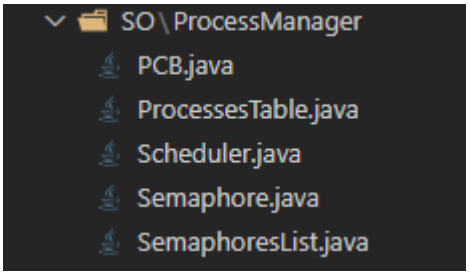
Implementações Sistemas Operacionais - Primeira unidade
Docente: Dr. Leiva Casemiro Oliveira

Grupo 2:
João Felipe Barros Silva
Riad Oliveira de Morais
Ricardo Cezar Fernandes de Melo Junior
Pedro David Rocha Saldanha
Vitor Duarte Bezerra de Oliveira

Arquivos .asm de testes e macros se encontram na pasta testeAssembly

Tarefa 1.1 - Criação da PCB do gerenciador de Processos

Pacote Mars.mips.SO foi criado conforme as requisições e contém os seguintes arquivos:

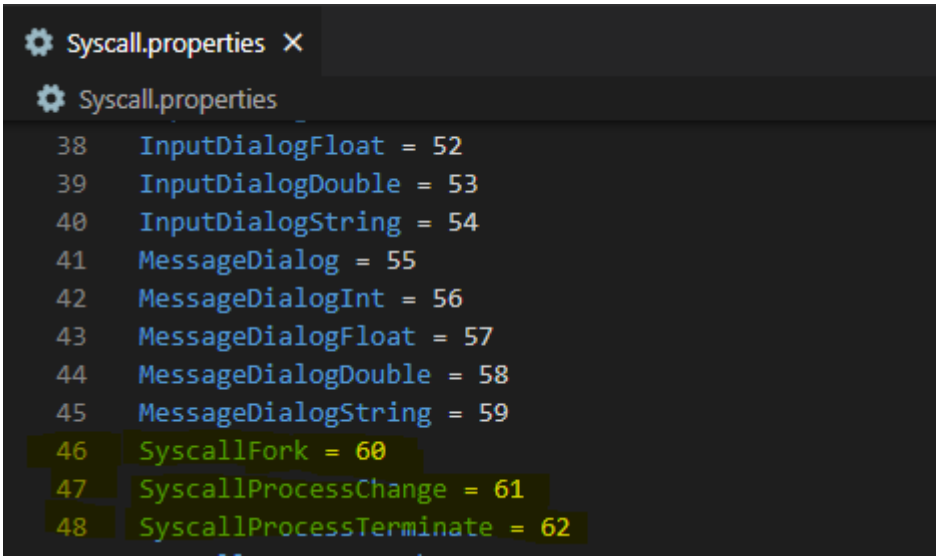


Prints das classes não foram adicionados pois o arquivo ficaria muito grande.

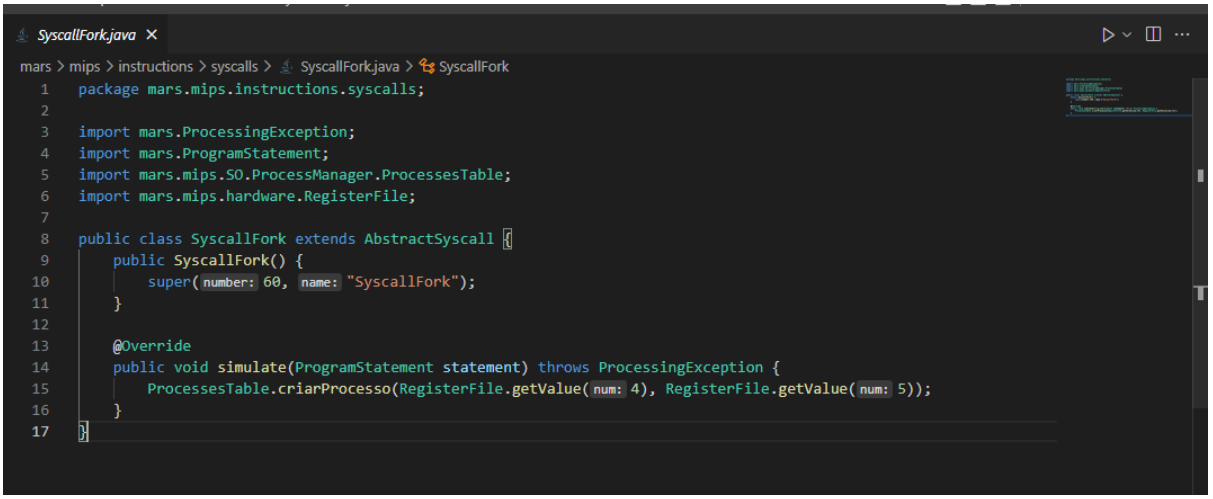
Tarefa 1.2 - Criação das Syscalls do gerenciador de Processos

Syscalls foram criadas no pacote “mars.mips.instructions.syscalls”

Cada uma das novas Syscalls ganhou seu próprio número no arquivo “Syscall.properties” (60, 61 e 62)

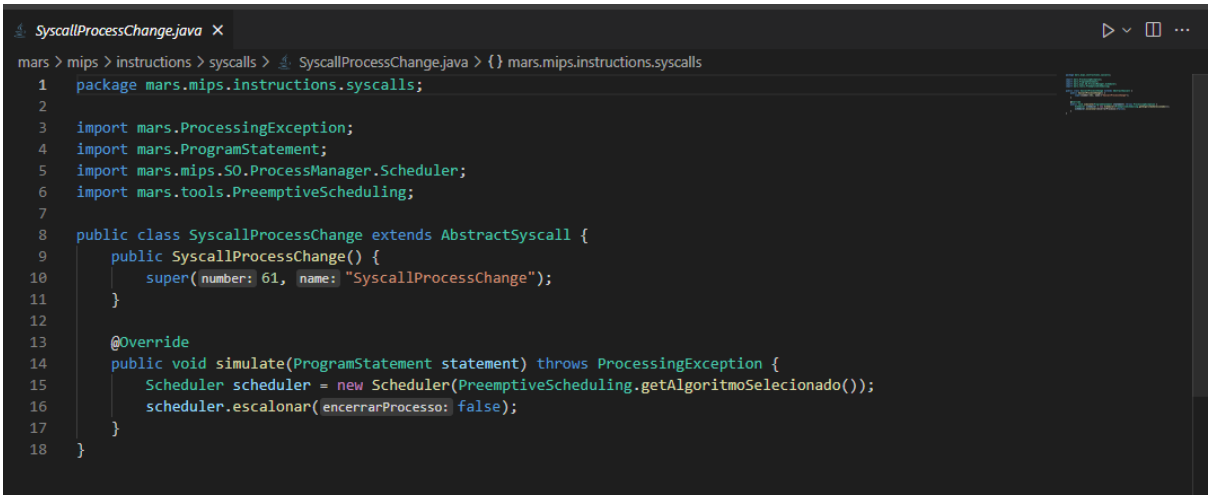


Primeira syscall - SyscallFork



```java
package mars.mips.instructions.syscalls;

import mars.ProcessingException;
import mars.ProgramStatement;
import mars.mips.SO.ProcessManager.ProcessesTable;
import mars.mips.hardware.RegisterFile;

public class SyscallFork extends AbstractSyscall {
    public SyscallFork() {
        super(number: 60, name: "SyscallFork");
    }

    @Override
    public void simulate(ProgramStatement statement) throws ProcessingException {
        ProcessesTable.criarProcesso(RegisterFile.getValue(num: 4), RegisterFile.getValue(num: 5));
    }
}
```

Segunda syscall - SyscallProcessChange

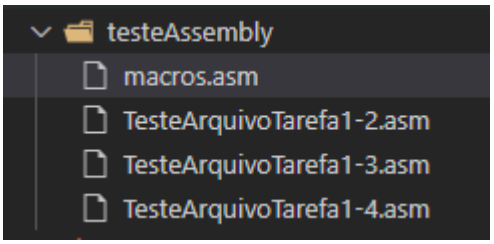


```java
package mars.mips.instructions.syscalls;

import mars.ProcessingException;
import mars.ProgramStatement;
import mars.mips.SO.ProcessManager.Scheduler;
import mars.tools.PreemptiveScheduling;

public class SyscallProcessChange extends AbstractSyscall {
    public SyscallProcessChange() {
        super(number: 61, name: "SyscallProcessChange");
    }

    @Override
    public void simulate(ProgramStatement statement) throws ProcessingException {
        Scheduler scheduler = new Scheduler(PreemptiveScheduling.getAlgoritmoSelecionado());
        scheduler.escalonar(encerrarProcesso: false);
    }
}
```

Terceira syscall - SyscallProcessTerminate

```java
package mars.mips.instructions.syscalls;

import mars.ProcessingException;
import mars.ProgramStatement;
import mars.mips.SO.ProcessManager.Scheduler;
import mars.tools.PreemptiveScheduling;

public class SyscallProcessTerminate extends AbstractSyscall {
    public SyscallProcessTerminate() {
        super(number: 62, name: "SyscallProcessTerminate");
    }

    @Override
    public void simulate(ProgramStatement statement) throws ProcessingException {
        Scheduler scheduler = new Scheduler(PreemptiveScheduling.getAlgoritmoSelecionado());
        scheduler.escalonar(encerrarProcesso: true);
    }
}
```

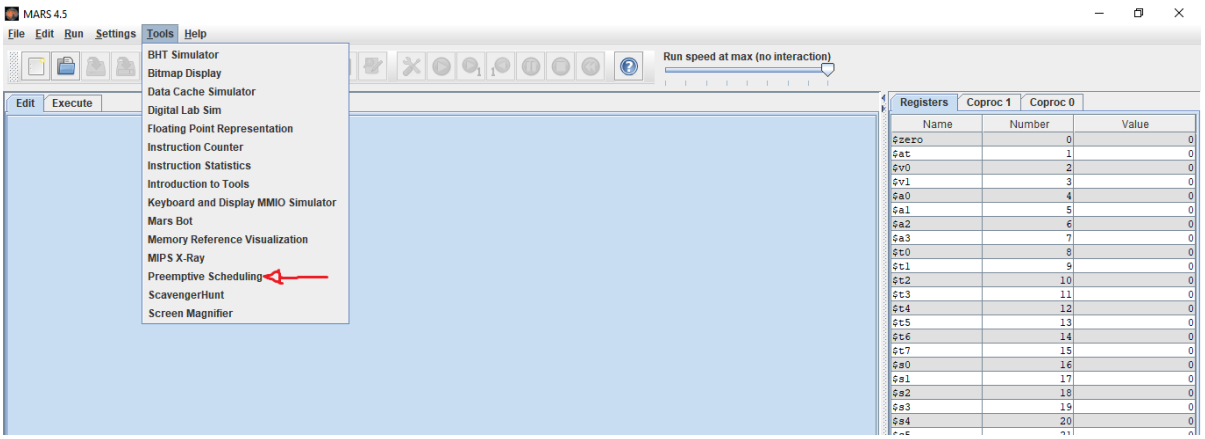
Arquivo macros.asm e o código de teste disponível na tarefa 1.2 se encontram na pasta testeAssembly (arquivos grandes para colocar prints)

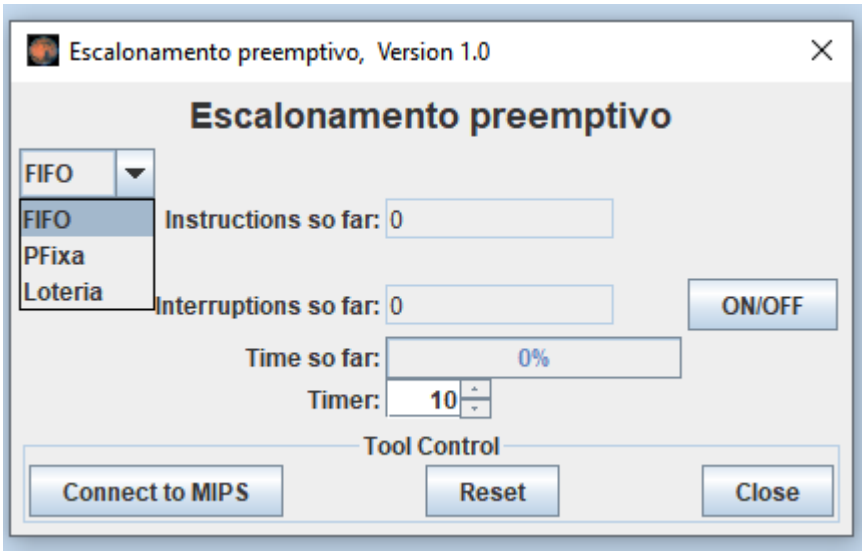


Tarefa 1.3 - Criação das Syscalls do gerenciador de Processos

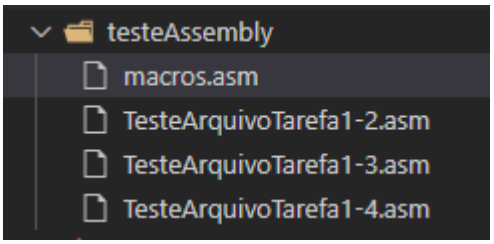
Parte 1 - Escalonamento Preemptivo

Criação de ferramenta (tool) do Mars que funcionará com timer:



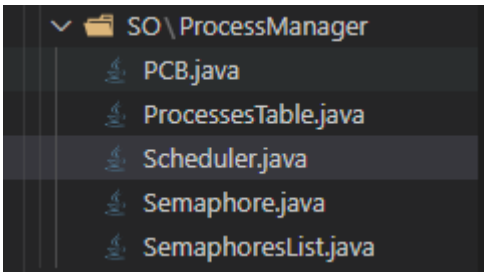


Arquivo de teste disponível na tarefa 1.3 se encontram na pasta testeAssembly (arquivo grande para colocar prints)



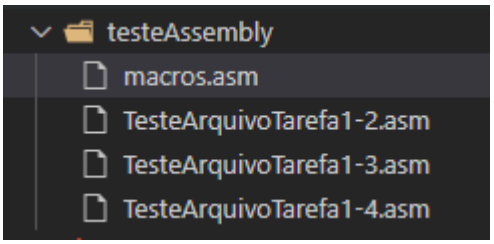
Parte 2 - Novos Algoritmos de escalonamento

Novos algoritmos foram criadas na classe Scheduler




```java
private void prioridadeFixa() {
    criarFilasDePrioridade();
    PCB maiorPrioridade = obterProcessoPrioritario(altaPrioridade);

    if(maiorPrioridade == null) {
        maiorPrioridade = obterProcessoPrioritario(mediaPrioridade);

        if(maiorPrioridade == null) {
            maiorPrioridade = obterProcessoPrioritario(baixaPrioridade);
        }
    }

    int indexProcessoPrioritario = ProcessesTable.getListaProcessos().indexOf(maiorPrioridade);
    trocarPosicoesDaLista(indexProcessoPrioritario);
}

// Funções loteria.
private void loteria() {
    Random random = new Random();
    int valorAleatorio = random.nextInt(ProcessesTable.getTamanhoLista()-1)+1;

    trocarPosicoesDaLista(valorAleatorio);
}
```

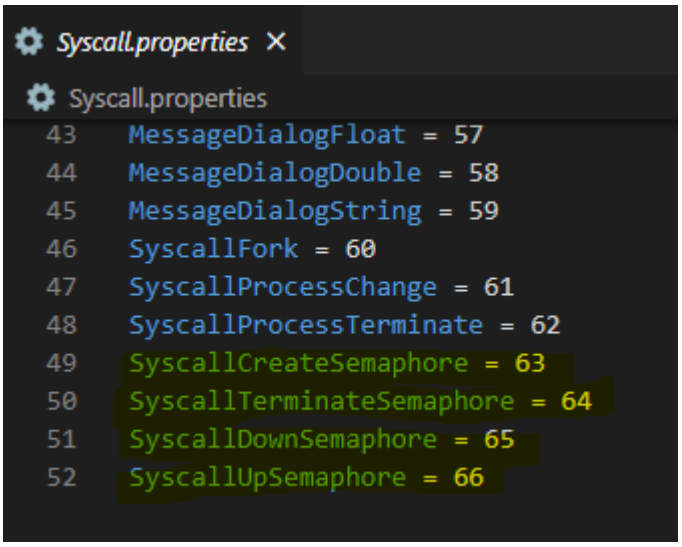
Arquivo de teste disponível na tarefa 1.3 se encontram na pasta testeAssembly (arquivo grande para colocar prints)



Tarefa 1.4 - Sincronização de processos por Semáforos

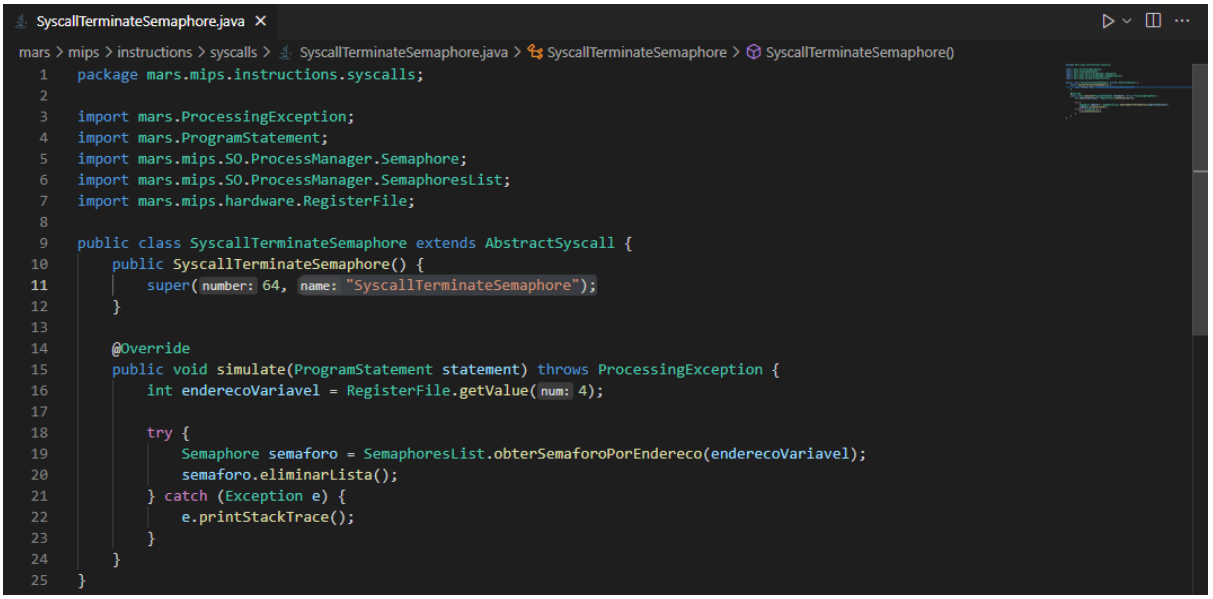
Syscalls foram criadas no pacote “mars.mips.instructions.syscalls”

Cada uma das novas Syscalls ganhou seu próprio número no arquivo “Syscall.properties” (63, 64, 65 e 66)

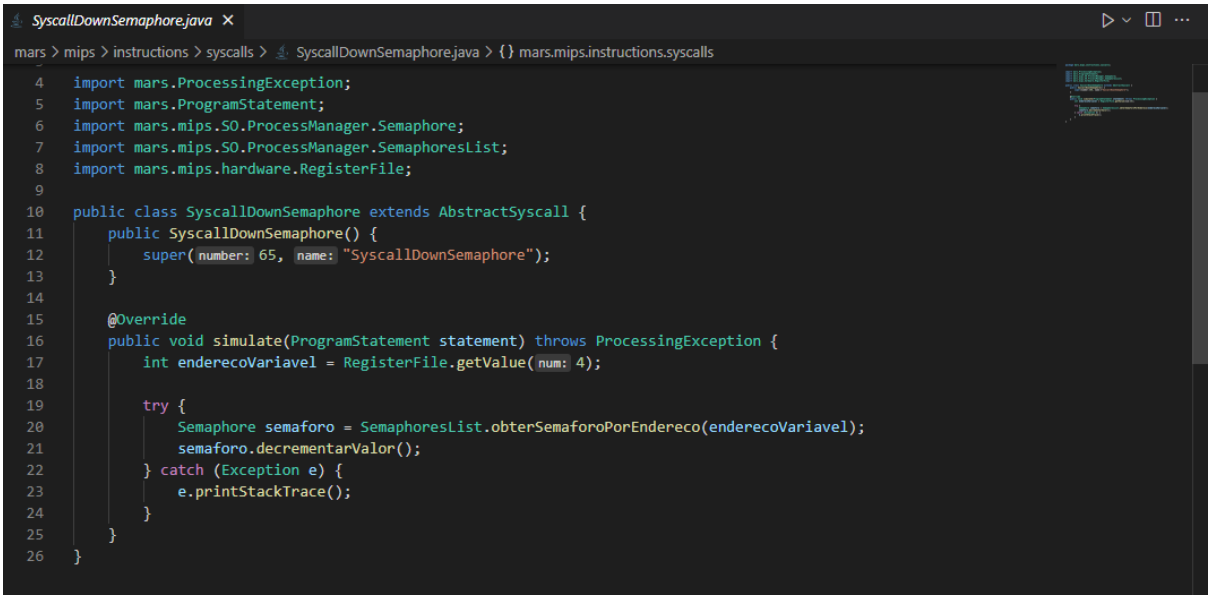


Primeira syscall - CreateSemaphore:

Segunda syscall - TerminateSemaphore:



Terceira syscall - DownSemaphore:



Quarta syscall - UpSemaphore:

SyscallUpSemaphore.java

mars > mips > instructions > syscalls > SyscallUpSemaphore.java > {} mars.mips.instructions.syscalls

```java
package mars.mips.instructions.syscalls;

import mars.ProcessingException;
import mars.ProgramStatement;
import mars.mips.SO.ProcessManager.Semaphore;
import mars.mips.SO.ProcessManager.SemaphoresList;
import mars.mips.hardware.RegisterFile;

public class SyscallUpSemaphore extends AbstractSyscall {
    public SyscallUpSemaphore() {
        super(number: 66, name: "SyscallUpSemaphore");
    }

    @Override
    public void simulate(ProgramStatement statement) throws ProcessingException {
        int enderecoVariavel = RegisterFile.getValue(num: 4);

        try {
            Semaphore semaforo = SemaphoresList.obterSemaforoPorEndereco(enderecoVariavel);
            semaforo.incrementarValor();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Arquivo de teste disponível na tarefa 1.4 se encontram na pasta testeAssembly (arquivo grande para colocar prints)

Validar Semáforo: