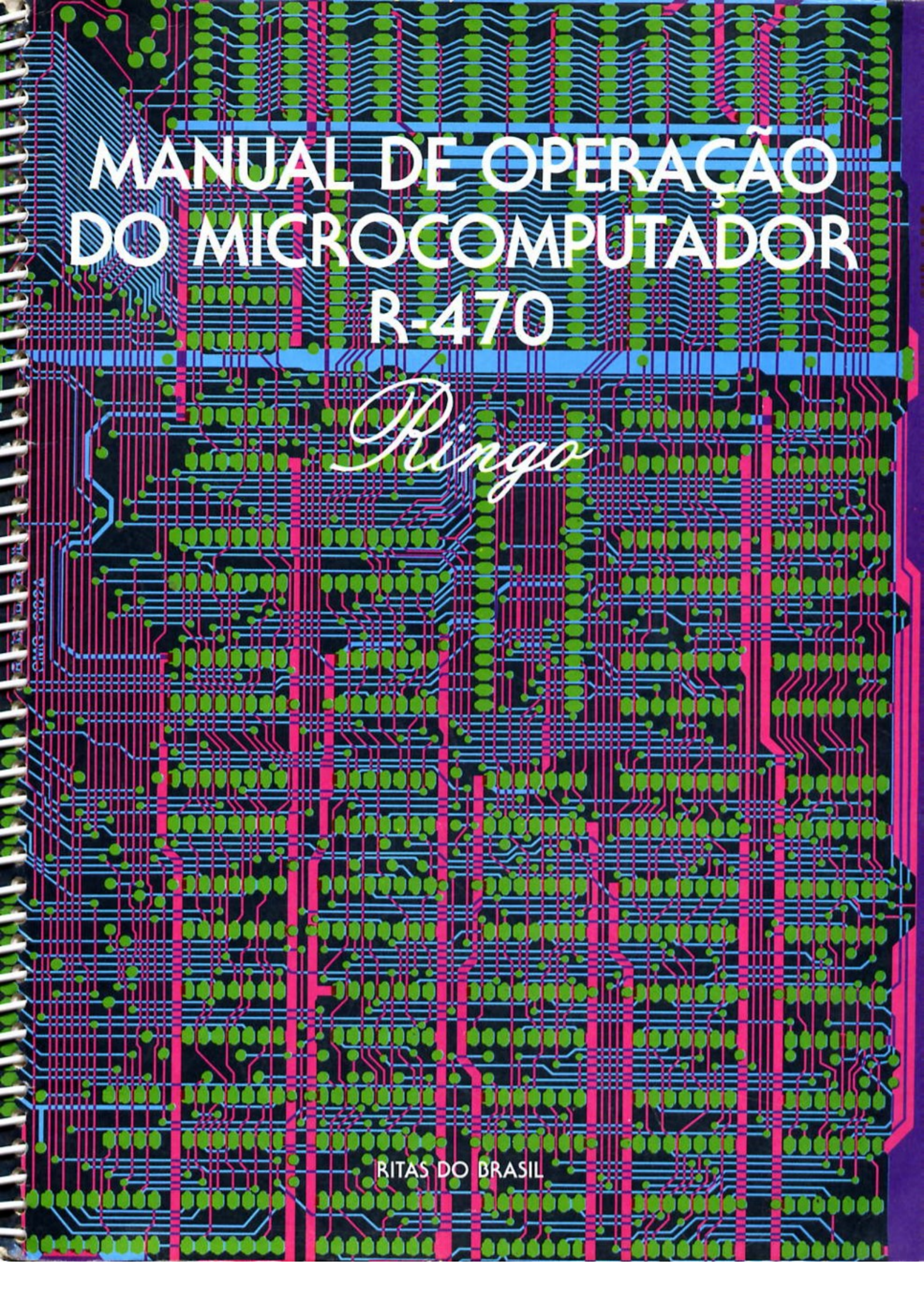
MANUAL DE OPERAÇÃO DO MICROCOMPUTADOR R-470

Ringo



RITAS DO BRASIL

MANUAL DE OPERAÇÃO DO MICROCOMPUTADOR R-470

Ringo

RITAS DO BRASIL

1983
1.º edição

Todos os direitos são reservados, sendo proibida a reprodução total ou parcial deste manual.
À Ritas do Brasil reserva-se o direito de alterar qualquer dado técnico, tanto do aparelho como do manual, sem prévio aviso.

ÍNDICE

1. INTRODUÇÃO

PARABÉNS: Você acaba de adquirir o nosso pequeno, mas poderoso **MICROCOMPUTADOR R-470**, o "**RINGO**".
BOA SORTE a você e a seu novo amigo.

O **MICROCOMPUTADOR R-470** lhe oferece as seguintes características:

- Acoplável a um televisor branco e preto ou em cores, sem necessidade de adaptações.
- Acoplável a qualquer gravador cassete que possua entrada AUX. e saída EAR, para a leitura e o armazenamento de programas em fitas. Estes programas devem ser feitos por você ou adquiridos nas lojas especializadas, devendo ser compatíveis com o **RINGO.**
- Teclado expandido com 49 teclas que possibilitam uma edição mais eficiente dos programas, sendo que, em funções específicas, existe o recurso da repetição, ou seja, enquanto a tecla estiver pressionada, a função se repetirá, aproximadamente, 3 vezes por segundo.
- Uma tecla **exclusiva** para a inversão de vídeo, tornando o fundo da tela branco com os caracteres pretos.
- Funções matemáticas e científicas calculadas em ponto flutuante com 8 algarismos significativos.
- Recursos de utilização gráfica (**PLOT** e **UNPLOT**).
- Comandos para a utilização e execução de programas em linguagem de máquina (**PEEK, POKE** e **USR**).
- Duas velocidades de operação.
- Uma versão de linguagem BASIC que fornece comprimento variável de linha, arranjos, séries de programas e muitas outras características avançadas.
- Possibilidade de expansão da memória à 48 KBytes.
- Conector de saída para JOYSTICK (Manopla para o comando de jogos animados).
- Permite acoplamento de um MODEM (modulador - demodulador), que possibilitará comunicação com outro **RINGO** através de uma linha telefônica.
- Sintetizador de sons opcional, acoplável ao aparelho, que permite a criação de efeitos sonoros especiais sob o controle do programa, sem a necessidade de uma caixa acústica, pois o **MICROCOMPUTADOR R-470** possui um exclusivo canal de som, que permite que os sons gerados sejam reproduzidos pelo próprio televisor.
- Cartuchos de programas "**INSTANT SOFT**" que substituem as já tradicionais fitas cassete. Sua principal vantagem é o tempo de leitura de programa que, como diz o nome, é instantâneo. Basta acoplar o cartucho ao aparelho e ligá-lo, pois o programa já estará à sua total disposição.
- Impressora opcional.
- Fonte de alimentação interna. Basta ligar o **RINGO** à qualquer tomada de 110 volts, sem a necessidade de equipamentos adicionais.

Para o surgimento do **R-470** foi empregada uma avançada tecnologia bem como equipamentos sofisticados e equipes de engenheiros. Mas para fazer com que o **RINGO** funcione corretamente, não é necessário possuir credenciais de engenharia. Pelo contrário. O nosso Microcomputador e este manual foram planejados para ajudar você a conseguir um rápido acesso à essa nova tecnologia.

2. ESPECIFICAÇÕES

2.1. Modelo

Microcomputador **R-470**
Número de dígitos de cálculos: 08
Sistemas de cálculo: de acordo com a fórmula matemática (com prioridade de função)
Linguagem de programação: BASIC

2.2. Capacidades

- CPU - NMOS 8 Bits (Z80 A) f = 3.25 MHz
- Sistemas ROM - 8 KBytes
- Capacidade de memória RAM - 16 KBytes
- Áreas de sistema - 125 Bytes (RAM)

2.3. Cálculos

Cálculos aritméticos, poder de cálculo de funções trigonométricas e trigonométricas inversas, funções logarítmicas e exponenciais, extração de raiz quadrada, função de sinal, absoluta e cálculos lógicos.

2.4. Tela

Tela com 24 linhas e 32 colunas, sendo 2 linhas reservadas para edição. Resolução gráfica atinge 64 x 48 pixels (unidade gráfica).

2.5. Teclado

Do tipo qwerty com teclas alfanuméricas, de edição, funções específicas, funções gráficas, comandos de execução direta e indireta.

2.6. Alimentação

110 Vac ± 10 %

2.7. Consumo

7 W D C

2.8. Dimensões

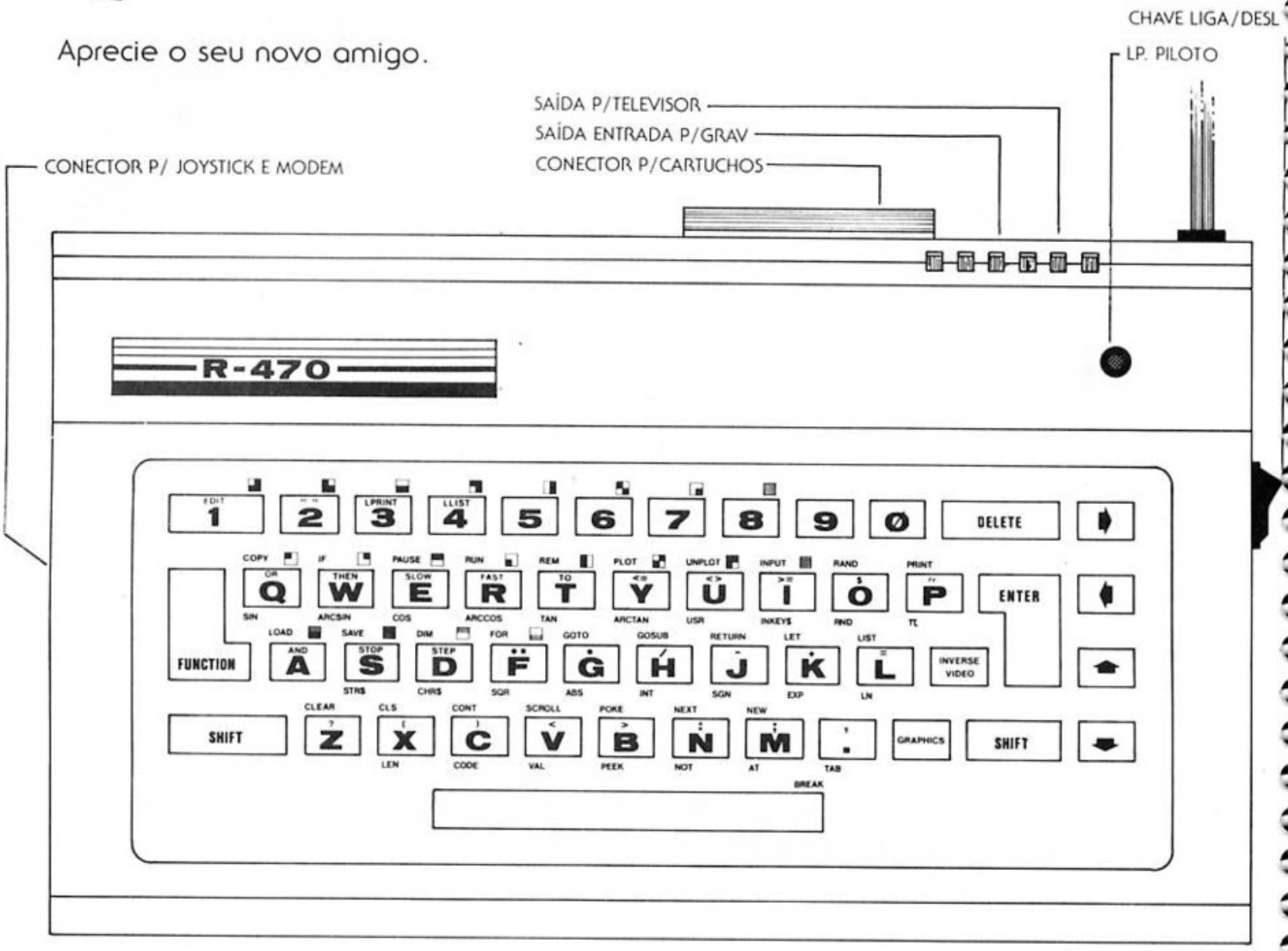
L = 310 mm
P = 180 mm
H = 60 mm

2.9. Acessórios

- Cabo coaxial para interligação com o aparelho de TV
- Cabo blindado duplo para o acoplamento do gravador
- Manual de instruções

3. COMO É O R-470

Aprecie o seu novo amigo.



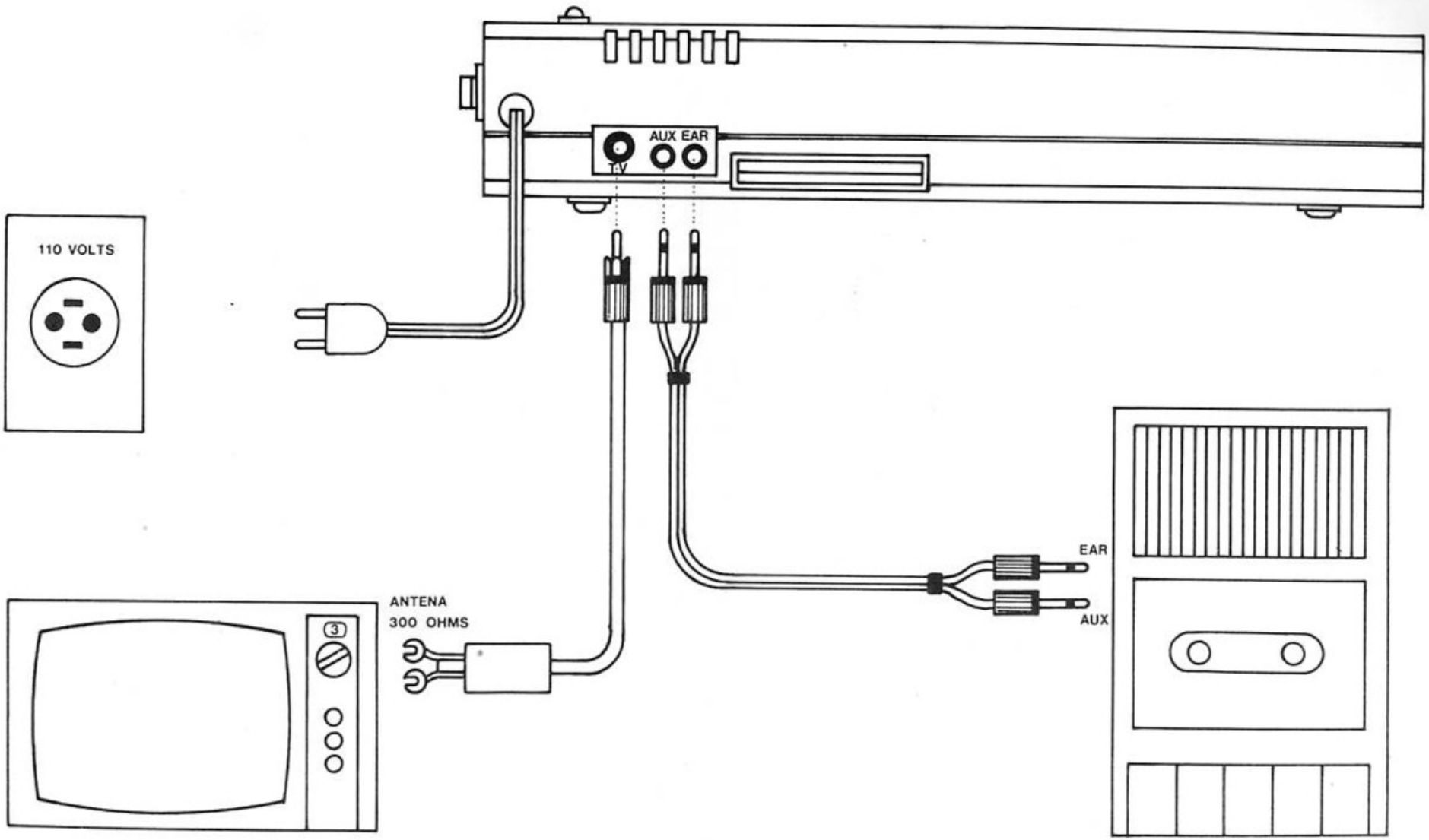
3.1. Conexões

Observe seu Micro por trás. Você facilmente encontrará, à sua esquerda, 3 orifícios. O 1.º é a saída para TV. Interligue o plug do fio coaxial (que acompanha o Micro) nesta saída com a entrada de antena da TV. Antes disso, porém, desligue o fio da mesma. Logo em seguida, encontram-se 2 tomadas que servem para usar programas em fita cassete: AUX e EAR. Conecte nelas o par de cabos com plugs (que também acompanha o **RINGO**) com o AUX e EAR do tipo JACK fêmea de um gravador comum.

Antes de ligar seu Micro, atente para que a rede elétrica seja de 110 volts. Caso não tenha acesso a essa voltagem, utilize-se de um transformador.

Para que você consiga receber a imagem emitida pelo Micro, ligue a TV e sintonize no canal 3. Mexa na sintonia fina até aparecer, o mais legível possível, um quadradinho com a letra K no canto inferior esquerdo da tela (K).

Ao digitar alguma tecla no **RINGO**, a TV emitirá sons agudos. Por isso, abaixe totalmente o volume de som.


AUX EAR
T-V
110 VOLTS
ANTENA
300 OHMS
EAR
AUX

4. CONHECENDO SEU MICRO

4.1. Chave LIGA-DESLIGA

Localizada no lado direito do aparelho, ela é que aciona a fonte de alimentação situada em seu interior. Esta fonte é responsável pelo fornecimento de energia a todos os componentes eletrônicos do circuito. Portanto, ao desligar o aparelho, suas memórias são desativadas, perdendo-se todo o seu conteúdo.

Com esta chave desligada, não é necessário desconectar o Micro da tomada.

4.2. Teclas ALFABÉTICAS E NUMÉRICAS

Através destas teclas que introduzimos os programas, os dados e os comandos para que o Microcomputador execute o que for desejado.

4.3. Tecla SHIFT

Esta é encontrada em dois lugares no teclado do **R-470**: nos dois cantos inferiores.

Sua função é nos possibilitar a introdução dos comandos localizados na parte superior de cada tecla. Ex: FAST, SLOW, STEP, AND, etc. Para que isso aconteça, proceda da seguinte forma: pressione o SHIFT e, em seguida, mantendo-o pressionado, digite a tecla desejada.

Em modo gráfico, ele nos permite utilizar os símbolos semi-gráficos existentes sobre algumas teclas.

4.4. Tecla ENTER

Toda vez que você digita um comando ou linha de programa, o Micro apenas transcreve para a tela o que foi digitado. Para que o comando seja executado ou a linha seja introduzida na memória, é necessária a utilização da tecla ENTER.

Podemos entender esta tecla como sendo o comando "execute", ou seja, tudo que estiver contido na 24.º linha da tela (linha de edição), só será executado quando a pressionarmos.

4.5. Tecla FUNCTION

Esta tecla existe para que tenhamos acesso às funções localizadas abaixo das teclas (impressas em azul-escuro). Ao pressioná-la, o cursor transforma-se em "Cursor Função" (F), bastando, em seguida, apertar a tecla desejada.

4.6. Tecla GRAPHICS

Converte o cursor em "Cursor Gráfico" (G). Toda vez que o cursor estiver em modo gráfico, se digitarmos qualquer letra ou número, estes surgirão invertidos na tela.

Para termos acesso aos símbolos semi-gráficos existentes, o SHIFT deve estar pressionado ao digitarmos a tecla correspondente. As teclas que não possuem estes símbolos, ao serem utilizadas desta forma, nos permitirão gerar os símbolos existentes em sua parte superior de forma invertida. Para que o cursor retorne à condição anterior ao modo gráfico, pressionamos, novamente, a tecla GRAPHICS.

4.7. Tecla INVERSE VÍDEO

Ao ligarmos o **R-470**, a tela do televisor aparecerá escura com os caracteres em branco. Esta tecla nos permite a inversão da tela, ou seja, o fundo passa a ser branco e os caracteres escuros. Pressionando-a mais uma vez, a tela voltará à situação inicial.

4.8. Tecla ESPAÇO-BREAK

Toda vez que não estivermos executando um programa, esta tecla tem a função de espaço (como numa máquina de escrever). Caso contrário, ela servirá como comando de interrupção, ou seja, bloqueará a execução do programa, qualquer que seja sua situação.

4.9. Cursores

Ao ligar o seu **RINGO**, você já deve ter notado que aparece no canto inferior esquerdo da tela a letra **K** em vídeo inverso. Este é o cursor, que se apresenta de 5 modos diferentes:

4.9.1. "Cursor Comando" (K)

Toda vez que o cursor estiver em modo K (Keyword), significa que o **R-470** interpretará a próxima tecla a ser digitada como sendo um comando (impressos acima das teclas, em preto). Ex: Se o cursor estiver em **K** e apertarmos a tecla **P**, surgirá na tela o comando PRINT.

4.9.2. "Cursor Letra" (L)

Em modo **L** (Letter), o cursor indica uma situação na qual toda tecla digitada será interpretada como letra, símbolo ou número correspondente. Ex: Estando o cursor em **L**, ao digitarmos a tecla **P**, esta será interpretada como sendo a própria letra P. Caso digitarmos a tecla **P** com o SHIFT pressionado, estaremos introduzindo as aspas (").

4.9.3. "Cursor Função" (F)

Para que possamos utilizar as funções, localizadas sob as teclas (impressas em azul escuro), devemos converter o cursor para o modo **F** (FUNCTION). Para tanto, basta pressionar a tecla FUNCTION.

4.9.4. "Cursor Gráfico" (G)

Em modo **G** (Graphics) poderemos utilizar os símbolos semigráficos, como descreve o item 4.6.

4.9.5. "Cursor Sintaxe" (S)

No caso de haver algum erro de edição em alguma linha do programa ou comando, o **RINGO** os identifica através da colocação do cursor **S** (Syntax) no ponto onde houver erro de sintaxe.

5. MANUTENÇÃO DO MICROCOMPUTADOR R-470

Para garantir a operação perfeita de seu Microcomputador, nós recomendamos que:

— Conserve-o num lugar livre de temperaturas extremas, umidade ou poeira. Durante o tempo quente, o aparelho deixado sob a luz direta do sol será sujeito à deformações da estrutura.

— Use apenas um pano seco e macio para limpar o seu Microcomputador. Não use solventes, água ou panos úmidos.

— Somente troque os cartuchos e as expansões com o aparelho **desligado**.

— Verifique sempre antes de ligá-lo na tomada se a mesma é de 110 volts.

— Se for necessário algum reparo, o **R-470** deve ser encaminhado apenas aos centros de serviços autorizados pela RITAS DO BRASIL.

— Este manual deve ser usado como referências futuras.

6. PROGRAMAÇÃO INSTANTÂNEA

Este artigo é exclusivamente dedicado às pessoas leigas em programação, sendo que as mesmas deverão ter paciência e muita dedicação com os programas.

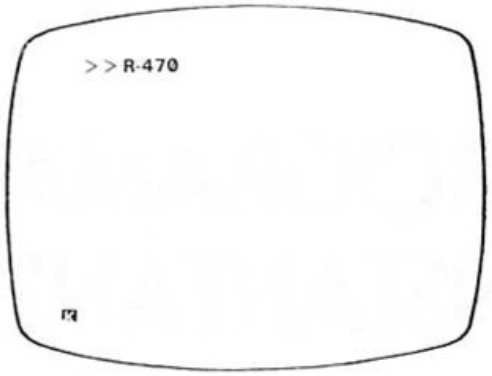
Se você não conseguir fazer esta programação imediata, aconselhamos a releitura do capítulo 4.

Antes de começar a programar, é necessário seguir os exemplos abaixo, passo a passo. Isto porque o Micro recebe suas instruções e executa-as logo em seguida. Está pronto? Então vamos iniciar.

EXEMPLO 1

Primeiramente, verifique se o Microcomputador **R-470** está conectado à tomada e à televisão. Depois, ligue-o através da chave LIGA-DESLIGA. Sua lâmpada piloto deverá acender. Após 1 segundo deverá aparecer o cursor **K** no canto inferior esquerdo da tela. Caso isso não aconteça, desligue e ligue novamente até que o cursor apareça.

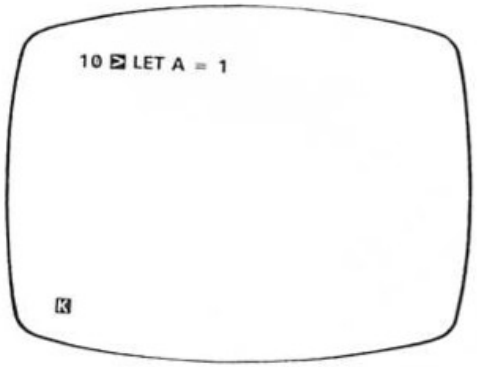
Representaremos a tela de seu televisor de acordo com a ilustração abaixo:



Digite a seguinte seqüência de teclas:

obs: para a utilização do sinal (=) mantenha a tecla SHIFT pressionada, digitando em seguida a tecla (L).

1	Ø	LET	A	SHIFT	=	1	ENTER



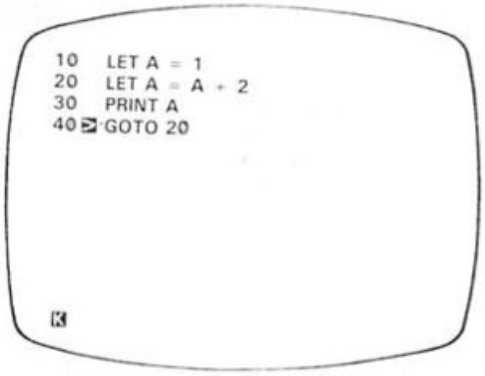
Usaremos o símbolo Ø para representar o número **zero**, diferenciando-o, assim, da letra **O**.

Vamos introduzir, agora, as seguintes linhas:

2	Ø	LET	A	SHIFT	=	A	SHIFT	+	2	ENTER

3	Ø	PRINT	A	ENTER

4	Ø	GOTO	2	Ø	ENTER



Pronto! Seu primeiro programa está completo. Agora, é só executá-lo. Para isso digite o comando RUN seguido do ENTER.

O seu programa está sendo executado. Você deve ter verificado que quando chegou ao fim da tela, ele parou e apresentou o código 5/30, significando que todas as linhas da tela foram preenchidas e o programa parou na linha 30. Para que se continue o programa é muito simples: pressione as teclas CONT e ENTER que, automaticamente, a tela se apagará, reiniciando a contagem, ou melhor, a execução do programa.

EXEMPLO 2

Antes de começar este exemplo, pressione as teclas NEW e ENTER para limpar o programa anterior.

```
10 LET A$ = "MARIA"
20 LET B$ = " E CARLOS"
30 PRINT A$;B$
40 STOP
```

TECLADO

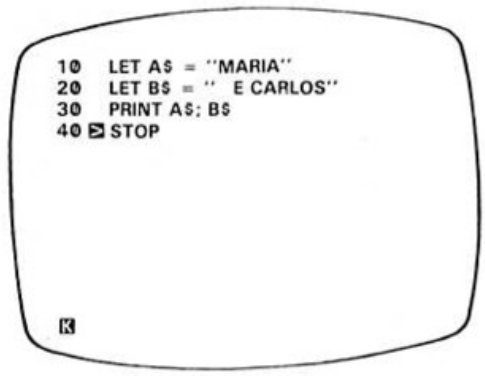
1	Ø	LET	A	SHIFT	$	SHIFT	=	SHIFT	"	M	A	R	I	A	SHIFT	"	ENTER

2	Ø	LET	B	SHIFT	$	SHIFT	=	SHIFT	"	ESPAÇO	E	ESPAÇO	C	A	R	L	O	S

SHIFT	"	ENTER

3	Ø	PRINT	A	SHIFT	$	SHIFT	;	B	SHIFT	$	ENTER

4	Ø	SHIFT	STOP	ENTER



RUN	ENTER

7. CÁLCULOS IMEDIATOS

Este capítulo vai ensinar a você como usar o **R-470** em operações aritméticas. Não se preocupe, pois não é tão terrível assim. Com a ajuda de seu Micro a matemática se torna muito fácil.

Aproxime-se dele sem receio, pois você não poderá feri-lo ou danificá-lo simplesmente por pressionar qualquer tecla.

7.1. Operações aritméticas

A operação aritmética é representada por elementos de operação como Adição, Subtração, Multiplicação, Divisão, Exponenciação e Notificação Científica.

OPERAÇÃO	SÍMBOLO
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Exponenciação	**
Notação Científica	**E** (n.º * 10 ** expoente)

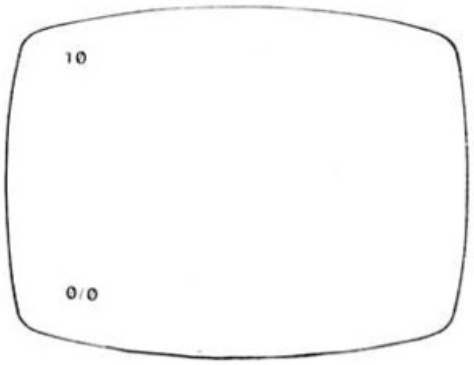
EXEMPLO 1 2 x 2 + 6

Para fazer esta conta proceda da seguinte forma:

*PRINT 2 * 2 + 6*

TECLADO

PRINT	2	SHIFT	*	2	SHIFT	+	6	ENTER



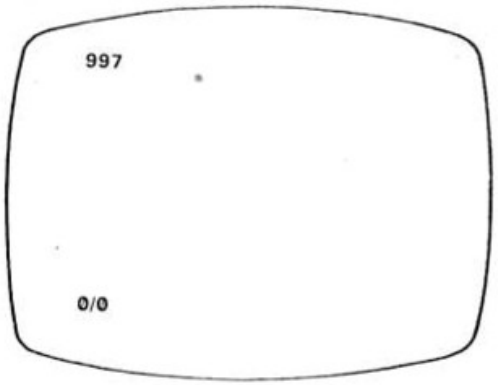
EXEMPLO 2 2 x 365 + 12^2 − 60 : 30 + 125

*PRINT 2 * 365 + 12 * * 2 - 60 / 30 + 125*

TECLADO

PRINT	2	SHIFT	*	3	6	5	SHIFT	+	1	2	SHIFT	**	2	SHIFT	-	6	Ø	SHIFT

/	3	Ø	SHIFT	+	1	2	5	ENTER

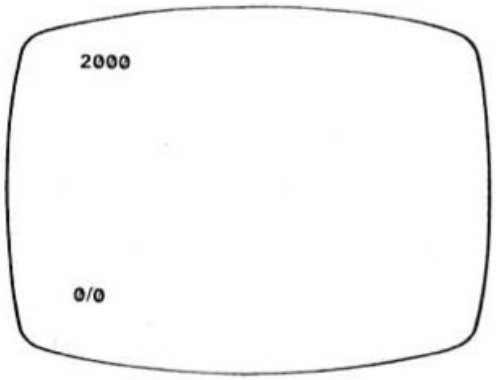


EXEMPLO 3 2×10^3

PRINT 2 E 3

TECLADO

PRINT	2	E	3	ENTER



8. A MISTERIOSA ARTE DA PROGRAMAÇÃO

Por um longo tempo, a arte da programação foi inutilmente envolvida num ar de mistério e a maioria das pessoas associavam-na com mágicas e gênios matemáticos. Na verdade, não é necessário nenhum talento especial para programar. Seus maiores recursos serão a paciência, habilidade de raciocínio lógico, atenção com detalhes e vontade de aprender.

Nossa intenção não é fazer de você um "expert" em programação e, sim, familiarizá-lo com suas operações e conceitos básicos.

8.1. O que é um programa?

Você, provavelmente, ficará surpreso ao saber que um programa é, justamente, um conjunto de comandos que o computador segue, um a um. Os comandos são dados a ele em uma linguagem própria. O **RINGO** fala o dialeto BASIC*— linguagem de programação muito usada e popular entre os microcomputadores. E, como toda linguagem, o BASIC possui um vocabulário especial e regras que são combinadas para formar comandos. Ao falar com o **R-470** em uma linguagem não familiar, ele alertará você para seu erro.

O BASIC foi originalmente desenvolvido para ensinar os princípios de programação. Seus comandos são palavras escritas em inglês e outros símbolos familiares.

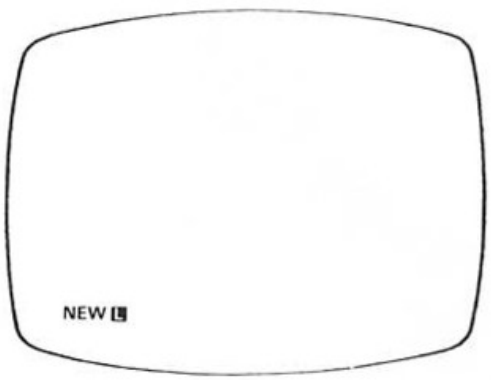
* Beginner's All-purpose Symbolic Instruction Code

8.2. Como fazer um programa

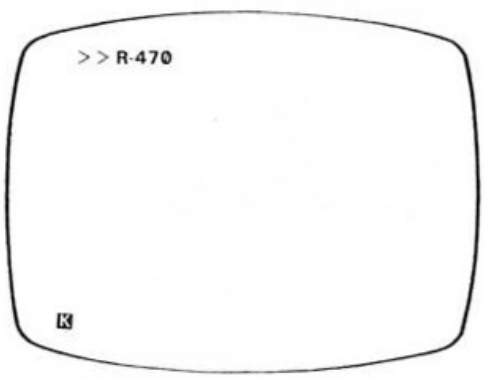
Para você programar o **RINGO**, basta seguir uma seqüência de comandos. Ao final da leitura deste manual você estará apto para realizar infinitos programas. Vamos desenvolver um, agora. Não se preocupe em entendê-lo. Com o decorrer da leitura você terá explicações específicas de cada passo.

Primeiramente, aperte a tecla NEW. Ela, seguida da tecla ENTER, apagará qualquer comando que possa estar na memória.

NEW



ENTER



Estando as memórias limpas, podemos iniciar a programação. Começaremos com uma linha de programa.

EXEMPLO 1

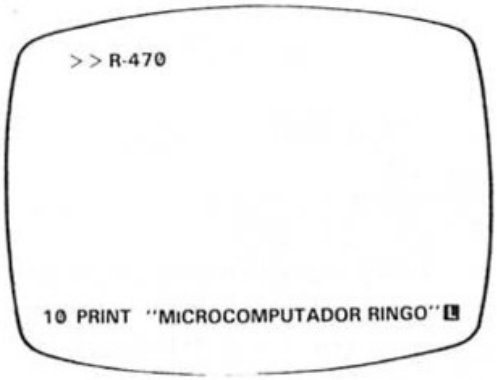
1Ø PRINT ''MICROCOMPUTADOR RINGO''

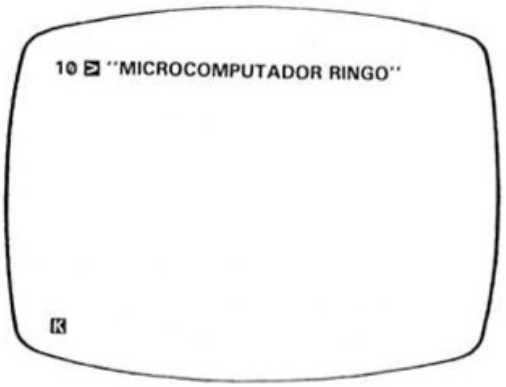
TECLADO

1	Ø	PRINT	SHIFT	''	M	I	C	R	O	C	O	M	P	U	T	A	D	O	R	ESPAÇO	R	I	N	G	O

SHIFT	''

ENTER





Agora, digite o RUN, seguido do ENTER, para que o programa seja executado.

RUN

ENTER

obs: alguns comandos como o NEW e RUN são executados imediatamente ao pressionarmos a tecla ENTER. Porém, aqueles precedidos de número de linha de programa (veja item seguinte), não são executados e, sim, introduzidos no programa que está sendo editado.

8.3. Linhas de programa

Um programa em BASIC consiste de uma série de linhas numeradas contendo um ou mais comandos. Esta numeração é usada pelo **RINGO** para manter uma seqüência correta de execução do programa, servindo, também, como referência ao programador para facilitar a sua construção.

Em praticamente todos os exemplos de programas sugeridos neste manual, as linhas estão numeradas de 10 em 10. Na verdade, a distância numérica entre as linhas de um programa pode ser qualquer número inteiro. Aconselhamos que a distância seja de 10, para que possamos intercalar — quando necessário — até 9 novas linhas de programa entre duas já editadas, sem a necessidade de se deslocar a numeração das linhas posteriores à modificação.

EXEMPLO

```
1 PRINT "R-47Ø"
2 GOTO 1
```

```
1Ø PRINT "R-47Ø"
2Ø GOTO 1Ø
```

Para introduzirmos um comando SCROLL após o PRINT, no primeiro caso seria necessário deslocar a linha 2, transformando-a em linha 3. Já no segundo caso isso não ocorre. Basta introduzir uma nova linha que pode ser de 11 a 19. Então:

```
1 PRINT "R-47Ø"
2 SCROLL
3 GOTO 1
```

```
1Ø PRINT "R-47Ø"
15 SCROLL
2Ø GOTO 1Ø
```

8.4. Edição

A 24.ª linha da tela é onde se encontra a linha de edição. Nela digitamos cada linha do programa, separadamente. Após a digitação da tecla ENTER esta linha passa para o programa.

8.5. Correção de linhas de programa

Existem 6 comandos que são usados para esta finalidade: DELETE, EDIT, MOVIMENTO DO CURSOR P/ESQUERDA, P/DIREITA, P/CIMA E P/BAIXO.

Durante uma programação poderá ocorrer algum erro na digitação de suas linhas. Corrija-o usando os comandos MOVIMENTO DO CURSOR PARA DIREITA (→), MOVIMENTO DO CURSOR PARA ESQUERDA (←) e DELETE.

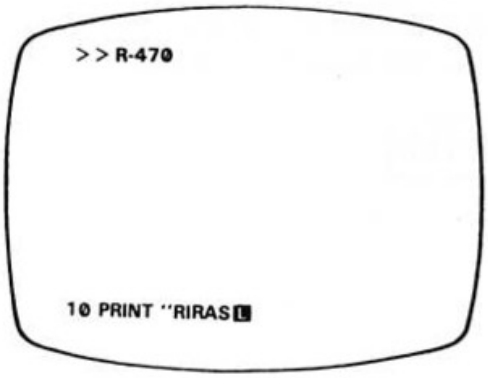
EXEMPLO

NEW

10 PRINT ''RITAS DO BRASIL''
20 GOTO 10

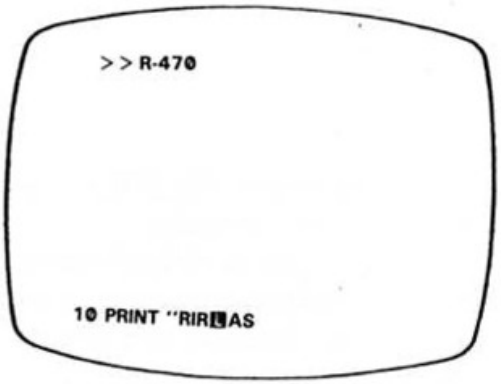
TECLADO

1	Ø	PRINT	SHIFT	''	R	I	R	A	S

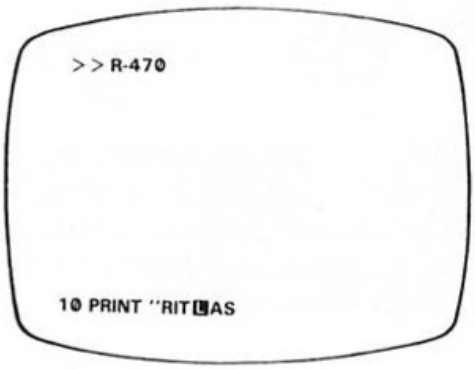


Verifique que foi cometido um erro. Ao escrever RITAS, você trocou o T pelo R. Para corrigir tal erro, aperte a tecla (←) até que o cursor se posicione no lado direito da letra incorreta. Feito isso, apague-a pressionando o comando DELETE e, logo em seguida, digite a letra desejada. Agora, utilizando a tecla (→), vá até a última letra digitada e continue a programação.

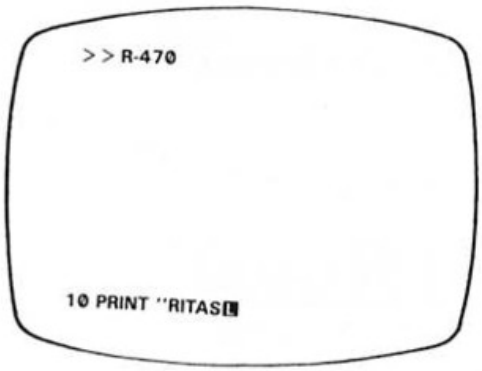
Pressione a tecla (←)



Pressione o DELETE e digite a letra certa (T)



Agora, pressione a tecla (→) até a última letra digitada e continue a programação



TECLADO

ESPAÇO	D	O	ESPAÇO	B	R	A	S	I	L	SHIFT	"	ENTER

2	Ø	GOTO	1	Ø	ENTER

RUN	ENTER

Você já deve ter notado que as teclas DELETE, (→) e (←) possuem repetição automática, facilitando seu trabalho de correção.

Caso haja necessidade de apagar a linha inteira que está sendo editada, pressione as teclas SHIFT e EDIT simultaneamente e ela desaparecerá de uma só vez, num processo bem mais rápido, ficando em seu lugar a linha em que o cursor

"Indicador de Linha" (▶) estiver posicionado. Pressione a tecla ENTER e reinicie o programa.

Se você digitar uma linha cujo n.º já esteja contido no programa, a 2.ª linha digitada substituirá automaticamente a anterior.

No caso de uma linha incorreta, já ter sido inserida no programa, devemos utilizar os comandos MOVIMENTO DO CURSOR PARA CIMA (↑), MOVIMENTO DO CURSOR PARA BAIXO (↓) e o EDIT para corrigi-la.

Para isso:

- consulte o cursor "Indicador de Linha" (▶) no programa para ver em que linha ele se encontra.
- pressione as teclas (↑) ou (↓) quantas vezes forem necessárias para que o cursor atinja a linha desejada.
- tecle o SHIFT e o EDIT, simultaneamente, para que ela seja transferida à linha de edição.
- corrija a linha utilizando as teclas (→), (←) e DELETE ou apenas acrescente a informação que faltou.

Para a linha retornar ao programa depois da correção, basta apertar o ENTER que ele voltará instantaneamente.

Se você quiser eliminar uma linha já inserida no programa, digite o n.º da linha seguido do ENTER; você notará que o cursor "Indicador de Linha" (▶) desaparece. Porém, não se preocupe, ele estará localizado entre as duas linhas que circundavam a linha eliminada. Se você, após isso, pressionar o EDIT, será editada a linha imediatamente abaixo da que foi eliminada. Caso a linha excluída tenha sido a última do programa, será editada a linha que tornou-se a última.

9. APROFUNDANDO SEUS CONHECIMENTOS

Até agora, você tem visto apenas comentários sobre alguns comandos e não uma explicação definitiva. Após ter se ambientado com o **RINGO**, você deve iniciar a leitura dos comandos em linguagem BASIC.

9.1. Início de programa

9.1.1. Comando **NEW**

Uma programação feita com linhas numeradas não se apaga da memória mesmo depois de sua execução. Significa que, ao introduzirmos novos comandos de programa, o que se consegue é uma edição ou mistura com o programa anterior.

A não ser que você desligue o **RINGO**, as memórias não se apagam. Portanto, antes de qualquer início de programa, devemos digitar o comando NEW, sendo esta a única forma de limparmos as memórias sem desligar o aparelho.

TECLADO

NEW	ENTER

9.2. Variáveis

9.2.1. O que são variáveis

Existem situações nos programas em que necessitamos guardar nomes, letras, valores e etc. A linguagem BASIC nos fornece um artifício muito importante para facilitar este procedimento: são as variáveis.

Estas variáveis funcionam como uma memória de calculadora. Damos um valor à letra e, para chamá-lo de volta, basta especificar a letra correspondente.

Podemos exemplificar assim:

Se denominarmos à variável A o valor 5, toda vez que utilizarmos esta variável, obteremos o valor numérico 5. Por exemplo, se multiplicarmos A por 3, o resultado será 15. Se somarmos A com 20, teremos como resultado o 25, e assim por diante. Veja:

A = 5
A × 3 = 15
A + 20 = 25
A ÷ 2 = 2,5

Existem dois tipos de variáveis: numéricas e alfanuméricas.

9.2.1.1. VARIÁVEIS NUMÉRICAS

Como o nome já diz, trabalham apenas com números. No que diz respeito ao nome da variável, este sim precisa ser formado por uma letra ou um conjunto de letras.

EXEMPLO 1

```
LET A=20
PRINT A
```

EXEMPLO 2

```
10 LET CARLOS=1
20 PRINT CARLOS
```

No exemplo acima, notamos que o nome da variável pode ser formado por uma palavra.

EXEMPLO 3

```
10 LET ABC123=95
20 PRINT ABC123
```

Agora ficou claro que o **R-470** também aceita números para dar nomes às variáveis numéricas.

ATENÇÃO: O Micro aceita números em nomes de variáveis numéricas, mas somente se estes nomes começarem com uma letra.

9.2.1.2. VARIÁVEIS ALFANUMÉRICAS

Este tipo de variável possibilita a utilização de letras, palavras, frases, mensagens, enfim, todo o tipo de combinações que não terão valor matemático, e sim, de reprodução.

EXEMPLO

```
10 LET A$ = "RINGO"
20 PRINT A$
```

9.2.2. Comando **LET**

Na execução de um programa em BASIC, se for necessário atribuirmos ou alterarmos o valor de alguma variável numérica ou alfanumérica, um dos recursos é a utilização do comando LET.

EXEMPLO 1

```
10 LET T=1
20 PRINT T
```

No exemplo acima, T é uma variável que, na linha 10, lhe foi atribuido o valor 1. Este valor pode ser, também, decorrente de uma expressão matemática, como no exemplo abaixo.

EXEMPLO 2

```
10 LET X=30/5+4
20 PRINT X
```

Neste caso, após o **RINGO** executar a linha 10, à variável X foi atribuído o valor numérico 10.

É muito comum que se queira, em algum ponto do programa, incrementar o valor de uma variável. O exemplo seguinte mostra como podemos executar esta operação sem se preocupar com o valor que a variável possuia antes desta linha.

EXEMPLO 3

```
10 LET N=0
20 PRINT N
30 LET N=N+1
40 PRINT N
```

Poderemos observar com maior clareza nos exemplos seguintes, onde trataremos com variáveis alfanuméricas, que o sinal "=", utilizado no comando LET, não significa igualdade, como nas expressões matemáticas. Neste caso, devemos entender este sinal como sendo uma atribuição de valores, ou seja, à variável que estiver ao lado esquerdo deste sinal, será atribuído o valor numérico ou alfanumérico da expressão que estiver ao seu lado direito.

EXEMPLO 4

```
10 LET F$="APOLLO 11"
20 PRINT F$
```

Como vimos anteriormente, toda variável que possuir o sinal "$" do seu lado direito é considerada uma variável alfanumérica (área da memória onde podemos armazenar palavras ao invés de números). Portanto, à estas variáveis devemos sempre atribuir conjunto de caracteres (STRINGs).

EXEMPLO 5

```
10 LET A=120
20 LET B$="(CENTO E VINTE)"
30 PRINT "NO CINEMA HA ";A;B$;"LUGARES"
```

Ao contrário das variáveis numéricas, as variáveis alfanuméricas só podem ser definidas por uma letra e o $ (STRING).

9.2.3. Manipulação de variáveis alfanuméricas

Quando necessitamos operar o Micro com o auxílio de variáveis alfanuméricas utilizamos o $ (STRING). Ele faz com que a mensagem contida em seu interior não tenha valor matemático, ficando apenas como simples combinação de letras e/ou números.

EXEMPLO

```
10 PRINT ''QUAL E SEU NOME?''
20 INPUT A$
30 PRINT A$;''. BONITO NOME''
```

No exemplo acima você viu que o **R-470** ficou esperando que fosse escrito alguma coisa para ele poder completar o programa. E a mensagem que você escreveu ficou entre aspas porque uma STRING é sempre entre aspas.

Se tivéssemos colocado na linha 40 um GOTO 10, o programa ficaria girando até a tela acabar. Experimente.

Agora, como pará-lo antes da tela ser preenchida, se o Micro considera tudo o que escrevemos como variável, já que as aspas estão sempre presentes? É simples. Basta apagar as aspas através da tecla EDIT ou DELETE e pressionar o STOP. Ao darmos o ENTER, o programa pára onde estiver.

obs.: podemos colocar quantas STRINGs desejarmos em um programa, bastando utilizar letras diferentes para cada uma.

9.2.3.1. SUB-STRING

Se quisermos manipular com apenas uma parte da STRING de um programa, utilizamos o artifício da SUB-STRING.

Existem 4 maneiras de se utilizar a SUB-STRING. Veja.

EXEMPLO 1

```
10 LET A$ = ''ABCDEF''
20 PRINT A$ (2)
30 GOTO 20
```

Neste exemplo aparecerá na tela apenas a segunda letra.

EXEMPLO 2

```
10 LET B$ = "ABCDEF"
20 PRINT B$ (2 TO)
30 GOTO 20
```

No exemplo acima, o Micro colocará na tela apenas da segunda letra em diante.

EXEMPLO 3

```
10 LET C$ = "ABCDEF"
20 PRINT C$ (2 TO 4)
30 GOTO 20
```

Já neste exemplo, o **R-470** só ocupará a tela da segunda à quarta letra.

EXEMPLO 4

```
10 LET D$ = "ABCDEF"
20 PRINT D$ (TO 4)
30 GOTO 20
```

Neste, o **RINGO** imprimirá na tela somente até a quarta letra.

9.2.4. Comando CLEAR

Existe uma região na memória do Micro onde ele armazena todas as variáveis definidas no decorrer de um programa BASIC. O comando CLEAR limpa esta região e faz com que estas variáveis deixem de existir.

EXEMPLO

```
10 LET A=2
20 PRINT A
30 CLEAR
40 PRINT A
```

Para que você entenda melhor este comando, vamos analisar o programa acima:

Na linha 10, atribuimos um valor para a variável A. Na linha 20, mandamos imprimi-la. Na 30, eliminamos a variável e, na linha 40, pedimos, novamente, a impressão do valor de A. Notamos que apareceu um **código de reportagem*** no canto inferior esquerdo da tela (2/40). Este código significa que tentamos utilizar uma variável não definida. Isto aconteceu porque solicitamos ao Micro a impressão de uma variável não mais existente em sua memória.

* Ver capítulo 11

9.3. Utilização da tela

9.3.1. Comando **PRINT**

Este é um dos comandos mais utilizados em um microcomputador. Ele faz com que os resultados de programas sejam colocados na tela. Sem o PRINT, o **R-470** guarda a resposta só para si, não nos mostrando o que queremos saber.

EXEMPLO 1

```
PRINT 8/4+22
```

Ao pressionarmos a tecla ENTER, o resultado aparecerá no canto superior esquerdo da tela.

EXEMPLO 2

```
10 PRINT "R-470:"
20 PRINT
30 PRINT "UM GRANDE MICRO"
```

Note que entre as mensagens "R-470:" e "UM GRANDE MICRO" existe uma linha com o comando PRINT sozinho. Isto fará com que esse comando mova a atual posição de impressão para o início da próxima linha disponível.

9.3.2 ASPAS

Para que possamos imprimir uma informação ou uma STRING na tela é necessário que utilizemos as aspas.

EXEMPLO 1

```
10 PRINT "DIGITE SEU NOME"
20 INPUT A$
30 PRINT A$
```

Agora digite seu nome ou outro qualquer.
IMPORTANTE: Tudo o que estiver entre as duas aspas utilizadas no comando PRINT, será interpretado pelo Micro apenas como palavras.

EXEMPLO 2

```
PRINT "FAST AND SLOW"
```

9.3.3. ASPAS ASPAS

Acabamos de ver para que servem as ASPAS. O ASPAS-ASPAS (SHIFT 2) serve para se colocar uma informação ou STRING dentro da informação ou STRING contida nas ASPAS. Para simplificar, acompanhe o exemplo abaixo.

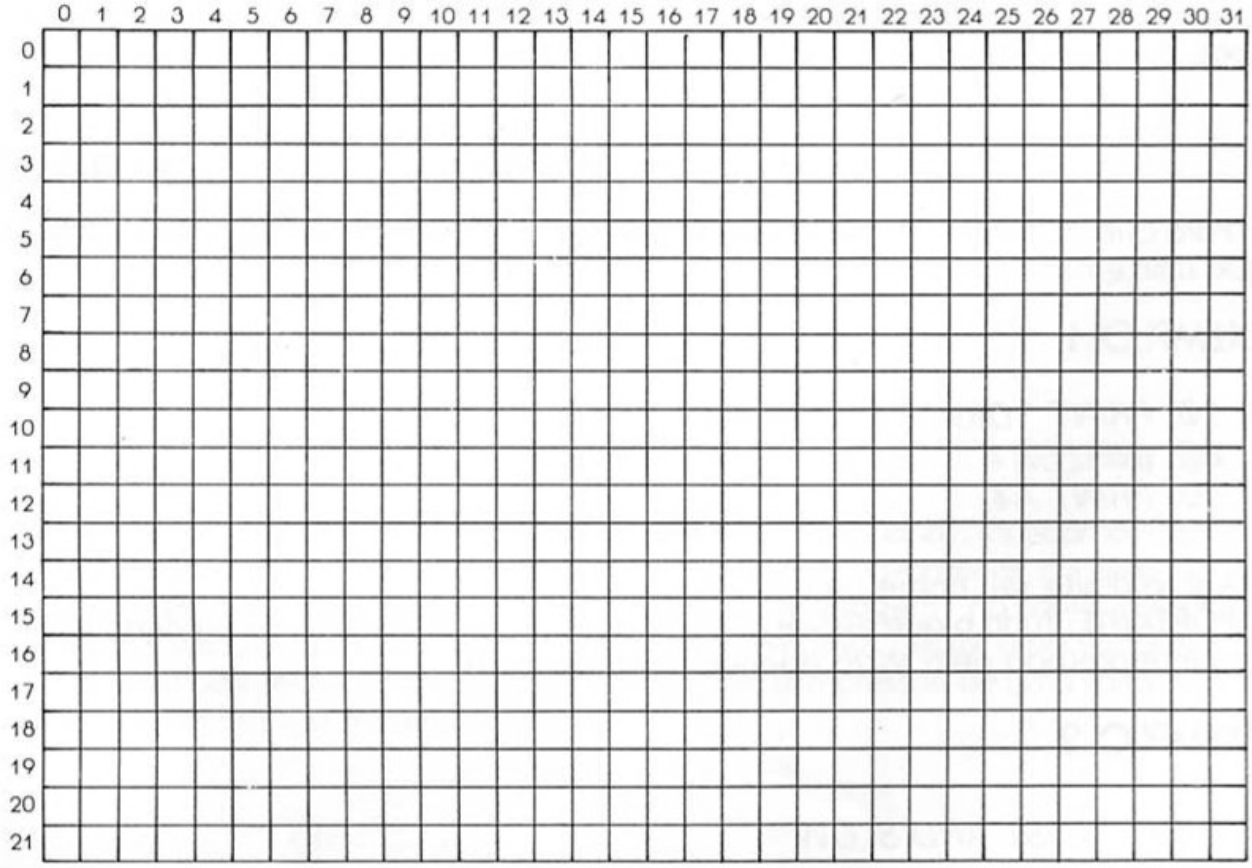
EXEMPLO

```
10 PRINT "R-470, "" THE BEST MICRO"" "
20 GOTO 10
```

obs: Você já deve estar curioso para saber o que significam os números e letras que aparecem no canto inferior esquerdo da tela toda vez que um programa é executado. Eles servem para indicar erro, fim de tela, programa executado e etc. A explicação completa destes códigos está no capítulo CÓDIGOS DE REPORTAGEM.

9.3.4. Função **AT**

A função AT, que só funciona com o comando PRINT, Pode percorrer qualquer ponto da tela, mesmo que já esteja impressa alguma coisa nele. Para isso, basta usar o PRINT AT seguido pelo número de linha e coluna, sendo que a linha é enumerada de cima para baixo, de 0 a 21, e a coluna, da esquerda para a direita, de 0 a 31, formando, assim, um quadro de 704 posições diferentes.



EXEMPLO 1

```
10 PRINT AT 10,9;"DISCO"
20 PRINT AT 11,13;"LASER"
```

Se quiser colocar esse programa em uma linha só, basta utilizar a vírgula ou o ponto e vírgula ligando as duas linhas. Verifique o exemplo:

EXEMPLO 2

```
PRINT AT 10,11; "DISCO"; AT 11,11; "LASER"
                    ou
PRINT AT 10,11; "DISCO", AT 11,11; "LASER"
```

9.3.5. Função **TAB**

A função TAB é utilizada juntamente com o comando PRINT para imprimir na tela em forma de tabulação, isto é, tudo o que digitamos no **R-470** será colocado em uma coluna pré-indicada. Caso esta coluna já estiver ocupada, o Micro imprimirá na próxima linha da tela.

EXEMPLO 1

```
10 PRINT TAB 8; "MICROCOMPUTADOR"
20 PRINT TAB 9; "PESSOAL R-470"
```

Como no AT, esse programa também poderá ser escrito em uma só linha. Para tanto, utilize a vírgula ou o ponto e vírgula.

EXEMPLO 2

```
PRINT TAB 8; "MICROCOMPUTADOR"; TAB 9; "PESSOAL R-470"
                    ou
PRINT TAB 8; "MICROCOMPUTADOR", TAB 9; "PESSOAL R-470"
```

9.3.6 PONTO E VÍRGULA

O PONTO E VÍRGULA faz com que o comando PRINT imprima suas mensagens sem deixar espaço, colocando-as imediatamente após a última impressão.

EXEMPLO

```
10 PRINT "VICTOR"
20 PRINT "OU"
30 PRINT "VICTORIA"
```

Se você quiser que estas palavras sejam escritas numa só linha, coloque o PONTO E VÍRGULA no fim das linhas 10 e 20 e dê um espaço depois do R e do U.

9.3.7 VÍRGULA

A VÍRGULA divide a tela em dois blocos de 15 colunas. Apesar de a tela possuir 32 colunas, cada bloco só imprimirá na mesma linha mensagens com, no máximo, 15 caracteres. Isso porque a vírgula, apesar de não aparecer na tela, ocupa seu espaço. O primeiro bloco começa na coluna 0e o segundo na 16.

EXEMPLO 1

```
PRINT "TIMES", "PONTOS",,,"PALMEIRAS", 10, "FLAMENGO", 8,
"ATLÉTICO", 5
```

Podemos concluir que a vírgula serve para tabular uma impressão na tela. Perceba que as 3 vírgulas situadas após a impressão "PONTOS" serviram para pular uma linha no programa, pois cada vírgula, significando meia tela, vai pulando de bloco a cada inserção, sendo que, de 2 em 2, pula 1 linha.

É possível utilizarmos numa mesma linha de um comando PRINT o PONTO E VÍRGULA e a VÍRGULA.

EXEMPLO 2

```
10 LET A = 15
20 LET B = 25
30 PRINT "A=";A,"B=";B
```

9.3.8. Comando SCROLL

Como já vimos anteriormente, a tela do **RINGO** possui 22 linhas. Se o programa utilizar um número de linhas superior a este, ele pára e indica, através do código de reportagem 5/nº da linha, que a tela está cheia.

Para evitar que isso ocorra devemos usar o comando SCROLL.

EXEMPLO

```
10 PRINT "PEGASUS"
20 PRINT "AGAMENON"
30 GOTO 10
```

"Rode" este programa. Agora coloque um SCROLL na linha 5 e outro na 15; modifique a linha 30 para GOTO 5 e veja o que acontece.

Neste exemplo fica claro que a imagem se move durante a execução do programa e que a primeira linha desaparece para dar lugar à última.

Para o programa parar de rodar, aperte a tecla BREAK.

9.3.9. Comando CLS

Este comando tem a função de limpar a tela, seja em programa, seja fora dele.

EXEMPLO

```
10 PRINT AT 10,10;"R-470"
20 CLS
30 GOTO 10
```

9.4. Introdução de dados

9.4.1. Comando **INPUT**

Em alguns programas existe a necessidade de introduzirmos dados durante a sua execução. Para que possamos fazer tal operação existe o comando INPUT que permite entrarmos com dados ou conjuntos de caracteres (STRINGs). Quando o computador interpreta um comando INPUT em uma linha do programa, ele bloqueia a seqüência de execução, solicitando ao operador que seja introduzido o dado desejado.

O comando INPUT é sempre seguido de uma variável (numérica ou alfanumérica), ou seja, à variável indicada será atribuido o dado digitado pelo operador.
Vamos fazer um programa de extração de raiz quadrada.

EXEMPLO 1

```
10 INPUT A
20 PRINT "RAIZ QUADRADA DE ";A;" = ";SQR A
```

Note que surgiu o cursor **L** no canto inferior esquerdo da tela. Esta é a indicação de que o **RINGO** está aguardando a digitação de um dado pelo operador. Basta agora colocar um número e, em seguida, pressionar a tecla ENTER, que o **RINGO** prosseguirá com o programa fazendo com que a variável correspondente ao comando INPUT (no exemplo acima a variável A) assuma o valor digitado.

Experimente executar o programa novamente, sendo que, quando o **R-470** solicitar a introdução de um dado, pressione somente a tecla ENTER.

Observe que surgirá o cursor **S**. Ele indica que houve um erro de sintaxe. Neste caso, o erro foi a não introdução de um valor numérico quando solicitado.

Mesmo quando o valor numérico desejado é zero, é necessário que ele seja colocado.

Se entrarmos com alguma letra ao invés de um número, o programa será interrompido com o surgimento de um **código de reportagem** na linha de edição.

Neste caso, o código acusa nossa tentativa indevida de atribuir a uma variável numérica um dado não numérico. Porém, se o caracter introduzido for uma variável já definida no programa, o Micro segue em sua execução atribuindo à variável do INPUT o valor numérico devido.

EXEMPLO 2

```
10 PRINT "ENTRE SEU PRIMEIRO NOME"
20 INPUT A$
30 CLS
40 PRINT "ENTRE SEU ULTIMO NOME"
50 INPUT B$
60 CLS
70 LET C$ = A$ (1) + "." + B$ (1) + " "
80 PRINT "SUAS INICIAIS SAO: "; C$
```

Este exemplo, como podemos observar, opera com variáveis alfanuméricas. Durante a execução do programa, o INPUT é distinguido através das duas aspas que são colocadas na "Linha de Edição" junto ao cursor **L**.

Lembre-se: como já explicamos anteriormente, toda variável alfanumérica (STRING) deve sempre estar entre aspas. No caso de um INPUT alfanumérico, o próprio comando se encarrega das aspas, bastando apenas digitarmos a STRING desejada.

O INPUT alfanumérico também possui uma maneira de transferir para sua variável correspondente o conteúdo de uma variável já definida no programa. Para efetuar esta transferência, espere surgir a mensagem "L" na tela e, em seguida, apague as aspas; assim, a variável desejada poderá ser digitada.

O **RINGO** tem um método bem rápido e prático para apagar as aspas: é só apertar a tecla EDIT e elas serão, automaticamente, eliminadas.

Verifique o exemplo abaixo:

EXEMPLO 3

```
10 INPUT A$
20 INPUT B$
30 PRINT A$ , B$
```

Na linha 10, o programa aguarda que se escreva uma STRING alfanumérica. Após você tê-la digitado, aperte a tecla ENTER para o Micro guardá-la em sua memória. Logo em seguida, surgirá na tela novamente as aspas. O programa já passou, então, para a linha 20. Agora ao invés de escrever outra STRING, apague as aspas utilizando a tecla EDIT e digite A$ (variável já definida). Apertando a tecla ENTER novamente, o **RINGO** passará para a linha 30 e escreverá na tela o resultado, que será a repetição da STRING.

9.4.2. Função **INKEY$**

Esta função testa se alguma tecla está sendo pressionada.

EXEMPLO 1

```
10 LET M$ = INKEY$
20 IF M$ = "1" THEN LET M$ = "  "
30 PRINT M$;
40 IF INKEY$ = M$ THEN GOTO 40
50 GOTO 10
```

Este exemplo mostra claramente a função do INKEY$. Nele, cada tecla que você apertar será imediatamente impressa na tela. Como a tecla ESPAÇO, durante a execução de um programa torna-se BREAK, utilizamos o artifício da linha 20. Ela faz com que toda vez que digitarmos o número 1, o **RINGO** considere como espaço. Cuidado para não errar o que estiver escrevendo, pois não tem como apagar.

EXEMPLO 2

```
10 PRINT INKEY$;
20 GOTO 10
```

9.5. Comandos Operacionais

9.5.1. Comando **RUN**

Ao introduzirmos um programa na memória do **RINGO**, é necessário "avisar" ao Micro que ele já pode executá-lo. Esta é a função do comando RUN.

O comando RUN determina ao **RINGO** que execute o programa que estiver na memória em ordem seqüencial e crescente (Daí o motivo de numerar as linhas do programa).

Para que não haja problema com os valores das variáveis utilizadas no programa a ser executado, o **R-470** após a introdução do comando RUN e antes da execução propriamente dita, limpa toda a região da memória reservada às variáveis, ou seja, quando um programa começa, não existe nenhuma variável definida.

Caso queiramos que um programa inicie a partir de uma determinada linha que não a primeira, basta indicar junto ao comando RUN o n.º da linha desejada.

9.5.2. Comando **REM**

Este comando tem uma característica interessante: não interfere nos programas. Serve, apenas, para fazer comentários, dar nomes a programas, sub-rotinas e etc., facilitando o entendimento dos mesmos.

EXEMPLO

```
10 REM ****************
20 REM * CONTAR ATÉ 10 *
30 REM ****************
40 FOR A=1 TO 10
50 PRINT A
60 NEXT A
```

obs: Não é necessária a utilização das aspas, pois não se trata de STRING.

9.5.3. Comando **STOP**

O comando STOP finaliza ou interrompe a execução de um programa. Saberemos que o programa foi interrompido pelo STOP pelo código de reportagem 9/(n.º da linha em que se encontrar este comando).

Podemos colocar o STOP em qualquer parte do programa.

EXEMPLO

```
10 PRINT "ESCREVA UM NUMERO"
20 INPUT Q
30 LET R=INT(Q/2)-Q/2
40 PRINT Q;
50 IF R=0 THEN GOTO 80
60 PRINT " - NUMERO IMPAR"
70 STOP
80 PRINT " - NUMERO PAR"
90 STOP
```

No exemplo acima, a função do STOP foi dar uma parada no programa no caso do número ser impar, senão ele imprimiria também a linha 80.

Para retomada de programa em caso de interrupção ou término, basta pressionar o comando CONT ou RUN, seguidos do ENTER.

9.5.4. Comando **BREAK**

Você já percebeu que às vezes é necessário que interrompamos um programa em andamento. Ou por não estar de acordo com o pretendido ou por já ter mostrado o que queríamos, enfim, existem vários motivos. Nestes casos, basta apertar a tecla ESPAÇO, que, durante a execução de um programa, tem a função de interrompê-lo, isto é, torna-se o comando BREAK.

EXEMPLO

```
10 PRINT "π";
20 GOTO 10
```

Perceba que, ao apertar o comando BREAK, o programa pára onde estiver e aparece, no canto inferior esquerdo, o código de reportagem, que, no caso, é representado pela letra D e o número da linha em que o programa foi interrompido.

9.5.5. Comando **CONT**

Se durante a execução, um programa for interrompido por: tela cheia ou a tecla BREAK, utilizamos o comando CONT, que irá recomeçar o programa do ponto em que parou. No caso de STOP ter provocado essa interrupção, será reiniciada na linha posterior a ele.

EXEMPLO

```
10 PRINT " RINGO ";
20 GOTO 10
```

9.5.6. Comando **LIST**

Com o decorrer do tempo você começará fazer programas mais complexos que, logicamente, serão mais extensos. Na tela de nosso Micro cabem apenas 22 linhas de programa e duas de edição. Vamos supor que você esteja fazendo um programa e que tenha ultrapassado às 22 linhas. Não precisa se preocupar, isso é muito normal. Nestes casos, o Micro vai guardando as linhas normalmente, apenas tirando de nossas vistas a última linha de cima. E assim vai: coloca uma linha em baixo, tira uma de cima. Agora, você quer ver alguma linha do programa que já não está mais aparecendo na tela. Para isso existem dois procedimentos:

1) QUER UMA LINHA ESPECÍFICA — Tecle LIST, o número da linha desejada e o ENTER.

2) QUER VER ALGUMAS DAS PRIMEIRAS LINHAS — Tecle apenas o LIST e o ENTER. Assim, aparecerão as primeiras linhas do programa.

9.6. Desvios

9.6.1. Comando **GOTO**

O **R-470** executa um programa contido em sua memória, numa ordem seqüencial crescente relacionada à numeração das linhas. Em alguns programas, porém, precisamos voltar ou ultrapassar várias linhas, dependendo do que queremos fazer; por isso, temos o comando GOTO. Ele "salta" para qualquer lugar do progra-

ma e executa as instruções a partir daquele ponto. Note algumas situações em que este comando pode ser utilizado:

EXEMPLO 1

```
10 PRINT "QUEM INVENTOU O AVIAO?"
20 PRINT
30 INPUT A$
40 IF A$ = "SANTOS DUMONT" THEN GOTO 70
50 PRINT "VOCE ERROU.";A$;" NAO E SEU NOME"
60 STOP
70 PRINT "VOCE ACERTOU.";A$;" E SEU NOME"
80 STOP
```

O GOTO, além de ser um comando de desvio, pode ser também, como o RUN, um comando de execução. A única diferença é que o GOTO não limpa as variáveis e o RUN sim. Ele pode executar desde a linha inicial ou da que você desejar, bastando que, em ambos os casos, seja determinado o número da linha.
Daremos um exemplo, iniciando na linha 30.

EXEMPLO 2

```
10 PRINT "14 BIS";
20 GOTO 10
30 PRINT "MIRAGE" ;
40 GOTO 30
```

9.6.2. Comandos FOR TO, NEXT e STEP

Às vezes, em um determinado programa é necessário repetir uma série de instruções. Para evitar que esta repetição ocupe muitas linhas utilizamos os comandos FOR TO, STEP e NEXT para fazer uma repetição controlada chamada de "Loop".

O comando FOR NEXT está associado à uma variável contadora. E como o número de repetições é controlado por esta variável, ele atinge seu final chegando à condição de teste. Também podemos, com este comando, especificar o valor inicial e o incremento do valor para a variável contadora.

Simplificando: FOR — variável contadora — valor inicial, TO — valor final, STEP — valor incrementado.

VARIÁVEL CONTADORA: É o nome da variável usada para contar o "Loop".
VALOR INICIAL: É o valor armazenado dentro da variável contadora antes do primeiro período do "Loop".

VALOR FINAL: É o valor que, quando atingido, conclui o "Loop".
VALOR INCREMENTADO: É um item opcional. Indica quanto aumenta ou diminui a variável contadora a cada período durante o "Loop". Se não for determinada assume o valor 1.
Atenção! O comando FOR não será executado sem a presença do TO.

A variável contadora sempre acompanha o comando NEXT. Sem ele, incorreremos num erro. No decorrer do "Loop", ao atingir a instrução NEXT, o valor da variável contadora é automaticamente incrementado com o valor do incremento. Se o valor da variável não foi ultrapassado, ele é desviado para a primeira instrução após o FOR. Quando ultrapassar o valor final, será executada a instrução seguinte ao comando NEXT.

EXEMPLO 1

```
10 FOR A = 1 TO 10
20 PRINT A ** 2
30 NEXT A
40 STOP
```

É possível a colocação de vários "Loops" dentro de outros, como mostra o exemplo abaixo.

EXEMPLO 2

```
10 FOR X=0 TO 21
20 FOR Y=1 TO 30
30 PRINT "■";
40 NEXT Y
50 PRINT X
60 NEXT X
```

obs: Sempre que um "Loop" estiver dentro de outro, o NEXT deve estar na ordem de colocação contrária ao FOR, como mostra o exemplo.

EXEMPLO 3

```
10 FOR A ...
20 FOR B ...
30 FOR C ...
.
.
.
.
.
60 NEXT C
70 NEXT B
80 NEXT A
```

Os comandos FOR-TO e NEXT são incrementados, automaticamente, de 1 em 1. Se for necessário mudar este valor, para mais ou menos, basta colocar o STEP na mesma linha do comando FOR. Veja.

EXEMPLO 4

```
10 FOR A=2 TO 40 STEP 2
20 PRINT A
30 NEXT A
```

Os comandos FOR-TO e NEXT são interdependentes, isto é, não podem aparecer em um programa sem que o outro esteja também. Isso já não acontece com o comando STEP. Ele pode ou não aparecer num programa em que os outros dois comandos estiverem.

9.7. Funções matemáticas e científicas

9.7.1. Função **RND**

Esta função serve para produzir números aleatórios entre 0 e 1, isto é $0 \leqslant RND < 1$.

São usados em programas que necessitam de números diferentes e indefinidos, sem uma seqüência lógica.

EXEMPLO 1

```
10 FOR A=1 TO 10
20 PRINT RND
30 NEXT A
```

Perceba que os números são completamente aleatórios.

No exemplo a seguir veremos um programa típico para o uso da função RND, onde os números solicitados têm que ser diferentes.

EXEMPLO 2

```
10 PRINT TAB 10; "PROGRAMA"
20 PRINT TAB 7; "ACERTE NA LOTO"
```

```
30 PRINT AT 4,0; "ESCOLHA QUANTAS  DEZENAS VOCE (4 espaços)
   QUER DE 5 A 10  (          )"
40 INPUT D
50 PRINT AT 5,16;D
60 IF D < 5 OR D > 10 THEN GOTO 200
70 DIM X(D)
80 FOR A=1 TO D
90 LET X (A) = INT (RND * 100)
100 FOR B = 1 TO A -1
110 IF X(A) = X(B) THEN GOTO 80
120 NEXT B
130 NEXT A
140 FOR A=1 TO D
150 PRINT AT 6+A,12;X(A)
160 NEXT A
170 STOP
200 PRINT,, "ESSE NÚMERO DE DEZENAS NÃO EXISTE NA LOTO, TENTE
    DE NOVO."
210 FOR A=1 TO 100
220 NEXT A
230 CLS
240 GOTO 10
```

Agora faça um teste e, se quiser, jogue na LOTO. Quem sabe você ganha.

9.7.2. Comando **RAND**

Toda vez que o **R-470** é ligado, uma série de números aleatórios é gerada por ele.

Esta série de números é requisitada através da função RND.

O comando RAND cria uma nova série para a geração destes números, tomando como referência o valor numérico ou variável que o acompanha, sendo que, para uma série de RNDs posteriores a um determinado RAND, a seqüência de resultados será sempre a mesma.

EXEMPLO

```
10 FOR A=0 TO 4
20 RAND 10
30 FOR B=0 TO 2
40 PRINT RND
50 NEXT B
60 NEXT A
```

É bom saber que toda vez que pedirmos um RND para um RAND 10, por exemplo, o resultado será sempre 0.01257342.

9.7.3. Função **SGN**

Com esta função podemos saber se a resposta de uma equação ou um número qualquer é igual a 0, positiva ou negativa, não informando, porém, seu valor numérico.

O seu resultado será:

1, se X > 0
0, se X = 0
-1, se X < 0

EXEMPLO 1

PRINT SGN (10 - 5)

EXEMPLO 2

PRINT SGN (10 - 10)

EXEMPLO 3

PRINT SGN -5

9.7.4. Função **ABS**

A função ABS é a função módulo, isto é, o seu resultado é sempre positivo.

EXEMPLO

PRINT ABS (25-86)

9.7.5. Função **SQR**

Esta função nos permite obter a raiz quadrada de qualquer número maior ou igual a 0.

EXEMPLO 1

PRINT SQR 25

EXEMPLO 2

Para calcularmos a expressão $3 \times \sqrt{5 + 3}$ procedemos da seguinte maneira:

*PRINT 3 * SQR (5+ 3)*

9.7.6. Função PI (π)

PI significa a relação entre o raio e o arco de uma circunferência. É muito usado nas funções trigonométricas, já que estas só operam em radianos. Seu valor é 3.1415926535 dízima.

EXEMPLO

*PRINT 42 - 5 * π*

9.7.7. Funções Trigonométricas

As funções trigonométricas são: SIN (seno), COS (cosseno), TAN (tangente), ASN (arco-seno), ACS (arco-cosseno) e ATN (arco-tangente), sendo possível calculá-las somente em radianos.

EXEMPLO 1

PRINT SIN (π/4)

EXEMPLO 2

PRINT COS (π/6)

EXEMPLO 3

*PRINT TAN (3*π/4)*

A função ASN é sempre calculada para ângulos entre -π/2 e π/2:

EXEMPLO 4

PRINT ASN 1

A função ACS é sempre calculada para ângulos entre 0 e π:

EXEMPLO 5

PRINT ACS Ø. 1

A função ATN também é sempre calculada entre -π/2 e π/2:

EXEMPLO 6

```
PRINT ATN 2.7
```

Para calcular em graus as funções SIN, COS e TAN, tome como exemplo o seguinte programa para SIN.

```
 5 INPUT A
10 PRINT SIN (A *π/180)
```

E, para calcular em graus as funções ASN, ACS e ATN, o seguinte:

```
 5 INPUT A
10 PRINT 180/π * ASN A
```

9.7.8. Funções Logarítmicas

As funções logarítmicas são LN e EXP.

9.7.8.1. Função **LN**

A função LN calcula o logarítmo natural na base e (2.7182818...).

EXEMPLO

```
PRINT LN 7.4
```

9.7.8.2. Função **EXP**

Esta função é inversa à LN. Calcula uma potência de base neperiana (e) e expoente igual ao argumento.

EXEMPLO

```
PRINT EXP 2. 00148
```

9.7.9. Função **INT**

A função INT considera somente a parte inteira de um número fracionário ou do resultado de uma expressão matemática.

EXEMPLO

```
10 PRINT "DIGITE UM NUMERO  FRACIONARIO"
20 INPUT N
```

```
30 PRINT N
40 LET I=INT N
50 PRINT "SUA PARTE INTEIRA E:"
60 PRINT I
```

Agora, experimente o número 3.2456

No exemplo acima podemos observar que a parte fracionária é ignorada pela função INT. Porém, caso você queira arredondar esse número, some 0.5 a ele antes de ser retirada a parte fracionária, colocando essa operação de adição entre parênteses:

```
70 LET I = INT(N + 0.5)
80 PRINT "ESSE N.º ARREDONDADO E:"
90 PRINT I
```

Para pré-fixar o número de casas depois da vírgula, sem arredondar, use a fórmula seguinte onde pré-fixamos em 2 casas:

```
100 LET I = INT(N*100)/100
110 PRINT "ATE 2 CASAS APOS A VIRGULA E:"
120 PRINT I
```

9.8. Testes

9.8.1. Operadores Matemáticos

Na linguagem BASIC encontramos sinais comparativos para que o Micro possa se localizar no que diz respeito à procedimento. Na verdade, ele não analisa profundamente uma expressão. Somente informa se é falsa ou verdadeira, através dos números: 0 (falso) e 1 (verdadeiro).

Este procedimento se repete em números, letras e STRINGs.

Os sinais são: = (igual)
> (maior que)
< (menor que)
>= (maior ou igual a)
<= (menor ou igual a)
<> (diferente de)

EXEMPLO 1

```
PRINT 3=3
```

EXEMPLO 2

```
PRINT 4>10
```

EXEMPLO 3

```
PRINT 305=10+5
```

Nos próximos exemplos veremos que o Micro também compara letras e palavras, tendo como escala a ordem alfabética.

EXEMPLO 4

```
PRINT "A">="A"
```

EXEMPLO 5

```
PRINT "BASICO"<"BASICA"
```

No exemplo abaixo, veremos que ele considera primeiro todas as letras do alfabeto para depois começar com números.

EXEMPLO 6

```
PRINT "Z" > "1"
```

Onde a comparação foi verdadeira o Micro colocou o número 1 e, onde foi falsa, o 0.

9.8.2. Comandos IF e THEN

O comando IF pode ser considerado como um comando inteligente ou um desvio condicional para tomada de decisões a partir de sinais já conhecidos de todos nós : =, >=, <=, <> e operadores lógicos. O THEN é um complemento de decisão, permitindo a execução de um desvio, impressão, atribuição de valores, etc, de acordo com o resultado do teste.

EXEMPLO

```
10 PRINT "DESCUBRA O NUMERO, ENTRE 0 E 10, QUE EU ESCOLHI."
20 LET X=INT(11*RND)
30 INPUT A
40 SCROLL
50 IF A=X THEN GOTO 80
60 PRINT "VOCE ERROU. TENTE DE NOVO"
70 GOTO 30
80 PRINT "PARABENS, VOCE ACERTOU."
```

Para que você entenda melhor, vamos analisar a linha 50.
IF (se) A = (igual a) X THEN (então) GOTO (vá para a linha) 80

LEMBRE-SE: O IF e o THEN têm sempre que estar juntos em cada linha de programa.

9.9. Sub-rotinas

9.9.1. Comandos GOSUB e RETURN

Durante o desenvolvimento de um programa, muitas vezes é necessário repetir uma mesma seqüência de linhas de instruções. Para poupar este trabalho, além de evitar o desperdício de linhas, devemos fazer uma sub-rotina, isto é, um programa fora do programa principal. Situando-se, na maioria das vezes, no fim do programa, a sub-rotina pode ser "chamada" tantas vezes quantas forem necessárias.

EXEMPLO 1

```
5 GOSUB 100
```

```
10 PRINT "O MAIS MODERNO"
20 GOSUB 100
25 PRINT "O MAIS PRATICO"
30 GOSUB 100
35 PRINT "O MAIS BONITO"
40 GOSUB 100
50 PRINT "UM GRANDE MICRO"
60 STOP
100 PRINT
105 FOR A = 0 TO 31
110 PRINT "■";
120 NEXT A
130 PRINT "RINGO"
140 PRINT
150 RETURN
```

O GOSUB é o comando que manda executar uma sub-rotina. Para que o **RINGO** saiba que sub-rotina executar, temos que indicar, ao lado do GOSUB, o número da linha em que ela se inicia.

O RETURN é o comando que faz retornar da sub-rotina para o programa principal. A partir daí, o programa continua na linha seguinte ao GOSUB que provocou o desvio.

Um programa pode conter vários comandos GOSUB e várias sub-rotinas. Em cada sub-rotina podemos ter vários comandos RETURN, como no seguinte exemplo:

EXEMPLO 2

```
10 INPUT A
20 GOSUB 100
30 IF B=0 THEN STOP
40 PRINT B
50 GOTO 10
100 LET B=0
110 IF A <=0 THEN RETURN
120 LET B=SQR A
130 RETURN
```

Este programa — para tirar raiz quadrada — pára se o número digitado fôr menor ou igual a 0.

Podemos, também, ter uma sub-rotina dentro da outra.

NÃO SE ESQUEÇA! PARA CADA GOSUB É PRECISO SEMPRE HAVER UM RETURN.

9.10. Gráficos

9.10.1. Comandos **PLOT** e **UNPLOT**

A tela do **RINGO** é formada de 22 linhas e 32 colunas, dando um total de 704 posições para impressão.

Com a utilização do PLOT ou UNPLOT, teremos 2816 posições, pois estes comandos usam apenas 1/4 de um caracter, multiplicando por 4 a capacidade gráfica da tela.

O comando PLOT imprime um elemento de imagem na tela, sendo necessário apenas fornecer as suas coordenadas cartesianas que serão representadas por X e Y.

EXEMPLO 1

```
PLOT 8,20
```

No exemplo, o número 8 representa a 8.ª posição da coordenada X, que é formada por 64 posições.

O número 20 representa a 20.ª posição da coordenada Y, que é formada por 44 posições.

O comando UNPLOT apaga os pontos que foram impressos pelo PLOT, usando os mesmos valores das coordenadas.

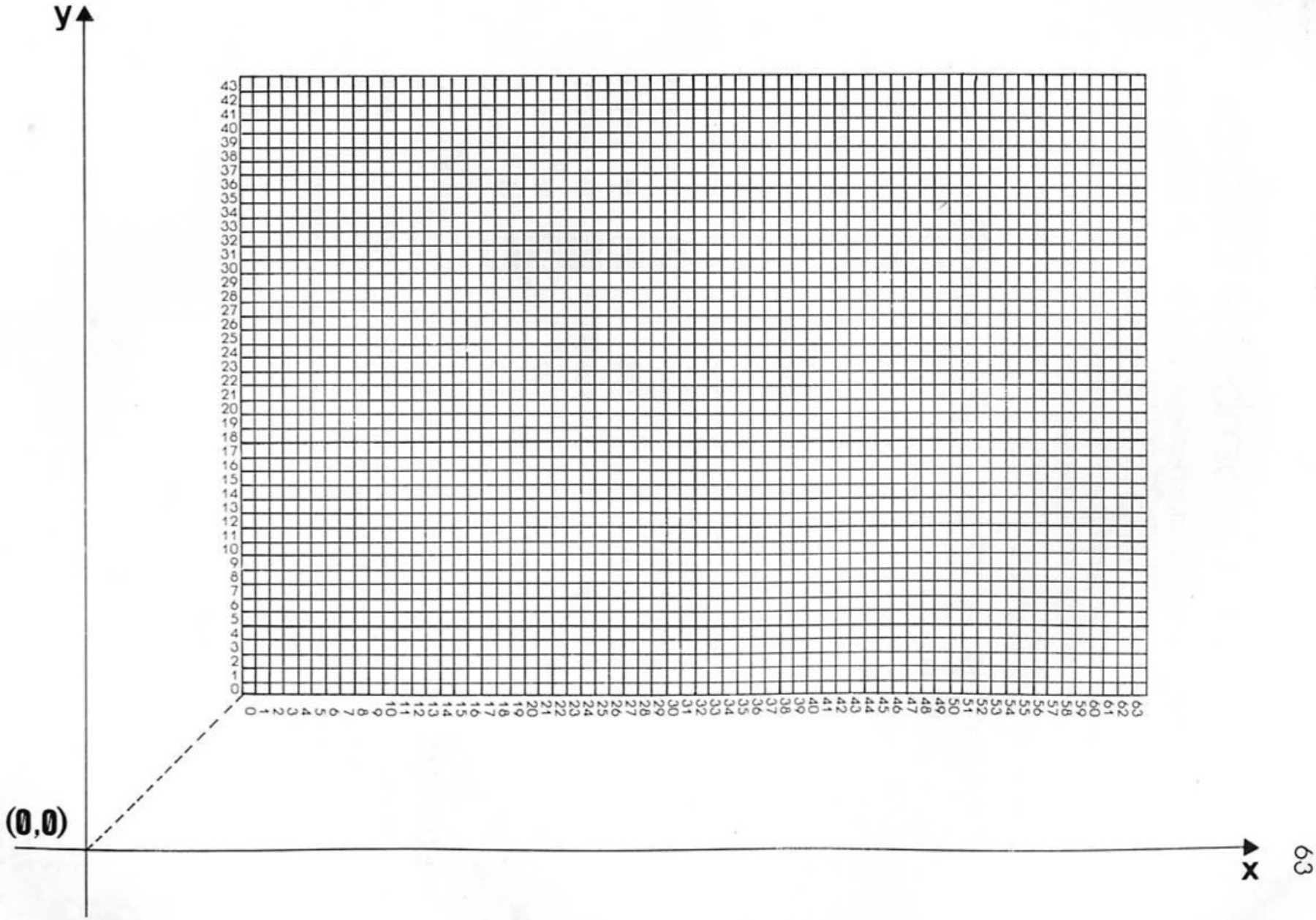
EXEMPLO 2

```
10 FOR Q=1 TO 50
20 PLOT Q,30
30 NEXT Q
40 FOR W=1 TO 50
50 UNPLOT W, 30
60 NEXT W
```

EXEMPLO 3

```
10 LET X = 0
20 LET Y = 0
30 PLOT X, Y
40 IF INKEY$ = "B" THEN LET Y = Y - 1
50 IF INKEY$ = "C" THEN LET Y = Y + 1
60 IF INKEY$ = "D" THEN LET X = X + 1
70 IF INKEY$ = "E" THEN LET X = X - 1
80 GOTO 30
```

PLOT x,y



9.10.2. Símbolos Gráficos

Quando queremos fazer algum desenho, uma moldura em um título ou tabela, enfim, qualquer tipo de decoração, nos utilizamos dos SÍMBOLOS GRÁFICOS. Para que tenhamos acesso a esses símbolos, basta apertar a tecla GRAPHICS (o cursor passará para G) e escolher entre os 22 SÍMBOLOS GRÁFICOS existentes no teclado, não esquecendo de manter a tecla SHIFT pressionada a cada símbolo digitado.

Estes são os 22 símbolos gráficos:

Código	Símbolo	Pressionar	Código	Símbolo	Pressionar
0		K ou L ESPAÇO	128		G ESPAÇO
1		G shift 1	129		G shift Q
2		G shift 2	130		G shift W
3		G shift 3	131		G shift E
4		G shift 4	132		G shift R
5		G shift 5	133		G shift T
6		G shift 6	134		G shift Y
7		G shift 7	135		G shift U
8		G shift 8	136		G shift I
9		G shift S	137		G shift F
10		G shift A	138		G shift D

EXEMPLO

```
10 PRINT TAB 7; "GRAFICO DE VENDAS",,,"MICRO A...▩",,
   "MICRO B...□",, "PROJECAO"
20 FOR A=0 TO 31
30 PRINT AT 19,A;"▬"
40 NEXT A
50 PRINT "ANO - 1983"; AT 20,15;"1984";AT 20,24;"1985"
60 FOR A=0 TO 2
70 PRINT AT 18-A,7;" □▩"
80 NEXT A
90 PRINT AT 15,8;"▩"
100 FOR A=18 TO 14 STEP - 1
110 PRINT AT A,16; " □▩"; AT A,25;"□"
120 NEXT A
130 PRINT AT 13,17;"▩"
140 FOR A =0 TO 9
150 PRINT AT 18-A,26;"▩"
160 NEXT A
```

9.11. Velocidade de processamento

9.11.1. Comandos **FAST** e **SLOW**

O **RINGO** ao ser ligado, inicia suas operações em SLOW. Nessa velocidade, ele executa qualquer programação, mostrando continuamente a tela. Porém, o Micro possui um segundo modo de operação chamado FAST, que trabalha com uma velocidade, aproximadamente, 4 vezes maior que no modo normal (SLOW), só que sem gerar sinal de vídeo. Ou seja, você só verá o resultado do programa na tela, ao fim da execução, ou se for interrompido por um comando PAUSE.

EXEMPLO

```
10 PRINT AT 10,10; "SLOW"
20 FOR A = 0 TO 70
30 NEXT A
40 CLS
50 GOSUB 200
60 FOR A = 0 TO 100
70 NEXT A
80 CLS
90 FAST
100 PRINT AT 10,10;"FAST"
110 PAUSE 150
120 CLS
130 GOSUB 200
140 SLOW
150 STOP
200 FOR A = 0 TO 63 STEP 0.5
210 PLOT A,22+20*SIN(A*π/32)
220 NEXT A
230 RETURN
```

obs: O comando FAST é muito útil para editar programas longos ou quando envolve cálculos matemáticos.

9.11.2. Comando **PAUSE**

Se for necessária uma pausa num determinado ponto do programa, você poderá utilizar o comando PAUSE em uma de suas linhas. Ele terá que vir sempre seguido de um n.º, que será o seu tempo de parada. O comando PAUSE pára a execução do programa e mostra a tela durante o tempo pré-determinado por você. Esse tempo pode variar de 0 a 32767, que equivale a ± 9 minutos e é contado da seguinte forma: o número que você colocar após o PAUSE será considerado como n.º simbólico, pois ele será dividido, pelo **RINGO**, por 60, que é o n.º de quadros que a televisão mostra por segundo. Se você ultrapassar essa faixa pré-estipulada de tempo, a pausa do programa será permanente.

EXEMPLO 1

```
PAUSE 240
240/60 = 4
```

O tempo real será de aproximadamente 4 segundos.

O PAUSE é melhor utilizado quando se trabalha em FAST, pois o SLOW produz, com esse comando, uma "piscagem" na tela, provocada por uma desincronização em sua varredura.

EXEMPLO 2

Pressione: FAST e ENTER.

```
10 FOR A = 0 TO 30
20 PRINT A
30 PAUSE 60
40 CLS
50 NEXT A
```

Caso queira interromper o tempo determinado por esse comando, aperte qualquer tecla, com exceção do BREAK, que seu programa voltará a executar.

Porém, se for necessário dar uma pausa na execução de um programa com velocidade SLOW, você poderá recorrer aos comandos FOR/NEXT para obter um tempo de parada.

EXEMPLO 3

```
10 PRINT "SLOW"
20 FOR A = 0 TO 200
30 NEXT A
40 CLS
```

Nesse exemplo, depois de aparecer na tela a palavra "SLOW", o programa deu uma pausa de 200 passos, isto é, o Micro executou o comando FOR/NEXT — linhas 20 e 30 — até o final da contagem e, em seguida, limpou a tela.

Utilizando

```
10 IF INKEY$ = "" THEN GOTO 10
```

você obterá uma pausa indeterminada, ou seja, o programa ficará interrompido até ser digitado alguma tecla (com exceção do BREAK).

9.12. Operações lógicas

9.12.1. Função **NOT** e Operações Lógicas **AND** e **OR**

Como vimos em capítulos anteriores, o Micro possui o poder de decisão em alguns casos específicos. Existem programas, porém, em que são necessárias condições simultâneas para que o **R-470** estabeleça seu próximo passo. Nestes programas usamos o NOT, o AND e o OR.

Uma maneira mais fácil de entendê-los — como na maioria das operações do BASIC — é traduzindo seus significados para o português:

AND = e
OR = ou
NOT = não

No interior do Micro não existe a matemática convencional, que é a que utilizamos em nosso dia-a-dia. Existe, sim, é a álgebra chamada Booleana. Nela, desconhecemos tudo o que for além da unidade, isto é, o número 1. Só existe o 0 e o 1. E é através destes dois números que o computador controla todo o seu funcionamento: memórias, circuitos e etc. É o princípio utilizado pelos computadores.

Já vimos no capítulo de Operadores Matemáticos que podemos ter acesso a esta álgebra através da análise que o Micro realiza de uma expressão: informa se é falsa (0) ou verdadeira (1).

Para que você entenda as tabelas a seguir, lembre-se que a álgebra Booleana segue os princípios da lógica (diferente da matemática convencional).

Podemos dizer que o AND é o produto lógico. Como vemos na tabela abaixo, só quando as duas entradas são 1 é que o resultado é 1.

AND

ENTRADA	ENTRADA	SAÍDA
0	0	0
0	1	0
1	0	0
1	1	1

A grande vantagem do **R-470** é que ele pode trabalhar não só com 0 e 1, mas também com variáveis e STRINGs. Sendo assim, ele funciona da seguinte forma:

Vamos supor que a primeira entrada da tabela acima é X e a segunda Y. Então:

X AND Y = X — se Y ≠ 0
X AND Y = 0 — se Y = 0

X$ AND Y = X$ se Y ≠ 0
X$ AND Y = "" se Y = 0

Para que o Micro possa trabalhar desta forma, repare que ele dá uma espécie de prioridade ao segundo operando, sendo este o que determina o resultado da operação lógica.

O OR pode ser considerado como uma soma lógica:

OR

ENTRADA	ENTRADA	SAÍDA
0	0	0
0	0	1
0	1	1
1	1	1

Nesta função, o Micro trabalha somente com variáveis numéricas, dando, igualmente, prioridade ao segundo operando. Veja.

X OR Y = 1 — se Y ≠ 0
X OR Y = X — se Y = 0

O NOT, por sua vez, é uma negação lógica.

NOT

ENTRADA	SAÍDA
1	0
0	1

NOT X = 0 — se X ≠ 0
NOT X = 1 — se X = 0

SIMPLIFICANDO

O AND, o OR e o NOT servem para que você possa obter resposta de uma operação que contenha vários testes em conjunto.

AND — Um **e** outro
OR — Um **ou** outro
NOT — Inverte

EXEMPLO 1

```
10 FOR A=1 TO 8
20 PRINT A;
30 IF A=3 OR A=6 THEN PRINT '' MULTIPLO DE 3'';
40 PRINT
50 NEXT A
```

Neste exemplo, a condição para que a frase solicitada fosse escrita era que o A fosse igual a 3 **ou** igual a 6.

EXEMPLO 2

```
10 PRINT ''DE DOIS NUMEROS ABAIXO DE 10''
20 INPUT A
30 INPUT B
40 IF A < 10 AND B < 10 THEN PRINT,, ''CERTO. '';A;'' E '';B;'' SAO
MENORES QUE 10''
50 IF A > = 10 AND B > = 10 THEN GOTO 90
60 IF A > 10 OR B > 10 THEN PRINT,,A; '' E '';B;'' NAO SERVEM; OS
DOIS TEM QUE SER MENORES QUE 10''
70 PRINT ,,,,
80 GOTO 10
90 PRINT,, ''ERRADO. '';A;'' E '';B;'' NAO SAO MENORES QUE 10''
100 PRINT ,,,,
110 GOTO 10
```

Neste exemplo vimos que, na linha 40, a condição só é satisfeita se o A **e** o B são menores que 10.

Na linha 50, a condição só é satisfeita se o A **e** o B são maiores ou iguais a 10.

Na linha 60, a condição só é satisfeita se o A **ou** o B são maiores que 10.

9.13. Matrizes

9.13.1. Comando **DIM**

Com o comando DIM é possível reservar espaço na memória do **RINGO** para uma matriz com n dimensões a serem estabelecidas por você, bastando apenas definir um nome para essa matriz, que precisa ser, necessariamente, uma única letra. E, logo em seguida, entre parênteses, as dimensões da mesma, podendo, assim, introduzir elementos alfanuméricos ou numéricos. Veja os exemplos abaixo:

EXEMPLO 1

```
10 DIM A (5)
20 LET A (1)=564
30 LET A (3)=10
40 LET A (4)=5
50 LET A (5)=92
```

Suponha que esse seja o espaço reservado na memória do Micro determinado pelo comando DIM para a matriz A:

	1	2	3	4	5
1	564	0	10	5	92

Agora, digite o comando PRINT juntamente com o nome da matriz e as coordenadas desejadas e o Micro mostrará o n.º correspondente na tela.

PRINT A (1)	Aparecerá na tela: 564
PRINT A (2)	Surgirá o n.º 0, pois o **RINGO** zera todo o espaço de memória reservado que não foi definido.
PRINT A (5)	Surgirá o n.º 92

EXEMPLO 2

```
10 DIM B (2,2,2)
20 LET A=1
30 FOR Z=1 TO 2
40 FOR Y=1 TO 2
50 FOR X=1 TO 2
60 LET B (X,Y,Z)=A
70 PRINT "B(";X;",";Y;",";Z;") = ";B(X,Y,Z)
80 LET A=A+1
90 NEXT X
100 NEXT Y
110 NEXT Z
120 STOP
```

Neste programa, como você já deve ter notado, o **R-470** guardou números de 1 a 8 na matriz B com dimensões (2,2,2) e escreveu na tela o processo de armazenamento.

Você também pode dimensionar matrizes alfanuméricas usando, após o nome da matriz, o $. Nesse caso, para cada coordenada deve-se colocar apenas um caracter.

EXEMPLO

```
10 DIM A$ (2,5)
20 LET A$ (1) = "MICRO"
30 LET A$ (2) = "R-470"
```

	1	2	3	4	5
1	M	I	C	R	O
2	R	—	4	7	0

PRINT A$ (1,3) Aparece: C

PRINT A$ (2,2) Aparece: —

PRINT A$ (1) Aparece: MICRO

PRINT A$ (2) Aparece: R-470

obs.: Em matrizes alfanuméricas de 2 dimensões (-,-), além de elementos isolados, também conseguimos recuperar da memória a linha inteira, digitando apenas seu nº; o que não ocorre em matrizes numéricas. Porém, em matrizes de mais dimensões será necessário que se escreva nºs de linha e coluna para que isso aconteça.

9.14. Operações com variáveis alfanuméricas

9.14.1. Função **LEN**

Para se obter o comprimento ou o número de caracteres de uma STRING é necessário o uso da função LEN. Esta função conta o número de caracteres dentro de uma STRING. Os caracteres não impressos e vazios também são contados.

EXEMPLO 1

```
10 LET A$ = "ATENAS"
20 PRINT LEN (A$)
```

EXEMPLO 2

```
10 INPUT A$
20 LET A = 12 - (LEN A$)/2
30 LET B = 15 + (LEN A$)/2
40 FOR X=A TO B
50 PRINT AT 8,X; "*"; AT 12,X;"*"
60 NEXT X
70 FOR X=1 TO 3
80 PRINT AT 8+X,A;"*";AT 8+X,B;"*"
90 NEXT X
100 PRINT AT 10,A+2;A$
```

9.14.2. Função STR$

A função STR$ converte um determinado número em uma STRING, isto é, o Micro lê um número como se fosse uma palavra, não podendo, portanto, esta palavra ser utilizada em operações matemáticas e sim em funções alfanuméricas.

EXEMPLO

```
10 INPUT A
20 PRINT "NUMERO: ";A;" + 1 = ";A + 1
30 PRINT
40 LET S$=STR$ A
50 PRINT "STRING: "; S$; " + 1 = "; S$ + "1"
```

Na linha 10 o programa pede para você determinar o valor de A. Coloque um número. Feito isso, aperte a tecla ENTER. Em seguida, aparecerão 2 frases.

A primeira, como você deve ter notado, compõe uma operação matemática de adição entre dois números. Já na segunda, quando o número é considerado uma STRING (palavra), o sinal (+) determina uma posição seqüencial, isto é, o resultado forma uma nova STRING, com números dispostos um após o outro. Observe que na linha 50 o nº 1 está entre aspas. Isto porque, como você deve lembrar-se, toda STRING precisa estar sempre entre aspas.

9.14.3. Função VAL

A função VAL é a forma inversa da função STR$. Ela transforma um número que

está como STRING (encarado como palavra) em um número que se pode manusear em operações matemáticas.

EXEMPLO

```
10 INPUT A$
20 PRINT "STRING: ";A$;" + 1 = ";A$ + "1"
30 LET V= VAL A$
40 PRINT
50 PRINT "NUMERO: ";V;" + 1 = ";V + 1
```

Entre com um n.º.

Este exemplo é do mesmo tipo àquele exposto na Função STR$. A única mudança é que a partir da linha 30, a STRING numérica introduzida, é transformada pela função VAL em número, efetuando, assim, a operação matemática de adição.

9.14.4. Função CHR$

O **RINGO** possui uma tabela onde guarda todos os seus caracteres, comandos e funções, num total de 256, diferenciados por números decimais. Para que possamos ter acesso a esta tabela, é que existe a função CHR$. Conheça-os através do exemplo abaixo.

EXEMPLO 1

```
10 FOR A= 0 TO 255
20 PRINT CHR$ A;
30 NEXT A
```

Repare que todos os caracteres, comandos e funções do teclado estão impressos na tela. Para a conferência dos mesmos, individualmente, proceda da seguinte forma:

EXEMPLO 2

```
PRINT CHR$ 36
```

9.14.5. Função CODE

Esta função é inversa à CHR$. Ao invés de nos mostrar o caracter que nós pedimos através de um código, nos mostra o código solicitado através de um caracter.

EXEMPLO

```
PRINT CODE "A"
```

Repare que aparece o número 38 no canto superior da tela. Este é o código da letra A. (Veja tabela de códigos no final do manual).

9.15. Fita cassete

9.15.1 Armazenamento em fita

O armazenamento de programas em fita é muito simples. A primeira coisa a fazer é pegar um gravador comum que possua as saídas EAR e AUX, do tipo JACK fêmea e conectar ao par de cabos com plugs que acompanha o computador. Olhe minuciosamente seu gravador; será muito fácil encontrar essas saídas. Depois de feita a devida conexão, verifique se seu gravador possui controle automático de nível de gravação. Caso não possua, você terá que efetuar ajustes de nível entre 0 e +2DB (quase atingindo a faixa vermelha). O próximo passo é arranjar uma fita virgem; você pode estar certo que ela lhe proporcionará grandes satisfações, pois, possibilitará a você salvar seus programas feitos com tanto carinho e dedicação.

Como já dissemos, ao desligar o Micro **RINGO** tudo que estiver na memória está perdido. Mas com a fita, você poderá usar o programa quantas vezes necessitar.

Utilizando apenas 2 comandos: SAVE e LOAD, podemos efetuar o armazenamento em fita.

9.15.2 Comando **SAVE**

Para a transferência de um programa completo, armazenado na memória RAM, para a fita cassete, devemos usar o comando SAVE.

EXEMPLO

```
10 REM PROGRAMA PARA TESTE DO SAVE
20 FOR A=0 TO 21
30 PRINT TAB A; "RINGO"
40 NEXT A
```

Não é necessário executar o programa para efetuar a transferência da memória à fita cassete.

Digite juntamente com o comando SAVE, o nome do programa sem apertar ainda a tecla ENTER.

```
SAVE "TESTE"
```

Obs: Para você acompanhar o processo de gravação, mantenha a tela com o fundo branco.

A esta hora, você já deve estar com o gravador preparado. Pressione, então, o PLAY e o REC do gravador e, logo em seguida, a tecla ENTER do computador. Feito isso observe a TV:

Durante 5 segundos, o vídeo ficará apagado, surgindo, logo após, listas horizontais. Se quiser, aumente o volume de som da televisão. Provavelmente, no início, você se espantará com os ruídos agudos que emanará da TV. Porém, fique tranqüilo: os ruídos são códigos sonoros sendo gravados em fita. Após mais alguns segundos, a tela voltará ao normal com a indicação 0/0. Imediatamente, pare o gravador, pois o armazenamento do programa em fita já foi completado.

O Micro **RINGO** estava mandando um sinal através do AUX do gravador e, ao mesmo tempo, um sinal para a TV. No período em que a tela apagou, o programa ainda não estava sendo processado, se efetivando somente com o aparecimento das listas horizontais. Depois disso, a tela voltou ao normal com a sinalização 0/0 no canto inferior esquerdo do vídeo, indicando que o programa já havia sido gravado.

Para ter certeza se o programa foi mesmo gravado, desconecte o cabo do gravador da saída EAR, volte a fita ao início e ouça-a. Você ouvirá um zumbido suave e depois 5 segundos de silêncio. O programa aparece na fita como um ruído estridente e alto, finalizando com outro mais suave.

Se você não ouvir nada disso, verifique se o gravador está interligado corretamente e se o nível de gravação está bem ajustado.

9.15.3 Comando LOAD

O comando LOAD, procedendo inversamente ao comando SAVE, carrega o programa da fita para o computador. Para tanto, coloque a fita no início e verifique se o nível de som do gravador está em torno de 50% a 75% de volume, pois, para o Micro captar o programa todo, juntamente com a parte silenciosa e não haver qualquer distorção, o volume precisa estar bem regulado. Se seu gravador tiver controle de tonalidade, ajuste o nível de agudo ao máximo e o grave, ao mínimo. Após essa averiguação, interligue corretamente os plugs e pressione a tecla LOAD seguida pelo nome do programa.

LOAD "TESTE"
ENTER

Agora, pressione o PLAY do gravador e, após alguns segundos, você verá novamente listas horizontais no vídeo. Quando a tela voltar ao normal e aparecer o código 0/0, isto estará indicando que o programa já foi carregado, bastando a você apertar a tecla LIST para ele aparecer na tela. O tempo que leva para carregar um programa depende de seu número de linhas; mas se você notar uma demora exagerada ou aparecer formas verticais na tela, pressione a tecla BREAK para interromper o processo, pois algo de errado deve estar acontecendo. Verifique se a conecção dos plugs, o nível de volume e a digitação do nome estão corretos. Após conferir estes itens, pressione LOAD e o nome do programa novamente.

Caso você não tenha o nome do programa, utilize um outro recurso que o **RINGO** oferece:

Pressione:
LOAD "" ENTER

Em seguida, pressione PLAY. Digitando uma STRING vazia no lugar do nome, será transferido para o computador o primeiro programa que o Micro encontrar na fita.

Os comandos SAVE e LOAD podem, também, ser utilizados como parte de um programa, isto é, ocupando uma linha.

Caso você queira "salvar" todo o programa e suas variáveis —gravando-os em fita— sem a necessidade de interrompê-lo, utilize o comando SAVE em uma linha de programação. Logo após o término da gravação, ele voltará automaticamente para a próxima linha àquela destinada ao comando SAVE.

EXEMPLO

```
10 FOR A=0 TO 20
20 PRINT A
30 IF A=15 THEN SAVE "TESTE"
40 NEXT A
```

AINDA NÃO DIGITE **RUN ENTER**

Esse exemplo faz uma contagem de 0 a 20. Porém, um pouco antes dele chegar ao n.º 15, prepare o gravador apertando o PLAY e o REC, pois, segundo a linha 30, exatamente nesse número, inicia-se a gravação, retornando, no final, para o próximo passo do programa, isto é, aparecendo em bloco do n.º 0 ao 15 e recomeçando a contagem a partir do n.º 16. Agora você já pode executar o programa. Aperte RUN ENTER e confira.

Pressionando a tecla ENTER, você notará que na linha 30 — a do SAVE — após o retorno do programa para o Micro, a última letra da palavra "TESTE" está escrita em vídeo-inverso. Esse recurso é utilizado pelo **RINGO** para detectar o término do nome do programa.

Quando utilizamos o comando LOAD para transferirmos, da fita para a memória do Micro, um programa que foi salvo por um SAVE contido numa de suas linhas, ele continuará também na linha posterior à do comando SAVE.

Recorrendo a esse processo de utilização dos comandos SAVE e LOAD, o programa ao ser carregado no Micro será executado sem precisar do comando RUN ou GOTO para isso.

Obs.: Numa fita, você poderá armazenar muitos programas. Para localizá-los com maior facilidade e saber corretamente o nome com o qual foi gravado (caso tenha esquecido), é aconselhável que, antes de utilizar o SAVE, você grave com sua voz os nomes dos programas. Para isso, desconecte o cabo da entrada Auxiliar e utilize o microfone. Após ter gravado o nome, não esqueça de ligar novamente o cabo da entrada AUX.

9.16. Impressora

9.16.1. Comandos **COPY, LPRINT** e **LLIST**

Estes comandos nos permitem a utilização de impressora acoplada ao Micro.

9.16.1.1. Comando **COPY**

Este comando faz com que a impressora copie o que estiver na tela. No caso de a tela estar em movimento, precisamos pará-la para que a impressora entre em ação.

9.16.1.2. Comando **LPRINT**

Este comando tem a mesma função do comando PRINT, só que ao invés de passar para a tela, passa para a impressora.

9.16.1.3. Comando **LLIST**

Tem a mesma função do comando LIST, só que em vez de listar o programa na tela, lista na impressora.

9.17. Código de máquina

9.17.1. Linguagem de máquina

Existe um outro modo de escrever programas no **RINGO** além do BASIC. É a linguagem de máquina. Essa linguagem é usada pelo Micro em seu processamento interno para ele entender e executar instruções.

Ao utilizar diretamente a linguagem de máquina, você evitará todo um trabalho de decodificação que o Micro comumente faz, aumentando sua velocidade. Outra vantagem é que, com isso, se ocupa um espaço menor de memória.

O **R-470** possui como centro de decisões a CPU-Z80. Ela lê, interpreta e executa instruções, faz operações matemáticas e controla o vídeo e o teclado. Porém, essa CPU tem um modo especial para trabalhar com números. Quando você digita um nº no sistema decimal, isto é, na base 10, o Micro, internamente, transforma-o em

sistema binário, pois a CPU só trabalha com n.^{os} na base 2; portanto, seus sinais elétricos só podem assumir 2 estados; 0 ou 1 — desligada ou ligada. Exemplo: 0 volt, ou 5 volts. Mas para mostrar o resultado na tela, o computador colocará o n.º no sistema decimal.

Quando elaborar um programa em linguagem de máquina, você sentirá maior facilidade efetuando um acompanhamento do processo de operação interna do Micro. Para isso, utilize a linguagem ASSEMBLER que são palavras-chave que dão uma noção de todo esse processo através de uma listagem (fora do Micro) feita por você.

9.17.2. Sistemas numéricos

A CPU-Z80 do **RINGO** trabalha com palavras — Bytes — que são localizações de memória. O lugar destinado ao armazenamento de dados chama-se RAM (RANDOM ACCESS MEMORIES) que vai da posição de memória 16384 a 32767, no caso dela ser de 16 Kbytes ou até a 65535, em memória com 48 Kbytes. Um Kbyte é formado por 1024 Bytes (2^{10}) e cada Byte por 8 bits. Ou seja: 1 Byte compõe 8 sinais elétricos: 0 ou 1. Porém, para você acompanhar todas operações exercidas pelos bits (0 ou 1) — sistema binário — será muito mais prático, transformá-los em HEXADECIMAL, que são números na base 16.

Você entenderá melhor, consultando a tabela abaixo e prestando atenção nos exemplos:

$DECIMAL_{(10)}$		$HEXADECIMAL_{(16)}$
0	=	0
1	=	1
2	=	2
3	=	3
4	=	4
5	=	5
6	=	6
7	=	7
8	=	8
9	=	9
10	=	A
11	=	B
12	=	C
13	=	D
14	=	E
15	=	F

EXEMPLO 1

$4350_{(10)} \rightarrow \_\_\_\_ _{(16)}$ = 4350 |16

115 271 |16

030 111 16 |16

[14] [15] [0] [1]

A cadeia de operações matemáticas de divisão deve ser interrompida quando o último quociente for menor que 16 e serão ordenados, para consulta de Tabela, a partir dele, ou seja: de maneira inversa.

[1] [0] [15] [14] = 10FE

EXEMPLO 2

$183_{(10)} \rightarrow \_\_\_\_ _{(16)}$ = 183 |16

023 [11]

[07]

Consulte agora a Tabela para transformar o quociente (11) e o resto (7) que estão no sistema decimal para o hexadecimal.

[11] [7] = B7

Caso você queira digitar um n.º no Micro e ele esteja na forma Hexadecimal, transforme-o para Decimal da seguinte forma:

10FE = [1] [0] [15] [14] =

$1 * 16^3 + 0 * 16^2 + 15 * 16^1 + 14 * 16^0 = 4350$

B7 = [11] [7] =

$11 * 16^1 + 7 * 16^0 = 183$

10. PENETRANDO NAS MEMÓRIAS DO R-470

Até agora, você tem visto instruções que lhe facilitam a utilização dos recursos da linguagem BASIC, porém, sem ter acesso direto aos valores contidos nas memórias do Micro. Para isso, temos a função PEEK e o comando POKE que lhe permitem completo acesso à memória RAM ou ROM* do **RINGO.**

* ROM (READ-ONLY MEMORIES) — Memória que contém todos os comandos e funções do BASIC. Ela permanece constante mesmo com o Micro desligado.

10.1 Função **PEEK**

A função PEEK permite que você tenha acesso ao conteúdo de uma posição de memória, isto é, ela sempre assume o valor desse conteúdo sem, todavia, alterá-lo. Para que isso ocorra, o PEEK deve vir sempre acompanhado do endereço de memória desejado. Observe:

EXEMPLO 1

```
PRINT PEEK 16389
```

Deverá aparecer o n.º 128 que é o conteúdo do endereço 16389 da memória RAM.

EXEMPLO 2

```
10 FOR A=0 TO 8191
20 SCROLL
30 LET X=INT (PEEK A/16)
40 LET Y=PEEK A-X*16
50 PRINT A; " - "; CHR$ (X+28); CHR$ (Y+28); " = "; PEEK A
60 NEXT A
```

Esse programa fará uma listagem completa da memória ROM, colocando em seqüência: Endereço — seu conteúdo em Hexadecimal — seu conteúdo em Decimal. Caso queira interromper sua execução, como você já sabe, basta pressionar o BREAK.

Leia atentamente todas as instruções contidas nas linhas deste programa. Você vai notar que poderá entendê-las facilmente. Na linha 50 foi somado 28 às variáveis X e Y para obtermos o valor correto de acordo com a Tabela de Caracteres.

10.2. Comando **POKE**

Com esse comando você pode guardar numa posição de memória da RAM um n.º desejado. Para tanto, basta colocar esse n.º — que pode variar de 0 a 255 — após

o n.º de endereço, que vai da posição 16384 até 32767 — caso a memória seja de 16 Kbytes ou até 65535 — se for de 48 Kbytes.

EXEMPLO

```
POKE 20000,147
```

Verifique se o n.º 147 está realmente nesse endereço:

```
PRINT PEEK 20000
```

Aparecerá na tela: 147

10.3 Função **USR**

É usada para se ter acesso a uma rotina em linguagem de máquina. Ela age como o GOSUB do BASIC.

A função USR vai até o endereço indicado de memória e considera os códigos lá encontrados como linguagem de máquina, continuando sua decodificação para os endereços posteriores e só retornando ao BASIC após localizar uma instrução RET do Z80 (Return).

O Micro comporta uma série de registradores que serão especificados no decorrer desse manual; porém, se você quiser usar algum resultado de rotina em linguagem de máquina, o n.º deverá ser colocado no par de registradores BC antes da instrução de retorno. Se não houver alterações durante toda a execução da função USR, o par BC assumirá o n.º do endereço indicado.

EXEMPLO 1

```
10 POKE 20000,201
20 PRINT USR 20000
```

Tela: 20000

Na linha 10 do exemplo, o POKE guardou o código 201 no endereço 20000. A função USR decodificou o 201 que significa RET e retornou sem alterações para o BASIC, deixando no par de registradores BC o n.º de endereçamento: 20000. O comando PRINT escreveu esse n.º na tela.

EXEMPLO 2

```
10 POKE 20000,1
20 POKE 20001, 65
30 POKE 20002, 2
40 POKE 20003,201
50 PRINT USR 20000
```

Resultado: 577

O POKE guardou os códigos nos endereços indicados. A função USR decodificou todos os n.ºs armazenados desde o endereço 20000 até encontrar o código referente a instrução RET(201), que fez retornar à linguagem BASIC. E o PRINT escreveu na tela o n.º colocado no par BC: 577.

Observe o que ocorreu no programa com a utilização desses códigos:

O código 1 da linha 10 carregou o par BC com os n.ºs dos 2 Bytes seguintes, só que de maneira inversa:

Decimal		Hexadecimal
1	=	01
65	=	41 — C
2	=	02 — B
201	=	C9

Par BC

0 2 4 1 h

Transformando para o Sistema Decimal, teremos:

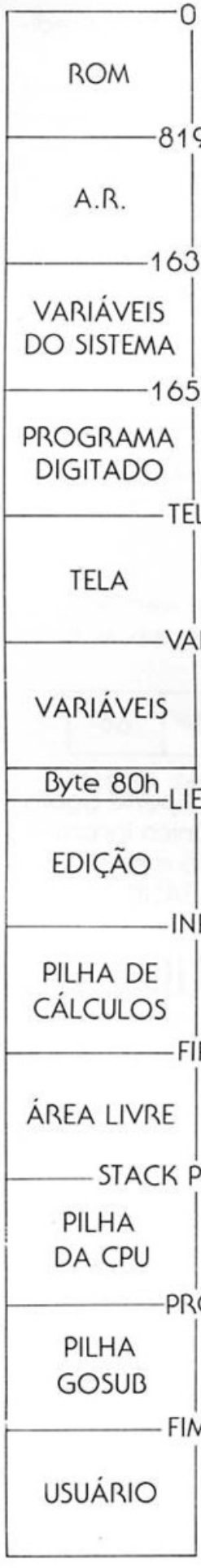
$$0 * 16^3 + 2 * 16^2 + 4 * 16^1 + 1 * 10^0 = 577$$

Quando o Micro estiver trabalhando em SLOW, você não poderá usar os registradores AF', IX, IY, I e R, pois eles são utilizados pela rotina que mostra a tela. Usando a velocidade de programação FAST existem também algumas restrições. Quando trabalhar com os registradores IY e I, se houver alterações em seus conteúdos, o IY deverá ter o valor 4000h e o I, o valor de 1Eh —antes de retornar ao BASIC— para que haja um perfeito funcionamento do programa.

O lugar seguro para se escrever rotinas em linguagem de máquina num programa, de forma que não sejam alteradas pelo BASIC, pode ser: nas primeiras linhas de programa em um comando REM ou no fim da memória RAM, reservando-se o espaço com o comando NEW.

10.4. Organização da memória

O **RINGO** divide sua memória em diversas partes, cada uma para um determinado fim. Veja, a seguir, a representação dessa divisão:



Da posição 0 a 8191 encontra-se a ROM (READ-ONLY MEMORIES) que é uma memória fixa, que se mantém constante mesmo com o Micro desligado. Nela está armazenado todo o conhecimento do **RINGO** sobre linguagem BASIC, controle de tela e teclado.

De 8192 a 16383: área reservada.

Aqui inicia a memória RAM (RANDOM ACCESS MEMORIES).

Da posição 16384 à 16508 existe o espaço para as variáveis do sistema.

A partir de 16509 ficam os programas que já foram digitados e aceitos pelo Micro.

Área reservada para a tela.

Espaço destinado para guardar os nomes e valores de variáveis numéricas e alfanuméricas.

Byte 80h — indica o fim da área de variáveis.

Área destinada à linha de programa que está sendo editada e para entrada de dados.

Caso seja preenchido o espaço da área de variáveis do sistema destinado ao armazenamento temporário dos resultados de operações matemáticas, o Micro passará a utilizar a pilha de cálculos.

Espaço livre de memória utilizável.

Área utilizada pela CPU — Z80 — para sua pilha (STACK).

Guarda o endereço de retorno do comando GOSUB — BASIC.

Espaço para armazenar programas em linguagem de máquina onde o BASIC não interfere.

Cada divisão de memória RAM, a partir do endereço 16509, é alterada a medida que vai se digitando instruções no Micro.

Você pode reservar um espaço para programas em linguagem de máquina no fim da memória, através do comando NEW. Antes disso, porém, deve-se determinar o n.º de Bytes que será utilizado e estabelecer o novo endereço final da RAM, guardando-a na variável FIME — FIM DA MEMÓRIA — (16388 e 16389). Ou seja: o Micro, ao receber o comando NEW, apagará todo o conteúdo da RAM e considerará como fim de memória o endereço colocado nesta variável e não o verdadeiro final.

Vamos, através de um exemplo, reservar um espaço de 100 Bytes. Em primeiro lugar, deve-se saber o endereço final da RAM:

```
PRINT PEEK 16388+PEEK 16389*256
```

Deverá aparecer o n.º 32768 que é o endereço final da memória (32767) + 1. Agora, subtraia o n.º de Bytes que deseja do endereço final.

```
32767-100=32667
```

Coloque esse novo endereço na variável FIME:

```
POKE 16389,INT (32667/256)
POKE 16388,32667-PEEK 16389*256
```

Se quiser conferir, digite novamente a linha do primeiro exemplo. Aperte agora a tecla NEW e ENTER e o espaço de memória estará reservado. O único inconveniente é que o SAVE não armazenará em fita esta região de memória e essa operação deve ser realizada antes de se fazer qualquer programa em BASIC.

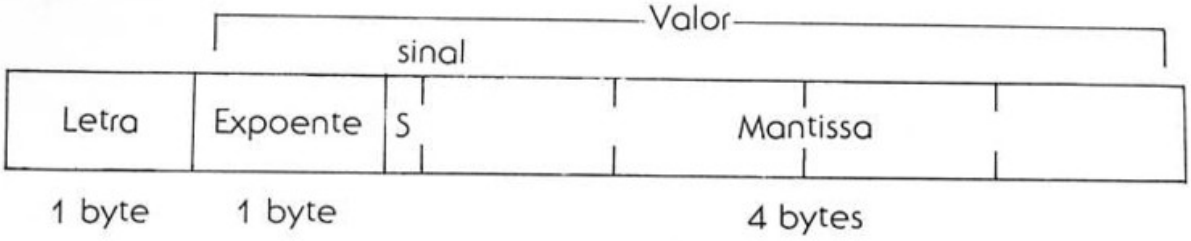
10.5. Processamento interno do **RINGO**

Uma linha de programa digitada no Micro tem, em seu processo interno, a seguinte forma:

Número de linha	Comprimento do texto + ENTER	Texto	76h (ENTER)
2 bytes	2 bytes	n bytes	1 byte

No **RINGO**, como você já sabe, existe um espaço de memória reservado às variáveis. Elas podem ser: numéricas, alfanuméricas, matrizes e do "Loop" FOR-NEXT. Para guardá-las nesse espaço e identificar cada tipo de variável, o Micro possui um código específico que, juntamente com o n.º da letra, gera um código final. Para facilitar seu entendimento, representamos abaixo todo o processo:

— variável numérica com somente uma letra:



EXEMPLO 1

LET A = 1Ø

Código da letra no Sist. Binário — 0010 0110
Código do Micro — 0110 0000 (OR)

Código final — 0110 0110 = 66h = (102)d

h = hexadecimal
d = decimal

66	84	20	00	00	00
(102)	(132)	(32)	(0)	(0)	(0)

LET B = – 1Ø

67	84	A0	00	00	00
(103)	(132)	(160)	(0)	(0)	(0)

LET C = 2E3

68	8B	7A	00	00	00
(104)	(139)	(122)	(0)	(0)	(0)

LET D = 2E – 3

69	78	03	12	6E	97
(105)	(120)	(3)	(18)	(110)	(151)

— variável numérica com mais de uma letra:

Letra	Segundo caracter		Último caracter	Valor
101 ------	00 ------		10 ------	5 bytes

EXEMPLO 2

LET RINGO = 10

B7	2E	33	2C	B4	84	20	00	00	00
(183)	(46)	(51)	(44)	(180)	(132)	(32)	(0)	(0)	(0)

— variáel alfanumérica:

Letra (— 20h)	Número de caracteres	Texto da STRING (pode ser vazia)
010 _ _ _ _ _	2 bytes	n bytes

EXEMPLO 3

LET A$ = "R-470"

46	06	00	37	16	20	23	1C
(70)	(6)	(0)	(55)	(22)	(32)	(35)	(28)

— matriz de números:

Letra (-20h)	Tamanho * da matriz	Número de dimensões	Primeira dimensão		Última dimensão	Elementos
100-----	2 bytes	1 byte	2 bytes		2 bytes	5 bytes cada

* Total de elementos & dimensões + 1 para número de dimensões.

EXEMPLO 4

DIM E (2,3)

8A	23	00	02	02	00	03	00	5 bytes para cada n.º
(138)	(35)	(0)	(2)	(2)	(0)	(3)	(0)	6 números

— matrizes alfanuméricas:

Para esse tipo de matriz, ao invés de 5 bytes para cada número, é reservado 1 byte para cada caracter.

EXEMPLO 5

DIM E$ (2,3) (1 byte para cada caracter: 6 bytes)

CA	0B	00	02	02	00	03	00	00	00	00	00	00	00
(202)	(11)	(0)	(2)	(2)	(0)	(3)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

— variável do loop-FOR-NEXT:

Letra	Valor	Limite	Passo	Linha de looping
111_____	5 bytes	5 bytes	5 bytes	2 bytes

EXEMPLO 6

10 FOR V=1 TO 10 STEP 2

FB	81	00	00	00	00	84	20	00
(251)	(129)	(0)	(0)	(0)	(0)	(132)	(32)	(0)

00	00	82	00	00	00	00	0B	00
(0)	(0)	(130)	(0)	(0)	(0)	(0)	(11)	(0)

10.6 Variáveis do sistema

DECIMAL	HEXA	NOME	NOTAS
16384	4000	CORE	Código de reportagem. O valor é incrementado antes de ser escrito.
16385	4001	BCT1	BIT 0 — supressão do espaço anterior ao comando. BIT 1 — controle da impressora. BIT 2 — seleciona modo do K ou L, ou F ou G. BIT 6 — N.º em ponto flutuante ou parâmetros de STRING. BIT 7 — reset durante verificação de sintaxe.
16386	4002	PRGO	Pilha de retorno do GOSUB.
	4003		
16388	4004	FIME	Fim da memória.
	4005		
16390	4006	MODO	Contém o código para K ou F.
16391	4007	NILSE	N.º da linha sendo executada.
	4008		
16393	4009	FASA	Início da RAM que é salva em fita.
16394	400A	LBAC	Linha do BASIC apontada pelo cursor.
	400B		
16396	400C	TELA	Início do arquivo de imagem.
	400D		
16398	400E	EPPA	Endereço para posição do PRINT AT.
	400F		
16400	4010	VARI	Início da área de variáveis.
	4011		
16402	4012	ENVA	Endereço da variável em atribuição.
	4013		
16404	4014	LIED	Início da linha de edição.
	4015		
16406	4016	ENCI	Endereço do caracter a ser interpretado. Pode ser na área de programa ou na linha de edição.
	4017		
16408	4018	EESI	Endereço do erro de sintaxe.
	4019		
16410	401A	INPC	Início da pilha de cálculos.
	401B		
16412	401C	FIPC	Fim da pilha de cálculos.
	401D		
16414	401E	REBC	Registro B do calculador.
16415	401F	EBPF	Endereço para base da tabela de n.ºs em ponto flutuante , podendo ser na pilha de cálculos.
	4020		
16417	4021	—	Não usado.
16418	4022	NLPI	Número de linhas na parte inferior da tela.
16419	4023	NLLA	N.º da linha em listagem automática.
	4024		
16421	4025	VUTP	Valor da última tecla pressionada.
	4026		

16423	4027	ESDT	Estado de debounce do teclado.
16424	4028	ADTV	Ajuste para diferentes padrões de TV.
16425	4029	NLBI	O n.º da próxima linha de BASIC a ser interpretada.
	402A		
16427	402B	NULS	O n.º da última linha está salvo em caso de necessidade.
	402C		
16429	402D	BCT2	BIT 0 — Reset indica uma variável-matriz. BIT 1 — Reset indica a existência da variável requisitada. BIT 5 — Set durante o INPUT. BIT 6 — Set quando o INPUT é numérico.
16430	402E	SULB	Comprimento de uma variável STRING ou uma linha de BASIC.
	402F		
16432	4030	TAPA	Aponta para a tabela de parâmetros.
	4031		
16434	4032	SRND	O valor da semente para a função RND.
	4033		
16436	4034	CNTE	Conta o n.º de telas.
	4035		
16438	4036	CDPL	As coordenadas X & Y do PLOT.
	4037		
16440	4038	COBI	Contador para o buffer da impressora.
16441	4039	NLCP	N.ºs de linha e coluna para o PRINT AT.
	403A		
16443	403B	BCT3	BIT 0 — set sempre que uma tecla é pressionada. BIT 6 — o "verdadeiro" indicador FAST/SLOW. BIT 7 — "cópia" do indicador de FAST/SLOW. set em modo SLOW.
16444	403C	BIMP	Buffer da impressora.
	—		
	405C		
16477	405D	ARPF	Área reservada para 6 números em ponto flutuante.
	—		
	407A		
16507	407B	—	Não usado.
	407C		
16509	407D		Aqui começa o Programa em BASIC.

11. APÊNDICES

11.1 Código dos caracteres

Código	Hex	Caracter	Código	Hex	Caracter	Código	Hex	Caracter	Código	Hex	Caracter
0	00	Espaço	64	40	RND	128	80	□	192	CO	""
1	01	▗	65	41	INKEY$	129	81	▘	193	C1	AT
2	02	▖	66	42	PI	130	82	▝	194	C2	TAB
3	03	▄	67	43		131	83	▐	195	C3	Não usado
4	04	▘	68	44		132	84	▚	196	C4	CODE
5	05	▌	69	45		133	85	▌	197	C5	VAL
6	06	▞	70	46		134	86	▞	198	C6	LEN
7	07	▟	71	47		135	87	▟	199	C7	SIN
8	08	▒	72	48		136	88	▒	200	C8	COS
9	09	▐	73	49		137	89	▐	201	C9	TAN
10	0A	▀	74	4A		138	8A	▀	202	CA	ASN
11	0B	"	75	4B		139	8B	"	203	CB	ACS
12	0C	π	76	4C		140	8C	π	204	CC	ATN
13	0D	$	77	4D		141	8D	$	205	CD	LN
14	0E	:	78	4E		142	8E	:	206	CE	EXP
15	0F	?	79	4F		143	8F	?	207	CF	INT
16	10	(	80	50		144	90	(	208	DO	SQR
17	11	)	81	51		145	91	)	209	D1	SGN
18	12	>	82	52		146	92	>	210	D2	ABS
19	13	<	83	53		147	93	<	211	D3	PEEK
20	14	=	84	54		148	94	=	212	D4	USR
21	15	+	85	55		149	95	+	213	D5	STR$
22	16	−	86	56		150	96	-	214	D6	CHR$
23	17	*	87	57		151	97	*	215	D7	NOT
24	18	/	88	58		152	98	/	216	D8	**
25	19	;	89	59	Não usado	153	99	;	217	D9	OR
26	1A	,	90	5A		154	9A	,	218	DA	AND
27	1B	.	91	5B		155	9B	.	219	DB	<=
28	1C	0	92	5C		156	9C	0	220	DC	>=
29	1D	1	93	5D		157	9D	1	221	DD	<>
30	1E	2	94	5E		158	9E	2	222	DE	THEN
31	1F	3	95	5F		159	9F	3	223	DF	TO
32	20	4	96	60		160	AO	4	224	EO	STEP
33	21	5	97	61		161	A1	5	225	E1	LPRINT
34	22	6	98	62		162	A2	6	226	E2	LLIST
35	23	7	99	63		163	A3	7	227	E3	STOP
36	24	8	100	64		164	A4	8	228	E4	SLOW
37	25	9	101	65		165	A5	9	229	E5	FAST
38	26	A	102	66		166	A6	A	230	E6	LOAD
39	27	B	103	67		167	A7	B	231	E7	POKE
40	28	C	104	68		168	A8	C	232	E8	CONT
41	29	D	105	69		169	A9	D	233	E9	DIM
42	2A	E	106	6A		170	AA	E	234	EA	PAUSE
43	2B	F	107	6B		171	AB	F	235	EB	FOR
44	2C	G	108	6C		172	AC	G	236	EC	GOTO
45	2D	H	109	6D		173	AD	H	237	ED	GOSUB
46	2E	I	110	6E		174	AE	I	238	EE	INPUT
47	2F	J	111	6F		175	AF	J	239	EF	RETURN
48	30	K	112	70	Cursor (↑)	176	B0	K	240	FO	LET
49	31	L	113	71	Cursor (↓)	177	B1	L	241	F1	LIST
50	32	M	114	72	Cursor (←)	178	B2	M	242	F2	NEW
51	33	N	115	73	Cursor (→)	179	B3	N	243	F3	NEXT
52	34	O	116	74	GRAPHICS	180	B4	O	244	F4	RAND
53	35	P	117	75	EDIT	181	B5	P	245	F5	PRINT
54	36	Q	118	76	ENTER	182	B6	Q	246	F6	COPY
55	37	R	119	77	DELETE	183	B7	R	247	F7	RUN
56	38	S	120	78	modo K/L	184	B8	S	248	F8	SAVE
57	39	T	121	79	FUNCTION	185	B9	T	249	F9	REM
58	3A	U	122	7A		186	BA	U	250	FA	UNPLOT
59	3B	V	123	7B	Não usado	187	BB	V	251	FB	SCROLL
60	3C	W	124	7C		188	BC	W	252	FC	IF
61	3D	X	125	7D		189	BD	X	253	FD	CLS
62	3E	Y	126	7E	Número	190	BE	Y	254	FE	PLOT
63	3F	Z	127	7F	Cursor	191	BF	Z	255	FF	CLEAR

vídeo inverso

11.2. Assembler Z80

Hex	Dec	Assembler	Hex	Dec	Assembler	Hex	Dec	Assembler	Hex	Dec	Assembler
00	0	NOP	40	64	LD B,B	80	128	ADD B	C0	192	RET NZ
01	1	LD BC, nn	41	65	LD B,C	81	129	ADD C	C1	193	POP BC
02	2	LD (BC), A	42	66	LD B,D	82	130	ADD D	C2	194	JP NZ, nn
03	3	INC BC	43	67	LD B,E	83	131	ADD E	C3	195	JP nn
04	4	INC B	44	68	LD B,H	84	132	ADD H	C4	196	CALL NZ,nn
05	5	DEC B	45	69	LD B,L	85	133	ADD L	C5	197	PUSH BC
06	6	LD B,n	46	70	LD B,(HL)	86	134	ADD (HL)	C6	198	ADD A,n
07	7	RLC A	47	71	LD B,A	87	135	ADD A	C7	199	RST 0
08	8	EX AF,AF'	48	72	LD C,B	88	136	ADC B	C8	200	RET Z
09	9	ADD HL,BC	49	73	LD C,C	89	137	ADC C	C9	201	RET
0A	10	LD A, (BC)	4A	74	LD C,D	8A	138	ADC D	CA	202	JP Z,nn
0B	11	DEC BC	4B	75	LD C,E	8B	139	ADC E	CB	203	Grupo CB
0C	12	INC C	4C	76	LD C,H	8C	140	ADC H	CC	204	CALL Z,nn
0D	13	DEC C	4D	77	LD C,L	8D	141	ADC L	CD	205	CALL nn
0E	14	LD C,n	4E	78	LD C,(HL)	8E	142	ADC (HL)	CE	206	ADC A,n
0F	15	RRC A	4F	79	LD C,A	8F	143	ADC A	CF	207	RST 8
10	16	DJNZ e	50	80	LD D,B	90	144	SUB B	D0	208	RET NC
11	17	LD DE,nn	51	81	LD D,C	91	145	SUB C	D1	209	POP DE
12	18	LD (DE),a	52	82	LD D,D	92	146	SUB D	D2	210	JP NC,nn
13	19	INC DE	53	83	LD D,E	93	147	SUB E	D3	211	OUT n,A
14	20	INC D	54	84	LD D,H	94	148	SUB H	D4	212	CALL NC,nn
15	21	DEC D	55	85	LD D,L	95	149	SUB L	D5	213	PUSH DE
16	22	LD D,n	56	86	LD D,(HL)	96	150	SUB (HL)	D6	214	SUB A,n
17	23	RL A	57	87	LD D,A	97	151	SUB A	D7	215	RST 10
18	24	JR e	58	88	LD E,B	98	152	SBC B	D8	216	RET C
19	25	ADD HL,DE	59	89	LD E,C	99	153	SBC C	D9	217	EXX
1A	26	LD A,(DE)	5A	90	LD E,D	9A	154	SBC D	DA	218	JP C,nn
1B	27	DEC DE	5B	91	LD E,E	9B	155	SBC E	DB	219	IN A,n
1C	28	INC E	5C	92	LD E,H	9C	156	SBC H	DC	220	CALL C,nn
1D	29	DEC E	5D	93	LD E,L	9D	157	SBC L	DD	221	Grupo DD
1E	30	LD E,n	5E	94	LD E,(HL)	9E	158	SBC (HL)	DE	222	SBC A,n
1F	31	RR A	5F	95	LD E,A	9F	159	SBC A	DF	223	RST 18
20	32	JR NZ,e	60	96	LD H,B	A0	160	AND B	E0	224	RET PO
21	33	LD HL,nn	61	97	LD H,C	A1	161	AND C	E1	225	POP HL
22	34	LD (nn), HL	62	98	LD H,D	A2	162	AND D	E2	226	JP PO,nn
23	35	INC HL	63	99	LD H,E	A3	163	AND E	E3	227	EX (SP),HL
24	36	INC H	64	100	LD H,H	A4	164	AND H	E4	228	CALL PO,nn
25	37	DEC H	65	101	LD H,L	A5	165	AND L	E5	229	PUSH HL
26	38	LD H,n	66	102	LD H,(HL)	A6	166	AND (HL)	E6	230	AND A,n
27	39	DAA	67	103	LD H,A	A7	167	AND A	E7	231	RST 20
28	40	JR Z,e	68	104	LD L,B	A8	168	XOR B	E8	232	RET PE
29	41	ADD HL,HL	69	105	LD L,C	A9	169	XOR C	E9	233	JP (HL)
2A	42	LD HL, (nn)	6A	106	LD L,D	AA	170	XOR D	EA	234	JP,PE,nn
2B	43	DEC HL	6B	107	LD L,E	AB	171	XOR E	EB	235	EX DE,HL
2C	44	INC L	6C	108	LD L,H	AC	172	XOR H	EC	236	CALL PE,nn
2D	45	DEC L	6D	109	LD L,L	AD	173	XOR L	ED	237	Grupo ED
2E	46	ID I,n	6E	110	LD L,(HL)	AE	174	XOR (HL)	EE	238	XOR A,n
2F	47	CPL	6F	111	LD L,A	AF	175	XOR A	EF	239	RST 28
30	48	JR NC, e	70	112	LD (HL),B	B0	176	OR B	F0	240	RET P
31	49	LD SP, nn	71	113	LD (HL),C	B1	177	OR C	F1	241	POP AF
32	50	ID (nn), a	72	114	LD (HL),D	B2	178	OR D	F2	242	JP P,nn
33	51	INC SP	73	115	LD (HL),E	B3	179	OR E	F3	243	DI
34	52	INC (HL)	74	116	LD (HL),H	B4	180	OR H	F4	244	CALL P,nn
35	53	DEC (HL)	75	117	LD (HL),L	B5	181	OR L	F5	245	PUSH AF
36	54	LD (HL), n	76	118	HALT	B6	182	OR (HL)	F6	246	OR A,n
37	55	SCF	77	119	LD (HL),A	B7	183	OR A	F7	247	RST 30
38	56	JR C, e	78	120	LD A,B	B8	184	CP B	F8	248	RET M
39	57	ADD HL, SP	79	121	LD A,C	B9	185	CP C	F9	249	LD SP,HL
3A	58	LD A, (nn)	7A	122	LD A,D	BA	186	CP D	FA	250	JP M,nn
3B	59	DEC SP	7B	123	LD A,E	BB	187	CP E	FB	251	EI
3C	60	INC A	7C	124	LD A,H	BC	188	CP H	FC	252	CAL M,nn
3D	61	DEC A	7D	125	LD A,L	BD	189	CP L	FD	253	Grupo FD
3E	62	LD A,n	7E	126	LD A,(HL)	BE	190	CP (HL)	FE	254	CP A,n
3F	63	CCF	7F	127	LD A,A	BF	191	CP A	FF	255	RST 38

Grupo CB

Hex	Dec	Assembler	Hex	Dec	Assembler	Hex	Dec	Assembler	Hex	Dec	Assembler
00	0	RLC B	40	64	BIT 0,B	80	128	RES 0,B	C0	192	SET 0,B
01	1	RLC C	41	65	BIT 0,C	81	129	RES 0,C	C1	193	SET 0,C
02	2	RLC D	42	66	BIT 0,D	82	130	RES 0,D	C2	194	SET 0,D
03	3	RLC E	43	67	BIT 0,E	83	131	RES 0,E	C3	195	SET 0,E
04	4	RLC H	44	68	BIT 0,H	84	132	RES 0,H	C4	196	SET 0,H
05	5	RLC L	45	69	BIT 0,L	85	133	RES 0,L	C5	197	SET 0,L
06	6	RLC (HL)	46	70	BIT 0,(HL)	86	134	RES 0,(HL)	C6	198	SET 0,(HL)
07	7	RLC A	47	71	BIT 0	87	135	RES 0,A	C7	199	SET 0,A
08	8	RRC B	48	72	BIT 1,B	88	136	RES 1,B	C8	200	SET 1,B
09	9	RRC C	49	73	BIT 1,C	89	137	RES 1,C	C9	201	SET 1,C
0A	10	RRC D	4A	74	BIT 1,D	8A	138	RES 1,D	CA	202	SET 1,D
0B	11	RRC E	4B	75	BIT 1,E	8B	139	RES 1,E	CB	203	SET 1,E
0C	12	RRC H	4C	76	BIT 1,H	8C	140	RES 1,H	CC	204	SET 1,H
0D	13	RRC L	4D	77	BIT 1,L	8D	141	RES 1,L	CD	205	SET 1,L
0E	14	RRC (LH)	4E	78	BIT 1,(HL)	8E	142	RES 1,(HL)	CE	206	SET 1,(HL)
0F	15	RRC A	4F	79	BIT 1,A	8F	143	RES 1,A	CF	207	SET 1,A
10	16	RL B	50	80	BIT 2,B	90	144	RES 2,B	D0	208	SET 2,B
11	17	RL C	51	81	BIT 2 C	91	145	RES 2,C	D1	209	SET 2,C
12	18	RL D	52	82	BIT 2,D	92	146	RES 2,D	D2	210	SET 2,D
13	19	RL E	53	83	BIT 2,E	93	147	RES 2,E	D3	211	SET 2,E
14	20	RL H	54	84	BIT 2,H	94	148	RES 2,H	D4	212	SET 2,H
15	21	RL L	55	85	BIT 2,L	95	149	RES 2,L	D5	213	SET 2,L
16	22	RL (HL)	56	86	BIT 2,(HL)	96	150	RES 2,(HL)	D6	214	SET 2,(HL)
17	23	RL A	57	87	BIT 2,A	97	151	RES 2,A	D7	215	SET 2,A
18	24	RR B	58	88	BIT 3,B	98	152	RES 3,B	D8	216	SET 3,B
19	25	RR C	59	89	BIT 3,C	99	153	RES 3,C	D9	217	SET 3,C
1A	26	RR D	5A	90	BIT 3,D	9A	154	RES 3,D	DA	218	SET 3,D
1B	27	RR E	5B	91	BIT 3,E	9B	155	RES 3,E	DB	219	SET 3,E
1C	28	RR H	5C	92	BIT 3,H	9C	156	RES 3,H	DC	220	SET 3,H
1D	29	RR L	5D	93	BIT 3,L	9D	157	RES 3,L	DD	221	SET 3,L
1E	30	RR (HL)	5E	94	BIT 3,(HL)	9E	158	RES 3,(HL)	DE	222	SET 3,(HL)
1F	31	RR A	5F	95	BIT 3,A	9F	159	RES 3,A	DF	223	SET 3,A
20	32	SLA B	60	96	BIT 4,B	A0	160	RES 4,B	E0	224	SET 4,B
21	33	SLA C	61	97	BIT 4,C	A1	161	RES 4,C	E1	225	SET 4,C
22	34	SLA D	62	98	BIT 4,D	A2	162	RES 4,D	E2	226	SET 4,D
23	35	SLA E	63	99	BIT 4,E	A3	163	RES 4,E	E3	227	SET 4,E
24	36	SLA H	64	100	BIT 4,H	A4	164	RES 4,H	E4	228	SET 4,H
25	37	SLA L	65	101	BIT 4,L	A5	165	RES 4,L	E5	229	SET 4,L
26	38	SLA (LH)	66	102	BIT 4,(HL)	A6	166	RES 4,(HL)	E6	230	SET 4,(HL)
27	39	SLA A	67	103	BIT 4,A	A7	167	RES 4,A	E7	231	SET 4,A
28	40	SRA B	68	104	BIT 5,B	A8	168	RES 5,B	E8	232	SET 5,B
29	41	SRA C	69	105	BIT 5,C	A9	169	RES 5,C	E9	233	SET 5,C
2A	42	SRA D	6A	106	BIT 5,D	AA	170	RES 5,D	EA	234	SET 5,D
2B	43	SRA E	6B	107	BIT 5,E	AB	171	RES 5,E	EB	235	SET 5,E
2C	44	SRA H	6C	108	BIT 5,H	AC	172	RES 5,H	EC	236	SET 5,H
2D	45	SRA L	6D	109	BIT 5,L	AD	173	RES 5,L	ED	237	SET 5,L
2E	46	SRA (HL)	6E	110	BIT 5,(HL)	AE	174	RES 5,(HL)	EE	237	SET 5,(HL)
2F	47	SRA A	6F	111	BIT 5,A	AF	175	RES 5,A	EF	239	SET 5,A
30	48		70	112	BIT 6,B	B0	176	RES 6,B	F0	240	SET 6,B
31	49		71	113	BIT 6,C	B1	177	RES 6,C	F1	241	SET 6,C
32	50		72	114	BIT 6,D	B2	178	RES 6,D	F2	242	SET 6,D
33	51		73	115	BIT 6,E	B3	179	RES 6,E	F3	243	SET 6,E
34	52		74	116	BIT 6,H	B4	180	RES 6,H	F4	244	SET 6,H
35	53		75	117	BIT 6,L	B5	181	RES 6,L	F5	245	SET 6,L
36	54		76	118	BIT 6,(HL)	B6	182	RES 6,(HL)	F6	246	SET 6,(HL)
37	55		77	119	BIT 6,A	B7	183	RES 6,A	F7	247	SET 6,A
38	56	SRL B	78	120	BIT 7,B	B8	184	RES 7,B	F8	248	SET 7,B
39	57	SRL C	79	121	BIT 7,C	B9	185	RES 7,C	F9	249	SET 7,C
3A	58	SRL D	7A	122	BIT 7,D	BA	186	RES 7,D	FA	250	SET 7,D
3B	59	SRL E	7B	123	BIT 7,E	BB	187	RES 7,E	FB	251	SET 7,E
3C	60	SRL H	7C	124	BIT 7,H	BC	188	RES 7,H	FC	252	SET 7,H
3D	61	SRL L	7D	125	BIT 7,L	BD	189	RES 7,L	FD	253	SET 7,L
3E	62	SRL (LH)	7E	126	BIT 7,(HL)	BE	190	RES 7,(HL)	FE	254	SET 7,(HL)
3F	63	SRL A	7F	127	BIT 7,A	BF	191	RES 7,A	FF	255	SET 7,A

Grupo DD

Hex	Dec	Assembler
09	9	ADD IX,BC
19	25	ADD IX,DE
21	33	LD IX,nn
22	34	LD (nn),IX
23	35	INC IX
29	41	ADD IX,IX
2A	42	LD IX,(nn)
2B	43	DEC IX
34	52	INC (IX+d)
35	53	DEC (IX+d)
36	54	LD (IX+d),n
39	57	ADD IX,SP
46	70	LD B,(IX+d)
4E	78	LD C,(IX+d)
56	86	LD D,(IX+d)
5E	94	LD E,(IX+d)
66	102	LD H,(IX+d)
6E	110	LD L,(IX+d)
70	112	LD (IX+d),B
71	113	LD (IX+d),C
72	114	LD (IX+d),D
73	115	LD (IX+d),E
74	116	LD (IX+d),H
75	117	LD (IX+d),L
77	119	LD (IX+d),A
7E	126	LD A,(IX+d)
86	134	ADD (IX+d)
8E	142	ADC (IX+d)
96	150	SUB (IX+d)
9E	158	SBC (IX+d)
A6	166	AND (IX+d)
AE	174	XOR (IX+d)
B6	182	OR (IX+d)
BE	190	CP (IX+d)
E1	225	POP IX
E3	227	EX (SP) IX
E5	229	PUSH IX
E9	233	JP (IX)
F9	249	LD SP,IX

Grupo DD CB

Hex	Dec	Assembler
06	6	RLC (IX+d)
0E	14	RRC (IX+d)
16	10	RL (IX+d)
1E	30	RR (IX+d)
26	38	SLA (IX+d)
2E	46	SRA (IX+d)
3E	62	SRL (IX+d)
46	70	BIT 0(IX+d)
4E	78	BIT 1,(IX+d)
56	86	BIT 2,(IX+d)
5E	94	BIT 3,(IX+d)
66	107	BIT 4,(IX+d)
6E	110	BIT 5,(IX+d)
76	118	BIT 6,(IX+d)
7E	126	BIT 7,(IX+d)
86	134	RES 0,(IX+d)
8E	142	RES 1,(IX+d)
96	150	RES 2,(IX+d)
9E	158	RES 3,(IX+d)
A6	166	RES 4,(IX+d)
AE	174	RES 5,(IX+d)
B6	182	RES 6,(IX+d)
BE	190	RES 7,(IX+d)
C6	198	SET 0,(IX+d)
CE	206	SET 1,(IX+d)
D6	214	SET 2,(IX+d)
DE	222	SET 3,(IX+d)
E6	230	SET 4,(IX+d)
EE	238	SET 5,(IX+d)
F6	240	SET 6,(IX+d)
FE	254	SET 7,(IX+d)

Grupo FD

Hex	Dec	Assembler
09	9	ADD IY,BC
19	31	ADD IY,DE
21	33	LD IY,nn
22	34	LD (nn),IY
23	35	INC IY
29	41	ADD IY,IY
2A	42	LD IY,(nn)
28	43	DEC IY
34	52	INC (IY+d)
35	53	DEC (IY+d)
36	54	LD (IY+d),n
39	57	ADD IY,SP
46	70	LD,B,(IY+d)
4E	78	LD C,(IY+d)
56	86	LD D,(IY +d)
5E	94	LD E,(IY+d)
66	102	LD E,(IY+d)
6E	110	LD L,(IY+d)
70	112	LD (IY+d),B
71	113	LD (IY+d),D
72	114	LD (IY+d),C
73	115	LD (IY+d),E
74	116	LD (IY+d),H
75	117	LD (IY+d),L
77	119	LD (IY+d),A
7E	126	LD A,(IY+d)
86	134	ADD (IY+d)
8E	142	ADC (IY+d)
96	150	SUB (IY+d)
9E	158	SBC (IY+d)
A6	166	AND (IY+d)
AE	174	XOR (IY+d)
B6	182	OR (IY+d)
BE	190	CP (IY+d)
E1	225	POP IY
E3	227	EX (SP), IY
E5	229	PUSH IY
E9	233	JP (IY)
F9	249	LD SP, IY

Grupo FD CB

Hex	Dec	Assembler
06	6	RLC (IY+d)
0E	14	RRC (IY+d)
16	10	RL (IY+d)
1E	30	RR (IY+d)
26	38	SLA (IY+d)
2E	46	SRA (IY+d)
3E	62	SRL (IY+d)
46	70	BIT 0(IY+d)
4E	78	BIT 1,(IY+d)
56	86	BIT 2,(IY+d)
5E	94	BIT 3,(IY+d)
66	107	BIT 4,(IY+d)
6E	110	BIT 5,(IY+d)
76	118	BIT 6,(IY+d)
7E	126	BIT 7,(IY+d)
86	134	RES 0,(IY+d)
8E	142	RES 1,(IY+d)
96	150	RES 2,(IY+d)
9E	158	RES 3,(IY+d)
A6	166	RES 4,(IY+d)
AE	174	RES 5,(IY+d)
B6	182	RES 6,(IY+d)
BE	190	RES 7,(IY+d)
C6	198	SET 0,(IY+d)
CE	206	SET 1,(IY+d)
D6	214	SET 2,(IY+d)
DE	222	SET 3,(IY+d)
E6	230	SET 4,(IY+d)
EE	238	SET 5,(IY+d)
F6	240	SET 6,(IY+d)
FE	254	SET 7,(IY+d)

Grupo ED

Hex	Dec	Assembler
40	64	IN B,(C)
41	65	OUT (C),B
42	66	SBC HL,BC
43	67	LD (nn)BC
44	68	NEG
45	69	RET N
46	70	IM 0
47	71	LD I,A
48	72	IN C,(C)
49	73	OUT (C),C
4A	74	ADC HL,BC
4B	75	LD BC, (nn)
4D	77	RETI
4F	79	LD R,A
50	80	IN D,(C)
51	81	OUT (C),D
52	82	SBC HL,DE
53	83	LD (nn),DE
56	86	IM 1
57	87	LD A,I
58	88	IN E,(C)
59	89	OUT (C),E)
5A	90	ADC HL,DE
5B	91	LD DE,(nn)
5E	94	IM 2
5F	95	LD A,R
60	96	IN H,(C)
61	97	OUT (C),H
62	98	SBC HL,HL
63	99	LD (nn),HL
67	103	RRD
68	104	IN L,(C)
69	105	OUT (C),L
6A	106	ADC HL,HL
6B	107	LD HL,(nn)
6F	111	RLD
72	114	SBC HL,SP
73	115	LD (nn),SP
78	120	IN A,(C)
79	121	OUT (C),A
7A	122	ADC HL,SP
7B	123	LD SP,(nn)
A0	160	LD I
A1	161	CP I
A2	162	IN I
A3	163	OUT I
A8	168	LD D
A9	169	CP D
AA	170	IN D
AB	171	OUT D
B0	176	LD IR
B1	177	CP IR
B2	178	IN IR
B3	179	OT IR
B8	184	LD DR
B9	185	CP DR
BA	186	IN DR
BB	187	OT DR

11.3 RESUMO

11.3.1 RESUMO DAS FUNÇÕES

FUNÇÕES

ABS	Valor absoluto
ACS	Arco-cosseno ($\cos^{-1}$) — Calculado entre 0 e π — Erro A se: $-1 > X > 1$
AND	X AND Y = X se Y ≠ 0; 0 se Y = 0 X$ AND Y = X$ se Y ≠ 0; "" se Y = 0
ASN	Arco-seno (seno $^{-1}$) — Calculado entre $-\pi/2$ e $\pi/2$ — Erro A se: $-1 > x > 1$
ATN	Arco-tangente ($\tan^{-1}$) — Calculada entre $-\pi/2$ e $\pi/2$.
CHR$	Converte um caracter em seu código correspondente.
COS	Cosseno (radiano)
EXP	e^X
INKEY$	Se uma tecla for pressionada durante a execução do INKEY$, o resultado será o caracter correspondente à tecla, caso contrário, será uma STRING vazia.
INT	Considera somente a parte inteira do número (arredonda para baixo).
LEN	Comprimento de uma STRING.
LN	LnX — Logarítmo natural (Na base e) — Erro A se: $X < 0$
NOT	NOT X = 0 se X ≠ 0; 1 se X = 0
OR	X OR Y = 1 se Y ≠ 0; X se Y = 0
PEEK	Dá o valor do byte na posição de memória X
π	PI (3.141592654)
RND	Produz n.°s aleatórios entre 0 e 1
SGN	Função sinal: sinal de X (-1, 0, +1)
SIN	Seno (radiano)
SQR	Raiz quadrada — Erro A se: $X < 0$.
STR$	Converte um número em uma STRING.
TAN	Tangente (radiano)
USR	Executa a sub-rotina em linguagem de máquina cujo endereço é X. O resultado deverá retornar no par de registradores BC.
VAL	Calcula uma STRING como uma expressão numérica.

11.3.2. OPERAÇÕES MATEMÁTICAS:

+	Adição
−	Subtração
*	Multiplicação
/	Divisão
**	Exponenciação
E	Notação científica
=	Igual
>	Maior
<	Menor
< =	Menor igual
> =	Maior igual
<>	Diferente

11.3.3. PRIORIDADES DE EXECUÇÃO:

Operação	Prioridade
Subscrito e SUB-STRING	12
Todas as funções com exceção do NOT e negação	11
**	10
Negação	9
*, /	8
+, − (− binário)	6
=, >, <, < =, > =, <>	5
NOT	4
AND	3
OR	2

11.3.4. RESUMO DOS COMANDOS

COMANDOS

CLEAR	Limpa as variáveis da memória.
CLS	Limpa a tela.
CONT	Recomeça o programa quando interrompido.
COPY	Faz uma cópia da tela na impressora.
DIM	Reserva espaço na memória para uma matriz de n dimensões.
FAST	Só mostra a tela no final da execução de um programa.
FOR A TO B STEP C	Define uma variável de controle com valor A, limite B e passo C.
GOSUB	Salva o endereço de retorno na pilha e executa a sub-rotina no endereço determinado.
GOTO	Vai para a linha indicada.
IF x THEN	Possibilita um desvio condicional em tomada de decisões.
INPUT	Pára e espera uma introdução de dados.
LET	Possibilita a atribuição ou alteração de valor de alguma variável numérica ou alfanumérica.
LIST	Lista o programa na tela a partir da linha indicada. Se não houver indicação, iniciará com a 1.º linha do programa.
LLIST	Como LIST, só que lista o programa na impressora.
LOAD	Carrega um programa da fita para o computador.
LPRINT	Como o PRINT, só que na impressora.
NEW	Limpa a memória RAM do Micro até o endereço anterior ao indicado na variável FIME.
NEXT	Controle do "Loop" FOR-NEXT.
PAUSE	Pára a execução de um programa durante o tempo pré-determinado.
PLOT X,Y,	Imprime um elemento de imagem nas coordenadas X,Y.
POKE A,B	Armazena o valor B na posição de memória A.
PRINT	Imprime informações na tela que podem ser: vazia, ou seja, nada; expressão numérica ou uma STRING. O PRINT utilizado com o AT imprime em qualquer ponto da tela, a partir da linha e coluna indicadas. Utilizado com o TAB imprime em forma de tabulação, a partir de uma coluna pré-estabelecida.
RAND	É usado para gerar o próximo valor do RND.
REM	Serve apenas para fazer comentários, não interferindo nos programas.
RETURN	Retorna da sub-rotina para o programa principal, isto é, retira o número da linha da pilha de GOSUB e salta para a linha seguinte.
RUN	Limpa as variáveis e executa o programa. Para que seja executado a partir de uma determinada linha que não a primeira, indica-se o n.º da linha desejada.
SAVE	Transfere o programa do computador para a fita.
SCROLL	Movimenta o arquivo de imagem para cima, fazendo com que a primeira linha desapareça para dar lugar à última.
SLOW	Executa o programa mostrando continuamente a tela.
STOP	Pára a execução do programa.
UNPLOT	Apaga os pontos que foram impressos pelo PLOT usando os mesmos valores das coordenadas.

11.4. CÓDIGO DE REPORTAGEM

Esta tabela apresenta os códigos que aparecem no canto inferior esquerdo da tela quando ocorre alguma parada no programa. Estes códigos sempre veem seguidos do número da linha em que esta parada aconteceu. Procure sempre saber das causas dos códigos nos programas, para um entendimento mais rápido da linguagem BASIC.

Código	***Significado***	***Situações***
0	Execução do programa foi completa, sem a ocorrência de erro.	qualquer
1	A variável estabelecida pelo comando FOR não existe, mas existe acompanhada da instrução NEXT.	NEXT
2	Variável indefinida. Em variáveis simples, ocorre se elas forem usadas antes de serem estabelecidas por um comando LET. Em variáveis subscritas ocorre se forem usadas antes de dimensionadas pelo comando DIM. Para a variável de controle, ocorre se for usada antes de ser definida por um comando FOR e não existir outra variável simples com o mesmo nome.	qualquer
3	Subscrito fora da faixa.	variáveis subscritas
4	Espaço insuficiente na memória. Devido a esta falta de memória, o código de reportagem pode também aparecer incompleto. Ex: 3/50 aparece 3/5.	PRINT, LET, LIST, DIM, PLOT, UNPLOT, FOR, GOSUB e, às vezes, durante avaliação de funções
5	Não há mais espaço na tela.	PRINT e LIST.
6	No caso de cálculos aritméticos, ocorreu uma sobrecarga na resposta, isto é, um número superior a 10^{38} ou inferior a 10^{-38}	Qualquer cálculo aritmético
7	RETURN sem GOSUB correspondente.	RETURN
8	INPUT não permitido.	INPUT
9	Comando STOP executado.	STOP
A	Ocorrência de erro por estar fora dos limites da função	SQR, ASN, ACS, LN.
B	Número inteiro fora da faixa permitida. (0 > x > 65535).	RAND, PLOT, UNPLOT, RUN, DIM, GOTO, PEEK, USR, LLIST, CHR$, LIST PAUSE, POKE e acesso a matriz
C	Dentro de uma STRING da VAL não contém uma expressão numérica.	VAL
D	1. Programa interrompido por BREAK	no fim de qualquer comando ou em LOAD, SAVE LPRINT, LLIST ou COPY
	2. A linha INPUT é iniciada com STOP	INPUT
E	Não utilizado	
F	O nome do programa é uma STRING vazia	SAVE

PROGRAMA BASIC		PÁG
NOME DO PROGRAMADOR	NOME DO PROGRAMA	DATA

Nº DA LINHA	COMANDOS

PROGRAMA EM LINGUAGEM DE MÁQUINA											PÁG
NOME DO PROGRAMADOR				NOME DO ☐PROGRAMA ☐SUB-ROTINA ☐FUNÇÃO							DATA
LBL	Nº	ENDEREÇO	CÓD. DE OPER.				ASSEMBLER		COMENTÁRIOS		

Dinheiro... pra que dinheiro?

Um certo programa foi pedir empréstimo em um dos vários bancos de memória do nosso querido R-470. Ao entrar no BANCO SAVE, famoso por salvar programas perdidos pelos integrados da vida, avistou aquela moça que fica na entrada de todos os bancos e deu um EDIT de sua linha que estava com problema. Ela, processando os dados obtidos, conclui com quem ele deveria conversar. Em seguida, deu um GOTO sr. $, o gerente.

— Bom digito, como vai?

— Vai-se indo. Naquele FOR NEXT de sempre. Mas, em que posso serví-lo?

— Bem. Vou dar logo a DIM de meu problema...

O programa deu um PAUSE 100, como quem está encontrando dificuldade para falar, e prosseguiu.

— Estou precisando urgentemente de uma expansão de memória. A que eu tenho já não é suficiente. Estou ficando sufocado.

De repente, toca o telefone e o sr. $ dá um BREAK na conversa. Era a sua esposa. Estava lhe mandando um LIST do que queria que ele levasse para casa depois do serviço.

Desligou o telefone e deu um CONT...

— Voltando ao assunto. Quanto é que o senhor está precisando?

— Aproximadamente, 16 bytes.

— Entendo...

E sua renda? É fixa?

— Não, é variável. Mas não se preocupe. Às vezes, tenho tanto serviço que até preciso de um SCROLL.

— Então, não tem problema nenhum.

Agora, o senhor vai responder um pequeno questionário. Não demora nada. É FAST.

— Tudo bem.

— Bem. Este questionário é divido em 3 partes. O senhor só prossegue se atender às exigências de cada fase. Entendeu?

— Entendi.

— Então, vamos à primeira. O senhor só passa para a 2.ª fase se tiver um veículo, um imóvel e uma renda mensal superior à 2 bytes. Traduzindo, é o seguinte: Passa para a 2.ª fase IF tem veículo **AND** imóvel **AND** renda > 2 bytes.

— O senhor tem tudo isso?

— Sim, tenho.

— Vamos, então, à 2.ª fase.

Para passar à terceira fase o senhor precisa ter um vídeo-inverso ou caracteres coloridos. Traduzindo... passa para a 3.ª fase IF tem vídeo-inverso **OR** caracteres coloridos.

— Eu só tenho vídeo-inverso.

— Ótimo. Chegamos, com isso, à 3.ª e última fase.

É preciso que o senhor não tenha nenhuma ENTER no DOPS (Departamento de Opressão à Programas Suspeitos).

— Não, nenhuma.

— Eu logo vi que o senhor era uma pessoa de caracter.

— Obrigado. Mas... fui aprovado?
— Claro. O empréstimo é seu. Só preciso de mais algumas coordenadas de praxe.
Seu nome por favor.
— REM Digit Cavalcanti.
— Profissão...
— Autônomo. Dou PRINT à GRAPHICS.
— Endereço...
— PLOT 10,20. É NEXT daqui.
— Agora, INPUT sua assinatura aqui e espere um comando para voltar e acoplar seu empréstimo.
— Muito obrigado.
E, muito feliz, o programa REM Digit Cavalcanti RUN ENTER para casa.

STOP

Autor da obra: Não tem registro

ANOTAÇÕES

Ritas do Brasil I.B.M. Ltda.
Rua Soldado José Reymão, 199 — Pque. Novo Mundo
São Paulo — SP — CEP 02178 — Telex: (011) 34673 RITA BR