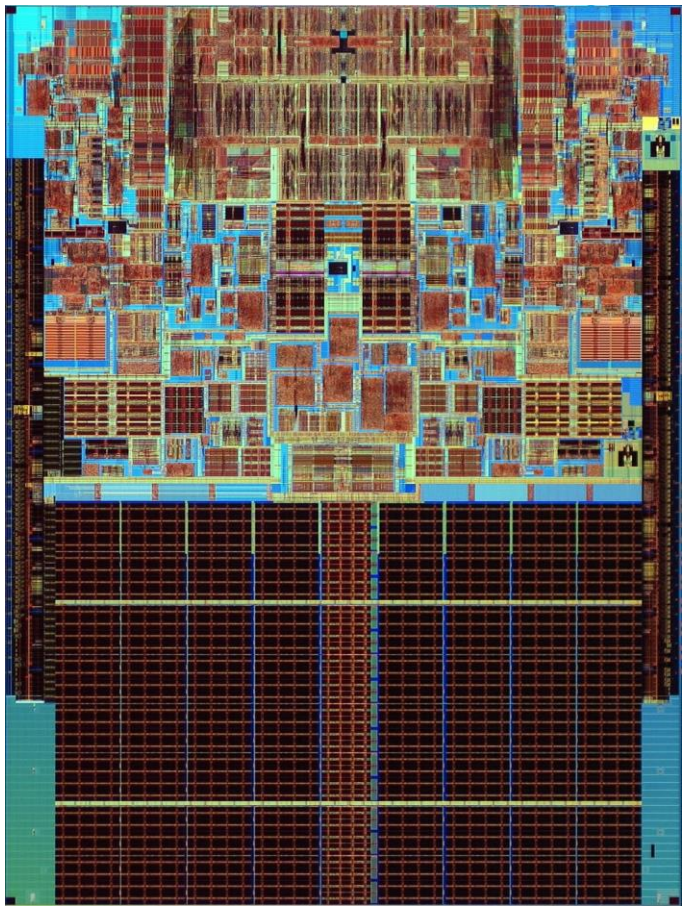
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

Microprocessadores 2: Projeto de um Jogo de Carrinho para o 8051



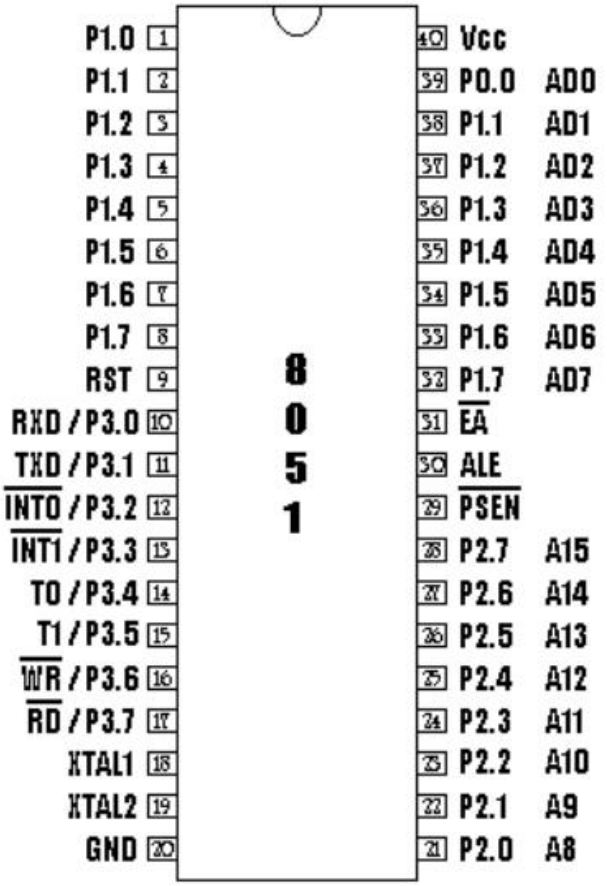
Rafael Delalibera Rodrigues – RA 207000519
Rafael Fae Tavares da Silva – RA 207000520
Bacharelado em Ciências da Computação Integral

Julho de 2007

Introdução

Este trabalho trata da implementação de um jogo de carrinho, baseado naqueles encontrados facilmente em mini-games populares. Para tanto, utilizamos um kit de desenvolvimento composto por um microcontrolador da família 8051, teclado com 16 teclas e display LCD de 2 linhas por 40 colunas.

Optamos por este projeto, pois ele utiliza muitos dos conceitos necessários ao domínio de um microcontrolador (microprocessador), tais como: Manipulação da memória (interna e externa), sincronização de tempos de resposta, configuração dos temporizadores, utilização das interrupções, controle de dispositivos externos (como LCD e Teclado) e a própria complexidade da engine (núcleo) de um jogo.



Pinagem do 8051

Display LCD

O display disponível no kit utilizado é do tipo texto (caractere) de 2 linhas por 40 colunas. Este possui uma CGRAM (Character Generator RAM) capaz de armazenar 8 caracteres definidos pelo usuário. Portanto, apesar de não ser um display gráfico, é possível criar alguns desenhos; o que possibilitou o desenvolvimento do projeto.

Display LCD de 2 Linhas por 40 Colunas

Para controlar o display é necessário conhecer os endereços de "Comandos", "Dados" e "Flag Ocupado" do controlador. Após isso é muito útil criar rotinas facilitando o acesso a essas posições. Também é necessário inicializar o display, o que é feito a partir dos comandos de controle apropriados para cada situação. O que pode ser analisado na tabela abaixo:

Command	Binary D7	D6	D5	D4	D3	D2	D1	D0	Hex
Clear Display	0	0	0	0	0	0	0	1	01
Display & Cursor Home	0	0	0	0	0	0	1	x	02 or 03
Character Entry Mode	0	0	0	0	0	1	1/D	S	04 to 07
Display On/Off & Cursor	0	0	0	0	1	D	U	B	08 to 0F
Display/Cursor Shift	0	0	0	1	D/C	R/L	x	x	10 to 1F
Function Set	0	0	1	8/4	2/1	10/7	x	x	20 to 3F
Set CGRAM Address	0	1	A	A	A	A	A	A	40 to 7F
Set Display Address	1	A	A	A	A	A	A	A	80 to FF

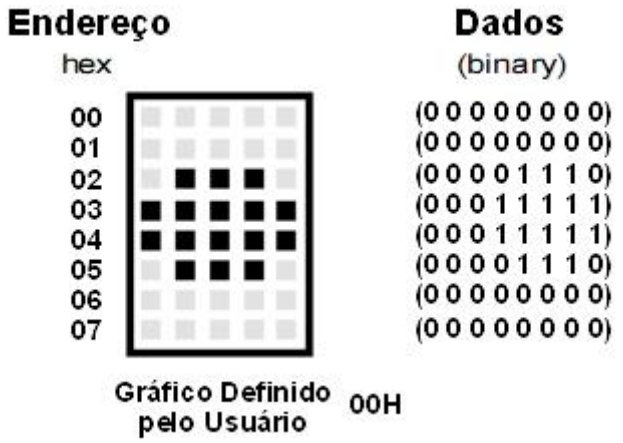
1/D:	1=Increment*, 0=Decrement	R/L:	1=Right shift, 0=Left shift
S:	1=Display shift on, 0=Off*	8/4:	1=8-bit interface*, 0=4-bit interface
D:	1=Display on, 0=Off*	2/1:	1=2 line mode, 0=1 line mode*
U:	1=Cursor underline on, 0=Off*	10/7:	1=5x10 dot format, 0=5x7 dot format*
B:	1=Cursor blink on, 0=Off*		
D/C:	1=Display shift, 0=Cursor move	x = Don't care	* = Initialization settings

Comandos de Controle de LCD

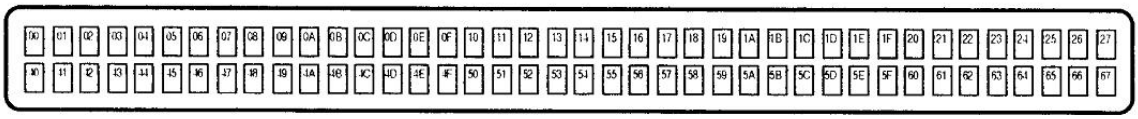
Para o nosso projeto os caracteres já disponíveis na CGROM não eram suficientes, logo foi necessário definir novos caracteres na CGRAM, que como dito anteriormente nos permite adicionar somente 8 novos caracteres. Estes novos caracteres serão endereçados nas posições de 00H até 07H ou 08H a 0F. Como é mostrado na tabela abaixo:

Tabela de Caracteres do LCD

A seguir é exemplificado o processo de escrita na CGRAM, onde cada byte é enviado para uma das 64 posições disponíveis (cada 8 bytes formam um caractere); deve se observar que somente são significativos os 5 bytes menos significativos (os outros são descartados) já que cada caractere possui 5 colunas de definição.



Abaixo está ilustrado o modo de endereçamento da tela do display, que se utiliza de uma DDRAM (Display Data RAM). Vale observar que qualquer número entre 00H e FFH endereça uma posição da tela. Logo a posição mostrada como 40H representa a mesma posição que a 28H.

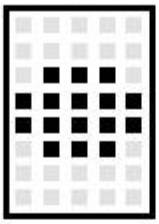


Display LCD de 2 Linhas por 40 Colunas

Os novos caracteres necessários ao nosso projeto foram criados para representar: uma pedra, um avião, um barco, um caminhão e uma pessoa. Para pedra foi utilizado somente um caractere (que também serviu como cabeça da pessoa) já para o avião, o barco e o caminhão foram necessários dois caracteres. Para a pessoa somente um novo (que representa o corpo).

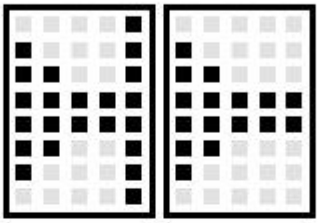
Caracteres Especiais
na CGRAM:

Pedra



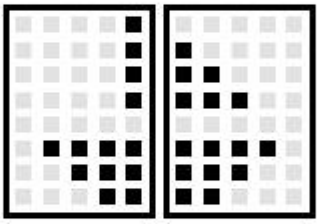
00H

Avião



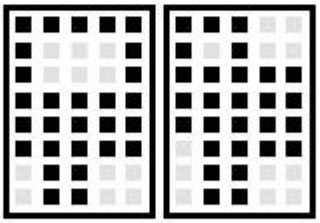
01H 02H

Barco



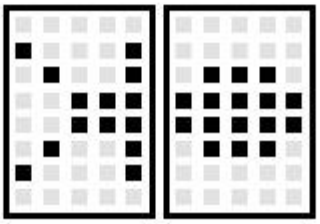
03H 04H

Caminhão



05H 06H

Homem

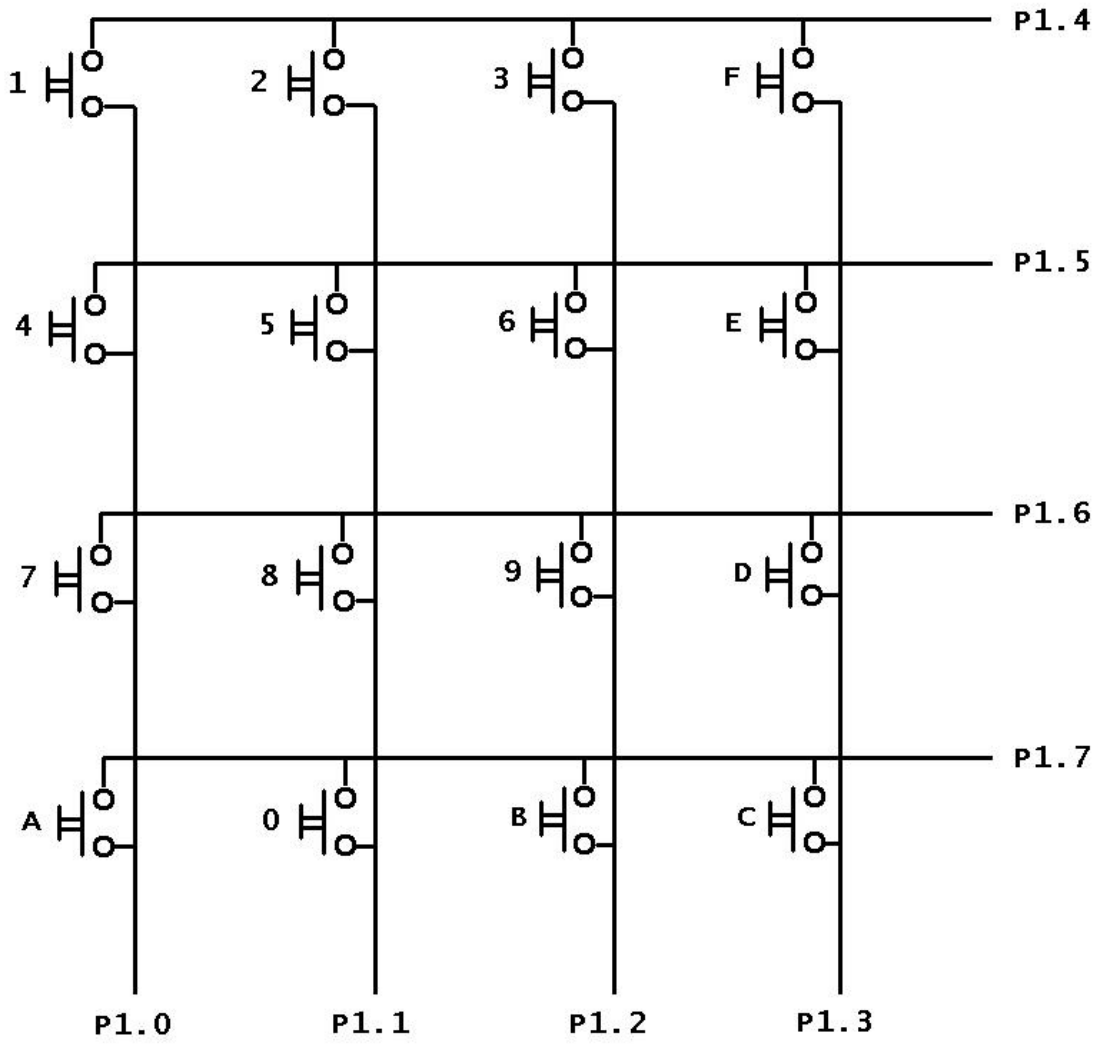


07H 00H

Teclado

Também foi necessário entender o funcionamento do teclado para que pudéssemos prosseguir com o projeto. Logo abaixo está ilustrado um diagrama simplificado do funcionamento do teclado (deve-se atentar para o fato de que o teclado tem seu funcionamento baseado em lógica reversa, o que não é explicitado no desenho devido a fatores de simplificação).

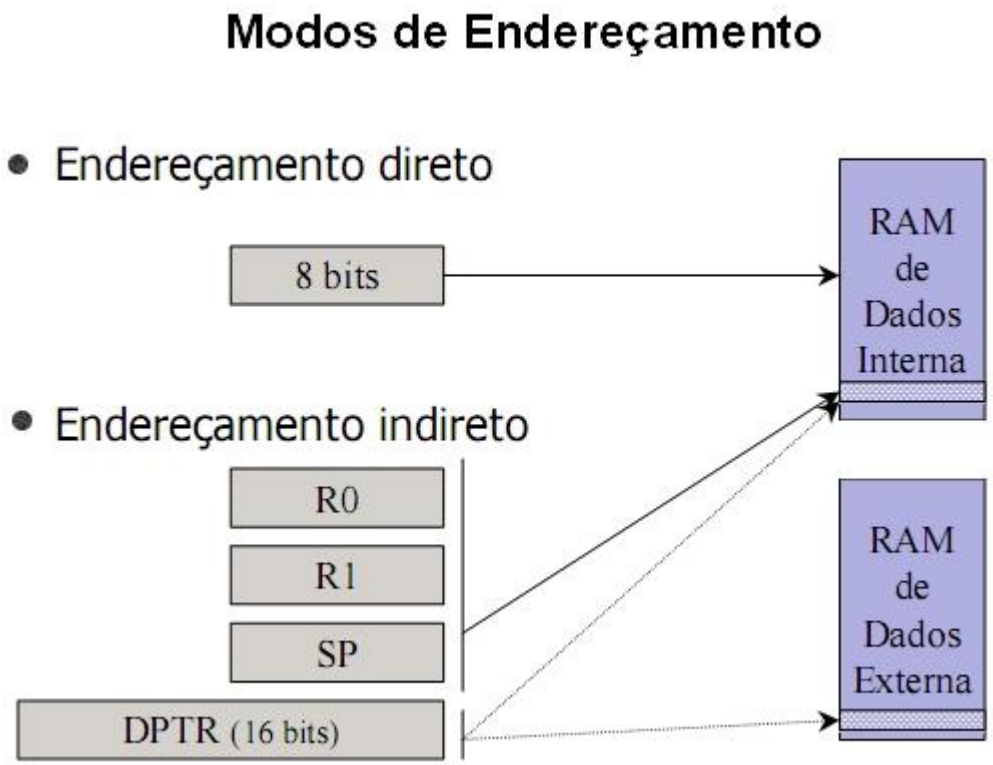
O teclado do kit encontra-se conectado ao microcontrolador através do porto 1 (P1), para acionar a leitura de suas teclas é necessário ativar o bit correspondente a linha desejada e a partir daí verificar se ocorreu alguma modificação nesse porto.



Esquema Representativo do Teclado

Organização da Memória

Devido ao modo de endereçamento do microcontrolador 8051 (que está representado abaixo de maneira simplificada), as variáveis do nosso programa estão alocadas na memória de dados interna; isso gera uma maior facilidade na programação, documentação além de otimizar o código.



Na Program Memory estão alocadas as posições do vetor de interrupção e o programa principal. O vetor de interrupções para o nosso projeto está configurado da seguinte maneira:

Endereços do Vetor de Interrupção na Memória de Programa

Timer 2 interrupt →		002B
Serial port interrupt →		0023
Timer 1 interrupt →		001B
External interrupt 1 →		0013
Timer 0 interrupt →	LJMP INTER	000B
External interrupt 0 →		0003
Reset →	LJMP BEGIN	0000

Organização das Variáveis na Memória de Dados Interna:

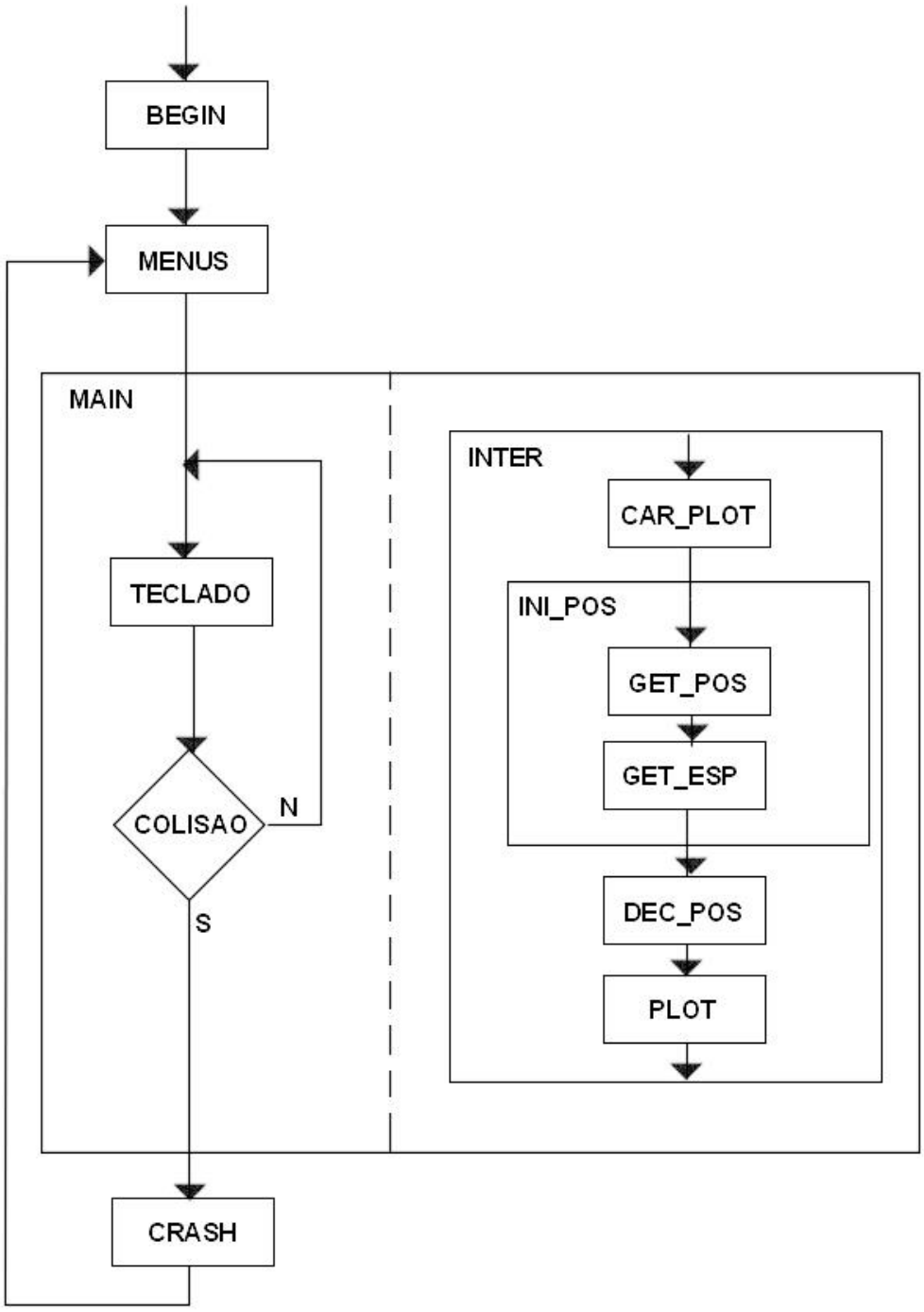
Os Primeiros 256 Bytes da Memória de Dados Interna

F8									Special function registers (SFR) (128 bytes)
F0	B								
E8									
E0	ACC								
D8									
D0	PSW								
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			
C0									
B8	IP								
B0	P3								
A8	IE								
A0	P2								
98	SCON	SBUF							
90	P1								
88	TCON	TMOD	TL0	TL1	TH0	TH1			
80	P0	SP	DPL	DPH				PCON	
78									Scratch pad area (80 bytes)
70									
68									
60									
58									
50									
48									
40									
38									
30									
28	Pode ser endereçado por 16 bytes ou 128 bits individuais.							SP	
20	Endereçamento por bit é de 00H até 7F.								
18	NIVEL	AUX_R0							Bank 3
10	POS9	CAR_T	CAR_F	CAR_POS	SCORE_0	SCORE_1	H_SCR0	H_SCR1	Bank 2
8	POS1	POS2	POS3	POS4	POS5	POS6	POS7	POS8	Bank 1
0	R0	R1				R5	R6		Bank 0

R0: Tempo de movimentação das pedras.
R1: Utilizado para endereçamento indireto (por GET_POS).
R5: Indicador utilizado para inicialização das pedras
R6: Contador do valor obtido para o espaçamento
POS1 ... POS9: Utilizadas para armazenar as posições das pedras.
CAR_T: Traseira do veículo que está sendo exibido.
CAR_F: Frente do veículo que está sendo exibido.
SCORE_1 | SCORE_0 : Pontuação obtida no jogo.
H_SCR1 | H_SCR0: Pontuação máxima já obtida.
NIVEL: Contador para a mudança de nível.
AUX_R0: Auxiliar para mudança no valor de R0.

Engine do Jogo:

É dita “engine” do jogo o conjunto das principais rotinas responsáveis por manter o correto funcionamento do jogo.



Descrição das Principais Rotinas Utilizadas Pela Engine:

CAR_PLOT (Desenhar Carrinho):

A rotina CAR_PLOT é responsável, simplesmente, por desenhar o carrinho no display. Obtendo sua posição através do tratamento do teclado (que se encontra no programa principal).

GET_POS (Obter Posição):

É responsável por gerar aleatoriamente a posição inicial de cada pedra (linha 1 ou linha 2). A aleatoriedade é baseada no valor do bit 0 da contagem do timer 1.

GET_ESP (Obter Espaçamento):

Obtém o espaçamento entre as pedras a partir da aleatoriedade gerada pelos bits 0 e 1 do Timer 1.

COLISAO (Verifica Colisões):

Verifica se o carrinho colidiu com alguma pedra desenhada. Esta rotina simplesmente compara a posição do carrinho e das pedras a cada instante.

DEC_POS (Decrementa Posições):

Responsabiliza-se pela movimentação das pedras no display. Esta movimentação ocorre a partir do decréscimo das posições das pedras ativadas, que não possuem o valor FFH (um sinalizador). Ao chegar ao final da tela a pedra recebe o valor FFH, que sinaliza a disponibilidade para reativação; impedindo que ela seja desenhada em posições indevidas.

Ela também auxilia a implementação do Nível e do Score do jogo.

INI_POS (Inicializador de Posições):

Gerencia a inicialização e reativação de todas as pedras, a partir do uso das rotinas GET_POS e GET_ESP. Trata as pedras como uma fila circular, iniciando uma nova pedra somente após a anterior ter obtido sua posição (através do GET_POS) e tendo sido respeitado o espaçamento definido por GET_ESP. Para assegurar que a fila não se esgote antes que uma pedra deva ser reutilizada, existem mais pedras do que a demanda.

```
;     UNESP - Universidade Estadual Paulista
;     Campus de Rio Claro
;
;        ###   ##             ###   ##             ###   ##
;     ## ###   ## #         ## ###   ## #         ## ###   ## #
;     ##############       ##############       ##############
;     ##############       ##############       ##############
;     ## ###   ## #         ## ###   ## #         ## ###   ## #
;        ###   ##             ###   ##             ###   ##
;
; Autores:
;           Rafael Delalibera Rodrigues - RA 207000519
;           Rafael Fae Tavares da Silva - RA 207000520
; Disciplina: Microprocessadores 2
; Proposta:   Jogo de Carrinho
; Linguagem:  Assembly do Microcontrolador 8051
;             Com Display de 2 Linhas por 40 Colunas
;             E Teclado Conectado em P1
; Data: 27 de Julho de 2007


;###############################################################################
;######################### RESET : Endereço de Partida #########################
;###############################################################################


      ORG   0000H
      LJMP  BEGIN       ; Direciona o início do programa


;###############################################################################
;################ Redirecionamento da Interrupção do Timer 0 ###################
;###############################################################################


      ORG   000Bh       ; Endereço da chamada de interrupção do Timer 0
      LJMP  INTER       ; Direciona a rotina da interrupção


;###############################################################################
;######################### Endereço de Início do Programa ######################
;###############################################################################


      ORG   0050H       ; Endereço de início do programa


;###############################################################################
;#################### Declaração de Constantes e Variáveis #####################
;###############################################################################


ALUNO1:     DB "Rafael Delalibera Rodrigues", 0             ; String Autor 1
ALUNO2:     DB "Rafael Fae Tavares da Silva", 0             ; String Autor 2
UNI1:       DB "UNESP - Universidade Estadual Paulista", 0  ; String Universidade
UNI2:       DB "Campus de Rio Claro", 0                     ; String Campus
TIT:        DB "CRAZY RACERS", 0                            ; String Título
STIT:       DB "Pressione a Tecla C para Continuar...", 0   ; String Auxiliar
PONTOS:     DB "Score:", 0                                  ; String Score
H_PONT:     DB "High Score:",0                              ; Pontuação Máxima
CARRO:      DB "Escolha o Seu Veiculo:", 0                  ; String Escolha
POS1  EQU 08D     ; Posição da Pedra 1 no Display (0-27H ; 40-67H) FF = Não Exibir
POS2  EQU 09D     ; Posição da Pedra 2 no Display (0-27H ; 40-67H) FF = Não Exibir
POS3  EQU 10D     ; Posição da Pedra 3 no Display (0-27H ; 40-67H) FF = Não Exibir
```

```
POS4  EQU 11D     ; Posição da Pedra 4 no Display (0-27H ; 40-67H) FF = Não Exibir
POS5  EQU 12D     ; Posição da Pedra 5 no Display (0-27H ; 40-67H) FF = Não Exibir
POS6  EQU 13D     ; Posição da Pedra 6 no Display (0-27H ; 40-67H) FF = Não Exibir
POS7  EQU 14D     ; Posição da Pedra 7 no Display (0-27H ; 40-67H) FF = Não Exibir
POS8  EQU 15D     ; Posição da Pedra 8 no Display (0-27H ; 40-67H) FF = Não Exibir
POS9  EQU 16D     ; Posição da Pedra 9 no Display (0-27H ; 40-67H) FF = Não Exibir
CAR_T EQU 17D     ; Traseira do Carrinho Escolhido
CAR_F EQU 18D     ; Frente do Carrinho Escolhido
CAR_POS     EQU 19D     ; Posição do Carrinho no Display
SCORE_0     EQU 20D     ; Guarda a Parte Baixa do Score Obtido
SCORE_1     EQU 21D     ; Guarda a Parte Alta do Score Obtido
H_SCR0      EQU 22D     ; Guarda Maior Pontuação Obtida (Parte Baixa)
H_SCR1      EQU 23D     ; Guarda Maior Pontuação Obtida (Parte Alta)
NIVEL       EQU 24D     ; Usadada Para Mudar de Nível
AUX_R0      EQU 25D     ; Auxiliar do Registrador 0

CONTROLE    EQU 0FF01H  ; Endereço para Controle do Display
DISPLAY     EQU 0FF11H  ; Endereço para Transmissão de Dados ao Display
STATUS      EQU 0FF09H  ; Endereço para Verificação do Status do Display


;###############################################################################
;############################## Início do Programa ##############################
;###############################################################################


BEGIN:

      MOV   IE, #00000000B    ; Zera o Vetor de Interrupções
      MOV   SP, #2FH          ; Endereço da Pilha em Relação à Memória Interna
      CLR   RS0               ; Seleciona o Banco de Registradores 0
      CLR   RS1               ; Outros Bancos São Endereçados como Variáveis
      MOV   H_SCR0, #00H
      MOV   H_SCR1, #00H


;###############################################################################
;########################### Inicialização do Display ###########################
;###############################################################################


      MOV   A, #38H           ; Comunicação com Display: 2 Linhas (5x8)/8 Bits
      LCALL LCD_CMD
      MOV   A, #00000110B     ; Modo de Deslocamento do Cursor (Esq->Dir)
      LCALL LCD_CMD
      MOV   A, #00001100B     ; Liga Display com Cursor Desligado
      LCALL LCD_CMD
      MOV   A, #00000001B     ; Limpa Display (Cursor para Posição Inicial)
      LCALL LCD_CMD
      MOV   A, #01000000B     ; Ativa o Endereçamento da CGRAM
      LCALL LCD_CMD           ; Para Possibilitar o Envio dos Caracteres

; Início do Envio dos Caractéres Especiais

; ############
; ## Pedra ##
; ############
; Endereço = 00H

      MOV   A, #00000B
      LCALL LCD_CHAR
      MOV   A, #00000B
      LCALL LCD_CHAR
      MOV   A, #01110B
```

```
      LCALL LCD_CHAR
      MOV   A, #11111B
      LCALL LCD_CHAR
      MOV   A, #11111B
      LCALL LCD_CHAR
      MOV   A, #01110B
      LCALL LCD_CHAR
      MOV   A, #00000B
      LCALL LCD_CHAR
      MOV   A, #00000B
      LCALL LCD_CHAR

; ############################
; ### Carrinho 1 Traseira ###
; ############################
; Endereço = 01H

      MOV   A, #00001B
      LCALL LCD_CHAR
      MOV   A, #10001B
      LCALL LCD_CHAR
      MOV   A, #11001B
      LCALL LCD_CHAR
      MOV   A, #11111B
      LCALL LCD_CHAR
      MOV   A, #11111B
      LCALL LCD_CHAR
      MOV   A, #11001B
      LCALL LCD_CHAR
      MOV   A, #10001B
      LCALL LCD_CHAR
      MOV   A, #00001B
      LCALL LCD_CHAR

; ############################
; #### Carrinho 1 Frente ####
; ############################
; Endereço = 02H

      MOV   A, #00000B
      LCALL LCD_CHAR
      MOV   A, #10000B
      LCALL LCD_CHAR
      MOV   A, #11000B
      LCALL LCD_CHAR
      MOV   A, #11111B
      LCALL LCD_CHAR
      MOV   A, #11111B
      LCALL LCD_CHAR
      MOV   A, #11000B
      LCALL LCD_CHAR
      MOV   A, #10000B
      LCALL LCD_CHAR
      MOV   A, #00000B
      LCALL LCD_CHAR

; ############################
; ### Carrinho 2 Traseira ###
; ############################
; Endereço = 03H

      MOV   A, #00001B
      LCALL LCD_CHAR
      MOV   A, #00001B
```

```
        LCALL LCD_CHAR
        MOV   A, #00001B
        LCALL LCD_CHAR
        MOV   A, #00001B
        LCALL LCD_CHAR
        MOV   A, #00000B
        LCALL LCD_CHAR
        MOV   A, #01111B
        LCALL LCD_CHAR
        MOV   A, #00111B
        LCALL LCD_CHAR
        MOV   A, #00011B
        LCALL LCD_CHAR

; ############################
; #### Carrinho 2 Frente ####
; ############################
; Endereço = 04H

        MOV   A, #00000B
        LCALL LCD_CHAR
        MOV   A, #10000B
        LCALL LCD_CHAR
        MOV   A, #11000B
        LCALL LCD_CHAR
        MOV   A, #11100B
        LCALL LCD_CHAR
        MOV   A, #00000B
        LCALL LCD_CHAR
        MOV   A, #11110B
        LCALL LCD_CHAR
        MOV   A, #11100B
        LCALL LCD_CHAR
        MOV   A, #11000B
        LCALL LCD_CHAR

; ############################
; ### Carrinho 3 Traseira ###
; ############################
; Endereço = 05H

        MOV   A, #11111B
        LCALL LCD_CHAR
        MOV   A, #10001B
        LCALL LCD_CHAR
        MOV   A, #10001B
        LCALL LCD_CHAR
        MOV   A, #11111B
        LCALL LCD_CHAR
        MOV   A, #11111B
        LCALL LCD_CHAR
        MOV   A, #11111B
        LCALL LCD_CHAR
        MOV   A, #01100B
        LCALL LCD_CHAR
        MOV   A, #01100B
        LCALL LCD_CHAR

; ############################
; #### Carrinho 3 Frente ####
; ############################
; Endereço = 06H

        MOV   A, #11100B
```

```
      LCALL LCD_CHAR
      MOV   A, #10100B
      LCALL LCD_CHAR
      MOV   A, #11111B
      LCALL LCD_CHAR
      MOV   A, #11111B
      LCALL LCD_CHAR
      MOV   A, #11111B
      LCALL LCD_CHAR
      MOV   A, #01111B
      LCALL LCD_CHAR
      MOV   A, #01100B
      LCALL LCD_CHAR
      MOV   A, #01100B
      LCALL LCD_CHAR

; ############################
; ########## Corpo ##########
; ############################
; Endereço = 07H

      MOV   A, #00000B
      LCALL LCD_CHAR
      MOV   A, #10001B
      LCALL LCD_CHAR
      MOV   A, #01001B
      LCALL LCD_CHAR
      MOV   A, #00111B
      LCALL LCD_CHAR
      MOV   A, #00111B
      LCALL LCD_CHAR
      MOV   A, #01001B
      LCALL LCD_CHAR
      MOV   A, #10001B
      LCALL LCD_CHAR
      MOV   A, #00000B
      LCALL LCD_CHAR

; Término do Envio dos Caracteres Especiais

      MOV   A, #00000001B     ; Limpa o Display / Cursor para Posição Inicial
      LCALL LCD_CMD           ; Motivo: Cursor Desloca-se com o Envio de
                              ; Caracteres para CGRAM
      MOV   A, #10000000B     ; Ativa o Endereçamento da DDRAM
      LCALL LCD_CMD


;#############################################################################
;################################### MENUS ###################################
;#############################################################################


; Apresentação: Universidade e Campus
      MOV   A, #0H                ; Posição no Display = 00H (Linha 1)
      MOV   DPTR, #UNI1           ; String a Ser Enviada
      LCALL LCD_POS_STRING        ; Escreve String Universidade
      MOV   A, #40H               ; Posição no Display = 40H (Linha 2)
      MOV   DPTR, #UNI2           ; String a Ser Enviada
      LCALL LCD_POS_STRING        ; Escreve String Campus
      MOV   A, #20D
      LCALL ESPERA                ; Espera 1 Segundo para Continuar
      MOV   A, #00000001B         ; Limpa a Tela
      LCALL LCD_CMD
```

```
; Apresentação: Autores
      MOV   A, #00H                 ; Posição no Display = 00H (Linha 1)
      MOV   DPTR, #ALUNO1           ; String a Ser Enviada
      LCALL LCD_POS_STRING          ; Escreve String Autor 1
      MOV   A, #40H                 ; Posição no Display = 40H (Linha 2)
      MOV   DPTR, #ALUNO2           ; String a Ser Enviada
      LCALL LCD_POS_STRING          ; Escreve String Autor 2
      MOV   A, #20D
      LCALL ESPERA                  ; Espera 1 Segundo para Continuar
      MOV   A, #00000001B           ; Limpa a Tela
      LCALL LCD_CMD

; Apresentação: Título
      MOV   A, #15D                 ; Posição no Display = 15D (Linha 1)
      MOV   DPTR, #TIT              ; String a Ser Enviada
      LCALL LCD_POS_STRING          ; Escreve String Título
      MOV   A, #40H                 ; Posição no Display = 40H (Linha 2)
      MOV   DPTR, #STIT             ; String a Ser Enviada
      LCALL LCD_POS_STRING          ; Escreve String Auxiliar

TECLA_INIT1:                        ; Espera Tecla C Ser Pressionada para
                                    ; Dar Continuidade ao Programa
      MOV   P1, #11111111B          ; Prepara o Porto 1
      CLR   P1.7                    ; Ativa a Linha 4
      MOV   A, P1                   ; Verifica o Porto 1
      XRL   A, #01110111B           ; Verifica Se a Tecla C foi Pressionada
      JNZ   TECLA_INIT1             ; Continua Se a Tecla foi Pressionada

MENU_CARRINHOS:                     ; Menu de Escolha dos Carrinhos
                                    ; Este Label Será Chamado Quando
                                    ; Houver Colisão
      MOV   SP, #2FH                ; A Pilha é Redefinida Para o Caso
                                    ; de Colisão
      MOV   A, #00000001B           ; Limpa a Tela
      LCALL LCD_CMD

      MOV   A, #0D                  ; Posição no Display = 00H (Linha 1)
      MOV   DPTR, #CARRO            ; String a Ser Enviada
      LCALL LCD_POS_STRING          ; Escreve String Escolha

      MOV   A, #64D                 ; Posição no Display = 64D (Linha 2)
      ORL   A, #10000000B           ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, #31H                 ; Escreve Caractere '1'
      LCALL LCD_CHAR
      MOV   A, #66D                 ; Posição no Display = 66D (Linha 2)
      ORL   A, #10000000B           ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, #01H                 ; Escreve Caractere Traseira Carrinho 1
      LCALL LCD_CHAR
      MOV   A, #02H                 ; Escreve Caractere Frente Carrinho 1
      LCALL LCD_CHAR

      MOV   A, #74D                 ; Posição no Display = 74D (Linha 2)
      ORL   A, #10000000B           ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, #32H                 ; Escreve Caractere '2'
      LCALL LCD_CHAR
      MOV   A, #76D                 ; Posição no Display = 76D (Linha 2)
      ORL   A, #10000000B           ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, #03H                 ; Escreve Caractere Traseira Carrinho 2
      LCALL LCD_CHAR
      MOV   A, #04H                 ; Escreve Caractere Frente Carrinho 2
```

```
      LCALL LCD_CHAR


      MOV   A, #84D                 ; Posição no Display = 84D (Linha 2)
      ORL   A, #10000000B           ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, #33H                 ; Escreve Caractere '3'
      LCALL LCD_CHAR
      MOV   A, #86D                 ; Posição no Display = 86D (Linha 2)
      ORL   A, #10000000B           ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, #05H                 ; Escreve Caractere Traseira Carrinho 3
      LCALL LCD_CHAR
      MOV   A, #06H                 ; Escreve Caractere Frente Carrinho 3
      LCALL LCD_CHAR


      MOV   A, #94D                 ; Posição no Display = 94D (Linha 2)
      ORL   A, #10000000B           ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, #46H                 ; Escreve Caractere 'F'
      LCALL LCD_CHAR
      MOV   A, #96D                 ; Posição no Display = 96D (Linha 2)
      ORL   A, #10000000B           ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, #07H                 ; Escreve Caractere Corpo
      LCALL LCD_CHAR
      MOV   A, #00H                 ; Escreve Caractere Pedra (Cabeça)
      LCALL LCD_CHAR


; Recebe do Teclado o Valor Correspondente ao Carrinho Escolhido
CARRINHO1:
      MOV   P1, #11111111B          ; Prepara o Porto 1
      CLR   P1.4                    ; Ativa a Linha 1
      MOV   A, P1                   ; Verifica o Porto 1
      XRL   A, #11101110B           ; Verifica Se a Tecla 1 foi Pressionada
      JNZ   CARRINHO2               ; Caso Não: Verifica a Próxima Tecla
      MOV   CAR_T, #01H             ; Caso Sim: Seta a Traseira do Carrinho
      MOV   CAR_F, #02H             ;           Seta a Frente do Carrinho
      LJMP  OK_CARRINHO             ;           Sai
CARRINHO2:
      MOV   P1, #11111111B          ; Prepara o Porto 1
      CLR   P1.4                    ; Ativa a Linha 1
      MOV   A, P1                   ; Verifica o Porto 1
      XRL   A, #11101101B           ; Verifica Se a Tecla 2 foi Pressionada
      JNZ   CARRINHO3               ; Caso Não: Verifica a Próxima Tecla
      MOV   CAR_T, #03H             ; Caso Sim: Seta a Traseira do Carrinho
      MOV   CAR_F, #04H             ;           Seta a Frente do Carrinho
      LJMP  OK_CARRINHO             ;           Sai
CARRINHO3:
      MOV   P1, #11111111B          ; Prepara Porto 1
      CLR   P1.4                    ; Ativa a Linha 1
      MOV   A, P1                   ; Verifica o Porto 1
      XRL   A, #11101011B           ; Verifica Se a Tecla 3 foi Pressionada
      JNZ   CARRINHO4               ; Caso Não: Verifica a Próxima Tecla
      MOV   CAR_T, #05H             ; Caso Sim: Seta a Traseira do Carrinho
      MOV   CAR_F, #06H             ;           Seta a Frente do Carrinho
      LJMP  OK_CARRINHO             ;           Sai
CARRINHO4:
      MOV   P1, #11111111B          ; Prepara Porto 1
      CLR   P1.4                    ; Ativa a Linha 1
      MOV   A, P1                   ; Verifica o Porto 1
```

```
      XRL   A, #11100111B          ; Verifica Se a Tecla F foi Pressionada
      JNZ   CARRINHO1              ; Caso Não: Reinicia Verificação
      MOV   CAR_T, #07H            ; Caso Sim: Seta a Traseira do Carrinho
      MOV   CAR_F, #00H            ;           Seta a Frente do Carrinho
      LJMP  OK_CARRINHO            ;           Sai
OK_CARRINHO:                       ; O Carrinho Já Foi Escolhido



;################################################################################
;######################### Inicialização das Variáveis ##########################
;################################################################################



      MOV   POS1, #FFH             ; Pedra 1 Não Inicializada (Em Espera)
      MOV   POS2, #FFH             ; Pedra 2 Não Inicializada (Em Espera)
      MOV   POS3, #FFH             ; Pedra 3 Não Inicializada (Em Espera)
      MOV   POS4, #FFH             ; Pedra 4 Não Inicializada (Em Espera)
      MOV   POS5, #FFH             ; Pedra 5 Não Inicializada (Em Espera)
      MOV   POS6, #FFH             ; Pedra 6 Não Inicializada (Em Espera)
      MOV   POS7, #FFH             ; Pedra 7 Não Inicializada (Em Espera)
      MOV   POS8, #FFH             ; Pedra 8 Não Inicializada (Em Espera)
      MOV   POS9, #FFH             ; Pedra 9 Não Inicializada (Em Espera)
      MOV   CAR_POS, #01H          ; Posição Inicial do Carrinho = Linha 1
      MOV   SCORE_0, #00H          ; Score Inicialmente em 0
      MOV   SCORE_1, #00H          ; Score Inicialmente em 0
      MOV   NIVEL, #00H            ; Contador: Deve Iniciar Zerado
      MOV   AUX_R0, #7D            ; Deve Conter o Mesmo Valor de R0



;################################################################################
;######################### Inicialização dos Timers / INT #######################
;################################################################################



; Timer 0 é Responsável Pelo Tempo de Atualização da Tela
; R0 é Responsável pelo Tempo de Mudança da Posição das Pedras
      MOV   TMOD, #01H         ; 0000 : Timer 1 : 0001 Timer 0
                               ; GATE0 = 0 => Interrupção por Software
                               ;  C/T0 = 0 => Temporizador
                               ;  M1.0 = 0 M0.0 = 1 => Modo 1 (16 bits)
                               ;  Sem Auto-Recarga
                               ;  Timer 0 Configurado
      MOV   R0, #7D
      MOV   R5, #0D            ; Indicador Iniciado em 00000000 (POS1)
                               ; R5 Indica a Pedra que Deve Ser Inicializada
                               ; POS1 = 00000000B  POS2 = 00000001B
                               ; POS3 = 00000010B  POS4 = 00000011B
                               ; POS5 = 00000100B  POS6 = 00000101B
                               ; POS7 = 00000110B  POS8 = 00000111B
                               ; POS9 = 00001000B
      MOV   R6, #FFH           ; Espaço Ainda Não Obtido
                               ; R6 Representa o Espaço Entre Uma Pedra e Outra


; Timer 1 é Responsável por Gerar a Aleatoriedade para Alguns Eventos do Jogo
; Como: A Linha em que a Próxima Pedra Aparecerá
;     E a Distância entre uma Pedra e Outra
```

```
      MOV   A, TMOD
      ANL   A, #00001111B      ; Mantem Valores do Timer 0 Inalterados
      ORL   A, #00100000B      ; GATE1 = 0 => Interrupção por Software
                               ;   C/T = 0 => Temporizador
                               ;  M1.1 = 1 M0.1 = 0 => Modo 2 (8 bits)
                               ;  Com Auto-Recarga
      MOV   TMOD, A            ; Timer 1 Configurado



; Carregando e Inicializando os Timers
      CLR   TR0                ; Desativa Timer 0
      CLR   TF0                ; Reseta Flag de Overflow
      MOV   TL0, #00H          ; Carrega Parte Baixa do Timer 0
      MOV   TH0, #80H          ; Carrega Parte Alta do Timer 0
      SETB  TR0                ; Habilita a Contagem
      CLR   TR1                ; Desativa Timer 1
      CLR   TF1                ; Reseta Flag de Overflow
      MOV   TH1, 0H            ; Carrega Timer 1
      SETB  TR1                ; Habilita a Contagem


;###############################################################################
;######################### Início do Programa Principal ########################
;###############################################################################



      MOV   IE, #10000010B     ; Habilita a Interrupção do Timer0

MAIN:                          ; Este é o Núcleo do Jogo
                               ; Nesta Seção de Código que Serão Chamadas
                               ; a Interrupção e Funções Referentes ao Jogo



TECLA9:
      MOV   P1, #11111111B     ; Prepara o Porto 1
      CLR   P1.6               ; Ativa a Linha 3
      MOV   A, P1              ; Verifica o Porto 1
      MOV   B, A               ; Salva o Valor Obtido em B
      XRL   A, #10111011B      ; Verifica Se a Tecla 9 foi Pressionada
      JNZ   TECLA9_8           ; Caso Não: Verifica Próxima Tecla
      MOV   R0, #01D           ; Caso Sim: Acelera o Carrinho
                               ;           Diminuindo o Tempo Entre as Pedras
  TECLA9_8:
      MOV   A, B               ; Resgata o Valor Guardado
      XRL   A, #10111001B      ; Verifica Se as Teclas 9 e 8 estão Pressionadas
      JNZ   TECLA8             ; Caso Não: Verifica Próxima Tecla
      MOV   R0, #01D           ; Caso Sim: Acelera o Carrinho
      MOV   CAR_POS, #01H      ;           E Sobe o Carrinho (Linha 1)
TECLA8:
      MOV   P1, #11111111B     ; Prepara o Porto 1
      CLR   P1.6               ; Ativa a Linha 3
      MOV   A, P1              ; Verifica o Porto 1
      XRL   A, #10111101B      ; Verifica Se a Tecla 8 foi Pressionada
      JNZ   TECLA0             ; Caso Não: Verifica Próxima Tecla
      MOV   CAR_POS, #01H      ; Caso Sim: Sobe o Carrinho (Linha 1)
TECLA0:
      MOV   P1, #11111111B     ; Prepara o Porto 1
      CLR   P1.7               ; Ativa a Linha 4
      MOV   A, P1              ; Verifica o Porto 1
```

```
      XRL   A, #01111101B     ; Verifica Se a Tecla 0 foi Pressionada
      JNZ   TECLAX            ; Caso Não: Não Altera Posição do Carrinho
      MOV   CAR_POS, #41H     ; Caso Sim: Desce o Carrinho (Linha 2)
  TECLAX:

      LCALL COLISAO           ; Verifica Se Houve Colisão

      LJMP  MAIN              ; Mantem o Jogo em Funcionamento


;###############################################################################
;########################### ROTINAS E SUBROTINAS ###############################
;###############################################################################


; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$
; $$$$$$$$$$$ DISPLAY $$$$$$$$$$
; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$


; Envia Comando de Controle para o Display
; O Comando de Controle Fica em 'A'
LCD_CMD:
      PUSH  DPL
      PUSH  DPH
      MOV   DPTR, #CONTROLE         ; Endereça Posição de Controle
      MOVX  @DPTR, A                ; Envia Dado de Controle
      MOV   DPTR, #STATUS           ; Endereça Posição de Status
  _LCD_CMD_WAIT:                    ; Inicia Ciclo de Verificação
      MOVX  A, @DPTR                ; Envia Flags do Display p/ Acumulador
      JB    ACC.7, _LCD_CMD_WAIT    ; Verifica Flag Busy : Segue = 0 Not Busy
      POP   DPH
      POP   DPL
      RET


; Envia um Caractere para o Display
; O Valor do Caractere Fica em 'A' (ASCII)
LCD_CHAR:
      PUSH  DPL
      PUSH  DPH
      MOV   DPTR, #DISPLAY          ; Endereça Posição de Dados
      MOVX  @DPTR, A                ; Envia Dado (Char em ASCII)
      MOV   DPTR, #STATUS
  _LCD_CHAR_WAIT:                   ; Verifica Status do Display
      MOVX  A, @DPTR
      JB    ACC.7, _LCD_CHAR_WAIT
      POP   DPH
      POP   DPL
      RET


; Envia String ao Display
; Em "A" Deve Estar a Posição de Início
; Em DPTR Deve Estar Endereçado Vetor da String
LCD_POS_STRING:
      SETB  ACC.7                   ; Seta o Bit 7 / Para Endereçar a DDRAM
      LCALL LCD_CMD
LCD_STRING:
      CLR   A                       ; Limpa 'A'
      MOVC  A, @A+DPTR              ; Endereça a String / 'A' é o Ponteiro
```

```
      JZ    _LCD_STRING_1
      LCALL LCD_CHAR
      INC   DPTR
      SJMP  LCD_STRING
  _LCD_STRING_1:
      RET




; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
; $$$$$$$ ENGINE DO JOGO $$$$$$$$
; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


; Rotina Chamada Pela Interrupção do Timer 0
INTER:
      PUSH  A                 ; Devido ao Telado / Colisão
      PUSH  PSW               ; Devido ao Teclado / Colisão
      PUSH  B                 ; Devido ao Teclado / Colisão
      CLR   TR0               ; Desliga o Timer 0
      CLR   TF0               ; Zera Flag de Overflow
      MOV   TL0, #00H         ; Carrega Parte Baixa do Timer
      MOV   TH0, #80H         ; Carrega Parte Alta do Timer
      MOV   A, #00000001B     ; Limpa Tela
      LCALL LCD_CMD

      LCALL CAR_PLOT          ; Desenha o Carrinho

      DJNZ  R0, PART1         ; Decrementa e Sai da Interrupção se R0<>0
      SJMP  PART2             ; PART 1 e PART 2 Necessários Devido
  PART1:                      ; Ao Estouro do Laço de DJNZ para PLOT
      LJMP  PLOT
  PART2:
      MOV   R0, AUX_R0        ; Restabelece o Valor de R0

      LCALL INI_POS           ; Rotina de Inicialização das Pedras
      LCALL DEC_POS           ; Rotina de Cálculo de Posição das Pedras

  PLOT:
  ; Este trecho de código é responsável por
  ; desenhar as pedras já inicializadas na tela
      MOV   A, POS1           ; Obtém Posição da Pedra 1
      XRL   A, #FFH           ; Verifica Se a Pedra 1 Já Foi Inicializada
      JZ    PLOTP2            ; Se Não: Pula Para Pedra 2
      MOV   A, POS1           ; Se Sim: Desenha
      ORL   A, #10000000B     ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, #00H           ; Escreve Caractere Pedra
      LCALL LCD_CHAR
    PLOTP2:
      MOV   A, POS2           ; Obtém Posição da Pedra 2
      XRL   A, #FFH           ; Verifica Se a Pedra 2 Já Foi Inicializada
      JZ    PLOTP3            ; Se Não: Pula Para Pedra 3
      MOV   A, POS2           ; Se Sim: Desenha
      ORL   A, #10000000B     ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, #00H           ; Escreve Caractere Pedra
      LCALL LCD_CHAR
    PLOTP3:
      MOV   A, POS3           ; Obtém Posição da Pedra 3
      XRL   A, #FFH           ; Verifica Se a Pedra 3 Já Foi Inicializada
      JZ    PLOTP4            ; Se Não: Pula Para Pedra 4
```

```
  MOV   A, POS3           ; Se Sim: Desenha
  ORL   A, #10000000B     ; Endereça a Posição na DDRAM
  LCALL LCD_CMD
  MOV   A, #00H           ; Escreve Caractere Pedra
  LCALL LCD_CHAR
PLOTP4:
  MOV   A, POS4           ; Obtém Posição da Pedra 4
  XRL   A, #FFH           ; Verifica Se a Pedra 4 Já Foi Inicializada
  JZ    PLOTP5            ; Se Não: Pula Para Pedra 5
  MOV   A, POS4           ; Se Sim: Desenha
  ORL   A, #10000000B     ; Endereça a Posição na DDRAM
  LCALL LCD_CMD
  MOV   A, #00H           ; Escreve Caractere Pedra
  LCALL LCD_CHAR
PLOTP5:
  MOV   A, POS5           ; Obtém Posição da Pedra 5
  XRL   A, #FFH           ; Verifica Se a Pedra 5 Já Foi Inicializada
  JZ    PLOTP6            ; Se Não: Pula Para Pedra 6
  MOV   A, POS5           ; Se Sim: Desenha
  ORL   A, #10000000B     ; Endereça a Posição na DDRAM
  LCALL LCD_CMD
  MOV   A, #00H           ; Escreve Caractere Pedra
  LCALL LCD_CHAR
PLOTP6:
  MOV   A, POS6           ; Obtém Posição da Pedra 6
  XRL   A, #FFH           ; Verifica Se a Pedra 6 Já Foi Inicializada
  JZ    PLOTP7            ; Se Não: Pula Para Pedra 7
  MOV   A, POS6           ; Se Sim: Desenha
  ORL   A, #10000000B     ; Endereça a Posição na DDRAM
  LCALL LCD_CMD
  MOV   A, #00H           ; Escreve Caractere Pedra
  LCALL LCD_CHAR
PLOTP7:
  MOV   A, POS7           ; Obtém Posição da Pedra 7
  XRL   A, #FFH           ; Verifica Se a Pedra 7 Já Foi Inicializada
  JZ    PLOTP8            ; Se Não: Pula Para Pedra 8
  MOV   A, POS7           ; Se Sim: Desenha
  ORL   A, #10000000B     ; Endereça a Posição na DDRAM
  LCALL LCD_CMD
  MOV   A, #00H           ; Escreve Caractere Pedra
  LCALL LCD_CHAR
PLOTP8:
  MOV   A, POS8           ; Obtém Posição da Pedra 8
  XRL   A, #FFH           ; Verifica Se a Pedra 8 Já Foi Inicializada
  JZ    PLOTP9            ; Se Não: Pula Para Pedra 9
  MOV   A, POS8           ; Se Sim: Desenha
  ORL   A, #10000000B     ; Endereça a Posição na DDRAM
  LCALL LCD_CMD
  MOV   A, #00H           ; Escreve Caractere Pedra
  LCALL LCD_CHAR
PLOTP9:
  MOV   A, POS9           ; Obtém Posição da Pedra 9
  XRL   A, #FFH           ; Verifica Se a Pedra 9 Já Foi Inicializada
  JZ    PLOTX             ; Se Não: Pula Para Fora da Interrupção
  MOV   A, POS9           ; Se Sim: Desenha
  ORL   A, #10000000B     ; Endereça a Posição na DDRAM
  LCALL LCD_CMD
  MOV   A, #00H           ; Escreve Caractere Pedra
  LCALL LCD_CHAR
PLOTX:

  MOV   A, NIVEL
  XRL   A, #50D           ; Verifica Se Deve Mudar de Nível
  JNZ   OUT_NIVEL         ; Se Não: Sai da Interrupção
```

```
      MOV   NIVEL, #0H        ; Se Sim: Zera o Contador
      MOV   A, R0
      XRL   A, #1D            ;         Verifica Se R0 Atingiu o Mínimo
      JZ    OUT_NIVEL         ;         Caso Sim: Sai da Interrupção
      DEC   R0                ;         Caso Não: Decrementa R0
      MOV   A, #5D
      LCALL ESPERA            ;                   Atraso
      MOV   AUX_R0, R0        ;                   Guarda R0 em AUX_R0
  OUT_NIVEL:
      POP   B
      POP   PSW
      POP   A
      SETB  TR0               ; Reinicia a Contagem
      RETI
; Término da Rotina de Tratamento da Interrupção do Timer 0


; Plota o Carrinho Escolhido na Tela
CAR_PLOT:
      MOV   A, CAR_POS        ; Posição do Carrinho no Display
      ORL   A, #10000000B     ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, CAR_F          ; Escreve Caractere Frente do Carrinho
      LCALL LCD_CHAR
      MOV   A, CAR_POS        ; Recupera Posição do Carrinho
      DEC   A                 ; Decrementa em 1 Unidade
      ORL   A, #10000000B     ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, CAR_T          ; Escreve Caractere Traseira do Carrinho
      LCALL LCD_CHAR
      RET


; Função Que Obtém Aleatóriamente a Linha da Nova Pedra
; EM R1 Deve Estar Qual A Pedra (POS1, ... , POS9)
GET_POS:
      CLR   TR1               ; Pausa o Timer 1
      MOV   A, TL1            ; Obtêm o Valor da Contagem
      ANL   A, #00000001B     ; Despreza os Bits 7 ao 1
      JNZ   SET_POS1          ; Se o Bit0 = 0
      MOV   A, #28H           ; 'A' Recebe o Valor 28H (Linha 1)
      SJMP  OUT_POS           ; E Sai da Função
  SET_POS1:
      MOV   A, #68H           ; Se Bit0 = 1
                              ; 'A' Recebe o Valor 68H (Linha 2)
  OUT_POS:
      MOV   @R1, A            ; Passa o Valor de 'A' para POSX
      SETB  TR1               ; Reabilita a Contagem
      RET


; Função Que Obtém Aleatóriamente o Tamanho do Espaço Entre Uma Pedra e Outra
; O Resultado é Passado em R6
GET_ESP:
      CLR   TR1               ; Pausa o Timer 1
      MOV   A, TL1            ; Obtêm o Valor da Contagem
      ANL   A, #00000011B     ; Despreza os Bits 7 ao 2
      JZ    CASE0             ; Se 'A' = 00 Pula Para Case0
      MOV   B, A              ; Salva o Valor Obtido em B
      XRL   A, #00000001B
      JZ    CASE1             ; Se 'A' = 01 Pula Para Case1
      MOV   A, B              ; Recupera Valor
      XRL   A, #00000010B
      JZ    CASE2             ; Se 'A' = 10 Pula Para Case3
```

```
  CASE3:                        ; Caso Contrário: 'A' = 11
      MOV   R6, #8D             ; Então R6 Recebe 8
      LJMP  OUT_ESP             ; E Sai da Função
  CASE0:
      MOV   R6, #5D             ; R6 Recebe 5
      LJMP  OUT_ESP             ; Sai da Função
  CASE1:
      MOV   R6, #6D             ; R6 Recebe 6
      LJMP  OUT_ESP             ; Sai da Função
  CASE2:
      MOV   R6, #7D             ; R6 Recebe 7
  OUT_ESP:
      SETB  TR1                 ; Reabilita a Contagem
      RET


; Rotina Que Calcula Se Houve Colisão Entre o Carrinho e Alguma das Pedra
; Que Já Foram Inicializadas; Em Caso Positivo o Jogo é Interrompido
COLISAO:
      MOV   B, CAR_POS          ; B Armazena a Posição do Carrinho

      MOV   A, B                ; Verifica Se A Frente do Carrinho
      XRL   A, POS1             ; Colidiu Com a Pedra 1
      JZ    IGUAL               ; Se Sim: Pula Para Igual
      MOV   A, B
      DEC   A                   ; Verifica Se A Traseira do Carrinho
      XRL   A, POS1             ; Colidiu Com a Pedra 1
      JZ    IGUAL               ; Se Sim: Pula Para Igual

      MOV   A, B                ; Verifica Se a Frente do Carrinho
      XRL   A, POS2             ; Colidiu Com a Pedra 2
      JZ    IGUAL               ; Se Sim: Pula Para Igual
      MOV   A, B
      DEC   A                   ; Verifica Se a Traseira do Carrinho
      XRL   A, POS2             ; Colidiu Com a Pedra 2
      JZ    IGUAL               ; Se Sim: Pula Para Igual

      MOV   A, B                ; Verifica Se a Frente do Carrinho
      XRL   A, POS3             ; Colidiu Com a Pedra 3
      JZ    IGUAL               ; Se Sim: Pula Para Igual
      MOV   A, B
      DEC   A                   ; Verifica Se A Traseira do Carrinho
      XRL   A, POS3             ; Colidiu Com a Pedra 3
      JZ    IGUAL               ; Se Sim: Pula Para Igual

      MOV   A, B                ; Verifica Se a Frente do Carrinho
      XRL   A, POS4             ; Colidiu Com a Pedra 4
      JZ    IGUAL               ; Se Sim: Pula Para Igual
      MOV   A, B
      DEC   A                   ; Verifica Se A Traseira do Carrinho
      XRL   A, POS4             ; Colidiu Com a Pedra 4
      JZ    IGUAL               ; Se Sim: Pula Para Igual

      MOV   A, B                ; Verifica Se a Frente do Carrinho
      XRL   A, POS5             ; Colidiu Com a Pedra 5
      JZ    IGUAL               ; Se Sim: Pula Para Igual
      MOV   A, B
      DEC   A                   ; Verifica Se A Traseira do Carrinho
      XRL   A, POS5             ; Colidiu Com a Pedra 5
      JZ    IGUAL               ; Se Sim: Pula Para Igual

      MOV   A, B                ; Verifica Se a Frente do Carrinho
      XRL   A, POS6             ; Colidiu Com a Pedra 6
      JZ    IGUAL               ; Se Sim: Pula Para Igual
```

```
      MOV   A, B
      DEC   A                 ; Verifica Se A Traseira do Carrinho
      XRL   A, POS6           ; Colidiu Com a Pedra 6
      JZ    IGUAL             ; Se Sim: Pula Para Igual

      MOV   A, B              ; Verifica Se a Frente do Carrinho
      XRL   A, POS7           ; Colidiu Com a Pedra 7
      JZ    IGUAL             ; Se Sim: Pula Para Igual
      MOV   A, B
      DEC   A                 ; Verifica Se A Traseira do Carrinho
      XRL   A, POS7           ; Colidiu Com a Pedra 7
      JZ    IGUAL             ; Se Sim: Pula Para Igual

      MOV   A, B              ; Verifica Se a Frente do Carrinho
      XRL   A, POS8           ; Colidiu Com a Pedra 8
      JZ    IGUAL             ; Se Sim: Pula Para Igual
      MOV   A, B
      DEC   A                 ; Verifica Se A Traseira do Carrinho
      XRL   A, POS8           ; Colidiu Com a Pedra 8
      JZ    IGUAL             ; Se Sim: Pula Para Igual

      MOV   A, B              ; Verifica Se a Frente do Carrinho
      XRL   A, POS9           ; Colidiu Com a Pedra 9
      JZ    IGUAL             ; Se Sim: Pula Para Igual
      MOV   A, B
      DEC   A                 ; Verifica Se A Traseira do Carrinho
      XRL   A, POS9           ; Colidiu Com a Pedra 9
      JZ    IGUAL             ; Se Sim: Pula Para Igual

      SJMP  OUT_COL           ; Não Colidiu => Retorna ao Programa

  IGUAL:
      LJMP  CRASH             ; Chama Rotina Que Desenha a Colisão

  OUT_COL:                    ; Retorna ao Programa
      RET


; Função Que Calcula as Posições das Pedras Que Já Foram Inicializadas
; Decrementando Suas Posições
DEC_POS:
      MOV   B, POS1           ; B Armazena a Pedra 1
      MOV   A, B
      XRL   A, #FFH           ; Verifica Se a Pedra 1 Foi Inicializada
      JZ    PEDRA2            ; Se Não: Pula Para Pedra 2
      MOV   A, B
                              ; Verifica Se Pedra 1 Chegou Ao Fim do Display
      JZ    RELOAD_P1         ; Se POS1 = 00H => Reinicializa Pedra 1
      XRL   A, #40H           ; Se POS1 = 40H => Reinicializa Pedra 1
      JNZ   DEC_P1            ; Se Não: Decrementa Posição
  RELOAD_P1:
      MOV   POS1, #FFH        ; POS1 Recebe Valor de Espera
      INC   NIVEL             ; Incrementa Contador Para Mudança de Nível
      INC   SCORE_0           ; Incrementa o Score Usando BCD 4bits
      MOV   A, SCORE_0
      XRL   A, #10101010B     ; Verifica Estouro de Ambas Partes
      JZ    SC_1              ; Se Sim: Pula Para SC_1
      MOV   A, SCORE_0        ; Recupera o Valor (Alterado por XRL)
      ANL   A, #00001111B     ; Descarta Parte Mais Alta
      XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
      JNZ   PEDRA2            ; Se Não: Pula Para Próxima Pedra
      MOV   A, SCORE_0        ; Se Sim: Recupera Score (Alterado por ANL)
      ADD   A, #00010000B     ;         Incrementa Parte Alta
      ANL   A, #11110000B     ;         Zera Parte Baixa
```

```
    MOV   SCORE_0, A
    SJMP  PEDRA2
SC_1:                       ; Caso SCORE_0 Tenha Estourado
    MOV   SCORE_0, #00H     ; Zera SCORE_0
    INC   SCORE_1           ; Incrementa Score 1
    MOV   A, SCORE_1
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA2            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_1        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_1, A
    LJMP  PEDRA2            ; Pula Para Pedra 2
DEC_P1:
    DEC   POS1              ; Decrementa POS1

PEDRA2:
    MOV   B, POS2           ; B Armazena a Pedra 2
    MOV   A, B
    XRL   A, #FFH           ; Verifica Se a Pedra 2 Foi Inicializada
    JZ    PEDRA3            ; Se Não: Pula Para Pedra 3
    MOV   A, B
                            ; Verifica Se Pedra 2 Chegou Ao Fim do Display
    JZ    RELOAD_P2         ; Se POS2 = 00H => Reinicializa Pedra 2
    XRL   A, #40H           ; Se POS2 = 40H => Reinicializa Pedra 2
    JNZ   DEC_P2            ; Se Não: Decrementa Posição
RELOAD_P2:
    MOV   POS2, #FFH        ; POS2 Recebe Valor de Espera
    INC   NIVEL             ; Incrementa Contador Para Mudança de Nível
    INC   SCORE_0           ; Incrementa o Score Usando BCD 4bits
    MOV   A, SCORE_0
    XRL   A, #10101010B     ; Verifica Estouro de Ambas Partes
    JZ    SC_2              ; Se Sim: Pula Para SC_1
    MOV   A, SCORE_0        ; Recupera o Valor (Alterado por XRL)
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA3            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_0        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_0, A
    SJMP  PEDRA3
SC_2:                       ; Caso SCORE_0 Tenha Estourado
    MOV   SCORE_0, #00H     ; Zera SCORE_0
    INC   SCORE_1           ; Incrementa Score 1
    MOV   A, SCORE_1
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA3            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_1        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_1, A
    LJMP  PEDRA3            ; Pula Para Pedra 3
DEC_P2:
    DEC   POS2              ; Decrementa POS2

PEDRA3:
    MOV   B, POS3           ; B Armazena a Pedra 3
    MOV   A, B
    XRL   A, #FFH           ; Verifica Se a Pedra 3 Foi Inicializada
    JZ    PEDRA4            ; Se Não: Pula Para Pedra 4
    MOV   A, B
```

```
                            ; Verifica Se Pedra 3 Chegou Ao Fim do Display
    JZ    RELOAD_P3         ; POS3 = 00H => Reinicializa Pedra 3
    XRL   A, #40H           ; POS3 = 40H => Reinicializa Pedra 3
    JNZ   DEC_P3            ; Se Não: Decrementa Posição
RELOAD_P3:
    MOV   POS3, #FFH        ; POS3 Recebe Valor de Espera
    INC   NIVEL             ; Incrementa Contador Para Mudança de Nível
    INC   SCORE_0           ; Incrementa o Score Usando BCD 4bits
    MOV   A, SCORE_0
    XRL   A, #10101010B     ; Verifica Estouro de Ambas Partes
    JZ    SC_3              ; Se Sim: Pula Para SC_1
    MOV   A, SCORE_0        ; Recupera o Valor (Alterado por XRL)
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA4            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_0        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_0, A
    SJMP  PEDRA4
SC_3:                       ; Caso SCORE_0 Tenha Estourado
    MOV   SCORE_0, #00H     ; Zera SCORE_0
    INC   SCORE_1           ; Incrementa Score 1
    MOV   A, SCORE_1
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA4            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_1        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_1, A
    LJMP  PEDRA4            ; Pula Para Pedra 4
DEC_P3:
    DEC   POS3              ; Decrementa POS3

PEDRA4:
    MOV   B, POS4           ; B Armazena a Pedra 4
    MOV   A, B
    XRL   A, #FFH           ; Verifica Se a Pedra 4 Foi Inicializada
    JZ    PEDRA5            ; Se Não: Pula Para Pedra 5
    MOV   A, B
                            ; Verifica Se Pedra 4 Chegou Ao Fim do Display
    JZ    RELOAD_P4         ; POS4 = 00H => Reinicializa Pedra 4
    XRL   A, #40H           ; POS4 = 40H => Reinicializa Pedra 4
    JNZ   DEC_P4            ; Se Não: Decrementa Posição
RELOAD_P4:
    MOV   POS4, #FFH        ; POS4 Recebe Valor de Espera
    INC   NIVEL             ; Incrementa Contador Para Mudança de Nível
    INC   SCORE_0           ; Incrementa o Score Usando BCD 4bits
    MOV   A, SCORE_0
    XRL   A, #10101010B     ; Verifica Estouro de Ambas Partes
    JZ    SC_4              ; Se Sim: Pula Para SC_1
    MOV   A, SCORE_0        ; Recupera o Valor (Alterado por XRL)
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA5            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_0        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_0, A
    SJMP  PEDRA5
SC_4:                       ; Caso SCORE_0 Tenha Estourado
    MOV   SCORE_0, #00H     ; Zera SCORE_0
    INC   SCORE_1           ; Incrementa Score 1
```

```
    MOV   A, SCORE_1
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA5            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_1        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_1, A
    LJMP  PEDRA5            ; Pula Para Pedra 5
DEC_P4:
    DEC   POS4              ; Decrementa POS4

PEDRA5:
    MOV   B, POS5           ; B Armazena a Pedra 5
    MOV   A, B
    XRL   A, #FFH           ; Verifica Se a Pedra 5 Foi Inicializada
    JZ    PEDRA6            ; Se Não: Pula Para Pedra 6
    MOV   A, B
                            ; Verifica Se Pedra 5 Chegou Ao Fim do Display
    JZ    RELOAD_P5         ; POS5 = 00H => Reinicializa Pedra 5
    XRL   A, #40H           ; POS5 = 40H => Reinicializa Pedra 5
    JNZ   DEC_P5            ; Se Não: Decrementa Posição
RELOAD_P5:
    MOV   POS5, #FFH        ; POS5 Recebe Valor de Espera
    INC   NIVEL             ; Incrementa Contador Para Mudança de Nível
    INC   SCORE_0           ; Incrementa o Score Usando BCD 4bits
    MOV   A, SCORE_0
    XRL   A, #10101010B     ; Verifica Estouro de Ambas Partes
    JZ    SC_5              ; Se Sim: Pula Para SC_1
    MOV   A, SCORE_0        ; Recupera o Valor (Alterado por XRL)
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA6            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_0        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_0, A
    SJMP  PEDRA6
SC_5:                       ; Caso SCORE_0 Tenha Estourado
    MOV   SCORE_0, #00H     ; Zera SCORE_0
    INC   SCORE_1           ; Incrementa Score 1
    MOV   A, SCORE_1
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA6            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_1        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_1, A
    LJMP  PEDRA6            ; Pula Para Pedra 6
DEC_P5:
    DEC   POS5              ; Decrementa POS5

PEDRA6:
    MOV   B, POS6           ; B Armazena a Pedra 6
    MOV   A, B
    XRL   A, #FFH           ; Verifica Se a Pedra 6 Foi Inicializada
    JZ    PEDRA7            ; Se Não: Pula Para Pedra 7
    MOV   A, B
                            ; Verifica Se Pedra 6 Chegou Ao Fim do Display
    JZ    RELOAD_P6         ; POS6 = 00H => Reinicializa Pedra 6
    XRL   A, #40H           ; POS6 = 40H => Reinicializa Pedra 6
    JNZ   DEC_P6            ; Se Não: Decrementa Posição
RELOAD_P6:
```

```
    MOV   POS6, #FFH        ; POS6 Recebe Valor de Espera
    INC   NIVEL             ; Incrementa Contador Para Mudança de Nível
    INC   SCORE_0           ; Incrementa o Score Usando BCD 4bits
    MOV   A, SCORE_0
    XRL   A, #10101010B     ; Verifica Estouro de Ambas Partes
    JZ    SC_6              ; Se Sim: Pula Para SC_1
    MOV   A, SCORE_0        ; Recupera o Valor (Alterado por XRL)
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA7            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_0        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_0, A
    SJMP  PEDRA7
SC_6:                       ; Caso SCORE_0 Tenha Estourado
    MOV   SCORE_0, #00H     ; Zera SCORE_0
    INC   SCORE_1           ; Incrementa Score 1
    MOV   A, SCORE_1
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA7            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_1        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_1, A
    LJMP  PEDRA7            ; Pula Para Pedra 7
DEC_P6:
    DEC   POS6              ; Decrementa POS6

PEDRA7:
    MOV   B, POS7           ; B Armazena a Pedra 7
    MOV   A, B
    XRL   A, #FFH           ; Verifica Se a Pedra 7 Foi Inicializada
    JZ    PEDRA8            ; Se Não: Pula Para Pedra 8
    MOV   A, B
                            ; Verifica Se Pedra 7 Chegou Ao Fim do Display
    JZ    RELOAD_P7         ; POS7 = 00H => Reinicializa Pedra 7
    XRL   A, #40H           ; POS7 = 40H => Reinicializa Pedra 7
    JNZ   DEC_P7            ; Se Não: Decrementa Posição
RELOAD_P7:
    MOV   POS7, #FFH        ; POS7 Recebe Valor de Espera
    INC   NIVEL             ; Incrementa Contador Para Mudança de Nível
    INC   SCORE_0           ; Incrementa o Score Usando BCD 4bits
    MOV   A, SCORE_0
    XRL   A, #10101010B     ; Verifica Estouro de Ambas Partes
    JZ    SC_7              ; Se Sim: Pula Para SC_1
    MOV   A, SCORE_0        ; Recupera o Valor (Alterado por XRL)
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA8            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_0        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_0, A
    SJMP  PEDRA8
SC_7:                       ; Caso SCORE_0 Tenha Estourado
    MOV   SCORE_0, #00H     ; Zera SCORE_0
    INC   SCORE_1           ; Incrementa Score 1
    MOV   A, SCORE_1
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA8            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_1        ; Se Sim: Recupera Score (Alterado por ANL)
```

```
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_1, A
    LJMP  PEDRA8            ; Pula Para Pedra 8
DEC_P7:
    DEC   POS7              ; Decrementa POS7

PEDRA8:
    MOV   B, POS8           ; B Armazena a Pedra 8
    MOV   A, B
    XRL   A, #FFH           ; Verifica Se a Pedra 8 Foi Inicializada
    JZ    PEDRA9            ; Se Não: Pula Para Pedra 9
    MOV   A, B
                            ; Verifica Se Pedra 8 Chegou Ao Fim do Display
    JZ    RELOAD_P8         ; POS8 = 00H => Reinicializa Pedra 8
    XRL   A, #40H           ; POS8 = 40H => Reinicializa Pedra 8
    JNZ   DEC_P8            ; Se Não: Decrementa Posição
RELOAD_P8:
    MOV   POS8, #FFH        ; POS8 Recebe Valor de Espera
    INC   NIVEL             ; Incrementa Contador Para Mudança de Nível
    INC   SCORE_0           ; Incrementa o Score Usando BCD 4bits
    MOV   A, SCORE_0
    XRL   A, #10101010B     ; Verifica Estouro de Ambas Partes
    JZ    SC_8              ; Se Sim: Pula Para SC_1
    MOV   A, SCORE_0        ; Recupera o Valor (Alterado por XRL)
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA9            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_0        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_0, A
    SJMP  PEDRA9
SC_8:                       ; Caso SCORE_0 Tenha Estourado
    MOV   SCORE_0, #00H     ; Zera SCORE_0
    INC   SCORE_1           ; Incrementa Score 1
    MOV   A, SCORE_1
    ANL   A, #00001111B     ; Descarta Parte Mais Alta
    XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
    JNZ   PEDRA9            ; Se Não: Pula Para Próxima Pedra
    MOV   A, SCORE_1        ; Se Sim: Recupera Score (Alterado por ANL)
    ADD   A, #00010000B     ;         Incrementa Parte Alta
    ANL   A, #11110000B     ;         Zera Parte Baixa
    MOV   SCORE_1, A
    LJMP  PEDRA9            ; Pula Para Pedra 9
DEC_P8:
    DEC   POS8              ; Decrementa POS8

PEDRA9:
    MOV   B, POS9           ; B Armazena a Pedra 9
    MOV   A, B
    XRL   A, #FFH           ; Verifica Se a Pedra 9 Foi Inicializada
    JZ    OUT_DEC           ; Se Não: Sai da Rotina
    MOV   A, B
                            ; Verifica Se Pedra 9 Chegou Ao Fim do Display
    JZ    RELOAD_P9         ; POS9 = 00H => Reinicializa Pedra 9
    XRL   A, #40H           ; POS9 = 40H => Reinicializa Pedra 9
    JNZ   DEC_P9
RELOAD_P9:
    MOV   POS9, #FFH        ; POS9 Recebe Valor de Espera
    INC   NIVEL             ; Incrementa Contador Para Mudança de Nível
    INC   SCORE_0           ; Incrementa o Score Usando BCD 4bits
    MOV   A, SCORE_0
    XRL   A, #10101010B     ; Verifica Estouro de Ambas Partes
```

```
        JZ    SC_9              ; Se Sim: Pula Para SC_1
        MOV   A, SCORE_0        ; Recupera o Valor (Alterado por XRL)
        ANL   A, #00001111B     ; Descarta Parte Mais Alta
        XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
        JNZ   OUT_DEC           ; Se Não: Pula Para Próxima Pedra
        MOV   A, SCORE_0        ; Se Sim: Recupera Score (Alterado por ANL)
        ADD   A, #00010000B     ;         Incrementa Parte Alta
        ANL   A, #11110000B     ;         Zera Parte Baixa
        MOV   SCORE_0, A
        SJMP  OUT_DEC
    SC_9:                       ; Caso SCORE_0 Tenha Estourado
        MOV   SCORE_0, #00H     ; Zera SCORE_0
        INC   SCORE_1           ; Incrementa Score 1
        MOV   A, SCORE_1
        ANL   A, #00001111B     ; Descarta Parte Mais Alta
        XRL   A, #00001010B     ; Verifica Estouro da Parte Baixa
        JNZ   OUT_DEC           ; Se Não: Pula Para Próxima Pedra
        MOV   A, SCORE_1        ; Se Sim: Recupera Score (Alterado por ANL)
        ADD   A, #00010000B     ;         Incrementa Parte Alta
        ANL   A, #11110000B     ;         Zera Parte Baixa
        MOV   SCORE_1, A
        LJMP  OUT_DEC           ; Sai da Rotina
    DEC_P9:
        DEC   POS9              ; Decrementa POS9

    OUT_DEC:
        RET


; Função Responsável Por Inicializar as Pedras (GET_POS)
; Mantendo o Tempo de Espera Aleatório (GET_ESP) Entre Uma e Outra
; Controla Através de R5 Qual Pedra Deve Ser Inicializada
INI_POS:
        MOV   B, R5             ; B Armazena R5
        MOV   A, B
        XRL   A, #00000000B     ; Verifica Se Deve Tratar a Pedra 1
        JNZ   INI_P2            ; Se Já Foi Inicializada Pula Para Pedra 2
        MOV   A, R6             ; Verifica Se o Espaço Já Foi Obtido
        XRL   A, #FFH
        JNZ   W_P1              ; Caso Sim: Trata o Espaço de Espera
        LCALL GET_ESP           ; Caso Não: Obtém Espaço de Espera
    W_P1:
        DEC   R6                ; Decrementa o Espaço de Espera
        MOV   A, R6             ; Verifica Se Acabou a Espera
;       JNZ   OUT_INI           ; Se Não: Sai de INI_POS
        JNZ   LOCAL_1           ; LOCAL_1 e LOCAL_11 Necessários Devido
        SJMP  LOCAL_11          ; Ao Estouro do Laço de JNZ para OUT_INI
    LOCAL_1:
        LJMP  OUT_INI
    LOCAL_11:
        MOV   R1, #POS1         ; Se Sim:
        LCALL GET_POS           ;    Obtém Posição Aleatória da Pedra 1
        MOV   R5, #00000001B    ;    Indica Pedra 2 em R5
        MOV   R6, #FFH          ;    Seta Espaço Como Não Obtido
        LJMP  OUT_INI           ;    Sai de INI_POS


    INI_P2:
        MOV   A, B              ; Recupera Valor em B (R5)
        XRL   A, #00000001B     ; Verifica Se Deve Tratar a Pedra 2
        JNZ   INI_P3            ; Se Já Foi Inicializada Pula Para Pedra 3
        MOV   A, R6             ; Verifica Se o Espaço Já Foi Obtido
        XRL   A, #FFH
        JNZ   W_P2              ; Caso Sim: Trata o Espaço de Espera
```

```
      LCALL GET_ESP           ; Caso Não: Obtém Espaço de Espera
  W_P2:
      DEC   R6                ; Decrementa o Espaço de Espera
      MOV   A, R6             ; Verifica Se Acabou a Espera
;     JNZ   OUT_INI           ; Se Não: Sai de INI_POS
      JNZ   LOCAL_2           ; LOCAL_2 e LOCAL_22 Necessários Devido
      SJMP  LOCAL_22          ; Ao Estouro do Laço de JNZ para OUT_INI
  LOCAL_2:
      LJMP  OUT_INI
  LOCAL_22:
      MOV   R1, #POS2         ; Se Sim:
      LCALL GET_POS           ;    Obtém Posição Aleatória da Pedra 2
      MOV   R5, #00000010B    ;    Indica Pedra 3 em R5
      MOV   R6, #FFH          ;    Seta Espaço Como Não Obtido
      LJMP  OUT_INI           ;    Sai de INI_POS


  INI_P3:
      MOV   A, B              ; Recupera Valor em B (R5)
      XRL   A, #00000010B     ; Verifica Se Deve Tratar Pedra 3
      JNZ   INI_P4            ; Se Já Foi Inicializada Pula Para Pedra 4
      MOV   A, R6             ; Verifica Se o Espaço Já Foi Obtido
      XRL   A, #FFH
      JNZ   W_P3              ; Caso Sim: Trata o Espaço de Espera
      LCALL GET_ESP           ; Caso Não: Obtém Espaço de Espera
  W_P3:
      DEC   R6                ; Decrementa o Espaço de Espera
      MOV   A, R6             ; Verifica Se Acabou a Espera
;     JNZ   OUT_INI           ; Se Não: Sai de INI_POS
      JNZ   LOCAL_3           ; LOCAL_3 e LOCAL_33 Necessários Devido
      SJMP  LOCAL_33          ; Ao Estouro do Laço de JNZ para OUT_INI
  LOCAL_3:
      LJMP  OUT_INI
  LOCAL_33:
      MOV   R1, #POS3         ; Se Sim:
      LCALL GET_POS           ;    Obtém Posição Aleatória da Pedra 3
      MOV   R5, #00000011B    ;    Indica Pedra 4 em R5
      MOV   R6, #FFH          ;    Seta Espaço Como Não Obtido
      LJMP  OUT_INI           ;    Sai de INI_POS

  INI_P4:
      MOV   A, B              ; Recupera Valor em B (R5)
      XRL   A, #00000011B     ; Verifica Se Deve Tratar Pedra 4
      JNZ   INI_P5            ; Se Já Foi Inicializada Pula Para Pedra 5
      MOV   A, R6             ; Verifica Se o Espaço Já Foi Obtido
      XRL   A, #FFH
      JNZ   W_P4              ; Caso Sim: Trata o Espaço de Espera
      LCALL GET_ESP           ; Caso Não: Obtém Espaço de Espera
  W_P4:
      DEC   R6                ; Decrementa o Espaço de Espera
      MOV   A, R6             ; Verifica Se Acabou a Espera
;     JNZ   OUT_INI           ; Se Não: Sai de INI_POS
      JNZ   LOCAL_4           ; LOCAL_4 e LOCAL_44 Necessários Devido
      SJMP  LOCAL_44          ; Ao Estouro do Laço de JNZ para OUT_INI
  LOCAL_4:
      LJMP  OUT_INI
  LOCAL_44:
      MOV   R1, #POS4         ; Se Sim:
      LCALL GET_POS           ;    Obtém Posição Aleatória da Pedra 4
      MOV   R5, #00000100B    ;    Indica Pedra 5 em R5
      MOV   R6, #FFH          ;    Seta Espaço Como Não Obtido
      LJMP  OUT_INI           ;    Sai de INI_POS

  INI_P5:
```

```
        MOV   A, B              ; Recupera Valor em B (R5)
        XRL   A, #00000100B     ; Verifica Se Deve Tratar Pedra 5
        JNZ   INI_P6            ; Se Já Foi Inicializada Pula Para Pedra 6
        MOV   A, R6             ; Verifica Se o Espaço Já Foi Obtido
        XRL   A, #FFH
        JNZ   W_P5              ; Caso Sim: Trata o Espaço de Espera
        LCALL GET_ESP           ; Caso Não: Obtém Espaço de Espera
    W_P5:
        DEC   R6                ; Decrementa o Espaço de Espera
        MOV   A, R6             ; Verifica Se Acabou a Espera
;       JNZ   OUT_INI           ; Se Não: Sai de INI_POS
        JNZ   LOCAL_5           ; LOCAL_5 e LOCAL_55 Necessários Devido
        SJMP  LOCAL_55          ; Ao Estouro do Laço de JNZ para OUT_INI
    LOCAL_5:
        LJMP  OUT_INI
    LOCAL_55:
        MOV   R1, #POS5         ; Se Sim:
        LCALL GET_POS           ;    Obtém Posição Aleatória da Pedra 5
        MOV   R5, #00000101B    ;    Indica Pedra 6 em R5
        MOV   R6, #FFH          ;    Seta Espaço Como Não Obtido
        LJMP  OUT_INI           ;    Sai de INI_POS

    INI_P6:
        MOV   A, B              ; Recupera Valor em B (R5)
        XRL   A, #00000101B     ; Verifica Se Deve Tratar Pedra 6
        JNZ   INI_P7            ; Se Já Foi Inicializada Pula Para Pedra 7
        MOV   A, R6             ; Verifica Se o Espaço Já Foi Obtido
        XRL   A, #FFH
        JNZ   W_P6              ; Caso Sim: Trata o Espaço de Espera
        LCALL GET_ESP           ; Caso Não: Obtém Espaço de Espera
    W_P6:
        DEC   R6                ; Decrementa o Espaço de Espera
        MOV   A, R6             ; Verifica Se Acabou a Espera
        JNZ   OUT_INI           ; Se Não: Sai de INI_POS
        MOV   R1, #POS6         ; Se Sim:
        LCALL GET_POS           ;    Obtém Posição Aleatória da Pedra 6
        MOV   R5, #00000110B    ;    Indica Pedra 7 em R5
        MOV   R6, #FFH          ;    Seta Espaço Como Não Obtido
        LJMP  OUT_INI           ;    Sai de INI_POS

    INI_P7:
        MOV   A, B              ; Recupera Valor em B (R5)
        XRL   A, #00000110B     ; Verifica Se Deve Tratar Pedra 7
        JNZ   INI_P8            ; Se Já Foi Inicializada Pula Para Pedra 8
        MOV   A, R6             ; Verifica Se o Espaço Já Foi Obtido
        XRL   A, #FFH
        JNZ   W_P7              ; Caso Sim: Trata o Espaço de Espera
        LCALL GET_ESP           ; Caso Não: Obtém Espaço de Espera
    W_P7:
        DEC   R6                ; Decrementa o Espaço de Espera
        MOV   A, R6             ; Verifica Se Acabou a Espera
        JNZ   OUT_INI           ; Se Não: Sai de INI_POS
        MOV   R1, #POS7         ; Se Sim:
        LCALL GET_POS           ;    Obtém Posição Aleatória da Pedra 7
        MOV   R5, #00000111B    ;    Indica Pedra 8 em R5
        MOV   R6, #FFH          ;    Seta Espaço Como Não Obtido
        LJMP  OUT_INI           ;    Sai de INI_POS

    INI_P8:
        MOV   A, B              ; Recupera Valor em B (R5)
        XRL   A, #00000111B     ; Verifica Se Deve Tratar Pedra 8
        JNZ   INI_P9            ; Se Já Foi Inicializada Pula Para Pedra 9
        MOV   A, R6             ; Verifica Se o Espaço Já Foi Obtido
        XRL   A, #FFH
```

```
        JNZ   W_P8              ; Caso Sim: Trata o Espaço de Espera
        LCALL GET_ESP           ; Caso Não: Obtém Espaço de Espera
    W_P8:
        DEC   R6                ; Decrementa o Espaço de Espera
        MOV   A, R6             ; Verifica Se Acabou a Espera
        JNZ   OUT_INI           ; Se Não: Sai de INI_POS
        MOV   R1, #POS8         ; Se Sim:
        LCALL GET_POS           ;    Obtém Posição Aleatória da Pedra 8
        MOV   R5, #00001000B    ;    Indica Pedra 9 em R5
        MOV   R6, #FFH          ;    Seta Espaço Como Não Obtido
        LJMP  OUT_INI           ;    Sai de INI_POS

    INI_P9:
        MOV   A, B              ; Recupera Valor em B (R5)
        XRL   A, #00001000B     ; Verifica Se Deve Tratar Pedra 9
        JNZ   OUT_INI           ; Se Já Foi Inicializada Sai de INI_POS
        MOV   A, R6             ; Verifica Se o Espaço Já Foi Obtido
        XRL   A, #FFH
        JNZ   W_P9              ; Caso Sim: Trata o Espaço de Espera
        LCALL GET_ESP           ; Caso Não: Obtém Espaço de Espera
    W_P9:
        DEC   R6                ; Decrementa o Espaço de Espera
        MOV   A, R6             ; Verifica Se Acabou a Espera
        JNZ   OUT_INI           ; Se Não: Sai de INI_POS
        MOV   R1, #POS9         ; Se Sim:
        LCALL GET_POS           ;    Obtém Posição Aleatória da Pedra 9
        MOV   R5, #00000000B    ;    Indica Pedra 1 em R5
        MOV   R6, #FFH          ;    Seta Espaço Como Não Obtido

    OUT_INI:                    ; Sai de INI_POS
        RET


; Rotina Responsável Por Tratar/Desenhar a Colisão
CRASH:
        MOV   IE, #00H          ; Desabilita as Interrupções
        MOV   A, #00H           ; Posição no Display = 00H (Linha 1)
        MOV   B, A              ; B Armazena A
        ORL   A, #10000000B     ; Endereça a Posição na DDRAM
        LCALL LCD_CMD
    DESLOC1:                    ; Desloca Caractere Pela Linha 1 (Esq-Dir)
        MOV   A, #FFH           ; Escreve Caractere Escuro
        LCALL LCD_CHAR
        MOV   A, #1D
        LCALL ESPERA            ; Atraso de 1/20 s
        MOV   A, B              ; Recupera B
        ORL   A, #10000000B     ; Endereça a Posição na DDRAM
        LCALL LCD_CMD
        INC   B                 ; Incrementa B
        MOV   A, B
        XRL   A, #41D           ; Verifica Se Acabou a Linha
        JZ    DESLOC2           ; Se Sim: Vai Para DESLOC2
        LJMP  DESLOC1           ; Se Não: Continua em DESLOC1
    DESLOC2:                    ; Desloca Caractere Pela Linha 2 (Dir-Esq)
        MOV   A, #67H           ; Posição no Display = 67H (Linha 2)
        MOV   B, A              ; B Armazena A
        ORL   A, #10000000B     ; Endereça a Posição na DDRAM
        LCALL LCD_CMD
    LOOP_D2:
        MOV   A, #FFH           ; Escreve Caractere Escuro
        LCALL LCD_CHAR
        MOV   A, #1D
        LCALL ESPERA            ; Atraso de 1/20 s
        MOV   A, B              ; Recupera B
```

```
      ORL   A, #10000000B     ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      DEC   B                 ; Decrementa B
      MOV   A, B
      XRL   A, #3EH           ; Verifica Se Acabou a Linha
      JZ    OUT_DES           ; Se Sim: Sai de CRASH
      LJMP  LOOP_D2           ; Se Não: Continua em LOOP_D2
  OUT_DES:
      MOV   A, #10D
      LCALL ESPERA            ; Atraso de 1/2 s

      MOV   A, #00000001B     ; Limpa a Tela
      LCALL LCD_CMD
      MOV   A, #0H            ; Posição no Display = 00H (Linha 1)
      MOV   DPTR, #PONTOS     ; String a Ser Enviada
      LCALL LCD_POS_STRING    ; Escreve a String Pontuação
      MOV   A, #22D           ; Posição no Display = 22D (Linha 1)
      MOV   DPTR, #H_PONT     ; String a Ser Enviada
      LCALL LCD_POS_STRING    ; Escreve a String Pontuação Máxima
      MOV   A, #5D
      LCALL ESPERA            ; Atraso
      MOV   A, #08D           ; Posição no Display = 08D (Linha 1)
      ORL   A, #10000000B     ; Endereça a Posição na DDRAM
      LCALL LCD_CMD
      MOV   A, SCORE_1
; Faz a Decodificação do Caractere da Casa do Milhar (Score)
      ANL   A, #11110000B     ; Despreza Parte Menos Significativa
      RR    A
      RR    A                 ; Traz Parte Mais Significativa Para Menos
      RR    A
      RR    A
      ADD   A, #30H           ; Adiciona 30H Devido ao ASCII ('0' = 30H)
      LCALL LCD_CHAR          ; Escreve na Tela o Número Obtido
      MOV   A, #5D
      LCALL ESPERA            ; Atraso
; Faz a Decodificação do Caractere da Casa da Centena (Score)
      MOV   A, SCORE_1
      ANL   A, #00001111B     ; Despreza Parte Mais Significativa
      ADD   A, #30H           ; Adiciona 30H Devido ao ASCII ('0' = 30H)
      LCALL LCD_CHAR          ; Escreve na Tela o Número Obtido
      MOV   A, #5D
      LCALL ESPERA            ; Atraso
; Faz a Decodificação do Caractere da Casa da Dezena (Score)
      MOV   A, SCORE_0
      ANL   A, #11110000B     ; Despreza Parte Menos Significativa
      RR    A
      RR    A                 ; Traz Parte Mais Significativa Para Menos
      RR    A
      RR    A
      ADD   A, #30H           ; Adiciona 30H Devido ao ASCII ('0' = 30H)
      LCALL LCD_CHAR          ; Escreve na Tela o Número Obtido
      MOV   A, #5D
      LCALL ESPERA            ; Atraso
; Faz a Decodificação do Caractere da Casa da Unidade (Score)
      MOV   A, SCORE_0
      ANL   A, #00001111B     ; Despreza Parte Mais Significativa
      ADD   A, #30H           ; Adiciona 30H Devido ao ASCII ('0' = 30H)
      LCALL LCD_CHAR          ; Escreve na Tela o Número Obtido
      MOV   A, #5D
      LCALL ESPERA            ; Atraso

      MOV   A, H_SCR1
      CLR   C                 ; Limpa o Flag Carry
      SUBB  A, SCORE_1        ; Verifica Se Score é Maior Que o High
```

```
        JC    NEW_SCORE         ; Se Sim: Pula Para New_Score
        MOV   A, H_SCR0         ; Se Não:
        CLR   C                 ;         Limpa o Flag Carry
        SUBB  A, SCORE_0        ;         Compara Com Parte Baixa do Score
        JC    NEW_SCORE         ; Se Maior: Pula Para New_Score
        JMP   OUT_HIGH          ; Se Não: Sai
    NEW_SCORE:                  ; Define o Novo High Score
        MOV   A, SCORE_0
        MOV   H_SCR0, A         ; Passa o Score Para o High Score
        MOV   A, SCORE_1
        MOV   H_SCR1, A
    OUT_HIGH:

; Faz a Decodificação do Caractere da Casa do Milhar (High Score)
        MOV   A, #35D           ; Posição no Display = 35D (Linha 1)
        ORL   A, #10000000B     ; Endereça a Posição na DDRAM
        LCALL LCD_CMD
        MOV   A, H_SCR1
        ANL   A, #11110000B     ; Despreza Parte Menos Significativa
        RR    A
        RR    A                 ; Traz Parte Mais Significativa Para Menos
        RR    A
        RR    A
        ADD   A, #30H           ; Adiciona 30H Devido ao ASCII ('0' = 30H)
        LCALL LCD_CHAR          ; Escreve na Tela o Número Obtido
        MOV   A, #5D
        LCALL ESPERA            ; Atraso
; Faz a Decodificação do Caractere da Casa da Centena (High Score)
        MOV   A, H_SCR1
        ANL   A, #00001111B     ; Despreza Parte Mais Significativa
        ADD   A, #30H           ; Adiciona 30H Devido ao ASCII ('0' = 30H)
        LCALL LCD_CHAR          ; Escreve na Tela o Número Obtido
        MOV   A, #5D
        LCALL ESPERA            ; Atraso
; Faz a Decodificação do Caractere da Casa da Dezena (High Score)
        MOV   A, H_SCR0
        ANL   A, #11110000B     ; Despreza Parte Menos Significativa
        RR    A
        RR    A                 ; Traz Parte Mais Significativa Para Menos
        RR    A
        RR    A
        ADD   A, #30H           ; Adiciona 30H Devido ao ASCII ('0' = 30H)
        LCALL LCD_CHAR          ; Escreve na Tela o Número Obtido
        MOV   A, #5D
        LCALL ESPERA            ; Atraso
; Faz a Decodificação do Caractere da Casa da Unidade (High Score)
        MOV   A, H_SCR0
        ANL   A, #00001111B     ; Despreza Parte Mais Significativa
        ADD   A, #30H           ; Adiciona 30H Devido ao ASCII ('0' = 30H)
        LCALL LCD_CHAR          ; Escreve na Tela o Número Obtido
        MOV   A, #5D
        LCALL ESPERA            ; Atraso
        MOV   A, #40H           ; Posição no Display = 40H (Linha 2)
        MOV   DPTR, #STIT       ; String a Ser Enviada
        LCALL LCD_POS_STRING    ; Escreve String Auxiliar

    TECLA_INIT2:                ; Espera Tecla C Ser Pressionada para
                                ; Dar Continuidade ao Programa
        MOV   P1, #11111111B    ; Prepara o Porto 1
        CLR   P1.7              ; Ativa a Linha 4
        MOV   A, P1             ; Verifica o Porto 1
        XRL   A, #01110111B     ; Verifica Se a Tecla C foi Pressionada
        JNZ   TECLA_INIT2       ; Continua Se a Tecla foi Pressionada
```

```
      LJMP  MENU_CARRINHOS    ; Pula Para o Menu de Escolha
                              ; Onde a Pilha é Tratada Novamente
                              ; Evitando um Possível 'Stack Overflow'


; Espera um Determinado Tempo
; A: Tempo em 1/20 segundos
; Esta Rotina Usa o Timer T1
; Esta Rotina Não é e Nem Deve Ser Chamada Após
; a Inicialização dos Timer da Engine do Jogo
ESPERA:
      ANL   TMOD, #0FH        ; Configura Timer para 16-bit
      ORL   TMOD, #10H
      MOV   TH1, #4CH
      MOV   TL1, #01H
      CLR   TF1               ; Limpa o Bit de Overflow
      SETB  TR1               ; Ativa o Timer
  _ESP_1:
      JNB   TF1, $            ; Espera Ocorrer o Overflow
      CLR   TF1               ; Limpa o Bit de Overflow
      DJNZ  A, _ESP_1         ; Repete Caso Não Tiver Atingido o Tempo
      CLR   TR1               ; Desliga o timer
      RET
```