CP 300

Operação e Programação

EDITELE

CP 300
Operação
e
Programação

1.ª Edição
1983

EDITELE

Editora Técnica Eletrônica Ltda.
São Paulo - Brasil

Editele — Editora Técnica Eletrônica Ltda.
Rua Casa do Ator, 1060
04546 — São Paulo — SP — Brasil
Caixa Postal 30141

1ª Edição

Impresso no Brasil

Advertência

Antes de ligar o seu CP 300, ou qualquer dos periféricos, verifique se as tensões indicadas nos equipamentos correspondem à tensão da rede local.

Leia com atenção, na primeira parte deste livro, as instruções sobre o procedimento correto para operar o computador e seus periféricos.

Caso não sejam seguidas as instruções do fabricante quanto a instalação do computador, este poderá causar interferência em receptores de rádio ou televisão.
Isto se deve ao fato dc os componentes do CP 300 operarem em velocidade elevada, gerando sinais elétricos de alta freqüência.

Pode-se tentar as seguintes medidas para eliminar a interferência no seu receptor:

- Mudar a posição da antena do receptor;
- Mudar a posição do computador em relação ao receptor;
- Afastar o computador do receptor;
- Conectar o computador a uma tomada que não esteja na mesma ramificação em que se encontra o receptor.

Sumário

Introdução

Basic

Apêndices

Introdução

Apresentação

Parabéns, você acaba de adquirir um microcomputador com a garantia da marca Prológica.

O CP 300 foi projetado para acompanhar o seu desenvolvimento técnico, na medida em que possibilita expandir gradualmente os seus recursos funcionais.

Devido a sua estrutura modular, o CP 300 poderá ser dimensionado de acordo com as suas possibilidades.

Num futuro próximo, o computador fará parte de sua vida.

Aprenda a conversar com ele, através de sua linguagem, o Basic, e ele fará tudo o que você quiser. Prepare-se para o futuro e vença esse desafio!

Como utilizar este manual

Todas as informações necessárias para manipular corretamente seu CP 300 estão neste livro.

Ele irá auxiliá-lo na instalação e operação do micro.

Você irá aprendendo a manipulá-lo aos poucos, de tal maneira que ao terminar a leitura deste manual, você será capaz de dominar todos os recursos que a máquina lhe oferece.

Aconselhamos ao principiante a consulta, sempre que necessária, ao glossário, o que lhe possibilitará entender as palavras comumente usadas em informática.

Boa Sorte!

Descrição do Computador

O CP 300 foi desenvolvido a partir de uma arquitetura de hardware que segue as mais avançadas tendências tecnológicas.

Sua apresentação é compacta, com caixa injetada em ABS, material de alta qualidade e resistência, devendo ser conectado a um televisor comum (P&B ou colorido), e permite ainda outras expansões: gravador cassete comum, até 4 (quatro) unidades de discos flexíveis de 5¼'', impressora, joystick, interface serial RS-232C, etc...

Como é constituído o CP 300

UCP

Unidade Central de Processamento com um microprocessador Z-80A, 2MHz; que é responsável pelo controle e organização de todas as funções do sistema.

Memória

16 k bytes de memória ROM, onde estão armazenados os programas internos do seu computador, incluindo a linguagem Basic-Residente.

48 k bytes de memória RAM, onde é permitido o armazenamento temporário de seus programas. Essa memória só se encontra ativada durante o funcionamento do equipamento.

1 k byte de memória RAM utilizada para buffer de vídeo.

Teclado

Teclado alfanumérico com 54 teclas e uma barra de espaço, que permite a entrada de todo o texto normal e de caracteres de controle.

Todo o teclado é auto repetitivo.

TV

O CP 300 utiliza qualquer televisor comum (P&B ou colorido) que funcionará como monitor de vídeo. É permitido ao usuário a formatação do vídeo feita por software que permite obter-se:

normal — 16 linhas com 64 caracteres por linha
expandido — 16 linhas com 32 caracteres por linha
gráficos — 48 linhas com 128 caracteres por linha

Fonte de Alimentação

A fonte de alimentação lhe permite usar seu microcomputador em 115 ou 230 V.

Som

Seu microcomputador possui um amplificador de áudio com auto-falante e ajuste de volume. A resposta em freqüência é de 10Hz a 10kHz com uma potência de saída de 1,5 watt RMS.

Expansões

O CP 300 é um equipamento modular, o que lhe permite expandi-lo conforme suas necessidades. Você poderá conectar:

Cassete

Utiliza-se um gravador cassete comum para armazenamento de programas e dados.

Pode-se utilizar duas velocidades de transferência de dados entre o micro e o gravador:

A — 500 bauds
B — 1.500 bauds

Drives

Existe a possibilidade de se trabalhar com até 4 unidades de disco flexível de 5 ¼ ''.

Você poderá armazenar e carregar programas e dados com grande rapidez e confiabilidade. Você terá disponível 720 k bytes para seu uso. Maiores informações lhe serão fornecidas no Sistema de Operação de Disco do CP 300.

Impressora

Você pode conectar qualquer impressora com interface paralela ou serial ao CP 300. Isto lhe será de grande utilidade para listagem de programas, relatórios, etc...

Interface RS 232C

Esta interface lhe permitirá comunicar-se com outros computadores, que possuam a mesma interface. O seu computador poderá ser programado para se comunicar através do telefone.

Joystick

Você poderá conectar um joystick ao seu CP 300, o que lhe permitirá um melhor desempenho em seus jogos.

Vídeo

Na saída para vídeo do seu CP 300 você poderá conectar um monitor de vídeo, o que lhe possibilitará a obtenção de uma imagem superior àquela obtida num televisor comum.

Instalação do Computador

Tenha cuidado ao desembalar seu computador. Tire-o com cuidado da embalagem e guarde todas as proteções para o caso de precisar transportá-lo futuramente. Tenha certeza de que retirou a fonte bem como todos os cabos e documentos que acompanham o seu CP 300.

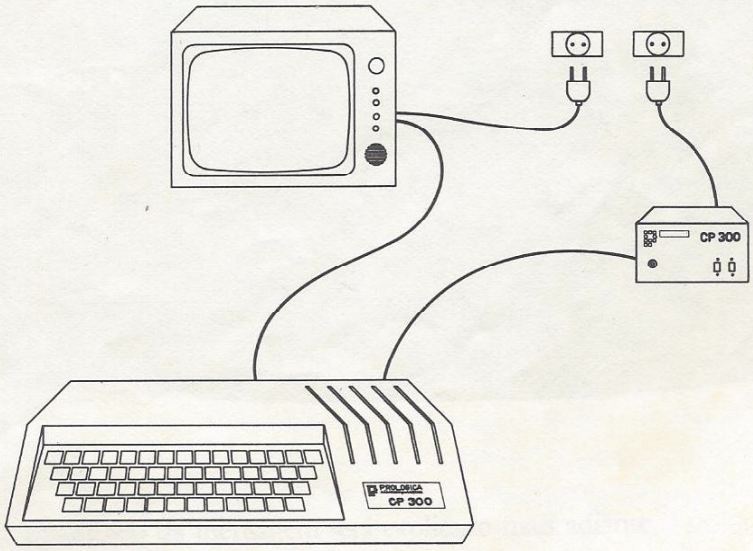
Coloque o computador e a fonte sobre a superfície em que irá manuseá-los. Uma tomada de força deve estar próxima, para ser evitado o uso de extensão.

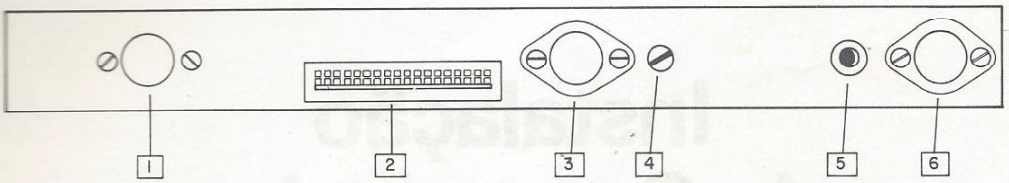
Montagem

Antes de conectar qualquer periférico (TV, gravador), certifique-se de que o computador e os periféricos estejam desligados.

Primeiramente conecte o cabo para televisor na entrada da antena do seu aparelho de TV e na tomada atrás do CP 300 marcado TV. Em seguida conecte o cabo da fonte no local marcado FONTE.

Ligue o televisor e a fonte à rede elétrica local verificando se a tensão local (115 ou 230 V) corresponde ao indicado na fonte (chave seletora de tensão).





1 — Fonte (conectar o cabo da fonte)
2 — Expansão (drives-impressora, etc.)
3 — Vídeo
4 — Volume
5 — TV (conectar a antena da TV)
6 — K7 (conectar o gravador cassete)

Operação

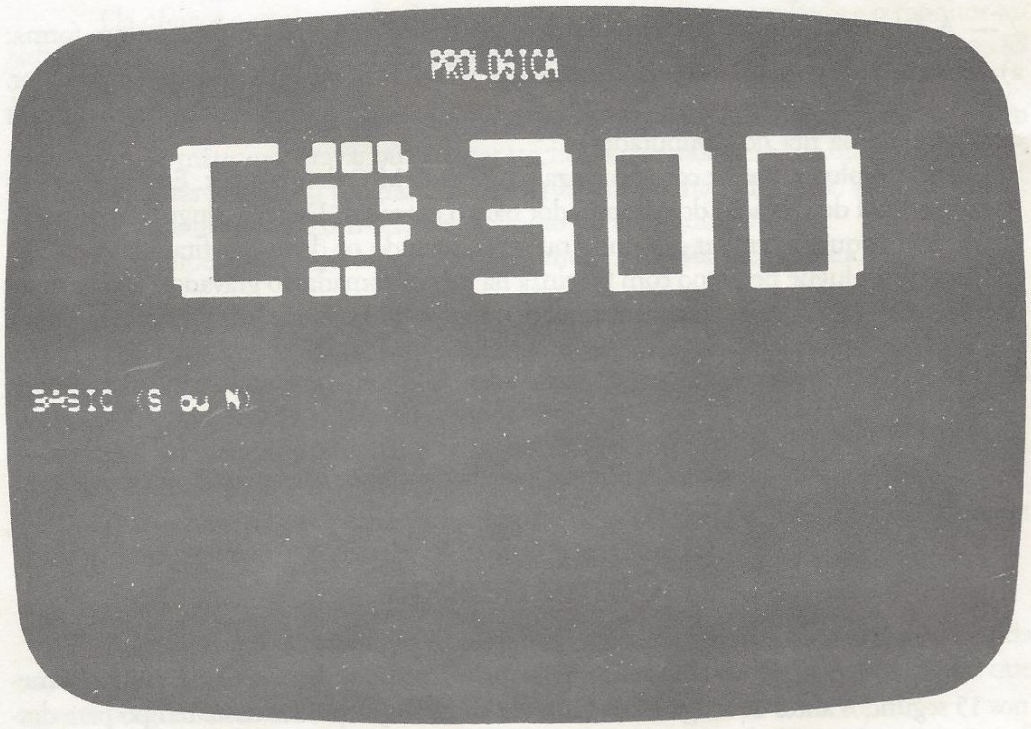
Seu computador está pronto para operar:

a) Ligue a TV

b) Coloque o seletor de canais no canal 3 VHF

c) Ligue a fonte

A tela de seu televisor deverá ficar assim:



O significado da mensagem será explicado mais adiante.

Se a mensagem não aparecer:

a) A TV pode estar precisando de ajuste de sintonia ou de brilho. Procure no seu televisor os respectivos controles e faça os ajustes necessários.
b) Se a mensagem ainda não apareceu, então desligue todo o sistema e verifique as conexões. Para uma verificação mais detalhada veja o capítulo "Detecção de Falhas e Manutenção".

Conexão com o gravador de fita

Você não precisa conectar o gravador, a menos que pretenda gravar ou carregar (copiar) programas do seu CP 300.

As instruções a seguir se baseiam num gravador cassete comum.

O cabo de conexão acompanha o equipamento; e sugerimos que seja sempre usado este cabo.

1 — Conecte o cabo (plugue DIN em uma extremidade e três plugues na outra) na tomada "K7" atrás do computador e certifique-se de que o plugue está corretamente conectado.

2 — A extremidade com três plugues deve ser ligada ao gravador, da seguinte forma:

a) Conecte o plugue com fio preto na tomada MONITOR ou EAR no lado do gravador. Esta conexão fornece o sinal de saída do gravador para o computador (para carregar programas da fita no computador).
b) Conecte o plugue maior com fio cinza na tomada MIC do gravador. Esta ligação fornece o sinal de gravação do computador para o gravador. Deixe o plugue monitor conectado enquanto estiver gravando ou reproduzindo os dados da fita.
c) Conecte o plugue pequeno com fio cinza na menor tomada do gravador. Isto permite que o CP 300 controle automaticamente o motor do gravador (movimenta ou pára a fita, para reproduzir, ou gravar, os programas e dados).

Atenção:

Certifique-se de que não existe nenhum microfone interno ou plugue postiço na entrada de microfone do gravador.

Como desligar

Primeiramente desligue a fonte e depois os periféricos.

Se você desligar o computador por qualquer razão, deixe-o como está pelo menos 15 segundos antes de religá-lo. A fonte de alimentação precisa deste tempo para descarregar completamente a energia armazenada.

Sempre que desligar seu computador, todos os programas e dados serão apagados. Então não se esqueça de guardar as informações (por exemplo, em fita) antes de desligar o computador.

Botão Reset

Seu CP 300 possui duas teclas de RESET (vermelhas).

Para acionar o Reset, você deverá pressionar as duas teclas ao mesmo tempo. Isto evitará um acionamento involuntário.

Para reiniciar a operação após uma falha no fornecimento de energia, você não precisará desligar e ligar o computador novamente. Basta apenas acionar o Reset e você conseguirá o mesmo efeito.

Para interromper um programa ou operação sem perder seu programa Basic, ou os dados, pressione a tecla BREAK.

Diálogo inicial

Quando você liga o computador, ou pressiona RESET, o computador começa suas operações com uma série de perguntas, denominada "diálogo inicial". A primeira pergunta é:

```
Basic (S ou N)?
```

Ela oferece a você a opção de escolher entre o interpretador Basic e o monitor-residente. Como resposta você deve digitar S para operar o computador em Basic, ou N para usar o monitor-residente.

O monitor-residente é um programa de apoio para depuração de programas em linguagem de máquina, gravar ou carregar trechos de memória em fita ou ainda, confeccionar pequenas rotinas ou programas em linguagem de máquina do Z 80. O apêndice G contém os comandos e o modo de operação do monitor-residente.

Respondendo com S à primeira pergunta, o CP 300 perguntará:

```
Cass?
```

Essa pergunta se refere à velocidade de transferência dos dados do computador para o gravador e vice-versa. Você pode escolher entre velocidade alta (1500 bauds), pressionando A, ou baixa (500 bauds), usando B.

Pode-se, ainda, pressionar a tecla ENTER ao invés dessas, e o CP 300 assumirá que foi escolhida a velocidade alta.

A terceira e última pergunta diz respeito à quantidade de memória que deve ser usada pelo computador, e aparece no vídeo da seguinte forma:

```
Mem. usada?
```

A resposta a essa pergunta estabelecerá o limite superior de endereçamento da memória RAM que o interpretador Basic poderá utilizar. Será dentro desses limites que os seus programas e dados serão armazenados.

Responda simplesmente com ENTER a essa pergunta. O CP 300 entenderá com isso que você deseja usar toda a capacidade de armazenamento da RAM. Programadores experientes podem querer reservar parte da memória para programas e rotinas em linguagem de máquina.

Terminado o diálogo inicial o computador indicará que está pronto para uso com a mensagem:

```
PROLOGICA
READY
>
```

Usando o teclado

A finalidade do teclado é permitir a introdução de todo o texto normal de um programa, bem como os dados e os caracteres de controle. Como no teclado das máquinas de escrever comuns, certas teclas possuem dois símbolos. Para introduzir o símbolo inferior basta pressionar a tecla, enquanto que para o símbolo superior deve ser pressionada a tecla desejada juntamente com SHIFT. O ponto de interrogação, por exemplo, é introduzido pressionando-se SHIFT e a tecla /.

Maiúsculas e minúsculas (SHIFT 0)

As teclas de A a Z podem produzir letras maiúsculas e minúsculas, como as das máquinas de escrever. Porém o CP 300 possui dois modos distintos de digitação: só maiúsculas e, maiúsculas e minúsculas. O computador começa suas operações aceitando apenas maiúsculas, o que facilita a digitação de programas. Quando você quiser usar maiúsculas e minúsculas, para digitar mensagens, por exemplo, deve pressionar SHIFT 0. Dessa forma as teclas passam a introduzir minúsculas diretamente e maiúsculas quando usadas juntamente com SHIFT. Para voltar ao modo inicial basta pressionar SHIFT 0 novamente.

Teclas especiais

Certas teclas têm funções especiais em Basic e, ao invés de aceitá-las como caracteres, o computador executa as funções a elas atribuídas.

Tecla	*Função*
←	Retrocede e apaga o último caractere digitado.
→	Passa o cursor para o início do próximo grupo de oito colunas, na mesma linha (tabulação).
SHIFT ←	Retorna ao começo da linha.
SHIFT →	Muda o vídeo para 32 caracteres por linha.
SHIFT @	Força uma pausa na execução de um programa. Pressione qualquer tecla para retomar o processamento.
ENTER	Introduz uma linha. O interpretador não reconhece nenhuma digitação até que esta tecla seja usada.
CLEAR	Cancela a linha que estiver sendo digitada, apaga toda a tela, converte para 64 caracteres por linha e posiciona o cursor no canto superior esquerdo.

Tecla	*Função*
BREAK	Interrompe qualquer operação do computador e devolve o controle para o teclado.
SHIFT ↓ * ou S P	Ativa a função de impressão do conteúdo da tela na impressora. A tecla BREAK interromperá essa função.

Outras características

Toda tecla tem função de auto-repetição, ou seja, quando você mantém qualquer tecla pressionada por mais que um segundo, o caractere corespondente passa a ser repetido na tela, enquanto a tecla assim permanecer.

Códigos de controle

Se você não está familiarizado com o conceito de códigos de controle, consulte o apêndice C, "Códigos de Caractere".

Existem, no código ASCII, 32 caracteres de controle (códigos 0 a 31). Alguns deles são introduzidos diretamente por teclas especiais (ENTER e BREAK , por exemplo). Outros são introduzidos por combinação das teclas SHIFT e ↓ com letras.

No apêndice C existe uma lista completa dos códigos de caractere de controle.

Nota:

Deve ser utilizada a tecla SHIFT da esquerda.

Basic

Comandos Basic

Antes de começarmos a codificar programas, precisamos nos familiarizar com alguns conceitos básicos:

— Diferenças entre comandos e instruções.
— Meios de representação (em Basic) de dados
— Constantes.
— Variáveis.

Existem dois grupos de palavras chaves, as quais diferenciamos por comandos e instruções de uma linguagem.

O principal objetivo dos comandos é ordenar à máquina que execute algum trabalho interna ou externamente. Sempre que a mensagem READY aparecer no vídeo, o computador estará a nível de comando, aguardando ordens do usuário. Portanto, temos dois tipos de comandos: diretos e indiretos.

Comandos diretos são aqueles que não devem estar nas linhas do programa.

Comandos indiretos são aqueles que podem estar nas linhas de programa, desta forma serão classificados como instruções.

Os comandos são:

AUTO

A finalidade deste comando é numerar automaticamente as linhas de programa, permitindo ao usuário definir o início da numeração e o incremento usado.

Sintaxe

a) — AUTO
b) — AUTO *nl, ni*

onde:

nl = número inicial da linha
ni = Incremento na numeração da linha

Obs.:

Na forma definida pelo exemplo (a) o computador assumirá o valor 10 para *nl* e *ni*. Toda vez que o usuário teclar ENTER o computador incrementará o valor atual de 10. Para sair do comando AUTO, o usuário deve digitar a tecla BREAK

Exemplos:

```
READY
>AUTO ENTER
10 REM PROGRAMA EXEMPLO ENTER
20 REM ENTER
30 BREAK
>

READY
>AUTO 2,2 ENTER
2 ENTER
4 ENTER
6 ENTER
8 BREAK
READY
>
```

CLOAD

A finalidade deste comando é ler um programa especificado pelo nome (somente um caracter alfabético) no cassete e carregar para a memória. Assim que for encontrado o programa, aparecerão no canto superior direito do vídeo dois asteriscos indicando que o mesmo está sendo carregado. Quando terminar o carregamento aparecerá a palavra READY no vídeo seguida do sinal >.

Sintaxe:

a) CLOAD
b) CLOAD "Nome do Programa"
c) CLOAD? "Nome do Programa"

onde:

a) Carrega o primeiro programa encontrado na memória.
b) Carrega o programa especificado para a memória.
c) Este comando compara o programa armazenado na fita com o que está na memória, normalmente usado depois de um comando CSAVE, que grava o programa que está na memória principal de uma fita. Este comando verifica se o programa foi gravado corretamente.

Exemplos:

```
READY
> CLOAD  "A"   ENTER
READY
> CLOAD? "A"   ENTER
```

Obs.:

Antes de ler um programa em Basic, procure seguir as orientações abaixo especificadas:

1 — Verifique se a velocidade do programa gravado coincide com a velocidade de transmissão do computador com o cassete (500 ou 1500 bauds).
2 — Ajuste o volume no nível 5-7.
3 — Ajuste o tom no nível máximo 10 (se seu gravador tiver este tipo de controle).
4 — Aperte a tecla play do cassete.

Pronto! O computador carregará o programa da fita para a memória, podendo ser executado.

CSAVE

A finalidade deste comando é gravar o programa Basic que está na memória principal em uma fita.

Gravando um programa na fita, o usuário não precisará digitar o programa novamente toda vez que quiser executá-lo; basta carregar através da instrução CLOAD.

Quando terminar a gravação aparecerá no vídeo a palavra READY e o sinal

Sintaxe:

CSAVE "Nome do Programa"

onde:

o nome do programa só pode ser um caractere alfabético.

Exemplos:

```
READY
> CSAVE "A"   ENTER
READY
>CSAVE "P"   ENTER
```

Obs.:

Antes de gravar um programa Basic, certifique-se de que:

1 — A velocidade de transmissão é a desejada (500 ou 1500 bauds). Em geral, deve ser usada a velocidade de 500 bauds, quando se utilizar fitas do tipo "LOW NOISE", pois é mais seguro.

2 — Coloque a fita em posição conveniente, de modo que não fique sobre nenhum programa já gravado.

3 — Verifique o controle de tonalidade do gravador, se tiver, o qual deve estar no máximo agudo.

4 — Verifique o controle de volume que deve estar no intervalo 2-4.

5 — Pressione as teclas (REC/PLAY) do seu gravador cassete.

Pronto. Digite o comando ou simplesmente a tecla ENTER e o programa estará sendo gravado em uma fita, e com um bom resultado.

NEW

A finalidade deste comando é apagar todas as linhas do programa, todas as variáveis numéricas e todas as alfanuméricas. Portanto elimine o programa que está armazenado na memória principal.

Sintaxe:

NEW

Exemplo:

```
READY
>NEW    ENTER
```

DELETE

A finalidade deste comando é apagar linhas específicas do programa que está armazenado na memória principal.

Sintaxe:

a) DELETE *n*
b) DELETE *n* — *nf*
c) DELETE *nf*

onde:

n = Número da linha específica.
ni = Número da linha inicial.
nf = Número da linha final.

a) Apaga somente a linha *n*.
b) Apaga as linhas compreendidas entre *ni* e *nf*, inclusive.
c) Apaga desde a primeira linha até a linha *nf*, inclusive.

Exemplo:

```
READY
>DELETE 50    ENTER
```

LIST

A finalidade deste comando é pedir ao computador que mostre no vídeo qualquer linha ou grupo de linhas do programa armazenado na memória principal.

Sintaxe:

a) List
b) List *n*
c) List *ni* – *nf*
d) List *ni* –
e) List – *nf*

onde:

n = Número da linha;
ni = Número da linha inicial, e
nf = Número da linha final.

a) Mostra todas as linhas do programa que está armazenado na memória.
b) Mostra somente a linha *n*.
c) Mostra a partir da linha *ni* até *nf*.
d) Mostra a partir da linha *ni* até o final do programa.
e) Mostra do início do programa até a linha *nf*.

Exemplos:

```
READY
>LIST 3 ENTER
READY
>LIST 10-20 ENTER
READY
>LIST 50 ENTER
READY
>LIST 20 ENTER
READY
>LIST ENTER
```

Obs.:

Se o usuário desejar parar a listagem, deve pressionar as teclas SHIFT e @, simultaneamente. A listagem continuará ao ser pressionada qualquer outra tecla.

RUN

A finalidade deste comando é ordenar ao computador a iniciar a execução do programa armazenado na memória principal.

Sintaxe:

a) RUN
b) RUN *n*

onde:

a) Executa o programa a partir do menor número de linha para o maior, ou seja, a partir da primeira linha do programa.
b) Executa o programa a partir da linha especificada por *n*.

Exemplos:

```
READY
>RUN  ENTER
READY
>RUN 50  ENTER
```

Obs.:

Sempre que um comando RUN é executado são zeradas todas as variáveis numéricas, sendo que as variáveis alfanuméricas serão preenchidas com caracteres nulos.
Ocorrerá um erro se for usado um número de linha que não existe.

TRON

A finalidade deste comando é permitir que o usuário siga a execução do programa, linha por linha. Ao digitar este comando a função TRACE é ligada. Ao se executar um programa é mostrado no vídeo entre um par de símbolos, o número de linha que está sendo executada.

Sintaxe:

TRON

Exemplos:

```
10 REM PROGRAMA EXEMPLO
20 REM
30 X=1
40 IF X=5 THEN GOTO 80
50 PRINT X
60 X=X+1
70 GOTO 40
80 PRINT "FIM DO EXEMPLO"
90 END
```

Digite:

```
TRON [ENTER]
RUN [ENTER]
<10><20><30><40><50>1
<60><70><40><50>2
<60><70><40><50>3
<60><70><40><50>4
<60><70><40><80>FIM DO EXEMPLO
<90>
READY
>
```

TROFF

A finalidade deste comando é desativar a função TRACE descrita no comando anterior.

Sintaxe:

TROFF

Exemplo:

```
READY
>TROFF [ENTER]
```

Obs.:

A utilização do comando TRON é para verificar se uma certa linha ou sub-rotina está sendo executada. Para fazer uma ou várias pausas na execução, pressione simultaneamente as teclas [SHIFT] e [@]. Para continuar digite qualquer outra tecla.

CONT

A finalidade deste comando é a de continuar a execução do programa após ser interrompido pela tecla [BREAK] ou por uma instrução Basic dentro do programa.

Sintaxe:

CONT

Obs.:

Não é válido o comando CONT após alterações nas linhas do programa.

SYSTEM

A finalidade deste comando é colocar o computador no modo SYSTEM, que permite ao usuário carregar arquivos objetos (linguagem máquina).

Vide apêndice G.

EDIT

Este comando permite ao usuário corrigir (editar) linhas do programa.

Sintaxe:

EDIT *nl*

onde:

nl = número da linha.

Exemplo:

```
READY
>EDIT 50 ENTER
```

Obs.:

O usuário não precisa registrar a linha inteira do programa. Basta digitar o comando EDIT e o número da linha que precisa ser corrigida, e utilizar-se dos subcomandos descritos abaixo:

ENTER

Se o usuário pressionar a tecla ENTER estando o computador no modo de edição, todas as mudanças na linha serão gravadas na memória principal.

n ESPAÇO

Se o usuário pressionar a barra de espaço, o cursor move-se para a direita e mostra o caracter; se n for igual ao número real e inteiro entre 0 e 255 o cursor moverá n caracteres para a direita.

SHIFT ↑

Se o usuário pressionar as teclas SHIFT e ↑ simultaneamente o computador voltará ao nível normal de edição (se ele estiver nos subníveis X, I, H).

L

Se o usuário pressionar a tecla L, o computador listará no vídeo o resto da linha do programa e retornará ao início da linha.

X

Se o usuário pressionar a letra **X**, o cursor se moverá para a direita até o final da linha e entrará no subcomando de inserção.

A

Se o usuário pressionar a tecla **A**, o cursor voltará ao início da linha e cancelará todas as correções feitas na linha.

Q

Se o usuário pressionar a tecla **Q**, cancelará todas as modificações feitas na linha e retornará ao nível de comando.

H

Se o usuário pressionar a tecla **H**, apagará o restante da linha (à direita do cursor) e entrará no subcomando de inserção.

nD

Se o usuário pressionar a tecla **D** irá apagar um caractere, a partir da posição do cursor. Se n for igual a um número real inteiro entre 0 e 255, apagará tantos caracteres quanto for o valor de n.

nSc

O subcomando S procura a enésima ocorrência do caractere na linha, contando a partir da posição do cursor.

nKc

O subcomando K apaga todos os caracteres da linha a partir da posição do cursor. Se n for igual a um número real inteiro entre 0 e 255, todos os caracteres a partir da posição do cursor até a enésima ocorrência do C (caractere) serão apagados.

I

O subcomando I indica o início da inserção de caracteres, a partir da posição do cursor.

Variáveis/Constantes

A memória do nosso computador manipula dados através de endereços; então, para guardarmos temporariamente dados na memória, atribuiremos nomes a esses endereços, utilizando-se de letras e números.

A esses nomes chamaremos de variáveis ou constantes.

No nosso computador existem quatro tipos de variáveis ou constantes: Inteira, Simples-Precisão, Dupla Precisão e variáveis ou constantes Alfa-Numéricas. Os 3 primeiros tipos são usados para guardar valores numéricos e o último tipo é para guardar caracteres.

Variáveis ou constantes inteiras

Valores máximos: – 32768 a 32767. O símbolo para identificação é o % (por cento); esta variável ocupa 5 bytes na memória (uso dinâmico), sendo 2 bytes para o valor e 3 bytes para o nome da variável.

Exemplos:

```
A% = 80
BE% = 800
```

Variáveis ou constantes de Simples-Precisão

Valores máximos: 6 dígitos ou notação científica – 1,701411E + 38 a . . . . . + 1,701411E + 38 (inclusive) sendo que o símbolo para identificação é a ! (exclamação). Esta variável ocupa 7 bytes na memória (uso dinâmico), sendo 4 bytes para o valor e 3 bytes para o nome da variável.

Exemplos:

```
A!= -99
A2!=999999
```

Variáveis ou constantes de Dupla Precisão

Valores máximos: 16 dígitos ou notação científica – 1701411834544556E + 38 ou + 1,701411834544556E + 38 (inclusive), sendo que o símbolo para identificação é o # (numeral). Esta variável ocupa 11 bytes na memória (uso dinâmico), sendo 8 bytes para o valor e 3 bytes para o nome da variável.

Exemplos:

```
A# = 3.1415926535989
A2# = 94567.8701234
```

Variáveis ou constantes Alfanuméricas

Valores máximos: 0 a 255 caracteres, sendo que o símbolo para identificação é o $ (cifrão). Esta variável ocupa 6 bytes na memória (uso dinâmico), sendo 3 para o nome, 3 para os ponteiros e 1 para cada caractere.

Exemplos:

```
A$ = " O seu COMPUTADOR PESSOAL CP300"
A2$ = " Salarios recebidos"
```

Obs.:

Se o usuário colocar informações erradas para o conteúdo de uma variável qualquer, o computador não entenderá e emitirá no vídeo uma mensagem de erro.

Exemplo:

`A!="DADO ERRADO"` (somente números inteiros)
`A$=100` (não está entre aspas o número 100)

O computador pessoal CP 300 aceita nomes de variáveis com mais de dois caracteres, mas somente os dois primeiros serão usados para identificá-los. Os nomes devem sempre começar com uma letra (A a Z); as outras podem ser letras ou números.

Os nomes das variáveis não podem ser palavras reservadas nem ter sentido especial na linguagem Basic.

Uma lista destas palavras está no apêndice A.

Operações/Funções

Operadores aritiméticos

Os operadores aritméticos são utilizados quando desejamos fazer operações aritméticas.

O computador CP 300 usa os seguintes símbolos para operações aritméticas:

SHIFT ↑ caractere representado E para potenciação
* para multiplicação
/ para divisão
\ para divisão inteira
\+ para adição
— para subtração

O grau de prioridade dos operadores aritméticos é o mesmo da álgebra, podendo ser alterado, utilizando-se de níveis de parênteses.

Operadores relacionais

Os operadores relacionais (de relação) são utilizados quando queremos fazer relações lógicas com as variáveis.

O computador CP 300 utiliza as seguintes operações:

= Igualdade
<> Diferente de
> Maior que
< Menor que
>= Maior ou igual que
<= Menor ou igual que

Operadores lógicos

Os operadores lógicos são utilizados quando queremos fazer comparações lógicas com os operadores booleanos.

Os operadores booleanos mais utilizados na informática são: AND, OR, NOT.

Exemplo:

Programa com utilização dos operadores aritméticos, relacionais e lógicos:

```
10 REM PROGRAMA EXEMPLO
20 REM COM APLICACAO DOS OPERADORES
30 REM ARITMETICOS RELECIONAIS E LOGICOS
40 REM CALCULO AREA DE UM TRIANGULO QUALQUER
50 CLS
60 INPUT "DIGITE O COMPRIMENTO DO LADO A";A%
70 INPUT "DIGITE O COMPRIMENTO DO LADO B";B%
80 INPUT "DIGITE O COMPRIMENTO DO LADO C";C%
90 IF A%=0 OR B% = 0.OR C% = 0 OR
   A%>=B%+C% OR B%>=A%+C% OR C%>=A%+B%
   THEN? "NAO E' TRIANGULO":GOTO 140
100 S%= (A%+B%+C%)/2
110 T%= SQR (S%*(S%-A%)*(S%-B%)*(S%-C%))
120 PRINT "O SEMI PERIMETRO E'--->";S%
130 PRINT "A AREA E'------------->";T%
140 INPUT "QUER MAIS CALCULOS";R$
150 IF R$="SIM" THEN GOTO 50
160 IF R$="NAO" THEN END
170 GOTO 140
```

Obs.:

Em operações alfanuméricas, os operadores relacionais são utilizados para comparar conteúdo de variáveis alfanuméricas. No exemplo acima, a variável R$ só pode assumir as palavras SIM ou NÃO. Qualquer palavra diferente destas volta a fazer novamente a pergunta "Quer mais calculos".

Os operadores cumprem a seguinte hierarquia, sem aplicação de níveis de parenteses:

1º — Potenciação
2º — Negação
3º — Multiplicação e Divisão
4º — Soma e Subtração
5º — Os Operadores Relacionais
6º — Os Operadores Lógicos

Funções matemáticas

Além dos operadores aritméticos, a linguagem Basic-residente no CP 300 apresenta uma série de funções que permite a realização dos mais complexos cálculos aritméticos. Essas são as funções:

ABS(X)

Calcula o valor absoluto (módulo) de X.

ATN(X)

Calcula a função arcotangente (em radianos) do argumento. Para obter o valor em graus, multiplique ATN(X) por 57.29578

CDBL(X)

Transforma o valor do argumento em dupla precisão. O valor calculado contém 17 digitos, entretanto somente os digitos do argumento (X) são signficativos.

CINT(X)

Calcula o maior inteiro que não é maior que o argumento. O argumento precisa estar no intervalo – 32768 a + 32767

Exemplo:

```
CINT (5,6)=5
CINT (-5,6)=-6
```

COS(X)

Calcula o valor da função cosseno do argumento (em radianos). Para obter o cosseno do argumento (X) quando o argumento (X) estiver em graus, multiplique o argumento (X) por 0.0174533.

CSNG(X)

Transforma o valor do argumento em Simples Precisão.

EXP(X)

Calcula a função antilogaritmo natural do argumento (X).

FIX(X)

Calcula a representação truncada do argumento, desprezando todos os dígitos a direita do ponto decimal.

INT(X)

Calcula o valor inteiro do argumento usando o maior número inteiro, que não for maior que o argumento.

Exemplo:

```
INT(5,5)=5
INT(-5,5)=-6
```

LOG(X)

Calcula o logaritmo natural do argumento (X).

SGN(X)

Esta é a função "sinal" que calcula –1 se argumento (X) for negativo e 1 se o argumento (X) for maior que zero e 0 se o argumento (X) for igual a zero.

SIN(X)

Calcula a função seno do argumento (em radianos)

Para se obter o valor seno do argumento (X) em graus multiplique (X) por 0.0174533.

SQR(X)

Calcula a raiz quadrada do argumento (X).

TAN(X)

Calcula o valor da função tangente do argumento (X) (em radianos).

Para se obter o valor tangente do argumento (X) em graus multiplique (X) por 0.0174533.

Instruções de programa

Instruções de Entrada e Saída

PRINT

A finalidade desta instrução é permitir que o usuário "Imprima" no vídeo um item ou uma lista de itens que podem ser:

a) Constantes Numéricas
b) Constantes Strings (Mensagens entre aspas)
c) Variáveis Numéricas
d) Variáveis Strings

Os itens a serem impressos podem ser separados por vírgula ou ponto e vírgula.

Quando utilizamos a vírgula para separação de itens, o cursor avança automaticamente à próxima zona de impressão, antes de imprimir o item seguinte do programa.

Se for utilizado o ponto e vírgula, não existirá nenhum espaço entre os itens alfabéticos e nos itens numéricos será inserido dois espaços que se referem; um para separar os itens e um outro referente ao sinal (+ ou —)

Note que se o ítem numérico for positivo o sinal não será impresso.

Sintaxe:

PRINT *xx*

onde:

xx pode ser uma variável numérica ou variável alfanumérica, ou ainda um comentário STRING entre aspas.

Exemplo:

```
READY
>AUTO ENTER
10 N = 15+5
20 PRINT "15+5 E' IGUAL A ";N
30 END
40 BREAK

READY
>RUN ENTER
15+5 E' IGUAL A 20

READY
>NEW ENTER

READY
>AUTO ENTER
10 A$= "COMPUTADOR "
20 B$= "PESSOAL"
30 PRINT"USE O SEU ";A$;B$
40 END
50 BREAK

READY
>RUN
USE O SEU COMPUTADOR PESSOAL
```

Obs.:

Quando se usam vírgulas para separar ítens, são aceitas até 4 zonas por linha, onde cada zona de impressão tem no máximo 16 caracteres. Qualquer caractere que for impresso além deste aparecerá na próxima zona ou linha.

Exemplos:

```
READY
>AUTO ENTER
10 ?"ZONA 1","ZONA 2","ZONA 3","ZONA 4","ZONA 5"
20 END
30 BREAK

READY
>RUN
ZONA 1          ZONA 2          ZONA 3          ZONA 4
ZONA 5
```

Obs.:

Se duas vírgulas ou mais forem usadas juntas; cada uma ocupará um espaço em branco no vídeo de 16 caracteres.

Um ponto e vírgula no final de um ítem ou sentença anula o retorno do cursor, sendo assim o próximo PRINT inicia onde o último parou.

Se não colocarmos nenhuma pontuação após um item ou sentença com o comando PRINT, o cursor passará para o início da próxima linha de vídeo.

Exemplo com ponto e vírgula no final:

```
READY
>AUTO ENTER
10 PRINT "IMPRESSAO NA MESMA ";
20 PRINT "LINHA USANDO O ";
30 PRINT "PONTO E VIRGULA";
40 BREAK

READY
>RUN
IMPRESSAO NA MESMA LINHA USANDO O PONTO E VIRGULA
```

PRINT @

A finalidade desta instrução é imprimir os itens ou lista de itens no ponto de vídeo especificado. O símbolo @ precisa estar logo após a instrução PRINT e o outro ponto do vídeo especificado deve ser um número entre 0 a 1023. Veja mapa de vídeo no apêndice C.

Sintaxe:

PRINT @ *nn*.

onde:

nn é o ponto do vídeo específico.

Exemplo:

```
READY
>AUTO ENTER
10 CLS
20 PRINT @ 100,"POSICAO 100"
30 BREAK

READY
>RUN

                                POSICAO 100
```

Obs.:

Se for introduzido um PRINT @ na última linha do vídeo ocorrerá uma mudança automática de linha, fazendo com que tudo o que está no vídeo suba uma linha. Para evitar que isto aconteça introduza no final da sentença um ponto e vírgula.

PRINT TAB

A finalidade desta instrução é garantir a impressão de itens em qualquer posição dentro de uma linha. Se a posição TAB(*nn*) for maior que 63 a impressão será na próxima linha. Podem ser usados mais que um TAB numa mesma instrução PRINT.

Sintaxe:

PRINT TAB(*nn*)

onde:

nn = posição da linha em que se encontra o cursor.

Exemplos:

```
READY
>AUTO ENTER
10 PRINT TAB(10)POSICAO 10"; TAB(20),"POSICAO 20"
20 BREAK

READY
>RUN ENTER
POSICAO 10          POSICAO 20

READY
>NEW ENTER

READY
>AUTO ENTER
10 N=5
20 PRINT TAB(N)"POSICAO";N;TAB(N+30)"POSICAO";N+30
30 BREAK

READY
>RUN ENTER
POSICAO 5                                   POSICAO 35
```

PRINT USING

A finalidade desta instrução é imprimir os dados com um formato de impressão. Os dados podem ser numéricos ou alfanuméricos.

Sintaxe:

PRINT USING *string* ou *constante*

onde:

string ou constante é o formato de impressão ou limite de impressão.

Exemplos:

```
READY
>AUTO ENTER
10 INPUT"DIGITE UM NUMERO";N!
20 PRINT USING"###.##";N!
30 BREAK

READY
>RUN ENTER
DIGITE UM NUMERO 124
124.00 ENTER

READY
>RUN ENTER
DIGITE UM NUMERO 124.445
124.45 ENTER

READY
>NEW ENTER

READY
>AUTO ENTER
10 INPUT"DIGITE O NOME DO FUNCIONARIO";N$
20 PRINT USING "%     %";N$
30 BREAK

READY
>RUN ENTER
DIGITE O NOME DO FUNCIONARIO?LEANDRO DA SILVA ENTER
LEANDRO
```

Obs.:

O sinal % aparecerá automaticamente sempre que o número de dígitos for maior que o formato de impressão para variáveis numéricas e para variáveis alfanuméricas, serão somente truncadas (desprezadas).

Os seguintes caracteres podem ser usados para especificar o formato de impressão:

#

Este caractere especifica a posição de cada dígito do valor numérico. Se o formato numérico for maior que o número de dígitos em um valor numérico as posições à esquerda em desuso do ponto decimal serão espaços e as posições a direita em desuso serão preenchidas com zeros.

.

Este caractere é o ponto decimal podendo ser exposto em qualquer lugar do formato numérico, especificado pelo caractere #. O arredondamento será feito quando os dígitos da direita forem suprimidos.

,

Este caractere quando disposto em qualquer posição entre o primeiro dígito e o ponto decimal no formato numérico imprimirá no vídeo uma vírgula, à esquerda de todo terceiro dígito, estabelecendo uma posição adicional no formato.

**

Dois asteristicos colocados no início do formato, farão com que todas as posições em desuso sejam preenchidas com asteristicos. Os dois asteristicos estabelecerão duas posições a mais no formato.

$

Este caractere (cifrão) será impresso à frente do número.

$$

Dois cifrões colocados no início do campo atuarão como um cifrão flutuante, isto é, ocupará a primeira posição anterior ao número.

% %

Estes caracteres limitam o tamanho do string a ser impresso ao comprimento do string será somado 2 ao número de espaços entre os sinais porcentual.

↑↑↑↑ ou [[[[

Estes caracteres fazem com que o número seja impresso no formato exponencial (E ou D)

+

Este caractere colocado no início ou no fim de um formato, fará com que o computador coloque o caractere "+" se o número for positivo. Se o número for negativo o computador colocará o caractere "—" de acordo com a posição do caractere dentro do formato.

—

Este caractere colocado no início ou no fim de um formato, fará com que o computador coloque o caractere "—" se o número for negativo. Se o número for positivo o computador colocará o caractere "+" de acordo com a posição do caractere dentro do formato.

INPUT

Esta instrução tem por finalidade permitir que o usuário introduza dados pelo teclado. Esses dados podem ser numéricos ou alfanuméricos, dependendo somente da variável que irá receber esses dados. Se o usuário introduzir um valor string a uma variável numérica, aparecerá no vídeo: "?REDO"; permitindo ao usuário introduzir novamente o valor.

Sintaxe:

INPUT *xx*

onde:

xx pode ser uma variável numérica ou uma variável alfanumérica ou ainda um comentário string, seguido de uma variável qualquer ou lista de variáveis.

Exemplo:

```
READY
>AUTO ENTER
10 CLS
20 INPUT A$,B$,A,B
30 PRINT A$,B$,A,B
40 BREAK

READY
>RUN
?LEANDRO,CRISTINA,98,137 ENTER
LEANDRO        CRISTINA        98          137
```

Esta instrução permite a entrada de dois valores alfanuméricos e dois numéricos, sendo necessária a entrada dos quatro valores de uma só vez, separados por vírgula.

Uma outra maneira de entrada para os mesmos dados seria utilizando-se de 4 instruções INPUT.

Exemplo:

```
READY
>AUTO ENTER
10 CLS
20 INPUT A$
30 INPUT B$
40 INPUT A
50 INPUT B
60 PRINT A$,B$,A,B
70 BREAK
```

```
READY
>RUN ENTER
?LEANDRO ENTER
?CRISTINA ENTER
?98 ENTER
?137 ENTER
LEANDRO          CRISTINA          98          137
```

Outra maneira de entrada poderá ser feita emitindo-se ao vídeo uma mensagem constante toda vez que é executada a mesma instrução INPUT, pois ajuda a introdução de dados corretos para cada tipo de variável. Essa mensagem deve seguir a palavra INPUT, devendo estar entre aspas e em seguida um ponto e vírgula.

Exemplo:

```
READY
>AUTO ENTER
10 CLS
20 INPUT "DIGITE O SALARIO BASE ";SB#
30 PRINT USING"$########.##";SB#
40 BREAK

READY
>RUN ENTER
DIGITE O SALARIO BASE 100000.25
$100.000.25
```

DATA

A finalidade desta instrução é permitir que o usuário armazene dados dentro do programa, podendo ter acesso a esses dados através da instrução READ. Os dados serão acessados seqüencialmente do menor número de linha DATA para o maior número de linha de DATA. Os itens numa linha de DATA podem ser numéricos ou alfanuméricos. Os alfanuméricos devem estar entre aspas separados por vírgulas e os numéricos apenas separados por vírgulas (não se permitem expressões).

Sintaxe:

DATA *xx, yy,..*

onde:

xx e *yy* são os itens da instrução DATA.

Exemplos:

```
READY
>AUTO ENTER
5 CLS
10 DATA "JANEIRO","FEVEREIRO","MARCO","ABRIL","MAIO
","JUNHO"
20 DATA "JULHO","AGOSTO","SETEMBRO","OUTUBRO","NOVE
MBRO","DEZEMBRO"
30 CT%=CT%+1:IF CT%>12 THEN GOTO 60
40 READ MES$:PRINT MES$
50 GOTO 30
60 END
70 BREAK

READY
>RUN

JANEIRO
FEVEREIRO
MARCO
ABRIL
MAIO
JUNHO
JULHO
AGOSTO
SETEMBRO
OUTUBRO
NOVEMBRO
DEZEMBRO

READY
>
```

READ

A finalidade desta instrução é ler os valores de uma instrução DATA e transferir esses valores às variáveis especificadas. Os valores das instruções DATA serão lidos seqüencialmente pela instrução READ. Se houver uma tentativa de leitura de uma variável quando não houver mais itens a ser lidos ocorrerá um erro OD.

Sintaxe:

READ *xx*

onde:

xx é a variável que irá armazenar os itens lidos na instrução DATA.

Exemplo:

Vide exemplo da instrução DATA.

RESTORE

A finalidade desta instrução é permitir que sejam relidos os itens das instruções DATA, a partir da primeira instrução DATA.

Exemplo:

```
READY
>AUTO ENTER
10 CLS
20 READ A$
30 PRINT A$
40 RESTORE
50 READ B$
60 PRINT B$
70 END
80 DATA "LEANDRO","CAROLINA","LEONARDO"
90 BREAK
READY
>RUN ENTER
LEANDRO
LEANDRO
```

DEFINT (INTEIRAS)

A finalidade desta instrução é declarar o tipo de variável, cujos nomes iniciem por caracteres contido nesta instrução. Entretanto uma declaração de tipo, $,!, # durante o programa, pode modificar o tipo de variável declarado.

Operações aritméticas com variáveis inteiras são mais rápidas e econômicas na memória do que as realizadas com variáveis de simples e dupla precisão. Entretanto, uma variável definida como inteira poderá somente atuar com valores entre — 32768 a 32767, inclusive.

Sintaxe:

DEFINT *caractere* ou *lista de caracteres*

Exemplo:

```
READY
>AUTO ENTER
10 DEFINT A,I,N
```

Após a linha 10, todas as variáveis de nome que comecem com as letras A,I,N serão tratadas como variáveis inteiras. Exceto se houver declaração de outro tipo como,A1#,I$,N!.

DEFSNG (SIMPLES PRECISÃO)

A finalidade desta instrução é declarar o tipo de variável, cujos nomes iniciem por caracteres contido nesta instrução. Entretanto, uma declaração do tipo %,#,$ durante o programa pode modificar o tipo de variável declarada.

Sintaxe:

DEFSNG *caracter ou lista de caracteres*

Exemplos:

```
READY
>AUTO ENTER
10 DEFSNG A,I,N
 .
 .
 .
```

Após a linha 10 todas as variáveis de nome que comecem com as letras A,I,N serão tratadas como variáveis simples precisão. Exceto se houver declaração de outro tipo como A$,I#,N%.

DEFDBL (DUPLA PRECISÃO)

A finalidade desta instrução é declarar o tipo de variável cujos nomes iniciem caracteres contidos nesta instrução. Entretanto uma declaração do tipo $,%,! durante o programa pode modificar o tipo de variável declarada.

Sintaxe:

DEFDBL *caractere* ou *lista de caracteres*

Exemplo:

```
READY
>AUTO ENTER
10 DEFDBL I-N
 .
 .
 .
```

Após a linha 10, todas as variáveis de nome que comecem com as letras I,J,K,L,M,N, serão tratadas como variáveis Dupla Precisão, exceto se houver declaração de outro tipo como I$,K!,N%.

DEFSTR (ALFANUMÉRICA)

A finalidade desta instrução é declarar o tipo de variável cujos nomes iniciem por caracteres contidos nesta instrução. Entretanto uma declaração de tipo # ,!, % durante o programa pode modificar o tipo de variável declarada.

Sintaxe:

DEFSTR *caractere* ou *lista de caracteres*

Exemplo:

```
10 DEFSTR S,T,U
 .
 .
 .
```

Após a linha 10 todas as variáveis de nomes que comecem com as letras S,T,U, serão tratadas como variáveis alfanuméricas, exceto se houver declaração de outro tipo como $,%,T#,U!.

CLEAR

A finalidade desta instrução é zerar todas as variáveis numéricas, e reservar *n* bytes para as strings.

n pode ser um valor qualquer ou uma expressão que indicará quantos bytes serão reservados para as strings.

Sintaxe:

CLEAR *n*

Exemplo:

```
10 CLEAR 3000
 .
 .
 .
```

Reservará 3000 bytes para armazenagem de caracteres alfanuméricos.

DIM

A finalidade desta instrução é definir uma variável como Matriz ou Lista de Matrizes. Se não for especificado a dimensão de uma variável essa pode assumir 11 elementos diferentes. O número de dimensão só é limitado pelo tamanho da memória disponível. A instrução DIM pode estar em qualquer parte do programa, e os subscritos podem ser valores ou expressões.

Sintaxe:

DIM *n(xx, yy...)*

onde:

n é o nome da variável
xx, yy é o número de elementos de cada dimensão.

Exemplos:

```
30 DIM MES$(12)
```

Esta dimensão atribui a matriz MES$, uma dimensão de elementos subscritos de 0 a 12

```
100 DIM B(3,4),C(2,3,3)
```

Estas dimensões atribuem a matriz B com 20 elementos (4,5) e a matriz C com 48 elementos.

```
10 INPUT " NUMERO DE ELEMENTOS";N
20 DIM M(N+2,4)
```

O número de elementos da matriz M pode variar de acordo com N.
Para um redimensionamento de uma matriz deve-se usar um comando CLEAR com ou sem argumento.

LET

A finalidade desta instrução é atribuir um valor a uma variável para fazer programas compatíveis com outros sistemas. A instrução LET é opcional no CP 300.

Sintaxe:

LET *a* = *xy*

onde:

a é o nome da variável
xy é qualquer valor ou expressão

Exemplo:

```
READY
>AUTO ENTER
10 LET A = 4.5
20 B = 25
30 LET M$= "PROGRAMA TESTE"
40 C%=C% +1
50 PRINT A,B,M$,C%
60 END

READY
>RUN ENTER
4.5          25          PROGRAMA TESTE
```

END

A finalidade desta instrução é definir o ponto de encerramento do programa, podendo estar em qualquer lugar do programa diferente do seu fim físico.

Exemplo:

Vide programa exemplo da instrução DATA.

STOP

A finalidade desta instrução é interromper o programa permitindo que o usuário examine ou modifique valores de variáveis. Esta é uma instrução auxiliar somente usada na preparação de programas.

Exemplo:

```
READY
>AUTO ENTER
10 INPUT A,B
20 C= A*B
30 STOP
40 PRINT A,B,C
50 END
60 BREAK
READY
>RUN ENTER
?2,4
Break na 30
READY
CONT ENTER
2              4              8
```

A Instrução STOP na linha 30 permite ao usuário conferir o valor de C antes da execução da linha 40.

Obs.:

Toda vez que o computador executar uma instrução STOP emitirá no vídeo a mensagem BREAK na *nn:*

onde:

nn é o número da linha que contém a instrução STOP

GOTO

A finalidade desta instrução é transferir o controle do programa para um número de linha especificado. Se esta instrução for usada sozinha, diremos que houve um salto incondicional. Entretanto se for usada junto com uma instrução de teste diremos que houve um salto condicional.

Exemplo:

Vide programa exemplo da instrução DATA

GOSUB

A finalidade desta instrução é transferir o controle do programa para a linha especificada como início da sub-rotina. O computador ao encontrar a instrução RETURN voltará para a próxima instrução imediatamente seguinte à GOSUB. Assim como a instrução GOTO, a instrução GOSUB pode também ser precedida de uma instrução de teste.

Exemplo:

```
10 FOR I=1 TO 10
20 GOSUB 100
30 NEXT I
100 PRINT "SUB-ROTINA",I
110 RETURN
```

RETURN

A finalidade desta instrução é finalizar uma sub-rotina e retornar o controle para a instrução seguinte à GOSUB. Ocorrerá um erro se for encontrado um RETURN sem o correspondente GOSUB.

ON ... GOTO

A finalidade desta instrução é transferir o controle do programa para várias linhas especificadas de acordo com o valor de um variável especificada no programa.

Sintaxe:

ON v GOTO *xx,yy,zz,...*

onde:

v é uma variável ou expressão onde o valor desta precisa estar entre 0 e 255.
xx, *yy* e *zz* números de linhas especificas do programa.

Exemplo:

```
10 CLS
20 PRINT"DIGITE UM NUMERO E <RETURN>";
30 INPUT N
40 ON N GOTO 100,200,300
50 PRINT"UM NUMERO ENTRE 1 E 3"
60 GOTO 30
100 PRINT"VOCE DIGITOU UM":END
200 PRINT"VOCE DIGITOU DOIS":END
300 PRINT"VOCE DIGITOU TRES":END
```

Obs.:

Quando a instrução ON—GOTO é executada e o desvio depende do resultado da expressão, então o computador calculará o valor da expressão em primeiro lugar sendo o resultado um número inteiro. Em seguida atribui valor a v, e conta *n* elementos na lista de números de linha, saltando para a linha especificada pelo elemento. Se existirem menos elementos na lista que o valor de v ou se o valor de v for igual ou zero, o computador executará a instrução da linha seguinte. A lista de números de linha pode ter qualquer quantidade de elementos:

ON ... GOSUB

A finalidade desta instrução é igual a instrução ON ... GOTO, exceto que os saltos são feitos a sub-rotinas onde se faz necessário no final de cada sub-rotina a instrução RETURN.

FOR ... TO ... STEP
...
...
...
NEXT ...

A finalidade destas instruções é fazer com que uma seqüência de instruções do programa seja executada um determinado número de vezes.

Sintaxe:

FOR v = *xx* TO *yy* STEP *zz*
NEXT

onde:

v = Nome da variável
xx = Valor inicial
yy = Valor final
zz = Incremento na variável
NEXT = contador

$(xx-yy+1)$ especifica o número de vezes a ser executada as instruções que estão entre FOR e NEXT.

Na instrução FOR o valor inicial, valor final e o incremento podem ser constantes ou variáveis ou expressões. A primeira vez que a instrução FOR é executada, estes três dados serão guardados e será ajustado através da instrução NEXT pelo valor especificado no incremento STEP. Se o incremento STEP não for usado o valor do incremento será 1, mas se o incremento STEP for negativo então o contador será decrementado.

Enquanto o contador não for maior ou menor que o valor final, o controle voltará para a instrução seguinte ao FOR.

Exemplo:

```
10 CLS
20 INPUT"QUANTAS REPETICOES";N
30 V$="VEZ"
40 FOR I=1 TO N
50 PRINT"EXECUTANDO ";I;V$
60 V$="VEZES"
70 NEXT I
```

Obs.:

Cada instrução NEXT especifica uma variável apropriada não sendo necessária a variável quando estiver com um LOOP simples, isto é: somente uma instrução FOR-NEXT de cada vez. Mas, se utilizar as instruções FOR-NEXT "ENTRELAÇADAS", se faz necessário que os nomes das variáveis e ainda as variáveis dos LOOPS mais internos devam vir primeiro.

ERROR

A finalidade desta instrução é para testar uma rotina ON ERROR GOTO.

Quando a instrução ERROR *código* for encontrada, o computador procederá como se realmente houvesse ocorrido um erro.

Sintaxe:

ERROR *x*

onde:

x é o código de erro

Exemplo:

```
30 IF A$<>"00"OR A$<>"01" OR A$<>"02"OR
      A$<>"03"OR A$<>"04" OR A$<>"05"OR
      A$<>"06"OR A$<>"07" OR A$<>"08"OR
      A$<>"09" THEN ERROR 13

?TM erro na 30
```

Neste exemplo se A$ tiver com conteúdo diferentes desses "00" a "09" o computador emitirá no vídeo a mensagem?TM ERRO NA 30

Os códigos de erros estão no apêndice B.

ON ERROR GOTO

A finalidade desta instrução é permitir que o usuário construa sua própria rotina de teste de erro para reconhecer o erro e continuar a execução normal do programa sem interrupção. Normalmente utilizada quando um tipo de erro pode ocorrer, e este tipo de erro foi previsto pelo programador.

Sintaxe:

ON ERROR GOTO *nn*

onde:

nn é o número da linha de desvio

Exemplo:

```
10 ON ERROR GOTO 100
20 A = 1/B
 .
 .
90 END
100 PRINT"NAO EXISTE DIVISAO POR ZERO":RESUME 90
```

Obs.:

Se *nn* for igual a zero é desativada a instrução ON ERROR GOTO e o Basic manipulará normalmente o erro.

A rotina de teste de erro deve terminar com uma instrução RESUME.

RESUME

A finalidade desta instrução é finalizar uma rotina de teste de erro especificando o número da linha a qual deve retornar o processamento.

Sintaxe:

RESUME *nn*

onde:

nn é o número da linha a ser resumido.

Exemplo:

```
10 ON ERROR GOTO 100
20 A = 1/B
90 END
100 PRINT "NAO EXISTE DIVISAO POR ZERO"
110 RESUME 90
```

Obs.:

Uma instrução RESUME sem numeração de linha ou RESUME 0, faz o computador retornar à instrução em que o erro ocorreu.

REM

A finalidade desta instrução é permitir que o usuário insira comentários em seu programa para melhor documentação.

Uma instrução REM em qualquer ponto da linha instrui ao computador a ignorar o resto da linha.

Exemplo:

```
10 REM PROGRAMA DE FOLHA DE PAGAMENTO:
20 REM     O OBJETIVO DESSE PROGRAMA E'
30 REM     EMITIR RELATORIOS DOS FUNCIONARIOS QUE
40 REM     GANHAM MENOS DE VINTE SALARIOS MINIMOS
```

Obs.:

Qualquer instrução após uma instrução REM numa mesma linha não será executada.
O apóstrofo (SHIFT 7) pode ser utilizado como abreviatura da instrução REM

IF ... THEN ... ELSE ...

A finalidade destas instruções é fazer com que o computador execute um teste lógico ou relacional da expressão. Se a expressão for verdadeira passará o controle do programa para instrução seguinte ao THEN. Se a expressão for falsa passará o controle do programa para a instrução seguinte ao ELSE (se existir a instrução ELSE) ou para a próxima linha do programa.

Sintaxe:

IF Expressão
THEN Instrução Ação
ELSE Instrução contrária à Instrução Ação

Exemplo:

```
10 IF A$= "SIM" THEN GOTO 60 ELSE GOTO 90
```

Neste exemplo, se o conteúdo de A$ for diferente de "SIM" o controle do programa passará para a linha 90.

As instruções IF — THEN — ELSE podem ser entrelaçadas, onde os IF e ELSE precisam ser pares correspondentes.

Exemplo:

```
10 INPUT "INTRODUZA TRES LETRAS";A$,B$,C$
20 PRINT "A MAIOR LETRA E':";
30 IF A$<B$ OR A$<C$ THEN IF B$<C$ THEN PRINT C$
   ELSE PRINT B$ ELSE PRINT A$
READY
>RUN
INTRODUZA TRES LETRAS? X,Y,Z  ENTER
A MAIOR LETRA E':Z
```

Esse programa exemplo aceita letras e números, portanto a comparação entre os valores das variáveis segue o valor de cada letra na tabela ASCII. Se a comparação for de igualdade será feita uma comparação caractere a caractere incluindo espaços em branco.

Instruções de programa que auxiliam operações com dados alfanuméricos.

ASC

A finalidade desta instrução é fornecer o código ASCII do primeiro byte da variável string.

Sintaxe:

ASC (*xx*)

onde:

xx pode ser uma variável ou um caractere.

Exemplos:

```
10 A$ = "10"
20 PRINT ASC (A$)
READY
>RUN
49
```

Nesse exemplo o valor 49 que corresponde ao número 1 na tabela ASCII.

```
READY
>PRINT ASC (A)
65
```

Nesse exemplo imprimiu o valor 65 que corresponde à letra A.

Obs.:

Se o argumento *xx* for um caractere, esse deve estar entre aspas dentro do parenteses.
Ocorrerá um erro se o argumento *xx* não for uma variável string ou o caractere não estiver entre aspas ou ainda se o argumento *xx* for nulo.

CHR$

A finalidade desta instrução é transformar o código (decimal) no caractere ou símbolo correspondente na tabela. Os argumentos podem ser variáveis ou qualquer número entre 0 e 255.

Sintaxe:

CHR$ (*xx*)

onde:

xx pode ser qualquer argumento.

Exemplo:

```
PRINT CHR$(89)
```

Esse exemplo imprimirá a letra Y.

LEFT$

A finalidade dessa instrução é separar ou extrair conteúdo de uma variável ou constante, a partir do 1º byte a esquerda.

Sintaxe:

LEFT$ (*nn, nc*)

onde:

nn é o nome da variável e *nc* é a quantidade de *bytes* a ser separado ou extraído.

Exemplo:

```
10 A$= "CP 300 MICROCOMPUTADOR PROLOGICA"
20 B$ = LEFT$ (A$,6)
30 PRINT B$
READY
>RUN
CP 300
```

RIGHT$

A finalidade dessa instrução é separar ou extrair conteúdo de uma variável ou constante, a partir do 1º byte a direita.

Sintaxe:

RIGHT$ (*nn, nc*)

onde:

nn é o nome da variável, *nc* é a quantidade de bytes a ser separado ou extraído.

Exemplo:

```
10 A$ "CP 300 MICROCOMPUTADOR PROLOGICA"
20 B$ RIGHT$ (A$,9)
30 PRINT B$
READY
>RUN

PROLOGICA
```

MID$

A finalidade dessa instrução é separar ou extrair conteúdo de uma variável ou constante, a partir de um ponto específico.

Sintaxe:

MID$ (*nn,pc,nc*)

onde:

nn é o nome da variável ou constante.
pc é a posição inicial de separação ou extração
nc é a quantidade de caracter a ser extraído ou separado.

Exemplo:

```
10 A$ = "CP 300 MICROCOMPUTADOR PROLOGICA"
20 B$ = MID$ (A$,8,15)
30 PRINT B$
READY
>RUN

MICROCOMPUTADOR
```

LEN

A finalidade desta instrução é retornar com o número de caracteres contidos numa variável string.

Sintaxe:

LEN (*nn*)

onde:

nn é o nome da variável.

Exemplo:

```
10 A$ = "CP 300 MICROCOMPUTADOR PROLOGICA"
20 PRINT "O NUMERO DE CARACTERES DA VARIAVEL A$ E':"
30 PRINT LEN (A$)

READY
>RUN
O NUMERO DE CARACTERES DA VARIAVEL A$ E':
32
```

STR$

A finalidade desta instrução é converter uma variável ou constante em caracteres.

Sintaxe:

STR$ (*xx*)

onde:

xx é a variável ou constante a ser convertido. A expressão ou variável deve estar entre parenteses.

Exemplo:

```
10 A=86.35
20 B=-86.35
30 PRINT STR$(A);STR$(B)
READY
>RUN
86.35-86.35
```

STRING$

A finalidade desta instrução é fornecer um string composto de *n* caracteres.

Sintaxe:

STRING$ (*n,c*)

onde:

n é quantidade de caracteres, *c* é o código ASCII do caractere especificado ou o caractere entre aspas.

Exemplos:

```
10 PRINT STRING$ (30,42)
READY
>RUN
******************************
```

```
10 PRINT STRING$ (10,"*")
READY
> RUN
**********
```

VAL

A finalidade desta instrução é converter os caracteres numéricos de uma variável em valores numéricos.

Sintaxe:

VAL (*xx*)

onde:

xx é o nome da variável string.

Exemplo:

```
10 A$ = "56"
20 B$ = "32"
30  C = VAL (A$+B$)
40 PRINT C
50 END

READY
> RUN
5632
```

Obs.:

Se o string não for numérico ou for nulo o valor resultará em erro.

INKEY$

A finalidade desta instrução é possibilitar a introdução de dados pelo teclado (sómente um caractere), sem a utilização da tecla ENTER. Se nenhuma tecla for pressionada um caractere nulo é considerado passando para a próxima instrução de programa. Portanto, essa instrução é geralmente usada dentro de um *loop* para o usuário digitar um caractere chave, para continuidade do programa.

```
100 PRINT "AJUSTE O VOLUME DO GRAVADOR ENTRE 5 E 7"
110 PRINT "EM SEGUIDA PRESSIONE P PARA CONTINUAR"
120 A$ = INKEY$
130 IF A$ <> "P" THEN GOTO 130
140 PRINT #-1, N$
150 PRINT #-1, E$
160 PRINT #-1, SB#
170 PRINT "TEM MAIS CADASTROS <S/N>";
180 INPUT RESP$
190 IF RESP$ = "S" THEN GOTO 50
200 IF RESP$ = "N" THEN END ELSE GOTO 180.
```

Processamento com variáveis dimensionadas (Matrizes)

Uma variável dimensionada é simplesmente uma variável que pode armazenar vários elementos.

Os elementos dessa variável serão identificados pelo nome da variável e acessado através de um índice cujo valor deve ser um número ou variável inteira.

Essas variáveis podem ser numéricas ou alfanuméricas. Entretanto os tipos de dados de uma variável dimensionada devem ser coerentes com as mesmas.

Para melhor aplicação com variáveis dimensionadas elaboramos vários programas-complexos. Portanto pedimos aos usuários que tentem compreender todos os programas.

Exemplos:

```
5 'EMISSAO DE DATA POR EXTENSO
10 CLEAR 100 REM REVERVA DE BYTES NA MEMORIA
20 DIM MES$ (12) REM DIMENSIONAMENTO DE VARIAVEL
30 FOR T% = 1 TO 12 REM LIMITE DO CONTADOR P/ LOOP
40 READ MES$ (T%) REM LEITURA DE INSTRUCOES DATA
50 NEXT T% REM CONTADOR
60 CLS
70 PRINT "EMISSAO DE DATA POR EXTENSO"
80 PRINT: PRINT: PRINT:
90 INPUT"DIGITE A DATA NA FORMA <DD/MM/AA>";DT$
100 DD$ = LEFT$ (DT$,2):MM% = VAL (MID$ (DT$,4,2)):
AA$ = RIGHT$ (DT$,2);
130 PRINT"SAO PAULO";DD$;"DE ";MES$(MM%);" DE 19";
AA$;
140 DATA "JANEIRO","FEVEREIRO","MARCO";"ABRIL","MAI
O","JUNHO"
150 DATA "JULHO","AGOSTO","SETEMBRO","OUTUBRO","NOV
EMBRO"
160 DATA "DEZEMBRO"
170 END
```

Na execução desse programa a variável MES$ estará carregada com todos os meses do ano, da seguinte maneira:

```
ESPACO VAZIO  (0)
JANEIRO        (1)
FEVERIERO      (2)
MARCO          (3)
ABRIL          (4)
MAIO           (5)
JUNHO          (6)
JULHO          (7)
AGOSTO         (8)
SETEMBRO       (9)
OUTUBRO       (10)
NOVEMBRO      (11)
DEZEMBRO      (12)
```

Observem que o indicador do mês a ser impresso é a variável MM%. Então, dependendo do valor de MM% estaríamos acessando o mês de referência que está na variável MES$.

```
5 REM O OBJETIVO DESSE PROGRAMA E DE LOCALIZAR.
10 REM UM ALUNO EM UMA SALA DE AULA QUE CONTEM:
15 '6 FILEIRAS DE CARTEIRAS-6 CARTEIRAS POR FILEIRA
20 CLEAR 2000.
25 DIM SALA$(6,6)'DIMENSIONAMENTO DA VARIAVEL SALA
30 FOR F = 1 TO 6
35 FOR C = 1 TO 6
40 INPUT "DIGITE O NOME DOS ALUNOS"; SALA$ (F,C)
45 NEXT C, F
50 CLS
55 INPUT "DIGITE O NU'MERO DA FILA"; F%
60 INPUT "DIGITE O NU'MERO DA CARTEIRA" C%
70 PRINT "O ALUNO E O"; SALA$ (F%, C%)
80 PRINT @ 896, "TEM MAIS CONSULTA <S/N>"; RESP$
90 IF RESP$="S" THEN GOTO 50
100 IF RESP$="N" THEN END ELSE GOTO 80.
```

Mapa da sala de aula

```
1,1   2,1   3,1   4,1   5,1   6,1
1,2   2,2   3,2   4,2   5,2   6,2
1,3   2,3   3,3   4,3   5,3   6,3
1,4   2,4   3,4   4,4   5,4   6,4
1,5   2,5   3,5   4,5   5,5   6,5
1,6   2,6   3,6   4,6   5,6   6,6
```

Observem que os indicadores das posições são as variáveis F% e C%.

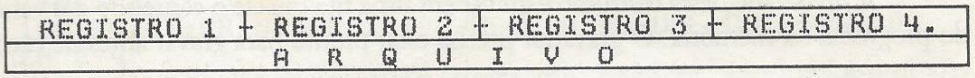
F% = FILEIRA
C% = CARTEIRA.

Arquivo seqüencial para fita

Arquivo seqüencial é o método convencional de organização, onde os registros estão ordenados em seqüência com base em um campo de dado comum em todos os registros.

O processo de leitura ou gravação é realizado seqüencialmente (um após o outro).

Exemplo:



Para ler ou gravar o registro 4, teriamos que realizar a operação a partir do registro 1 até o registro 4.

As instruções utilizadas em programas que trabalham com esse tipo de organização de dados em BASIC são:

INPUT # —1
PRINT # —1

E todas as instruções da linguagem.

INPUT # —1

A finalidade desta instrução é de ler os dados que estiverem gravados na fita cassete e atribuir esses ás variáveis do programa.

Sintaxe:

INPUT # —1, *xx*

onde:

xx é o nome da variável que receberá campo do registro gravado e lido na fita.

Exemplo:

```
 90"AJUSTE O VOLUME DO GRAVADOR EM 5-7 DEPOIS DIG P"
100 A$ = INKEY$
110 IF A$ <> "P" THEN GOTO 100
120 INPUT #-1, N$
130 INPUT #-1, E$
140 INPUT #-1, SB#
```

Esse programa lê os dados gravados na fita e o primeiro campo lido carregará para a variável N$ o segundo para a variável E$ e o último campo do registro á variável SB#

Quando o computador executa esta instrução o gravador é ligado automaticamente e desligado quando terminar a leitura dos dados.

PRINT#—1

A finalidade desta instrução é gravar em uma fita cassete o conteúdo das variáveis do programa na memória. O gravador precisa estar pronto para gravar antes de executar esta instrução.

Sintaxe:

PRINT#—1, *xx*

onde:

xx é a variável que contém o dado a ser gravado.

Exemplo:

```
20 PRINT"AJUSTE O VOLUME EM 2-3 APOS DIG. P"
30 A$ = INKEY$
40 IF A$ <> "P" THEN GOTO 30
50 PRINT #-1, N$
60 PRINT #-1, E$
70 PRINT #-1, SB#
```

Esse programa grava os dados das variáveis N$, E$, SB# na fita, permitindo uma leitura posterior através da instrução INPUT#—1.

Obs.:

O número total de caracteres contidos nas variáveis não pode exceder 255.

Exemplo:

```
5 REM PROGRAMA P/ LEITURA E GRAVACAO EM FITA K7
10 REM PROGRAMA TESTE
20 CLS
30 PRINT TAB(20)"OPCOES DO PROGRAMA"
40 PRINT:PRINT:PRINTTAB(10)"(1) - CADASTRAMENTO"
50 PRINTTAB(10)"(2) - CONSULTA NO VIDEO"
60 PRINT:PRINTTAB(10)"DIGITE SUA OPCAO";INPUT O%
70 IF O%<1 OR O%>2 THEN PRINTTAB(10)"OPCAO INVALIDA
":FOR I=1 TO 1000:NEXT I:GOTO 20
80 IF O%=1 GOTO 100
90 IF O%=2 GOTO 250
100 CLS
110 PRINTTAB(20)"** CADASTRAMENTO **"
120 PRINT:PRINTTAB(10)"DIGITE O NOME";:INPUT N$
130 PRINTTAB(10)"DIGITE O ENDERECO";:INPUT EN$
140 PRINTTAB(10)"DIGITE O SAL. BRUTO";:INPUT SB#
150 PRINT:PRINTTAB(10)"DADOS OK <S/N>";:INPUT R$
160 IF R$="S" GOTO 180
170 IF R$="N" THEN GOTO 100 ELSE GOTO 150
180 PRINT:PRINTTAB(10)"AJUSTE O VOLUME DO GRAVADOR
EM 2-4"
190 PRINTTAB(10)"QUANDO ESTIVER PRONTO TECLE 'P'"
200 A$=INKEY$:IF A$<>"P" THEN GOTO 200
210 PRINT #-1,N$
220 PRINT #-1,EN$
230 PRINT #-1,SB#
240 GOTO 400
250 CLS
260 PRINTTAB(20)"** CONSULTA **"
270 PRINT:PRINTTAB(10)"AJUSTE O VOLUME DO GRAVADOR
EM 5-7"
280 PRINTTAB(10)"QUANDO ESTIVER PRONTO TECLE 'P'"
290 B$=INKEY$:IF B$<>"P" THEN GOTO 290
300 INPUT #-1,N$
310 INPUT #-1,EN$
320 INPUT #-1,SB#
330 IAP#=SB#*.08
340 SL#=SB#-IAP#
350 PRINT:PRINTTAB(10)"NOME.....:";TAB(21)N$
360 PRINTTAB(10)"ENDERECO.:";TAB(21)EN$
370 PRINTTAB(10)"SAL BRUTO:";TAB(21)SB#
380 PRINTTAB(10)"IAPAS....:";TAB(21)IAP#
390 PRINTTAB(10)"SAL. LIQ.:";TAB(21)SL#
400 INPUT"TECLE 'C' PARA CONTINUAR";C$
410 IF C$="C" THEN GOTO 20 ELSE END
```

Funções de gráficos

SET

A finalidade desta instrução é imprimir no vídeo um ponto gráfico em uma posição determinada.

Quando o programa estiver utilizando essa função o vídeo estará dividido em 6144 pontos, sendo:
128 pontos horizontais e 48 pontos verticais.

Sintaxe:

SET (*x*,*y*)

onde:

x é a coordenada da coluna que varia entre 0 e 127, começando da esquerda para a direita.
y é a coordenada da linha que varia entre 0 e 47.

Exemplo:

```
10 CLS
20 FOR I = 1 TO 127
30 SET (I,1)
40 NEXT I
50 FOR J = 1 TO 41
60 SET (127, J)
70 NEXT J
80 FOR T = 127 TO 1 STEP - 1
90 SET (T,42)
100 NEXT T
110 FOR W = 42 TO 1 STEP - 1
120 SET (1,W)
130 NEXT W
140 END
```

RESET

A finalidade desta instrução é apagar no vídeo um ponto gráfico em uma posição determinada.

Esta instrução tem os mesmos parametros da instrução SET.

Como exemplo podemos utilizar o programa-exemplo da instrução SET, fazendo a seguinte modificação no mesmo:
Nas linhas, 30, 60, 90 e 120 — troque a instrução SET por RESET.

POINT

A finalidade desta instrução é verificar se um ponto do vídeo está ativado ou não. Se o ponto estiver ativado, a instrução POINT associa o valor (—1); se estiver desativado associa o valor (0) zero, e atribui a uma variável qualquer do programa. Esta instrução tem os mesmos parametros da instrução SET.

Sintaxe:

POINT (*x,y*)

Exemplo:

```
B = POINT (24,64)
```

Se o ponto (24,64) estiver aceso, B terá o valor (—1) e se estiver apagado B terá o valor (0) zero.

Instruções especiais

PEEK

A finalidade desta instrução é ler um valor de 8 bits armazenado na memória principal, no endereço especificado, e atribuir a uma variável de programa.

Sintaxe:

PEEK (*xx*)

onde:

xx é o endereço da memória principal.

Exemplo:

```
10 FOR T = 1 TO 10
20 FOR A = 20 TO 40
30 PRINT@64*10+A," ";:PRINTCHR$(PEEK(16419));
:PRINT@64*10+A-1,";"
40 PRINTCHR$(PEEK(16419));:PRINT@64*10+A,".";:NEXTA
50 FOR A=40 TO 20 STEP-1:PRINT@64*10+A,"";:PRINT
CHR$ (PEEK (16419));
60 PRINT@64*10+A-1,"";:PRINTCHR$(PEEK(16419));
:PRINT@64*10+A," "
70 NEXT A,T
80 END
```

POKE

A finalidade desta instrução é colocar um valor de 8 bits na memória principal, no endereço especificado.

Sintaxe:

POKE (*x, y*)

onde:

x é o endereço de memória
y é o valor a ser colocado.
O valor precisa estar entre 0 e 255.

Exemplo:

```
 5 CLS
10 X = 250
20 POKE 15000, X
30 Y = PEEK (15000)
40 PRINT Y
50 END
```

MEM

A finalidade desta instrução é fornecer o número de bytes livres na memória.

Sintaxe:

MEM

Exemplo:

```
READY
> PRINT MEM
48082
```

INP

A finalidade desta instrução é introduzir e fornecer o valor de 8 bits lido na porta especificada. O CP 300 pode controlar até 256 portas sendo essas numeradas de 0 a 255.

Sintaxe:

INP *(xx)*

onde:

xx é a porta de controle de periféricos.

OUT

A finalidade desta instrução é colocar um valor de 8 bits na porta especificada.

Sintaxe:

OUT *xx,yy*

onde:

xx é o número da porta
yy é o valor.

Os dois argumentos *xx*, *yy* precisam estar nos intervalos de 0 a 255.

Principais endereços de memória

Este capítulo é destinado aos programadores Basic que estejam familiarizados com aritmética binária e hexadecimal e com noções de hardware.

As informações apresentadas têm como propósito permitir ao programador controlar todas as vantagens que o computador CP 300 oferece.

Endereços e seus conteúdos

16412

Este endereço de memória controla a intermitência do cursor. Se o conteúdo desse endereço for nulo, o cursor é intermitente; caso contrário, não haverá intermitência.

Exemplo:

```
POKE 16412,1 ENTER
```

Essa linha fixa o cursor no vídeo.

```
POKE 16412,0 ENTER
```

Essa linha retorna à condição inicial (intermitente).

16419

Este endereço de memória contém o código ASCII do caractere correspondente ao cursor. Inicialmente seu conteúdo é 176. A mudança desse valor resulta numa troca de caractere que identifica o cursor.

Exemplo:

```
POKE 16419,63 ENTER
```

Essa linha troca o cursor normal por um ponto de interrogação.

```
POKE 16419,176 ENTER
```

Essa linha retorna o caractere normal que identifica o cursor.

16913

Este endereço de memória controla a velocidade de transferência de bytes do ou para o gravador. Se contiver um valor diferente de zero será usada a velocidade "alta" (1500 bauds); se for zero, a velocidade será a "baixa" (500 bauds).

16916

Este endereço de memória controla o número de linhas que você pode proteger, no alto do vídeo. O número máximo de linhas para o cabeçalho no vídeo é sete, o que deixa as nove linhas inferiores dentro do campo de deslocamento da tela.

Exemplo:

```
POKE 16916,4 ENTER
```

Essa linha protege as quatro primeiras linhas da tela para que seja usado um cabeçalho em listagem de longas tabelas.

```
POKE 16916,0 ENTER
```

Essa linha retorna para a condição normal (sem proteção).

Obs.:

Os valores maiores que sete nesse endereço são divididos por oito e apenas o resto da divisão é considerado.
Esse endereço só controla as linhas do topo do vídeo.

Exemplo:

```
POKE 16916,26 ENTER
```

Ao executar essa linha, o computador divide o número (26) por oito, o que resulta em três e sobram dois. Como o computador só considera o resto da divisão, serão protegidas apenas duas linhas, no topo do vídeo.

O programa a seguir ilustra a característica de proteção:

```
10 CLS:POKE 16916,3     'PROTEGE AS 3 LINHAS DO TOPO
20 PRINT"AS 3 LINHAS DE CIMA NAO SERAO DESLOCADAS"
30 PRINT"AS DEMAIS SERAO"
40 PRINTSTRING$(30,"-")
50 FOR I=1 TO 100
60 PRINT"ESTAS LINHAS NAO ESTAO EM AREA PROTEGIDA."
70 PRINT"PORTANTO, SERAO DESLOCADAS"
80 NEXT I
90 POKE 16916,0          'REMOVE A PROTECAO
```

15360 a 16383

Endereço de memória do vídeo (posição de vídeo).

Exemplo:

```
10 CLS
20 FOR I=16383 TO 15360 STEP -1
30 POKE I,166
40 NEXT I
```

16919 a 16924

Endereço de memória onde está o relógio, que está quase sempre em funcionamento, parando apenas durante as operações de acesso ao gravador ou ao disquete.

Esse relógio possui lógica para meses de 28, 30 e 31 dias, mas não reconhece anos bissextos.

Ao ligar o seu CP 300 esses endereços são preenchidos por zeros.

Dado	*Abreviatura*	*Faixa*	*Posição na memória*
Mês	MM	01-12	16924
Dia	DD	01-31	16923
Ano	AA	00-99	16922
Hora	HH	00-23	16921
Minuto	MN	00-59	16920
Segundo	SS	00-59	16919

Programa para acertar o relógio:

```
10 DEFINT A-Z
20 DIM TM(5)
30 CL=16924
40 PRINT"DIGITE OS SEIS VALORES:MM,DD,AA,HR,MN,SS
50 INPUT TM(0),TM(1),TM(2),TM(3),TM(4),TM(5)
60 FOR I=0 TO 5
70 POKE CL-I,TM(I)
80 NEXT I
90 PRINT"O RELOGIO ESTA CORRETO"
100 END
```

Para saber se o relógio está correto, digite apenas `PRINT TIME$` ENTER

Para colocar a hora certa na tela (HH:MM:SS), nas colunas 57 a 64 da primeira linha, introduza o programa abaixo:

```
10 DEFINT A-Z
20 AT=152:DS=161          'BYTES MENOS SIGNIF
30 PRINT "DIGITE C PARA COLOCAR A HORA NA TELA"
40 PRINT "OU R PARA RETIRAR"
50 INPUT A$
```

(continua)

```
60 IF A$="C" THEN SW=AT:GOTO 100
70 IF A$="R" THEN SW=DS:GOTO 100
80 GOTO 30
100 POKE 16526,SW          'BYTE MENOS SIGNIF
110 POKE 16527,0           'BYTE MAIS SIGNIF
120 X=USR(0)               'CHAMADA USR
130 END
```

16409

Este endereço contém a seleção de letras maiúsculas e minúsculas.

Exemplo:

```
POKE 16409,0    Letras minúsculas.
POKE 16409,1    Letras maiúsculas.
```

16424

Este endereço controla o número máximo de linhas por página a ser impresso em uma impressora. Seu valor é um acima do real. Por exemplo, inicialmente o valor nesse endereço é 67, o que indica 66 linhas em cada página impressa.

16425

Este endereço contém o número de linhas impressas na página atual. Seu valor inicial é um.

16427

Este endereço controla o comprimento máximo da linha impressa.
Seu valor inicial é 255. Esse valor é o máximo.

16526

Este é o endereço do byte menos significativo para se chamar sub-rotinas assembly em seu CP 300.

O valor colocado nesse endereço, juntamente com o do endereço 16527, identifica o endereço onde começa a sub-rotina assembly que se quer executar.

16527

Este é o endereço do byte mais significativo para se chamar sub-rotinas em assembly. Esse valor, juntamente com o do endereço 16526, identifica o endereço onde começa a sub-rotina assembly que se deseja acionar.

Alocação de memória

Em muitas aplicações, é necessário interligar programas Basic com outros em linguagem assembler. Quando isso ocorrer, uma parte da memória deverá ser separada para acomodar o programa Z 80 (assembly).

Durante o "diálogo inicial", você tem a opção de realizar essa operação respondendo à pergunta:

Mem usada?

Em resposta a essa questão, você deve fornecer o endereço limite, na forma decimal, que o interpretador Basic poderá usar. O endereço limite se refere ao último byte na memória para armazenar programas e dados relacionados com a linguagem Basic.

Detecção de falhas e Manutenção

Este capítulo apresenta uma relação de sintomas e causas que, provavelmente, o impossibilitará de operar seu CP 300. Mas não se esqueça: uma instrução que não tenha sido seguida corretamente também representa um problema na seqüência de um programa. E se o seu problema ainda persistir, leve a unidade até o representante local da Prológica. A solução será imediata.

Sintoma/Providência

O computador é ligado e a mensagem Cass? não aparece na tela.

1 — Não há energia. Verifique as conexões do cabo.

2 — Seqüência incorreta ao ligar o computador.

3 — Algum periférico não está devidamente conectado. Reveja as ligações.

4 — Em sistemas com drives para disquete, mantenha pressionada BREAK enquanto liga ou pressiona RESET.

5 — A tela pode estar precisando de um ajuste de brilho. Verifique.

O computador não carrega um programa da fita.

1 — Conexão incorreta do gravador. Verifique as instruções de conexão no capítulo referente à instalação do equipamento.

2 — A velocidade de carregamento do computador não é a mesma em que foi gravado o programa.

3 — Ajuste incorreto de volume. Tente outro nível.

4 — As informações na fita foram danificadas por campo magnético ou mesmo pela própria deterioração da fita. Se você tiver uma outra cópia, tente usá-la.

Durante uma operação normal, o computador trava, sendo necessário desligá-lo e ligá-lo novamente, ou usar RESET.

1 — Isso pode ocorrer quando há flutuação de energia. Veja adiante em "Alimentação".

2 — Conector defeituoso ou mal instalado. Verifique se todos os cabos de conexão estão bem colocados e se não estão danificados.

3 — A programação pode estar irregular. Verifique-a. Provavelmente o computador entrou em looping.

Alimentação

Os computadores são sensíveis a flutuações da rede elétrica. Isto raramente constitui um problema, a menos que você esteja operando próximo a equipamentos elétricos pesados. A alimentação pode também ser instável se, nas proximidades, algum eletrodoméstico ou equipamento de escritório estiver com o interruptor defeituoso, isso provocará um faiscamento ao se acionar o equipamento.

Para suportar as condições que mencionamos é que o computador foi projetado com um filtro de linha CA interno. Ele deverá eliminar os efeitos de flutuações normais. Entretanto, se as flutuações forem severas será necessário tomar algumas das seguintes providências abaixo:

a) Instale circuitos de isolação ou "bypass" nos aparelhos que causam interferência;
b) Substitua os interruptores defeituosos;
c) Instale uma linha individual para o computador;
d) Instale um filtro especial de linha projetado para computadores e outros equipamentos sensíveis.

Os problemas com a rede elétrica são raros e, muitas vezes, podem ser evitados pela escolha correta do local de instalação de seu computador. Quanto mais complexo for o sistema, mais séria será a aplicação e maior a importância que deverá ser dada à rede de alimentação do computador.

Manutenção

Em termos de manutenção, o computador é bem econômico. É bom mantê-lo sempre limpo e livre de poeira; isto, especialmente para o teclado.

Quando você precisar limpar o gabinete do computador, utilize uma flanela limpa e seca. Os periféricos (gravador, impressora, etc.) requerem maiores cuidados. Cada periférico tem um manual específico; verifique o sistema de manutenção de cada um.

Apêndices

Apêndice A
Sumário do CP 300

Abreviaturas e caracteres especiais

Modo de comando (imediato)

Comandos	*Função*
ENTER	Introduz a linha digitada e interpreta o comando.
←	Retrocede o cursor e apaga o último caractere digitado.
SHIFT ←	Posiciona o cursor no começo da linha e a apaga.
	Muda para a próxima linha.
:	Delimitador de instrução, usado entre instruções na mesma linha lógica.
→	Move o cursor para a próxima parada de tabulação. As tabulações estão nas posições 0, 8, 16, 24, 32, 48 e 56.
SHIFT →	Converte a tela para 32 caracteres por linha.
CLEAR	Limpa a tela e converte para 64 caracteres por linha.

Modo de execução

SHIFT @	Pausa na execução. Isso "congela" a tela durante um comando LIST.
BREAK	Suspende a execução do programa e retorna o comando ao INPUT.

Abreviaturas

?	Usado no lugar de PRINT
'	Usado no lugar de REM
.	"linha corrente" usada no lugar do número de linha com LIST, EDIT, etc.

SHIFT ↓ Para dar saída de um caractere de controle, pressione SHIFT e depois ↓. Enquanto se retêm as duas teclas, pressione a tecla para um caractere de controle desejado. Por exemplo: para digitar um controle-Z pressione SHIFT ↓ Z.

Nota:

Você deve usar a tecla SHIFT da esquerda.

Comandos e instruções

Comando	*Função*	*Exemplos*

AUTO *xx,yy*	Ativa a numeração automática de linha, iniciando por *xx* e utilizando incrementos *yy*.	`AUTO`
		`AUTO 10`
		`AUTO 5.5`
		`AUTO .,10`

CLEAR	Zera as variáveis e anula as strings.	`CLEAR` `CLEAR 500`
CLEAR *x*	O mesmo que CLEAR, mas também separa *x* bytes para strings.	`CLEAR MEM/4`
CLOAD	Copia um programa BASIC da fita.	`CLOAD "A"`
CLOAD?	Compara um programa BASIC na fita com um contido na memória.	`CLOAD? "A"`
CONT	Continua a execução após um BREAK ou STOP.	`CONT`
CSAVE	Armazena um programa BASIC na fita.	`CSAVE "A"`

DELETE *xx,yy*	Apaga as linhas de programa da linha *xx* à linha *yy*.	`DELETE 100`
		`DELETE 10-50`
		`DELETE .`

EDIT *xx*	Introduz o modo de edição para a linha *xx*. Veja subcomandos do modo de Edição.	`EDIT 100` `EDIT .`

LLIST *xx,yy*	Lista todas as linhas do programa *xx* a *yy* na impressora.	`LIST`
		`LIST 30-90`
		`LIST 90-`
		`LIST -90`
		`LIST 90`
		`LIST .`

LIST *xx,yy*	Lista todas as linhas do programa de *xx* a *yy*.	`LLIST .` `LLIST 30-60`
NEW	Apaga o programa inteiro e inicializa todas as variáveis, indicadoras, etc.	`NEW`

Instruções de programas

DEFDBL ***lista ou faixa de letras***

Define como precisão-dupla todas as variáveis começando por letras dentro da faixa de letras especificada.

Exemplos

```
DEFDBL J
DEFDBL X,Y,Z
DEFDBL A-E, J
```

DEFINT ***lista ou faixa de letras***

Define como sendo inteiras todas as variáveis começando por letras dentro da faixa de letras especificada.

```
DEFINT A
DEFINT C,E,G
DEFINT A-K
```

DEFSNG ***lista ou faixa de letras***

Define como sendo de precisão simples todas as variáveis, começando por letras ou faixa de letras especificada.

```
DEFSNG L
DEFSNG A-L,Z
DEFSNG P,R,A-K
```

DEFSTR ***lista ou faixa de letras***

Define como sendo string todas as variáveis começando por letra ou faixa de letras especificada.

```
DEFSTR A-J
```

Atribuição ou alocação

CLEAR ***n***

Separa um número especificado de *n* bytes para armazenagem de string.
Limpa valores e tipos de todas as variáveis.

```
CLEAR 750
CLEAR MEM/10
CLEAR 0
```

DIM ***matriz (dim#1...dim#k)***

Define armazenagem para matriz de *k* dimensões com tamanho específico por dimensão: *dim#1, dim#2, ..., etc...* pode ser seguida por uma lista de dimensões separadas por vírgulas.

```
DIM A (2,3)
DIM A1(5),A2(15)
DIM B(X+2)
DIM T(3,3,5)
```

LET ***variável = expressão***

Atribui o valor da expressão à variável.
A palavra LET é opcional.

```
LET A$="MARIO"
LET AX=1#
LET B1=C1
```

Seqüência de execução

END

Finaliza a execução, retorna para o modo imediato.

Exemplos

```
100 END
```

STOP

Para a execução, imprime a mensagem "Break na" com o número da linha onde o programa foi interrompido. A execução pode ser retomada com a instrução CONT.

```
100 STOP
```

GOTO *número de linha*

Passa a execução do programa para a linha de número especificado.

```
GOTO 100
```

GOSUB *número de linha*

Passa para a sub-rotina que começa pela linha de número especificado.

```
GOSUB 300
```

RETURN

Passa a execução do programa para a linha subseqüente à que contém a última instrução GOSUB executada.

```
100 RETURN
```

ON *exp* GOTO *linha 1, linha 2, ..., linha k*

Avalia *exp*; se INT *(exp)* for igual a um dos números de um a *k*, a execução passará para o número de linha correspondente. De outra forma, avançará para a próxima linha de programa.

```
ONKGOTO 20,100
```

ON *exp* GOSUB

Opera de forma similar a ON... GOTO, exceto no fato de que a execução é destinada para uma sub-rotina.

```
ONJGOSUB 20,70
```

FOR *var* = *exp* TO *exp* STEP *exp*

Abre um loop FOR—NEXT, STEP é opcional. Caso não seja usado STEP, o incremento será de uma unidade.

```
FOR I=1 TO 50
 STEP 1.5
```

NEXT *variável*

Fecha um LOOP FOR—NEXT. Variável pode ser omitida.

Exemplos

```
NEXT
NEXT ITEM
NEXT A,B,COD,D
```

ON ERROR GOTO *número de linha*

Define o início de uma rotina de manipulação de erro; em caso de ocorrência de erro, a execução será desviada para a linha de número especificado.

```
ON ERROR
 GOTO 3330
```

ERROR (*código*)

Simula um erro especificado pelo código.

```
ERROR (14)
```

RESUME *n*

Retorna da rotina de erro para a linha especificada por *n*. Se *n* for zero ou não for especificado, retorna à instrução contendo erro. Se *n* for NEXT, retorna para a instrução seguinte a instrução de erro.

```
RESUME
RESUME 0
RESUME 100
```

RANDOM

Alimenta o gerador de números aleatórios.

```
RANDOM
```

REM

Indicador de mensagem (REMark); ignora o restante da linha.

```
REM ATENCAO
' Y=A+BX
```

Testes-Instruções condicionais

IF *exp-1* THEN *instrução-1* ELSE...

Testa *exp-1*: se verdadeira, executa a *instrução-1* e então salta para a próxima linha de programa (exceto se *instrução-1* for GOTO). Se *exp-1* é falso, salta diretamente para instrução ELSE e executa as instruções subseqüentes.

```
IF A=0 THEN
 PRINT"ZERO"
 ELSE ?"ZERO"
```

Instruções gráficas

CLS

Limpeza da tela.

```
CLS
```

RESET (*x,y*)

Exemplos

Desativa uma posição (ponto da tela) especificada pela coordenada horizontal *x* e pela vertical *y*. $0 <= X < 128$ e $0 <= Y < 48$

```
RESET (8+B,11)
```

SET (*x,y*)

Ativa a posição (ponto da tela) especificada pelas coordenadas *x* e *y*. Mesma limitação de argumentos do RESET.

```
SET (A*2,B+C)
```

Instruções especiais

POKE *local, valor*

Carrega o *valor* em um certo *local* de memória (ambos os argumentos sob a forma decimal). $0 <= valor <= 255$

```
POKE 15635,34
POKE 17770,A+N
```

OUT *porta, valor*

Envia *valor* à *porta* (ambos os argumentos entre 0 e 255, inclusive).

```
OUT 255,10
OUT 55,A
```

Funções string

ASC (*string)**

Retorna código ASCII do 1º caractere no argumento *string*.

```
ASC (B$)
ASC ("H")
```

CHR$ (*código*)

Fornece um string de um caractere, definido pelo *código*. Se o *código* especifica uma função de controle, aquela função é ativada.

```
CHR$ (1)
CHR$ (A)
CHR$ (34)
```

FRE (*string*)

Fornece o espaço disponível de memória para armazenagem de string. O argumento é uma variável simples.

```
FRE (A$)
```

**string* pode ser uma variável, expressão ou constante string.

INKEY$

Varre o teclado e fornece um string correspondente à tecla pressionada durante uma varredura (string nulo se nenhuma tecla for pressionada).

Exemplos

```
INKEY$
```

LEFT$ (*string,n*)

Fornece os primeiros *n* caracteres de um *string*.

```
LEFT$ (A$,1)
LEFT$ (LI$+Cr$,8)
LEFT$ (A$,M+L)
```

LEN (*string*)

Fornece o comprimento do *string* (zero para *string* nulo).

```
LEN (A$+B$)
LEN ("HORA")
```

MID$ (*string,p,n*)

Fornece substring do *string* com extensão *n* e iniciado na posição da string.

```
MID$ (M$,5,2)
MID$(M$,F,L)
```

RIGHT$ (*string, n*)

Fornece os últimos *n* caracteres do *string*.

```
RIGHT$(NA$,7)
RIGHT$ (AB$,M2)
```

STR$ (*exp numérica*)

Fornece a representação.

```
STR$ (A+B*2)
```

STRING$ (*n*, carac)

Fornece uma seqüência de *n* símbolos de caracteres, empregando o primeiro caractere de *carac*.

```
STRING$(30,"-")
STRING$(25,"A")
STRING$ (5,C$)
```

TIME$

Fornece data e horas.

```
TIME$
```

VAL (*string*)

Fornece um valor numérico correspondente a um string dotado devalor numérico.

```
VAL ("1"+A$)
VAL (G1$)
```

Instruções entrada/saída

PRINT *exp** *Exemplos*

Dá saída na tela do valor da expressão *exp*.
A expressão *exp* pode ser uma constante, uma expressão numérica, ou string ou uma combinação delas.
A vírgula funciona como um modificador PRINT. Faz o cursor avançar para a próxima zona de impressão.
O ponto e vírgula funciona como um modificador PRINT. Insere um espaço após um item numérico na listagem PRINT. Não insere nenhum espaço após um item string. No final da listagem PRINT, suprime o retorno automático de carro.

```
PRINT A$
PRINT X+3
PRINT "D="D
PRINT 1,2,3,4
PRINT "1","2"
PRINT 1,2
PRINT X;Y;Z
PRINT "OK";
```

PRINT *n*

Modificar PRINT; inicia a impressão na posição *n* especificada da tela.

```
PRINT @ 540, "X"
```

PRINT TAB (*n*)

Modificador PRINT move o cursor para a posição *n* (expressão) especificada na tela.

```
PRINT TAB (n)n
```

PRINT USING *string; exp*

Especificador de formato PRINT, da saída de *exp* na forma especificada por um campo *string* (veja a seguir).

```
PRINTUSINGA$;X
? USING"#.#";Y
```

INPUT *"mensagem"; variável*

Imprime a mensagem (se houver) e espera uma entrada pelo teclado.

```
INPUT"NOME";A$
INPUT"VALOR",X
INPUT"A,B";X,Y
INPUT A,B,C,D$
```

LPRINT

Dá saída para a impressora.

```
LPRINT A$
```

**exp* é qualquer expressão ou constante numérica.

PRINT # —1

Dá saída ao gravador

Exemplos

```
PRINT#-1,A,B,C
```

INPUT # —1

Recebe dados do cassete

```
INPUT#-1,A,D$
```

DATA

Lista de itens. Retém os dados para acesso pela instrução READ.

```
DATA 22,23,11,
DATA "JOSE"
```

READ *lista de variáveis*

Atribui valores às variáveis especificadas.

```
READ A,A1,A2
READ A$,B$,C$
```

RESTORE

Reajusta o indicador de dados ao primeiro item na primeira instrução DATA.

```
RESTORE
```

Especificadores de campo para instruções PRINT USING

#	Campo numérico (um dígito por #).	# # #
.	Posição do ponto decimal.	# # #.# #
+/-	Imprime sinais dianteiros e traseiros. + para números positivos — para negativos.	+ # #.# #
-	Imprime sinal traseiro apenas se o valor impresso for negativo.	# # #.# # -
* *	Preenche espaços em branco com asteriscos.	**# # #.# #
$ $	Coloca o sinal cifrão imediatamente à esquerda do primeiro dígito.	$$ # # # # #
	****$**	
	Coloca o sinal cifrão à esquerda do primeiro dígito e preenche os espaços em branco anteriores com asteriscos.	**$ # # #.# #

↑↑↑↑ ou ↑↑↑↑

Exemplos

Formato exponencial com um dígito significativo à esquerda do decimal.
Pressione ↑ para introduzir este caractere

```
#.# # ↑↑↑↑
```

,

Imprime números com vírgulas, como em 1,356,000

```
#,# # #.# #
```

!

Caractere único

```
!
```

% espaço%

String com comprimento igual a 2 mais o número de espaços entre os símbolos de porcentagem.

```
% %
```

Funções aritméticas

(Exceto onde especificado, $-1.7E+38 = exp = 1.7E+38$)

ABS (*exp*)

Fornece o valor absoluto.

```
ABS(L*7)
```

ATN (*exp*)

Fornece o arco-tangente em radianos

```
ATN(2.7)
ATN(3.14)
```

CDBL (*exp*)

Fornece representação da *exp* com precisão dupla.

```
CDBL (A)
CDBL (A+1/3#)
```

CINT (*exp*)

Fornece o maior inteiro não maior do que a *exp*.
Limite: $-32768 <= exp < +32768$

```
CINT (A#+B)
```

COS (*exp*)

Fornece o cosseno de *exp*, assumindo *exp* em radianos.

```
COS (2*A)
COS (A/57.296)
```

CSNG (*exp*)

Fornece representação com precisão simples, com arredondamentos 5/4 do decimal menos significativo, quando *exp* tem precisão dupla.

```
CSNG (33*B#)
CSNG (A#)
```

EXP (*exp*)

Fornece o exponencial natural, e^{exp} = EXP *(exp)*.

Exemplos

```
EXP(34,5)
EXP(A*B*C-1)
```

FIX (*exp*)

Fornece o inteiro equivalente a *exp* truncada (a parte fracionária de *exp* é eliminada).

```
FIX (A - B)
```

INT (*exp*)

Fornece o maior inteiro não maior que *exp*.

```
INT (A+B*C)
```

LOG (*exp*)

Fornece o logaritmo natural (base e) de *exp*.
Limites: *exp* deve ser positivo.

```
LOG(12.33)
LOG (A*B+B)
```

RND(*0*)

Fornece um número pseudo-aleatório entre 0,000001 e 0,999999, inclusive.

```
RND(0)
```

RND (*exp*)

Fornece um número pseudo-aleatório entre 1 e INT (exp), inclusive. Limites: $1 <= exp < 32768$

```
RND(40)
RND(A + B)
```

SGN (*exp*)

Fornece — 1 para uma *exp* negativa, 0 para $exp = 0$ e + 1 para *exp* positiva.

```
SIN(A/B)
SIN(90/57.296)
```

SIN (*exp*)

Fornece o seno de *exp*, sendo *exp* em radianos.

```
TAN(X)
TAN(X*.017453)
```

SQR (*exp*)

Fornece a raiz quadrada de *exp*. Limites: *exp* não deve ser negativa.

```
SQR(A*A-B*B)
```

TAN *(exp)*

Fornece a tangente de *exp*, sendo *exp* em radianos.

```
TAN(X)
TAN(X*.01745329)
```

Funções especiais

ERL

Fornece o número da linha de erro atual.

Exemplos

```
ERL
```

ERR

Fornece um valor relacionado ao código do erro corrente (caso haja algum erro). ERR = (cód. erro - 1)* 2 também: (ERR/2) + 1 = código erro.

```
ERR/2+1
```

INP (*porta*)

Introduz e fornece o valor corrente da *porta* especificada. Tanto o resultado como o argumento encontram-se na faixa de 0 a 255, inclusive.

```
INP (55)
```

MEM

Fornece o número total de bytes desprotegidos e não utilizados da memória. Não inclui o espaço não utilizado de strings.

```
MEM
```

PEEK (*local*)

Fornece o valor guardado no byte especificado de memória. O *local* deve ser um endereço válido de memória, sob a forma decimal (veja Principais Endereços de Memória).

```
PEEK (15730)
```

POINT (*x,y*)

Verifica a posição do gráfico especificada pelas coordenadas *x* e *y*. Se a posição é válida, fornece um nível verdadeiro (-1); se é falsa, fornece um nível falso (0). Limites: 0 < = x < 128;0 < = y < 48

```
POINT(30,40)
```

POS (*0*)

Fornece um número indicando a posição corrente do cursor. O argumento "0" é irrelevante.

```
POS (0)
```

USR (*n*)

Desvia para uma sub-rotina em linguagem de máquina.

```
USR(0)
```

VARPTR (*var*)

Fornece o endereço onde estão armazenados o nome, o valor e o indicador de uma variável; *var* deve ser um nome válido de variável.

Exemplo

```
VARPTR(var)
```

Palavras reservadas para o BASIC*

@	ELSE	LLIST	RENAME
ABS	END	LPRINT	RESET
AND	EOF*	LOAD*	RESTORE
ASC	ERL	LOC*	RESUME
ATN	ERR	LOF*	RETURN
AUTO	ERROR	LOG	RIGHT$
CDBL	EXP	MEM	RND
CHR$	FIELD*	MERGE*	RSET*
CINT	FIX	MID$	RUN
CLEAR	FN	MKD$*	SAVE*
CLOCK	FOR	MKI$*	SET
CLOSE*	FORMAT	MKS$*	SGN
CLS	FRE	NAME*	SIN
CMD*	FREE*	NEW	SQR
CONT	GET*	NEXT	STEP
COS	GOSUB	NOT	STOP
CSNG	GOTO	ON	STRING$
CVD*	IF	OPEN*	STR$
CVI*	INKEY$	OR	SYSTEM
CVS*	INP	OUT	TAB
DATA	INPUT	PEEK	TAN
DEFDBL	INSTR	POINT	THEN
DEFFN	INT	POKE	TIME$
DEFINT	KILL*	POS	TO
DEFSNG	LEFT$	POSN	TROFF
DEFUSR	LET	PRINT	TRON
DEFSTR	LSET*	PUT	USING
DELETE	LEN	RANDOM	USR
DIM	LINE*	READ	VAL
EDIT	LIST	REM	VARPTR
			VERIFY*

Obs.:

* Algumas destas palavras não têm função no BASIC do CP 300; elas estão reservadas para uso do Basic com Disco. Nenhuma delas pode ser utilizada como nome de variável; você incorrerá em um erro de sintaxe, caso tente usar alguma delas com esse objetivo.

Limites de memória para uso dinâmico

Faixas abrangidas

Inteiros: −32768 a +32767, inclusive.
Precisão simples: −1.701411E ± 38 inclusive.
Precisão dupla: −1.701411834544556D ± 38 a 1.701411834544556D ± 38, inclusive.
Faixa de strings: até 255 caracteres

Números de linhas permitidas: de 0 a 65529, inclusive.

Extensão de linha de programa: até 255 caracteres (entrada 240, edição 255).

Requisitos de memória

As linhas de programa requerem no mínimo 5 bytes, da seguinte forma:
número de linha — 2 bytes
indicador de linha — 2 bytes
retorno da linha — 1 byte
Além disso, cada palavra, operador, nome de variável, caractere especial e caractere constante reservado exige 1 byte.

Alocação dinâmica de memória (em operação)

Variáveis inteiras: 5 bytes cada
(2 para o valor, 3 para o nome)
Variáveis de precisão simples: 7 bytes cada
(4 para o valor, 3 para o nome)
Variáveis de precisão dupla: 11 bytes cada
(8 para o nome, 3 para indicador de pilha e variável, 1 para cada caractere)
Variáveis matriz: 12 bytes no mínimo
(3 para o nome, 2 para o tamanho total, 1 para número de dimensões, 2 para tamanho de cada dimensão e 2, 3, 4 ou 8 para cada elemento da matriz, dependendo do tipo da mesma).
Cada loop ativo de FOR-NEXT requer 16 bytes.
Cada nível de parênteses requer 4 bytes, mais 12 bytes para cada valor temporário.

Fórmula geral para computar os requisitos de memória para matrizes

A matriz G(N1, N2,...,Nk) requer o seguinte espaço de memória:

$$14 + (k*2) + T\,(N1+1)*(N2+1)*\ldots.*(Nk+1)$$

onde *k* é o número de dimensões da matriz, e o valor T depende do tipo de matriz:

TIPO	T =
Inteiro	2
Precisão simples	4
Precisão dupla	8
string*	3

Precisão

Cálculos de precisão simples envolvendo + , – , * e / tem precisão de até 6 dígitos significativos; cálculos de precisão dupla, envolvendo as mesmas operações, têm precisão de até 16 dígitos significativos.

O operador de exponenciação (exibido como "["), tem precisão simples.

As funções logarítmicas e trigonométricas têm precisão simples; outras funções têm a precisão em função do argumento de entrada e da função.

Assim, por exemplo, CDBL fornece um valor de precisão dupla, e ABS fornece um valor com a mesma precisão do argumento de entrada.

Ao converter de precisão simples para dupla, utilize a seguinte técnica para evitar a introdução de valores incorretos nos dígitos adicionais de precisão:
variável de precisão dupla = VAL (STR$ (variável de precisão simples)).

*Ao computar os requisitos de memória para matriz em string, você deve acrescentar a extensão do texto de cada elemento da matriz. Quando a matriz é dimensionada pela primeira vez, todos os elementos tem extensão 0.

O texto da string será armazenado no espaço reservado para strings (reservado pela instrução CLEAR n).

Apêndice B
Códigos de erros

Códigos	*Abreviação*	*Erro*
01	NF	NEXT sem o FOR
02	SN	Erro de sintaxe
03	RS	RETURN sem GOSUB
04	OD	Insuficiência de dados
05	FC	Chamada ilegal da função
06	OV	Overflow
07	OM	Insuficiência de memória
08	UL	Linha indefinida
09	BS	Subscrito fora da caixa
10	DD	Matriz dimensionada
11	/0	Divisão por zero
12	ID	Imediato ilegal
13	TM	Introdução ilegal
14	OS	Falta de espaço
15	LS	String muito extenso
16	ST	Fórmula de string muito complexa
17	CN	Impossibilidade de continuar
18	NR	Falta RESUME
19	RW	RESUME sem Erro
20	UE	Erro não imprimível
21	MO	Falta de operando
22	FD	Dados defeituosos de arquivo
23	L3	Somente em BASIC-DISCO

Explicação das mensagens de erro

NF

NEXT sem FOR. NEXT utilizado sem uma instrução FOR corresponde. Esse erro também pode ocorrer se a instrução NEXT variável for invertida num loop entrelaçado.

SN

Erro de sintaxe: é o resultado, normalmente, de pontuação incorreta; parênteses abertos, um caractere ilegal ou um comando errado.

RG

RETURN sem GOSUB: uma instrução RETURN foi encontrada antes que um GOSUB correspondente fosse executado.

OD

Insuficiência de Dados: uma instrução READ ou INPUT foi executada com dados insuficientes. A instrução DATA pode ter sido deixada de fora, ou todos os dados podem ter sido lidos, da fita ou DATA.

FC

Chamada ilegal de função: tentativa de extrair raiz quadrada de um argumento negativo, dimensão negativa de matriz, argumentos negativos ou nulos em logaritmos, etc. Ou então, chamada de USR sem que o ponto de entrada tenha passado por um POKE.

OV

Overflow (sobrecarga): a magnitude do número introduzido, ou derivado, é muito grande para o computador.

Nota:

Matemáticamente, não existe erro de sobrecarga.

Os números inferiores a ± 1.701411E − 38 (precisão simples) ou a ± 1.701411834544556D − 38 (precisão dupla) são arredondados para 0. Vide /0 a seguir.

OM

Insuficiência de memória: toda a memória disponível foi utilizada ou reservada. Pode ocorrer com grandes dimensões de matrizes, ramificações como o GOTO, GOSUB e FOR-NEXT.

UL

Linha indefinida: tentativa de referir ou desviar para uma linha inexistente.

BS

Subscrito fora de faixa: tentativa de estipular um elemento de matriz com um subscrito fora da faixa dimensionada.

DD

Matriz redimensionada: tentativa de dimensionar uma matriz já previamente dimensionada por uma instrução DIM. É bom colocar todas as instruções de dimensionamento no início de cada programa.

/0

Divisão por zero: tentativa de utilizar o valor zero no denominador.

Nota:

Se for impossível localizar uma divisão por zero, que seja óbvia, verifique divisões por números inferiores ás faixas permitidas. Veja também erro OV, acima, e as faixas válidas na pág. 165.

ID

Imediato ilegal: utilização do comando INPUT como sendo imediato.

TM

Introdução ilegal: tentativa de atribuir uma string a uma variável simples.

OS

Falta de espaço para string: o espaço reservado para string foi excedido.

LS

String muito extensa: uma variável string recebeu um valor string que excedeu a extensão de 255 caracteres.

ST

Fórmula de string muito complexa: operação string muito complexa para o computador. A operação deve ser dividida em etapas.

CN

Impossibilidade de continuar um programa: O CONT foi introduzido em um determinado ponto, onde se torna impossível dar continuidade ao programa. Isto, após um END ou EDIT.

NR

Falta RESUME: quando o final do programa é encontrado na rotina de manipulação de erros.

RW

Resume sem erro: um RESUME foi encontrado antes que um ON ERROR GO-TO fosse executado.

UE

Erro não imprimível: tentativa de gerar um erro, usando uma instrução ERROR, com um código inválido.

MO

Falta de operando: tentativa de operação sem o fornecimento de um dos operandos necessários.

FD

Dados defeituosos em arquivo: entrada de dados de uma fonte externa (fita por exemplo) de forma incorreta ou em uma seqüência imprópria.

L3

Somente o BASIC—DISCO: tentativa de utilizar uma instrução, função ou comando, somente disponível no BASIC—DISCO (DOS 300).

Erros de carregamento de programas em fita

Durante a operação de carregamento da memória com programas em fita podem surgir três mensagens diferentes no canto superior direito da tela. Essas mensagens indicam que a operação não foi bem sucedida e deve ser repetida. A tabela abaixo explica o significado de cada uma.

Mensagem	*Significado*
*C**	Ocorre quando há um certo erro de verificação (checksum) durante o carregamento de programas em linguagem Z 80.
D*	Ocorre quando há um erro de dados (data error) durante o carregamento de um programa em Basic.
BK	Indica que a tecla BREAK foi pressionada cancelando a operação.

Os dois primeiros podem ser causados por conexão incorreta do gravador ou volume inadequado do sinal. Verifique essas condições e tente novamente. Se ainda persistir o problema, pode ser que haja sujeira na cabeça de gravação. Limpe-a conforme está explicado no manual do gravador e tente mais uma vez. Se mesmo assim a operação não se realizar, então, provavelmente, a gravação na fita foi danificada por eletricidade estática, campo magnético intenso ou alguma outra causa.

Apêndice C
Códigos de caracteres do CP 300

Código de caracteres do CP 300 é quando um texto é representado por códigos no computador. Por exemplo: a letra A é representada pelo código 65; as funções de controle e os gráficos também são representados por códigos; e os códigos de caracteres assumem valores de zero a 255.

Os códigos de zero a 31 normalmente representam funções de controle. Por exemplo: o código 13 representa um retorno de carro ou "final de linha".

Entretanto, no CP 300, estes mesmos códigos também representam 32 caracteres especiais na tela. Para esta aplicação, eles devem ser colocados (POKE) na memória e não impressos (PRINT).

Os códigos de 32 a 127 representam todos os caracteres normalmente utilizados em texto: as letras, números e outros. Os caracteres do texto do CP 300 obedecem ao padrão ASCII (American National Standard Code for Information Interchange).

Os códigos 128 a 191, quando endereçados à tela do vídeo, apresentam os 64 caracteres gráficos.

Os códigos 192 a 255, quando endereçados à tela do vídeo, representam códigos de compressão de espaços ou caracteres especiais, conforme determinado por software.

Muito dos códigos podem ser introduzidos pelo teclado; todos eles podem ser armazenados em um "string" e endereçados para qualquer periférico.

Por exemplo: para enviar um código 31 para a tela use uma instrução como:
PRINT CHR$ (31)

Nota:

Na tabela a seguir, endereço se refere à memória RAM da Tela (endereços 15360 a 16383).

Na tabela a seguir, são relacionados todos os 255 caracteres do CP 300. A coluna "teclado" mostra a tecla, ou combinação de teclas, necessária para a introdução do caractere. A coluna "vídeo" mostra o resultado na tela da impressão do caractere (PRINT CHR$). A última coluna mostra o resultado de se colocar o código diretamente na memória de vídeo (POKE). Nas páginas 118 e 119 são mostrados os caracteres especiais e gráficos do CP 500 e na página 120, encontra-se um lay-out da tela.

Código		*Teclado*	*Vídeo*	*POKE*
Dec.	Hex		PRINT CHR$ (código)	End./Cód.
0	00		Sem efeito	Ver caracteres especiais de 0 a 31 na pág. 118
1	01	BREAK SHIFT ↓ A	Sem efeito	
2	02	SHIFT ↓ B	Sem efeito	
3	03	SHIFT ↓ C	Sem efeito	
4	04	SHIFT ↓ D	Sem efeito	
5	05	SHIFT ↓ E	Sem efeito	
6	06	SHIFT ↓ F	Sem efeito	
7	07	SHIFT ↓ G	Sem efeito	
8	08	← SHIFT ↓ H	Retrocesso e limpeza	
9	09	→ SHIFT ↓ I	Tabulação (0, 8, 16, 24...)	
10	0A	↓ SHIFT ↓ J	Move o cursor para o começo da próxima linha e apaga a linha	
11	0B	SHIFT ↓ K	Sem efeito	
12	0C	SHIFT ↓ L	Sem efeito	
13	0D	ENTER SHIFT ↓ M	Move o cursor para o começo da próxima linha e apaga a linha	
14	0E	SHIFT ↓ N	Ativa o cursor	
15	0F	SHIFT ↓ O	Desativa o cursor	
16	10	SHIFT ↓ P	Sem efeito	
17	11	SHIFT ↓ Q	Sem efeito	
18	12	SHIFT ↓ R	Sem efeito	
19	13	SHIFT ↓ S	Sem efeito	
20	14	SHIFT ↓ T	Sem efeito	
21	15	SHIFT ↓ U	Controla compressão de espaços/caracteres especiais	
22	16	SHIFT ↓ V	Controla caracteres especiais/alternativos	
23	17	SHIFT ↓ W	Caracteres tamanho duplo (32 cpl)	
24	18	SHIFT ← SHIFT ↓ X	Retrocesso sem limpeza	
25	19	SHIFT ↓ Y	Avança o cursor	
26	1A	SHIFT ↓ Z	Desde o cursor	
27	1B	SHIFT ↑	Sobe o cursor	

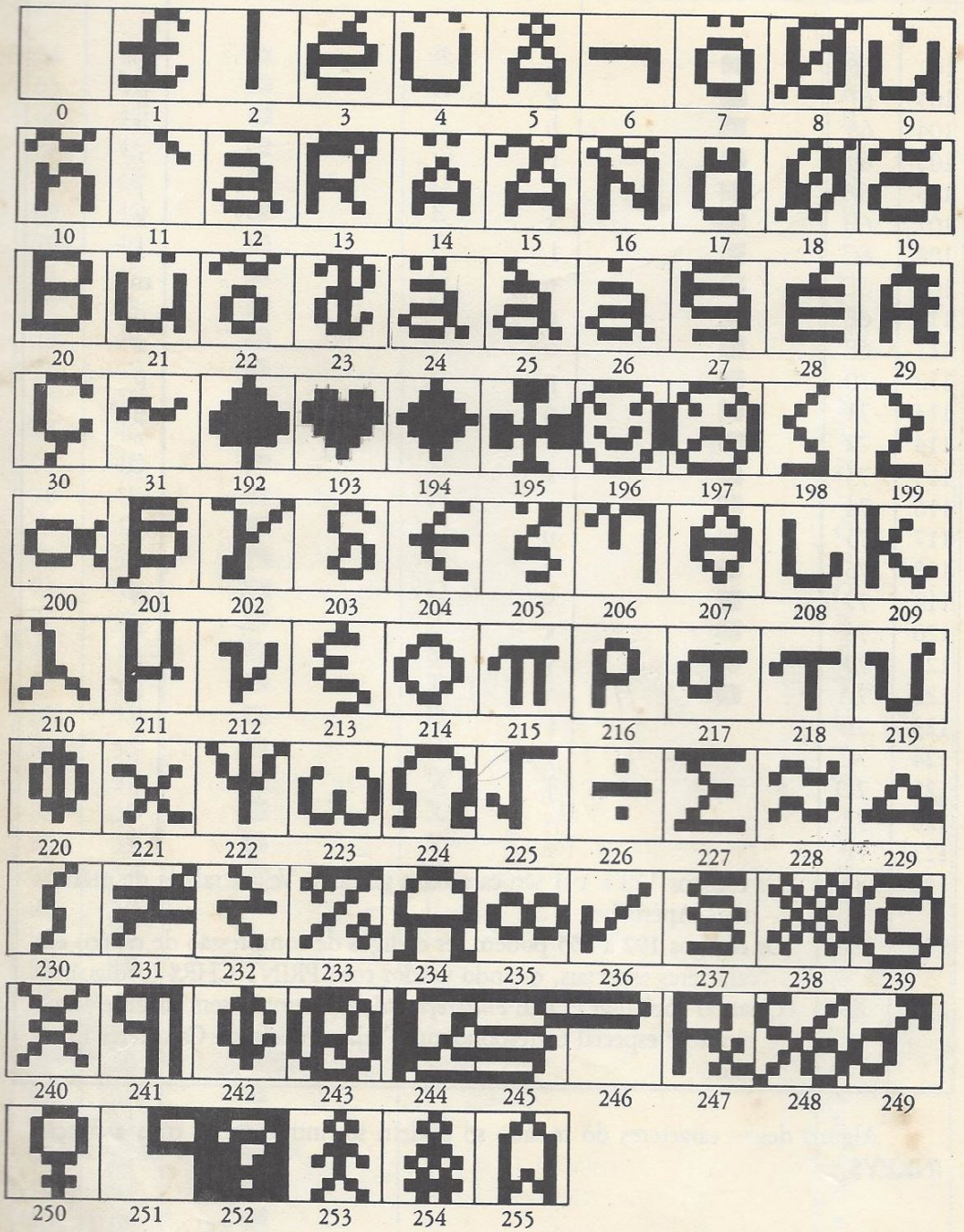
Código		*Teclado*	*Vídeo*	*POKE*
Dec.	Hex		PRINT CHR$ (código)	End./Cód.
28	1C	SHIFT ↓ ,	Move o cursor para o canto superior esquerdo	
29	1D	SHIFT ↓ 9	Apaga a linha e recomeça	
30	1E	SHIFT ↓ .	Apaga o final da linha	
31	1F	CLEAR SHIFT ↓ /	Apaga toda a tela	
32	20	ESPAÇO	♭	♭
33	21	!	!	!
34	22	"	"	"
35	23	#	#	#
36	24	$	$	$
37	25	%	%	%
38	26	&	&	&
39	27	'	'	'
40	28	(	(	(
41	29	)	)	)
42	2A	*	*	*
43	2B	+	+	+
44	2C	,	,	,
45	2D	—	—	—
46	2E	.	.	.
47	EF	/	/	/
48	30	0	0	0
49	31	1	1	1
50	32	2	2	2
51	33	3	3	3
52	34	4	4	4
53	35	5	5	5
54	36	6	6	6
55	37	7	7	7
56	38	8	8	8
57	39	9	9	9
58	3A	:	:	:
59	3B	;	;	;
60	3C	<	<	<
61	3D	=	=	=
62	3E	>	>	>
63	3F	?	?	?

Código		*Teclado*	*Vídeo*	*POKE*
Dec.	Hex		PRINT CHR$ (código)	End./Cód.
64	40	@	@	@
65	41	A	A	A
66	42	B	B	B
67	43	C	C	C
68	44	D	D	D
69	45	E	E	E
70	46	F	F	F
71	47	G	G	G
72	48	H	H	H
73	49	I	I	I
74	4A	J	J	J
75	4B	K	K	K
76	4C	L	L	L
77	4D	M	M	M
78	4E	N	N	N
79	4F	O	O	O
80	50	P	P	P
81	51	Q	Q	Q
82	52	R	R	R
83	53	S	S	S
84	54	T	T	T
85	55	U	U	U
86	56	V	V	V
87	57	W	W	W
88	58	X	X	X
89	59	Y	Y	Y
90	5A	U	Z	Z
91	5B		[	[
92	5C		\	\
93	5D		]	]
94	5E		^	^
95	5F		—	—
96	60	SHIFT	`	`
97	1	A	a	a
98	62	B	b	b
99	63	C	c	c
100	64	D	d	d
101	65	E	e	e

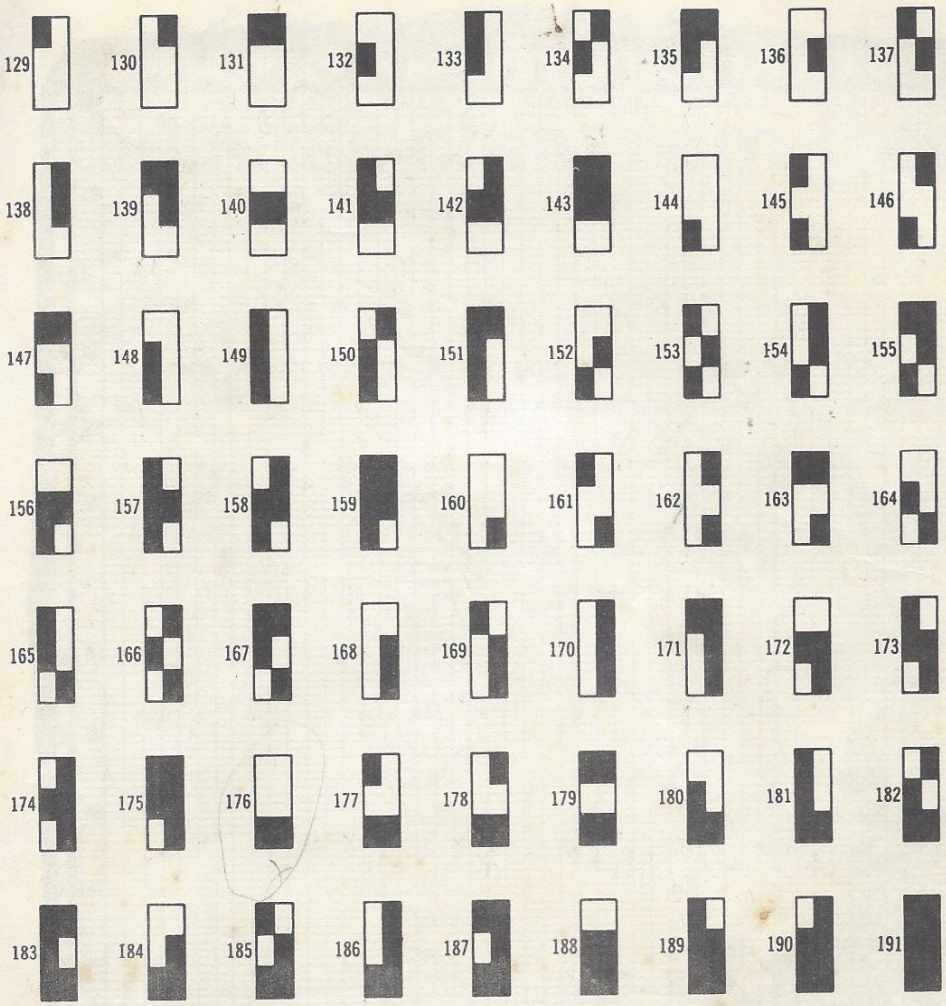
Código		*Teclado*	*Vídeo*	*POKE*
Dec.	Hex		PRINT CHR$ (código)	End./Cód.
102	66	F	f	f
103	67	G	g	g
104	68	H	h	h
105	69	I	i	i
106	6A	J	j	j
107	6B	K	k	k
108	6C	L	l	l
109	6D	M	m	m
110	6E	N	n	n
111	6F	O	o	o
112	70	P	p	p
113	71	Q	q	q
114	72	R	r	r
115	73	S	s	s
116	74	T	t	t
117	75	U	u	u
118	76	V	v	v
119	77	W	w	w
120	78	X	x	x
121	79	Y	y	y
122	7A	Z	z	z
123	7B		{	{
124	7C		:	:
125	7D		}	}
126	7E		~	~
127	7F		±	±
128	80	Os códigos 128 a 191 são caracteres gráficos. Veja a tabela de gráficos neste Apêndice.		
192	C0	Os códigos 192 a 255 podem ser códigos de compressão de espaço em caracteres especiais, quando usados com PRINT CHR$ (código).		
255	FF	Quando você usar POKE *endereço, código*, sempre será considerado o caractere especial correspondente. Veja a tabela de Caracteres Especiais.		

Alguns desses caracteres do teclado só podem ser introduzidos com a função INKEY$.

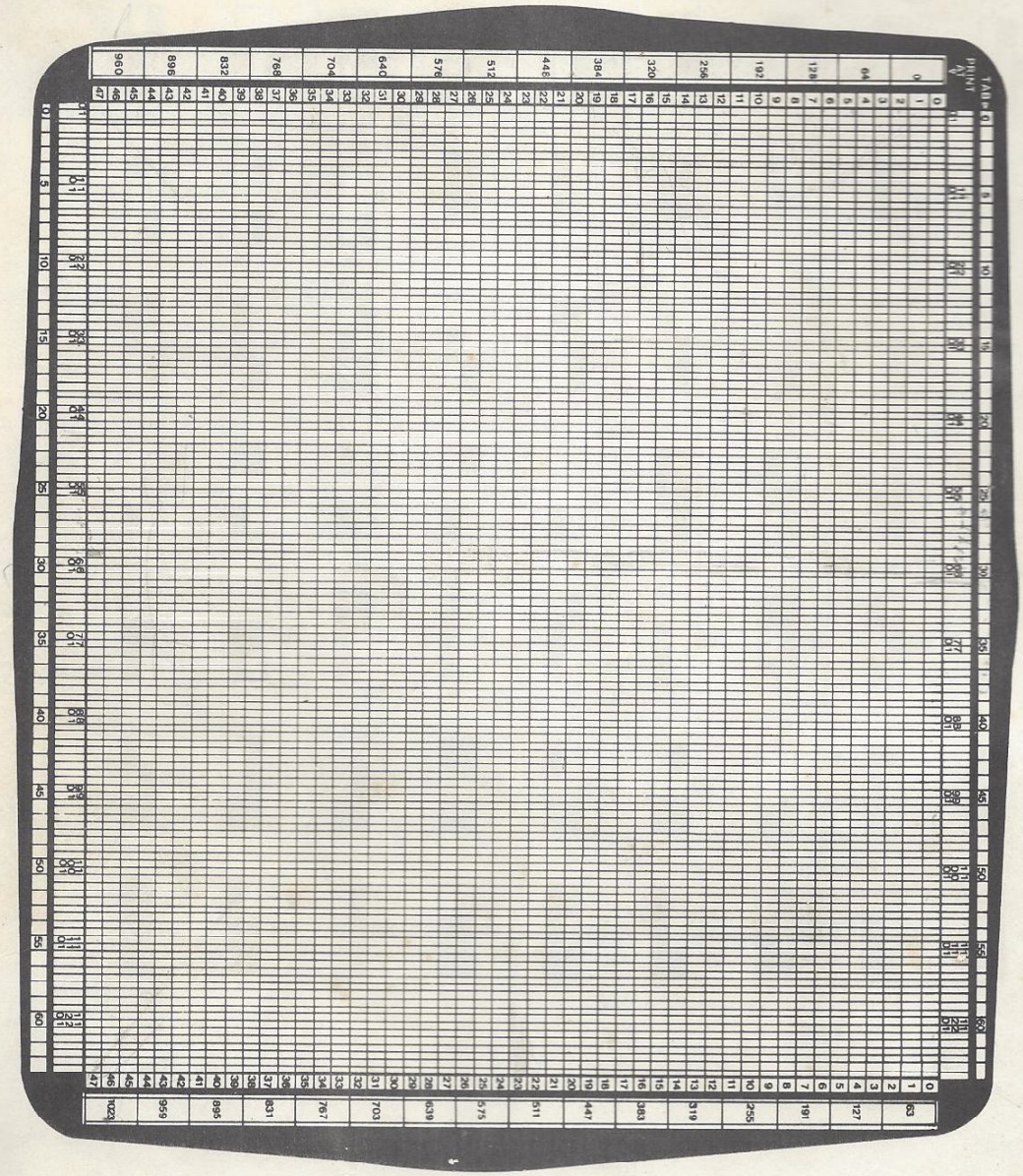
Caracteres especiais (0 a 31 e de 192 a 255)



Caracteres gráficos (129 a 191)



Lay-out da tela (endereços de 15360 a 16383)



Apêndice D
Códigos Internos das palavras chave em Basic

São os seguintes os códigos que o computador usa para armazenar as palavras-chaves em BASIC.

Se você pegar (instrução PEEK) no buffer de programa (começa no endereço 17129 em decimal) você encontrará seu programa armazenado com estes códigos:

Código Decimal	*Palavra Chave BASIC*	*Código Decimal*	*Palavra Chave BASIC*
129	FOR	153	DEFINT
130	RESET	154	DEFSNG
131	SET	155	DEFDBL
132	CLS	156	LINE
133	CMD	157	EDIT
134	RANDOM	158	ERROR
135	NEXT	159	RESUME
136	DATA	160	OUT
137	INPUT	161	ON
138	DIM	162	OPEN
139	READ	163	FIELD
140	LET	164	GET
141	GOTO	165	PUT
142	RUN	166	CLOSE
143	IF	167	LOAD
144	RESTORE	168	MERGE
145	GOSUB	169	NAME
146	RETURN	170	KILL
147	REM	171	LSET
148	STOP	172	RSET
149	ELSE	173	SAVE
150	TRON	174	SYSTEM
151	TROFF	175	LPRINT
152	DEFSTR	176	DEF

Código Decimal	*Palavra Chave BASIC*	*Código Decimal*	*Palavra Chave BASIC*
177	POKE	214	<
178	PRINT	215	SGN
179	CONT	216	INT
180	LIST	217	ABS
181	LLIST	218	FRE
182	DELETE	219	INP
183	AUTO	220	POS
184	CLEAR	221	SQR
185	CLOAD	222	RND
186	CSAVE	223	LOG
187	NEW	243	EXP
188	TAB	225	COS
189	TO	226	SIN
190	FN	227	TAN
191	USING	228	ATN
192	VARPTR	229	PEEK
193	USR	230	CVI
194	ERL	231	CVS
195	ERR	232	CVD
196	STRING$	233	EDF
197	INSTR	234	LOC
198	POINT	235	LOF
199	TIME$	236	MKI$
200	MEM	237	MKS$
201	INKEY$	238	MKD$
202	THEN	239	CINT
203	NOT	240	CSNG
204	STEP	241	CDBL
205	+	242	FIX
206	—	243	LEN
207	*	244	STR$
208	/	245	VAL
209		246	ASC
210	AND	247	CHR$
211	OR	248	LEFT$
212	>	249	RIGHT$
213	=	250	MID$

Apêndice E
Funções derivadas

Função/Expressão em relação às funções Basic (X em radianos)

Secante
SEC(X) = 1/COS(X)

Cossecante
CSC(X) = 1/SIN(X)

Cotangente
COT(X) = 1/TAN(X)

Arco-seno
ARCSIN(X) = ATN(X/SQR(—X.X + 1))

Arco-cosseno
ARCCOS(X) = —ATN(X/SQR(—X.X = 1)) + 1.5708

Arco-secante
ARCSEC(X) = ATN(SQR(X.X—1)) + (SG N(X)—1)*1.5708

Arco-cossecante
ARCCSC(X) = ATN(1/SQR(X.X1)) + (SGN(X)—1)*1.5708

Arco-cotangente
ARCOT(X) = —ATN(X) + 1.5708

Seno hiperbólico
SINH(X) = (EXP(X)—EXP(—X))/2

Cosseno hiperbólico
COSH(X) = (EXP(X) + EXP(—X))/2

Tangente hiperbólica
TAN(X) = —EXP(—X)/(EXP(X) + EXP(—X))*2 + 1

Secante hiperbólica
SECH(X) = 2/(EXP(X) + EXP(—X))

Cossecante hiperbólica
CSCH(X) = 2/(EXP(X)—EXP(—X))

Cotangente hiperbólica
COTH(X) = EXP(—X)EXP(X) — EXP(—X))*2 + 1

Arco-seno hiperbólico
ARGSINH(X) = LOG(X + SQR(X*X + 1))

Arco-cosseno hiperbólico
ARGCOSH(X) = LOG(X + SQR(X*X—1))

Arco-tangente hiperbólica
ARGTANH(X) = LOG((1 + X)/(1—X))/2

Arco-secante hiperbólica
ARGSECH(X) = LOG((SQR(—X*X + 1) + 1)/X)

Arco-cossecante hiperbólica
ARGCSCH(X) = LOG((SGN(X)*SQR(X*X + 1) + 1)/X)

Arco-cotangente hiperbólica
ARGCOTH(X) = LOG(X + 1)/(X—1)/2)

Faixas válidas de entrada

Arco-seno
—1<X<1

Arco-cosseno
—1<X<1

Arco-secante
X<—1,X>1

Arco-cossecante
X<—1,X>1

Arco-cosseno hiperb.
X>1

Arco-tangente hiperb.
X*X<1

Arco-secante hiperb.
0<X<1

Arco-cossecante hiperb.
X<>0

Arco-cotangente hiperb.
X * X > 1

Certos valores são matematicamente indefinidos, porém, nossas funções podem calcular valores inválidos:

TAN e SEC de 90 e 270 graus
COT e CSC de 0 e 180 graus

Por exemplo. TAN (1.5708) é calculado mas TAN (90*0.01745329) resulta em um erro de DIVISÃO POR ZERO;90*0.01745329 = 1.5708.

ARCSIN(—1) = —PI/2
ARCSIN(1) = PI/2
ARCCOS(—1) = PI
ARCCOS(1) = 0
ARCESEC(—1) = —PI
ARCSEG(1) = 0
ARCCSC(—1) = —PI/2
ARCCS(1) = PI/2

Note que as informações acima podem estar incompletas.

Apêndice F
Conversão de base

Dec.	*Hex.*	*Binário*
0	00	00000000
1	01	00000001
2	02	00000010
3	03	00000011
4	04	00000100
5	05	00000101
6	06	00000110
7	07	00000111
8	08	00001000
9	09	00001001
10	0A	00001010
11	0B	00001011
12	0C	00001100
13	0D	00001101
14	0E	00001110
15	0F	00001111
16	10	00010000
17	11	00010001
18	12	00010010
19	13	00010011
20	14	00010100
21	15	00010101
22	16	00010110
23	17	00010111
24	18	00011000
25	19	00011001
26	1A	00011010
27	1B	00011011
28	1C	00011100
29	1D	00011101
30	1E	00011110

Dec.	*Hex.*	*Binário*
31	1F	00011111
32	20	00100000
33	21	00100001
34	22	00100010
35	23	00100011
36	24	00100100
37	25	00100101
38	26	00100110
39	27	00100111
40	28	00101000
41	29	00101001
42	2A	00101010
43	2B	00101011
44	2C	00101100
45	2D	00101101
46	2E	00101110
47	2F	00101111
48	30	00110000
49	31	00110001
50	32	00110010
51	33	00110011
52	34	00110100
53	35	00110101
54	36	00110110
55	37	00110111
56	38	00111000
57	39	00111001
58	3A	00111010
59	3B	00111011
60	3C	00111100
61	3D	00111101

Dec.	*Hex.*	*Binário*
62	3E	00111110
63	3F	00111111
64	40	01000000
65	41	01000001
66	42	01000010
67	43	01000011
68	44	01000100
69	45	01000101
70	46	01000110
71	47	01000111
72	48	01001000
73	49	01001001
74	4A	01001010
75	4B	01001011
76	4C	01001100
77	4D	01001101
78	4E	01001110
79	4F	01001111
80	50	01010000
81	51	01010001
82	52	01010010
83	53	01010011
84	54	01010100
85	55	01010101
86	56	01010110
87	57	01010111
88	58	01011000
89	59	01011001
90	5A	01011010
91	5B	01011011
92	5C	01011100
93	5D	01011101
94	5E	01011110
95	5F	01011111
96	60	01100000
97	61	01100001
98	62	01100010
99	63	01100011
100	64	01100100
101	65	01100101
102	66	01100110

Dec.	*Hex.*	*Binário*
103	67	01100111
104	68	01101000
105	69	01101001
106	6A	01101010
107	6B	01101011
108	6C	01101100
109	6D	01101101
110	6E	01101110
111	6F	01101111
112	70	01110000
113	71	01110001
114	72	01110010
115	73	01110011
116	74	01110100
117	75	01110101
118	76	01110110
119	77	01110111
120	78	01111000
121	79	01111001
122	7A	01111010
123	7B	01111011
124	7C	01111100
125	7D	01111101
126	7E	01111110
127	7F	01111111
128	80	10000000
129	81	10000001
130	82	10000010
131	83	10000011
132	84	10000100
133	85	10000101
134	86	10000110
135	87	10000111
136	88	10001000
137	89	10001001
138	8A	10001010
139	8B	10001011
140	8C	10001100
141	8D	10001101
142	8E	10001110
143	8F	10001111

Dec.	*Hex.*	*Binário*
144	90	10010000
145	91	10010001
146	92	10010010
147	93	10010011
148	94	10010100
149	95	10010101
150	96	10010110
151	97	10010111
152	98	10011000
153	99	10011001
154	9A	10011010
155	9B	10011011
156	9C	10011100
157	9D	10011101
158	9E	10011110
159	9F	10011111
160	A0	10100000
161	A1	10100001
162	A2	10100010
163	A3	10100011
164	A4	10100100
165	A5	10100101
166	A6	10100110
167	A7	10100111
168	A8	10101000
169	A9	10101001
170	AA	10101010
171	AB	10101011
172	AC	10101100
173	AD	10101101
174	AE	10101110
175	AF	10101111
176	B0	10110000
177	B1	10110001
178	B2	10110010
179	B3	10110011
180	B4	10110111
181	B5	10110101
182	B6	10110110
183	B7	10110111
184	B8	10111000

Dec.	*Hex.*	*Binário*
185	B9	10111001
186	BA	10111010
187	BB	10111011
188	BC	10111100
189	BD	10111101
190	BE	10111110
191	BF	10111111
192	C0	11000000
193	C1	11000001
194	C2	11000010
195	C3	11000011
196	C4	11000100
197	C5	11000101
198	C6	11000110
199	C7	11000111
200	C8	11001000
201	C9	11001001
202	CA	11001010
203	CB	11001011
204	CC	11001100
205	CD	11001101
206	CE	11001110
207	CF	11001111
208	DO	11010000
209	DI	11010001
210	D2	11010010
211	D3	11010011
212	D4	11010100
213	D5	11010101
214	D6	11010110
215	D7	11010111
216	D8	11011000
217	D9	11011001
218	DA	11011010
219	DB	11011011
220	DC	11011100
221	DD	11011101
222	DE	11011110
223	DF	11011111
224	EO	11100000
225	E1	11100001

Dec.	*Hex.*	*Binário*
226	E2	11100010
227	E3	11100011
228	E4	11100100
229	E5	11100101
230	E6	11100110
231	E7	11100111
232	E8	11101000
233	E9	11101001
234	EA	11101010
235	EB	11101011
236	EC	11101100
237	ED	11101101
238	EE	11101110
239	EF	11101111
240	FO	11110000

Dec.	*Hex.*	*Binário*
241	F1	11110001
242	F2	11110010
243	F3	11110011
244	F4	11110100
245	F5	11110101
246	F6	11110110
247	F7	11110111
248	F8	11111000
249	F9	11111001
250	FA	11111010
251	FB	11111011
252	FC	11111100
253	FD	11111101
254	FE	11111110
255	FF	11111111

Apêndice G
Monitor-residente

O CP 300 além da linguagem BASIC, possui a capacidade de operar, através de seu Monitor Residente, em linguagem de máquina (Z-80).

Com o Monitor Residente pode-se verificar as condições reais do microprocessador, verificando seu conteúdo, conferindo os programas na memória. Para tanto, proceda da forma descrita a seguir.

Procedimento

CP 300 cassete:
Ao ligar o equipamento pressione o botão RESET
`BASIC (S ou N)?`
Para acionar o monitor digite "N" e o sistema passará o controle para o monitor.
Quando o controle é passado ao monitor aparecerá a seguinte mensagem:
`MONITOR VERSAO 1.1  1982`

Comandos do Monitor

D

Lista memória (Hexa e ASCII)

Sintaxe:

D endereço inicial em hexa, número de bytes em hexa ENTER.

Nota:

O endereço inicial deve ser qualquer endereço válido:
Para 16K temos de 0 até 7FFF
Para 32K temos de 0 até BFFF
Para 48K temos de 0 até FFFF
O número de bytes indica quantos bytes deverão ser mostrados.

Exemplo:

```
D 6CD0,B0 ENTER
```

Efeito: Lista 176(BO) bytes a partir do endereço 6CD0 da memória. Lista do endereço 6CD0 até D7F.

S

Substitui o conteúdo do endereço da memória fornecido.

Sintaxe:

S endereço a ser alterado ENTER

Será então apresentado o seguinte:
Endereço a ser alterado Conteúdo Sub-comando.
O computador aguarda um dos sub-comandos

ENTER Para sair do comando sem alterar o conteúdo
/ Para passar para o endereço seguinte
— Voltar ao endereço anterior.
xx Onde *xx* é o novo conteúdo do endereço formado por dois dígitos (hexadecimal).

Exemplos:

```
S 64D3 ENTER
```

Endereço	*Conteúdo*	*Sub-comando*	*Efeito*
64D3	FC	/	Passa para endereço seguinte
64D4	7F	4D	Alteração do conteúdo
64D5	3C	—	Retorna ao endereço anterior
64D4	4D	ENTER	Saída do sub-comando

R

Altera os registradores do Z-80

Sintaxe: R

Ao ser dado do comando será visualizado os registradores e seus respectivos conteúdos da seguinte forma:

```
I    AF    BC    DE    HL    AF'   BC'   DE'
```

conteúdo:

```
(xx)(xxxx)(xxxx)(xxxx)(xxxx)(xxxx)(xxxx)(xxxx)
HL'   IY    IX    SP    PC
(xxxx)(xxxx)(xxxx)(xxxx)(xxxx)
```

Onde *xx* representa o conteúdo em hexadecimal dos pares de registradores.

Para alterar um dos registradores deve ser dado a sigla do mesmo seguida de um espaço. Será então apresentado o seguinte formato:

sigla do registrador ESPAÇO **conteúdo sub-comando**

Os sub-comandos aceitos após R são os seguintes:

ESPAÇO

Para alterar o conteúdo anterior do registrador.
Deve-se digitar o novo conteúdo após a barra de espaço.
O computador só aceita uma alteração por comando R.

CLEAR

Para saltar para o próximo registrador.

ENTER

Para sair do comando.

Nota:

Registradores de 8 bits são apresentados separadamente e não aos pares. Seu conteúdo é 2 dígitos hexadecimais.
Os registradores de 16 bits são apresentados em siglas de dois caracteres. Seu conteúdo é 4 dígitos hexadecimais.

Exemplos:

```
R
I   AF   BC   DE   HL   AF'  BC'  DE'  HL'  IY   IX   SP   PC
3D 4D53 327A D8F0 3CFC 54D9 A9DF 5F3C 7524 6328 632A 403B 62B7
```

Registrador	*Conteúdo*	*Subcomando*	*Efeito*
A ESPAÇO	4D		Salta para o próximo registrador
B	53	ENTER 32	Altera conteúdo do registrador F e salta para o próximo.
C	7A	CLEAR	Salta para o próximo registrador.
D	D8	ESPAÇO 6D ENTER	Altera o conteúdo de D e sai do comando.

J

Salta para o endereço especificado e inicia a execução.

Sintaxe:

J endereço (hexa) de início, endereço (hexa) do break point ENTER.

Ao ser dado comando haverá uma execução que começa no endereço de início e terminará ao encontrar o break point. Cuidado ao definir o break point, pois se colocado em uma área de dados ou em um ponto que não seja executável a operação não passará por ele.

Exemplo:

```
J8000 ENTER
```

Inicia a execução em 8000 sem haver um break point.

```
J8000,803B ENTER
```

Inicia a execução em 8000 e finaliza ao encontrar o break point no endereço 803B.

G

Continua a execução a partir do endereço do último break point.

Sintaxe:

G endereço do novo break point ENTER.
Recomeçar a instrução do endereço do último break point e todas até o próximo.

Exemplo:

```
G813D ENTER
```

Continúa a execução que foi interrompida em 803B (do break point anterior) até o endereço 813D que é o novo break point.

```
G ENTER
```

Prossegue a execução que foi interrompida pelo break point em 813D, sem parar em outro break point.

T

Transfere o conteúdo da RAM para o cassete (veja exemplo)
Cod. Baud Rate

Sintaxe:

T Código Baud Rate
Nome do Programa ENTER
Endereço inicial (hexa) ENTER
Número de bytes ENTER

Nota:

Sempre que o cassete for usado a mensagem "Cass?" irá aparecer devendo então ser informada a taxa de transferência.
A para uma baud rate de 1500 baud
B para uma baud rate de 500 baud

Exemplo:

```
T
Cass? (A ou B)
MON ENTER
8000 ENTER
8400 ENTER
```

Salvará o conteúdo da RAM do endereço 8000 até 83F F(1K) no arquivo de nome MON na fita.

C

Carrega um programa da fita para a memória.

Sintaxe:

C
Código do baud rate
Nome do arquivo ENTER

O programa será procurado na fita cassete e quando for encontrado será carregado na memória. Durante a carga será visualizado os dois asteriscos no canto superior direito, um dos quais é intermitente. Após a carga do programa será visualizado na tela o endereço de início do programa.

A operação poderá ser abortada pela tecla BREAK.

Exemplo:

```
C
Cass?(responda A ou B)
NOME
MON ENTER
8000
```

NOTAS GERAIS

a) Programas do tipo SYSTEM são incompatíveis com o monitor, isto é, fitas SYSTEM não são carregadas pelo monitor é vice-versa.

b) O RESET não apaga a memória para o monitor, desde que:
   1) Não se pressione RESET por muito tempo;
   2) Ao sair do monitor com RESET se entre novamente no monitor, sem "BOOT" do DOS 300 e sem nenhuma entrada no BASIC-RESIDENTE. (Responda sempre não para BASIC(S/N)?).

c) A tecla . apesar de ativa não é interpretada pelo monitor. Em caso de erro digite os bytes na seqüência desejada.

Exemplo:

Examinar a memória de 6250 até 6260
D6246250,0010

Serão considerados apenas os últimos bytes, sendo desprezados os excedentes à esquerda.

d) Nomes de arquivos podem ter quantos caracteres quantos se desejar (não há limite), sendo que para a carga do arquivo será exigido o mesmo nome.

e) Se alguma rotina criada ou carregada pelo monitor for ser utilizada pelo Basic, pressione RESET para sair do monitor (mantenha essa tecla pressionada se o sistema tiver drive de disquete), chame Basic (BASIC S/N?) e proteja a área de memória onde se encontra a rotina para que o BASIC não a sobreponha. Mem. usada? (máximo 17686 (decimal)).
   A rotina criada pelo monitor deverá estar após o endereço 45FF (hexa) para não ser encoberta pelo Basic.

f) Se algum engano for cometido em qualquer comando (exceto J, G ou T), poderá ser interrompido pressionando-se a tecla BREAK, voltando o monitor ao seu início sem alteração dos registradores ou da memória.

g) O retorno de um programa ao monitor é feito executando-se um "jump" para o endereço 4300H (C30043).

h) O registrador SP (stack pointer) não é iniciado pelo monitor devendo o usuário iniciá-lo para uma área de RAM reservada para o "stack" antes de qualquer comando, toda vez que entrar no monitor.

Apêndice H
Glossário

A

Acesso — maneira pela qual o computador obtém ou atinge um conjunto de dados.

Acumulador — registrador para armazenamento de dados durante a operação do computador. Endereços e informações podem ser guardados temporariamente nesse dispositivo. Certos tipos são colocados à disposição do usuário, enquanto outros ficam reservados para uso interno do processador.

Aleatorizar — dispor dados de forma que não respeitem nenhuma seqüência ou relação pré-determinada. Dispor dados de forma aleatória.

Alfanumérico — qualquer um dos caracteres usados nas linguagens de computação. Pode ser uma letra, um número ou um símbolo padronizado.

ALGOL — abreviação de ALGOrithmic Language (linguagem algorítmica). Linguagem aplicação geral, mas com ênfase nas aplicações de cunho numérico. Linguagem algorítmica de orientação científica.

Algoritmo — processo ou conjunto de regras fixas que, numa seqüência passo a passo, leva a um resultado determinado.

Alimentar — suprir o computador com os dados que deve processar. Pode significar, ainda, o ato de fornecer ao equipamento energia elétrica.

Alocação — preencher àreas específicas de memória com blocos de dados determinados. Reservar áreas de armazenamento para as rotinas e sub-rotinas, fixando assim os valores absolutos de todos os endereços simbólicos.

ALU — Arithmetic Logic Unit (unidade lógico-aritmética). Parte do hardware do computador onde são executadas as operações lógicas e aritméticas.

Analógico — representação de dados de forma contínua normalmente apresentados como grandezas físicas, como tensão, corrente, rotação, etc.

APL — abreviatura de A Programming Language (uma linguagem de programação). Linguagem com características de notação, ainda não completamente implementada, dedicada especialmente a operações matemáticas, como valores matrizes, etc.

Aquisição de dados — captação e reunião de dados, a fim de que possam ser aproveitados pelo computador.

Arquivo — conjunto de dados reunidos, formando uma unidade. Acumulação de informação na memória do computador.

Array — o mesmo que arranjo ou distribuição. Lista de dados indexados, que podem ser acessados diretamente.

ASCII — abreviação de American Standard Code for Information Interchange (código padrão americano para intercâmbio de informações). Codificação que emprega palavras de 8 bits (incluindo o bit verificador de paridade), adotada com o objetivo de facilitar a troca de dados entre sistemas de processamento ou entre o computador e seus periféricos.

Assembler — pode ser traduzido como "programa de montagem". Consiste em um programa encarregado de atuar sobre dados simbólicos para produzir, a partir deles, instruções de máquina capazes de efetuar determinadas funções, tais como: tradução de códigos de operação em instruções de computador, determinação de locais de memória para instruções sucessivas ou computação de endereços absolutos a partir de endereços simbólicos.

Assembly — o mesmo que "montagem". Tradução de programa fonte capaz de ser executado diretamente pelo microprocessador.

B

Bandeira — Bit de informação acrescentado a um caractere ou palavra, para indicar a fronteira de um campo de dados. Caractere que sinaliza a ocorrência de alguma condição, tal como o fim de uma palavra.

Barra de dados (data bus) — meio físico de transporte de dados entre o microprocessador e os demais elementos que compõem o computador (memórias, registradoras, etc.)

Barra de endereços — (address bus) — meio físico de informação sobre as coordenadas ("endereços") que os dados devem ocupar na memória RAM.

BASIC — Abreviatura de Beginner's All-purpose Symbolic Instruction Code (código simbólico de instrução, universal para principiantes). Linguagem de aplicação geral, desenvolvida tanto em programas comerciais como científicos, mas adapta-se melhor a estes.

Baud — unidade de velocidade de sinais, igual ao número de dados por segundo. Deriva do nome Baudot, que deu origem também ao código de mesmo nome.

BCD — abreviação de Bynary-Coded Decimal (decimal codificado em binário). Representação numérica pela qual os dígitos decimais são apresentados sob a forma de números binários. Exemplo: o número 13, em decimal, tem como equivalente 0001 0011, em BCD.

Biblioteca de programas — conjunto organizado de programas, rotinas ou de software específico ou genérico, para uso em um determinado sistema.

Binário — sistema numérico que emprega a base 2, ao invés da base 10 normal, dispondo assim dos dígitos 0 e 1, apenas para formar números.

Bit — abreviatura de BInary digiT (digíto binário). Caractere de um número binário. Unidade de informação ou de capacidade de memória.

Bit mais (ou menos) significativo — bit que contribui com a maior (ou menor) porção do valor de uma palavra. O bit mais à esquerda (ou direita) de um número binário.

Busca — processo de recolher instruções que serão aproveitadas pela CPU, armazenando-as num registrador adequado.

Branch — ver Ramo ou ramificação.

Byte — número de bits que o computador considera como uma unidade. A menor unidade endererável na memória de um computador. Normalmente, consiste de oito bits de dados e um bit de paridade.

C

Cadeia, cordão — seqüência interligada de caracteres, palavras ou outros elementos.

Caractere — símbolo utilizado na representação gráfica de dados.

Campo (field) — conjunto de um ou mais caracteres tratados como um todo. Área especificada de um registro usada para uma categoria específica de dados.

Carregar — preencher a memória interna de um computador com informações provindas de sistemas externos ou auxiliares de armazenagem.

Chamada de sub-rotina — passagem do controle do computador do programa principal para uma determinada sub-rotina.

Clock — dispositivo interno do computador encarregado de sincronizar os vários estágios do sistema. Também chamado de "relógio".

COBOL — abreviatura de COmmon Business Oriented Language (linguagem de orientação comercial). Trata-se de uma linguagem especialmente projetada para processamento de dados da área comercial, definida e desenvolvida por um comitê americano de fabricantes e usuários de computadores.

Código Baudot — codificação padronizada de dados para teletipo, em 5 canais, consistindo de um impulso de partida, 5 impulsos de caracteres, todos de igual extensão, e um impulso de parada.

Código objeto — codificação produzida por um compilador ou um assemble especial para ser executada num computador.

Código operacional — conjunto de símbolos que designam a execução de uma operação básica do computador. Parte da instrução que determina a operação lógica, aritmética ou de transferência que deve ser executada.

Comando — Sinal ou conjunto de sinais elétricos que dá início, encerra ou dá seqüência a uma operação. Parte da instrução que especifica a operação a ser efetuada. (Obs.: é incorreto utilizar o termo "comando" como sinônimo de "instrução").

Compilação — conversão de um programa fonte em rotinas específicas, escritas em linguagem de máquina. Desenvolver ou produzir um programa ordenado lógica ou seqüencialmente, em linguagem de máquina, a partir de uma série de códigos operacionais mnemônicos ou simbólicos.

Compilador (compiler) — rotina de montagem de programas, capaz de produzir um programa específico para um determinado fim, ao estabelecer o sentido inerente a elementos de informação. Programa de computador ainda mais poderoso que o assembler; além da função de tradução, é também capaz de substituir certos itens de entrada por séries de instruções, normalmente chamadas de sub-rotinas.

Compressão de dados — série de técnicas usadas para a redução de espaço, largura de banda, custo, transmissão, e armazenagem de dados. Tais técnicas visam a eliminação de repetições, a remoção de dados irrelevantes e o emprego de processos especiais de codificação. Também chamado de compactação de dados.

Comprimento de palavra — medida de extensão de uma palavra de computador, normalmente dividida em sílabas, bytes, bits, etc.

Computador — dispositivo capaz de aceitar informações, aplicar processos pré-determinados às mesmas e fornecer os resultados desses processos. Consiste, normalmente, de dispositivos de entrada e saída (VO), unidade de armazenagem, de cálculos aritméticos e lógicos, além de uma unidade de controle (CPU).

Computer-aided design — ver Projeto auxiliado por computador.

Contador de programa — registrador que retém a identificação da palavra de instrução prestes a ser executada. Também chamado de registrador de controle.

CPU — ver Unidade Central de Processamento.

Cross-assembler — tradutor de linguagem simbólica que "roda" em um tipo de computador, a fim de produzir linguagem de máquina para outro tipo de computador. Pode ser chamado, também, de assembler cruzado.

D

Dados — nome genérico dado aos elementos básicos de informação que podem ser processados ou produzidos por um computador.

Dados formatados — arranjo pré-determinado de caracteres, campos, linhas, pontuação, números de páginas, etc.

Debug — processo de localização e correção de quaisquer erros num programa de computador. Teste de programa, a fim de assegurar de que funciona corretamente.

Decisão — processo de verificação, normalmente por comparação, da existência ou não de uma certa condição, como resultado de uma ação alternativa. Operação efetuada pelo computador, de determinar a existência de alguma relação entre palavras guardadas na memória ou em registrador e a partir daí, seguir caminhos alternativos.

Decodificação — execução das operações internas pelas quais o computador determina o sentido do código operacional de uma instrução.

Delete — remover, eliminar dados, como por exemplo, cancelar um registro de arquivo principal.

Demodulação — processo de se obter o sinal original a partir de uma portadora modulada. Técnica normalmente utilizada para tornar os sinais de comunicação compatíveis com os sinais de máquinas comerciais.

Digitação — normalmente, a introdução de dados num computador através de teclado apropriado.

Digital — referente à utilização de números discretos integrais, numa determinada base, para representar todas as quantidades que ocorrem num problema ou cálculo.

Dígito — um dos "n" símbolos de valor integral, variando de 0 a n-1, pertence a um sistema de numeração de base n (os dígitos de 0 a 9 pertencem ao sistema decimal, enquanto 0 e 1 fazem parte do sistema binário, por exemplo).

E

Editar — processo de rearranjar dados ou informações, que pode consistir da remoção de dados indesejáveis, seleção de dados importantes, aplicação de técnicas de formatação, inserção de símbolos, aplicação de processos padronizados e teste de dados.

Enable — ver Habilitar.

Endereço — identificação do local, registrador ou unidade em que se encontram informações armazenadas.

Entrada/saída (E/S) — termo genérico usado para o equipamento com que o computador se comunica com o exterior. Os dados envolvidos em tal comunicação.

Evento — Ocasião ou ação que faz os dados afetarem o conteúdo dos arquivos.

F

Feed — ver Alimentar.

Fetch — ver Busca.

File — ver Arquivo.

Fim de arquivo (end of file) — término ou ponto final de uma certa quantidade de dados; indicações especiais de fim de arquivo, demarcam esse ponto.

Flag — ver Bandeira.

Fluxo — termo genérico que indica uma seqüência de eventos.

Fluxograma — representação gráfica de uma seqüência de operações, num programa, empregando símbolos para representar as várias operações, tais como comparação, salto, leitura, escrita, etc.

FORTRAN — abreviação de FORmula TRANslator (conversor de fórmulas). Sistema de programação, incluindo uma linguagem e um compilador, que permite escrever programas com notação matemática, foi desenvolvido originalmente pela IBM, destinado a fins científicos, mas atualmente presta-se também a resolução de inúmeros problemas comerciais.

Frase — na programação, consiste em uma expressão ou uma instrução generalizada em linguagem fonte.

Firmware — programas ou instruções armazenados em memória ROM. O firmware é análogo ao software, sob forma de hardware.

H

Habilitar — permitir o início ou seqüência de operação de um determinado dispositivo.

Hardware — conjunto dos elementos mecânicos, elétricos, magnéticos e eletrônicos que compõem o computador.

Hexadecimal — sistema de numeração que emprega a base 16. Nesse sistema, os dígitos são representados pelos algarismos de 0 a 9 e pelas letras de A a F (no lugar dos números de 10 a 15).

I

Impressora — dispositivo que atua como periférico do computador, fornecendo dados impressos, sob a forma de uma listagem.

Indexação — modificação de endereços, normalmente efetuado pelos chamados registradores de indexação (ou indexadores).

Índice — tabela de palavras ou campos que contém endereços de dados localizados nos arquivos.

Indicador — registrador da CPU que contém os endereços de memória

Interface — ponto ou dispositivo de contato entre dois elementos, no interior do computador, ou entre estes dispositivos externos.

Interrupção — quebra no fluxo normal de um sistema ou uma rotina, de forma que possa ser retomado mais tarde. Sinal de controle que desvia a atenção do computador, quando este está atarefado com o programa principal, para um endereço específico, diretamente relacionado com o tipo de interrupção ocorrida.

Instrução — passo codificado de programa que diz ao computador o que fazer a cada operação. Conjunto de caracteres que, juntamente com endereço, define uma operação e faz com que o computador aja como pedido sobre as quantidades indicadas.

I/O (input/output) — ver Entrada/saída.

J

JUMP — ver Salto

L

Laço — um conjunto de instruções que deve ser repetido vários vezes.

Linguagem — em computação, um meio de comunicação entre o homem e máquina, ou de partes do computador entre si ou, ainda, entre computadores.

Linguagem de alto nível — uma linguagem em que os comandos são passíveis de serem entendidos pelo operador, sem que o mesmo tenha que conhecer, necessariamente, a arquitetura do computador ou sua linguagem de máquina.

Linguagem de máquina — conjunto de símbolos, caracteres ou sinais e suas regras de combinação, para formar instruções para um determinado computador.

Listagem — impressão de um conjunto de dados, instruções ou resultados de um programa.

Load — ver Carregar.

Loader — rotina de serviço, cuja finalidade é ler programas para a memória central, para fins de execução.

Loop — ver Laço.

LSB — ver Bit menos significativo.

M

Macro-instrução — Instrução de uma linguagem de alto nível, formada por várias instru-

ções em linguagem de máquina. Ou então uma instrução de um microprocessador formada por várias micro-instruções do mesmo.

Mainframe — conjunto formado pela CPU, unidade de entrada/saída e memória. É o computador, menos periféricos.

Memória — Dispositivo ou conjunto de dispositivos que armazena um dado ou conjunto de dados para aproveitamento posterior.

Memória dinâmica — Memória em que os dados precisam ser continuamente realimentados para que não se percam.

Memória estática — Memória que não precisa de realimentação para manter seus dados.

Memória rascunho — Memória em que os dados são resultados parciais de uma operação qualquer, e posteriormente poderão ser modificados. Ao se terminar a operação, o resultado final será transferido para outra posição de memória, caso seja necessária a sua preservação para uso posterior ou para uma saída.

Memória Virtual — Memória com capacidade de usar um tipo de algoritmo de paginação ou segmentação. Por este meio, pode-se simular uma memória maior que aquela que realmente existe.

Memory Dump — Uma listagem do conteúdo da memória ou parte dele, visando a pesquisa de erros ou defeitos no computador.

Microcomputador — Computador cuja CPU é formada por um microprocessador.

Micro-instrução — Uma instrução em linguagem de máquina, que é usada para formar uma instrução em linguagem de alto nível. Em uma unidade de controle microprogramada, cada uma das pequenas instruções que forma uma instrução em linguagem de máquina.

Microprocessador — a unidade central de processamento de um microcomputador, formada por um único CI.

Microprograma — programa em linguagem de máquina que forma uma instrução em uma linguagem de alto nível (Macro-instrução). Em uma unidade de controle microprogramada, um microprograma é um programa formado por micro-instruções, que forma uma instrução em linguagem de máquina.

Modem (modelador/demodulador) — tem a função, em teleprocessamento, de codificar e decodificar os dados em função de sua transmissão por via de cabos telefônicos.

MSB — Ver dígito mais significativo.

N

Nested subroutine — Ver Sub-rotina aninhada.

O

Octal — número na base oito.

On line (em linha)— equipamento ou dispositivo sob controle direto da CPU.

Op code — Ver código operacional.

P

Página — num agrupamento de memórias, cada uma dessas memórias forma uma página. Num microcomputador de oito bits, uma página pode ser constituída por uma memória de 2 8 = 256 bits.

Palavra — um grupo de caracteres ocupando uma localização da memória de um computador. Para a unidade de controle, uma palavra é uma instrução e, para a unidade lógico-aritmética, uma quantidade.

Paridade — um método de checagem da precisão de um número binário. Um bit adicional, chamado bit de paridade, é acrescentado ao número. Se a paridade par for usada, a soma de todos os bits 1 do número, mais o bit de paridade, é sempre par; se for usada a paridade impar, esta soma sempre é ímpar; ocorrendo resultado par, houve erro.

PASCAL — linguagem estruturada, baseada no ALGOL 60, destinada ao ensino de programação e uso geral em microcomputadores.

Passo de programa — operação numa rotina de um computador. Cada uma das instruções executadas num programa.

Pattern — modelo, padrão.

Periférico — unidade auxiliar que pode ser colocada sob controle do processador central, tal como unidades de discos, terminais de vídeo, impressoras, gravadores, etc.

Pilha — tipo de memória rascunho usada durante operações de expressões numéricas, em que o primeiro dado que entra é o último que sai.

Pista — parte da área gravada de um meio de armazenamento (fita, tambor, disco ou disquete), a que se tem acesso por meio de um determinado posicionamento da cabeça de leitura.

Pointer — ver indicador.

Polling (interrogação periódica) — técnica em que cada um dos terminais que compartilham uma linha de comunicação é periodicamente interrogado, para determinar se necessitam de serviços da CPU.

Porta (Port) — dispositivo que comanda a entrada e saída de um sistema. Em comunicação de dados, a parte de um processador de dados que é dedicada exclusivamente a um único canal de dados, com o fim de recebê-los e transmiti-los para um ou mais terminais remotos.

Print — instrução comum a várias linguagens, que coloca os dados em uma saída de forma legível ao operador humano. Esta saída pode ser uma folha impressa por impressora ou dados num terminal de vídeo.

Prioridade — gradação atribuída a uma tarefa que determina sua precedência quanto ao uso dos recursos do sistema.

Procedimento (procedure) — curso de ação adotado para a solução de um determinado problema.

Programa — seqüência de instruções que determinam ao computador o procedimento adotado para resolver determinados problema.

Programa fonte (source program) — programa em linguagem de máquina executável pelo processador. É criado a partir de um programa compilado.

Programa compilado — programa em códigos relocáveis traduzido a partir do programa fonte.

Projeto e Fabricação Auxiliados por Computador (CAD/CAM) — uso do computador em automação industrial, engenharia, biologia, estatística, etc., para agilizar a compilação e análise de dados, projeto físico de equipamentos, cálculo estrutural, e outras tarefas que envolvam um grande número de informações.

R

RAM — Memória de Acesso Selecionável. Memória em que se tem acesso a qualquer posição, tanto para escrita quanto para leitura.

Randomize — Ver Aleatorizar.

Ramo ou ramificação — desvio na seqüência normal do programa, por uma instrução de tipo

Read — instrução comum a várias linguagens, que indica ao computador que ele deve ler determinados dados vindos de um dispositivo de entrada (como em Cobol ou Fortran), ou de uma lista de dados dentro do próprio programa (como em Basic).

Reforço da memória — técnica usada em memória dinâmicas, em que o dado presente em cada uma das células é realimentado para evitar que, com o passar do tempo, ele se perca.

Refresh — ver Reforço de memória.

Registrados — parte interna de um microprocessador que armazena dados para serem processados pela CPU.

Registrador de deslocamento — registrador em que os dados podem ser deslocados, quer para a direita, quer para a esquerda, de forma serial.

Registrador de instruções — um dos registradores de uma CPU cuja função é armazenar a instrução, enquanto esta está sendo interpretada.

Relocar — deslocar uma rotina ou bloco de dados dentro da memória física do computador, realizando as devidas correções no endereçamento envolvido.

Reset — recolocar um dispositivo em sua condição inicial de operação. O mesmo que inicializar.

ROM — memória apenas de leitura. Memória que contém funções e rotinas permanentes, necessárias ao funcionamento do computador e que não podem ser alteradas

Rotina — conjunto de instruções dispostas em posição adequada que instrui o computador a realizar uma operação determinada.

Rotina de checagem (check routine) — rotina que tem por finalidade verificar se o computador está operando corretamente.

Run — instrução comum a várias linguagens, que indica ao computador para executar um determinado programa. O mesmo que rodar um programa.

S

Salto (jump) — tipo de instrução que ordena ao computador que se dirija a um ponto determinado do programa, que não seja a instrução seguinte na seqüência normal.
Scratchpad memory — ver Memória rascunho.
Serial — entrada ou saída em que os bits estão dispostos um após o outro, para facilitar a transmissão e recepção de dados.
Shift register — ver Registrador de deslocamento.
Sistema — conjunto de componentes ou equipamentos arranjados de forma a realizar uma ou várias ações determinadas.
Software — conjunto de programas, procedimentos e documentação relativo à operação de um sistema de processamento de dados.
Stack — ver Pilha.
Statement — ver Frase.
String — ver cadeia ou cordão.
Sub-rotina — rotina repetitiva. Em geral trata-se de um procedimento que é utilizado várias vezes dentro de um programa.
Sub-rotina aberta — procedimento que aparece em vários pontos de um mesmo programa.
Sub-rotina fechada — procedimento indexado. Toda vez que for necessário seu uso, será realizado um salto para esta sub-rotina e, ao terminar sua função, o processamento retorna ao endereço seguinte ao ponto de desvio.
Sub-rotina aninhada — sub-rotina que foi acionada por uma outra sub-rotina. O processador empilha os endereços de retorno de modo que, a medida que as sub-rotinas forem sendo concluídas, o processamento retorna na seqüência inversa das chamadas.

T

Teletipo (TTY, teletype) — equipamento tele-impressor usado em sistemas de comunicação. Podem ser usados também em transmissão de dados, como equipamento de entrada ou saída.
Tempo partilhado — meio de usar um sistema de computação, de forma a permitir executar e interagir vários programas de diferentes usuários ao mesmo tempo.
Tempo real — regime de trabalho de um computador ou controlador em que as respostas são dadas imediatamente após à entrada dos dados.
Terminal — dispositivo de entrada ou saída que permite a comunicação entre o usuário e a unidade central de processamento (CPU).
Time shared — ver Tempo partilhado.
Track — ver pista.
Trilha — ver pista.

U

Unidade Central de Processamento (UCP ou CPU) — parte de um computador que o controla e realiza as operações aritméticas e lógicas.

V

Variável em ponto flutuante — variável representada por números decimais, cujo ponto (que separa a parte inteira da fracionária) pode variar de posição ao ser realizada uma operação.

Variável em ponto fixo — variável inteira ou decimal, cujo número de casas decimais está pré-estabelecido.

Verificação de paridade — verificação de precisão de um número, visando detecção de erro, baseada no conceito de paridade.

W

Write — instrução comum a diversas linguagens, que indica ao computador que ele deve fornecer uma saída em uma forma legível, como um texto em uma impressora ou linhas em uma tela de vídeo.

Este livro foi impresso pela
artes gráficas guaru s/a.
Rod. Presidente Dutra, km 214
Fone: 913-1422 - Guarulhos