Deixe com Autosaver a tarefa de gravar periodicamente o seu programa no Apple e não esquente mais a cabeça com falta de energia elétrica.

Autosaver

Antonio Augusto Gorni

Todo usuário de microcomputadores certamente já teve a oportunidade de verificar a total vulnerabilidade da memória de seu equipamento por ocasião de uma falha no fornecimento de energia elétrica. Para complicar as coisas, a famigerada lei de *Murphy* fará com que tal evento ocorra quando você está prestes a salvar no disco um enorme programa que lhe tomou horas de digitação.

Justamente por causa de tais problemas, programadores experientes salvam periodicamente seus programas durante a fase de digitação. É um sábio conselho, mas que não funciona com programadores esquecidos, que é justamente o meu caso. Outra solução é o uso de sistemas tipo no-break, ou seja, geradores eletrônicos para garantir o funcionamento do computador por um período mínimo de tempo após a falta de energia. Entretanto, o alto custo desta solução só a torna viável para sistemas comerciais.

Para resolver esse problema de forma razoável, resolvi dotar meu micro com um recurso existente nos processadores de texto dos computadores de grande porte, ou seja, fazendo com que ele salve automaticamente (em disco) o programa em BASIC cada vez que for digitado um certo número de linhas. Para tal, desenvolvi uma pequena rotina em linguagem de máquina, ligada ao sistema operacional. Esta rotina foi desenvolvida para o microcomputador Exato Pro e serve para todas as máquinas compatíveis com o Apple Plus que utilizem o DOS 3.3.

Listagem 1

```
10  REM  ***
15  REM  ***                    A U
       T O     -     S A V E
      R
20  REM  ***
25  REM  ***          Modulo BASIC p
    ara Instalacao das Subrotina
    s Assembler
30  REM  ***
35  REM  ***            Antonio Aug
    usto Gorni --- 27 de Outubro
     de 1985
40  REM  ***
100 TEXT : HOME :D$ =  CHR$ (4):
    B$ =  CHR$ (7): PRINT D$;
110 INVERSE : HTAB 15: PRINT "AU
    TO-SAVER"
120 VTAB 9
130 NORMAL : INPUT "PERIODO DE L
    INHAS PARA SALVAMENTO? ";NL%

140 IF NL% > 255 THEN  PRINT B$;
    : GOTO 120
150 VTAB 15
160 INPUT "NOME DO ARQUIVO PROVI
    SORIO? ";A0$
170 IF  LEN (A0$) > 10 THEN  PRINT
    B$;: GOTO 150
180 VTAB 22
190 FLASH : PRINT "EM INSTALACAO
    ...";
200 NORMAL
210 A0$ = "SAVE" + A0$
220 PRINT : PRINT D$"BLOAD AUTOS
    AVER.ASMB"
230 POKE 796,NL%: POKE 797,0: POKE
    798,0
240 M = 768: FOR I = 40657 TO 406
    81: POKE M, PEEK (I):M = M +
    1: NEXT
250 POKE 40577,32: POKE 40578,0:
     POKE 40579,3
260 L =  LEN (A0$):ED = 799
270 FOR I = 1 TO L: POKE ED + I -
    1, ASC ( MID$ (A0$,I,1)) + 1
    28: NEXT : POKE ED + L,141
280 TEXT : HOME
290 PRINT B$;
300 VTAB 6
310 INVERSE : HTAB 11: PRINT "AU
    TOSAVER     INSTALADO!"
320 VTAB 14
330 NORMAL : PRINT "    PARA DES
    ATIVA-LO:     POKE 793,96"
340 VTAB 18
350 PRINT "    PARA REATIVA-LO:
        POKE 793,76"4
360 PRINT
370 NEW
```

Listagem 2

```
0319- 4C 2E 03 A0 C9 84 FF
0320- A0 8A AA AC C1 FA C1 A0
0328- 9C C5 85 A0 A2 A0 AD 1E
0330- 03 C9 01 D0 08 C6 24 A9
0338- 00 8D 1E 03 60 A4 24 88
0340- B1 28 C9 DD F0 01 60 AE
0348- 1D 03 E8 EC 1C 03 F0 04
0350- BE 1D 03 60 A2 00 8E 1D
0358- 03 A0 15 B9 1F 03 99 00
0360- 02 88 10 F7 A9 01 8D 1E
0368- 03 4C CD 9F
```

PROCEDIMENTO BÁSICO DO PROGRAMA

As ações executadas pela rotina são as seguintes:

1 – Interceptar todo caráter digitado;
2 – Analisar o caráter e verificar se foi terminada a digitação de uma linha;
3 – Se a linha foi terminada, somar 1 ao contador de linhas, caso contrário voltar ao passo 1;
4 – Caso o número de linhas digitadas até o momento for inferior ao estabelecido, voltar ao passo 1. Entretanto, se já foi atingido o número de linhas especificado, o programa em BASIC na memória é salvo e o contador de linhas é zerado, voltando-se ao passo 1.

Pelas ações que a rotina executa, é possível observar que ela deve estar ligada ao Sistema Operacional para que possa interceptar os caracteres digitados. Outra questão a ser resolvida consiste em executar a instrução SAVE do DOS 3.3 em nível de linguagem de máquina. A informação necessária para a solução destes dois problemas está no livro **"Beneath Apple DOS"** de *Don Worth* e *Pieter Lechner*.

OS PROGRAMAS

A listagem 1 mostra o programa em BASIC utilizado para instalar as rotinas em linguagem de máquina que estão na listagem 2.

A função do programa em BASIC é montar as sub-rotinas em linguagem de máquina na memória e ligá-las ao DOS, bem como permitir a entrada dos dados necessários.

Veremos agora como foram implantadas e como funcionam as rotinas em linguagem de máquina.

Em primeiro lugar, foi estabelecido que a rotina em linguagem de máquina começaria a partir da posição de memória $300 (hexadecimal), pois a região de memória entre $300-$3FF está disponível ao usuário sob condições normais de operação para a implantação de pequenas rotinas em linguagem de máquina.

A primeira coisa que nossa rotina deve fazer é interceptar o caráter digitado. Ora, a rotina padrão do DOS 3.3 para interceptar os caracteres do teclado começa na posição $9EBD. Ao examinarse este trecho do DOS com o auxílio do Monitor verificamos que esta rotina começa chamando outra sub-rotina, cuja função é salvar o conteúdo dos registradores do processador 6502, e está localizada nas posições de memória $9ED1-$9EEA.

Tal fato pode ser utilizado de modo a conseguirmos ligar nossa rotina ao DOS. Primeiramente, transferimos essa rotina de sua posição original ($9ED1-$9EEA) para as posições originais de nossa sub-rotina ($300-$318). A nossa rotina propriamente dita para salvar o programa será colocada após esta rotina transplantada do DOS, ou seja, a partir da posição $319.

Esta transferência de sub-rotina é realizada na linha 240 do programa em BASIC da listagem 1. Obviamente temos de avisar o DOS sobre a modificação efetuada, o que é feito na linha 250 do programa em BASIC, alterando-se o endereço de chamada da sub-rotina original.

Conforme já foi dito, após a rotina transferida do DOS coloca-se a sub-rotina para salvamento automático, a qual é comentada na listagem 3. Pode-se observar que primeiramente é definida uma área de variáveis, as quais são preenchidas pelo programa em BASIC; a seguir, a linha 230 zera o contador de linhas (Count) e o indicador de programa salvo (Flag), bem como coloca o número de linha estabelecido (Limite). A linha 270 monta o comando **SAVE** <nome do arquivo> na variável Buf.

Os comentários da listagem 3 explicam o funcionamento da rotina em linguagem de máquina.

Listagem 3

```
 1          ORG $319       - MONTA PROGRAMA A PARTIR DE $319
 2          JMP CONT       - VAI PARA A SUBROTINA
 3 *
 4 * ***    DEFINICAO DE VARIAVEIS
 5 *
 6 LIMITE   DS 1           - NUMERO DE LINHAS
 7 COUNT    DS 1           - CONTADOR DE LINHAS
 8 FLAG     DS 1           - INDICADOR DE PROGRAMA SALVO
 9 BUF      DS 15          - NOME DO ARQUIVO PROVISORIO
10 *
11 * ***    SUBROTINA
12 *
13 CONT     LDA FLAG       - CARREGA FLAG
14          BNE * $01      - PROGRAMA ACABOU DE SER SALVO?
15          BNE COUNT1     - NAO, CONTINUA NORMALMENTE
16          DEC $24        - SIM, REPOSICIONA CURSOR NA TELA
17          LDA #$00       - RESETA FLAG
18          STA FLAG
19          RTS            - RETORNA CONTROLE AO TECLADO
20 COUNT1   LDY $24
21          DEY
22          LDA($28),Y     - CARREGA EM "A" O CARATER DIGITADO
23          CMP #$DD       - CARATER E' "]"(cursor)
24          BEQ INCR       - SIM, CONTINUA NORMALMENTE
25          RTS            - NAO, RETORNA CONTROLE AO TECLADO
26 INCR     LDX COUNT
27          INX            - INCREMENTA CONTADOR DE LINHAS
28          CPX LIMITE     - NRO DE LINHAS SUFICIENTE PARA SALVAR PROGRAMA ?
29          BEQ SAVE       - SIM, CONTINUA
30          STX COUNT      - NAO,ARMAZENA NOVO VALOR DO CONTADOR
31          RTS            - RETORNA CONTROLE AO TECLADO
32 SAVE     LDX #$00       - ZERA CONTADOR DE LINHAS
33          STX COUNT
34          LDY #$15       - MONTA COMANDO "SAVE" NO BUFFER INTERNO
35 LOOP     LDA BUF,Y
36          STA $200,Y
37          DEY
38          BPL LOOP
39          LDA #$01       - SETA FLAG DE PROGRAMA SALVO
40          STA FLAG
41          JMP $9FCD      - EXECUTA COMANDO "SAVE"
```

O único problema mais sério a ser resolvido nesta rotina é como usar o comando **SAVE** do DOS 3.3 em nível de linguagem de máquina. A maneira mais viável que consegui foi montar esse comando, juntamente com o nome do arquivo (fornecido pelo usuário), no buffer do teclado, o qual começa na posição $200. Uma vez montado o comando, é chamada a rotina de identificação e execução de comandos do DOS 3.3, a qual começa na posição $9FCD.

Um inconveniente da utilização da rotina $9FCD é que, após o salvamento do programa, o cursor da tela fica adiantado de uma posição. Para corrigir tal problema, após a execução do comando **SAVE**, a nossa sub-rotina recua automaticamente o cursor de uma posição, normalizando-o.

COMO IMPLANTAR E USAR

Se você dispor de um Editor Assembler, a preparação da rotina em linguagem de máquina pode ser feita a partir da listagem 3, a qual é compatível com o Editor Assembler do DOS Tool Kit. O arquivo-objeto que será gerado deverá ter o nome de Autosaver.ASMB, pois ele será utilizado posteriormente pelo programa de instalação em BASIC.

Entretanto, caso você não disponha de um Editor Assembler, utilize o Monitor para entrar com os códigos de máquina. Para tal, você deve digitar CALL -151 e digitar os códigos da listagem 2. Após isto, dê o comando **BSAVE AUTOSAVER.ASMB,A$319,L65**.

A seguir, digite e salve o programa em BASIC da listagem 1. Ao executá-lo, primeiramente, ele lhe perguntará de quantas em quantas linhas você deseja que o programa seja automaticamente salvo. Após isto, o programa instalará e colocará em funcionamento as rotinas em linguagem de máquina.

Assim, supondo que você tenha digitado um número de linhas igual a 20 e nome de arquivo igual a PROGRAMA, a rotina irá salvar automaticamente o programa em BASIC que se encontra na memória a cada 20 linhas digitadas, num arquivo tipo "A" chamado PROGRAMA.

Há duas restrições nesses dados: o número de linhas deve ser menor ou igual a 255 e o nome do arquivo deve ter no máximo dez caracteres.

A rotina pode ser desativada a qualquer momento utilizando-se o comando **POKE 793,96** e reativada posteriormente usando **POKE 793,76**. Estes comandos POKE modificam a instrução presente na linha 2 da listagem 3, respectivamente para RTS (retorno ao programa principal) ou JMP (instrução original da sub-rotina).

Espero que esta rotina possa lhe evitar dissabores futuros por ocasião de uma falta de energia durante a digitação de programas, e que você tenha aprendido um pouquinho mais sobre o sistema operacional DOS 3.3 do Apple.

Antonio Augusto Gorni é formado em engenharia de materiais pela Universidade Federal de São Carlos. Atualmente está cursando pós-graduação em metalurgia pela Escola Politécnica da USP.