Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

DCC819 - Arquitetura de Computadores

Relatório V - Pipeline

Iuri Silva Castro
João Mateus de Freitas Veneroso
Ricardo Pagoto Marinho

Belo Horizonte - MG
3 de dezembro de 2017

1 Introdução

Este documento descreve a implementação do trabalho prático V da disciplina Organização de Computadores II. O trabalho visou incorporar uma *Pipeline* de três estágios sobre o caminho de dados implementado nos trabalhos anteriores. O processador de 16 bits finalizado também faz encaminhamento de dados.

2 Descrição

O *Pipeline* é uma técnica de *hardware* para promover paralelismo à nível de instrução dentro de um único processador. O objetivo da técnica é manter todas os módulos do processador ocupados com alguma instrução pelo máximo de tempo possível. Esse objetivo é realizado por meio da divisão das instruções em múltiplas etapas sequenciais, de forma que diferentes etapas de diferentes instruções possam ser executadas em paralelo como em uma linha de montagem. Essa técnica permite aumentar consideravelmente a velocidade do processador em comparação à execução puramente sequencial, pois várias tarefas podem ser executadas em um mesmo ciclo de *clock*.

No entanto, a técnica de *Pipelining* complexifica o controle do processador, uma vez que a execução paralela introduz *Hazards* no caminho de dados que não existiriam no caso da execução sequencial:

- *Hazards Estruturais* impõem restrições no número de instruções que podem utilizar um módulo do processador ao mesmo tempo. No caso do nosso processador, o caminho de dados possui uma única via, portanto existe apenas um módulo para executar cada etapa da *Pipeline*.
- *Hazards de Dados* forçam que uma instrução dependente de dados de instruções anteriores espere até que os resultados estejam disponíveis antes de ser executada. Caso não haja encaminhamento de dados, o processador é forçado a paralizar a execução de novas instruções por meio de *stalls* até que o dado esteja disponível. Nosso processador implementa o encaminhamento de dados da saída da unidade multiplicadora e da Unidade Lógica Aritmética para evitar o *stall*.
- *Hazards de Controle* acontecem quando existe um desvio de fluxo que altera o *Program Counter* e torna a próxima execução indefinida até que o processador avalie o novo valor do *Program Counter*. Nosso processador introduz um *stall* nos *branches* com o intuito de terminar a avaliação do *Program Counter* antes de prosseguir com a execução da próxima instrução.

O processador desenvolvido até o trabalho prático IV executava instruções de maneira sequencial. Com a introdução do *Pipeline* neste trabalho, obtivemos ganhos significativos na velocidade de execução como será mostrado na seção de experimentos.

3 Implementação

Mostrar implementação de cada módulo. Descrever ISA final do sistema

4 Integração

Mostrar divisão de estágios. Mostrar buffers entre estágios. Mostrar sistema inteiro

5 Simulação e Testes

Descrever processo de simulação e programas de teste. Comparar programas rodando com pipeline e sem (trabalho IV anterior).

6 Discussões

Comentar dos ganhos de desempenho, custos dos *Stalls*, dificuldades de mudança para pipeline.