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 «Компьютерные системы и сети (ИУ6)»

НАПРАВЛЕНИЕ ПОДГОТОВКИ «09.03.04 Программная инженерия»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
по курсу «Архитектура ЭВМ»

«Изучение принципов работы микропроцессорного ядра RISC-V»

Студент: ИУ7-53Б _____ М. Д. Маслова
(группа) (подпись, дата) (И. О. Фамилия)

Преподаватель: _____ Е. Н. Дубровин
(подпись, дата) (И. О. Фамилия)

Москва, 2021

Содержание

1 Задание 1 **5**

2 Задание 2 **8**

3 Задание 3 **9**

4 Задание 5 **11**

4.1 Проверка результата 11

4.2 Временные диаграммы сигналов стадий выполнения 11

4.3 Трасса выполнения программы 12

4.4 Оптимизация программы 13

4.5 Вывод 16

opn

jkdf

1 Задание 1

На листинге 1.1 приведен текст программы по индивидуальному варианту.

Листинг 1.1 – Текст программы по индивидуальному варианту

```
# ВАРИАНТ 12
        .section .text
        .globl _start;
        len = 8 #Размер массива
        enroll = 4 #Количество обрабатываемых элементов за одну итерацию
        elem_sz = 4 #Размер одного элемента массива

_start:
        la x1, _x
        addi x20, x1, elem_sz*len #Адрес последнего элемента
lp:
        lw x2, 0(x1)
        lw x3, 4(x1)
        add x31, x31, x2 #!
        add x31, x31, x3
        lw x4, 8(x1)
        lw x5, 12(x1)
        add x31, x31, x4
        add x31, x31, x5
        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
```

На листиге 1.2 приведен псевдокод на языке C, соответствующий программе варианта.

Листинг 1.2 – Псевдокод на языке C программы варианта

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

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

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

    do
    {
        int x2 = x1[0];
        int x3 = x1[1];
        x31 += x2;
        x31 += x3;
        int x4 = x1[2];
        int x5 = x1[3];
        x31 += x4;
        x31 += x5;
        x1 += enroll;
    } while(x1 != x20);

    x31++;

    while(1){}
}
```

Проанализировав исходный текст программы можно сделать вывод, что в регистре x31 в конце выполнения программы должна содержаться сумма элементов массива + 1 (37).

На листиге 1.3 приведен дизассемблерный код.

Листинг 1.3 – Дизассемблерный листинг

```
80000000 <_start>:
80000000:       00000097                auipc   x1,0x0
80000004:       03c08093                addi    x1,x1,60 # 8000003c <_x>
80000008:       02008a13                addi    x20,x1,32

8000000c <lp>:
8000000c:       0000a103                lw      x2,0(x1)
80000010:       0040a183                lw      x3,4(x1)
80000014:       002f8fb3                add     x31,x31,x2
80000018:       003f8fb3                add     x31,x31,x3
8000001c:       0080a203                lw      x4,8(x1)
80000020:       00c0a283                lw      x5,12(x1)
80000024:       004f8fb3                add     x31,x31,x4
80000028:       005f8fb3                add     x31,x31,x5
8000002c:       01008093                addi    x1,x1,16
80000030:       fd409ee3                bne     x1,x20,8000000c <lp>
80000034:       001f8f93                addi    x31,x31,1

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

2 Задание 2

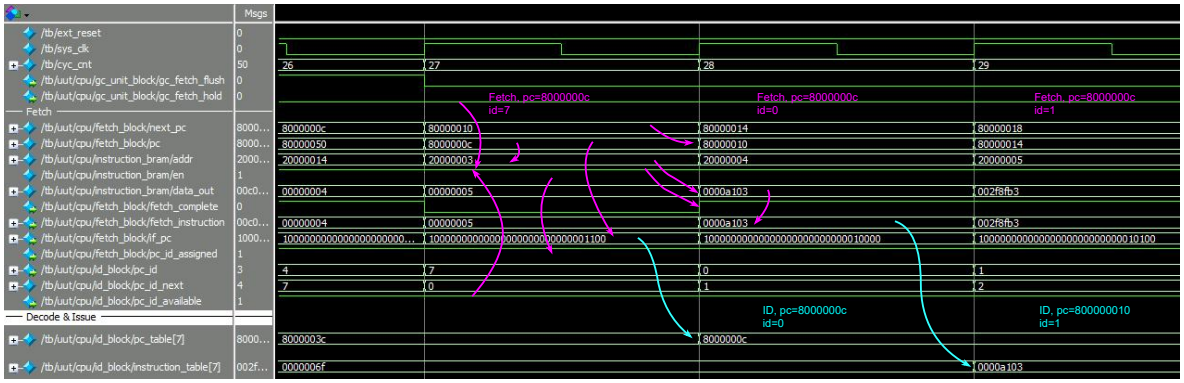


Рисунок 2.1 – Выборка и диспетчеризация команды с адресом `8000000c` на второй итерации

3 Задание 3

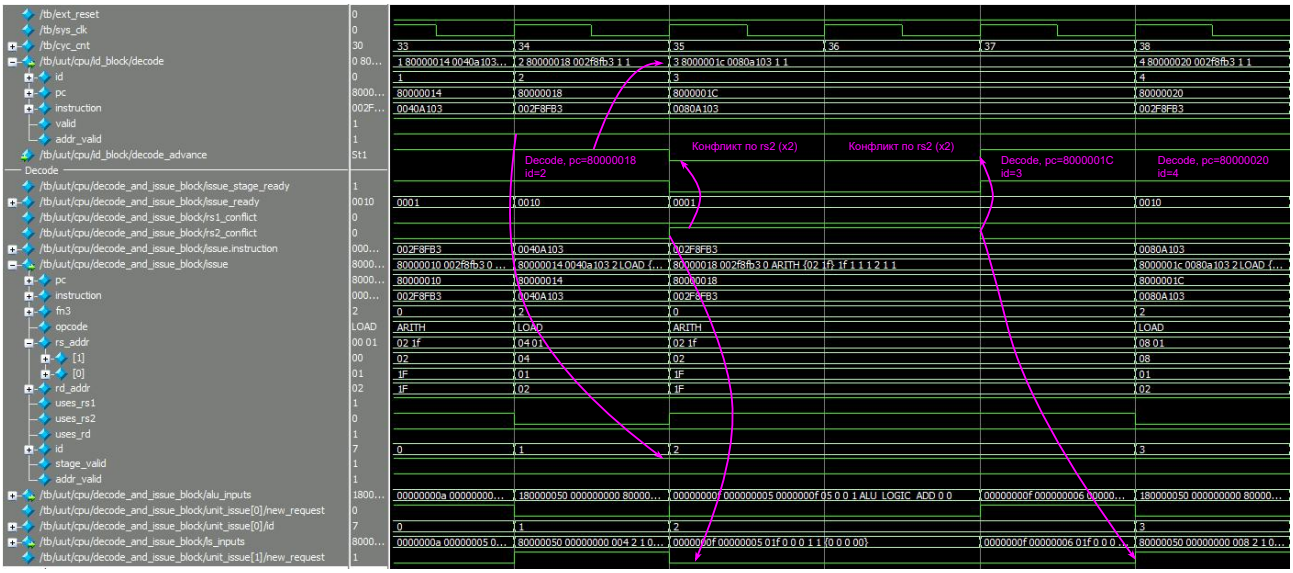


Рисунок 3.1 – Декодирование и планирование на выполнение команды с адресом `80000018` на второй итерации

Задание 4

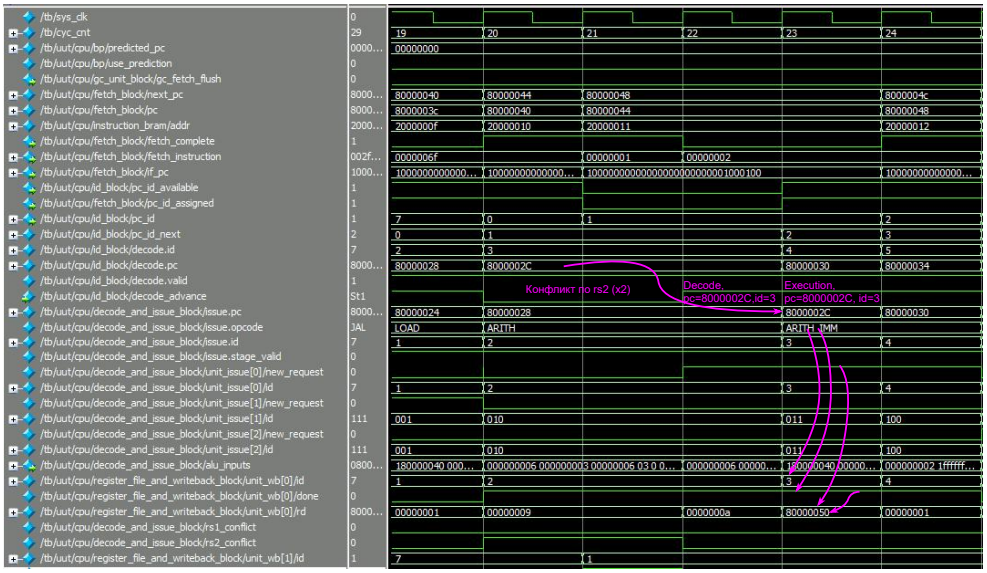


Рисунок 3.2 – Выполнение команды с адресом `8000002c` на первой итерации

4 Задание 5

4.1 Проверка результата

Значение регистра x31 представлено на рисунке 4.1. Значение равно 25 в шестнадцатиричной системе, что соответствует значению 37 в десятиричной системе счисления, приведенном в задании 1.

Рисунок 4.1 – Проверка значения регистра `x31`

4.2 Временные диаграммы сигналов стадий выполнения

В данном подразделе представлены временные диаграммы сигналов стадий выполнения сигналов команды `add x31, x31, x2`, находящейся по адресу `0x80000014` (на первой итерации).

На рисунке 4.2 представлены временные диаграммы сигналов стадий выборки и диспетчеризации. На рисунке 4.3 представлены временные диаграммы сигналов стадий декодирования и планирования на выполнение. На рисунке 4.4 представлены временные диаграммы сигналов стадии выполнения.

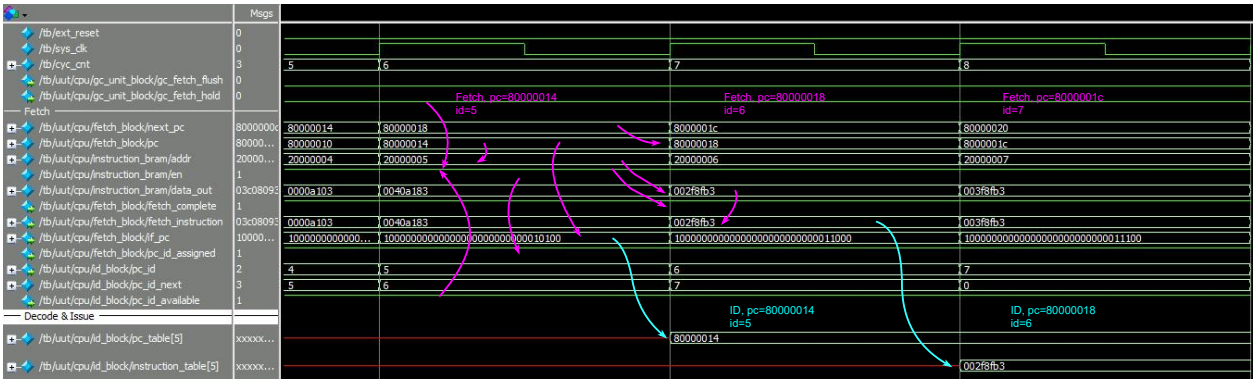


Рисунок 4.2 – Выборка и диспетчеризация команды #!

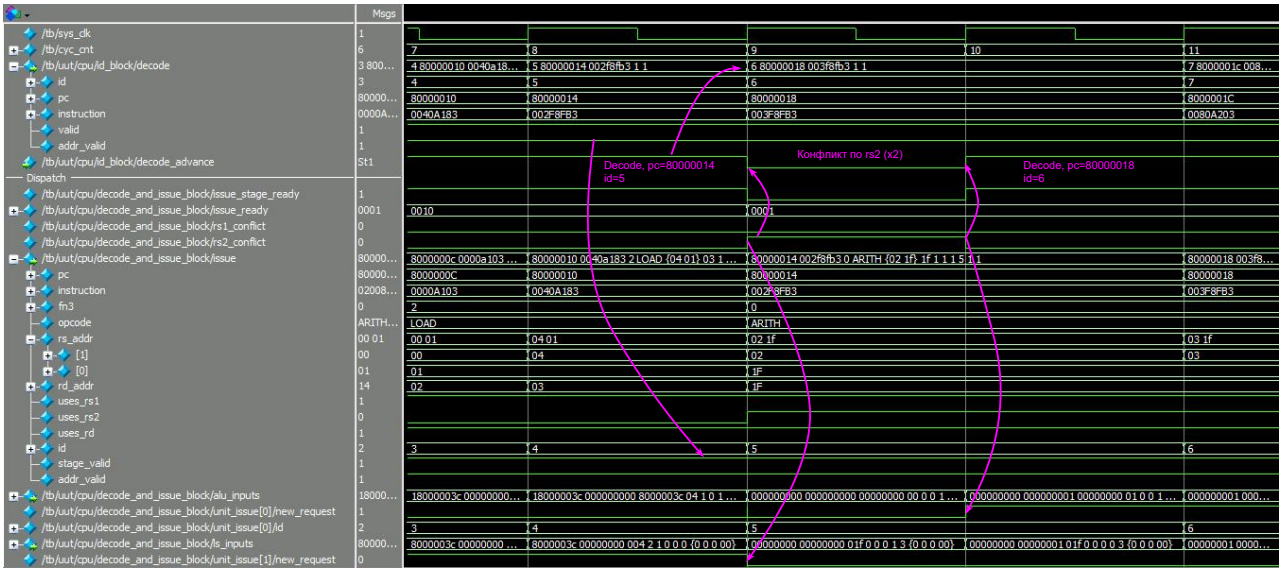


Рисунок 4.3 – Декодирование и планирование на выполнение команды #!

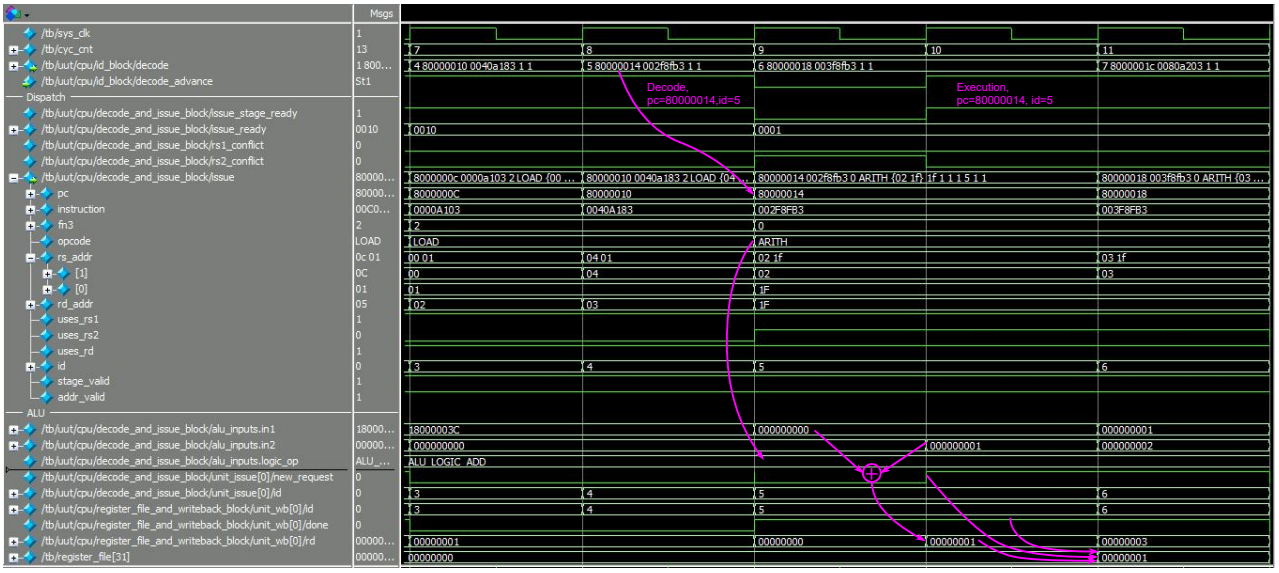
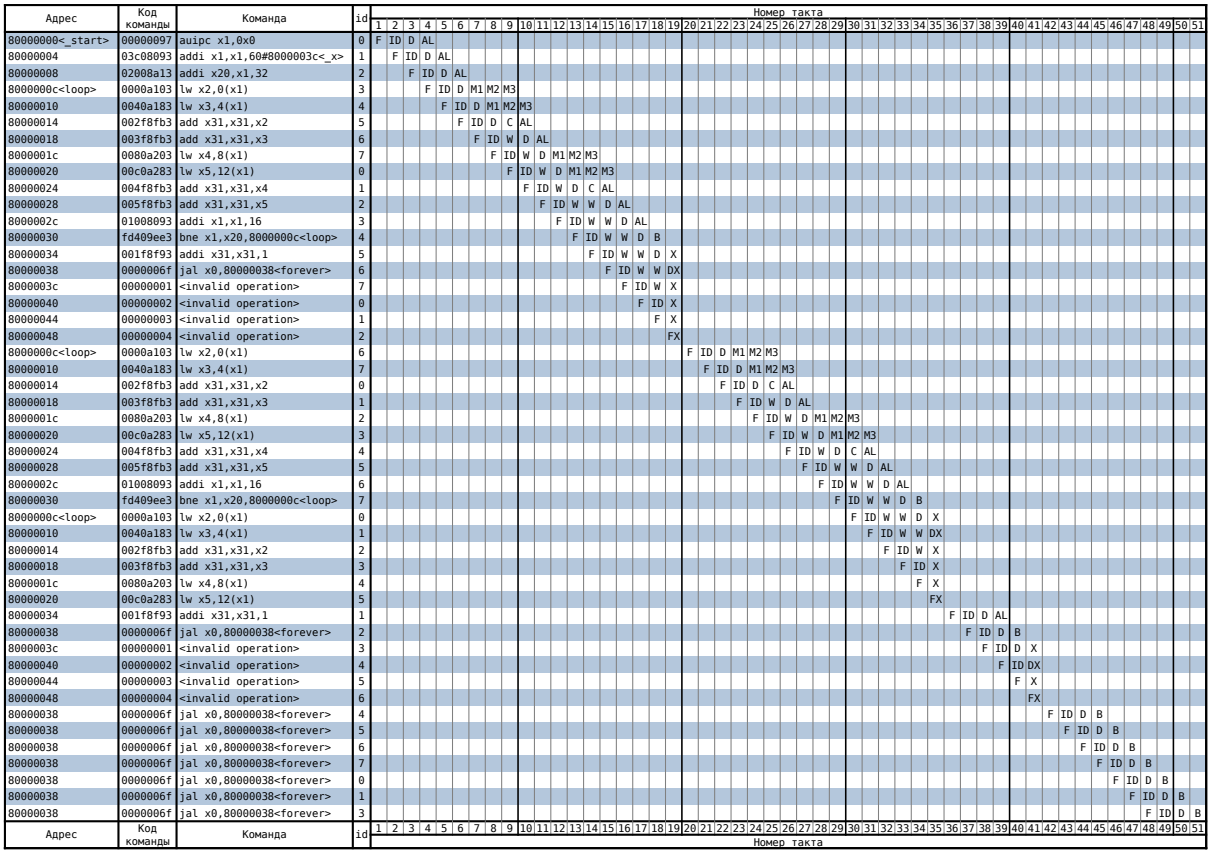


Рисунок 4.4 – Выполнение команды #!

4.3 Трасса выполнения программы

На рисунке 4.5 представлена трасса выполнения программы.

Проанализировав трассу выполнения программы, можно сделать вывод о том, что конфликты возникают тогда, когда блок обращения к памяти не успевает загрузить необходимое для выполнения арифметической операции значение. Это происходит потому, что блок обращения к памяти выполняет загрузку за три такта, а арифметическая команда следует за загрузкой нужной переменной через одну команду, таким образом блок обращения к памяти завершает только 2 такта обработки.



Адрес	Код команды	Команда	id	Номер такта
				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
80000000<_start>	00000097	auipc x1,0x0	0	F ID D AL
80000004	03c08093	addi x1,x1,60#8000003c<_x>	1	F ID D AL
80000008	02008a13	addi x20,x1,32	2	F ID D AL
8000000c<loop>	0000a103	lw x2,0(x1)	3	F ID D M1 M2 M3
80000010	0040a183	lw x3,4(x1)	4	F ID D M1 M2 M3
80000014	002f8fb3	add x31,x31,x2	5	F ID D C AL
80000018	003f8fb3	add x31,x31,x3	6	F ID W D AL
8000001c	0080a203	lw x4,8(x1)	7	F ID W D M1 M2 M3
80000020	00c0a283	lw x5,12(x1)	0	F ID W D M1 M2 M3
80000024	004f8fb3	add x31,x31,x4	1	F ID W D C AL
80000028	005f8fb3	add x31,x31,x5	2	F ID W W D AL
8000002c	01008093	addi x1,x1,16	3	F ID W W D AL
80000030	fd409ee3	bne x1,x20,8000000c<loop>	4	F ID W W D B
80000034	001f8f93	addi x31,x31,1	5	F ID W W D X
80000038	0000006f	jal x0,80000038<forever>	6	F ID W W DX
8000003c	00000001	<invalid operation>	7	F ID W X
80000040	00000002	<invalid operation>	0	F ID X
80000044	00000003	<invalid operation>	1	F X
80000048	00000004	<invalid operation>	2	FX
8000000c<loop>	0000a103	lw x2,0(x1)	6	F ID D M1 M2 M3
80000010	0040a183	lw x3,4(x1)	7	F ID D M1 M2 M3
80000014	002f8fb3	add x31,x31,x2	0	F ID D C AL
80000018	003f8fb3	add x31,x31,x3	1	F ID W D AL
8000001c	0080a203	lw x4,8(x1)	2	F ID W D M1 M2 M3
80000020	00c0a283	lw x5,12(x1)	3	F ID W D M1 M2 M3
80000024	004f8fb3	add x31,x31,x4	4	F ID W D C AL
80000028	005f8fb3	add x31,x31,x5	5	F ID W W D AL
8000002c	01008093	addi x1,x1,16	6	F ID W W D AL
80000030	fd409ee3	bne x1,x20,8000000c<loop>	7	F ID W W D B
8000000c<loop>	0000a103	lw x2,0(x1)	0	F ID W W D X
80000010	0040a183	lw x3,4(x1)	1	F ID W W DX
80000014	002f8fb3	add x31,x31,x2	2	F ID W X
80000018	003f8fb3	add x31,x31,x3	3	F ID X
8000001c	0080a203	lw x4,8(x1)	4	F X
80000020	00c0a283	lw x5,12(x1)	5	FX
80000034	001f8f93	addi x31,x31,1	1	F ID D AL
80000038	0000006f	jal x0,80000038<forever>	2	F ID D B
8000003c	00000001	<invalid operation>	3	F ID D X
80000040	00000002	<invalid operation>	4	F ID DX
80000044	00000003	<invalid operation>	5	F X
80000048	00000004	<invalid operation>	6	FX
80000038	0000006f	jal x0,80000038<forever>	4	F ID D B
80000038	0000006f	jal x0,80000038<forever>	5	F ID D B
80000038	0000006f	jal x0,80000038<forever>	6	F ID D B
80000038	0000006f	jal x0,80000038<forever>	7	F ID D B
80000038	0000006f	jal x0,80000038<forever>	0	F ID D B
80000038	0000006f	jal x0,80000038<forever>	1	F ID D B
80000038	0000006f	jal x0,80000038<forever>	3	F ID D B
Адрес	Код команды	Команда	id	Номер такта
				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

Рисунок 4.5 – Трасса выполнения программы

4.4 Оптимизация программы

Хотя блок обращения к памяти выполняет команду за 3 такта, он может принимать команду на вход каждый такт, таким образом, при последовательном расположении команд загрузки мы сможем увеличить количество команд между загрузкой и арифметической операцией, исключив возникающие конфликты и недопустив новые.

На листинге 4.1 приведен текст оптимизированной программы по индивидуальному варианту.

Листинг 4.1 – Текст оптимизированной программы по индивидуальному варианту

```
# ВАРИАНТ 12 (оптимизация)
        .section .text
        .globl _start;
        len = 8 #Размер массива
        enroll = 4 #Количество обрабатываемых элементов за одну итерацию
        elem_sz = 4 #Размер одного элемента массива

_start:
        la x1, _x
        addi x20, x1, elem_sz*len #Адрес последнего элемента
lp:
        lw x2, 0(x1)
        lw x3, 4(x1)
        lw x4, 8(x1)
        lw x5, 12(x1)
        add x31, x31, x2 #!
        add x31, x31, x3
        add x31, x31, x4
        add x31, x31, x5
        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
```

На листиге 4.2 приведен псевдокод на языке C, соответствующий оптимизированной программе варианта.

Листинг 4.2 – Псевдокод на языке C оптимизированной программы варианта

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

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

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

    do
    {
        int x2 = x1[0];
        int x3 = x1[1];
        int x4 = x1[2];
        int x5 = x1[3];
        x31 += x2;
        x31 += x3;
        x31 += x4;
        x31 += x5;
        x1 += enroll;
    } while(x1 != x20);

    x31++;

    while(1){}
}
```

На листиге 4.3 приведен дизассемблерный код оптимизированной программы.

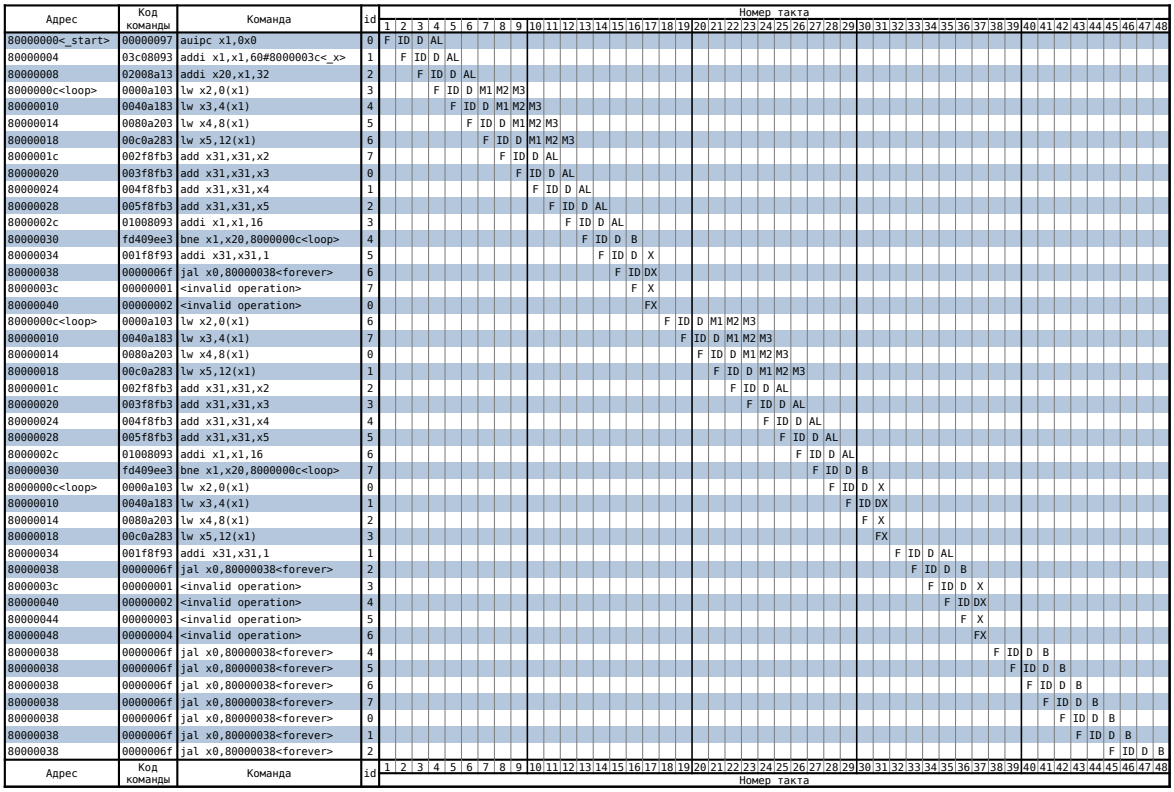
Листинг 4.3 – Дизассемблерный листинг

```
80000000 <_start>:
80000000:       00000097                auipc   x1,0x0
80000004:       03c08093                addi    x1,x1,60 # 8000003c <_x>
80000008:       02008a13                addi    x20,x1,32

8000000c <lp>:
8000000c:       0000a103                lw      x2,0(x1)
80000010:       0040a183                lw      x3,4(x1)
80000014:       0080a203                lw      x4,8(x1)
80000018:       00c0a283                lw      x5,12(x1)
8000001c:       002f8fb3                add     x31,x31,x2
80000020:       003f8fb3                add     x31,x31,x3
80000024:       004f8fb3                add     x31,x31,x4
80000028:       005f8fb3                add     x31,x31,x5
8000002c:       01008093                addi    x1,x1,16
80000030:       fd409ee3                bne     x1,x20,8000000c <lp>
80000034:       001f8f93                addi    x31,x31,1

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

На рисунке 4.6 представлена трасса выполнения оптимизированной программы.

4.5 Вывод

При выполнении неоптимизированной программы вычисления завершаются за 41 такт, при выполнении оптимизированной – за 37 (без учета бесконечного цикла). Таким образом, в результате оптимизации удалось сократить выполнение на 4 такта, то есть получилось ускорить программу на $\frac{4}{41} \approx 0.1(10\%)$



Адрес	Код команды	Команда	id	Номер такта
80000000<_start>	00000097	auipc x1,0x0	0	F ID D AL
80000004	03c08093	addi x1,x1,60#8000003c<_x>	1	F ID D AL
80000008	02008a13	addi x20,x1,32	2	F ID D AL
8000000c<loop>	0000a103	lw x2,0(x1)	3	F ID D M1 M2 M3
80000010	0040a183	lw x3,4(x1)	4	F ID D M1 M2 M3
80000014	0080a203	lw x4,8(x1)	5	F ID D M1 M2 M3
80000018	00c0a283	lw x5,12(x1)	6	F ID D M1 M2 M3
8000001c	002f8fb3	add x31,x31,x2	7	F ID D AL
80000020	003f8fb3	add x31,x31,x3	0	F ID D AL
80000024	004f8fb3	add x31,x31,x4	1	F ID D AL
80000028	005f8fb3	add x31,x31,x5	2	F ID D AL
8000002c	01008093	addi x1,x1,16	3	F ID D AL
80000030	fd409ee3	bne x1,x20,8000000c<loop>	4	F ID D B
80000034	001f8f93	addi x31,x31,1	5	F ID D X
80000038	0000006f	jal x0,80000038<forever>	6	F ID DX
8000003c	00000001	<invalid operation>	7	F X
80000040	00000002	<invalid operation>	0	FX
8000000c<loop>	0000a103	lw x2,0(x1)	6	F ID D M1 M2 M3
80000010	0040a183	lw x3,4(x1)	7	F ID D M1 M2 M3
80000014	0080a203	lw x4,8(x1)	0	F ID D M1 M2 M3
80000018	00c0a283	lw x5,12(x1)	1	F ID D M1 M2 M3
8000001c	002f8fb3	add x31,x31,x2	2	F ID D AL
80000020	003f8fb3	add x31,x31,x3	3	F ID D AL
80000024	004f8fb3	add x31,x31,x4	4	F ID D AL
80000028	005f8fb3	add x31,x31,x5	5	F ID D AL
8000002c	01008093	addi x1,x1,16	6	F ID D AL
80000030	fd409ee3	bne x1,x20,8000000c<loop>	7	F ID D B
8000000c<loop>	0000a103	lw x2,0(x1)	0	F ID D X
80000010	0040a183	lw x3,4(x1)	1	F ID DX
80000014	0080a203	lw x4,8(x1)	2	F X
80000018	00c0a283	lw x5,12(x1)	3	FX
80000034	001f8f93	addi x31,x31,1	1	F ID D AL
80000038	0000006f	jal x0,80000038<forever>	2	F ID D B
8000003c	00000001	<invalid operation>	3	F ID D X
80000040	00000002	<invalid operation>	4	F ID DX
80000044	00000003	<invalid operation>	5	F X
80000048	00000004	<invalid operation>	6	FX
80000038	0000006f	jal x0,80000038<forever>	4	F ID D B
80000038	0000006f	jal x0,80000038<forever>	5	F ID D B
80000038	0000006f	jal x0,80000038<forever>	6	F ID D B
80000038	0000006f	jal x0,80000038<forever>	7	F ID D B
80000038	0000006f	jal x0,80000038<forever>	0	F ID D B
80000038	0000006f	jal x0,80000038<forever>	1	F ID D B
80000038	0000006f	jal x0,80000038<forever>	2	F ID D B

Рисунок 4.6 – Трасса выполнения оптимизированной программы