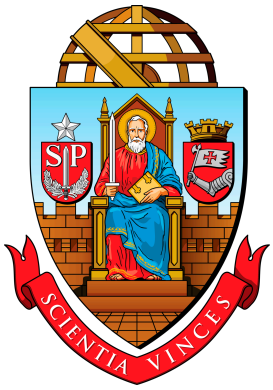


**UNIVERSIDADE DE SÃO PAULO**

**ESCOLA DE ENGENHARIA DE SÃO CARLOS - EESC**

**INSTITUTO DE CIÊNCIAS MATEMÁTICAS E COMPUTAÇÃO - ICMC**

Engenharia de Computação

**SSC0902 – Organização e Arquitetura de Computadores**

# Trabalho Prático 01: Implementação do jogo Jokenpo em Assembly

Bárbara Naomi Morimoto Hatano - 13678755

Luiza Rodrigues Cardoso - 14593332

Letícia Crepaldi da Cunha - 11800879

São Carlos–SP

2024

## 1. Desenvolvimento e Implementação

O trabalho foi desenvolvido em Assembly utilizando a arquitetura de conjunto de instruções RISC-V e o simulador “RARS”.

O jogo Jokenpo tem como princípio uma escolha entre as opções pedra, papel e tesoura e a partir da comparação entre os jogadores, aqueles que tiver escolhido a melhor opção da tentativa se sagra vencedor.

## 2. Regras

Para implementar o jogo Jokenpo em RISC_V - Assembly, definimos algumas estratégias:

- Utilizar valores numéricos para representar os termos pedra, papel e tesoura, sendo neste caso:
    - 0 - corresponde ao termo Pedra;
    - 1 - corresponde ao termo Papel;
    - 2 - corresponde ao termo Tesoura;

- O valor numérico 3 corresponde a instrução de encerrar a partida;
- Tomamos como consenso comum que a opção pedra sempre ganha da opção tesoura, tesoura ganha da opção papel e papel ganha da opção pedra, regras básicas do jogo.

## 3. Apresentação dos rótulos implementados

No primeiro rótulo (_start), realizamos a apresentação das opções de jogada e seus respectivos valores numéricos, bem como a leitura da opção escolhida através da syscall ReadInt (5). Caso o jogador tenha optado pela opção 3, se encerra a partida e imprime o histórico.

No rótulo principal do jogo (“game_loop”), é realizada uma chamada de sistema para gerar um inteiro pseudoaleatório entre 0 e 2, com limites superior e inferior armazenados nos registradores a1 e a0 de forma que:

$$\text{limite inferior (a0)} \leq \text{número aleatório} < \text{limite superior (a1)}$$

Depois disso, as jogadas são comparadas e o fluxo de instruções é direcionado para o rótulo _empate caso as escolhas sejam iguais. Se forem diferentes, o programa verifica

a jogada e se redireciona para “checar_pedra” “checar_papel” ou  “checar_tesoura”, dependendo da jogada do usuário, e então para os procedimentos “_derrota” ou “_vitoria”, conforme os critérios estipulados acima.

Por sua vez, as instruções de derrota e vitória realizam uma chamada de sistema para impressão da mensagem correspondente e desviam o programa para o procedimento de inserção das jogadas na lista ligada, rotulado por “add_historico”.

Para armazenar cada jogada, alocamos 12 bytes via a syscall sbrk, sendo 4 bytes para a escolha do jogador, 4 para a do computador e 4 para o ponteiro do próximo nó. O loop “achar_ultimo” percorre a lista até encontrar o último nó, e “colocar_final” atualiza o ponteiro desse nó para apontar o novo nó.

Por fim, os rótulos “mostrar_historico” e “mostrar_resultados” imprimem o histórico das jogadas realizadas.

## 4. Conclusão

Com este trabalho, por meio do simulador “RARS”, foi possível compreender o funcionamento da arquitetura RISC-V e suas especificidades, bem como a implementação de algoritmos. O desenvolvimento deste projeto melhorou nossas habilidades de resolução de problemas e se adaptar a uma nova tecnologia.