as variáveis Fortran de uma lista de entrada ou saída e os valores a ler ou escrever.

(*) Não confundir os tipos $E$, $F$, $I$ com variáveis $E$, $F$, $I$.

Para ler num cartão de dados um número com ponto decimal não seguido de expoente e transmiti-lo a uma variável real, usa-se a especificação $F$; se o número estiver seguido de expoente, como $10.46E-01$, usa-se a especificação $E$. Um número inteiro será atribuível a uma variável inteira pela especificação $I$.

Normalmente, em saída de informação numérica, variáveis inteiras exigem especificação $I$; variáveis reais podem dar os seus valores por especificação $E$ (e o número sai representado por um número com ponto decimal seguido de um expoente) ou $F$ (e o número sai com ponto decimal e sem expoente).

Mais precisamente, as especificações de formato para campos numéricos apresentam-se nas formas

$$Ew.d \quad , \quad Fw.d \quad \text{e} \quad Iw$$

onde $w$ e $d$ são inteiros não precedidos de sinal. $E$, $F$ e $I$ referem o tipo de especificação, $w$ o espaço reservado à representação do número (amplitude do campo), e $d$ o número de casas decimais que aparecem à direita do ponto decimal.

a) *A especificação Iw*

As instruções
```
    M=-1
    I=6278
    K1=12345
    L=-82
1   FORMAT (I2, I4, I5, I6, I4)
    PRINT 1, M, I, I, K1, L
```
fariam imprimir, com $k=5$

$$-16278b6278b12345b-82$$

$b$ não aparece na folha de resultados; indica aqui um espaço em branco. Na saída de resultados o sinal de um inteiro positivo é ignorado, de modo que não é essencial reservar-lhe uma unidade de campo. O sinal de um número negativo ocupa uma das posições $w$; se $w$ não for suficientemente grande para conter todos os dígitos de um número e o sinal (se for negativo), o número sai na forma correcta, num campo de amplitude $k+1$, mas é impressa uma mensagem de erro.

Exemplo: As instruções
```
    I=-12350
    PRINT 1, I
1   FORMAT (I3)
```
fariam imprimir (se $k = 5$) *ER F8 —12350*

b) *A especificação Ew.d*

Com
```
B=-8.1415935
C=-B
PRINT 3, B, B, B, C
3 FORMAT (E14.8, E10.3, E11.2, E16.6)
```

teríamos (com $f=8$)

$$-.81415935E+01-8.141E+00-814.15E-02bbb81.415935E-01$$

Na saída de resultados, se uma especificação não puder ser satisfeita ($d>w-6$), como $E11.8$, a máquina utiliza automàticamente $E(f+6).f$ e assinala erro, prosseguindo o programa.

Se $f \leqslant w-6$ serão impressos ou perfurados os $f$ dígitos de mantissa; se $f>w-6$ sairão os $w-6$ dígitos mais significativos.

c) *A especificação Fw.d*

Com
```
A=123.45678
PRINT 1, A, A, A, A
1 FORMAT (F10.5, F10.3, F8.1, F6.0)
```

a máquina escreve (para $f \geqslant 8$)

$$b123.45678bbb123.456bbb123.4bb123.$$

Quando o comprimento do campo não for suficiente para escrever o resultado acompanhado do sinal — ou de um espaço em branco (que substitui +), a máquina transmite o número na forma $E(f+6).f$ anotando o erro.

```
PRINT 2, A, A
2 FORMAT (F6.3, F4.0)
```

originava a escrita (para $f=8$) de

```
ER F8  .12345678E+03
ER F8  .12345678E+03
```

d) *Transmissão de valores com mudança de tipo de representação*

É também possível ler um inteiro e transmiti-lo a uma variável real ou ler um número com ponto decimal (com ou sem expoente) e atribuir *a sua parte inteira* a uma variável inteira.

Assim, por exemplo, são compatíveis as duas instruções

```
    READ 1, REAL, I, J
1   FORMAT (I5, F10.5, E10.2)
```

e equivalentes a

```
    READ 1, N, A, B
1   FORMAT (I5, F10.5, E10.2)
    REAL = N
    I = A
    J = B
```

É necessário acautelar que a parte inteira de $A$ e $B$ caiba nos $k$ dígitos de precisão de $I$ e $J$.

Anàlogamente, é possível fazer sair a parte inteira de uma variável real, sem ponto decimal, ou transmitir o valor de uma variável inteira, com ponto decimal seguido ou não de expoente.

```
    PRINT 2, D, K, L
2   FORMAT (I10, F5.1, E10.4)
```

é equivalente a

```
    M = D
    G = K
    H = L
    PRINT 2, M, G, H
2   FORMAT (I10, F5.1, E10.4)
```

Se $D$, $K$ e $L$ contivessem os valores 24.7, —30 e 258 respectivamente, a máquina de escrever dar-nos-ia

```
bbbbbbbb24—30.0b.2580E+03
```

Com $k = 5$ (precisão de inteiros) e com 513987.6 em $D$ o computador daria mensagem de erro ao tentar transmitir 513987 porque se excederam as $k$ posições do registo interno de inteiros, embora a especificação $I10$ fornecesse dez casas. Obter-se-ia

```
ER E9
bbbbb99999—30.0b.2580E+03
```

### 5.4.3 Especificações de campos alfanuméricos

Em Fortran é possível ler ou escrever informação alfanumérica. Dispomos de várias especificações para esse fim.

a) *A especificação wH*

A especificação *wH* é seguida na instrução *FORMAT* por *w* caracteres alfanuméricos (*).

Por exemplo: 1 *FORMAT* (23*HINFORMACAO ALFANUMERICA*)

Notar que os espaços em branco são também caracteres e têm de contribuir para o número *w*.

As instruções

*PRINT* 4
4 *FORMAT* (8*HINTEGRAL*)

obrigariam a máquina de escrever a imprimir a palavra «*INTEGRAL*»

A acção de *wH* pode ser modificada durante a execução do programa. Quando é lida uma parte dum cartão em obediência a *wH*, os *w* caracteres da instrução *FORMAT* escritos inicialmente são substituídos pelos *w* caracteres colhidos no cartão.

Na execução das instruções

*READ* 4
4 *FORMAT* (8*HINTEGRAL*)

se for lido o cartão

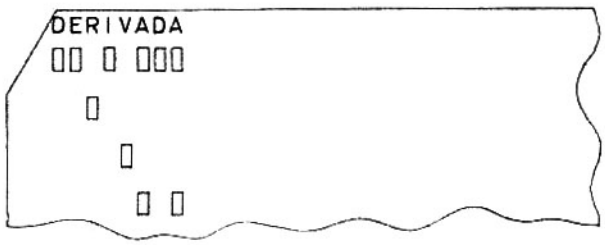



FIG. 13

(*) No IBM 1620 podemos fazer uso, além dos símbolos da fig. 5 (pág. 17), de @ e $.

Em *wH*, o *H* recorda o nome de Herman Hollerith, que cerca de 1890 deu um notável impulso ao uso de máquinas de cartões perfurados.

a instrução *FORMAT* passa a actuar, para efeitos de escrita, como se fosse 4 *FORMAT* (8*HDERIVADA*).

A leitura de novo cartão de acordo com o mesmo formato voltaria a substituir os $w$ caracteres.

Anàlogamente se passaria com leitura, não por cartões, mas por máquina de escrever, com as instruções

```
    ACCEPT 4
4 FORMAT (8HINTEGRAL)
```

b) *A especificação* $wX$

Campos de espaços em branco podem ser especificados por $wX$ onde $w$ designa o número de espaços; $w$ não pode exceder 87 de cada vez que se utiliza $wX$.

Exemplo:

```
    PRINT 3, A, B
3 FORMAT (E14.8, 14X, E14.8)
```

fará aparecer 14 espaços em branco entre os dois números a transmitir.

O mesmo *FORMAT*, associado a uma leitura, ignoraria os 14 elementos a seguir ao primeiro campo.

c) *A especificação* $Aw$

Esta especificação, ligada a uma instrução de leitura, transmite $w$ símbolos alfanuméricos para a memória, e associa-os *a uma variável*; inversamente, permite a saída de $w$ símbolos alfanuméricos pela chamada dessa variável numa instrução de escrita.

Assim, com $k=5$, $f=8$

```
    READ 1, INF, ALFA
1 FORMAT (A2, A4)
    PRINT 1, INF, ALFA
```

poder-se-ia ler num cartão 6 símbolos, como *DELTA* = e reproduzi-los depois com a máquina de escrever. *INF* conteria os símbolos *DE*, e *ALFA* os símbolos *LTA* =. A uma variável que comece por *I*, *J*, *K*, *L*, *M* ou *N* pode associar-se um número de símbolos que não exceda $\frac{k}{2}$ ou $\frac{k-1}{2}$ conforme $k$ for par ou ímpar *.

(*) No computador, cada símbolo alfanumérico ocupa 2 posições de memória.

No espaço reservado por uma variável que não comece por aquelas letras podem colocar-se símbolos em número não superior a $\frac{f}{2}$ ($f$ par) ou $\frac{f-1}{2}$ ($f$ ímpar).

O quadro seguinte esclarece algumas situações de utilização de variáveis inteiras e reais como meio de memorizar informação alfanumérica.

| Cadeia de símbolos | k | Registo inteiro | Cadeia de símbolos | f | Registo real |
|---|---|---|---|---|---|
| ABI | 6 | $\bar{4}14249$ | C | 2 | $\bar{4}3\bar{0}0$ |
| JKR | 7 | $\bar{5}152590$ | +$* | 7 | $\bar{1}01314\bar{0}\bar{0}0$ |
| STZ | 8 | $\bar{6}2636900$ | —/,( | 8 | $\bar{2}021232\bar{4}\bar{0}0$ |
| 109 | 6 | $\bar{7}17079$ | =@ | 5 | $\bar{3}334\bar{0}\bar{0}0$ |
| 2$ | 4 | $\bar{7}213$ | b.) | 6 | $\bar{0}0030\bar{4}\bar{0}0$ |
| ETA | 6 | $\bar{4}56341$ | ETA | 6 | $\bar{4}5634\bar{1}00$ |

O registo interno de símbolos numa variável real (com excepção das cadeias começadas por «b», «.» ou «)», em que o primeiro dígito de codificação é um zero) constitui um número de ponto flutuante e pode intervir nos cálculos.

Uma instrução como

$$IF\ (SIMB - 0.456341)\ 1,\ 2,\ 3$$

supondo $f \geqslant 6$, averigua se os símbolos associados à variável *SIMB* precedem, igualam ou seguem «alfabèticamente» a palavra *ETA*.

Também se podem manipular as variáveis inteiras que registem cadeias alfanuméricas.

Assim com $k = 6$ em

$$IF\ (INT - 456341)\ 1,\ 2,\ 3$$

teríamos situação análoga à anterior.

### 5.4.4. Associação de especificações de formato

Em geral, uma instrução *FORMAT* pode associar especificações de qualquer dos tipos referidos.

Assim, com as instruções

```
      PRINT 1, L, B
    1 FORMAT (2X,2HP(,I3,2H)=,E14.8)
```

tendo $L$ o valor 45 e $B$ o valor 2.5, a máquina de escrever imprimiria

$$bbP(b45)=b.25000000E+01$$

As especificações $wH$ e $wX$ não necessitam de ser seguidas de vírgula. O formato anterior é equivalente a

$$1\ FORMAT\ (2X2HP(I3,2H)=E14.8)$$

### 5.4.5 Utilização múltipla de especificações

Numa instrução de leitura ou escrita o programa procura alternadamente a lista de variáveis e as especificações de formato. Repete o processo até esgotar a lista e assim cada variável exige, em princípio, uma especificação.

a) *Grupos de especificações*

Em geral as especificações apresentam-se separadas por vírgulas. No entanto, permite-se que conjuntos de uma ou mais especificações apareçam dentro de parêntesis, como por exemplo em

$$1\ FORMAT\ (F10.4,\ (E14.8,I6))$$

Se durante uma transmissão de dados ou resultados as variáveis forem em número superior às especificações numéricas, o programa volta ao último parêntesis aberto da instrução *FORMAT* e prossegue, *mudando de linha de informação.*

Por exemplo:

```
    READ 2, A, B, C, D
2 FORMAT (E14.8)
```

atribuiria valores às 4 variáveis lendo as 14 primeiras colunas de 4 cartões sucessivos.

```
    PRINT 3, A, B, C, D, I, E, KB
3 FORMAT (F3.1, F14.6, E14.4, (E14.8,I5))
```

dar-nos-ia $A$, $B$, $C$, $D$ e $I$ numa linha, e na linha seguinte, $E$ e $KB$, segundo as duas últimas especificações.

b) *Repetição de especificações e grupos de especificações*

Pode condensar-se a escrita de especificações iguais e consecutivas, o mesmo sucedendo a grupos de especificações.

Assim,

```
    1 FORMAT (E12.3, E12.3, E12.3, (I4,I6,I4,I6))
```

é equivalente a

```
    1 FORMAT (3E12.3, 2(I4,I6))
```

A multiplicidade (menor que 100) dum grupo de especificações é tomada em conta quando a lista de variáveis obriga a um retrocesso na consulta das especificações.

```
      I=-6
      PRINT 2, I, I, I, I, I, I, I
    2 FORMAT (2(I2,I4))
      ...
```

produziria

```
-6bb-6-6bb-6
-6bb-6-6
```

5.4.6. Uso do símbolo / nas instruções *FORMAT*

A ocorrência do símbolo / numa instrução *FORMAT* deve interpretar-se como mudança de linha de informação (novo cartão ou nova linha da máquina de escrever).

```
      PRINT 3, A, B, C, D
    3 FORMAT (F4.2, E14.8/F15.6/E14.7)
```

daria origem à impressão de $A$ e $B$ numa primeira linha, de $C$ na segunda e de $D$ na terceira.

```
      READ 7, N, (X(I), I=1,N)
    7 FORMAT (I5/(F10.5))
```

manda ler $N$ num cartão e pede novo cartão para ler $X(1)$ devido ao /; para ler $X(2)$ tem de se retroceder na especificação $F10.5$ pedindo-se por isso novo cartão; anàlogamente para $X(3)$, ..., $X(N)$. Em suma, é preciso um cartão para cada um dos elementos da lista.

$N+1$ sinais / consecutivos, separando duas especificações, correspondem a $N$ linhas de informação em branco; o mesmo efeito se opera com $N$ sinais / antes (depois) da primeira (última) especificação.

## 5.5 Entrada de dados

Em princípio a linha de informação deve seguir as exigências das especificações.

Assim, o cartão figurado poderia ser lido de acordo com

5 *FORMAT* (*F*8.2, 2*X*, *E*9.1, *I*5)

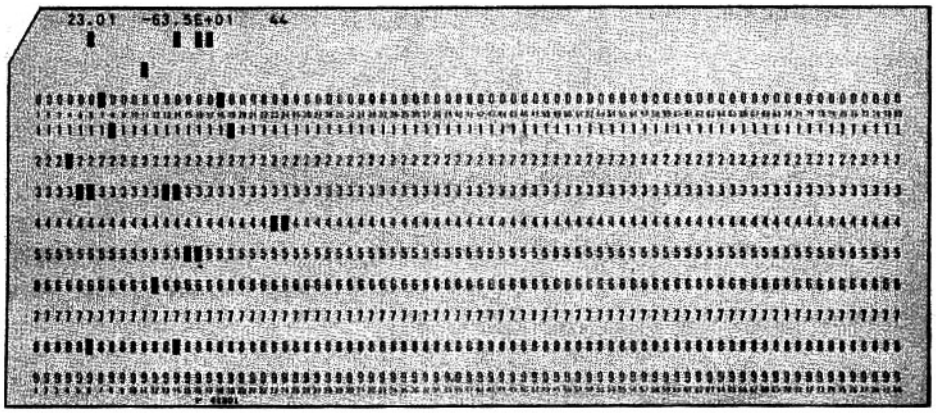



FIG. 14

A informação relativa a cada campo numérico deve estar encostada à direita. Um espaço em branco num campo numérico é tomado como zero. Admitem-se no entanto algumas liberdades no fornecimento de dados:

*a)* O sinal + pode ser omitido ou substituído por um espaço em branco.

*b)* O campo de um expoente pode não utilizar 4 colunas. A letra *E* pode mesmo ser omitida desde que o expoente seja acompanhado de + ou —.

Assim,

$$E+03,\ E03,\ Eb3,\ E+3,\ E3,\ +03,\ +3$$

são formas possíveis para o campo de um expoente (*).

(*) Mas *Eb*—3 é lido incorrectamente.

*c)* O ponto decimal pode ser suprimido (mas não substituído por espaços em branco); neste caso a especificação, *Ew.d* ou *Fw.d*, através de *d*, indica a localização do ponto.

*d)* Se a posição do ponto decimal do número não corresponde à especificação de formato, o computador dá prioridade à informação fornecida pelo número. Assim a constante $-568.35-2$ poderia ser lida como

$-568.35\times10^{-2}$ mesmo com a especificação *E*14.5.

Notar que o dígito 2 deveria ocupar a 14.ª casa do campo.

## Cap. 6 — FUNÇÕES DEFINIDAS POR EXPRESSÕES.

Podem utilizar-se em Fortran II outras funções além das de biblioteca (3.2). As funções de que nos ocuparemos neste capítulo são definidas por expressões Fortran.

Definem-se por instruções da forma

«*nome da função*» («argumentos»)=«expressão»

O nome da função é formado por 1 a 6 símbolos, letras ou números, sendo o primeiro uma letra.

Os argumentos, separados por vírgulas, são variáveis sem índices, que só dizem respeito à definição da função. São variáveis mudas. Designam locais especiais ainda que possam ter o nome de outras variáveis usadas no programa.

À direita do sinal = aparece a expressão, que pode conter os argumentos, outras variáveis (sem índice) e funções.

As instruções definidoras de funções colocam-se antes da primeira instrução executável do programa.

Se uma função definida por uma expressão utiliza uma função definida da mesma forma, esta deve preceder a primeira.

O nome de uma função não pode aparecer na sua própria definição.

Exemplos:

$HOMOG\ (X)=(A*X+B)/(C*X+D)$

$F3(Y)=((Y+1.)*Y+1.)*Y+1.$

$DIST\ (X1,\ Y1,\ X2,\ Y2)=SQRTF\ ((X2-X1)**2+(Y2-Y1)**2)$

$PF3\ (A,B)=F3\ (A)*F3(B)$

Quando se pretende utilizar uma destas funções numa instrução aritmética do programa, basta escrever o primeiro membro da instrução definidora, depois de substituir os argumentos por expressões aritméticas.

Nas instruções de utilização já podem aparecer variáveis com índices.

Exemplos:

```
Z=1.-HOMOG (R**2+5.)
A=F3(B(I))+F3(C(I)+D(I))
```

O resultado depende dos valores das expressões aritméticas (da instrução de utilização) em correspondência com os argumentos e dos valores correntes das variáveis do programa utilizadas na instrução de definição. O tipo de representação do resultado é inteiro ou real, conforme o nome da função começa ou não por *I*, *J*, *K*, *L*, *M* ou *N*.