

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»	
КАФЕЛРА «Программное обеспечение ЭВМ и информационные технолог	ии»

ОТЧЕТ по лабораторной работе N 2

Название Изучение принципов работы г	микропроцессорного ядра	a RISC-V	
Дисциплина Архитектура элекронно-вы	числительных машин		
Студент:		Динь Вьет Ань	
Преподаватель:	подпись, дата	Фамилия, И.О. Попов А. Ю.	
преподаватель.	подпись, дата	<u> </u>	

Содержание

Ц	ель ј	работы	3
1	Осн	ювные теоретические сведения	4
	1.1	Модель памяти	4
	1.2	Система команд	4
2	Обі	цая программа	5
3	Рез	ультаты исследования программы	9
	3.1	Задание №1	9
	3.2	Задание №2	14
	3.3	Задание №3	17
	3.4	Задание №4	18
	3.5	Задание №5	19
За	клю	рчение	27
Π_1	рилс	эжение	28

Цель 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 Основные теоретические сведения

RISC-V является открытым современным набором команд, который может использоваться для построения как микроконтроллеров, так и высокопроизводительных микропроцессоров. 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.

1.1 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.

1.2 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 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.

2 Общая программа

Рассмотрим пример небольшой программы для RV32I, которым мы будем пользоваться далее для исследования процесса выполнения команд.

Данная программа выполняет суммирование значений элементов массива слов и увеличивает это значение на 1 - Листинг 2.1.

Листинг 2.1 – Пример программы

```
section text (1)
1
       .globl start; (2)
2
3
       len = 8 #Размер массива (3)
       enroll = 4 #Количество обрабатываемых элементов за одну итерацию
4
5
       elem sz = 4 #Размер одного элемента массива
6
   start: (4)
7
       addi x20, x0, len/enroll (5)
8
       la x1, x(6)
9
       loop:
       lw \times 2, 0(\times 1) (7)
10
       add x31, x31, x2 (8)
11
       lw x2, 4(x1)
12
       add x31, x31, x2
13
       lw x2, 8(x1)
14
       add x31, x31, x2
15
       lw x2, 12(x1)
16
       add x31, x31, x2
17
       addi x1, x1, elem sz*enroll (9)
18
19
       addi x20, x20, -1 (10)
20
       bne x20, x0, loop (11)
       addi x31, x31, 1
21
22 forever: j forever (12)
23
|24| section data (13)
25
   x: 4 byte 0x1 (14)
26
       4 byte 0x2
       .4 byte 0x3
27
28
       .4 byte 0x4
29
       4 byte 0x5
30
       4 byte 0x6
       .4 \, \text{byte} \, 0 \, \text{x}7
31
32
       .4 byte 0x8
```

- 1. Объявление секции .text, содержащей исполняемый код.
- 2. Объявление символа $_start$, имеющего глобальную видимость. Символ $_start$ это специальный символ, обозначающий точку входа в программу.
- 3. Метка.
- 4. Объявление констант.
- 5. Арифметические выражения над константами могут использоваться в командах на месте непосредственного операнда.
- 6. Загрузка в x1 адреса символа x (то есть, начала массива).
- 7. Загрузка в x2 числа по адресу, содержащемуся в x1 по смещению 0.
- 8. Добавление к x31 (который хранит результат) значения x2.
- 9. Смещение указателя x1.
- 10. Уменьшение счетчика цикла.
- 11. Условный переход на метку *loop*.
- 12. Бесконечный цикл.
- 13. Объявление секции данных.
- 14. Начало описания массива.

Дизассемблерный код представлен на листинге 2.2.

Листинг 2.2 – Дизассемблированный код примера программы

```
1 Disassembly of section .text:
3 80000000 < start >:
                                      addi
4 80000000:
                    00200a13
                                               x20, x0,2
5 80000004:
                                               x1,0x0
                    00000097
                                      auipc
6 80000008:
                    03c08093
                                      addi
                                               x1, x1, 60 \# 80000040 <_x>
7
8 8000000c < lp >:
9 8000000c:
                                      lw
                                               x2,0(x1)
                    0000a103
10 80000010:
                    002f8fb3
                                      add
                                               x31,x31,x2
11 80000014:
                    0040a183
                                      lw
                                               x3,4(x1)
12 80000018:
                    003f8fb3
                                      add
                                               x31,x31,x3
13 8000001c:
                                      lw
                                               x4,8(x1)
                    0080a203
14 80000020:
                    00c0a283
                                               x5,12(x1)
                                      lw
                                               x31,x31,x4
15 80000024:
                    004f8fb3
                                      add
16 80000028:
                    005f8fb3
                                      add
                                               x31,x31,x5
17 8000002c:
                    01008093
                                      addi
                                               x1, x1, 16
                    fffa0a13
                                      addi
                                               x20, x20, -1
18 80000030:
                    fc0a1ce3
                                      bne
                                               x20, x0, 8000000c < lp >
19 80000034:
20 80000038:
                    001f8f93
                                      addi
                                               x31,x31,1
21
22 \mid 8000003c < lp2 >:
23 8000003c:
                    0000006 f
                                      jal
                                               x0,8000003c < lp2 >
```

Можно сказать, что данная программа эквивалентна следующему псевдокоду на языке С, представленному на листинге 2.3.

Листинг 2.3 – Псевдокод общей программы

```
1 | #define len 8
2 #define enroll 4
3 #define elem sz 4
4 \mid int _x[] = \{1, 2, 3, 4, 5, 6, 7, 8\};
5 void _start() {
       int x20 = len/enroll;
6
7
       int *x1 = _x;
8
       do {
9
            int x2 = x1[0];
10
11
            x31 += x2;
           x2 = x1[1];
12
            x31 += x2;
13
           x2 = x1[2];
14
            x31 += x2;
15
           x2 = x1[3];
16
            x31 += x2;
17
            x1 += enroll;
18
            x20 --;
19
       } while (x20 != 0);
20
21
       x31++;
       while (1) {}
22
23|}
```

3 Результаты исследования программы

3.1 Задание №1

Условие задания

В процесссе выполнения задания необходимо выполнить следующие действия:

- 1. Ознакомиться с теоретической частью, внимательно изучить примеры.
- 2. Перейти в подкаталог src командой $cd\ riscv-lab/src$.
- 3. Выполнить сборку, запустив команду *gmake*. Убедиться, что был создан файл test.hex, содержащий шестнадцатеричное представление программы, а в окне терминала отобразился дизассемблерный листинг. Сравнить дизассемблерный листинг с тем, который приведен в примере.
- 4. Создать новый файл, содержащий текст программы по индивидуальному варианту (см. Индивидуальные варианты). Поместить его в каталог src. Текст программы сохранить в файле с расширением .s.
- 5. Изучить текст программы по индивидуальному варианту. Поместить в отчете псевдокод, соответствующий данной программе.
- 6. Анализируя исходный текст программы, ответьте на вопрос: какое значение должно содержаться в регистре x31 в конце выполнения программы?
- 7. Изменить в Makefile строку SRC =так, чтобы ее содержимое соответствовало имени файла с текстом программы без расширения .s.
- 8. Выполнить компиляцию командой *gmake*. В процессе будет создан файл с расширением .hex, хранящий содержимое памяти команд и данных, а в окне терминала отобразится дизассемблерный листинг, который необходимо поместить в отчет вместе с исходным текстом.

Результаты выполнения

Листинг 3.1 – Код программы варианта

```
1
       section text
2
       .globl _start;
       len = 8
3
4
       enroll = 4
5
       elem sz = 4
6
7
   start:
8
       addi x20, x0, len/enroll
9
       la x1, _x
       add x31, x0, x0
10
       lp:
11
       lw x2, 0(x1)
12
       add x31, x31, x2 #!
13
       lw x3, 4(x1)
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
20
       addi x1, x1, elem sz*enroll
       addi x20, x20, -1
21
       bne x20, x0, lp
22
       addi x31, x31, 1
24|lp2: j lp2
25
       section data
26
27
   x 4 byte 0x1
28
       .4 byte 0x2
29
       .4 byte 0x3
30
       .4 byte 0x4
       .4 byte 0x5
31
32
       .4 byte 0x6
33
       .4 byte 0x7
34
       .4 byte 0x8
```

Выполнение команды gmake показано на рисунке 3.1

```
@WIN-40TB9R26C78 MINGW32 /c/User/mansurov/riscv-lab/src (main)
$ omake
riscv64-unknown-elf-as --march=rv32i lab_02_09.s -o lab_02_09.o
lab_02_09.s: Assembler messages:
lab_02_09.s: Warning: end of file not at end of a line; newline inserted riscv64-unknown-elf-ld -b elf32-littleriscv -T link.ld lab_02_09.o -o lab_02_09.
riscv64-unknown-elf-objdump -D -M numeric,no-aliases -t lab_02_09.elf
lab_02_09.elf:
                    file format elf32-littleriscv
SYMBOL TABLE:
                 .text 00000000 .text
80000000 1
               d
80000040 1
               d .data
df *ABS*
                          00000000 .data
00000000 lab_02_09.o
00000000
8000000
                  *ABS*
                          00000000 len
00000004
                  *AB5*
                          00000000 enroll
                  *AB5*
                          00000000 elem_sz
00000004
                  .data 00000000 _x
text 00000000 lp
80000040 1
8000000c
8000003c 1
                          00000000 1p2
                  .text
                          00000000 _start
80000000 g
                  .text
80000060 g
                          00000000 _end
                  .data
Disassembly of section .text:
80000000 <_start>:
80000000:
                 00200a13
                                                     x20,x0,2
                 00000097
                                                     x1,0x0
80000004:
                                            auipc
80000008:
                 03c08093
                                                     x1,x1,60 # 80000040 <_x>
                                            addi
8000000c <1p>:
8000000c:
                 0000a103
                                                     x2,0(x1)
80000010:
                 002f8fb3
                                            add
                                                     x31,x31,x2
                 0040a183
                                                     x3,4(x1)
80000014:
                                            1w
80000018:
                 003f8fb3
                                            add
                                                     x31,x31,x3
8000001c:
                 0080a203
                                            1w
                                                     x4,8(x1)
                                                     x5,12(x1)
x31,x31,x4
80000020:
                 00c0a283
                                            1w
80000024:
                 004f8fb3
                                            add
                 005f8fb3
                                                     x31,x31,x5
80000028:
                                            add
8000002c:
                 01008093
                                                     x1,x1,16
                                            addi
80000030:
                 fffa0a13
                                            addi
                                                     x20,x20,-1
80000034:
                  fc0a1ce3
                                            bne
                                                     x20,x0,8000000c <1p>
80000038:
                 001f8f93
                                                     x31,x31,1
                                            addi
8000003c <1p2>:
                 0000006f
                                                     x0,8000003c <1p2>
8000003c:
                                            jal
Disassembly of section .data:
80000040 <_x>:
80000040:
                 0001
                                            c. addi
                                                     x0,0
80000042:
                 0000
                                            unimp
80000044:
                 0002
                                            0x2
80000046:
                 0000
                                            unimp
80000048:
                 00000003
                                                     x0,0(x0) # 0 <elem_sz-0x4>
                                            1b
8000004c:
                 0004
                                            c. addi4spn
                                                              x9,x2,0
8000004e:
                 0000
                                            unimp
80000050:
                 0005
                                            c. addi
                                                     x0,1
80000052:
                 0000
                                            unimp
80000054:
                 0006
                                            0x6
80000056:
                 0000
                                            unimp
```

Рисунок 3.1 – Скриншот запуска gmake для программы по варианту

Листинг 3.2 – Дизассемблированный код

```
Disassembly of section .text:
2
3 80000000 < start >:
4 80000000:
                     00200a13
                                       addi
                                                x20, x0,2
5 80000004:
                                                x1,0x0
                     00000097
                                       auipc
6 80000008:
                                       addi
                                                x1, x1, 60 \# 80000040 < x>
                     03c08093
7
8 8000000c < lp >:
9 8000000c:
                                                x2,0(x1)
                     0000a103
                                       lw
10 80000010:
                     0040a183
                                       add
                                                x31, x31, x2
11 80000014:
                     0080a203
                                       lw
                                                x4,8(x1)
12 80000018:
                     00c0a283
                                       \mathsf{add}
                                                x31,x31,x3
13 8000001c:
                     002f8fb3
                                       lw
                                                x3,4(x1)
14 80000020:
                                                x31,x31,x4
                     003f8fb3
                                       add
15 80000024:
                     004f8fb3
                                                x5,12(x1)
                                       lw
16 80000028:
                                       add
                                                x31,x31,x5
                     005 f8 fb3
17 8000002c:
                     01008093
                                       addi
                                                x1,x1,16
18 80000030:
                     fffa0a13
                                       addi
                                                x20, x20, -1
19 80000034:
                     fc0a1ce3
                                       bne
                                                x20, x0, 8000000c < lp >
20 80000038:
                     001f8f93
                                       addi
                                                x31,x31,1
21
22 \mid 8000003c < lp2 >:
23 8000003c:
                     0000006 f
                                                x0,8000003c < lp2 >
                                       jal
```

Можно сказать, что данная программа эквивалентна следующему псевдокоду на языке С, представленному на листинге 3.3.

Листинг 3.3 – Псевдокод программы

```
1 #define len 8
2 #define enroll 4
3 #define elem sz 4
5 void _start() {
      int x20 = len/enroll;
6
7
      int *x1 = _x;
8
9
      do {
          int x2 = x1[0];
10
11
          x31 += x2;
12
          int x3 = x1[1];
          x31 += x3;
13
          int x4 = x1[2];
14
          x31 += x4;
15
16
          int x5 = x1[3];
17
          x31 += x5;
          x1 += enroll;
18
          x20 --;
19
      } while (x20 != 0);
20
21
      x31++;
22
      while (1) {}
23|}
```

После выполнения программы в x31 будет записано число 38 (подсчитано вручную и написанной программы на языке С по псевдокоду).

3.2 Задание №2

Условие задания

В ходе выполнения данного задания необходимо выполнить следующие действия:

- 1. Открыть проект riscv-lab/taiga/taiga.qpf в среде Intel~Quartus. При запуске Intel~Quartus, если потребуется, выбрать опцию Run~the~Quartus Prime~software.
- 2. Выполнить синтез проекта выбрав пункт меню Processing \to Start \to Start Analysis & Synthesis.
- 3. Запустить симуляцию в среде Modelsim. Для этого выбрать в меню Quartus пункт Tools \to Run Simulation Tool \to RTL Simulation.
- 4. Запустить симуляцию, набрав в командной строке Modelsim команду $run\ 460us.$
- 5. Изучить список сигналов, приведенных в окне Wave.
- 6. Получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки и диспетчеризации команды с адресом 8000002с на первой итерации.

Результаты выполнения

После выполнения пунктов 1-4 запускается симуляция в среде Modelsim, как видно на рисунке 3.2.

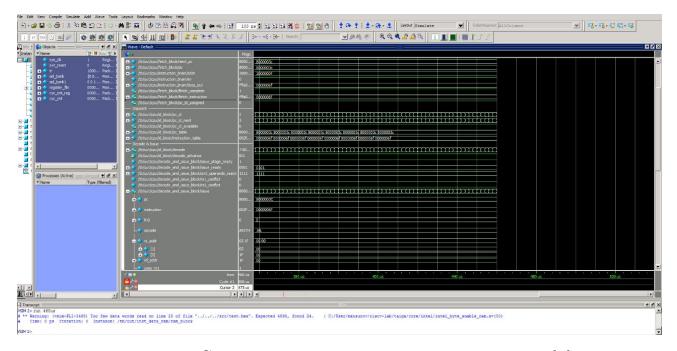


Рисунок 3.2 – Скриншот запуска симуляция в среде Modelsim

На рисунке 3.3 снимок экране симуляции в среде Modelsim на стадии выборки и диспетчеризации команды с адресом 8000002с на первой итерации.

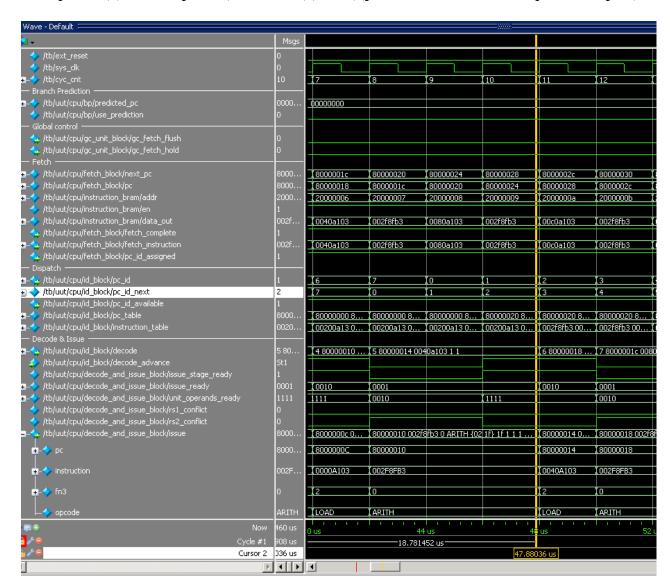


Рисунок 3.3 – Скриншот запуска симуляция в среде Modelsim – команды с адресом 8000002с на первой итерации.

3.3 Задание №3

Условие задания

Получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с адресом 8000000с на второй итерации.

Результаты выполнения

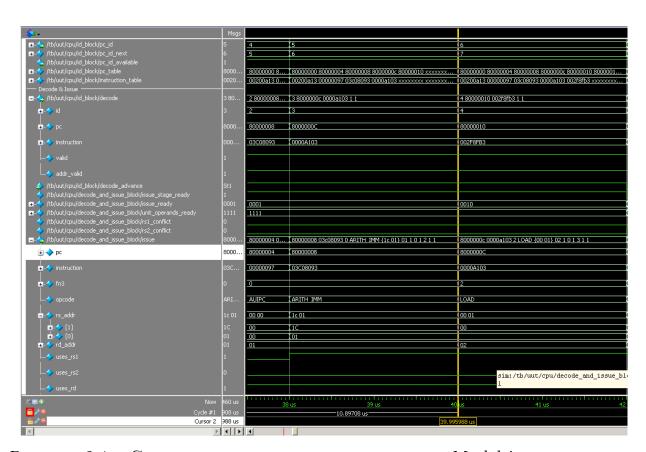


Рисунок 3.4 – Скриншот запуска симуляция в среде Modelsim – команды с адресом 8000000с на второй итерации.

3.4 Задание №4

Условие задания

Получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с адресом 80000020 на первой итерации.

Результаты выполнения

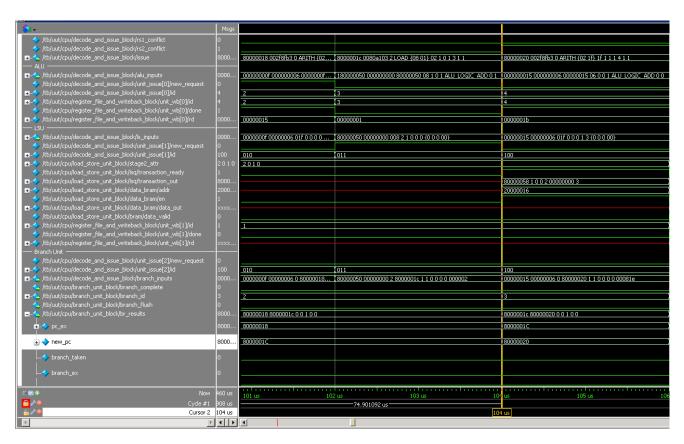


Рисунок 3.5 – Скриншот запуска симуляция в среде Modelsim – команды с адресом 80000020 на первой итерации.

3.5 Задание №5

Условие задания

В процесссе выполнения задания необходимо выполнить следующие действия:

- 1. Исправить 76-ю строку файла $taiga/examples/zedboard/taiga_wrapper.sv$ так, чтобы там был указан путь к файлу .hex, соответствующему программе по индивидуальному варианту. Сохранить файл.
- 2. Перекомпилировать исправленный файл. Для этого в окне программы Modelsim найти вкладку Library, в этой вкладке найти модуль work → taiga wrapper. В контекстном меню модуля выбрать пункт Recompile.
- 3. Ввести в командой строке Modelsim команду restart; run~460us для перезапуска симуляции.
- 4. Получить временную диаграмму сигналов выполнения программы индивидуального варианта.
- 5. Сравнить значение регистра х31 (сигнал /tb/register_file[31]) на момент окончания выполнения программы с тем, который был получен в Задании №1.
- 6. Получить снимок экрана, содержащий временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом #!.
- 7. Анализируя диаграмму заполнить трассу выполнения программы. Рекомендуется использовать для этого файл pipeline.ods, содержащий трассу тестового примера.
- 8. Сделать вывод об эффективности выполнения программы и о путях оптимизации.
- 9. Провести оптимизацию программы путем перестановки команд для устранения конфликтов.

- 10. Перекомпилировать программу и перезапустить симуляцию.
- 11. Заполнить трассу выполнения оптимизированной программы.
- 12. Сравнить трассы выполнения неоптимизированной и оптимизированной версии, сделать выводы.

Результаты выполнения

Трасса работы программы

Трасса работы представлена на рисунке 3.6.

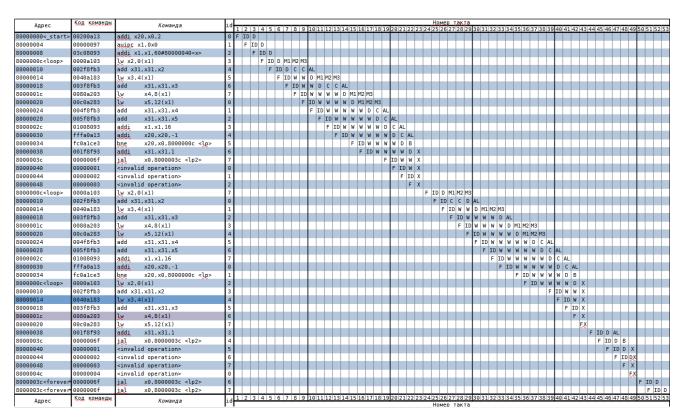


Рисунок 3.6 – Трасса работы программы.

Временные диаграммы

Временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом #! (add x31, x31, x2) с адресом 80000010 представлены на рисунке 3.7-3.9.

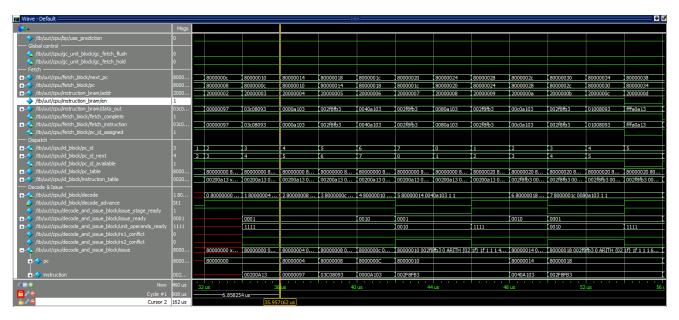


Рисунок 3.7 – Временные диаграммы сигналов выборки и диспетчеризации (Fetch & Dispatch) с адресом 80000010.



Рисунок 3.8 – Временные диаграммы сигналов декодирование и планирование (Decode & Issue) с адресом 80000010.

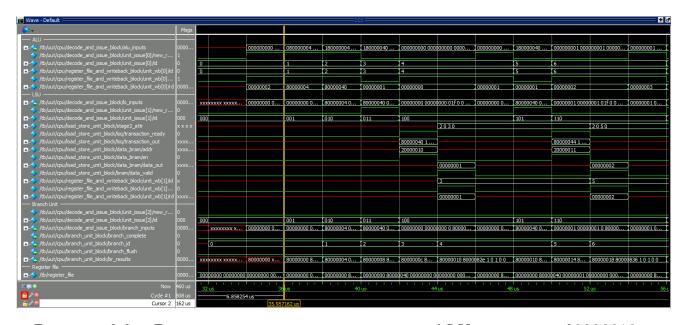


Рисунок 3.9 – Временные диаграммы сигналов ALU с адресом 80000010.

Вывод и предложение по оптимизации

Как видно на трассе работы программы, представленной на рисунке, конфликты возникают из-за того, что данные загружаются в память тогда, когда уже готова выполниться операция сложения тех данных, которые загружаются. Из-за этого и возникают конфликты, так как нечего складывать, так как в памяти пока нет ничего.

Можно заметить, что трижды возникает ситуация ошибочной выборки, которая негативно сказывается на производительности, так как приводит к очистки конвейера.

Оптимизировать программы можно тем, что сначала загрузить все данные в память, а потом их складывать. Тем самым у нас не будет конфликтов, не будет ожидания конца загрузки данных в память.

В итоге, можно будет уменьшить программу на 2 такта 4 раза в программе, на 1 такт 3 раза в программе, то есть на 11/53=20% программа будет работать быстрее.

Оптимизированная программа

Код программы представлен в листинте 3.4

Листинг 3.4 – Код программы (оптимизированный)

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

Дизассемблерный код представлен на листинге 3.5.

Листинг 3.5 – Дизассемблированный код (оптимизированный)

```
1 Disassembly of section text:
3 80000000 < start >:
4 80000000:
                    00200a13
                                       a d d i
                                                x20, x0, 2
5 80000004:
                                                x1,0x0
                    00000097
                                       auipc
6 80000008:
                    03c08093
                                       addi
                                                x1, x1, 60 \# 80000040 <_x>
7
8 8000000c < lp >:
9 8000000c:
                                                x2,0(x1)
                    0000a103
                                       lw
10 80000010:
                                                x3,4(x1)
                    0040a183
                                       lw
11 80000014:
                    0080a203
                                       lw
                                                x4,8(x1)
12 80000018:
                    00c0a283
                                                x5,12(x1)
                                       lw
13 8000001c:
                    002f8fb3
                                                x31,x31,x2
                                       add
14 80000020:
                                                x31,x31,x3
                    003f8fb3
                                       add
                                                x31,x31,x4
15 80000024:
                    004f8fb3
                                       add
16 80000028:
                    005f8fb3
                                       add
                                                x31,x31,x5
17 8000002c:
                    01008093
                                       addi
                                                x1, x1, 16
                                                x20, x20, -1
                                       addi
18 80000030:
                    fffa0a13
                    fc0a1ce3
                                       bne
                                                x20, x0, 8000000c < lp >
19 80000034:
20 80000038:
                    001f8f93
                                       addi
                                                x31,x31,1
21
22 \mid 8000003c < lp2 >:
23 8000003c:
                    0000006 f
                                       jal
                                                x0,8000003c < lp2 >
```

Можно сказать, что данная программа эквивалентна следующему псевдокоду на языке С, представленному на листинге 3.6.

Листинг 3.6 – Псевдокод программы (оптимизированный)

```
1 | #define len 8
2 #define enroll 4
3 #define elem sz 4
4 \mid int _x[] = \{1,2,3,4,5,6,7,8\};
5 void _start() {
       int x20 = len/enroll;
6
7
       int *x1 = _x;
8
       do {
9
10
           int x2 = x1[0];
           int x3 = x1[1];
11
           int x4 = x1[2];
12
           int x5 = x1[3];
13
           x31 += x2;
14
           x31 += x3;
15
16
           x31 += x4;
17
           x31 += x5;
           x1 += enroll;
18
           x20 --;
19
       } while (x20 != 0);
20
21
       x31++;
       while (1) {}
22
23|}
```

Трасса работы оптимизированной программы

Трасса работы представлена на рисунке 3.10.

Адрес	Код команды	Команда	id																Но	мер) Та	жт	a														_	_
	DOM: DOMONIO							6	7	8 9	16	11	12	13 1	41	5 16	17	18	19 2	02	1 22	23	24	25	26 2	7 2	3 29	30	31	32	33 3	4 35	36	37	38	39	40 4	1 4
80000000<_start>		addi x20,x0,2				AL					П																											
80000004	00000097	auipc x1,0x0	1			D D																															Ш	
80000008	03c08093	addi x1,x1,60#80000040 <x></x>	2		F		D				П								-									L										4
8000000c <loop></loop>	0000a103	lw x2,0(x1)	3			F				12 M3																												
80000010	0040a183	lw x3,4(x1)	4 5				F	ID	D M	11 M2	2 M3	3							-									L										4
80000014	0080a203	lw x4,8(x1)	5	П						D MI			П		Т	Т		П	Т								Т	Г		П						П	П	Т
80000018	00c0a283	lw x5,12(x1)	6 7							D D			МЗ															П										
8000001c	002f8fb3	add x31,x31,x2	7	П						F II	D	AL	П		Т	Т		П	Т								Т	Г		П						П		
80000020	003f8fb3	add x31,x31,x3	Θ							F	III	D	AL															L										
80000024	004f8fb3	add x31,x31,x4	1	П	Т	Т	П	П	Т		F	ID	D	AL	Т	Т	П	П	Т		Т	Т			П		Т	Г		П		Т	Т		П	П	П	Т
80000028	005f8fb3	add x31,x31,x5	2									F	ID	D A	AL.																							
8000002c	01008093	addi x1,x1,16	3	П	Т	Т	П	П	Т	\top	Т	Т	F	ID	D A	L	Т	П	Т	Т	Т	Т		П	Т	Т	Т	Г		П	Т	Т	Т	П	П	╛	П	Т
80000030	fffa0a13	addi x20,x20,-1	4								L			F I	D I) AL																						
80000034	fc0alce3	bne x20,x0,8000000c <lp></lp>	4 5					П	Т						F I	D D	В			Т					T													Т
80000038	001f8f93	addi x31,x31,1	6								L					- II	D	Х																				
800003c	0000006f	jal x0,8000003c <lp2></lp2>	6 7								Г						ID		Т								Т	Г										Т
80000040	00000001	<invalid operation=""></invalid>															F																					
80000044	00000002	<invalid operation=""></invalid>	0	П		Т	П		Т		П	т	П	т	т	т	т	FΧ	Т			Т			т	Т	т	П		П	т	т	Т	П	П	╗	П	т
8000000c <loop></loop>	0000a103	lw x2.0(x1)	7								н							~	FΙ	D D	M3	M2	МЗ															
80000010	0040a183	lw x3,4(x1)	Θ								П	т	П		т					- I	D D	М1	M2	мз	7	т	т	П				т	Т			_	П	т
80000014	0080a203	lw x4,8(x1)	1								н									F	II	D	M1	M2	43													
80000018	00c0a283	lw x5.12(x1)	1 2								П				т				т					M1		13	т	П					Т			_	П	т
8000001c	002f8fb3	add x31.x31.x2	3								ш											F	ID	D	AL													
80000020	003f8fb3	add x31.x31.x3	3								т				т				т					ID			т	П								_		т
80000024	004f8fb3	add x31.x31.x4	5								н													F	TD I) AI												
80000028	005f8fb3	add x31.x31.x5	5								т				т				т							D D												т
8000002c	01008093	addi x1.x1.16	7								н															= II												
80000030	fffa0a13	addi x20,x20,-1	0								т				т				т						7	F	II	D	AL							_		т
80000034	fc0alce3	bne x20,x0,8000000c <lp></lp>									н															Ė		ID		В								
8000000c <loop></loop>	0000a103	lw x2.0(x1)	2								т								т						7		T.			D	x		т			_		т
80000010	0040a183	lw x3,4(x1)	3								н																	i i	_	ID	_							
80000014	0080a203	lw x4,8(x1)	3								т								т						7			П			X		П			_		т
80000018	00c0a283	lw x5,12(x1)	5								н																				FX							
80000038	001f8f93	addi x31,x31,1	3								т								т						7			П				I	ם מ	AI				т
8000003c	0000000f	ial x0,8000003c <lp2></lp2>	4																												ď			D	В			
80000030	00000001	<invalid operation=""></invalid>	4 5																													1	1	ID	-	X		
80000040	00000001	<invalid operation=""></invalid>	6								L																						Ė		ID			
80000044	00000002	<invalid operation=""></invalid>	6 7																																	X		7
8000004c	00000003	<invalid operation=""></invalid>	Θ								L																								-	FX		
8000003c	00000004 0000006f	ial x0,8000003c <lp2></lp2>	6																																	\sim	FI	n r
	Код команды	MAC.		1	2 :	1 4	5	6	7 !	8 9	10	111	12	13 1	4 1	5 16	17	18	192	θ2	1 22	23	24	25	26 2	7 2	3 20	30	31	32	33 3	43	36	37	38	_	_	_
Адрес	SSA SSERVAN	Команда	1d	-	- '	, , ,		_			120	-121	-4	20 3	7 2		, 1) T				- 0 /2	. 2	-14	.,50	-		000	.,,	,00	,,,,,	-	00		

Рисунок 3.10 – Трасса работы оптимизированной программы.

Заключение

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

Также были рассмотрены принципы проектирования и верификации сложных цифровых устройств с использованием языка описания аппаратуры SystemVerilog и ПЛИС.

На основе изученных материалов был найден способ оптимизации программы.

Поставленная цель достигнута.

Приложение

																																		_	_	_			_	_			_	_	_	_
Адрес	Код команды	Команда	id 1	2	3 4	5 6	7	8	9 116	111	12 1	3 17	115	16 1	7 15	19	20 2	1 2	2 2 3	24	Ho	мер 26	27 2	KT 8 2	9 3	03	1 32	33	34	35 3	6 37	38	394	40 4	11 4	2 43	44	45	46	47	48	49	Isa	51	52	53
80000000<_start>	00200a13	addi x20,x0,2	0 F	ID	D		Ť	Ť	1									Ť			Ĭ		1		Ť	Ť			Ť		1		Ť	Ť	Ť	Ť	i i	Ť	Ť	Ť	ľ	Ť	ř	Ť		Ť
80000004	00000097	auipc x1,0x0	1	F I	D D		Т		Т	П		Т	П	П	Т	П	Т	Т	Т	Т			П	Т	Т	Т	Т		П	Т	Т	П	Т	Т	Т	Т			П					Г	П	П
80000008	03c08093	addi x1,x1,60#80000040 <x></x>	2		F ID	D																																								
8000000c <loop></loop>	0000a103	lw x2,0(x1)	3		F	ID C	M1	M2 N	13								Т	Т	Т	Т			П	Т	Т	Т	Т	П	П	Т			т	Т	т	т			П	П			Г	Г	П	П
80000010	002f8fb3	add x31,x31,x2	4			FI	D D	C	C AL																																					
80000014	0040a183	lw x3,4(x1)	5			F	ID	W I	W D	M1	M2 P	13						Т	Т	Т			П	Т	Т	Т	Т	П	П	Т			т	Т	т	т			П	П			Г	Г	П	П
80000018	003f8fb3	add x31,x31,x3	6				F	ID !	w W	D	C	C AL																																		
8000001c	0080a203	lw x4,8(x1)	7				Т	F]	D W	W	W	W D	M1	M2 M	13	П		т	Т	Т			Т	т	Т	Т	Т		П	т	Т	П	Т	т	т	т				П			Г	П	П	П
80000020	00c0a283	lw x5,12(x1)	Θ						FIL	W	W	W W	D	M1 M	12 M3																															
80000024	004f8fb3	add x31,x31,x4	1				Т		F	ID	W	w w	W	W E	ОС	AL		т	Т	Т			Т	т	Т	т	т		П	т	Т	П	Т	т	Т	т				П				г		г
80000028	005f8fb3	add x31,x31,x5	2							F	ID	W W	W	W	W D	C	AL																													
8000002c	01008093	addi x1.x1.16	3						т		F I	D W	W	W	w w	D	C A	L.	Т	т			т	т	т	т	т		П	т			т	т	т	т						г	Г	П		г
80000030	fffa0a13	addi x20.x201	4									FI	W	W	N W	W	D (C A	L						ı																					
80000034	fc0alce3	bne x20,x0,8000000c <lp></lp>	5						Т					ID V			W								T	Т	Т							т	Т	т										г
80000038		addi x31,x31,1	6				i.							FI				M E		į.					t		į.																			
899993c	0000006f	ial x0,8000003c <lp2></lp2>	7						т				П		-	F	ID V	N V	ı x	т				т	т	т	т		П		т	П	т	т	т	т							г	т		г
80000040	00000001	<invalid operation=""></invalid>	0														FI								ı																					
80000044	00000002	<invalid operation=""></invalid>	lil.																D X					7	т		т		П				-	т	_	_				г			г	т		г
80000048	000000002	<invalid operation=""></invalid>	2															F	X																											
80000004C 8000000C <loop></loop>	0000a103	lw x2.0(x1)	7															۳.	^		TD	D	M1 N	12 N	13		т		П			П	1	т		-							г			г
80000010	002f8fb3	add x31,x31,x2	Θ																	Ė		ID				1																				
80000014	0040a183	lw x3.4(x1)	1 i																			F					1 M2	мз						т		-							г			г
80000014	003f8fb3	add x31,x31,x3	2																									D	AI					٠												
8000001c	0080a203	lw x4,8(x1)	3																								W			M1 M	2 M3		-	7	-	-							г			г
80000010		lw x5,12(x1)	4																					-		D V					1 M2			÷										-		
80000020	00C00203	add x31,x31,x4	5																	Н				Η,	Ŧ,			W				C	A1	-	-	-							г			г
80000024	004101b3	add x31,x31,x5	6			-			-																ď			W		W					-									-		
80000026 8000002c	01008093	addi x1.x1.16	7						-															-	т	- 1						W		C A		-							Р			
80000030	fffa0a13	addi x20,x20,-1	0			-			٠	н	-		н		-			-		н				-	٠	-	ľ			W					C AL			_		-	-	-		۰		
80000034	fc0alce3		I,						-															-	т			r				W			D B								Р			П
800000054 8000000c <loop></loop>	0000a103	bne x20,x0,8000000c <lp>lw x2,0(x1)</lp>	2			-			-		-	-	-			ш		-		-				-	4	-	-	Н	ш			W			W D					_		_		۰		-
80000010	0000a103	add x31,x31,x2	3						н																н				Н	- 1	110				W W								Р			
80000010	0040a183	lw x3.4(x1)	4			-			-				-					-						-	4	-	-								W W					_		_		۰		-
80000014	0040a183	add x31.x31.x3	5						1																۱											D X										П
						-			-		-	_	_		-	ш		-		-				-	4	-	-		ш	-	-	ш	4	4		- X				ш		-		۰	ш	Н
8000001c	0080a203		6																															4												П
80000020	00c0a283		1/1			-			1									-							1		-				-		4	4	4	EX						Н	Н	-		
80000038 8000003c	001f8f93 0000006f	addi x31,x31,1 ial x0,8000003c <lp2></lp2>	3																						۱									4	-		r		D ID		В					r
80000036	000000061	xu,8000003c <ip2> <invalid operation=""></invalid></ip2>	5			-					-				-			-					-	-	٠	-							4	+	-	-				ID		v	Н	-		
80000040	000000002	<invalid operation=""> <invalid operation=""></invalid></invalid>	6																					7	1									4	4	-					ID					r
	000000002		7			-					-				-			-					-	-	٠	-	-			-			4	-	-	-				F	F			-		
80000048		<invalid operation=""></invalid>	1.																															4							F	X				
8999994c	00000004	<invalid operation=""></invalid>	0						1							ы									ł			1					_	-	-	-						ιX	-			
8000003c <forever< td=""><td></td><td>ial x0,8000003c <lp2></lp2></td><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>ID</td><td></td><td></td></forever<>		ial x0,8000003c <lp2></lp2>	6																						1									4										ID		
8000003c <forever< td=""><td></td><td>jal x0,8000003c <lp2></lp2></td><td>7</td><td>4</td><td>٠.</td><td>Ψ.</td><td></td><td><u>.</u></td><td>١.</td><td>Щ.</td><td></td><td></td><td></td><td></td><td>٠.</td><td>Н</td><td></td><td></td><td></td><td>١.</td><td></td><td>_</td><td></td><td>-</td><td>_</td><td>_</td><td></td><td><u> </u></td><td></td><td></td><td></td><td>ш</td><td>_</td><td></td><td></td><td>٠</td><td>٠.</td><td>щ.</td><td>پ</td><td>پ</td><td></td><td>٠.</td><td></td><td></td><td>ID</td><td></td></forever<>		jal x0,8000003c <lp2></lp2>	7	4	٠.	Ψ.		<u>.</u>	١.	Щ.					٠.	Н				١.		_		-	_	_		<u> </u>				ш	_			٠	٠.	щ.	پ	پ		٠.			ID	
Адрес	Код команды	Команда	id 1	2	3 4	5 6	17	8	9 16	11	12 1	3 14	4 15	16 1	/(18	19	26 2	1 2	2 23	5 24		26 мер				B 3	1 32	133	34	35 3	6 37	38	39 4	10 4	11 42	2 43	44	45	46	47	48	49	150	51	52	53

Рисунок 3.11 – Трасса работы программы.

Адрес	Код команды	Команда	100																Ном	ep	так	та			_					_						_	_	_
			10	1	2 3	4	5	6	7 8	3 9	10	11	12 1	3 14	15	16	17 1	8 19	20	21	22 2	3 2	4 2	26	27	28	29	9 3	1 32	33	34	35 3	6 3	7 38	39	40	41	47
80000000<_start>		addi x20,x0,2		F I																																		
80000004	00000097	auipc x1,0x0	1		F I							Ш			Ш				Ш								_									Ш		L
80000008	03c08093	addi x1,x1,60#80000040 <x></x>	2		F		D																															
8000000c <loop></loop>	0000a103	lw x2,0(x1)	3	Ш		F	ID																													Ш		L
80000010	0040a183	lw x3,4(x1)	4							11 M2																	- 1											
80000014	0080a203	lw x4,8(x1)	5							M1																												L
80000018	00c0a283	اپ x5,12(x1)	6	ш				- 1		D D			М3														-1											
8000001c	002f8fb3	add x31,x31,x2	7						F	ID																												
80000020	003f8fb3	add x31,x31,x3	Θ	ш						F		D															-1											
80000024	004f8fb3	add x31,x31,x4	1								F		D A		П									П			П		Т									
80000028	005f8fb3	add x31,x31,x5	2									F	ID D	AL																								
8000002c	01008093	addi x1,x1,16	3	П	Т	Т	П	Т	Т	Т	Г	П	FI				Т				Т	Т	Т	Т		П	Т	Т	Т			Т	Т	Т		П		
80000030	fffa0a13	addi x20,x20,-1	4										F	ID	D	AL																						
80000034	fc0alce3	bne x20,x0,8000000c <lp></lp>	5	П	Т	Т	П	т	Т	т	Г	П		F	ID	D	В	Т	П		т	Т	т	Т		П	П	т	Т		╗	т	т	Т	П	П		г
80000038	001f8f93	addi x31,x31,1	6												F	ID	D 3	(
8000003c	0000006f	ial x0,8000003c <lp2></lp2>	7	П		т	П	т		т	П	П		Т	П	F :	ID D	Х	П		т		Т	т			т	т	т			т	т	т				г
80000040	00000001	<invalid operation=""></invalid>	Θ														F 2	Č																				
80000044	00000002	<invalid operation=""></invalid>	1	П		т		т			П			Т	П		F	X	П		т			т		7	т		т				т	т				П
8000000c <loop></loop>	0000a103	lw x2.0(x1)	7														ľ	F	ID	D	M1 N	12 M	3															
80000010	0040a183	lw x3,4(x1)	0	П		т	П	т			П	П			П				F	ID	D N	11 M	2 M	al T			т		т				т	т				г
80000014	0080a203	lw x4,8(x1)	1																	F	ID	D M	п ма	мз														
80000018	00c0a283	lw x5.12(x1)	2								П										F I	D I	M	M2	МЗ		т		т									П
8000001c	002f8fb3	add x31.x31.x2	3																				D D															
80000020	003f8fb3	add x31,x31,x3	4			т		7			П								П				11				7		т				т	т				П
80000024	004f8fb3	add x31,x31,x4	5																				E	TD	D	AI												
80000028	005f8fb3	add x31,x31,x5	6								П												1		ID													П
8000002c	01008093	addi x1.x1.16	7																					Ė		ID		1										
80000030	fffa0a13	addi x20,x20,-1	9	П							П								П									D A										П
80000034	fc0alce3	bne x20,x0,8000000c <lp></lp>	1																									D										
8000000c <loop></loop>	0000a103	lw x2.0(x1)	2								П								П										D D	Y		т						П
80000000000000	0040a183	lw x3,4(x1)	3																										ID									
80000014	0080a203	lw x4.8(x1)	4								Н																7	Τ.		X				т		Н		г
80000014	00c0a203	lw x5,12(x1)	5	н																										FX								
80000018	00C00203	addi x31,x31,1	3	П							Н								П								7				E .	ID I	Δ ΔΙ			П		г
80000036	00110193	jal x0,8000003c <lp2></lp2>	4																													FI						i
80000030	00000001	<invalid operation=""></invalid>	5																								1						- II		Y			r
80000040	00000001	<invalid operation=""></invalid>	6																															ID				i
80000044	00000002	<invalid operation=""></invalid>	6																								1						-		X			r
8000004c	00000003	<invalid operation=""></invalid>	θ																								1								FX			i
8000003c	00000004 0000006f	ial x0.8000003c <lp2></lp2>	6																								1								100	-	TD	h
	Код команды	ANA.	10	+	2 2	-	-	6	7 0	0	10	11	12 1	3 1.4	15	16	17 1	8 14	120	21	22 2	3 2	4 21	26	27	28	0	0 2	1 32	33	34	35 2	6 2	7 30	30			
Адрес	DOM DOMORRAN	Команда	id	-	2 3	- 4	١,	0	,	9	110	111	12 1	J 14	123	10	1/ 1				Tak		4 2:) Z 0	21	20	3 3	00 3	1132	J	J4	د ادد	0 3	/ 30	ودرر	40	14.1	-42

Рисунок 3.12 – Трасса работы оптимизированной программы.