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

Содержание

1. Введение

Разработчик компьютера ENIAC, Джон фон Нейман, был первым создателем АЛУ. В 1945 году он опубликовал первые научные работы по новому компьютеру, названному EDVAC (Electronic Discrete Variable Computer).

Годом позже он работал со своими коллегами над разработкой компьютера для Принстонского института новейших исследований (IAS).

Архитектура этого компьютера позже стала прототипом архитектур большинства последующих компьютеров. В своих работах фон Нейман указывал устройства, которые, как он считал, должны присутствовать в компьютерах. Среди этих устройств присутствовало и АЛУ.

Фон Нейман отмечал, что АЛУ необходимо для компьютера, поскольку оно гарантирует, что компьютер будет способен выполнять базовые математические операции включая сложение, вычитание, умножение и деление.

2. Теоретическая часть

Арифметико-логическое устройство (АЛУ) – блок процессора, который под управлением устройства управления служит для выполнения арифметических и логических преобразований (начиная от элементарных) над данными, представляемыми в виде машинных слов, называемыми в этом случае операндами.

Операции АЛУ подразделяются на три основные категории: арифметические, логические и операции над битами. Арифметической операцией называют процедуру обработки данных, аргументы и результат которой являются числами (сложение, вычитание, умножение, деление,...). Логической операцией именуют процедуру, осуществляющую построение сложного высказывания (операции И, ИЛИ, НЕ,...). Операции над битами обычно подразумевают сдвиги.

АЛУ состоит из регистров, сумматора с соответствующими логическими схемами и элемента управления выполняемым процессом. Устройство работает в соответствии с сообщаемыми ему именами (кодами) операций, которые при пересылке данных нужно выполнить над переменными, помещаемыми в регистры.

Арифметико-логическое устройство функционально можно разделить на две части:

1. микропрограммное устройство (устройство управления), задающее последовательность микрокоманд (команд);
2. операционное устройство (АЛУ), в котором реализуется заданная последовательность микрокоманд (команд).

RISC-V— это свободная архитектура набора команд. Проект зародился в Калифорнийском университете в Беркли в 2010 году. Важную роль в его успехе сыграла открытость кода и свобода использования, что резко отличалось от многих других архитектур.

Все современные архитектуры считаются архитектурами RISC. Для этого есть много причин:

- Инструкции в процессоре RISC проще, внутри микросхемы больше места для регистров. Регистры - это самый быстрый тип памяти, и в конечном итоге все инструкции должны выполняться для значений,

хранящихся в регистрах, поэтому большее количество регистров приведет к более высокой производительности.

- Поскольку все инструкции выполняются одновременно, конвейерная обработка возможна.

Недостаток заключается в том, что RISC усложняет программирование на ассемблере, поскольку почти все программисты используют языки высокого уровня и оставляют тяжелую работу по генерации кода ассемблера компилятору.

MIPS – это сокращение от Millions of Instructions Per Second — «миллионы команд в секунду. Этот предполагает, что параллелизм может играть главную роль в улучшении производительности, поскольку приступать к большому количеству команд за короткий промежуток времени можно только в том случае, если одновременно может выполняться несколько команд.

Процессоры MIPS основаны на архитектуре RISC.

Ранние модели процессора имели 32-битную структуру, позднее появились его 64-битные версии. Существует множество модификаций процессора, включая MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, MIPS32 и MIPS64, из них действующими являются MIPS32 (для 32-битной реализации) и MIPS64 (для 64-битной реализации). MIPS32 и MIPS64 определяют как набор регистров управления, так и набор команд.

3. Ход работы.

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.

3.1. Задача №1.

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

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

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

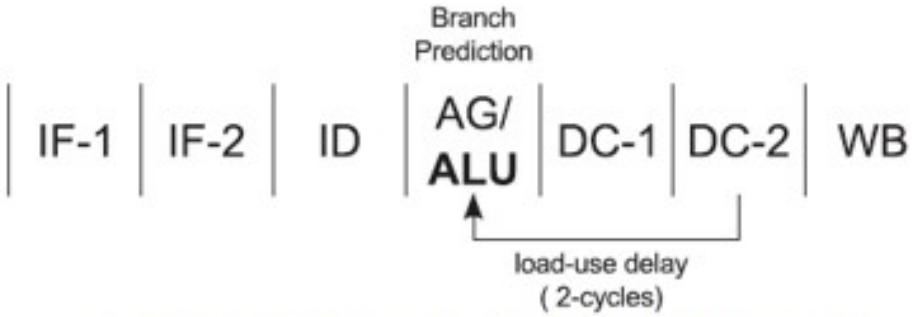
Нам показаны 4 различных конфигурации конвейеров, важными для нас являются только два:

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

1. **Эталонный конвейер LUI-типа**.

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

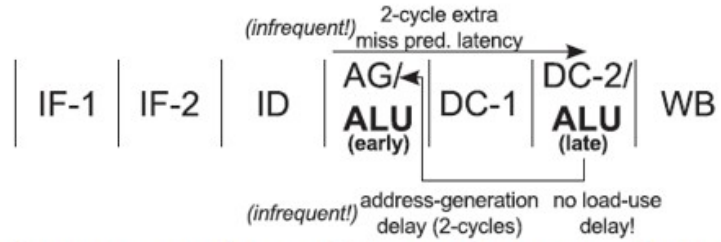
В конвейере симулятора MIPT-MIPS блок АЛУ находятся в стадии выполнения (EXE), поэтому мы выбрали для рассмотрения именно этот пример.



(a) Reference configuration with ALU in EXE stage (LUI).

2. **Конвейер, в котором используются два идентичные АЛУ 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, (DC-2)



(d) Configuration with ALUs in both EXE and DC-2 stages (dual-ALU).

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

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

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

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

По сравнению с конвейером с одним АЛУ, для конвейера с двумя АЛУ 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.

3.2. Задача №2.

Конвейере MIPT-MIPS схож с конвейером которым мы рассмотрели ранее, а именно, где доступ к АЛУ 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.

3.3. Задача 3.

Теперь запустим набор тестов на модифицированном конвейере и оригинальном и сравним количество симуляционных тактов на выполнение тестов.

Тесты собирались с помощью riscv-gnu-toolchain и riscv-tests.

Название теста	Оригинальный конвейер	Модифицированный конвейер
Pmp.riscv	133739	
Multiply.riscv	117315	
Median,riscv	118939	
Vvadd.riscv	119377	
Rsort.riscv	137660	
Towers.riscv	116276	

5. Литература.

1. https://github.com/MIPT-ILab/mipt-mips/wiki/Lectures-on-Computer-Architecture-in-2018
2. https://riscv-config.readthedocs.io/en/latest/:
3. http://publications.lib.chalmers.se/records/fulltext/224962/local_224962.pdf
4. https://github.com/MIPT-ILab/mipt-mips/wiki/RISC-V-binutils
5. https://www.jetbrains.com/help/clion/build-actions.html
6. http://scipro.ru/conf/computerarchitecture.pdf
7. https://habr.com/ru/post/454208/#1
8. http://wiki.mvtom.ru/index.php/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D0%BA%D0%BE-%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE#:~:text=%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0%20%D0%90%D0%9B%D0%A3,-%D0%90%D0%9B%D0%A3%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D0%B8%D1%82%20%D0%B8%D0%B7&text=%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D0%BA%D0%BE%2D%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5%20%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE%20%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%BC%D0%BE%D0%B6%D0%BD%D0%BE,%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F%20%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%20%D0%BC%D0%B8%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%20(%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4)