



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»

---

## Отчет по лабораторной работе №2 по курсу "Архитектура ЭВМ"

Тема Изучение принципов работы микропроцессорного ядра RISC-V

---

Студент Цветков И.А.

---

Группа ИУ7-53Б

---

Оценка (баллы) \_\_\_\_\_

---

Преподаватель Дубровин Е.Н.

---

# Содержание

|   |           |
|---|-----------|
| <b>1 Теоритические основы</b>                 | <b>4</b>  |
| 1.1 Архитектура набора команд RV32I . . . . . | 4         |
| <b>2 Выполнение лабораторной работы</b>       | <b>5</b>  |
| 2.1 Задание 1 . . . . .                       | 5         |
| 2.1.1 Результат выполнения задания . . . . .  | 5         |
| 2.1.2 Вывод . . . . .                         | 8         |
| 2.2 Задание 2 . . . . .                       | 8         |
| 2.2.1 Результат выполнения задания . . . . .  | 8         |
| 2.2.2 Вывод . . . . .                         | 10        |
| 2.3 Задание 3 . . . . .                       | 10        |
| 2.3.1 Результат выполнения задания . . . . .  | 10        |
| 2.3.2 Вывод . . . . .                         | 12        |
| 2.4 Задание 4 . . . . .                       | 12        |
| 2.4.1 Результат выполнения задания . . . . .  | 12        |
| 2.4.2 Вывод . . . . .                         | 13        |
| 2.5 Задание 5 . . . . .                       | 13        |
| 2.5.1 Часть 1 . . . . .                       | 13        |
| 2.5.2 Часть 2 . . . . .                       | 14        |
| 2.5.3 Часть 3 . . . . .                       | 17        |
| 2.5.4 Часть 4 . . . . .                       | 18        |
| <b>Заключение</b>                             | <b>21</b> |

# **Введение**

**Основной целью данной 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.

# 1 Теоритические основы

При исследовании производительности будет использована архитектура набора команд **RV32I**.

## 1.1 Архитектура набора команд RV32I

**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.

## 2 Выполнение лабораторной работы

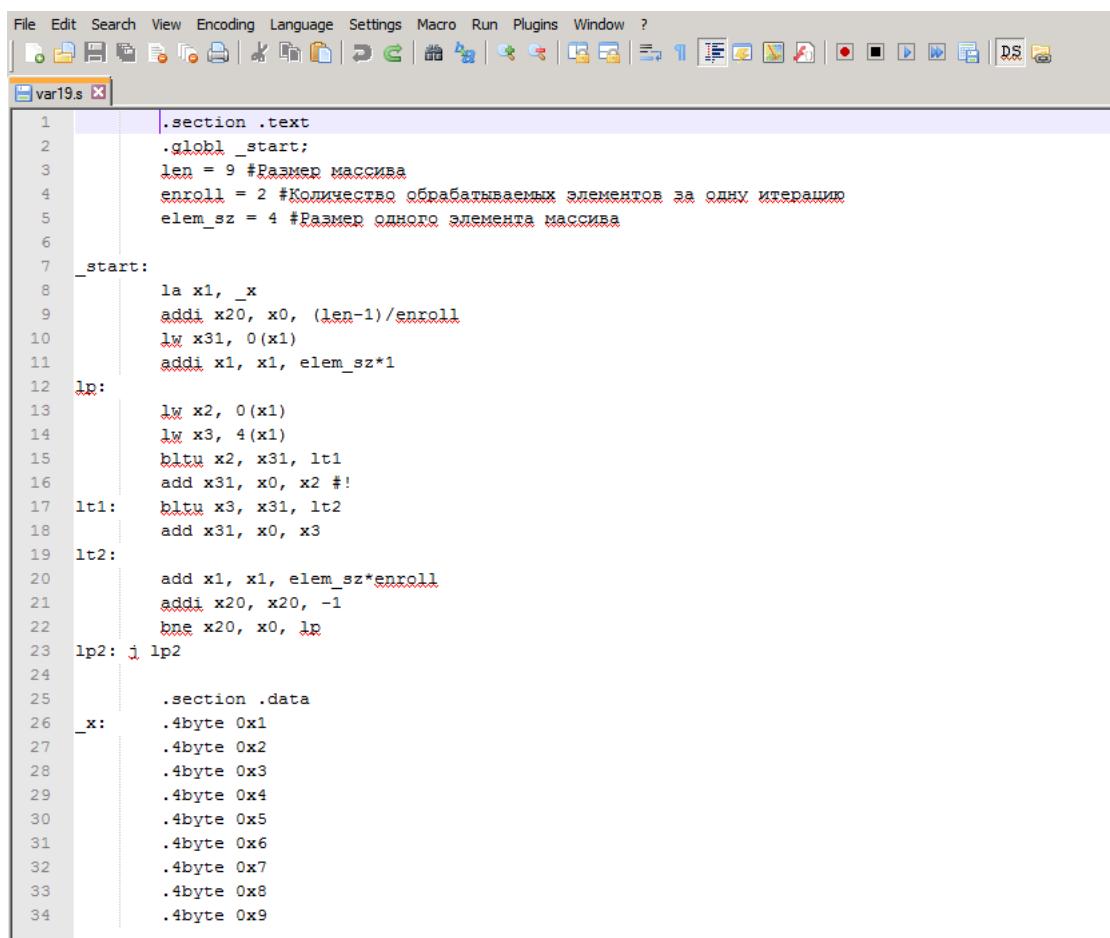
Для знакомства с RISC-V будет выполнены необходимые задачи.

### 2.1 Задание 1

Дизассемблирование индивидуальной программы.

#### 2.1.1 Результат выполнения задания

На рисунке 2.1 представлен код программы для индивидуального задания по 19 варианту. При этом на рисунке 2.2, а в листинге 2.1 код на языке Си для данной программы.



```
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
var19.s
1 .section .text
2 .globl _start;
3 len = 9 #Размер массива
4 enroll = 2 #Количество обрабатываемых элементов за одну итерацию
5 elem_sz = 4 #Размер одного элемента массива
6
7 _start:
8     la x1, _x
9     addi x20, x0, (len-1)/enroll
10    lw x31, 0(x1)
11    addi x1, x1, elem_sz*1
12    lp:
13        lw x2, 0(x1)
14        lw x3, 4(x1)
15        bltu x2, x31, lt1
16        add x31, x0, x2 !
17    lt1: bltu x3, x31, lt2
18        add x31, x0, x3
19    lt2:
20        add x1, x1, elem_sz*enroll
21        addi x20, x20, -1
22        bne x20, x0, lp
23    lp2: j lp2
24
25 .section .data
26 _x:
27     .4byte 0x1
28     .4byte 0x2
29     .4byte 0x3
30     .4byte 0x4
31     .4byte 0x5
32     .4byte 0x6
33     .4byte 0x7
34     .4byte 0x8
35     .4byte 0x9
```

Рисунок 2.1 – Код программы

```

MINGW32:/c/User/TsvetkovNew/riscv-lab/src
riscv64-unknown-elf-ld -b elf32-littleriscv -T link.ld var19.o -o var19.elf
riscv64-unknown-elf-objdump -D -M numeric,no-aliases -t var19.elf

var19.elf:      file format elf32-littleriscv

SYMBOL TABLE:
80000000 l    d  .text  00000000 .text
8000003c l    d  .data  00000000 .data
00000000 l    df  *ABS*  00000000 var19.o
00000009 l    *ABS*  00000000 len
00000002 l    *ABS*  00000000 enroll
00000004 l    *ABS*  00000000 elem_sz
8000003c l    .data  00000000 _x
80000014 l    .text  00000000 lp
80000024 l    .text  00000000 lt1
8000002c l    .text  00000000 lt2
80000038 l    .text  00000000 lp2
80000000 g    .text  00000000 _start
80000060 g    .data  00000000 _end

Disassembly of section .text:
80000000 <_start>:
80000000: 00000097          auipc   x1,0x0
80000004: 03c08093          addi    x1,x1,60 # 8000003c <_x>
80000008: 00400a13          addi    x20,x0,4
8000000c: 0000af83          lw      x31,0(x1)
80000010: 00408093          addi    x1,x1,4

80000014 <lp>:
80000014: 0000a103          lw      x2,0(x1)
80000018: 0040a183          lw      x3,4(x1)
8000001c: 01f1e463          bltu   x2,x31,80000024 <lt1>
80000020: 00200fb3          add    x31,x0,x2

80000024 <lt1>:
80000024: 01f1e463          bltu   x3,x31,8000002c <lt2>
80000028: 00300fb3          add    x31,x0,x3

8000002c <lt2>:
8000002c: 00808093          addi    x1,x1,-8
80000030: fffa0a13          addi    x20,x20,-1
80000034: fe0a10e3          bne    x20,x0,80000014 <lp>

80000038 <lp2>:
80000038: 0000006f          jal    x0,80000038 <lp2>

Disassembly of section .data:
8000003c <_x>:
8000003c: 0001              c.addi  x0,0
8000003e: 0000              unimp
80000040: 0002              0x2
80000042: 0000              unimp
80000044: 00000003          lb     x0,0(x0) # 0 <enroll-0x2>
80000048: 0004              c.addi4spn x9,x2,0
8000004a: 0000              unimp
8000004c: 0005              c.addi  x0,1
8000004e: 0000              unimp
80000050: 0006              0x6
80000052: 0000              unimp
80000054: 00000007          0x7
80000058: 0008              c.addi4spn x10,x2,0
8000005a: 0000              unimp
8000005c: 0009              c.addi  x0,2

```

Рисунок 2.2 – Код программы (дизассемблированный)

Листинг 2.1 – Код программы (на языке Си)

```
1 #define LEN 9
2 #define ENROLL 2
3 #define ELEM_SZ 4
4
5 int _x[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
6
7 int main(void)
8 {
9     int *x1 = _x;
10    int x20 = (LEN - 1) / ENROLL;
11    int x31 = x1[0];
12
13    x1 += ELEM_SZ;
14
15    do
16    {
17        int x2 = x1[0];
18        int x3 = x1[1];
19
20        if (!(x3 < x31))
21        {
22            x31 = x2;
23        }
24
25        if (!(x3 < x31))
26        {
27            x31 = x3;
28        }
29
30        x1 += ELEM_SZ * ENROLL;
31        x20 -= 1;
32    } while (x20 != 0);
33
34    while(1) {}
35
36    return 0;
37 }
```

## 2.1.2 Вывод

В программе, которая представлена на рисунке 2.1, находится максимум из элементов массива. В результате, в **x31** будет находиться максимальное число из массива **\_x**, причем, в **x31** будет помещен последний из максимумов, если их было встречено несколько одинаковых. Тогда в результате выполнения программы в **x31** будет лежать 0x9.

## 2.2 Задание 2

**Задание:** Получить снимок экрана, содержащий временную диаграмму выполнения стадий *выборки и диспетчеризации* команды со следующим адресом: 80000028, 2-ая итерация.

### 2.2.1 Результат выполнения задания

Сигнал **fetch\_complete = 1** – диспетчеризация произошла предыдущем такте. На рисунках 2.3–2.4 представлены временные диаграммы этапов выборки и диспетчеризации.

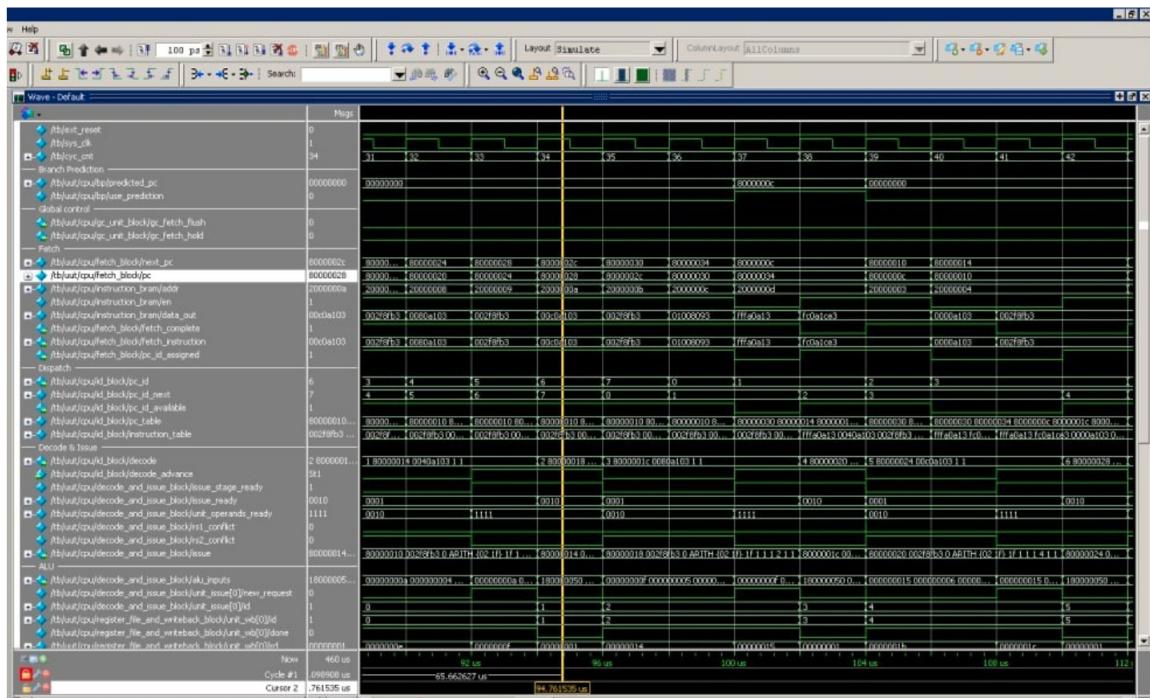


Рисунок 2.3 – Диаграмма этапа выборки

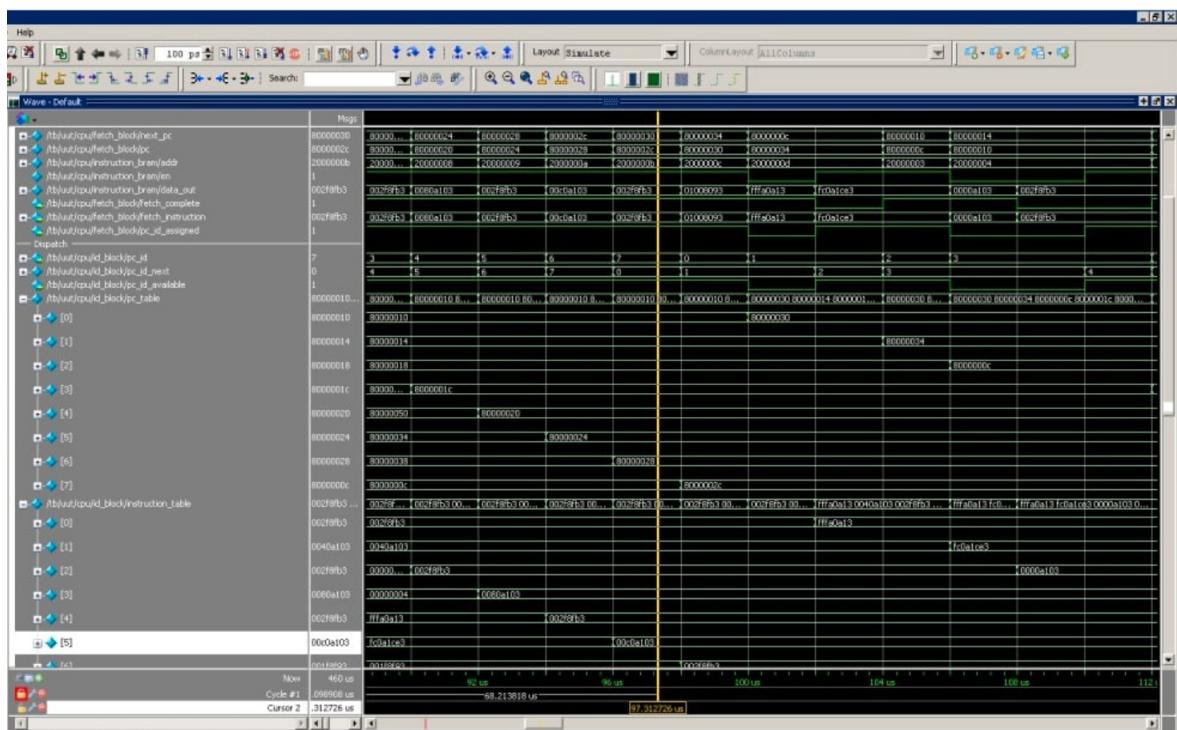


Рисунок 2.4 – Диаграмма этапа диспетчеризации

## 2.2.2 Вывод

Поскольку выборка и диспетчеризация предыдущих команд уже произошли, то выборка и диспетчеризация вышеуказанной команды произошли последовательно, то есть без задержки по тактам.

## 2.3 Задание 3

**Задание:** Получить снимок экрана, содержащий временную диаграмму выполнения стадий декодирования и планирования команды со следующим адресом: 80000034, 2-ая итерация.

### 2.3.1 Результат выполнения задания

На рисунках 2.5–2.6 представлены временные диаграммы этапов декодирования и планирования. Как видно, в конце 47-го такта виден результат декодирования, а 48-ой такт – планирование на выполнение.

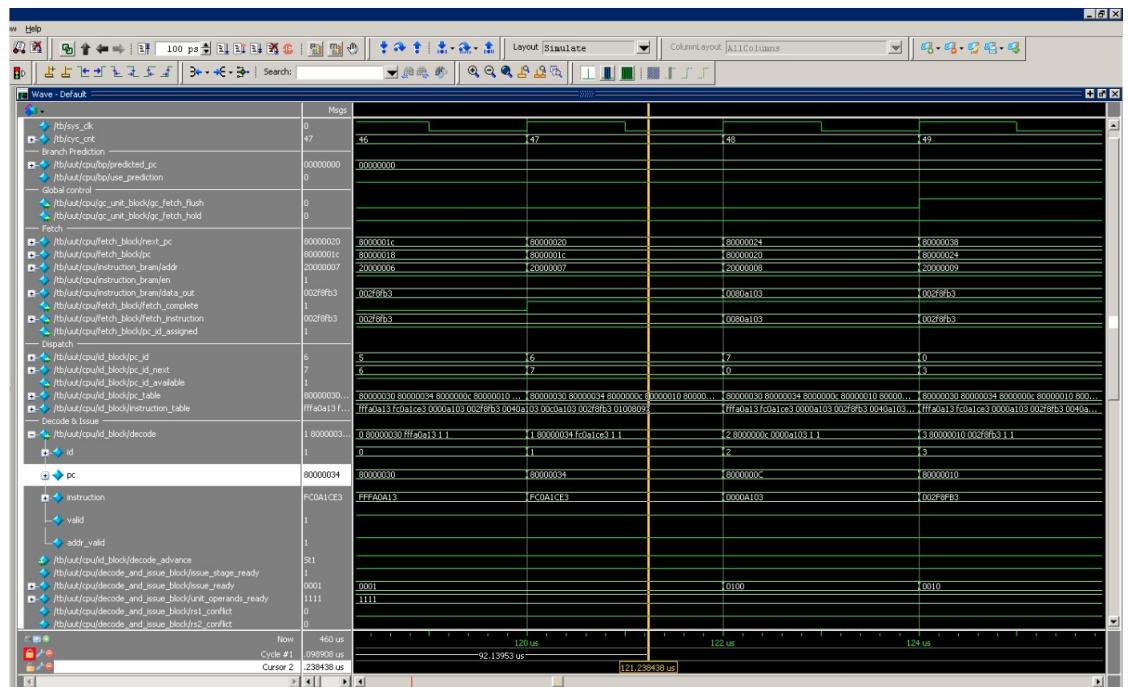


Рисунок 2.5 – Диаграмма этапа декодирования

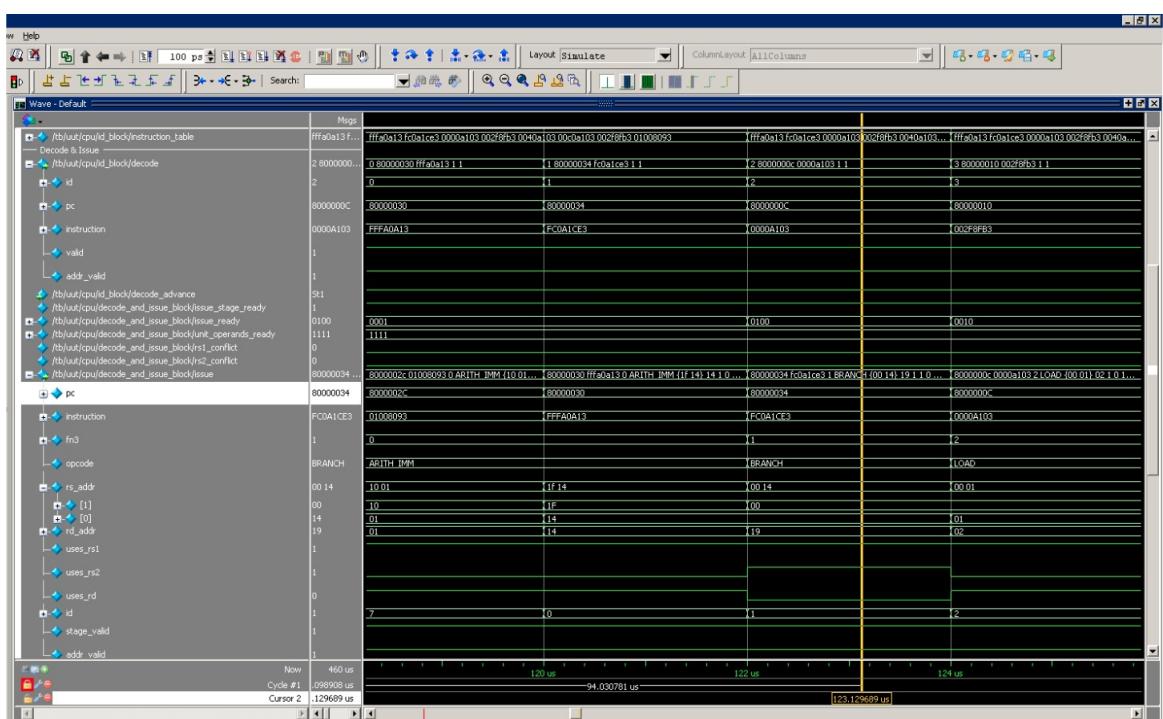


Рисунок 2.6 – Диаграмма этапа планирования

### 2.3.2 Вывод

Поскольку декодирование и планирование предыдущих команд уже произошли, то декодирования и планирования вышеуказанной команды произошли последовательно, то есть без задержки по тактам.

## 2.4 Задание 4

**Задание:** Получить снимок экрана, содержащий временную диаграмму стадии выполнения команды со следующим адресом: 80000020, 2-ая итерация.

## 2.4.1 Результат выполнения задания

На рисунке 2.7 представлена временная диаграмма этапа выполнения. Выполнение команды происходит, когда сигналы  $unit\_issue[0] / new\_request = 1$  и  $unit\_wb[0] / done = 1$ .

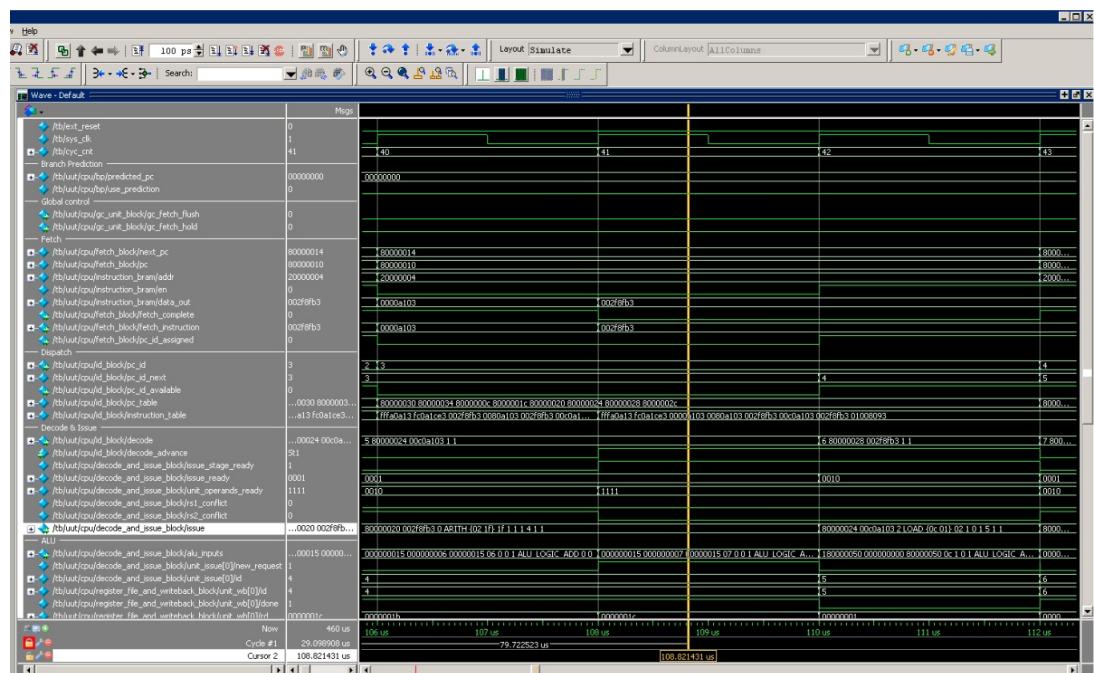


Рисунок 2.7 – Диаграмма этапа выполнения

## 2.4.2 Вывод

Выполнение произошло не на следующем же такте после стадии декодирования (41 и 38), так как возникли конфликты, вызванные тем, что регистр был занят выполнением другой команды.

## 2.5 Задание 5

Выполнить необходимые задания.

### 2.5.1 Часть 1

**Задание:** Сравнить значение регистра x31 (сигнал /tb/register\_file[31]) на момент окончания выполнения программы с тем, который был получен в задании 1.

На рисунке 2.8 представлен результат выполнения программы. Как видно, значение равно 0x9, что соответствует результату, полученному в задании 1.

|      |          |          |  |
|------|----------|----------|--|
| [29] | 00000000 | 00000000 |  |
| [30] | 00000000 | 00000000 |  |
| [31] | 00000009 | 00000009 |  |

Рисунок 2.8 – Результат выполнения программы

## 2.5.2 Часть 2

**Задание:** Получить снимок экрана, содержащий временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом `#!`

Для программы из Варианта 19: `add x31, x0, x2 #!`

На рисунках 2.9-2.12 представлены этапы выполнения команды.

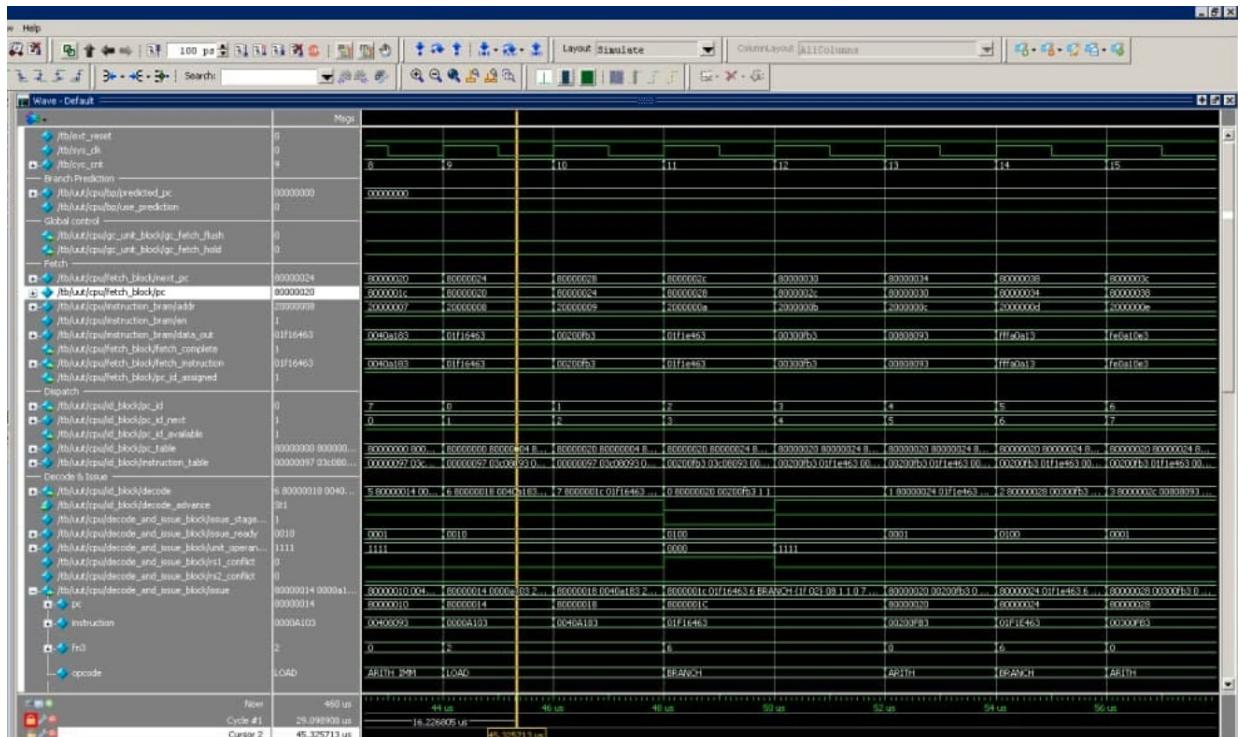


Рисунок 2.9 – Выборка команды с символом `#!`

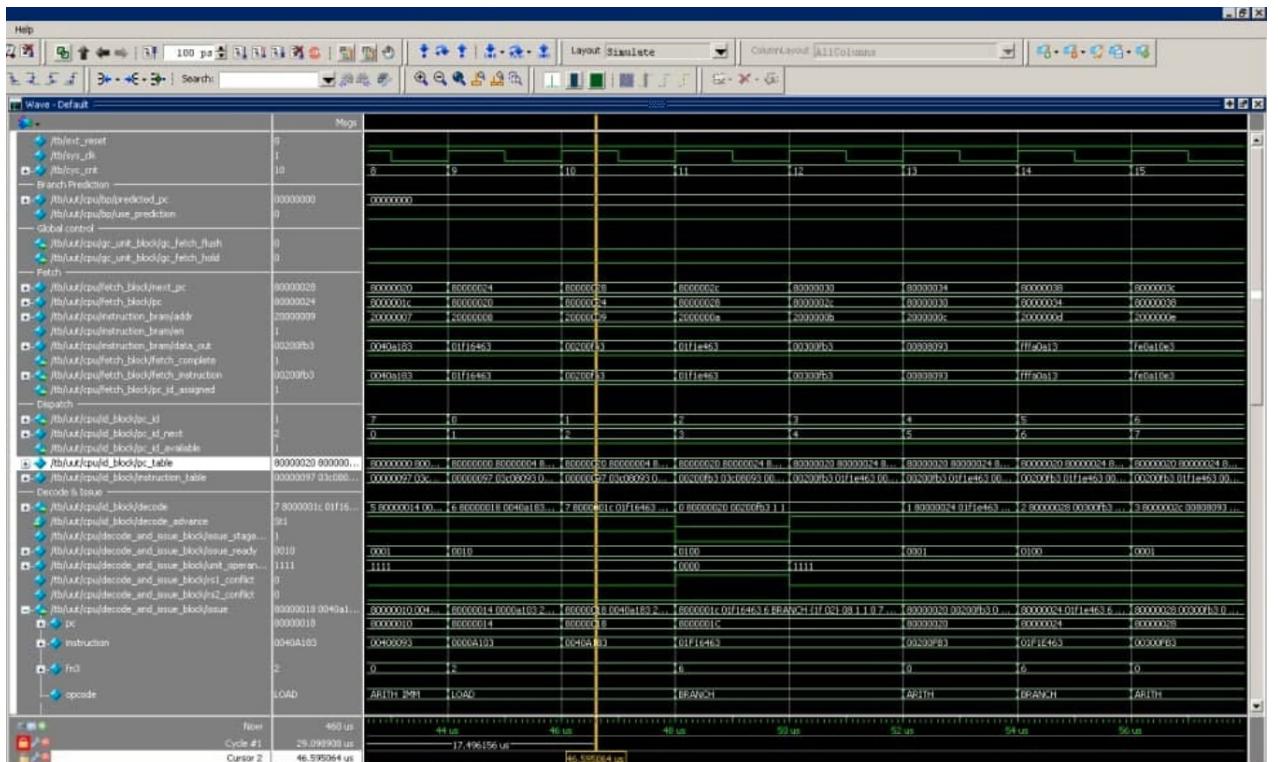


Рисунок 2.10 – Диспетчеризация команды с символом #!

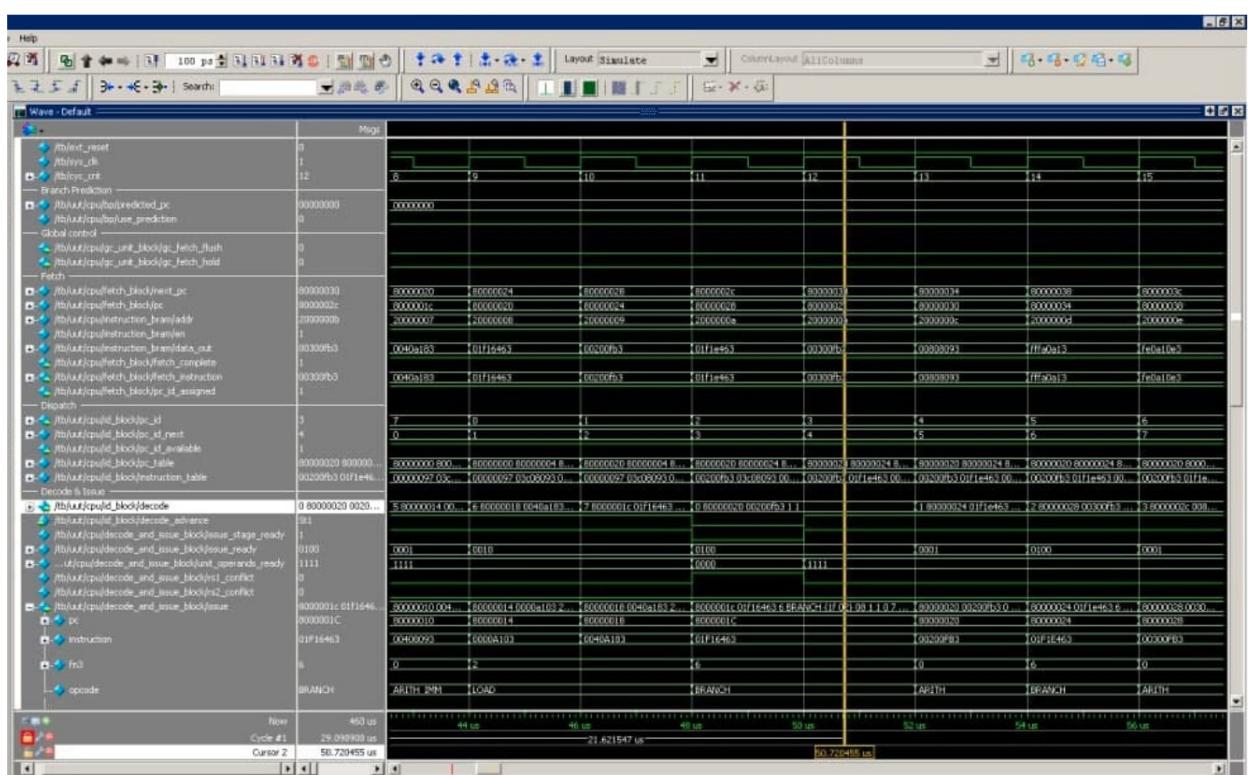


Рисунок 2.11 – Декодирование команды с символом #!

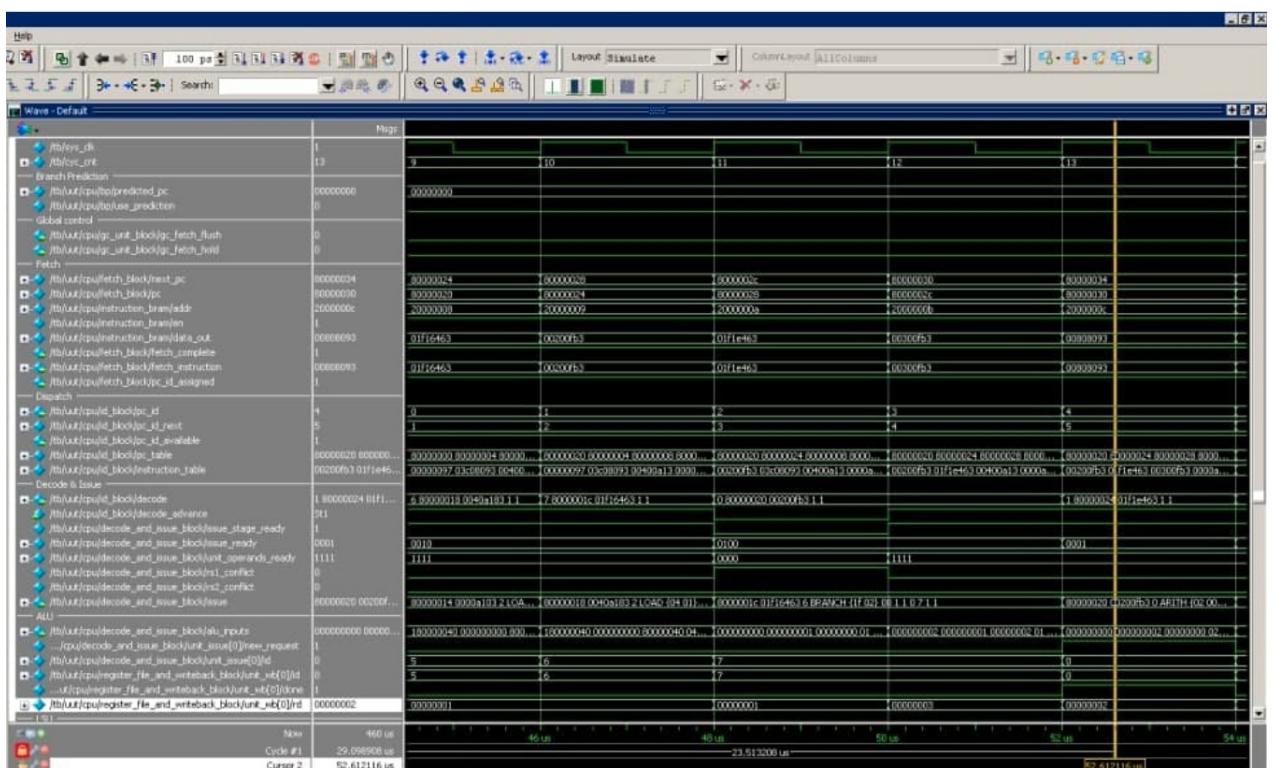


Рисунок 2.12 – Выполнение команды с символом  $\#$ !

## 2.5.3 Часть 3

**Задание:** Анализируя диаграмму, заполнить трассу выполнения программы.

Мы.

На рисунке 2.13 представлена трасса программы.

| Адрес            | Код команды                  | Команда                       | id | Номер такта   |
|------------------|------------------------------|-------------------------------|----|---|
| 00000000 <start> | 00000097                     | adlpc x1,0xb                  | 0  | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 |
| 00000004         | 03c08093                     | addi x1,x1,60 # 8000003c <_x> | 1  | F ID D AL   |
| 00000008         | 00400a13                     | addi x20,x0,4                 | 2  | F ID D AL   |
| 0000000c         | 00000f83                     | lw x31,0(x1)                  | 3  | F ID D M1[M0]M3   |
| 00000010         | 00400809                     | addi x1,x1,4                  | 4  | F ID D AL   |
| 00000014 <l1>    | 00000103                     | lw x2,0(x1)                   | 5  | F ID D M1[M2]M3   |
| 00000016         | 00400803                     | lw x3,4(x1)                   | 6  | F ID D M1[M2]M3   |
| 0000001c         | 01f1e463                     | bltu x2,x3,0x80000024 <l1t>   | 7  | F ID D C B  |
| 00000020         | 00000f63                     | add x2,x0,x2                  | 8  | F ID W D AL   |
| 00000024 <l1t>   | 01f1e463                     | bltu x3,x3,0x8000002c <l2t>   | 1  | F ID W D B  |
| 00000028         | 00300f63                     | add x3,x0,x3                  | 2  | F ID W D AL   |
| 0000002c <l2t>   | 008008093                    | addi x1,x1,8                  | 3  | F ID W D AL   |
| 00000030         | 00000103                     | lw x20,x0,80000014 <l1>       | 4  | F ID W D AL   |
| 00000034         | fc01c0e3                     | bee x20,x0,80000014 <l1>      | 5  | F ID W D AL   |
| 00000038 <l1>    | 00000103                     | lw x2,0(x1)                   | 5  | F ID D M1[M0]M3   |
| 0000003a         | 00400a13                     | lw x3,4(x1)                   | 6  | F ID D M1[M2]M3   |
| 00000040         | 01f1e463                     | bltu x2,x3,0x80000024 <l1t>   | 7  | F ID D C B  |
| 00000044         | 00000001                     | c-addi x0,0                   | 7  | F ID W D X  |
| 00000048         | 00000002 <invalid operation> |                               | 8  | F ID W DX   |
| 00000052         | 00000002 <invalid operation> |                               | 1  | F ID X  |
| 00000056         | 00000003 <invalid operation> |                               | 2  | FX  |
| 0000005a <l1p>   | 00000103                     | lw x2,0(x1)                   | 5  | F ID D M1[M0]M3   |
| 0000005c         | 00400a13                     | lw x3,4(x1)                   | 6  | F ID D M1[M2]M3   |
| 00000060         | 01f1e463                     | bltu x2,x3,0x80000024 <l1t>   | 7  | F ID D C B  |
| 00000064 <l1t>   | 01f1e463                     | add x2,x3,0x8000002c <l2t>    | 1  | F ID W D AL   |
| 00000068         | 00300f63                     | add x3,x0,x3                  | 2  | F ID W D B  |
| 00000072 <l2t>   | 008008093                    | addi x1,x1,8                  | 3  | F ID W D AL   |
| 00000078         | 0fffa0a13                    | addi x20,x0,80000014 <l1>     | 4  | F ID W D B  |
| 00000082         | fc01c0e3                     | bee x20,x0,80000014 <l1>      | 5  | F ID W D B  |
| 00000084 <l1p>   | 00000103                     | lw x2,0(x1)                   | 5  | F ID D M1[M0]M3   |
| 00000086         | 00400a13                     | lw x3,4(x1)                   | 6  | F ID D M1[M2]M3   |
| 00000090         | 01f1e463                     | bltu x2,x3,0x80000024 <l1t>   | 7  | F ID D C B  |
| 00000094         | 00200f53                     | add x31,x0,x2                 | 8  | F ID W D AL   |
| 00000098 <l1t>   | 01f1e463                     | bltu x3,x3,0x8000002c <l2t>   | 1  | F ID W D B  |
| 0000009c         | 00300f53                     | add x31,x0,x3                 | 2  | F ID W D AL   |
| 000000a0 <l2t>   | 008008093                    | addi x1,x1,8                  | 3  | F ID W D AL   |
| 000000a4         | 0fffa0a13                    | addi x20,x0,80000014 <l1>     | 4  | F ID W D B  |
| 000000a8         | fc01c0e3                     | bee x20,x0,80000014 <l1>      | 5  | F ID W D B  |
| 000000a0 <l1p>   | 00000103                     | lw x2,0(x1)                   | 5  | F ID D M1[M0]M3   |
| 000000a6         | 00400a13                     | lw x3,4(x1)                   | 6  | F ID D M1[M2]M3   |
| 000000b0         | 01f1e463                     | bltu x2,x3,0x80000024 <l1t>   | 7  | F ID D C B  |
| 000000b4 <l1t>   | 01f1e463                     | add x2,x3,0x8000002c <l2t>    | 1  | F ID W D B  |
| 000000b8         | 00300f63                     | add x31,x0,x3                 | 2  | F ID W D AL   |
| 000000bc <l2t>   | 008008093                    | addi x1,x1,8                  | 3  | F ID W D AL   |
| 000000c0         | 0fffa0a13                    | addi x20,x0,80000014 <l1>     | 4  | F ID W D B  |
| 000000c4         | fc01c0e3                     | bee x20,x0,80000014 <l1>      | 5  | F ID W D B  |
| 000000c8 <l1p>   | 00000103                     | lw x2,0(x1)                   | 5  | F ID D M1[M0]M3   |
| 000000cc         | 00400a13                     | lw x3,4(x1)                   | 6  | F ID D M1[M2]M3   |
| 000000d0         | 01f1e463                     | bltu x2,x3,0x80000024 <l1t>   | 7  | F ID D C B  |
| 000000d4 <l1t>   | 01f1e463                     | add x31,x0,x3                 | 8  | F ID W D B  |
| 000000d8         | 00300f63                     | add x31,x0,x3                 | 1  | F ID W D B  |
| 000000dc <l2t>   | 008008093                    | addi x1,x1,8                  | 2  | F ID W D AL   |
| 000000e0         | 0fffa0a13                    | addi x20,x0,80000014 <l1>     | 3  | F ID W D AL   |
| 000000e4         | fc01c0e3                     | bee x20,x0,80000014 <l1>      | 4  | F ID W D AL   |
| 000000e8 <l1p>   | 00000103                     | lw x2,0(x1)                   | 5  | F ID D M1[M0]M3   |
| 000000f2         | 00400a13                     | lw x3,4(x1)                   | 6  | F ID D M1[M2]M3   |
| 000000f6         | 01f1e463                     | bltu x2,x3,0x80000024 <l1t>   | 7  | F ID D C B  |
| 000000f0 <_x>    | 00000001                     | addi x0,0                     | 7  | F ID D X  |
| 000000f4         | 00000002 <invalid operation> |                               | 8  | F ID X  |
| 000000f8         | 00000003 <invalid operation> |                               | 1  | FX  |
| 000000f2 <l1p>   | 00000f6f                     | jal x0,80000038 <l1p>         | 6  | F ID D B  |
| 000000f6 <l1p>   | 00000f6f                     | jal x0,80000038 <l1p>         | 7  | F ID D B  |

Рисунок 2.13 – Трасса программы

## 2.5.4 Часть 4

**Задание:** Оптимизировать программу с целью устранения конфликтов.

Как видно из трассы для неоптимизированной программы на рисунке 2.13, во время работы программы произошло 4 конфликта, которые связаны с командой bltu x2, x31, lt1.

В качестве оптимизации переставим некоторые команды местами. Оптимизированный код представлен на рисунке 2.14, на рисунке 2.15 – дизассемблированный код оптимизированной программы, а на рисунке 2.16 – трасса оптимизированной программы. Как видно, удалось сократить количество конфликтов до 1.

```
C:\User\Tsvetkov\New\riscv-lab\src\var19_optimized.s - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
var19.s var19_optimized.s
1 .section .text
2 .globl _start;
3 len = 9 #Размер массива
4 enroll = 2 #Количество обрабатываемых элементов за одну итерацию
5 elem_sz = 4 #Размер одного элемента массива
6
7 _start:
8     la x1, _x
9     addi x20, x0, (len-1)/enroll
10    lw x31, 0(x1)
11    addi x1, x1, elem_sz*1
12    lw x2, 0(x1)
13    lw x3, 4(x1)
14 lp:
15     bltu x2, x31, lt1
16     add x31, x0, x2 #!
17 ltl:
18     bltu x3, x31, lt2
19     add x31, x0, x3
20     add x1, x1, elem_sz*enroll
21     addi x20, x20, -1
22     lw x2, 0(x1)
23     lw x3, 4(x1) ; Conflict line
24     bne x20, x0, lp
25 lp2: j lp2
26
27 .section .data
28 _x:
29     .4byte 0x1
30     .4byte 0x2
31     .4byte 0x3
32     .4byte 0x4
33     .4byte 0x5
34     .4byte 0x6
35     .4byte 0x7
36     .4byte 0x8
37     .4byte 0x9
```

Рисунок 2.14 – Оптимизированный код программы

```

MINGW32:/c/User/TsvetkovNew/riscv-lab/src
riscv64-unknown-elf-objdump -D -M numeric,no-aliases -t var19_optimized.elf

var19_optimized.elf:      file format elf32-littleriscv

SYMBOL TABLE:
80000000 l  d .text 00000000 .text
80000044 l  d .data 00000000 .data
00000000 1  df "%ABS%" 00000000 var19_optimized.o
00000009 1  "ABS%" 00000000 len
00000021 1  "ABS%" 00000000 enroll
00000004 1  "ABS%" 00000000 elem_sz
80000044 l  .data 00000000 _x
80000001c l  .text 00000000 lp
80000024 l  .text 00000000 lt1
8000002c l  .text 00000000 lt2
80000040 l  .text 00000000 lp2
80000000 g  .text 00000000 _start
80000068 g  .data 00000000 _end

Disassembly of section .text:
80000000 <_start>:
80000000: 00000097          auipc   x1,0x0
80000004: 04408093          addi    x1,x1,68 # 80000044 <_x>
80000008: 00400a13          addi    x20,x0,4
8000000c: 0000aaf83         lw      x31,0(x1)
80000010: 00408093          addi    x1,x1,4
80000014: 0000a103          lw      x2,0(x1)
80000018: 0040a183          lw      x3,4(x1)

80000001c <lp:>
8000001c: 01f16463          bltu   x2,x31,80000024 <lt1>
80000020: 00200fb3          add    x31,x0,x2

80000024 <lt1>:
80000024: 01f16463          bltu   x3,x31,8000002c <lt2>
80000028: 00300fb3          add    x31,x0,x3

8000002c <lt2>:
8000002c: 00808093          addi    x1,x1,8
80000030: ffffffa013        addi    x20,x20,-1
80000034: 0000a103          lw      x2,0(x1)
80000038: 0040a183          lw      x3,4(x1)
8000003c: fe0a10e3          bne   x20,x0,80000001c <lp>

80000040 <lp2>:
80000040: 0000006f          jal    x0,80000040 <lp2>

Disassembly of section .data:
80000044 <x>:
80000044: 0001              c.addi  x0,0
80000046: 0000              unimp
80000048: 0002              0x2
8000004a: 0000              unimp
8000004c: 00000003          lb     x0,0(x0) # 0 <enroll-0x2>
80000050: 0004              c.addi4spn x9,x2,0
80000052: 0000              unimp
80000054: 0005              c.addi  x0,1
80000056: 0000              unimp
80000058: 0006              0x6
8000005a: 0000              unimp
8000005c: 00000007          0x7
80000060: 0008              c.addi4spn x10,x2,0
80000062: 0000              unimp

```

Рисунок 2.15 – Дизассемблированный код оптимизированной программы

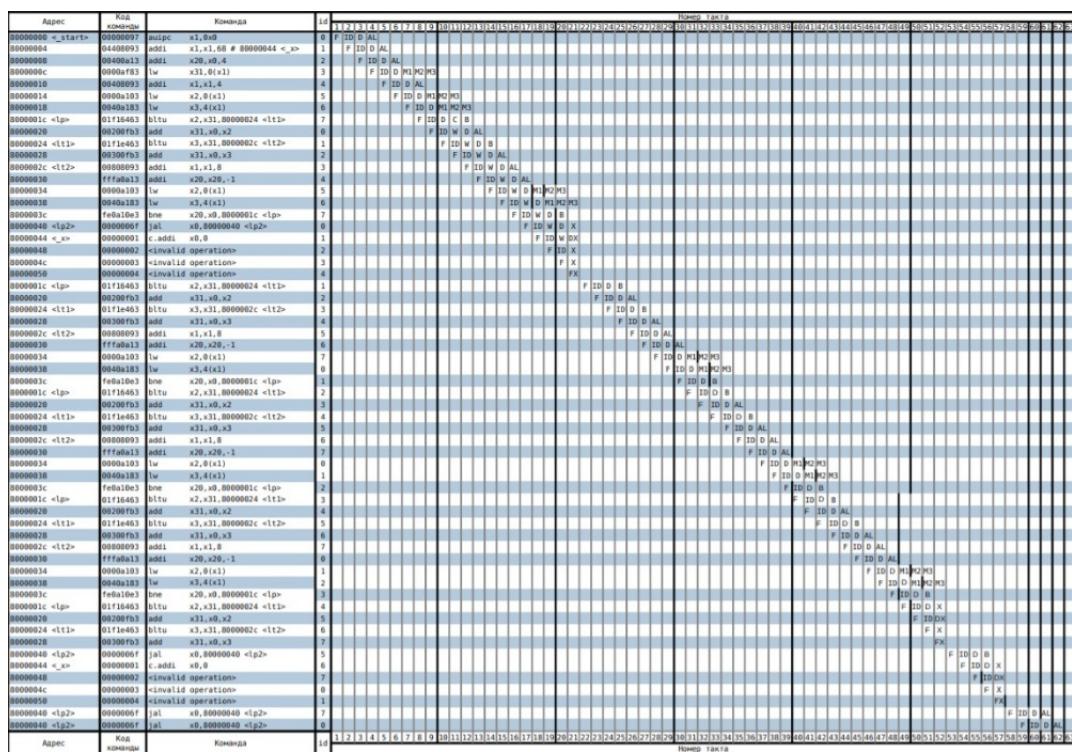


Рисунок 2.16 – Трасса оптимизированной программы

**Вывод:** оптимизация программ позволяет устраниć конфликты. В данной ситуации удалось устраниć 3 конфликта из 4, переставив местами некоторые команды. Это говорит о том, что правильное расположение команд имеет большое значение при работе с программами.

# Заключение

В данной лабораторной работе были рассмотрены принципы построения и работы с архитектурой суперскалярных конвейрных микропроцессоров. Также рассмотрена работа с микропроцессорным ядром RISC-V.