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»

Отчёт

по лабораторной работе №2

Название «Изучение принципов работы микропроцессорного ядра RISC-V»

Дисциплина «Архитектура ЭВМ»

Вариант 3

Студент ИУ7-55Б _____ Клименко А.К.
(подпись, дата) (Фамилия И.О.)

Преподователь _____ Ибрагимов С.В.
(подпись, дата) (Фамилия И.О.)

Москва, 2022

Содержание

Введение 3
1 Теоритический раздел 4
1.1 Архитектура набора команд RV32I 4
1.1.1 Регистровая модель 4
1.1.2 Модель памяти 4
1.1.3 Система команд 4
2 Экспериментальный раздел 6
2.1 Исследование программы 6
2.1.1 Исходный код программы 6
2.1.2 Дизассемблированный листинг 7
2.1.3 Псевдокод 8
2.1.4 Трасса работы программы 12
2.1.5 Выводы об эффективности 12
2.2 Оптимизация программы 13
2.2.1 Исходный код оптимизированной программы 13
2.2.2 Дизассемблированный листинг оптимизированной программы 14
2.2.3 Трасса работы оптимизированной программы 15
2.2.4 Выводы 15
Заключение 16

Введение

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 RISC-V. Данное описание выполнено на языке описания аппаратуры SystemVerilog.

В ходе лабораторной работы используется средство моделирования MentorGraphics Modelsim для моделирования работы исследуемого микропроцессора в процессе выполнения программы и наблюдения формы внутренних сигналов.

Для работы с проектом используется САПР Intel Quartus.

1 Теоритический раздел

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.

1.1 Архитектура набора команд RV32I

В настоящем разделе описывается архитектура набора команд, то есть архитектура абстрактной вычислительной машины с точки зрения набора команд без связи с конкретной аппаратной реализацией.

1.1.1 Регистровая модель

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

1.1.2 Модель памяти

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

1.1.3 Система команд

Большая часть команд RV32I является трехадресными, выполняющими операции над двумя заданными явно операндами, и сохраняющими результат в регистре. 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 разделение команд на команды доступа к памяти (чтение данных из памяти в регистр или запись данных из регистра в память) и команды обработки данных в регистрах.

Таким образом, команды RV32I можно разделить на следующие категории:

1) Команды обработки данных
    а) Арифметические и логические команды
    б) Команды сравнения
    в) Команды сдвига
    г) Команды формирования значения в старшей части регистра
2) Команды передачи управления
    а) Команды безусловного перехода с сохранением адреса возврата
    б) Команды условного перехода
3) Команды доступа к памяти
    а) Команды загрузки
    б) Команды сохранения
4) Системные команды

2 Экспериментальный раздел

2.1 Исследование программы

2.1.1 Исходный код программы

```
        .section .text
        .globl _start;
        len = 8      # Размер массива
        enroll = 1   # Количество обрабатываемых элементов за одну итерацию
        elem_sz = 4  # Размер одного элемента массива

_start:
        la x1, _x
        addi x20, x1, elem_sz*(len-1) # Адрес последнего элемента
lp:
        lw x2, 0(x1)
        add x31, x31, x2 \#!
        addi x1, x1, elem_sz*enroll
        bne x1, x20, lp
        addi x31, x31, 1
lp2:    j lp2

        .section .data
_x:     .4byte 0x1
        .4byte 0x2
        .4byte 0x3
        .4byte 0x4
        .4byte 0x5
        .4byte 0x6
        .4byte 0x7
        .4byte 0x8
```

2.1.2 Дизассемблированный листинг

```
Disassembly of section .text:

80000000 <_start>:
80000000:       00000097                auipc   x1,0x0
80000004:       02408093                addi    x1,x1,36 # 80000024 <_x>
80000008:       01c08a13                addi    x20,x1,28

8000000c <lp>:
8000000c:       0000a103                lw      x2,0(x1)
80000010:       002f8fb3                add     x31,x31,x2
80000014:       00408093                addi    x1,x1,4
80000018:       ff409ae3                bne     x1,x20,8000000c <lp>
8000001c:       001f8f93                addi    x31,x31,1

80000020 <lp2>:
80000020:       0000006f                jal     x0,80000020 <lp2>

Disassembly of section .data:

80000024 <_x>:
80000024:       0001                    c.addi  x0,0
80000026:       0000                    unimp
80000028:       0002                    0x2
8000002a:       0000                    unimp
8000002c:       00000003                lb      x0,0(x0) # 0 <enroll-0x1>
80000030:       0004                    c.addi4spn      x9,x2,0
80000032:       0000                    unimp
80000034:       0005                    c.addi  x0,1
80000036:       0000                    unimp
80000038:       0006                    0x6
8000003a:       0000                    unimp
8000003c:       00000007                0x7
80000040:       0008                    c.addi4spn      x10,x2,0
```

2.1.3 Псевдокод

Листинг 2.1 — Эквивалентный код на C.

```
#define len     8
#define enroll  1
#define elem_sz 4

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

void _start()
{
    int *x1 = &_x;
    int *x20 = x1 + elem_si * (len - 1);
    int x31 = 0;

    do
    {
        int x2 = *x1;
        x31 += x2;
        x1 += elem_sz * enroll;
    }
    while (x1 != x20);

    x31++;

    while (1) {}
}
```

Задание 2.

Получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки и диспетчеризации команды с указанным адресом.

Адрес команды: 80000014, номер итерации: 1.

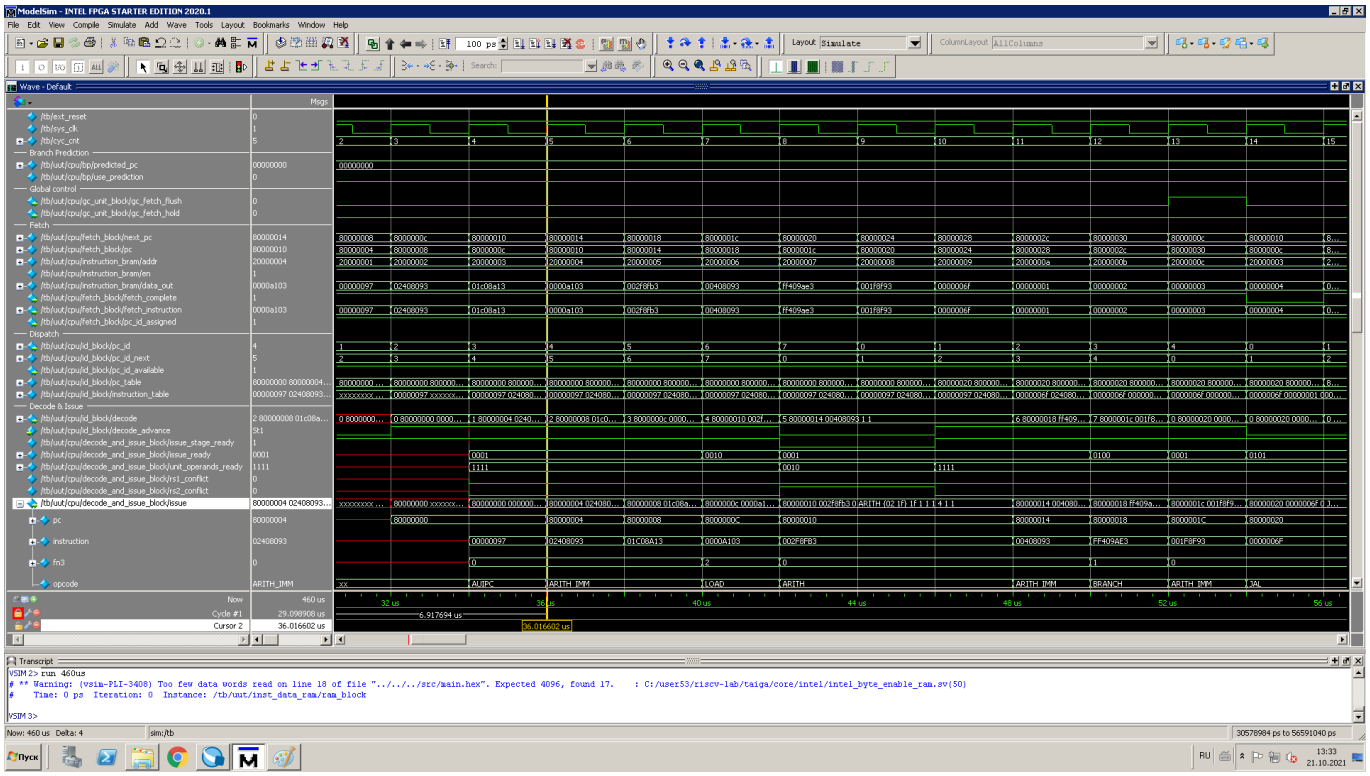


Рисунок 2.1 — Скриншот для задания 2

Задание 3.

Получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с указанным адресом.

Адрес команды: 80000020, номер итерации: 1.

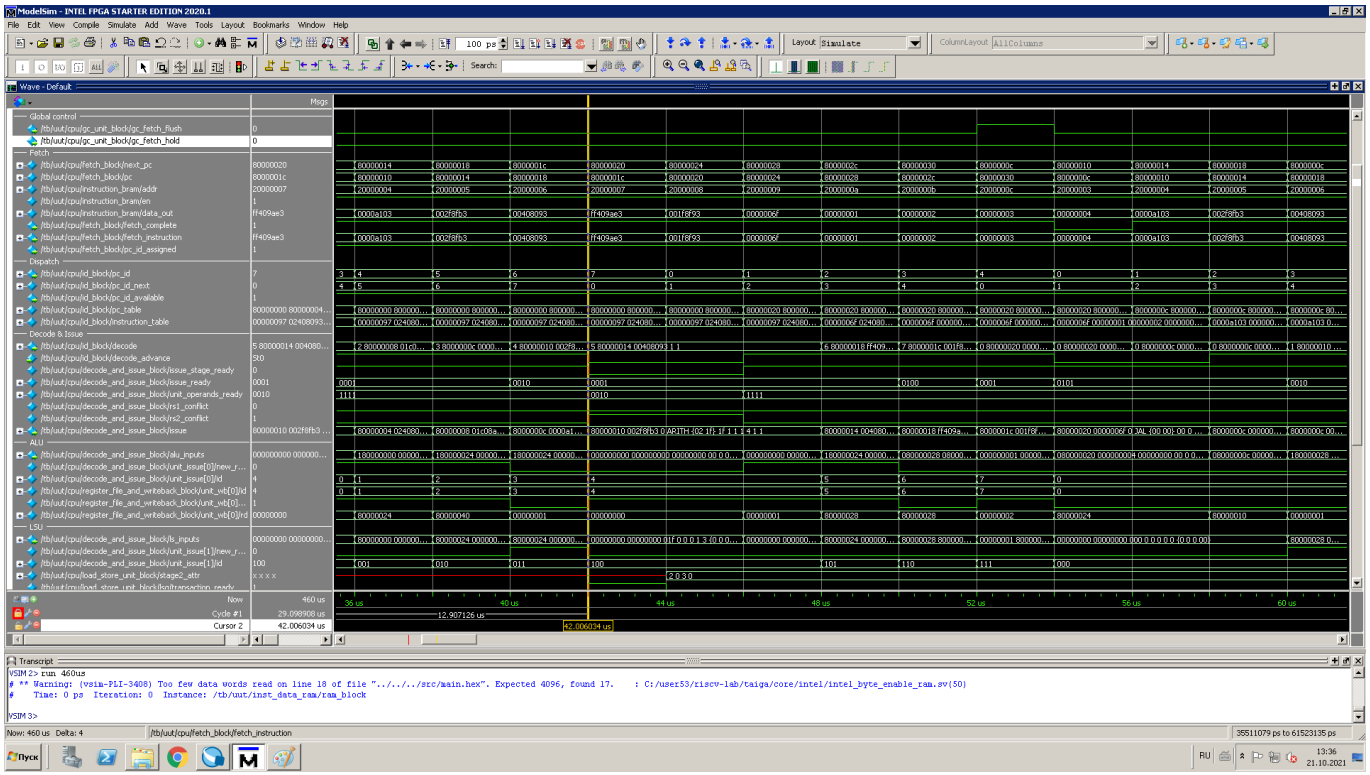


Рисунок 2.2 — Скриншот для задания 3

Задание 4.

Получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с указанным адресом.

Адрес команды: 80000008.

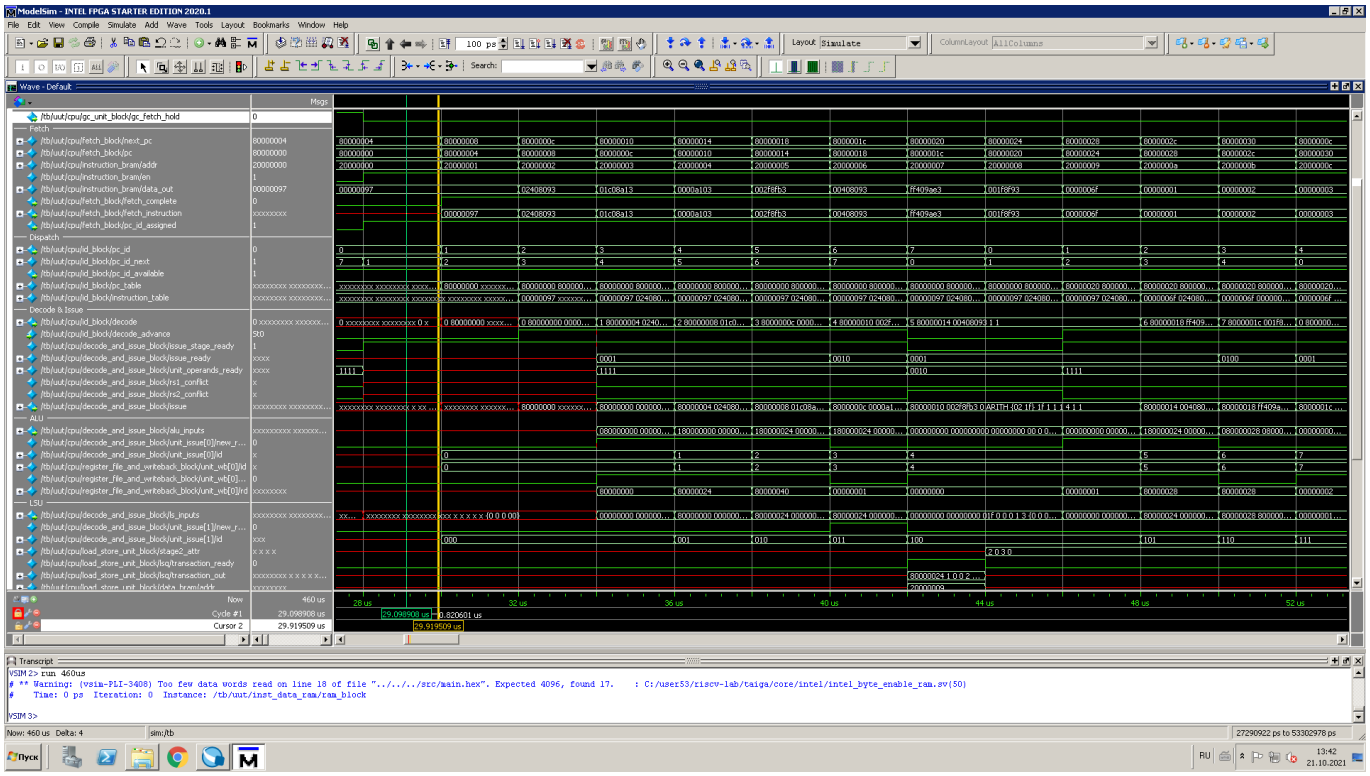


Рисунок 2.3 — Скриншот для задания 4

2.1.4 Трасса работы программы

Адрес	Код команды	Команда	id	Номер такта
80000000<_start>	00000097	auipc x1,0x0	0	1: F ID D AL
80000004	02408093	addi x1,x1,36	1	2: F ID D AL
80000008	01c08a13	addi x20,x1,28#80000024<_x>	2	3: F ID D AL
8000000c <lp>	0000a103	lw x2,0(x1)	3	4: F ID D M1 M2 M3
80000010	002f8fb3	add x31,x31,x2	4	5: F ID D C C AL
80000014	00408093	addi x1,x1,4	5	6: F ID W W D AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	6	7: F ID W W D B
8000001c	001f8f93	addi x31,x31,1	7	8: F ID W W D X
80000020 <lp2>	0000006f	jal x0,80000020<lp2>	0	9: F ID W W DX
80000024	00000001	<invalid operation>	1	10: F ID W X
80000028	00000002	<invalid operation>	2	11: F ID X
8000002C	00000003	<invalid operation>	3	12: F X
80000030	00000004	<invalid operation>	4	13: FX
8000000c <lp>	0000a103	lw x2,0(x1)	7	14: F ID D M1 M2 M3
80000010	002f8fb3	add x31,x31,x2	0	15: F ID D C C AL
80000014	00408093	addi x1,x1,4	1	16: F ID W W D AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	2	17: F ID W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	3	18: F ID W W D M1 M2 M3
80000010	002f8fb3	add x31,x31,x2	4	19: F ID W W D C C AL
80000014	00408093	addi x1,x1,4	5	20: F ID W W W W D AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	6	21: F ID W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	7	22: F ID W W W W D M1 M2 M3
80000010	002f8fb3	add x31,x31,x2	0	23: F ID W W W W D C C AL
80000014	00408093	addi x1,x1,4	1	24: F ID W W W W W W D AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	2	25: F ID W W W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	3	27: F ID W W W W W D M1 M2 M3
80000010	002f8fb3	add x31,x31,x2	4	28: F ID W W W W W D C C AL
80000014	00408093	addi x1,x1,4	5	29: F ID W W W W W W W D AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	6	30: F ID W W W W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	7	33: F ID W W W W W D M1 M2 M3
80000010	002f8fb3	add x31,x31,x2	0	34: F ID W W W W W D C C AL
80000014	00408093	addi x1,x1,4	1	35: F ID W W W W W W W D AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	2	36: F ID W W W W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	3	39: F ID W W W W W D M1 M2 M3
80000010	002f8fb3	add x31,x31,x2	4	40: F ID W W W W W D C C AL
80000014	00408093	addi x1,x1,4	5	41: F ID W W W W W W W D AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	6	42: F ID W W W W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	7	45: F ID W W W W W D M1 M2 M3
80000010	002f8fb3	add x31,x31,x2	0	46: F ID W W W W W D C C AL
80000014	00408093	addi x1,x1,4	1	47: F ID W W W W W W W D AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	2	48: F ID W W W W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	3	51: F ID W W W W W D M1 M2 M3
80000010	002f8fb3	add x31,x31,x2	4	52: F ID W W W W W D C C AL
80000014	00408093	addi x1,x1,4	5	53: F ID W W W W W W W D AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	6	54: F ID W W W W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	7	57: F ID W W W W W W X
80000010	002f8fb3	add x31,x31,x2	0	58: F ID W W W W W DX
80000014	00408093	addi x1,x1,4	1	59: F ID W W W W X
80000018	ff409ae3	bne x1,x20,8000000c<lp>	2	60: F ID W W W X
8000000c <lp>	0000a103	lw x2,0(x1)	3	63: F ID X
80000010	002f8fb3	add x31,x31,x2	4	64: F X
80000014	00408093	addi x1,x1,4	5	65: FX
8000001c	001f8f93	addi x31,x31,1	0	66: F ID D AL
80000020	0000006f	jal x0,80000020<lp2>	1	67: F ID D B
80000024	00000001	<invalid operation>	2	68: F ID D X
80000028	00000002	<invalid operation>	3	69: F ID DX
8000002	00000003	<invalid operation>	4	70: F X
-64000024	00000004	<invalid operation>	5	71: FX
80000020	0000006f	jal x0,80000020<lp2> <forever>	3	72: F ID D B
80000020	0000006f	jal x0,80000020<lp2> <forever>	4	73: F ID D B

2.1.5 Выводы об эффективности

Из рассмотрения трассы видно, что выполнение программы до окончания команды `addi x31,x31,1` заняло 69 тактов (1-69). Из них, в течение 23 тактов (7, 8, 13-15, 17, 18, 23, 24, 29, 30, 35, 36, 41, 42, 47, 48, 53, 54, 60, 65-67) не происходило планирование на выполнение новых команд, то есть, 50% времени было потрачено не эффективно из-за ошибок предсказания ветвлений и конфликтов на конвейере.

Для оптимизации данной программы можно поменять местами команды на строках 12 и 13. После этого конвеер сможет приступить к обработке следующей команды после планирования `lw x2,0(x1)`.

2.2 Оптимизация программы

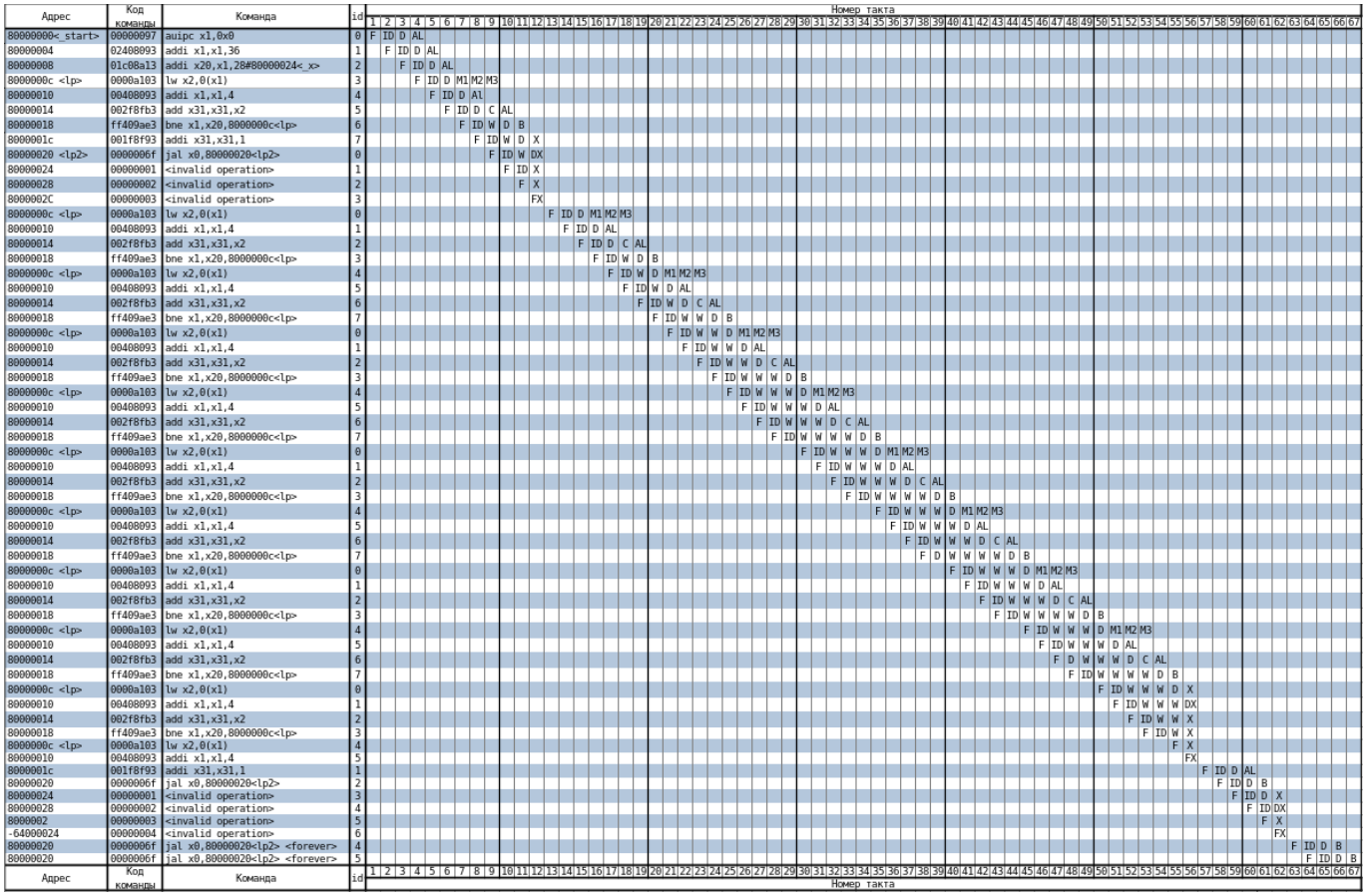
2.2.1 Исходный код оптимизированной программы

```
        .section .text
        .globl _start;
        len = 8      # Размер массива
        enroll = 1   # Количество обрабатываемых элементов за одну итерацию
        elem_sz = 4  # Размер одного элемента массива

_start:
        la x1, _x
        addi x20, x1, elem_sz*(len-1) # Адрес последнего элемента
lp:
        lw x2, 0(x1)
        addi x1, x1, elem_sz*enroll
        add x31, x31, x2
        bne x1, x20, lp
        addi x31, x31, 1
lp2:    j lp2

        .section .data
_x:     .4byte 0x1
        .4byte 0x2
        .4byte 0x3
        .4byte 0x4
        .4byte 0x5
        .4byte 0x6
        .4byte 0x7
        .4byte 0x8
```

2.2.2 Дизассемблированный листинг оптимизированной программы

```
Disassembly of section .text:

80000000 <_start>:
80000000:       00000097                auipc   x1,0x0
80000004:       02408093                addi    x1,x1,36 # 80000024 <_x>
80000008:       01c08a13                addi    x20,x1,28

8000000c <lp>:
8000000c:       0000a103                lw      x2,0(x1)
80000010:       00408093                addi    x1,x1,4
80000014:       002f8fb3                add     x31,x31,x2
80000018:       ff409ae3                bne     x1,x20,8000000c <lp>
8000001c:       001f8f93                addi    x31,x31,1

80000020 <lp2>:
80000020:       0000006f                jal     x0,80000020 <lp2>

Disassembly of section .data:

80000024 <_x>:
80000024:       0001                    c.addi  x0,0
80000026:       0000                    unimp
80000028:       0002                    0x2
8000002a:       0000                    unimp
8000002c:       00000003                lb      x0,0(x0) # 0 <enroll-0x1>
80000030:       0004                    c.addi4spn      x9,x2,0
80000032:       0000                    unimp
80000034:       0005                    c.addi  x0,1
80000036:       0000                    unimp
80000038:       0006                    0x6
8000003a:       0000                    unimp
8000003c:       00000007                0x7
80000040:       0008                    c.addi4spn      x10,x2,0
```

2.2.3 Трасса работы оптимизированной программы



Адрес	Код команды	Команда	id	Номер такта
80000000<_start>	00000097	auipc x1,0x0	0	F ID D AL
80000004	02408093	addi x1,x1,36	1	F ID D AL
80000008	01c08a13	addi x20,x1,28#80000024<_x>	2	F ID D AL
8000000c <lp>	0000a103	lw x2,0(x1)	3	F ID D M1 M2 M3
80000010	00408093	addi x1,x1,4	4	F ID D Al
80000014	002f8fb3	add x31,x31,x2	5	F ID D C AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	6	F ID W D B
8000001c	001f8f93	addi x31,x31,1	7	F ID W D X
80000020 <lp2>	0000006f	jal x0,80000020<lp2>	0	F ID W DX
80000024	00000001	<invalid operation>	1	F ID X
80000028	00000002	<invalid operation>	2	F X
8000002C	00000003	<invalid operation>	3	FX
8000000c <lp>	0000a103	lw x2,0(x1)	0	F ID D M1 M2 M3
80000010	00408093	addi x1,x1,4	1	F ID D AL
80000014	002f8fb3	add x31,x31,x2	2	F ID D C AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	3	F ID W D B
8000000c <lp>	0000a103	lw x2,0(x1)	4	F ID W D M1 M2 M3
80000010	00408093	addi x1,x1,4	5	F ID W D AL
80000014	002f8fb3	add x31,x31,x2	6	F ID W D C AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	7	F ID W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	0	F ID W W D M1 M2 M3
80000010	00408093	addi x1,x1,4	1	F ID W W D AL
80000014	002f8fb3	add x31,x31,x2	2	F ID W W D C AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	3	F ID W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	4	F ID W W W D M1 M2 M3
80000010	00408093	addi x1,x1,4	5	F ID W W W D AL
80000014	002f8fb3	add x31,x31,x2	6	F ID W W W D C AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	7	F ID W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	0	F ID W W W W D M1 M2 M3
80000010	00408093	addi x1,x1,4	1	F ID W W W W D AL
80000014	002f8fb3	add x31,x31,x2	2	F ID W W W W D C AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	3	F ID W W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	4	F ID W W W W W D M1 M2 M3
80000010	00408093	addi x1,x1,4	5	F ID W W W W W D AL
80000014	002f8fb3	add x31,x31,x2	6	F ID W W W W W D C AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	7	F ID W W W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	0	F ID W W W W W W D M1 M2 M3
80000010	00408093	addi x1,x1,4	1	F ID W W W W W W D AL
80000014	002f8fb3	add x31,x31,x2	2	F ID W W W W W W D C AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	3	F ID W W W W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	4	F ID W W W W W W W D M1 M2 M3
80000010	00408093	addi x1,x1,4	5	F ID W W W W W W W D AL
80000014	002f8fb3	add x31,x31,x2	6	F D W W W W W W D C AL
80000018	ff409ae3	bne x1,x20,8000000c<lp>	7	F ID W W W W W W D B
8000000c <lp>	0000a103	lw x2,0(x1)	0	F ID W W W W W W D X
80000010	00408093	addi x1,x1,4	1	F ID W W W W W W DX
80000014	002f8fb3	add x31,x31,x2	2	F ID W W W W W X
80000018	ff409ae3	bne x1,x20,8000000c<lp>	3	F ID W W W W X
8000000c <lp>	0000a103	lw x2,0(x1)	4	F ID W W W W X
80000010	00408093	addi x1,x1,4	5	F ID W W W W X
8000001c	001f8f93	addi x31,x31,1	1	F ID D AL
80000020	0000006f	jal x0,80000020<lp2>	2	F ID D B
80000024	00000001	<invalid operation>	3	F ID D X
80000028	00000002	<invalid operation>	4	F ID DX
8000002	00000003	<invalid operation>	5	F X
-64000024	00000004	<invalid operation>	6	FX
80000020	0000006f	jal x0,80000020<lp2> <forever>	4	F ID D B
80000020	0000006f	jal x0,80000020<lp2> <forever>	5	F ID D B

2.2.4 Выводы

Выполнение оптимизированной программы до окончания каманды `addi x31,x31,1` заняло 60 тактов, что на 13% быстрее неоптимизированной программы.

Рассмотрев трассу работы оптимизированной программы, можно обнаружить, что планирование на выполнение новых команд происходило только 13 тактов (8, 12, 13, 17, 22, 27, 32, 37, 42, 47, 52, 56, 57).

Заключение

В результате проделанной работы было получено представление о принципах работы и особенностях суперскалярных конвеерных микропроцессоров.

В ходе выполнения практикума приобретён опыт работы с конвеером команд, потактовой отладкой программ, а также моделирования работы исследуемого микропроцессора в процессе выполнения программы и наблюдения внутренних сигналов.