INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

ARQUITETURA DE COMPUTADORES

LEIC/LERC/LEE

IST-TAGUSPARK

Batalha Espacial

Paulo Cabeças 76358

Marisa Roque 76653

Sara Santos 76503

1. Introdução

O projecto de Batalha Espacial pretende desenvolver um jogo em linguagem Assembly para o Processador Especial Para Ensino (PEPE), no âmbito da cadeira de Arquitectura de Computadores.

O jogo consiste numa nave controlada pelo jogador e em vários aliens controlados autonomamente. Os aliens movem-se num intervalo periódico na direcção da nave.

A nave possui um canhão para atacar os aliens. Esta pode ser movida em oito direções, rodar o canhão em dois sentidos e dispará-lo.

A nave possui energia que é gasta ao movimentar-se e ao disparar, por outro lado, recupera energia ao destruir um alien.

O jogo termina por falta de energia da nave, por colisão com um alien ou por opção do utilizador.

Foi considerado a origem do sistema de eixos XY o canto superior esquerdo do pixelscreen.

As teclas escolhidas para movimentação foram 0, 1, 2, 4, 6, 8, 9 e A. Para rodar o canhão foram escolhidas as teclas C e D. As teclas 3, 7 e B correspondem a suspender, terminar e reiniciar o jogo, respectivamente. Os comandos do jogo associados às teclas do teclado do simulador estão representadas na Figura 1.

Figura 1 – Representação dos comandos do jogo.

Na próxima parte, concepção e implementação é descrito a estrutura de hardware utilizada e visão sumária do código desenvolvido.

A parte seguinte, conclusões, recorda os objetivos iniciais que foram alcançados, quais não o foram, principais dificuldades, e sugestões de melhoramento. Por fim, apresenta se o código do projeto.

2. Concepção e Implementação

2.1. Estrutura Geral

O hardware em que se desenvolveu o projecto encontra-se representado no diagrama de blocos da Figura 2.

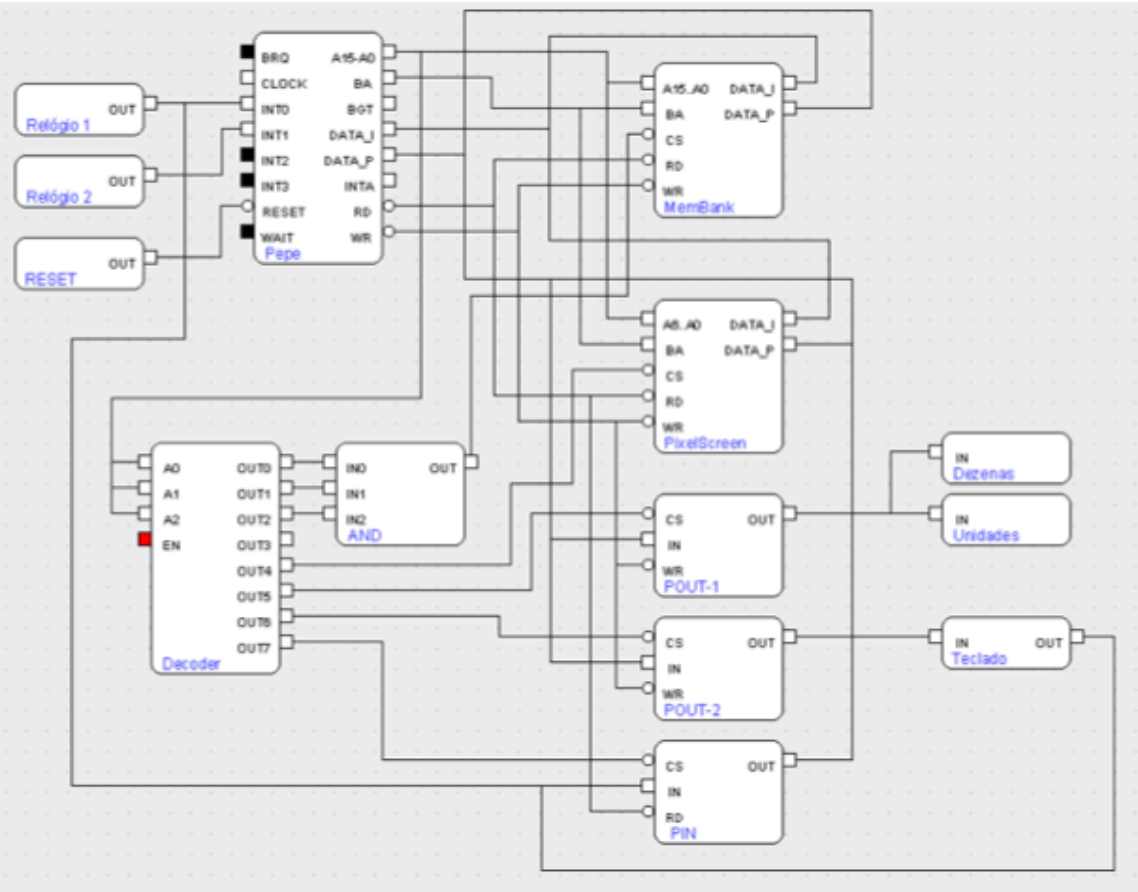

Figura 2 – Diagrama de blocos do hardware utilizado no simulador.

Os principais dispositivos são:

- PEPE – Processador de 16 bits.
- Relógio 1 – Relógio de tempo real, ligado à porta de interrupção 1 do PEPE. Usado como base temporal da evolução autónoma dos aliens.
- Relógio 2 – Relógio de tempo real, ligado à porta de interrupção 2 do PEPE. Usado como base temporal do gasto de energia da nave.
- MemBank – RAM de 16 bits, com capacidade de endereçamento de byte.
- PushMatrix – Teclado de 4 x 4 botões, com 4 bits ligados ao POUT – 2 e outros 4 ligados ao PIN. A detecção do botão carregado é feita por varrimento.

- PixelScreen - Ecrã de 32 x 32. É acedido como se fosse uma memória pois tem 32 linhas e 4 bytes por linhas.
- Dois displays de 7 segmentos - Utilizado para mostrar a energia da nave. Um para as dezenas e outro para as unidades.

A Figura 3 representa um fluxograma do ciclo de execução principal do software desenvolvido para o jogo da batalha espacial.

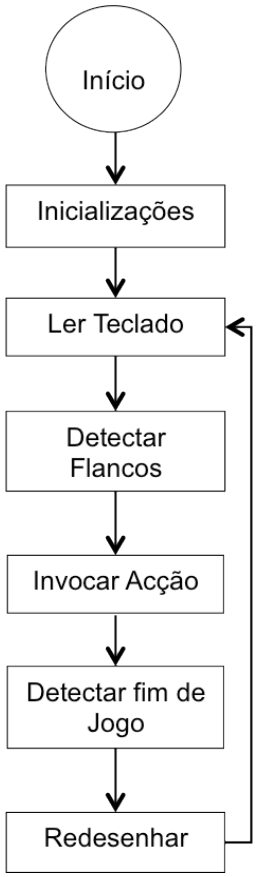

Figura 3 - Fluxo principal do programa

2.1.1. Mapa de endereçamento escolhido

Optou se por usar o mapa de endereços fornecido pelo docente no enunciado do projecto.

Dispositivo	Endereços
RAM (MemBank)	0000H a 5FFFH
PixelScreen	8000H a 807FH
POUT-1 (porto de saída de 8 bits)	0A000H
POUT-2 (porto de saída de 8 bits)	0C000H
PIN (porto de entrada de 8 bits)	0E000H

Tabela 1 - Endereços utilizados nos dispositivos do simulador.

2.1.2. Comunicação entre processos

A comunicação entre os processos é feita por registos e variáveis de estado, que serão aprofundados no próximo tópico deste relatório.

São usados registos, por exemplo, na rotina *desenha_objecto* que recebe os registos R1 e R2 para as coordenadas X e Y, respectivamente.

Outro exemplo é o da rotina *teclado* que usa o registo R10 para conter o valor da tecla lida.

São usadas variáveis de estado, por exemplo no processo *desenha_tudo*. A variável de estado *estado_jogo* é que decide se são chamados os processos que desenham a nave e os aliens no ecrã.

Também é usada a variável de estado *raio_on* para fazer a comunicação entre os processos *desenha_nave*, *desenha_raio* e *desenha_canhão*. Quando *desenha_nave* é invocado e se a variável *raio_on* estiver a 0 (raio desligado) apenas é desenhado o canhão, chamando o processo *desenha_canhão*. Se a variável estiver a 1, significa que o raio está ligado, logo é desenhado o raio.

Desta forma existe uma coesão entre os vários processos envolvidos no programa, de modo que não hajam simultaneamente dois processos a realizar operações opostas preservando a continuidade do programa, ou seja, há uma ligação do processo seguinte com o processo anterior de modo que o ciclo só esteja terminado quando for para terminar o programa (fim do jogo).

2.1.3. Variáveis de Estado

Foram utilizadas as seguintes variáveis de estado para conter informação sobre a nave:

- *nave_x* – guarda a posição X da nave.
- *nave_y* – guarda a posição Y da nave.
- *raio_on* – contém informação sobre o estado do raio.
- *energia* – guarda a energia que a nave possui.
- *canhao_*index – guarda a orientação do canhão, com valores no intervalo de 0 a 7.

Foram utilizadas as seguintes variáveis de estado para conter informação sobre os aliens:

- *alien_x* – tabela com as posições X dos aliens.
- alien_y – tabela com as posições Y dos aliens.

Foram utilizados as seguintes variáveis de estado para conter informação geral sobre o jogo:

- *estado_jogo* – contém um de três estados: normal, pausado e terminado. Esta variável é inicializada a normal. Esta variável é consultada em diversas rotinas como as de desenho, de movimentação para desistir se for caso disso.
- *redesenha* – indica a necessidade de redesenhar no pixelscreen.
- *redesenha_energia* – indica a necessidade de redesenhar no pixelscreen.
- *tecla_transicao* – indica que uma tecla acabou de ser pressionada ou largada, isto é, houve uma transição no estado da tecla.
- *tecla_flanco* – pode conter os valores 1 ou 0, para o flanco descendente ou ascendente da tecla, respectivamente.
- *tecla_anterior* – guarda a tecla lida pela rotina *teclado*, entre invocações sucessivas desta. É utilizada para detectar transição do estado da tecla.

2.1.4. Interrupções

As interrupções existentes são *relogio_energia* e *relogio_aliens*.

relogio_energia

Esta rotina é chamada em cada interrupção provocada pelo relógio 2. Serve para controlar a energia da nave que pode ser decrementada consoante a variável de estado *raio_on*, activando a variável de estado *redesenha_energia*.

relogio_aliens

Esta rotina é chamada em cada interrupção provocada pelo relógio 1. A rotina vai percorrer as posições X e Y de todos os aliens e actualiza-as no sentido de os aliens se aproximarem da nave.

2.1.5. Rotinas

As rotinas consideradas principais na realização deste projecto foram:

modifica_pixel

Esta rotina recebe uma posição X em R1, uma posição Y em R2 e um estado em R9, permitindo que seja possível modificar o pixel no pixelscreen nas coordenadas X e Y.

Divide-se em cinco passos:

- Descobrir qual o endereço do byte a ser lido do pixelscreen, através da expressão: $8000_h + (4y + x/8))$
- Descobrir qual a posição do bit a ser modificado do pixelscreen, obtendo o resto da divisão inteira da coordenada X por 8.
- Ler do endereço calculado o byte a ser modificado.
- Modificar o bit de interesse.
- Reescrever o byte modificado no endereço de interesse no ecrã.

detecta_game_over

Esta rotina consulta a energia da nave e avalia as posições dos aliens e da nave para detectar colisões e escreve na variável *estado_jogo*.

andar

Esta rotina obedece ao comando do jogador movendo a nave na direcção solicitada.

Para converter a tecla numa direcção é usada a tabela de mapeamento *teclado_direccao*. Uma vez obtida a direcção são usadas as tabelas *table_x* e *table_y*.

Existem ainda nesta rotina restrições para fazer com que a nave não se movimente para fora do pixelscreen, tendo em atenção a orientação do canhão.

desenha_objecto

Esta rotina recebe uma posição X em R1 uma posição Y em R2 e um endereço de uma tabela em R10. A tabela corresponde aos pixéis do objecto a desenhar, com um tamanho de 3x3 pixéis.

detecta_ataque

Esta rotina começa por determinar as posições X e Y da ponta do raio. Com estas posições vai percorrer os aliens e verificar se existe colisão com as coordenadas dos aliens. Em caso positivo a energia é incrementada.

detecta_colisao

Esta rotina recebe dois pares de coordenadas X e Y e uma distância d em R5 e avalia o resultado da seguinte expressão: $|x_1 - x_2| < d \ \wedge \ |y_1 - y_2| < d$.

gira_direita* e *gira esquerda

Estas rotinas alteram a orientação do canhão, tendo atenção às fronteiras do jogo.

coordenadas_raio

Esta rotina recebe um par de coordenadas X e Y do centro da nave e uma distância, devolve as coordenadas X e Y do pixel que se encontra à distancia dada do centro da nave, segundo a direcção do canhão.

teclado

Esta rotina percorre as quatro linhas do teclado procurando um valor diferente de zero nas colunas. Optou-se por usar ao invés de uma tabela, potências de dois nos valores a colocar nas linhas e a ler das colunas.

A tecla lida é obtida com a seguinte expressão: $tecla = 4 \times linha + coluna$.

Se não for detectada nenhuma tecla devolve -1.

detecta_flancos

Esta rotina começa por limpar a variável de estado *tecla_transição*, garantindo que nunca está activa em interações consecutivas do ciclo do programa principal.

Compara a *tecla_anterior* com a tecla actual e se for diferente activa a variável de estado *tecla_transicao.*

É ainda feita uma detecção do flanco ascendente ou descendente e guardado o estado em *tecla_flanco.*

invoca_accao

São consultadas as variáveis de estado *tecla_transicao*, *tecla_flanco* e o registo R10 com a tecla actual para decidir se é altura correcta para invocar a acção.

Para descortinar a acção a executar é usada a tabela *table_teclado*, que contém endereços das rotinas indexadas pela tecla.

3. Conclusões

A estratégia de implementação seguiu de perto a recomenda pelo enunciado do projecto.

O objectivo de implementação de um jogo da batalha espacial para o processador PEPE foi conseguido com sucesso, no entanto destacam-se alguns desafios e escolhas que se destacam em seguida.

A configuração do período temporal dos aliens não foi implementada devido a uma dúvida na interpretação do enunciado.

A solução apresentada possui erros visíveis no pixelscreen. Estes artefactos desaparecem quando as rotinas são executadas passo a passo. Uma vez que o simulador em si também possui alguns erros, atribui-se a isso uma possível causa dos erros visíveis.

Optou-se por ser mais óbvio em termos de jogo não fazer a colisão da nave com um alien quando os seus centros coincidem, mas quando um invade o espaço do outro.

Devido à natureza do simulador onde foi desenvolvido o projecto não foi possível testar qual o é o comportamento quando são pressionadas duas ou mais teclas.

4. Código assembly

```
; *********************************************************************************
; * Jogo Batalha Espacial
; * Arquitectura de Computadores (2012/2013)
; *
; * Autores:  Paulo Cabeças   76358
; *           Marisa Roque    76653
; *           Sara Santos     76358
; *********************************************************************************


; *********************************************************************************
; * Constantes
; *********************************************************************************
DOIS                    EQU 2H
QUATRO                  EQU 4H
OITO                    EQU     8H
DEZ                     EQU 10                  ; base decimal
TAMANHO_PX              EQU 32                  ; pixeis, ecra é sempre quadrado
LIMITE_64               EQU 64                  ; limite superior na escrita de palavras no
pixelscreen
LIMITE_128              EQU 128                 ; limite superior na escrita de palavras no
pixelscreen
PX_SCREEN               EQU 8000H               ; endereço do pixelscreen (inicio)
TECLADO_OUT             EQU     0C000H          ; endereço do porto de E/S do teclado
TECLADO_IN              EQU 0E000h
DISPLAY                 EQU 0A000H              ; endereço do porto dos displays
hexadecimais
NORMAL                  EQU 0                   ; estado_jogo: situação normal de jogo
PAUSADO                 EQU 1                   ; estado_jogo: jogo em pausa
TERMINADO               EQU 2                   ; estado_jogo: jogo terminado

; *********************************************************************************
; * Configuracoes
; *********************************************************************************
ENERGIA_INICIAL                 EQU     80
ENERGIA_MAXIMO          EQU     99
BONUS_ENERGIA           EQU 10          ; valor do bonus de energia ao destruir um alien
N_ALIENS                        EQU 4           ; numero de aliens
NAVE_X_INICIAL          EQU 20
NAVE_Y_INICIAL          EQU     15
ORIENTACAO_INICIAL      EQU     1
RAIO_INICIAL            EQU 0
ENERGIA_TEMP            EQU 2
TAMANHO_RAIO            EQU 5           ; tamanho do raio, a contar do centro da nave
ALIEN_X0_INICIAL    EQU 30
ALIEN_Y0_INICIAL    EQU 30
ALIEN_X1_INICIAL    EQU 1
ALIEN_Y1_INICIAL    EQU 1
ALIEN_X2_INICIAL    EQU 30
ALIEN_Y2_INICIAL    EQU 1
ALIEN_X3_INICIAL    EQU 1
ALIEN_Y3_INICIAL    EQU 30

; *********************************************************************************
; * Stack
; *********************************************************************************
PLACE           1000H                                   ; localiza blocos de dados

pilha:          TABLE 200H                              ; espaço reservado para a pilha

SP_inicial:                                             ; este é o endereço com que o SP
deve ser inicializado.

; *********************************************************************************
; * Dados
; *********************************************************************************

table_pxscreen:         WORD 10000000b ; 0 tabela de 8 máscaras (cada uma só com um bit a
1)
```

```
                                WORD 01000000b ; 1     para desenhar no pixel
screen
                                WORD 00100000b ; 2
                                WORD 00010000b ; 3
                                WORD 00001000b ; 4
                                WORD 00000100b ; 5
                                WORD 00000010b ; 6
                                WORD 00000001b ; 7

table_alien:          WORD 00000101b ; 1a linha
                                WORD 00000010b ; 2a linha
                                WORD 00000101b ; 3a linha

table_nave:           WORD 00000111b ; 1a linha
                                WORD 00000101b ; 2a linha
                                WORD 00000111b ; 3a linha


table_x:                      WORD 0                ; componente no eixo y da direcao no
sistema de eixos do ecra
                                WORD 1
                                WORD 1
                                WORD 1
                                WORD 0
                                WORD -1
                                WORD -1
                                WORD -1

table_y:                      WORD -1               ; componente no eixo y da direcao no
sistema de eixos do ecra
                                WORD -1
                                WORD 0
                                WORD 1
                                WORD 1
                                WORD 1
                                WORD 0
                                WORD -1

teclado_direccao:     WORD 7                ; tabela para converter as teclas em
orientacoes
                                WORD 0                ; o valor da tecla sera o
indice
                                WORD 1                ; usa-se -1 para perfazer
uma tabela de 16 entradas
                                WORD -1
                                WORD 6
                                WORD -1
                                WORD 2
                                WORD -1
                                WORD 5
                                WORD 4
                                WORD 3
                                WORD -1
                                WORD -1
                                WORD -1
                                WORD -1


nave_x:                       WORD NAVE_X_INICIAL
nave_y:                       WORD NAVE_Y_INICIAL
raio_on:                      WORD RAIO_INICIAL
energia:                      WORD ENERGIA_INICIAL
canhao_index:         WORD ORIENTACAO_INICIAL               ; indice das tabelas y e x


table_teclado:        WORD andar                    ; 0
                                WORD andar                    ; 1
                                WORD andar                    ; 2
                                WORD pausa                    ; 3
                                WORD andar                    ; 4
                                WORD disparar         ; 5
                                WORD andar                    ; 6
                                WORD sair                     ; 7
                                WORD andar                    ; 8
                                WORD andar                    ; 9
                                WORD andar                    ; A
```

```
                                WORD reset                      ; B
                                WORD gira_esquerda  ; C
                                WORD gira_direita      ; D
                                WORD nada                       ; E
                                WORD nada                       ; F

alien_x_inicial:        WORD ALIEN_X0_INICIAL ; tabela com posições x iniciais dos aliens
para repor mais facilmente
                                WORD ALIEN_X1_INICIAL
                                WORD ALIEN_X2_INICIAL
                                WORD ALIEN_X3_INICIAL

alien_y_inicial:        WORD ALIEN_Y0_INICIAL ; tabela com posições y iniciais dos aliens
para repor mais facilmente
                                WORD ALIEN_Y1_INICIAL
                                WORD ALIEN_Y2_INICIAL
                                WORD ALIEN_Y3_INICIAL

alien_x:                WORD ALIEN_X0_INICIAL ;  tabela com posições x dos aliens
                                WORD ALIEN_X1_INICIAL
                                WORD ALIEN_X2_INICIAL
                                WORD ALIEN_X3_INICIAL

alien_y:                WORD ALIEN_Y0_INICIAL ;  tabela com posições y dos aliens
                                WORD ALIEN_Y1_INICIAL
                                WORD ALIEN_Y2_INICIAL
                                WORD ALIEN_Y3_INICIAL


estado_jogo:            WORD NORMAL
redesenha:                      WORD 1
redesenha_energia:      WORD 1


tecla_flanco:           WORD 0                                  ; 0 = flanco ascendente; 1
flanco descendente
tecla_transicao:        WORD 0                                  ; 0 = sem transição; 1 = com
transicção
tecla_anterior:                 WORD -1                                 ; guarda a tecla
pressionada entre iterações, para detectar transições

; Tabela de vectores de interrupção
tabela_interrupcoes:            WORD relogio_aliens
                                                        WORD relogio_energia


; ********************************************************************************
; * Código
; ********************************************************************************
PLACE                   0000H                                                   ; o código
tem de começar em 0000H
inicio:                 MOV     SP, SP_inicial                  ; inicializa SP para
a palavra a seguir à última da pilha
                                MOV     BTE, tabela_interrupcoes        ; incializa BTE

                                CALL desenha_tudo

                                EI0
        ; permite interrupções tipo 0
                                EI1
        ; permite interrupções tipo 1
                                EI
        ; activa interrupçoes globais
ciclo_inicio:  CALL teclado

                                CALL detecta_flancos                    ; detectar tecla
pressionada ou largada


                                CALL invoca_accao
                                CALL detecta_game_over
                                CALL detecta_ataque
                                CALL desenha_tudo

                                JMP ciclo_inicio
```

```
; *********************************************************************************
; * Rotinas de Interrupções
; *********************************************************************************

; *********************************************************************************
; * Descrição: Trata a interrupção do relógio da energia.
; *********************************************************************************
relogio_energia:
                            PUSH R1
                            PUSH R2
                            PUSH R3

                            MOV R3, estado_jogo                         ; nao detecta
se já terminado
                            MOV R2, [R3]
                            CMP R2, TERMINADO
                            JEQ f_rel_energia

                            MOV R1, raio_on                                     ; se
raio desligado não faz nada
                            MOV R2, [R1]
                            CMP R2,0
                            JEQ f_rel_energia

                            MOV R1, energia                                     ; lê
energia
                            MOV R2, [R1]
                            MOV R3, ENERGIA_TEMP
                            SUB R2, R3                                          ;
decrementa
                            MOV [R1], R2                                ; escreve na
memória

                            MOV R2, redesenha_energia           ; activar flag, para
redesenhar
                            MOV R1, 1
                            MOV [R2], R1

f_rel_energia: POP R3
                            POP R2
                            POP R1
                            RFE

; *********************************************************************************
; * Descrição: Trata a interrupção do relógio dos aliens.
; *********************************************************************************
relogio_aliens:
                            CALL mover_aliens
                            RFE


; *********************************************************************************
; * Rotinas
; *********************************************************************************


; *********************************************************************************
; * Descrição: Detecta fim de jogo, por colisão ou falta de energia
; * Entrada:          --
; * Saída:                    --
; * Destrói:          --
; *********************************************************************************

detecta_game_over:
                                    PUSH R1
                                    PUSH R2
                                    PUSH R3
                                    PUSH R9

                                    MOV R3, estado_jogo                         ; nao
detecta se já terminado
                                    MOV R2, [R3]
                                    CMP R2, TERMINADO
                                    JEQ go_fim
```

```
                                MOV R1, energia
                                MOV R2, [R1]
                                CMP R2, 0
                                JLE go_terminado

                                CALL detecta_colisao_aliens          ; saida em R9
                                CMP R9, 1
                                JNE go_fim

go_terminado:           DI

                                MOV R2, 0
                                MOV [R1], R2                                 ;
energia a zero

                                MOV R2, TERMINADO                            ; muda
o estado de jogo para terminado
                                MOV [R3], R2

                                MOV R2, redesenha                            ;
activar flag, para redesenhar
                                MOV R1, 1
                                MOV [R2], R1

                                MOV R2, redesenha_energia            ; activar
flag, para redesenhar energia
                                MOV R1, 1
                                MOV [R2], R1

go_fim:                 POP R9
                                POP R3
                                POP R2
                                POP R1
                                RET

; *********************************************************************************
; * Descrição: Desenha aliens e nave
; * Entrada:          --
; * Saída:                  --
; * Destrói:          --
; *********************************************************************************
desenha_tudo:
                                PUSH R0
                                PUSH R1
                                PUSH R2

                                MOV R2, redesenha_energia            ; não mostra energia,
se a flag estiver a 0. Para não piscar
                                MOV R1, [R2]
                                CMP R1, 0
                                JEQ dt_screen
                                CALL mostra_energia

                                MOV R1, 0                                    ; já
desenhou, limpa flag
                                MOV [R2], R1


dt_screen:              MOV R2, redesenha                            ; não desenha, se a
flag estiver a 0. Para não piscar
                                MOV R1, [R2]
                                CMP R1, 0
                                JEQ dt_fim

                                MOV R0, estado_jogo                          ; não
desenha, se terminado
                                MOV R1, [R0]
                                CMP R1, TERMINADO
                                JEQ dt_terminado                             ; if
terminado { enche_screen } else {desenha nave + aliens }


                                CALL limpa_screen
                                CALL desenha_nave
                                CALL desenha_aliens
```

```
                              JMP dt_if_fim

dt_terminado:   CALL enche_screen


dt_if_fim:      MOV R1, 0                                       ; já desenhou, limpa
flag
                              MOV [R2], R1

dt_fim:                 POP R2
                              POP R1
                              POP R0
                              RET


; *********************************************************************************
; * Descrição: Altera/modifica no ecra o pixel nas coordenadas X e Y.
; * Entrada:   R1 (Coordenada X do ecra)
;                       R2 (Coordenada Y do ecra)
;               R9 (Accao a fazer: apagar = 0 / acender = 1)
; * Saída:                 --
; * Destrói:           --
; *********************************************************************************
; * Notas:
; *(0,0) do pixelscreen é o canto superior esquerdo.

modifica_pixel:
                              PUSH R3                                         ; salvaguarda
registos
                              PUSH R4
                              PUSH R5
                              PUSH R6
                              PUSH R7
                              PUSH R8

                              ; 1ª fase: descobrir qual o endereço do byte a ser lido do
pixel screen

                              ; 8000h + (y * 4 + x / 8) = endereco do byte de interesse,
resultado em R3
                              MOV R3, R2                                      ; primeira
parte: y * 4, resultado em R3
                              MOV R4, QUATRO
                              MUL R3, R4
                              MOV R4, R1                                      ; segunda
parte: x / 8, resultado em R4
                              MOV R5, OITO
                              DIV R4, R5
                              ADD R3, R4                                      ; junta as
duas partes, resultado em R3
                              MOV R5, PX_SCREEN                       ; somar offset do
endereco do ecra
                              ADD R3, R5

                              ; 2ª fase: descobrir qual a posição do bit a ser
modificado do pixel screen
                              MOV R6, R1                                      ; x % 8 =
posição do bit de interesse, resultado em R6
                              MOV R7, OITO
                              MOD R6, R7

                              ; multiplicar por dois, devido à cena par/impar da RAM
                              MOV R7, DOIS
                              MUL R6, R7

                              MOV R7, table_pxscreen
                              ADD R7, R6
                              ADD R7, 1
                              MOVB R8, [R7]                           ; máscara

                              ; 3ª fase: ler do endereço calculado o byte a ser
modificado
                              ; ler o byte de interesse do ecra, resultado em R4
                              MOVB R4, [R3]

                              ; 4ª fase: modificar o bit de interesse
```

```
                        ; if acende = true
                        CMP R9, 0
                        JZ apaga
                        OR R8, R4
                        JMP fim_if

apaga:          NOT R8
                        AND R8, R4

                        ; 5ª fase: reescrever o byte modificado no endereco de
interesse no ecra
fim_if:         MOVB [R3], R8

                        POP R8                                  ; restaura
registos
                        POP R7
                        POP R6
                        POP R5
                        POP R4
                        POP R3
                        RET

; *********************************************************************************
; * Descrição: Move N aliens com base nas coordenadas da tabela.
; * Entrada:          --
; * Saída:                  --
; * Destrói:          --
; *********************************************************************************

mover_aliens:   PUSH R1
                                PUSH R2
                                PUSH R3
                                PUSH R4
                                PUSH R5
                                PUSH R6

                                MOV R5, 0                               ;
inicializa variavel de indice
                                MOV R6, N_ALIENS                ; inicializa
numero de aliens (N)
                                SHL R6, 1                               ;
N_ALIENS * 2
                                MOV R3, alien_x
                                MOV R4, alien_y


percorre_aliens1:       CMP R5, R6
                                JGE terminou1

                                MOV R1, [R3+R5]                 ;
alien_x
                                MOV R2, [R4+R5]                 ;
alien_y

                                CALL andar_alien                ; modifica R1
e R2

                                MOV [R3+R5], R1                 ;
actualiza em memoria
                                MOV [R4+R5], R2

                                ADD R5, 2                               ;
par/impar
                                JMP percorre_aliens1

terminou1:                      MOV R2, redesenha               ; activar flag, para
redesenhar
                                MOV R1, 1
                                MOV [R2], R1

                                POP R6
                                POP R5
                                POP R4
                                POP R3
                                POP R2
```

```
                                    POP R1
                                    RET

; *********************************************************************************
; * Descrição: Andar aliens
; * Entrada:          R1 (x_alien), R2 (y_alien)
; * Saída:                  R1 (x_alien), R2 (y_alien)
; * Destrói:          --
; *********************************************************************************

andar_alien:
                            PUSH R4
                            PUSH R5
                            PUSH R6

                            ;;;;;;;;;
                            ;       X       ;
                            ;;;;;;;;;

                            MOV R6, nave_x          ; endereço nave_x
                            MOV R4,[R6]                     ; nave_x

                            CMP R4,R1
                            JZ fim_x                        ; nave_x (R4) = alien_x (R1)
                            JGT aumenta_x           ; nave_x (R4) > alien_x (R1)
                            SUB R1,1              ; senao diminui o x uma unidade
                            JMP fim_x
aumenta_x:          ADD R1,1                ; aumenta uma unidade ao x do alien

                            ;;;;;;;;;
                            ;       Y       ;
                            ;;;;;;;;;

fim_x:                  MOV R6,nave_y           ; endereço nave_y
                            MOV R4,[R6]                     ; nave_y

                            CMP R4,R2
                            JZ fim_y                        ; nave_y = alien_y
                            JGT aumenta_y           ; nave_y (R4) > alien_y (R2)
                            SUB R2,1
                            JMP fim_y

aumenta_y:          ADD R2,1


fim_y:                  POP R6
                            POP R5
                            POP R4
                            RET

; *********************************************************************************
; * Descrição: Desenha N aliens com base nas coordenadas da tabela.
; * Entrada:          --
; * Saída:                  --
; * Destrói:          --
; *********************************************************************************

desenha_aliens:             PUSH R1
                                    PUSH R2
                                    PUSH R3
                                    PUSH R4
                                    PUSH R5
                                    PUSH R6
                                    PUSH R7

                                    MOV R5, 0                       ; inicializa variavel
de indice
                                    MOV R6, N_ALIENS        ; inicializa numero de
aliens
                                    SHL R6, 1                       ; N_ALIENS * 2
                                    MOV R3, alien_x
                                    MOV R4, alien_y


percorre_aliens:        CMP R5, R6
                                    JGE terminou
```

```
                                MOV R1,[R3+R5]          ; alien_x
                                MOV R2, [R4+R5]                 ; alien_y

                                CALL desenha_alien

                                ADD R5, 2                       ; par/impar
                                JMP percorre_aliens

terminou:                       POP R7
                                POP R6
                                POP R5
                                POP R4
                                POP R3
                                POP R2
                                POP R1
                                RET

; *********************************************************************************
; * Descrição: Desenha um alien no ecra com base nas coordenadas X e Y, que é o seu
centro.
; * Entrada:          R1 (Coordenada X do centro do alien)
;                                 R2 (Coordenada Y do centro do alien)
; * Saída:                  --
; * Destrói:          --
; *********************************************************************************

desenha_alien:
                                PUSH R10
                                MOV R10, table_alien            ; tabela com os
pixeis do alien
                                CALL desenha_objecto            ; entra R1, R2 e R10
                                POP R10
                                RET

; *********************************************************************************
; * Descrição: Desenha um nave no ecra com base nas coordenadas X e Y, que é o seu
centro.
; * Entrada:          --
; * Saída:                  --
; * Destrói:          --
; *********************************************************************************

desenha_nave:
                                PUSH R1
                                PUSH R2
                                PUSH R3
                                PUSH R4
                                PUSH R10

                                MOV R10, table_nave             ; tabela com os
pixeis da nave
                                MOV R3, nave_x                          ; endereço
x_nave
                                MOV R4, nave_y                          ; endereço
y_nave
                                MOV R1, [R3]                            ; x_nave
                                MOV R2, [R4]                            ; y_nave

                                CALL desenha_objecto            ; entra R1, R2 e R10

                                MOV R3, raio_on                                 ;
desenha raio ou canhão conforme variavel raio_on
                                MOV R4, [R3]
                                CMP R4, 0
                                JEQ canhao
                                CALL desenha_raio
                                JMP fim_des_nave
canhao:                 CALL desenha_canhao

fim_des_nave:  POP R10
                                POP R4
                                POP R3
                                POP R2
                                POP R1
                                RET
```

```
; *********************************************************************************
; * Descrição: Desenha um objecto (nave/alien) no ecra com base nas coordenadas X e Y,
que é o seu centro.
; * Entrada:          R1 (Coordenada X do centro do objecto)
;                                 R2 (Coordenada Y do centro do objecto)
;                                 R10 (Tabelas da nave ou do alien)
; * Saída:                  --
; * Destrói:          --
; *********************************************************************************
; * Notas:
; * O objecto tem 3x3 pixeis

desenha_objecto:
                            PUSH R3                                         ;
salvaguarda registos
                            PUSH R4
                            PUSH R5
                            PUSH R6
                            PUSH R7
                            PUSH R8
                            PUSH R9

                            MOV R3, R1                                      ;
backup de R1 (x_centro)
                            MOV R4, R2                                      ;
backup de R2 (y_centro)

                            MOV R5, -1                                      ;
variavel do ciclo for

ciclo_nave:           CMP R5, 2                                     ; sair do
ciclo quando for >= 3
                            JGE fim_nave

                            MOV R2, R4                                      ;
y_centro
                            ADD R2, R5                                      ;
y_pixel = y_centro + i

                            MOV R7, R5                                      ;
posicao_memoria = 3 + 2 * R5 + table_nave
                            SHL R7, 1                                       ;
equivale a multiplicar por 2
                            ADD R7, 3
                            ADD R7, R10

                            MOVB R6, [R7]                                 ; ler o byte
original da tabela

                            MOV R9, R6
                            MOV R8, 00000100b
                            AND R9, R8                                      ;
mascara para definir o bit de interesse
                            SHR R9, 2                                       ;
meter o bit de interesse na posicao de menor peso
                            MOV R1, R3                                      ;
x_centro
                            SUB R1, 1                                       ;
x_pixel = x_centro - 1
                            CALL modifica_pixel

                            MOV R9, R6
                            MOV R8, 00000010b
                            AND R9, R8
                            SHR R9, 1
                            MOV R1, R3                                      ;
x_pixel = x_centro
                            CALL modifica_pixel

                            MOV R9, R6
                            MOV R8, 00000001b
                            AND R9, R8
                            MOV R1, R3                                      ;
x_centro
```

```
                            ADD R1, 1                                       ;
x_pixel = x_centro + 1
                            CALL modifica_pixel

                            ADD R5, 1                                       ;
incrementa a variavel do ciclo
                            JMP ciclo_nave



fim_nave:           MOV R1, R3                                      ; restore de
R1 (x_centro)
                            MOV R2, R4                                      ;
restore de R2 (y_centro)
                            POP R9                                          ;
restaura registos
                            POP R8
                            POP R7
                            POP R6
                            POP R5
                            POP R4
                            POP R3
                            RET

; *********************************************************************************
; * Descrição:        Verifica ataque aos aliens.
; * Entrada:          --
; * Saída:                    --
; * Destrói:          --
; *********************************************************************************
detecta_ataque:
                            PUSH R0
                            PUSH R1
                            PUSH R2
                            PUSH R3
                            PUSH R4
                            PUSH R5
                            PUSH R6
                            PUSH R7
                            PUSH R8
                            PUSH R9
                            PUSH R10
                            PUSH R11


                            MOV R1, estado_jogo                     ; nada, se
jogo terminado
                            MOV R1, [R1]
                            CMP R1, TERMINADO
                            JEQ da_fim

                            MOV R1, raio_on                                 ;
nada, se raio desligado
                            MOV R1, [R1]
                            CMP R1, 0
                            JEQ da_fim


                            MOV R1, nave_x
                            MOV R2, nave_y
                            MOV R1, [R1]                            ; x_nave
                            MOV R2, [R2]
                            MOV R10, TAMANHO_RAIO           ; tamanho do raio, a
contar do centro da nave

                            CALL coordenadas_raio           ; entra R1, R2, R10.
calcular coordenadas da ponta do raio. resultado em R3, R4


                            MOV R0, 0                                       ;
indice do loop
                            MOV R11 ,N_ALIENS
                            SHL R11 ,1
da_loop:            CMP R0, R11
                            JEQ da_fim
```

```
                                CALL detecta_ataque_alien


da_proximo:             ADD R0, 2                                       ; par/impar
                                JMP da_loop

da_fim:                 POP R11
                                POP R10
                                POP R9
                                POP R8
                                POP R7
                                POP R6
                                POP R5
                                POP R4
                                POP R3
                                POP R2
                                POP R1
                                POP R0
                                RET

; *********************************************************************************
; * Descrição:            Verifica se a ponta do raio colide com alien.
; * Entrada:              R0 (offset das tabelas alien_x, alien_y, alien_x_inicial,
alien_y_inicial
;                                       R3 (x_raio)
;                                       R4 (y_raio)
; * Saída:                       --
; * Destrói:             --
; *********************************************************************************
detecta_ataque_alien:
                                PUSH R0
                                PUSH R1
                                PUSH R2
                                PUSH R3
                                PUSH R4
                                PUSH R5
                                PUSH R6
                                PUSH R7
                                PUSH R8
                                PUSH R9
                                PUSH R10
                                PUSH R11

                                MOV R5, 1
        ; distância limite da colisão entre raio e aliens
                                MOV R7, alien_x
                                MOV R8, alien_y
                                MOV R10, alien_x_inicial                        ; endereço da
tabela de posições iniciais dos aliens
                                MOV R6, alien_y_inicial


                                MOV R1, [R7 + R0]                                       ;
x_alien
                                MOV R2, [R8 + R0]

                                CALL detecta_colisao                            ; entra
R1,R2,R3,R4,R5, saida em R9
                                CMP R9, 1
                                JNE daa_fim


                                MOV R1, [R10 + R0]                                      ;
conteudo da tabela de posições iniciais dos aliens
                                MOV R2, [R6 + R0]

                                MOV [R7 + R0], R1                                       ;
copiar para a tabela das posições dos aliens
                                MOV [R8 + R0], R2

                                CALL add_bonus_energia                          ; incrementa
o bonus de energia

daa_fim:                POP R11
                                POP R10
```

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

; *********************************************************************************
; * Descrição:        Incrementa a energia da NAVE, depois da morte de um alien
; * Entrada:          --
; * Saída:                    --
; * Destrói:          --
; *********************************************************************************
add_bonus_energia:
                            PUSH R1
                            PUSH R2
                            PUSH R3
                            PUSH R4

                            MOV R2, energia                                     ;
incrementar BONUS_ENERGIA de energia
                            MOV R1, [R2]
                            MOV R3, BONUS_ENERGIA
                            ADD R1, R3

                            MOV R4, ENERGIA_MAXIMO              ; há um máximo de
energia
                            CMP R1, R4
                            JLT abe_escreve

                            MOV  R1, R4

abe_escreve:   MOV [R2], R1

                            MOV R2, redesenha_energia           ; activar flag, para
redesenhar
                            MOV R1, 1
                            MOV [R2], R1

                            POP R4
                            POP R3
                            POP R2
                            POP R1
                            RET


; *********************************************************************************
; * Descrição:        Verifica se existe colisao entre objectos.
;                                 |x_1 - x_2 | <= d AND |y_1 - y_2| <= d
; * Entrada:          R1 (x do objecto 1)
;                                 R2 (y do objecto 1)
;                                 R3 (x do objecto 2)
;                                 R4 (y do objecto 2)
;                                 R5 ( distancia limite da colisão )
; * Saída:                  R9 (Colisao = 1 / Sem colisao = 0)
; * Destrói:          R3, R4
; *********************************************************************************
detecta_colisao:

                            MOV R9, 0                                       ; inicializar
R9 = 0: sem colisao

                            SUB R3, R1
                            JNN positivo_x
                            NEG R3
positivo_x:           CMP R3, R5                                    ; |x_1 - x_2| <= d
                            JGT dc_fim

                            SUB R4, R2
                            JNN positivo_y
                            NEG R4
```

```
positivo_y:             CMP R4, R5
                        JGT dc_fim

                        MOV R9, 1                                   ; houve
colisao entre os objectos

dc_fim:                 RET

; *********************************************************************************
; * Descrição:          Verifica se existe colisao entre os aliens e a nave.
; * Entrada:            --
; * Saída:                     R9 (Colisao = 1 / Sem colisao = 0)
; * Destrói:           --
; *********************************************************************************
detecta_colisao_aliens:
                        PUSH R0
                        PUSH R1
                        PUSH R2
                        PUSH R3
                        PUSH R4
                        PUSH R5
                        PUSH R6
                        PUSH R7
                        PUSH R8
                        PUSH R10

                        MOV R7, alien_x                             ; endereços
das coordenadas dos alien
                        MOV R8, alien_y

                        MOV R1, nave_x                      ; endereços das
coordenadas da nave
                        MOV R2, nave_y
                        MOV R1, [R1]                        ; x_nave
                        MOV R2, [R2]                        ; y_nave

                        MOV R5, DOIS                        ; distância limite da
colisão entre nave e aliens

                        MOV R10, 0                                  ; inicializa
variavel do indice
                        MOV R0, N_ALIENS
                        SHL R0, 1                                   ; N_ALIENS *
2


dca_loop:               CMP R10, R0
                        JGE dca_fim_loop

                        MOV R3, [R7+R10]                    ; x_alien
                        MOV R4, [R8+R10]                    ; y_alien

                        CALL detecta_colisao          ; retorna 1/0 em R9
                        CMP R9, 1
                        JEQ dca_fim_loop

                        ADD R10, 2                                  ; par/impar

                        JMP dca_loop

dca_fim_loop:  POP R10
                        POP R8
                        POP R7
                        POP R6
                        POP R5
                        POP R4
                        POP R3
                        POP R2
                        POP R1
                        POP R0
                        RET

; *********************************************************************************
; * Descrição: Incrementa o indice da direccao do canhao, fazendo-o virar à direita.
; * Entrada:            --
; * Saída:                     --
```

```
; * Destrói:          --
; ********************************************************************************
gira_direita:
                            PUSH R1
                            PUSH R2
                            PUSH R3
                            PUSH R4
                            PUSH R5
                            PUSH R6
                            PUSH R7
                            PUSH R8

                            MOV R7, estado_jogo                 ; verificar o estado
do jogo
                            MOV R8, [R7]
                            CMP R8, NORMAL                              ; em caso
afirmativo o canhão não gira
                            JNE fim

                            MOV R3, canhao_index
                            MOV R4, [R3]

                            MOV R6, nave_x
                            MOV R6, [R6]                                ; x_nave
                            MOV R7, nave_y
                            MOV R7, [R7]                                ; y_nave

                            ; parede à esquerda
                            CMP R6, 1                                           ;
x_nave = 1, canhao_index = 4, não incrementar
                            JNE fim_parede_e
                            CMP R4, 4
                            JEQ fim

                            ; parede à direita
fim_parede_e:  MOV R5, TAMANHO_PX
                            SUB R5, 2
                            CMP R6, R5                                          ;
x_nave = 31, canhao_index = 0, não incrementar
                            JNE fim_parede_d
                            CMP R4, 0
                            JEQ fim

                            ; parede em cima
fim_parede_d:  CMP R7, 1                                        ; y_nave = 1,
canhao_index = 6, não incrementar
                            JNE fim_parede_c
                            CMP R4, 6
                            JEQ fim

                            ; parede em baixo
fim_parede_c:  CMP R7, R5                                       ; y_nave = 31,
canhao_index = 2, não incrementar
                            JNE fim_parede_b
                            CMP R4, 2
                            JEQ fim

fim_parede_b:  ADD R4, 1                                        ; incrementa o indice
da direccao
                            MOV R5, 00000111b
                            AND R4, R5                                          ;
mascara para voltar para o zero  (incrementa entre 0 e 7)
                            MOV [R3], R4

fim:                    MOV R2, redesenha                   ; activar flag, para
redesenhar
                            MOV R1, 1
                            MOV [R2], R1

                            POP R8
                            POP R7
                            POP R6
                            POP R5
                            POP R4
                            POP R3
                            POP R2
```

```
                            POP R1
                            RET


; ********************************************************************************
; * Descrição: Decrementa o indice da direccao do canhão, fazendo-o virar à esquerda.
; * Entrada:          --
; * Saída:                  --
; * Destrói:          --
; ********************************************************************************

gira_esquerda:
                            PUSH R1
                            PUSH R2
                            PUSH R3
                            PUSH R4
                            PUSH R5
                            PUSH R6
                            PUSH R7
                            PUSH R8

                            MOV R7, estado_jogo                 ; verificar o estado
do jogo
                            MOV R8, [R7]
                            CMP R8, NORMAL                              ; em caso
afirmativo o canhao não gira
                            JNE fim1


                            MOV R3, canhao_index
                            MOV R4, [R3]

                            MOV R6, nave_x
                            MOV R6, [R6]                                ; x_nave
                            MOV R7, nave_y
                            MOV R7, [R7]                                ; y_nave

                            ; parede à esquerda
                            CMP R6, 1                                           ;
x_nave = 1, canhao_index = 0, não incrementar
                            JNE fim_parede_e1
                            CMP R4, 0
                            JEQ fim1

                            ; parede à direita
fim_parede_e1: MOV R5, TAMANHO_PX
                            SUB R5, 2
                            CMP R6, R5                                          ;
x_nave = 31, canhao_index = 4, não incrementar
                            JNE fim_parede_d1
                            CMP R4, 4
                            JEQ fim1

                            ; parede em cima
fim_parede_d1: CMP R7, 1                                    ; y_nave = 1,
canhao_index = 2, não incrementar
                            JNE fim_parede_c1
                            CMP R4, 2
                            JEQ fim1

                            ; parede em baixo
fim_parede_c1: CMP R7, R5                                   ; y_nave = 31,
canhao_index = 6, não incrementar
                            JNE fim_parede_b1
                            CMP R4, 6
                            JEQ fim1


fim_parede_b1: SUB R4, 1                                    ; decrementa o indice
da direccao
                            MOV R5, 00000111b
                            AND R4, R5                                          ;
mascara para voltar para o zero (decrementa entre 0 e 7)
                            MOV [R3], R4
```

```
fim1:                   MOV R2, redesenha                           ; activar flag, para
redesenhar
                                MOV R1, 1
                                MOV [R2], R1

                                POP R8
                                POP R7
                                POP R6
                                POP R5
                                POP R4
                                POP R3
                                POP R2
                                POP R1
                                RET


; *********************************************************************************
; * Descrição: Calcula as coordenadas a uma determinada distância, segundo a orientação
do canhão.
; * Entrada:           R1 (posiçao x do centro da nave)
;                                 R2 (posicao y do centro da nave)
;                                 R10 (distancia do raio)
;
; * Saída:                    R3 (posição x do raio)
;                                 R4 (posição y do raio)
; * Destrói:          --
; *********************************************************************************
coordenadas_raio:
                                PUSH R5
                                PUSH R6

                                MOV R6, canhao_index
                                MOV R5, [R6]
                                SHL R5,1                            ; corresponde a
multiplicar por 2. par/impar

                                MOV R6, table_x
                                MOV R3,[R6 + R5]

                                ; x_canhao = x_centro + (table_x * 2)
                                MUL R3, R10                         ; multiplicar pelo
tamanho do raio (entre 2 e 5)
                                ADD R3, R1


                                MOV R6, table_y
                                MOV R4, [R6 + R5]

                                ; y_canhao = y_centro + (table_y * 2)
                                MUL R4, R10                         ; multiplicar pelo
tamanho do raio (entre 2 e 5)
                                ADD R4, R2

                                POP R6
                                POP R5
                                RET


; *********************************************************************************
; * Descrição: Função que desenha o canhão da nave.
; * Entrada:   R1 ( posiçao x do centro da nave )
;                                 R2 ( posicao y do centro da nave )
; * Saída:             --
; * Destrói:   --
; *********************************************************************************

desenha_canhao:
                                PUSH R3
                                PUSH R4
                                PUSH R9
                                PUSH R10

                                MOV R10, 2                                          ;
tamanho 2 do raio (corresponde ao canhao)

                                CALL coordenadas_raio
```

```
                            MOV R9, 1                                       ; 1
para acender o pixel

                            MOV R1, R3
                            MOV R2, R4

                            CALL modifica_pixel

                            POP R10
                            POP R9
                            POP R4
                            POP R3
                            RET

; *********************************************************************************
; * Descrição: Função que desenha o raio da nave.
; * Entrada:   R1 ( posiçao x do centro da nave )
;                           R2 ( posicao y do centro da nave )
; * Saída:            --
; * Destrói:   --
; *********************************************************************************

desenha_raio:
                            PUSH R3
                            PUSH R4
                            PUSH R5
                            PUSH R6
                            PUSH R7
                            PUSH R9
                            PUSH R10

                            MOV R10, 2                                      ;
tamanho 2 do raio (corresponde ao canhao)

dr_loop:            CMP R10, TAMANHO_RAIO
                            JGT dr_fim

                            MOV R5, R1                                      ;
guardar x_nave, porque modifica_pixel precisa de R1 com outro valor
                            MOV R6, R2
                            CALL coordenadas_raio               ; saidas em R3 e R4


                            MOV R1, R3
                            MOV R2, R4
                            MOV R9, 1                                       ; 1
para acender o pixel

                            ; não deixa desejar para além das paredes do ecra
                            MOV R7, TAMANHO_PX                          ; parede da
direita
                            CMP R1, R7
                            JGE dr_fim

                            CMP R2, R7                                      ;
parede de baixo
                            JGE dr_fim

                            CMP R1, 0                                       ;
parede da esquerda
                            JLT dr_fim

                            CMP R2, 0                                       ;
parede de cima
                            JLT dr_fim

                            CALL modifica_pixel

                            MOV R1, R5                                      ;
repor x_nave em R1
                            MOV R2, R6

                            ADD R10, 1                                      ;
próximo pixel do raio
                            JMP dr_loop
```

```
dr_fim:                 POP R10
                                POP R9
                                POP R7
                                POP R6
                                POP R5
                                POP R4
                                POP R3
                                RET


; *********************************************************************************
; * Descrição:          Função que altera a posição da nave.
; * Entrada:            R10 (Tecla premida)
; * Saída:                      --
; * Destrói:            --
; *********************************************************************************

andar:
                                PUSH R9
                                PUSH R8
                                PUSH R7
                                PUSH R6
                                PUSH R5
                                PUSH R4
                                PUSH R3
                                PUSH R2
                                PUSH R1
                                PUSH R0

                                MOV R0, 0                                       ; flag
para detectar movimento para a energia

                                MOV R7, estado_jogo                     ; verificar o estado
do jogo
                                MOV R8, [R7]
                                CMP R8, NORMAL                                  ; em caso
afirmativo não anda
                                JNE fim_andar_y

                                MOV R7, teclado_direccao

                                MOV R4, R10
                                SHL R4, 1                                       ; R4 *
2, par/impar
                                MOV R3, [R7+R4]                                 ;
orientacao do movimento em R3
                                SHL R3, 1                                       ; R3 *
2, par/impar

                                ;;;;;;;;;
                                ;       X       ;
                                ;;;;;;;;;

                                MOV R7, table_x
                                MOV R1, [R7 + R3]                               ; componente
x do movimento
                                CMP R1, 0
                                JEQ fim_andar_x

                                MOV R9, canhao_index
                                MOV R9, [R9]
                                SHL R9, 1                                       ; R9 *
2
                                MOV R9, [R7+R9]                                 ;
componente x da orientacao do canhao, x_canhao (R9)

                                ; x_novo (R5) = x_antigo (R5) + e_x (R1)
                                MOV R7, nave_x
                                MOV R5, [R7]
                                ADD R5, R1

                                ; testar x_novo com a parede da direita
                                MOV R8, TAMANHO_PX                              ; limite (R8)
a ser construido
                                SUB R8, 1
```

```
                              CMP R9,0                                          ; se
x_canhao (R9) >= 0, o limite (R8) decresce 1
                              JLT fim_canhao_x_d
                              SUB R8, R9
fim_canhao_x_d:         CMP R5, R8
                              JGE fim_andar_x                                   ; fora
da parede da direita. nao se actualiza e nao se testa a parede da esquerda

                              ; testar x_novo com a parede da esquerda
                              MOV R8, 1
                              CMP R9, 0                                         ; se
x_canhao (R9) <= 0, o limite (R8) cresce 1
                              JGT fim_canhao_x_e
                              SUB R8, R9
fim_canhao_x_e: CMP R5,R8
                              JLT fim_andar_x

                              ; limites X testados, actualizar memoria
                              MOV [R7], R5                                ; actualiza
x_novo em memoria
                              MOV R0, 1                                         ;
movimento efectuado


                              ;;;;;;;;;
                              ;       Y       ;
                              ;;;;;;;;;

fim_andar_x:    MOV R7, table_y
                              MOV R2, [R7 + R3]                           ; componente
y do movimento
                              CMP R2, 0
                              JEQ fim_andar_y

                              MOV R9, canhao_index
                              MOV R9, [R9]
                              SHL R9, 1                                         ; R9 *
2
                              MOV R9, [R7+R9]                                   ;
componente y da orientacao do canhao, y_canhao (R9)


                              ; y_novo (R6) = y_antigo (R6) + e_x (R2)
                              MOV R7, nave_y
                              MOV R6, [R7]
                              ADD R6, R2

                              ; testar y_novo com a parede de baixo
                              MOV R8, TAMANHO_PX                          ; limite (R8)
a ser construido
                              SUB R8, 1
                              CMP R9,0                                          ; se
y_canhao (R9) >= 0, o limite (R8) decresce 1
                              JLT fim_canhao_y_b
                              SUB R8, R9
fim_canhao_y_b:         CMP R6, R8
                              JGE fim_andar_y                                   ; fora
da parede de baixo. nao se actualiza e nao se testa a parede de cima

                              ; testar y_novo com a parede de cima
                              MOV R8, 1
                              CMP R9, 0                                         ; se
x_canhao (R9) <= 0, o limite (R8) cresce 1
                              JGT fim_canhao_y_c
                              SUB R8, R9
fim_canhao_y_c: CMP R6,R8
                              JLT fim_andar_y

                              ; limites Y testados, actualizar memoria
                              MOV [R7], R6
                              MOV R0, 1                                         ;
movimento efectuado

fim_andar_y:    MOV R7, energia
                              MOV R6, [R7]
```

```
                            SUB R6, R0                                          ; R0
tem 0 ou 1 consoante movimentos efectuados ou nao
                            MOV [R7], R6

                            MOV R2, redesenha                           ; activar
flag, para redesenhar
                            MOV R1, R0                                          ;  R0
tem 0 ou 1 consoante movimentos efectuados ou nao
                            MOV [R2], R1

                            MOV R2, redesenha_energia           ; activar flag, para
redesenhar energia
                            MOV R1, R0
                            MOV [R2], R1

                            POP R0
                            POP R1
                            POP R2
                            POP R3
                            POP R4
                            POP R5
                            POP R6
                            POP R7
                            POP R8
                            POP R9
                            RET

; *************************************************************************************
; * Descrição:        Função que activa/desactiva o raio.
; * Entrada:          --
; * Saída:                    --
; * Destrói:          --
; *************************************************************************************

disparar:
                            PUSH R0
                            PUSH R1
                            PUSH R2
                            PUSH R7
                            PUSH R6

                            MOV R7, estado_jogo    ; verificar o estado do jogo
                            MOV R6, [R7]
                            CMP R6, NORMAL                ; em caso afirmativo não
dispara
                            JNE dispa_fim

                            MOV R0, raio_on
                            MOV R1, [R0]                  ; ler estado do raio


                            CMP R1, 0
                            JEQ dispara

                            MOV R1, 0                           ; se raio ligado,
desliga
                            JMP fim_disparar


dispara:              MOV R1, 1                             ; se raio desligado, liga
                            MOV R7, energia                     ; decrementar 2
unidades de energia
                            MOV R6, [R7]
                            SUB R6, 2
                            MOV [R7], R6


fim_disparar:  MOV [R0], R1                 ; escrever estado do raio

                            MOV R2, redesenha             ; activar flag, para
redesenhar
                            MOV R1, 1
                            MOV [R2], R1

                            MOV R2, redesenha_energia           ; activar flag, para
redesenhar
```

```
                        MOV R1, 1
                        MOV [R2], R1

dispa_fim:      POP R6
                        POP R7
                        POP R2
                        POP R1
                        POP R0
                        RET

; *********************************************************************************
; * Descrição:        Interrompe o jogo até ser chamada novamente ou o jogo recomeçado.
; * Entrada:          --
; * Saída:                   --
; * Destrói:          --
; *********************************************************************************

pausa:
                        PUSH R0
                        PUSH R1

                        MOV R0, estado_jogo
                        MOV R1, [R0]

                        CMP R1, TERMINADO
                        JEQ p_fim                       ; estado terminado,
sair

                        CMP R1, NORMAL
                        JNE p_normal

                        DI                                      ; parar
interrupções global
                        MOV R1, PAUSADO                 ; jogo normal, pausar

                        JMP p_fim

p_normal:       MOV R1, NORMAL          ; jogo pausado,
                        EI                                      ; recomeçar
interrupções
p_fim:          MOV [R0], R1

                        POP R1
                        POP R0
                        RET

; *********************************************************************************
; * Descrição:        Função vazia para preencher na tabela de acções do teclado.
; * Entrada:          --
; * Saída:                   --
; * Destrói:          --
; *********************************************************************************

nada:
                        NOP
                        RET

; *********************************************************************************
; * Descrição:        Pára o jogo independentemente do estado actual.
; * Entrada:          --
; * Saída:                   --
; * Destrói:          --
; *********************************************************************************
sair:
                        PUSH R0
                        PUSH R1

                        DI                                      ; parar
interrupções
                        MOV R0, estado_jogo             ; endereço do estado_jogo
                        MOV R1, TERMINADO               ; jogo pausado
                        MOV [R0], R1                    ; escreve o estado em
memória

                        CALL enche_screen               ; mostra visualmente que o
jogo acabou
```

```
                            MOV R0, redesenha           ; activar flag, para
redesenhar
                            MOV R1, 1
                            MOV [R0], R1

                            POP R1
                            POP R0
                            RET


; *************************************************************************
; * Descrição:        Recomeça o jogo, independentemente do estado actual.
; * Entrada:          --
; * Saída:                    --
; * Destrói:          --
; *************************************************************************
reset:
                            PUSH R0
                            PUSH R1
                            PUSH R2
                            PUSH R3
                            PUSH R4
                            PUSH R5
                            PUSH R6

                            DI                                              ;
suspender interrupções, enquanto repomos o estado inicial

                            MOV R0, estado_jogo                 ; endereço do
estado_jogo
                            MOV R1, NORMAL                      ; recomeçar o estado
                            MOV [R0], R1                        ; escreve o estado em
memoria

                            MOV R0, energia                             ; repor
energia inicial
                            MOV R1, ENERGIA_INICIAL
                            MOV [R0], R1

                            MOV R0, nave_x                      ; repor x_nave
                            MOV R1, NAVE_X_INICIAL
                            MOV [R0], R1

                            MOV R0, nave_y                      ; repor y_nave
                            MOV R1, NAVE_Y_INICIAL
                            MOV [R0], R1

                            MOV R0, canhao_index          ; repor orientação da nave
                            MOV R1, ORIENTACAO_INICIAL
                            MOV [R0], R1

                            MOV R0, raio_on                             ; repor
estado do raio da nave
                            MOV R1, RAIO_INICIAL
                            MOV [R0], R1

                            ; repor posições dos aliens
                            MOV R3, alien_x_inicial
                            MOV R0, alien_x
                            MOV R5, alien_y_inicial
                            MOV R6, alien_y

                            MOV R1, 0
                            MOV R2, N_ALIENS
                            SHL R2, 1                                   ; R2 * 2;
par/impar
r_loop:             CMP R1, R2
                            JGE r_fim_loop

                            MOV R4, [R3+R1]                             ; copia da
tabela de valores iniciais de x
                            MOV [R0+R1], R4
                            MOV R4, [R5+R1]                             ; copia da
tabela de valores iniciais de y
                            MOV [R6+R1], R4
```

```
                            ADD R1, 2
                            JMP r_loop

r_fim_loop:           MOV R0, redesenha                    ; activar flag, para
redesenhar
                            MOV R1, 1
                            MOV [R0], R1

                            MOV R0, redesenha_energia     ; activar flag, para
redesenhar energia nos displays
                            MOV R1, 1
                            MOV [R0], R1

                            EI                                                  ;
recomeçar interrupções

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

; *********************************************************************************
; * Descrição: Função que limpa completamente o Pixel Screen.
; * Entrada:   --
; * Saída:            --
; * Destrói:   --
; *********************************************************************************
limpa_screen:
                            PUSH    R1                                          ;
salvaguarda registos
                            PUSH    R2
                            PUSH    R3

                            MOV           R1, PX_SCREEN                 ; mete o
endereço do pixel screen em R1
                            MOV           R3, LIMITE_128                ; define o
limite de bytes do pixel screen
                            ADD     R3, R1
                            MOV           R2, 0                                 ;
pixeis todos a 0

ciclo_screen:  CMP            R1, R3                              ; testa se chegou ao
fim da matriz
                            JZ            fim_screen                            ;
salta para fim_matriz caso tenha chegado

                            MOVB    [R1], R2                              ; preenche os
primeiros 8 bits com o valor
                            ADD           R1, 1                                 ;
passa para os proximos 8 bits
                            JMP     ciclo_screen                  ; repete o ciclo até
ter limpo todo o pixel screen

fim_screen:           POP     R3                                          ; restaura
registos
                            POP     R2
                            POP     R1
                            RET


; *********************************************************************************
; * Descrição: Função que preenche completamente o Pixel Screen.
; * Entrada:   --
; * Saída:            --
; * Destrói:   --
; *********************************************************************************

enche_screen:
                            PUSH    R1                                  ; salvaguarda
registos
                            PUSH    R2
```

```
                            PUSH    R3

                            MOV             R1, PX_SCREEN           ; mete o endereço do
pixel screen em R1
                            MOV             R3, LIMITE_128          ; define o limite de
bytes do pixel screen
                            ADD     R3, R1
                            MOV             R2, 0FFh                        ; pixeis
todos a 1

ciclo_preenche:         CMP             R1, R3                          ; testa se chegou ao
fim
                            JZ              fim_preenche

                            MOVB    [R1], R2                        ; preenche os
primeiros 8 bits com o valor
                            ADD             R1, 1                           ; passa para
os proximos 8 bits
                            JMP     ciclo_preenche          ; repete o ciclo até ter
limpo toda a matriz

fim_preenche:  POP      R3                                      ; restaura registos
                            POP     R2
                            POP     R1
                            RET


; ********************************************************************************
; * Descrição:          Funçao que faz a descodificação da tecla que foi premida.
; * Entrada:            --
; * Saída:                      R10 (Tecla premida)
; * Destrói:            --
; ********************************************************************************

teclado:
                            PUSH R1
                            PUSH R2
                            PUSH R3
                            PUSH R4
                            PUSH R5
                            PUSH R6
                            PUSH R7
                            PUSH R8

                            MOV     R2, TECLADO_OUT                 ; R2 com o endereço
do periférico
                            MOV R7, TECLADO_IN
                            MOV R5,2                                ; é usado as
potencias de dois para relacionar a linha com as entradas e saidas do teclado

                            MOV R6, 000FH                   ; máscara para isolar o bit
de menor peso

                            MOV R8, 0                               ; R8 vai indicar qual
a linha em teste
                            MOV     R1, 1                           ; R1 vai ter o input
correspondente à linha em teste. Inicia com linha 0 (0001b)
                            MOV R4, 4                               ; Limite para
recomeçar o ciclo

ciclo_scan:             MOVB [R2], R1                   ; escrever no porto de saída
(teclado)
                            MOVB R3, [R7]                   ; ler do porto de entrada
(teclado)
                            AND R3, R6                              ; afectar as flags
(MOVs não afectam as flags)
                            JNZ fim_scan                    ; R3 != 0,  tecla premida
                            MUL R1, R5                              ; proxima linha
                            ADD R8, 1
                            CMP R8, R4
                            JLE ciclo_scan
                            MOV R10, -1                             ; se a linha >= 4,
não foi detectada nenhuma tecla. Retorna-se com R10 = -1
                            JMP teclado_fim

fim_scan:               MOV R4, 0                               ; inicializa o contador
```

```
ciclo_tecla:   CMP R3, 1                          ; verifica se é igual a 1
                              JZ fim_coluna
                              DIV R3, R5                          ; divide sempre por 2
                              ADD R4, 1                           ; incrementa o
contador
                              JMP ciclo_tecla

                              ; tecla (R10) = linha (R8) * 4 + coluna (R4)
fim_coluna:           SHL R8,2
                              ADD R8, R4
                              MOV R10, R8                  ; guardar tecla premida em
registo

teclado_fim:   POP R8
                              POP R7
                              POP R6
                              POP R5
                              POP R4
                              POP R3
                              POP R2
                              POP R1
                              RET

; *********************************************************************************
; * Descrição:         Detectar flanco ascendente e descendente da tecla.
; * Entrada:           R10 (Tecla premida)
; * Saída:                    --
; * Destrói:           --
; *********************************************************************************

detecta_flancos:
                              PUSH R1
                              PUSH R2
                              PUSH R3
                              PUSH R4

                              MOV R1, tecla_transicao
                              MOV R2, tecla_anterior
                              MOV R4, tecla_flanco

                              MOV R3, 0
                              MOV [R1], R3                                    ; limpar
transição

                              MOV R3, [R2]
                              CMP R10, R3                                             ;
comparar com tecla anterior
                              JEQ dflancos_fim

                              MOV R3, 1
                              MOV [R1], R3                                    ; houve
transição

                              CMP R10, -1                                             ;
tecla (R10) == -1, é flanco descendente
                              JEQ dflancos_desc

                              MOV R3, 0                                               ;
flanco ascendente = 0
                              MOV [R4], R3
                              JMP dflancos_fim

dflancos_desc: MOV R3, 1                                      ; flanco descendente
= 1
                              MOV [R4], R3


dflancos_fim:  POP R4
                              POP R3
                              POP R2
                              POP R1
                              RET


; *********************************************************************************
```

```
; * Descrição:         Determina qual a acção a realizar e executa-a.
; * Entrada:          R10 (tecla premida)
; * Saída:                   --
; * Destrói:          --
; ********************************************************************************

invoca_accao:
                              PUSH R0
                              PUSH R1
                              PUSH R2
                              PUSH R3
                              PUSH R4
                              PUSH R5
                              PUSH R6
                              PUSH R7

                              MOV R0, table_teclado                         ; tabela com
endereços das rotinas a chamar
                              MOV R1, tecla_transicao
                              MOV R2, disparar                                      ;
endereço da rotina dispara
                              MOV R3, tecla_flanco
                              MOV R6, tecla_anterior                        ; tecla
guardada da iteração anterior

                              MOV R7, R10
                              MOV R10, [R6]
                              MOV [R6], R7                                          ;
guarda a tecla premida nesta iteração para a próxima

                              MOV R1, [R1]                                          ; R1 =
1 se houve transicção entre tecla pressionada e não pressionada
                              CMP R1, 0
        ; testar se alguma transicção foi detectada
                              JEQ ia_fim

                              CMP R7, 5
        ; 5 é a tecla de dispara
                              JNE ia_else

                              CALL disparar
                              JMP ia_fim

ia_else:              MOV R4, [R3]
                              CMP R4, 1
        ; flanco descendente
                              JNE ia_fim

                              MOV R5, R10
                              SHL R5, 1
        ; R1 * 2
                              MOV R0, [R0+R5]
                              CALL R0

ia_fim:               MOV R10, R7                                                   ;
repor tecla premida

                              POP R7
                              POP R6
                              POP R5
                              POP R4
                              POP R3
                              POP R2
                              POP R1
                              POP R0
                              RET

; ********************************************************************************
; * Descrição:         Mostra energia nos displays.
; * Entrada:          --
; * Saída:                   --
; * Destrói:          --
; ********************************************************************************

mostra_energia:
                              PUSH R1
```

```
                            PUSH R10

                            MOV R1, energia
                            MOV R10, [R1]
                            CALL display

                            POP R10
                            POP R1
                            RET

; *************************************************************************************
; * Descrição:        Mostra em formato decimal nos dois displays.
; * Entrada:          R10 (valor em hexadecimal)
; * Saída:                --
; * Destrói:          --
; *************************************************************************************

display:
                        PUSH R2                                 ; guardar registos
                        PUSH R3
                        PUSH R4
                        PUSH R5

                        MOV R3, R10                             ; coloca valor
hexadecimal a mostrar num registo para as unidades
                        MOV R4, R10                             ; coloca valor
hexadecimal a mostrar num registo para as dezenas
                            MOV R2, DEZ                         ; guarda constante
10, útil para fazer a divisao inteira e descobrir o resto da divisao inteira

                            DIV R4, R2                          ; guarda o valor das
dezenas (R4)
                        MOD R3, R2                      ; guarda o valor das
unidades (R3)

                        SHL R4, 4                       ; empurra nibble das dezenas
para o nibble de maior peso
                        OR  R4, R3                      ; junta o nibble das dezenas
com os das unidades
                        MOV R5, DISPLAY
                        MOVB [R5], R4                   ; escreve no porto de saída
(POUT-1)

                            POP R5                              ; restaurar registos
                            POP R4
                        POP R3
                        POP R2
                        RET
```