Título: Relatório Técnico - Implementação do Jogo Batalha Naval em Assembly x86

Introdução: Este documento apresenta o desenvolvimento do código de um jogo de Batalha Naval em Assembly para o processador 8086. O jogo foi estruturado em uma matriz 10x10 onde o jogador tenta localizar embarcações escondidas. O objetivo principal foi desenvolver um programa funcional e otimizado, aproveitando as funcionalidades do Assembly.

Estrutura Geral do Código: O código foi dividido em seções principais:

1. **Segmento de Dados (`.DATA`)**:
    - Contém as variáveis e estruturas de dados necessárias para o jogo, como a matriz do tabuleiro (`TABULEIRO`) e a matriz de tiros (`TIROS`).
    - Mensagens de instrução e feedback para o usuário.
2. **Segmento de Código (`.CODE`)**:
    - Inicia a execução com a configuração do segmento de dados e controle do loop principal do jogo.
    - Contém as principais rotinas, incluindo:
        - **Entrada de Dados**: Recebe a linha e a coluna do usuário para efetuar um "tiro".
        - **Cálculo do Endereço da Célula**: Utiliza registros `SI` e `BX` para navegar pelo tabuleiro com acesso direto à matriz como `TABULEIRO[SI+BX]`.
        - **Verificação de Acerto**: Verifica se a célula contém uma embarcação (valor `1`).
        - **Marcação de Resultados**: Atualiza a matriz de tiros com `*` para acertos e `x` para erros.
    - **Rotina de Impressão**: Imprime o tabuleiro de tiros atualizado a cada jogada.

Detalhamento das Funções e Rotinas:

1. **Configuração Inicial**:
    - `MOV AX, @DATA`, seguido por `MOV DS, AX` e `MOV ES, AX` para definir os segmentos de dados e pilha do programa.
2. **Loop do Jogo (`LOOP_JOGO`)**:
    - Este é o ciclo principal onde o jogador insere suas jogadas.
    - Inclui chamadas para exibir o tabuleiro, solicitar linha e coluna, e atualizar o status de jogo.
    - **Entrada de Linha e Coluna**: Utiliza interrupções de DOS (`INT 21h`) para ler valores de entrada.
    - **Desistência**: Condição para encerrar o jogo com a tecla 'F' ou 'f'.
3. **Cálculo de Índice**:
    - O índice na matriz é acessado como `TABULEIRO[SI+BX]` onde:
        - `SI` representa a coluna (multiplicada por 10 para acessar a posição correta na linha).
        - `BX` representa a linha diretamente.

4. **Marcadores de Acertos e Erros**:
    - **Acerto**: Marca a célula com `*` e incrementa o contador de acertos `CONT_ACERTOS`.
    - **Erro**: Marca a célula com `x`.
    - Após 19 acertos, o programa finaliza com uma mensagem de vitória.
5. **Finalização do Jogo**:
    - Inclui uma mensagem de vitória ou de desistência conforme a condição final do jogo.

Considerações sobre Desempenho e Otimização:

- **Uso de Registros**: O código foi otimizado para minimizar operações extras, utilizando `SI` e `BX` para navegação direta pela matriz.
- **Interrupções**: As interrupções foram utilizadas para manipular entrada e saída de dados, facilitando a interação com o jogador e economizando recursos.

Conclusão: Este programa de Batalha Naval implementado em Assembly proporciona uma prática eficaz da manipulação de matrizes e controle de fluxo em Assembly x86. O uso de registros `SI` e `BX` como índices multidimensionais simplificou o código e permitiu uma navegação eficiente pela matriz.

Possíveis Melhorias:

- Incluir mensagens detalhadas ou indicadores visuais para cada tipo de embarcação acertada.