

# Projeto Batalha Naval Assembly x86



**Disciplina:** Organização de Sistemas Computacionais (Prática)

**Curso:** Engenharia de Computação

## **Integrantes:**

Artur Yano Contarelli | RA: 24014303

Rafael Martiniano Nogueira Filho | RA: 24008538

Tiago Alves Rodrigues | RA: 24001623

## Capítulo 1: Descrição do Projeto

Este projeto foi inspirado no jogo de tabuleiro Batalha Naval, o qual, há dois jogadores que posicionam suas embarcações dentro de um tabuleiro e adivinham as posições das embarcações do oponente, o que derrubar todas embarcações do adversário primeiro ganha.

Entretanto neste projeto, foi feito de uma forma diferente, ao invés de jogador contra jogador, fez-se jogador contra computador.

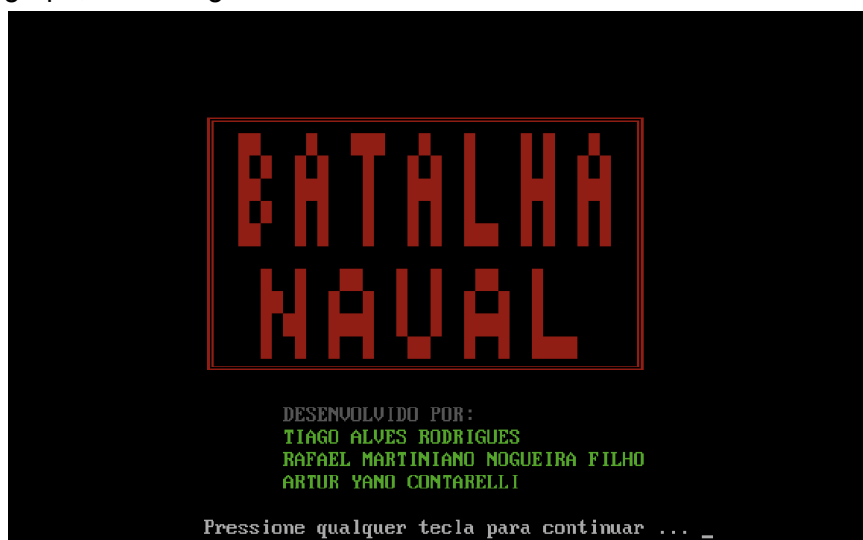
## Lógica do jogo:

Neste jogo, o computador gera um tabuleiro, entre 10 opções diferentes, com embarcações posicionadas aleatoriamente, após isto, esse é copiado para um tabuleiro auxiliar, o qual será responsável pelo registro de danos e comparado com o tabuleiro principal, o que é exibido ao jogador. Estes são compostos por uma matriz 13X16, entretanto, apenas uma seção 10x10 desta matriz é possível interagir. O computador pede ao jogador para selecionar o número da linha de ataque, seguido pela letra da coluna, então o jogo atualiza o tabuleiro, mostrando o caractere 0F7h (≈), caso o jogador tenha acertado água ou o caractere 16h, caso o jogador tenha acertado uma embarcação, no caso de coordenadas inválidas ou repetidas, o computador mostra uma mensagem indicando o que o jogador fez de errado. Ademais implementou-se uma lógica de tiros, o jogador possui 50 tiros para realizar, caso afunde todas embarcações antes do fim destes, o jogador ganha, caso contrário perde, e ao final há uma tela de derrota/vitória, perguntando ao jogador se este deseja jogar novamente, em caso positivo, o jogo reinicia, em caso negativo, encerra.

Em suma, o jogo consiste em uma matriz gerada aleatoriamente na qual o jogador realiza os tiros, tentando acertar todas as embarcações antes que seus tiros se esgotem.

## Escolhas estilísticas:

No começo do código, gera-se uma tela com o nome do jogo e nome dos integrantes do grupo, foto a seguir:



A seguir, é mostrado ao jogador uma tela de regras, explicando brevemente o funcionamento do jogo, foto a seguir:

```
=====BEM-VINDO AO BATALHA NAVAL=====

-> NESTE JOGO EH VOCE CONTRA A CPU
-> EH GERADO UM TABULEIRO 10x10, COM 6 EMBARCACOES
-> SENDO ESTAS 1 ENCOURACADO, 1 FRAGATA, 2 SUBMARINOS E 2 HIDROAVIOES
  - ENCOURCADO: 4 BLOCOS, NA HORIZONTAL OU VERTICAL
  - FRAGATA: 3 BLOCOS, NA HORIZONTAL OU VERTICAL
  - SUBMARINO: 2 BLOCOS, NA HORIZONTAL OU VERTICAL
  - HIDROAVIAO: 4 BLOCOS EM FORMATO DE T
-> ACERTOS SAO INDICADOS COM _ E ERROS COM ~
-> O JOGO ACABA QUANDO VOCE AFUNDA-LAS OU ACABAREM SEUS TIROS
-> DIGITE A COORDENADA QUE DESEJA ATACAR, EX: OC
-> APERTAR ESC FINALIZA AUTOMATICAMENTE O JOGO

=====BEM-VINDO AO BATALHA NAVAL=====
Pressione qualquer tecla para continuar ... _
```

A próxima tela já é a tela de jogo, nesta está presente no topo uma linha em azul escrita um cabeçalho “Batalha Naval” seguido por uma indicação que pressionar esc encerrará o programa, a esquerda no centro da tela, há uma indicação de quais embarcações ainda estão presentes no tabuleiro, a esquerda no canto inferior, aparece a matriz do tabuleiro e ao centro as mensagens de prompt ao usuário além dos avisos de embarcação afundada e casa inválida, canto inferior direito aparece a contagem de tiros restantes do usuário, imagem a seguir:



Prompts de tiro:

```
Digite o numero da linha para o ataque: 1
Digite a letra da coluna para o ataque: [
```

Mensagem de erro:

```
Coordenada invalida!!
```

Mensagem de coordenada já atacada:

```
ESTA COORDENADA JA FOI ATACADA
Pressione qualquer tecla para continuar ... _
```

Mensagem de embarcação afundada:

```
Voce afundou o Encouracado!
```

```
Voce afundou o Fragata!
```

Voce afundou um Hidroaviao!

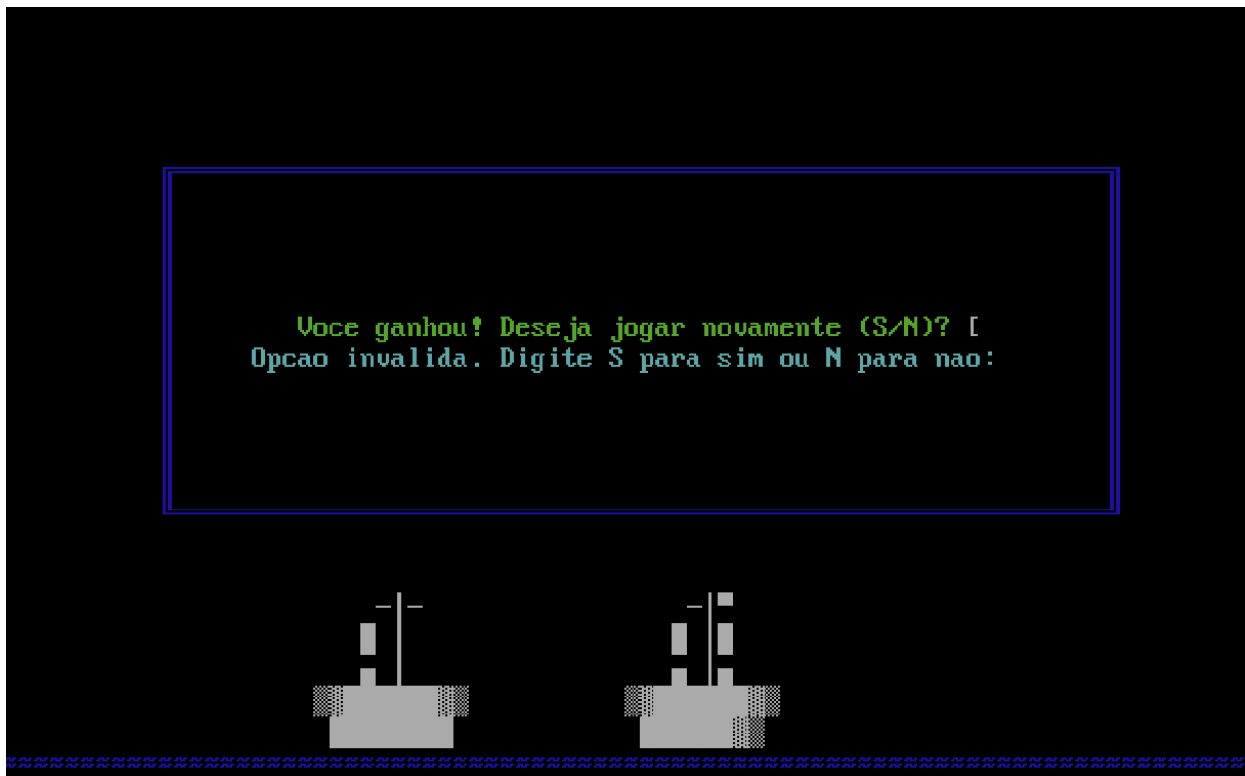
Voce afundou um Submarino!

Tela de fim de jogo, há 2 variações, tela de derrota e tela de vitória, fotos a seguir:

Tela de derrota com mensagem de caracter inválido, no caso de n\N, jogo encerra, no caso de s\S, jogo reinicia:



Tela de vitória com mensagem de tecla invalida, mesma logica da tela de derrota:



## Capítulo 2: Macros e Procedimentos:

Neste código foram usados uma série de macros e procedimentos, os quais estão bem comentados ao decorrer do código, logo será feita apenas uma breve explicação sobre estes aqui

### Macros:

Macros de push\_all e pop\_all:

Função: preservar e restaurar contexto de registradores

```

;-----MACRO DE PUSH ALL-----{
; FUNÇÃO DO MACRO: PRESERVAR CONTEXTO
;
; ONDE USAR: DENTRO DE PROCEDIMENTOS OU MACROS
; QUE UTILIZAM VARIOS REGISTRADORES
;
; COMO USAR: SOMENTE CHAMAR O MACRO E EM CON-
; TEXTOS NECESSÁRIOS
;
; NOME: PUSH_ALL
;-----MACRO DE PUSH ALL-----}

PUSH_ALL MACRO
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    PUSH SI
    PUSH DI
    PUSH BP
ENDM

;-----MACRO DE POP ALL-----{
; FUNÇÃO DO MACRO: RESTAURAR CONTEXTO
;
; ONDE USAR: DENTRO DE PROCEDIMENTOS OU MACROS
;
; COMO USAR: SOMENTE CHAMAR O MACRO E EM CON-
; TEXTOS NECESSÁRIOS
;
; NOME: POP_ALL
;-----MACRO DE POP ALL-----}

POP_ALL MACRO
    POP BP
    POP DI
    POP SI
    POP DX
    POP CX
    POP BX
    POP AX
ENDM

```

Macro de fim de linha:  
passa para a próxima linha

```

;-----MACRO DE FIM DE LINHA-----{
;
;
; FUNÇÃO DO MACRO: PULAR PARA A PRÓXIMA LINHA
;
;
; ONDE USAR: QUANDO QUISE PULAR PARA A PRÓ-
; XIMA LINHA
;
;
; COMO USAR: SOMENTE CHAMAR O MACRO
;
;
; NOME: ENDL; VEM DE→ (END LINE)
;
;-----MACRO DE FIM DE LINHA-----}
ENDL MACRO
    PUSH AX
    PUSH DX

    MOV AH, 2
    MOV DL, 10 ; Line Feed
    INT 21h

    POP AX
    POP DX
ENDM

```

Macro de limpar tela:

Utiliza a interrupção 10h

```

;-----MACRO PARA LIMPAR TELA-----{
;
;
; FUNÇÃO DO MACRO: LIMPAR TELA
;
;
; ONDE USAR: QUANDO QUISE LIMPAR TODA A TELA,
; SEJA DENTRO DE PROCEDIMENTOS OU MACROS
;
;
; COMO USAR: SOMENTE CHAMAR O MACRO
;
;
; NOME: CLEAR_SCREEN
;
;-----MACRO PARA LIMPAR TELA-----}
CLEAR_SCREEN MACRO
    PUSH_ALL
    MOV AX, 600H
    MOV BH, 7H
    MOV CX, 0H
    MOV DX, 184FH
    INT 10H
    POP_ALL
ENDM

```



Macro para exibir string:

```
;-----MACRO PARA EXIBIR STRING-----{  
;  
; FUNÇÃO DO MACRO: EXIBIR STRING  
;  
;  
; ONDE USAR: QUANDO QUISEER EXIBIR UMA STRING, E  
; AO INVES DE USAR AH, 9; INT 21H, USAR ESTE MACRO  
;  
;  
; COMO USAR: CHAMAR O MACRO, E PASSAR A STRING AP  
;  
;  
; EXEMPLO DE USO: PRINTS STRING, ONDE STRING POSS  
; UMA STRING  
;  
;  
; NOME: PRINTS → (PRINT STRING)  
;  
;-----MACRO PARA EXIBIR STRING-----}  
PRINTS MACRO STRING  
    PUSH DX  
    PUSH AX  
  
    LEA DX, STRING  
    MOV AH, 9  
    INT 21h  
  
    POP AX  
    POP DX  
ENDM
```

Macro para exibir caractere:

```
;-----MACRO PARA PRINT CHAR-----{  
;  
; FUNÇÃO DO MACRO: EXIBIR CHARACTER  
;  
; ONDE USAR: QUANDO QUISER EXIBIR UM CHARACTERER  
;  
; COMO USAR: CHAMAR O MACRO, E PASSAR O CHARACTER  
;  
; EXEMPLO DE USO: PRINTC CHAR, ONDE CHAR POSSUI  
; UM CARACTE  
;  
; NOME: PRINTC → (PRINT CHAR)  
;  
;-----MACRO PARA PRINT CHAR-----}  
PRINTC MACRO CHAR  
    PUSH DX  
    PUSH AX  
    MOV AH, 2  
    MOV DX, CHAR  
    INT 21h  
    POP AX  
    POP DX  
ENDM
```

Macro de posicionar cursor na tela:

coloca em dh a linha que deseja posicionar e em dl, a coluna, utiliza a interrupção 10h.

```

;-----MACRO PARA POSICIONAR CURSOR-----{
;
;  FUNÇÃO DO MACRO: POSICIONA O CURSOR NA TELA
;
;  ONDE USAR: QUANDO PRECISAR POSICIONAR O CURSOR
;  EM UMA LINHA E COLUNA ESPECIFICOS DA TELA
;
;  COMO USAR: CHAMAR MACRO E INDICAR PARAMETROS DE
;  LINHA E COLUNA RESPECTIVAMENTE
;
;  EXEMPLO DE USO:  POS_CURSOR 10, 20
;
;  NOME: POS_CURSOR → (POSICIONAR CURSOR)
;
;-----MACRO PARA POSICIONAR CURSOR-----}
POS_CURSOR MACRO linha, coluna
    PUSH AX
    PUSH DX
    PUSH BX

    MOV AH, 2
    MOV BH, 0
    MOV DH, linha
    MOV DL, coluna
    INT 10H

    POP BX
    POP DX
    POP AX
ENDM

```

Macro de input sem echo:

```

;-----MACRO DE INPUT SEM ECHO-----{
;
;  FUNÇÃO DO MACRO: FAZ UM INPUT SEM ECHO
;
;  ONDE USAR: QUANDO QUISE QUE O USUÁRIO DIGITE
;  ALGO SEM QUE O CARACTER DIGITADO SEJA ARMAZENADO
;
;  COMO USAR: SOMENTE CHAMAR O MACRO
;
;  NOME: PPC → (PRESSIONE PARA CONTINUAR)
;
;-----MACRO DE INPUT SEM ECHO-----}
PPC MACRO
    PUSH AX
    MOV AH, 7
    INT 21H
    POP AX
ENDM

```

Macro de mudar cor de strings:

recebe em bl a cor que deseja que a string seja e em si a string a mudar de cor, ambas são passadas para o procedimento muda cor, o qual utiliza interrupção 10h

```
;-----MACRO PARA MUDAR COR DE STRING-----{  
;  
; FUNÇÃO DO MACRO: MUDAR COR DE STRINGS TERMINADAS EM $  
;  
;  
; COMO USAR: CHAMAR MACRO, PASSAR A STRING QUE QUER MUDAR A  
; COR E A COR PARA QUAL QUER MUDAR, EM BL É ARMAZENADA A COR  
; E EM [SI] A STRING  
;  
;  
; NOME → PRINT_COR  
;  
;-----MACRO PARA MUDAR COR DE STRING-----}  
  
PRINT_COR MACRO STRING, COR  
PUSH SI  
PUSH BX  
  
    LEA SI, STRING  
    MOV BL, COR  
    CALL MUDA_COR  
  
POP BX  
POP SI  
ENDM
```

Macro de tabulação:

usa int 10h

```

;-----MACRO PARA TABULAÇÃO-----{
;
;   FUNÇÃO DO MACRO: TABULAÇÃO, CRIAR ESPAÇOS EM BRANCO
;
;   COMO USAR: CHAMAR MACRO MAIS NUMERO DE VEZES QUE QUER QUE
;   A TABULAÇÃO SEJA APLICADA
;
;   NOME → TAB
;
;-----MACRO PARA TABULAÇÃO-----}

TAB MACRO N
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX

    MOV AH, 3
    MOV BH, 0
    INT 10h

    MOV AH, 2
    ADD DL, N
    INT 10h

    POP DX
    POP CX
    POP BX
    POP AX
ENDM

```

Macro para exibir número:  
similar a printf, entretanto soma 30h a bl

```

;-----MACRO PARA IMPRIMIR NÚMERO-----{
;
;   FUNÇÃO DO MACRO: IMPRIMIR UM NÚMERO INTEIRO DE UMA CASA
;
;   COMO USAR: CHAMAR O MACRO COM O NÚMERO QUE DESEJA IMPRIMIR
;
;   NOME → PRINTNUM
;
;-----MACRO PARA IMPRIMIR NÚMERO-----}

PRINTNUM MACRO NUM
    PUSH_ALL

    MOV BL, NUM
    ADD BL, 30H
    MOV AH, 2
    MOV DL, BL
    INT 21H

    POP_ALL
ENDM

```

## Procedimentos:

obs: procedimentos são mais longos, logo aconselho olhá-los em nosso github

Procedimento de Exibir tela inicial:

```

;=====PROCEDIMENTO DE TELA DE INICIAL=====
;
; FUNÇÃO: MOSTRAR O LOGO "BATALHA NAVAL" AO INICIO DO CÓDIGO
;
; COMO USAR: CHAMAR O PROCEDIMENTO AO INICIO DO CODIGO SOMENTE
;
; NOME: TELA_INICIAL
;
;=====PROCEDIMENTO DE TELA DE INICIAL=====}

```

TELA\_INICIAL PROC

CLEAR\_SCREEN

```

POS_CURSOR      5, 18
PRINT_COR        L0, VERMELHO
POS_CURSOR      6, 18
PRINT_COR        L1, VERMELHO
POS_CURSOR      7, 18
PRINT_COR        L2, VERMELHO
POS_CURSOR      8, 18
PRINT_COR        L3, VERMELHO
POS_CURSOR      9, 18
PRINT_COR        L4, VERMELHO
POS_CURSOR     10, 18
PRINT_COR        L5, VERMELHO
POS_CURSOR     11, 18
PRINT_COR        L, VERMELHO
POS_CURSOR     12, 18
PRINT_COR        L6, VERMELHO
POS_CURSOR     13, 18
PRINT_COR        L7, VERMELHO
POS_CURSOR     14, 18
PRINT_COR        L8, VERMELHO
POS_CURSOR     15, 18
PRINT_COR        L9, VERMELHO
POS_CURSOR     16, 18
PRINT_COR       L10, VERMELHO

POS_CURSOR     18, 25
PRINT_COR       BY, CINZA_ESCURO

POS_CURSOR     19, 25
PRINT_COR      TIAGO, VERDE

POS_CURSOR     20, 25
PRINT_COR     RAFAEL, VERDE

POS_CURSOR     21, 25
PRINT_COR     ARTUR, VERDE

```

Procedimento de mostrar tela de regras:



```
;=====PROCEDIMENTO DE MANUAL DE INSTRUÇÃO=====
;
;
; FUNÇÃO: MOSTRAR O MANUAL DE INTRUÇÃO DO JOGO
;
;
; COMO USAR: CHAMAR O PROCEDIMENTO AO INICIO DO CODIGO SOMENTE
;
;
; NOME: MANUAL_INSTRUCAO
;
;=====PROCEDIMENTO DE MANUAL DE INSTRUÇÃO=====}
```

MANUAL\_INSTRUCAO PROC

CLEAR\_SCREEN

; IMPRIMIR MANUAL DE INSTRUÇÕES

POS\_CURSOR 3, 5

PRINT\_COR REGRAS, CINZA\_ESCURO

POS\_CURSOR 6, 5

PRINT\_COR REGRA1, CINZA\_CLARO

POS\_CURSOR 7, 5

PRINT\_COR REGRA2, CINZA\_CLARO

POS\_CURSOR 8, 5

PRINT\_COR REGRA3, CINZA\_CLARO

POS\_CURSOR 9, 8

PRINT\_COR FORM\_E, VERMELHO

POS\_CURSOR 10, 8

PRINT\_COR FORM\_F, VERMELHO

POS\_CURSOR 11, 8

PRINT\_COR FORM\_S, VERMELHO

POS\_CURSOR 12, 8

PRINT\_COR FORM\_H, VERMELHO

POS\_CURSOR 13, 5

PRINT\_COR REGRA4, CINZA\_CLARO

POS\_CURSOR 14, 5

PRINT\_COR REGRA5, CINZA\_CLARO

POS\_CURSOR 15, 5

PRINT\_COR REGRA6, CINZA\_CLARO

POS\_CURSOR 16, 5

PRINT\_COR REGRA7, CINZA\_CLARO

POS\_CURSOR 19, 5

PRINT\_COR REGRAS, CINZA\_ESCURO

POS\_CURSOR 20,15

PRINTS PTC

PPC

RET

MANUAL\_INSTRUCAO ENDP

Procedimento de exibir matriz:

```
;=====PROCEDIMENTO PARA IMPRIMIR MATRIZ=====
;
;
; FUNÇÃO DO PROCEDIMENTO: IMPRIMIR MATRIZ DE 16BITS,(DW)
;
;
; ONDE USAR: QUANDO QUISER IMPRIMIR UMA MATRIZ DW
;
;
; COMO USAR: CHAMAR O PROCEDIMENTO
;
;
; NOME: PRINT_MATRIZ
;
;=====PROCEDIMENTO PARA IMPRIMIR MATRIZ=====}

PRINT_MATRIZ PROC
    PUSH_ALL

    XOR     BX, BX
    XOR     SI, SI
    XOR     DX, DX

    MOV     CX, CONTADOR ; CX RECEBE TAMANHO DA MATRIZ

MOSTRAR_MATRIZ:
    PRINTC  TABULEIRO[BX][SI]

    ADD     SI, 2

    CMP     SI, FIM_LINHA
    JA      NOVA_LINHA

    LOOP    MOSTRAR_MATRIZ

NOVA_LINHA:
    ENDL
    ADD     BX, FIM_LINHA + 2
    XOR     SI, SI
    CMP     BX, ULTIMA_POS

    JA      FIM_PRINT
    JMP     MOSTRAR_MATRIZ

FIM_PRINT:

    POP_ALL
    RET

PRINT_MATRIZ ENDP
```

Procedimento para imprimir tiros restantes:

```
;=====PROCEDIMENTO DE IMPRIMIR TIROS RESTANTES=====
;
; FUNÇÃO: IMPRIMIR O NÚMERO DE TIROS RESTANTES NA TELA
;
; COMO USAR: CHAMAR QUANDO PRECISAR EXIBIR O NÚMERO DE TIROS RESTANTES
;
; COMO FUNCIONA: POSICIONA O CURSOR, IMPRIME O TEXTO "TIROS RESTANTES" EM VERMELHO,
; CONVERTE O NÚMERO DE TIROS RESTANTES PARA CARACTERES E IMPRIME NA TELA
;
; PROCEDIMENTOS CHAMADOS: POS_CURSOR, PRINT_COR, PUSH_ALL, POP_ALL
;
; NOME: PRINT_TIROS
;=====PROCEDIMENTO DE IMPRIMIR TIROS RESTANTES=====}

PRINT_TIROS PROC
PUSH_ALL
POS_CURSOR      24, 60
PRINT_COR       TIROS_RESTANTES, VERMELHO
    XOR         AX, AX
    XOR         BX, BX
    XOR         CX, CX      ; contador de dígitos
    MOV         AL, TIROS
    MOV         BX, 10      ; divisor
REPEAT:
    XOR         DX, DX      ; prepara parte alta do dividendo
    DIV         BX          ; AX = quociente  DX = resto
    PUSH        DX          ; salva resto na pilha
    INC         CX          ; contador = contador +1
    OR          AX, AX      ; quociente = 0?
    JNE         REPEAT      ; SE NÃO FOR, PULA PARA REP1

    MOV         AH, 2 ;LOOP POR CX VEZES
IMP_LOOP:
    POP         DX          ; dígito em DL
    OR          DL, 30H
    INT         21H
    LOOP        IMP_LOOP
POP_ALL
PRINT_TIROS ENDP
```

Procedimento de gerar números aleatórios:

```

;=====PROCEDIMENTO DE GERAR NUMERO ALEATORIO=====
;
; FUNÇÃO: GERAR UM NÚMERO ALEATÓRIO ENTRE 0 E 9
;
; COMO USAR: CHAMAR QUANDO PRECISAR POSICIONAR EMBARCAÇÕES AO COMEÇO DO JOGO
;
; COMO FUNCIONA: GERA UM NÚMERO ALEATÓRIO ENTRE 0 E 9 COM A INTERRUPÇÃO 1AH
; DIVIDE POR 10 PARA GERAR ESTE NÚMERO EX  $X \% 10 = 0 \leq X \leq 9$ 
;
; PROCEDIMENTOS CHAMADOS: NENHUM
;
; NOME: ALEATORIO
;
;=====PROCEDIMENTO DE GERAR NÚMERO ALEATÓRIO=====}
ALEATORIO PROC
    MOV     AH, 0H                ; Chama a interrupção 1Ah para obter o número de ticks
    INT     1AH

    MOV     AX, DX                ; Coloca o valor do timer em AX
    XOR     DX, DX                ; Limpa DX para a divisão
    MOV     BX, 10                ; O divisor é 10 para limitar o valor de 0 a 9
    DIV     BX                    ; Divide AX por 10
    MOV     NUM_ALEATORIO, DX     ; Armazena o resto (0-9) em NUM_ALEATORIO
    RET

ALEATORIO ENDP

```

Procedimento de pegar coordenada:

```

934 ;=====PROCEDIMENTO PARA PEGAR POSIÇÃO DE ATAQUE DO JOGADOR=====
935 ;
936 ; FUNÇÃO: PEGAR RESPECTIVAMENTE LINHA E COLUNA A QUAL O JOGADOR DESEJA ATACAR,
937 ; VERIFICA SE O LOCAL ESTÁ DENTRO OU FORA DA ÁREA DE ATAQUE, ADEMAIS VERIFICA
938 ; SE 'ESC' FOI PRESSIONADO, CASO TENHA, ACABA O JOGO
939 ;
940 ; COMO USAR: CHAMAR QUANDO O JOGADOR FOR ATACAR
941 ;
942 ; NOME: PEGAR_COORDENADAS
943 ;
944 ;=====PROCEDIMENTO PARA PEGAR POSIÇÃO DE ATAQUE DO JOGADOR=====}
945 PEGAR_COORDENADAS PROC

```

obs: procedimento muito longo, veja-o aqui nas linhas indicadas ao lado:

<https://github.com/tiago-rods/BattleShip-Assembly.git>

Procedimento para gerar tabuleiro aleatoriamente

```

1029 ;=====TABULEIRO PSEUDOALEATÓRIO=====
1030 ;
1031 ; O QUE FAZ: GERA UM TABULEIRO COM EMBARCAÇÕES POSICIONADAS
1032 ; PSEUDOALEATÓRIAMENTE
1033 ;
1034 ; COMO FAZ: GERA UM NUMERO ALEATORIO, E FAZ UM SWITCH CASE,
1035 ; DEPENDENDO DE QUAL NÚMERO CAIR, USA UM PRESET DIFERENTE
1036 ; DE TABULEIRO
1037 ;
1038 ; PROCEDIMENTOS CHAMADOS: ALEATORIO
1039 ;
1040 ; NOME: GERA_TABULEIRO
1041 ;
1042 ;=====TABULEIRO PSEUDOALEATÓRIO=====
1043 GERA_TABULEIRO PROC

```

obs: procedimento muito longo, veja-o aqui nas linhas indicadas ao lado:

<https://github.com/tiago-rods/BattleShip-Assembly.git>

Procedimento de atualizar ataque

```

1184 ;=====PROCEDIMENTO DE UPDATE TABULEIRO=====
1185 ;
1186 ; O QUE FAZ: ATUALIZA O TABULEIRO AUXILIAR E TABULEIRO DE JOGO
1187 ;
1188 ; COMO FAZ: COMPARA A POSIÇÃO DE AMBOS TABULEIROS PARA VER SE A CASA
1189 ; É VALIDA, CASO SEJA, ATUALIZA PARA ACERTO OU ERRO, ADEMAIS VERIFICA
1190 ; SE EMBARCAÇÕES AFUNDARAM
1191 ;
1192 ; NOME: UPDATE_ATAQUE
1193 ;
1194 ;=====PROCEDIMENTO DE UPDATE TABULEIRO=====
1195
1196 > UPDATE_ATAQUE PROC;ATUALIZA A MATRIZ COM O ATAQUE DO USUÁRIO PROC

```

obs: procedimento muito longo, veja-o aqui nas linhas indicadas ao lado:

<https://github.com/tiago-rods/BattleShip-Assembly.git>

Procedimento para exibir embarcações restantes:

```

;=====PROCEDIMENTO DE ATUALIZAÇÃO DE TABELA DE NAVIOS=====}{
;
; FUNÇÃO: ATUALIZAR A TABELA DE NAVIOS NO JOGO
;
; NOME: UPDATE_TABELA_NAVIOS
;
;=====PROCEDIMENTO DE ATUALIZAÇÃO DE TABELA DE NAVIOS=====}
UPDATE_TABELA_NAVIOS PROC

    POS_CURSOR      4, 3
    PRINT_COR        TABELA_FRAGATA, CIANO
    PRINTNUM          COUNT_TABELA_FRAGATA

    POS_CURSOR      5, 3
    PRINT_COR        TABELA_ENCOURACADO, CIANO
    PRINTNUM          COUNT_TABELA_ENCOURACADO

    POS_CURSOR      6, 3
    PRINT_COR        TABELA_SUBMARINO, CIANO
    PRINTNUM          COUNT_TABELA_SUBMARINO

    POS_CURSOR      7, 3
    PRINT_COR        TABELA_HIDROAVIAO, CIANO
    PRINTNUM          COUNT_TABELA_HIDROAVIAO
    RET
UPDATE_TABELA_NAVIOS ENDP

```

Procedimentos para verificar navios afundados:

```

1334 ;=====PROCEDIMENTO DE VERIFICAR AFUNDADOS=====}{
1335 ;
1336 ; O QUE FAZ: VERIFICA SE EMBARCAÇÃDEC, CASO POSITIVO, MOSTRA MENSAGEM
1337 ; COM O NOME DA EMBARCAÇÃO, AVISANDO QUE ELA FOI AFUNDADA
1338 ;
1339 ; COMO FAZ: VERIFICA A MATRIZ BUSCANDO UM ELEMENTO, CASO NÃO O ENCONTRE,
1340 ; MOSTRA MENSAGEM DE AFUNDADA, CASO ENCONTRE, EMBARCAÇÃO NÃO FOI AFUNDADA
1341 ;
1342 ; PROCEDIMENTOS CHAMADOS: NENHUM
1343 ;
1344 ; MACROS UTILIZADOS: AFUNDOU E PRINTS
1345 ;
1346 ; NOME: VERIFICA_AFUNDOU
1347 ;
1348 ;=====PROCEDIMENTO DE VERIFICAR AFUNDADOS=====}
1349 ~ VERIFICA_AFUNDOU PROC
1350

```

obs: procedimento muito longo, veja-o aqui nas linhas indicadas ao lado:

<https://github.com/tiago-rods/BattleShip-Assembly.git>

Procedimento para verificar fim do jogo:

```
1439 ;=====PROCEDIMENTO DE VERIFICAÇÃO DE FIM DE JOGO=====
1440 ;
1441 ; FUNÇÃO: VERIFICAR SE AINDA EXISTE EMBARCAÇÃO E SE O JOGADOR DESEJA JOGAR NOVAMENTE
1442 ;
1443 ; COMO FAZ: PARA CADA EMBARCAÇÃO ATINGIDA, ADICIONAR UM CONTADOR, E QUANDO O CONTADOR CHEGAR A 6, ACABA.
1444 ;
1445 ; NOME: VERIFICA_FIM_JOGO
1446 ;
1447 ;=====PROCEDIMENTO DE VERIFICAÇÃO DE FIM DE JOGO=====
1448
1449 VERIFICA_FIM_JOGO PROC
```

obs: procedimento muito longo, veja-o aqui nas linhas indicadas ao lado:

<https://github.com/tiago-rods/BattleShip-Assembly.git>

Procedimento para mudar cor de strings:

```
=====PROCEDIMENTO PARA MUDAR COR DE STRINGS=====
;
; COMO FUNCIONA: SALVA O ENDEREÇO DA STRING EM SI E DA COR EM BL
;
; ONDE USAR: QUANDO QUISEER COLORIR UM TEXTO
;
; PROCEDIMENTOS CHAMADOS: NENHUM
;
; MACROS USADOS: TAB
;
=====PROCEDIMENTO PARA MUDAR COR DE STRINGS=====
MUDA_COR PROC
    PUSH        AX                ;Feita para padronizar a impressão de cor ao longo do programa e
    PUSH        CX                ;quando for feita a chamada dela apenas precisa passar o parametro
    PUSH        SI                ;de cor para BL.

    XOR         BH, BH            ;Zera bit superior de bx(bh é utilizado para mudar a cor do fundo)
    MOV         CX, 1

    REPETE:
    MOV         AH, 9             ;chama a função de colocar o caracter na tela
    MOV         AL, [SI]          ;incrementa para poder ir para o proximo caracter e imprimir a string totalmente
    INT         10H

    INC         SI
    TAB         1

    MOV         AL, [SI]
    CMP         AL, '$'
    JNE         REPETE            ;Fim de string marcada com $, logo enquanto for diferente, percorre a string

    POP         SI
    POP         CX
    POP         AX
    RET
MUDA_COR ENDP

END MAIN
```

## Capítulo 3

## **Explicações de matéria não dada em aula:**

INT 1Ah, esta interrupção utiliza dados do horário do sistema, ao utilizar a função ah, 0, obtém-se o número de ticks do sistema, o qual é atualizado constantemente, e utilizar o módulo 10, garante que o número que resulta, será entre 0 e 9, assim escolhendo um dos tabuleiros.

INT 10h, esta interrupção é utilizada para manipular elementos de video, ela foi utilizada nos seguintes casos:

macro POS\_CURSOR: este pega a linha em DH e a coluna em DL e move o cursor para esta posição

macro de TAB: quando você chama int 10h com AH = 3h, o cursor se move para a posição (DL, DH). O valor de DL representa a coluna e DH representa a linha onde o cursor será posicionado.

Procedimento MUDA\_COR: mov ah, 9 com int 10h, faz com que o código em BL seja transformado em cor de letra da string e o código em BH, a cor do fundo da exibição.

## **Capítulo 4 Manual de uso:**

O jogo inicial exibindo a tela do logo, pressione qualquer tecla para continuar, após serão exibidas as regras, leia-as e pressione qualquer tecla para continuar.

Agora será exibida a tela com o tabuleiro principal, perceba que você possui 50 tiros e 6 embarcações para serem afundadas, ademais ESC pode ser pressionado a qualquer hora para encerrar o programa.

Para jogar, pressione primeiramente o número da linha que deseja atacar, após isso aperte a tecla da letra da coluna que deseja atacar, caso uma posição inválida seja digitada, tiros não serão decrementados e um aviso aparecerá. Seu tiro será contabilizado no tabuleiro, caso erre uma embarcação “~” aparecerá, indicando que água foi acertada, logo você errou o tiro, agora caso veja “■”, parabéns você acertou uma embarcação.

Continue atirando até que afunde todas embarcações ou acabe seus tiros, quando uma posição repetida for digitada, uma mensagem de aviso aparecerá, quando uma embarcação for afundada você também será notificado e será decrementado do contador na parte superior esquerda, a quantidade desta embarcação

Ao final do jogo aparecerá uma tela de rejogabilidade, caso deseje jogar novamente, aperte (S/s), caso deseje sair do programa aperte (N/n).

## **Capítulo 5:**



## Referências Bibliográficas:

PUC Campinas. *OSC Pannain INTEL.pdf*. Universidade Pontifícia Católica de Campinas, 2024. Disponível em: [https://puc-campinas.instructure.com/files/2591190/download?download\\_frd=1](https://puc-campinas.instructure.com/files/2591190/download?download_frd=1). Acesso em: 20 out. 2024.

PUC Campinas. *OSC Pannain OrgComp.pdf*. Universidade Pontifícia Católica de Campinas, 2024. Disponível em: [https://puc-campinas.instructure.com/files/2591191/download?download\\_frd=1](https://puc-campinas.instructure.com/files/2591191/download?download_frd=1). Acesso em: 20 out. 2024.

Fraggle, 2024. Disponível em: <https://fragglet.github.io/dos-help-files/along.hlp/1Ah.html>. Acesso em: 26 out. 2024.

Alkar, C. “ELE336 Week 5 Lecture Notes.” ELE336: Computer Systems and Programming, Hacettepe University, 2014, [https://www.ee.hacettepe.edu.tr/~alkar/ELE336/ele336\\_2014\\_week5.pdf](https://www.ee.hacettepe.edu.tr/~alkar/ELE336/ele336_2014_week5.pdf). Acesso em 2 nov. 2024

Pannain, José. *The Art of Intel x86 Assembly Language*. Universidade Estadual de Campinas, 2007. Disponível em: <https://www.ic.unicamp.br/~pannain/mc404/aulas/pdfs/Art%20Of%20Intel%20x86%20Assembly.pdf>. Acesso em: 6 nov. 2024.

Obs: vale citar que utilizamos IA's somente para ajudar a desenvolver a lógica em algumas parte mas não foram escritas linhas de códigos por estas.