TRABALHO ASSEMBLY - RISC-V

JOGO MANCALA

OBJETIVO:

Neste trabalho deve-se implementar, uma versão simplificada do jogo Mancala (também conhecido como *Kalah*) em Assembly para a arquitetura RISC-V (RV32). O objetivo do trabalho é a compreensão e aplicação de conceitos de manipulação de dados, controle de fluxo e interação com o usuário através do terminal, utilizando a arquitetura do conjunto de instruções (ISA) do processador RISC-V. A aplicação deverá ser executada no simulador RARS.

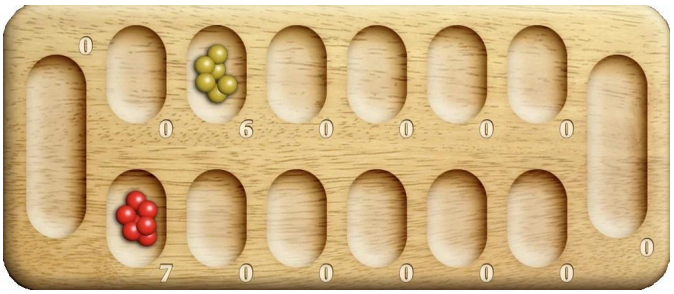
DESCRIÇÃO DO JOGO:

Mancala é uma família de jogos de tabuleiro de estratégia para dois jogadores, originária da África e do Oriente Médio. O nome "Mancala" deriva da palavra árabe naqala (نقلة), que significa "mover". O jogo básico envolve a "semeadura" de sementes (ou pedras) em poços (cavidades) e a "captura" das sementes do oponente.

Tabuleiro: O tabuleiro tradicional possui 12 pequenas cavidades (6 para cada jogador) e duas cavidades maiores, chamadas "poços", uma em cada extremidade. O poço à direita pertence ao Jogador 1 e o poço à esquerda pertence ao Jogador 2.

Configuração Inicial: No início do jogo, 4 sementes são colocadas em cada uma das 12 cavidades. Os poços começam vazios.

Objetivo: O objetivo principal é capturar mais sementes no seu poço do que o seu adversário.



Turno de um Jogador (Semeadura):

- No seu turno, um jogador escolhe uma de suas cavidades (as 6 mais próximas de si) que contém sementes.
- Todas as sementes dessa cavidade são distribuídas nas cavidades seguintes, começando pela próxima cavidade no sentido anti-horário, ele deposita uma semente em cada cavidade sucessiva, incluindo seu próprio poço, mas pulando o poço do adversário caso passe por ele.

Regras Especiais de Fim de Semeadura:

- **Ganha Outro Turno:** Se a última semente depositada cair no poço do próprio jogador, ele ganha um turno extra e joga novamente.
- **Captura:** Se a última semente depositada cair em uma cavidade VAZIA do próprio jogador, e a cavidade oposta do adversário contiver sementes, o jogador captura todas as

sementes da cavidade oposta do adversário, mais a sua última semente. Todas as sementes capturadas são movidas para o poço do jogador atual.

- **Troca de Turno:** Se nenhuma das condições acima for atendida, o turno termina e passa para o outro jogador.

Fim de Jogo:

- O jogo termina quando todas as 6 cavidades de um dos lados do tabuleiro ficam vazias.
- O jogador que ainda tem sementes em suas cavidades pega todas elas e as move para o seu próprio poço.
- O jogador com o maior número total de sementes em seu poço é o vencedor.

Requisitos Técnicos:

1. Armazenamento de Dados:

- Armazene na memória a quantidade de sementes existentes em cada cavidade e em cada poço;
- A quantidade de sementes padrão nas cavidades no início do jogo: SEEDS_INIT = 4;
- Armazene na memória quem é o jogador (1 ou 2) que está jogando (de quem é o turno atual);
- Armazene na memória o total de vitórias de cada um dos jogadores;

2. Controle de Fluxo:

- O jogo deve ser baseado em loops que controlam as rodadas de cada um dos jogadores e, ao final de cada interação com qualquer jogador, deve-se mostrar o estado atual do tabuleiro;
- Deve-se identificar as condições de fim de jogo e apresentar qual o jogador que venceu a partida, incrementando o número de vitórias deste e em seguida perguntar se deseja iniciar uma nova partida. Caso sim, deve-se reiniciar o tabuleiro.

3. Modularidade do código

- O jogo deve utilizar funções que recebem parâmetros via registradores (a0, a1, a2, etc) ou via pilha, evitando a repetição de trechos de código.
- Funções sugeridas:
  a) `inicia_tabuleiro`
  b) `mostra_tabuleiro`
  c) `distribui_sementes`
  d) `verifica_vencedor`

4. Interação com o Usuário:

- O programa deve interagir com o usuário via entrada e saída padrão no terminal. Use chamadas de sistema para ler entradas do jogador e exibir informações no terminal.

Exemplo de Saída:

```
      Bem-vindo ao jogo de Mancala

Tabuleiro Atual:
       12     <-- JOGADOR 2     7
+----+----+----+----+----+----+----+----+
|    | 4  | 4  | 4  | 4  | 4  | 4  |    |
| 0  |----+----+----+----+----+----|  0 |
|    | 4  | 4  | 4  | 4  | 4  | 4  |    |
+----+----+----+----+----+----+----+----+
       0      --> JOGADOR 1     5

Jogador 1
Escolha a cavidade [0-5]: 2

       12     <-- JOGADOR 2     7
+----+----+----+----+----+----+----+----+
|    | 4  | 4  | 4  | 4  | 4  | 4  |    |
| 0  |----+----+----+----+----+----|  1 |
|    | 4  | 4  | 0  | 5  | 5  | 5  |    |
+----+----+----+----+----+----+----+----+
       0      --> JOGADOR 1     5

TURNO EXTRA
Jogador 1
Escolha a cavidade [0-5]: 3

       12     <-- JOGADOR 2     7
+----+----+----+----+----+----+----+----+
|    | 4  | 4  | 4  | 4  | 5  | 5  |    |
| 0  |----+----+----+----+----+----|  2 |
|    | 4  | 4  | 0  | 0  | 6  | 6  |    |
+----+----+----+----+----+----+----+----+
       0      --> JOGADOR 1     5

NOVO JOGADOR
Jogador 2
Escolha a cavidade [7-12]: 12

       12     <-- JOGADOR 2     7
+----+----+----+----+----+----+----+----+
|    | 0  | 4  | 4  | 4  | 5  | 5  |    |
| 1  |----+----+----+----+----+----|  2 |
|    | 5  | 5  | 1  | 0  | 6  | 6  |    |
+----+----+----+----+----+----+----+----+
       0      --> JOGADOR 1     5

NOVO JOGADOR
Jogador 1
Escolha a cavidade [0-5]: ……
```

Exemplo de fluxo para o programa:

```
MAIN:
   CALL INICIA_TABULEIRO
MAIN_LOOP:
   CALL MOSTRA_TABULEIRO
   IF JOGADOR_ATUAL == 1:
      CALL DISTRIBUI_SEMENTES(1)
      IF RETURN == TURNO_EXTRA
J MAIN_LOOP
   ELSE:
      CALL   DISTRIBUI_SEMENTES(2)
      IF RETURN == TURNO_EXTRA
J MAIN_LOOP
   CALL VERIFICA_VENCEDOR
   IF GAME_OVER
        J FIM_DA_PARTIDA
   TROCA JOGADOR_ATUAL
   J MAIN_LOOP


FIM_DA_PARTIDA:
   CALL MOSTRA_VENCEDOR
   CALL ATUALIZA_PLACAR
   IF JOGAR_NOVAMENTE == 1
       J MAIN
   EXIT
```

Critérios de Avaliação:

- **Funcionalidade:** O jogo funciona corretamente no terminal, com interação completa (distribuição de cartas, jogadas, vitória/derrota).
- **Estrutura e modularidade do Código:** O código está bem organizado, modular e comentado. Utiliza conceitos adequados de Assembly, como controle de fluxo e manipulação de registradores e memória, passagem de parâmetros e retorno das funções.
- **Interatividade e Saída no Terminal:** A interação com o usuário é clara, e o jogo é exibido de forma legível no terminal.
- **Arquivo de documentação**: atendimento das solicitações da documentação, capacidade de síntese e clareza das explicações.

ENTREGA:

- Este trabalho deve ser realizado em **grupos de até 2 pessoas**.
- As entregas serão realizadas no SIGAA.
- **Entrega parcial:**
  - **Data: 27/10/2025**
  - O código em Assembly em um arquivo .asm contendo o nome e matrícula dos integrantes, com as seguintes funcionalidades implementadas:
    - Função de inicialização do tabuleiro;
    - Função que mostra o tabuleiro;
    - Leitura do índice da cavidade e validação de acordo com o jogador;
    - Principais mensagens e interação para realizar uma jogada completa

- **Entrega final**
  - **Data: 13/11/2025**
  - Deve ser entregue, por apenas um dos integrantes do grupo, um arquivo **.zip** contendo:
    - O código em Assembly para a arquitetura RISC-V deve ser entregue em um arquivo *.asm* contendo a implementação, os comentários necessários, o nome e a matrícula dos integrantes.
    - Um arquivo de documentação (sem capa, apenas nome dos integrantes, **no máximo 4 páginas**) explicando o uso da memória e dos registradores, enumerando e descrevendo as funções implementadas, o fluxograma geral de funcionamento do programa.

- **Apresentações:**
  - Conforme planilha de horários disponibilizada no SIGAA
  - 30 minutos por grupo;
  - Obrigatório a presença conjunta dos integrantes do grupo.

REFERÊNCIAS:

- https://pt.wikipedia.org/wiki/Mancala
- https://www.crazygames.com/game/mancala-classic