**imagesUniversidade Federal Rural Semi-Árido - UFERSA**

**Centro de Ciências Exatas e Naturais - CCEN**

**Departamento de Computação - DC**

**Curso Ciência da Computação**

**Disciplina: Sistemas Operacionais**

**Prof. Leiva Casemiro Oliveira**

# 1.4 – Tarefa de Sistemas Operacionais

**INSTRUÇÕES:** Essa atividade pode ser feita em grupo e os códigos (projeto Eclipse) e respostas correspondentes as questões devem ser enviados em um único arquivo (.zip) até a data estabelecida no SIGAA pelo líder do grupo como parte da nota da Unidade em avaliação.

**Orientações gerais**

* A implementação deve ser feita no MARS versão 4.5, disponível em <http://courses.missouristate.edu/kenvollmar/mars/>
* A implementação corresponde de parte da infraestrutura de sistema operacional para realizar gerenciamento de processos.
* Para se ter acesso ao código do MARS, após baixar o arquivo .jar utilize uma ferramenta de descompactação (ex: Winrar) e extraia seu conteúdo. Em seguida crie um projeto em uma IDE para Java (ex: Eclipse) com os arquivos extraídos.
  + O “Apêndice A” do livro “PATTERSON, D. A. ; HENNESSY, J.L. Organização e projeto de computadores – a interface hardware software. 3. ed. Editora Campus, 2005” apresenta detalhes sobre o MIPS e chamadas de sistema.

**Descrição da Implementação:** Sincronização de processos por Semáforos

* Esta atividade depende do correto funcionamento da ferramenta da atividade anterior (*timer*) e a PCB tenha o estado de bloqueado.
* Nesta atividade permitirá disponibilizar o serviço de semáforos para os processos executando no MARS
* Para isso deverão ser criadas 4 *sycalls:* CreateSemaphore, TerminateSemaphore, DownSemaphore e UpSemaphore
  + **CreateSemaphore**: recebe como parâmetro o endereço de uma variável inteira, e assumirá o valor atual dessa variável como o valor inicial do semáforo. A chamada de sistema cria, internamente, uma lista na qual os processos que acessarem esse semáforo poderão ficar bloqueados.
  + **TerminateSemaphore**: recebe como parâmetro o endereço da variável associada a um semáforo, de modo a eliminar a lista associada ao semáforo; se o endereço da variável não for de um semáforo previamente criado retorna um código de erro.
  + **DownSemaphore**: recebe como parâmetro o endereço da variável associada a um semáforo de modo a acessar as estruturas associadas ao mesmo; se o valor atual do semáforo for maior que zero então decrementa este valor; caso o valor do semáforo for igual a zero o processo que chamou é colocado na fila de bloqueado.
  + **UpSemaphore**: recebe como parâmetro o endereço da variável associada a um semáforo; se existir pelo menos um processo bloqueado então retira um dos processos da fila de bloqueados do semáforo e o transfere para fila de prontos senãoincrementa o valor do semáforo.
* Para validar o semáforo:
  + Implemente, em assembly do MIPS, o problema do produtor consumidor de modo que cada uma seja um processo.
  + A alternância entre os processos deve ser feita pelo timer.
  + A criação dos semáforos (mutex, empty e full) devem ser feitas antes da execução dos processos produtor e consumidor
  + O buffer de dados compartilhado entre produtor e consumidor deve ser implementado como uma pilha na memória (pode-se declarar uma variável array e usar um registrador de propósito geral como ponteiro do topo; ou usar a pilha propriamente dita do MARS e manipular o registrador $SP)
  + Inserir e remover dados no buffer devem ser realizados por LW e SW, de modo que possamos observar o estado da pilha durante a execução.