PUC
CAMPINAS
PONTIFÍCIA UNIVERSIDADE CATÓLICA

Projeto Batalha Naval em Assembly 8086

Disciplina: Organização de computadores e linguagem de montagem

Curso: Engenharia de Computação

Grupo: Pedro Ninci (20086120)

e Julio Moura (20126231)

Professor: Ricardo Pannain

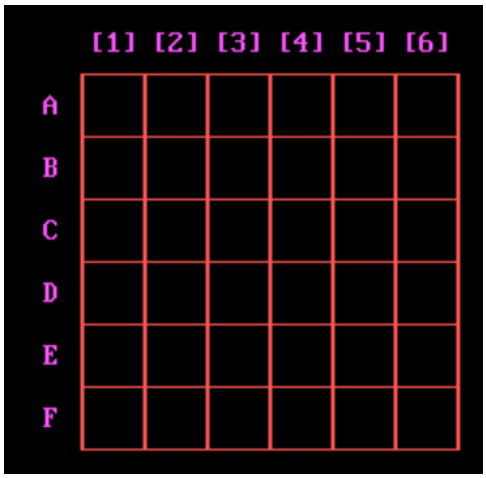
14 de dezembro de 2020

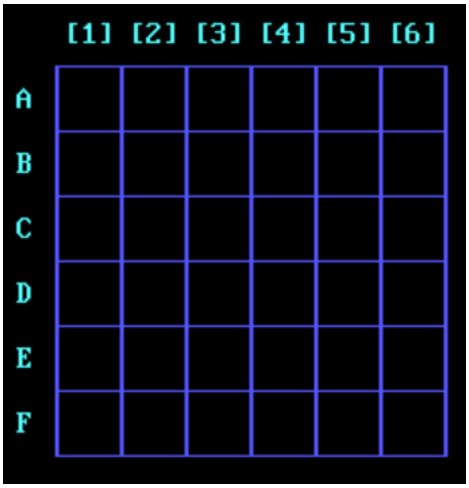
SUMÁRIO

Capítulo 1

Descrição do projeto

O Projeto consiste na implementação do jogo de tabuleiro “Batalha Naval” em Linguagem Assembly. Batalha naval é um jogo de tabuleiro de dois jogadores, no qual os jogadores têm de adivinhar em quais quadrados estão os navios do oponente. Cada jogador terá um tabuleiro de 6X6 quadrados e terão 5 navios para posicionar no mesmo.





Neste jogo, as chamadas para dar os tiros nas respectivas posições consiste em uma letra e um número, elas funcionam da seguinte maneira: a pessoa da vez escolhe uma letra que definirá a linha que deseja acertar e um número que definirá a coluna que deseja acertar, com isso ela tem a posição exata do seu tiro podendo acertar ou não o navio inimigo. Ganha quem afundar todas as embarcações inimigas.

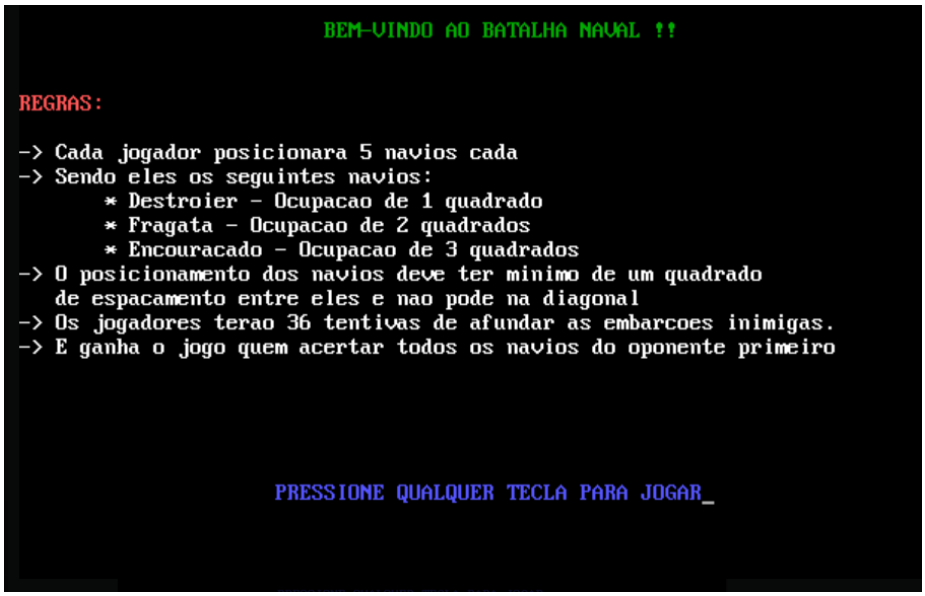
O número de chances para tal equivale ao número total de posições do tabuleiro tendo assim total certeza que um dos jogadores conseguirá afundar todas as embarcações inimigas.

Capítulo 2

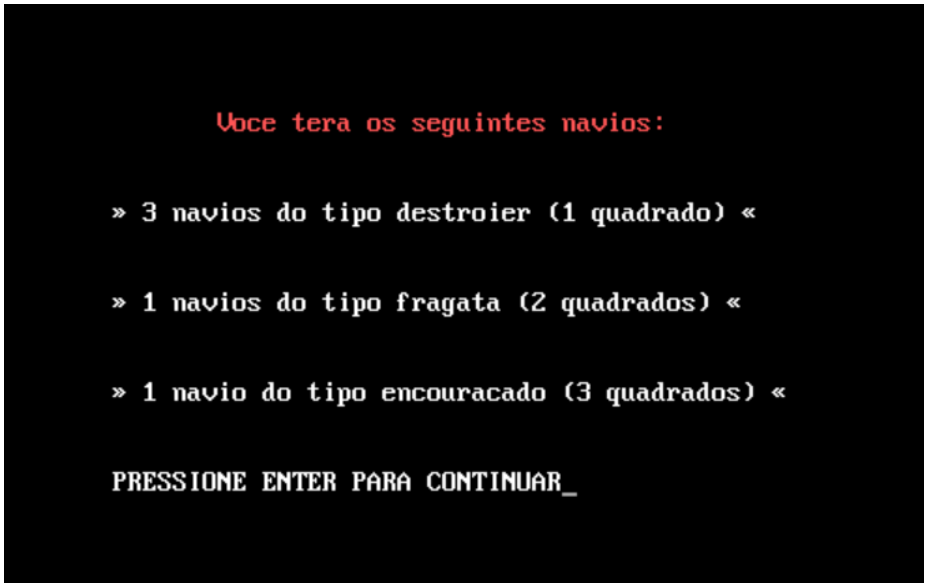
Especificação técnica

2.1 Detalhes do projeto

Quando o jogo é iniciado a primeira tela mostrada tem o objetivo de informar as regras para os jogadores explicando quantos navios tem a disposição para os posicionamentos, quantos quadrados cada navio ocupa, as chances que cada jogador tem e como que termina o jogo.



Na tela subsequente, é mostrado quantos navios de cada tipo os jogadores têm a disposição.

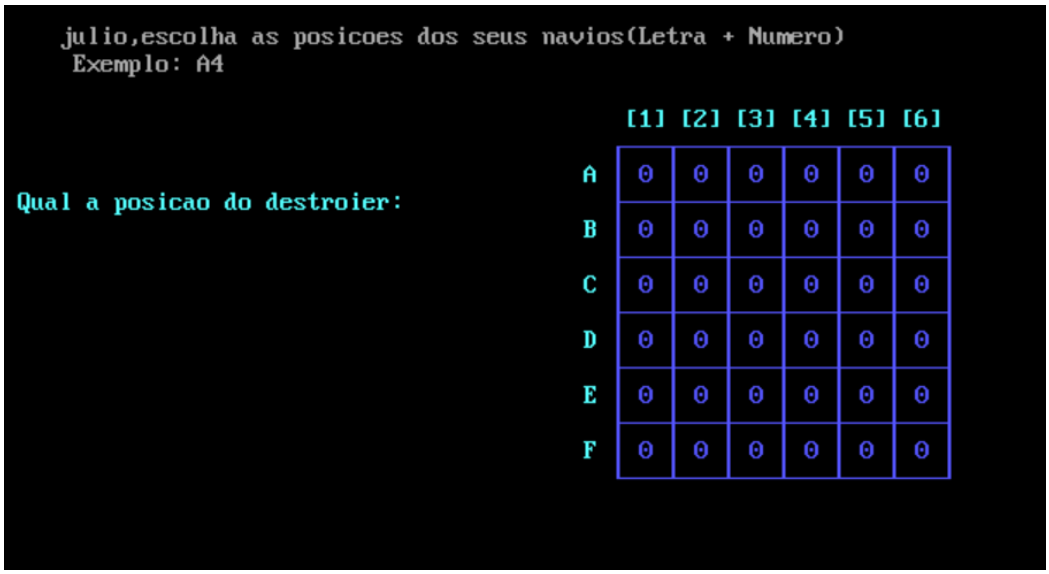


Logo após, é pedido os nomes dos jogadores para serem chamados nas telas de posicionamento dos navios e nas chamadas de tiros:

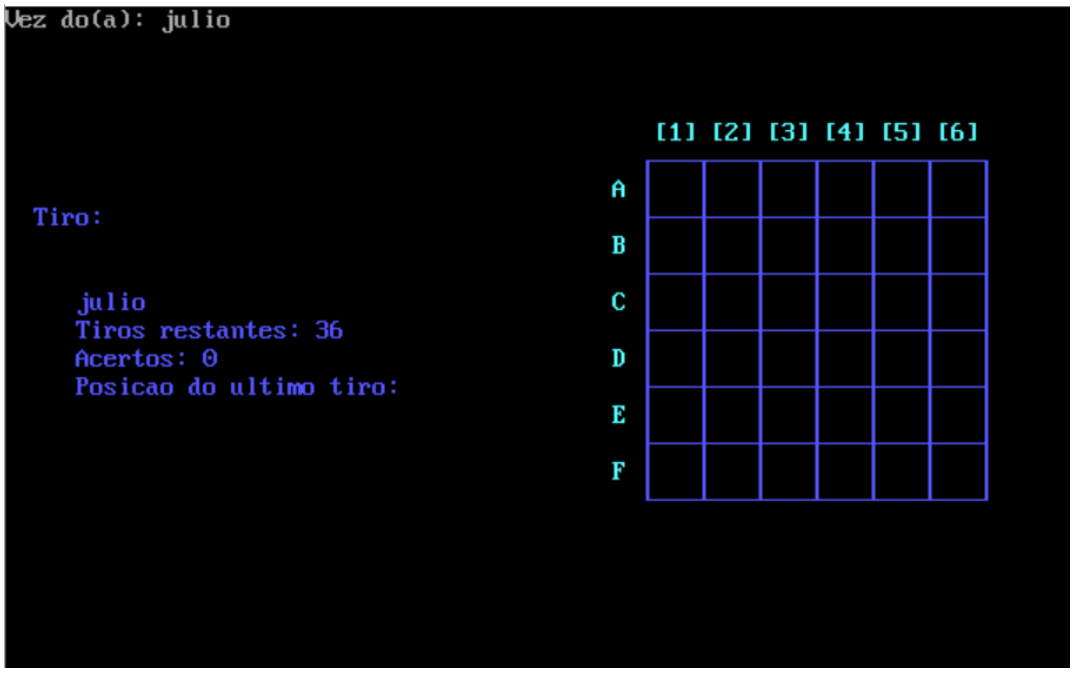
```
Jogador 1, informe seu nome:
```

```
Jogador 2, informe seu nome:
```

A próxima refere-se ao início do jogo, perguntando as escolhas de posição dos respectivos jogadores:



Depois, a tela mostrada é da leitura dos tiros para saber se o jogador acertou ou não o navio inimigo além de mostrar o relatório dos tiros:



2.2 Detalhes de Implementação

Todos os .DATA

```
    INTRODUCAO DB "BEM-VINDO AO BATALHA NAVAL !!$"
    ESPACE DB 10,13,"$"

    REGRA0 DB "REGRAS:$"
    REGRA1 DB "-> Cada jogador posicionara 5 navios cada$"
    REGRA2 DB "-> Sendo eles os seguintes navios:$"
    REGRA3 DB "* Destroier - Ocupacao de 1 quadrado$"
    REGRA4 DB "* Fragata - Ocupacao de 2 quadrados$"
    REGRA5 DB "* Encouracado - Ocupacao de 3 quadrados$"
    REGRA6 DB "-> O posicionamento dos navios deve ter minimo de um quadrado$"
    REGRA7 DB "    de espacamento entre eles e nao pode na diagonal$"
    REGRA8 DB "-> Os jogadores terao 36 tentivas de afundar as embarcoes inimigas.$"
    REGRA9 DB "-> E ganha o jogo quem acertar todos os navios do oponente primeiro$"
    ESCOLHA1 DB "PRESSIONE QUALQUER TECLA PARA JOGAR$"
    NAVIOS1  DB "Voce tera os seguintes navios: $"
    NAVIOS2  DB 175," 3 navios do tipo destroier (1 quadrado) ",174,"$"
    NAVIOS3  DB 175," 1 navios do tipo fragata (2 quadrados) ",174,"$"
    NAVIOS4  DB 175," 1 navio do tipo encouracado (3 quadrados) ",174,"$"
    NAVIOS5  DB "PRESSIONE ENTER PARA CONTINUAR$"
    NOME_JOGADOR1 DB "Jogador 1, informe seu nome: $"
    NOME_JOGADOR2 DB "Jogador 2, informe seu nome: $"
    EXEMPLO1 DB ",escolha as posicoes dos seus navios(Letra + Numero)$"
    EXEMPLO2 DB " Exemplo: A4$"
    PEDE_POSICAO_DESTROIER    DB "Qual a posicao do destroier: $"
    PEDE_POSICAO_FRAGATA      DB "Quais as posicoes do fragata: $"
    PEDE_POSICAO_ENCOURACADO DB "Quais as posicoes do encouracado: $"
    RELATORIO_JOGADOR_TIROS     DB "Tiros restantes: $"
    RELATORIO_JOGADOR_ACERTOS   DB "Acertos: $"
    RELATORIO_JOGADOR_POS       DB "Posicao do ultimo tiro: $"
    JOGADOR1 DB "NOME 1, qual a posicao do tiro: $"
    JOGADOR2 DB ", qual a posicao do tiro: $"
    ACERTO DB "Parabens, voce atingiu um navio inimigo! $"
    ERRO   DB "Opss, voce errou o tiro! $"
    PRINT_ACERTO DB "X$"
    PRINT_ERRO DB "~$"
    POSICAO_ESCOLHIDA DB "Voce ja escolheu essa posicao,tente outra$"
    POSICAO_REPITIDA DB "Voce ja atirou nessa posicao, tente novamente! $"
    POSICAO_INVALIDA DB "Posicao invalida, tente novamente$"
    TIRO_INVALIDO DB "Tiro invalido, tente novamente $"
    ENCERRAMENTO DB "PARABENS $"
                 DB "Voce acertou todas as embarcacoes inimigas e ganho o jogo!$"


NOME1 DB 10 DUP (?)
NOME2 DB 10 DUP (?)


COLUNAS DB " [1] [2] [3] [4] [5] [6] $"
DIV_1   DB 218,196,196,196,194,196,196,196,194,196,196,196,194,196,196,196,194,196,196,196,194,196,196,196,191,"$"
DIV_2   DB 192,196,196,196,193,196,196,196,193,196,196,196,193,196,196,196,193,196,196,196,193,196,196,196,217,10,13,"$"
DIV_3   DB 195,196,196,196,197,196,196,196,197,196,196,196,197,196,196,196,197,196,196,196,197,196,196,196,180,10,13,"$"

VEZ DB "Vez do(a): $"
TIRO DB "Tiro: $"

MATRIZ1 DB 36 DUP (0)
MATRIZ2 DB 36 DUP (0)
MATRIZ_AUXILIAR1 DB 36 DUP(0)
MATRIZ_AUXILIAR2 DB 36 DUP(0)

INVALIDO DB 0
EMUSO DB 0
EMUSO2 DB 0
POSICAO1 DB ?
POSICAO2 DB ?
LETRA DW ?
NUMERO DW ?
CONTADOR DW 0

CONT_TIROS_RESTANTES DB 36
CONT_TIROS_RESTANTES2 DB 36
CONT_ACERTOS DB 0
CONT_ACERTOS2 DB 0
ULTIMAPOS DW 0
ULTIMAPOS2 DW 0
```

Nesse tópico iremos apresentar e explicar os procedimentos e macros usadas no programa:

- **Leitura do nome**: Função que irá fazer a leitura do nome dos jogadores.

```
LEITURA_NOME PROC
      PRIMEIRO:
            MOV AH,1
            INT 21H

            CMP AL,0DH
            JE FINALIZA

            MOV [SI],AL

            INC SI
            JMP PRIMEIRO
      FINALIZA:
            MOV AL, '$'
            MOV [SI],AL

            RET
LEITURA_NOME ENDP
```

- **Printa Matriz**: Função que printa matriz na tela assim como as letras ao lado referente as linhas e os números acima referente as colunas.

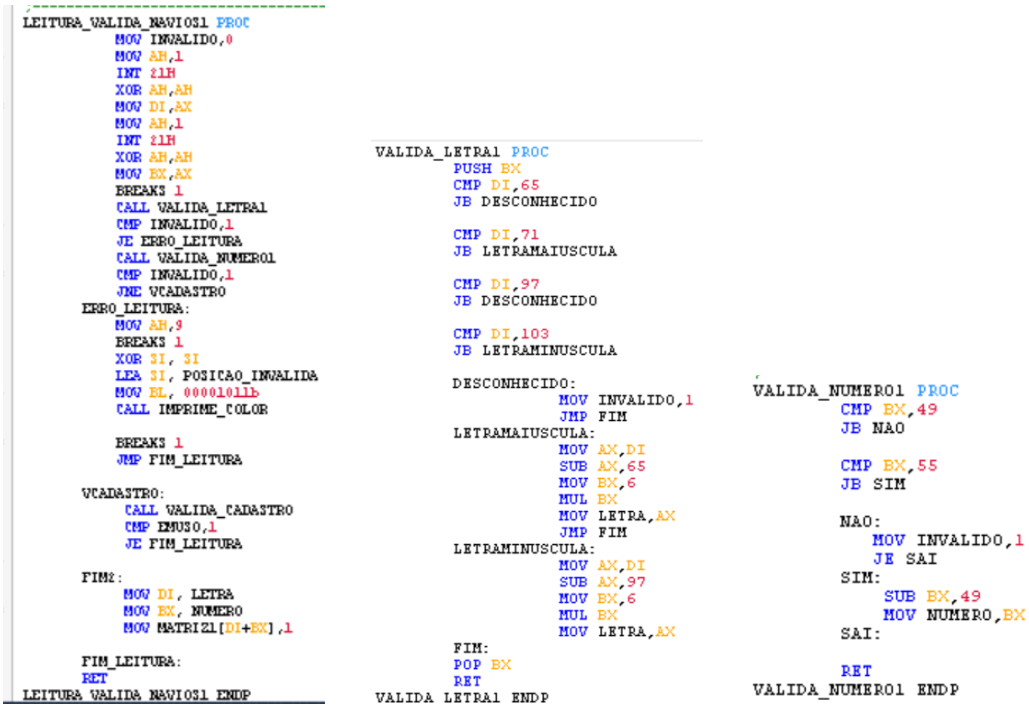
```
PRINTA_MATRIZ PROC

    MOV AH, 2
    MOV BH, 0
    XOR DX, DX
    INT 10h
    TAB 45
    MOVE_Y 4
    LEA SI,COLUNAS
    MOV BL,00001011b
    CALL IMPRIME_COLOR
    INT 21H
    BREAKS 2
    TAB 15
    GOTOXY 5, 45
    MOV AH,9
    MOV CX,1000
    MOV AL,' '
    MOV BH,0
    MOV BL,00001001b
    INT 10H
    MOV AH,9
    LEA DX,DIV_1
    INT 21H
    XOR DI,DI
    MOV CL,6
    LACOA:
        XOR BX,BX
        MOV CH,6
        MOV AH,9
        LEA DX,ESPACE
        INT 21H
        TAB 45
        CMP CL,6
        JE LACOB
        LEA DX,DIV_3
        INT 21H
        TAB 45
        LACOB:
            MOV AH,2
            MOV DL,179
            INT 21H
            TAB 45
        LACOB:
            MOV AH,2
            MOV DL,179
            INT 21H
            MOV DL,20H
            INT 21H
            MOV DL, MATRIZ1[DI+BX]
            ADD DL,30H
            INT 21H
            MOV DL,20H
            INT 21H
            INC BX
            DEC CH
            JNZ LACOB
            MOV DL,179
            INT 21H
            ADD DI,6
            DEC CL
            JNZ LACOA
            BREAKS 1
            TAB 45
        MOV AH,9
        LEA DX,DIV_2
        INT 21H
        GOTOXY 6,42
        MOV AL, 65
        RPT:
            MOV AH, 9
            MOV BL,00001011b
            MOV CX, 1
            INT 10h
            MOVE_Y 2
            INC AL
            CMP AL, 71
            JNE RPT
        RET
PRINTA_MATRIZ ENDP
```

- **Printa matriz auxiliar**: Função referente a matriz auxiliar que aparece na chamada dos tiros o qual é comparada com a matriz para verificar acertos e erros dos tiros.

```
PRINTA_MATRIZ_AUX1 PROC
    MOV AH, 2
    MOV BH, 0
    XOR DX, DX
    INT 10h
    TAB 45
    MOVE_Y 4
    LEA SI,COLUNAS
    MOV BL,00001101b
    CALL IMPRIME_COLOR
    INT 21H
    BREAKS 2
    TAB 15
    GOTOXY 5, 45
    MOV AH,9
    MOV CX,1000
    MOV AL,' '
    MOV BH,0
    MOV BL,00001100b
    INT 10H
    MOV AH,9
    LEA DX,DIV_1
    INT 21H
    XOR DI,DI
    MOV CL,6
    LACOC:
        XOR BX,BX
        MOV CH,6
        MOV AH,9
        LEA DX,ESPACE
        INT 21H
        TAB 45
        CMP CL,6
        JE LACOD
        LEA DX,DIV_3
        INT 21H
        TAB 45
        LACOD:
            MOV AH,2
            MOV DL,179
            INT 21H
            MOV DL,20H
            INT 21H
            MOV DL, MATRIZ_AUXILIAR1[DI+BX]
            ADD DL,''
            INT 21H
            MOV DL,20H
            INT 21H
            INC BX
            DEC CH
            JNZ LACOD
            MOV DL,179
            INT 21H
    ADD DI,6
    DEC CL
    JNZ LACOC
    BREAKS 1
    TAB 45
    MOV AH,9
    LEA DX,DIV_2
    INT 21H
    GOTOXY 6,42
    MOV AL, 65
    AGAIN:
        MOV AH, 9
        MOV BL,00001101b
        MOV CX, 1
        INT 10h
        MOVE_Y 2
        INC AL
        CMP AL, 71
        JNE AGAIN
        RET
PRINTA_MATRIZ_AUX1 ENDP
```

- **Leitura para validação dos navios**: Função que irá fazer a leitura das posições que os jogadores escolherão para seus tabuleiros ao mesmo tempo que reconhece se a letra e número da posição é valida ou não. Na parte de validação da letra e número foram feitos outros procedimentos para as validações específicas.



```
;-------------------------------
LEITURA_VALIDA_NAVIOS1 PROC
        MOV INVALIDO,0
        MOV AH,1
        INT 21H
        XOR AH,AH
        MOV DI,AX
        MOV AH,1
        INT 21H
        XOR AH,AH
        MOV BX,AX
        BREAKS 1
        CALL VALIDA_LETRA1
        CMP INVALIDO,1
        JE ERRO_LEITURA
        CALL VALIDA_NUMERO1
        CMP INVALIDO,1
        JNE VCADASTRO
    ERRO_LEITURA:
        MOV AH,9
        BREAKS 1
        XOR SI, SI
        LEA SI, POSICAO_INVALIDA
        MOV BL, 00001011b
        CALL IMPRIME_COLOR

        BREAKS 1
        JMP FIM_LEITURA

    VCADASTRO:
         CALL VALIDA_CADASTRO
         CMP EMUSO,1
         JE FIM_LEITURA

    FIM2:
         MOV DI, LETRA
         MOV BX, NUMERO
         MOV MATRIZ1[DI+BX],1

    FIM_LEITURA:
    RET
LEITURA_VALIDA_NAVIOS1 ENDP
```

```
VALIDA_LETRA1 PROC
        PUSH BX
        CMP DI,65
        JB DESCONHECIDO

        CMP DI,71
        JB LETRAMAIUSCULA

        CMP DI,97
        JB DESCONHECIDO

        CMP DI,103
        JB LETRAMINUSCULA

        DESCONHECIDO:
                MOV INVALIDO,1
                JMP FIM
        LETRAMAIUSCULA:
                MOV AX,DI
                SUB AX,65
                MOV BX,6
                MUL BX
                MOV LETRA,AX
                JMP FIM
        LETRAMINUSCULA:
                MOV AX,DI
                SUB AX,97
                MOV BX,6
                MUL BX
                MOV LETRA,AX
        FIM:
        POP BX
        RET
VALIDA_LETRA1 ENDP
```

```
;
VALIDA_NUMERO1 PROC
        CMP BX,49
        JB NAO

        CMP BX,55
        JB SIM

        NAO:
           MOV INVALIDO,1
           JE SAI
        SIM:
            SUB BX,49
            MOV NUMERO,BX
        SAI:

        RET
VALIDA_NUMERO1 ENDP
```

- **Salva as posições escolhidas pelos jogadores**: Nessa função as letras e números são passadas para registradores, salvando-as. Desta forma, ele consegue reconhecer se aquela posição já está sendo ocupada ou não.

```
;--------------------------------------
VALIDA_CADASTRO PROC
        MOV EMUSO,0

        MOV DI, LETRA
        MOV BX, NUMERO

        CMP NUMERO, 5
        JA FIM_VALIDA

        CMP MATRIZ1[DI + BX],1
        JE  JA_FOI

        JMP FIM_VALIDA
        JA_FOI:
              MOV EMUSO,1
              MOV AH,9
              LEA DX,POSICAO_ESCOLHIDA
              INT 21H
              BREAKS 1
        FIM_VALIDA:
            RET
VALIDA_CADASTRO ENDP
```

- **Leitura do tiro**: Função que lê a letra e número do tiro dado pelos jogadores afim de acertar embarcações inimigas e com isso reconhece se a posição existe de fato, se nela tem ou não embarcação e se aquela posição já foi atirada.

```
LEITURA_TIRO1 PROC
    VOLTEIAQUI:
            MOV INVALIDO,0
            MOV AH,1
            INT 21H
            XOR AH,AH
            MOV DI,AX
            MOV AH,1
            INT 21H
            XOR AH,AH
            MOV BX,AX
            CALL VALIDA_LETRA1
            CALL VALIDA_NUMERO1
            CALL VALIDA_TIROS1
            CMP JA_ACERTADO,1
            JNE NAO_TENTE_NOVAMENTE
            MOV AH,9
            BREAKS 1
            XOR SI, SI
            LEA SI, TIRO_INVALIDO
            MOV BL, 00001011b
            CALL IMPRIME_COLOR
            BREAKS 1
            MOV AH,9
            LEA DX,TIRO
            INT 21H
            JMP VOLTEIAQUI
            NAO_TENTE_NOVAMENTE:
            CMP INVALIDO,0
            JE FIM4
            MOV AH,9
            BREAKS 1
            XOR SI, SI
            LEA SI, TIRO_INVALIDO
            MOV BL, 00001011b
            CMP JA_ACERTADO,1
            JNE NAO_TENTE_NOVAMENTE
            MOV AH,9
            BREAKS 1
            XOR SI, SI
            LEA SI, TIRO_INVALIDO
            MOV BL, 00001011b
            CALL IMPRIME_COLOR
            BREAKS 1
            MOV AH,9
            LEA DX,TIRO
            INT 21H
            JMP VOLTEIAQUI
            NAO_TENTE_NOVAMENTE:
            CMP INVALIDO,0
            JE FIM4
            MOV AH,9
            BREAKS 1
            XOR SI, SI
            LEA SI, TIRO_INVALIDO
            MOV BL, 00001011b
            CALL IMPRIME_COLOR
            JMP VOLTEIAQUI
    FIM4:
            MOV DI, LETRA
            MOV BX, NUMERO
            DEC CONT_TIROS_RESTANTES
            CMP LETRA, 5
            JA FIM_TUDO
            CMP NUMERO, 5
            JA FIM_TUDO
FIM_TUDO:
RET
LEITURA_TIRO1 ENDP
```

- **Validação dos tiros:** Função que reconhece se o tiro foi acertado ou não e com isso ele imprime na tela os textos referentes aos acertos ou erros. Nela também é feito a movimentação dos símbolos: (X) acertou e (~) errou nas posições da matriz auxiliar.

```
;=============================================
VALIDA_TIROS1 PROC
        MOV EMUSO2,0
        MOV JA_ACERTADO,0
        MOV DI, LETRA
        MOV BX, NUMERO
        CMP NUMERO, 5
        JA FIM_GERAL
        CMP MATRIZ_AUXILIAR2[DI+BX],88
        JE DESTRUIDO
        CMP MATRIZ2[DI + BX],1
        JE  ACERTOS

        JMP FIM_TIROS

        ACERTOS:
              MOV EMUSO2,1
              MOV MATRIZ_AUXILIAR2[DI+BX],88
              MOV AH,9
              XOR DH,DH
              LEA DX,ACERTO
              INC CONT_ACERTOS
              BREAKS 1
              INT 21H
              RET

        DESTRUIDO:
              MOV JA_ACERTADO,1
              RET

        FIM_TIROS:
             MOV MATRIZ_AUXILIAR2[DI+BX],126
             MOV AH,9
             XOR DH,DH
             LEA DX,ERRO
             BREAKS 1
             INT 21H
        FIM_GERAL:
            RET
VALIDA_TIROS1 ENDP
```

- **Printa relatório:** Função que printa na tela o relatório dos tiros dos jogadores.

```
PRINTA_RELATORIO1 PROC

            TAB 3
            MOV AH,9
            LEA DX,NOME1
            INT 21H
            BREAKS 1
            TAB 5
            MOV AH,9
            LEA DX,RELATORIO_JOGADOR_TIROS
            INT 21H
            CALL IMPRIME_RESTO
            BREAKS 1
            TAB 5
            MOV AH,9
            LEA DX,RELATORIO_JOGADOR_ACERTOS
            INT 21H
            MOV AH,2
            MOV DL,CONT_ACERTOS
            ADD DL,48
            INT 21H

            BREAKS 1
            TAB 5

            MOV AH,9
            LEA DX,RELATORIO_JOGADOR_POS
            INT 21H

            MOV AH,2
            MOV DX,ULTIMAPOS
            INT 21H

            MOV AH,2
            MOV DX,ULTIMAPOS2
            INT 21H

            RET
PRINTA_RELATORIO1 ENDP
```

- **Informa a vez do jogador:** chama funções de leitura/verificação de forma alternada.

```
ATIRAR_JOG1 PROC

    GOTOXY 0,0
    MOV AH,9
    LEA DX,VEZ
    INT 21H

    MOV AH,9
    LEA DX,NOME1
    INT 21H

    CALL PRINTA_MATRIZ_AUX2

    GOTOXY 13,2
    CALL PRINTA_RELATORIO1

    GOTOXY 7,2
    MOV AH,9
    LEA DX,TIRO
    INT 21H

    CALL LEITURA_TIRO1
    MOV AH,1
    INT 21H

    RET
ATIRAR_JOG1 ENDP
```

- **Imprime resto**: Função que realiza a divisão dos tiros restantes e salva o quociente e o resto para que possa ser feita a impressão de ambos os caracteres.

```
IMPRIME_RESTO PROC
        PUSH AX
        PUSH BX
        PUSH DX

        XOR AX,AX
        MOV AL,CONT_TIROS_RESTANTES
        MOV BL,10
        DIV BL

        MOV BH, AH

        MOV AH, 2
        MOV DL, AL
        ADD DL,48
        INT 21h

        MOV DL, BH
        ADD DL,48
        INT 21h

        POP DX
        POP BX
        POP AX
        RET
IMPRIME_RESTO ENDP
```

- **Fim do jogo**: Função que faz as impressões dos textos referentes ao final do jogo.

```
ENDGAME1 PROC

        BREAKS 5
        TAB 36

        LEA SI,NOME1
        MOV BL,00000010b
        CALL IMPRIME_COLOR

        BREAKS 2
        TAB 35

        LEA SI,ENCERRAMENTO
        MOV BL,00000010b
        CALL IMPRIME_COLOR

        BREAKS 2
        TAB 10

        LEA SI,ENCERRAMENTO2
        MOV BL,00000010b
        CALL IMPRIME_COLOR
        INT 21H

        RET
ENDGAME1 ENDP
```

Macros:

- Macro para movimentação do eixo y na tela de saída:

```
MOVE_Y MACRO N
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX

    MOV AH, 3
    MOV BH, 0
    INT 10h

    MOV AH, 2
    ADD DH, N
    INT 10H

    POP DX
    POP CX
    POP BX
    POP AX
ENDM
```

- Macro para movimentação dos eixos x e y simultaneamente:

```
GOTOXY MACRO X, Y
    PUSH AX
    PUSH BX
    PUSH DX

    MOV AH, 2
    MOV BH, 0
    MOV DH, X
    MOV DL, Y
    INT 10h

    POP DX
    POP BX
    POP AX
ENDM
```

- Macro para quebrar linhas:

```
BREAKS MACRO N
     PUSH AX
     PUSH BX
     PUSH CX
     PUSH DX

     MOV AH, 3
     MOV BH, 0
     INT 10h

     MOV AH, 2
     ADD DH, N
     XOR DL, DL
     INT 10H

     POP DX
     POP CX
     POP BX
     POP AX
ENDM
```

- Macro para dar TAB (dar espaço na lateral):

```
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
```

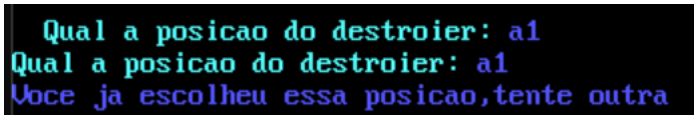
Capítulo 3

Resultados

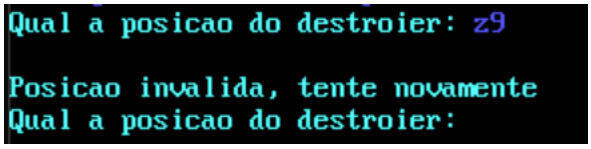
3.1 Descrição dos testes realizados

O primeiro teste que foi preciso realizar no programa foi o de leitura das posições dos jogadores

Se a posição escolhida foi repetida:



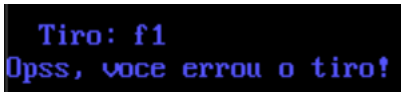
Se a posição escolhida está fora do alcance da matriz:



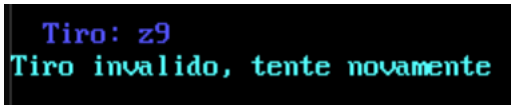
O Segundo teste foi referente a validação dos tiros dos jogadores

Se a posição atirada tem embarcação:

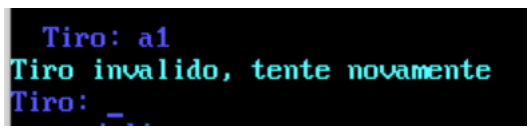
Se a posição atirada não tem embarcação:



Se a posição não existe no tabuleiro:



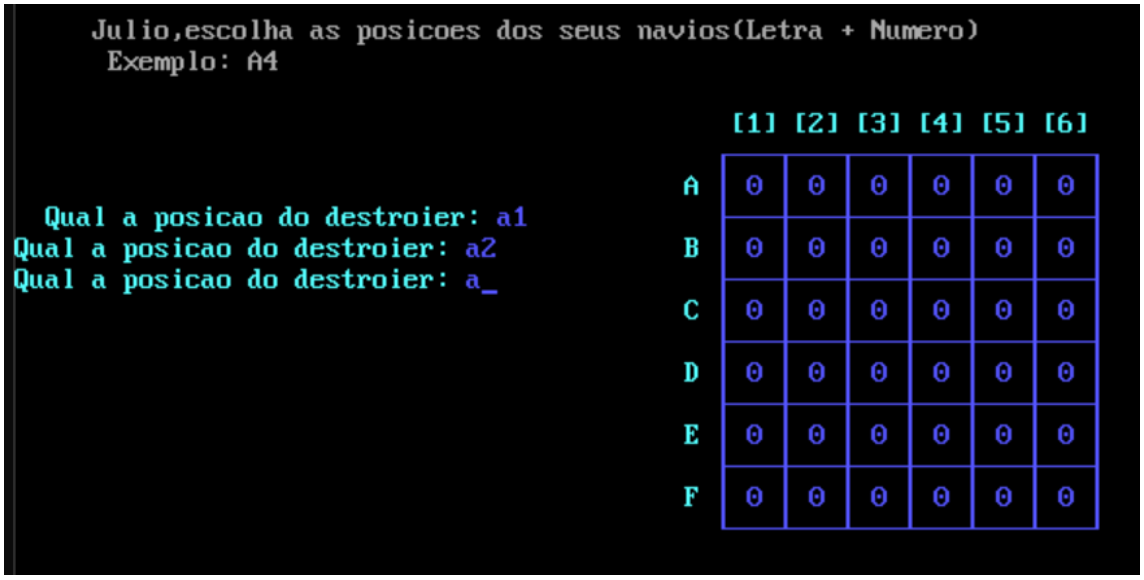
Se o tiro foi repetido em alguma posição já digitada:



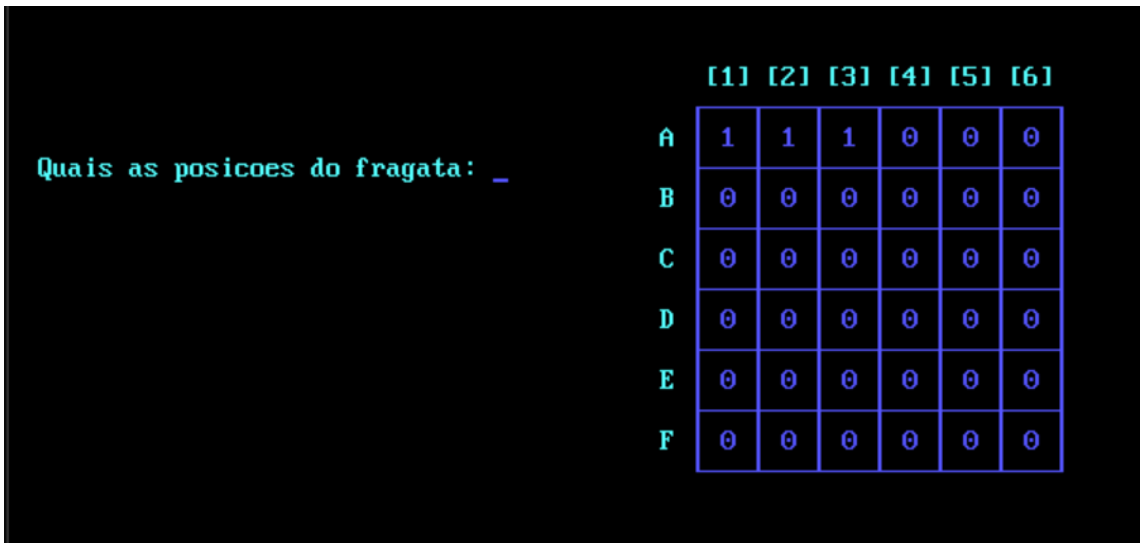
3.2 Resultados e Discussões

Resultados do jogo:

Nesta tela o jogo pede para o usuário digitar os destroiers que deseja colocar em seu tabuleiro

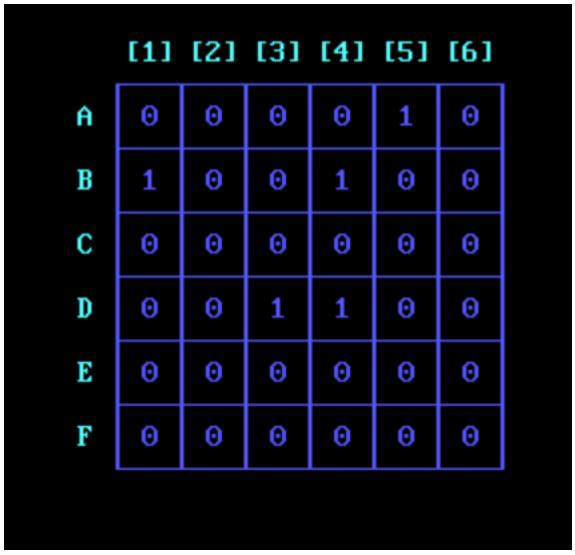


Depois que o jogador escolhe as posições dos destroiers a matriz auxiliar é preenchida e inicia os pedidos do fragata e assim sucessivamente até o encouraçado.



Este é um exemplo depois que todos os destroiers e fragatas estão posicionados

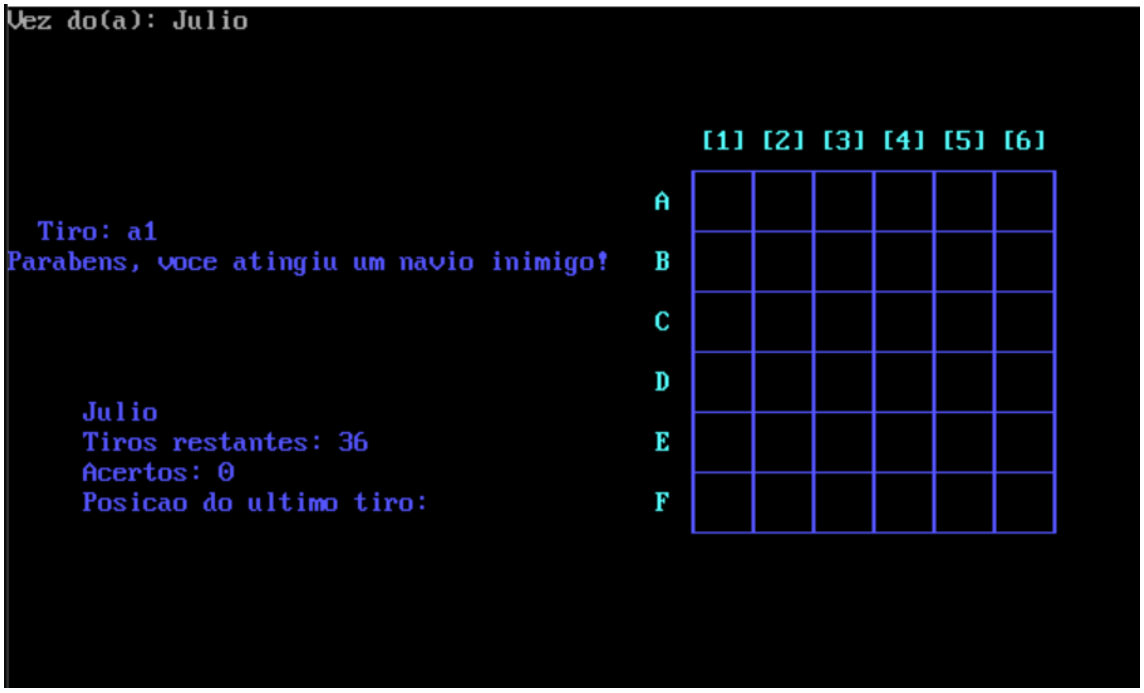
Foi feita a troca de zero por um nas posições selecionadas



Após as posições serem selecionadas, o jogo começa.

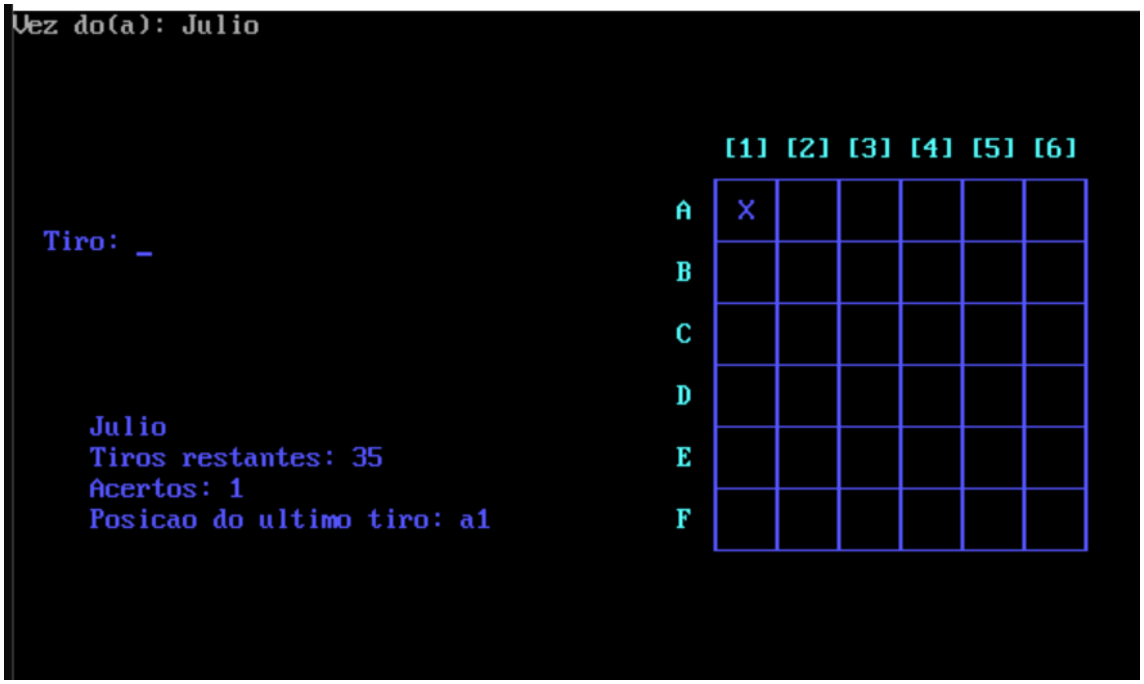
Nesta tela do início do jogo já percebemos que a posição que o jogador 1 escolheu para tentar acertar a embarcação está correta, logo a mensagem abaixo do tiro é mostrada.

Detalhe para o relatório dos tiros no canto inferior esquerdo. Ao decorrer das imagens ela irá mudar em relação aos tiros.



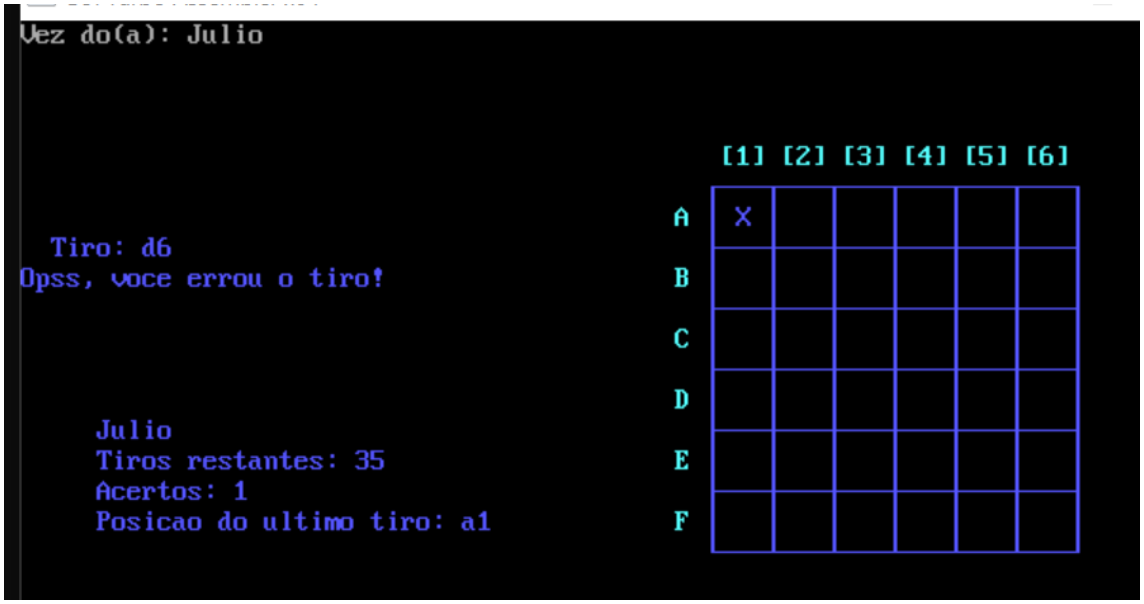
Esta tela refere-se à segunda jogada do jogador 1 após acertar a posição A1 da jogada anterior.

Repare no ‘X’ da matriz auxiliar mostrando o acerto e as mudanças do relatório.

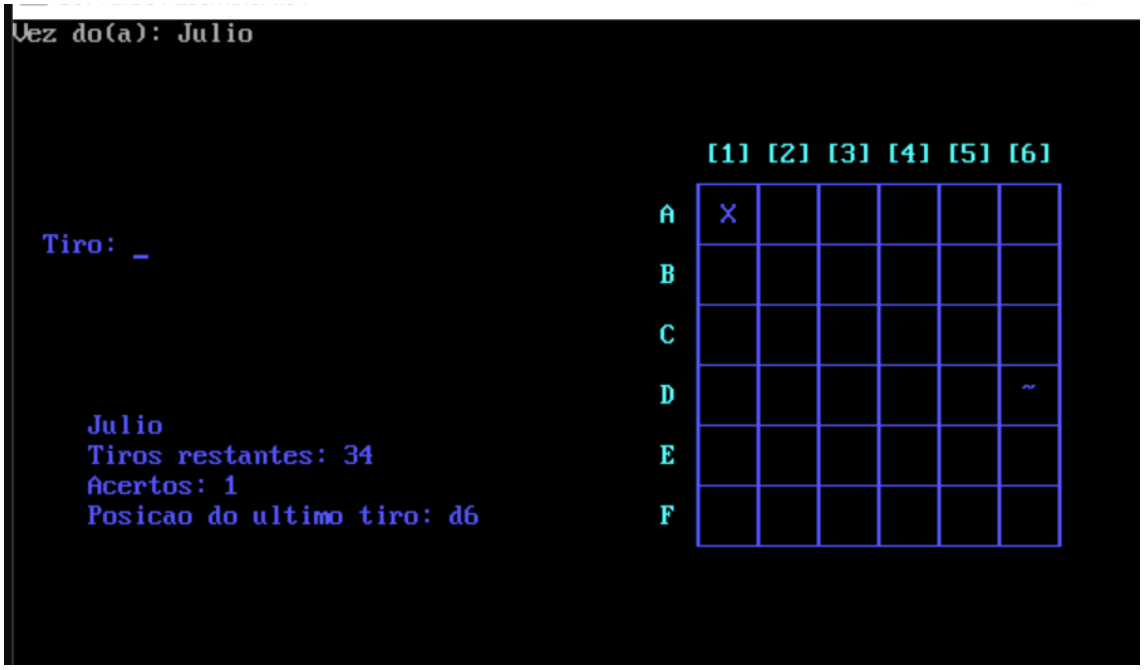


Agora iremos ver o que acontece se ocorrer o erro do tiro:

Mensagem de erro abaixo do tiro

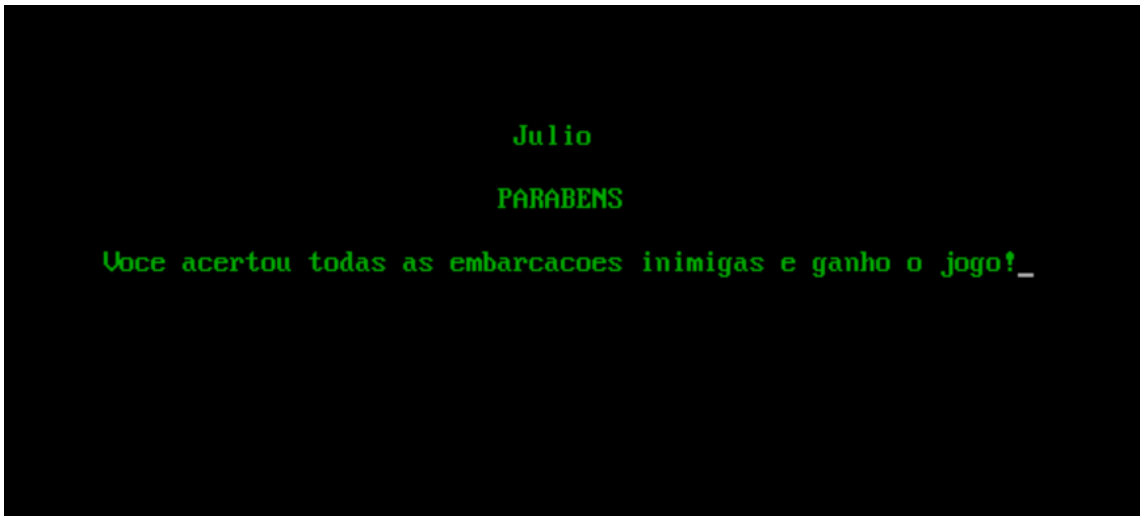


Como pode ser visto na posição D6 por conta do acerto na água foi impresso na posição o ‘~’.



Essa é referente a tela do vencedor:

Ela é apenas chamada caso um dos jogadores conseguir acertar todas as posições inimigas



Discussões e conclusão:

Com esse projeto nos deparamos com vários problemas nas estruturações do código em diferentes partes, porém conseguimos sanar todos os problemas a partir de pesquisas, revisões de aula, ajuda de colegas e o Debugger.

Concluímos que um projeto grande como esse é preciso ter muita paciência, calma e uma boa organização para que o desenvolvimento do projeto saia como desejado. Na hora de seguir em frente com todos os códigos foi preciso o cuidado dos integrantes para que as estruturações presentes não afetassem o resto do desenvolvimento. Junto com esse cuidado, foram realizados os testes vistos acima no relatório, para que nós pudéssemos solucionar ou prevenir conflitos entre trechos de códigos solucionados.

Além de sabedoria nas partes técnicas da linguagem utilizada foi necessário um bom raciocínio lógico para que houvesse uma fluidez entre os seguimentos de códigos. E um ponto positivo foi que por conta de ser um projeto complexo, conseguimos ampliar nosso conhecimento na linguagem Assembly 8086.

Capítulo 4

Referências Bibliográficas

Links e Aulas:

- Organização Básica de Computadores e Linguagem de Montagem, Ricardo Pannain (Aulas)

- https://theasciicode.com.ar/extended-ascii-code/box-drawings-double-line-horizontal-vertical-character-ascii-code-206.html

- https://stackoverflow.com/questions/28665528/while-do-while-for-loops-in-assembly-language-emu8086

- http://unixwiz.net/techtips/x86-jumps.html

- http://spike.scu.edu.au/~barry/interrupts.html