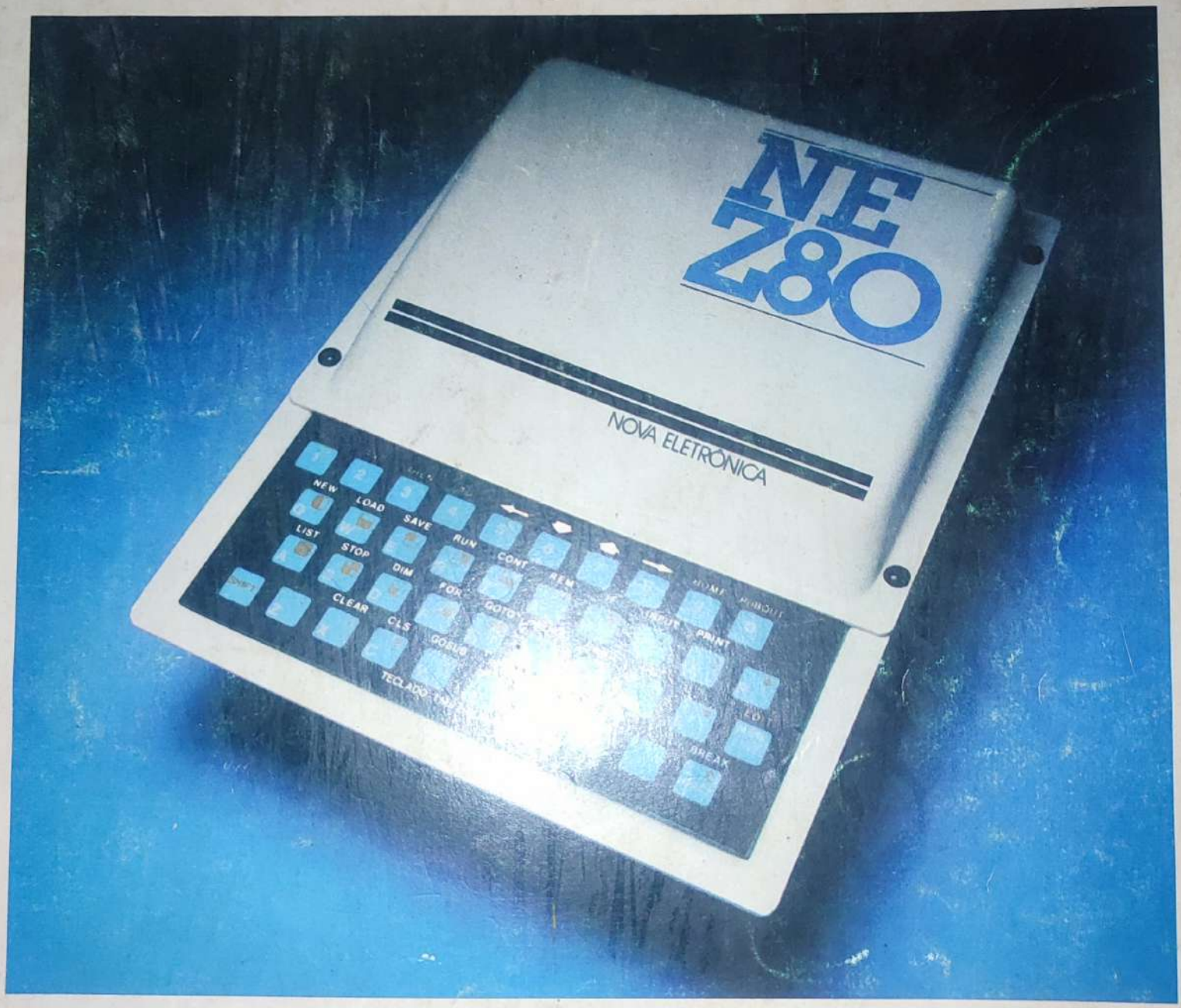
MANUAL DE INSTRUÇÕES
OPERAÇÃO E PROGRAMAÇÃO



NE Z80

Sumário

Manual do NE-Z80

Neste manual do computador pessoal NE-Z80, você encontrará tudo o que precisa saber sobre o **Hardware** e o **Software** do aparelho e tomará conhecimento da linguagem utilizada pelo mesmo (**Basic**), a qual lhe permitirá escrever seus próprios programas.

Adotamos o método de, na maior parte do livro, agrupar os capítulos que compõem o manual em pares. Desse modo, o primeiro capítulo de cada par é puramente descritivo e de preparação para o segundo, cujo objetivo é familiarizá-lo com o **Basic** do NE-Z80 ao mesmo tempo que vamos demonstrando o uso do teclado do microcomputador.

Para quem já tiver experiência no uso da linguagem **Basic** não haverá necessidade de ler todo o manual — provavelmente apenas o capítulo 2 e o sumário de linguagem **Basic** para o NE-Z80, serão suficientes.

Os menos experimentados deverão ter paciência e ler todo o livro, capítulo por capítulo.

Capítulo 1

Recomendações importantes

Esse equipamento gera e se utiliza de sinais da faixa de rádio-freqüência e, se não for instalado e usado corretamente, isto é, de acordo com nossas instruções, poderá causar interferências nos receptores de rádio e televisão que estiverem no seu raio de ação.

O NE-Z80 possue caixa totalmente blindada, que o protege contra sinais que eventualmente possam influir no seu funcionamento, o que assegura baixa probabilidade de que ocorra algum fenômeno dessa ordem. No entanto, não há possibilidade de garantir que não acontecerão interferências numa determinada instalação. Caso se observe alguma interferência no seu receptor de rádio ou TV, proceda segundo algumas recomendações que passamos a enumerar:

— reoriente a antena do receptor;
— mude a posição do computador em relação ao receptor;
— afaste o computador do receptor;
— ligue o computador e o receptor em ramos diferentes do circuito.

Capítulo 2

Preparando o micro

Este capítulo é o mais importante do manual, porque nele você aprenderá como ligar o NE-Z80 e conectar a ele as unidades auxiliares.

Duas unidades constituem o sistema essencial do NE-Z80:

— o microcomputador;
— a unidade de alimentação.

A fonte de alimentação é de 9 volts CC, 600 mA, não-regulada.

NÃO LIGUE O COMPUTADOR AINDA.

Olhe atrás de seu NE-Z80. Você verá quatro plugues: três deles são do tipo jaque fêmea, marcados com os dizeres MIC, EAR e 9V DC; o quarto é do tipo RCA e será usado para o cabo de TV.

O diagrama de conexão (figura 1) mostra como conectar a fonte de alimentação e o televisor que será usado como **display** para a saída do micro. Para ligação à TV use um cabo coaxial e, opcionalmente, uma chave de comutação para evitar a necessidade de ficar conectando e desconectando a antena do aparelho a cada vez que for usá-lo com o NE-Z80.

A figura mostra também como deverá se fazer a ligação com o gravador cassete, unidade auxiliar de memória para o microcomputador.

Faça, então, as ligações da unidade de alimentação e da televisão ao micro, de modo apropriado. Se for o caso, selecione a chave de comutação para a posição adequada, caso você tenha a agrupado ao sistema.

Agora, ainda sem ligar o televisor, posicione-o no canal 2. Ligue a TV, deixando-a totalmente sem volume.

Ligue o NE-Z80.

Não se desaponte se o que aparecer na tela for uma horrível confusão acinzentada! Tente sintonizar a TV através do "ajuste fino de sintonia". Você verá que, subitamente, a tela ficará clara e, no canto inferior esquerdo, aparecerá um quadrado preto com uma letra K branca dentro.

Se não puder ver a letra K, mexa no controle de brilho até que ela fique visível. (Obs.: A sintonia poderá ser feita no canal 3, também).

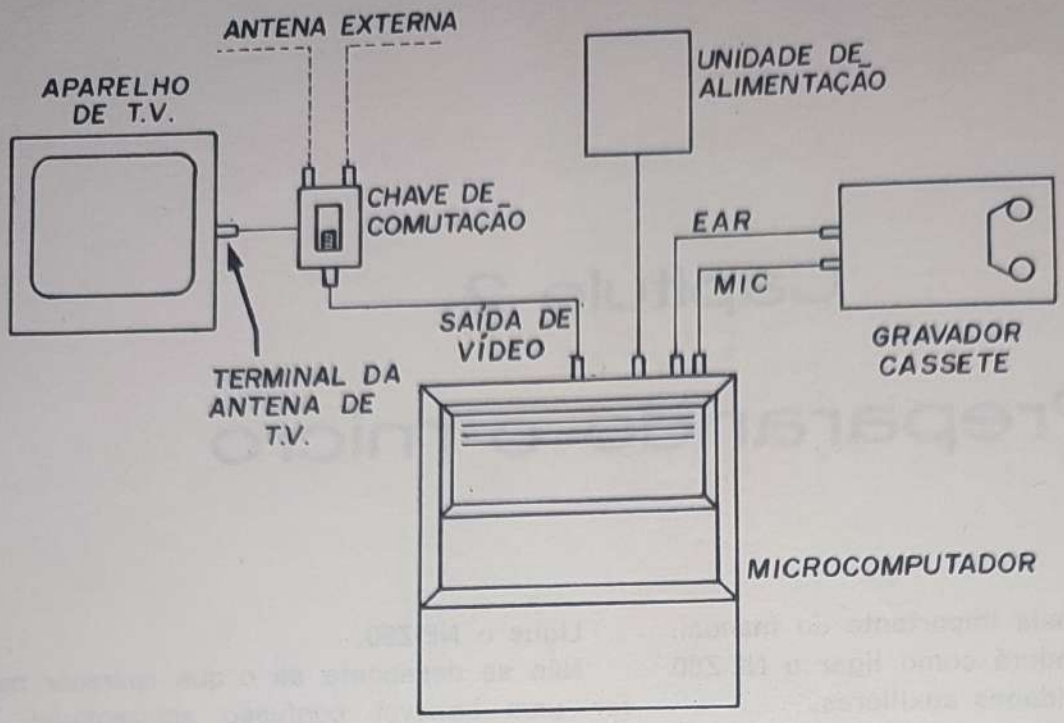
ARMAZENANDO PROGRAMAS NA FITA CASSETE

Antes que possa armazenar programas numa fita, você terá de escrevê-los. O restante desse manual, a partir do próximo capítulo, é todo sobre como escrever um programa. Então, a primeira coisa a fazer é ir em frente, lendo, até ter um programa para armazenar.

Mas, supondo que você já está apto e já tem um programa escrito...

Um cabo duplo deve ser utilizado para conectar o gravador ao NE-Z80. A maioria dos gravadores cassete têm jaques de 3,5 mm para MICROPHONE e EARPIECE. Dispondo de um desses, conecte os plugues como mostramos na figura 1 (Em alguns gravadores são usados plugues DIN. O nível de sinal necessário deve ser de, no mínimo, 4 volts pico a pico, o que geralmente não se consegue com plugue DIN).

Tendo já efetuado a conexão, ajuste o controle de tonalidade do gravador para o máximo. Caso seu gravador tenha controles separados de gra-



ves (BASS) e agudos (TREBLE), regule os agudos para o máximo e os graves no mínimo. Ajuste, também, o controle de volume no máximo.

Você já tem um programa?

Desconecte o plugue do gravador e use o microfone do mesmo para gravar o título do programa com a sua voz. Isso o ajudará quando quiser localizar determinado programa na fita.

Reconecte o plugue ao gravador. Aperte o comando SAVE no micro (tecla E), comece a gravação e depois aperte a tecla NEWLINE.

A tela de seu televisor ficará cinza por uns 5 segundos e então você verá uma série de listras horizontais percorrendo-a. Passados alguns segundos, a tela ficará clara e você poderá observar a listagem do programa.

Naturalmente você desejará certificar-se de que o programa foi gravado na fita. Para isso, desconecte o plugue EAR. Volte a fita até o início, no ponto onde gravou o título com sua voz. Aperte a tecla PLAY do gravador. Você ouvirá o título e então um zumbido seguido de uns 5 segundos de silêncio. A partir daí, escutará um som bem peculiar. Este ruído é o programa sendo transmitido pelo alto-falante do gravador. Depois de um certo tempo o som cessará, voltando a ser produzido um zumbido.

Volte novamente a fita até o ponto onde se iniciam os 5 segundos de silêncio. Reconecte o plugue EAR. Pressione primeiramente a tecla W (comando LOAD) do micro e, depois, a chave PLAY do seu gravador; em seguida, pressione a tecla NEWLINE do computador. A tela se tornará cinza ou preta e, após alguns segundos, ficará cinza, com algum chuvisco. Mais alguns segundos e ela irá clarear e listar o programa.

NOTA: Ao armazenar um programa você também armazena todos os dados e variáveis envolvidas. É possível impedir que isso ocorra utilizando-se o comando GO TO 1.

Se o que descrevemos não acontecer, use o comando BREAK (SPACE) para interromper a carga (LOAD) do programa. Se assim mesmo não for possível, desligue o micro por alguns segundos. Tente repetir todo o procedimento, porém, mudando a posição do controle de volume. Outras possibilidades são verificar a fiação de ligação e, se o gravador estiver ligado à rede, tentar repetir a operação com alimentação por pilhas.

Mas, realmente, alguns tipos de gravadores não se adaptarão às operações descritas do modo como indicamos. Você poderá verificar se o seu gravador é um desses modelos, observando se, no intervalo de 5 segundos de "silêncio", aparece alguma oscilação. No caso afirmativo, passe a proceder da seguinte forma: quando for armazenar um programa (SAVE), deixe ligado apenas o plugue MIC ao gravador; e quando for carregar o programa no computador (LOAD), somente o plugue EAR deverá ser mantido.

Capítulo 3

BASIC, mas não totalmente

Quase todos os computadores (inclusive o NE-Z80) trabalham em código binário. Nós conversamos utilizando-nos da língua portuguesa. Portanto, para nos comunicarmos com o computador temos de aprender a usar o código binário ou então ensinar uma língua a ele. Há uns 40 anos era obrigatório, para quem quisesse trabalhar com um computador digital, aprender binário. Mas porque ter todo esse trabalho se dispomos do próprio computador para fazê-lo por nós?

Portanto, podemos ensinar uma linguagem mais adequada à máquina. Os computadores, entretanto, não são tão brilhantes a ponto de conseguir conversar em português, ou qualquer outra língua humana, diretamente, algo que uma criança de dois anos aprende com facilidade. Isso principalmente porque temos muitos meios e muitas palavras para dizer as coisas. A solução está em códigos intermediários, mistos, mas bem determinados. Várias linguagens foram criadas, algumas mais próximas da língua binária da máquina. Essas são chamadas de linguagens de **baixo nível.** Outras, bem mais chegadas ao falar humano (à língua inglesa), são as consideradas de alto nível, por exemplo: a ALGOL, a PL/1, a PASCAL, a FORTRAN e a BASIC.

O NE-Z80 utiliza BASIC, porque é uma das mais fáceis de aprender e mais adequada para a grande maioria dos problemas que você solucionará. Como todas as línguas, a Basic também tem uma variedade de "dialetos", dependendo do computador que está sendo usado. A Basic do NE-Z80 difere de outras similares em alguns aspectos que estão listados no "Sumário da Basic do NE-Z80", no fim deste manual.

Como já dissemos, o computador não pensa. Nós temos de lhe dizer, passo a passo, as instruções que ele deve executar. Isso é o que chamamos de programa. Naturalmente, para se escrever um programa, é preciso conhecer todo um vocabulário especial, o que faz parte do nosso próximo assunto.

INSTRUÇÕES E COMANDOS

A BASIC do NE-Z80 oferece a você 22 instruções (**statements**). Elas estão divididas em algumas categorias:

1) Comandos do sistema

NEW — "Limpa" o NE-Z80 para um novo programa
RUN — Faz correr o programa
LIST — Dá toda a listagem do programa
LOAD — Carrega o computador com o programa contido na fita
SAVE — Armazena o programa na fita magnética

2) Instruções de controle

GOTO N — Vai até a linha indicada (N)
IF... THEN — Compara ou checa uma condição (IF) e então executa (THEN)
GOSUB N — Pula para uma subrotina que se inicia na linha N
STOP — Comando de parada final do programa
RETURN — Usado no término da subrotina para retorno ao programa principal
FOR... TO — Usado para repetição de uma seqüência (loop)
NEXT — Volta para o começo da seqüência e incrementa em 1 a variável
CONTINUE — Continua a execução do programa depois de uma parada que não STOP

Essas instruções dão ao programador a possibilidade de controlar a ordem na qual as operações são ou devem ser executadas. Elas serão descritas com detalhe nos capítulos vindouros.

3) Instruções de Entrada/Saída

PRINT — O computador fornece os dados de saída

INPUT — Permite a entrada de dados

4) Instrução de transferência

LET...=...

Essa instrução é usada sempre que se deseja efetuar uma operação aritmética.

5) Outras instruções

CLEAR — Apaga os valores das variáveis armazenadas

CLS — Apaga (limpa) a tela

DIM — Delimita um espaço de memória para uma variável.

REM — Indica que a seguir vem uma observação

RAND — Ativa o gerador de números aleatórios

POKE — Permite ao usuário "conversar" com o computador em código binário

Existem algumas instruções normais Basic que não são incluídas na linguagem Basic do NE-Z80:

READ
RESTORE
DATA
END
ON

Quase tudo que pode ser feito com essas instruções pode ser conseguido com o NE-Z80 utilizando outras instruções (a instrução END não é necessária no nosso microcomputador).

Num programa em Basic, todas as instruções ou linhas são precedidas por um **número de instrução** ou **número de linha.** O número de linha pode variar entre 1 e 9999. O computador executa as instruções na ordem em que elas estão numeradas. Isso é fácil de entender, porque os programas são sempre mostrados na tela em ordem crescente com o número das instruções, de modo que a ordem na qual o programa é listado também é a ordem na qual ele deve ser executado.

VARIÁVEIS

Todas as partes de uma informação armazenada no computador para uso num programa são identificadas, para que a máquina possa segui-las. Cada parte da informação, ou variável, tem um nome.

Existem dois tipos de variáveis:

1. Números (variáveis integrais) — podem ter qualquer valor entre — 32768 e 32767 inclusive. Exemplo: 142.

2. **Strings** (frases) — podem ser qualquer seqüência de caracteres (exceto ") de qualquer comprimento.

Exemplo: "JUCA PIRAMA"

As variáveis integrais têm nomes que devem sempre começar por uma letra e conter somente letras e dígitos.

Podem ser de qualquer comprimento, inclusive mnemônicos pode ser usados.

Alguns exemplos:

A, A3, AB, AB3, ANSWER, A4X
Y, Y8, YZ, YZ9, FRIDB
AAA, Z123BC, QTOTAL

Observe que não há espaço entre os dígitos.

Alguns exemplos errôneos:

4, 45, 4AD (o nome não começa por uma letra)
FRED BLOGGS (há espaço)
A.B (um caracter não permitido incluído)
FRED-BLOGGS (isso não é uma variável, são duas, uma subtraída da outra)

Capítulo 4

Conversando com o NE-Z80

Ligue o computador como foi descrito no capítulo 2. Você verá um quadrado, com um K em seu interior (K), aparecer no canto interior esquerdo da tela. Esse é o **cursor** e mostrará sempre em que ponto da tela você está com o programa.

O TECLADO

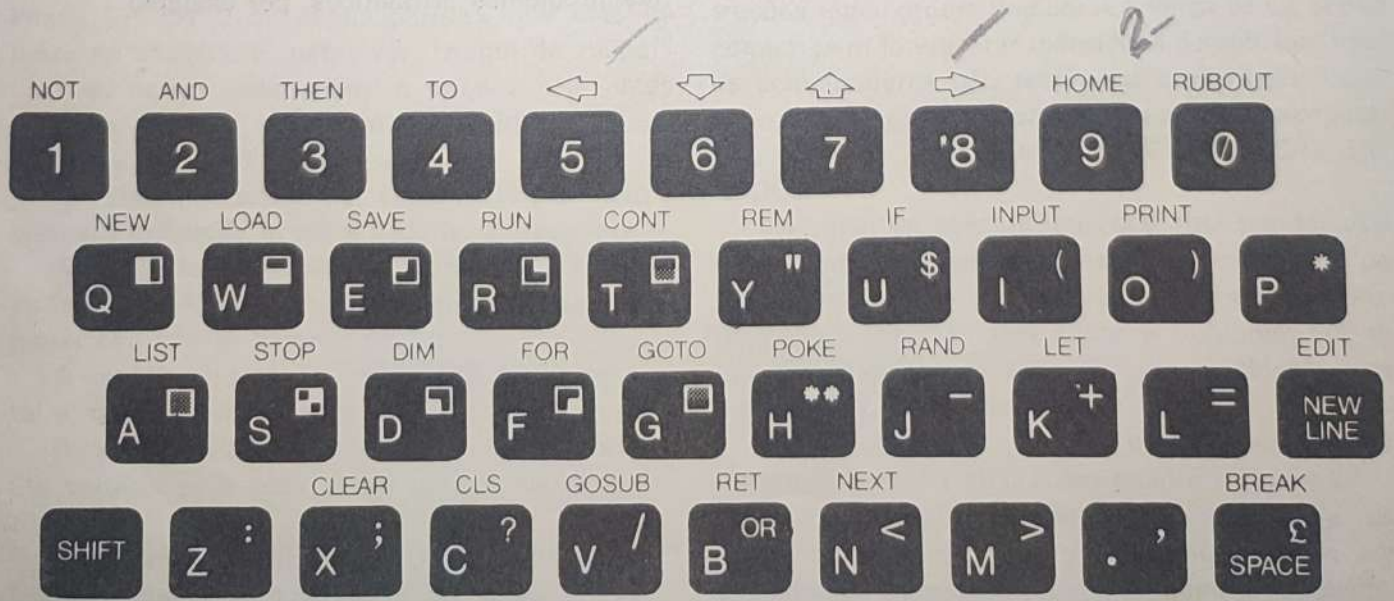
Olhando com atenção para o teclado, você verá que ele se parece com o de uma máquina de escrever, mas apresenta algumas diferenças. Por exemplo, não há a tecla retorno (em seu lugar existe a chave chamada NEWLINE) e a barra de espaço é substituída por uma chave denominada SPACE.

As letras ou palavras (comandos) que se encontram fora (acima) de cada tecla, são obtidas usando-se a chave SHIFT.

Um ponto muito importante: como distinguir a letra O do número 0. Sempre usaremos 0 como letra e 0 como número zero. Na tela, o computador usa um quadrado (▢) para a letra O e um hexágono (⬡) para o número zero. A razão pela qual adotamos o símbolo 0 para o número zero reside no fato de que todos os livros e literatura sobre programação o utilizam.

O [S] é uma indicação de que houve um erro de sintaxe e aparece se houver algum erro na programação, na linha onde está o erro.

O [L] indica que o computador está esperando



alguma coisa de você. Esperando, por exemplo, que você entre com uma variável.

Exemplo:

```
22 PRINT "ENTER A NUMBER" (Entre com um número)
24 INPUT A
4Ø PRINT A
```

Agora tente fazer o programa correr.

Primeiro, o computador irá pedir a variável e depois pedirá o número. Enquanto ele espera que você entre com um número, ele indicará (L)(S) e não só(L) . Entre com o número e finalmente ele irá imprimi-lo.

EDITANDO PROGRAMAS

Suponha que você queira alterar alguma coisa num programa que já escreveu (isto é chamado "editar um programa").

Verifique a tela (o cursor (>) aponta para a linha 4Ø).

Agora veja o que acontece quando você aperta SHIFT 7 (↑).

O cursor sobe, certo?! Agora SHIFT 6 (↓), o cursor desce.

Essas chaves permitem que você selecione qualquer linha de seu programa, que eventualmente queira modificar.

Agora selecione uma linha, apertando 22 por exemplo. Acione SHIFT NEWLINE (EDIT) e a linha 22 aparecerá na parte superior da tela.

SHIFT 8 (→) e verá que o cursor passa por cima da palavra PRINT. SHIFT 5 (←) e ele retornará para o lado esquerdo.

Para apagar qualquer erro use SHIFT Ø (RUBOUT). Cada vez que você apertar RUBOUT, o caracter, palavra ou sinal à esquerda do cursor será eliminado.

O COMANDO LIST

Quando você já tiver experiência com o NE-Z80, irá escrever programas com mais de 24 linhas (que é o número máximo de linhas que pode aparecer na tela). Isso poderia constituir-se num problema. Pois então façamos um teste.

Aperte NEW e depois NEWLINE.

Entre com o seguinte programa:

```
1Ø PRINT
2Ø PRINT
3Ø PRINT
4Ø PRINT
```

E assim por diante até

```
26Ø PRINT
```

De olho na tela, você notará que as primeiras linhas saíram do campo de visão.

Agora aperte LIST (tecla A), seguida de NEWLINE. O programa será totalmente apresentado a partir da linha 1Ø.

Então, aperte LIST 2ØØ. O programa será mostrado a partir da linha 2ØØ. Conclusão: Você pode listar o programa a partir da linha que desejar.

Você já verificou que o NE-Z80 dispõe de inúmeros recursos que tornam bastante conveniente escrever programas com ele. Só resta, então, daqui prá frente, aprender a escrever programas que realmente façam alguma coisa útil — solucionar problemas aritméticos, por exemplo.

Capítulo 5

Sistematizando os problemas

As pessoas são capazes de interpretar instruções bastante confusas e solucionar problemas complexos. O computador não — o máximo que ele pode fazer é seguir uma lista de instruções e executar as instruções na forma como elas lhe chegam.

Como ilustração para o que estamos dizendo, tome as seguintes instruções que uma mãe poderia dar a seu filho pequeno:

"Você poderia ir até a padaria comprar pão? Pegue Cr$ 50,00 da minha carteira que está na mesa da cozinha. E, por favor, troque de roupa!"

Tudo parece estar claro: a criança sabe onde ir, o que comprar, onde está o dinheiro e também que deve mudar de roupa, antes.

Um robô controlado por computador, tomaria a primeira instrução: ir até a padaria, e a executaria.

A segunda instrução: comprar pão. Ele não poderia fazê-lo, está sem dinheiro. Nesse ponto provavelmente já pararia, confuso.

A terceira instrução: pegar Cr$ 50,00 na carteira, também não poderia segui-la.

A quarta instrução: ir até a mesa da cozinha. Ele voltaria para casa e iria para a mesa da cozinha.

Então a quinta instrução: trocar de roupa. O ingênuo robô procuraria trocar de roupa.

Uma criança instintivamente entende as instruções na seguinte ordem:

1. trocar-se
2. ir até a mesa da cozinha
3. pegar Cr$ 50,00 na carteira
4. ir até a padaria
5. comprar pão

E voltaria para casa!

A mãe, nesse exemplo, omitiu uma informação vital (volte para casa) e também colocou as instruções numa ordem ilógica. A criança usa o senso comum para interpretar instruções complexas, mas os computadores não têm essa capacidade. Você deve pensar em tudo antes de usar um computador. Por isso lembre-se: PONHA AS COISAS EM ORDEM!

Um meio de assegurar-se que está pondo tudo em ordem, é desenhar uma carta de fluxo, ou fluxograma, antes de começar realmente a programação. Isso também ajuda a perceber coisas que poderiam lhe escapar, por serem óbvias, mas que farão falta ao computador.

Por exemplo, o fluxograma para o exemplo que demos deverá ser algo como mostra a figura 2.

Cada retângulo contém uma instrução e as setas mostram a ordem em que elas devem ser executadas. Bem, isso parece simples e adequado.

Agora que o fluxograma está desenhado no papel, podemos checar se a criança (ou o computador) poderia fazer isso sem dúvidas, indo de retângulo em retângulo.

Começando pela primeira instrução (trocar de roupa), podemos achá-la óbvia e clara. Entretanto, a criança pode não encontrar suas roupas, por que a empregada as guardou no lugar errado. Talvez devamos adicionar uma instrução no início do programa: encontrar as roupas.

Ir até a mesa da cozinha. Essa parece OK.

Pegar Cr$ 50,00 da carteira. Bem, e se a carteira estiver vazia?

Ir à padaria. Será que a criança sabe onde é a padaria?

Comprar pão: de que tipo?

Voltar para casa. Essa também parece nada ter de errado.

Bem, parece que o fluxograma não estava tão completo quanto parecia. Mesmo as tarefas mais simples podem ser mais complicadas do que imaginamos. A maioria dos problemas que apontamos devem-se a falta de informação e isso pode ser coberto com o acréscimo de instruções extras no início do programa. Estas incluem:

1. Perguntar onde é a padaria
2. Achar as roupas

Portanto, quando você estiver escrevendo programas para o NE-Z80, é de vital importância dar ao computador toda a informação que ele necessitará, antes de levá-lo a executar uma tarefa.

Capítulo 6

Obtendo as respostas

Iremos, agora, usar o microcomputador para fazer alguns cálculos.

A INSTRUÇÃO LET

Quase todas as operações aritméticas são feitas usando a instrução LET. Da seguinte forma:

LET variável = expressão

Nesse caso, a variável (que pode ter qualquer nome que você queira) é o que se deseja calcular. A expressão, portanto, descreve como você quer definir a variável.

Por exemplo:

LET A = 2 + 4

variável expressão

Isso significa, "some 2 e 4 e faça com que A seja igual ao resultado".

Podemos usar outras variáveis:

LET A = 2 + B

variável expressão

ou

LET A = B + C

Nesse último caso há uma condição: a expressão é válida desde que o computador conheça **a priori** os valores de B e C.

Outra observação importante é que o sinal (=) não é usado de acordo com as regras matemáticas com as quais você está habituado.

Por exemplo:

LET J = J + 1

Obviamente J não é igual a J + 1. A instrução LET nesse caso quer dizer:

"Pegue o valor de J, adicione 1 a ele, e faça com que a variável J passe então a ter o valor dessa soma".

O sinal + é chamado de operador e define a operação que se deseja fazer. Os outros operadores aritméticos são:

- (menos)
* (multiplicação — não é usado o símbolo X para não haver confusão com a letra X)
/ (dividir)
** (potenciação)

Agora um exemplo de um programa que multiplica dois números.

```
10 PRINT "Multiplicação"
20 PRINT "Entre com o primeiro número"
30 INPUT A
40 PRINT "Entre com o segundo número"
50 INPUT B
60 LET C = A * B
70 PRINT "A resposta é", C
```

Entre com o programa e tente rodá-lo, usando dois números pequenos, tais como 17 e 25.

O computador mostrará a resposta na tela:

A resposta é 425

Rode novamente o programa com os números 255 e 129.

Aparecerá uma mensagem de erro dessa forma: 6/60. O 6 é o código para **overflow** (sobrecarga), porque a resposta é maior que 32767 (que, como vimos antes, é o número máximo que o computador pode escrever). E o 60 é a linha onde ocorreu tal erro.

Agora, tente editar o programa, fazendo uma adição ao invés de multiplicação — apenas mude de * para + na linha 60. Você também pode tentar uma subtração e verificar que obtém números negativos quando A é menor que B.

Agora, tente uma divisão. Altere a linha 60 para

60 LET C = A/B

Rode o programa (RUN) para A = 24 e B = 12

O computador mostrará: A resposta é 2

Rode o programa novamente, só que para A = 18 e B = 12

Aparecerá na tela: A resposta é 1 !! ??

Não deveria ser 1,5? Claro, mas o computador trabalha com aritmética inteira — ele pode expressar somente números inteiros. Acontece que ele efetua a divisão normalmente e considera o resto zero. Assim, por exemplo:

2,58 será 2
— 0,01 será 0
212,1 será 212
— 549,98 será — 549

Isso significa que é preciso tomar cuidado ao fazer uma divisão. O micro é bastante preciso quando se está dividindo um número grande por um pequeno, mas pode ser bem menos preciso quando os dois números são muito próximos.

Porém, existem formas de contornar esse problema; o programa a seguir é um exemplo de como fazer uma divisão com 3 casas decimais, usando apenas aritmética inteira.

```
10 PRINT "Programa Divisão"
20 PRINT "Dividendo = ?"
30 INPUT X
40 PRINT "Divisor=?"
50 INPUT Y
60 LET Z = X/Y (divide X por Y)
70 LET R1 = X - Z * Y (calcula o resto)
80 LET D1 = 10 * R1/Y (divide 10 * o resto por Y, o que dá a 1.ª casa decimal)
90 LET R2 = 10 * R1 - D1 * Y (calcula o segundo resto)
100 LET D2 = 10 * R2/Y (divide, dando a segunda casa decimal)
110 LET R3 = 10 * R2 - D2 * Y (calcula o 3.º resto)
120 LET D3 = 10 * R3/Y * (última casa decimal)
130 PRINT "A resposta é"; Z; ", "; D1; D2; D3
```

Observando bem, você nota que esse programa calcula a divisão exatamente como você o faz no papel.

Tente rodar o programa.

Observe a instrução PRINT da linha 130. Ela contém diversas variáveis separadas por ponto e vírgula. Os ponto e vírgulas indicam ao computador que cada coisa a imprimir deve ser impressa imediatamente após o item precedente, sem espaço entre eles.

Note também a ordem como são executadas as instruções a partir da linha 70. Elas mostram que você pode usar mais de uma operação por instrução. Mas deve seguir a ordem de prioridades das operações aritméticas.

Aqui está uma lista das prioridades para operações aritméticas:

Primeiro A ** B (A elevado a B)
Segundo —A (Negação ou multiplicação por —1)
Terceiro A * B
Quarto A/B
Quinto A+B ou A—B

Agora façamos algumas experiências.

```
10 REM PROGRAMA DE TESTE (REM é a chave Y)
20 PRINT "ENTRE com A"
30 INPUT A
40 PRINT "ENTRE COM B"
50 INPUT B
60 PRINT "ENTRE COM C"
70 INPUT C
80 LET Z = A + B * C
90 PRINT Z
```

Tente rodar o programa (RUN) com A = 1, B = 2 e C = 3, mas, antes de começar, use as regras de prioridades, nas operações, para saber qual resposta o NE-Z80 lhe dará: 7 ou 9?

Muito bem, rode o programa.

A resposta encontrada foi 7. Isso se explica porque primeiro o computador multiplicou 2 por três e depois adicionou 1 ao resultado.

Tente outras combinações substituindo (através do EDIT) a linha 80, por exemplo, por:

A * B/C
A/B * C
A ** B (A elevado a B, usando pequenos números com C = 0)
— A ** B
A * — B

Em caso de dúvida com essas regras, utilize o recurso dos parênteses, que facilitarão sensivelmente as operações. Como por exemplo em

Z = A/B * C

A seqüência normal seria o computador multiplicar B por C e então dividir A pelo resultado.

Mas, se reescrevermos essa equação colocando parênteses em torno de A/B, do seguinte modo:

80 LET Z = (A/B) * C

O computador fará primeiro a operação colocada entre parênteses, mesmo sendo ela de menor prioridade.

Observe, portanto, que existe uma importante diferença entre:

Z = A * (B/C)

e

Z = A * B/C

A primeira vista, pode-se pensar que dão a mesma resposta. E realmente isso pode acontecer — em certos casos!

Rode o programa com as duas formas de equação, usando A = 100, B = 25 e C = 5. A resposta será 500, para ambas.

Agora, tente novamente para A = 100, B = 3 e C = 5. A resposta deve ser 60.

Mas o que aconteceu quando do uso dos parênteses? Pense a respeito.

O computador primeiro fez 3/5, o que foi aproximado para zero. Depois ele multiplicou 100 por zero e naturalmente obteve ZERO como resposta final.

Esse exemplo destaca como é importante estar atento ao executar operações de multiplicação e divisão. As multiplicações podem causar sobrecargas (que farão o programa parar), enquanto o uso de números menores na divisão poderá resultar em respostas estranhas, sem indicação de que existe erro.

Se você perceber que irá cair nesse caso, rode o programa de modo que a multiplicação seja feita primeiro. De outro modo, é melhor fazer as divisões antes, para evitar sobrecarga.

Capítulo 7

Decisões

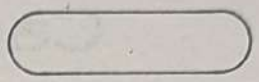
Até aqui consideramos problemas que puderam ser resolvidos executando uma lista de instruções, estritamente seguida do início até a última. Neste capítulo, e nos próximos, mostraremos alguns meios de fazer o NE-Z80 executar programas muito mais complexos, que ajudarão a resolver algumas das nossas tarefas comuns mais enfadonhas.

Um pouco mais atrás falamos rapidamente do uso de fluxogramas para verificação dos passos de um programa quanto a uma seqüência lógica. Demonstramos a utilidade desse recurso para os programas mais simples. Quando chegamos a programas mais complexos, o fluxograma torna-se vital.

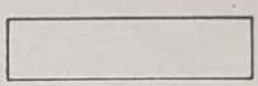
Consideremos um problema do nosso cotidiano — levantar-se pela manhã. Na figura 2 apresentamos um fluxograma para esse ato do dia-a-dia. Nele você observa que existem ramos que são seguidos apenas mediante o resultado prévio de certas alternativas. Isto é, o computador toma a decisão de seguir determinado caminho, em função de um resultado pré-estabelecido.

Os programas podem tornar-se bastante difíceis de entender se aparecerem muitos ramos e aí é que os fluxogramas podem ajudar. Por isso é uma idéia muito boa a de desenhar um fluxograma do seu programa antes de começar a escrevê-lo em definitivo.

Abaixo, 3 tipos de simbolos(ou blocos)



Inicio ou término



Bloco de processamento.

Nele se encontra qualquer operação a ser realizada incondicionalmente, isto é, sem que dependa de uma forma de decisão.

Bloco de decisão.

Em função da alternativa presente no bloco, o programa é desviado para um caminho ou outro do fluxograma.

Capítulo 8

Ramificações

Antes de partirmos para decisões, realmente, vejamos a instrução GO TO.

Ela deve ser escrita da seguinte forma:

GO TO n, onde n é normalmente o número de uma linha. Mas, no caso do NE-Z80, poderá ser também uma variável, uma capacidade exclusiva desse microcomputador.

Quando o NE-Z80 recebe a instrução GO TO n, ele salta imediatamente para essa linha do programa (n) e executa a instrução nela contida. Se n for uma variável, X por exemplo, o computador verificará o valor de X e pulará para a instrução com aquele número — supondo que haja uma instrução com tal número no seu programa.

Além disso, n poderá ser até mesmo uma operação, tal como A/1Ø, outra possibilidade prática exclusiva do NE-Z80.

Agora um pequeno programa que ilustra o uso de GO TO.

```
1Ø PRINT "Esta é a instrução 1Ø"
2Ø GO TO 4Ø
3Ø PRINT "Esta é a instrução 3Ø"
4Ø PRINT "Fim do programa"
```

Agora rode esse programa e veja o que o computador irá imprimir:

```
Esta é a instrução 1Ø
Fim do programa
```

E a instrução 3Ø? O computador simplesmente "saltou" sobre ela, por que a instrução 2Ø mandava ir (GO TO) para 4Ø.

Com isso você pode concluir que a instrução GO TO n provoca um salto incondicional no programa.

Outra utilidade do comando GO TO. Faça agora GO TO 3Ø e em seguida aperte NEWLINE. O micro imprimirá:

```
Esta é a instrução 3Ø
Fim do programa
```

O que demonstra o uso de GO TO para fazer o computador executar o programa a partir de determinada linha que se queira (no caso ilustrado à linha 3Ø).

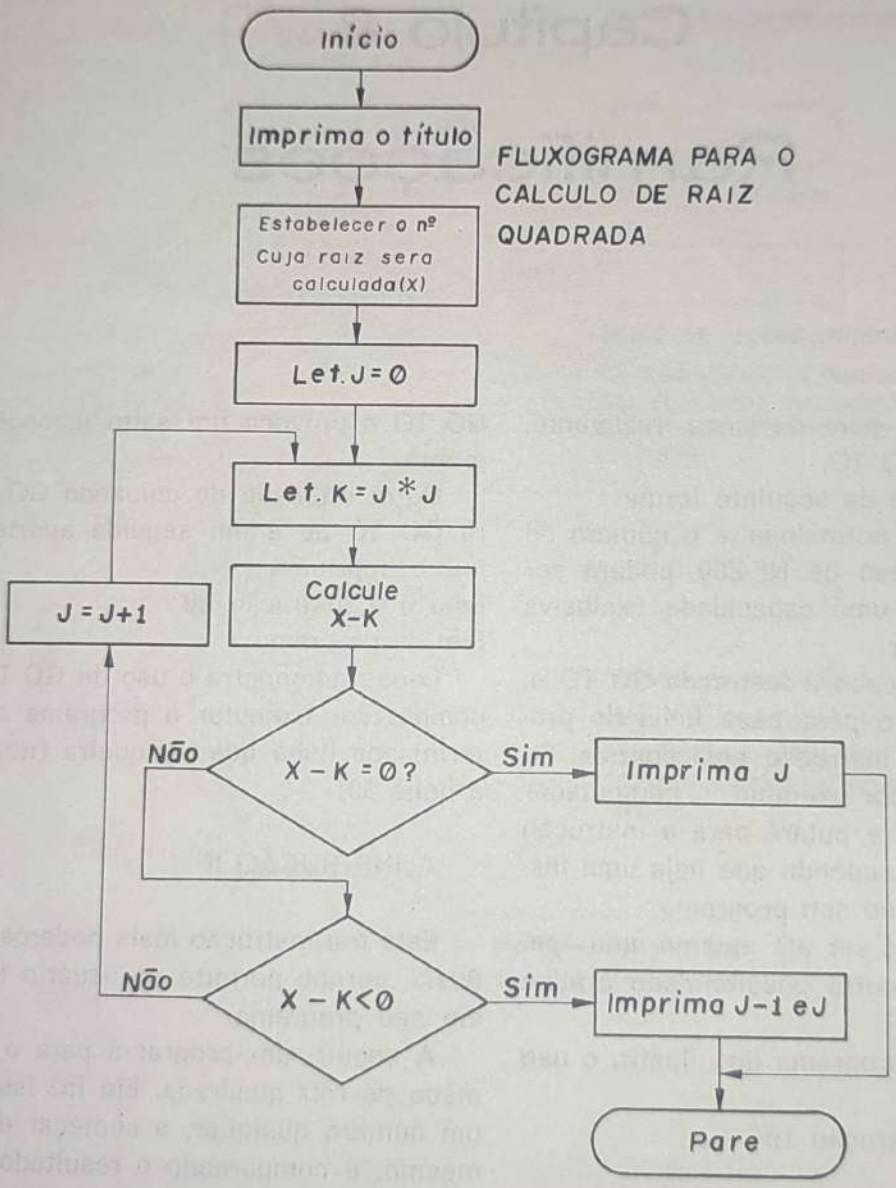
A INSTRUÇÃO IF

Esta é a instrução mais poderosa da linguagem Basic, porque permite ao usuário incluir decisões em seu programa.

A seguir, um programa para o cálculo aproximado de raiz quadrada. Ele faz isso multiplicando um número qualquer, à começar de zero, por ele mesmo, e comparando o resultado com o número do qual se deseja extrair a raiz. Sempre que o produto for menor que o número a ter sua raiz calculada, a variável será incrementada em 1 e o computador tentará o mesmo processo novamente.

```
1Ø PRINT "Rotina de raiz quadrada"
2Ø PRINT "Entre com o número desejado"
3Ø INPUT X
4Ø LET J = Ø
5Ø LET K = J * J
6Ø LET D = X - K
```

FLUXOGRAMA PARA O CALCULO DE RAIZ QUADRADA



```
7Ø IF D = Ø THEN GO TO 11Ø (Use SHIFT 3 para obter THEN)
8Ø IF D < Ø THEN GO TO 13Ø
9Ø LET J = J + 1
1ØØ GO TO 5Ø
11Ø PRINT "A raiz é"; J
12Ø GO TO 14Ø
13Ø PRINT "A raiz está entre"; (J−1); "e"; J
14Ø STOP
```

Para melhor entender esse programa, você também pode acompanhá-lo através do fluxograma da figura 3.

A instrução IF (se) sempre deverá ser seguida de THEN (então), o que resumidamente implica na seguinte forma para a instrução:

IF (condição a ser satisfeita) THEN (então faça isso)

Os operadores que acompanham a instrução IF são:

= igual a
< menor que
> maior que

Exemplos:

A = 2 A igual a 2
A < 1Ø A menor que 1Ø
A + B > C + D A+B maior que C+D

Nesses casos, A, 2, 10, A + B, C + D, são as expressões.

As expressões não necessariamente precisam ser números inteiros ou variáveis inteiras. Também é válido usar esse comando para uma instrução do tipo:

IF A = Z − (Z/B) B THEN GO TO 1Ø
(Z − (Z/B) = expressão)

A condição a ser satisfeita poderá ser ainda mais complexa porque **operadores lógicos** podem ser utilizados.

Por exemplo,

```
1Ø IF NOT A = 2 THEN GO TO 1ØØ
```

Aqui, a função NOT (NÃO) nega a condição subseqüente, de modo que se A **não** for igual a 2 é que o computador saltará para 1ØØ.

Agora um programa que ilustra o uso do operador lógico AND (E).

```
1Ø INPUT A
2Ø INPUT B
3Ø INPUT C
4Ø IF A = 1 AND B = 1 AND C = 1 THEN PRINT "OK"
```

Para obter AND, use SHIFT 2.

Assim para as operações aritméticas existem prioridades, para esses operadores lógicos, também há uma ordem:

NOT, AND, OR

Exemplificando, na expressão IF A = 1 OR B = 1 AND C = 1 THEN PRINT "OK", primeiro é visto se B = 1 e C = 1 para imprimir OK. Só após essa condição ser verificada, se as variáveis A e B não forem iguais a 1, o computador opta (OR) pela verificação de A = 1, para imprimir OK.

Portanto, a condição para impressão de OK seria:

B e C = 1 ou então A = 1

Os operadores NOT, AND e OR, também possibilitam que você produza **expressões condicionais.** Por exemplo:

X = 3 se A > B
X = P se A < B

Estas condições poderão ser escritas como expressões assim:

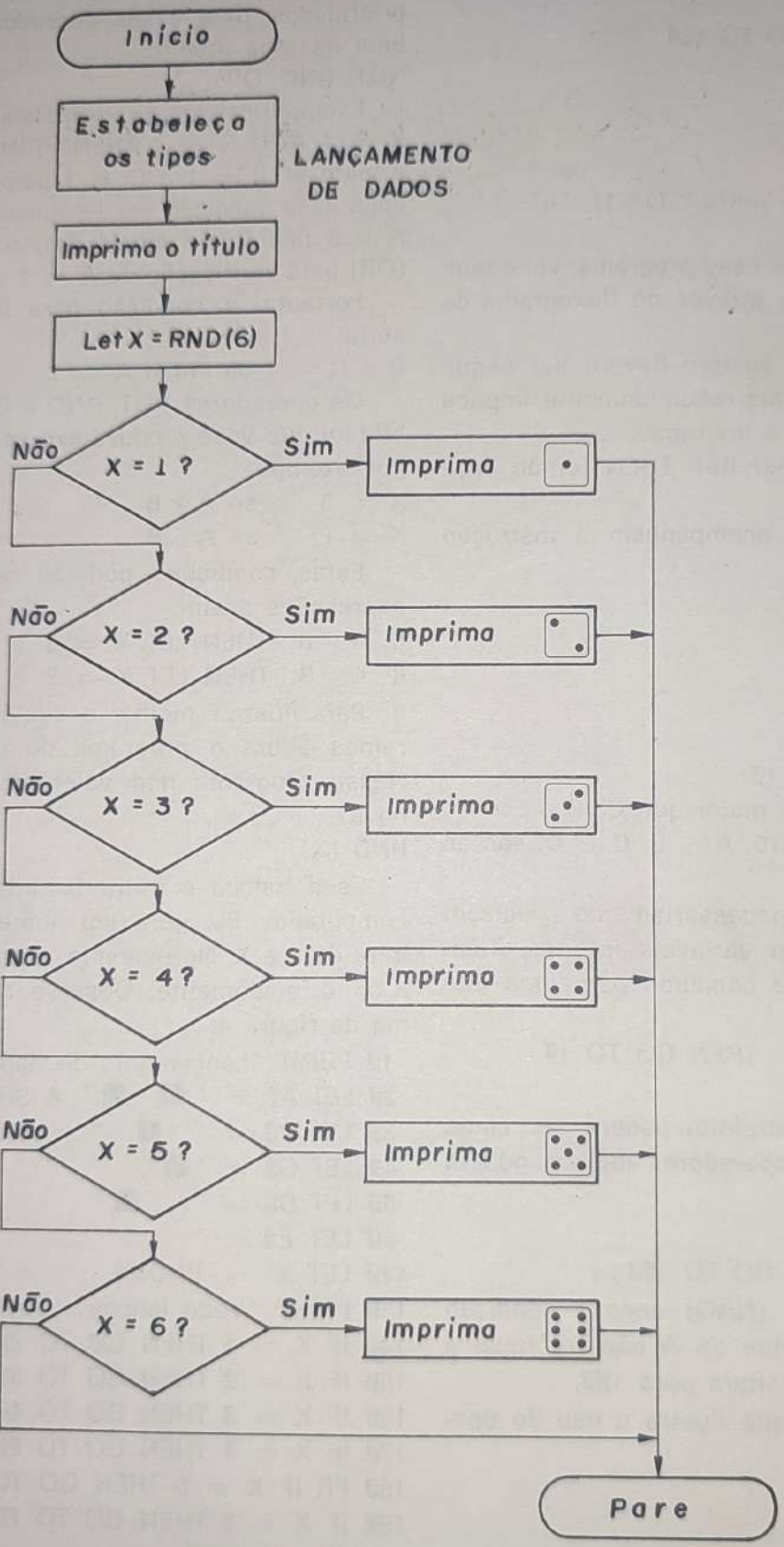
IF A > B THEN LET X = 3
IF A < B THEN LET X = P

Para ilustrar melhor o que vimos até aqui, teremos agora o programa de um jogo de dado. Nesse programa nos valeremos de uma função nova:

RND (X)

Essa função é outra facilidade oferecida pelo computador. Ela gera um número aleatório no limite de 1 a X. No exemplo, para um jogo de dado, X = 6, obviamente. Observe também o fluxograma da figura 4.

```
1Ø PRINT "Lançamento do dado"
2Ø LET A$ = "■...■" é SHIFT A
3Ø LET B$ = "..■.."
4Ø LET C$ = "■....."
5Ø LET D$ = "....■"
6Ø LET E$ = "....."
12Ø LET X = RND(6)
13Ø PRINT "Você lançou..."
14Ø IF X = 1 THEN GO TO 2ØØ
15Ø IF X = 2 THEN GO TO 3ØØ
16Ø IF X = 3 THEN GO TO 4ØØ
17Ø IF X = 4 THEN GO TO 5ØØ
18Ø PR IF X = 5 THEN GO TO 6ØØ
19Ø IF X = 6 THEN GO TO 7ØØ
195 GO TO 1ØØØ
2ØØ PRINT E$
2Ø5 PRINT E$
21Ø PRINT B$
```


Início
Estabeleça os tipos
LANÇAMENTO DE DADOS
Imprima o título
Let X = RND(6)
Não
X = 1 ?
Sim
Imprima
Não
X = 2 ?
Sim
Imprima
Não
X = 3 ?
Sim
Imprima
Não
X = 4 ?
Sim
Imprima
Não
X = 5 ?
Sim
Imprima
Não
X = 6 ?
Sim
Imprima
Pare

```
215 PRINT E$
220 PRINT E$
230 GO TO 1000
300 PRINT C$
305 PRINT E$
310 PRINT E$
315 PRINT E$
320 PRINT D$
330 GO TO 1000
400 PRINT D$
405 PRINT E$
410 PRINT B$
415 PRINT E$
420 PRINT C$
430 GO TO 1000
500 PRINT A$
505 PRINT E$
510 PRINT E$
515 PRINT E$
520 PRINT A$
530 GO TO 1000
600 PRINT A$
605 PRINT E$
610 PRINT B$
615 PRINT E$
620 PRINT A$
630 GO TO 1000
700 PRINT A$
705 PRINT E$
710 PRINT E$
715 PRINT A$
720 PRINT A$
1000STOP
```

Se você quiser fazer o programa estar apto para recomeçar o jogo por si mesmo, proceda do seguinte modo:

```
1000 PRINT "Aperte NEWLINE para jogar novamente"
1100 INPUT X$
1200 CLS     (aperte SHIFT C para CLS)
1300 IF X$ = " " THEN GO TO 120
```

Capítulo 9

Programas interativos

É muito freqüente nos depararmos com problemas que envolvam repetição de operações para serem solucionados. Um exemplo desse tipo de problema é a extração de raiz quadrada, como vimos no capítulo 8. Chamamos os programas que resolvem esses problemas, de **interativos**. Através de técnicas de interação, podemos fazer o computador trabalhar bastante a partir de um programa bem curto.

Observe o exemplo a seguir:

```
10 LET J = 1
20 PRINT "$"
30 IF J = 152 THEN GO TO 60
40 LET J = J + 1
50 GO TO 20
60 STOP
```

Esse programa imprime $ 152 vezes, valendo-se do uso de **loops**, através da instrução GO TO. As setas desenhadas no programa servem para facilitar a sua visualização da seqüência que o programa percorre. No caso de programas curtos fica mais fácil desenhar essas setas do que um fluxograma.

Os **loops** são tão úteis que você já pode estar imaginando que existe um meio mais simples de escrever programas usando-os. E realmente há. Como veremos no capítulo 10.

Capítulo 10

Loops

Devido à utilidade dos **loops**, algumas instruções especiais foram inventadas para torná-los ainda mais fáceis de usar.

Veja o programa para imprimir $:

```
1Ø FOR J = 1 TO 152 (use SHIFT 4 para TO)
2Ø PRINT "$";
6Ø NEXT J
```

No capítulo 9 foram tomadas cinco linhas de programa, usando as instruções IF e GO TO, para se conseguir o mesmo resultado que você poderá obter agora rodando esse programa.

Observe o uso do ponto e vírgula (;) depois do comando PRINT. Isso informa ao computador para imprimir cada $ imediatamente após seu predecessor.

Em linguagem Basic esses **loops** são normalmente chamados de FOR **loops**. Mas também são bastante conhecidos como DO **loops**, porque outras linguagens, como DO **loops**, porque outras linguagens, como FORTRAN, usam DO em lugar de FOR.

Nada nos impede de fazer **loops** dentro de **loops**, ou colocar vários deles dentro de um maior. Adicione as linhas a seguir ao programa que vimos há pouco:

```
3Ø FOR I = 1 TO 3
4Ø PRINT "£"
5Ø NEXT I
```

O fluxograma da figura 5 ilustra esse novo programa com diversos **loops**.

SUB-ROTINAS

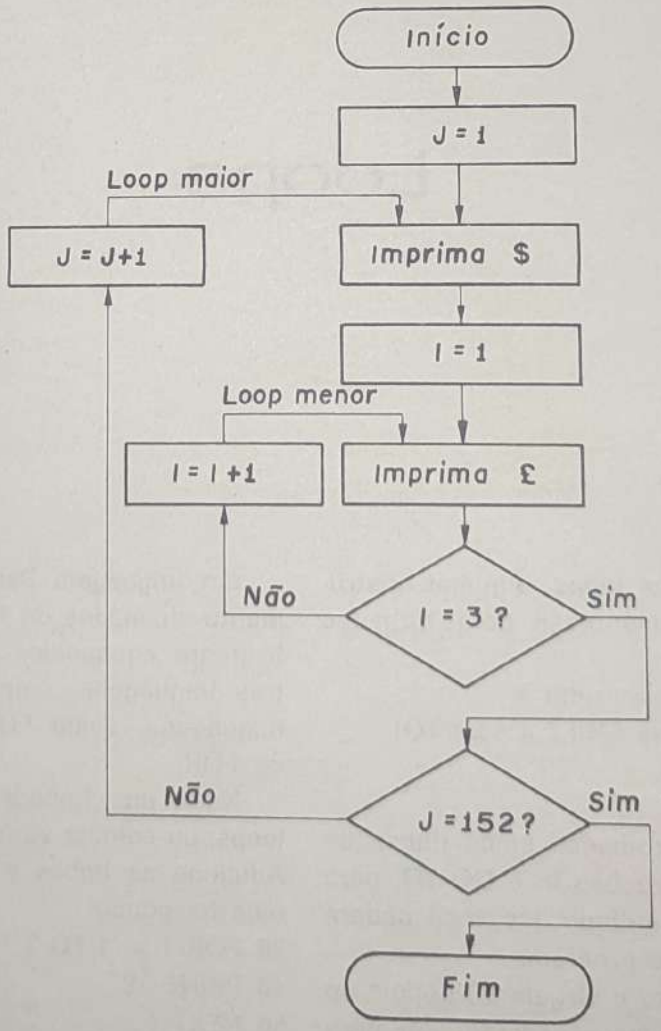
Uma sub-rotina é um sub-programa que pode ser usado uma ou mais vezes pelo programa principal.

Exemplificando:

```
10 FOR J = 1 TO 10
20 GO SUB 1000
30 NEXT J
40 PRINT "FIM"
900 GO TO 1200
1000 PRINT "Sub-rotina executada"
1100 RETURN (SHIFT tecla B)
1200 STOP
```

A instrução 20 manda o computador ir para a linha 1000, que é a sub-rotina. No exemplo trata-se apenas de imprimir "Sub-rotina executada".

A instrução 1100 informa ao micro que a sub-rotina terminou e que ele deve retornar à execução do programa principal. O computador então salta para a linha imediatamente seguinte à GO SUB e executa a instrução nela contida.



Capítulo 11

Como imprimir

Já estamos usando a instrução PRINT há um bom tempo, sempre que há necessidade de imprimir variáveis e expressões. De fato, a instrução PRINT é muito versátil. A forma genérica dessa instrução é

PRINT expressão ch...expressão ch... e assim por diante. Ch (caracter de controle) pode ser uma vírgula (,), um ponto e vírgula (;) ou nada (somente no fim da instrução). O objetivo do uso de caracteres de controle é permitir o controle do espaçamento da linha a ser impressa. Já observamos o uso do ponto e vírgula (;) nas instruções PRINT.

Tomemos um exemplo:

```
10 LET X = 4
20 PRINT "A resposta é"; X; "unidades"
```

(controle: ; ; expressões: "A resposta é", X, "unidades")

Esse programa deverá imprimir: A resposta é 4 unidades

O ponto e vírgula (;) faz com que o computador imprima as expressões sem qualquer espaço entre elas. Se você quiser espaço será preciso prevê-lo na sentença literal a ser impressa.

A vírgula (,) é usada como tabulador. Cada linha de 32 caracteres mostrada na tela é dividida em quatro campos de 8 caracteres de comprimento. Cada vez que o computador percebe uma vírgula numa instrução PRINT, ele começa a imprimir a expressão a partir do póximo campo disponível.

O objetivo disso é a divisão da tela em quatro colunas. Se uma variável tem mais de 7 caracteres de comprimento e é seguida por uma vírgula, o computador ocupa dois ou mais campos para imprimir a sentença e começa a próxima expressão no início do terceiro campo.

Por exemplo:

```
PRINT "Campo 1", "Campo 2", "Campo 3"
```

Deve aparecer na tela:

```
Campo 1     Campo 2     Campo 3
```

Mas,

```
PRINT "Campo 1",, "Campo 2", "Campo 3"
```

(vírgula extra)

Deve produzir

```
Campo 1             Campo 2   Campo 3
```

Já a instrução:

```
PRINT "Este é um campo",, "Campo 1", "Campo 2"
```

Irá resultar:

```
Este é um campo        Campo 1
Campo 2
```

Nesse último caso, o computador ocupou dois campos para a primeira expressão, pulou um campo e continuou. Ele saiu fora dos campos da primeira linha e continuou no primeiro campo da segunda linha.

Não há limites para o número de campos que você pode pular utilizando vírgulas (a menos que saia da área visível da tela, é claro).

A melhor forma de aprender a lidar com a instrução PRINT é experimentá-la de várias maneiras.

Capítulo 12

Copiando com caracteres

Você deve ter notado que, além dos caracteres normais de uma máquina de escrever, o teclado do NE-Z80 apresenta vários outros que não são normalmente encontrados. E, ainda mais, existem também alguns caracteres úteis que não estão escritos no teclado.

Antes de estudarmos as possibilidades do teclado, observemos uma função bastante interessante:
CHR$(X), onde X é um número, uma variável ou expressão.

Essa função é empregada principalmente em conjunto com a instrução PRINT. O seu significado é "o caracter cujo código é X".

Tome o programa a seguir:

```
1Ø INPUT X
2Ø PRINT CHR$(X)
3Ø GO TO 1Ø
```

Esse programa pede um número e então imprime o caracter cujo código é o número introduzido. Os códigos situam-se entre Ø e 255.

Um programa cuja utilidade é listar os símbolos, é dado a seguir.

```
1Ø PRINT "Entre com o valor do código"
2Ø INPUT X
3Ø PRINT X;" . . . ";CHR$(X)
4Ø PRINT
5Ø LET X = X + 1
6Ø GO TO 3Ø
```

Ele lista 11 símbolos com seus respectivos códigos, começando com o código introduzido no início do programa.

Veja agora a tabela de caracteres correspondentes aos diferentes códigos:

Código	Caracter
0	espaço
1	
2	
3	
4	
5	gráficos
6	
7	
8	
9	
10	
11	
12	£
13	$
14	:
15	?
16	(
17	)
18	–
19	+
20	*
21	/
22	=
23	>
24	<
25	
26	
27	
28	0
29	1
30	2
31	3
32	4
33	5
34	6
35	7
36	8
37	9
38	A
39	B
40	C

Código	Caracter
41	D
42	E
43	F
44	G
45	H
46	I
47	J
48	K
49	L
50	M
51	N
52	O
53	P
54	Q
55	R
56	S
57	T
58	U
59	V
60	W
61	X
62	Y
63	Z
64 a 127	?
128 a 191	Repete-se a seqüência de 0 até no inverso
192 a 211	?
212	"
213	THEN
214	T O
215	;
216	,
217	)
218	(
219	NOT
220	–
221	+
222	*
223	/

Código	Caracter
224	AND
225	OR
226	**
227	=
228	>
229	<
230	LIST
231	RETURN
232	CLS
233	DIM
234	SAVE
235	FOR
236	GO TO
237	POKE
238	INPUT
239	RAND

Código	Caracter
240	LET
241	?
242	?
243	NEXT
244	PRINT
245	?
246	NEW
247	RUN
248	STOP
249	CONTINUE
250	IF
251	GO SUB
252	LOAD
253	CLEAR
254	REM
255	?

O inverso, a que nos referimos para os códigos 128 a 191, significa que os caracteres, no caso, aparecerão em branco sobre um fundo preto, ou seja "no negativo". O espaço inverso, por exemplo, é um quadrado preto.

Os caracteres gráficos estão ilustrados na figura 6.

Existem ainda algumas facilidades que nos permitem manipular os caracteres:

a) TL$ (seqüência)

Esta mostra a seqüência retirando da mesma o primeiro caracter. Por exemplo:

10 PRINT TL$ ("ABC")

Isso daria:

BC

b) CODE (seqüência)

Este mostra o código do primeiro caracter da seqüência. Exemplo:

10 PRINT CODE ("ABC")

O computador deve imprimir 38, que é o código de A.

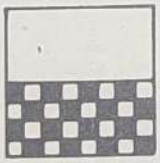
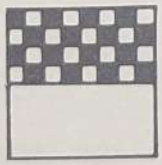
SÍMBOLO	CÓDIGO	SÍMBOLO	CÓDIGO	SÍMBOLO	CÓDIGO	SÍMBOLO	CÓDIGO
	2		7		130		135
	3		8		131		136
	4		9		132		137
	5		10		133		138
	6		11		134		139

Tabela de Simbolos Gráficos

Capítulo 13

Miscelânea de funções

Este capítulo dedica-se às funções não vistas até aqui e àquelas que já foram vistas sem muito detalhe.

RND (X) — Fornece um número aleatório entre 1 e X.

Uma instrução típica usando RND

10 LET J = RND(X)

Outra instrução útil, POKE. Ela é usada na forma:

POKE A,B — onde A é o endereço de um lugar na memória e B é uma expressão.

Tipicamente, A pode ser o endereço de um dos dois bytes que formam a variável que atua na tela.

Uma outra instrução:

LET X = PEEK(A) — Isso faz com que X assuma o valor que se encontra na posição de memória dada por A. A é sempre menor que 256 (0 – 255).

A razão porque as variáveis associadas com POKE e PEEK devem ser sempre menores que 256 é que tudo na memória do NE-Z80 é armazenado em 8 bytes (1 byte). O número máximo que se pode escrever com 8 bits é $2^8 = 256$ e assim todas as variáveis usam até dois bytes, permitindo que o número máximo possível de ser guardado seja 32767.

Cada metade de uma variável é armazenada em um byte, e todo byte tem um endereço associado.

Outra instrução não vista é o módulo:

ABS(n) — É uma função que fornece o módulo de n.

Exemplo:

ABS(5) = 5

ABS(−5) = 5

Capítulo 14

O computador em suas mãos

Se você leu este manual e acompanhou os exemplos, rodando os programas e escrevendo os seus próprios, já está bem encaminhado para tornar-se um fluente programador Basic. Lembre-se, porém, que com o NE-Z80 será possível fazer muitas coisas que não são alcançadas usando outros tipos de computador. Do mesmo modo, alguns Basic influem características não presentes no NE-Z80.

Finalizamos com um apêndice que contém um código de erros e algumas informações adicionais sobre o uso do NE-Z80 com o vídeo.

Apêndice I

Códigos de erros

Quando os resultados são mostrados na tela, também aparece um código n/m: n é o número de erros e m é um número de linha no programa.

instrução	código	significado
	Ø	(m = próxima linha que teria sido executada) BREAK pressionado. (m = −1 ou −2) comando completado com sucesso. (m<Ø ou > que o maior número de linha no programa) GO TO m foi executado. (m = maior número de linha no programa) fim do programa.
NEXT	1	* NEXT <id> onde <id> não é a variável de controle de um **loop** ativo FOR.
qualquer	2	* nome da variável não encontrado (qualquer variável ou mesmo o nome de qualquer elemento sendo usado).
qualquer	3	* subscrição fora da faixa ou erro de qualquer tipo na avaliação de uma subscrição.
LET INPUT DIM PRINT	4	* Não há espaço para adicionar novas variáveis ou uma frase maior na tela.
PRINT	5	* Não há mais espaço na tela.
qualquer	6	Sobrecarga aritmética (resultado > 32767 ou < − 32768, sendo também resultado = − 32768, em algumas circunstâncias.
RETURN	7	* RETURN com a correspondente GO SUB.
INPUT	8	A entrada só pode ser usada num programa, indiretamente.
STOP	9	Instrução STOP executada.

m = número da linha da instrução problemática
CONTINUE é o mesmo que GO TO m, exceto depois do código 9, quando ela é GO TO m + 1.

Apêndice II

a) **Listando programas** — Você insere, através do teclado, linhas de Basic (programa e comandos) para execução imediata. Enquanto está fazendo isso, você observa que a tela está dividida em duas partes. A parte superior é uma "janela" onde o programa é listado, enquanto a parte inferior mostra a linha ou comando que está sendo chamado.

b) **Área de entrada** — A parte inferior da tela contém a linha que você está teclando. Essa linha pode ser um comando ou uma linha do programa.

Algumas vezes, um cursor é mostrado na linha. Isso indica duas coisas: a posição na linha onde os símbolos serão inseridos; e se uma tecla foi apertada como letra (por exemplo "A") ou como comando (isto é "LIST"). A forma do cursor no vídeo, para palavras é [K] e [L] para letras.

Note que esse cursor, embora apareça numa linha e ocupe uma posição de caracter na tela, não toma parte da linha e é ignorado em qualquer interpretação da linha.

Um segundo símbolo, em princípio similar ao cursor, pode também aparecer no vídeo. Sua forma é [S] e ele indica que a linha não está sintaticamente correta como instrução Basic.

As teclas a seguir são usadas para alterar a entrada.

(1) — Caracteres simples: letras, dígitos, pontuação, etc., o símbolo é inserido à esquerda do cursor.

(2) — Multi-caracteres: "sinais"; "AND"; "OR"; "NOT"; "**"; "TO", "THEN";

Cada um desses caracteres está armazenado no computador através de um único byte que, como no caso (1), são inseridos à esquerda do cursor. No entanto, eles aparecem na tela de forma maior que um caracter. Aqueles que são alfanuméricos (isto é, todos exceto "**"), são precedidos e seguidos por um espaço.

(3) "RUBOUT", elimina o símbolo ou sinal à esquerda do cursor.

(4) Chaves de controle " → " e " ← ", deslocam o sinal ou símbolo para a direita ou esquerda.

(5) "EDIT", já visto anteriormente.

(6) "NEWLINE", é ignorada se o símbolo [S] estiver presente. Fora isso, é usada para identificar o fim da linha.

ANOTAÇÕES