**Universidade Federal Rural do Semi-Árido - UFERSA**

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

**Departamento de Computação - DC**

**Curso Ciência da Computação**

**Disciplina: Sistemas Operacionais**

**Prof. Leiva Casemiro Oliveira**

# 2.1- 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. A avaliação também conta com a apresentação dos alunos em sala de aula.

**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:** Estruturas para o gerenciamento de memória

* Adicione atributos na sua classe PCB (*Process Control Block*) para manter informações a respeito do gerenciamento de memória do processo tais como:
  + Registrador de limite superior da memória do processo
  + Registrador de limite inferior da memória do processo
  + Os registradores de limites de memória devem ser configurados como:
    - Limite superior: endereço inicial do processo (*label* que o identifica, que também é usado pelo *Fork*)
    - Limite inferior: endereço imediatamente antes do *label* do próximo processo. Caso seja o último processo, deve ser configurado com o endereço final do programa
* Cada vez que o processo for escalonado, além dos registradores físicos do processador, os valores dos registradores de limites da memória também devem atualizados das PCBs de cada processo
* Criar uma classe de gerenciador de memória com atributos globais (e seus respectivos métodos *get* e *set*) para o todos os processos, tais como:
  + Tamanho de bloco de alocação de memória (tamanho da página virtual) em quantidade de instruções do MIPS
  + Quantidade máxima de blocos de alocação por processo
  + Configuração do tipo de algoritmo de substituição de páginas da memória virtual
* A cada acesso a memória (segmento de código) o gerenciador deve:
  + Comparar se endereço acessado está dentro dos limites de memória para o processo em execução
  + Se endereço for válido continue executando normalmente
  + Caso contrário imprime uma mensagem de erro na saída padrão do MARS, informando a tentativa de endereço a ser acessado e os limites do processo, em seguida pare a simulação.
* Para validar
  + Execute o código a seguir, verificando se os limites de memória são os esperados
  + Em seguida modifique o código de um dos Programas para que salte para o código do outro Programa e verifique se o erro é detectado.

Programa1:

addi $s1, $zero, 1 # valor inicial do contador

addi $s2, $zero, 10 # valor limite do contador

loop1: addi $s1, $s1, 1

beq $s1, $s2, fim1

j loop1

fim1: SyscallProcessTerminate

Programa2:

addi $s1, $zero, -1 # valor inicial do contador

addi $s2, $zero, -10 # valor limite do contador

loop2: addi $s1, $s1, -1

beq $s1, $s2, fim2

j loop2

fim2: SyscallProcessTerminate

.include "macros.asm"

.data

.text

#criação dos processos

SyscallFork(Programa1)

SyscallFork(Programa2)

SyscallFork(Idle)

#escalonando o primeiro processo

SyscallProcessChange

Idle:

loop:

NOP

j loop