Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»(ИУ7)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.04 «Программная инженерия»

## О Т Ч Е Т

по лабораторной работе № 2

Название Изучение принципов работы микропроцессорного ядра RISC-V

Дисциплина Архитектура элекронно-вычислительных машин

|  |  |  |
| --- | --- | --- |
| Студент: |  | Золотухин А В. |
| Преподаватель: | подпись, дата | Фамилия, И.О.  Попов А. Ю. |
|  | подпись, дата | Фамилия, И. О. |

Москва — 2022 г.

# Цель работы

Основной целью работы является ознакомление с принципами функциониро- вания, построения и особенностями архитектуры суперскалярных конвейерных микропроцессоров. Дополнительной целью работы является знакомство с прин- ципами проектирования и верификации сложных цифровых устройств с исполь- зованием языка описания аппаратуры SystemVerilog и ПЛИС.

# Основные теоретические сведения

RISC-V является открытым современным набором команд, который может использоваться для построения как микроконтроллеров, так и высокопроизводи- тельных микропроцессоров.

В данной работе исследуется набор команд RV32I, который включает в се- бя основные команды 32-битной целочисленной арифметики кроме умножения и деления.

Набор команд RV32I предполагает использование 32 регистров общего назна- чения x0-x31 размером в 32 бита каждый и регистр pc, хранящего адрес следу- ющей команды. Все регистры общего назначения равноправны, в любой команде могут использоваться любые из регистров. Регистр pc не может использоваться в командах.

Архитектура RV32I предполагает плоское линейное 32-х битное адресное про- странство. Минимальной адресуемой единицей информации является 1 байт. Ис- пользуется порядок байтов от младшего к старшему (Little Endian), то есть, млад- ший байт 32-х битного слова находится по младшему адресу (по смещению 0). От- сутствует разделение на адресные пространства команд, данных и ввода-вывода. Распределение областей памяти между различными устройствами (ОЗУ, ПЗУ, устройства ввода-вывода) определяется реализацией.

Большая часть команд RV32I является трехадресными, выполняющими опе- рации над двумя заданными явно операндами, и сохраняющими результат в реги- стре. Операндами могут являться регистры или константы, явно заданные в коде команды. Операнды всех команд (кроме команды auipc) задаются явно.

Архитектура RV32I, как и большая часть RISC-архитектур, предполагает раз- деление команд на команды доступа к памяти (чтение данных из памяти в ре- гистр или запись данных из регистра в память) и команды обработки данных в регистрах.

# Общая для всех вариантов программа

## Исследуемая программа

Исходный текст исследуемой программы представлен на рисунке [1.](#_bookmark0)

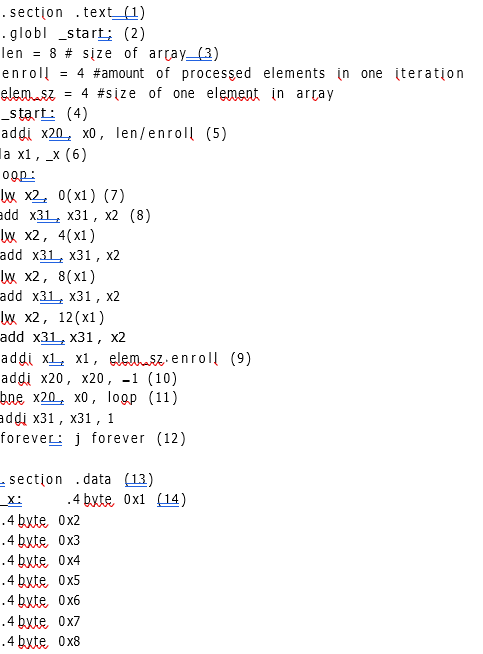


Рисунок 1  Исходный код программы

Дизассемблерный листинг исследуемой программы представлен на рисунке 2

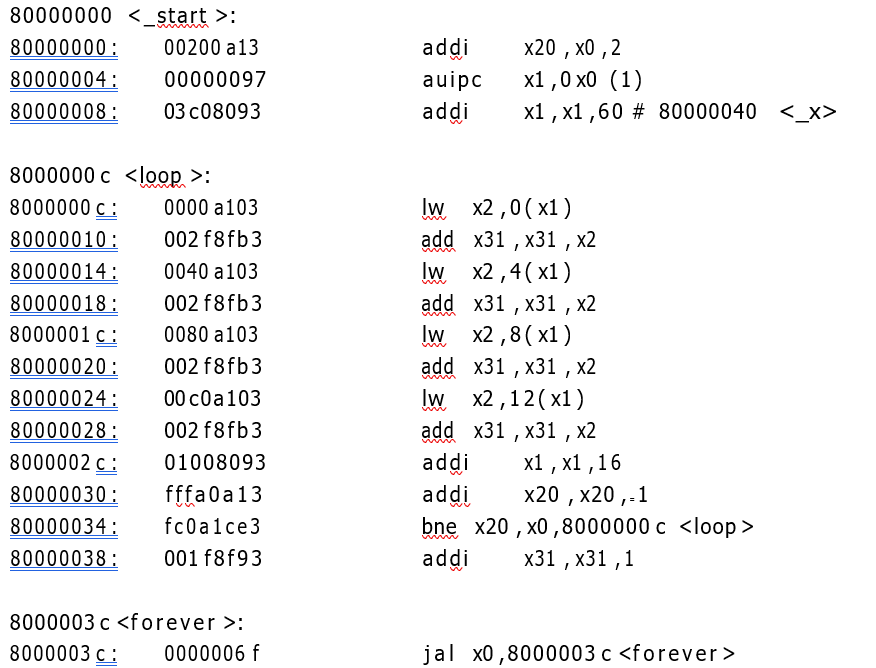


Рисунок 2  Дизассемблированный код программы

Можно сказать, что данная программа эквивалентна псевдокоду на языке C, представленному на рисунке [3.](#_bookmark2)

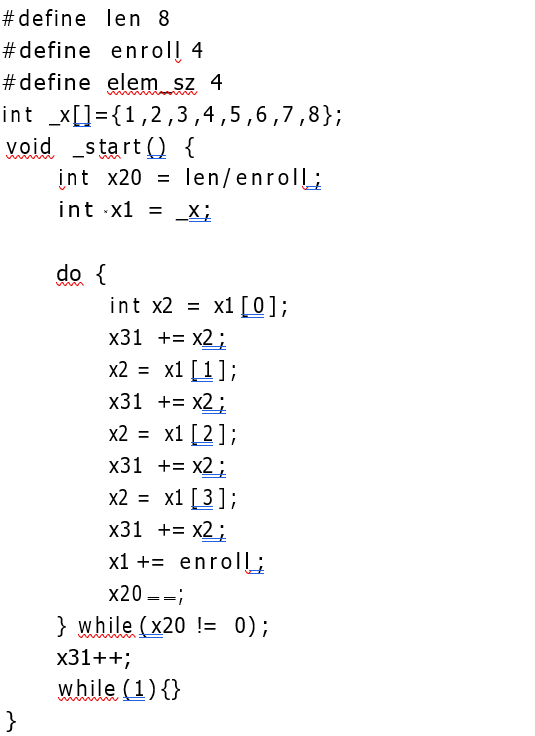


Рисунок 3  Псевдокод программы

# Результаты исследования программы

Все задания выполнялись по индивидуальному варианту (5).

Скриншот, полученный в ходе выполнения задания №2 (получить снимок экра- на, содержащий временную диаграмму выполнения стадий выборки и диспетче- ризации команды с адресом 8000001c на первой итерации) представлен на рисунке [4.](#_bookmark3)

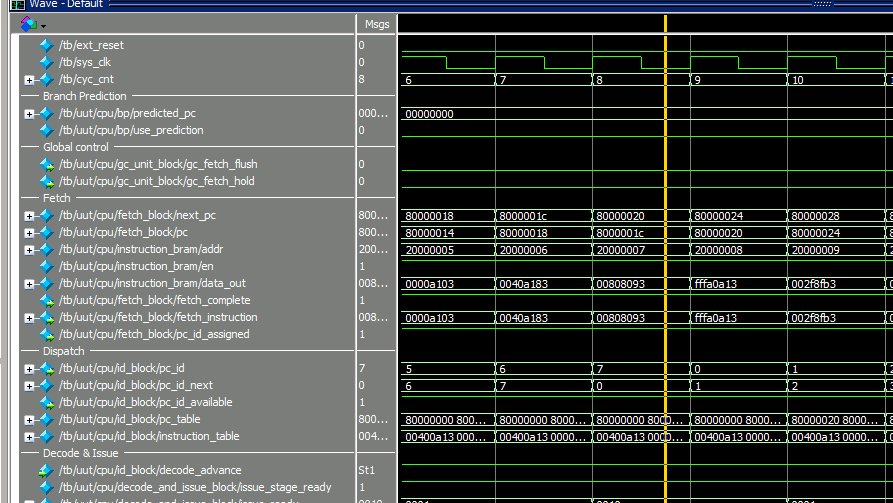


Рисунок 4 – Временная диаграмма выполнения стадий выборки и диспетчеризации

Скриншот, полученный в ходе выполнения задания №3 (получить снимок экра- на, содержащий временную диаграмму выполнения стадии декодирования и пла- нирования на выполнение команды с адресом 80000028 на первой итерации) пред- ставлен на рисунке [5.](#_bookmark4)

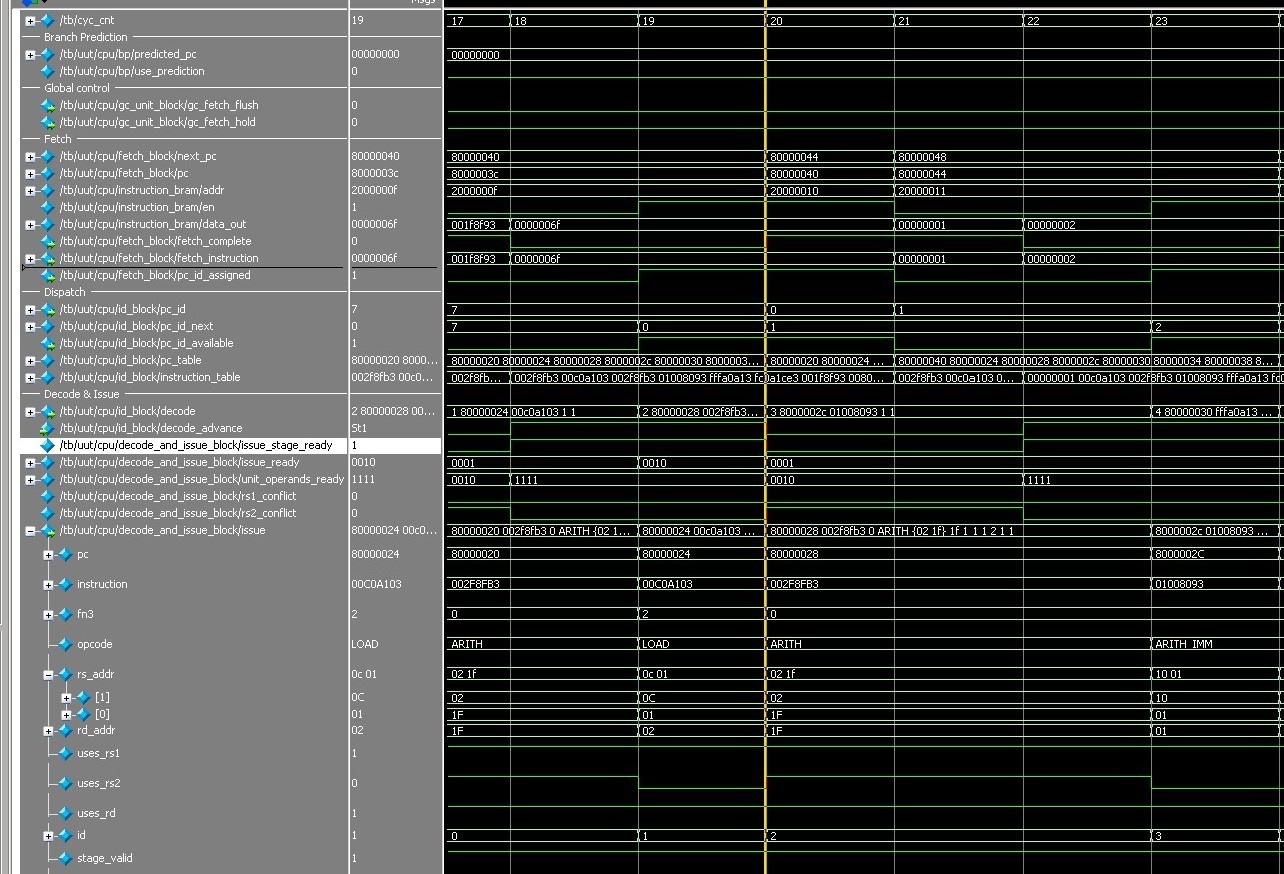


Рисунок 5 – Временная диаграмма выполнения стадии декодирования и планирования на выполнение

Скриншот, полученный в ходе выполнения задания №4 (получить снимок экра- на, содержащий временную диаграмму выполнения стадии выполнения команды с адресом 80000010 на первой итерации) представлен на рисунке [6.](#_bookmark5)

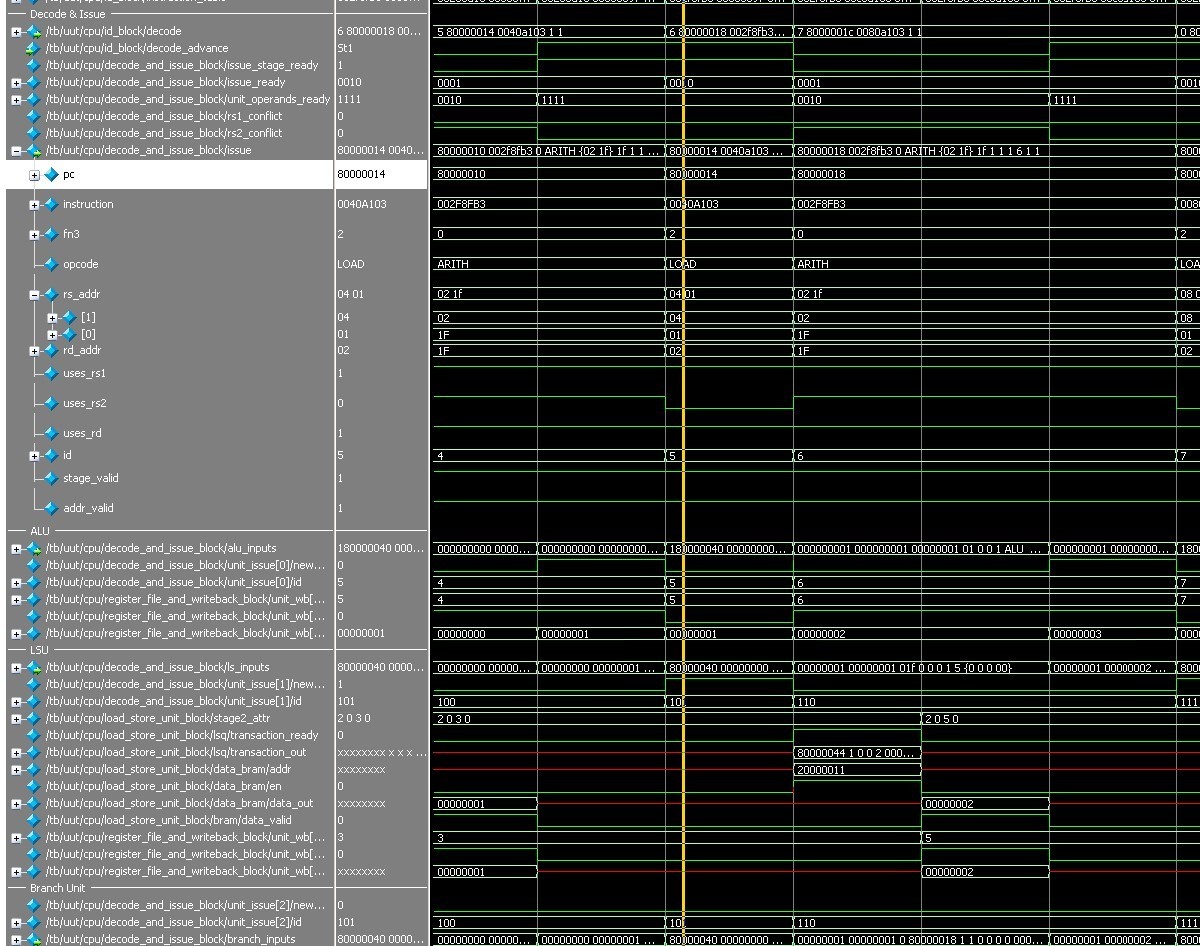


Рисунок 6 – Временная диаграмма выполнения стадии выполнения

# Программа по варианту

Все задания выполнялись по индивидуальному варианту (5).

## Исследуемая программа

Исходный текст исследуемой программы представлен на рисунке [7.](#_bookmark6)

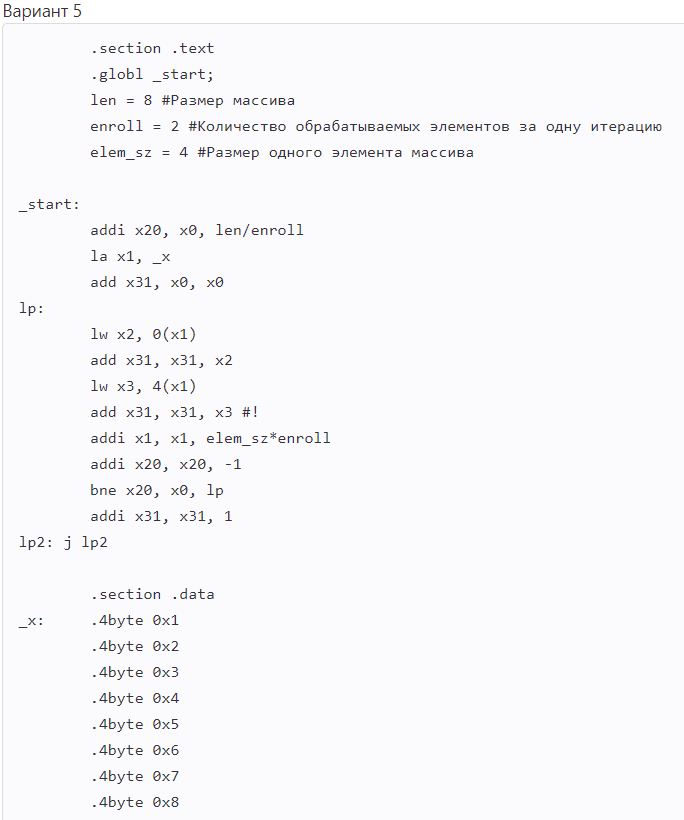


Рисунок 7 – Исходный текст исследуемой программы

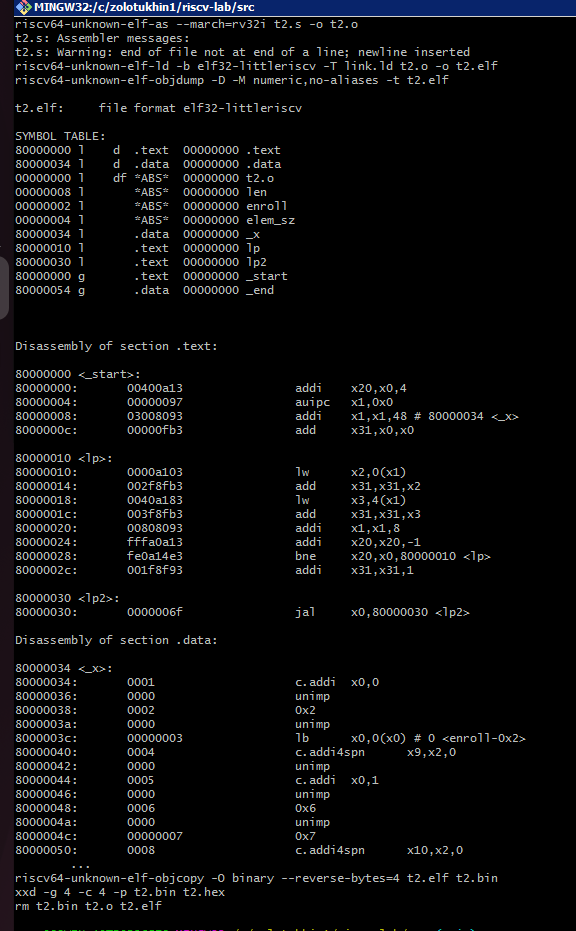
Дизассемблерный листинг исследуемой программы представлен на рисунке [8.](#_bookmark7) 

Рисунок 8 – Дизассемблерный листинг исследуемой программы

## Трасса работы программы

Трасса работы программы представлена на рисунке [9.](#_bookmark9)

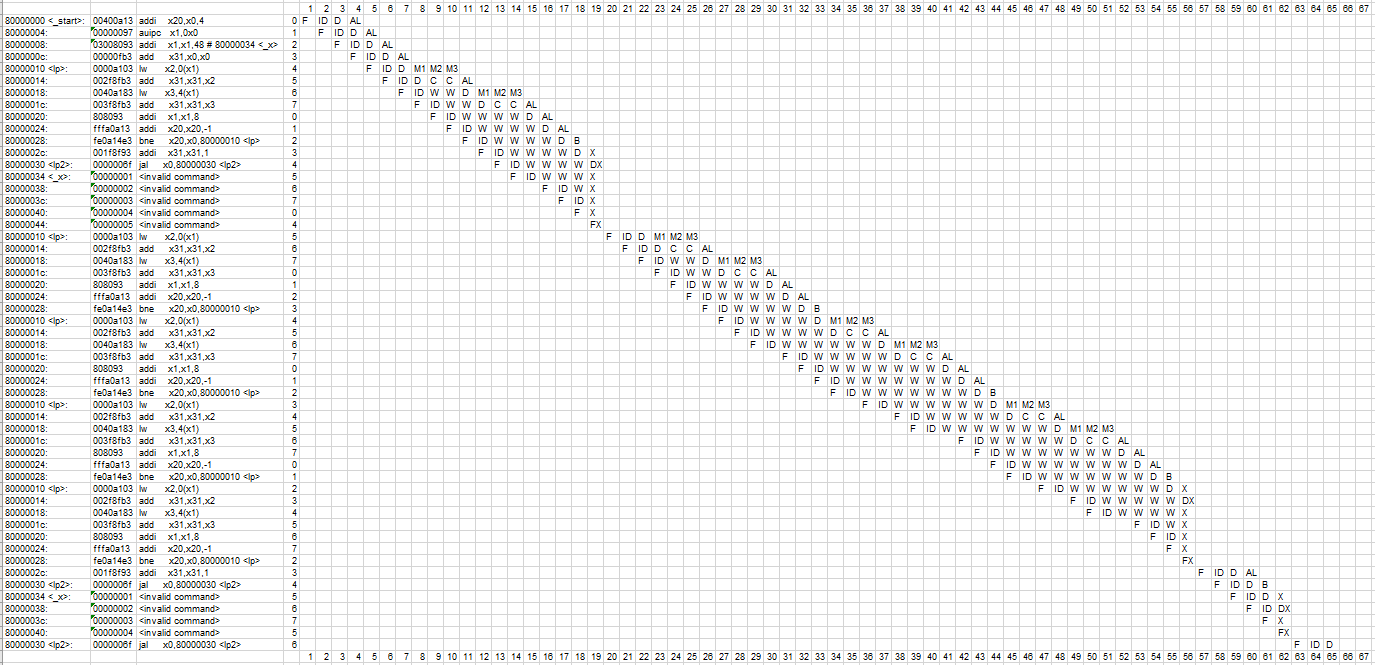


Рисунок 9 – Трасса работы программы

## Временные диаграммы

Временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом #! (add x31, x31, x3), представлены на рисунках [10, 11, 12.](#_bookmark10)

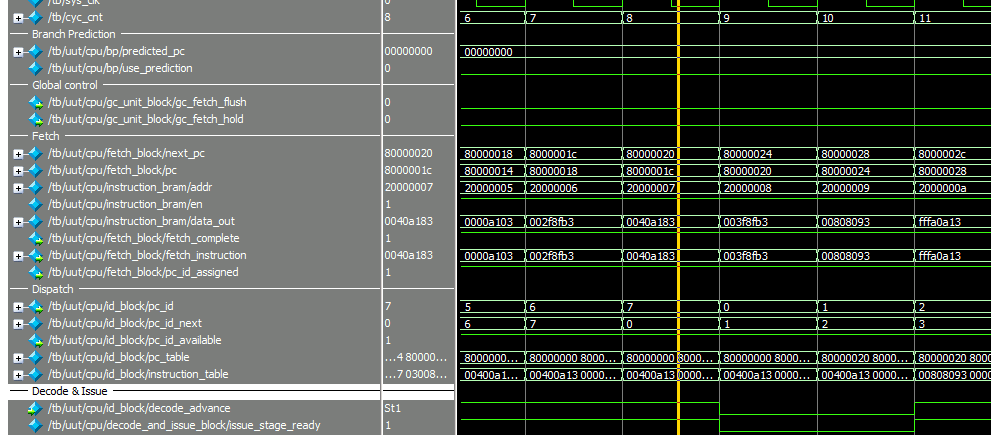


Рисунок 10 – Временная диаграмма стадии выборки и диспетчеризации

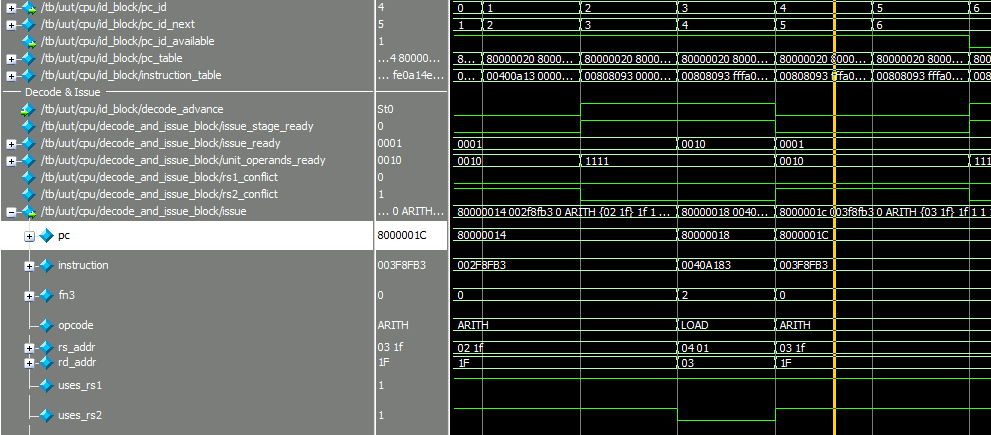


Рисунок 11 – Временная диаграмма стадии декодирования

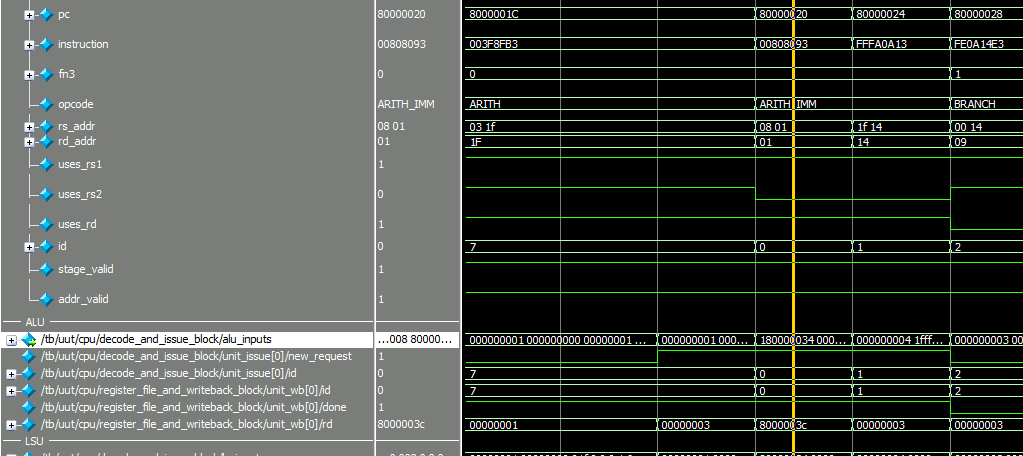


Рисунок 12 – Временная диаграмма стадии выполнения

## Вывод об эффективности работы программы

Как видно на трассе работы программы, представленой на рисунке [9,](#_bookmark9) конфлик- ты по регистрам возникают.

Оптимизировать же программу можно, если сначала производить все команды загрузки, затем выполнять команды, обслуживающие цикл и в конце команды обрабатывающие данные.

## Оптимизированная программа

Исходный текст оптимизированной программы представлен на рисунке [13.](#_bookmark11)

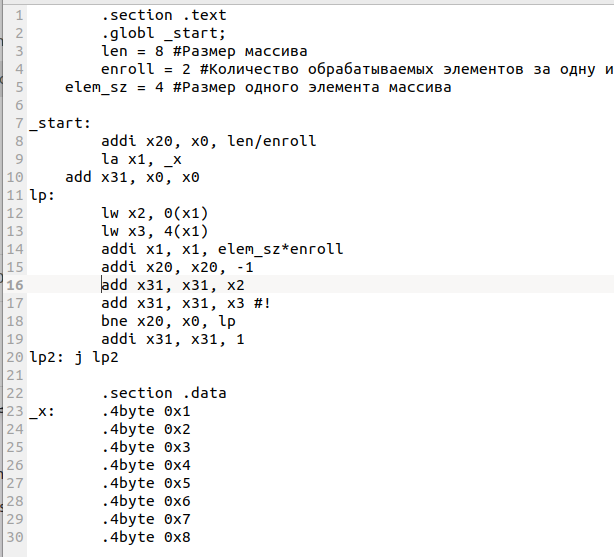


Рисунок 13 — Код оптимизированной программы

Дизассемблерный листинг оптимизированной программы представлен на рисунке 14[.](#_bookmark12)

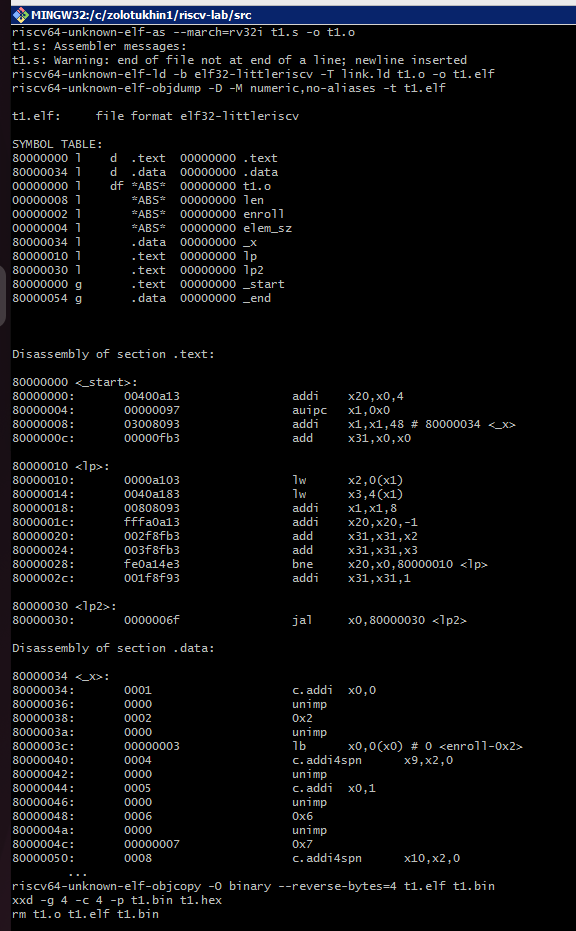


Рисунок 14 – Дизассемблированный листинг программы.

## Трасса работы программы

Трасса работы оптимизированной программы представлена на рисунке [15.](#_bookmark13)

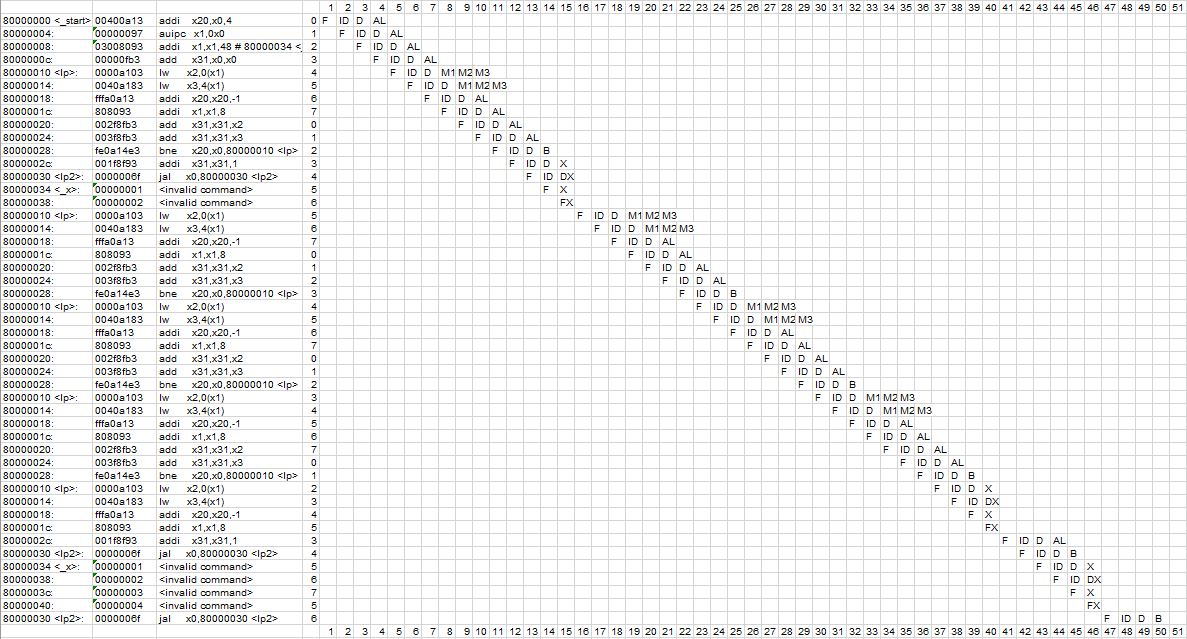


Рисунок 15 – Трасса работы оптимизированной программы

# Вывод

В результате выполнения лабораторной работы были изучены принципы функ- ционирования, построения и особенности архитектуры суперскалярных конвейер- ных микропроцессоров.

Также были рассмотрены принципы проектирования и верификации сложных цифровых устройств с использованием языка описания аппаратуры SystemVerilog и ПЛИС.

На основе изученных материалов был найден способ оптимизации программы.