**imagesUniversidade Federal Rural do 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**

# 0.2– Tarefa de Sistemas Operacionais

**INSTRUÇÕES:** Essa atividade é em grupo e os código 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 para abrir o projeto do MARS no Eclipse**

1. Descompacte (usando ex. Winrar) o projeto MARS do .jar disponível em <http://courses.missouristate.edu/kenvollmar/mars/>
2. Criar um projeto no Eclipse usando os arquivos extraídos seguindo os passos:
   1. File -> New -> Java Project
   2. Dê um nome para o projeto
   3. Desmarque a opção “Use default location”
   4. Escolha o caminho do diretório onde o projeto foi descompactado
   5. Clique em finalizar
3. Compilar o projeto
   1. Quando compilar a 1ª. vez uma janela será aberta para escolha de um dos arquivos com “Main”
   2. No caso do MARS, escolha “Mars – (default package)
4. Criação de Chamada de Sistema (CS)
   1. Pode extender ***AbstractSyscall***
   2. public class MySyscall extends AbstractSyscall{…}
   3. Ou implementando a interface **Syscall** e seus métodos *String getName(), setNumber(), getNumber()* e *simulate()*
5. Criação de *Tool*
   1. Implementando a interface **mars.tools.MarsTool**
   2. Extendendo a classe abstrata **mars.tools.AbstractMarsToolAndApplication**
   3. Em ambos os casos sua classe deve fazer parte do pacote **mars.tools**
   4. O melhor método é o 2o.
   5. Dos 21 métodos, 19 já tem implementação default
6. Extendendo a classe abstrata **mars.tools.AbstractMarsToolAndApplication**
   1. Define pelo menos
   2. String getName(): retorn o nome da ferramenta
   3. JComponent buildMainDisplayArea(): para construir a área central da interface gráfica
   4. Será colocada automaticamente no CENTER de um BorderLayout com o título no NORTH e botões de controle no SOUTH
   5. Vários métodos addAsObserver() estão disponíveis verificar registradores e memória
   6. Adicione os métodos que quiser
   7. Depois de compilar será adicionada automaticamente no MARS

**Parte 1 - Aumentando abstração sem modificar o MARS**

1. Use o código a seguir como exemplo para testar as próximas atividades

|  |  |
| --- | --- |
|  | # Calcula f = (g+h)-(i+j)  .data  g: .word 5 # valor 5 armazenado em g  h: .word 2 # valor 2 armanazenado em h  i: .word 1 # valor 1 armanazenado em i  j: .word 3 # valor 3 armanazenado em j  f: .word 0 # valor 0 armanazenado em f  .text  # carrengando os valores da memoria  la $t0, g # carrega o endereco de g em $t0  lw $s1, 0($t0) # carrega o valor de g em $s1  la $t0, h # carrega o endereco de h em $t0  lw $s2, 0($t0) # carrega o valor de h em $s2  la $t0, i # carrega o endereco de i em $t0  lw $s3, 0($t0) # carrega o valor de i em $s3  la $t0, j # carrega o endereco de j em $t0  lw $s4, 0($t0) # carrega o valor de j em $s4    #Realizando o calculo  add $t0, $s1, $s2 #soma g e h  add $t1, $s3, $s4 #soma i e j  sub $s0, $t0, $t1 #(g+h)-(i+j)    #Armazena o resultado na memoria  la $t0, f # carrega o endereco de f em $t0  sw $s0, 0($t0) # armazena o valor    add $a0, $s0, $zero #copiando o valor de $s0 para $a0  li $v0, 1 #imprimir inteiro  syscall    # Terminando o programa  li $v0, 10 # system call for exit  syscall # we are out of here. |

1. Crie macros para abstrair as seguintes *syscalls*, passando os devidos parâmetros. Salve todas as macros em um mesmo arquivo chamado “macros.asm”.
   1. **done**: para finalizar programa
   2. **print\_int**: para imprimir inteiro
   3. **print\_char**: para imprimir caracter
   4. **print\_string**: para imprimir string
   5. **read\_int**: para leitura de inteiro
   6. **read\_char**: para leitura de inteiro
   7. **read\_string**: para leitura de inteiro
2. Utilize o exemplo da questão 1 e inclua as macros usando a diretiva **.include “macros.asm”** no início do código e substitua as respectivas *syscalls* pela macros criadas.
3. Código em C finalizam com um *return*, o qual acompanha um valor zero quando o programa termina como esperado ou com valor negativo quando acontece algum erro e este erro deve ser reportado para quem chamou o programa. Então, crie uma macro **return** a qual recebe um parâmetro. Se o parâmetro for menor que zero chama a *syscall* exit2 com o parâmetro (indicando um código de erro) senão chama a macro **done** criada na questão 2.
4. Crie 2 macros para manipular variáveis: **load\_var** e **store\_var**. Ambas devem receber 2 parâmetros: o label da variável a ser carregada e o nome do registrador de/para onde o valor está sendo lido/escrito. Em seguida substitua no exemplo da questão 1.

**Parte 2- Criando Chamadas de Sistemas (CS) ou *Syscalls***

1. Crie uma classe “MySyscall” no pacote ***mars.mips.instructions.syscalls***
2. Para utilizar as definições do MARS faça as importações mínimas

import mars.util.\*;

import mars.\*;

1. Nossa classe deve extender ***AbstractSyscall***

public class MySyscall extends AbstractSyscall{…}

1. O construtor da classe deve ter o número e o nome da CS

public MySyscall() {

super(18, "MySyscall");

}

1. Implementar o método **simulate()**

public void simulate(ProgramStatement statement) throws ProcessingException {

SystemIO.printString(“Olá Syscall!\n”);

}

* A E/S padrão do MARS é implementado em **mars.util.SystemIO.java**

1. Verificar a disponibilidade de número de CS e adicione a sua no arquivo **Syscall.properties** que está no mesmo pacote

No nosso exemplo: MySyscall = 18

1. Compila e executa o MARS, clicando no botão “run” do Eclipse
2. Para testar escreva um programa .asm e inclua a *syscall*:

li $v0, 18

syscall

**Parte 3- Criando uma ferramenta do MARS ou *Tool***

1. Vamos construir a nossa tool modificando a classe **mars.tools.IntroToTools.java**
2. Abra essa implementação de tool e salve com o nome “MyTool”
3. Substituir nome da classe e construtores para MyTool
4. Modificar o atributo **String *name*** *para* “MyTool”
5. Modificar o atributo **String *version*** *para* “Version 1.0 (NOME do GRUPO)”
6. Modificar o atributo **String *heading*** *para* “My First Tool”
7. Modificar no método **getName()** para "MyTool“
8. Modificar no método **buildMainDisplayArea()** o texto para "Essa ferramenta foi criada para fins didáticos no curso de SO da UFERSA"
9. Compile o projeto e execute o MARS, em seguida procure sua ferramenta no menu “Tools”.