;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Projeto 1 de Arquitetura de Computadores, LETI \*

;\* 93577 - Fabio Sousa \*

;\* 93587 - Joao Pereira \*

;\* 93608 - Pedro Godinho \*

;\* \*

;\* Controlos: \*

;\* 0, 1, 2, 4, 6, 8, 9, A - Movimento do Submarino \*

;\* E - Stop \*

;\* F - Restart \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DISPLAY EQU 8000H ; Endereco do display

KEYPININ EQU 0E000H ; Endereco de onde ler do keyboard

KEYPINOUT EQU 0C000H ; Endereco de onde escrever para o keyboard

HEX\_DISPLAY1 EQU 0A000H ; Display hexadecimal 1

HEX\_DISPLAY2 EQU 06000H ; Display hexadecimal 2

PLACE 2000H

stack: TABLE 100H ; Stack vai ter 100H words

SP\_start:

; Tabela do movimento ao qual cada tecla corresponde (2, 2 indica nao mover)

movement: STRING -1, -1 ; 0

STRING 0, -1 ; 1

STRING 1, -1 ; 2

STRING 2, 2 ; 3

STRING -1, 0 ; 4

STRING 2, 2 ; 5

STRING 1, 0 ; 6

STRING 2, 2 ; 7

STRING -1, 1 ; 8

STRING 0, 1 ; 9

STRING 1, 1 ; a

; uma tabela dos numeros binarios 1000 0000 a 0000 0001 (one-hot)

onehot\_table: WORD 80H

WORD 40H

WORD 20H

WORD 10H

WORD 8H

WORD 4H

WORD 2H

WORD 1H

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Cada string que se pode desenhar no ecra comeca com o x e o y onde devem \*

;\* ser desenhadas, seguidos do seu tamanho em x e em y (para a rotina saber \*

;\* onde para a leitura), seguido dos 0s e 1s a desenhar em cada posicao. \*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Ecra "Press any key" a mostrar no inicio do jogo

start\_screen: STRING 0H, 0H ; coordenadas

STRING 20H, 20H ; tamanho

STRING 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

STRING 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

; Ecra "Game over" a mostrar no fim do jogo

end\_screen: STRING 0H, 0H

STRING 20H, 20H

STRING 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

STRING 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

; Desenho do submarino

submarine: STRING 0CH, 12H

STRING 6, 3

STRING 2, 2, 1, 1, 2, 2

STRING 2, 2, 2, 1, 2, 2

STRING 1, 1, 1, 1, 1, 1

; Retangulo de 0s do tamanho do submarino, mantido sincroniado com o submarino para o apagar sempre que necessario

erase\_submarine: STRING 0CH, 12H

STRING 6, 3

STRING 2, 2, 0, 0, 2, 2

STRING 2, 2, 2, 0, 2, 2

STRING 0, 0, 0, 0, 0, 0

; Desenho do barco grande

boat1: STRING 0H, 0H

STRING 8H, 6H

STRING 2, 1, 2, 2, 2, 2, 2, 2

STRING 2, 2, 1, 2, 2, 2, 2, 2

STRING 2, 2, 1, 2, 2, 2, 2, 2

STRING 1, 1, 1, 1, 1, 1, 1, 1

STRING 2, 1, 1, 1, 1, 1, 1, 2

STRING 2, 2, 1, 1, 1, 1, 2, 2

; Desenho dos pixeis a apagar do barco grande (alguns nao tem de ser apagados)

erase\_boat1: STRING 0H, 0H

STRING 8H, 6H

STRING 2, 0, 2, 2, 2, 2, 2, 2

STRING 2, 2, 0, 2, 2, 2, 2, 2

STRING 2, 2, 0, 2, 2, 2, 2, 2

STRING 0, 2, 2, 2, 2, 2, 2, 2

STRING 2, 0, 2, 2, 2, 2, 2, 2

STRING 2, 2, 0, 2, 2, 2, 2, 2

; Desenho dos pixeis a apagar para limpar completamente o barco grande

fully\_erase\_boat1: STRING 0H, 0H

STRING 8H, 6H

STRING 2, 0, 2, 2, 2, 2, 2, 2

STRING 2, 2, 0, 2, 2, 2, 2, 2

STRING 2, 2, 0, 2, 2, 2, 2, 2

STRING 0, 0, 0, 0, 0, 0, 0, 0

STRING 2, 0, 0, 0, 0, 0, 0, 2

STRING 2, 2, 0, 0, 0, 0, 2, 2

; Desenho do barco pequeno

boat2: STRING 0H, 0H

STRING 6H, 5H

STRING 2, 1, 2, 2, 2, 2

STRING 2, 2, 1, 2, 2, 2

STRING 2, 2, 1, 2, 2, 2

STRING 1, 1, 1, 1, 1, 1

STRING 2, 1, 1, 1, 1, 2

; Desenho dos pixeis a apagar do barco pequeno (alguns nao tem de ser apagados)

erase\_boat2: STRING 0H, 0H

STRING 6H, 5H

STRING 2, 0, 2, 2, 2, 2

STRING 2, 2, 0, 2, 2, 2

STRING 2, 2, 0, 2, 2, 2

STRING 0, 2, 2, 2, 2, 2

STRING 2, 0, 2, 2, 2, 2

; Desenho dos pixeis a apagar para limpar completamente o barco pequeno

fully\_erase\_boat2: STRING 0H, 0H

STRING 6H, 5H

STRING 2, 0, 2, 2, 2, 2

STRING 2, 2, 0, 2, 2, 2

STRING 2, 2, 0, 2, 2, 2

STRING 0, 0, 0, 0, 0, 0

STRING 2, 0, 0, 0, 0, 2

; Desenho do torpedo

torpedo: STRING 0H, 0H

STRING 1, 3

STRING 1

STRING 1

STRING 1

; Desenho com que apagar o torpedo

erase\_torpedo: STRING 0H, 0H

STRING 1, 3

STRING 0

STRING 0

STRING 0

; Desenho da bala

bullet: STRING 0H, 0H

STRING 1, 1

STRING 1

; Desenho com que apagar a bala

erase\_bullet: STRING 0H, 0H

STRING 1, 1

STRING 0

; Ultima tecla lida

last\_key: word -1

;Tabela de interrupcoes para os relogios

exception\_table: word clock\_0\_exception

word clock\_1\_exception

; Estados

submarine\_state: word 0

torpedo\_state: word 0

bullet\_state: word 0

boat1\_state: word 0

boat2\_state: word 0

torpedo\_clock: word 0

bullet\_clock: word 0

boat1\_clock: word 0

boat2\_clock: word 0

; Pontuacao do jogador

score: word 0

; Comecar o programa

PLACE 0

MOV BTE, exception\_table ; Escrever a tabela de interrupecoes

EI1

EI0

EI

restart:

MOV SP, SP\_start ; (Re)iniciar o stack

; Inicializar todos os estados a 0

MOV R0, submarine\_state

MOV R1, 0

MOV [R0], R1

MOV R0, torpedo\_state

MOV R1, 0

MOV [R0], R1

MOV R0, bullet\_state

MOV R1, 0

MOV [R0], R1

MOV R0, boat1\_state

MOV R1, 0

MOV [R0], R1

MOV R0, boat2\_state

MOV R1, 0

MOV [R0], R1

; Reiniciar a last\_key para -1

MOV R0, last\_key

MOV R1, -1

MOV [R0], R1

; Reenviar o score para 0

MOV R0, score

MOV R1, 0

MOV [R0], R1

MOV R0, HEX\_DISPLAY1

MOVB [R0], R1

; Reiniciar os registos

MOV R0, 0

MOV R1, 0

MOV R2, 0

MOV R3, 0

MOV R4, 0

MOV R5, 0

MOV R6, 0

MOV R7, 0

MOV R8, 0

MOV R10, 0

; Mostrar o ecra inicial

MOV R0, start\_screen

CALL draw\_string

MOV R0, last\_key ; R0 - endereco da last\_key

restart\_wait:

CALL get\_key

MOV R1, [R0] ; ler a ultima tecla carregada

CMP R1, -1

JEQ restart\_wait ; Se nenhuma tecla foi carregada, continuar a esperar

; Comecar o jogo:

call clear\_screen ; Limpar o ecra

main\_loop:

; Chamadas as varias funcoes que gerem as varias partes do jogo.

CALL submarine\_handler

CALL bullet\_handler

CALL torpedo\_handler

CALL boat1\_handler

CALL boat2\_handler

CALL detect\_torpedo\_collision

CALL detect\_bullet\_collision

CMP R0, 1 ; verifica se a last\_key foi E

JEQ stop

CALL get\_key ; gravar tecla lida para last\_key

MOV R1, last\_key

MOV R0, [R1] ; valor de last\_key para R0

CMP R0, -1

JEQ main\_loop ; nao foi dado input

; Submarino

MOV R1, 0AH ; A ultima tecla de movimento e o A

CMP R0, R1

JGT main\_no\_movement ; Se nao for uma tecla de movimento, nao alterar o estado do submarino

MOV R2, submarine\_state

MOV R3, 2

MOV [R2], R3 ; atualiza o estado do submarino para o estado 2

main\_no\_movement:

; Torpedo

MOV R1, 5 ; A tecla para disparar e o 5

CMP R0, R1

JNZ main\_no\_shooting

MOV R2, torpedo\_state ; Alterar o estado do torpedo apenas se este nao e 1

MOV R3, [R2]

CMP R3, 1 ; verifica se o estado do torpedo nao e o estado 1

JNZ main\_no\_shooting

MOV R3, 2

MOV [R2], R3 ; muda o estado do torpedo para o estado 2

main\_no\_shooting:

MOV R1, 0EH

CMP R0, R1

JEQ stop ; o input e a tecla de stop (e)

MOV R1, 0FH

CMP R0, R1

JEQ restart ; o input e a tecla de restart (f)

JMP main\_loop ; proxima iteracao do main\_loop

end: JMP end

clock\_0\_exception:

PUSH R0

PUSH R1

MOV R1, 1

MOV R0, boat1\_clock

MOV [R0], R1 ; muda o boat1\_clock para 1

MOV R0, boat2\_clock

MOV [R0], R1 ; muda o boat2\_clock para 1

POP R1

POP R0

RFE

clock\_1\_exception:

PUSH R0

PUSH R1

MOV R1, 1

MOV R0, torpedo\_clock

MOV [R0], R1 ; muda o torpedo\_clock para 1

MOV R0, bullet\_clock

MOV [R0], R1 ; muda o bullet\_clock para 1

POP R1

POP R0

RFE

stop:

MOV R0, end\_screen

CALL draw\_string

MOV R2, last\_key

stop\_loop:

CALL get\_key ;gravar tecla lida para last\_key

MOV R0, [R2] ; valor de last\_key para R0

MOV R1, 0FH

CMP R0, R1

JEQ restart ; o input e a tecla de restart (f)

JMP stop\_loop

; Rotina que apaga tudo no ecra (da esquerda para a direita, para ser diferente)

clear\_screen:

PUSH R0

PUSH R1

PUSH R2

PUSH R3

MOV R3, 0 ; Queremos sempre desenhar um 0

MOV R0, 20H ; R0: 20H, para comparacoes

MOV R1, 0 ; R1: x

clear\_screen\_for1: ;for (x = 0; x < 20H, x++) {

CMP R1, R0

JGE clear\_screen\_for1\_end

MOV R2, 0 ; R2: y

clear\_screen\_for2: ; for (y = 0; y < 20H, y++) {

CMP R2, R0

JGE clear\_screen\_for2\_end

CALL draw\_pixel ; R1 e R2 ja tem as coordenadas a desenhar

ADD R2, 1 ; fim do loop interno

JMP clear\_screen\_for2

clear\_screen\_for2\_end:

ADD R1, 1 ; fim do loop externo

JMP clear\_screen\_for1

clear\_screen\_for1\_end:

clear\_screen\_end:

POP R3

POP R2

POP R1

POP R0

RET

random\_num\_gen:

PUSH R1

MOV R1, HEX\_DISPLAY2;

MOV R0, [R1]

POP R1

RET

detect\_torpedo\_collision:

PUSH R0

PUSH R1

PUSH R2

PUSH R4

PUSH R4

PUSH R5

PUSH R6

PUSH R7

MOV R0, torpedo\_state

MOV R1, [R0]

CMP R1, 2

JLT detect\_torpedo\_collision\_end

MOV R0, torpedo

MOVB R1, [R0] ; x do torpedo

ADD R0, 1

MOVB R2, [R0] ; y do torpedo

MOV R3, boat1\_state

MOV R4, [R3]

CMP R4, 0

JEQ detect\_torpedo\_collision\_boat2

MOV R0, boat1

MOVB R3, [R0] ; x do boat1

MOV R4, 20H

CMP R3, R4

JLE detect\_torpedo\_collision\_continue1

MOV R4, 0FF00H

OR R3, R4

detect\_torpedo\_collision\_continue1:

ADD R0, 1

MOVB R4, [R0] ; y do boat1

ADD R0, 1

MOVB R5, [R0] ; tamanho\_x do boat1

ADD R0, 1

MOVB R6, [R0] ; tamanho\_y do boat1

CMP R1, R3

JLT detect\_torpedo\_collision\_boat2

CMP R2, R4

JLT detect\_torpedo\_collision\_boat2

MOV R7, R3

ADD R7, R5 ; coordenadas\_x do boat1

CMP R1, R7

JGT detect\_torpedo\_collision\_boat2

MOV R7, R4

ADD R7, R6 ; coordenadas\_y do boat1

CMP R2, R7

JGT detect\_torpedo\_collision\_boat2

MOV R0, boat1\_state

MOV R1, 0

MOV [R0], R1

MOV R0, fully\_erase\_boat1

CALL draw\_string

MOV R0, score

MOV R1, [R0]

ADD R1, 1

MOV [R0], R1

MOV R0, HEX\_DISPLAY1

MOVB [R0], R1

JMP detect\_torpedo\_collision\_end

detect\_torpedo\_collision\_boat2:

MOV R3, boat2\_state

MOV R4, [R3]

CMP R4, 0

JEQ detect\_torpedo\_collision\_end

MOV R0, boat2

MOVB R3, [R0] ; x do boat2

MOV R4, 20H

CMP R3, R4

JLE detect\_torpedo\_collision\_continue2

MOV R4, 0FF00H

OR R3, R4

detect\_torpedo\_collision\_continue2:

ADD R0, 1

MOVB R4, [R0] ; y do boat2

ADD R0, 1

MOVB R5, [R0] ; tamanho\_x do boat2

ADD R0, 1

MOVB R6, [R0] ; tamanho\_y do boat2

CMP R1, R3

JLT detect\_torpedo\_collision\_end

CMP R2, R4

JLT detect\_torpedo\_collision\_end

MOV R7, R3

ADD R7, R5 ; coordenadas\_x do boat2

CMP R1, R7

JGT detect\_torpedo\_collision\_end

MOV R7, R4

ADD R7, R6 ; coordenadas\_y do boat2

CMP R2, R7

JGT detect\_torpedo\_collision\_end

MOV R0, boat2\_state

MOV R1, 0

MOV [R0], R1

MOV R0, fully\_erase\_boat2

CALL draw\_string

MOV R0, score

MOV R1, [R0]

ADD R1, 1

MOV [R0], R1

MOV R0, HEX\_DISPLAY1

MOVB [R0], R1

JMP detect\_torpedo\_collision\_end

detect\_torpedo\_collision\_end:

POP R7

POP R6

POP R5

POP R4

POP R3

POP R2

POP R1

POP R0

RET

detect\_bullet\_collision:

PUSH R1

PUSH R2

PUSH R4

PUSH R4

PUSH R5

PUSH R6

PUSH R7

PUSH R8

PUSH R9

MOV R0, bullet

MOVB R1, [R0] ; x da bullet

ADD R0, 1

MOVB R2, [R0] ; y da bullet

MOV R0, submarine

MOVB R3, [R0] ; x do submarino

ADD R0, 1

MOVB R4, [R0] ; y do submarino

ADD R0, 1

MOVB R5, [R0] ; tamanho\_x do submarino

ADD R0, 1

MOVB R6, [R0] ; tamanho\_y do submarino

ADD R0, 1 ; R0 fica no primeiro elemento do submarino

CMP R1, R3

JLT detect\_bullet\_collision\_x\_end

CMP R2, R4

JLT detect\_bullet\_collision\_x\_end

MOV R7, R3

ADD R7, R5

CMP R1, R7

JGT detect\_bullet\_collision\_x\_end

MOV R7, R4

ADD R7, R6

CMP R2, R7

JGT detect\_bullet\_collision\_x\_end

MOV R7, 0

detect\_bullet\_collision\_y: ; for (y = 0; y < tamanho\_y\_sub; y++)

CMP R7, R6

JGE detect\_bullet\_collision\_y\_end

MOV R8, 0

detect\_bullet\_collision\_x: ; for (x = 0; x < tamanho\_x\_sub; x++)

CMP R8, R5

JGE detect\_bullet\_collision\_x\_end

MOVB R9, [R0]

CMP R9, 1

JNE detect\_bullet\_collision\_no\_collision ; so verificar se o pixel e 1

MOV R9, R8 ; R9 - x na box do submarine

ADD R9, R3 ; R9 - x do pixel no display

CMP R9, R1

JNE detect\_bullet\_collision\_no\_collision

MOV R9, R7 ; R9 - y na box do submarine

ADD R9, R4 ; R9 - y do pixel no display

CMP R9, R2

JNE detect\_bullet\_collision\_no\_collision

; Houve colisao!

MOV R0, 1

JMP detect\_bullet\_collision\_end

detect\_bullet\_collision\_no\_collision:

ADD R0, 1

ADD R8, 1

JMP detect\_bullet\_collision\_x

detect\_bullet\_collision\_x\_end:

ADD R7, 1

JMP detect\_bullet\_collision\_y

detect\_bullet\_collision\_y\_end:

MOV R0, 0

detect\_bullet\_collision\_end:

POP R9

POP R8

POP R7

POP R6

POP R5

POP R4

POP R3

POP R2

POP R1

RET

submarine\_handler:

PUSH R0

PUSH R1

MOV R1, submarine\_state

MOV R0, [R1]

CMP R0, 0

JEQ submarine\_handler\_0

CMP R0, 1

JEQ submarine\_handler\_1

CMP R0, 2

JEQ submarine\_handler\_2

POP R1

POP R0

RET

submarine\_handler\_0: ; Estado 0: inicializar as variaveis

; Reiniciar o sitio do submarino

PUSH R2

PUSH R3

MOV R1, submarine

MOV R2, erase\_submarine

MOV R3, 0CH ; x para 0CH

MOVB [R1], R3

MOVB [R2], R3

MOV R3, 12H ; y para 12H

ADD R1, 1 ; avancar o endereco para y

ADD R2, 1 ; avancar o endereco para y

MOVB [R1], R3

MOVB [R2], R3

MOV R0, submarine

CALL draw\_string

MOV R0, submarine\_state

MOV R1, 1

MOV [R0], R1 ; Passa o estado do submarino para 1

POP R3

POP R2

POP R1

POP R0

RET

submarine\_handler\_1: ; Estado 1: O jogador ao carregou em nenhuma tecla

POP R1

POP R0

RET

submarine\_handler\_2: ; Estado 2: O jogador carregou em alguma tecla

PUSH R2

PUSH R3

PUSH R4

PUSH R5

PUSH R6

PUSH R7

PUSH R8

PUSH R9

MOV R1, submarine ; R1 - endereco do submarino

MOV R2, erase\_submarine ; R2 - endereco do erase\_submarine

MOV R4, last\_key

MOV R3, [R4] ; R3 - last\_key

MOV R4, 2

MUL R3, R4 ; R3 - last\_key \* 2

MOV R4, movement ; R4 - endereco da tabela de movimentos

ADD R4, R3 ; R4 - endereco do movimento a fazer

MOVB R5, [R4] ; R5 - movimento x a fazer

ADD R4, 1 ; Avancar para o y na tabela de movimentos

MOVB R6, [R4] ; R6 - movimento y a fazer

MOV R7, 0

MOV R8, 0

MOVB R7, [R1]

ADD R7, R5 ; R7 - x final

ADD R1, 1

MOVB R8, [R1]

ADD R8, R6 ; R8 - y final

MOV R1, submarine

MOV R3, 00FFH ; Mask para 8 bits

AND R7, R3 ; Aplicar a mask em R7 e R8

AND R8, R3

MOV R3, 1BH ; R3: 1BH (20H - tamanho em x do submarino)

MOV R4, 1EH ; R4: 1EH (20H - tamanho em y do submarino)

MOV R6, 0BH ; R6: Limite de cima do ecra (10H)

CMP R5, 2 ; Consideramos 2 no movimento como indicativo de movimento nulo

JEQ submarine\_handler\_end

; Verificar se o movimento e valido

AND R7, R7 ; Se o x ficaria negativo (fora do ecra a esquerda)

JN submarine\_handler\_end

CMP R7, R3 ; Se o x + tamanho do submarino ficaria acima de 20H (fora do ecra a direita)

JGE submarine\_handler\_end

CMP R8, R6 ; Se o y ficaria acima do limite de cima

JLT submarine\_handler\_end

CMP R8, R4 ; Se o y + tamanho do submarino ficaria acima de 20H (fora do ecra para baixo)

JGE submarine\_handler\_end

; Se o movimento e valido:

MOV R0, erase\_submarine

CALL draw\_string ; apagar o submarino

MOVB [R1], R7 ; Fazer o movimento em x em submarine

MOVB [R2], R7 ; Fazer o movimento em x em erase\_submarine

ADD R1, 1 ; Avancar para y

ADD R2, 1 ; Avancar para y

MOVB [R1], R8 ; Fazer o movimento em y em submarine

MOVB [R2], R8 ; Fazer o movimento em y em erase\_submarine

MOV R0, submarine

CALL draw\_string ; Desenhar de novo o submarino

MOV R0, submarine\_state

MOV R1, 1

MOV [R0], R1

submarine\_handler\_end:

POP R9

POP R8

POP R7

POP R6

POP R5

POP R4

POP R3

POP R2

POP R1

POP R0

RET

torpedo\_handler:

PUSH R0

PUSH R1

MOV R1, torpedo\_state

MOV R0, [R1]

CMP R0, 0

JEQ torpedo\_handler\_0

CMP R0, 1

JEQ torpedo\_handler\_1

CMP R0, 2

JEQ torpedo\_handler\_2

CMP R0, 3

JEQ torpedo\_handler\_3

POP R1

POP R0

RET

torpedo\_handler\_0: ; Estado 0: Inicializar o torpedo, quando o jogo e (re)iniciado

PUSH R2

PUSH R3

MOV R1, torpedo

MOV R2, erase\_torpedo

MOV R3, 0

MOVB [R1], R3 ; coordenada\_x do torpedo = 0

MOVB [R2], R3 ; coordenada\_x do apaga torpedo = 0

MOV R3, 0H

ADD R1, 1

ADD R2, 1

MOVB [R1], R3 ; coordenada\_y do torpedo = 0

MOVB [R2], R3 ; coordenada\_y do apaga torpedo = 0

MOV R1, torpedo\_state

MOV R0, 1

MOV [R1], R0 ; muda o estado do torpedo para 1

POP R3

POP R2

POP R1

POP R0

RET

torpedo\_handler\_1: ; Estado 1: O torpedo nao existe

POP R1

POP R0

RET

torpedo\_handler\_2: ; Estado 2: Quando o torpedo e disparado

PUSH R2

PUSH R3

MOV R0, submarine ; endereco da coordenada x do submarino

MOVB R1, [R0] ; R1: coordenada x do submarino

ADD R0, 1

MOVB R2, [R0] ; R2: coordenada\_y do submarino

ADD R1, 5 ; R1: coordenada\_x no submarino onde o torpedo vai ser disparado

SUB R2, 1 ; R2: coordenada\_y no submarine onde o torpedo vai ser disparado

MOV R0, torpedo ; R0: coordenada\_x do torpedo

MOV R3, erase\_torpedo ; R3: coordenada\_y do torpedo

MOVB [R0], R1 ; muda a coordenada\_x do torpedo para onde vai ser disparado

MOVB [R3], R1 ; muda a coordenada\_x do apaga torpedo para onde vai ser disparado

ADD R0, 1 ; endereco da coordenada\_y do torpedo

ADD R3, 1 ; endereco da coordenada\_y do apaga torpedo

MOVB [R0], R2 ; muda a coordenada\_y do torpedo para onde vai ser disparado

MOVB [R3], R2 ; muda a coordenada\_y do apaga torpedo para onde vai ser disparado

SUB R0, 1 ; R0 : coordenada\_x onde o torpedo vai ser desenhado

CALL draw\_string ; desenha o torpedo

MOV R1, 3

MOV R0, torpedo\_state

MOV [R0], R1 ; muda o estado do torpedo para o estado 3

POP R3

POP R2

POP R1

POP R0

RET

torpedo\_handler\_3: ; Estado 3: O torpedo esta em movimento, tem de subir

PUSH R2

MOV R0, torpedo\_clock

MOV R1, [R0]

CMP R1, 1 ; So se move se o torpedo\_clock estiver a 1

JNE torpedo\_handler\_3\_end

MOV R0, erase\_torpedo

CALL draw\_string ; apaga o torpedo

MOV R0, torpedo

MOV R1, erase\_torpedo

ADD R0, 1 ; R0: coordenada\_y do torpedo

ADD R1, 1 ; R1: coordenada\_y do apaga torpedo

MOVB R2, [R0]

SUB R2, 1 ; diminui a coordenada\_y em 1 unidade (o torpedo sobe)

CMP R2, 0 ; se o torpedo bater no topo do ecra

JLT torpedo\_handler\_3\_destroy\_torpedo

MOVB [R0], R2 ; atualiza as novas coordenadas do torpedo

MOVB [R1], R2 ; atualiza as novas coordenadas do apaga torpedo

SUB R0, 1 ; R0: coordenada\_x do torpedo

CALL draw\_string ; desenha o torpedo com as novas coordenadas

MOV R0, torpedo\_clock

MOV R1, 0

MOV [R0], R1 ; muda o torpedo\_clock para 0

JMP torpedo\_handler\_3\_end

torpedo\_handler\_3\_destroy\_torpedo: ; Destroi o torpedo

MOV R0, torpedo\_state

MOV R1, 1

MOV [R0], R1 ; muda o estado do torpedo para 1

torpedo\_handler\_3\_end:

POP R2

POP R1

POP R0

RET

boat1\_handler:

PUSH R0

PUSH R1

MOV R1, boat1\_state ; vai buscar o estado do barco 1

MOV R0, [R1]

CMP R0, 0

JEQ boat1\_handler\_0 ; se o estado do barco 1 for 0

CMP R0, 1

JEQ boat1\_handler\_1 ; se o estado do barco 1 for 1

POP R1

POP R0

RET

boat1\_handler\_0: ; Estado 0: Colocar o barco a esquerda do ecra

PUSH R2

PUSH R3

PUSH R4

PUSH R5

MOV R1, boat2\_state

MOV R2, [R1] ; obtem o estado do barco 2

CMP R2, 1 ; verifica se o estado do barco 2 nao e o estado 1

JNE boat1\_handler\_0\_continue

MOV R1, boat2 ; R1: endereco da cordenada\_x do barco 2

MOVB R2, [R1] ; R2: coordenada\_x do barco 2

CMP R2, 3 ; verifica se a distancia entre os dois barco e de 3 ou mais pixeis

JLT boat1\_handler\_0\_end

MOV R1, 20H

CMP R2, R1 ; verifica se as coordenada\_x e negativa (fora do ecra para a esquerda)

JGE boat1\_handler\_0\_end

boat1\_handler\_0\_continue:

MOV R1, boat1 ; R1: endereco da coordenada\_x do barco 1

MOV R2, erase\_boat1 ; R2: endereco da coordenada\_x do apaga barco 1

MOV R5, fully\_erase\_boat1 ; R5: endereco da coordenada\_x do apaga tudo barco 1

MOV R3, 0F8H ; 8 pixeis a esquerda do ecra (-8)

MOVB [R1], R3 ; atualiza a coordenada\_x dos 3 enderecos

MOVB [R2], R3

MOVB [R5], R3

CALL random\_num\_gen ; obtem um numero aleatorio

MOV R4, 03H ; 0011 - mask para o numero aleatorio

AND R0, R4

ADD R1, 1 ; R1: endereco da coordenada\_y do barco 1

ADD R2, 1 ; R2: endereco da coordenada\_y do apaga barco 1

ADD R5, 1 ; R5: endereco da coordenada\_y do apaga tudo barco 1

MOVB [R1], R0 ; atualiza a coordenada\_y dos 3 enderecos

MOVB [R2], R0

MOVB [R5], R0

MOV R1, boat1\_state

MOV R0, 1

MOV [R1], R0 ; atualiza o estado do boat 1 para o estado 1

boat1\_handler\_0\_end:

POP R5

POP R4

POP R3

POP R2

POP R1

POP R0

RET

boat1\_handler\_1: ; Estado 1: Mover o barco para a direita em 1 pixel

PUSH R2

PUSH R3

PUSH R4

MOV R0, boat1\_clock

MOV R1, [R0]

CMP R1, 1 ; verifica se o boat1\_clock esta a 1

JNE boat1\_handler\_1\_end

MOV R0, erase\_boat1

CALL draw\_string ; apaga o barco 1

MOV R0, boat1 ; R0: endereco da coordenada\_x do barco 1

MOV R1, erase\_boat1 ; R1: endereco da coordenada\_x do apaga barco 1

MOV R4, fully\_erase\_boat1 ; R4: endereco da coordenada\_x do apaga tudo barco 1

MOVB R2, [R0] ; R2: coordenada\_x do barco 1

ADD R2, 1 ; aumenta a coordenada\_x do barco 1 em uma unidade

MOV R3, 20H

CMP R2, R3 ; verifica se o barco esta dentro do ecra (x =< 32)

JLT boat1\_handler\_1\_continue

MOV R3, 0F0H

CMP R2, R3 ; verifica se barco saiu do ecra (x > 32)

JLT boat1\_handler\_1\_destroy\_boat

boat1\_handler\_1\_continue:

MOVB [R0], R2 ; atualiza a coordenada\_x dos 3 enderecos

MOVB [R1], R2

MOVB [R4], R2

CALL draw\_string ; desenha o barco 1 nas novas coordenadas (x+1,y)

MOV R0, boat1\_clock

MOV R1, 0

MOV [R0], R1 ; muda o boat1\_clock para 0

JMP boat1\_handler\_1\_end

boat1\_handler\_1\_destroy\_boat: ; destroi o barco porque saiu do ecra

MOV R0, boat1\_state

MOV R1, 0

MOV [R0], R1 ; muda o estado do barco 1 para o estado 0

boat1\_handler\_1\_end:

POP R4

POP R3

POP R2

POP R1

POP R0

RET

boat2\_handler:

PUSH R0

PUSH R1

MOV R1, boat2\_state

MOV R0, [R1]

CMP R0, 0 ; verifica se o estado do barco 2 e o estado 0

JEQ boat2\_handler\_0

CMP R0, 1 ; verifica se o estado do barco 2 e o estado 1

JEQ boat2\_handler\_1

POP R1

POP R0

RET

boat2\_handler\_0:

PUSH R2

PUSH R3

PUSH R4

PUSH R5

MOV R1, boat1\_state

MOV R2, [R1] ; obtem o estado do barco 1

CMP R2, 1 ; verifica se o estado do barco 1 nao e o estado 1

JNE boat2\_handler\_0\_continue

MOV R1, boat1 ; R1: endereco da coordenada\_x do barco 1

MOVB R2, [R1] ; R2: coordenada\_x do barco 1

CMP R2, 3 ; verfica se a distancia entre os dois barcos e de 3 ou mais pixeis

JLT boat2\_handler\_0\_end

MOV R1, 20H

CMP R2, R1 ; verifica se as coordenada\_x e negativa (fora do ecra para a esquerda)

JGE boat2\_handler\_0\_end

boat2\_handler\_0\_continue:

MOV R1, boat2 ; R1: endereco da coordenada\_x do barco 2

MOV R2, erase\_boat2 ; R2: endereco da coordenada\_x do apaga barco 2

MOV R5, fully\_erase\_boat2 ; R5: endereco da coordenada\_x do apaga tudo barco 2

MOV R3, 0FAH ; 8 pixeis a esquerda do ecra (-8)

MOVB [R1], R3 ; atualiza a coordenada\_x dos 3 enderecos

MOVB [R2], R3

MOVB [R5], R3

CALL random\_num\_gen ; obtem um numero aleatorio

MOV R4, 04H ; 0011 - mask para o numero aleatorio

AND R0, R4

ADD R1, 1 ; R1: endereco da coordenada\_y do barco 2

ADD R2, 1 ; R2: endereco da coordenada\_y do apaga barco 2

ADD R5, 1 ; R5: endereco da coordenada\_y do apaga tudo barco 2

MOVB [R1], R0 ; atualiza a coordenada\_y dos 3 enderecos

MOVB [R2], R0

MOVB [R5], R0

MOV R1, boat2\_state

MOV R0, 1

MOV [R1], R0 ; atualiza o estado do boat 2 para o estado 1

boat2\_handler\_0\_end:

POP R5

POP R4

POP R3

POP R2

POP R1

POP R0

RET

boat2\_handler\_1: ; Estado 1: Mover o barco para a direita em 1 pixel

PUSH R2

PUSH R3

PUSH R4

MOV R0, boat2\_clock

MOV R1, [R0]

CMP R1, 1 ; verifica se o boat2\_clock esta a 1

JNE boat2\_handler\_1\_end

MOV R0, erase\_boat2

CALL draw\_string ; apaga o barco 2

MOV R0, boat2 ; R0: endereco da coordenada\_x do barco 2

MOV R1, erase\_boat2 ; R1: endereco da coordenada\_x do apaga barco 2

MOV R4, fully\_erase\_boat2 ; R4: endereco da coordenada\_x do apaga tudo barco 2

MOVB R2, [R0] ; R2: coordenada\_x do barco 2

ADD R2, 1 ; aumenta a coordenada\_x do barco 2 em uma unidade

MOV R3, 20H

CMP R2, R3 ; verifica se o barco esta dentro do ecra (x =< 32)

JLT boat2\_handler\_1\_continue

MOV R3, 0F0H

CMP R2, R3 ; verifica se barco saiu do ecra (x > 32)

JLT boat2\_handler\_1\_destroy\_boat

boat2\_handler\_1\_continue:

MOVB [R0], R2 ; atualiza a coordenada\_x dos 3 enderecos

MOVB [R1], R2

MOVB [R4], R2

CALL draw\_string ; desenha o barco 1 nas novas coordenadas (x+1,y)

MOV R0, boat2\_clock

MOV R1, 0

MOV [R0], R1 ; muda o boat1\_clock para 0

JMP boat2\_handler\_1\_end

boat2\_handler\_1\_destroy\_boat: ; destroi o barco porque saiu do ecra

MOV R0, boat2\_state

MOV R1, 0

MOV [R0], R1 ; muda o estado do barco 1 para o estado 0

boat2\_handler\_1\_end:

POP R4

POP R3

POP R2

POP R1

POP R0

RET

bullet\_handler:

PUSH R0

PUSH R1

MOV R1, bullet\_state ; vai buscar o estado da bala

MOV R0, [R1]

CMP R0, 0

JEQ bullet\_handler\_0 ; se o estado da bala for 0

CMP R0, 1

JEQ bullet\_handler\_1 ; se o estado da bala for 1

POP R1

POP R0

RET

bullet\_handler\_0: ; Estado 0: Quando a bala aparece no ecra

PUSH R2

PUSH R3

PUSH R4

MOV R1, bullet ; R1: endereco da coordenada\_x da bala

MOV R2, erase\_bullet ; R2: endereco da coordenada\_x do apaga bala

MOV R3, 0H

MOVB [R1], R3 ; coordenada\_x da bala passa para 0

MOVB [R2], R3 ; coordenada\_x do apaga bala passa para 0

MOV R4, submarine

ADD R4, 1 ; R4: endereco da coordenada\_y do submarino

MOVB R3, [R4] ; R3: coordenada\_y do submarino

ADD R3, 1 ; Aumenta em 1 unidade a coordenada\_y da bala

ADD R1, 1 ; endereco da coordenada\_y da bala

ADD R2, 1 ; endereco da coordenada\_y do apaga bala

MOVB [R1], R3 ; atualiza a coordenada\_y da bala

MOVB [R2], R3 ; atualiza a coordenada\_y do apaga bala

MOV R1, bullet\_state

MOV R0, 1

MOV [R1], R0 ; atualiza o estado da bala para o estado 1

POP R4

POP R3

POP R2

POP R1

POP R0

RET

bullet\_handler\_1: ; Estado 1: Movimento da bala

PUSH R2

PUSH R3

MOV R0, bullet\_clock

MOV R1, [R0]

CMP R1, 1 ; verifica se o bullet\_clock nao e 1

JNE bullet\_handler\_1\_end

MOV R0, erase\_bullet

CALL draw\_string ; apaga a bala

MOV R0, bullet ; R0: endereco da coordenada\_x da bala

MOV R1, erase\_bullet ; R1: endereco da coordenada\_x do apaga bala

MOVB R2, [R0] ; R2: coordenada\_x da bala

ADD R2, 1 ; aumenta a coordenada\_x da bala em 1 unidade

MOV R3, 20H

CMP R2, R3 ; verifica se a coordenada\_x da bala saiu do ecra (x > 32)

JGE bullet\_handler\_1\_destroy\_bullet

MOVB [R0], R2 ; atualiza a coordenada\_x da bala

MOVB [R1], R2 ; atualiza a coordenada\_x do apaga bala

CALL draw\_string ; desenha a bala nas novas coordenadas

MOV R0, bullet\_clock

MOV R1, 0

MOV [R0], R1 ; muda o bullet\_clock para 0

JMP bullet\_handler\_1\_end

bullet\_handler\_1\_destroy\_bullet: ; Destroi a bala

MOV R0, bullet\_state

MOV R1, 0

MOV [R0], R1 ; muda o estado da bala para 0

bullet\_handler\_1\_end:

POP R3

POP R2

POP R1

POP R0

RET

; Funcao que desenha uma string no ecra

; Args: R0 - String

draw\_string:

PUSH R0

PUSH R1

PUSH R2

PUSH R3

PUSH R4

PUSH R5

PUSH R6

PUSH R7

PUSH R8

PUSH R9

MOVB R4, [R0] ; R4: canto superior esquerdo do desenho (x)

ADD R0, 1

MOVB R5, [R0] ; R5: canto superior esquerdo do desenho (y)

ADD R0, 1

MOVB R6, [R0] ; R6: tamanho do x do desenho

ADD R0, 1

MOVB R7, [R0] ; R7: tamanho do y do desenho

ADD R0, 1 ; Colocar R0 no endereco do primeiro pixel a desenhar

MOV R9, 0 ; R9: Contador do y atual (i)

draw\_string\_y: ; for (i = 0; i < tamanho\_y; i++)

CMP R9, R7

JGE draw\_string\_y\_end

MOV R8, 0 ; R8: Contador do x atual (j)

draw\_string\_x: ; for (j = 0; j < tamanho\_x; j++)

CMP R8, R6

JGE draw\_string\_x\_end

MOV R1, R8 ; Copia do x atual

ADD R1, R4 ; Adicionar o canto para ver o x onde escrever

MOV R2, R9 ; Copia do y atual

ADD R2, R5 ; Adicionar o canto para ver o y onde escrever

MOVB R3, [R0] ; Ler o pixel a desenhar

CALL draw\_pixel ; Desenhar o pixel em questao (R1, R2 e R3 ja tem os args)

ADD R0, 1 ; Avancar para o proximo pixel

ADD R8, 1

JMP draw\_string\_x

draw\_string\_x\_end:

ADD R9, 1

JMP draw\_string\_y

draw\_string\_y\_end:

POP R9

POP R8

POP R7

POP R6

POP R5

POP R4

POP R3

POP R2

POP R1

POP R0

RET

; Funcao que desenha um determinado pixel no ecra

; Args: R1 - x

; R2 - y

; R3 - 2, 1 ou 0 (ignorar, escrever ou apagar)

draw\_pixel:

PUSH R0

PUSH R1

PUSH R2

PUSH R3

PUSH R4

PUSH R5

PUSH R6

CMP R3, 2

JEQ draw\_end

MOV R4, 00FFH ; Aplicar uma mascara para 8 bits

AND R1, R4

AND R2, R4

CMP R1, 0

JLT draw\_end

CMP R2, 0

JLT draw\_end

MOV R0, 1FH

CMP R1, R0

JGT draw\_end

CMP R2, R0

JGT draw\_end

MOV R0, DISPLAY ; Endereco base

MOV R6, 8 ; Usado para operacoes

MOV R4, R1 ; Criar uma copia de R2

DIV R4, R6 ; Dividir Coluna por 8

ADD R0, R4 ; Somar isso ao endereco

MOD R1, R6 ; Modulo da coluna por 8, para saber onde estamos a contar do inicio do grupo

MOV R6, 4 ; Para multiplicacoes

MUL R2, R6 ; Multiplicar a linha por 4 (grupos por linha)

ADD R0, R2 ; Somar ao endereco

; Obter o bit a desenhar (por exemplo, 0010 0000 para desenhar o terceiro pixel de um grupo)

MOV R4, onehot\_table ; Endereco da lista de onehots

MOV R6, 2 ; Para a multiplicacao

MUL R1, R6 ; Multiplicar o x por 2, para ver o indice na lista

ADD R4, R1 ; Adicionar o indice na lista ao endereco na lista

MOV R4, [R4] ; E ler para R4

draw\_draw:

MOVB R5, [R0] ; Ler o que ja estava no display nesse grupo de pixeis

AND R3, R3 ; Atualizar bits de estado para R3

JZ draw\_0

draw\_1:

OR R5, R4 ; Colocar o pixel no sitio

JMP draw\_apply

draw\_0:

NOT R4 ; Negar R4 para apagar

AND R5, R4 ; Colorcar o pixel no sitio

draw\_apply:

MOVB [R0], R5 ; Desenhar o grupo de novo

draw\_end:

POP R6

POP R5

POP R4

POP R3

POP R2

POP R1

POP R0

RET

; Funcao que le do teclado

; Returns: last\_key: -1 se nenhuma, a tecla se alguma

get\_key:

PUSH R0

PUSH R1

PUSH R2

PUSH R3

PUSH R4

PUSH R5

PUSH R6

PUSH R7

MOV R1, KEYPININ ; R1: endereco in do periferico

MOV R2, KEYPINOUT ; R2: endereco out do periferico

MOV R3, 1 ; R3: linha atual

MOV R6, 8 ; R6: constante para comparar

MOV R7, 0FH

get\_key\_cicle:

MOVB [R2], R3 ; Escrever a linha no out

MOVB R4, [R1] ; R4: Coluna (ler a coluna do in)

AND R4, R7

AND R4, R4 ; Atualizar bits de estado

JNZ get\_key\_save ; Guardar a tecla encontrada

SHL R3, 1

CMP R3, R6

JGT get\_key\_return\_null ; Percorremos todas as linhas

JMP get\_key\_cicle

get\_key\_save:

MOV R5, 0 ; iniciar um contador a 0

get\_key\_count1:

ADD R5, 1

SHR R4, 1

JNZ get\_key\_count1

SUB R5, 1 ; numero de vezes que podemos fazer SHL a R4

MOV R0, R5 ; guardar no return (R0)

MOV R5, 0 ; outro contador a 0

get\_key\_count2:

ADD R5, 1

SHR R3, 1

JNZ get\_key\_count2

SUB R5, 1 ; numero de vezes que podemos fazer SHL a R3

MOV R1, 4 ; reutilizar um registo ja nao usado para a multiplicacao

MUL R5, R1 ; linha\*4+coluna

ADD R0, R5

JMP get\_key\_end

get\_key\_return\_null: ; nenhuma tecla carregada, return -1

MOV R0, -1

get\_key\_end:

MOV R1, last\_key

MOV [R1], R0

POP R7

POP R6

POP R5

POP R4

POP R3

POP R2

POP R1

POP R0

RET