UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

PAULO EDUARDO RODRIGUES WERLE

JOGO DE BATALHA NAVAL

CHAPECÓ
2021

Problema

Esse trabalho foi realizado com o propósito de implementar o jogo 'Batalha Naval', onde duas pessoas tem o objetivo de adivinhar em quais espaços do tabuleiro adversário estão situados os seus navios, que podem apresentar tamanhos variados e estão espalhados pelo tabuleiro. O jogo conta com uma série de adivinhações e só termina quando um dos jogadores consegue destruir completamente todos os navios do adversário.

Na implementação temos o seguinte cenário: um jogador fará suas ações contra o computador, os navios estarão espalhados pelo tabuleiro, e o jogador tentará destruir todos na menor quantidade de ações possíveis. Quando conseguir adivinhar onde estão todas as partes dos navios e destruí-los, vence.

Solução

A implementação do jogo Batalha Naval foi feita por um conjunto de instruções do RISC-V. Vamos nos deparar com várias funções, cada uma com um papel na composição do jogo, sendo que eventualmente pode acontecer de algumas chamarem outras.

A seguir as funções criadas e um pouco sobre seu funcionamento:

1) **novoTabuleiro(a2)** – criação de um novo tabuleiro cada vez que o jogo é iniciado, ela recebe como parâmetro uma matriz de 10x10, que vai ser aonde vai haver os dados do jogo, como navios e tiros.

   Parâmetros:
   - **a2** → int board[BOARD_SIZE][BOARD_SIZE]

2) **inserirTabuleiro(a2, a4, a5, a6, a7, a3)** – inserção de dados em um tabuleiro, recebe como parâmetro o tabuleiro do jogo, para poder fazer a inserção nele, e informações sobre onde deve ser inserido o navio, temos parâmetros indicando qual é o ângulo do navio(vertical/horizontal), o tamanho do navio, qual vai ser a coluna e a linha que ele vai ser inserido na matriz, e qual é o navio que está sendo inserido

   Parâmetros:
   - **a2** → int board[BOARD_SIZE][BOARD_SIZE]
   - **a4** → int angle
   - **a5** → int length
   - **a6** → int row
   - **a7** → int col
   - **a3** → int current

3) **imprimeBarcos(a2)** – impressão da posição dos barcos na matriz, é recebido como parâmetro a matriz onde contém dados sobre os barcos.

Parâmetros:

- **a2** → int board[BOARD_SIZE][BOARD_SIZE]

4) **imprimeTiros(a2)** - mostra ao jogador onde foi feito tiros, e se seu tiro acertou ou errou o alvo, nesta função vamos ter diferentes tipos de impressões na tela para o usuário, caso na tela seja mostrado o símbolo(**ø)** em uma posição da matriz, significa que o tiro errou o alvo e acabou atingindo a água, caso seja mostrado o símbolo(o) significa que ainda não houve algum tiro nessa posição da matriz, e caso seja mostrado o símbolo(x) significa que acertou um navio.

   Parâmetros:

   - **a2** → int board[BOARD_SIZE][BOARD_SIZE]

5) **imprimeDados(a1, a2, a3, a4, a5, a6)** - Utilizado para poder mostrar aos jogadores dados sobre a partida, como quantos tiros foi feito, quantos tiros acertaram algum navio, e quantos navios foram afundados, também possui dados sobre record do jogo, que leva como base a menor quantidade de tiros para poder destruir todos os navios.

   Parâmetros:

   - **a1** → int shots
   - **a2** → int hits
   - **a3** → int sunk
   - **a4** → int shots_record
   - **a5** → int hits_record
   - **a6** → int sunk_record

6) **confereBarcos(a2, a5, a6)** - Utilizado para conferir se ao ser realizado um tiro, e esse ter acertado um alvo, é conferido se o navio foi totalmente destruído, utilizado também para verificar se ainda possui navios no tabuleiro

   Parâmetros:

   - **a2** → int board[BOARD_SIZE][BOARD_SIZE]
   - **a5** → int *sunk
   - **a6** → int ships

7) **validaInsercoes(a2, a4, a5, a6, a7, s11)** - Utilizado para validar se não há nenhum erro no processo de obter os dados informados e no processo de inserir os dados no tabuleiro, caso aconteça algum erro é retornado o erro

   Parâmetros:

   - **a2** → int board[BOARD_SIZE][BOARD_SIZE]
   - **a4** → int angle
   - **a5** → int length

- **a6** → int row
- **a7** → int col
- **s11** → int *error

Retorno:

- **s11** → int

8) **converterInteiro(s11, s2, s3)** - Utilizado para converter tipos de dados string para tipos de números inteiros, a função conta com um processo de validação para verificar se os números presentes apresentam apenas uma casa decimal, caso acontecer de um número ter duas casas decimais, isto é pertencer a casa de dezenas é emitido um erro que será retornado para que a função não seja executado novamente.

   Parâmetros:

   - **s11** → int *error
   - **s2** → char current
   - **s3** → char next

   Retorno:

   - **a0** → int

9) **insereEmbarcacoes(a2, s11, s9)** - Utilizado para inserir os navios no tabuleiro, pegando os dados propostos de cada navio e gerenciando sua inserção, também é responsável por avisar caso tenha dado algum erro ao pegar os dados para inseri los

   Parâmetros:

   - **a2** → int board[BOARD_SIZE][BOARD_SIZE]
   - **s11** → int *error
   - **s9** → int *quantity_ships

10) **novaJogada(a2, a3, a4, a5)** - gerenciar cada jogada que o jogador irá fazer, atualizando os dados necessários na matriz, e os dados que serão mostrados ao jogador a cada nova jogada.

    Parâmetros:

    - **a2** → int board[BOARD_SIZE][BOARD_SIZE]
    - **a3** → int *shots
    - **a4** → int *hits
    - **a5** → int *sunk

11) **menu()** - mostrar as opções que o jogador pode realizar no jogo, como fazer uma nova jogada, poder ver a matriz de navios, ou reiniciar o jogo caso ache necessário

    Retorno:

    - **a0** → int

Conclusões

Durante o processo de implementação algumas dificuldades foram encontradas; primeiramente em relação ao nível de abstração da linguagem, por se tratar de uma linguagem de baixo nível pela qual o desempenho foi muito inferior comparado a produções nas linguagens estudadas anteriormente.

Em contrapartida, o fato de lidar diretamente com registradores, que muitas vezes ao manipular dados de um registrador para outro, acontece de sobrescrever dados que ainda estão em uso ou são de necessidade no decorrer do programa, limitando algumas ações.

Por último, o processo de validações para que um navio seja somente inserido ao tabuleiro, se atender a certas exigências propostas. Posto isso, a manipulação de strings foi um ponto custoso na hora do desenvolvimento.

Assembly é uma linguagem de baixo nível, mas de excelência e apesar das inúmeras dificuldades no percurso de aprendizagem dessa linguagem, é muito importante ter conhecimento na programação e no uso de conjuntos de instruções em assembly, isso devido a complexidade com que são tratados os dados em diferentes tipos de arquiteturas de processadores.