Arquitetura de Processadores na Prática – TP2

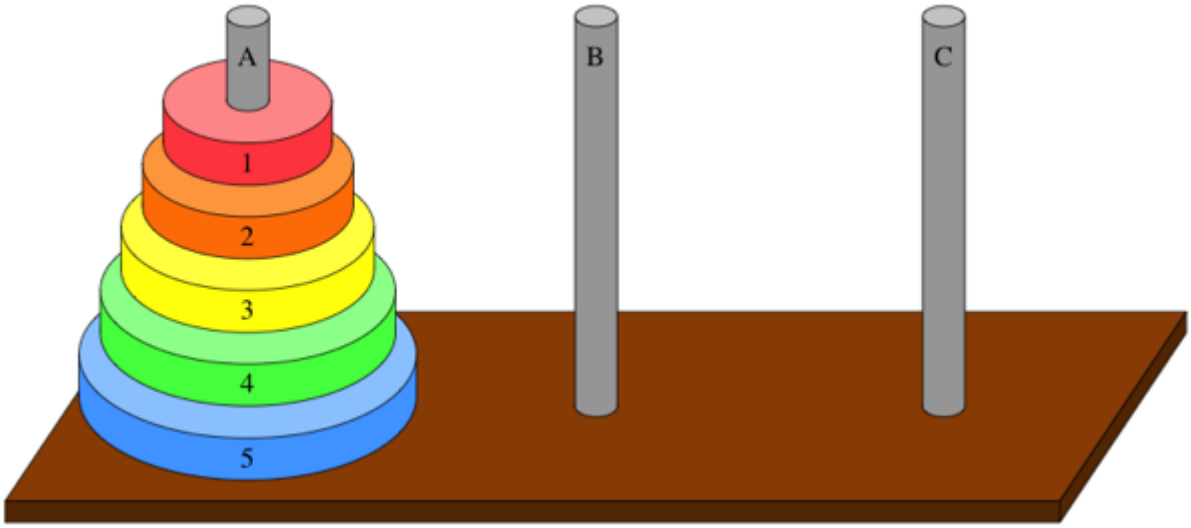
1 FORMAÇÃO DOS GRUPOS

Formação dos grupos: Os grupos devem ser de 2 ou 3 alunos. Não há trabalhos individuais ou de grupos com mais de 3 alunos.

2 TRABALHO A SER DESENVOLVIDO E REGRAS DO JOGO

Torres de Hanoi

1. Contexto: Trata-se de um problema clássico de uso de um conjunto limitado de recursos e regras simples que devem ser respeitadas. Refere-se à Figura abaixo para explicar o problema de forma simples:



2. Conforme mostra a Figura, existem exatamente três postes (A, B e C) e um número parametrizável de discos perfurados de tamanhos diferentes (na Figura aparecem 5, onde inteiros menores correspondem a discos menores). O número de discos poderia ser apenas 1, ou 10, ou 64, ou 100 etc.). A situação inicial é sempre a mesma, todos os discos encontram-se em um poste e os outros dois postes estão vazios. Vejam uma ilustração de como se pode visualizar a solução deste problema executando a aplicação web disponível em https://archive.org/details/TowersHanoi.
3. Resolver o problema de Torres de Hanoi consiste em mover todos os discos do poste fonte (no caso da Figura acima, o poste A) para um poste destino (que segundo a Figura pode ser o poste B ou o poste C). Para tanto, duas regras devem ser seguidas:
   3.1. Só se pode mover apenas um disco por vez.
   3.2. Um disco nunca pode ficar em cima de um disco menor que ele. Por exemplo, se o disco 3 estiver em um pino, então todos os discos abaixo dele devem ter número maior do que 3.
4. Procurem na Internet **um algoritmo recursivo** (chamemos ele de **`hanoi_T`**) que seja capaz de:
   4.1. Resolver o problema de Torres de Hanoi para um número arbitrário de discos;
   4.2. Calcular o número de passos (movimentos individuais de discos) necessários para resolver o problema para **n discos**.
   4.3. Um exemplo (**incompleto**) de tal rotina seria o dado abaixo. Por que este exemplo está incompleto? Não se limitem a ele...

```
1.   FUNCTION han_move_tower (disk, source, dest, spare):
2.   IF disk == 1, THEN:
3.       move disk from source to dest
4.   ELSE:
5.       han_move_tower(disk - 1, source, spare, dest)
6.       move disk from source to dest
7.       han_move_tower(disk - 1, spare, dest, source)
8.   END IF.
```

5. Elabore um programa na linguagem de montagem do MIPS **(3 pontos)** para testar a o algoritmo **`hanoi_T`**, implementando sob a forma de uma subrotina recursiva escrita na mesma linguagem de montagem **(4 pontos)**. Lembre-se que o programa principal deve não apenas **(1)** chamar a rotina, mas também **(2)** passar argumentos para esta, como exigido no item 6 abaixo, todos via pilha, **(3)** receber o valor de retorno, **(4)** imprimir este valor na console do simulador. Crie também um menu de opções e execute seu programa principal como um laço que permita testar a execução da rotina tantas quanto se deseje. Note que quem usar a pilha deve sempre esvaziá-la dos dados lá colocados antes de retornar para quem lhe chamou. Defina uma área de dados adequada para o programa. Acrescente variáveis, se considerar necessário. **IMPORTANTE: A rotina `hanoi_T` deve ser responsável por acumular o número de movimentos realizados.**
6. Respeite as seguintes convenções:
   6.1. Passagem de argumentos – todos os argumentos para a função devem ser passados através da pilha, apontada pelo registrador $sp. Isso obviamente vale também para argumentos que a função passa para uma chamada recursiva de si própria.
   6.2. O retorno do valor resultante da execução de qualquer instância da função deve também ocorrer através da pilha para quem a chamou.
7. Responda às seguintes questões no relatório final do trabalho **(1 ponto)**: (1) Qual o número do registrador $sp no conjunto de registradores do MIPS e qual o seu valor inicial em hexadecimal (valor inicial atribuído pelo simulador MARS)? (2) Qual é o primeiro valor escrito na pilha pelos seu programa, e qual o significado dele? (3) Mostre o conteúdo da pilha ao entrar na **terceira** chamada aninhada de alguma recursão (use a opção File→Dump Memory do simulador MARS). (4) Qual o conteúdo do registrador $sp neste momento? (5) Isto implica ter quanto espaço alocado na pilha? (6) Observar o retorno do procedimento recursivo. O valor do registrador $sp volta ao valor original? (Se isto não ocorrer seu programa está incorreto, pois sua execução deixa lixo na pilha) (7) Em qual linha de código este valor é reestabelecido?
8. Formato do trabalho e entrega: O trabalho deverá ser entregue via sala do Moodle até a data de **13/05/2022 (sexta-feira)**, contendo o código fonte da aplicação **COMENTADO SEMANTICAMENTE**, as respostas das perguntas do item 5 acima e um relatório final, contendo uma explicação da solução, exemplos de telas do simulador MARS, mostrando alguns dos passos da execução do programa, devidamente comentados **(2 pontos)**.
9. Valor do trabalho: Este trabalho vale **30% da nota de G1** da disciplina.