# Snake

**Alana Schwendler, Felipe Camargo Gruendemann**

[1]Disciplina de AOC I – Prof. Guilherme Correa e Prof. Marcelo Porto
Curso de Ciência da Computação

{aschwendler, fcgruendemann}@inf.ufpel.edu.br

## 1. Introdução

Com a proposta de realizar um projeto final para a disciplina de Arquitetura e Organização de Computadores I (AOC I) e que envolvesse os conhecimentos adquiridos na cadeira, tivemos a ideia de recriar o jogo Snake na linguagem Assembly do MIPS utlizando o simulador *MIPS Assembler and Runtime Simulator* (MARS).

## 2. Snake

O jogo snake se popularizou em celulares antigos e com pouca capacidade de processamento e armazenamento. É um jogo simples onde o objetivo é deslocar o personagem, Snake, para que ele consiga se alimentar com a comida que aparece na arena. Quando ele consegue comer a comida, seu corpo cresce. Se o personagem acabar batendo nas bordas da arena ou no seu próprio corpo, o jogo acaba e o jogador perdeu a partida. Para vencer, o jogador deve fazer o personagem comer sua comida até que seu corpo atinja um determinado tamanho.

## 3. Implementação

Para implementar o jogo, foi utilizado o simulador MARS. Na programação do jogo, foram utilizadas diversas subrotinas e macros. Para o preenchimento da arena, utilizamos subrotinas distintas para as bordas, o fundo da arena e a "comida".

O personagem se relaciona com outras subrotinas para realizar seus movimentos na arena do jogo. Para isto, são realizadas operações de multiplicação e divisão, para calcular em quais pixeis o corpo do personagem deve aparecer. A comida irá aparecer de maneira aleatória na arena, utilizando uma chamada de sistema que gera um número ramdômico.

Também foram utilizados macros para realizar deslocamento na arena e para fazer manipulação na pilha (*Stack*), e.g., os macros para a pilha são nomeados como push e pop. Estes macros recebem um registrador como parâmetro e operam de acordo com o valor armazenado no registrador.

### 3.1. Configuração

Antes de rodar o código, é necessário conectar o *Bitmap Display* e o *Keyboard and Display MMIO Simulator* ao MIPS, e configurá-lo de maneira correta. O *Bitmap Display* deve utilizar as seguintes configurações:

- Unit width in pixels: 16;
- Unit height in pixels: 16;
- Display width in pixels: 512;

- Display height in pixels: 512;
- Base address for display: 0x10010000 (static data).

Além disto, é necessário conectar o teclado do MIPS para que o movimento do personagem seja dado pelas entradas do teclado. Os movimentos são:

- w: cima;
- a: esquerda;
- s: baixo;
- d: direita.

Quando o jogo começa a executar, aparece para o jogador um menu com opção de jogar e sair. Para jogar, deve ser pressionada a tecla **w**. Para sair, deve ser pressionada a tecla **s**. As configurações foram escolhidas de maneira que o jogo ficasse com o layout limpo e claro.

### 3.2. O jogo

A arena do jogo foi preenchida de maneira que todas as bordas sejam fechadas e o jogador tem um campo limitado para percorrer. Com as configurações ajustadas, a arena do jogo se apresenta de acordo com a figura 1.


**Figura 1. Ilustração do menu inicial.**

Se o jogador quiser iniciar o jogo, após pressionar a tecla **w**, a tela mostrada para ele é apresentadao como segue na figura 2

Ao final do jogo, se o jogador colide com as bordas da arena, ou com seu próprio corpo,e o programa é finalizado e mostra uma mensagem de fim de jogo para o jogador. A mensagem pode se vista na figura 3. Existem então duas opções: caso o jogador queira jogar novamente, basta pressionar a tecla **w**. Caso deseje sair, ele deve pressionar a tecla **s**.

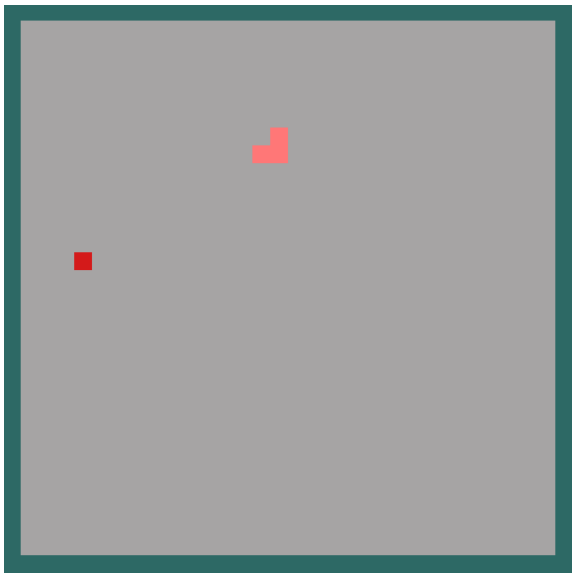

**Figura 2. Ilustração da arena.**

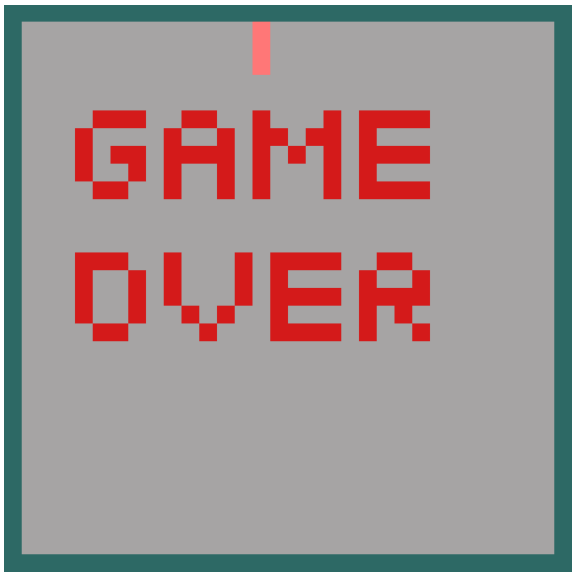


**Figura 3. Mensagem de fim de jogo.**