UNIOESTE - UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ

CENTRO DE ENGENHARIAS E CIÊNCIAS EXATAS

CURSO DE CIÊNCIA DA COMPUTAÇÃO

**Relatório do trabalho 4**

Bruna Luiza Stefanelli Eyng

Guilherme Silva Altarugio

Pedro Henrique Correa Kraus

FOZ DO IGUAÇU

2024

**1.DESCRIÇAO DO CÓDIGO**

O código define uma classe contendo as informações do pipeline, onde é possível encontrar a memória e os registradores, assim como cada estágio do pipeline e os registradores temporários onde são armazenadas as informações para não serem perdidas. Inicialmente, os registradores e a memória são inicializados. Em seguida, é criada uma lista de instruções que serão executadas.

O ciclo começa com o valor 0 e é inicializado um loop para processar todas as instruções que se encontram nos estágios do pipeline (IF, ID, EX, MEM, WB). No final de cada ciclo, os registradores temporários são atualizados com os valores processados.

Cada estágio é processado por uma função específica:

* IF (Instruction Fetch): Este estágio incrementa o contador de programa (PC) para buscar a próxima instrução a ser executada e armazena essa instrução no registrador temporário nextIFID.
* ID (Instruction Decode): Dependendo do tipo de instrução, os sinais de controle são ativados para configurar o próximo estágio. O conteúdo do estágio é então impresso.
* EX (Execute): Neste estágio, as operações aritméticas e lógicas são realizadas com base nos sinais de controle e nas instruções recebidas. Os resultados são armazenados e preparados para o próximo estágio.
* MEM (Memory Access): Dependendo da instrução, pode haver leitura ou escrita na memória. As informações são atualizadas nos registradores conforme necessário.
* WB (Write Back): Este estágio escreve o resultado no registrador apropriado e atualiza o sinal de controle MemtoReg.

Durante cada ciclo, o código espera que o usuário pressione Enter para continuar permitindo a inspeção do estado do pipeline após cada etapa. No final da execução, os valores dos registradores e da memória são impressos, mostrando o resultado do processamento das instruções.

Assim, o código implementa um pipeline de cinco estágios que simula a execução de um conjunto de instruções, permitindo observar como as instruções são processadas e como os dados fluem através do pipeline.

# **2. INSTRUÇÕES QUE SÃO EXECUTADAS PELO PIPELINE:**

new Inst("I", 19, 3, 0, 4, 0, 0, 100), // addi x4, x1, 100

new Inst("R", 0, 3, 1, 3, 0, 0, 0), // add x3, x3, x1

new Inst("R", 2, 3, 1, 3, 0, 0, 0), // sub x3, x3, x1

new Inst("l", 3, 6, 0, 7, 0, 0, 20), // lw x7, 20(x6)

new Inst("S", 19, 4, 5, 0, 0, 0, 0)); // sw x5, 0(x4)

Texto

Descrição gerada automaticamente

# **3.Instruções para uso do programa:**

Após compilado o código fonte, é preciso usar o enter para da continuação, e as instruções serão executadas a capa etapa, mostrando o ciclo em que a instrução está, os registradores atualizados, o estágio, os sinais de controle e a memória e como estão os outros estágios neste mesmo ciclo, e em seguida ao pressionar enter novamente é atualizado para o próximo ciclo.