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 №1

по курсу «Архитектура ЭВМ»

на тему: *«Изучение принципов работы микропроцессорного ядра RISC-V»*

Вариант: **17**

Студент группы ИУ7-51Б _____ (Подпись, дата) Савинова М. Г. (Фамилия И.О.)

Преподаватель _____ (Подпись, дата) Ибрагимов С. В. (Фамилия И.О.)

Москва — 2023 г.

Содержание

1 Цели

1) Ознакомиться с принципами функционирования построения и особенностями архитектуры суперскалярных конвейерных микропроцессоров;

2) Ознакомиться с принципами проектирования и верификации сложных цифровых устройств с использованием языка описания аппаратуры *SystemVerilog* и *ПЛИС*.

2 Задания

2.1 Задание 1

Содержимое файла *mine.s*, который соответсвует **варианту 17**:

```
1     .section .text
2     .globl _start;
3     len = 9 #Размер массива
4     enroll = 2 #Количество обрабатываемых элементов за одну итер
          ацию
5     elem_sz = 4 #Размер одного элемента массива
6
7 _start:
8     la x1, _x
9     addi x20, x1, elem_sz*len #Адрес элемента, следующего за пос
          ледним
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    bne x1, x20, lp
22 lp2: j lp2
23
24    .section .data
25 _x: .4byte 0x1
26    .4byte 0x2
27    .4byte 0x3
28    .4byte 0x4
29    .4byte 0x5
30    .4byte 0x6
31    .4byte 0x7
32    .4byte 0x8
33    .4byte 0x9
```

Псевдокод:

```
#define len 9
#define enroll 2
#define elem_sz 4

int _x[] = [1, 2, 3, 4, 5, 6, 7, 8, 9];

void start() {
    int *x1 = _x;
    int *x20 = x1 + elem_sz * len;

    int x31 = _x[0];
    int x1 += 1;

    do {
        x2 = x1[0];
        x3 = x1[1];

        if (x2 >= x31)
            x31 = x2;

        if (x3 >= x31)
            x31 = x3;

        x1 += enroll;
    } while (x1 != x20);
    while (1) {}
}
```

Результат выполнения компиляции:

```
Disassembly of section .text:

80000000 <_start>:
80000000:       00000097                auipc   x1,0x0
80000004:       03808093                addi    x1,x1,56 #
   80000038 <_x>
80000008:       02408a13                addi    x20,x1,36
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:       01f16463                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:       ff4092e3                bne     x1,x20,80000014
   <lp>

80000034 <lp2>:
80000034:       0000006f                jal     x0,80000034
   <lp2>
```

Содержимое файла *mine.hex:*

```
00000097
03808093
02408a13
0000af83
00408093
0000a103
0040a183
01f16463
00200fb3
01f1e463
00300fb3
00808093
ff4092e3
0000006f
00000001
00000002
00000003
00000004
00000005
00000006
00000007
```

```
00000008
00000009
```

В регистре x31 в конце выполнения программы должно содержаться **максимальное значение** в массива. В нашем случае: **9**.

2.2 Задание 2

Адрес команды: 80000020; *итерация:* 2.

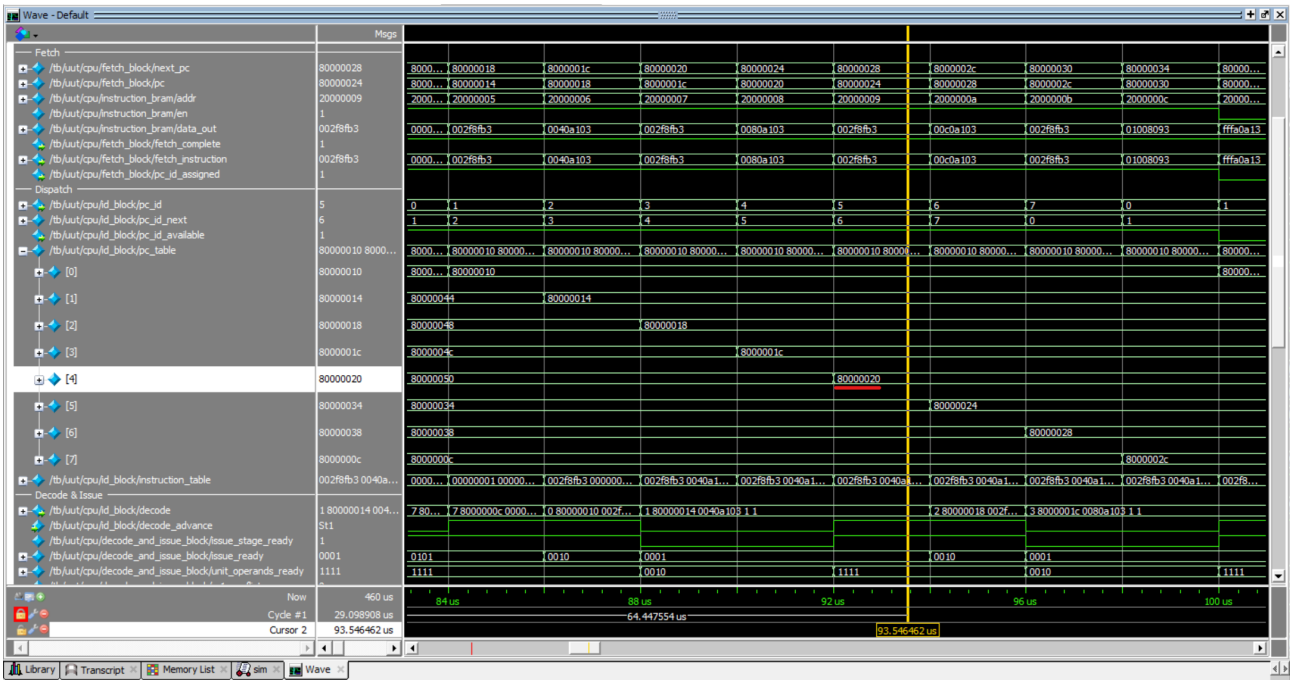

Рисунок 2.1 – Выборка и диспетчерезация

2.3 Задание 3

Адрес команды: 8000002c; *итерация:* 2.

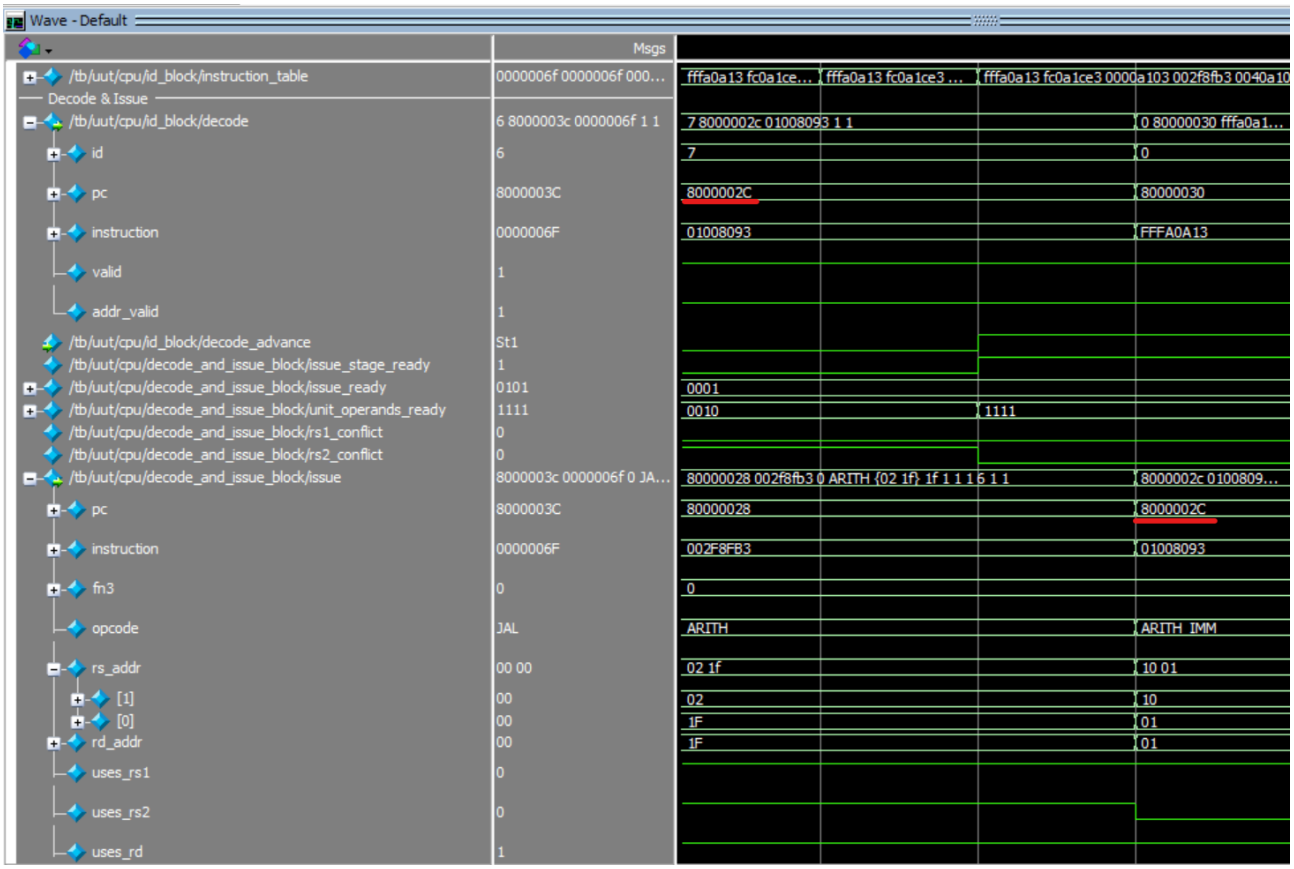

Рисунок 2.2 – Декодирование и планирование

2.4 Задание 4

Адрес команды: 8000018c; *итерация:* 2.

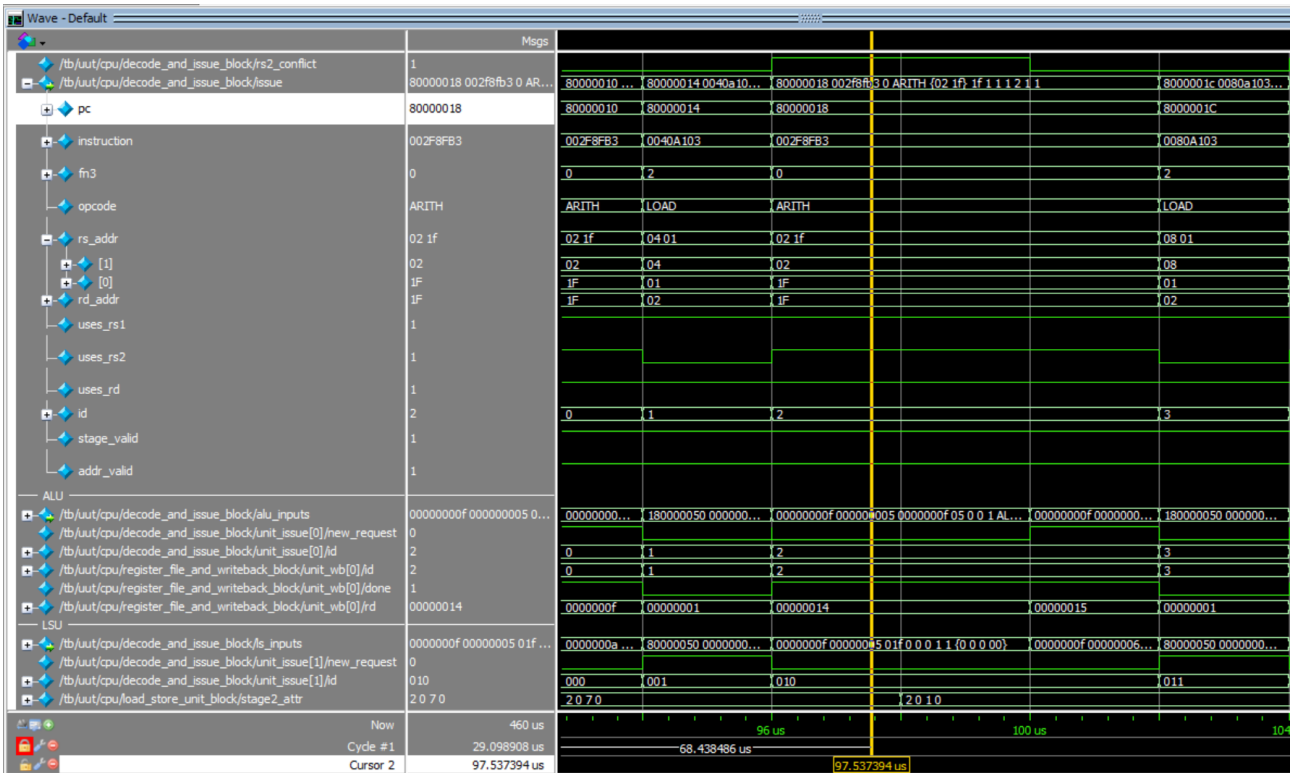

Рисунок 2.3 – Выполнение

2.5 Задание 5

Адрес команды #!: 80000018

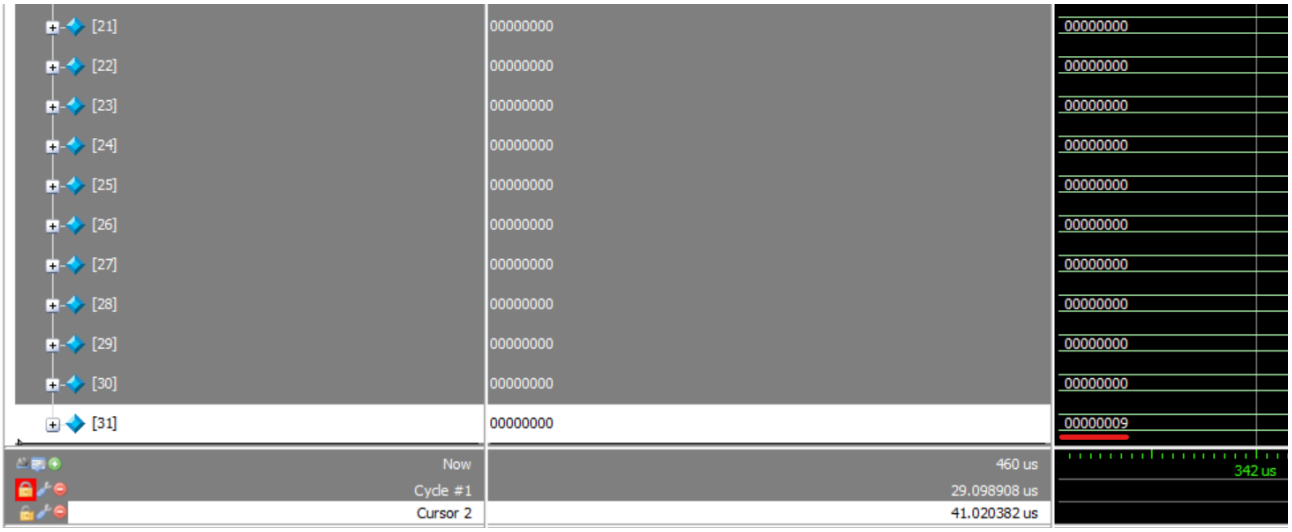

Рисунок 2.4 – Результат в регистре x31

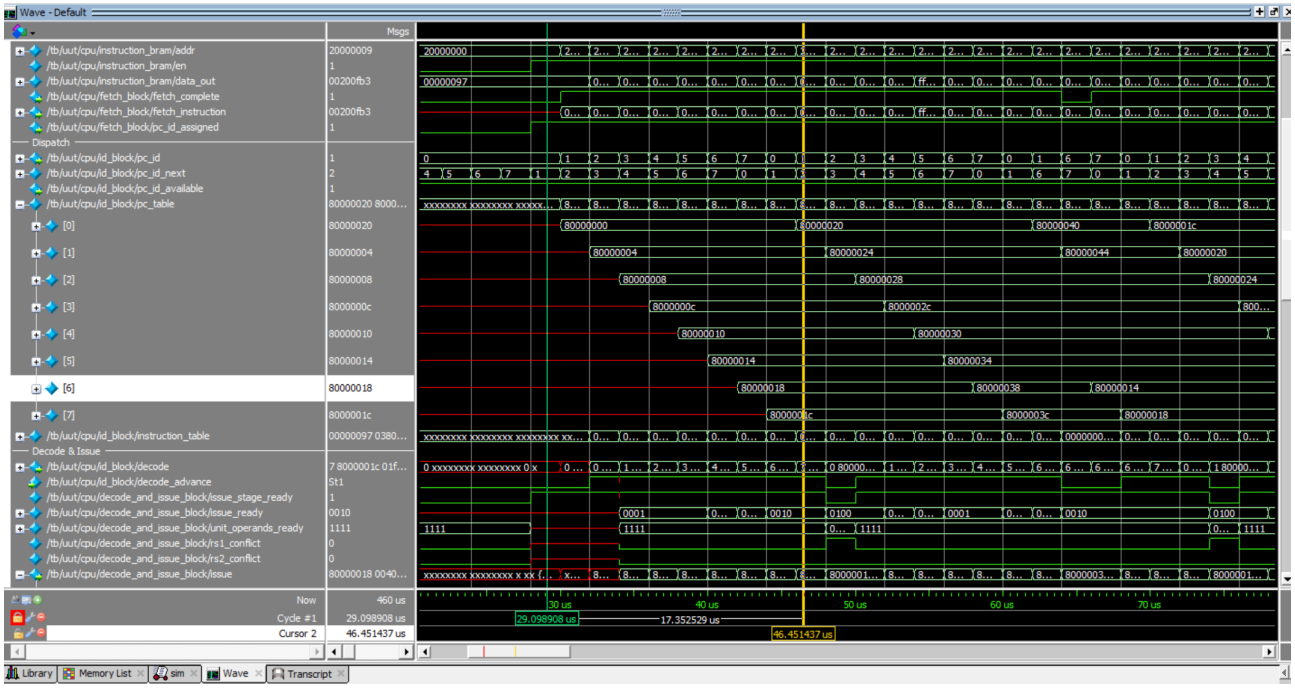

Рисунок 2.5 – Выборка и диспетчерезация

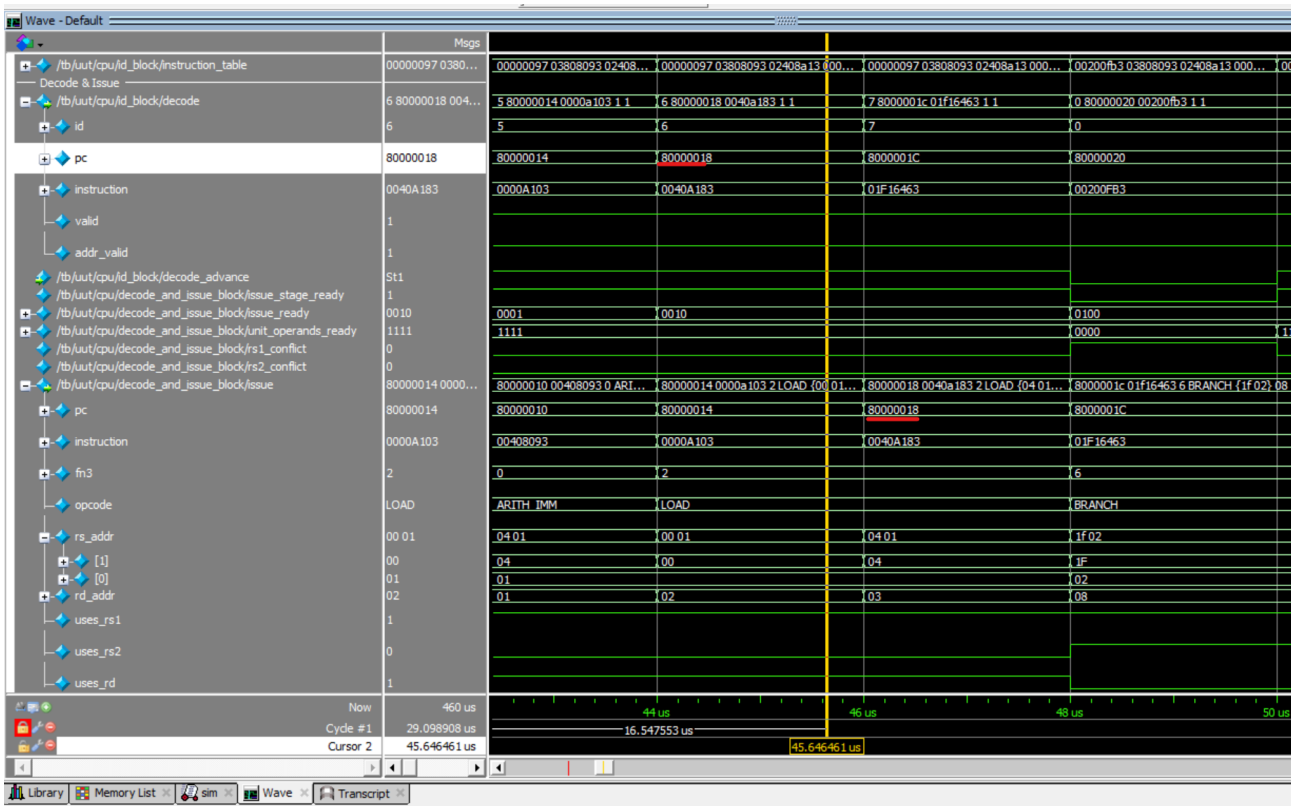

Рисунок 2.6 – Декодирование и планирование

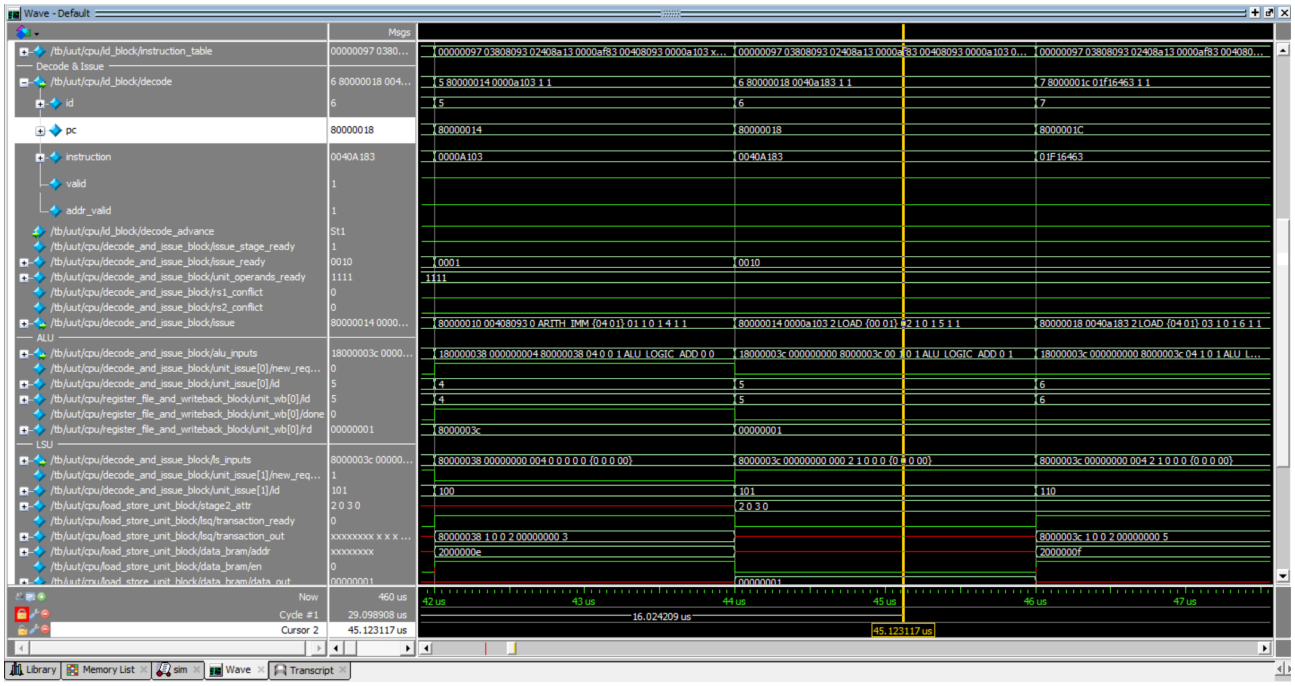

Рисунок 2.7 – Выполнение

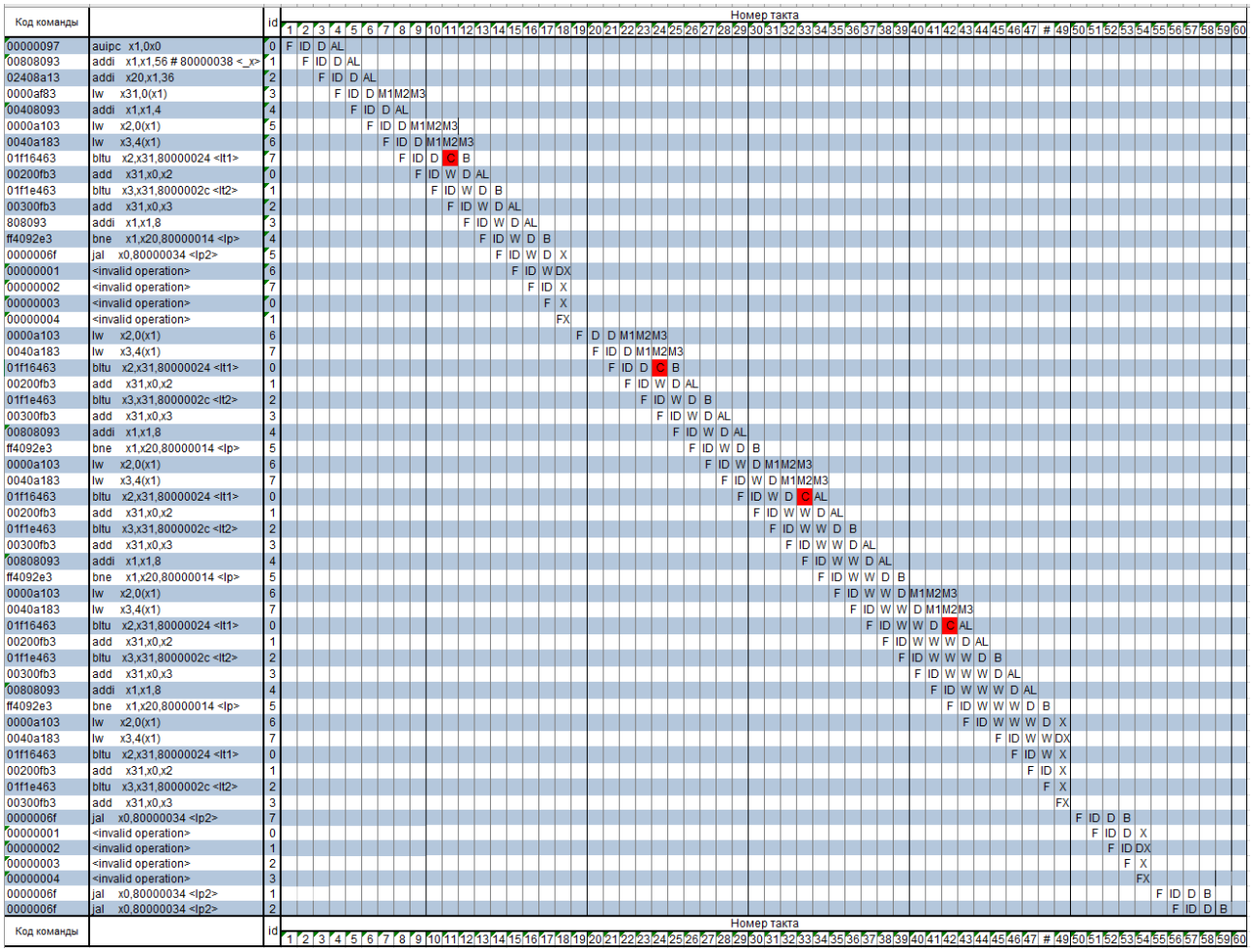

Рисунок 2.8 – Трасса работы программы

Конфликты возникают из-за того, что мы обращаемся к значению в регистре, до того как оно было еще записано. Оптимизировать программу можно тем, что сначала загрузить одно значение, затем увеличить указатель, затем то же само сделать и для второго числа, и только после этого работать с регистрами.

Оптимизированная программа:

```
1     .section .text
2     .globl _start;
3     len = 9 #Размер массива
4     enroll = 2 #Количество обрабатываемых элементов за одну итер
        ацию
5     elem_sz = 4 #Размер одного элемента массива
6
7  _start:
8     la x1, _x
9     addi x20, x1, elem_sz*len #Адрес элемента, следующего за пос
        ледним
10    lw x31, 0(x1)
11    addi x1, x1, elem_sz*1
12 lp:
13    lw x2, 0(x1)
14    addi x1, x1, elem_sz
15    lw x3, 0(x1) #!
16    addi x1, x1, elem_sz
17    bltu x2, x31, lt1
18    add x31, x0, x2
19 lt1:    bltu x3, x31, lt2
20    add x31, x0, x3
21 lt2:
22    bne x1, x20, lp
23 lp2: j lp2
24
25    .section .data
26 _x: .4byte 0x1
27    .4byte 0x2
28    .4byte 0x3
29    .4byte 0x4
30    .4byte 0x5
31    .4byte 0x6
32    .4byte 0x7
33    .4byte 0x8
34    .4byte 0x9
```

Дизассемблированный код:

```
1  Disassembly of section .text:
2
3  80000000 <_start>:
4  80000000:       00000097                auipc   x1,0x0
5  80000004:       03c08093                addi    x1,x1,60 #
     8000003c <_x>
6  80000008:       02408a13                addi    x20,x1,36
7  8000000c:       0000af83                lw      x31,0(x1)
8  80000010:       00408093                addi    x1,x1,4
9
10 80000014 <lp>:
11 80000014:       0000a103                lw      x2,0(x1)
12 80000018:       00408093                addi    x1,x1,4
13 8000001c:       0000a183                lw      x3,0(x1)
14 80000020:       00408093                addi    x1,x1,4
15 80000024:       01f16463                bltu
     x2,x31,8000002c <lt1>
16 80000028:       00200fb3                add     x31,x0,x2
17
18 8000002c <lt1>:
19 8000002c:       01f1e463                bltu
     x3,x31,80000034 <lt2>
20 80000030:       00300fb3                add     x31,x0,x3
21
22 80000034 <lt2>:
23 80000034:       ff4090e3                bne
     x1,x20,80000014 <lp>
24
25 80000038 <lp2>:
26 80000038:       0000006f                jal     x0,80000038
     <lp2>
```

Псевдокод:

```
28 #define len 9
29 #define enroll 2
30 #define elem_sz 4
31
32 int _x[] = [1, 2, 3, 4, 5, 6, 7, 8, 9];
33
34 void start() {
35     int *x1 = _x;
36     int *x20 = x1 + elem_sz * len;
37
38     int x31 = _x[0];
39     int x1 += 1;
40
41     do {
42         x2 = x1[0];
43         x1 += 1;
44
45         x3 = x1[1];
46         x1 += 1;
47
48         if (x2 >= x31)
49             x31 = x2;
50
51         if (x3 >= x31)
52             x31 = x3;
53
54     } while (x1 != x20);
55     while (1) {}
56 }
```

Трасса работы оптимизированной программы:

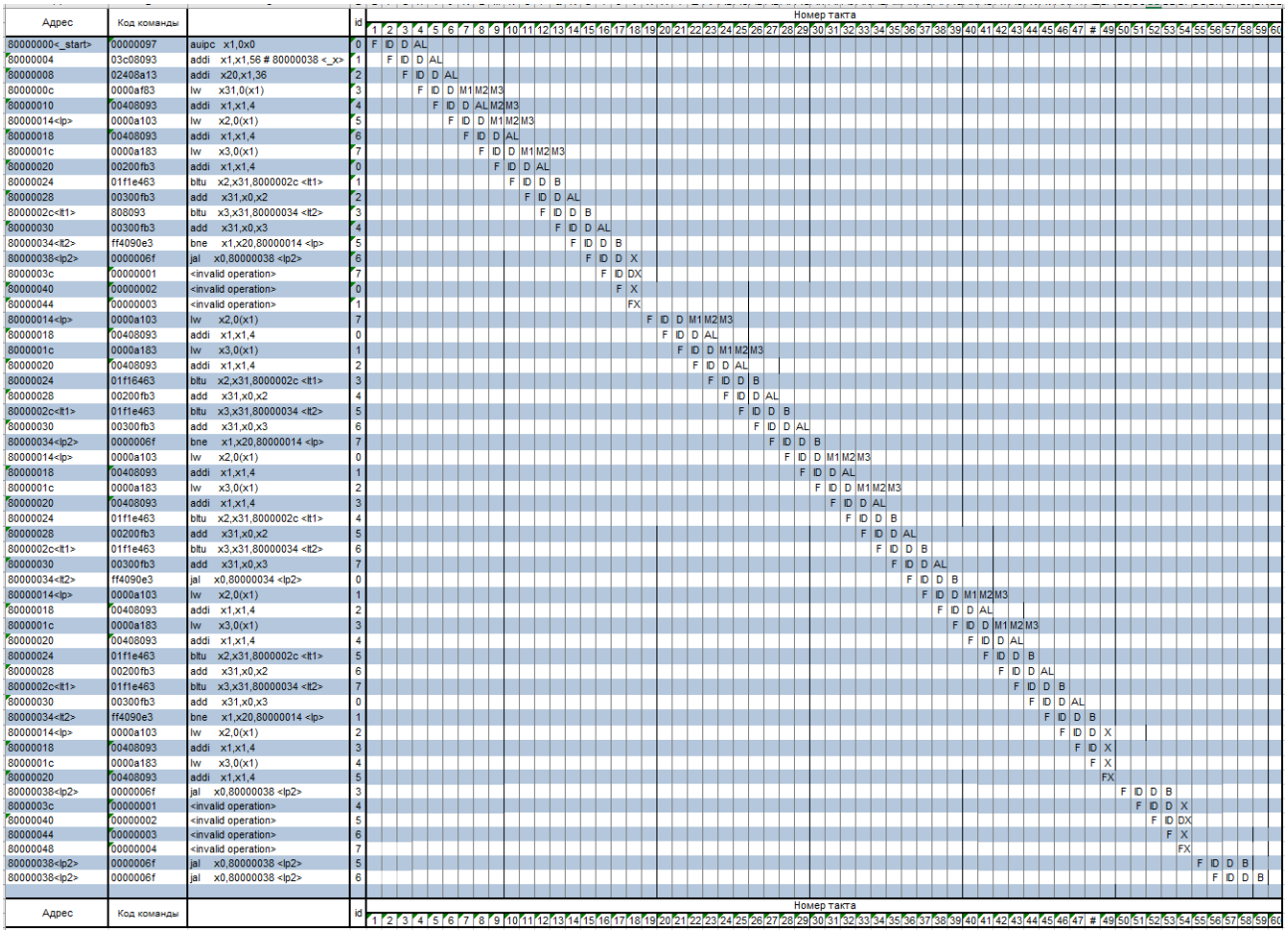

Рисунок 2.9 – Трасса работы оптимизированной программы

3 Заключение

В результате выполнения работы были изучены принципы функционирования, построения и особенности архитектуры суперскалярных конвейерных микропроцессоров.

На основе изученных материалов был найден способ оптимизировать программу.

Поставленная цель достигнута.