**Universidade Federal de São Carlos - Bacharelado em Sistemas de Informação**

**Arquitetura de Computadores**

**Wiki do Trabalho Prático de Arquitetura de Computadores – Jogo da Batalha Naval**

Professor Ricardo Menotti

Amarildo Reis (482927)

Fabiano Sousa (482919)

Marcelo Barbosa (482811)

Marco Aurélio (482960)

Paulo Cesar (483028)

Rodolfo Souza (482862)

Wanderson Sampaio (482978)

**2013**

*Dedico este trabalho a Deus que me abençoou durante toda esta jornada, sendo Ele a razão de minhas vitórias e conquistas. Dedico também a minha família, que me apoiou quando precisei me ausentar em alguns momentos de dedicação ao estudo. (Marcelo Barbosa)*

*“É impossível para um homem aprender aquilo que ele acha que já sabe”.*

*Epíteto*

Sumário

[Lista de Siglas 5](#_Toc355639882)

[Lista de Figuras 6](#_Toc355639883)

[Resumo 7](#_Toc355639884)

[1. Objetivos do Projeto 8](#_Toc355639885)

[1.1 Conhecendo o Jogo 9](#_Toc355639886)

[2. Batalha Naval 1.0 em Assembly 12](#_Toc355639887)

[2.1 Como jogar a Batalha Naval 1.0 em Assembly 12](#_Toc355639888)

[2.2. Divisão de Trabalho 14](#_Toc355639889)

[3. Especificações do sistema da Batalha Naval 1.0 em Assembly e Conclusões finais. 15](#_Toc355639890)

# Lista de Siglas

AC – Arquitetura de Computadores

MARS - MIPS Assembler and Runtime Simulator

MIPS - Microprocessor without Interlocked Pipeline Stages (microprocessador sem estágios interligados de pipeline)

# Lista de Figuras

[Figura 1 - Painel de ajuda do MARS 11](#_Toc355639896)

[Figura 2 - Tela inicial do jogo da Batalha Naval 1.0 em Assembly 12](#_Toc355639897)

[Figura 3 - entrada de dados para declarar um tiro em uma posição do quadro naval 13](#_Toc355639898)

[Figura 4 - visualização do tiro certeiro no quadro naval 13](#_Toc355639899)

[Figura 5 - Quadro naval de resposta 14](#_Toc355639900)

[Figura 6 - maximização da tela de execução RUN I/O 15](#_Toc355639901)

# Resumo

Este artigo visa documentar e explicar de um modo geral todos os passos e métodos utilizados durante o desenvolvimento do jogo da batalha naval.

A ideia central do projeto em grupo proposto na disciplina de AC é desenvolver um jogo que oferecesse uma interatividade empolgante com o usuário, de maneira dual e primordial, visou ampliar os conhecimentos dos alunos em linguagem Assembly.

O que foi estabelecido como critério é que utilizássemos o Simulador do processador MIPS, MARS. Foi estabelecido também que o jogo possuísse um sistema de pontuação, para ter maior controle de ações durante o jogo.

Este trabalho traz a documentação do trabalho em grupo, desenvolvido pelos alunos do grupo denominado The Masters.

O jogo escolhido foi o jogo de Batalha Naval, pelo motivo de alguns integrantes jogarem este jogo quando crianças, e terem então uma certa afinidade com o mesmo.

O grupo The Masters, assim intitulado por seus integrantes, definiu como projeto de desenvolvimento, o jogo da batalha naval, sendo este um jogo perfeito em termos que atendem as necessidades e critérios do projeto.

Foram adquiridos novos conhecimentos em termos de instruções em formatos R, I e J, bem como no uso de syscalls e macros, que foram essenciais no desenvolvimento do jogo.

# 1. Objetivos do Projeto

Desenvolver um jogo em linguagem Assembly do processador MIPS por meio do simulador MARS e seus recursos.

O simulador MARS oferece uma gama de recursos que tornam a programação em Assembly possível de ser escrita e interpretada em tempo real, mesmo não sendo um IDE.

O jogo deve ter critérios lógicos para controle de pontuação, estruturas condicionais para avaliação dos eventos disparados pelo usuário.

O jogo deve saber indicar quando o usuário vence ou perde uma partida.

O jogo pode ser desenvolvido em modo gráfico ou textual, de modo que ambos os meios não interferem na qualidade de seu desenvolvimento, pois é a lógica computacional que irá compor e coordenar todos os eventos do jogo.

Um mesmo evento pode disparar outros eventos e assim sucessivamente, para que cada ação do jogo seja bem sucedida.

O que é válido em um jogo no mundo real deve ser válido para a sua versão computacional. Assim, é fundamental que o jogo virtual seja uma abstração fiel do jogo do mundo real.

A aquisição de novos conhecimentos em linguagem Assembly é uma consequência esperada, pois por meio desta linguagem, aplicações e até mesmo grandes softwares podem ser escritos, dado que todo sistema computacional, independente de sua linguagem de programação, é convertido em Assembly compatível com o processador da máquina e seu SO por meio da compilação ou interpretação.

Esperamos com este projeto auxiliar agora e no futuro, outros grupos com os conhecimentos que obtivemos durante seu desenvolvimento.

## 1.1 Conhecendo o Jogo

A batalha naval é um jogo de raciocínio lógico, baseado em um plano cartesiano com cada posição de sua matriz ocupada por um número distinto de elementos, que representam embarcações.

Cada embarcação tem uma característica pelo número de posições que ocupa na matriz. As embarcações conhecidas são:

* Submarino: 1 posição;
* Cruzador: 2 posições;
* Hidro Avião: 3 posições;
* Encouraçado: 4 posições;
* Porta-Aviões: 5 posições;

Batalha naval é um jogo de tabuleiro onde os jogadores precisam adivinhar onde os navios do oponente estão em uma grade.

Antigamente, era jogado somente utilizando papel e lápis, mas em 1931 a empresa Milton Bradley Company o comercializou.

O objetivo do jogo é simples, quem eliminar todos os barcos do adversário primeiro ganha, sendo que cada um escolhe uma coordenada por vez (por exemplo: A5, C6...).

O objetivo do jogo é afundar todos os navios do adversário. Para afundar uma embarcação, deve-se acertar com tiros, todas as suas posições ocupadas.

Um tiro é dado informando a coordenada em que se posiciona uma embarcação. Assim, uma posição é informada com a linha e a coluna do elemento, no formato (número, letra).

Quando um tiro é certeiro, ele é marcado no quadro naval, que é a matriz e cenário em que se passa o jogo.

Em uma partida real, o jogo só termina quando todas as embarcações foram a pique. Ou quando o navio principal do jogo (porta-aviões) foi afundado.

Este jogo é altamente recomendado para quem busca desenvolver raciocínio lógico e agilidade mental, pois testa a tomada de decisões de uma pessoal por meio da lógica. É também uma fonte de diversão que pode ser jogado inicialmente por dois jogadores.

**1.2**. **Razões pela qual escolhemos o jogo**

Escolhemos este jogo por apresentar alta taxa de compatibilidade com os critérios avaliativos do projeto.

O jogo em si é entre o usuário e o computador. O usuário tem 5 chances para afundar todos os navios do adversário. Caso acabem suas chances, o jogo é encerrado com a derrota do usuário.

Para vencer este jogo, o usuário deve encontrar e destruir todos os navios do adversário, sem exceção de embarcações, pois todos devem ser afundados.

Ao escolhermos este jogo, tivemos a intenção de criar um desafio para os usuários, pois é quase impossível afundar todos os navios apenas com 5 chances. Assim, o jogo contém duas opções de jogabilidade: posicionar navios no quadro naval e navios automaticamente posicionados.

A primeira opção permite a um usuário posicionar os navios no quadro naval conforme sua vontade. Porém o usuário deve fazer isso com o intuito de desafiar outra pessoa a afundar todos os navios. Assim, essa segunda pessoa não deve ver o usuário posicionar os navios, mas sim tentar adivinhar onde estão para afundá-los.

A segunda opção possibilita o preenchimento automático do quadro naval, com posições pré-definidas pelo próprio jogo.

**1.3. O que foi utilizado no desenvolvimento do jogo:**

Para facilitar o desenvolvimento do jogo utilizamos a abordagem top-down (pensar no funcionamento do jogo deste os últimos detalhes, até os detalhes principais).

Foram utilizadas macros (pequenas sub-rotinas ou procedimentos), para a manipulação da entrada e saída de dados.

Para melhorar a entrada de dados via teclado, foram limitados os tipos de dados que seriam utilizados, de modo coerente com a realidade do jogo, assim os dados permitidos são: do tipo inteiro, do tipo carácter e string (cadeia de caracteres).

A entrada de dados é feita pela chamada das macros (sub-rotinas) que recebem como passagem parâmetro por referência, um registrador do processador.

A escrita destes dados também é feita por meio das macros (sub-rotinas), para facilitar a sua utilização.

Uma label permite criar um endereço de memória que cumpre todo um conjunto de instruções abaixo de seu nome, assim estas foram muito utilizadas para laços de repetição em geral e desvios condicionais por meio das instruções beq (salte se os registradores forem iguais) bne (salte se os registradores forem diferentes).

Os vetores são todos do tipo "word", que permite o armazenamento de até 4bytes por posição, podem ser dados numéricos, ou até mesmos caracteres (caso seja caractere, apenas um caractere pode ser armazenado).

**1.4. Fonte de referência para ampliação de conhecimento em Assembly**

A fonte de referência para criar todo o jogo e ampliar os nossos conhecimentos em Assembly foi retirada do painel de ajuda do simulador MARS, de modo que toda instrução nova, syscalls e macros foram assimilados neste painel de ajuda. Através da figura 1, pode-se ver um verdadeiro referencial para a programação em Assembly:

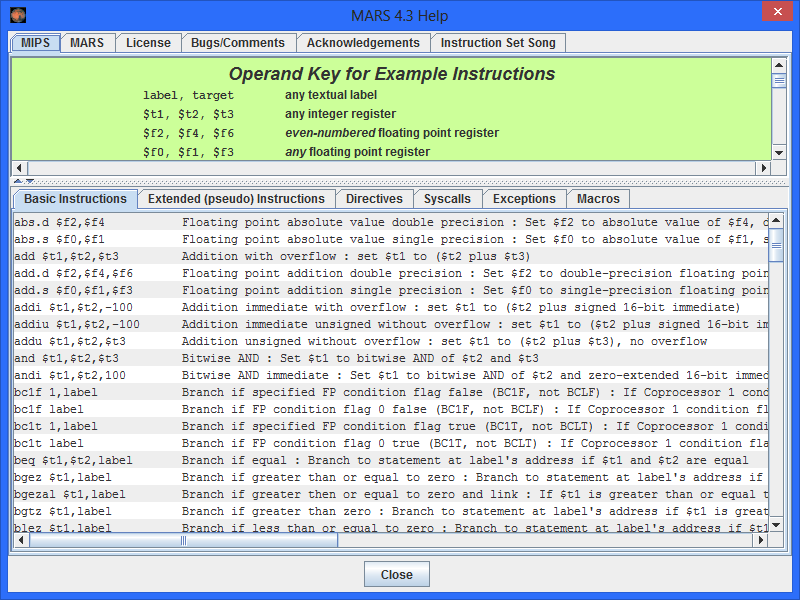


Figura 1 - Painel de ajuda do MARS

Perceba que o painel de ajuda do MARS oferece mais do que referencias para a utilização correta das instruções, pois ele permite a utilização de diretivas, syscalls e macros. Com base nesta ferramenta de consulta, foi possível converter toda a realidade do jogo da batalha naval em uma abstração coerente com o seu contexto e funcionalidade.

# 2. Batalha Naval 1.0 em Assembly

A Batalha Naval 1.0 em Assembly, assim intitulada, é uma abstração do jogo real, com eventos bem definidos e coerentes com o contexto do jogo original. A ideia central do jogo é afundar todos os navios. O usuário tem apenas 5 chances, o que torna o jogo um verdadeiro desafio.

Em uma partida cujo posicionamento dos navios é pré-definido, o usuário terá que acertar uma soma de 59 posições ocupadas no quadro naval, ou seja, até 11,8 embarcações por 5 tipos de embarcações.

## 2.1 Como jogar a Batalha Naval 1.0 em Assembly

Aqui descreveremos como jogar a Batalha Naval 1.0 em Assembly. A figura 2 ilustra perfeitamente

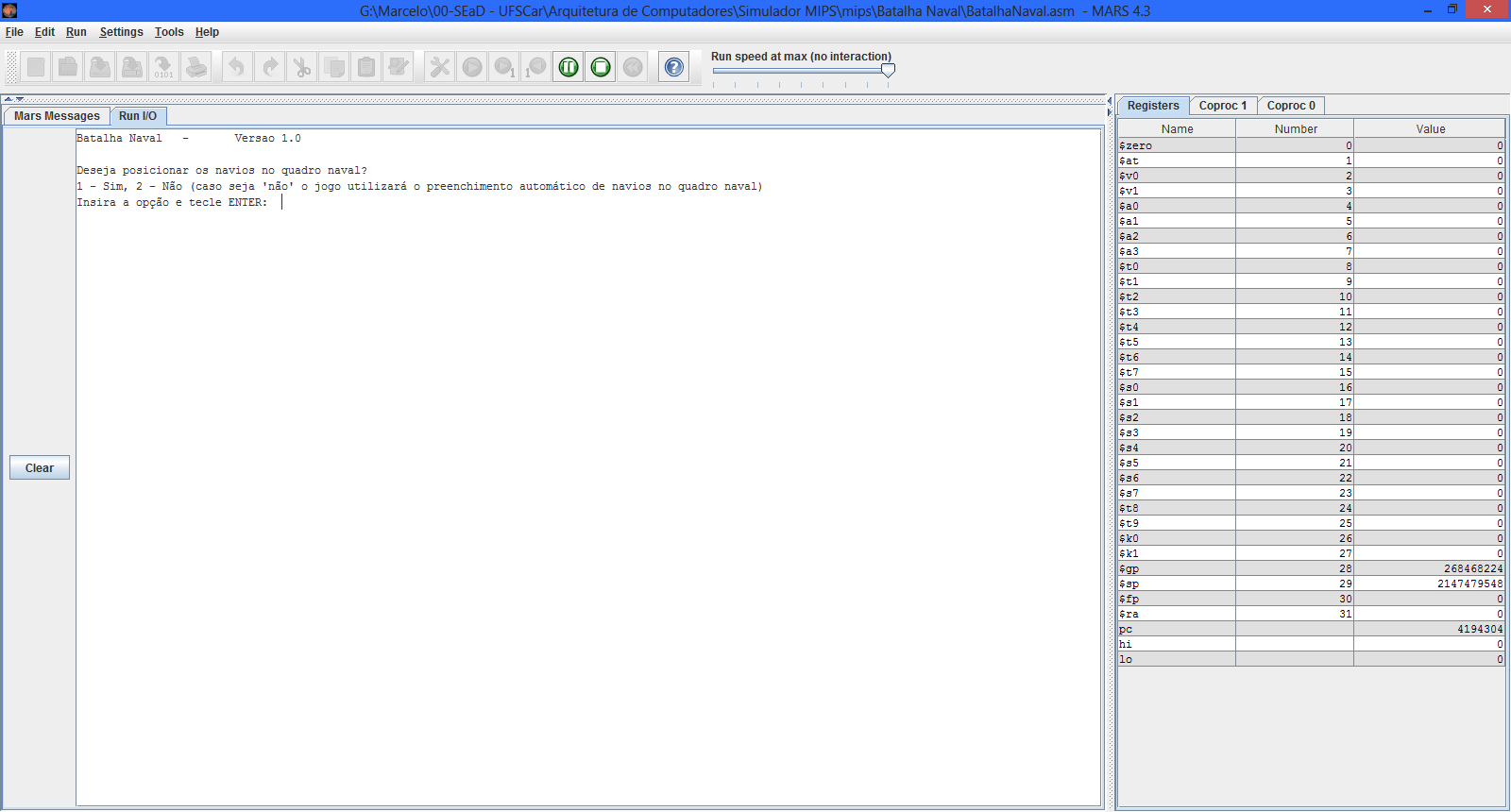


Figura 2 - Tela inicial do jogo da Batalha Naval 1.0 em Assembly

O usuário deve entrar com uma das opções (sim ou não, 1 ou 2 respectivamente), para posicionar os navios no quadro naval.

Caso a resposta seja 1 (sim), o usuário deve utilizar exclusivamente a letra “o” minúscula ou maiúscula para indicar uma embarcação no quadro naval. Para indicar uma posição vazia, deve-se pressionar a tecla “espaço” para definir esta posição como nula.

As embarcações são inseridas linha a linha, até a décima linha do quadro naval. Lembrando que se é um usuário que fixa estes navios, ele deve desafiar um oponente para tentar acertar as embarcações.

Caso a resposta seja 2 (não), os navios são posicionados automaticamente no quadro naval. Assim, o usuário pode jogar e tentar fazer o máximo de pontos que puder. Sempre tendo em mente, que possui apenas 5 chances para prosseguir com suas jogadas.

A figura 3 ilustra como é possível dar um tiro em uma posição do quadro naval, de modo a utilizar coordenadas com o formato (número da linha, letra da coluna), como mencionado anteriormente.

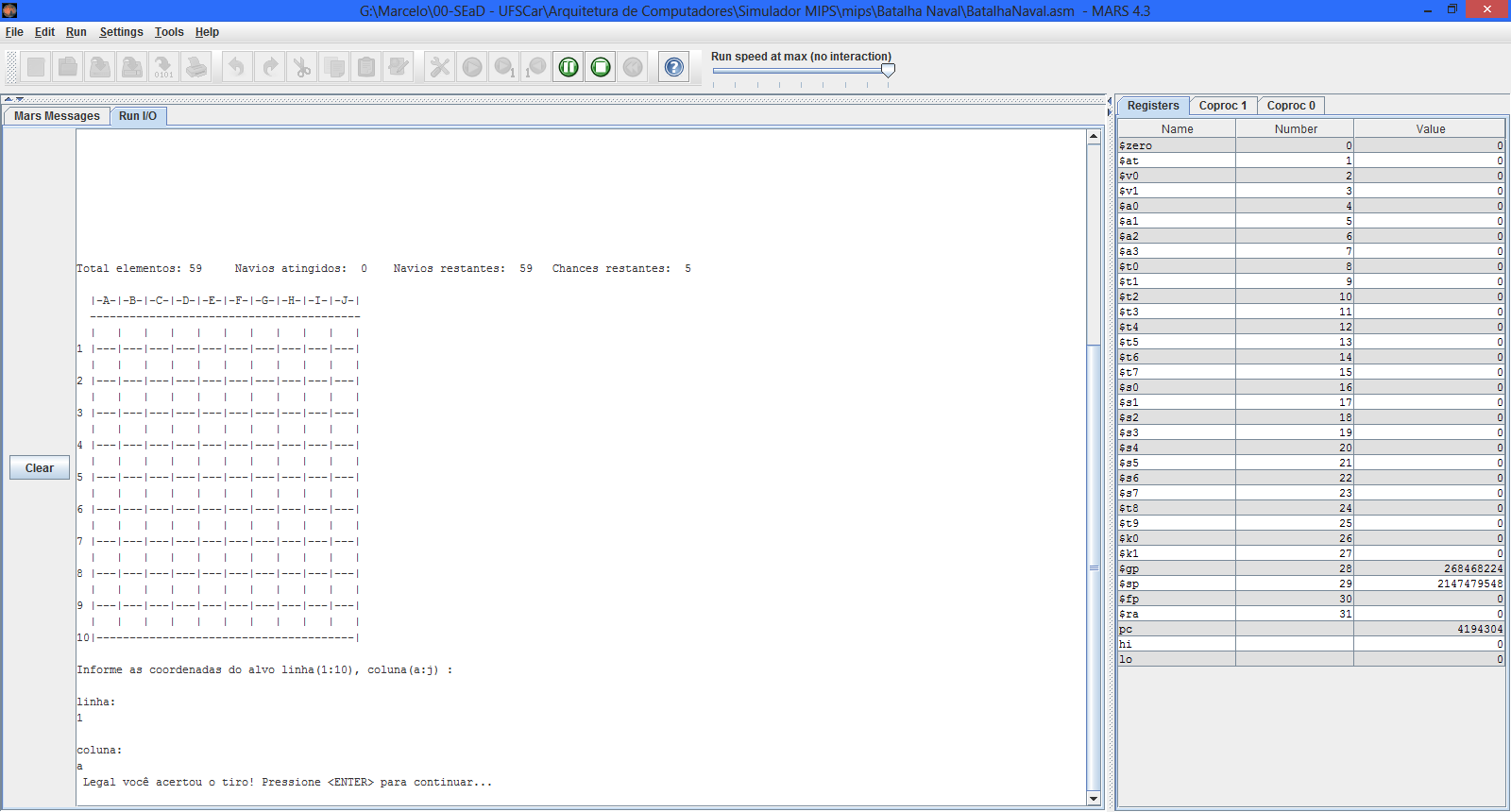


Figura 3 - entrada de dados para declarar um tiro em uma posição do quadro naval

Caso o usuário acerte o tiro, o quadro naval indica o tiro acertado colocando o caractere “O” no quadro naval, assim como pode-se notar na figura 4:

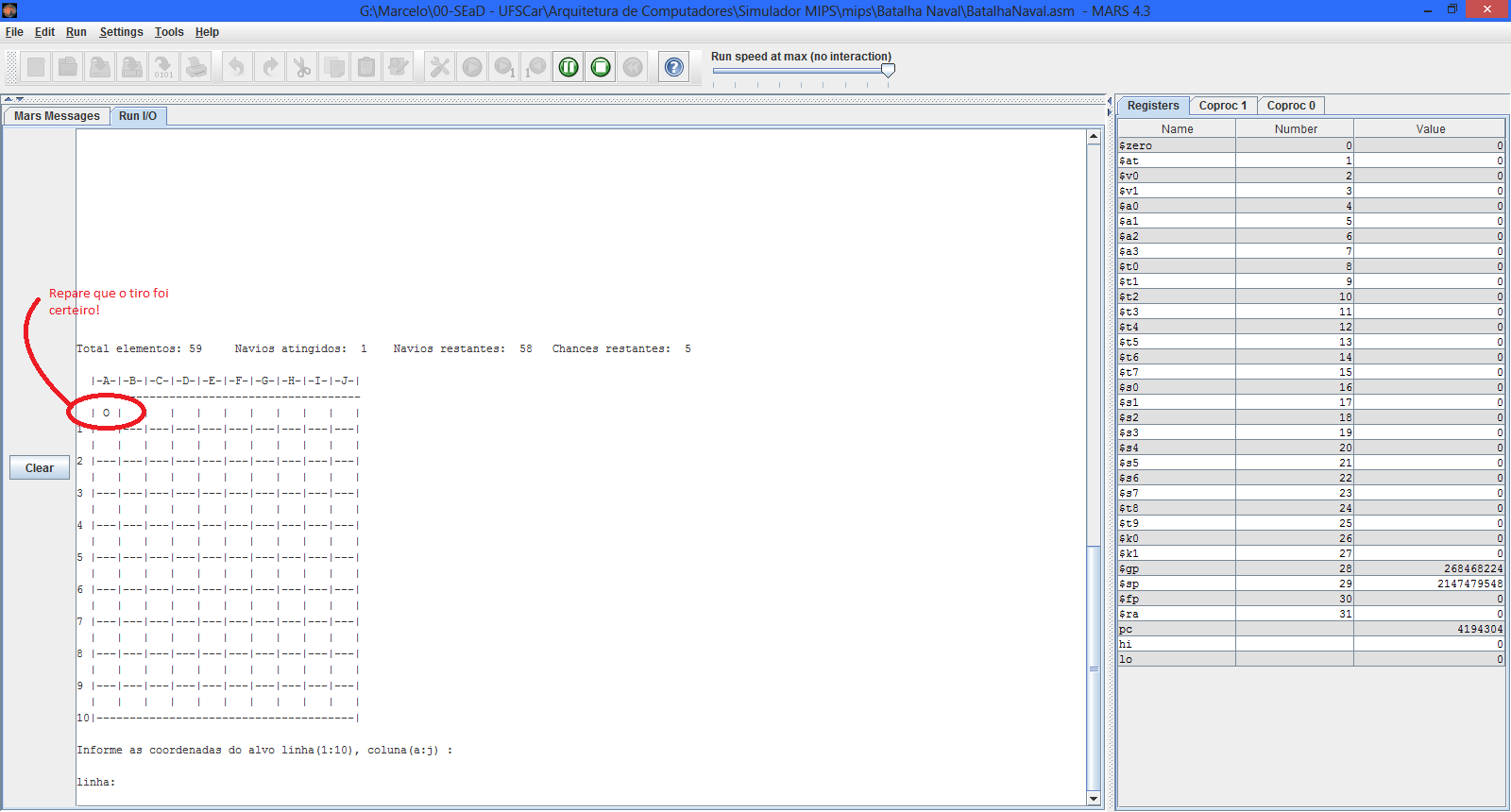


Figura 4 - visualização do tiro certeiro no quadro naval

Ao final do jogo, independente da vitória ou derrota do jogador (a), o sistema do jogo imprime na tela o quadro naval de resposta, como pode ser observado na figura 5. Este quadro naval mostra todas as posições das embarcações, para que o jogador reflita e planeje novas estratégias de jogo.

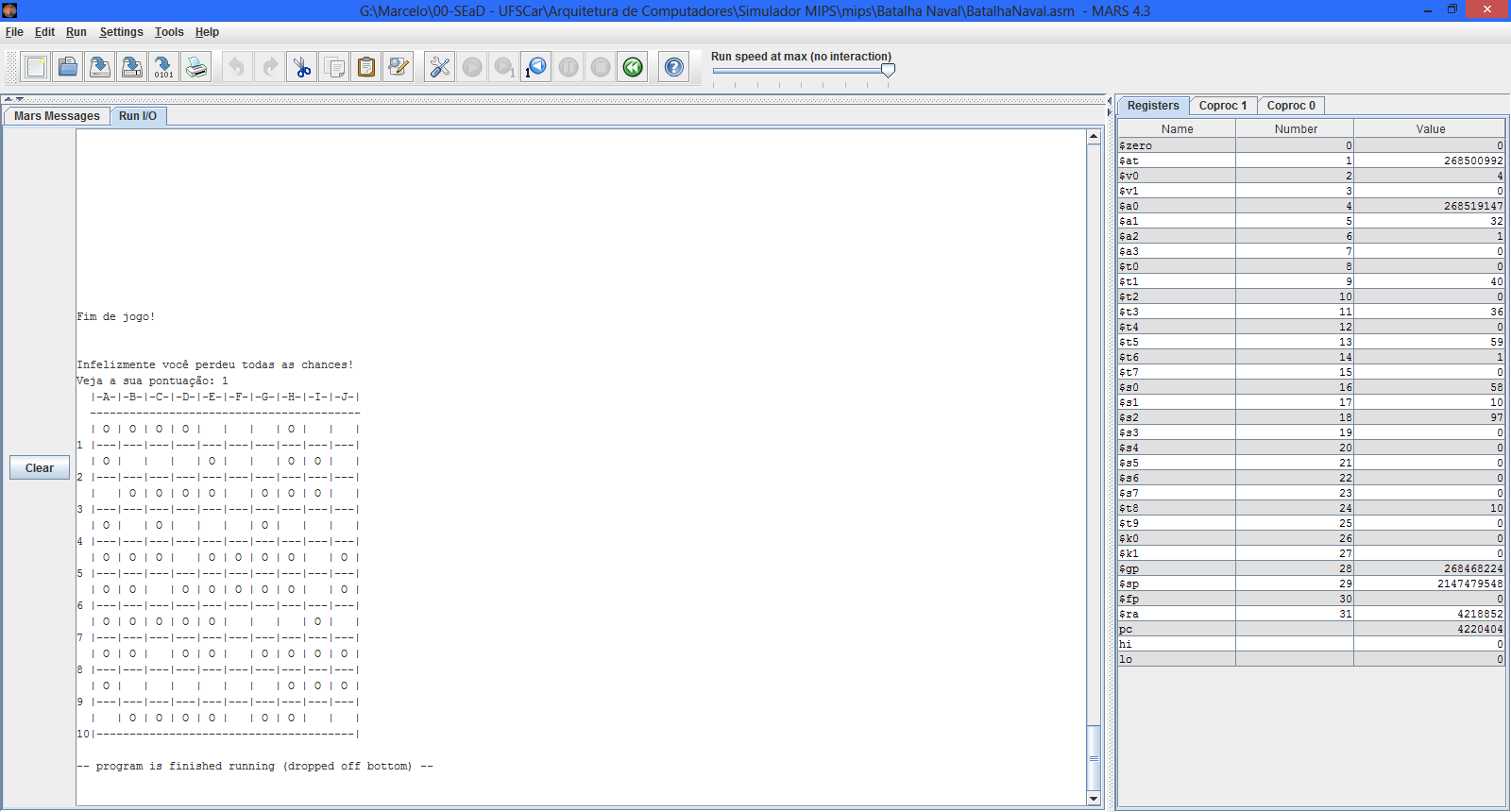


Figura 5 - Quadro naval de resposta

Juntamente com este quadro naval de resposta, é visualizada a pontuação geral do jogador (a), com o total de tiros certeiros.

## 2.2. Divisão de Trabalho

O trabalho teve sua divisão segundo os integrantes que conheciam o jogo e aqueles que não o conheciam.

Para os que conheciam, ficou a parte de implementação do mesmo na linguagem Assembly, para os que não conheciam, ficaram as partes de pesquisa e formas de melhorias da linguagem, para que pudéssemos extrair o melhor da linguagem.

# 3. Especificações do sistema da Batalha Naval 1.0 em Assembly e Conclusões finais.

Para utilizar e visualizar com qualidade o jogo da Batalha Naval 1.0 em Assembly no simulador MARS, o jogador deve maximizar a tela de execução do programa, com o título de “RUN I/O”, assim como mostra a figura 6:

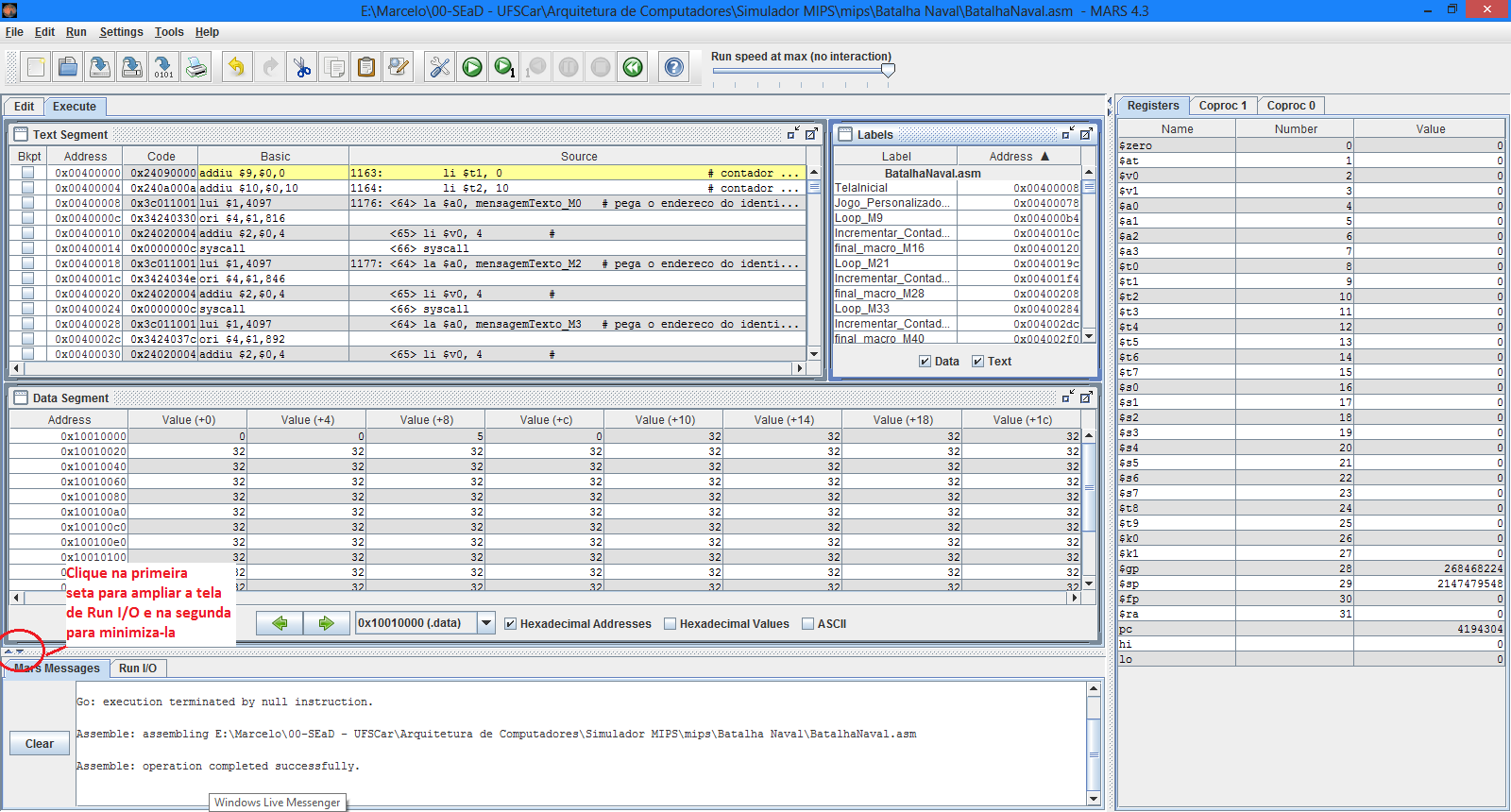


Figura 6 - maximização da tela de execução RUN I/O

Assim, a tela será maximizada e o jogo poderá ser melhor visualizado e o jogador poderá ter uma visão geral do jogo e do quadro naval.

Para acessar as figuras utilizadas neste trabalho com resolução de boa qualidade, basta acessá-las no repositório no diretório Imagens em nossa pasta TheMasters.

Para concluir, percebemos que ao criar um jogo na linguagem Assembly, fizemos mais do que praticar o conhecimento que obtivemos ao longo da disciplina, pois buscamos novos conhecimentos, que possibilitaram solidificar o conteúdo assimilado até o presente momento.

Escolhemos a batalha naval pela interessante lógica que este jogo exige para com os seus jogadores, de modo a desafiar o raciocínio destes em empolgantes e difíceis partidas deste fantástico jogo.

Todo o código fonte foi desenvolvido com o apoio do painel de ajuda do MARS, mostrando assim o quando este simulador é eficiente para fins didáticos.

Esperamos que nosso projeto sirva de fonte de conhecimento para futuros alunos e que possam aproveitar o máximo do desenvolvimento em linguagem Assembly no simulador MARS.