NOTAS TÉCNICAS

VOLUME I

Nº 1

MANUAL
DE
PROGRAMAÇÃO EM UNICODE

(Tradução com adaptações e ampliações)

preparado por:

Alfredo Marques
Erasmo Ferreira
Georges Schwachheim
Ricardo Palmeira

CENTRO BRASILEIRO DE PESQUISAS FÍSICAS
Av. Wenceslau Braz, 71
RIO DE JANEIRO
1961

APRESENTAÇÃO

Com êste fascículo, o Centro Brasileiro de Pesquisas Físicas inicia a publicação das Notas Técnicas, que se vêm juntar às Notas de Física e às Monografias de Física. Enquanto as Notas de Física são reproduções antecipadas de trabalhos científicos originais realizados neste Centro e enviados para publicação em conceituadas revistas especializadas, pretendemos que as Notas Técnicas constituam um meio de divulgação de trabalhos de interêsse para as técnicas modernas de física e ciências correlatas.

As Monografias de Física são reproduções de cursos de nível universitário de graduação ou pós-graduação.

O conjunto destas publicações reflete os objetivos e as atividades dêste Centro, voltadas para a cada vez maior associação da pesquisa científica pura com a técnica e o ensino especializado.

J. Leite Lopes
Diretor Científico

PREFACIO

O Instituto Brasileiro de Geografia e Estatística possui um Computador Univac 1105, e o Centro Brasileiro de Pesquisas Físicas tem tido oportunidade de utilizá-lo. As perspectivas de continuidade da colaboração entre as duas instituições, nos levaram a preparar essa edição dêsse Manual de Programação em Unicode.

Esperamos que não só elementos do Instituto Brasileiro de Geografia e Estatística e do Centro Brasileiro de Pesquisas Físicas aproveitem êsse Manual. Um Centro de Processamento de Dados, do qual o Computador Univac será a espinha dorsal, está sendo criado, e a êle poderão recorrer indivíduos ou instituições que tenham problemas de cálculo numérico a resolver. Será desejável que o próprio interessado em resolver um problema possa preparar o seu programa para o Computador - e isso será possível graças à simplicidade e à eficiência da programação em linguagem científica tratada nesse Manual.

Para o físico, que trabalha com matemática complexa e que frequentemente se defronta com cálculos proibitivamente extensos, um computador como êsse é um instrumento de inestimável valor. Na dura concorrência científica internacional, entre as desvantagens que nós no Brasil levamos figura a falta de computadores apropriados a cálculos de alta complexidade que aumentem a eficiência de nosso trabalho. Mais talvez do que ninguem sabemos por isso apreciar o valor dêsse computador. Nosso desenvolvimento não permitiu ainda que tenhamos em nossos Centros de pesquisa científica instrumentos dispendiosos como computadores eletrônicos de elevadas velocidade e capacidade. A utilização dêsse magnífico Univac abrirá muitas possibilidades para a pesquisa científica e o desenvolvimento tecnológico no Brasil.

Se a divulgação dêsse Manual entre indivíduos e instituições de todo o país possibilitar o seu uso por um grande número de pessoas e na solução de um grande número de problemas, nosso pequeno esfôrço estará amplamente justificado.

Em grande parte êsse Manual é tradução do correspondente em língua inglesa fornecido pela Remington Rand Corporation. Houve

algumas adaptações e modificações. Alguns exemplos foram acrescentados aos já existentes na versão original. Outros exemplos, informações adicionais e soluções de problemas de interêsse geral em diferentes campos de trabalho serão publicados em anexos ao Manual.

Críticas sôbre falta de clareza, sôbre impropriedade no uso de termos técnicos em língua portuguesa, sôbre omissões que sempre existem num texto com êsse - tôdas serão benvindas. Sugestões poderão facilitar o trabalho de preparação de anexos ou corrigendas ou talvez de uma futura edição melhorada.

Agradecemos a todos que colaboraram na preparação dessa publicação, especialmente ao pessoal do Serviço Nacional de Recenseamento e à Srta. Yara Chaudon que fêz criteriosamente todo o trabalho de datilografia.

Alfredo Marques

Erasmo Ferreira

Georges Schwachheim

Ricardo Palmeira

ÍNDICE

INTRODUÇÃO

COMPONENTES DO UNIVAC 1105

A. Observações Gerais.

O computador UNIVAC 1105 executa várias operações. Além das operações aritméticas (adição, subtração, multiplicação e divisão), ele executa comunicações com unidades de fita magnética (os Uniservos), para ler dados das fitas ou nelas escrever, comunicações com equipamento de perfurar cartões de 80 colunas, etc. Para dar conta de sua tarefa o computador é dividido em várias seções, cada uma das quais executa um tipo especial de operação e é solicitada sòmente durante a execução dessa particular operação.

É conveniente separar o computador pròpriamente dito do equipamento externo, utilizado para trazer dados para o computador ou receber resultados do computador.

Os passos que devem ser dados para, partindo da proposição de um problema, obter resultados podem ser sumarizados da seguinte forma:

a) Análise do problema.

Durante esta fase o problema é decomposto numa série de passos lógicos por uma pessoa familiarizada com o tipo de problema matemático, físico, comercial ou de engenharia.

b) Codificação do problema.

O problema analisado é então codificado numa linguagem acessível ao computador. Isto constitui a programação do problema.

c) Preparação do programa para alimentação do computador.

O programa deve ser colocado no computador. Isto feito, dá-se a partida na máquina que então operará automàticamente segundo o programa. O programa deverá ser preparado de maneira a ser aceito pela máquina. Possibilidades para a 1105 são: fita magnética, fita de papel, cartão perfurado.

d) Entrada do programa no computador.

Esta é a primeira fase durante a qual o operador é envolvi -

do. Êle coloca, por exemplo, fitas magnéticas nas unidades manipuladoras de fitas (Uniservos) e prepara o computador para a partida por intermédio de operações manuais no Painel Supervisor de Contrôle. Após terem sido dados estes passos o operador dará a partida na máquina. Ao cabo desta operação o programa estará na máquina.

e) Computação.

Esta é a primeira fase que usa o computador para a finalidade que justifica a sua existência, isto é, computar. O computador segue o programa passo a passo e executa automàticamente todas as operações necessárias para chegar aos resultados programados. Esta fase é também supervisionada pelo operador.

f) Saida de resultados.

Executada a computação, os resultados tornados disponíveis devem ser retirados do computador. O momento em que isto deve ocorrer é determinado pelo programa. Esta fase envolve, novamente, comunicações entre o computador e o chamado equipamento externo associado, por exemplo, fita magnética. O tipo de receptor (fita magnética, fita de papel, impressão direta pela máquina de escrever associada, etc.) é também comandado pelo programa.

A saída de resultado pode ser programada de modo a ocorrer a qualquer momento durante a computação ou no intervalo entre computações.

g) Preparação de cópias dos resultados.

Dependendo do formato dos resultados, uma cópia deles pode ser obtida por meio do equipamento não associado à máquina. Êste equipamento não está fìsicamente ligado ao computador, ao contrário do equipamento associado que está ligado ao computador de modo a permitir inter-comunicações a qualquer tempo.

Em resumo a situação é a seguinte:

1) Análise do problema } Programação
2) Codificação }
3) Preparação da entrada
4) Entrada no computador } Sob a supervisão e contrôle do operador
5) Computação }
6) Saída de resultados }

B. Discussão Geral das Seções do Computador.

Vamos admitir que os passos 1, 2 e 3 acima já foram dados e que estamos prontos para executar o passo 4. Nessa fase muitos dados são introduzidos no computador. Êsses dados consistem de comandos também chamados instruções que são o resultado da programação, e constantes ou outros dados necessários para a computação.

A questão que nós temos de considerar em 1º lugar é: onde vai parar tôda esta informação. A resposta é: o computador possui uma "seção-reservatório" muitas vêzes chamada memória. Tôda informação vai para o reservatório. Para cada informação poder ser usada posteriormente, deve haver um meio de tomar conhecimento dela no momento em qué isto seja necessário.

Para isto a memória é dividida num número muito grande de repartições individuais chamadas registros. Cada registro pode conter um número. Afim de referir a êste número é preciso que se dê um "nome" ao registro no qual o número foi colocado. Êsse "nome" é chamado o endereço do registro.

Assim cada número é colocado num registro da memória e poderá ser reconhecido mais tarde porque o registro possui seu endereço individual.

A memória consiste de dois tipos de meios de armazenagem: a memória de núcleos magnéticos e a memória de tambor magnético. A memória de núcleos magnéticos contém um número menor de registros porém é de mais fácil acesso que a do tambor.

As operações aritméticas são executadas na unidade aritméti-

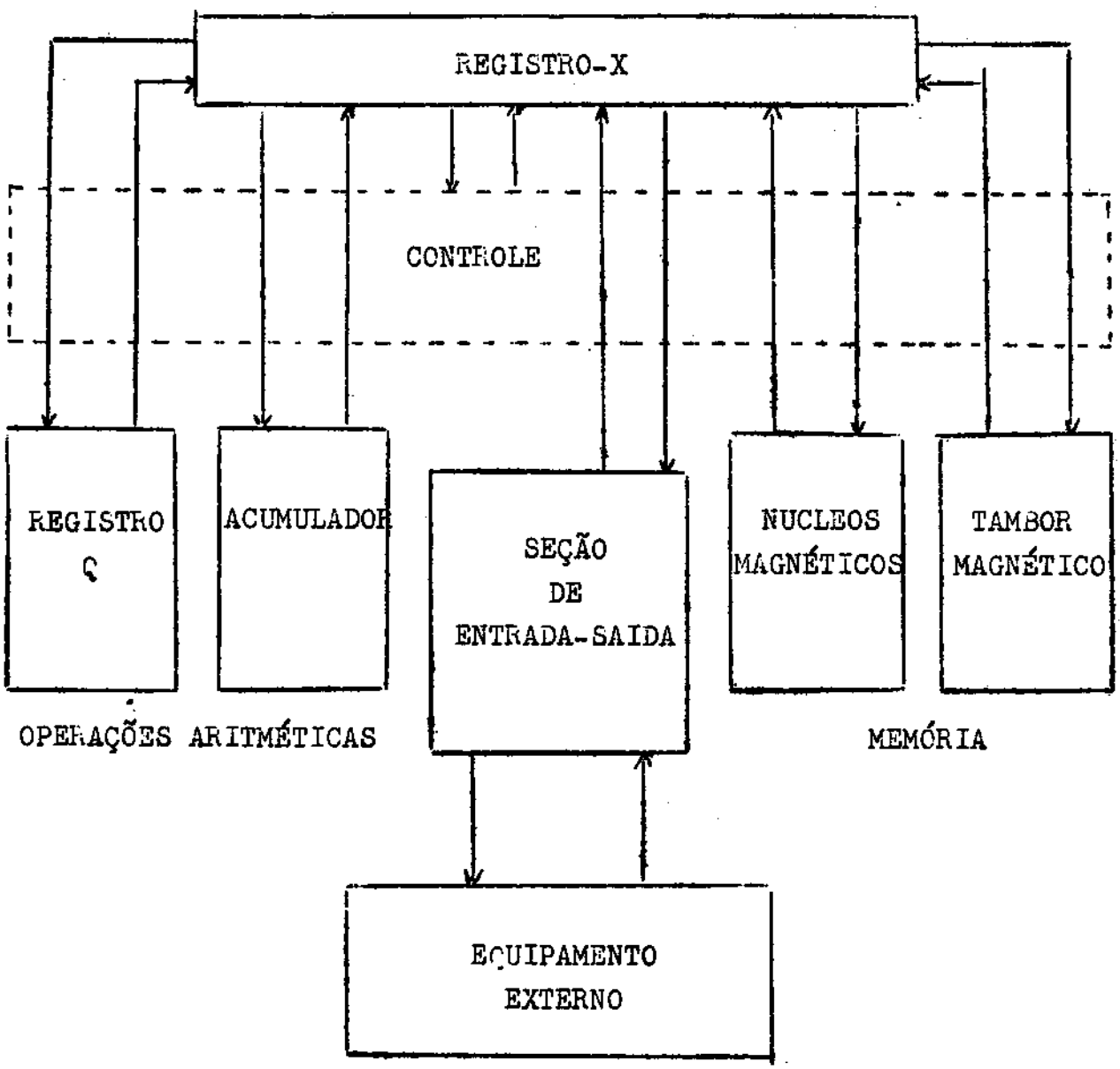
ca do computador. Ela consta de três partes importantes: o Acumulador, o Registro-Q e o Registro-X (Registro de câmbio).

No Acumulador são realizadas as adições, multiplicações e divisões. O Registro-Q retém o quociente durante divisões e é também usado em outras operações. O Registro-X é usado durante as operações aritméticas e durante todas as tranferências de números entre registros da memória e da unidade aritmética bem como entre fontes de entrada e saida.

Todas as operações do computador requerem que a máquina obedeça a uma sequência única de passos. Para conseguí-lo existe uma seção de contrôle que estabelece e controla essa sequência.

A última seção do computador é a seção de entrada/saída. Ela regula e controla todas as comunicações entre o computador e o equipamento externo associado.

A Figura 1 representa um diagrama lógico das seções do computador, mostrando o fluxo de dados de uma seção para outra. Observe como o Registro-X é utilizado para transmissões de dados. Dados que vão de um lugar para outro no computador seguem sempre via X (mesmo que estejam sendo transferidos de um registro da memória de núcleos para outro da mesma memória).


REGISTRO-X
CONTROLE
REGISTRO Q
ACUMULADOR
SEÇÃO DE ENTRADA-SAIDA
NUCLEOS MAGNÉTICOS
TAMBOR MAGNÉTICO
OPERAÇÕES ARITMÉTICAS
MEMÓRIA
EQUIPAMENTO EXTERNO

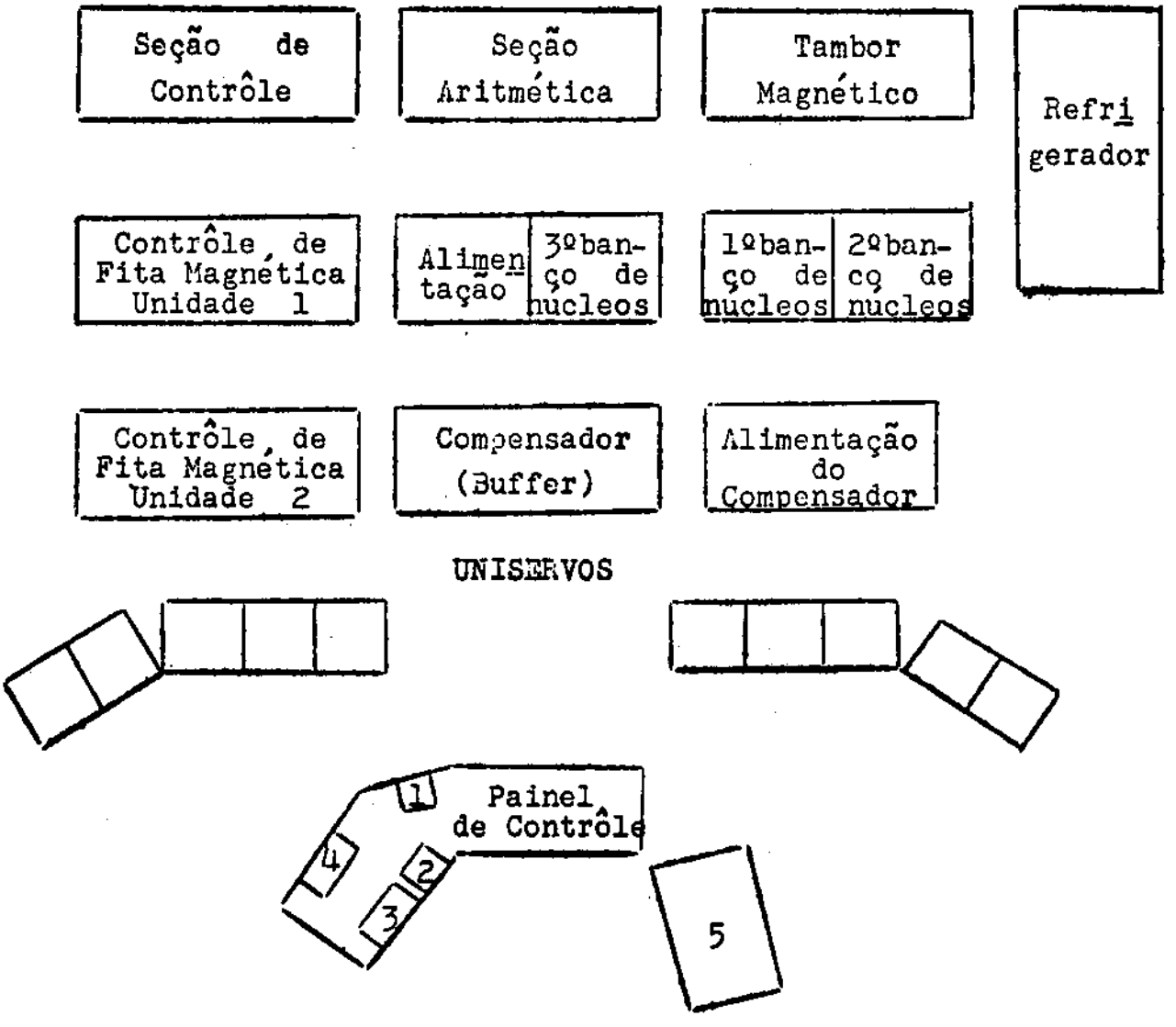


Fig. 2. Disposição dos Componentes do Computador

1. OSCILOSCÓPIO
2. FLEXOWRITER
3. UNIDADE DE LEITURA FERRANTI
4. PERFURADORA DE ALTA VELOCIDADE
5. PAINEL AUXILIAR

CAPÍTULO ZERO

O SISTEMA UNICODE

No UNICODE, a programação automática atingiu um elevado grau de desenvolvimento. O UNICODE capacita o engenheiro, o físico, o matemático, enfim qualquer pessoa familiarizada com a álgebra, a tornar-se um programador, a exprimir seu problema de uma maneira concisa, numa forma fácil de aprender, a qual o UNIVAC CIENTÍFICO 1103 ou 1105 pode utilizar para produzir um programa de cálculo. O UNIVAC pode ser empregado para codificar um programa para qualquer problema matemático com solução numérica. Sua melhor aplicação verifica-se no campo dos problemas não repetitivos, isto é, problemas para os quais são desejadas soluções rápidas e que serão rodados no computador um número limitado de vêzes. Se um problema é para ser rodado muitas vêzes, o tempo de operação no computador poderá justificar uma codificação por programador profissional para economizar tempo do computador. Mas mesmo para tais problemas, o programador pode reduzir seu trabalho, utilizando o UNICODE para preparar partes do programa. Os problemas contendo seções que requeiram muitas operações lógicas, podem também ser mais eficientemente preparados por um programador profissional. Mas até em problemas dêste tipo, estas seções podem ser programadas em código , como sub-rotinas e colocadas na biblioteca UNICODE, sendo o resto do problema codificado em UNICODE. Assim, a biblioteca UNICODE de sub-rotinas é tal que é fácil acrescentar, subtrair ou mudar rotinas, o que permite ao grupo trabalhando no computador desenvolver uma biblioteca ajustada às suas necessidades específicas.

Expressão de Constantes e Variáveis

O sistema de UNICODE permite exprimir qualquer constante decimal. Tôdas as constantes, exceto aquelas usadas sòmente com variáveis ou números de ponto fixo , são transformados em números de ponto flutuante com oito algarismos decimais significativos.

Com apenas poucas exceções, qualquer conjunto de símbolos alfa-numéricos (o primeiro dos quais deve ser alfabético), pode ser usado para representar uma variável. O sistema UNICODE inclui no-

tação funcional com exatamente a mesma significação e a mesma notação que a comumente usada em matemática, a não ser em poucos casos. Uma função pode ser expressa com até quatro argumentos. Os argumentos podem ser todos variáveis rotuladas, todos variáveis não-rotuladas ou uma combinação de rotuladas e não-rotuladas. Uma equação pode ser considerada tanto em um sentido geral como incluida em uma igualdade numérica. Os parêntesis são utilizados da maneira usual, para ordenar os cálculos numa expressão matemática.

As Instruções do Sistema UNICODE

Para facilitar o uso de UNICODE, o número de instruções no sistema foi reduzido a um mínimo compatível com a simplicidade e flexibilidade no uso e com a eficiência na obtenção dos resultados. As instruções são:

COMPUTE - (1) Faz referência a um sub-programa ou (2) calcula uma variável usando sua equação de definição. Se a variável estiver escrita em forma rotulada serão conservados pelo computador, para uso em cálculos posteriores do mesmo programa, os valores dessa variável correspondentes aos valores que seu índice puder tomar. O número máximo de valores que se deseja que uma variável rotulada tome ao mesmo tempo é especificado pelo programador, na sentença chamada "Dimension". Se a variável não possui índice subscrito, só o mais recente valor atribuido a ela é conservado: cada vez que uma variável toma um novo valor, o anterior é apagado. Duas ou mais instruções COMPUTE podem ser combinadas em uma única sentença pela palavra de ligação AND. Equações escritas na região de execução (depois da palavra START no programa) exercem as mesmas funções que a instrução COMPUTE.

VARY - Controla os valores de uma variável usada repetidamente numa certa série de proposições. Consiste de três partes: modificação, alcance e transferência. A palavra de ligação WITH pode ser usada para combinar di -

versos componentes de modificação numa única instrução VARY, as variáveis tomando seus sucessivos valores concorrentemente.

JUMP - Transfere o contrôle para uma outra proposição.

IF - Transfere o contrôle para uma outra proposição, quando uma certa condição é satisfeita; se a condição especificada não fôr satisfeita, a computação prosseguirá como se a transferência de contrôle não houvesse sido ordenada, passando à proposição seguinte na ordem natural.

RESUME - Faz voltar o contrôle à instrução VARY, de modo que o processo de repetição com modificação pretendido com a instrução VARY siga seu curso, sendo continuado a partir do ponto em que foi usado pela última vez. Caso a transferência de contrôle para uma instrução VARY seja efetuada por outros meios, a sequência ordenada pela instrução VARY começa com as condições existentes no momento em que esta sentença é atingida.

LIST - Registra em uma fita magnética informação que será impressa pela Impressora de Alta Velocidade existente independentemente do computador. Todo resultado impresso terá a representação de ponto fixo, a menos que o ponto decimal apareça fora do alcance do grupo de algarismos significativos disponíveis. Se o número não pode ser escrito em notação de ponto fixo, usa-se a notação científica comum.

READ - Faz com que os dados numéricos decimais sejam lidos da fita de dados, convertidos em números binários de ponto flutuante, e armazenados nas regiões apropriadas da memória de alta velocidade.

TYPE - Permite indicar as variáveis cujos valores se quer sejam imprimidos pela máquina de escrever do computador

(Flexowriter); comumente usado para imprimir pequenas quantidades de informação. A variável referida pela instrução TYPE pode ser rotulada, não-rotulada, ou uma função.

PRINT - Permite a indicação de títulos ou de condições especiais durante o decurso da computação. Também usada em combinação com a máquina de escrever do computador, esta instrução imprime exatamente as palavras ou símbolos que seguem a palavra PRINT na sentença.

START - Indica o ponto no qual a execução de um programa deve começar; as proposições que o precedem não são executadas, são apenas usadas para referência.

STOP - Indica um ponto no qual a execução do Programa Objeto terá fim com uma parada do computador.

END OF TAPE - Indica que toda informação útil sôbre o problema, contida numa fita, foi fornecida.

DIMENSION - Especifica o número de valores de uma variável rotulada que deverão ser conservados a um dado tempo.

DELETE - Elimina sentenças inteiras do programa UNICODE, na fase de correção dos erros.

EXIT - Faz voltar o contrôle para a sentença do programa que referiu a pseudo-operação.

Execução Automática de Tarefas Rotineiras da Programação

A maior parte do tempo gasto em detalhes da programação de um problema é eliminada ou enormemente reduzida pelo UNICODE. A pseudo-operação, por exemplo, possibilita a redação de rotinas específicas para um problema particular. Isto torna disponível, como se fosse suplemento à biblioteca existente, qualquer sequência de instruções que serão usadas repetidamente no problema, quer os operan-

dos sejam vários ou não. A pessoa que estiver programando pode assim exprimir seu problema na linguagem UNICODE sem a confusão de excessivos detalhes. Uma pseudo-operação pode abranger até 20 operandos, bastando que a linha de referência (COMPUTE) e o título do sub-programa tenham o mesmo número de operandos correspondentes.

O mesmo símbolo de pseudo-operação pode ser utilizado tantas vêzes quantas forem necessárias em qualquer número de sentenças.

Tôdas as instruções UNICODE têm números que lhes são atribuidos pela pessoa que programou o problema. Assim, em suas correções para um programa UNICODE, o programador pode fazer referência a qualquer sentença pelo seu número. A pessoa que estiver programando não necessita de subdividir o Programa Objeto caso êle seja muito grande para o computador; UNICODE faz isso por si só, automàticamente, se houver necessidade.

O UNICODE reduz a tarefa de depuração a uma pequena fração do que seria com a programação em linguagem de máquina. Um grande fator que contribui para esta redução é a eliminação pelo UNICODE da fase de "arrumação" de programação, as modificações da instrução, escolhas de endereços, sub-divisão de programas e tarefas de rotina similares. Enquanto a facilidade de programação com o UNICODE elimina a maioria dos erros, aqueles que ainda ocorrem são fàcilmente localizados com a ajuda dos vários testes de erros incorporados a êste sistema de programação automática. São fáceis também as correções num programa UNICODE. Uma fita contendo as informações corretas é montada num outro Uniservo e encaixada no programa original para produzir o programa corrigido.

Preparo dos Programas em UNICODE para Entrada no Computador

Não é necessário colocar-se o programa UNICODE em cartões perfurados e depois fazê-los passar através de um conversor de registros em cartões para registros em fita magnética - o programa pode ser escrito diretamente com a máquina Unityper na fita magnética. Cada linha da cópia do programa UNICODE obtida nessa máquina conterá 120 símbolos (incluindo símbolos de espaços) representando um bloquete de informação em fita magnética. Fitas adicionais podem ser usadas para manipular um programa UNICODE que seja grande

demais para um único carretel de fita; estas fitas são adaptadas ao programa original da mesma maneira que as fitas de correção.

A descrição do programa em linguagem de máquina produzida pelo UNICODE é também fácil de se entender. Imprimida na Impressora de Alta Velocidade essa descrição do programa dá numa lista, lado a lado, os números das sentenças do programa original e as correspondentes instruções geradas no Programa Objeto (Programa em linguagem de máquina obtido após a compilação do programa escrito em UNICODE). Também está incluida a localização das variáveis rotuladas tanto no núcleo como no tambor e as das variáveis não-rotuladas no núcleo. O conjunto de constantes é impresso. Rotinas incluidas no programa e Rotinas da Biblioteca que são usadas em mais de um segmento, são mostradas em cada segmento em que ocorrem.

Há duas versões disponíveis de UNICODE, uma para o UNIVAC ≠ 1103A e outra para o 1105. As únicas diferenças entre as duas estão nas instruções de manejo da fita. Ambas requerem instruções em ponto flutuante e 5 unidades Uniservo. Um ajuste MJ1 de início exigirá sete Uniservos para a compilação. Apenas uma bancada de matrizes de 4096 palavras da memória de alta-velocidade é necessária. Entretanto, a Fita Mestra do UNICODE pode ser fàcilmente modificada de modo a fornecer Programas-Objeto que possam usar duas ou mais matrizes da memória de alta-velocidade.

CAPÍTULO 1

Êste capítulo ilustra o aspecto geral do programa, do sub-programa e da sentença no sistema UNICODE. Uso de letras maiúsculas indica as sentenças que obrigatòriamente têm que aparecer.

PROGRAMA - Os últimos cinco bloquetes (linhas) no primeiro bloco (conjunto de 6 linhas) de cada programa são destinados ao título do programa e a uma descrição do problema. No caso de não haver título ou descrição a dar, estes bloquetes devem ser cheios com espaços. O primeiro bloquete dêste primeiro bloco deve necessàriamente conter o título UNICODE PROGRAM . Em todo êste manual o símbolo Δ indica "espaço". Na Unityper êste símbolo é obtido como ao dar-se o espaço numa máquina de escrever comum. Todo programa em UNICODE deve ser escrito como segue:

Folha de Programa UNICODE pag.		
Número das sentenças (Posições 1 a 6)	Instruções (Posições 7 a 120)	
	UNICODE PROGRAMΔ.	1º bloquete
	Título e Descrição	2º ao 6º bloquete do 1º bloco
	Sentença de "Dimension" Equaçces	Inicia-se 2º bloco na fita
	START Δ.	Indica que seguem-se os comandos de calculo
	Prcpcsições STOPΔ.	
	Sub-programas	EXIT deve aparecer ao menos uma vez em cada sub-programa
ZZZZZZ	END OF TAPE Δ.	

Fig. 1 - Forma geral de um programa em UNICODE

SUB-PROGRAMA - Um sub-programa em um problema indica o uso de uma pseudo-operação. A forma geral de um sub-programa é semelhante à de um programa, exceto que a primeira sentença de um sub-programa é reservada ao seu título. Para clareza na figura abaixo, imagine-se que o símbolo da pseudo-operação é OPER e que os operandos são A e B.

Número das sentenças (Posições 1 a 6)	Instruções (Posições 7 a 120)	
	OPER (A, B)Δ.	Título
	Proposições	
	EXITΔ.	

Fig. 2 - Forma Geral de um Sub-Programa

SENTENÇA - A sentença é a unidade fundamental da linguagem UNICODE, consistindo de um número designativo da sentença, e de uma instrução (a qual é interpretada por UNICODE).

Número de Sentença (Posições 1 a 6)	Instrução (Posições 7 a 120)

Fig. 3 - Forma Geral de uma Sentença

Número da Sentença	Instruções
38	VARY Z 1 (0.1) 100 SENTENCE 39 THEN JUMP TO 5 Δ.
10	F(0) = 13 Δ.
17	COMPUTE Y(I,J) Δ.

Δ. é o símbolo que indica o fim da sentença

Fig. 4 - Exemplos de 3 sentenças não relacionadas

CAPÍTULO 2

NOTAÇÃO

CONSTANTES - O sistema UNICODE permite a expressão de qualquer constante decimal. Com exceção dos números indicativos das sentenças, tôdas as constantes que contêm um ponto decimal são convertidas em números de ponto flutuante com oito algarismos decimais significativos. Isto permite a expressão de zero ou qualquer número entre 10^{-38} e 10^{38}. Inteiros sem um ponto decimal podem ou não ser convertidos a ponto flutuante, dependendo de serem ou não usados juntamente com números ou variáveis de ponto flutuante. Números de ponto fixo no sistema UNICODE são definidos como inteiros. A grandeza de um número de ponto fixo não pode exceder 999999. Esta restrição de grandeza nas constantes de ponto fixo escritas em um programa UNICODE não impede a geração, uso ou extração de números em ponto fixo maiores do que 999999 na execução no Programa Objeto. Desde que a combinação de símbolos espaço-ponto(Δ.) indica o fim de uma sentença, uma constante que não tenha parte inteira deve ser escrita com um zero ponto (0.---.) e não apenas com ponto (.---). Os símbolos + - * / indicam soma , subtração, multiplicação e divisão respectivamente.Exemplos:

123.95
0.12395* 10 POW 3
12395* 10^{-2}

Não se pode usar vírgula em lugar de ponto decimal.

VARIÁVEIS - Um qualquer conjunto de 1,2,3,4,5 ou 6 símbolos alfabéticos ou numéricos, o primeiro dos quais deve ser uma letra, pode ser usado para exprimir uma variável. Exemplos:

X
CAT
EE2L1
D81958

Nenhuma das várias combinações de símbolos que são usados nos nomes das rotinas da Biblioteca de rotinas (Ver cap. 11) pode ser usado como variável. Assim por exemplo o símbolo EXP, que indica a rotina que calcula a função exponencial, não poderá ser usado para designar uma variável em um programa. Também as palavras chave da linguagem UNICODE não podem ser usadas como variáveis. Assim os seguintes símbolos, pelo menos, estão excluídos do uso como variáveis:

FLEXPT	DIMENS	JUMP
GENPOW	START	STOP
VAREXP	VARY	END
LN	COMPUT	EXIT
EXP	READ	POW
SQRT	LIST	NOT
FLTCVT	TYPE	TAPE
LISTRN	PRINT	WITH
READRN	IF	THEN
INTCVT	RESUME	AND

Qualquer variável começando por I,J,K,L ou M é definida como uma variável de ponto fixo no sistema UNICODE. Tôdas as outras são variáveis de ponto flutuante.

ÍNDICES - Uma sequência de símbolos ou expressões matemáticas escritas dentro de parêntesis imediatamente após uma variável é usada para denotar os índices dessa variável. Qualquer variável usada como índice deve começar com uma das letras I,J,K,L ou M. Índices múltiplos em uma variável, num máximo de quatro, são separados por vírgulas. Um índice não pode ter um índice e pode assumir apenas valores inteiros não negativos. **Exemplos:**

X(5)
A(0)
Y(I,J)
Z(I+3,J+K)

Chamaremos as variáveis que possuem índices de variáveis rotuladas. Em um dado programa uma variável é usada sem índice ou com índice, mas as duas formas não podem ser usadas simultâneamente. Assim, se se especifica no início de um programa que a variável X terá um índice, X(J), o UNICODE não aceitará o uso no mesmo programa da variável não rotulada X, ou da mesma variável com dois ou mais índices, X(J,K) por exemplo.

EQUAÇÕES E FUNÇÕES - Uma equação na linguagem UNICODE é a definição de uma variável (em ponto fixo, em ponto flutuante, rotulada ou função escrita à esquerda de um sinal de igualdade por uma expressão matemática que está à direita. Nenhum sinal de operação (+ - / , etc) ou símbolo da Biblioteca de Rotinas pode aparecer à esquerda; logo os únicos símbolos permitidos à esquerda de equações escritas antes da instrução START são: variável ( ) ; , Em equações que apareçam depois de START, constantes são também permitidas à esquerda, além dos símbolos acima, mas elas podem apenas aparecer como valores de sub-índices. Exemplo: X(J,5,K) =
A computação do valor de uma variável definida por uma equação escrita antes de START não é feita até que assim seja ordenado por uma instrução COMPUTE. Referência a tal variável pode ser feita tantas vêzes quantas se queira. Quando uma equação está numa sentença depois de START cálculo imediato da expressão ocorrerá, com a correspondente modificação do valor numérico da variável definida por ela.

Uma equação de ponto fixo é uma na qual o membro da esquerda é uma única variável de ponto fixo. Apenas essa variável pode aparecer à esquerda do sinal de i-

gualdade. Apenas cinco tipos de sinais de operação po dem ser usados em uma equação de ponto fixo além do si nal de igualdade. Êles são: + - * / e | . Parêntesis podem ser usados para agrupamento.

Símbolos de funções (rotinas da Biblioteca) ou expoentes não são permitidos em equações de ponto fixo. Quan do uma divisão é realizada com operandos de ponto fixo, o resto da divisão, caso exista, é ignorado e apenas a parte inteira do quociente é usada em computação poste rior. Exemplos de equações de ponto fixo são:

```
I=|J-K|+L*M/I .
KX50=K4+J7+159 .
J3=I*(K-L) .
```

O membro da esquerda de uma equação de ponto flutuante pode ser um dos seguintes:

1. Uma variável rotulada com seus índices. Exemplo: X(I,J,K,L) = . Apenas quando a equação vem de - pois de START pode um ou mais dos índices ser constante de ponto fixo. Exemplo: X(4,J,K)=
2. Uma função com argumentos, se a equação está antes de START. Exemplo: F(X, Y(I))=
3. Uma função sem argumentos se a equação vem depois de START. Exemplo: F=
4. Uma variável de ponto flutuante sem índices ou argumentos. Exemplo: W=

O membro da direita de uma equação de ponto flutuante pode conter quase quaisquer dos símbolos de operação do UNICODE, inclusive expoentes e símbolos designativos de rotinas da Biblioteca. Os símbolos não permitidos são: $>\!<$ e o uso da letra E para exponenciação. O símbolo POW ou expoentes numéricos são usados para exponenciação em equações. (Em fitas de dados - veja cap. 5 - e na sentença IF, a letra E é usada na notação científica usual).

Variáveis de ponto fixo não podem ser combinadas com constantes e variáveis de ponto flutuante em uma equação, exceto se usadas como índices.

Símbolos disponíveis para uso em expoentes são . - / e algarismos. Não mais do que quatro símbolos podem aparecer em sequência num expoente, e o - , se aparece, deve ser o primeiro símbolo. A equação seguinte mostra a variável U seguida por algumas sequências de símbolos no expoente:

$$F(U)=U^{1/2}+U^{-1/2}*U^{2}-U^{3.1416}\ \Delta.$$

Desde que não se pode usar letras em expoente, o símbolo de operação POW é necessário. Assim X^Y é escrito X△POW△Y na linguagem UNICODE. Uma só equação antes de START pode ser escrita para cada variável; de outro modo ter-se-ia uma definição ambígua daquela variável.

Referências a rotinas da Biblioteca só podem ser feitas a partir de equações, e de um de dois modos. O método geral de referência aplica-se a qualquer número de operandos (até o máximo de 7) e é escrito por exemplo:

```
QUALK(A,B,C,D,E,F,G)
```

onde QUALK é o nome da rotina da Biblioteca e A,B,C,D, E,F e G são os operandos. Quando uma rotina de Biblioteca tem um só operando, a referência pode ser escrita, por exemplo,

```
SIN△A
```

onde SIN é o nome da rotina e A é o operando único.

Se tôdas as condições seguintes aplicam-se ao símbolo de uma variável de ponto flutuante, êste símbolo é considerado como representando uma função:

1. É o primeiro símbolo de uma equação escrita antes de START.

2. Não aparece na sentença de DIMENSION.

3. É seguido pela abertura de parêntesis.

Os argumentos de uma função devem ser variáveis de ponto flutuante; assim, um argumento não pode começar por uma das letras I,J,K,L,M.

O sistema UNICODE permite a uma função ser expressa com até quatro argumentos. Estes argumentos podem ser todos variáveis simples, variáveis rotuladas ou uma combinação de variáveis simples e rotuladas. Se uma variável é usada como argumento de uma função, ela não pode ter mais do que um índice.

Quando uma variável rotulada serve como argumento de uma função, parêntesis dentro de parêntesis são necessários para separar o índice do argumento.

O sistema UNICODE não permite expressão direta de uma função que tenha outra função como seu argumento.

Um valor apenas de uma função é retido a um dado tempo. Armazenar um valor enquanto outro é computado pode ser feito igualando-o a uma outra variável. Numa referência direta à função obtem-se apenas o valor que foi calculado por último.

Para ilustrar algumas das observações acima sôbre funções, suponhamos que temos a seguinte sequência de sentenças em um programa UNICODE:

1. DIMENSION F(43,9), X(3,9), Z(50)Δ.
2. F(I,K)=H-Z(K)Δ.
3. H(R,S)=$(R^2+S^2)^{1/2}$Δ.
4. V(P(K),Q)=P(I)-F(I,K)*Q+VΔ.
5. T=V+RΔ.
6. STARTΔ.

Aquí vemos que H e V representarão funções, mas F é uma variável rotulada desde que ela aparece na sentença

de DIMENSION. P(K) é uma variável rotulada usada como argumento da função V.

Tôdas variáveis que aparecem à esquerda de uma equação antes de START e que estão dentro de parêntesis são variáveis mudas para esta equação. No exemplo acima as variáveis mudas são:

Sentença 2. I e K

Sentença 3. R e S

Sentença 4. P, K e Q

Sentença 5. Nenhuma.

Estas variáveis mudas são análogas às da integração em matemática, que nos permitem escrever

$$\int_0^u f(x)dx = \int_0^u f(y)dy$$

onde x e y são as variáveis mudas da integração. Desde que os valores dos limites superior e inferior substituem x ou y após realizada a integração, não tem efeito no resultado se se usa a letra x ou a letra y.

No UNICODE as variáveis mudas meramente mostram a forma de uma função e quaisquer valores podem ser postos em lugar delas.

A sentença 3 acima poderia ser considerada como uma formula para o comprimento da hipotenusa de um triangulo retângulo com lados R e S. Se se deseja achar a hipotenusa de vários triângulos, não se precisa reescrever a fórmula cada vez, se ela está escrita como uma função antes de START.

Para resolver o problema com um triângulo de lados 3 e 4, pode-se escrever a seguinte sentença COMPUTE depois

de START:

10. COMPUTE H(3,4)Δ.

Quando H está sendo calculado, R terá o valor 3 e S terá o valor 4. Se posteriormente se deseja achar H com lados 7.96 e 13.21, escreve-se:

45. COMPUTE H(7.96,13.21)Δ.

Pode-se também colocar variáveis em lugar de R e S escrevendo, por exemplo:

50. COMPUTE H(A,B)Δ.

Assim H será calculado com os valores correntes de A e B nos lugares de R e S.

Na sentença 4 acima, a variável muda P é rotulada, e logo apenas uma variável com um sub-índice pode ser substituida por P. Mas qualquer constante positiva de ponto fixo ou variável de ponto fixo pode ser substituida por K. Na sentença de DIMENSION apenas Z aparece, com um índice. Assim, pode-se escrever depois de START:

30. COMPUTE V(Z(7),W)Δ.

P será igualado ao valor de Z, dependendo do índice usado à direita, K será 7 e Q será igual a W. Note-se que à direita da equação, P tem o índice I; logo, durante a computação P(I) será igualado a Z(I). Suponha que I = 37; então o cálculo à direita será efetuado como segue:

Z(37)-F(37,7)*W+VΔ.

Equações escritas depois de START não contêm variáveis mudas a menos que a equação apareça em um sub-programa

e contenha variáveis mudas referentes à pseudo-operação (veja-se o capítulo 4). Argumentos de uma função não devem ser escritos em seguida ao símbolo da função nos seguintes casos:

1. No membro da direita de uma equação.
2. No 1º membro de uma equação escrita depois de START.
3. No título de um sub-programa.
4. Numa sentença IF.
5. Quando usado como um operando de uma pseudo-operação em uma sentença COMPUTE.

Quando uma sentença COMPUTE refere-se diretamente a uma função, argumentos da função devem aparecer tanto se a sentença está no programa principal ou em um sub-programa.

HIERARQUIA DE OPERAÇÕES - Parêntesis são usados do modo usual para ordenar o cálculo em uma expressão matemática. Quando a ordem de cálculo não é especificada por parêntesis, as operações são efetuadas na seguinte ordem:

1. Funções de Biblioteca
2. Exponenciação
3. Multiplicação e Divisão
4. Adição e Subtração

Na equação $Z=X+Y^2/3 * SIN\ A-B$

o cálculo será efetuado como se estivesse escrito

$$Z=X+[(Y^2/3)*(SIN\ A)]-B$$

Se parêntesis são omitidos em uma sequência de operações de mesma hierarquia, o agrupamento é tomado como sendo da esquerda para a direita. Assim, X/Y * Z significa ((X/Y)*Z) calculando-se dos parêntesis mais internos para os mais externos. Z = sin y^2 é interpretado como $Z=(\sin y)^2$. Um termo como sen^2y causa a acusação de um êrro. Para se obter o seno de y^2 deve-se escrever $\sin(y^2)$. Uso semelhante deve ser feito de outras rotinas de biblioteca.

CAPÍTULO 3

REPERTÓRIO DE INSTRUÇÕES

Para facilitar a apresentação e explicação das instruções do UNICODE, a seguinte notação especial será usada:

1. As letras X, Y, e Z denotarão variáveis simples
2. As letras a, b e c denotarão expressões
3. As letras k, m e n denotarão números de sentenças. Essa convenção usada aquí por conveniência não deve ser interpretada como significando que os números de sentenças possam ser outra coisa que não constantes. Veja o Capítulo 6 para detalhes de como escrever números de sentenças.

COMPUTE - A instrução COMPUTE possibilita dois tipos de chamada:

1. Ela pode chamar uma equação escrita antes da instrução START.
2. Ela pode chamar um sub-programa.

A chamada de uma equação pela instrução COMPUTE faz executar o cálculo de variável ou função mencionada, usando sua equação de definição. Antes de tal cálculo ocorrer, a tôdas as variáveis que aparecem no lado direito da equação de definição devem ter sido atribuidos valores numéricos, quer como dados de entrada quer por meio de instruções VARY, quer através de anterior execução de suas respectivas equações de definição.

Exemplo:

```
COMPUTE Z .
```

(Quando as equações são escritas na região de execução - isto é, depois da instrução START - elas desempenham a mesma função que a instrução COMPUTE .

Quando durante a execução do programa uma tal equação é atingida, a variável é calculada usando a expressão à direita. Uma equação aparecendo depois de START não pode ser chamada por uma instrução COMPUTE. (Ver a seção EQUAÇÕES, Cap. 2).

A instrução COMPUTE também é usada como um meio de executar um sub-programa. Se chamarmos uma pseudo-operação de SUBR, a referência ao sub-programa para SUBR poderá ser:

COMPUTE SUBR (A,B,C,...).

onde os símbolos A,B,C,... denotam quaisquer variáveis ou funções (mas não expressões) as quais serão usadas como operandos pelo sub-programa. Entretanto, quando qualquer dos A,B,C,... representa uma função, seus argumentos não devem ser mencionados dentro da instrução COMPUTE. Como exemplo, se F(X(I), Y(J)) é uma função no problema, seu aparecimento na referência ao sub-programa deverá ser:

COMPUTE SUBR (F,...).Ver também o cap. 4.

Duas ou mais instruções COMPUTE consecutivas podem ser combinadas numa instrução única por meio da palavra de conexão AND.

Exemplo:

COMPUTE X AND Y AND Z .

Os cálculos são feitos da esquerda para a direita . No exemplo acima, X é calculado primeiro, depois Y e depois Z.

- A instrução VARY determina a sequência de valores de uma variável a ser usada em repetições de determinado número de instruções. Consiste de 3 partes:

as componentes Modificação, Alcance e Tranferência.

A componente Modificação identifica a variável afetada pela instrução VARY e determina os valores sucessivos assumidos por essa variável.

VARY X p(q)r--- .

À variável X é dado o incremento q desde o valor inicial p até um valor tal que $|r - X| < |q|$. Os quatro elementos da componente Modificação, podem ser todos elementos de ponto fixo ou de ponto flutuante, mas nunca uma combinação dos dois.

A componente Alcance especifica as sentenças que são controladas pela instrução VARY. Se o alcance de uma instrução VARY A inclui outra instrução VARY B, então o alcance completo de VARY B deve estar incluído no alcance de VARY A. A componente Alcance pode tomar qualquer das duas formas:

(1) ___SENTENCES k THRU m___

Essa forma indica que as sentenças, a partir da sentença de número k e até e inclusive a sentença de número m, serão repetidas cada vez que a variável na componente Modificação tomar um novo valor.

(2) ___SENTENCE k___

Êsse é um caso especial do mostrado acima no qual a instrução de número k é a única sujeita à repetição.

Em ambos os casos, a instrução de número k deve seguir imediatamente a instrução VARY.

A componente Transferência especifica o número da sentença para a qual será transferido o contrôle depois da sequência VARY ter sido executada; isto é, depois que tôdas as instruções dentro do alcance fo

ram executadas para todos os valores da variável mencionada na componente Modificação. A presença da componente Transferência não é necessária; entretanto, se não for especificada, o contrôle será transferido para a primeira sentença que segue o Alcance da instrução VARY no programa. Se especificada, a componente Transferência pode tomar uma das formas seguintes:

```
___THEN JUMP TO n .
___THEN RESUME  k .
```

Quando os Alcances de duas instruções VARY terminam exatamente na mesma sentença, a não especificação da componente de Transferência na instrução VARY interna tem um significado especial. Em vez da transferência usual para a instrução seguindo o alcance da instrução VARY, ela causa automàticamente um RESUME do VARY anterior, cujo alcance inclui a instrução VARY interna. Portanto, a especificação da componente de Transferência THEN RESUME k ou omissão da componente de Transferência, numa instrução VARY interna, têm efeitos idênticos, desde que a sentença número k, seja a sentença com a instrução VARY externa.

Só uma componente de Transferência do tipo THEN JUMP TO k, para uma qualquer instrução VARY (mesmo uma anterior) é usada, a variável modificada pela instrução VARY na sentença número k toma outra vez o valor inicial.

Ilustração do VARY interno:

```
1.  VARY X---SENTENCE 2 THRU 7.
2.  |
3.  | VARY Y---SENTENCE 4 THRU 7.[THEN RESUME  1]
4.  | |                          [THEN JUMP TO m]
5.  | |                          [THEN JUMP TO 1]
6.  | |
7.  |_|_____
8.
9.
```

A primeira instrução VARY (sentença 1) segue a regra convencional de Transferência; isto é, depois de completada a instrução VARY, o contrôle é transferido para a sentença 8. A Transferência do segundo VARY pode ser:

1. Não mencionada - o contrôle é transferido para 1, X toma o valor seguinte na sequência.
2. THEN RESUME 1 - o contrôle é transferido para 1, X toma o valor seguinte na sequência.
3. THEN JUMP TO m - o contrôle é transferido para a instrução de número m.
4. THEN JUMP TO 1 - o contrôle é transferido para 1, X retoma o valor inicial.

A palavra de ligação WITH pode ser usada para combinar várias componentes de Modificação em uma única instrução VARY quando as variáveis das componentes Modificação tomam seus valores sucessivos simultaneamente. As variáveis tomam novos valores na ordem na qual elas são escritas na instrução VARY. As variáveis de tôdas as componentes de Modificação tomam seus valores iniciais a primeira vez através do Alcance, seus segundos valores a segunda vez, etc... até que o valor limite de uma das variáveis é atingido e nesse ponto a instrução VARY é terminada. Um máximo de 15 WITH e portanto, 16 variáveis é permitido em uma instrução VARY qualquer.

VARY X p(q)r WITH Y s(t)u ---

Abaixo damos exemplos das formas que a instrução VARY completa pode tomar:

```
VARY X r(s)t SENTENCE k THRU m THEN JUMP TO n .
VARY X r(s)t SENTENCE k THRU m .
VARY X p(q)r WITH Y s(t)u SENTENCE k THRU m .
VARY I J(K)L SENTENCES m THRU n .
VARY X P(Q)R WITH M J(K)KA SENTENCE n .
```

Quando o valor inicial de uma variável, como está escrito na componente Modificação, é êle mesmo uma variável, essa variável deve assumir um valor numérico antes de na execução do programa ser atingida a instrução VARY. No terceiro exemplo dado acima, p e s devem tomar valores numéricos antes do início da execução da instrução VARY. Além disso, valores numéricos devem ser estabelecidos para tôdas as variáveis numa componente Modificação antes do fim da primeira passagem pelo Alcance do VARY. Quando se volta à sentença contendo a instrução VARY por meio de uma instrução RESUME, a sequência VARY é continuada no processo de modificação a partir do ponto em que foi deixada. Se entretanto a transferência para uma instrução VARY é efetuada por meio de um JUMP ou simplesmente pelo seguimento da ordem das sentenças UNICODE no programa, a sequência VARY começa com as condições iniciais como escrito na própria instrução VARY.

"THEN RESUME k," cujo uso foi explicado acima, só pode ser usado como a última parte de uma instrução VARY. Não pode ser acrescentada a nenhuma outro tipo de sentença UNICODE.

O número máximo permitido de sentenças VARY interpenetradas é de 30. Por sequências interpenetradas queremos nos referir ao caso em que o alcance de cada instrução de VARY exceto a primeira está contida no alcance do VARY precedente.

O número máximo permitido de sentenças VARY num programa é 50.

JUMP - A instrução JUMP transfere o comando de cálculo para outra instrução.

Exemplo:

JUMP TO SENTENCE k

Nesse exemplo, o comando é transferido para a sentença de número k, e daí prossegue em sequência a partir dessa instrução até ser interrompido por outra ordem de transferência.

A instrução JUMP pode ser usada em um sub-programa somente para transferir o comando para outras sentenças dentro do mesmo sub-programa. JUMP também não pode ser programado para transferir comando de fora para dentro de um sub-programa, nem mesmo para a primeira linha ou título do sub-programa.

- A instrução condicional IF transfere o comando de operação a outra instrução quando uma determinada condição é satisfeita. Se a condição não for satisfeita, o comando de cálculo será passar à instrução seguinte na sequência natural. Os exemplos seguintes mostram formas simples de sentenças IF.

```
IF X = Y, JUMP TO SENTENCE k .
IF X NOT = Y, JUMP TO SENTENCE k .
IF X < Y, JUMP TO SENTENCE k .
IF X > Y, JUMP TO SENTENCE k .
IF X < =Y, JUMP TO SENTENCE k .
IF X > =Y, JUMP TO SENTENCE k .
```

Além dêsses exemplos, uma instrução IF pode conter até três cláusulas se não houver duplicação de relações e se o mesmo conjunto de X e Y for usado em cada uma das três cláusulas. Exemplos:

```
IF X < Y JUMP TO SENTENCE 7, IF X > Y JUMP TO SENTENCE 74, IF X = Y JUMP TO SENTENCE 32.5 .

IF X > = Y JUMP TO SENTENCE 65, IF X < Y JUMP TO SENTENCE 44 .

IF X = Y JUMP TO SENTENCE 41, IF X NOT = Y JUMP TO SENTENCE 31 .
```

IF X (i, j, k, l)<= Y (ia, jt) JUMP TO SENTENCE 5.2, IF X (i, j, k, l)> Y (ia, jt) JUMP TO SENTENCE 3.

IF X>Y JUMP TO SENTENCE 4, IF X<Y JUMP TO SENTENCE 3.

Nos exemplos acima, uma transferência de comando para o número da sentença indicada é feito sòmente se a relação especificada entre X e Y existe. X e Y podem ser funções, constantes, variáveis rotuladas, ou não rotuladas mas não podem ser expressões.

Sinais de valor absoluto podem ser usados com X ou Y ou ambos. Sinais negativos também são permitidos. Sinais positivos são proibidos. Notação científica é permitida. Se X e Y são ambos constantes, uma comparação imediata é feita na compilação e a sentença é ou transformada em um JUMP ou eliminada com uma advertência de êrro.

RESUME - A instrução RESUME retorna o comando a uma instrução VARY para continuar o processo de modificação a partir do ponto em que o mesmo foi interrompido. Isso quer dizer que a variável da componente Modificação na instrução VARY assume o valor seguinte, em vez do valor inicial que tinha antes de entrar no alcance do VARY. A instrução RESUME pode sòmente referir-se a uma instrução VARY. No exemplo abaixo o comando é transferido para a instrução VARY de número k, a qual continua o processo de modificação.

Exemplo:

```
RESUME k
```

LIST A instrução LIST faz registrar em uma fita especificada, informação que deve ser impressa pela Impressora de Alta Velocidade. Qualquer resultado impresso terá uma representação em ponto fixo, a menos que o valor do número não esteja compreendido entre

0.1 e 10^9 - 1 inclusive, em cujo caso notação científica usual é usada, isto é, o número será dado em forma decimal entre 1 e 10 seguido pela letra e e uma potência de 10.

Uma sentença LIST pode fazer referência a até cinco quantidades, essas podendo ser quaisquer combinações de funções, variáveis rotuladas ou não rotuladas ou índices que possam ser requeridos. Se apenas uma variável é especificada, ela é impressa em cinco colunas, valores sucessivos sendo lidos através da página; para de duas a cinco variáveis, cada uma é impressa em uma coluna simples. Se se desejar imprimir cabeçalhos das colunas, êstes podem ser colocados dentro de parentesis em seguida ao título, o qual é colocado dentro de parentesis duplos. Os cabeçalhos e o título são ambos optativos. Se não se desejar nenhum cabeçalho para a coluna entre um cabeçalho ou título e outro cabeçalho, um par de parentesis sem nada dentro deve ser inserido na sentença no lugar apropriado. Vêr o exemplo C abaixo. Êsses cabeçalhos podem conter um máximo de 23 símbolos, e o título um máximo de 119 símbolos. Nunca há necessidade de indicação para omissão do título.

Devemos notar que todas as quantidades a serem impressas devem ser mencionadas explìcitamente. Se por exemplo os valores de uma variável rotulada e os índices são desejados, a sentença LIST deverá ser

LIST A(I, J), I, J, TAPE 2 .

O programador tem a opção de escolher um valor ou valores específicos de uma variável rotulada, inserindo um valor numérico para um ou mais índices. Isto ocasionará a não tomada em consideração dos valores dos índices no instante em que o código gerado está em operação, e os valores da variável serão impressos do acôrdo com os índices escolhidos.

Cada sentença LIST causa a impressão de um valor da variável mencionada, de modo que quando se deseja um conjunto de valores de uma variável rotulada a sentença LIST deve estar dentro do Alcance de uma sentença VARY apropriada. É possível, embora não recomendado, ter mais de uma sentença LIST no alcance de uma instrução VARY. Ter mais de uma, significa que cada conjunto de uma a cinco variáveis impressas por uma sentença LIST terá linhas de título separadas. Por outro lado, com sòmente uma sentença LIST, sòmente um conjunto de linhas de título é impresso, independente de quantas vezes a sentença é chamada dentro do ciclo de VARY.

Todas as sentenças LIST devem especificar a fita na qual o registro deve ser feito. Normalmente, isso significará um número específico de fita aparecendo na sentença. A critério do programador, êsse poderá ser uma variável em ponto fixo, a fita realmente a ser gravada dependendo do valor da variável ao tempo em que o código da LIST esteja em operação pela primeira vez. A designação das fitas devem ser ou constantes de ponto fixo ou variáveis de ponto fixo. Ponto decimal não deve ser usado em números de fitas - assim TAPE 1.0 ou TAPE 2. não são permitidos.

Os exemplos seguintes são supostos aparecer dentro de ciclos VARY apropriados:

A. LIST R(I, J), TAPE 3, ((INVERSE MATRIX)).

B. LIST F(X(I), Y(J), T), X(I), Y(J), T, I, TAPE K .

C. LIST C(I, K), I, K, TAPE 4, ((MATRIX MULTIPLICATION)), ( ), (ROW), (COLUMN) .

Então o exemplo A dá a seguinte gravação na fita Nº 3:

INVERSE MATRIX

R(I, J)

|R(1, 1)| [R(1, 2)] [R(1, 3)] [R(1, 4)] |R(1, 5)]
[R(1, 6)] -
- - - - - |R(1, J_{max})| |R(2, 1)| |R(2, 2)| - - - - -
- - - - - - - - - - - - - - - - - -[R(I_{max}, J_{max})]

O exemplo B produz o seguinte registro na fita Nº k (o número do uniservo é definido pelo valor de k):

F(X(I), Y(J), T)	X(I)	Y(J)	T	I
[F(X(1), Y(1), T)]	[X(1)]	[Y(1)]	[T]	[I]
[F(X(1), Y(2), T)]	[X(1)]	[Y(2)]	[T]	[I]
¦	¦	¦	¦	¦

etc, onde [ ] denota o "valor de".

O exemplo C produz a seguinte gravação na fita Nº 4:

MATRIX MULTIPLICATION

	ROW	COLUMN
C (I, K)	I	K
|C (0, 0)|	0	0
[C (0, 1)]	0	1
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.

onde I e R variam a partir de zero.

Note nos exemplos acima, que vírgulas devem sempre seguir cada ítem separado e que o número da fita deve seguir a enumeração do conjunto de variáveis e preceder o primeiro título central principal se existir algum. Note como os cabeçalhos das colunas seguindo o título central no exemplo C correspondem

aos nomes das variáveis dadas na primeira parte da sentença LIST, e estão concentradas com respeito a êles.

A instrução LIST não pode ser usada num sub-programa para fazer gravar uma variável muda.

EAD

A instrução READ ordena que informação decimal codificada em XS3 seja lida de fitas magnéticas especificadas, traduzida em números binários em ponto flutuante e guardada em regiões apropriadas da memória de alta velocidade.

Sòmente variáveis rotuladas podem ser lidas por essa instrução, e para uma tal variável, o código gerado pela instrução tentará ler tantas variáveis quantas necessárias para encher o espaço reservado para ela pela sentença DIMENSION correspondente (isto é, um conjunto completo). Se houver menos valores na fita do que os exigidos por essa sentença o computador parará e indicará êsse fato depois do que o programa poderá ser reiniciado, deixando de atribuir novos valores as variáveis que restam. Se houver mais valores na fita do que os exigidos, os excedentes serão desprezados.

Pode haver na fita diversos conjuntos de valores para uma dada variável rotulada (A por exemplo). Então, a primeira vez que o programa de execução encontrar READ A, a fita será posta em tal posição que o primeiro grupo seja lido. Se nêsse interim nenhuma outra sentença READ ordenando leitura de dados na mesma fita fôr encontrada, a segunda READ A fará a leitura do segundo grupo, e assim por diante até que o último grupo de dados tenha sido lido. Se então, uma outra instrução READ A fôr dada, dependendo da forma da sentença READ um dos dois cursos possíveis será tomado:

1. READ A .
2. READ A, IF END OF DATA, JUMP DO SENTENCE n .

Se tivermos a forma (1), a fita será novamente colocada na posição do primeiro conjunto A e êsses dados serão lidos novamente. Se tivermos a forma (2), então o salto especificado será efetuado.

Há um ponto importante que o programador deve ter em mente no caso acima descrito onde existe mais de um grupo de valores para uma ou mais variáveis. Se entre duas referências a uma variável com vários grupos de dados numa fita, ordenamos a leitura dos valores de uma variável diferente na mesma fita (tais referências naturalmente sendo feitas por meio de sentenças READ) então, ao encontrar a segunda referência à primeira variável, a fita será recolocada na posição do primeiro grupo de dados desta variável.

Muitas variáveis rotuladas separadas por vírgulas podem ser mencionadas numa sentença READ única.

Exemplos:

1. READ A, B, C, D, E, F, G, H, P, IF END OF DATA, JUMP TO SENTENCE k .

2. READ P, H, A, B, C, D .

É necessário fornecer ao programa em execução informação sôbre a localização de dados. Para detalhes sôbre isto, bem como sôbre formato, especificações, etc, deve-se consultar o capítulo 5 dêste manual.

A instrução READ não pode ser usada num sub-programa para fazer ler uma variável muda.

TYPE

Pela instrução TYPE se faz a designação de variáveis cujos valores se quer apareçam escritos pela máquina de escrever automática ligada ao computador. Ela é mais conveniente quando se quer obter pequenas quantidades de informação. A variável cujo valor se quer conhecer dessa forma pode ser rotulada, não rotulada ou uma função. Se ela é rotulada ou uma função os valores dos índices ou argumentos de-

vem ser determinados antes da execução da instru - ção.

Exemplo:

```
TYPE A(I, J), F(X), Y .
```

Se cada um dos valores de I e J fôr 2 e o valor de X fôr 2.5 quando a instrução TYPE fôr executada, a máquina de escrever escreverá:

```
A(2,2) = [A(2,2)]
F(2.5) = [F]
   Y   = |Y|
```

onde os colchetes representam o "valor de".

A instrução TYPE não pode ser usada num sub-programa para fazer imprimir o valor de uma variável mu - da.

PRINT

A instrução PRINT, também comandando a máquina de escrever ligada ao computador, permite indicação de títulos ou de condições especiais durante a execu - ção do programa. A impressão feita é sempre igual ao que aparece seguindo a palavra PRINT no programa.

Exemplo:

```
PRINT A(I, J) SINGULAR .
```

No exemplo acima a impressão será sempre

```
A(I, J) SINGULAR .
```

quer I e J tenham valores específicos durante a execução ou não.

Qualquer sentença PRINT não deve ser mais longa do que 6 linhas de escrita UNITYPER. Material que seja mais longo que isso deve ser subdividido em duas ou mais sentenças PRINT.

START

A instrução START indica o ponto no qual a verdadeira execução dos cálculos do programa deve começar Sentenças precedendo a instrução START não são executadas. Essas sentenças geralmente incluem o nome do programa, a sentença DIMENSION e as equações de

definição a serem chamadas por instruções COMPUTE.

STOP — A instrução STOP indica o ponto no qual a execução do Programa Objeto deve ser terminada. Pode aparecer em diversos pontos num programa e é traduzida em uma instrução que faz parar a operação do computador. A primeira instrução STOP encontrada termina o programa. Pode se ter várias em um só programa porque pode se querer dispôr de várias maneiras alternativas de terminar os cálculos.

END OF TAPE — A instrução END OF TAPE indica que todas as informações relativas a um problema precederam essa sentença na fita. Esta sentença deve aparecer em todas as fitas usadas em um problema e em lugar de ter um número ela é precedida por ZZZZZZ ocupando os seis primeiros espaços (Vêr problemas, Apendice B).

DIMENSION — A instrução DIMENSION especifica o número de valores de uma variável rotulada que serão retidos a qualquer tempo. Se exigido pelo problema, essa instrução deve aparecer como a primeira sentença do programa. Para qualquer programa UNICODE, há <u>sòmente uma</u> sentença DIMENSION. Desde que qualquer combinação de variáveis rotuladas é tratada tendo como módulo o produto das dimensões especificadas, os valores assumidos pelos índices pode exceder às dimensões especificadas. Como um simples exemplo, suponha que a dimensão em X(I) é dada por

DIMENSION X(6) .

<u>Então a variável X tem seis valores retidos</u> durante a execução do programa. Então um valor calculado X(8) ficará no lugar de X(2), X(12) por cima de X(6), etc. Explicações semelhantes aplicam-se à variáveis com 2, 3 ou 4 índices. Êsse fato permite ao programador, em certos programas, reduzir a quantidade de espaço na memória exigido por alguns grupos de dados.

O módulo ou produto das dimensões especificadas de uma variável rotulada deve ser maior que um.

Com duas exceções, todas as variáveis rotuladas de um problema devem aparecer na instrução DIMENSION. As exceções são:

1. As variáveis rotuladas cujos únicos aparecimentos no problema são como argumentos de uma equação funcional que é escrita antes da instrução START.
2. As variáveis rotuladas cujos únicos aparecimentos no problema são como variáveis mudas num sub-programa. (Vêr capítulo 4).

INSTRUÇÃO DE SUB-PROGRAMA

A instrução EXIT é usada sòmente em sub-programas (Vêr capítulo 4).

INSTRUÇÃO DE CORREÇÃO

A instrução DELETE é usada para fazer correções no programa UNICODE (Vêr capítulo 7).

CAPÍTULO 4

PSEUDO-OPERAÇÕES E SUB-PROGRAMAS

A pseudo-operação permite escrever rotinas específicas para um problema particular. Êste recurso do sistema UNICODE fornece ao programador, como um suplemento à Biblioteca existente, qualquer sequência de instruções a ser usada repetidamente mesmo com operandos diferentes.

SÍMBOLO - O símbolo caracterizando uma pseudo-operação, é formado de uma maneira semelhante ao de uma variável(de um a seis símbolos alfabéticos e numéricos, o primeiro dos quais deve ser alfabético).

USO A chamada de uma pseudo-operação pode ser feita sòmente por meio de uma instrução COMPUTE no programa principal. Tôdas as variáveis do programa principal podem ser usadas livremente no sub-programa, mas variáveis "mudas" (Ver abaixo), podem ser usadas como tais sòmente no sub-programa.

Uma constante em ponto flutuante, escrita numa sentença COMPUTE que chama um sub-programa, é caracterizado pelo aparecimento de um ponto decimal. A variável muda correspondente no título do sub-programa não deve ter I, J, K, L, ou M como seu primeiro símbolo. Inversamente uma constante em ponto fixo (inteiro) aparecendo na sentença COMPUTE não deve conter um ponto decimal, e sua variável correspondente no sub-programa deve começar com I, J, K, L, ou M.

TÍTULO DE UM SUB-PROGRAMA - O título de um sub-programa indica que as sentenças do sub-programa vem a seguir e também nomeia os operandos "mudos" a serem usados nas sentenças. O título consiste do símbolo de pseudo-operação seguido pelos operandos "mudos" (Ver abaixo) os quais são encerrados entre parêntesis. Dentro dêsses parêntesis os

operandos são separados por vírgulas.Por exemplo, usemos o símbolo DOT para uma pseudo-operação que forme o produto escalar de dois vetîres.O título poderá ser:

DOT (A(I),B(J))

onde A(I) e B(J) são vetôres "mudos",Nenhum operando "mudo" no título do sub-programa pode ser uma constante.Nenhum operando mudo(inclusive índices)pode ter a mesma designação que outro operando mudo no mesmo título de subprograma,embora diferentes operandos mudos possam ser igualados à mesma variável real na sentença COMPUTE que chama o sub-programa.

VARIÁVEIS MUDAS

Tôdas as variáveis que aparecem no título de um sub-programa são variáveis mudas e podem ser usadas sòmente por sentenças do sub-programa;entretanto, estas sentenças podem também conter outras variáveis do programa.

As sentenças READ, LIST e TYPE,não podem se referir às variáveis mudas num sub-programa.

Variáveis rotuladas mudas não precisam aparecer na instrução DIMENSION,e nenhuma variável muda precisa ser prèviamente definida por uma equação.Um mesmo símbolo pode ser usado para denotar uma variável muda e outra variável do mesmo programa,porém isso não é recomendável porque pode haver alguma confusão;torna-se também impossível dessa maneira,usar a variável muda como uma variável ordinária nas sentenças do sub-programa.

As variáveis mudas são igualadas às variáveis do problema que vêm imediatamente depois do símbolo de pseudo-operação na instrução COMPUTE.

Uma equação funcional muda num sub-programa,é aquela que tem uma função muda sem argumentos à esquerda do sinal de igualdade.Da mesma maneira,podemos ter uma variável rotulada muda à esquerda do sinal de igualdade numa equação num sub-programa.Quando qualquer dêsses dois tipos de equações é executado num sub-programa,a função ou a variável rotulada que es

tava a princípio igualada à função muda ou à variá - vel rotulada muda no título da pseudo-operação, é substituida pelos valores mudos calculados correspon dentes.

Entretanto, quando uma variável simples é igualada a uma variável muda no título de uma pseudo-operação e essa variável muda é posteriormente calculada por meio de uma equação no sub-programa, o valor dessa variável não é substituido pelo novo valor da variá- vel muda.

FUNÇÕES MUDAS - Nem uma sentença COMPUTE que comanda um sub-programa, nem o título de um sub-programa, pode conter argumen tos de símbolos de função. Entretanto, uma sentença COMPUTE dentro do sub-programa, referindo-se a uma função muda, deve ter argumentos, porque ela efetua o cálculo de uma equação precedendo START.

O problema de exemplo mostrado na Figura 5, apêndice B, ilustra o uso de funções mudas num sub-programa.

EXIT - A instrução EXIT retorna o contrôle para a sentença COMPUTE do programa que comandou a pseudo-operação . Deve aparecer pelo menos uma vez em cada sub-programa.

TAMANHO - Um máximo de 20 operandos pode ser usado numa pseu- do-operação. Além disso, a fórmula restritiva se - guinte, a qual é usada internamente pelo UNICODE para determinar se um sub-programa caberá num dado espaço disponível, é apresentada aquí para explicar possíveis dificuldades de espaço dentro de um sub- programa.

$$\left[S + 2F + \sum_{k=1}^{4} N_K\,(2k + 1)\right] \leq 64$$

onde:

S = número de variáveis simples (não-rotula das),

F = número de funções mudas,

K = número de índices nas variáveis rotuladas mudas,

N_K = número de variáveis rotuladas mudas com K índices.

CAPÍTULO 5

DADOS

Tôdas variáveis rotuladas são armazenadas em regiões específicas da memória do tambor magnético durante a computação do programa, ao passo que as variáveis sem índices permanecem todo o tempo na memória de alta velocidade. Cada segmento do programa em execução tem um Prefácio e uma Terminação os quais transferem dados entre o tambor e a memória de alta velocidade durante o intervalo entre segmentos.

ENTRADA

- (Em todo o restante desta seção "dados" significarão valores de variáveis rotuladas, e não consideraremos variáveis simples).

Dados podem ser fornecidos ao programa em execução de duas maneiras:

1. Usando a instrução READ.
2. Pelo recurso do "Leitor Automático de Dados".

O "Leitor Automático de Dados", que opera antes do segmento 1 do Programa Objeto, lê dados para tôdas variáveis rotuladas, exceto para as que estão mencionadas em sentenças de READ ou definidas por equações. Se os dados para uma variável são inadequados, zeros são usados para preencher à área de dados do tambor. Falha no suprimento de dados para uma variável não mencionada em um READ ou não definida por uma equação resultará na impressão de um êrro e numa parada do computador. O número máximo de elementos permissível em uma qualquer tábela de dados a ser introduzido pelo "Leitor Automático de Dados" é 2500. Exceder êste limite tem como consequência a impressão de um êrro e uma parada no computador.

O número máximo de variáveis rotuladas que pode ser lido por UNICODE é determinado pela seguinte rela-

ção

$$V \leq \frac{103 - 3T}{2}$$

onde V é o número de variáveis diferentes e T é o número de Uniservos requeridos para receber os dados. Assim o número máximo de diferentes tabelas de dados que podem entrar no computador é 50 se todos os dados forem colocados no mesmo Uniservo. Uma tabela é o conjunto de dados para uma mesma variável rotulada, cada valor numérico da tabela indo corresponder a um índice da variável (ou a um conjunto de índices quando a variável tem vários índices). A fórmula acima aplica-se ao total de tabelas introduzidas, tanto as que entram pelo " Leitor Automático de Dados" como as que entram comandadas pela sentença READ.

O "Leitor Automático de Dados" transfere dados a seus endereços no tambor, enquanto que a instrução READ coloca os dados na apropriada memória de alta velocidade. Certamente, no fim de cada segmento, a Terminação atualiza as memórias do tambor com as variáveis rotuladas usadas no segmento.

Variáveis definidas por equações podem ser mencionadas em uma sentença de READ.

Ambos os tipos de introdução de dados utilizam fita magnética como meio externo de armazenamento, e é portanto necessário fornecer ao Programa Objeto(não ao programa original em UNICODE) um "ÍNDICE DE DADOS". Isto está explicado mais adiante neste capítulo.

Qualquer Uniservo, exceto o de número 1, que é reservado para a fita de programa, pode receber uma fita de dados, inclusive o número 2, mesmo quando êste também recebe o índice.

FORMATO DA FITA - O primeiro bloco em qualquer fita de dados (a menos que nesta fita também esteja o Índice de dados, caso em que esta informação precede a tôda que é descrita a seguir) é um bloco sentinela:

1ª Palavra	Z Z Z Z Z Z
2ª Palavra	Z Z Z Z Z Z
3ª Palavra	△ D A T A △
4ª Palavra	T A P E △ △
5ª Palavra	sem efeito (podem ser usadas para comentários descritivos).
120ª Palavra	

Uma "palavra" é formada por 6 símbolos. Numa linha de escrita em Unityper, isto é, em um bloquete de fita, tem-se 120 espaços, logo 20 palavras. Em um bloco temos 6 bloquetes, logo 120 palavras. A forma explicada acima aparece do seguinte modo na folha escrita em Unityper.

ZZZZZZZZZZZZ△DATA△TAPE△_____

_____

_____

_____

_____

_____

Seguem-se então tantos blocos quantos necessários contendo os dados, e então o bloco sentinela final:

1ª Palavra	Z Z Z Z Z Z
2ª Palavra	Z Z Z Z Z Z
3ª Palavra	△ E N D △ O
4ª Palavra	F △ D A T A
5ª Palavra	sem efeito
120ª Palavra	

Tem-se que respeitar rìgidamente êste formato.

FORMATO DOS BLOCOS DE DADOS - Cada conjunto de valores para uma dada variável deve aparecer em blocos consecutivos, e dentro de cada conjunto os valores devem estar em ordem de endereço relativo crescente como determinado pela fórmula de manipulação de índices explicada adiante neste capítulo.

Cada conjunto de valores é convenientemente arrumado de modo a ocupar um número inteiro de blocos e é identificável por duas palavras de Z's (isto é, por 12 Z's) no início do primeiro bloco e pela terceira palavra, que deve conter o nome da variável. O primeiro conjunto de valores para cada variável é indicado pela quarta palavra do primeiro bloco dêste conjunto, a qual deve ser " START".

Dentro de cada conjunto, os dados podem começar em qualquer posição depois do fim do primeiro bloquete do primeiro bloco (o qual é reservado exclusivamente à palavras de identificação e às sentinelas, como descrito acima). Todos símbolos na fita, com exceção de símbolos numéricos, o ponto decimal, E, espaço, vírgula, ponto e vírgula e menos, são completamente ignorados. Um número pode consistir de no máximo doze algarismos, ou onze algarismos e um ponto decimal. Estes podem ser precedidos por um sinal menos, e/ou seguidos pela letra E, indicando um expoente decimal, o qual por sua vez deve ser seguido pelo sinal (se negativo) do expoente, e de um ou dois algarismos constituindo êste expoente. Então, o número máximo de símbolos aparecendo juntos é 17. Todos os símbolos constituindo uma quantidade numérica devem ser datilografados continuamente, e o fim do conjunto de símbolos formando uma quantidade numérica é, de fato, assinalado por um Δ(espaço) ou , (vírgula) ou ; (ponto e vírgula). Os últimos dois símbolos têm um significado extra. Êles indicam o "fim da linha", e se presentes, evitam que o restante bloquete seja lido, assim economizando em tempo

de operação e possibilitando que qualquer espécie de comentário (inclusive números) seja escrito em seguida no mesmo bloquete. Ao menos um símbolo deve separar números consecutivos, mas o número de espaços usado não tem qualquer efeito. Os símbolos de preenchimento de espaços devem ser todos 01 e não 00.

O exemplo abaixo ilustra o aspecto de uma fita contendo duas tabelas para a variável A, cada uma ocupando um bloco, e uma tabela para a variável BBB, ocupando dois blocos. Também estão mostrados no exemplo os vários modos diferentes de escrever a quantidade 10^{-3}. Note que uma variável deve estar na posição mais à direita na 3ª palavra do primeiro bloquete de um bloco e ser precedida por espaços, se ela é de menos do que 6 símbolos.

EXEMPLO:

```
Bco. 1, Bqt 1  Z Z Z Z Z Z Z Z Z Z Z Z Δ D A T A Δ T A P E Δ Δ ——completar——→
        Bqt 2
        Bqt 3
        Bqt 4      } Completar
        Bqt 5
        Bqt 6
Bco. 2, Bqt 1  Z Z Z Z Z Z Z Z Z Z Z Z Δ Δ Δ Δ Δ A Δ S T A R T ——completar——→
        Bqt 2  0 . 0 0 1 Δ Δ . 0 0 1 Δ Δ 1 E - 3 Δ Δ 1 0 0 . 0 E - 5 Δ Δ etc
        Bqt 3
        Bqt 4      } Dados Numéricos
        Bqt 5
        Bqt 6
Bco. 3, Bqt 1  Z Z Z Z Z Z Z Z Z Z Z Z Δ Δ Δ Δ Δ A Δ Δ Δ Δ Δ Δ ——completar——→
        Bqt 2
        Bqt 3
        Bqt 4      } Dados Numéricos
        Bqt 5
        Bqt 6
```

Bco. 4, Bqt 1 Z Z Z Z Z Z Z Z Z Z Z Z Δ ΔΔB B B Δ S T A R T completar →
Bqt 2
Bqt 3
Bqt 4 } Dados Numéricos
Bqt 5
Bqt 6

Bco. 5, Bqt 1
Bqt 2
Bqt 3
Bqt 4 } Dados Numéricos
Bqt 5
Bqt 6

Bco. 6, Bqt 1 Z Z Z Z Z Z Z Z Z Z Z Z Δ E N D Δ O F Δ D A T A Δ completar →
Bqt 2
Bqt 3
Bqt 4 } Completar
Bqt 5
Bqt 6

ÍNDICE DE DADOS - O Índice de Dados fornece ao programa em execução, tôda a informação necessária para descrever os dados a serem introduzidos em um problema. Estão aí incluídos os nomes das variáveis, as fitas nas quais estas variáveis são achadas, e nomes identificativos das fitas no caso de as fitas serem designadas por nomes diferentes dos nomes das variáveis mencionadas no programa.

O Índice pode ser fornecido ou em fita de papel perfurado em código Flex ou em fita magnética gravada em código Excesso-Três. Introdução de MJ1 indica que está em fita magnética. Quando o Índice está em fita magnética, êle deve estar escrito nos primeiros blocos da fita montada no Uniservo 2. Uma verificação é feita no momento em que o Índice é lido para assegurar que todos os dados a serem lidos em qualquer estágio do problema, quer durante a Leitura Automática de Dados ou sob o comando de uma sentença

de READ, está mencionado no Índice.

FORMATO DO ÍNDICE DE DADOS

- 1. Em Fita de Papel

O seguinte exemplo ilustra o formato recomendado:

```
INPUT DATA
TAPE 2 A, B = W, CCAT, DDOG = XX3Y, CHECK 4
TAPE 3 F, CHECK 1
TAPE 7 GGG = V, HIJ, CHECK 2
END
```

A primeira linha é o título de identificação, que deve sempre estar presente. Então segue a lista de variáveis aparecendo em cada fita. Precedendo cada lista deve estar o número do Uniservo no qual a fita é montada, e seguindo cada lista está a palavra CHECK, seguida pelo número de variáveis contidas na lista. Isto possibilita uma verificação na leitura do índice. As variáveis têm que estar mencionadas na mesma ordem em que elas aparecem na fita, embora não seja necessário incluir os nomes de variáveis que apareçam na fita mas que não são usadas no programa em execução.

No caso em que dados para uma variável apareçam na fita com nome diferente daquele que será usado no programa em execução, deve-se escrever

```
X = Y
```

onde X é o nome usado no programa e Y é o nome usado na fita de dados.

Não há restrições sôbre o uso do símbolo de espaço, mas é essencial que cada variável seja separada por vírgula daquela que lhe segue.

A linha final, depois que todos os dados foram mencionados, deve conter END seguido por ao menos um

espaço, ponto, vírgula ou retrocesso do carro.

2. Em Fita Magnética

O formato é pràticamente idêntico ao usado em fita de papel; há entretanto uns poucos requisitos adicionais.

1. As duas primeiras palavras da fita devem ser cheias com símbolos Z (isto é, deve-se iniciar a fita com doze Z's), então ao menos um espaço, e então o título INPUT DATA, como em fita de papel.
2. Os símbolos de preenchimento devem ser os 01 e não os 00.
3. Como não há símbolos de retrocesso do carro na Unityper, não é permitido terminar uma palavra na 120ª posição de um bloquete (isto é, linha) e iniciar o próximo na 1ª posição da linha seguinte, porque isto não deixaria espaço entre êles na fita.
4. Depois da linha de END, bloquetes de espaços devem ser acrescentados para completar o bloco.

ORDENAÇÃO DOS DADOS - Consideremos uma tabela "n" dimensional de elementos numéricos representados pela variável rotulada $X(s_1,\ldots, s_n)$ para n = 1,2 3 ou 4. Os índices $s_1,\ldots s_n$ podem assumir apenas valores inteiros não-negativos (isto é, zero e números inteiros positivos). Consideremos esta tabela subdividida em partições (sub-tabelas) de iguais dimensões, tais que uma só partição será requisitada para a memória de alta velocidade de cada vez. As dimensões dessas partições são especificadas na sentença de DIMENSION na forma $X(d_1, \ldots , d_n)$ onde $d_1, \ldots , d_n$ são inteiros positivos indicando o tamanho das "n" dimensões das partições. Se a tabela tôda tem que estar na memória de alta velocidade a um só tempo, haverá apenas uma partição, e as dimensões especi-

ficadas coincidirão com as da tabela completa. As dimensões portanto podem - mas no caso geral isto não ocorre - representar o valor máximo atingido por um índice.

O número de endereços da memória de alta velocidade reservados para valores de X é igual ao produto das dimensões de uma partição e é chamado módulo "M" da tabela. O enderêço H na memória de alta velocidade para um elemento específico de uma tabela é dado pela seguinte fórmula de manipulação de índices:

$$H = B + (m_1 s_1 + \ldots + m_{n-1} s_{n-1} + s_n) \bmod M$$

onde B = enderêço-base da partição na memória de alta velocidade.

$m_1 = (d_2) \ldots (d_n)$, isto é produto das n-1 dimensões à direita.

$m_2 = (d_3) \ldots (d_n)$, isto é, produto das n-2 dimensões à direita.

$m_3 = d_4$

$M = (d_1) \ldots (d_n)$, isto é, módulo da tabela.

Desde que s_1 mais a soma dos produtos dos índices $s_1, \ldots, s_{n-1}$ e dos multiplicadores $m_1, \ldots, m_{n-1}$ é reduzida módulo "M", o mesmo enderêço na memória de alta velocidade pode ser associado com muitos elementos diferentes de uma tabela, dependendo das dimensões da partição. Logo, é possível ir lendo(com a instrução READ) cada partição da tabela e gravando-a na memória de alta velocidade à medida que for sendo necessário. Cada nova gravação faz-se sôbre a anterior, que é então apagada. Referência aos elementos escritos na memória de alta velocidade po-

de ser feita pelos próprios índices associados a êles na tabela original.

Os elementos de uma partição têm que aparecer na fita de dados na ordem de endereços crescentes como dado pela fórmula precedente. Esta ordem não é necessàriamente a "ordem natural" dos elementos da partição, isto é, com cada índice variando por unidades a partir do valor mais baixo na partição e com o índice mais da direita variando primeiro. Além disso, as partições devem aparecer na fita de dados na ordem em que elas devem ser lidas para a memória durante a execução do Programa Objeto.

Como um exemplo, suponhamos que a variável X(I,J) represente a seguinte tabela de duas dimensões, com os índices I e J representando respectivamente a linha e a coluna em que o elemento aparece. Suponhamos a tabela dividida em quatro partições, uma apenas das quais deve estar presente na memória de alta velocidade de cada vez.

e.g.

X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	X_{16}
X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}
X_{31}	X_{32}	X_{33}	X_{34}	X_{35}	X_{36}
X_{41}	X_{42}	X_{43}	X_{44}	X_{45}	X_{46}

Partição 1: X_{11}, X_{12}, X_{13}, X_{21}, X_{22}, X_{23}

Partição 2: X_{14}, X_{15}, X_{16}, X_{24}, X_{25}, X_{26}

Partição 3: X_{31}, X_{32}, X_{33}, X_{41}, X_{42}, X_{43}

Partição 4: X_{34}, X_{35}, X_{36}, X_{44}, X_{45}, X_{46}

Na sentença de DIMENSION associada com a variável X(I,J) esc

Teremos: $n = 2$, $d_1 = 2$, $d_2 = 3$, $m_1 = 3$, $M = (d_1)($

$$H = B + [3s_1 + s_2] \bmod 6$$

Na equação (1) podemos calcular o endereço na memória relativamente ao endereço de base B, para os elemento

Por exemplo:

Para X_{13}: H = B + [3(1) + 3] mod 6 = B + 6 mod 6 = B + 0 = B

Para X_{24}: H = B + [3(2) + 4] mod 6 = B +10 mod 6 = B + 4

Para X_{32}: H = B + [3(3) + 2] mod 6 = B +11 mod 6 = B + 5

Para X_{46}: H = B + [3(4) + 6] mod 6 = B +18 mod 6 = B + 0 = B

Assim, para os elementos das quatro partições obtemos os seguintes endereços:

Endereços Relativos	Elementos da Partição n. 1	Elementos da Partição n. 2	Elementos da Partição n. 3	Elementos da Partição n.4
B	X_{13}	X_{26}	X_{33}	X_{46}
B + 1	X_{21}	X_{14}	X_{41}	X_{34}
B + 2	X_{22}	X_{15}	X_{42}	X_{35}
B + 3	X_{23}	X_{16}	X_{43}	X_{36}
B + 4	X_{11}	X_{24}	X_{31}	X_{44}
B + 5	X_{12}	X_{25}	X_{32}	X_{45}

O compilador escolherá um enderêço absoluto correspondente ao enderêço relativo B. Cada vez que uma sentença READ referindo-se à variável X fôr encontrada no programa original em UNICODE, o compilador fará com que seis elementos da tabela de X sejam lidos em ordem na fita de dados e armazenados nos endereços próprios da memória de alta velocidade, começando no enderêço B. Portanto, é imperativo que os elementos de cada partição apareçam na fita de dados na ordem acima, isto é, na ordem de endereços relativos crescentes como calculado pela fórmula de manipulação de índices. Além disso, as partições devem aparecer na fita na ordem na qual elas serão lidas para a memória durante a execução do Programa Objeto. Esta não

é necessàriamente a ordem física das sentenças READ no programa original.

Se os índices associados com os elementos de uma tabela ou partição de uma tabela começam com zero, então a ordem dêstes elementos na memória de alta velocidade será a sua "ordem natural" na tabela ou partição, isto é, com o último índice variando primeiro. Note que com uma tal tabela nenhuma redistribuição de dados é necessária antes de se preparar a fita de dados.

Exemplo:

X_{00}	X_{01}	X_{02}
X_{10}	X_{11}	X_{12}

Dimension X(2, 3)

Endereço Relativo	Elementos
B	X_{00}
B + 1	X_{01}
B + 2	X_{02}
B + 3	X_{10}
B + 4	X_{11}
B + 5	X_{12}

Tudo o que foi dito sôbre ordenação de dados aplica-se igualmente aos dados que serão lidos pelo "Leitor Automático de Dados" e aos que serão lidos sob comando de uma instrução READ. Entretanto, tabelas que serão "lidas automàticamente" não podem ser divididas em partições; isto é, as dimensões especificadas para uma variável rotulada representando uma tal tabela devem ser as dimensões da tabela inteira.

CAPÍTULO 6

FORMATO

NUMERAÇÃO DAS SENTENÇAS

- Cada sentença num programa UNICODE deve ter um número maior que o número da sentença anterior. Entretanto, a numeração das sentenças não precisa ser consecutiva.

Um total de no máximo seis símbolos pode ser usado para denotar um número de sentença, incluindo um ponto decimal e três algarismos no máximo para a esquerda e dois algarismos no máximo para a direita. O número indicativo da sentença só pode aparecer nas seis primeiras posições do bloquete no qual a sentença começa. As linhas de continuação requeridas para completar uma sentença longa não são numeradas.

Exemplos de números de sentenças permissíveis:

123.45
023.4Δ
Δ2ΔΔΔ
999.99 (o maior possível)
Δ.20ΔΔ

SÍMBOLOS DE ESPAÇO

- O símbolo Δ que aparece no teclado da Unityper é usado para denotar o espaço no programa UNICODE. Em geral, o símbolo deve ser usado tôda vez que um espaço aparecer na escrita inglesa normal, isto é, para separar uma palavra de outra palavra, uma palavra de uma variável ou constante, uma variável de uma constante, etc. O símbolo de espaço Δ pode ser usado a critério do programador sempre que seu uso não violar nenhuma das restrições que se seguem. O uso de mais de um símbolo de espaço como separador também é permitido . As restrições seguintes governando o uso do símbolo Δ em casos específicos devem ser fielmente obedecidas pelo programador.

(1) △ deve ser usado para separar um símbolo de operação (SIN, COS, LOG, etc.) dos seus operandos, a menos que um sinal de mais (+) ou de menos (-), uma abertura de parêntesis ou de símbolo de valor absoluto seja usado.

Exemplos:

X = SIN △Y
X = SIN - Z
X = SIN + Y
X = SIN (Y)
X = SIN |Y|

(2) △ deve ser usado para separar o símbolo de exponenciação POW do seu operando.

Exemplo:

Z = X △ POW △ Y

Nota: O expoente sempre segue o símbolo POW; por exemplo, a forma matemática da equação acima seria $Z = X^Y$.

(3) △ deve ser usado conforme indicado na lista dos exemplos de instruções UNICODE apresentada no Apêndice A (Linguagem UNICODE).

Exemplo:

VARY△X△p(q)r△SENTENCES△K△THRU△M△.
IF△X = Y, JUMP△TO△SENTENCE△K△.

(4) △ deve ser usado junto com um ponto, isto é, na combinação △., terminando cada sentença num programa UNICODE.

Exemplo:

COMPUTE△T△.

(5) Δ não deve ser usado como um componente do símbolo de uma variável ou pseudo-operação. Por exemplo, a combinação MAXΔHT não pode ser usada como símbolo de uma variável.

PROGRAMA UNICODE - Cada linha da cópia, obtida na Unityper, de um programa UNICODE conterá 120 posições, o que representa um bloquete na fita magnética. As 6 primeiras posições são reservadas para o número da sentença e as posições 7-120 são destinadas às instruções do programa. Tantos espaços quantos desejados podem aparecer entre o número de sentença e o comêço da instrução; entretanto, linhas adicionais de uma sentença devem começar na posição 7.

Um exemplo dêsse formato de entrada é mostrado na Fig. 1. Embora as teclas FILL e SKIP FILL não imprimam nenhum símbolo na cópia, a Fig. 1 mostra os Δ para salientar que o "espaço" deve ser usado como símbolo de preenchimento.

LISTAGEM DO PROGRAMA - O sistema UNICODE fornece uma listagem do programa (optativa) em fita magnética, própria para impressão na Impressora de Alta Velocidade. Essa listagem inclue o título do programa, o conjunto de constantes, as variáveis não rotuladas, as variáveis rotuladas e cada um dos segmentos do Programa Objeto. A forma de listagem correspondente ao programa original da Fig. 1 é indicada na Fig. 2.

Os cinco bloquetes reservados para o título do programa são impressos exatamente como aparecem na cópia obtida na Unityper; portanto, sòmente os símbolos da Unityper que podem ser reproduzidos na Impressora de Alta Velocidade devem ser usados nos bloquetes de título. No item sôbre variáveis não rotuladas são dados o nome e a localização na memória de alta velocidade de cada uma dessas variáveis, o enderêço na memória de alta velocidade e a repre-

sentação octal de cada constante podem ser obtidas no item referente às constantes. Todas as variáveis rotuladas são listadas, sendo dados seus nomes, e sua localização no tambor.

Na listagem de cada segmento são dados os nomes das variáveis rotuladas que são nele usadas, juntamente com os endereços na memória de alta velocidade reservados para elas. Também estão incluidos uma listagem do Prefácio (instruções geradas na compilação e que transferem do tambor magnético para a memória de alta velocidade os grupos de variáveis rotuladas necessárias para execução do segmento), e da Terminação (instruções que transferem de volta para o tambor magnético os grupos de variáveis rotuladas devidamente atualizadas depois da execução do segmento).

Finalmente, as instruções do Programa Objeto produzidas na compilação, inclusive as rotinas utilizadas, são listadas para cada segmento, juntamente com o número da sentença correspondente do programa UNICODE ou o título da rotina da Biblioteca. As instruções são listadas na ordem crescente dos endereços das memórias. Portanto, a ordem na qual os números das sentenças aparecem na listagem não é necessàriamente sua ordem no programa UNICODE. As sentenças que formam os comandos de contrôle para cada segmento aparecerão primeiro e serão seguidas pelas equações individuais, sub-rotinas da Biblioteca e sub-programas em uma ordem qualquer.

As instruções do Programa Objeto e as constantes são listadas da esquerda para a direita, sob a forma de números octais em quatro colunas ao longo da página, dispostas de tal maneira que instruções ou constantes cujos endereços de memória terminem em zero ou quatro, aparecem na coluna mais à esquerda. Os endereços são enumerados numa coluna extra à esquerda das quatro colunas. Cada endereço terminando em

zero é escrito, bem como o primeiro endereço de cada sentença ou rotina de Biblioteca, o primeiro endereço aparecendo entre parêntesis se não terminar em zero. A listegem é paginada, com os números das páginas indo até 999; daí em diante a palavra "continued" é usada em lugar do número da página. Caso a listagem ocupe mais do que uma fita magnética de 1500 pés de comprimento, a listagem é prosseguida em uma fita adicional.

A listagem do programa é destinada a ser impressa em uma impressora de alta velocidade usando espaço simples, papel de 14 7/8" de largura e 11" de comprimento em blocos especiais fornecidos com o sistema UNICODE. O papel deve ser inserido na impressora de modo a que a primeira linha da lista seja impressa na quarta linha da página a contar de cima. Os únicos símbolos especiais usados são FAST FEED I e PRINTER STOP. As palavras END OF LISTING seguindo PRINTER STOP indica o fim da listagem.

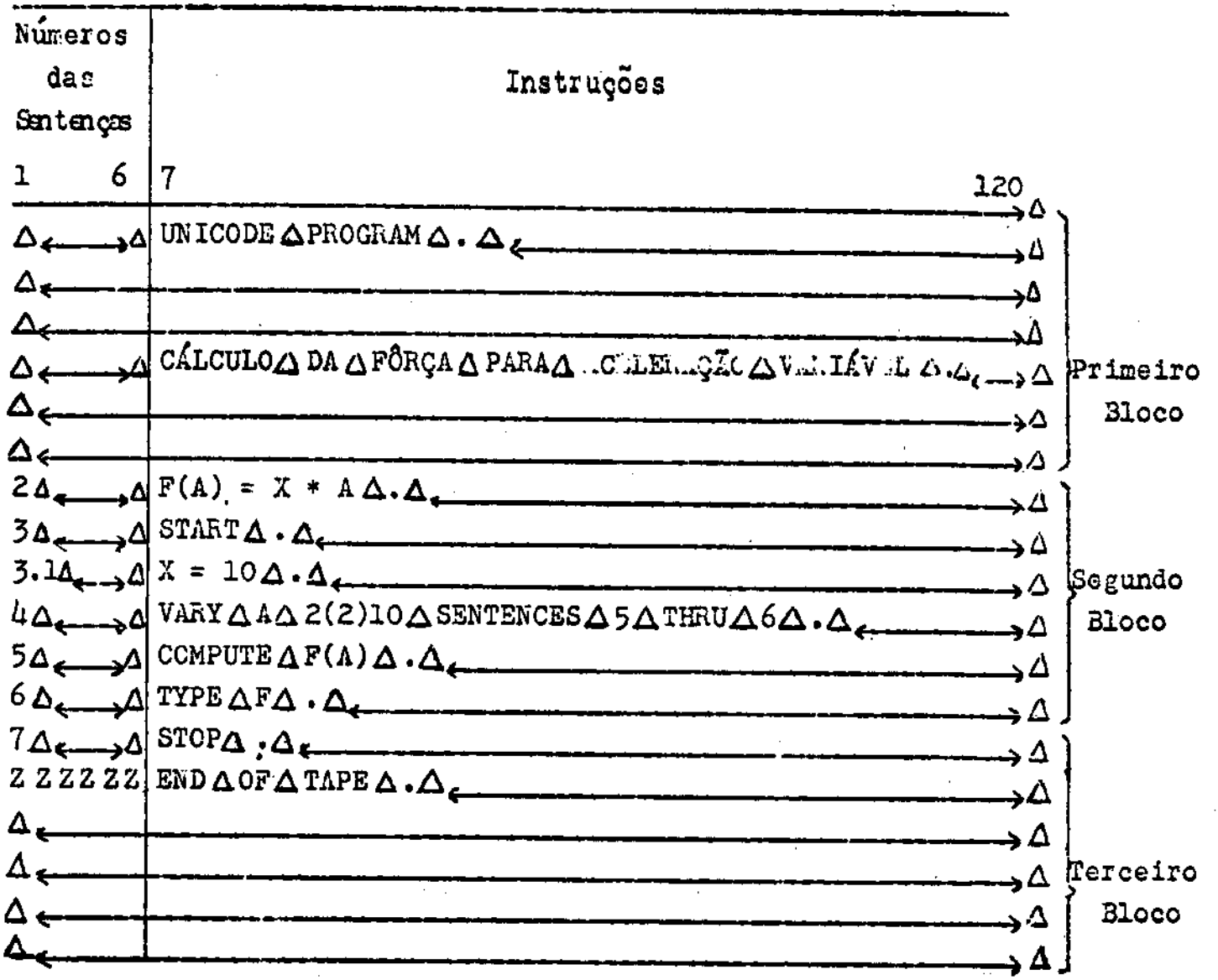


Figura 1 - Programa Original Escrito em Unityper

PROGRAM LISTING

CÁLCULO DA FÔRÇA PARA ACELERAÇÃO VARIÁVEL

NONSUBSCRIPTED VARIABLES
IMPRESSÃO DE CADA VARIÁVEL E RESPECTIVO ENDERÊÇO

CONSTANT POOL
IMPRESSÃO DAS CONSTANTES

SEGMENT 1

SENTENCE NUMBER 3.
CODIFICAÇÃO OCTAL (Leitura da esquerda para a direita em quatro colunas, com o enderêço inicial e cada oitavo enderêço numa coluna extra à esquerda).

SENTENCE NUMBER 3.1
CODIFICAÇÃO OCTAL

SENTENCE NUMBER 4.
CODIFICAÇÃO OCTAL

SENTENCE NUMBER 5.
CODIFICAÇÃO OCTAL

SENTENCE NUMBER 6.
CODIFICAÇÃO OCTAL

SENTENCE NUMBER 7.
CODIFICAÇÃO OCTAL

SENTENCE 2.
CODIFICAÇÃO OCTAL

LIBRARY ROUTINE FLEXPT
CODIFICAÇÃO OCTAL

LIBRARY ROUTINE FLTCVT
CODIFICAÇÃO OCTAL

END OF LISTING

Figura 2. -- Listagem do Programa

CAPÍTULO 7

CORREÇÕES E TRANSBORDAMENTO DE FITA.

Correções de um programa UNICODE são feitas perparando-se uma fita com a informação corrigida. A fita é então montada em um Uniservo separado e encaixada no programa original para produzir a nova fita modificada. Uma fita de correção deve ter ao menos uma polegada de espaçamento entre blocos e uma polegada de espaçamento entre bloquetes. Esta condição é satisfeita automàticamente se a fita fôr preparada na Unityper ou por meio da Rotina "Flex-para-Excesso-Três" descrita no Cap. 12.

Os números das sentenças numa fita de programa devem estar em ordem crescente. Se uma sentença "fora-de-sequência" aparecer numa fita de programa enquanto a rotina de correção estiver operando, a sentença será excluida e a seguinte impressão ocorrerá:

SENTENCE_____OUT OF SEQUENCE

A ordem das sentenças no programa não será conferida quando a rotina de correção não fôr usada.

As correções não precisam estar na ordem numérica das sentenças a que elas se referem. Elas são devidamente ordenadas pelo sistema UNICODE antes de serem encaixadas no programa original.

O número máximo de correções permitidas é 300. Se se ultrapassar êste máximo a máquina o acusará imprimindo o êrro, e em seguida usará só as 300 primeiras correções para produzir uma fita corrigida.

O primeiro bloquete do primeiro bloco de uma fita de correção tem que conter o título UNICODE CORRECTIONS. Êste título, se precedido apenas por espaços, poderá iniciar-se em qualquer lugar nas primeiras 100 posições do bloquete. O restante dêsse primeiro bloco tem que ser preenchido por espaços. As correções são escritas a partir do primeiro bloquete do segundo bloco. No fim da lista de correções usa-se uma linha para escrever a instrução END OF TAPE, precedida por ZZZZZZ na posição correspondente ao número da sentença. Estes 6 Z's indicam ao UNICODE que todas as correções foram já dadas. Omissão desta linha em uma fita de correção ou de programa ocasionaria uma procura de informação adicional na fita ,

adiante dos dados registrados referentes ao problema em consideração. Como adiante daquilo que se gravou na fita pode estar qualquer tipo de marca(remanescentes de gravações anteriores), isto pode causar uma suspensão na operação do computador, ou na indicação de uma falta ou numa variedade de impressões de êrro.

Qualquer correção de programa no sistema UNICODE é de uma das seguintes três categorias:

1. Mudança de sentenças existentes.
2. Inserção de sentenças adicionais.
3. Cancelamento de sentenças existentes.

Para mudar uma sentença que está no programa é necessário apenas que na fita de correção seja escrito o número da sentença, seguido da versão correta da instrução.

Uma sentença na fita de correção com um número inexistente no programa original, é automàticamente inserida no programa na ordem indicada pelo seu número.

O número da sentença na fita de correção deve estar contido no grupo dos 6 primeiros espaços da linha. A nova instrução começa na posição 7.

Se houver mais do que uma mudança ou inserção com o mesmo número na fita de correção, o último que aparecer é que será usado, os anteriores sendo ignorados.

Se o número de uma sentença na fita de correção fôr ilegal (ver Capítulo 6), a correção será ignorada.

DELETE - Para eliminar linhas do programa original, uma instrução especial é necessária. Esta instrução aparece na fita de correção, sem um número de sentença.

Exemplos:

ΔΔΔΔDELETEΔSENTENCESΔkΔTHRUΔmΔ
ΔΔΔΔΔΔDELETEΔSENTENCEΔkΔ

No 1º exemplo tôdas as sentenças entre e inclusive as

sentenças numeradas k e m seriam eliminadas do programa original. No segundo exemplo só a sentença com número k seria eliminada. Tôdas as posições do 1 a 6 de uma sentença DELETE têm que ser espaços. Mais espaço antes da palavra DELETE é permitido mas os espaços e a escrita dentro da sentença devem ser exatamente como indicado nos exemplos acima. Deixar de usar o plural no primeiro exemplo resultaria em apenas a sentença k ser eliminada.

Uma sentença DELETE cancela na fita de correção quaisquer outras correções referentes à sentença (ou sentenças) mencionadas, juntamente com a eliminação da própria sentença no programa original.

Se um número ilegal de sentença ocorrer dentro de uma ordem de DELETE, a ordem não será executada.

TRANSBORDAMENTO DE FITA - Se não fôr possível escrever todo o programa em uma única fita de Unityper, uma segunda fita poderá ser usada. A fita adicional poderá ser tratada como se fosse uma fita de correção desde que os números das instruções nela ocorrendo sejam maiores do que os números existentes na primeira fita. As duas fitas são então combinadas no computador para formar uma fita maior que conterá então o programa completo.

CAPÍTULO 8

PREPARO DE FITAS

Três fitas podem ser necessárias para resolver um problema pelo sistema UNICODE: a fita de programa, (contendo um conjunto de instruções que o sistema UNICODE interpreta como operações a executar), a fita de correções ao programa e a fita de dados (valores numéricos de certas variáveis ou parâmetros que ocorrem no programa). Essas três fitas são usadas na preparação de um chamado Programa Objeto. Esta preparação do Programa Objeto é feita no próprio computador, pela pessoa que o opera.

Muito da informação necessária para o preparo de fitas já foi dada nos capítulos I,V,VI e VII. Fitas de programae de correção usam números de sentença ocupando os primeiros seis lugares, e as instruções começam na sétima posição. Se uma instrução ocupar mais do que uma linha pula-se os seis primeiros espaços ao se entrar em cada nova linha (isto é, na fita de programa os seis primeiros espaços de cada bloquete só podem ser usados para números de sentenças). Por outro lado, as linhas da fita de dados não são numeradas, e quando se passa de uma linha para outra não se pula os seis primeiros espaços.

Se uma palavra fôr interrompida no meio pelo fim de uma linha da Unityper, não deverá ser usado traço de união nem se deve tentar observar regras de separação de sílabas. As letras de uma palavra que não puderem ser postas em uma linha ao se escrever até o 120º espaço devem ser continuadas a partir do sétimo espaço da linha seguinte se a fita fôr de programa ou de correção; se a fita fôr de dados a continuação se fará a partir da primeira posição.

Se uma palavra ou símbolo terminar precisamente na 120ª posição de uma linha de Unityper, um espaço extra deverá ser dado no início da linha seguinte para conservar o sentido. O fim de uma linha de Unityper de modo nenhum age como separador ou símbolo de espaço no sistema UNICODE. Os seis espaços deixados livres no início de uma linha de continuação de uma sentença longa são ignorados completamente na interpretação de uma sentença.

Tôdas as linhas têm sempre que ser ocupadas até a 120ª posição. Se uma sentença fôr completada antes desta posição (como é geralmen

te o caso), ela deverá ser preenchida até o fim pressionando-se a tecla de enchimento de espaço.

Adicionando-se cinco linhas de espaço no fim de uma fita fica se seguro de que a última linha contendo informação faz parte de um bloco completo. Os 15 símbolos na primeira coluna da tabela abaixo, que não se encontram no teclado regular da Unityper, podem ser obtidos batendo-se nas teclas indicadas na segunda coluna. O UNICODE foi adaptado para reconhecer a representação no código Excesso-Três dêsses símbolos na segunda coluna como equivalentes aos símbolos correspondentes na 1a coluna.

Expoente zero	\0/	Sigma	Σ
Expoente um	\1/	Beta	β
Expoente dois	\2/	t minúsculo	t
Expoente três	\3/	r minúsculo	r
Expoente quatro	\4/	Aspas	"
Expoente cinco	\5/	Número	#
Expoente seis	\6/	Dolar	$
Expoente sete	\7/	Porcentagem	%
Expoente oito	\8/	Cento	¢
Expoente nove	\9/	Interrogação	?
Ponto no expoente	\./	Dois pontos	:
Menos no Expoente	\-/	i minúsculo	i
Divisão no Expoente	\//	Apóstrofe	'
Maior do que	>	"e" comercial	&
Menor do que	<	"a" comercial	@

Quando uma fita é completada, ela deve ser passada na Impressora de Alta Velocidade na posição "computer-digit straight" e então cuidadosamente conferida com o original. Levando-se em consideração as variantes em "computer-digit" dêsses símbolos, esta verificação pode mostrar mui

tos êrros que podem ser corrigidos antes da fita ir para o computador. Uma fita de correção de programa pode então ser preparada como explicado no Capítulo 7.

Se não se dispõe de uma Unityper, fitas de programa, de correção ou de dados poderão ser produzidas em uma máquina Flexwriter em combinação com o computador como explicado no Capítulo 12, em que se trata da rotina de conversão do código de Flexwriter para o código Excesso-Três.

CAPÍTULO 9

OPERAÇÃO DO UNICODE

COMPILAÇÃO. O primeiro passo para a compilação em UNICODE é carregar o tambor, na posição "8 interlace", com a fita bioctal das Rotinas de Serviço do UNICODE. O grande número de saltos de igualdade escritos no tambor torna pouco prático o uso da posição "4 interlace". Há duas versões dessa fita bioctal - uma para a 1103A e uma para a 1105.

Anàlogamente há versões para 1103A e 1105 da Fita Mestra de UNICODE, uma fita magnética que contém o UNICODE e deve ser montada no Uniservo 1.

No caso em que rotinas de biblioteca sejam referidas nas equações do problema em UNICODE, uma Fita-Biblioteca deverá ser montada no Uniservo 2. Se essa fita não se encontrar aí a máquina o acusará por escrito, dando além disso uma opção de contornar a exigência.

O Uniservo 3 é carregado com o programa em UNICODE do problema em consideração.

No Uniservo 4 será colocada uma fita de correções, se fôr o caso.

No Uniservo 5 uma fita em branco deverá ser montada. Nessa fita será escrito o problema corrigido - obtido através da combinação das fitas 3 e 4, ou, se não há correções a fazer, será escrita uma cópia da fita 3, sem alterações.

No caso em que sete Uniservos sejam usados para a compilação, MJ1 deverá ser introduzido e fitas em branco deverão ser montadas nos Uniservos 6 e 7. Se se opera com TCU2 na 1105, ou na 1103 em qualquer caso, MJ2 deverá ser introduzido. Se uma combinação das fitas 3 e 4 é desejada, MJ3 deverá ser introduzido.

Todos os Uniservos, de 1 a 5 ou de 1 a 7 conforme a seleção, deverão ser ligados.

Assim que os passos acima tenham sido dados, se introduzirá o endereço inicial em 77000 e, em seguida, se pressionará o botão de partida (START).

Se a "Flexowriter" associada ao computador não imprimir êrro

algum, o problema será prosseguido até o fim da geração, quando então o computador acusará por escrito: END OF GENERATION. Para interromper a compilação introduzir A NOT=0.START. Aqui, START apenas, sem alteração de A, fará prosseguir a compilação. Se a opção de interrupção fôr tomada, tôdas as fitas serão novamente enroladas ao ponto de partida. Para recomeçar, mais tarde, se deverá verificar se todos os Uniservos estão ligados, com as fitas montadas e posicionadas como estavam no fim da geração, os MJ apropriados introduzidos e as Rotinas de Serviço do UNICODE carregadas como anteriormente. O enderêço inicial será então introduzido em 77004 e o botão de partida será novamente pressionado.

Se apenas cinco Uniservos forem utilizados, logo após a operação de combinação das fitas 3 e 4 o computador requisitará por escrito : PUT 1500 FT. TAPES ON S. 3 AND 4. Assim que essas fitas tenham sido montadas nesses Uniservos, o botão "start" deverá ser pressionado para dar início à fase seguinte - tradução.

Se operando com cinco Uniservos, o computador acusará por escrito: COMPUTER CODING PRODUCED ON TAPE 3. Se com sete Uniservos o número da fita será 6. Em seguida o computador perguntará por escrito se haverá alguma opção de listagem: IF PROGRAM LISTING IS NOT DESIRED, SET A NOT=0.START.

Aquí, START apenas, sem alteração de A, fará com que uma cópia do programa produzido, dividido segundo segmentos e números de sentença, seja dada na fita 4 ou na fita 7, conforme 5 ou 7 Uniservos estejam sendo usados.

No fim da compilação o computador acusará por escrito: COMPILATION COMPLETED.

A fita contendo a Listagem do Programa poderá ser traduzida na impressora de alta velocidade e mais tarde usada na depuração do problema, se necessário. Ver capítulo 6 para detalhes.

PROGRAMA-OBJETO. A Fita Mestra de UNICODE não é usada durante a ~~confecção~~ execução do Programa-Objeto. A fita bioctal das Rotinas de Serviço do UNICODE é usada, e deve ser carregada no tambor.

Se na 1105, TCU2 deve ser usado. MJ2 deve ser sempre introduzido. Isto significa que TCU2, 1105, ou 1103A está sendo usada.

No Uniservo 1 deverá ser colocada a fita contendo o código

produzido pelo computador na compilação.

Se o Índice de Dados estiver em fita magnética, a fita que o contém deverá ser colocada no Uniservo 2 e MJ1 introduzido.

Se existir um Índice de Dados, a não introdução de MJ1 indicará que êle está em fita de papel, codificada pela Flexowriter. Nesse caso a fita deverá ser colocada na unidade de leitura Ferranti, antes de iniciar a operação. Se não existir um Índice de Dados a introdução de MJ1 será irrelevante,

Fitas em branco deverão ser montadas nos Uniservos referidos por meio de sentenças de LIST no programa original em UNICODE.

As fitas contendo dados deverão ser carregadas nos Uniservos apropriados, de acôrdo com as indicações do Índice de Dados.

No caso em que se queira uma parada precedendo imediatamente a operação de qualquer segmento, MS1 deverá ser introduzido. Se a parada fôr desejada imediatamente após a realização de qualquer segmento , MS2 deverá ser introduzido.

O comando de partida será feito com endereço inicial em 77300.

As paradas comandadas por MS1 ocorrem logo depois que variáveis rotuladas são transportadas do tambor para a matriz. Variáveis ou constantes poderão ser alteradas nessa fase por intermédio de comando de fita perfurada, correções de console ou outras medidas apropriadas. Para retornar à execução do problema a partida deverá ser dada com endereço inicial em 57.

A primeira parada comandada por MS2 ocorre logo depois que o Leitor Automático de Dados transfere as tabelas de dados para o tambor. A massa de constantes está na memória de alta velocidade nesta fase mas o primeiro segmento não foi ainda inserido. A segunda bem como as paradas sob o comando de MS2 ocorrem após operação do segmento, antes que o tambor seja atualizado com os últimos valores das variáveis rotuladas, obtidos na matriz. Assim no tambor estão as variáveis rotuladas, antes da operação do segmento, e na memória de alta velocidade os valores das variáveis após operação. Depois que quaisquer alterações de valor tenham sido executadas o programa será retomado com endereço de partida em 17.

Se o Índice de Dados estiver correto a máquina o confirmará por escrito, mencionando as fitas de entrada por êle referidas. Por oca-

são da conclusão do problema o computador fará por escrito e sucessiva - mente as seguintes declarações: END OF RUN. TO REWIND I/O TAPES SET SERVO NO IN A HIT START. Isto significa que o número de um Uniservo de entrada ou saída deve ser introduzido em A, em escrita octal, e o botão de partida deve ser pressionado para que aquele Uniservo seja re-enrolado. Em seguida a esta operação o computador relembrará por escrito que o mesmo deverá ser feito para qualquer Uniservo adicional que requeira ser re-enrolado.

CAPÍTULO 10

ALARMAS E ADVERTÊNCIAS

A maioria das advertências dadas por escrito pelo computador operando com UNICODE são detalhadas e claras. Não foi feita neste capítulo qualquer tentativa de documentar tôdas as possíveis advertências. Explicações são dadas apenas para aquelas poucas que ocorrem sob forma mais abreviada.

Quando 25 êrros ~~tenham~~ tiverem sido encontrados pelo computador na fase de tradução, as fitas são re-enroladas e a tradução é encerrada com uma advertência por escrito daquele fato. A análise de qualquer sentença não é prosseguida depois que cinco êrros são nela encontrados.

Frequentemente acontecerá que a ocorrência de um êrro ou êrros produzirá muitos êrros adicionais. Por exemplo, omissão ou má colocação da sentença DIMENSION causará uma acusação por escrito de êrro cada vez que uma variável rotulada fôr encontrada. Ausência do número apropriado de sub-índices seguindo uma variável rotulada numa sentença IF dará origem a uma série de advertências.

Má colocação de elementos dentro de uma sentença é uma causa frequente de advertências. Por exemplo, um símbolo ocorrendo em lugar errado numa sentença IF será apontado por escrito, seguido das palavras: Symbol Rejected. Nesse caso específico, se o símbolo puder ser omitido sem sacrifício do sentido da sentença, o código produzido estará ainda correto.

A omissão de um símbolo essencial num estágio específico da análise de uma sentença dará origem também a uma advertência. Um exemplo disso ocorre na sentença IF, caso em que a advertência dada é: Sentence_____ Incorrectly Written _____.

O último elemento dessa advertência - o símbolo sob análise naquele estágio - indica até que extensão o exame foi efetuado, antes que a sentença fôsse julgada inadequada.

Na fase de geração um único êrro acarretará o re-enrolamento de tôdas as fitas e a parada do computador, acompanhada de uma advertência do êrro. Quando um número de sentença é referido numa instrução e ês

se número não ocorre entre os números de sentença do programa, a advertência e parada correspondente não ocorrerão senão na geração.

Outras situações que podem conduzir a uma parada são comandos de saltos ilegais.

A seguinte advertência poderá ocorrer na geração ou qualquer fase posterior: ALARM a. COMPILATION INCONSISTENCY 'POSSIBLE COMPUTER ERROR'. RECOMPILE.

a no exemplo acima vale por qualquer número de 1 a 11. Detalhes sôbre o UNICODE não apresentados neste manual são envolvidos na determinação do número particular usado nessa advertência. Essa advertência, quando não fôr causada por mal funcionamento do computador, poderá resultar apenas de um êrro numa rotina de Biblioteca ou na Fita Mestra de UNICODE.

O número de segmentos permitidos para qualquer problema é 63. Quando êsse limite é excedido, uma advertência é dada, o computador pára e as fitas são re-enroladas. A solução - quando existe - dessa situação é quebrar o problema em duas ou mais seções e compilar cada uma isoladamente.

Se uma sentença fôr longa demais e contiver muitas referências a seguinte advertência poderá ocorrer:

ONE STATEMENT WITH ALL REFERENCES
IS TOO LARGE FOR SEGMENT

O remédio é quebrar a sentença em duas ou mais sentenças.

Durante a execução do Programa-Objeto, o uso da Biblioteca de Rotinas Permanentes (Ver Cap. 11) poderá causar as advertências explicadas na seguinte tabela:

ADVERTÊNCIA	EXPLICAÇÃO
RUN ERROR 1	Cálculo de X^Y está sendo tentado para X=0 e Y=0.
RUN ERROR 2	Cálculo de X^Y está sendo tentado para X=0 e Y<0.
RUN ERROR 3	Cálculo de X^Y está sendo tentado para X<0 e Y não inteiro.
RUN ERROR 4	Cálculo de $\log_e X$ está sendo tentado para $X \leq 0$
RUN ERROR 5	X ou e^X estão fora da capacidade da máquina
RUN ERROR 6	Cálculo da raiz quadrada de X está sendo tentado para X<0.

Se o prosseguimento da execução do Programa Objeto, após qualquer das advertências acima fôr desejado para fins de depuração, um valor razoável em ponto flutuante poderá ser introduzido em Q e então pressionado o botão de partida.

Durante a execução do Programa Objeto, tôda vez que uma variável tiver que ser lida, será procurada a fita de dados que a contém. Falha em localizá-la resultará na seguinte advertência:

DATA INDEX ERROR: SEE Q

Em Q será encontrada a representação em código Excesso-Três da variável que está sendo procurada à esquerda, seguido por 77's à direita. Êsse tipo de êrro poderá ser causado pela ausência do nome da variável na última posição à direita da terceira palavra do primeiro bloquete de um bloco (veja Formato de Bloco de Dados no Capítulo 5).

CAPÍTULO 11

BIBLIOTECA DE ROTINAS

Biblioteca Permanente de Rotinas

Há dois tipos de biblioteca de rotinas usados em UNICODE - aquela que inclui as rotinas contidas na Fita Mestra do UNICODE e a que se compõe das rotinas introduzidas na Fita Biblioteca, montada no Uniservo 2. As rotinas na Fita Biblioteca poderão ser construidas e incluidas pelos utilisadores do UNICODE. As rotinas da Fita Mestra do UNICODE, conhecidas como Biblioteca Permanente de Rotinas, foram construidas no mesmo formato requerido para as rotinas da Fita Biblioteca, a qual será descrita adiante neste capítulo.

Três dessas rotinas da Biblioteca Permanente podem ser chamadas a partir de equações num programa em UNICODE. Essas rotinas são: a que dá o logaritmo natural, a da exponencial e a da raiz quadrada.LN seguido de uma variável ou constante numa equação chama a primeira, EXP seguido de uma variável ou constante retira da segunda rotina mencionada o valor de e^x, onde e = 2.7182818 e x é a variável ou constante. Exemplos de chamada da rotina de raiz quadrada são: Y=SQRTΔXΔ. Y=XΔPOWΔ(1/2)Δ. $Y=X^{1/2}\Delta$, $Y=X^{0.5}\Delta$.

A Biblioteca Permanente de Rotinas será encontrada frequentemente nas Listagens de Programas e será referida pelos nomes apresentados na descrição que delas é dada abaixo.

Nenhum dêsses nomes poderá ser dado a uma outra rotina de Biblioteca e, exceto para referência específica a uma das três rotinas acima mencionadas, nenhum deles poderá aparecer num programa em UNICODE, denotando uma variável ou uma função.

As advertências associadas com essas rotinas, originadas na execução do Programa Objeto são explicadas no Capítulo 10. Segue-se uma descrição breve de suas funções e operações:

1	FLEXPT	A máquina de escrever associada ao computador é operada com entrada em código flex durante a execução do Programa Objeto.
2	GENPOW	Rotina de exponencial generalizada, incluida no Programa

Objeto como o resultado do aparecimento de um têrmo tal como X△POW△Y no PROGRAMA ORIGINAL, sendo Y desconhecido durante a compilação. Esta rotina chama as de números 3 e 6 abaixo, conforme a necessidade.

3 VAREXP — Termos tais como X△POW△A ou X^A no Programa Original, onde A é conhecido durante a compilação mas ≠ 1/2 e não é um inteiro < 64, causam a chamada desta rotina a qual, por sua vez, chama automàticamente as de número 4 e 5.

4 LN — Rotina de logaritmo natural, que pode ser chamada diretamente do Programa Original, ou indiretamente, através de VAREXP, por exemplo. Exemplo de uso X=LN△Y△.

5 EXP — Referida internamente por VAREXP mas também diretamente do Programa Original. A entrada de EXP Y dá como saída e^Y, onde e=2.7182818.

6 SQRT — Rotina de raiz quadrada que pode ser chamada interna ou externamente.

7 FLTCVT — Converte ponto flutuante em decimal e dá o comando de impressão na máquina de escrever associada; resulta de uma chamada interna desta rotina por uma sentença de TYPE.

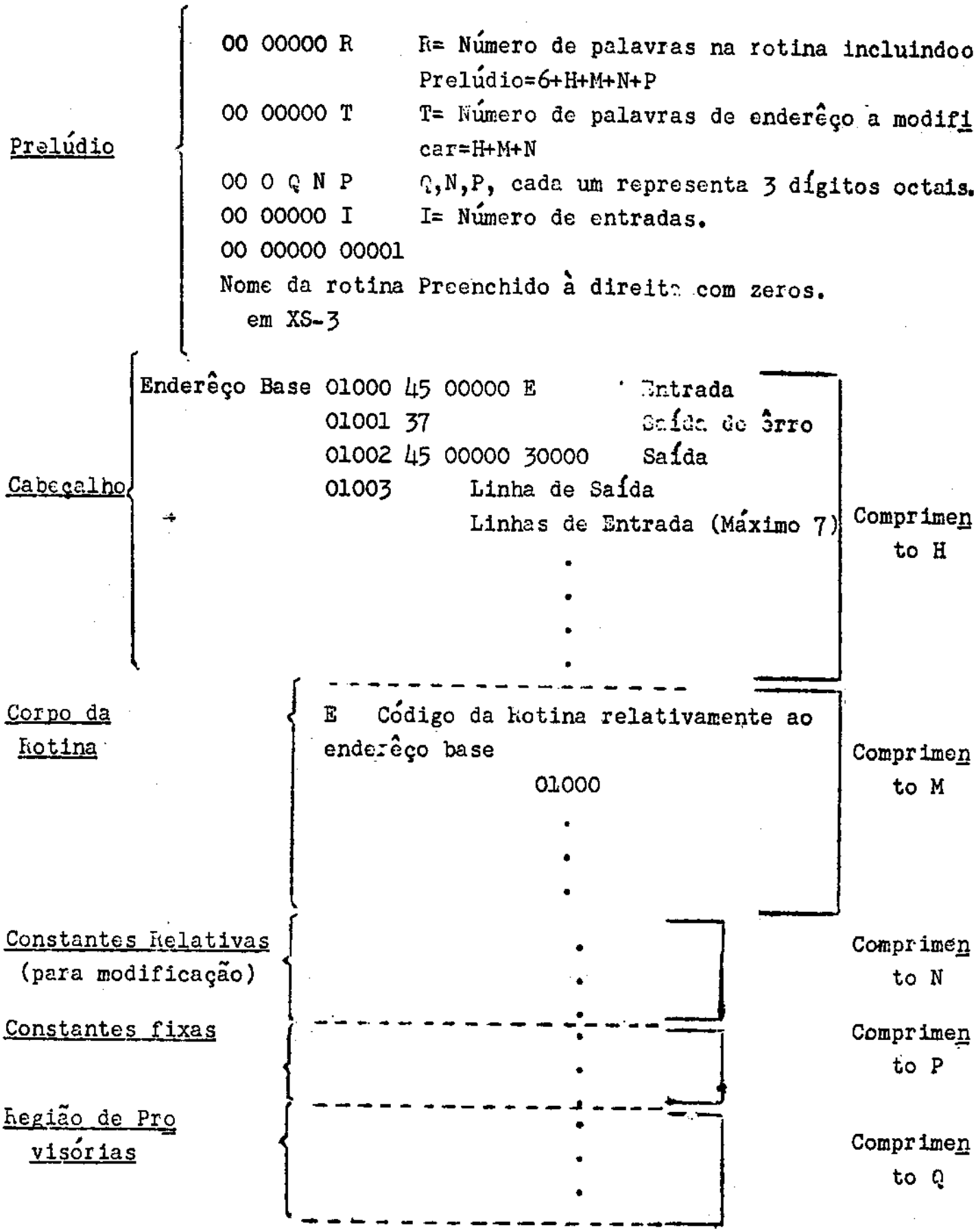
8 LISTRN — Rotina de saída que escreve em fita magnética listas apropriadas para a Impressora de Alta Velocidade.

9 READRN — Rotina de entrada que lê valores de fita magnética e os converte à notação de ponto flutuante.

10 INTCVT — Converte ponto fixo em decimal e dá o comando de impressão na máquina de escrever associada ao computador; resulta de uma chamada interna desta sentença por uma sentença TYPE.

Formato das Rotinas de Biblioteca.

Tôdas as rotinas a serem adicionadas à Fita de Biblioteca deverão ser escritas no seguinte formato:



A ordem, acima descrita, das várias seções numa rotina da Biblioteca, é essencial para a correta manipulação pelo UNICODE.

Para facilitar a escrita das rotinas da Biblioteca, um conjunto de "senhas" poderá ser usado para fazer referência relativa a palavras específicas dentro de qualquer das últimas três seções de uma tal rotina. As senhas básicas são:

10 - - - Constantes Relativas
20 - - - Constantes Fixas
70 - - - Reserva Temporária

onde os três últimos dígitos octais das senhas indicam a posição relativa dentro da região designada. Por exemplo, a senha 10003 se refere à quarta constante relativa, e 70000 se refere ao primeiro enderêço temporário. Essas senhas facilitam a escrita das instruções do computador que devem chamar essas áreas, sem predeterminar os enderêços dos ítens relativamente ao enderêço base (01000) da rotina.

O nome em código excesso-três da rotina poderá consistir de qualquer combinação de símbolos alfabéticos ou numéricos que não tenham sido já usados na Biblioteca Permanente de Rotinas do UNICODE ou como palavras-chave da linguagem do UNICODE.

Recìprocamente, nenhuma combinação de símbolos alfabéticos e numéricos usados para denotar uma rotina da Biblioteca poderá ser usada , num programa em UNICODE, como variável.

Regras adicionais para a escrita em UNICODE de uma rotina da Biblioteca:

1. O valor de saída deve passar da rotina ao registro Q.
2. Valores de entrada e saída são supostos estarem em notação de ponto flutuante. Assim rotinas da Biblioteca não podem ser usadas por equações de ponto fixo. Elas podem ser chamadas apenas por equações de ponto flutuante.
3. Referências internas não podem ultrapassar o enderêço relativo 07777, exceto quando usando as senhas 10---, 20---, 70---, etc.
4. Uma rotina da Biblioteca em UNICODE pode ter até sete linhas de entrada mas sòmente uma de saída. Assim a contribuição da contagem das linhas de saída no Prelúdio é sempre 1.

O Bibliotecário do UNICODE.

O Bibliotecário do UNICODE é uma rotina de serviço, independente da Fita Biblioteca, que supre os meios de fazer inserções e retiradas na Biblioteca de sub-rotinas. O Bibliotecário constrói e insere, automàticamente, em posições apropriadas da Fita Biblioteca tôda informação requerida pelas fases de tradução e alojamento da compilação. Com o Bibliotecário o utilizador poderá construir e expandir uma Biblioteca de rotinas adequada para a manipulação de seus problemas especiais.

O Bibliotecário é chamado pela sequência de instruções

```
          75    30003   L
          11      B     00007
L         37    00006   00004
```

onde B é o primeiro endereço de três parâmetros consecutivos, assim definidos:

	xxx	xxxx	x	xx	xx
B	D	W	I	X	Y
	xxx	xxx	xxx	xxx	
B + 1	A	Q	N	P	
B + 2	Nome da rotina em XS-3.				

D é a posição de uma rotina a ser apagada. (Se D = 0, nenhuma rotina será apagada; se D = 3, a terceira rotina da Biblioteca será apagada).

W é o número em palavras da rotina a ser adicionada. (Não inclui Q, o número de provisórios.)

I é o número de entradas da rotina a ser adicionada.

X é o número do Uniservo onde a Fita Biblioteca estiver montada. (Se X = 0, o Bibliotecário toma como inexistente a Fita Biblioteca e escreve uma Biblioteca "muda" no Uniservo 4 antes de escrever a Biblioteca normal no Uniservo para isso destacado e adicionar-lhe a primeira rotina.)

Y é o Uniservo onde a Biblioteca revista deverá ser escrita, (Se X = 0, Y = 4.)

A é a posição de uma rotina a ser adicionada. (Se A = 0, nenhuma rotina será adicionada.)

Q é o número de provisórios requeridos pela rotina a ser adicionada.

N é o número de constantes relativas na rotina a ser adicionada.

P é o número de constantes fixas na rotina a ser adicionada.

Se o nome em XS-3 da rotina a ser adicionada contiver menos de seis caracteres, deverá ser completado com zeros à direita.

O Bibliotecário (uma fita bioctal) ocupa os endereços 00004-01332, e também usa endereços de 01333 a 03315 para reserva provisória. Qualquer rotina a ser adicionada é considerada armazenada com endereço inicial em 60000. Êste endereço pode, porém, ser mudado modificando a parte-u do endereço 00345. O Uniservo 3 é reservado para escrever uma fita de listagem contendo uma descrição das rotinas na Biblioteca. Essa lista será obtida cada vez que o Bibliotecário fôr usado.

CAPÍTULO 12

ROTINA DE CÓDIGO FLEX PARA EXCESSO-TRÊS

O preparo de fitas em UNICODE por intermédio da Flexowriter se torna possível pelo uso desta rotina. Ela é conveniente para programas curtos, fitas de correção, e fitas de dados, especialmente no caso em que não se tenha uma Unityper a disposição. A rotina está incluida na fita bioctal das Rotinas de Serviços do UNICODE, mencionada no Capítulo 9.

Quando a Flexowriter é utilizada da maneira descrita no fim dêste Capítulo, a rotina escreve uma fita que pode ser usada como entrada de UNICODE e que também pode ser listada na Impressora de Alta Velocidade.

Para operar a rotina a fita bioctal da Rotina de Serviços do UNICODE apropriada (1105 ou 1103A) deverá ser carregada no tambor. A fita da Flexowriter deve ser introduzida na unidade de leitura Ferranti. Se na 1105, o número da Unidade de Contrôle de Fita (TCU) é posto em Q. Se na 1103A, 2 é posto em Q. Em A é introduzido, em octal, o número do Uniservo onde a fita de saída é montada. Se fôr desejado re-enrolar o Uniservo após escrever, MJ1 deverá ser introduzido. Com a unidade de leitura, máquina de escrever e Uniservo ligados, a partida será dada com endereço inicial em 76300.

Duas advertências que poderão ocorrer são:

p

error _ _ _

A primeira significa um parâmetro ilegal em A ou Q e será causada se em Q não estiver 1 ou 2 ou se o número em A fôr menor que 1 ou maior que 10_{10}. Corrigir os parâmetros e dar nova partida sem mudar PAK remediará êsse êrro.

A segunda advertência significa que um símbolo flex ilegal apareceu na linha número - - -, o último sendo dado em forma decimal. Dando a partida após êsse êrro a conversão será continuada com omissão do símbolo errado.

Ao preparar a fita na Flexowriter se deve tomar cuidado para que nenhum "carriage return" adicional seja feito. As regras de preparação de fita dadas no Capítulo 9 têm de ser repetidas aqui.

Na preparação de fitas em código flex a seguinte correspondência entre símbolos "flex" e XS-3 deverá ser observada:

Caráter Flex	Caráter Excesso-Três
Letras Minúsculas	Letras Correspondentes
Números Minúsculos*	Números Correspondentes
Números Maiúsculos*	Números em expoentes
" - " Minúsculo	-
" - " Maiúsculo	Superscrito " - " (menos no expoente)
" . " Minúsculo	.
" . " Maiúsculo	Superscrito " . " (ponto decimal no expoente)
/	/
D Maiúsculo	Superscrito / (Divisão no expoente)
=	=
+	+
,	,
|	|
(	(
)	)
Espaço Maiúsculo	Δ
Espaço Minúsculo	Δ
G Maiúsculo	>
L Maiúsculo	<
X Maiúsculo	* (Multiplicação)

* Maiúsculo e Minúsculo aqui se referem às duas posições do carro da máquina de escrever com relação à cabeça do tipo impressor. Aplica-se também a números, no caso da Flexowriter. No caso de out os caracteres, ocorrem sempre dois por tecla; minúsculo é o que aparece escrito na parte inferior da tecla, maiúsculo o outro.

Tôdas as letras maiúsculas excetp X,D,G,L são ilegais. Estas quatro letras foram usadas para denotar símbolos usados por UNICODE que não estão no teclado da Flexowriter. Alguns dos códigos flex normais têm

significação especial. São êles:

Código Flex	Significação
Tape Feed or delete	Ignorado
Shift up or down	Permite a rotina decidir entre minúsculo e maiúsculo.
Tab	Imaginando o bloquete a ser impresso em grupos de seis caracteres XS-3, começando da esquerda, tab faz com que o resto do grupo seguinte de seis dígitos seja preenchido com Δ's.
Carriage Return	Faz com que o resto de um bloquete seja preenchido com Δ's. Um "carriage return" adicional preenche com Δ's o bloquete seguinte.
Stop	Interrompe a conversão e enche o resto do último bloco com Δ's.

Todos os códigos não mencionados especificamente acima são ilegais e darão origem a uma advertência de êrro, caso apareçam na fita de entrada.

O primeiro caráter significante numa fita flex tem de ser um shift up ou shift down; o último tem de ser um stop.

APÊNDICE A

LINGUAGEM DO UNICODE

Exemplos de Instruções em UNICODE.

1. COMPUTE△X△.
2. X = A△.
3.1 VARY△X△P(Q)R△SENTENCES△N△THRU△M△THEN△JUMP△TO△K△.
3.2 VARY△X△P(Q)R△SENTENCE△N△.
3.3 VARY△X△P(Q)R△WITH△Y△S(T)U△SENTENCES△N△THRU△M△.
4. JUMP△TO△SENTENCE△K△.
5.1 IF△X = Y△JUMP△TO△SENTENCE△K△.
5.2 IF△X△NOT = Y△JUMP△TO△SENTENCE△K△.
5.3 IF△X<Y△JUMP△TO△SENTENCE△K△.
5.4 IF△X>Y△JUMP△TO△SENTENCE△k,△IF△X<= Y△JUMP△TO△SENTENCE△m△.
5.5 IF△X<Y△JUMP△TO△SENTENCE△k,△IF△X = Y△JUMP△TO△SENTENCE△m,△IF△X>Y△JUMP△TO△SENTENCE△n△.
5.6 IF△X>= Y△JUMP△TO△SENTENCE△K△.
6. RESUME△K△.
7.1 LIST△X(I, J),△TAPE△L,△((Title))△.
7.2 LIST△F(X, Y, Z, T),△X,△Y,△Z,△T,△TAPE△L,△((Title)),△(Column Heading),△(Col. Hdg),△(Col. Hdg),△(Col. Hdg),△(Col. Hdg)△.
8. TYPE△X(I, J),△Y,△---,△F(X),△Z△.
9.1 READ△A△.
9.2 READ△A,IF△END△OF△DATA,△JUMP△TO△SENTENCE△K△.
10. PRINT△---------△.
11. START△.
12. STOP△.
13. END△OF△TAPE△.
14. DIMENSION△X(--, --),△Y(--, --, --, --),△---,△Z(--, --, --)△.

EXEMPLO DE INSTRUÇÃO DE CORREÇÃO

1. DELETE △SENTENCES△N△THRU△M△,

OPERAÇÕES SINGULARES

SÍMBOLO	OPERAÇÃO	EXEMPLO
+ (optativo)	mais	+ 157.25
-	menos	- 157.25
||	valor absoluto	| X - Y |

OPERAÇÕES BINÁRIAS

SÍMBOLO	OPERAÇÃO	EXEMPLO
+	adição	X + 5
-	subtração	Y - Z
*	multiplicação	Y*X*2. 5632
/	divisão	X/7.5
Super-índice numérico.	exponenciação(expoente numérico)	Z^5
POW	exponenciação(expoente numérico ou literal)	Z POW Y

RELAÇÕES

SÍMBOLO	RELAÇÃO	EXEMPLO
=	é igual a	Z = X + Y
Not =	é não igual a	X△NOT = Y
<	é menor que	X < Y
>	é maior que	X > Y
<=	é menor ou igual que	X <= Z
>=	é maior ou igual que	Y >= X

Fig. 1 (Primeiro Método).

APÊNDICE B

EXEMPLOS

I. Tabelamento de uma Função. Suponhamos que queremos tabelar os valores de y dados por

$$y(x) = \left[\sqrt{a^2+b^2}\, \log_e\left(\frac{x}{a}\right) + \left(\frac{b}{a}\right)^{(x/a)} \right] \cdot \left[1 - e^{-\left(\frac{a+d}{x}\right)} \right]$$

onde $d = e^{-\sqrt{\frac{a}{b}}} \log_e \left(\frac{a+1}{b+a} \right)$

para valores de x no intervalo de 1 a 10, os valores de b e a sendo b = 1, 3, 5 ; a = 0.9, 1.8, 3.6, 7.2, 14.4, 28.8.

Um programa possível para efetuar êsse cálculo é o seguinte:

Nº das Sentenças	Instruções
	△
△ △	UNICODE △PROGRAM △.
△ △	TABELAMENTO △DE △UMA △FUNÇÃO △.
△ △	△
△ △	△
△ △	△
△ △	△
1△ △	START △.
2△ △	VARY △B △1(3)5 △SENTENCES △3 △THRU △11 △.
3△ △	TYPE △B △.
4△ △	A=0.45 △.
5△ △	VARY △I △1(1)6 △SENTENCES △6 △THRU △11 △.
6△ △	A=2*A △.
7△ △	D=EXP(-(A/B)**1/2)*LN((A+1)/(B+A)) △.
8△ △	TYPE △A, △D △.
9△ △	VARY △X △1(0.1)10.01 △SENTENCES △10 △THRU △11 △.
10△ △	Y=(SQRT(A*A+B*B)*LN(X/A)+(B/A) △POW △(X/A)*(1-EXP(-(A+D)/X)) △.
11△ △	TYPE △Y △.
12△ △	STOP △.
13△ △	END △OF △TAPE △.

Nota: Ao valor máximo que x deve tomar na instrução VARY da sentença 9 (x = 10) foi dado um pequeno acréscimo para segurança, afim de garantir que o último valor seja incluido entre os valores assumidos por x. Exclusão pode ocorrer devido ao fato de que na conversão de números fracionários ao sistema binário uma aproximação ocorre, e essa aproxima - ção pode causar que ao ser dado o último acréscimo o valor superior fique excluido, por estar já fora do intervalo requerido.

II. Cálculo da Soma dos Termos de uma Série Convergente. Seja, por exemplo, calcular o valor de sen x para x=1, 2,3,---99,100 usando a fórmula

$$y = \text{sen } x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \text{--------} + (-1)^n \frac{x^{(2n-1)}}{(2n-1)!} + \text{---}$$

com uma precisão de 5 algarismos decimais.

A convergência dessa série é lenta para valores de x muito maiores que.1. É conveniente então usar as propriedades de periodicidade da função representada pela série, reduzindo os valores de x a números entre 0 e $\pm\ \pi/2$. Para isso podemos primeiro subtrair $2n\pi$ do valor original de x, de modo a obter um número entre $+\pi$ e $-\pi$. Depois usamos o fato de que para $\pi > x > \pi/2$ temos sen x = sen $(\pi - x)$, onde $\pi - x < \pi/2$ e o fato de que para $-\pi < x < -\pi/2$ temos sen x = = sen $(-x - \pi)$ onde $|-x-\pi| < \pi/2$. O seguinte programa resolve o problema. Mandaremos imprimir o valor original de x juntamente com o valor s do arco entre $-\pi/2$ e $+\pi/2$ que tem o mesmo seno, e o valor calculado para y = sen x.

Nº das Sentenças	Instruções
	UNICODE PROGRAM
	TABELA PARA A FUNÇÃO SENO
	_____
	_____
	_____
	_____

```
1       | START .
2       | PRINT TABELA DE VALORES DA FUNÇÃO SENO .
3       | VARY X 0(1)100 SENTENCES 9 THRU 28 .
9       | S = X .
10      | IF |S| < 3.1415926 JUMP TO SENTENCE 13 .
11      | S = S - 2 * 3.1415926 .
12      | JUMP TO SENTENCE 10 .
13      | IF |S| =< 1.5707963 JUMP TO SENTENCE 19 .
14      | IF S =< - 1.5707963 JUMP TO SENTENCE 17 .
15      | S = 3.1415926 - S .
16      | JUMP TO SENTENCE 19 .
17      | S = - 3.1415926 - S .
19      | A = 2 .
20      | Z = S .
21      | Y = S .
22      | B = A * ( A+1 ) .
23      | Z = - Z * S * S/B .
24      | Y = Y + Z .
25      | IF |Z| < 0.0001 JUMP TO SENTENCE 28 .
26      | A = A + 2 .
27      | JUMP TO SENTENCE 22 .
28      | TYPE X, S, Y .
29      | STOP .
ZZZZZZ  | END OF TAPE .
```

Abaixo mostramos a parte inicial da folha de resultados:

```
TABELA DE VALORES DA FUNÇÃO SENO .
X = 0
S = 0
```

```
Y = 0
X = 1.
S = 1.
Y = 0.841471008
X = 2.
S = 1.14159259
Y = 0.909297503
X = 3.
S = 0.141592592
Y = 0.141119947
X = 4.
S = - 0.858407407
Y = - 0.756801843
X = 5.
S = - 1.28318518
Y = - 0.958924636
X = 6.
S = - 0.283185184
Y = - 0.279415410
X = 7.
S = 0.716814815
Y = 0.656986549
X = 8.
S = 1.42477777
Y = 0.989359438
.................
.................
```

III. Preparação de um histograma. O que se deseja, no caso, é a distribuição de frequência de certos dados numéricos por intervalos de amplitude pre-fixada.

No exemplo que se segue obtém-se uma classificação de um grupo

de 100 números, não negativos e menores que 100, em intervalos sucessivos de amplitude 5, a partir do intervalo (0,5). Os dados são alimentados através de uma fita de dados e, neste caso, são chamados automàticamente pelo Leitor Automático de Dados (ver cap. 5). A apresentação dos resultados na Flexowriter terá o formato:

```
ZMIN = 0          (limite inferior do intervalo)
ZMAX = 5          (limite superior do intervalo)
M =  ...          (frequência de dados no intervalo)
ZMIN = 5
ZMAX = 10
M = ...
     .
     .
     .
ZMIN = 95
ZMAX = 100
M =  ...
I =  ...
```

I dará o número total de dados diferentes de zero e menores que 100 encontrados na fita.

Nº das Sentenças	Instruções
	UNICODE PROGRAM.
	HISTOGRAMA.
	_____
	_____
	_____
	_____
1	DIMENSION ΔX(100)Δ.
2	START Δ.
3	I=0Δ.
4	VARY Δ ZMIN Δ 0(5)95 Δ WITH Δ ZMAX Δ 5(5)100 Δ SENTENCES Δ 5 Δ THRU Δ 13 Δ.
5	TYPE Δ ZMIN, Δ ZMAX Δ.

```
6       | M=0Δ.
7       | VARYΔKΔ1(1)100ΔSENTENCESΔ8ΔTHRUΔ11Δ.
8       | IFΔX(K)>ZMAXΔJUMPΔTOΔSENTENCEΔ11Δ.
9       | IFΔX(K)<=ZMINΔJUMPΔTOΔSENTENCEΔ11Δ.
10      | M=M+1Δ.
11      | RESUMEΔ7Δ.
12      | TYPEΔMΔ.
13      | I=I+MΔ.
14      | TYPEΔIΔ.
15      | STOPΔ.
ZZZZZZ  | ENDΔOFΔTAPEΔ.
```

IV - Ajuste de uma Reta a um Grupo de Pontos pelo Método dos Mínimos Quadrados. Neste exemplo são obtidos os parâmetros da reta que melhor representa a relação entre dois grupos de dados, de acôrdo com o critério dos mínimos quadrados.

Sejam $\{X_i\}$, $\{Y_i\}$, $i = 1,2 \ldots N$, dois tais conjuntos. Se uma relação linear é suposta existir entre seus elementos, o critério dos mínimos quadrados fornece um meio de selecionar os valores dos parâmetros A e B da reta

$$Y = A + B*X$$

que melhor representará aquela relação.

Em geral os elementos de um dos grupos, por exemplo de $\{Y_i\}$, não têm a mesma significação estatística, de modo que, com cada elemento, Y_i, vem associada uma variância intrínseca, S_i, que funciona como um índice da precisão com que cada Y_i pode ser dado.

De acôrdo com o critério dos mínimos quadrados, A e B podem ser determinados a partir de:

$$\Sigma\ W_i\ [Y_i - (A + B*X_i)]^2 = \text{mínimo}$$

Os W_i são pêsos atribuidos aos diferentes elementos de $\{Y_i\}$ e que refletem o fato de que alguns são mais bem conhecidos que outros.

É muito conveniente a escôlha:

$$W_i = S^2/S_i^2 ,$$

onde

$$S^2 = 1/\Sigma(1/S_i^2)$$

Nesse caso $\Sigma W_i = 1$ e os valores de A e B que satisfazem à condição do mínimo acima são dados por:

$$A = \overline{Y} + \frac{X}{D} (\overline{X}*\overline{Y} - \overline{X*Y})$$

$$B = \frac{1}{D}(X*Y - \overline{X*Y}),$$

com $D^2 = \overline{X^2} - (X)^2$

As variâncias de A e de B, V(A), V(B) serão dadas por:

$$V(A) = S \quad ; \quad V(B) = S/D$$

(Nas fórmulas acima, uma barra superior denota valor médio ; assim:

$$\overline{XY} = \sum_{i=1}^{N} W_i \, X_i*Y_i \text{ , etc.}$$

No programa que se segue a notação é:

$$\overline{X} \equiv XM \; ; \; \overline{Y} \equiv YM \; ; \; \overline{X^2} \equiv XXM \; ; \; \overline{XY} \equiv XYM.$$

Os dados serão fornecidos por meio de uma fita de dados (ver capítulo 5).

O programa foi feito para o caso específico de dez valores de cada variável de sentença READ.

Nº das Sentenças	Instruções
	UNICODE PROGRAM . RETA MMQ . _____ _____

```
1       DIMENSION X(10), Y(10), S(10), W(10) .
2       START .
3       READ X, Y, S .
4       SS = 0 .
5       XM = 0 .
6       YM = 0 .
7       XYM = 0 .
8       XXM = 0 .
9       VARY K 0(1)9 SENTENCES 10 THRU 15 .
10      W(K) = 1/(S(K) * S(K)) .
11      SS = W(K) + SS .
12      XM = W(K) * X(K) + XM .
13      YM = W(K) * Y(K) + YM .
14      XYM = W(K) * X(K) * Y(K) + XYM .
15      XXM = W(K) * (X(K)) POW(2) .
16      SS = 1/SS .
17      D = SS * XXM - (SS*XM) POW(2) .
18      A = SS*(YM + XM*(SS*SS*XM*YM-SS*XYM)/D) .
19      B = (SS*SS*XM*YM-SS*XYM)/D .
20      VA = SQRT(SS) .
21      VB = SQRT(SS)/SQRT(D) .
22      TYPE A, B, VA, VB .
23      STOP .
ZZZZZZ  END OF TAPE .
```

V - Integração: Vamos programar êste problema de dois modos.

As vantagens e desvantagens dos dois métodos serão discutidas em seguida.

Problema: Calcular a seguinte integral por meio da regra de Simpson, com um intervalo de cinco milésimos. (Regra de Simpson é $\frac{1}{3} h \left[y(x) + 4y(x + h) + y(x + 2h) \right]$; h = 0.005.

$$T = \int_0^{\pi/2} \frac{dx}{\sqrt{1 - \frac{1}{2}\sin^2 x}}$$

1º método:

Nº das Sentenças	Instruções
	UNICODE PROGRAM . PRIMEIRO MÉTODO DE INTEGRAÇÃO .
1	DIMENSION Y(3) .
2	Y(I) = 1/(1 - 0.5 * (SIN(X + Z))^2)$^{1/2}$.
3	U = (1/600)*(Y(1) + 4 * Y(2) + Y(3)) .
4	START .
5	T = 0 .
6	VARY X 0(0.01)1.56 SENTENCES 7 THRU 10 .
7	VARY I 1(1)3 WITH Z 0(0.005)0.01 SENTENCE 8 .
8.	COMPUTE Y(I) .
9	COMPUTE U .
10	T = T + U .
11	TYPE T .
12	STOP .
ZZZZZZ	END OF TAPE .

2º método:

Nº de Sentenças	Instruções
	UNICODE PROGRAM . SEGUNDO MÉTODO DE INTEGRAÇÃO .
1	DIMENSION Y(3) .
2	$Y(I) = 1/(1 - 0.5 * (SIN (X + Z))^2)^{1/2}$.
3	U = (1/600)*(Y(1) + 4 * Y(2) + Y(3)) .
4	START .
5	T = 0 .
6	Y(1) = 1 .
7	VARY X 0(0.01)1.56 SENTENCES 8 THRU 12 .
8	VARY 1 2(1)3 WITH Z 0.005(0.005)0.01 SENTENCE 9 .
9	COMPUTE Y(I) .
10	COMPUTE U .
11	T = T + U .
12	Y(1) = Y(3) .
13	TYPE T .
14	STOP .
ZZZZZZ	END OF TAPE .

O primeiro método usa variáveis rotuladas, e, afim de obter os argumentos apropriados para a função seno, duas instruções VARY são requeridas.

O segundo método também usa variáveis rotuladas e duas instruções VARY, mas elimina o cálculo redundante de Y(1) dentro dos ciclos de VARY. Entretanto, êste método requer que Y(1) tenha o valor inicial igual a 1.

VI - Geração de uma tabela de Integrais. Enquanto que os programas precedentes calculam apenas um valor da integral elítica mostrada, o seguinte programa vai um passo adiante e computa tôda uma tabela de valôres.

Problema : Compute uma tabela de valôres aproximados da integral elítica, usando a regra de Simpson com intervalo de um milésimo.

$$F(S, Z) = \int_0^Z \frac{dx}{\sqrt{1-S^2 \sin X}}$$

para $1^o \leq Z \leq 90^o$, $5^o \leq \text{arc sin } S < 90^o$.

Nº das Sentenças	Instruções
	UNICODE PROGRAM .
	TABELA DA INTEGRAL ELÍTICA .
1	U = (1/3000) * (1/(1 - S**2 * SIN X)**1/2 .
	+ 4 * (1/(1 - S**2 * SIN (X + 0.001))**1/2) .
	+ 1/(1 - S**2 * SIN (X + 0.002))**1/2) .
2	S = SIN A .
3	V = R * A .
4	W = R * Z .
5	START .
5.1	PI = 3.14159265536 .
5.2	R = 180/PI .
5.3	T = PI/180 .
5.4	H = 0.0872664626 .
5.5	G = 1.5708963268 .
5.6	E = 1.4907963268 .
6	VARY A H (H) E SENTENCES 7 THRU 13 .
7	VARY Z T (T) G SENTENCES 8 THRU 13 .
8	F = 0 .
9	VARY X 0(0.002) Z SENTENCES 10 THRU 11 .
10	COMPUTE S AND U .
11	F = F + U .

```
12      COMPUTE V AND W .
13      LIST F, V, W, TAPE 3, ((ELLIPTIC INTEGRALS OF THE FIRST
        KIND)),(INTEGRAL),(ARCSIN S),(UPPER LIMIT Z) .
14      STOP .
ZZZZZZ  END OF TAPE .
```

O limite superior G, correspondendo a medida em radianos do ângulo de 90°, foi acrescido de .0001 para assegurar que uma aproxima - ção por excesso do valor 89°, devido ao uso de aritmética de ponto flutu ante, não resultaria em suspender a variação do ciclo, comandado pela sentença de VARY, cedo demais. Semelhantemente, E é maior que a medida de 85° em radianos.

Alguns valores da tabela gerada são dados abaixo (V variando em intervalos de 5° aproximadamente e W em intervalos de 1°; para cada valor de V há 90 de W):

INTEGRAIS ELÍTICAS DE PRIMEIRA ESPÉCIE

F INTEGRAL		V ARCSIN S	W UPPER LIMIT Z
1.80006155	E -2	5.	1.00000001
3.60024619	E -2	5.	2.00000002
5.40055382	E -2	5.	3.00000002
.		.	.
.		.	.
.		.	.
.		.	.
1.5781968		5.	90.0000085
1.80024430	E -2	10.	1.00000001
3.60097736	E -2	10.	2.00000002
.		.	.
.		.	.
.		.	.
.		.	.
1.58736017		10.	90.0000085
1.80054290	E -2	15.	1.00000001

VII - Solução de Uma Equação Diferencial. Dada a equação diferencial:

$$\frac{dy}{dx} = x^3 - 3xy^3 + 2, \quad \text{com a condição } y(0) = 0$$

encontrar a sua solução no intervalo (0,3) usando o método de Runge-Kutta com um intervalo H entre valores sucessivos de x.

Nº das Sentenças	Instruções
	UNICODE PROGRAM . RUNGE-KUTTA METHOD FOR SOLVING DIFFERENTIAL EQUATION .
1	R1 = (X**3 - 3 * X * Y**3 + 2) * H .
2	R2 = ((X + H/2)**3 - 3 * (X + H/2) * (Y + R1/2)**3 + 2) * H .
3	R3 = ((X + H/2)**3 - 3 * (X + H/2) * (Y + R2/2)**3 + 2) * H .
4	R4 = ((X + H)**3 - 3 * (X + H) * (Y + R3)**3 + 2) * H .
5	DY = 1/6 * (R1 + 2 * R2 + 2 * R3 + R4) .
6	START .
7	Y = 0 .
8	H = 0.01 .
9	VARY X 0(H)3 SENTENCES 10 THRU 12 .
10	LIST Y, X, TAPE 4 .
11	COMPUTE R1 AND R2 AND R3 AND R4 AND DY .
12	Y = Y + DY .
13	STOP .
ZZZZZZ	END OF TAPE .

Em lugar de usar a lêtra H, seu valor, 0.01 no caso, poderia ser escrito nas equações e na sentença VARY. O método apresentado, porém, facilita a alteração de H quando requerida.

Neste problema a sentença LIST é colocada seguindo o VARY de modo que valores correspondentes de X e Y serão impressos em duas colunas adjacentes.

VIII - Integração usando uma Pseudo-Operação.

Problema: Integre as três funções:

$$G(X) = X^2$$
$$H(X) = X^3 - 3$$
$$P(X) = X^4 + 4$$

no intervalo (1,100), escolhendo um sub-intervalo de 0.01.

Nº das Sentenças	Instruções
	UNICODE PROGRAM .
	INTEGRAÇÃO DE FUNÇÕES .
1	DIMENSION Z(3) .
2	G(X) = X**2 .
3	H(X) = X**3 - 3 .
4	P(X) = X**4 + 4 .
5	START .
6	COMPUTE INT(G, Z(L)) AND INT(H, Z(2)) AND INT(P, Z(3)) .
8	TYPE Z(1), Z(2), Z(3) .
9	STOP .
10	INT(F, R(I)) .
11	R(I) = 0 .
12	VARY Y 1(0.01)99.99 SENTENCES 13 THRU 14 .
13	COMPUTE F(Y) .
14	R(I) = R(I) + F * 0.01 .
15	EXIT .
ZZZZZZ	END OF TAPE .

No programa acima, quando computando INT(G, Z(1)), R(I) é igualado a Z(1), de modo que a sentença R(I) = 0 no sub-programa faz Z(1) = 0. A sentença COMPUTE F(Y) faz computar G(Y), e na equa-

ção seguinte, F usa o valor de G(Y) já computado. Quando a instrução EXIT fôr atingida, a integral de G(Y) estará em Z(1). INT(H, Z(2)) e INT(P, Z(3)) são processados anàlogamente. Observe que R(I), uma variável rotulada muda, não aparece na sentença DIMENSION, e F, uma função muda, não tem uma equação de definição.

GLOSSÁRIO

BIBLIOTECA - Uma coleção de rotinas de cálculo usadas frequentemente, tais como as de seno, coseno, raiz quadrada.

BIBLIOTECÁRIO - Uma rotina de serviços, independente da fita de Biblioteca, que fornece os meios de fazer inserir e retirar subrotinas na fita de Biblioteca.

BLOCO - Conjunto de 6 bloquetes, constituindo o quantum de leitura de fita magnética pelo computador.

BLOQUETE - Conjunto de 20 palavras, equivalendo a uma linha de escrita na Unityper.

CODIFICAÇÃO - Uma sequência de instruções para computador.

CÓDIGO FLEX - Código produzido em fita perfurada pela Flexowriter.

CÓDIGO EXCESSO TRÊS - Habitualmente representado por XS-3 - é um sistema de código que representa cada algarismo, letra, símbolo de operação, etc, por um número de dois digitos octais. É o código produzido em fita magnética pela Unityper.

COMPILAÇÃO - Processo pelo qual se produz o programa Objeto a partir do programa Original utilizando-se o Compilador apropriado.

COMPILADOR - Consiste em um conjunto de instruções escritas em fita magnética, servindo para transformação de um programa escrito em um código qualquer para produzir um programa em linguagem diretamente aceita pela máquina.

EQUAÇÃO - Definição de uma variável por uma expressão escrita à direita do símbolo de igualdade.

FITA MESTRA DO UNICODE - Fita magnética gravada com as instruções que constituem o compilador do sistema Unicode.

FLEXOWRITER - Máquina de escrever com a qual se produz fita de papel perfurada em código Flex.

INSTRUÇÃO - Parte de uma sentença que o sistema Unicode utiliza para produzir uma sequência de instruções em linguagem de máquina ou para indicar condições especiais.

LISTAGEM - Gravação em fita magnética de tabelas de resultados de cálculo ou de descrição detalhada do Programa Objeto.

MEMÓRIA DE ALTA-VELOCIDADE - Reservatório de acesso rápido de nucleos magnéticos consistindo de um máximo de 4096 locações.

OPERAÇÃO BINÁRIA - Operação que se aplica a dois operandos.

OPERAÇÃO SINGULAR - Operação aplicada a um só operando.

OPERANDO - Variável função ou constante à qual uma operação se refere.

PALAVRA - Conjunto de seis símbolos de escrita (isto é, de letras, algarismos, símbolos de operação, etc.) correspondendo a 36 digitos binários.

PREFACIO - Codificação necessária para transferir dados do tambor magnético para a memória de alta velocidade antes da execução de um segmento.

PROGRAMA OBJETO - Conjunto de instruções escritas em linguagem de máquina, produzido na compilação do programa Original.

PROGRAMA ORIGINAL - Sequência de instruções e equações escritas em linguagem de Unicode.

SEGMENTAÇÃO - Processo, realizado na compilação, que consiste em dividir o programa objeto em segmentos.

SEGMENTO - Um conjunto de instruções e de dados por ela referidos, ocupando um determinado número de locações da memória de alta velocidade.

SUB-PROGRAMA - Sequência de instruções colocada após STOP, encabeçadas por um título que funciona como símbolo para referência da sequência, e que é executada quando requerido no corpo do programa principal.

TAMBOR MAGNÉTICO - Reservatório de acesso lento, consistindo de 16384 locações.

TERMINAÇÃO - Codificação necessária para transferir dados da memória de alta velocidade para o tambor magnético após a execução de um segmento.

UNICODE - Sistema automático de codificação para o Univac 1105.

UNITYPER - Máquina de escrever com a qual se imprime em fita magnética símbolos em código Excesso-Três e simultâneamente em folha de papel em escrita usual.

VARIÁVEIS DE PONTO FIXO - Variáveis cujos símbolos iniciam-se por uma das letras I,J,K,L,M e que podem assumir apenas valores inteiros.

VARIÁVEIS DE PONTO FLUTUANTE - Variáveis que podem assumir qualquer valor cujo módulo esteja entre 10^{-38} e 10^{38}.

VARIÁVEIS ROTULADAS - Variáveis possuindo índices.

APÊNDICE D - CÓDIGO FLEXOWRITER DE PERFURAÇÃO

Símbolos	Perfuração	Octal
	•	
A a	oo•	30
B b	o • oo	23
C c	o•oo	16
D d	o • o	22
E e	o •	20
F f	o •oo	26
G g	o• oo	13
H h	•o o	05
I i	o•o	14
J j	oo• o	32
K k	oo•oo	36
L l	o• o	11
M m	•ooo	07
N n	•oo	06
O o	• oo	03
P p	o•o o	15
Q q	oo•o o	35
R r	o• o	12
S s	o •o	24
T t	• o	01
U u	oo•o	34
V v	o•ooo	17
W w	oo• o	31
X x	o •ooo	27
Y y	o •o o	25
Z z	o • o	21

Símbolos	Perfuração	Octal
- -	o o•oo	56
. =	o •o	44
/ +	o o•o	54
(,	o •oo	46
) .	o • o	42
- |	o o•	50
1 1	o o• o	52
2 2	ooo•o	74
3 3	ooo•	70
4 4	oo •o	64
5 5	oo • o	62
6 6	oo •oo	66
7 7	ooo• o	72
8 8	oo •	60
9 9	oo• oo	33
0 0	oo•ooo	37
SPACE	•o	04
BACK SPACE	oo • o	61
CAR.RETURN	o •o o	45
TABULATOR	o o• o	51
STOP	o • oo	43
SHIFT UP	o •ooo	47
SHIFT DOWN	o o•ooo	57
COLOUR SH.	• o	02
CODE DELETE	ooo•ooo	77
	•	