TÉCNICO LISBOA

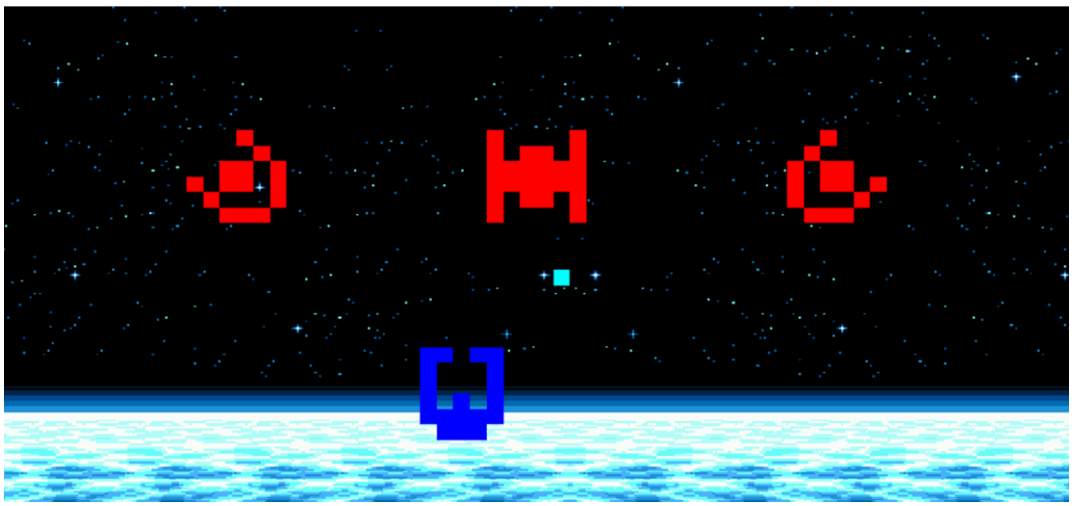
ARQUITETURA DE COMPUTADORES

LETI

IST-TAGUSPARK

RELATÓRIO DO PROJETO

BATALHA ESPACIAL



GRUPO 14:

Diogo Lopes Nº96732

Francisco Silva Nº97366

Luís Semedo Nº96754

1. Introdução

Este projeto surge no âmbito da cadeira de Arquitetura de Computadores, onde nos é proposto programar um jogo de batalha espacial em linguagem Assembly, usando o PEPE-16 (Processador Especial Para Ensino – 16 bits), com o objetivo de pormos em prática o conhecimento adquirido sobre a cadeira, nomeadamente programação em linguagem Assembly, periféricos e interrupções.

O jogador tem como objetivo acertar no máximo número de naves inimigas possíveis até alcançar 100 de pontuação. Por cada nave em que acerta o jogador ganha 5 pontos. Por outro lado, se deixar as naves inimigas chegarem ao fundo do ecrã perde 10 pontos.

O jogador perde o jogo se colidir com algumas das naves inimigas ou se a pontuação chegar a 0.

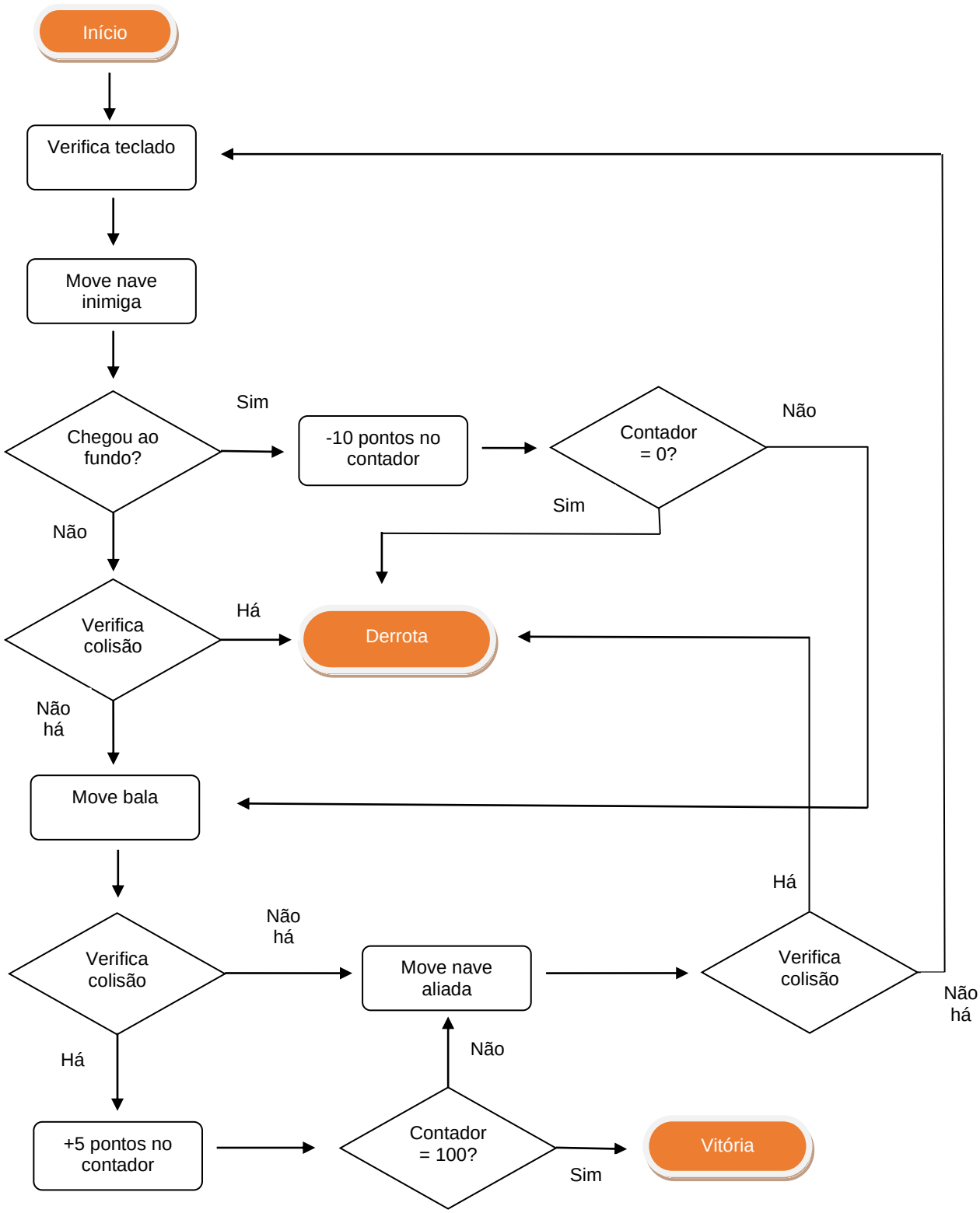
Tecla	Ação
0	Move nave Noroeste
1	Move nave Norte
2	Move nave Nordeste
4	Move nave Oeste
5	Dispara Bala
6	Move nave Este
8	Move nave Sudoeste
9	Move nave Sul
A	Move nave Sudeste
7	Fim voluntário jogo
B	Início/Pausa jogo

Figura 1 - Instruções do teclado

Na secção seguinte apresentamos a estrutura do software. Posteriormente, na secção 3, tiramos conclusões acerca da elaboração do projeto, nomeadamente em relação ao código realizado, aos objetivos que conseguimos concluir e aos que não concretizámos.

2. Conceção e Implementação

2.1. Software



2.2. Rotinas e Interrupções

TECLADO – a rotina do teclado é idêntica à do relatório com a ligeira diferença que é desbloqueado e tem outra rotina lá dentro que chama as funções de cada tecla.

BACKG – rotina que coloca o cenário desejado ao comparar o conteúdo de uma variável de seleção SWITCH_BACKG. Coloca o cenário de inicialização/pausa, modo de jogo, vitória e derrota.

ESCOLHA_SOM – o seu modo de funcionamento é idêntico ao da rotina BACKG. A sua variável de seleção é SWITCH_SOM, e escolhe o som da inicialização, disparo, colisão, derrota e vitória.

DES_OBJ – esta rotina serve para desenhar uma imagem no pixelscreen se receber registos com informações referentes à coordenada do X, do Y, número de pixéis, constante para a cor que servirá para comparar dentro desta mesma função, um switch para identificar se o objetivo é para ligar (1) ou desligar (8) o objeto em questão e um modelo que serve para adicionar ao X e Y de forma a imprimir uma figura linha a linha.

DES_BALA – serve para desenhar a primeira aparência da bala e colocar as coordenadas dentro da string BALA1 para depois ser animada na rotina de interrupção.

ROT_MOVIMENTO – desliga a nave na posição atual, avalia a tecla que foi premida, conforme a tecla escolhe qual a direção que a nave avançar, muda as coordenadas principais e volta a desenhar a nave nesta nova posição.

SCOREBOARD_AD_SUB – se o valor do R1 que entra nesta rotina for 3, esta irá adicionar 5 ao contador, caso não seja 3 irá subtrair 10 ao contador. Devido ao contador não fazer a conversão para decimal são necessárias verificações para certos casos como os de o contador ser 100, 95, 5 e 0 nas quais os valores são inseridos manualmente na variável do contador. Também se avalia se o jogador ganhou o jogo ou perdeu por pontos ao comparar o novo valor a ser colocado e tomando a devida ação.

LIGA_INT_INIM_BALAS – esta rotina serve de interruptor para as variáveis de estado das interrupções, fazendo com que estas sejam suspensas até que a tecla B seja premida de novo. Ao mesmo tempo irá também colocar um cenário de pausa.

ROT_INT_0 | ROT_INT_1 – nas interrupções serão averiguados os valores das variáveis de estado ATIVA_INT_BALA e ATIVA_INT_INIM e caso estejam a 1, as balas e as naves inimigas serão animadas.

ANIMA_INIM – as naves serão animadas todas ao mesmo tempo nesta rotina. As naves das extremidades terão as suas coordenadas trocadas de forma a se movimentarem num ângulo de 45 graus para o centro do pixelscreen. Desta

forma todas as naves irão colidir no centro do ecrã onde a nave aliada é desenhada da primeira vez.

VERIFICAR_NAVE_INIM /B/C/D – de forma a saber se houve colisão de naves, cada uma destas rotinas vai verificar um canto do desenho da nave aliada e compará-lo com as atuais coordenadas das naves inimigas. A primeira rotina avalia o canto superior esquerdo, a seguinte o canto superior direito, depois o canto inferior esquerdo e por fim o canto inferior direito.

ANIMA_BALA – para animar a bala simplesmente e necessário subtrair 1 ao y atual da bala, e compará-la com o topo do pixelscreen e com as coordenadas das naves inimigas de forma a perceber se houve ou não impacto.

VERIFICAR_BALA_INIM – de forma semelhante ao funcionamento das rotinas de verificação de colisão entre nave e inimigas, nesta apenas será verificado um X e Y pois a bala é caracterizada apenas por um pixel. No caso de colisão as naves terão as suas coordenadas trocadas para as iniciais de forma a regressarem ao início do pixelscreen.

2.3. Mapa de endereçamento escolhido

Dispositivo	Endereços
RAM	0000H a 3FFFH
PixelScreen (acesso aos comandos)	6000H e seguintes
PixelScreen (acesso à sua memória)	8000H a 80FFH
POUT-1 (periférico de saída de 16 bits)	0A000H
POUT-2 (periférico de saída de 8 bits)	0C000H
PIN (periférico de entrada de 8 bits)	0E000H

Figura 2 - Mapa de endereços

3. Conclusões

Este trabalho consistia essencialmente num jogo de simulação de voo de uma nave espacial que tem de enfrentar 3 naves inimigas através de balas.

O jogador acumula 5 pontos de cada vez que acerta com uma bala numa nave inimiga e perde 10 caso uma das naves inimigas chegue ao fim do pixelscreen. Se o chegar a 100 pontos, ganhou o jogo e é derrotado caso o contador chegue ao 0.

O jogo que conseguimos realizar contém a maioria dos objetivos iniciais. A parte que não chegamos a resolver por completo encontra-se no conjunto das naves inimigas. No nosso trabalho as 3 naves funcionam como uma unidade, ou seja, o que afetar uma, afeta as outras duas também. Pensamos que poderíamos ter contornado este problema ao termos separado as rotinas relativas à unidade de naves em 3 módulos diferentes com mais variáveis de estado, dessa forma seria possível fazer as verificações nave a nave ao invés de estarem ligadas.

Como só usamos uma variável para as balas, só é possível estar uma no ecrã a qualquer dada altura. Apesar deste défice de munição, a bala pode ser disparada a qualquer altura do jogo, dando assim hipótese ao jogador de corrigir um erro pontaria.

A nave aliada, tem pequenas interferências no movimento que fazem com que sejam ligados pixéis aleatórios quando esta se move com uma velocidade elevada.

A única parte do projeto que está ausente é o Gerador de um número pseudoaleatório para a escolha da nave inimiga a ser desenhada.

Tomamos certas rotas diferentes do objetivo inicial, nomeadamente a chamada da rotina do movimento duas vezes de forma a que a nave se mova ligeiramente mais depressa.

Em relação a melhorias ao enunciado, pusemos mais cenários e sons, tais como um cenário de pausa, vitória e derrota. Implementámos sons para diferentes processos como inicialização do jogo, disparo, colisões, vitória e derrota.

4. Código assembly

```
;###### DIOGO LOPES - 96732 #########

;###### FRANCISCO DA SILVA - 97366 ##

;###### LUIS SEMEDO - 96754 #########



PLACE 1000H

ECRA            EQU 6000H   ; endereço do ecra e da cor vermelha do pixel

SONORO          EQU 6012H   ; endereço som

DISPLAYS        EQU 0A000H  ; endereço dos displays de 7 segmentos (periférico POUT-1)

TEC_LIN         EQU 0C000H  ; endereço das linhas do teclado (periférico POUT-2)

TEC_COL         EQU 0E000H  ; endereço das colunas do teclado (periférico PIN)

LINHA           EQU 8H      ; primeira linha a ser testada

PIXEL_Y         EQU 600AH      ; endereco da linha do pixel

PIXEL_X         EQU 600CH       ; endereco da coluna do pixel

PIX_G           EQU 6002H       ; endereco da cor verde do pixel

PIX_B           EQU 6004H       ; endereço da cor azul do pixel

ESTADO_PIXEL    EQU 6008H      ; define se o pixel está ligado

CONTADOR_MIN    EQU 0H         ; minimo valor do contador

CONTADOR_MAX    EQU 100H       ; maximo valor do contador



L_ESQ_SUP       EQU 0H                ; limite do pixelscreen

N_LINHAS    EQU  27         ; número de linhas do écrã

PLACE           2000H

pilha:          TABLE   100H

sp_inicial:



CONTADOR:       WORD 0000H     ; variavel do contador



SWITCH_BACKG:  WORD 2H ; identifica o cenario a ser usado

SWITCH_SOM:            WORD 1H



NAVE_xy:        string 29, 24, 19, 1, 1       ; atributos das naves, por order
```

```
INIM_xy:        string 29, 0, 24, 0, 1 ; coordenada X, coordenada Y, nr de pixeis nas naves,

                                                                    ; 0 ou 1 para identificar se é aliado ou inimigo, e um switch para ligar/desligar

INIM2_xy:       string 3, 0, 14, 0, 1

INIM3_xy:       string 55, 0, 14, 0, 1


MODELO_Ax:    string  0, 1, 3, 4, 0, 4, 0, 4, 0, 2, 4, 0, 1, 2, 3, 4, 1, 2, 3  ; coordenadas em relação ao pixel superior esquerdo da nave,


MODELO_Ay:   string  0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5 ; ao somar os elemento da string MODELO_Ax juntamente com os elementos



                                    ; do MODELO_Ay, o pixel será ligado/desligado

MODELO_Ix:   string 0, 5, 0, 2, 3, 5, 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 2, 3, 5, 0, 5

MODELO_Iy:   string 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5


MODELO_I2x:   string 3, 4, 2, 3, 5, 0, 2, 3, 5, 1, 5, 2, 3, 4

MODELO_I2y:   string 0, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 5, 5, 5


MODELO_I3x:   string 2, 1, 0, 2, 3, 0, 2, 3, 5, 0, 4, 1, 2, 3

MODELO_I3y:   string 0, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 5, 5, 5


BALA1:                string 0, 0, 0        ; X, Y, switch

bala_ativa:           string 0                    ; variavel de estado para ativar/desativar a bala


ACABA_JOGO:           string 1                    ;variavel de estado para acabar o jogo


HA_COLISAO:           WORD 0                      ;variavel de estado para verificar colisao entre bala e inimigos



MOVIMENTO: string -1,-1,0,-1,1,-1,2,2,-1,0,2,2,1,0,2,2,-1,1,0,1,1,1              ; numeros que seram adicionados as coordenadas de forma a movimentar a nave (agrupados de 2 em 2)
```

```
tab:            WORD rot_int_0        ; rotina de atendimento da interrupção balas
                WORD  rot_int_1          ;  rotina  de  atendimento  da  interrupção
inimigos


ativa_int_inim:        WORD 0H                      ;   variavel   de   estado   para   a
interrupcao dos inimigos

ativa_int_bala:        WORD 0H                      ;   variavel   de   estado   para   a
interrupcao das balas




;################################################

;#                      Teclas                                                  #

;################################################

; conjunto de variaveis para comparacao na funcao_tecla


TECLA_0 EQU 0H

TECLA_1 EQU 1H

TECLA_2 EQU 2H

TECLA_3 EQU 3H

TECLA_4 EQU 4H

TECLA_5 EQU 5H

TECLA_6 EQU 6H

TECLA_7 EQU 7H

TECLA_8 EQU 8H

TECLA_9 EQU 9H

TECLA_A EQU 0AH

TECLA_B EQU 0BH

TECLA_C EQU 0CH

TECLA_D EQU 0DH

TECLA_E EQU 0EH

TECLA_F EQU 0FH


;#######################################

;#                            Código                  #

;#######################################
```

```
; O programa é iniciado com o background do start, um som de inicializacao, o SP inicial
e a tabela de interrupcoes;

; depois o programa entra num ciclo para buscar uma tecla

PLACE      0H

init:

        MOV BTE, tab

        MOV SP, sp_inicial

        EI0

        EI1

        EI

        CALL desenha_nave

        CALL backg     ; chama rotina que mete o fundo do ecra

        CALL escolha_som       ; chama a rotina do som

        MOV R1, 3

        CALL scoreboard_ad_sub ; finicializar o contador


main:

        MOV R1, ACABA_JOGO            ; verifica se a variavel de estado para acabar o
jogo foi ativada

        MOVB R2, [R1]

        CMP R2, 0

        JZ main

        CALL teclado ; chama a rotina do teclado uma vez e repete o ciclo

        JMP main


;#######################################

;#                    Teclado               #

;#######################################

; funcao que testa as 4 linhas do teclado de cada vez de forma a ao tornar o teclado
bloqueante

; ao premir uma tecla esta vai ser convertida para a respetiva posicao em hexadecimal

; dentro desta funcao será tambem chamada outra rotina que ira decidir que acao sera
tomada tendo em conta a tecla premida

teclado:

        MOV  R2, TEC_LIN   ; endereço do periférico das linhas (saida)

        MOV  R3, TEC_COL   ; endereço do periférico das colunas (entrada)
```

```
        MOV  R5, LINHA    ; primeira linha a ser testada
        MOV  R6, 0
        MOV  R7, 0
        MOV  R1, 0



ciclo_teclado:
        MOV  R6, 0              ; registo da linha e que guardará o valor da tecla
        MOV  R7, 0              ; registo da coluna
        CMP  R5, 1              ; compara se chegou à ultima
        JZ      desbloqueia    ; quando a linha chegar a ultima a ser testada o teclado
desbloqueia
        SHR R5, 1               ; muda de linha


espera_tecla:          ; neste ciclo espera-se até uma tecla ser premida
        MOV  R1, R5        ; testar a linha
        MOVB [R2], R1      ; escrever no periférico de saída (linhas)
        MOVB R0, [R3]      ; ler do periférico de entrada (colunas)
        CMP  R0, 0         ; há tecla premida?
        JZ   ciclo_teclado


converter_l:                ; conversao de linhas
        SHR R1, 1
        CMP R1, 0                   ; ve se chegou ao ultimo shift
        JZ     converter_c                  ; passa para a conversao das colunas
        ADD R6, 1                   ; conta os shifts
        JMP converter_l


converter_c:                ; conversao das colunas
        SHR R0, 1
        CMP R0, 0                   ; ve se chegou ao ultimo shift
        JZ     converter_final      ; passa para a ultima conversao necessaria
        ADD R7, 1                   ; conta os shifts
        JMP converter_c
```

```
converter_final:
        SHL R6, 2                       ; mult por 2
        ADD R6, R7                      ; soma as colunas e torna a tecla em hexa
        MOV R10, R6
        CALL funcao_tecla      ; verifca a acao a ser efetuada


desbloqueia:
        RET


ha_tecla:               ; neste ciclo espera-se até NENHUMA tecla estar premida
        MOV  R1, R5         ; testar a linha 4   (R1 tinha sido alterado)
        MOVB [R2], R1       ; escrever no periférico de saída (linhas)
        MOVB R0, [R3]       ; ler do periférico de entrada (colunas)
        CMP  R0, 0          ; há tecla premida?
        JNZ  ha_tecla       ; se ainda houver uma tecla premida, espera até não haver
        RET                         ; repete ciclo



;#########################################################
;#                     Colocar os atributos dos desenho              #
;#########################################################
; esta rotina serve para separa a string com os atributos de cada nave
; em registo de forma a serem utilizaveis na contrucao do desenho
; sao funcoes que irao inicializar os registos para a funcao "des_obj"
desenha_nave:
        PUSH R6
        MOV R1, NAVE_xy
        MOVB R2, [R1]  ; X DA NAVE
        ADD R1, 1
        MOVB R3, [R1]  ; Y DA NAVE
        ADD R1, 1
        MOVB R4, [R1]  ; NUMERO DE PIXELS DA NAVE
```

```
        ADD R1, 1
        MOVB R5, [R1]  ; DECIDIR COR ALIADA/INIMIGA
        ADD R1, 1
        MOVB R6, [R1]  ; LIGA/DESLIGA OS PIXELS DA NAVE
        MOV R7, MODELO_Ax      ; coordenadas dos Xs da nave
        MOV R8, MODELO_Ay      ; coordenadas dos Ys da nave
        CALL des_obj
        POP R6
        RET

desenha_inim:
        MOV R1, INIM_xy
        MOVB R2, [R1]  ; X DA INIMIGA
        ADD R1, 1
        MOVB R3, [R1]  ; Y DA INIMIGA
        ADD R1, 1
        MOVB R4, [R1]  ; NUMERO DE PIXELS DA INIMIGA
        ADD R1, 1
        MOVB R5, [R1]  ; DECIDIR COR ALIADA/INIMIGA
        ADD R1, 1
        MOVB R6, [R1]  ; LIGA/DESLIGA OS PIXELS DA NAVE
        MOV R7, MODELO_Ix      ; coordenadas dos Xs da nave
        MOV R8, MODELO_Iy      ; coordenadas dos Ys da nave
        CALL des_obj
        RET

desenha_inim2:
        MOV R1, INIM2_xy
        MOVB R2, [R1]  ; X DA INIMIGA
        ADD R1, 1
        MOVB R3, [R1]  ; Y DA INIMIGA
        ADD R1, 1
        MOVB R4, [R1]  ; NUMERO DE PIXELS DA INIMIGA
```

```
        ADD R1, 1
        MOVB R5, [R1]  ; DECIDIR COR ALIADA/INIMIGA
        ADD R1, 1
        MOVB R6, [R1]  ; LIGA/DESLIGA OS PIXELS DA NAVE
        MOV R7, MODELO_I2x     ; coordenadas dos Xs da nave
        MOV R8, MODELO_I2y     ; coordenadas dos Ys da nave
        CALL des_obj
        RET

desenha_inim3:
        MOV R1, INIM3_xy
        MOVB R2, [R1]  ; X DA INIMIGA
        ADD R1, 1
        MOVB R3, [R1]  ; Y DA INIMIGA
        ADD R1, 1
        MOVB R4, [R1]  ; NUMERO DE PIXELS DA INIMIGA
        ADD R1, 1
        MOVB R5, [R1]  ; DECIDIR COR ALIADA/INIMIGA
        ADD R1, 1
        MOVB R6, [R1]  ; LIGA/DESLIGA OS PIXELS DA NAVE
        MOV R7, MODELO_I3x     ; coordenadas dos Xs da nave
        MOV R8, MODELO_I3y     ; coordenadas dos Ys da nave
        CALL des_obj
        RET



;##########################################;
;#                          Background                                    #;
;##########################################;
; rotina que coloca o cenario dentro do pixelscreen
; recebe a variavel de seleçao dos cenarios
backg:
        PUSH R1
```

```
        PUSH R2
        PUSH R3

        MOV R1, SWITCH_BACKG          ; verifica com que valor se econtra a variavel que
decide
        MOV R2, [R1]                          ; o cenario ser tocado
        CMP R2, 1
        JZ ENDGAME                                    ; fim do jogo
        CMP R2, 0
        JZ NORMAL                                     ; cenario de jogo
        CMP R2, 3
        JZ WIN                                        ; vitoria

        MOV R1, ECRA          ; endereço do ecra
        MOV R2, 2H                    ; cenario do start
        MOV R3, 0EH                   ; endereco do fundo
        MOV [R1 + R3], R2     ; mete a imagem dentro ecra
        JMP FIM_BACKG

        NORMAL:
        MOV R1, ECRA          ; endereço do ecra
        MOV R2, 0H                    ; cenario principal ou seja primeira imagem
        MOV R3, 0EH                   ; endereco do fundo
        MOV [R1 + R3], R2     ; mete a imagem dentro ecra
        JMP FIM_BACKG

        ENDGAME:
        MOV R1, ECRA          ; endereço do ecra
        MOV R2, 1H                    ; cenario de loss
        MOV R3, 0EH                   ; endereco do fundo
        MOV [R1 + R3], R2     ; mete a imagem dentro ecra
        JMP FIM_BACKG

        WIN:
```

```
        MOV R1, ECRA          ; endereço do ecra
        MOV R2, 3H                    ; cenario do vitorioso
        MOV R3, 0EH                   ; endereco do fundo
        MOV [R1 + R3], R2     ; mete a imagem dentro ecra
        JMP FIM_BACKG


        FIM_BACKG:
        POP R3
        POP R2
        POP R1
        RET


;###################################
;#                   Sons                          #
;###################################
; rotinas para serem emitidos sons
; recebe a variavel de selecao dos sons


escolha_som:
        PUSH R8
        PUSH R9
        PUSH R10


        MOV R8, SWITCH_SOM            ; verifica como esta a variavel de selecao
        MOV R9, [R8]
        CMP R9, 0
        JZ DISPARO
        CMP R9, 2
        JZ COL_BALA
        CMP R9, 3
        JZ END_SOUND
        CMP R9, 4
        JZ WINWIN
```

```
MOV R8, SONORO ; coloca o endereco dos sons em r9
MOV R9, 1H             ; 1 é o som inicial
MOV [R8], R9   ; toca o som
JMP FIM_SOM

DISPARO:
MOV R8, SONORO ; coloca o endereco dos sons em r9
MOV R9, 0H             ; 0 é o som dos disparos
MOV [R8], R9   ; toca o som
JMP FIM_SOM

WINWIN:
MOV R8, SONORO ; coloca o endereco dos sons em r9
MOV R9, 4H             ; 4 é o som da sweet victory
MOV [R8], R9   ; toca o som
JMP FIM_SOM

COL_BALA:
MOV R8, SONORO ; coloca o endereco dos sons em r9
MOV R9, 2H             ; 2 é o som das colisoes entre nave e balas
MOV [R8], R9   ; toca o som
JMP FIM_SOM

END_SOUND:
MOV R8, SONORO ; coloca o endereco dos sons em r9
MOV R9, 3H             ; 3 é o som de fim do jogo
MOV [R8], R9   ; toca o som
JMP FIM_SOM

FIM_SOM:
POP R10
POP R9
```

```
        POP R8

        RET


;##################################

;#                      fim                     #

;##################################

; rotinas para trocados os elementos do jogo para mostrar que o jogador ganhou ou perdeu
o jogo

; sao apenas chamadas paara acabar o jogo, pelo que o jogo tera de ser reiniciado


acabar_o_jogo:                ; acabar o jogo com ua derrota

        PUSH R1

        PUSH R2


        MOV R1, ACABA_JOGO            ; muda a variavel de estado que faz com que o
programa entre num ciclo infinito e seja necessario dar restart

        MOVB R2, [R1]

        SUB R2, 1

        MOVB [R1], R2

        MOV R1, ativa_int_bala ; muda a variavel de estado que ira desligar a interrupcao
das balas

        MOV R2, [R1]

        MOV R2, 0

        MOV [R1], R2

        MOV R1, ativa_int_inim ; muda a variavel de estado que ira desligar a interrupcao
das inimigas

        MOV R2, [R1]

        MOV R2, 0

        MOV [R1], R2

        MOV R1, SWITCH_BACKG   ; troca a variavel de verificacao de cenarios para mostrar
o da derrota

        MOV R2, [R1]

        MOV R2, 1

        MOV [R1], R2

        CALL backg

        MOV R1, SWITCH_SOM            ; muda a variavel de verificacao do som de forma a
tocar a som da derrota
```

```
        MOV R2, 3

        MOV [R1], R2

        CALL escolha_som


        POP R2

        POP R1

        RET


WIN_acabar_o_jogo:                  ; acabar o jogo com uma vitoria

        PUSH R1

        PUSH R2

        CALL desliga_INIM                   ; desliga as naves

        CALL desliga_INIM2

        CALL desliga_INIM3

        CALL desliga_NAVE

        MOV R1, ACABA_JOGO                  ; muda a variavel de estado que faz com que
o programa entre num ciclo infinito e seja necessario dar restart

        MOVB R2, [R1]

        SUB R2, 1

        MOVB [R1], R2

        MOV R1, ativa_int_bala        ; muda a variavel de estado que ira desligar a
interrupcao das balas

        MOV R2, [R1]

        MOV R2, 0

        MOV [R1], R2

        MOV R1, ativa_int_inim        ; muda a variavel de estado que ira desligar a
interrupcao das inimigas

        MOV R2, [R1]

        MOV R2, 0

        MOV [R1], R2

        MOV R1, SWITCH_BACKG          ; troca a variavel de verificacao de cenarios para
mostrar o da vitoria

        MOV R2, [R1]

        MOV R2, 3

        MOV [R1], R2

        CALL backg
```

```
        MOV R1, SWITCH_SOM                  ; muda a variavel de verificacao do som de
forma a tocar a som da vitoria

        MOV R2, 4

        MOV [R1], R2

        CALL escolha_som


        POP R2

        POP R1

        RET




;#######################################;

;#                    Ligar/Desligar Nave                  #;

;#######################################;

; estas rotinas servem como interruptores para acender ou desligar os pixels da nave

; para que estas possam ser desenhadas em outros lugares

liga_NAVE:

        PUSH R1

        PUSH R2

        MOV R1, 4

        MOV R2, NAVE_xy

        ADD R2, R1                    ; escolher o atributo de ligar/desligar dentro da
string

        MOV R1, 1                     ; ligar o "switch"

        MOVB [R2], R1          ; mete-lo na string

        CALL desenha_nave

        POP R2

        POP R1

        RET


desliga_NAVE:

        PUSH R1

        PUSH R2

        MOV R1, 4
```

```
        MOV R2, NAVE_xy
        ADD R2, R1                  ; escolher o atributo de ligar/desligar dentro da
string
        MOV R1, 0                   ; desligar o "switch"
        MOVB [R2], R1          ; mete-lo na string
        CALL desenha_nave
        POP R2
        POP R1
        RET


liga_INIM:
        PUSH R1
        PUSH R2
        MOV R1, 4
        MOV R2, INIM_xy
        ADD R2, R1                  ; escolher o atributo de ligar/desligar dentro da
string
        MOV R1, 1                   ; ligar o "switch"
        MOVB [R2], R1          ; mete-lo na string
        CALL desenha_inim
        POP R2
        POP R1
        RET


desliga_INIM:
        PUSH R1
        PUSH R2
        MOV R1, 4
        MOV R2, INIM_xy
        ADD R2, R1                  ; escolher o atributo de ligar/desligar dentro da
string
        MOV R1, 0                   ; desligar o "switch"
        MOVB [R2], R1          ; mete-lo na string
        CALL desenha_inim
        POP R2
```

```
        POP R1
        RET


liga_INIM2:
        PUSH R1
        PUSH R2
        MOV R1, 4
        MOV R2, INIM2_xy
        ADD R2, R1                      ; escolher o atributo de ligar/desligar dentro da
string
        MOV R1, 1                       ; ligar o "switch"
        MOVB [R2], R1           ; mete-lo na string
        CALL desenha_inim2
        POP R2
        POP R1
        RET


desliga_INIM2:
        PUSH R1
        PUSH R2
        MOV R1, 4
        MOV R2, INIM2_xy
        ADD R2, R1                      ; escolher o atributo de ligar/desligar dentro da
string
        MOV R1, 0                       ; desligar o "switch"
        MOVB [R2], R1           ; mete-lo na string
        CALL desenha_inim2
        POP R2
        POP R1
        RET


liga_INIM3:
        PUSH R1
        PUSH R2
```

```
        MOV R1, 4
        MOV R2, INIM3_xy
        ADD R2, R1                      ; escolher o atributo de ligar/desligar dentro da
string
        MOV R1, 1                       ; ligar o "switch"
        MOVB [R2], R1           ; mete-lo na string
        CALL desenha_inim3
        POP R2
        POP R1
        RET


desliga_INIM3:
        PUSH R1
        PUSH R2
        MOV R1, 4
        MOV R2, INIM3_xy
        ADD R2, R1                      ; escolher o atributo de ligar/desligar dentro da
string
        MOV R1, 0                       ; desligar o "switch"
        MOVB [R2], R1           ; mete-lo na string
        CALL desenha_inim3
        POP R2
        POP R1
        RET


;################################################
;#             Desenhar o objeto no pixel screen            #
;################################################
; rotina generalizada para desenhar qualquer objeto no pixelscreen
; utiliza todos os atributos separados anteriormente
des_obj:
        MOVB R1, [R7]           ; primeiro numero a ser adicionado ao referencial x
        ADD R7, 1                       ; proximo numero a ser adicionado
        ADD R2, R1                      ; adiciona ao referencial x
```

```
        MOVB R0, [R8]          ; primeiro numero a ser adicionado ao referncial y
        ADD R8, 1                       ; proximo numero a ser adicionado
        ADD R3, R0                      ; adiciona ao referencial y

        MOV R9, PIXEL_X                 ; mete o endereco dos X's em R9
        MOV [R9], R2           ; mete a coordenada no endereço
        MOV R9, PIXEL_Y                 ; mete o endereco  dos Y's em R9
        MOV [R9], R3           ; mete a coordenada no endereço
        CALL cor_obj           ; chama a rotina para decidir a cor da nave

        MOV R9, ESTADO_PIXEL   ; coloca o endereço que serve para acender o pixel em R9
        MOV [R9], R6                    ; coloca o "switch" ligado ou desligado em R9

        SUB R4, 1                               ; subtrai o numero de pixeis
        SUB R2, R1                              ; subtrai o que foi adicionado no inicio
para que o referencial x não mude
        SUB R3, R0                              ; subtrai o que foi adicionado no inicio
para que o referencial y não mude
        CMP R4, 0                               ; verifica se já nao ha mais pixeis para
serem ligados
        JZ fim_d                                ; se não houverem, acaba
        JMP des_obj                             ; se houverem volta ao inicio
fim_d:
        RET

cor_obj:                                ; rotina para selecionar a cor
        PUSH R9
        PUSH R4
        PUSH R5
        CMP R5, 0                       ; le o identificador de aliado/inimigo
        JZ cor_inim                     ; se for 0 a nave será vermelha
        JMP cor_nave           ; se for 1 a nave será azul
cor_nave:
        MOV     R5, ECRA                ; coloca em R5 o endereço que escolhe a cor
vermelha
        MOV R9, 0                       ; coloca a cor vermelha com valor 0
```

```
        MOV [R5], R9            ; mete o valor da cor dentro do endereço

        MOV R5, PIX_B           ; coloca em R5 o endereço que escolhe a cor azul

        MOV R9, 255                     ; coloca a cor azul com valor maximo

        MOV [R5], R9            ; mete o valor da cor dentro do endereço

        MOV     R5, PIX_G               ;  coloca  em  R5  o  endereço  que  escolhe  a  cor
vermelha

        MOV R9, 0                       ; coloca a cor vermelha com valor 0

        MOV [R5], R9            ; mete o valor da cor dentro do endereço

        JMP fim_c

cor_inim:

        MOV     R5, PIX_B               ; coloca em R5 o endereço que escolhe a cor azul

        MOV R9, 0                       ; coloca a cor azul com valor 0

        MOV [R5], R9            ; mete o valor da cor dentro do endereço

        MOV R5, ECRA            ; coloca em R5 o endereço que escolhe a cor vermelha

        MOV R9, 255                     ; coloca a cor vermelha com valor maximo

        MOV [R5], R9            ; mete o valor da cor dentro do endereço

        MOV [R5], R9            ; mete o valor da cor dentro do endereço

        MOV     R5, PIX_G               ;  coloca  em  R5  o  endereço  que  escolhe  a  cor
vermelha

        MOV R9, 0                       ; coloca a cor vermelha com valor 0

        JMP fim_c

fim_c:

        POP R5

        POP R4

        POP R9

        RET



;#############################################################################

; desenhar a primeira posicso da bala, e colocar as coordenadsa iniciais


des_BALA:

        PUSH R1

        PUSH R2
```

```
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH R6
        PUSH R9

        MOV R1, BALA1          ; apaga a bala antes de a disparar
        MOVB R2, [R1]          ; de forma a nao ficar na ultima posicao
        ADD R1, 1                      ; que foi desenhada
        MOVB R3, [R1]

        MOV R9, PIXEL_X               ; mete o endereco dos X's em R9
        MOV [R9], R2           ; mete a coordenada no endereço
        MOV R9, PIXEL_Y               ; mete o endereco  dos Y's em R9
        MOV [R9], R3           ; mete a coordenada no endereço
        MOV R9, ESTADO_PIXEL
        MOV R6, 0
        MOV [R9], R6                   ; coloca o "switch" ligado ou desligado em R9

        MOV R1, SWITCH_SOM     ; escolher o som a ser tocado (disparo)
        MOV R2, 0
        MOV [R1], R2
        CALL escolha_som

        MOV R1, NAVE_xy               ; inicializa as strings a serem usadas
        MOV R4, BALA1

        MOVB R2, [R1]                  ;X DA BALA
        ADD R2, 2                              ; adiociona se 2 de forma a bala sair
centrada em relacao ao modelo da nave
        ADD R1, 1
        MOVB R3, [R1]                  ; Y DA BALA

        MOV R9, PIXEL_X               ; mete o endereco dos X's em R9
```

```
        MOV [R9], R2          ; mete a coordenada no endereço
        MOV R9, PIXEL_Y               ; mete o endereco  dos Y's em R9
        MOV [R9], R3          ; mete a coordenada no endereço
        CALL cor_obj_BALA             ; chama a rotina para decidir a cor da bala

        MOV R9, ESTADO_PIXEL
        MOV R6, 1
        MOV [R9], R6                  ; coloca o "switch" ligado ou desligado em R9
        MOVB [R4], R2                 ; volta a colocar coordenadas na string BALA1
        ADD R4, 1
        MOVB [R4], R3
        ADD R4, 1
        MOV R9, 1
        MOVB [R4], R9

        MOV R1, bala_ativa            ; coloca na variavel de estado que existe uma bala
ativa
        MOV R2, 1
        MOVB [R1], R2
        JMP fim_d_BALA



fim_d_BALA:

        POP R9
        POP R6
        POP R5
        POP R4
        POP R3
        POP R2
        POP R1
        RET

cor_obj_BALA:                         ; rotina para selecionar a cor
```

```
        PUSH R9

        PUSH R8

        MOV     R8, ECRA              ;  coloca  em  R5  o  endereço  que  escolhe  a  cor
vermelha

        MOV R9, 0                     ; coloca a cor vermelha com valor 0

        MOV [R8], R9          ; mete o valor da cor dentro do endereço

        MOV R8, PIX_B         ; coloca em R5 o endereço que escolhe a cor azul

        MOV R9, 255                   ; coloca a cor azul com valor maximo

        MOV [R8], R9          ; mete o valor da cor dentro do endereço

        MOV R8, PIX_G         ; coloca em R5 o endereço que escolhe a cor verde

        MOV R9, 255                   ; coloca a cor azul com valor maximo

        MOV [R8], R9          ; mete o valor da cor dentro do endereço

        POP R8

        POP R9

        RET



;#########################################################################

;#             Rotina para decidir a acao a ser realizada por cada tecla      #

;#########################################################################

; rotina que decide a funcao de cada tecla do teclado

; ira fazer uso das variveis TECLA_X de forma a achar qual tecla foi premida

; na funcao teclado. utiliza CMPs e JMPs para funcoes auxiliares que vao chamar as
rotinas


funcao_tecla:

        PUSH R1

        PUSH R6


        MOV R1, TECLA_0                             ; move a tecla a ser comparada com o
R& que tem a tecla armazenada

        CMP R6, R1

        JZ chama_movimento                  ;  salta  para  o  bloco  que  faz  a  call  da
funcao a ser chamada


        MOV R1, TECLA_1
```

```
CMP R6, R1
JZ chama_movimento

MOV R1, TECLA_2
CMP R6, R1
JZ chama_movimento

MOV R1, TECLA_4
CMP R6, R1
JZ chama_movimento

MOV R1, TECLA_6
CMP R6, R1
JZ chama_movimento

MOV R1, TECLA_8
CMP R6, R1
JZ chama_movimento

MOV R1, TECLA_9
CMP R6, R1
JZ chama_movimento

MOV R1, TECLA_A
CMP R6, R1
JZ chama_movimento

MOV R1, TECLA_2
CMP R6, R1
JZ chama_movimento

MOV R1, TECLA_5
CMP R6, R1
```

```
        JZ chama_bala


        MOV R1, TECLA_B
        CMP R6, R1
        JZ pausa_das_int

        MOV R1, TECLA_7
        CMP R6, R1
        JZ chama_fim_jogo



        fim_funcao:
                POP R6
                POP R1
                RET

;#############################################
; funcoes auxiliares de chamada de CALLs       #
;#############################################
; depois de serem executadas voltam parao fim da funcao_tecla
chama_fim_jogo:
        CALL acabar_o_jogo
        JMP fim_funcao



pausa_das_int:

        CALL liga_int_inim_balas
        CALL ha_tecla
        JMP fim_funcao

chama_scb:
        CALL scoreboard_ad_sub
```

```
        JMP fim_funcao



chama_bala:
        CALL des_BALA
        CALL ha_tecla                   ; para o som nao repetir e sair apenas uma bala
        JMP fim_funcao


chama_movimento:
        CALL rot_movimento
        CALL rot_movimento
        CALL ha_tecla
        CALL atraso
        JMP fim_funcao



atraso:
        PUSH R1
        PUSH R2
        MOV R1, 4000
        A:
                SUB R1, 1
                CMP R1, 0
                JZ fim_atraso
                JMP A
        fim_atraso:
        POP R2
        POP R1
        RET


;################################

; rotina que apaga a ultima posicao da nave e reescreve numa nova posicao conforme a
tecla pressionada

; recebe as coordenadas na nave e a string do movimento
```

```
rot_movimento:
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH R6
        PUSH R7
        CALL desliga_NAVE      ; DESLIGA A POSICAO ATUAL DA NAVE
        MOV R0, NAVE_xy    ; TEM AS COORDENADAS ORIGINAIS


        MOV R3, MOVIMENTO      ; STRING DO QUE VAI SER SOMADO AS COORDENADAS ORIGINAIS
        SHL R6, 1                       ; LETRA*2 PARA IR AO ENCONTRO DA POSICAO DENTRO DA
STRING
        ADD R3, R6                      ; IR A POSICAO NA STRING


        MOVB R1, [R0]          ; X DAS COORDENADAS ORIGINAIS
        MOVB R4, [R3]          ; R4 TERA O QUE VAI SER ADICIONADO A COORDENADA X
        CMP R6, 0                       ;  COMPARACOES  PARA  VERIFICAR  SE  A  TECLA  VAI
ADICIONAR OU SUBTRAIR À POSICAO ATUAL
        JZ SU                           ; VALOR DA TECLA * 2, PARA IR PARA A STRING DO
MOVIMENTO
        MOV R7, 8
        CMP R6, R7
        JZ SU
        MOV R7, 16
        CMP R6, R7
        JZ SU
        ADD R1, R4                      ; NOVA COORDENADA DO X
        JMP CONT
        SU:
                SUB R1, 1


        CONT:
```

```
        ADD R0, 1               ;VA PARA O Y

        ADD R3, 1               ; ANDAR UMA POSICAO NA STRING


        MOVB R2, [R0]           ; Y DAS COORDENADAS ORIGINAIS

        MOVB R5, [R3]           ; R5 TERA O QUE VAI SER ADICIONADO A COORDENADA     Y

        CMP R6, 0                       ;  COMPARACOES  PARA  VERIFICAR  SE  A  TECLA  VAI
ADICIONAR OU SUBTRAIR À POSICAO ATUAL

        JZ SU2                          ; VALOR DA TECLA * 2, PARA IR PARA A STRING DO
MOVIMENTO

        CMP R6, 2

        JZ SU2

        CMP R6, 4

        JZ SU2

        ADD R2, R5                      ; NOVA COORDENADA DO Y

        JMP check_limites

        SU2:

                SUB R2, 1

        check_limites: ; 0 <= X <= 56    0 <= Y <= 25               devido  a  largura  e
altura da nave

                MOV R7, L_ESQ_SUP

                CMP R1, R7

                JN fim_mo


                MOV R7, L_ESQ_SUP

                CMP R2, R7

                JN fim_mo




                MOV R7, 27

                CMP R2, R7      ; Y = 26 - 26

                JZ fim_mo


                MOV R7, 60

                CMP R1, R7
```

```
                JZ fim_mo



        MOV R0, NAVE_xy     ; TEM AS COORDENADAS ORIGINAIS
        MOVB [R0], R1       ; SUBSTITUI COM AS NOVAS
        ADD R0, 1
        MOVB [R0], R2
        fim_mo:
                CALL liga_NAVE
                CALL VERIFICAR_NAVE_INIM
                CALL VERIFICAR_NAVE_INIMB
                CALL VERIFICAR_NAVE_INIMC
                CALL VERIFICAR_NAVE_INIMD
                POP R7
                POP R6
                POP R5
                POP R4
                POP R3
                POP R2
                POP R1
                POP R0
                RET


;###################################
;#             Funções do contador      #
;###################################
; rotinas para o contador que aumenta 5 e diminui 10 da variavel CONTADOR
; vao buscar o valor à variavel, faz a devida operacao, volta a coloca-lo dentro da
variavel
; e dá display do novo valor



scoreboard_ad_sub:
        PUSH R0
```

```
        PUSH R1                         ; tecla
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH R6
        MOV R0, DISPLAYS            ; endereço dos displays
        MOV R6, CONTADOR            ; variavel do contador
        MOV R2, [R6]                ; R2 tem o valor atual do contador
        CMP R1, 3
        JZ adic_5
        JMP subt_10
        adic_5:
                MOV R3, CONTADOR_MAX   ; R3 tem o valor maximo do contador
                CMP R2, R3                      ; verifica se o contador está no
maximo
                JZ score                        ; se estiver acaba a rotina
                MOV R3, 95H
                CMP R2, R3
                JZ  score_95                ; no caso de ser 95, mete o valor
"manualmente"
                MOV R4, 0FH                     ; "mascara" para verificar se o nr
acaba em 5
                AND R4, R2                      ; ao fazer AND entre os dois valores
apenas vao ficar ligados os bits de menor valor
                CMP R4, 5H                      ; que servem para a comparação
                JZ acaba_5
                JMP acaba_0                     ; se o numero acabar em 0, basta uma
soma normal
        score_95:
                MOV R5, 100H                ; meter o 100 manualmente
                MOV R2, R5                      ; meter o valor dentro de R2
                JMP score
        acaba_5:
                MOV R5, 0BH                     ; R5 tem o valor 0BH, pois o
contador nao converte valores, tendo na mesma
```

```
                ADD R2, R5                          ;   de   somar   os   valores   em
hexadecimal, para que no contador, o que é mostrado nos

                JMP score                           ;  displays  seja  0  na  esquerda  e
incremente 1 no do meio

        acaba_0:

                MOV R5, 5H                          ;  R5  tem  o  valor  a  adicionar  a
variavel

                ADD R2, R5

                JMP score

        subt_10:

                MOV R3, CONTADOR_MIN   ; R3 tem o valor minimo do contador

                CMP R2, R3                          ;  verifica  se  o  contador  está  no
minimo

                JZ score                            ; se estiver acaba a rotina

                MOV R3, CONTADOR_MAX   ; se o contador estiver no maximo inserimos o valor
manualmente

                CMP R2, R3

                JZ  score_100_10

                MOV R3, 5H                          ;  se  o  contador  estiver  no  minimo
inserimos o valor manualmente

                CMP R2, R3

                JZ score_5_10

                JMP sub_normal_10             ;  subtracao  normal  caso  nenhum  dos  caso
anteriores seja ativado

        score_100_10:

                MOV R5, 90H

                MOV R2, R5                          ; insere 90 na constante

                JMP score

        score_5_10:

                MOV R5, 0H                          ; insere 0 na constante

                MOV R2, R5

                JMP score

        sub_normal_10:

                MOV R5, 10H                         ; subtrai 10 ao contador

                SUB R2, R5

                JMP score

score:
```

```
        MOV [R6], R2                ; coloca o novo valor dentro da constante
        MOV [R0], R2                ; mostra no display

        MOV R1, 100H
        CMP R2, R1
        JZ PONTOS_100

        MOV R1, 0H
        CMP R2, R1
        JZ PERDER

        JMP FIM_SCB
        PERDER:                     ; se o contador esta a 0, perdeu
        CALL acabar_o_jogo
        JMP FIM_SCB
        PONTOS_100:                 ; se o contador esta a 100, ganhou
        CALL WIN_acabar_o_jogo
        FIM_SCB:
        POP R6
        POP R5
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET


;#########################################
;#            auxiliares INTERRUPCOES                      #
;#########################################
; rotinas relacionadas com as interrupcoes das balas e das naves inimigas
```

```
liga_int_inim_balas:          ; rotina que ativa/destiva as interrupcoes conforme os
valores das variaves de estado

        PUSH R0                             ; ativa_int_inim e  ativa_int_bala

        PUSH R1

        PUSH R2

        PUSH R3



        MOV R0, ativa_int_inim

        MOV R3, ativa_int_bala

        MOV R2, [R0]




        CMP R2, 0                                   ; caso a variavel esteja a 0, a
rotina vai ligar as interrupcoes

        JZ acionar_int                      ; so é necesario ver uma das variaveis de
estado pois as interrupcoes

        JMP desativa_int                    ; sao ligadas e desligadas ao mesmo tempo

        acionar_int:                        ; aciona as duas ao meter a variavel de
estado a 1

                MOV R1, 1

                MOV [R0], R1

                MOV [R3], R1


                MOV R1, SWITCH_BACKG   ; mete o background da pausa

                MOV R2, 0

                MOV [R1], R2

                CALL backg




                JMP fim_aciona

        desativa_int:                       ; desativa as duas ao meter a variavel de
estado a 0

                MOV R1, 0

                MOV [R0], R1

                MOV [R3], R1
```

```
                MOV R1, SWITCH_BACKG   ; mete o background normal
                MOV R2, 2
                MOV [R1], R2
                CALL backg

                JMP fim_aciona
        fim_aciona:
                POP R3
                POP R2
                POP R1
                POP R0
                RET

;##########################################
;#             INTERRUPCOES                                #
;##########################################

rot_int_0:                    ; executa a cada 0,3 segundos
        PUSH R0
        PUSH R1
    MOV  R0, ativa_int_bala

    MOV  R1, [R0]                    ; na interrupcao vai comparar a variavel de estado
e decidir se vai executar a rotina das balas
    CMP R1, 0
        JZ fim_int_0
        CALL VERIFICA_INT_B                  ; ativa a rotina de animacao das balas
        fim_int_0:
    POP  R1
    POP  R0
    RFE

rot_int_1:                    ; executa a cada 0,5 segundos
        PUSH R0
```

```
        PUSH R1
    MOV  R0, ativa_int_inim

    MOV  R1, [R0]                    ; na interrupcao vai comparar a variavel de estado
e decidir se vai executar a rotina das inimigas
    CMP R1, 0
        JZ fim_int_1
        CALL VERIFICA_INT_INIM_I      ; ativa a rotina de animacao das inimigas
        fim_int_1:
    POP  R1
    POP  R0
    RFE

;##########################################
;#             rotinas INTERRUPCOES                    #
;##########################################
; rotina que irá fazer com que as naves inimigas descam em diversas direcoes

VERIFICA_INT_INIM_I:
        PUSH R0
        PUSH R1
        PUSH R2

    MOV  R0, ativa_int_inim                  ; se a variavel de estado estiver a 1, as
naves inimigas sao animadas
        MOV R2, [R0]
        CMP R2, 1
        JZ CHAMA_ANIM_INIM
        JMP fim_ver_inim
CHAMA_ANIM_INIM:
        CALL anima_inim

fim_ver_inim:
        POP  R2
```

```
        POP  R1
    POP  R0
        RET

anima_inim:                                             ; rotina que vai apagar a
bala anterior, verificar colisao, mudar as coordenadas e redescreve-la
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH R6
                                        ; desliga as posicoes antigas
        CALL desliga_INIM
        CALL desliga_INIM2
        CALL desliga_INIM3
        MOV R0, INIM_xy  ; X , Y
        MOV R3, INIM2_xy
        MOV R5, INIM3_xy
        ADD R0, 1               ; coloca a adress nos Ys das naves

        MOVB R1, [R0]           ; Y das naves

        MOV R3, INIM2_xy
        MOV R5, INIM3_xy
        MOVB R2, [R3]           ;xs das naves laterais para as mover num angulo
        MOVB R4, [R5]

        ADD R1, 1                       ; soma 1 aos ys das naves
        ADD R2, 1                       ; soma 1 ao x da nave da esquerda
        SUB R4, 1                       ; subtrai 1 ao x da nave da direita
```

```
        MOV R6, N_LINHAS            ;  serve  para  verificar  se  chegou  ao  fim  do
pixelscreen

        CMP R1, R6

        JLT escreve                         ; nao chegou, continua normalmente

        CALL scoreboard_ad_sub

        MOV R1, 0                           ; chegou, vai colocar os ys e xs das naves
no inicio do teclado

        MOV R2, 0

        MOV R4, 58

        MOV R0, SWITCH_SOM

        MOV R3, 2

        MOV [R0], R3

        CALL escolha_som



escreve:

        MOV R0, INIM_xy  ; X , Y

        MOV R3, INIM2_xy

        MOV R5, INIM3_xy

        MOVB [R3], R2               ; muda os xs

        MOVB [R5], R4

        ADD R0, 1             ; Y

        ADD R3, 1

        ADD R5, 1

        MOVB [R0], R1               ; muda os ys

        MOVB [R3], R1

        MOVB [R5], R1


        CALL liga_INIM

        CALL liga_INIM2

        CALL liga_INIM3

        JMP fim


fim:
```

```
        CALL VERIFICAR_NAVE_INIM      ; verifica se houve colisao com as naves inimigas

        CALL VERIFICAR_NAVE_INIMB

        CALL VERIFICAR_NAVE_INIMC

        CALL VERIFICAR_NAVE_INIMD

        POP R6

        POP R5

        POP R4

        POP R3

        POP R2

        POP R1

        POP R0

        RET


;###############################################################################


VERIFICAR_NAVE_INIM:  ; vai compara as coordenadas das naves inimigas com a da nave aliada e aaveriguar se se sobrepoem

        PUSH R0                       ; nesta rotina será avaliado o ponto central das coordenadas, ou seja, o canto superior esquerdo

        PUSH R1

        PUSH R2

        PUSH R3

        PUSH R4

        PUSH R5


        MOV R0, NAVE_xy

        MOV R1, INIM_xy

        MOV R2, INIM2_xy

        MOV R3, INIM3_xy


        MOVB R4, [R0]          ; TEM O X DA NAVE

        MOVB R5, [R1]          ; TEM O X DA INIM DO MEIO


        CMP R4, R5                    ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE
```

```
        JN CONTINUAA

        ADD R5, 5

        CMP R5, R4                  ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUAA



        ADD R0, 1

        ADD     R1, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        MOVB R5, [R1]          ; TEM O Y DA INIM DO MEIO



        CMP R4, R5                  ;  SE  O  Y  DA  NAVE  FOR  MENOR  QUE  O  Y  DA  INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUAA

        ADD R5, 6

        CMP R5, R4                  ;  SE  O  Y  DA  NAVE  FOR  MAIOR  QUE  O  Y  DA  NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUAA

        JMP COLISAO_N          ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA DENTRO
DA INIM







        CONTINUAA:

        MOV R0, NAVE_xy



        MOVB R4, [R0]          ; TEM O X DA NAVE

        MOVB R5, [R2]          ; TEM O X DA INIM DA ESQ



        CMP R4, R5                  ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUAB

        ADD R5, 5

        CMP R5, R4                  ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUAB
```

```
        ADD R0, 1

        ADD     R2, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        MOVB R5, [R2]          ; TEM O Y DA INIM DO MEIO


        CMP R4, R5                    ; SE O Y DA NAVE FOR MENOR QUE O Y DA INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUAB

        ADD R5, 6

        CMP R5, R4                    ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUAB

        JMP COLISAO_N          ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA DENTRO
DA INIM




        CONTINUAB:

        MOV R0, NAVE_xy


        MOVB R4, [R0]          ; TEM O X DA NAVE

        MOVB R5, [R3]          ; TEM O X DA INIM DA ESQ


        CMP R4, R5                    ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUAC

        ADD R5, 5

        CMP R5, R4                    ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUAC


        ADD R0, 1

        ADD     R3, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        MOVB R5, [R3]          ; TEM O Y DA INIM DO MEIO


        CMP R4, R5                    ; SE O Y DA NAVE FOR MENOR QUE O Y DA INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM
```

```
        JN CONTINUAC

        ADD R5, 6

        CMP R5, R4                  ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUAC

        JMP COLISAO_N          ; SE NAO PASSOU NESTES CRITERIOS É PORQUE SE ENCONTRA
DENTRO DA INIM




        COLISAO_N:                  ; simplesmente acaba o jogo

        CALL acabar_o_jogo



CONTINUAC:

        POP R5

        POP R4

        POP R3

        POP R2

        POP R1

        POP R0

        RET


;#################################


VERIFICAR_NAVE_INIMB: ; vai comparar as coordenadas das naves inimigas com a da nave
aliada e averiguar se se sobrepoem

        PUSH R0                     ; nesta rotina será avaliado o canto superior
direito

        PUSH R1

        PUSH R2

        PUSH R3

        PUSH R4

        PUSH R5

        PUSH R6

        MOV R6, 4                   ; ao adicionar 4 ao X da coordenada principal
iremos comparar o canto direito
```

```
        MOV R0, NAVE_xy

        MOV R1, INIM_xy

        MOV R2, INIM2_xy

        MOV R3, INIM3_xy


        MOVB R4, [R0]          ; TEM O X DA NAVE

        ADD R4, R6

        MOVB R5, [R1]          ; TEM O X DA INIM DO MEIO


        CMP R4, R5                    ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUAAB

        ADD R5, 5

        CMP R5, R4                    ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUAAB


        ADD R0, 1

        ADD     R1, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        MOVB R5, [R1]          ; TEM O Y DA INIM DO MEIO


        CMP R4, R5                    ; SE O Y DA NAVE FOR MENOR QUE O Y DA INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUAAB

        ADD R5, 6

        CMP R5, R4                    ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUAAB

        JMP COLISAO_NB         ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA DENTRO
DA INIM




        CONTINUAAB:

        MOV R0, NAVE_xy
```

```
        MOVB R4, [R0]          ; TEM O X DA NAVE

        ADD R4, R6

        MOVB R5, [R2]          ; TEM O X DA INIM DA ESQ


        CMP R4, R5                    ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUABB

        ADD R5, 5

        CMP R5, R4                    ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUABB


        ADD R0, 1

        ADD     R2, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        MOVB R5, [R2]          ; TEM O Y DA INIM DO MEIO


        CMP R4, R5                    ; SE O Y DA NAVE FOR MENOR QUE O Y DA INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUABB

        ADD R5, 6

        CMP R5, R4                    ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUABB

        JMP COLISAO_NB         ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA DENTRO
DA INIM



        CONTINUABB:

        MOV R0, NAVE_xy


        MOVB R4, [R0]          ; TEM O X DA NAVE

        ADD R4, R6

        MOVB R5, [R3]          ; TEM O X DA INIM DA ESQ
```

```
        CMP R4, R5                  ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUACB

        ADD R5, 5

        CMP R5, R4                  ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUACB



        ADD R0, 1

        ADD     R3, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        MOVB R5, [R3]          ; TEM O Y DA INIM DO MEIO



        CMP R4, R5                  ; SE O Y DA NAVE FOR MENOR QUE O Y DA INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUACB

        ADD R5, 6

        CMP R5, R4                  ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUACB

        JMP COLISAO_NB         ; SE NAO PASSOU NESTES CRITERIOS É PORQUE SE ENCONTRA
DENTRO DA INIM




        COLISAO_NB:                 ; simplesmente acaba o jogo

        CALL acabar_o_jogo




CONTINUACB:

        POP R6

        POP R5

        POP R4

        POP R3

        POP R2

        POP R1
```

```
        POP R0

        RET


;#################################################################################


VERIFICAR_NAVE_INIMC: ; vai compara as coordenadas das naves inimigas com a da nave aliada e aaveriguar se se sobrepoem

        PUSH R0                         ; nesta rotina será avaliado o canto inferior esquerdo

        PUSH R1

        PUSH R2

        PUSH R3

        PUSH R4

        PUSH R5

        PUSH R6

        MOV R6, 5                       ; ao adicionar 5 a coordenada principal no y iremos comparar o canto inferior esquerdo

        MOV R0, NAVE_xy

        MOV R1, INIM_xy

        MOV R2, INIM2_xy

        MOV R3, INIM3_xy


        MOVB R4, [R0]           ; TEM O X DA NAVE

        MOVB R5, [R1]           ; TEM O X DA INIM DO MEIO


        CMP R4, R5                      ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUAAC

        ADD R5, 5

        CMP R5, R4                      ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA A NAVE ESTA A DIREITA DA INIM

        JN CONTINUAAC


        ADD R0, 1
```

```
        ADD     R1, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        ADD R4, R6

        MOVB R5, [R1]          ; TEM O Y DA INIM DO MEIO


        CMP R4, R5                    ; SE O Y DA NAVE FOR MENOR QUE O Y DA INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUAAC

        ADD R5, 6

        CMP R5, R4                    ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUAAC

        JMP COLISAO_NC         ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA DENTRO
DA INIM




        CONTINUAAC:

        MOV R0, NAVE_xy


        MOVB R4, [R0]          ; TEM O X DA NAVE

        MOVB R5, [R2]          ; TEM O X DA INIM DA ESQ


        CMP R4, R5                    ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUABC

        ADD R5, 5

        CMP R5, R4                    ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUABC


        ADD R0, 1

        ADD     R2, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        ADD R4, R6

        MOVB R5, [R2]          ; TEM O Y DA INIM DO MEIO
```

```
        CMP R4, R5                      ; SE O Y DA NAVE FOR MENOR QUE O Y DA NAVE
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUABC

        ADD R5, 6

        CMP R5, R4                      ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUABC

        JMP COLISAO_NC          ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA DENTRO
DA INIM




        CONTINUABC:

        MOV R0, NAVE_xy


        MOVB R4, [R0]           ; TEM O X DA NAVE

        MOVB R5, [R3]           ; TEM O X DA INIM DA ESQ


        CMP R4, R5                      ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUACC

        ADD R5, 5

        CMP R5, R4                      ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUACC


        ADD R0, 1

        ADD     R3, 1

        MOVB R4, [R0]           ; TEM O Y DA NAVE

        ADD R4, R6

        MOVB R5, [R3]           ; TEM O Y DA INIM DO MEIO


        CMP R4, R5                      ; SE O Y DA NAVE FOR MENOR QUE O Y DA INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUACC

        ADD R5, 6
```

```
        CMP R5, R4                      ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUACC

        JMP COLISAO_NC          ; SE NAO PASSOU NESTES CRITERIOS É PORQUE SE ENCONTRA
DENTRO DA INIM




        COLISAO_NC:                     ; simplesmente acaba o jogo

        CALL acabar_o_jogo




CONTINUACC:

        POP R6

        POP R5

        POP R4

        POP R3

        POP R2

        POP R1

        POP R0

        RET


;#################################################


VERIFICAR_NAVE_INIMD: ; vai compara as coordenadas das naves inimigas com a da nave
aliada e aaveriguar se se sobrepoem

        PUSH R0                         ; nesta rotina será avaliado o canto inferior
direito

        PUSH R1

        PUSH R2

        PUSH R3

        PUSH R4

        PUSH R5

        PUSH R6

        PUSH R7
```

```
        MOV R6, 4                       ; ao adicionar 4 a coordenada do x e 4 a coordenada
do y obtem-se o canto inferior esquerdo

        MOV R7, 5

        MOV R0, NAVE_xy

        MOV R1, INIM_xy

        MOV R2, INIM2_xy

        MOV R3, INIM3_xy


        MOVB R4, [R0]           ; TEM O X DA NAVE

        ADD R4, R6

        MOVB R5, [R1]           ; TEM O X DA INIM DO MEIO


        CMP R4, R5                      ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUAAD

        ADD R5, 5

        CMP R5, R4                      ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUAAD


        ADD R0, 1

        ADD     R1, 1

        MOVB R4, [R0]           ; TEM O Y DA NAVE

        ADD R4, R7

        MOVB R5, [R1]           ; TEM O Y DA INIM DO MEIO


        CMP R4, R5                      ; SE O Y DA NAVE FOR MENOR QUE O Y DA INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUAAD

        ADD R5, 6

        CMP R5, R4                      ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUAAD

        JMP COLISAO_ND          ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA DENTRO
DA INIM
```

```
        CONTINUAAD:

        MOV R0, NAVE_xy


        MOVB R4, [R0]          ; TEM O X DA NAVE

        ADD R4, R6

        MOVB R5, [R2]          ; TEM O X DA INIM DA ESQ


        CMP R4, R5                    ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUABD

        ADD R5, 5

        CMP R5, R4                    ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUABD


        ADD R0, 1

        ADD    R2, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        ADD R4, R7

        MOVB R5, [R2]          ; TEM O Y DA INIM DO MEIO


        CMP R4, R5                    ; SE O Y DA NAVE FOR MENOR QUE O Y DA INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUABD

        ADD R5, 6

        CMP R5, R4                    ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUABD

        JMP COLISAO_ND         ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA DENTRO
DA INIM



        CONTINUABD:

        MOV R0, NAVE_xy
```

```
        MOVB R4, [R0]          ; TEM O X DA NAVE

        ADD R4, R6

        MOVB R5, [R3]          ; TEM O X DA INIM DA ESQ


        CMP R4, R5                    ; SE X DA NAVE - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A NAVE ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUACD

        ADD R5, 5

        CMP R5, R4                    ; SE X DA INIM+5 - X DA NAVE FOR NEGATIVO SIGNIFICA
A NAVE ESTA A DIREITA DA INIM

        JN CONTINUACD


        ADD R0, 1

        ADD     R3, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        ADD R4, R7

        MOVB R5, [R3]          ; TEM O Y DA INIM DO MEIO


        CMP R4, R5                    ; SE O Y DA NAVE FOR MENOR QUE O Y DA INIM
SIGNIFICA QUE ESTA POR CIMA DA INIM

        JN CONTINUACD

        ADD R5, 6

        CMP R5, R4                    ; SE O Y DA NAVE FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA INIM

        JN CONTINUACD

        JMP COLISAO_ND         ; SE NAO PASSOU NESTES CRITERIOS É PORQUE SE ENCONTRA
DENTRO DA INIM




        COLISAO_ND:                   ; simplesmente acaba o jogo

        CALL acabar_o_jogo




CONTINUACD:

        POP R7
```

```
        POP R6
        POP R5
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET


;#############################################################################################
; rotina que irá fazer com que as balas subam até chegar ao y = 0 ou embater num inimigo


VERIFICA_INT_B:
        PUSH R0
        PUSH R1
        PUSH R2
    MOV  R0, ativa_int_bala          ; se a variavel de estado estiver a 1, as balas
serao animadas
        MOV R2, [R0]
        CMP R2, 1
        JZ CHAMA_ANIM_BALA
        JMP fim_ver_bala


CHAMA_ANIM_BALA:                            ; verifica se a bala esta ativa
        MOV R0, BALA1
        MOVB R1, [R0]
        CMP R1, 0
        JZ fim_ver_bala
        CALL anima_bala
fim_ver_bala:
        POP  R2
        POP  R1
    POP  R0
        RET
```

```
anima_bala:
        PUSH R0
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        PUSH R5
        PUSH R6
        PUSH R7
        PUSH R8
        PUSH R9
        MOV R5, 0
        MOV R6, 1
        MOV R0, BALA1                   ;R0 TEM A STRING DAS BALAS
        MOV R3, R0                              ; COPIA DA STRINGF DAS BALAS
        ADD R0, 2
        MOVB R1, [R0]                   ; R1 TEM O SWITCH DA BALA
        CMP R1, 0                               ;VERIIFCAR SE JA FOI INICIALIZADA
        JZ fim_anima_bala_anima

        MOVB R1, [R3]                   ; R1 TEM O X DA BALA
        ADD R3, 1
        MOVB R2, [R3]                   ; R2 TEM O Y DA BALA

        MOV R9, PIXEL_X                 ; mete o endereco dos X's em R9
        MOV [R9], R1            ; mete a coordenada no endereço
        MOV R9, PIXEL_Y                 ; mete o endereco  dos Y's em R9
        MOV [R9], R2            ; mete a coordenada no endereço
        MOV R4, ESTADO_PIXEL
        MOV [R4], R5                            ; DESLIGAR A BALA ANTERIOR

        SUB R2, 1
```

```
CMP     R2, 0
JZ APAGA_BALA



MOV R9, PIXEL_X               ; mete o endereco dos X's em R9
MOV [R9], R1           ; mete a coordenada no endereço
MOV R9, PIXEL_Y               ; mete o endereco  dos Y's em R9
MOV [R9], R2           ; mete a coordenada no endereço
CALL cor_ANIMA_BALA
MOV R4, ESTADO_PIXEL
MOV [R4], R6                          ; LIGA A BALA
MOVB [R0], R6
SUB R0, 1
MOVB [R0], R2
SUB R0, 1
MOVB [R0], R1
CALL VERIFICAR_BALA_INIM
MOV R7, HA_COLISAO
MOV R8, [R7]
CMP R8, 1
JZ APAGA_BALA


fim_anima_bala_anima:
        POP R9
        POP R8
        POP R7
        POP R6
        POP R5
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
```

```
                RET


APAGA_BALA:                     ; reset as caracteristicas da bala para as de origem, muda
o nr de balas para 0


        MOV R9, PIXEL_X                 ; mete o endereco dos X's em R9

        MOV [R9], R1            ; mete a coordenada no endereço

        MOV R9, PIXEL_Y                 ; mete o endereco  dos Y's em R9

        MOV [R9], R2            ; mete a coordenada no endereço

        MOV R4, ESTADO_PIXEL

        MOV R6, 0

        MOV [R4], R6                            ; DESLIGA A BALA


        MOV R1, BALA1

        MOV R2, 0

        MOVB [R1], R2

        ADD R1, 1

        MOVB [R1], R2

        ADD R1, 1

        MOVB [R1], R2



        MOV R1, bala_ativa

        MOV R2, 0

        MOVB [R1], R2


        MOV R1, HA_COLISAO

        MOV R2, 1

        MOVB [R1], R2


        JMP fim_anima_bala_anima
```

```
cor_ANIMA_BALA:                         ; rotina para selecionar a cor

        PUSH R9

        PUSH R8

        MOV     R8, ECRA                ;  coloca  em  R5  o  endereço  que  escolhe  a  cor
vermelha

        MOV R9, 0                       ; coloca a cor vermelha com valor 0

        MOV [R8], R9            ; mete o valor da cor dentro do endereço

        MOV R8, PIX_B           ; coloca em R5 o endereço que escolhe a cor azul

        MOV R9, 255                     ; coloca a cor azul com valor maximo

        MOV [R8], R9            ; mete o valor da cor dentro do endereço

        MOV R8, PIX_G           ; coloca em R5 o endereço que escolhe a cor azul

        MOV R9, 255                     ; coloca a cor azul com valor maximo

        MOV [R8], R9            ; mete o valor da cor dentro do endereço

        POP R8

        POP R9

        RET


;###################################

; as rotinas VERIFICAR_BALA_INIM[], recebem as coordenadas atuais de todas as naves
(aliada e inims) e verifica se os seus pixeis se sobrepoem

; caso se sobreponham o jogo acaba


VERIFICAR_BALA_INIM:

        PUSH R0

        PUSH R1

        PUSH R2

        PUSH R3

        PUSH R4

        PUSH R5


        MOV R0, BALA1

        MOV R1, INIM_xy

        MOV R2, INIM2_xy

        MOV R3, INIM3_xy
```

```
        MOVB R4, [R0]          ; TEM O X DA BALA

        MOVB R5, [R1]          ; TEM O X DA NAVE DO MEIO


        CMP R4, R5                    ; SE X DA BALA - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A BALA ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUA1

        ADD R5, 5

        CMP R5, R4                    ; SE X DA INIM+5 - X DA BALA FOR NEGATIVO SIGNIFICA
A BALA ESTA A DIREITA DA INIM

        JN CONTINUA1


        ADD R0, 1

        ADD     R1, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        MOVB R5, [R1]          ; TEM O Y DA NAVE DO MEIO


        CMP R4, R5                    ; SE O Y DA BALA FOR MENOR QUE O Y DA NAVE
SIGNIFICA QUE ESTA POR CIMA DA NAVE

        JN CONTINUA1

        ADD R5, 6

        CMP R5, R4                    ; SE O Y DA BALA FOR MAIOR QUE O Y DA NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA NAVE

        JN CONTINUA1

        JMP COLISAO                   ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA
DENTRO DA INIM




        CONTINUA1:

        MOV R0, BALA1


        MOVB R4, [R0]          ; TEM O X DA BALA

        MOVB R5, [R2]          ; TEM O X DA NAVE DA ESQ


        CMP R4, R5                    ; SE X DA BALA - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A BALA ESTA A ESQUERDA DA INIM E NAO COLIDE
```

```
        JN CONTINUA2

        ADD R5, 5

        CMP R5, R4                  ; SE X DA INIM+5 - X DA BALA FOR NEGATIVO SIGNIFICA
A BALA ESTA A DIREITA DA INIM

        JN CONTINUA2



        ADD R0, 1

        ADD     R2, 1

        MOVB R4, [R0]          ; TEM O Y DA NAVE

        MOVB R5, [R2]          ; TEM O Y DA NAVE DO MEIO



        CMP R4, R5                  ;  SE  O  Y  DA  BALA  FOR  MENOR  QUE  O  Y  DA  NAVE
SIGNIFICA QUE ESTA POR CIMA DA NAVE

        JN CONTINUA2

        ADD R5, 6

        CMP R5, R4                  ;  SE  O  Y  DA  BALA  FOR  MAIOR  QUE  O  Y  DA  NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA NAVE

        JN CONTINUA2

        JMP COLISAO                 ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA
DENTRO DA INIM




        CONTINUA2:

        MOV R0, BALA1

        MOVB R4, [R0]          ; TEM O X DA BALA

        MOVB R5, [R3]          ; TEM O X DA NAVE DA ESQ



        CMP R4, R5                  ; SE X DA BALA - X DA INIM FOR NEGATIVO SIGNIFICA
QUE A BALA ESTA A ESQUERDA DA INIM E NAO COLIDE

        JN CONTINUA3

        ADD R5, 5

        CMP R5, R4                  ; SE X DA INIM+5 - X DA BALA FOR NEGATIVO SIGNIFICA
A BALA ESTA A DIREITA DA INIM

        JN CONTINUA3



        ADD R0, 1

        ADD     R3, 1
```

```
        MOVB R4, [R0]          ; TEM O Y DA NAVE

        MOVB R5, [R3]          ; TEM O Y DA NAVE DO MEIO


        CMP R4, R5                    ;  SE  O  Y  DA  BALA  FOR  MENOR  QUE  O  Y  DA  NAVE
SIGNIFICA QUE ESTA POR CIMA DA NAVE

        JN CONTINUA3

        ADD R5, 6

        CMP R5, R4                    ;  SE  O  Y  DA  BALA  FOR  MAIOR  QUE  O  Y  DA  NAVE+6
SIGNIFICA QUE ESTA POR BAIXO DA NAVE

        JN CONTINUA3


        JMP COLISAO                   ; SE NAO PASSOU NESTE CRITERIO É PORQUE SE ENCONTRA
DENTRO DA INIM



        COLISAO:                      ;  no  bloco  da  colisao  vai  ser  emitido  um  som,  as
naves inimigas serao devolvidas ao topo do pixelscreen

        MOV R1, 3                     ; irao ser adicionados 5 pontos ao jogador

        CALL scoreboard_ad_sub

        MOV R1, HA_COLISAO

        MOV R2, 1

        MOV [R1], R2


        MOV R1, SWITCH_SOM

        MOV R2, 2

        MOV [R1], R2

        CALL escolha_som


        CALL desliga_INIM

        CALL desliga_INIM2

        CALL desliga_INIM3


        MOV R1, INIM_xy

        MOV R2, INIM2_xy

        MOV R3, INIM3_xy
```

```
        ; inimigos terao as coordendas originais
        MOV R4, 29
        MOVB [R1], R4
        ADD R1, 1
        MOV R4, 0
        MOVB [R1], R4

        MOV R4, 3
        MOVB [R2], R4
        ADD R2, 1
        MOV R4, 0
        MOVB [R2], R4

        MOV R4, 55
        MOVB [R3], R4
        ADD R3, 1
        MOV R4, 0
        MOVB [R3], R4


CONTINUA3:
        POP R5
        POP R4
        POP R3
        POP R2
        POP R1
        POP R0
        RET

;###############################################
```