|  |  |
| --- | --- |
| Gerb-BMSTU_01**00** | **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**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе № 2** |  |

**Название:**

Изучение принципов работы микропроцессорного ядра RISCV

**Дисциплина:** Архитектура ЭВМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-51Б |  |  | Костев Д.И. |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Дубровин Е.Н. |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2021

Цели работы

Основной целью работы является ознакомление с принципами функционирования, построения и особенностями архитектуры суперскалярных конвейерных микропроцессоров. Дополнительной целью работы является знакомство с принципами проектирования и верификации сложных цифровых устройств с использованием языка описания аппаратуры SystemVerilog и ПЛИС.

Для достижения поставленных целей в настоящей лабораторной работе используется синтезируемое описание микропроцессорного ядра Taiga, реализующего систему команд RV32I семейства RISC­V. Данное описание выполнено на языке описания аппаратуры SystemVerilog.

RISC-V является открытым современным набором команд, который может использоваться для построения как микроконтроллеров, так и высокопроизводительных микропроцессоров. В связи с такой широкой областью применения в систему команд введена вариативность. Таким образом, термин RISC-V фактически является названием для семейства различных систем команд, которые строятся вокруг базового набора команд, путем внесения в него различных расширений.

В данной работе исследуется набор команд RV32I, который включает в себя основные команды 32-битной целочисленной арифметики кроме умножения и деления. В рамках данного набора команд мы не будем рассматривать системные команды, связанные с таймерами, системными регистрами, управлением привилегиями, прерываниями и исключениями.

## Задание №1

Скомпилированная тестовая программа:

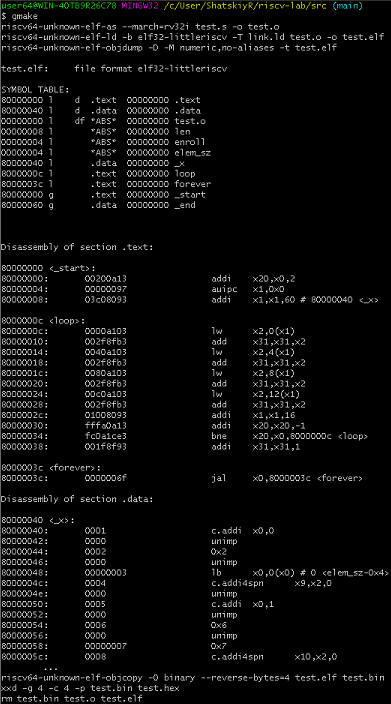


Рисунок 1. Скомпилированная программа test

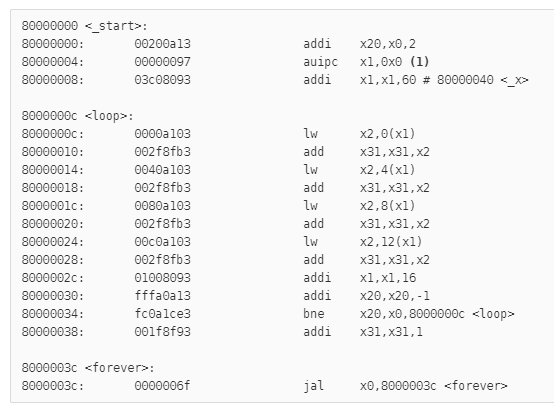


Рисунок 2. Код тестовой программа

Рассматриваемый вариант – 14

Ниже приведен листинг программы этого варианта.

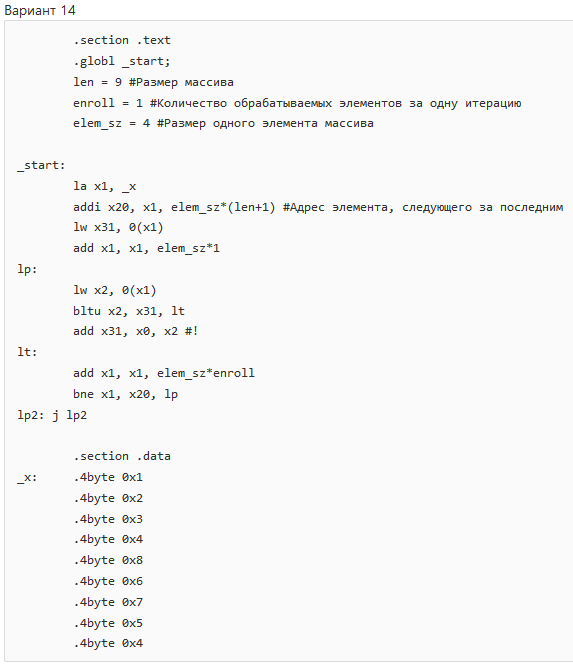


Рисунок 3. Код программы в.14

Код на языке C, соответствующий данной программе:

#include <stdio.h>

#include <stdlib.h>

#define len 9

#define enroll 1

#define elem\_sz 4

int \_x[] = {1, 2, 3, 4, 8, 6, 7, 5, 4};

int main() {

int \*x20 = \_x + len + 1;

int \*x1 = \_x;

int x31 = x1[0];

x1 += 1;

do {

int x2 = x1[0];

if (x2 >= x31) {

x31 = x2;

}

x1 += enroll;

} while(x1 != x20);

printf("%d\n", x31);

while (1) {};

return 0;

}

В x31 в конце программы будет находиться число 8.

Ниже приведен дизассемблерный листинг программы.

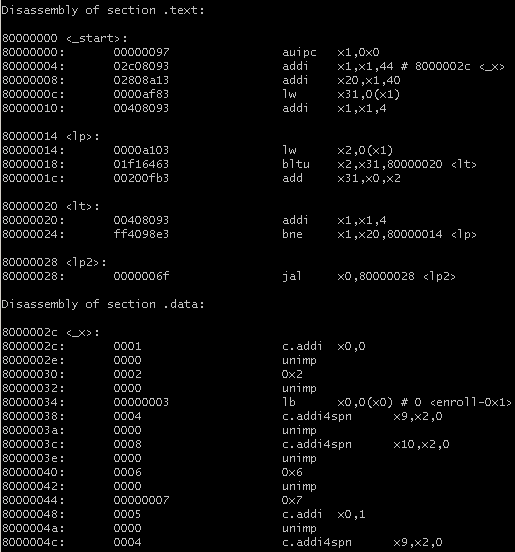


Рисунок 4. Скомпилированная программа

## Задание 2

Адрес команды: 80000014

Номер итерации: 2

Выборка и диспетчеризация.

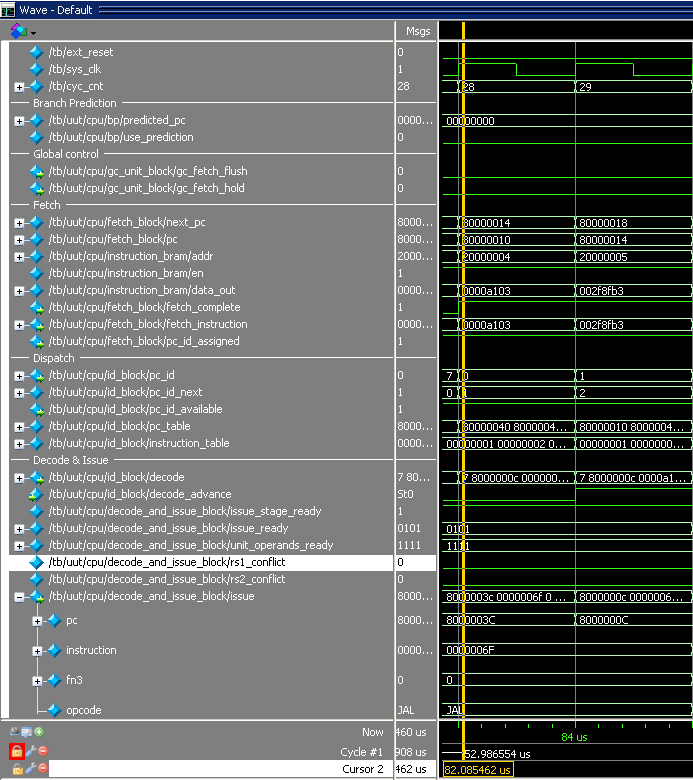


Рисунок 5. Выборка и диспетчеризация

## Задание 3

Адрес команды: 80000020

Номер итерации: 2

Декодирование и планирование.

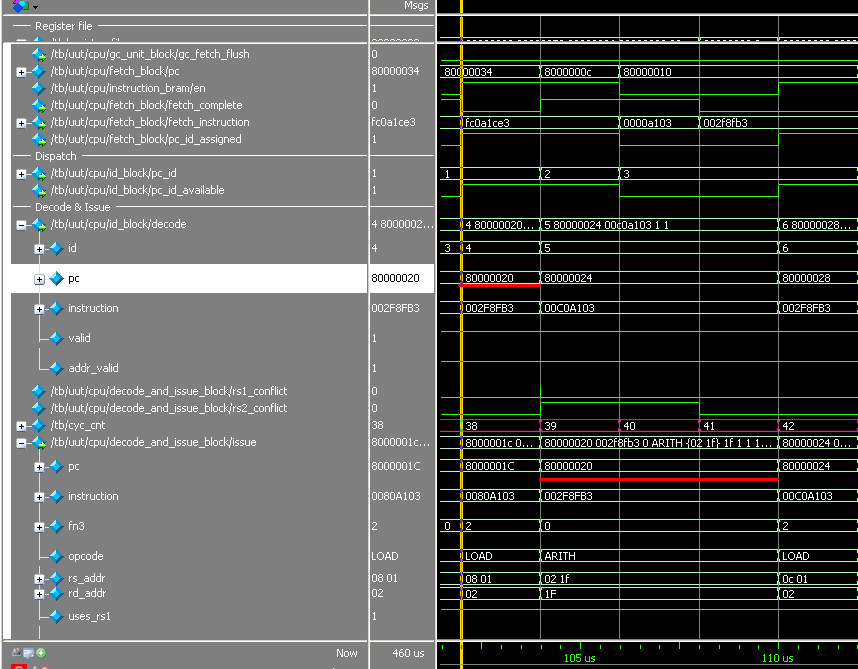


Рисунок 6. Декодирование и планирование

## Задание 4

Адрес команды: 8000000c

Номер итерации: 2

Выполнение.

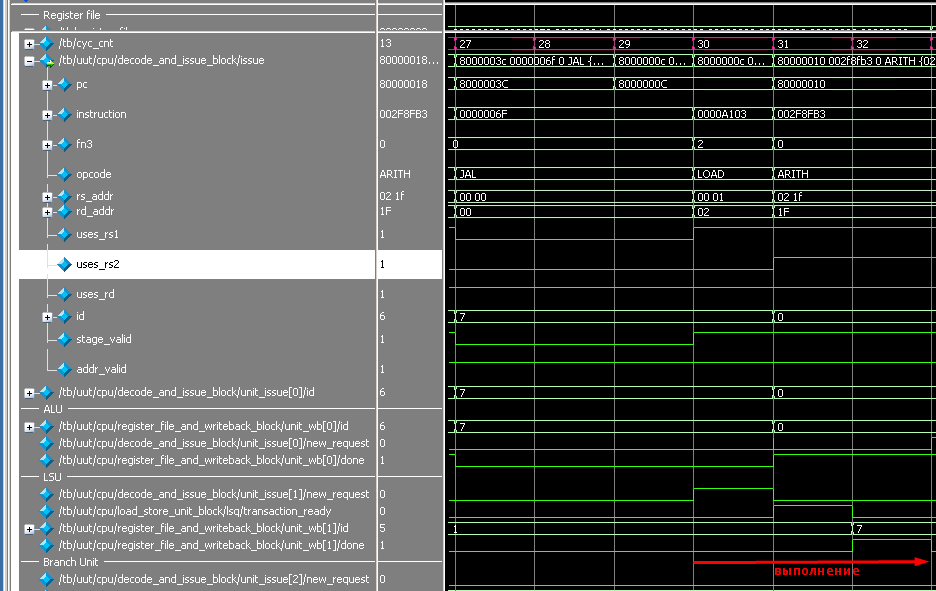


Рисунок 7. Выполнение

## Задание 5

Адрес команды с #!: 80000001c

Выборка, диспетчеризация, декодирование, выполнение:

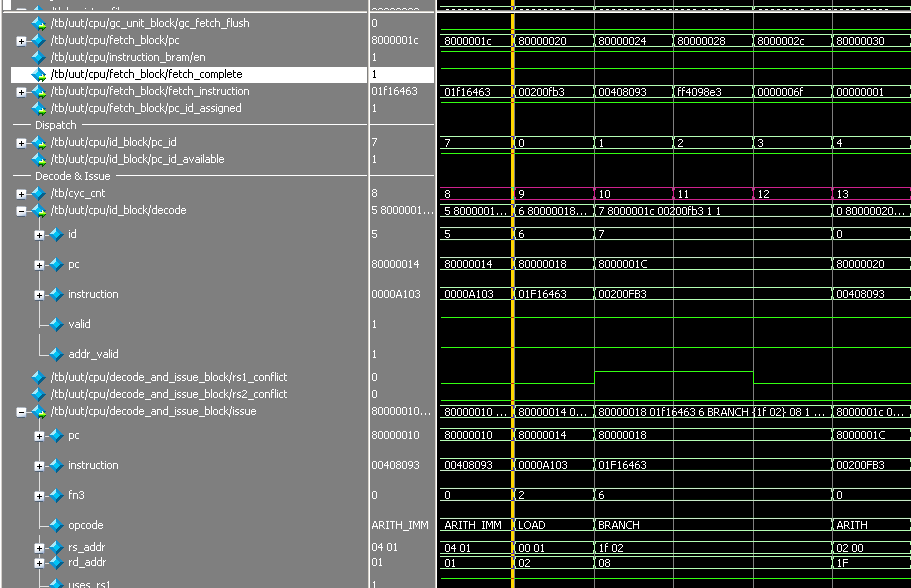


Рисунок 8. Выборка, диспетчеризация, декодирование

Выполнение:

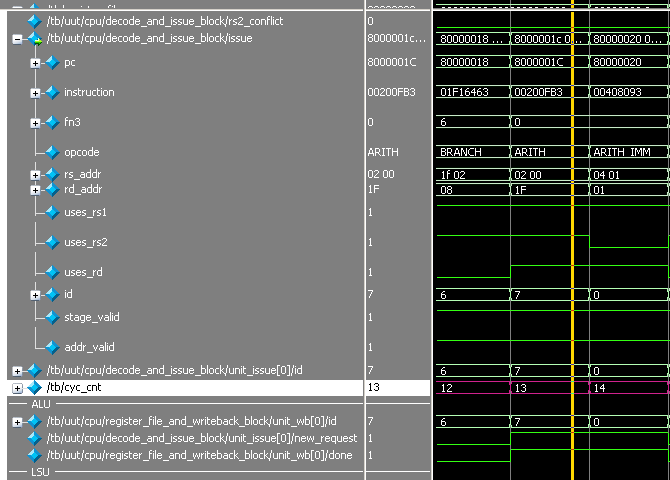


Рисунок 9. Выполнение

Трасса программы:

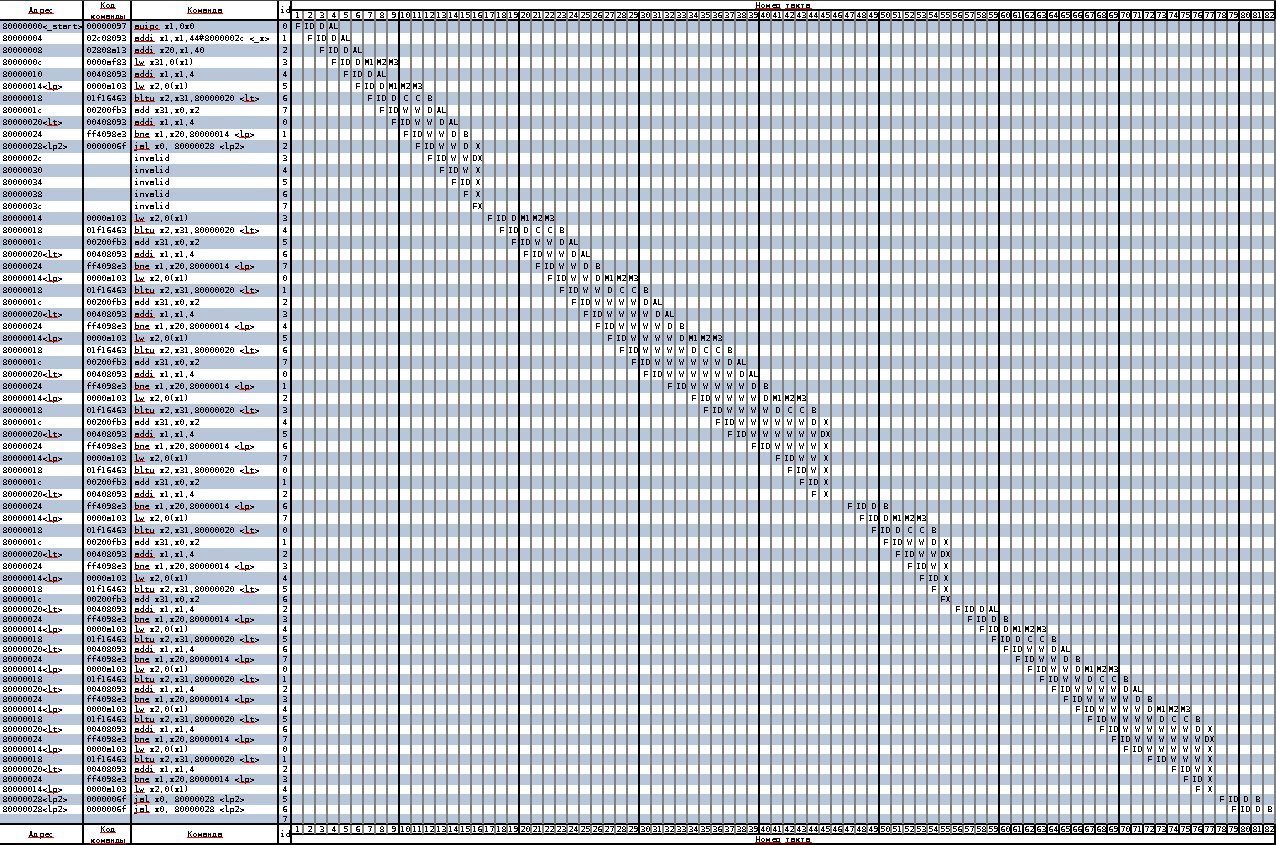


Рисунок 10. Трасса изначальной программы

Из трассы видно возникновение конфликтов, которые замедляют работу программы. Для оптимизации можно перенести команду addi x1,x1,4 в место между конфликтующими командами.

Ниже приведены исходный и оптимизированный коды программы моего варианта.

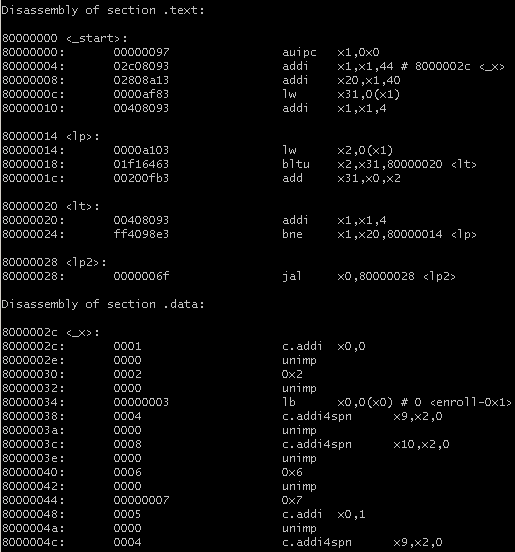


Рисунок 11. Код изначальной программы

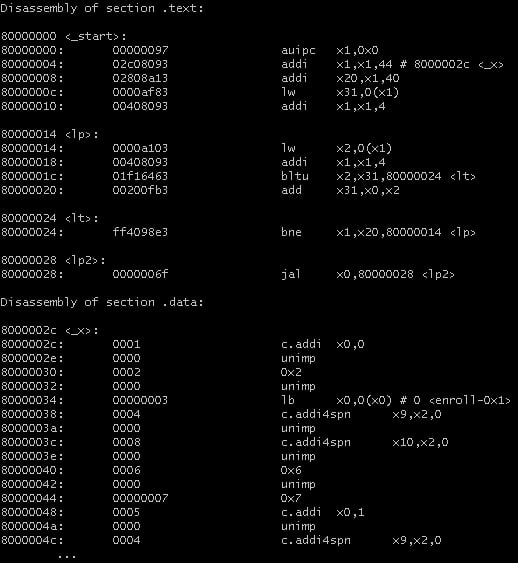


Рисунок 12. Код оптимизированной программы

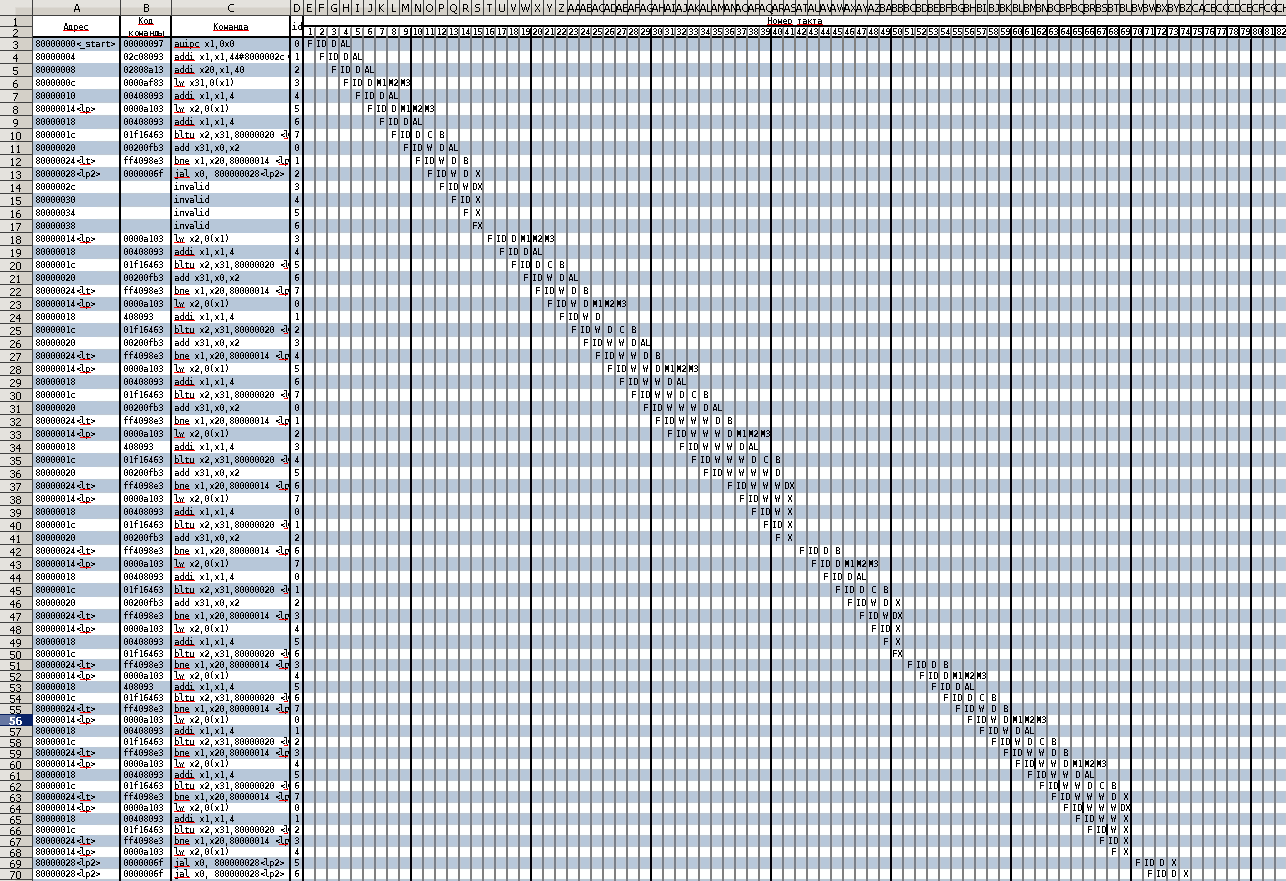


Рисунок 13. Трасса оптимизированной программы

Оптимизированная программа работает на 8 тактов быстрее.

## Заключение

В ходе лабораторной работы я ознакомился с принципами функционирования и построения суперскалярных конвейерных микропроцессоров.