Centro Federal de Educação Tecnológica



Simulador de Algoritmo Tomasulo

Por: Rogerio Pereira

**Índice**

Diagrama de classes …………………………………………………………….…...2

Pseudocódigo…………………………………………………………………….…..2

Características …………………………………………………………………….....5

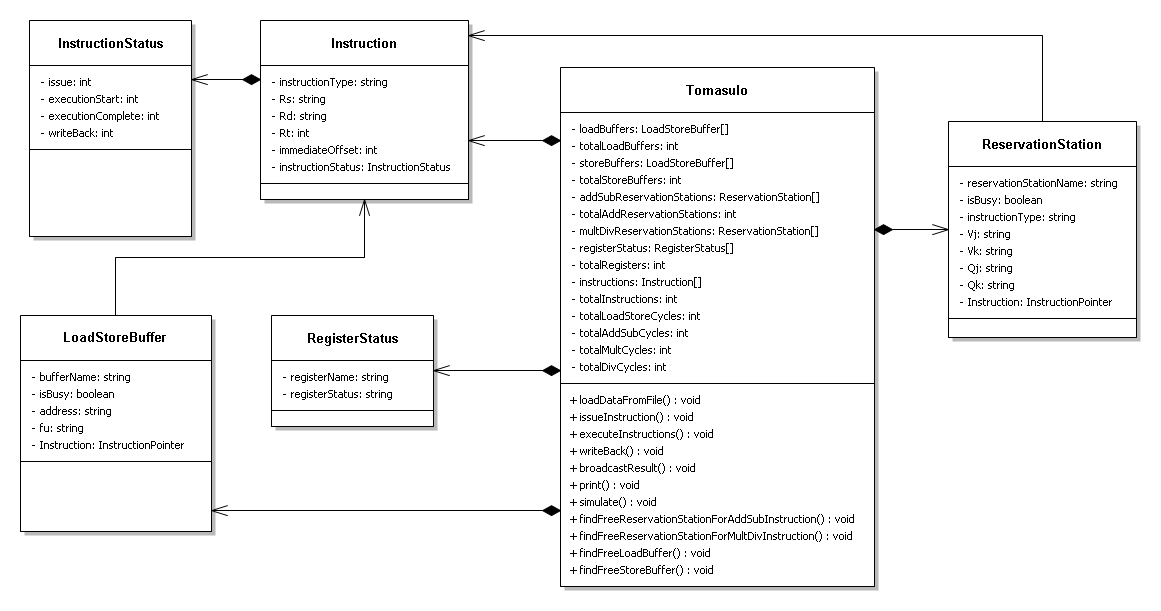
O que pode ser adicionado posteriormente…......................………………….…..….5

Limitações ………………………………………………………………..............….5

Ferramentas usadas………………………………………………………….....…......6

Captura de tela da simulação…………...………………………………………….…6

1. **Diagrama de Classe:**



1. **Pseudo-código:**

**2.1 Função Simular( ):**

atualCycleNumber curto = 0;

while (há uma instrução restante)

{

printInformationOnScreen();

waitForUserToPressAKey ( );

númeroCicloAtual++;

emitirInstrução ( );

executeInstruções ( );

writeBackInstructions ( );

}

**2.2 As funções necessárias usadas pela função Simulate( ):**

emissãoInstrução ( )

{

/\*apenas uma instrução pode ser emitida por vez.

A emissão de instruções estará em ordem\*/

if ( Estações de reserva obrigatórias /buffers estão ocupados)

{

retornar ; //não emite a instrução atual porque existe um risco estrutural

}

else

{

1. Leia os valores dos registros de origem Rs e Rt nos campos Vj e Vk respectivamente.
2. Se algum(s) registro(s) de origem não estiver(ão) disponível(s). Escreva o nome da Estação/Buffer de reserva, que produzirá o resultado para aquele registrador, em Qj / Qk . //O perigo RAW é tratado.
3. Na tabela Register Status registre a informação que esta estação/buffer de reserva irá gravar o resultado no registrador de destino Rd.
4. Marcar a estação/buffer de reserva como ocupado.
5. Defina IssuedInstruction.issued = currentCycleNumber .
6. currentInstructionToBeIssued ++ ; (Esta variável inteira é contador de instruções )

}

} //fim da função IssueInstruction ( )

executeInstruções ( )

{

//Tomar cuidado para não executar a instrução emitida no Ciclo atual

Para Cada instrução emitida

{

1. se todos os valores do registro de origem estiverem disponíveis, execute a instrução.( restingExecutionCycles --); caso contrário, continue a próxima iteração do loop.
2. Se este for o primeiro ciclo de execução da instrução, defina Instruction.ExecutionStartedOn = currentCycleNumber .
3. Se este for o último ciclo de execução da instrução, defina Instructin.ExecutionFinishedOn = currentCycleNumber .

} //fim do loop

} //fim da função executeInstructions ( )

writeBackInstructions ( )

{

//Tome cuidado para não escrever de volta a instrução que completou a execução no ciclo atual

Para Cada instrução emitida

{

1. se a instrução completou seus ciclos de execução ( Instrution.remainingExecutionCycles = 0 ). vá para a linha 2, caso contrário, continue a próxima iteração do loop.
2. Defina Instrução.writeBack = currentCycleNumber .
3. Transmitir os resultados para estações/buffers de reserva dependentes
4. Marque a estação/buffer de reserva como gratuita.

} //fim do loop

} //fim da função writeBackInstructions ( )

1. **Características:**
2. Embora o conjunto de instruções atual seja limitado, as instruções podem ocorrer em qualquer ordem e não há limitação no número de instruções que podem ser executadas pelo simulador.
3. Você pode aumentar/diminuir o número de buffers, bem como de estações de reserva.
4. Você pode especificar o número de ciclos que cada tipo de instrução leva em seu estágio de execução.
5. O número de registros pode ser aumentado ou diminuído.
6. **O que pode ser adicionado ainda:**
7. O programa pode ser modificado para simular o algoritmo de pontuação.
8. O buffer de reordenação pode ser adicionado.
9. O conjunto de instruções pode ser aprimorado para acumular mais instruções
10. A previsão de ramificação pode ser adicionada (pode exigir muitas alterações)
11. **Limitações:**
12. A saída pode não parecer boa abaixo da resolução 1366\*768; a saída definitivamente não aparece intacta em resoluções abaixo de 1024\*768.
13. O programa só pode ser executado na plataforma Windows devido à sua dependência da biblioteca Windows.h .
14. O programa não funciona de forma eficiente. (Por exemplo, ao executar e escrever instruções, o programa visita cada estação/buffer de reserva). Com algumas modificações, essa ineficiência pode ser eliminada.
15. O conjunto de instruções atual é limitado a:

Instruções ADDD, STORE, LOAD, SUB e MULT.

1. **Ferramentas usadas:**
2. Para diagrama de classes, foi utilizado NClass .
3. O programa de simulação foi escrito em C++.
4. O IDE utilizado para programação foi o Visual Studio Code juntamente com o Code Blocks.
5. Microsoft Word foi utilizado para desenvolver esta documentação

**8. \_\_ Captura de tela da simulação:**

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente