

UNIVERSIDADE FEDERAL DE SERGIPE

CAMPUS UNIVERSITÁRIO PROF. ALBERTO CARVALHO

DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO

GUILHERME SANTOS COSTA

RAUL ALVES DA SILVA

JOGO DA FORCA - ASSEMBLY

ITABAIANA

2021

GUILHERME SANTOS COSTA

RAUL ALVES DA SILVA

**JOGO DA FORCA - ASSEMBLY**

**Artigo referente à implementação de um jogo da forca feito no simulador do processador MIPS – Mars, que deverá ser entregue junto à disciplina OAC (organização e arquitetura de computadores), como nota da unidade.**

**ITABAIANA**

**2021**

SUMÁRIO

[RESUMO 4](#_Toc64833895)

[1. INTRODUÇÃO 5](#_Toc64833896)

[2. OBJETIVOS 5](#_Toc64833897)

[3. ASPECTOS RELEVANTES DA IMPLEMENTAÇÃO 5](#_Toc64833898)

[3.1 MANIPULAÇÃO DO ARQUIVO 6](#_Toc64833899)

3.2 EXIBIÇÃO DA PALAVRA ESCOLHIDA...........................................................................................7

3.3 VERIFICAÇÃO DA LETRA INFORMADA......................................................................................8

[4. DIFICULDADES NA IMPLEMENTAÇÃO 8](#_Toc64833900)

[5. CONCLUSÃO 9](#_Toc64833901)

# 

# RESUMO

O presente artigo tem por escopo mostrar as principais funcionalidades de um jogo da forca implementado em Assembly (mais especificamente no simulador para o MIPS – Mars), desde a fase preparatória do projeto, mostrando os objetivos pretendidos, até a sua conclusão, com a lógica do código feito e as dificuldades para se chegar até ela.

**ABSTRACT**

The purpose of this article is to show the main features of a hangman game implemented in Assembly (more specifically in the MIPS - Mars simulator), from the preparatory phase of the project, showing the intended objectives, until its conclusion, with the logic of the code made and the difficulties to reach it.

# INTRODUÇÃO

No modelo de projeto proposto, foi dado uma série de temas a serem escolhidos pelas respectivas duplas e desenvolvidos na linguagem Assembly. O jogo da forca, pelo qual ficamos responsáveis, trata-se de um jogo de adivinhação onde o competidor tenta acertar uma palavra escondida com base na quantidade de letras e em uma dica relacionado a ela (fruta, marca de carro, esporte...). O jogador tem seis chances para acertar a palavra, levando-se em consideração a quantidade de membros do corpo humano que será desenhado na forca a cada erro, e caso não consiga nessas chances, ele perde. Atualmente, este jogo é uma brincadeira muito conhecida entre as crianças, e utilizada principalmente para estimular o vocabulário delas.

# OBJETIVOS

**2.1 Gerais**

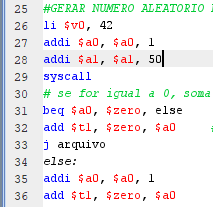
* Aprimorar o conhecimento na linguagem Assembly por meio de novas experiências vivenciadas na implementação de um projeto prático.
* Implementar um Jogo da Forca totalmente funcional, simples de executar/utilizar e semelhante, na medida do possível, com o jogo original e conhecido por todos.

**2.2 Específicos**

* Fazer uso do maior número possível de comandos, syscalls e diretivas que o MARS proporciona, para conseguir completar o primeiro objetivo geral citado anteriormente.
* Compreender todo o gerenciamento de memória do MIPS (tamanho ocupado, registradores utilizados) durante a realização e execução do projeto.

# ASPECTOS RELEVANTES DA IMPLEMENTAÇÃO

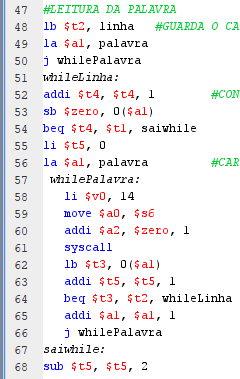
O projeto conta com um arquivo base (“palavras.txt”) contendo 50 palavras quaisquer que poderão ser utilizadas no jogo. Primeiramente, por meio de um syscall, o código gera um número aleatório por meio do qual sorteará a palavra futuramente.



Em $a1 adicionamos um valor que representará o limite máximo entre os números sorteados, portanto será entre 0 e 50. Contudo, é importante frisar que foi constatado um erro de execução quando o número sorteado dava 0, por isso, fez-se um tratamento de erro para sempre que o número sorteado desse 0, adicionasse 1 a ele.

# MANIPULAÇÃO DO ARQUIVO

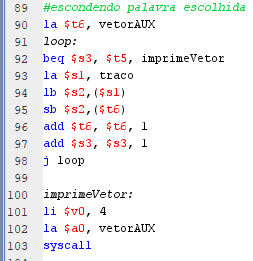
Depois de escolhido o número, passa-se à fase de manipulação do arquivo (isso inclui abertura, leitura e fechamento) para carregar a palavra numa variável do programa. De forma simples e sem dificuldade foram feitas as syscalls de abertura e fechamento, o que demandou mais atenção foi o processo de leitura da palavra.



Esse processo foi feito lendo byte a byte da palavra até encontrar a quebra de linha do arquivo. Para isso, logo no início guardou-se em registradores auxiliares um endereço que iria armazenar a própria palavra ($a1) e o caractere “\n” ($t2) para usar como parâmetro na quebra de linha. Seguindo com a lógica, foram feitos dois laços de repetição, um para cada palavra (“whilePalavra”) e outro para cada linha (“whileLinha”) do arquivo; o primeiro vai executar a syscall lendo 1 byte por vez da palavra e comparando, em seguida, com o byte da quebra de linha anteriormente lido em $t2, já o segundo será ativado quando ambos os bytes foram iguais, e adicionará 1 a um registrador auxiliar que irá parar exatamente no valor do número aleatório gerado, em outras palavras, na palavra escolhida. Concomitantemente a esse processo, existe um registrador de utilidade futura, mas muito importante ($t5), responsável por contar o número de letras de cada palavra lida. Ele é zerado toda vez que encontra-se uma quebra de linha, e incrementado toda vez que lê-se um byte.

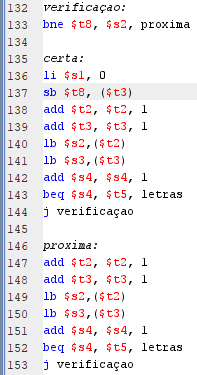
* 1. **EXIBIÇÃO DA PALAVRA ESCOLHIDA**

Outro ponto que faz parte do esqueleto principal do jogo é a parte de exibição da palavra escolhida, de forma que ela esteja oculta ao competidor, claro.



Esse processo foi feito utilizando-se a variável vetorAUX (.space) que contém 50 posições totalmente vazias inicialmente. Então, através do comando “store byte” (sb), armazenou-se neste vetor ($t6) o byte do caractere “-” que foi lido anteriormente ($s1), numa quantidade igual a quantidade de letras da palavra escolhida (valor que ficou guardado em $t5 desde a parte de manipulação do arquivo). Para finalizar, imprimimos ele na tela por meio de um syscall.

* 1. **VERIFICAÇÃO DA LETRA INFORMADA**



Por fim, tem-se a parte chave do projeto que é a verificação da letra informada pelo usuário. Anteriormente a esse pedaço de código foram feitos 2 syscalls responsáveis por imprimir uma mensagem pedindo ao usuário que informa uma letra, e por ler essa letra informada. Vale ressaltar que não foi feita nenhuma distinção quanto ao caractere informado, ou seja, o usuário pode informar um “5” que o programa considerará como um caractere e contará como um erro por não existir na palavra. Além disso, caso o usuário informe mais de um caractere por vez (“ba”), será considerado apenas o primeiro informado também. Voltando para a lógica da verificação, ela acontece da seguinte maneira: primeiro compara-se o byte da letra informada com o byte da primeira posição da palavra (linha 133), caso sejam iguais, a letra informada será armazenada no vetor de letras usadas (por meio do comando “store byte”), a palavra será atualizada e os endereços de ambos os vetores serão incrementados em 1, para a próxima verificação; caso sejam diferentes, ocorrerá o “jump” para a label “proxima” que irá apenas incrementar os endereços dos vetores para comparar novamente, até encontrar a igual.

# DIFICULDADES NA IMPLEMENTAÇÃO

No que diz respeito à implementação do projeto, à priori houve uma dificuldade geral por ser algo prático e que requeria uma técnica mais avançada, num ambiente e estilo de programação totalmente diferente dos anteriores. A falta de experiência com a linguagem e a escassez de conteúdo na internet, podem ter sido fatores chave para tornar a implementação um pouco mais difícil. Citando algo mais específico agora, a dupla também achou complicado o gerenciamento dos registradores. Muitas vezes era necessário sobrescrever o valor de algum, mas não sabia mais que dado aquele registrador estava guardando e se poderia ser sobrescrito.

# CONCLUSÃO

Por todas as dificuldades expostas anteriormente, desenvolver o jogo da forca em Assembly foi um desafio, mas ao mesmo tempo foi gratificante. De nenhuma outra forma o aprendizado da linguagem poderia ser tão engrandecedor, do que fazendo um projeto prático que envolvesse ela. No futuro, talvez esse jogo possa ser melhorado, acrescentando uma interface que desenhe o próprio boneco ao errar a letra (no atual tem apenas um score contando a pontuação).