БЕЛОРУСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информатики

Курсовая работа

по дисциплине: “Архитектура вычислительных систем”

по теме

“Исследование отложенного АЛУ для последовательного процессора.”

Выполнил: студент гр. 853505 Нетецкая Ю.В.

Проверил: руководитель работы Леченко А.В.

  Минск 2020

**Введение**

В данной курсовой работе имеется три основных задачи, а именно:

1. Изучить статью “Exploring Early and Late ALUs for Single-Issue In-Order Pipelines” и разобраться как такой эксперимент провести в симуляторе MIPT-MIPS;
2. Переделать конвейер MIPT-MIPS для проведения эксперимента;
3. Запустить набор тестов на модифицированном конвейере и оригинальном. Сравнить количество симуляционных тактов на выполнение тестов.

**Задача №1.**

В предоставленной статье, рассмотрен вопрос, как лучше всего использовать ресурсы Арифметико-логического устройства в конвейере.

Начинается с анализа наиболее эффективного способа размещения один ALU в рабочем конвейере. Оценивается как наиболее эффективно использовать два ALU, один ранний и один поздний ALU. Покажем, что использование двух ALU на разных этапах трубопровод обеспечивает лучшую производительность и энергоэффективность, чем любой другая конфигурация конвейера с одним АЛУ.

Как результат, в этой работе основное внимание уделяется целочисленному ALU.

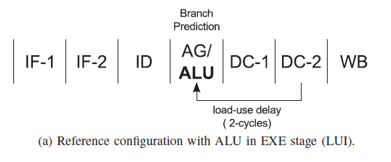
Нам показаны 4 различных конфигурации конвейеров, важными для нас являются только два:

* **эталонный конвейер LUI-типа,** который похож на процессор MIPS R2000
* конвейер, в котором используются **два идентичные ALU** **на разных стадиях**

1. **Эталонный конвейер LUI-типа**.

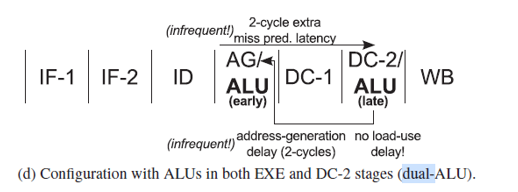
Он похож на процессор MIPS R2000, поскольку доступ к ALU осуществляется непосредственно на этапе выполнения, на котором генерация адреса также обрабатывается.

**В конвейере симулятора MIPT-MIPS блок ALU находятся в стадии выполнения (EXE), поэтому мы выбрали для рассмотрения именно этот пример.**



1. **Конвейер, в котором используются два идентичные ALU на разных стадиях.**

В данном конвейере используются два Арифметико-логических устройства, а именно ранний АЛУ, который находится в стадии выполнения (EXE), и поздний АЛУ, который расположен в ступени data cache access 2,  **(DC-2)**



В этой конфигурации задержка использования нагрузки полностью устраняется. Кроме того, большая часть генерации адресов также устраняются за счет использования раннего ALU.

Преимуществом является то, что АЛУ могут выполнять разные операции АЛУ в одном цикле. Следует отметить, что перенос ALU на более поздние этапы конвейера, требует все меньше и меньше переадресации от этапов впереди.

Конфигурация с двумя ALU сокращает циклы простоя примерно на 83%.

В некоторых случаях перенос ALU на более поздние этапы может вызвать больше циклов остановки из-за увеличения штрафа за переход, поэтому такой метод реализации нам не подходит.

Для **конвейера с двумя ALU улучшение составляет 4,4% для MiBench и 5,0% для SPEC2000int. Если взять во внимание то, что** для конвейера AGI() улучшение составляет 2,8% и 2,4%, а для промежуточного конвейера LUI / AGI время выполнения улучшается на 2,1% и 1,7%, то использование конвейера с двойным АЛУ показывает наивысший результат выполнения.

Проанализировав, можно сделать вывод что для лучшей производительности конвейера, является использование двойного АЛУ.

**Задача №2. П**еределать конвейер MIPT-MIPS для проведения эксперимента.

Конвейере MIPT-MIPS схож с конвейером которым мы рассмотрели ранее, а именно, где доступ к AЛУ осуществляется непосредственно на этапе выполнения.

В функциональном симуляторе все действия разделены на 5 этапов:

* **Fetch**
* **Decode**, read sources
* **Execute**, calculate address
* **Memory** **access**
* **Writeback**, PC update, information dump

Одноцикловый - простейшая реализация архитектуры. В его основе лежат три основных состояния:

* Все операции выполняются строго последовательно
* Выполнение инструкции не начинается до тех пор, пока предыдущая не будет полностью выполнена (без перекрытия)
* Все инструкции занимают одинаковое количество времени - один цикл

Это делает разработку функционального симулятора очень простой. Симулятор будет иметь структуру с внутренним состоянием, автономными инструкциями и одним методом, который будет выполнять инструкции.

**Литература**

<https://github.com/MIPT-ILab/mipt-mips/wiki/Lectures-on-Computer-Architecture-in-2018>

<https://riscv-config.readthedocs.io/en/latest/>:

<http://publications.lib.chalmers.se/records/fulltext/224962/local_224962.pdf>

<https://www.jetbrains.com/help/clion/build-actions.html>

<http://scipro.ru/conf/computerarchitecture.pdf>