



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»

НАПРАВЛЕНИЕ ПОДГОТОВКИ _____ «09.03.04 Программная инженерия»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Название: _____ Изучение принципов работы микропроцессорного ядра RISC-V

Дисциплина: _____ Архитектура ЭВМ

Студент	ИУ7-54Б	_____	С. Д. Параскун
	Группа	Подпись, дата	И. О. Фамилия

Преподаватель	_____	А. Ю. Попов
	Подпись, дата	И. О. Фамилия

Москва, 2022 г.

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.

1. Задание №1

В данной работе будет выполнен 12 вариант.

Листинг 1.1 – Код рассматриваемой программы

```
1  .section .text
2      .globl _start;
3      len = 8 #array length
4      enroll = 4 #count of elems in one loop
5      elem_sz = 4 #size of one element
6
7  _start:
8      la x1, _x
9      addi x20, x1, elem_sz*len #address of last elem
10 lp:
11     lw x2, 0(x1)
12     lw x3, 4(x1)
13     add x31, x31, x2 #!
14     add x31, x31, x3
15     lw x4, 8(x1)
16     lw x5, 12(x1)
17     add x31, x31, x4
18     add x31, x31, x5
19     addi x1, x1, elem_sz*enroll
20     bne x1, x20, lp
21     addi x31, x31, 1
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
```

В регистре $x31$ в конце программы будет находится сумма элементов массива + 1, т.е. значение его будет 37. За один цикл обрабатывается 4 элемента, при этом сначала считываются 2, потом прибавляются к результату, потом считываются еще 2 и прибавляются.

Листинг 1.2 – Псевдокод рассматриваемой программы

```

1 #define len 8
2 #define enroll 4
3 #define elem_sz 4
4 int _x[ ]={1,2,3,4,5,6,7,8};
5 void _start() {
6     int *x1 = _x;
7     int x20 = x1 + len;
8
9     do {
10         int x2 = x1[0];
11         int x3 = x1[1];
12         x31 += x2;
13         x31 += x3;
14         int x4 = x1[2];
15         int x5 = x1[3];
16         x31 += x4;
17         x31 += x5;
18         x1 += enroll;
19     } while(x1 != x20);
20     x31++;
21     while(1){}
22 }

```

Листинг 1.3 – Дизассемблированный листинг рассматриваемой программы

```

1 80000000 <_start>:
2 80000000:      00000097          auipc    x1,0x0
3 80000004:      03c08093          addi     x1,x1,60 # 8000003c <_x>
4 80000008:      02008a13          addi     x20,x1,32
5
6 8000000c <lp>:
7 8000000c:      0000a103          lw       x2,0(x1)
8 80000010:      0040a183          lw       x3,4(x1)
9 80000014:      002f8fb3          add      x31,x31,x2
10 80000018:      003f8fb3          add      x31,x31,x3
11 8000001c:      0080a203          lw       x4,8(x1)
12 80000020:      00c0a283          lw       x5,12(x1)
13 80000024:      004f8fb3          add      x31,x31,x4
14 80000028:      005f8fb3          add      x31,x31,x5
15 8000002c:      01008093          addi     x1,x1,16
16 80000030:      fd409ee3          bne      x1,x20,8000000c <lp>
17 80000034:      001f8f93          addi     x31,x31,1
18
19 80000038 <lp2>:
20 80000038:      0000006f          jal      x0,80000038 <lp2>

```

2. Задание №2

На рисунке ниже рассмотренна временная диаграмма стадий выборки и диспетчеризации команды 8000000с на 2-й итерации цикла. Выборка происходит в 27 такте, так как сигнал `fetch_complete` сброшен в 0, диспетчеризация заканчивается в 29 (заканчивается записью 0000a103 в `instruction_table`).

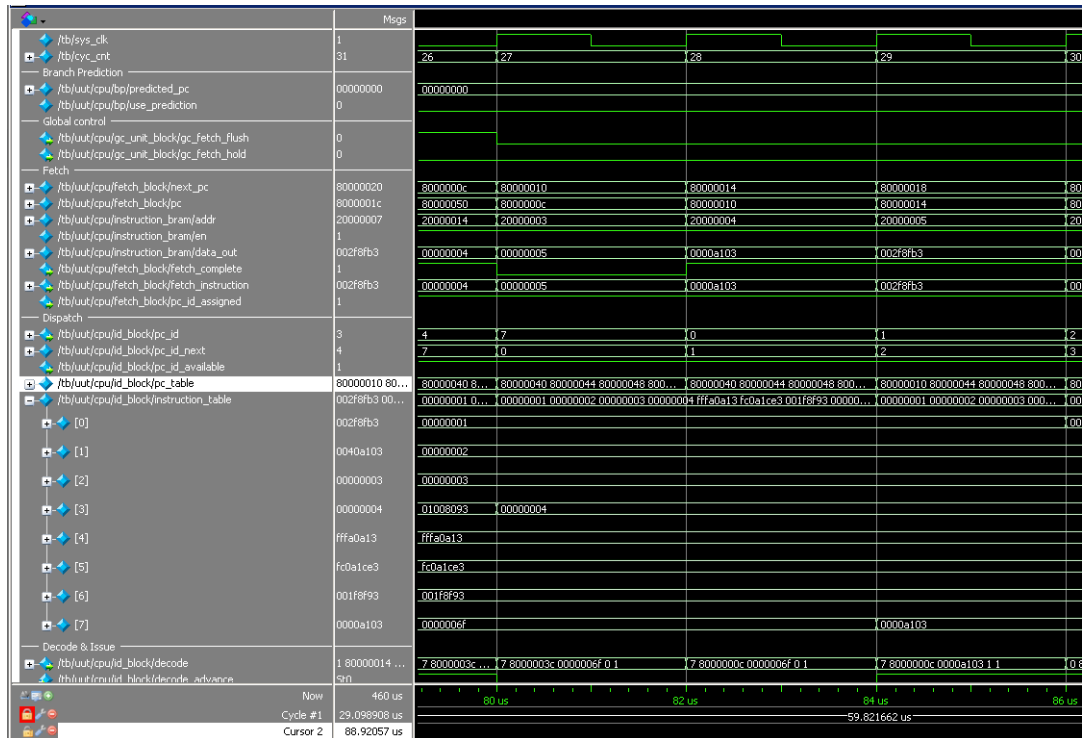


Рисунок 2.1 – Временная диаграмма выполнения стадий выборки и диспетчеризации команды 8000000с на 2-й итерации

3. Задание №3

На рисунке ниже рассмотренна временная диаграмма стадии декодирования и планирования на выполнение команды 80000018 на 2-й итерации цикла. В 34 такте происходит декодирование команды, в 35 она планируется на выполнение, возникает конфликт по регистру rs2, в 36 такте завершается выполнение предыдущей команды.

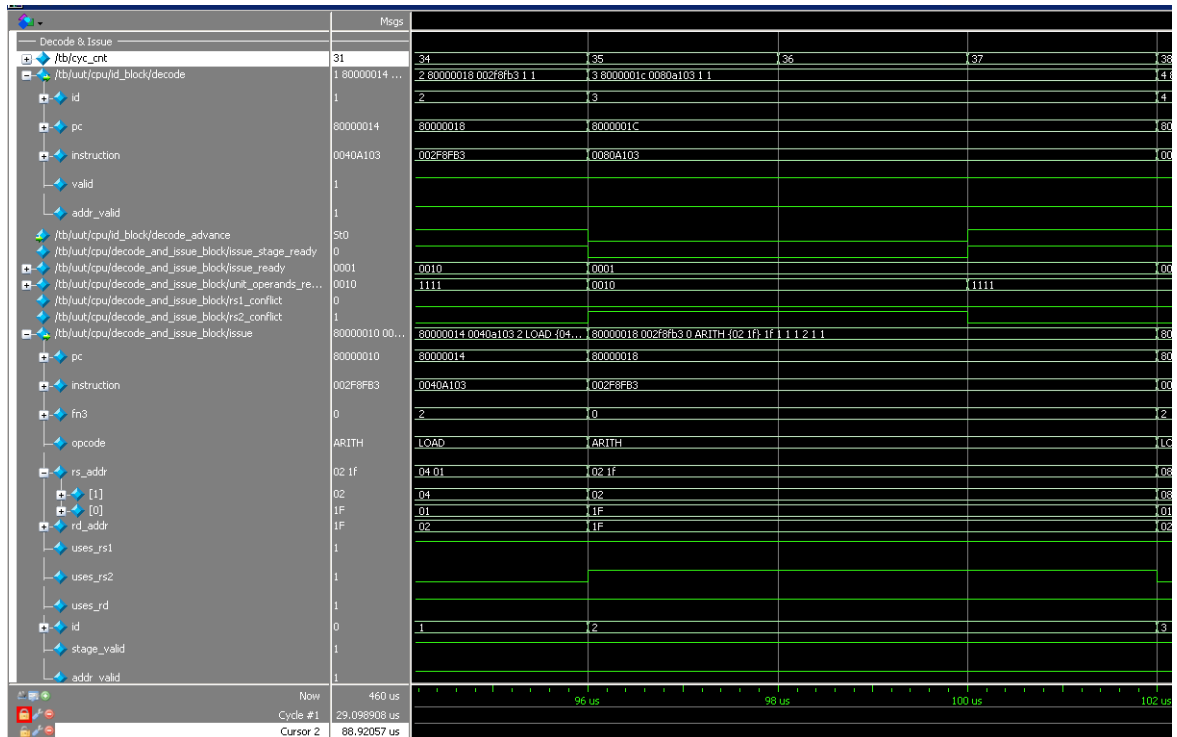


Рисунок 3.1 – Временная диаграмма выполнения стадий декодирования и планирования на выполнение команды 80000018 на 2-й итерации

4. Задание №4

На рисунке ниже рассмотренна временная диаграмма стадии выполнения команды 8000002с на 1-й итерации цикла. Видно, что в 23 такте данная команда поступает на выполнение, так как это арифметическая операция, она выполняется блоком ALU, в unit_wb[0]/id записывается 3 (как и у исследуемой команды), сигнал unit_wb[0]/done выставлен в 1. Следовательно выполнение данной команды завершается в этом же такте.

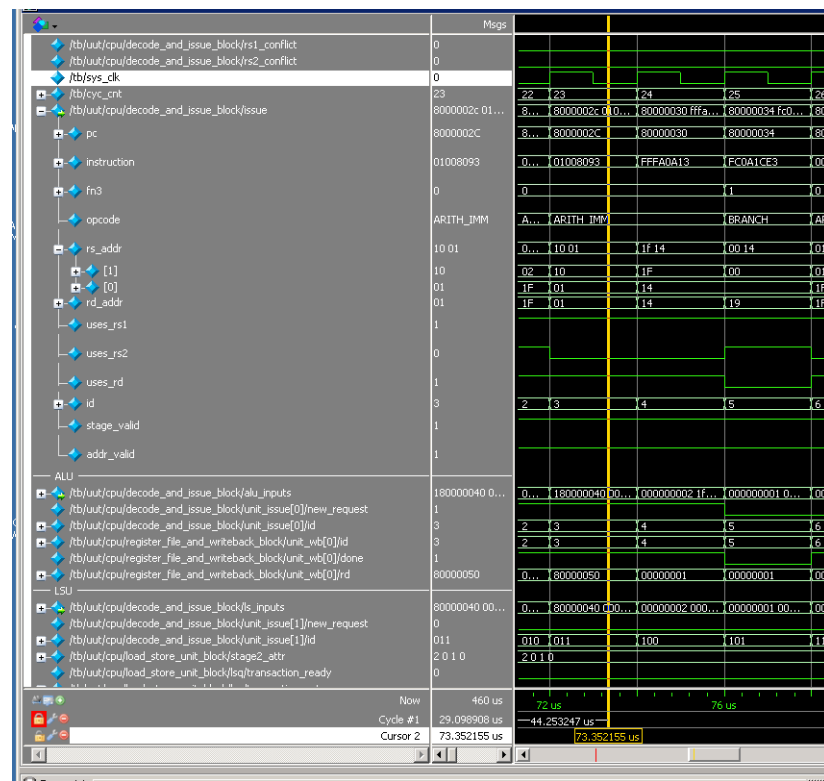


Рисунок 4.1 – Временная диаграмма выполнения стадии выполнения команды 8000002с на 1-й итерации

5. Задание №5

Для начала узнаем значение регистра $x31$, из рисунка ниже видно, что вычисленное в первом задании значение совпадает с хранящимся там

$$00000025(hex) = 37(dec) \quad (5.1)$$

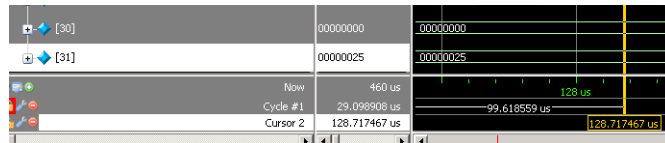


Рисунок 5.1 – Значение регистра $x31$ после выполнения программы

Символом #! в программе помечена команда `addi x31, x31, x2`. Рассмотрим стадии ее выполнения. На рисунке 5.1 представлены стадии выборки (6 такт), диспетчеризации (7 такт) и декодирования (8 такт).

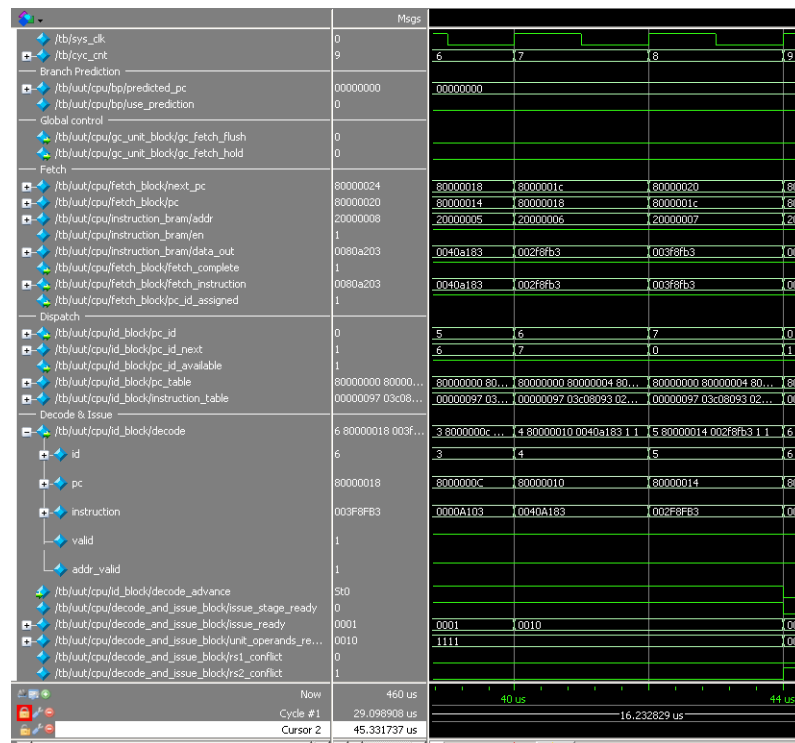


Рисунок 5.2 – Выборка, диспетчеризация и декодирование команды

В 9 такте команда планируется на выполнение, но так как сигнал `rs2_conflict` равен 1 возникает конфликт. В следующем такте этот конфликт разрешается, управление передается блоку ALU, о чем свидетельствует совпадение

issue/id и unit_wb[0]/id, они равны 5. В 10 такте выполнение завершается, о чем свидетельствует 1 на сигнале unit_wb[0]/done, и в следующем такте переходит к выполнению следующей команды.

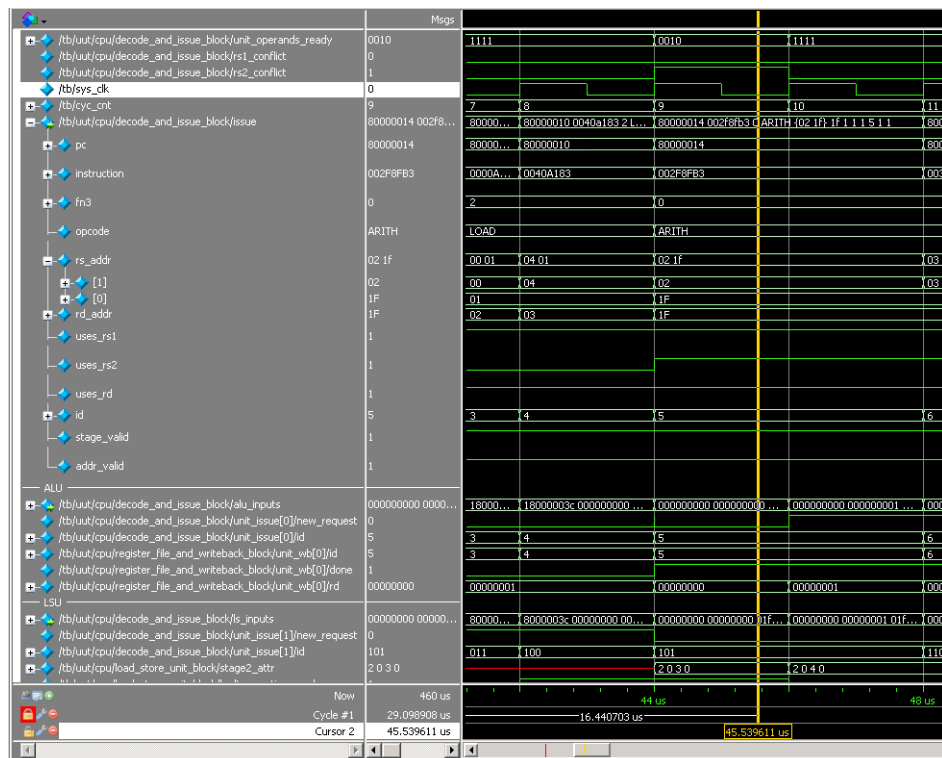


Рисунок 5.3 – Планирование на выполнение и выполнение команды

Ниже представлена временная диаграмма сигналов выполнения программы по варианту 12.

[illegible]

Рисунок 5.4 – Временная диаграмма сигналов выполнения программы

Из трассы видно, что выполнение программы заканчивается выполнением команды `add x31, x31, 1` в 39 такте. То есть выполнение программы происходит с 1 по 39 такт. Из них планирование на выполнение не происходило в течение 16 тактов (7, 8, 12, 13, 19 - 24, 28, 29, 35 - 38). Это составляет 41%. Чтобы сократить время простоя можно провести оптимизацию путем перестановки команд.

В качестве оптимизации было выбрано перемещение строк загрузки данных друг к другу. Оптимизированный код представлен на листинге ниже.

Листинг 5.1 – Код оптимизированной программы

```
1  .section .text
2      .globl _start;
3      len = 8 #array length
4      enroll = 4 #count of elems in one loop
5      elem_sz = 4 #size of one element
6
7  _start:
8      la x1, _x
9      addi x20, x1, elem_sz*len #address of last elem
10 lp:
11     lw x2, 0(x1)
12     lw x3, 4(x1)
13     lw x4, 8(x1)
14     lw x5, 12(x1)
15     add x31, x31, x2 #!
16     add x31, x31, x3
17     add x31, x31, x4
18     add x31, x31, x5
19     addi x1, x1, elem_sz*enroll
20     bne x1, x20, lp
21     addi x31, x31, 1
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
```

Временная диаграмма сигналов выполнения представлена ниже.

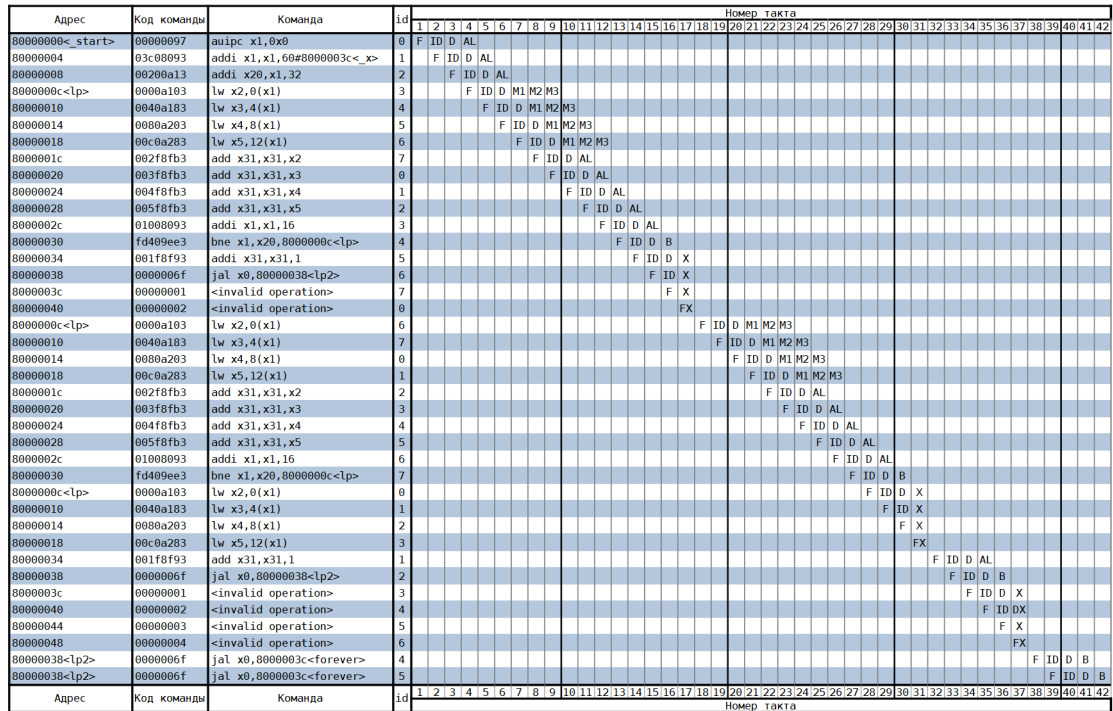


Рисунок 5.5 – Временная диаграмма сигналов выполнения оптимизированной программы

Из трассы видно, что выполнение программы заканчивается выполнением команды add x31, x31, 1 в 35 такте. То есть выполнение программы происходит с 1 по 35 такт. Из них планирование на выполнение не происходило в течение 12 тактов (7, 8, 17 - 22, 31 - 34). Это составляет 31%. Удалось сократить простой на 10%.