



ARQUITETURA DE COMPUTADORES

LETI

2018/2019

IST-TAGUSPARK

RELATÓRIO DO PROJETO

Miguel Ricardo – 87552

Francisco Rosa – 93578

Francisco Bento - 93581

1. Introdução

No âmbito da disciplina de Arquitetura de Computadores, foi pedido a cada grupo que desenvolvesse um jogo: “Batalha Naval”. Com isto, exercitámos e aprofundámos os conhecimentos de programação em linguagem *assembly*, os periféricos e as interrupções.

O objetivo do projeto é acertar nos barcos sem ser afundado.

O jogo é iniciado com o submarino na posição central na metade inferior do ecrã. Provenientes do lado esquerdo no terço superior do ecrã, vão surgindo barcos que pertencem a uma frota de navios. O objetivo do submarino é afundar o máximo número de barcos, lançando torpedos. No entanto, o submarino pode ser afundado por balas disparadas pelos barcos. Estes barcos não se veem, pois encontram-se fora do desenho no lado esquerdo do ecrã. Apenas se observam as balas a aparecerem do lado esquerdo. Quando o submarino atinge um barco, ganha um ponto. A pontuação final, apresentada num dos displays de 7 segmentos, mede a qualidade do jogador e representa os pontos obtidos a acertar nos barcos. Apenas se poderão deslocar 2 barcos no máximo, um submarino, um torpedo e uma bala. O ecrã é apresentado em perspetiva, do ponto de vista aéreo.

Presentes na secção 2, estarão as várias explicações e observações do como e porquê do nosso código. Diversos pontos servirão para diferentes objetivos ao longo do projeto.

A secção 3 servirá para apresentar as conclusões do grupo e possíveis melhorias a termos em mente no futuro.

Por último, a secção 4 corresponderá à transposição do código *assembly* para este ficheiro.

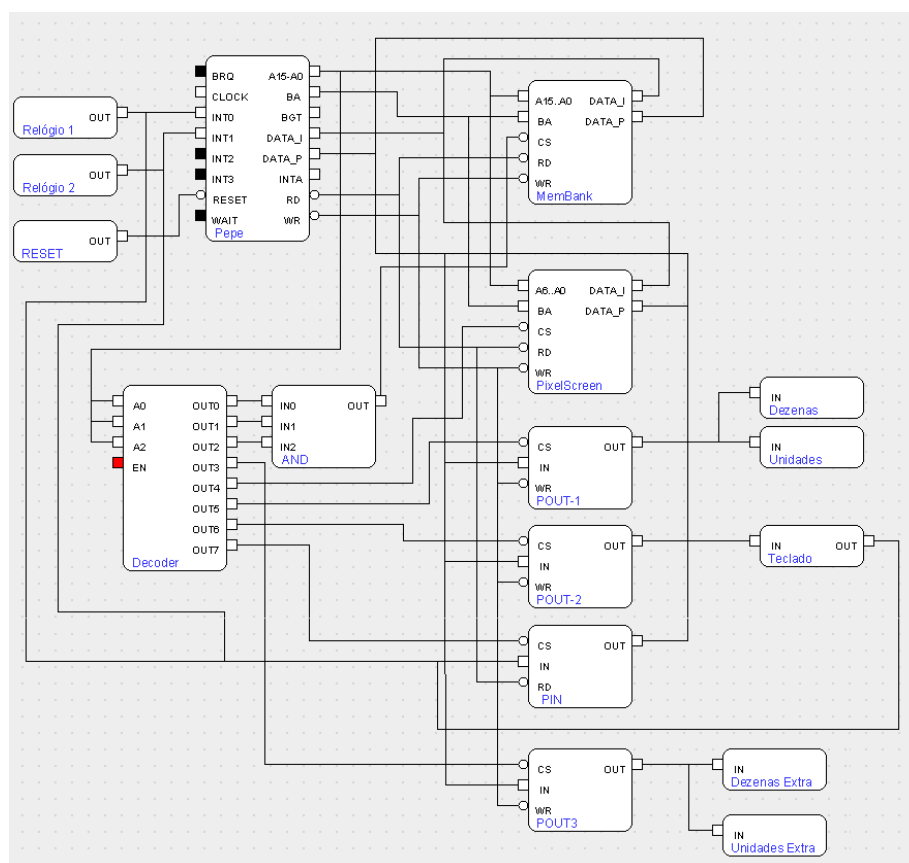
2. Conceção e Implementação

2.1. Estrutura Geral

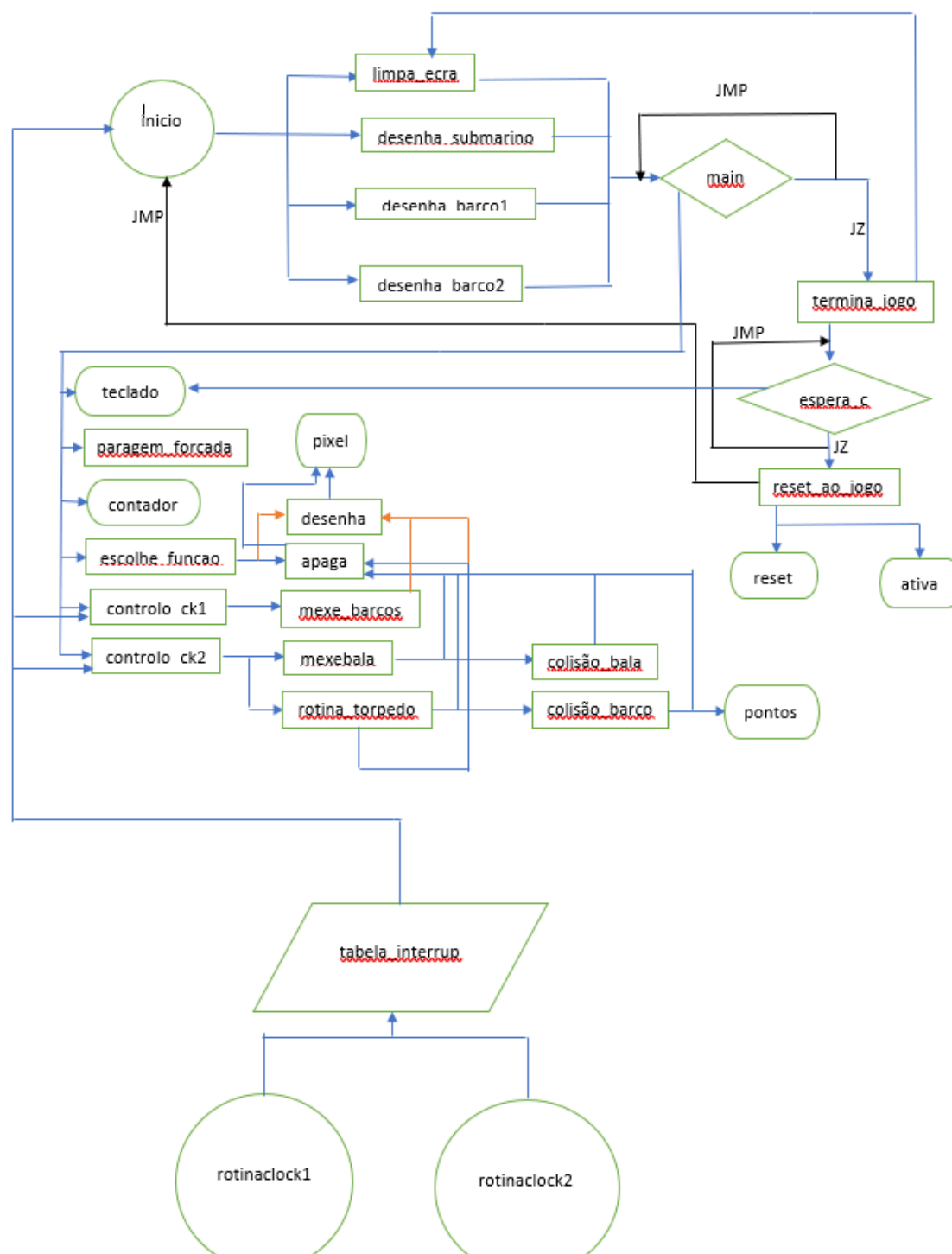
Aqui deve ser dada uma panorâmica do que se concebeu, nomeadamente a estrutura do hardware e do software.

No hardware, deve ser apresentado um diagrama de blocos e explicar muito sucintamente a orgânica de funcionamento desses blocos com vista ao objetivo do trabalho.

2.1.1. Hardware







2.1.2. Software



2.2. Implementação do Programa

Quanto à implementação do programa, há diversos aspetos a ter em conta que facilitam a compreensão do mesmo.

- Teclas (as que têm cruz não são utilizadas):
 - 1 – mexe submarino para cima e para a esquerda
 - 2 – mexe submarino para cima
 - 3 – mexe submarino para cima e para a direita
 - 5 – mexe submarino para a esquerda
 - 6 – dispara torpedo
 - 7 – mexe submarino para a direita
 - 9 – mexe submarino para baixo e para a esquerda
 - a – mexe submarino para baixo
 - b – mexe submarino para baixo e para a direita
 - c – faz reset ao programa
 - f – força a paragem do programa

	1	2	3
	5	6	7
	9	a	b
c			f

- Parâmetros das rotinas (exemplo)
 - Teclas: as rotinas para mexer o submarino utilizam as teclas como argumento para formar um switch. Também temos a tecla 'c' (variável geral) para terminar o jogo.
 - Coordenadas: são utilizadas como parâmetros para se desenhar, apagar e movimentar os objetos numa determinada posição e para determinar os estados das colisões. As coordenadas são guardadas em memória, através de WORDs, para não se perder os valores atuais.

- Valor dos píxeis: são apenas utilizados nas rotinas de desenhar os objetos, tendo em conta que as funções de apagar usam um registo geral com o píxel a 0. Determinam se a função *desenha_pixel* deverá ou desenhar ou apagar/deixar apagado ou píxel em questão.
- Divisão do ecrã: O submarino não se pode mover para cima de metade do ecrã, onde se situam os barcos.

2.3. Mapa de endereçamento escolhido

Dispositivo	Endereços
RAM (MemBank)	0000H a 5FFFH
POUT-3 (periférico de saída de 8 bits)	06000H
PixelScreen	8000H a 807FH
POUT-1 (periférico de saída de 8 bits)	0A000H
POUT-2 (periférico de saída de 8 bits)	0C000H
PIN (periférico de entrada de 8 bits)	0E000H

2.4. Comunicação entre processos

Quanto à main, existe uma comunicação entre processos no sentido em que este ciclo interliga as rotinas. Por exemplo, a rotina *teclado* está ligada à rotina *escolhe_funcao* pois retorna a tecla premida e, ao guardá-la num registo, é lida e permite o movimento do submarino.

Para coordenar os movimentos dos barcos, do torpedo e da bala criámos rotinas de interrupções (*rotinaclock1* e *rotinaclock2*) que entram em consonância com os estados referidos na secção 2.5. Por exemplo, a função *controlo_ck1* controla as rotinas associadas à *mexe_barco* (*mexebarco1* e *mexebarco2*) pois, dependendo do estado do clock 1, decide se as executa ou não. De modo a controlar estes processos, utilizam-se os estados dos clocks que depois são guardados num endereço de uma WORD.

2.5. Variáveis de Estado

Neste projeto, usamos vários estados relacionados com os objetos e tornam a abordagem das colisões, por exemplo, muito mais eficiente para o programa de calcular. Os estados são, em geral, ON e OFF que têm como valor 1 e 0, respetivamente.

- *estado_geral* (permite saber se o jogo está a decorrer):
 - 1 = está a decorrer
 - 0 = está parado
- *estado_clock_1* / *estado_clock_2* (clock ativo/inativo):
 - 1 = ativo
 - 0 = inativo
- *estado_torpedo* (torpedo disparado/encontrar-se no ecrã ou não):
 - 1 = disparado
 - 0 = por disparar
- *estado_barco1* / *estado_barco2* (barco desenhados ou não):
 - 1 = desenhado
 - 0 = apagado

2.6. Interrupções

Rotinas de interrupções (*rotinaclock1* e *rotinaclock2*) são inicializadas numa tabela (*tabela_interrup*) e, por conseguinte, tratam de ligar/desligar as variáveis (em memória) que representam os estados do clock 1 e clock 2.

Ligada a estas rotinas estão as rotinas de controlo (*controlo_ck1* e *controlo_ck2*) que utilizam os valores destas variáveis para averiguar se há de chamar as rotinas respetivas (clock 1 – mexer os barcos; clock 2 – mexer o torpedo e a bala).

2.7. Rotinas

- *paragem_forcada*: Rotina que recebe a tecla premida e coloca o estado geral do jogo a OFF caso a tecla premida seja a correspondente à designada (F).

- *ativa_geral*: Rotina que ativa o estado geral do jogo (altera o estado para ON).
- *reset_jogo*: Rotina que faz reset a todas as coordenadas, estados e display em memória.

Acedemos a todas as variáveis que inicializámos nas WORDs, acendendo diretamente aos endereços e restituindo os seus conteúdos.

- *contador_random*: Rotina que recebe o endereço de memória que contém o valor random, incrementa 1 e atualiza o valor na memória, que mais tarde será usado para definir em que linha os barcos fazem spawn quando chegam a coluna final.

Começamos com o contador a 0 (que dá reset quando chega a 1000), que incrementa assincronamente com os flancos ascendentes do clock 1 e depois de guardar o valor do contador quando o barco chega à última coluna, recupera-se os 3 bits de menor peso com uma máscara (0007H) e guarda esse valor na memória da coordenada das linhas.

- *controlo_ck1*: Rotina de controlo que verifica o estado do clock1 e caso este esteja ativo executa as duas rotinas designadas pelo enunciado, as rotinas que movimentam os barcos
- *controlo_ck2*: Rotina de controlo que verifica o estado do clock2 e caso este esteja ativo executa as duas rotinas designadas pelo enunciado, a rotina que mexe a bala e a rotina que envolve todo a questão do torpedo (ser lançado, continuar a mover e colisões).
- *rotinaclock1* e *rotinaclock2*: Rotinas de interrupção que colocam os estados do clock1 e do clock2 ativos.
- *rotina_torpedo*: Rotina que trata de ver se o torpedo foi disparado, do seu movimento e das suas colisões.

Somente guarda em registos os valores do estado e das coordenadas e depois chama as rotinas *mexe_torpedo*, *colisao_barco1*, *colisao_barco2*.

- *teclado*: Rotina que varre o teclado e guarda no R0 o valor dessa tecla premida.

Varre todo o teclado utilizando contadores de linhas e colunas, lê o input do porto de entrada e utiliza uma máscara para “retirar” os 4 bits de menor peso (000FH) e retorna esse valor para ser utilizado pela rotina *escolhe_funcao*.

- *desenha*: Rotina que recebe como argumentos as coordenadas de referência de todos os objetos juntamente com o primeiro endereço da string a desenhar e percorre todas as linhas e colunas da string, enviando de cada vez o seu conteúdo para a rotina *desenha_pixel*.

Recebe as rotinas que inserem os endereços dos dados de cada objeto em registos/parâmetros (e.g. *desenha_submarino*) e chama a rotina *desenha_pixel*, coluna*linha vezes.

- *desenha_pixel*: Rotina auxiliar que consoante os parâmetros da linha e coluna que recebe, liga/desliga o pixel correspondente, dependendo do valor da variável geral R10.
- *apaga*: Rotina que recebe como argumentos as coordenadas de referência de todos os objetos juntamente com o primeiro endereço da string a apagar e percorre todas as linhas e colunas da string, enviando de cada vez o seu conteúdo para a rotina *desenha_pixel*.

Recebe as rotinas que inserem os endereços dos dados de cada objeto em registos/parâmetros (e.g. *apaga_submarino*) e chama a rotina *desenha_pixel*, coluna*linha vezes, mas com o registo geral R10 = 0 (valor dos pixéis).

- *escolhe_funcao*: Rotina que avalia a tecla premida e chama a função por utilizar.

Chama funções (e.g. *mexec*, para mexer para cima) que incrementam/decrementam nas coordenadas do submarino guardadas em memória. Caso a tecla não esteja definida funcionalmente, apenas sai da rotina.

- *mexe_barcos*:

Rotina que recebe argumentos de um dos barcos, incrementa os valores das colunas e caso se encontrem já encostados ao lado direito, através do endereço que contém o número random, geram uma linha random onde irão reaparecer.

- *mexebala:*

Rotina que trata de todas as operações da bala, desde de o seu apagar, desenhar e verificar colisões com o submarino.

- *colisao_bala: Rotina que determina se há colisão entre a bala e o submarino. Se este caso se apresentar, o estado geral do jogo passa a OFF, interrompendo o ciclo main.*

Caso haja colisão, altera o estado geral para OFF, o que se traduz em terminar o jogo. Este estado define-se pela colisão entre a bala e o submarino.

- *colisao_barcos: Rotina que determina se há colisão entre os barcos e o torpedo.*

Se há colisão, chama a rotina *pontuacao* e altera o estado do barco e do torpedo para OFF de modo a apagá-los posteriormente.

- *pontuacao: Rotina que trata de atualizar o display e de desligar o estado do torpedo, como também apagá-lo.*

3. Conclusões

Perante o que nos foi proposto (o desenvolvimento de um jogo com base nos parâmetros referidos na introdução), fomos capazes de concretizar todos os objetivos iniciais do trabalho. Por outro lado, não realizámos certas ideias e planos de maneira a melhorar o código existente. Para além de estes objetivos complementares que não viram fruição (como criar um efeito de explosão após as colisões dos barcos e fazer os barcos desaparecer aos poucos), a única parte do projeto que foi feita a mais foram os comentários que se encontravam presentes em todas as instâncias de PUSHs e POPs.

Após a escrita e diversos testes do nosso código, o programa agora corre sem qualquer tipo de percalços ou erros. Relativamente às opções tomadas, optámos por este caminho por se apresentar como o mais lógico e mais em linha com o nosso pensamento e os ensinamentos ao longo do semestre.

Ultimamente, certas melhorias poderiam ter sido executadas. Uma destas, seria o desenvolvimento de rotinas mais abstratas, de maneira a englobar mais casos. Uma outra que também se poderia ter concretizado, a criação de funções auxiliares, serviria para reduzir a quantidade de código duplicado presente na versão final.

4. Código Assembly

[illegible]



```
;Implementacao do barco1:
;
; O barco1(maior) esta inserido numa tabela 6x8, onde o ponto superior
; esquerdo serve de referencia para o seu desenho e movimentacoes.

;Implementacao do barco2:
;
; O barco2(menor) esta inserido numa tabela 5x6, onde o ponto superior
; esquerdo serve de referencia para o seu desenho e movimentacoes.

;Implementacao do torpedo:
;
; O barco2(menor) esta inserido numa tabela 3x1, onde o ponto superior
; esquerdo serve de referencia para o seu desenho, movimentacoes e colisao.

;Implementacao da bala:
;
; O bala esta inserido numa tabela 1x1, onde o ponto superior
; esquerdo serve de referencia para o seu desenho, movimentacoes e colisao.

PLACE 3000H
submarino:                                STRING  3,7

                                         STRING  0,0,0,1,1,0,0
                                         STRING  0,0,0,0,1,0,0
                                         STRING  1,1,1,1,1,1,1

barco1:                                  STRING  6,8

                                         STRING  0,1,0,0,0,0,0,0
                                         STRING  0,0,1,0,0,0,0,0
                                         STRING  0,0,1,0,0,0,0,0
                                         STRING  1,1,1,1,1,1,1,1
                                         STRING  0,1,1,1,1,1,1,0
                                         STRING  0,0,1,1,1,1,0,0

barco2:                                  STRING  5,6

                                         STRING  0,1,0,0,0,0
                                         STRING  0,0,1,0,0,0
                                         STRING  0,0,1,0,0,0
                                         STRING  1,1,1,1,1,1
                                         STRING  0,1,1,1,1,0

torpedo:                                 STRING  3,1

                                         STRING  1
                                         STRING  1
                                         STRING  1

bala:                                    STRING  1,1

                                         STRING  1

game_over:

                                         STRING 0FFH, 0FFH, 0FFH, 0FFH
                                         STRING 0FFH, 0FFH, 0FFH, 0FFH
                                         STRING 0BFH, 0FFH, 0FFH, 0FDH
                                         STRING 09FH, 0FFH, 0FFH, 0F9H
                                         STRING 080H, 000H, 000H, 001H
                                         STRING 08FH, 0BEH, 0FBH, 0E1H
                                         STRING 088H, 022H, 0AAH, 001H
                                         STRING 08BH, 0BEH, 0ABH, 0E1H
                                         STRING 088H, 0A2H, 0AAH, 001H
                                         STRING 08FH, 0A2H, 0ABH, 0E1H
                                         STRING 080H, 000H, 000H, 001H
                                         STRING 08FH, 0A2H, 0FBH, 0C1H
                                         STRING 088H, 0B6H, 082H, 021H
                                         STRING 088H, 094H, 0FBH, 0E1H
```



game_over:

```
STRING 0FFH, 0FFH, 0FFH, 0FFH
STRING 0FFH, 0FFH, 0FFH, 0FFH
STRING 0BFH, 0FFH, 0FFH, 0FDH
STRING 09FH, 0FFH, 0FFH, 0F9H
STRING 080H, 000H, 000H, 001H
STRING 08FH, 0BEH, 0FBH, 0E1H
STRING 088H, 022H, 0AAH, 001H
STRING 08BH, 0BEH, 0ABH, 0E1H
STRING 088H, 0A2H, 0AAH, 001H
STRING 08FH, 0A2H, 0ABH, 0E1H
STRING 080H, 000H, 000H, 001H
STRING 08FH, 0A2H, 0FBH, 0C1H
STRING 088H, 0B6H, 082H, 021H
STRING 088H, 094H, 0FBH, 0E1H
STRING 088H, 09CH, 082H, 061H
STRING 08FH, 088H, 0FAH, 031H
STRING 080H, 000H, 000H, 001H
STRING 080H, 000H, 000H, 001H
STRING 080H, 000H, 000H, 001H
STRING 080H, 000H, 000H, 001H
STRING 088H, 020H, 040H, 001H
STRING 08BH, 0A0H, 060H, 0C1H
STRING 082H, 007H, 0F0H, 041H
STRING 083H, 080H, 067H, 0F1H
STRING 080H, 000H, 040H, 001H
STRING 080H, 000H, 000H, 001H
STRING 080H, 000H, 000H, 001H
STRING 080H, 000H, 000H, 001H
STRING 09FH, 0FFH, 0FFH, 0F9H
STRING 0BFH, 0FFH, 0FFH, 0FDH
STRING 0FFH, 0FFH, 0FFH, 0FFH
STRING 0FFH, 0FFH, 0FFH, 0FFH
```

PLACE 5000H

coor_mem_sub:	WORD	22	;coordenadas do ponto de referência do submarino
	WORD	13	
coor_mem_b1:	WORD	4	;coordenadas do ponto de referência do barco 1
	WORD	1	
coor_mem_b2:	WORD	5	;coordenadas do ponto de referência do barco 2
	WORD	15	
coor_torpedo:	WORD	0	;coordenadas do ponto de referência do torpedo
	WORD	0	
coor_bala:	WORD	27	;coordenadas do ponto de referência da bala
	WORD	1	
tabela_interrup:	WORD	rotinaclock1	
	WORD	rotinaclock2	
estado_torpedo:	WORD	0	;estado que permite saber se o torpedo se encontra disparado
estado_clock_1:	WORD	0	;estado que permite saber se o clock1 se encontra ativo/inativo
estado_clock_2:	WORD	0	;estado que permite saber se o clock2 se encontra ativo/inativo
estado_geral:	WORD	1	;estado que permite saber se o jogo esta a decorrer ou nao
gera_random:	WORD	0	;valor que o contador vai incrementando
hitpoints:	WORD	0	;local onde se regista a quantidade de pontos do jogador numa partida
estado_barcol:	WORD	1	;estado do barcol que permite saber se este se encontra desenhado ou nao



```
estado_barco2:      WORD    1          ;estado do barco2 que permite saber se este se encontra desenhado ou nao
; *****
; * Código
; *****
PLACE      0000H          ;codigo tem de começar em 0000H

inicio:
    MOV     BTE,tabela_interrup      ;mexe o endereço da tabela de interrupcoes para o registo especial
    EIO
    EII
    MOV     SP,SP_inicial             ;inicializa SP para a palavra a seguir a ultima da pilha
    MOV     R0,Tecla                  ;variavel global que contem o valor da tecla
    MOV     R10,0                     ;variavel global que permite ligar/desligar o pixel pretendido
    MOV     R7,ON                     ;variavel global com valor 1
    MOV     R2, Tecla_Reset           ;variavel global com o valor da tecla que permite fazer reset
    CALL    limpa_ecra                ;chama a rotina que limpa o ecrã
    CALL    desenha_submarino         ;chama a rotina que desenha o submarino
    CALL    desenha_barco1           ;chama a rotina que desenha o barco1
    CALL    desenha_barco2           ;chama a rotina que desenha o barco2
    EI                                  ;enable das interrupcoes em geral

main:
    ;ciclo geral

    MOV     R0,Tecla                  ;faz reset a variavel da tecla
    MOV     R1,LinhaInicial           ;reset no número da linha em que o teclado se encontra

    CALL    teclado                   ;rotina que percorre o teclado e devolve a tecla premida/nao premida
    MOV     R8,estado_geral           ;variavel global que contem o endereço de memoria do estado do jogo
    MOV     R9,[R8]                  ;variavel global com o valor do estado do jogo
    CMP     R7,R9                     ;verifica se o jogo esta a decorrer ou nao
    JNE     termina_jogo             ;caso o jogo nao esteja a decorrer
    CALL    paragem_forçada           ;rotina que verifica se a tecla de reset for premida,caso seja coloca o estado geral OFF
    CALL    contador_random           ;rotina que devolve um numero random

    CALL    escolhe_funcao            ;rotina que escolhe a funcao de movimento consoante a tecla premida
    CALL    controlo_ck1              ;rotina que executa as rotinas associadas ao clock1 caso este se encontre ativo
    CALL    controlo_ck2              ;rotina que executa as rotinas associadas ao clock2 caso este se encontre ativo
    JMP     main                      ;volta a repetir o ciclo main

termina_jogo:
    CALL    limpa_ecra                ;chama a rotina para limpar o ecrã
    CALL    ecrã_final               ;chama a rotina que imprime o ecrã final

espera_c:
    CALL    teclado                   ;chama a rotina do teclado quando se encontra no ecrã final
    CMP     R0,R2                     ;compara a tecla premida com a tecla de reset
    JZ      reset_ao_jogo             ;se a tecla corresponde,coloca todos os objetos e variaveis com os seus valores iniciais e ativa o geral
    JMP     espera_c                 ;enquanto for diferente , fica em loop a receber o teclado

reset_ao_jogo:
    CALL    reset_jogo                ;chama a rotina para dar reset as variaveis de coordenadas e de estado
    CALL    ativa_geral               ;chama a rotina que ativa o estado geral do jogo
    JMP     inicio                    ;volta a correr o programa inteiro

; * -- Ativa Geral -----
; *
; * Descricao: Rotina que ativa o estado geral do jogo
; *
; * Parametros: --
; * Retorna: --
; * Destroi: R1,R2
; * Notas: --
ativa_geral:
    PUSH    R1
    PUSH    R2
    MOV     R1,estado_geral
```



```
; * -- Ativa Geral -----  
; *  
; * Descricao: Rotina que ativa o estado geral do jogo  
; *  
; * Parametros: --  
; * Retorna: --  
; * Destroi: R1,R2  
; * Notas: --  
ativa_geral:  
    PUSH R1  
    PUSH R2  
    MOV R1,estado_geral  
    MOV R2,ON  
    MOV [R1],R2  
    POP R2  
    POP R1  
    RET  
; * -- Reset jogo -----  
; *  
; * Descricao: Rotina que faz reset a todas as coordenadas ,estados e display em memoria  
; *  
; * Parametros: --  
; * Retorna: --  
; * Destroi: R1,R2,R3,R4  
; * Notas: --  
reset_jogo:  
  
    PUSH R1  
    PUSH R2  
    PUSH R3  
    PUSH R4  
    MOV R2, coor_mem_sub  
    MOV R3, 22  
    MOV [R2], R3  
    ADD R2, 2  
    MOV R3, 13  
    MOV [R2], R3  
    MOV R2, coor_mem_b1  
    MOV R3, 4  
    MOV [R2], R3  
    ADD R2, 2  
    MOV R3, 1  
    MOV [R2], R3  
    MOV R2, coor_mem_b2  
    MOV R3, 5  
    MOV [R2], R3  
    ADD R2, 2  
    MOV R3, 15  
    MOV [R2], R3  
    MOV R2, coor_torpedo  
    MOV R3, 0  
    MOV [R2], R3  
    ADD R2, 2  
    MOV R3, 0  
    MOV [R2], R3  
    MOV R2, coor_bala  
    MOV R3, 27  
    MOV [R2], R3  
    ADD R2, 2  
    MOV R3, 1  
    MOV [R2], R3  
    MOV R2, estado_torpedo  
    MOV R3, 0  
    MOV [R2], R3  
    MOV R2, estado_clock_1  
    MOV R3, 0  
    MOV [R2], R3  
    MOV R2, estado_clock_2  
    MOV R3, 0
```



```
MOV R3, 15
MOV [R2], R3
MOV R2, coor_torpedo
MOV R3, 0
MOV [R2], R3
ADD R2, 2
MOV R3, 0
MOV [R2], R3
MOV R2, coor_bala
MOV R3, 27
MOV [R2], R3
ADD R2, 2
MOV R3, 1
MOV [R2], R3
MOV R2, estado_torpedo
MOV R3, 0
MOV [R2], R3
MOV R2, estado_clock_1
MOV R3, 0
MOV [R2], R3
MOV R2, estado_clock_2
MOV R3, 0
MOV [R2], R3
MOV R2, estado_geral
MOV R3, 1
MOV [R2], R3
MOV R2, gera_random
MOV R3, 0
MOV [R2], R3
MOV R2, hitpoints
MOV R3, 0
MOV R4, displays
MOV [R4], R3
MOV [R2], R3
MOV R2, estado_barcol
MOV R3, 1
MOV [R2], R3
MOV R2, estado_barco2
MOV R3, 1
MOV [R2], R3

sai_reset_jogo:
POP R4
POP R3
POP R2
POP R1
RET

/* -- Ecra Final -----
/*
/* Descricao: Rotina que imprime o ecra final
/*
/* Parametros: --
/* Retorna: --
/* Destroi: R1,R2,R3,R4,R5
/* Notas: --
ecra_final:
PUSH R1
PUSH R2
PUSH R3
PUSH R4
PUSH R5
MOV R1, 0
MOV R2, game_over
MOV R3, bytes_ecra
MOV R5, PrimeiroPixel
imprime_byte:
MOV R4, [R2]
MOV [R5], R4
;primeiro endereco da string que contem o ecra final
;numero de bytes que o ecra contem
;primeiro endereco do ecra
;passa o valor da string atual para o R4
;passa o valor da string para o endereco do ecra
```



```
        MOVB [R5],R4                ;passa o valor da string para o endereço do ecrã
        ADD  R1,1                  ;incrementa o contador que serve de condição de paragem
        ADD  R5,1                  ;incrementa o endereço do byte do ecrã a analisar
        ADD  R2,1                  ;incrementa o endereço da string a utilizar
        CMP  R1,R3                 ;verifica se já chegamos ao último byte do ecrã
        JZ   sai_ecra_final        ;se sim sai da rotina
        JMP  imprime_byte          ;caso contrário volta a repetir o ciclo

sai_ecra_final:
        POP  R5
        POP  R4
        POP  R3
        POP  R2
        POP  R1
        RET

;* -- Paragem Forçada -----
;*
;* Descrição: Rotina que recebe a tecla premida e coloca o estado geral do jogo OFF caso
;* a tecla premida seja a correspondente a designada(F).
;*
;* Parametros: R0(tecla premida)
;* Retorna: --
;* Destroi: R1,R2,R3
;* Notas: --
paragem_forcada:
        PUSH R1
        PUSH R2
        PUSH R3
        MOV  R1,Tecla_paragem      ;tecla designada para parar o jogo forcadamente
        CMP  R0,R1                 ;compara a tecla premida com a tecla designada
        JZ   desliga_geral         ;se corresponder,desligar o estado geral do programa
sai_paragem_forcada:
        POP  R3
        POP  R2
        POP  R1
        RET

desliga_geral:
        MOV  R2,estado_geral
        MOV  R3,OFF
        MOV  [R2],R3
        JMP  sai_paragem_forcada

;* -- Contador Random -----
;*
;* Descrição: Rotina que recebe o endereço de memória que contém o valor random, incrementa 1
;* e atualiza o valor na memória, que mais tarde será usado para definir em que linha os barcos
;* fazem spawn quando chegam a coluna final
;*
;* Parametros: --
;* Retorna: --
;* Destroi: R1,R2,R3,R4
;* Notas: O número máximo permitido ao contador chegar é 1000,quando da sua chegada retorna a 0
contador_random:
        PUSH R1
        PUSH R2
        PUSH R3
        PUSH R4
        MOV  R1,gera_random        ;endereço que contém o valor do número random
        MOV  R2,[R1]               ;coloca-se em R2 esse valor
        MOV  R3,max_random         ;coloca-se em R3 o valor máximo que pode atingir
        MOV  R4,0                  ;reset ao contador
        ADD  R2,1                  ;incrementa no contador
        MOV  [R1],R2               ;ativa o valor na memória
        CMP  R2,R3                 ;compara se já chegou ao valor máximo
        JZ   reset_contador
```




```
sai_contador_random:
    POP R4
    POP R3
    POP R2
    POP R1
    RET

reset_contador:
    MOV [R1],R4
    JMP sai_contador_random

;* -- Controlo CK1 -----
;*
;* Descricao: Rotina de controlo que verifica o estado do clock1 e caso este esteja ativo
;* executa as duas rotinas designadas pelo enunciado, as rotinas que movimentam os barcos
;*
;* Parametros: --
;* Retorna: --
;* Destroi: R1,R2,R3
;* Notas: --
controlo_ck1:
    PUSH R1
    PUSH R2
    PUSH R3
    MOV R1,estado_clock_1
    MOV R2,[R1]
    MOV R3,OFF
    CMP R3,R2
    JNZ ativa_movimento_barcos      ; salta caso o estado do clock1 esteja ligado

sai_controlo_ck1:
    POP R3
    POP R2
    POP R1
    RET

ativa_movimento_barcos:
    CALL mexebarco1                ;chama a rotina que movimenta o barco1
    CALL mexebarco2                ;chama a rotina que movimenta o barco2
    MOV [R1],R3                    ;volta a desligar o estado do clock1
    JMP sai_controlo_ck1

;* -- Controlo CK2 -----
;*
;* Descricao: Rotina de controlo que verifica o estado do clock2 e caso este esteja ativo
;* executa as duas rotinas designadas pelo enunciado, a rotina que mexe a bala e a
;* rotina que envolve todo a questao do torpedo(ser lancado,continuar a mover e colisoes).
;*
;* Parametros: --
;* Retorna: --
;* Destroi: R1,R2,R3
;* Notas: --
controlo_ck2:
    PUSH R1
    PUSH R2
    PUSH R3
    MOV R1,estado_clock_2
    MOV R2,[R1]
    MOV R3,OFF
    CMP R3,R2
    JNZ ativa_rotina_torpedo
    JMP sai_controlo_ck2      ; salta caso o estado do clock2 esteja ligado

sai_controlo_ck2:
    POP R3
    POP R2
```



```
sai_controlo_ck2:
    POP R3
    POP R2
    POP R1
    RET
ativa_rotina_torpedo:
    CALL mexebala
    CALL rotina_torpedo
    MOV [R1],R3
    JMP sai_controlo_ck2
; * -- Rotinaclock1 -----
; *
; * Descricao: Rotina de interrupcao que coloca o estado do clock1 ativo.
; *
; * Parametros: --
; * Retorna: --
; * Destroi: R1,R2
; * Notas: --
rotinaclock1:
    PUSH R1
    PUSH R2
    MOV R1,estado_clock_1
    MOV R2,ON
    MOV [R1],R2
    POP R2
    POP R1
    RFE
; * -- Rotinaclock2 -----
; *
; * Descricao: Rotina de interrupcao que coloca o estado do clock2 ativo.
; *
; * Parametros: --
; * Retorna: --
; * Destroi: R1,R2
; * Notas: --
rotinaclock2:
    PUSH R1
    PUSH R2
    MOV R1,estado_clock_2
    MOV R2,ON
    MOV [R1],R2
    POP R2
    POP R1
    RFE
; * -- Rotina Torpedo -----
; *
; * Descricao: Rotina que trata de ver se o torpedo foi disparado, do seu movimento
; * e das suas colisoes.
; *
; * Parametros: --
; * Retorna: --
; * Destroi: R2,R3,R4,R5,R6,R7,R8
; * Notas: --
rotina_torpedo:
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8
    MOV R7,OFF
    MOV R8,estado_clock_2
    MOV R5,estado_torpedo
    MOV R6,[R5]
    MOV R2,0
;estado atual do torpedo
;limite superior do ecran
```



```
MOV R8,estado_clock2
MOV R5,estado_torpedo
MOV R6,[R5] ;estado atual do torpedo
MOV R2,0 ;limite superior do ecrã
MOV R3,coord_torpedo
MOV R4,[R3] ;coordenada y do torpedo

CMP R7,R6 ;Caso o estado do torpedo esteja a 0, entramos na rotina que verifica se foi premida a tecla para disparar
JZ talvez_dispare ;Caso contrario movimentamos o torpedo ate chegar ao limite superior ou ter uma colisao com um dos barcos

CMP R2,R4 ;caso o torpedo esteja no ecrã,verifica se ja chegou ao limite superior
JZ reset_torpedo ;se chegou ao limite,apaga o torpedo e mete o estado do torpedo OFF
CALL mexe_torpedo ;nao tendo chegado ao limite , chama a rotina de mexer o torpedo
CALL colisao_barco1 ;chama a rotina para verificar se ha colisao entre o torpedo e o barco1
CALL colisao_barco2 ;chama a rotina para verificar se ha colisao entre o torpedo e o barco2
JMP mete_clock2_off

reset_torpedo:
CALL apaga_torpedo ;chama a rotina de apagar o torpedo
MOV R6,OFF
MOV [R5],R6 ;desliga o estado do torpedo
JMP mete_clock2_off

talvez_dispare:
CALL verifica_lanca_torpedo ;chama a rotina que verifica se e para lancar o torpedo
JZ mete_clock2_off

mete_clock2_off:
MOV [R8],R7
POP R8
POP R7
POP R6
POP R5
POP R4
POP R3
POP R2
RET

;* -- Limpa Ecrã -----
;*
;* Descricao: Rotina inicial que limpa o ecrã.
;*
;* Parametros: --
;* Retorna: --
;* Destroi: R1,R2,R3
;* Notas: Esta rotina serve apenas para eliminar a hipotese do
;* ecrã de pixeis começar com algum pixel inicializado

limpa_ecra:
PUSH R1
PUSH R2
PUSH R3
MOV R1,PrimeiroPixel ;atribui a R1 o endereço do primeiro pixel
MOV R2,UltimoPixel ;atribui a R2 o endereço do ultimo pixel
MOV R3,limpa_pixel ;mascara para limpar cada pixel do ecrã de 128bytes

proximo_endereco:
CMP R1,R2 ;compara endereços do atual presente com o ultimo
JZ sai_limpa_ecra ;se for o ultimo endereço termina a rotina de limpar o ecrã
MOV [R1],R3 ;coloca o presente endereço com o valor pretendido
ADD R1,2 ;incrementa 2 para avançar de endereço
JMP proximo_endereco ;se nao for o ultimo endereço

sai_limpa_ecra:
POP R3
POP R2
```



```
sai_limpa_ecra:
    POP R3
    POP R2
    POP R1
    RET

;* -- Teclado -----
;*
;* Descricao: Rotina que varre o teclado e guarda no R0 o valor dessa tecla premida
;*
;* Parametros: R1,R3(linha testada,coluna premida)
;* Retorna:    R0 -(tecla premida)
;* Destroi:   R2,R3,R4,R6,R7,R8,R9,R10
;* Notas:    --

teclado:
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8
    PUSH R9
    PUSH R10
    MOV R2,OUT2 ;R2 com o endereco do porto de saida do teclado
    MOV R6,FimdeCiclo ;R6 com a variavel da linha final para comparacoes
    MOV R8,ContadorC ;inicializa variavel do contador das colunas do teclado
    MOV R9,ContadorL ;inicializa variavel do contador das linhas do teclado
    MOV R10,PIN ;R10 com o endereco de porto de entrada
    MOV R5,mascara_teclado

ciclo_teclado:
    MOVB [R2], R1 ;escrever no porto de saida
    MOV R7,R1 ;copia da linha presente
    MOVB R3, [R10] ;ler do porto de entrada
    AND R3, R5 ;afectar as flags (MOV's nao afectam as flags)
    JZ verificaciclo_teclado ;se nenhuma tecla premida salta
    MOV R4, R3 ;guardar tecla premida em R4

;transformacoes das linhas e colunas para decimal

trocaC:
    ADD R8,1 ;incrementa contador por 1
    SHR R3,1 ;divide o numero da coluna por 2
    JNZ trocaC ;se R3 nao for 0 ainda,repete
    SUB R8,1 ;subtrai 1 para ajustar as contas
    JMP trocaL ;prosegue para transformar agora as linhas

trocaL:
    ADD R9,1 ;incrementa contador por 1
    SHR R7,1 ;divide o numero da linha por 2
    JNZ trocaL ; se R7 nao for 0 ainda,repete
    SUB R9,1 ;subtrai 1 para ajustar as contas
    SHL R9,2 ;formula fornecida pelo professor
    MOV R0,0
    ADD R0, R9 ;
    ADD R0, R8 ;
    JMP sai_teclado ;trata agora de resetar os contadores

verificaciclo_teclado:
    SHL R1,1 ;avança para a linha seguinte
    SHR R6,1 ;verifica se ja esta na 4 linha
    JZ resetciclo_teclado ;no caso de termos chegado a ultima linha
    JMP ciclo_teclado ;avança para a linha seguinte se nao estiver concluido o varrimento

resetciclo_teclado:
```



```
resetciclo_teclado:
    MOV R6,FimdeCiclo           ;reset a variavel de contagem de linhas
    MOV R1,LinhaInicial
    JMP sai_teclado

sai_teclado:
    POP R10
    POP R9
    POP R8
    POP R7
    POP R6
    POP R5
    POP R4
    POP R3
    POP R2
    RET

;* -- Desenha Submarino -----
;*
;* Descricao: Rotina que recebe as coordenadas do ponto de referencia do submarino
;* e passa as coordenadas e o primeiro endereço da string do submarino para a rotina desenha geral
;*
;* Parametros: --
;* Retorna:    --
;* Destroi: R1,R7,R8
;* Notas:    --

desenha_submarino:
    PUSH R1
    PUSH R7
    PUSH R8
    MOV R1, coor_mem_sub
    MOV R7, [R1]           ;coordenada x do submarino
    ADD R1, 2
    MOV R8, [R1]           ;coordenada y do submarino
    MOV R1,submarino       ;contem o endereço que contem o numero de linhas do submarino
    CALL desenha           ;chama a rotina desenha geral

    POP R8
    POP R7
    POP R1
    RET

;* -- Desenha Barcol -----
;*
;* Descricao: Rotina que recebe as coordenadas do ponto de referencia do barcol
;* e passa as coordenadas e o primeiro endereço da string do barcol para a rotina desenha geral
;*
;* Parametros: --
;* Retorna:    --
;* Destroi: R1,R7,R8
;* Notas:    --

desenha_barcol:
    PUSH R1
    PUSH R7
    PUSH R8
    MOV R1, coor_mem_b1
    MOV R7, [R1]           ;coordenada x do barcol
    ADD R1, 2
    MOV R8, [R1]           ;coordenada y do barcol
    MOV R1, barcol         ;contem o endereço que contem o numero de linhas do barcol
    CALL desenha           ;chama a rotina desenha geral

    POP R8
    POP R7
    POP R1
    RET

;* -- Desenha Barco2 -----
```



```
;* -- Desenha Barco2 -----
;*
;* Descricao: Rotina que recebe as coordenadas do ponto de referencia do barco2
;* e passa as coordenadas e o primeiro endereco da string do barco2 para a rotina desenha geral
;*
;* Parametros: --
;* Retorna: --
;* Destroi: R1,R7,R8
;* Notas: --
desenha_barco2:
    PUSH    R1
    PUSH    R7
    PUSH    R8
    MOV     R1, coor_mem_b2
    MOV     R7, [R1]          ;coordenada x do barco2
    ADD     R1, 2
    MOV     R8, [R1]          ;coordenada y do barco2
    MOV     R1,barco2         ;contem o endereco que contem o numero de linhas do barco2
    CALL    desenha           ;chama a rotina desenha geral

    POP     R8
    POP     R7
    POP     R1
    RET

;* -- Desenha Bala -----
;*
;* Descricao: Rotina que recebe as coordenadas do ponto de referencia do bala
;* e passa as coordenadas e o primeiro endereco da string da bala para a rotina desenha geral
;*
;* Parametros: --
;* Retorna: --
;* Destroi: R1,R7,R8
;* Notas: --
desenha_bala:
    PUSH    R1
    PUSH    R7
    PUSH    R8
    MOV     R1, coor_bala
    MOV     R7, [R1]          ;coordenada x da bala
    ADD     R1, 2
    MOV     R8, [R1]          ;coordenada y da bala
    MOV     R1,bala           ;contem o endereco que contem o numero de linhas da bala
    CALL    desenha           ;chama a rotina desenha geral

    POP     R8
    POP     R7
    POP     R1
    RET

;* -- Desenha -----
;*
;* Descricao: Rotina que recebe como argumentos as coordenadas de referencia de todos os objetos
;* juntamente com o primeiro endereco da string a desenhar,e percorre todas as linhas e colunas da
;* string , enviando de cada vez o seu conteudo para a rotina desenha pixel.
;*
;* Parametros: R1,R7,R8
;* Retorna: --
;* Destroi: R2,R3,R4,R5,R6,R10
;* Notas: --
desenha:
    PUSH    R2
    PUSH    R3
    PUSH    R4
    PUSH    R5
    PUSH    R6
    PUSH    R10
    MOV     R2, 0             ;contador colunas
    MOV     R6, R8             ;copia coordenada y
    MOV     R4, [R1]           ;numero de linhas
```



```
MOV B    R4, [R1]      ;numero de linhas
ADD      R1, 1         ;incrementa endereço
MOV B    R5, [R1]      ;numero de colunas
ADD      R1, 1
MOV      R3, 1         ;contador linhas

desenha_colunas:
  CMP     R2, R5        ;verifica se ja chegou a coluna final
  JZ      desenha_linhas ;se sim avança para a linha seguinte e faz reset nas colunas
  MOV B   R10, [R1]     ;passa a R10 o estado do proximo pixel a desenhar
  CALL    desenha_pixel ;chama a rotina que desenha o pixel
  ADD     R1, 1         ;incrementa o endereço
  ADD     R2, 1         ;incrementa o contador de colunas
  ADD     R8, 1         ;incrementa a coordenada y
  JMP     desenha_colunas

desenha_linhas:
  CMP     R4, R3        ;verifica se ja desenhamos todas as linhas da string
  JZ      sai_desenha   ;caso se verifique a condicao anterior, sai da rotina
  SUB     R4, 1         ;decrementa o numero da linha
  ADD     R7, 1         ;incrementa a coordenada x
  MOV     R8, R6        ;volta a restituir o valor das colunas
  MOV     R2, 0         ;reset ao contador das colunas
  JMP     desenha_colunas

sai_desenha:
  POP     R10
  POP     R6
  POP     R5
  POP     R4
  POP     R3
  POP     R2
  RET

; * -- Desenha Pixel -----
; *
; * Descriçao: Rotina auxiliar que consoante os parametros da linha e coluna que recebe,
; * liga/desliga o o pixel correspondente, dependendo do valor da variavel global R10.
; *
; * Parametros: R7,R8,R10(copias dos valores do centro x,y de quaisquer objetos,interruptor do p:
; * Retorna:    --
; * Destroi:   R0,R1,R2,R3,R4,R5,R6,R9
; * Notas:    R10 sera o interruptor do pixel, caso seja utilizado pela rotina desenha submarino,
; * R10 tera o valor do pixel analisado,caso seja com a rotina apaga submarino,R10 tera o seu
; * o seu valor global,0.
desenha_pixel:
  PUSH R0
  PUSH R1
  PUSH R2
  PUSH R3
  PUSH R4
  PUSH R5
  PUSH R6
  PUSH R9

  MOV R0,PrimeiroPixel ;primeiro endereço do ecrã
  MOV R1,0              ;variavel inicializada a 0 para a formula
  MOV R9,7              ;bit de maior peso do byte
  MOV R3,R8              ;copia da coluna
  MOV R6,4

  MUL R6,R7              ;multiplica-se a linha por 4
  ADD R1,R0
  ADD R1,R6
  MOV R6,div_por_8
  DIV R3,R6
  ADD R1,R3              ;formula concluida 8000 + 4linha + coluna div 8
```



```
ADD R1,R3 ;formula concluida 8000 + 4linha + coluna div 8
MOV R3,R8

MOV R2,[R1] ;coloca em R2 o conteudo do byte calculado
MOV R4,div_por_8
MOD R3,R4 ;permite chegar a coluna dentro do byte selecionado
MOV R5,mascara_pixel ;coloca inicialmente 01H
procura_mascara:
CMP R3,R9 ;compara se e o bit de maior peso, senao decrementa
JZ liga_desliga ;se for o bit correto vai analisar se liga ou desliga
SHL R5,1 ;decrementa a mascara
SUB R9,1 ;decrementa a posicao do bit de maior peso
JMP procura_mascara ;volta a repetir o ciclo

liga_desliga:
CMP R10,1 ;salta para mascara1 se for para ligar o pixel
JZ mascara1
CMP R10,0 ;salta para mascara0 se for para desligar o pixel
JZ mascara0

mascara1:
OR R2,R5 ;ligo o bit na posicao requisitada
MOV R1,[R2] ;atualiza-se o conteudo do byte
JMP sai_desenha_pixel ;sai do ciclo para recuperar os valores da rotina e sair

mascara0:
NOT R5 ;nega-se a mascara, todos os bit ficam a 1 exceto o que queremos
AND R2,R5 ;desliga-se o bit que se pretende
MOV R1,[R2] ;atualiza-se o conteudo do byte
JMP sai_desenha_pixel ;sai do ciclo para recuperar os valores da rotina e sair

sai_desenha_pixel:
POP R9
POP R6
POP R5
POP R4
POP R3
POP R2
POP R1
POP R0
RET

;* -- Apaga Submarino -----
;*
;* Descricao: Rotina que recebe as coordenadas do ponto de referencia do submarino
;* e passa as coordenadas e o primeiro endereço da string do submarino para a rotina desenha geral
;*
;* Parametros:
;* Retorna: --
;* Destroi: R1,R7,R8
;* Notas: --

apaga_submarino:
PUSH R1
PUSH R7
PUSH R8
MOV R1, coor_mem_sub
MOV R7, [R1] ;coordenada x do submarino
ADD R1, 2
MOV R8, [R1] ;coordenada y do submarino
MOV R1,submarino ;tem o endereço que contem o numero de linhas
CALL apaga ;chama a rotina apaga geral
POP R8
POP R7
POP R1
RET
```




```
;* -- Apaga Barcol -----
;*
;* Descricao: Rotina que recebe as coordenadas do ponto de referencia do barcol
;* e passa as coordenadas e o primeiro endereço da string do barcol para a rotina desenha geral
;*
;* Parametros:
;* Retorna:  --
;* Destroi:   R1,R7,R8
;* Notas:  --

apaga_barcol:
    PUSH    R1
    PUSH    R7
    PUSH    R8
    MOV     R1, coor_mem_b1
    MOV     R7, [R1]          ;coordenada x do barcol
    ADD     R1, 2
    MOV     R8, [R1]          ;coordenada y do barcol
    MOV     R1, barcol         ;contem o endereço que contem o numero de linhas do barcol
    CALL    apaga              ;chama a rotina apaga geral
    POP     R8
    POP     R7
    POP     R1
    RET

;* -- Apaga Barco2 -----
;*
;* Descricao: Rotina que recebe as coordenadas do ponto de referencia do barco2
;* e passa as coordenadas e o primeiro endereço da string do barco2 para a rotina desenha geral
;*
;* Parametros:
;* Retorna:  --
;* Destroi:   R1,R7,R8
;* Notas:  --

apaga_barco2:
    PUSH    R1
    PUSH    R7
    PUSH    R8
    MOV     R1, coor_mem_b2
    MOV     R7, [R1]          ;coordenada x
    ADD     R1, 2
    MOV     R8, [R1]          ;coordenada y
    MOV     R1,barco2         ;contem o endereço que contem o numero de linhas do barco2
    CALL    apaga              ;chama a rotina apaga geral
    POP     R8
    POP     R7
    POP     R1
    RET

;* -- Apaga Bala -----
;*
;* Descricao: Rotina que recebe as coordenadas do ponto de referencia da bala
;* e passa as coordenadas e o primeiro endereço da string da bala para a rotina desenha geral
;*
;* Parametros:
;* Retorna:  --
;* Destroi:   R1,R7,R8
;* Notas:  --

apaga_bala:
    PUSH    R1
    PUSH    R7
    PUSH    R8
    MOV     R1, coor_bala
    MOV     R7, [R1]          ;coordenada x
    ADD     R1, 2
    MOV     R8, [R1]          ;coordenada y
    MOV     R1,bala           ;contem o endereço que contem o numero de linhas da bala
```



```
apaga_bala:
    PUSH    R1
    PUSH    R7
    PUSH    R8
    MOV     R1, coor_bala
    MOV     R7, [R1]          ;coordenada x
    ADD     R1, 2
    MOV     R8, [R1]          ;coordenada y
    MOV     R1,bala           ;contem o endereco que contem o numero de linhas da bala
    CALL    apaga             ;chama a rotina apaga geral
    POP     R8
    POP     R7
    POP     R1
    RET

; * -- Apaga -----
; *
; * Descricao: Rotina que recebe como argumentos as coordenadas de referencia de todos os objetos
; * juntamente com o primeiro endereco da string a apagar,e percorre todas as linhas e colunas da
; * string , enviando de cada vez o seu conteudo para a rotina desenha pixel.
; *
; * Parametros: R1,R7,R8
; * Retorna:    --
; * Destroi:   R2,R3,R4,R5,R6,R10
; * Notas:    --
apaga:
    PUSH    R2
    PUSH    R3
    PUSH    R4
    PUSH    R5
    PUSH    R6
    MOV     R2, 0              ;contador colunas
    MOV     R6, R8             ;copia coordenada y
    MOV     R4, [R1]           ;numero de linhas
    ADD     R1, 1              ;incrementa endereço
    MOV     R5, [R1]           ;numero de colunas
    MOV     R3, 1              ;contador linhas

apaga_colunas:
    CMP     R2, R5             ;verifica se ja chegou a coluna final
    JZ      apaga_linhas       ;se sim avanca para a linha seguinte e faz reset nas colunas
    CALL    desenha_pixel      ;passa a R10 o estado do proximo pixel a desenhar
    ADD     R2, 1              ;incrementa o contador de colunas
    ADD     R8, 1              ;incrementa a coordenada y
    JMP     apaga_colunas

apaga_linhas:
    CMP     R4, R3             ;verifica se ja desenhemos todas as linhas de string
    JZ      sai_apaga          ;caso se verifique a condicao anterior,sai da rotina
    SUB     R4, 1              ;decrementa o numero da linha
    ADD     R7, 1              ;incrementa a coordenada x
    MOV     R8, R6             ;volta a restituir o valor das colunas
    MOV     R2, 0
    JMP     apaga_colunas

sai_apaga:
    POP     R6
    POP     R5
    POP     R4
    POP     R3
    POP     R2
    RET

; * -- Escolhe Função -----
; *
; * Descricao: Rotina que avalia a tecla premida e chama a função por utilizar
; *
; * Parametros: R0(tecla premida)
```



```
;* -- Escolhe Função -----  
;*  
;* Descricao: Rotina que avalia a tecla premida e chama a função por utilizar  
;*  
;* Parametros: R0 (tecla premida)  
;* Retorna: --  
;* Destroi: R1,R2,R3,R5,R6  
;* Notas: --  
  
escolhe_funcao:  
    PUSH R1  
    PUSH R2  
    PUSH R3  
    PUSH R5  
    PUSH R6  
  
    MOV R3,coord_mem_sub ;endereço de memoria que contem a posicao x do ponto de referencia do submarino  
    MOV R5,[R3] ;coordenada x do submarino  
    MOV R6,[R3+2] ;coordenada y do submarino  
  
cima:  
    MOV R1,02H ;valor a testar da tecla  
    CMP R0,R1 ;verifica se a tecla premida corresponde ao movimento  
    JZ mexec ;se nao corresponder salta para testar nova condicao  
  
baixo:  
    MOV R1,0AH ;valor a testar da tecla  
    CMP R0,R1 ;verifica se a tecla premida corresponde ao movimento  
    JZ mexeb ;se nao corresponder salta para testar nova condicao  
  
direita:  
    MOV R1,07H ;valor a testar da tecla  
    CMP R0,R1 ;verifica se a tecla premida corresponde ao movimento  
    JZ mexed ;se nao corresponder salta para testar nova condicao  
  
esquerda:  
    MOV R1,05H ;valor a testar da tecla  
    CMP R0,R1 ;verifica se a tecla premida corresponde ao movimento  
    JZ mexee ;se nao corresponder salta para testar nova condicao  
  
direita_cima:  
    MOV R1,03H ;valor a testar da tecla  
    CMP R0,R1 ;verifica se a tecla premida corresponde ao movimento  
    JZ mexedc ;se nao corresponder salta para testar nova condicao  
  
direita_baixo:  
    MOV R1,0BH ;valor a testar da tecla  
    CMP R0,R1 ;verifica se a tecla premida corresponde ao movimento  
    JZ mexedb ;se nao corresponder salta para testar nova condicao  
  
esquerda_baixo:  
    MOV R1,09H ;valor a testar da tecla  
    CMP R0,R1 ;verifica se a tecla premida corresponde ao movimento  
    JZ mexeeb ;se nao corresponder salta para testar nova condicao  
  
esquerda_cima:  
    MOV R1,01H ;valor a testar da tecla  
    CMP R0,R1 ;verifica se a tecla premida corresponde ao movimento  
    JZ mexeec ;se nao corresponder salta para testar nova condicao  
  
sai_escolhe_funcao:  
    POP R6  
    POP R5  
    POP R3  
    POP R2  
    POP R1  
    RET  
  
;* -- Mexe para Cima -----
```



```
;* -- Mexe para Cima -----
;*
;* Descriçao: Rotina que calcula as novas posições centro x,y do submarino e chama as
;* rotinas necessarias para a movimentação da nave pretendida
;*
;* Parametros: R5(posicao centro x do submarino)
;* Retorna: --
;* Destroi: --
;* Notas: Esta rotina utiliza duas rotinas, uma delas para apagar o submarino na sua posição
;* atual, calcula a nova posição do centro x,y consoante o movimento e por fim chama a rotina
;* que permite desenhar o submarino, de acordo com o movimento

mexec:
    MOV R2,16
    CMP R2,R5
    JZ sai_escolhe_funcao

    CALL apaga_submarino          ;chama a rotina do apaga submarino

    SUB R5,1                      ;movimentação necessaria
    CALL atualiza_coor            ;chama a rotina que atualiza as coordenadas do submarino apos alteracao
    CALL desenha_submarino       ;chama a rotina do desenhar submarino novamente
    JMP sai_escolhe_funcao

;* -- Mexe para Baixo -----
;*
;* Descriçao: Rotina que calcula as novas posições centro x,y do submarino e chama as
;* rotinas necessarias para a movimentação da nave pretendida
;*
;* Parametros: R5(posicao centro x do submarino)
;* Retorna: --
;* Destroi: --
;* Notas: Esta rotina utiliza duas rotinas, uma delas para apagar o submarino na sua posição
;* atual, calcula a nova posição do centro x,y consoante o movimento e por fim chama a rotina
;* que permite desenhar o submarino, de acordo com o movimento

mexeb:
    MOV R2,29
    CMP R2,R5
    JZ sai_escolhe_funcao

    CALL apaga_submarino          ;chama a rotina do apaga submarino
    ADD R5,1                      ;movimentação necessaria
    CALL atualiza_coor            ;chama a rotina que atualiza as coordenadas do submarino apos alteracao
    CALL desenha_submarino       ;chama a rotina do desenhar submarino novamente
    JMP sai_escolhe_funcao

;* -- Mexe para a Direita -----
;*
;* Descriçao: Rotina que calcula as novas posições centro x,y do submarino e chama as
;* rotinas necessarias para a movimentação da nave pretendida
;*
;* Parametros: R6(posicao centro y do submarino)
;* Retorna: --
;* Destroi: --
;* Notas: Esta rotina utiliza duas rotinas, uma delas para apagar o submarino na sua posição
;* atual, calcula a nova posição do centro x,y consoante o movimento e por fim chama a rotina
;* que permite desenhar o submarino, de acordo com o movimento

mexed:
    MOV R2,25
    CMP R2,R6
    JZ sai_escolhe_funcao

    CALL apaga_submarino          ;chama a rotina do apaga submarino
    ADD R6,1                      ;movimentação necessaria
    CALL atualiza_coor            ;chama a rotina que atualiza as coordenadas do submarino apos alteracao
```



```
;* que permite desenhar o submarino, de acordo com o movimento

mexed:
    MOV R2,25
    CMP R2,R6
    JZ sai_escolhe_funcao

    CALL apaga_submarino           ;chama a rotina do apaga submarino
    ADD R6,1                      ;movimentacao necessaria
    CALL atualiza_coor             ;chama a rotina que atualiza as coordenadas do submarino apos alteracao
    CALL desenha_submarino        ;chama a rotina do desenhar submarino novamente
    JMP sai_escolhe_funcao

;* -- Mexe para a Esquerda -----
;*
;* Descricao: Rotina que calcula as novas posicoes centro x,y do submarino e chama as
;* rotinas necessarias para a movimentacao da nave pretendida
;*
;* Parametros: R6(posicao centro y do submarino)
;* Retorna: --
;* Destroi: --
;* Notas: Esta rotina utiliza duas rotinas, uma delas para apagar o submarino na sua posicao
;* atual, calcula a nova posicao do centro x,y consoante o movimento e por fim chama a rotina
;* que permite desenhar o submarino, de acordo com o movimento

mexee:
    MOV R2,0
    CMP R2,R6
    JZ sai_escolhe_funcao

    CALL apaga_submarino           ;chama a rotina do apaga submarino
    SUB R6,1                      ;movimentacao necessaria
    CALL atualiza_coor             ;chama a rotina que atualiza as coordenadas do submarino apos alteracao
    CALL desenha_submarino        ;chama a rotina do desenhar submarino novamente
    JMP sai_escolhe_funcao

;* -- Mexe para Direita e para Cima -----
;*
;* Descricao: Rotina que calcula as novas posicoes centro x,y do submarino e chama as
;* rotinas necessarias para a movimentacao da nave pretendida
;*
;* Parametros: R5,R6(posicao centro x do submarino,posicao centro y do submarino)
;* Retorna: --
;* Destroi: --
;* Notas: Esta rotina utiliza duas rotinas, uma delas para apagar o submarino na sua posicao
;* atual, calcula a nova posicao do centro x,y consoante o movimento e por fim chama a rotina
;* que permite desenhar o submarino, de acordo com o movimento

mexedc:
    MOV R2,16
    CMP R2,R5
    JZ sai_escolhe_funcao
    MOV R2,25
    CMP R2,R6
    JZ sai_escolhe_funcao

    CALL apaga_submarino           ;chama a rotina do apaga submarino
    SUB R5,1                      ;movimentacao necessaria
    ADD R6,1                      ;movimentacao necessaria
    CALL atualiza_coor             ;chama a rotina que atualiza as coordenadas do submarino apos alteracao
    CALL desenha_submarino        ;chama a rotina do desenhar submarino novamente
    JMP sai_escolhe_funcao

;* -- Mexe para Direita e para Baixo -----
;*
;* Descricao: Rotina que calcula as novas posicoes centro x,y do submarino e chama as
;* rotinas necessarias para a movimentacao da nave pretendida
```



```
;* -- Mexe para Direita e para Baixo -----
;*
;* Descricao: Rotina que calcula as novas posicoes centro x,y do submarino e chama as
;* rotinas necessarias para a movimentacao da nave pretendida
;*
;* Parametros: R5,R6(posicao centro x do submarino,posicao centro y do submarino)
;* Retorna: --
;* Destroi: --
;* Notas: Esta rotina utiliza duas rotinas, uma delas para apagar o submarino na sua posicao
;* atual, calcula a nova posicao do centro x,y consoante o movimento e por fim chama a rotina
;* que permite desenhar o submarino, de acordo com o movimento

mexedb:
    MOV R2,29
    CMP R2,R5
    JZ sai_escolhe_funcao
    MOV R2,25
    CMP R2,R6
    JZ sai_escolhe_funcao

    CALL apaga_submarino           ;chama a rotina do apaga submarino
    ADD R5,1                      ;movimentacao necessaria
    ADD R6,1                      ;movimentacao necessaria
    CALL atualiza_coor             ;chama a rotina que atualiza as coordenadas do submarino apos alteracao
    CALL desenha_submarino        ;chama a rotina do desenhar submarino novamente
    JMP sai_escolhe_funcao

;* -- Mexe para Baixo e para Esquerda -----
;*
;* Descricao: Rotina que calcula as novas posicoes centro x,y do submarino e chama as
;* rotinas necessarias para a movimentacao da nave pretendida
;*
;* Parametros: R5,R6(posicao centro x do submarino,posicao centro y do submarino)
;* Retorna: --
;* Destroi: --
;* Notas: Esta rotina utiliza duas rotinas, uma delas para apagar o submarino na sua posicao
;* atual, calcula a nova posicao do centro x,y consoante o movimento e por fim chama a rotina
;* que permite desenhar o submarino, de acordo com o movimento

mexeeb:
    MOV R2,29
    CMP R2,R5
    JZ sai_escolhe_funcao
    MOV R2,0
    CMP R2,R6
    JZ sai_escolhe_funcao

    CALL apaga_submarino           ;chama a rotina do apaga submarino
    ADD R5,1                      ;movimentacao necessaria
    SUB R6,1                      ;movimentacao necessaria
    CALL atualiza_coor             ;chama a rotina que atualiza as coordenadas do submarino apos alteracao
    CALL desenha_submarino        ;chama a rotina do desenhar submarino novamente
    JMP sai_escolhe_funcao

;* -- Mexe para Cima e para a Esquerda -----
;*
;* Descricao: Rotina que calcula as novas posicoes centro x,y do submarino e chama as
;* rotinas necessarias para a movimentacao da nave pretendida
;*
;* Parametros: R5,R6(posicao centro x do submarino,posicao centro y do submarino)
;* Retorna: --
;* Destroi: --
;* Notas: Esta rotina utiliza duas rotinas, uma delas para apagar o submarino na sua posicao
;* atual, calcula a nova posicao do centro x,y consoante o movimento e por fim chama a rotina
;* que permite desenhar o submarino, de acordo com o movimento

mexeec:
    MOV R2,16
    CMP R2,R5
    JZ sai_escolhe_funcao
```



```
mexeec:
    MOV R2,16
    CMP R2,R5
    JZ sai_escolhe_funcao
    CMP R2,R6
    JZ sai_escolhe_funcao

    CALL apaga_submarino           ;chama a rotina do apaga submarino
    SUB R5,1                      ;movimentacao necessaria
    SUB R6,1                      ;movimentacao necessaria
    CALL atualiza_coor             ;chama a rotina que atualiza as coordenadas do submarino apos alteracao
    CALL desenha_submarino        ;chama a rotina do desenhar submarino novamente
    JMP sai_escolhe_funcao

; * -- Atualiza Coordenadas Submarino -----
; *
; * Descricao: Rotina que recebe como argumentos R5,R6 e atualiza na memoria as coordenadas do ponto de
; * referencia do submarino.
; *
; * Parametros: R5,R6
; * Retorna: --
; * Destroi: --
; * Notas: --
atualiza_coor:
    MOV [R3],R5
    MOV [R3+2],R6
    RET

; * -- Calcula Coordenada Torpedo -----
; *
; * Descricao: Rotina que utiliza as coordenadas do submarino e atraves delas calcula as coordenadas
; * de saida do torpedo(de maneira a que ele saia do meio do submarino).
; *
; * Parametros: --
; * Retorna: --
; * Destroi: R1,R2,R3
; * Notas: --
calcula_coor_torpedo:
    PUSH R1
    PUSH R2
    PUSH R3
    MOV R1,coor_mem_sub           ;endereço de memoria que contem as coordenadas do submarino
    MOV R3,coor_torpedo           ;endereço de memoria que contem as coordenadas do torpedo
    MOV R2,[R1]                   ;coordenada x do submarino
    SUB R2,3                       ;calcula necessario
    MOV [R3],R2                   ;atualiza a posicao x do torpedo
    ADD R1,2                       ;incrementa endereço
    MOV R2,[R1]                   ;coordenada y do submarino
    ADD R2,4                       ;calcula necessario
    MOV [R3+2],R2                 ;atualiza posicao y do torpedo
    POP R3
    POP R2
    POP R1
    RET

; * -- Desenha Torpedo -----
; *
; * Descricao: Rotina que recebe as coordenadas do ponto de referencia do torpedo
; * e passa as coordenadas e o primeiro endereço da string do torpedo para a rotina desenha geral
; *
; * Parametros: --
; * Retorna: --
; * Destroi: R1,R7,R8
; * Notas: --
desenha_torpedo:
    PUSH R1
    PUSH R7
    PUSH R8
    MOV R1, coor_torpedo
    MOV R7, [R1]                  ;coordenada x
```



```
;* -- Desenha Torpedo -----  
;*  
;* Descricao: Rotina que recebe as coordenadas do ponto de referencia do torpedo  
;* e passa as coordenadas e o primeiro endereço da string do torpedo para a rotina desenha geral  
;*  
;* Parametros: --  
;* Retorna: --  
;* Destroi: R1,R7,R8  
;* Notas: --  
desenha_torpedo:  
    PUSH    R1  
    PUSH    R7  
    PUSH    R8  
    MOV     R1, coor_torpedo  
    MOV     R7, [R1]          ;coordenada x  
    ADD     R1, 2  
    MOV     R8, [R1]          ;coordenada y  
    MOV     R1, torpedo       ;contem o endereco que contem o numero de linhas do torpedo  
    CALL    desenha           ;chama a rotina desenha geral  
    POP     R8  
    POP     R7  
    POP     R1  
    RET  
;* -- Apaga Torpedo -----  
;*  
;* Descricao: Rotina que recebe as coordenadas do ponto de referencia do torpedo  
;* e passa as coordenadas e o primeiro endereço da string do submarino para o torpedo desenha geral  
;*  
;* Parametros: --  
;* Retorna: --  
;* Destroi: R1,R7,R8  
;* Notas: --  
apaga_torpedo:  
    PUSH    R1  
    PUSH    R7  
    PUSH    R8  
    MOV     R1, coor_torpedo  
    MOV     R7, [R1]          ;coordenada x  
    ADD     R1, 2  
    MOV     R8, [R1]          ;coordenada y  
    MOV     R1, torpedo       ;contem o endereco que contem o numero de linhas do torpedo  
    CALL    apaga             ;chama a rotina desenha geral  
    POP     R8  
    POP     R7  
    POP     R1  
    RET  
;* -- Mexe Torpedo -----  
;*  
;* Descricao: Rotina que utiliza as coordenadas atuais do torpedo e realiza o movimento  
;* necessario, atualizando por fim as coordenadas na memoria  
;*  
;* Parametros: --  
;* Retorna: --  
;* Destroi: R1,R7,R8  
;* Notas: Esta rotina utiliza duas rotinas para realizar o seu proposito, começa por apagar  
;* o torpedo, faz o movimento em termos de contas, e por fim atualizamos na memoria e desenhamos.  
mexe_torpedo:  
    PUSH    R1  
    PUSH    R2  
    PUSH    R3  
    CALL    apaga_torpedo     ;chama a rotina apaga torpedo  
    MOV     R1, coor_torpedo  
    MOV     R2, [R1]  
    SUB     R2, 1  
    MOV     [R1], R2          ;atualiza a nova coordenada x do torpedo  
    CALL    desenha_torpedo   ;chama a rotina desenha torpedo  
    POP     R3  
    POP     R2
```




```
;* -- Mexe Torpedo -----  
;*  
;* Descricao: Rotina que utiliza as coordenadas atuais do torpedo e realiza o movimento  
;* necessario, atualizando por fim as coordenadas na memoria  
;*  
;* Parametros:  
;* Retorna: --  
;* Destroi: R1,R7,R8  
;* Notas: Esta rotina utiliza duas rotinas para realizar o seu proposito, começa por apagar  
;* o torpedo, faz o movimento em termos de contas,e por fim atualizamos na memoria e desenhamos.  
mexe_torpedo:  
    PUSH    R1  
    PUSH    R2  
    PUSH    R3  
    CALL    apaga_torpedo        ;chama a rotina apaga torpedo  
    MOV     R1,coor_torpedo  
    MOV     R2,[R1]  
    SUB     R2,1  
    MOV     [R1],R2            ;atualiza a nova coordenada x do torpedo  
    CALL    desenha_torpedo      ;chama a rotina desenha torpedo  
    POP     R3  
    POP     R2  
    POP     R1  
    RET  
  
;* -- Verifica Lanca Torpedo -----  
;*  
;* Descricao: Rotina que recebe como argumento a tecla premida e verifica se esta coincide com  
;* com a tecla designada para disparar.  
;*  
;* Parametros: R0(tecla premida)  
;* Retorna: --  
;* Destroi: R1,R6  
;* Notas: --  
verifica_lanca_torpedo:  
    PUSH    R1  
    PUSH    R6  
    MOV     R1,06H  
    CMP     R0,R1  
    JZ      lanca                ;caso a tecla coincida com a tecla designada,lanca o torpedo  
    JMP     sai_verifica_torpedo ;senao simplesmente sai da rotina  
lanca:  
    CALL    calcula_coor_torpedo ;chama a rotina para saber quais sao as coordenadas com o torpedo sera lancado  
    CALL    desenha_torpedo      ;chama a rotina para desenhar o torpedo  
    MOV     R6,ON                ;ativa-se o estado do torpedo  
    MOV     [R6],R6              ;atualiza-se o estado do torpedo  
    JMP     sai_verifica_torpedo  
  
sai_verifica_torpedo:  
    POP     R6  
    POP     R1  
    RET  
  
;* -- Mexebarcol -----  
;*  
;* Descricao: Rotina que realiza o movimento do barcos de acordo com o estado do  
;* clock1.  
;*  
;* Parametros:  
;* Retorna: --  
;* Destroi: R1,R2,R5,R6,R7,R8,R9  
;* Notas: --  
mexebarcol:  
    PUSH    R1  
    PUSH    R2  
    PUSH    R5  
    PUSH    R6  
    PUSH    R7  
    PUSH    R8
```



```
;* -- Mexebarcol -----
;*
;* Descricao: Rotina que realiza o movimento do barcos de acordo com o estado do
;* clock1.
;*
;* Parametros:
;* Retorna: --
;* Destroi: R1,R2,R5,R6,R7,R8,R9
;* Notas: --
mexebarcol:
    PUSH R1
    PUSH R2
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8
    PUSH R9
    CALL apaga_barcol ;chama a rotina apaga barcol

    MOV R8,1
    MOV R1,coor_mem_b1 ;endereço do valor da coordenada x do barcol
    MOV R2,[R1+2] ;coordenada y do barcol
    MOV R9,estado_barcol ;R9 fica com o endereço do estado do barcol

    CALL mexe_barcol ;chama a rotina mexe barcol
    MOV R5,estado_barcol
    MOV R6,[R5]
    MOV R7,OFF
    CMP R6,R7 ;compara o estado do barcol com OFF
    JZ naoprintabl ;caso o estado do barcol se encontre desligado, nao desenha o barcol, o que significa que foi
    ; atingido e so reaparecera quando for suposto aparecer no lado esquerdo do ecrã(fantasma).

    CALL desenha_barcol
naoprintabl:
    POP R9
    POP R8
    POP R7
    POP R6
    POP R5
    POP R2
    POP R1

    RET

;* -- Mexebarco2 -----
;*
;* Descricao: Rotina que realiza o movimento do barco2 de acordo com o estado do
;* clock1.
;*
;* Parametros:
;* Retorna: --
;* Destroi: R1,R2,R5,R6,R7,R8,R9
;* Notas: --
mexebarco2:
    PUSH R1
    PUSH R2
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8
    PUSH R9
    CALL apaga_barco2 ;chama a rotina apaga barco2

    MOV R8,1
    MOV R1,coor_mem_b2 ;endereço do valor da coordenada x do barco2
    MOV R2,[R1+2] ;coordenada y do barco2
    MOV R9,estado_barco2 ;R9 fica com o endereço do estado do barco2
```



```

; * Notas: --
mexebarco2:

    PUSH R1
    PUSH R2
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8
    PUSH R9
    CALL apaga_barco2                ;chama a rotina apaga barco2

    MOV R8,1
    MOV R1,coord_mem_b2            ;endereço do valor da coordenada x do barco2
    MOV R2,[R1+2]                  ;coordenada y do barco2
    MOV R9,estado_barco2           ;R9 fica com o endereço do estado do barco2

    CALL mexe_barco2                ;chama a rotina mexe barcos
    MOV R5,estado_barco2
    MOV R6,[R5]
    MOV R7,OFF
    CMP R6,R7                      ;compara o estado do barco2 com OFF
    JZ naoprintab2                 ;caso o estado do barco2 se encontre desligado, não desenha o barco2, o que significa que foi
                                   ;atingido e só reapareceria quando for suposto aparecer no lado esquerdo do ecrã(fantasma).

    CALL desenha_barco2
naoprintab2:

    POP R9
    POP R8
    POP R7
    POP R6
    POP R5
    POP R2
    POP R1

    RET

; * -- Mexebarcos -----
; *
; * Descrição: Rotina que recebe argumentos de um dos barcos, incrementa os valores das
; * colunas e caso se encontrem já encostados ao lado direito,atraves do endereço que contem
; * o numero random, geram uma linha random onde irão reaparecer.
; *
; * Parametros:
; * Retorna: --
; * Destroi: R0,R3,R5,R6,R7
; * Notas: --
mexe_barco2:
    PUSH R0
    PUSH R3
    PUSH R5
    PUSH R6
    PUSH R7
    MOV R0,25                      ;valor de coluna maximo que os barcos poderao ter
    ADD R2,2                      ;incrementa a posicao em termos de colunas
    MOV [R1+2],R2                 ;atualiza na memoria o valor y dos barcos
    CMP R2,R0                    ;se o valor que foi calculado for identico ao valor maximo
    JNZ sai_mexe_barco2          ;caso não seja igual simplesmente sai da rotina
    MOV R2,ON                    ;ativa o estado do barco em questao
    MOV [R9],R2

    ; *Gera a nova coordenada x do barco que ira reaparecer do lado esquerdo superior do programa
    MOV R5,gera_random
    MOV R6,[R5]
    MOV R7,mascara_random
    AND R6,R7
    MOV [R1],R6
    MOV [R1+2],R8                ;reset na coordenada y dos barcos
sai_mexe_barco2:

```



```
mexe_barcos:
    PUSH R0
    PUSH R3
    PUSH R5
    PUSH R6
    PUSH R7
    MOV R0,25                ;valor de coluna maximo que os barcos poderao ter
    ADD R2,2                 ;incrementa a posicao em termos de colunas
    MOV [R1+2],R2            ;atualiza na memoria o valor y dos barcos
    CMP R2,R0                ;se o valor que foi calculado for identico ao valor maximo
    JNZ sai_mexe_barcos      ;caso nao seja igual simplesmente sai da rotina
    MOV R2,ON                ;ativa o estado do barco em questao
    MOV [R9],R2

    ;*Gera a nova coordenada x do barco que ira reaparecer do lado esquerdo superior do programa
    MOV R5,gera_random
    MOV R6,[R5]
    MOV R7,mascara_random
    AND R6,R7
    MOV [R1],R6
    MOV [R1+2],R8            ;reset na coordenada y dos barcos
sai_mexe_barcos:
    POP R7
    POP R6
    POP R5
    POP R3
    POP R0
    RET

;* -- Mexebala -----
;*
;* Descricao: Rotina que trata de todas as operacoes da bala, desde de o seu apagar,
;* desenhar e verificar colisoes com o submarino.
;*
;* Parametros: --
;* Retorna: --
;* Destroi: R0,R1,R2,R3,R4
;* Notas: --
mexebala:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    CALL apaga_bala          ;chama a rotina apaga bala
    MOV R1,coor_bala         ;endereco das coordenadas da bala
    MOV R2,[R1]              ;valor da coordenada x da bala
    MOV R3,[R1+2]            ;valor da coordenada y da bala
    MOV R0,30                ;valor maximo que a coordenada y da bala pode alcancar
    MOV R4,1                 ;reset da coordenada y da bala
    CALL mexe_bala           ;chama a rotina mexe bala
    CALL desenha_bala        ;chama a rotina desenha bala
    CALL colisao_bala        ;chama a rotina que verifica se houve colisao da bala com submarino
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET

;* -- Mexe_bala -----
;*
;* Descricao: Rotina que trata de incrementar a coluna e verifica se ja chegou ao fim,
;* calculando a nova linha de reaparecer caso necessario
;*
```



```
;* -- Mexe_bala -----
;*
;* Descricao: Rotina que trata de incrementar a coluna e verifica se ja chegou ao fim,
;* calculando a nova linha de reaparecer caso necessario
;*
;* Parametros: R1,R2,R3,R4
;* Retorna:    --
;* Destroi:    R5,R6,R7,R8
;* Notas:     --
mexe_bala:
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8
    ADD R3,1                ;incrementa valor de y da bala por 1
    MOV[R1+2],R3            ;atualiza a coordenada y na memoria
    CMP R3,R0               ;compara se ja chegou ao valor maximo da coluna
    JZ linha_random_bala    ;se for esse o caso, salta para a linha_random_bala para definir esse novo valor

sai_mexe_bala:
    POP R8
    POP R7
    POP R6
    POP R5
    RET

linha_random_bala:
    MOV R5,gera_random
    MOV R6,[R5]
    MOV R7,mascara_random_bala
    MOV R8,15
    AND R6,R7
    ADD R6,R8
    SUB R6,1                ;para acertar contas
    MOV [R1],R6
    MOV [R1+2],R4
    JMP sai_mexe_bala

;* -- Colisao Bala -----
;*
;* Descricao: Rotina que determina se ha colisao entre a bala e o submarino ,se este
;* caso se apresentar, o estado geral do jogo passa a OFF,interrompendo o ciclo main.
;*
;* Parametros:
;* Retorna:    --
;* Destroi:    R1,R2,R3,R4,R5,R6,R7,R8,R9,R10
;* Notas:     --
colisao_bala:
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8
    PUSH R9
    PUSH R10
    MOV R1,coor_bala
    MOV R2,[R1]              ;valor da coordenada x da bala
    MOV R3,[R1+2]            ;valor da coordenada y da bala
    MOV R4,coor_mem_sub
    MOV R5,[R4]              ;valor da coordenada x do submarino
    MOV R6,[R4+2]            ;valor da coordenada y do submarino
    MOV R9,submarino
    MOVB R10,[R9]            ;numero de linhas que o submarino ocupa

    ;*Verifico se as coordenadas da bala se encontram fora da "caixa"
    ;*do submarino , caso isso se de simplesmente sai da rotina,caso
```



```
;* -- Colisao Bala -----
;*
;* Descricao: Rotina que determina se ha colisao entre a bala e o submarino ,se este
;* caso se apresentar, o estado geral do jogo passa a OFF,interrompendo o ciclo main.
;*
;* Parametros:
;* Retorna:  --
;* Destroi:   R1,R2,R3,R4,R5,R6,R7,R8,R9,R10
;* Notas:  --
colisao_bala:
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8
    PUSH R9
    PUSH R10
    MOV R1,coor_bala
    MOV R2,[R1]                ;valor da coordenada x da bala
    MOV R3,[R1+2]              ;valor da coordenada y da bala
    MOV R4,coor_mem_sub
    MOV R5,[R4]                ;valor da coordenada x do submarino
    MOV R6,[R4+2]              ;valor da coordenada y do submarino
    MOV R9,submarino
    MOVB R10,[R9]              ;numero de linhas que o submarino ocupa

    ;*Verifico se as coordenadas da bala se encontram fora da "caixa"
    ;*do submarino , caso isso se de simplesmente sai da rotina,caso
    ;*contrario o estado geral é colocado a OFF.
    CMP R2,R5
    JLT sai_colisao_bala
    CMP R3,R6
    JLT sai_colisao_bala
    ADD R5,R10
    CMP R2,R5
    JGT sai_colisao_bala
    ADD R9,1
    MOVB R10,[R9]
    ADD R6,R10
    CMP R3,R6
    JGT sai_colisao_bala

    ;*HA colisao
    MOV R7,estado_geral
    MOV R8,OFF
    MOV [R7],R8

sai_colisao_bala:
    POP R10
    POP R9
    POP R8
    POP R7
    POP R6
    POP R5
    POP R4
    POP R3
    POP R2
    POP R1
    RET

;* -- Colisao Barcos1 -----
;*
;* Descricao: Rotina que determina se ha colisao ou nao entre barcol e o torpedo disparado,
;* chamando a rotina geral para as colisoes entre barcos e o torpedo
;*
;* Parametros:
;* Retorna:  --
```



```
;* -- Colisao Barcos1 -----  
;*  
;* Descricao: Rotina que determina se ha colisao ou nao entre barcol e o torpedo disparado,  
;* chamando a rotina geral para as colisoes entre barcos e o torpedo  
;*  
;* Parametros:  
;* Retorna: --  
;* Destroi: R0,R1,R2,R3,R4,R5,R6,R9  
;* Notas: --  
colisao_barcol:  
    PUSH R0  
    PUSH R1  
    PUSH R2  
    PUSH R3  
    PUSH R4  
    PUSH R5  
    PUSH R6  
    PUSH R9  
    MOV R0,ON  
    MOV R6,ON  
    MOV R1,coor_mem_b1  
    MOV R2,[R1] ;valor da coordenada x do barcol  
    MOV R3,[R1+2] ;valor da coordenada y do barcol  
    MOV R1,barcol  
    MOVB R4,[R1] ;numero de linhas do barcol  
    ADD R1,1  
    MOVB R5,[R1] ;numero de colunas do barcol  
    MOV R1, estado_barcol  
    MOV R9,[R1] ;estado do barcol  
  
    CALL colisao_barcos ;chama a rotina de colisao barcos  
    CMP R0,R6 ;compara o registo que deteta se ha colisao com ON  
    JZ sai_colisao_b1 ;caso nao haja colisao simplesmente sai da rotina  
    CALL apaga_barcol ;caso contrario chama a rotina que apaga o barcol e coloca o seu estado a OFF  
    MOV R0,estado_barcol  
    MOV R9,OFF  
    MOV [R0],R9  
  
sai_colisao_b1:  
    POP R9  
    POP R6  
    POP R5  
    POP R4  
    POP R3  
    POP R2  
    POP R1  
    POP R0  
    RET  
  
;* -- Colisao Barcos2 -----  
;*  
;* Descricao: Rotina que determina se ha colisao ou nao entre barcol2 e o torpedo disparado,  
;* chamando a rotina geral para as colisoes entre barcos e o torpedo  
;*  
;* Parametros:  
;* Retorna: --  
;* Destroi: R0,R1,R2,R3,R4,R5,R6,R9  
;* Notas: --  
colisao_barco2:  
    PUSH R0  
    PUSH R1  
    PUSH R2  
    PUSH R3  
    PUSH R4  
    PUSH R5  
    PUSH R6  
    PUSH R9  
    MOV R0,ON  
    MOV R6,ON  
    MOV R1,coor_mem_b2
```



```
;* -- Colisao Barcos2 -----
;*
;* Descricao: Rotina que determina se ha colisao ou nao entre barco2 e o torpedo disparado,
;* chamando a rotina geral para as colisoes entre barcos e o torpedo
;*
;* Parametros:
;* Retorna: --
;* Destroi: R0,R1,R2,R3,R4,R5,R6,R9
;* Notas: --
colisao_barco2:
    PUSH R0
    PUSH R1
    PUSH R2
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R9
    MOV R0,ON
    MOV R6,ON
    MOV R1,coor_mem_b2
    MOV R2,[R1] ;valor da coordenada x do barco2
    MOV R3,[R1+2] ;valor da coordenada y do barco2
    MOV R1,barco2
    MOVB R4,[R1] ;numero de linhas do barco2
    ADD R1,1
    MOVB R5,[R1] ;numero de colunas do barco2
    MOV R1,estado_barco2
    MOV R9,[R1] ;estado do barcol

    CALL colisao_barcos ;chama a rotina de colisao barcos
    CMP R0,R6 ;compara o registo que deteta se ha colisao com ON
    JZ sai_colisao_b2 ;caso nao haja colisao simplesmente sai da rotina
    CALL apaga_barco2 ;caso contrario chama a rotina que apaga o barcol e coloca o seu estado a OFF
    MOV R0,estado_barco2
    MOV R9,OFF
    MOV [R0],R9

sai_colisao_b2:
    POP R9
    POP R6
    POP R5
    POP R4
    POP R3
    POP R2
    POP R1
    POP R0
    RET

;* -- Colisao Barcos -----
;*
;* Descricao: Rotina que determina se ha colisao entre os barcos e o torpedo,
;*
;* Parametros: R1,R2,R3,R4,R5,R6
;* Retorna: --
;* Destroi: R0,R7,R8
;* Notas: --
colisao_barcos:
    PUSH R0
    PUSH R7
    PUSH R8
    MOV R0,OFF
    MOV R1,coor_torpedo
    MOV R7,[R1] ;valor da coordenada x do torpedo
    MOV R8,[R1+2] ;valor da coordenada y do torpedo
    CMP R9,R0 ;compara o estado dos barcos com o OFF
```




```
;* -- Colisao Barcos -----
;*
;* Descricao: Rotina que determina se ha colisao entre os barcos e o torpedo,
;*
;* Parametros: R1,R2,R3,R4,R5,R6
;* Retorna: --
;* Destroi: R0,R7,R8
;* Notas: --
colisao_barcos:
    PUSH R0
    PUSH R7
    PUSH R8
    MOV R0,OFF
    MOV R1,coord_torpedo
    MOV R7,[R1] ;valor da coordenada x do torpedo
    MOV R8,[R1+2] ;valor da coordenada y do torpedo
    CMP R9,R0 ;compara o estado dos barcos com o OFF
    JZ sai_colisao_barcos ;caso o barcos esteja invisivel, devera sair imediatamente da rotina

    ;*Verifico se as coordenadas do torpedo se encontram fora da "caixa"
    ;*dos barcos , caso isso se de simplesmente sai da rotina,caso
    ;*contrario o estado do barco selecionado e colocado a OFF.
    CMP R7,R2
    JLT sai_colisao_barcos
    CMP R8,R3
    JLT sai_colisao_barcos
    ADD R2,R4
    CMP R7,R2
    JGT sai_colisao_barcos
    ADD R3,R5
    CMP R8,R3
    JGT sai_colisao_barcos

    ;* Ha colisao
    CALL pontuacao ;chama a rotina da pontuacao
    MOV R6,OFF ;registo que servira para desligar o barco selecionado

sai_colisao_barcos:
    POP R8
    POP R7
    POP R0
    RET

;* -- Pontuacao -----
;*
;* Descricao: Rotina que trata de atualizar o display e de desligar o estado do torpedo,
;* como tambem apaga-lo.
;*
;* Parametros: --
;* Retorna: --
;* Destroi: R1,R3,R4,R5,R6,R7,R8
;* Notas: --
pontuacao:

    PUSH R1
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8

    MOV R1,displays
    MOV R3,hitpoints
    MOV R4,[R3] ;valor atual do display de pontos
    ;* Tradução dos pontos para decimal
    MOV R7,0AH
    MOV R8,mascara_displays
```



```
CMP R7,R2
JLT sai_colisao_barcos
CMP R8,R3
JLT sai_colisao_barcos
ADD R2,R4
CMP R7,R2
JGT sai_colisao_barcos
ADD R3,R5
CMP R8,R3
JGT sai_colisao_barcos

    ;* Ha colisao
CALL pontuacao                ;chama a rotina da pontuacao
MOV R6 ,OFF                  ;registo que servira para desligar o barco selecionado

sai_colisao_barcos:
POP R8
POP R7
POP R0
RET

;* -- Pontuacao -----
;*
;* Descricao: Rotina que trata de atualizar o display e de desligar o estado do torpedo,
;* como tambem apaga-lo.
;*
;* Parametros: --
;* Retorna:    --
;* Destroi:    R1,R3,R4,R5,R6,R7,R8
;* Notas:     --
pontuacao:

    PUSH R1
    PUSH R3
    PUSH R4
    PUSH R5
    PUSH R6
    PUSH R7
    PUSH R8

    MOV R1,displays
    MOV R3,hitpoints
    MOV R4,[R3]                ;valor atual do display de pontos
    ;* Tradução dos pontos para decimal
    MOV R7,0AH
    MOV R8,mascara_displays
    ADD R4,1                    ;incrementa os pontos por 1
    AND R8,R4
    CMP R8,R7
    JNZ conversao_decimal
    ADD R4,6
conversao_decimal:

    MOV [R3],R4
    MOV [R1],R4
    MOV R5,estado_torpedo
    MOV R6,OFF
    MOV [R5],R6                ;coloca o estado do torpedo a OFF
    CALL apaga_torpedo         ;chama a rotina apaga torpedo

sai_pontuacao:
POP R8
POP R7
POP R6
POP R5
POP R4
POP R3
POP R1
RET
```