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»

Студент Рунов К.А.

Группа ИУ7-54Б

Вариант 18

Преподаватели Попов А.Ю., Ибрагимов С.В.

2024 г.

СОДЕРЖАНИЕ

1 Цели

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.

2 Задания

Далее представлен ход выполнения заданий лабораторной работы.

2.1 Задание 1

Далее представлены листинги, полученные в процессе выполнения первого задания лабораторной работы.

```
1  .section .text
2  .globl _start;
3          len = 9
4          enroll = 2
5          elem_sz = 4
6
7  _start:
8          la x1, _x
9          addi x20, x1, elem_sz*len
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 0x8
30         .4byte 0x6
31         .4byte 0x7
32         .4byte 0x5
33         .4byte 0x4
```

Листинг 1 – Код программы по варианту на языке ассемблера

```
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>

Disassembly of section .data:
80000038 <_x>:
80000038:       0001        c.addi  x0,0
8000003a:       0000        unimp
8000003c:       0002        0x2
8000003e:       0000        unimp
80000040:       00000003    lb      x0,0(x0) # 0 <enroll-0x2>
80000044:       0004        c.addi4spn      x9,x2,0
80000046:       0000        unimp
80000048:       0008        c.addi4spn      x10,x2,0
8000004a:       0000        unimp
8000004c:       0006        0x6
8000004e:       0000        unimp
80000050:       00000007    0x7
80000054:       0005        c.addi  x0,1
80000056:       0000        unimp
80000058:       0004        c.addi4spn      x9,x2,0
```

Листинг 2 – Дизассемблированный листинг кода программы по варианту

```
00000097
03808093
02408a13
0000af83
00408093
0000a103
0040a183
01f16463
00200fb3
01f1e463
00300fb3
00808093
ff4092e3
0000006f
00000001
00000002
00000003
00000004
00000008
00000006
00000007
00000005
00000004
```

Листинг 3 – Код программы по варианту в шестнадцатеричном представлении

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

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

void _start() {
    unsigned *x1 = _x; // la x1, _x
    unsigned *x20 = x1 + len; // addi x20, x1, elem_sz*len
    unsigned x31 = x1[0]; // lw x31, 0(x1)
    x1 += 1; // addi x1, x1, elem_sz*1

    do {
        // lp
        unsigned x2 = x1[0]; // lw x2, 0(x1)
        unsigned x3 = x1[1]; // lw x3, 4(x1)

        if (x2 < x31) { // bltu x2, x31, lt1
        } else {
            x31 = x2; // add x31, x0, x2
        }
        // lt1
        if (x3 < x31) { // bltu x3, x31, lt2
        } else {
            x31 = x3; // add x31, x0, x3
        }
        // lt2
        x1 += enroll; // add x1, x1, elem_sz*enroll
    } while (x1 != x20); // bne x1, x20, lp

    for (;;); // lp2: j lp2
}
```

Листинг 4 – Псевдокод программы по варианту

В результате выполнения программы, в регистре x31 будет содержаться число, соответствующее максимальному элементу массива x_, то есть 8.

2.2 Задание 2

```
1  Disassembly of section .text:
2  80000000 <_start>:
3  80000000:        00200a13    addi    x20,x0,2
4  80000004:        00000097    auipc   x1,0x0
5  80000008:        03c08093    addi    x1,x1,60 # 80000040 <_x>
6  8000000c <loop>:
7  8000000c:        0000a103    lw      x2,0(x1)
8  80000010:        002f8fb3    add     x31,x31,x2
9  80000014:        0040a103    lw      x2,4(x1)
10 80000018:        002f8fb3    add     x31,x31,x2
11 8000001c:        0080a103    lw      x2,8(x1)
12 80000020:        002f8fb3    add     x31,x31,x2
13 80000024:        00c0a103    lw      x2,12(x1)
14 80000028:        002f8fb3    add     x31,x31,x2
15 8000002c:        01008093    addi    x1,x1,16
16 80000030:        fffa0a13    addi    x20,x20,-1
17 80000034:        fc0a1ce3    bne     x20,x0,8000000c <loop>
18 80000038:        001f8f93    addi    x31,x31,1
19 8000003c <forever>:
20 8000003c:        0000006f    jal     x0,8000003c <forever>
21
22 Disassembly of section .data:
23 80000040 <_x>:
24 80000040:        0001        c.addi  x0,0
25 80000042:        0000        unimp
26 80000044:        0002        0x2
27 80000046:        0000        unimp
28 80000048:        00000003    lb      x0,0(x0) # 0 <elem_sz-0x4>
29 8000004c:        0004        c.addi4spn      x9,x2,0
30 8000004e:        0000        unimp
31 80000050:        0005        c.addi  x0,1
32 80000052:        0000        unimp
33 80000054:        0006        0x6
34 80000056:        0000        unimp
35 80000058:        00000007    0x7
36 8000005c:        0008        c.addi4spn      x10,x2,0
```

Листинг 5 – Дизассемблированный листинг кода тестовой программы test.s

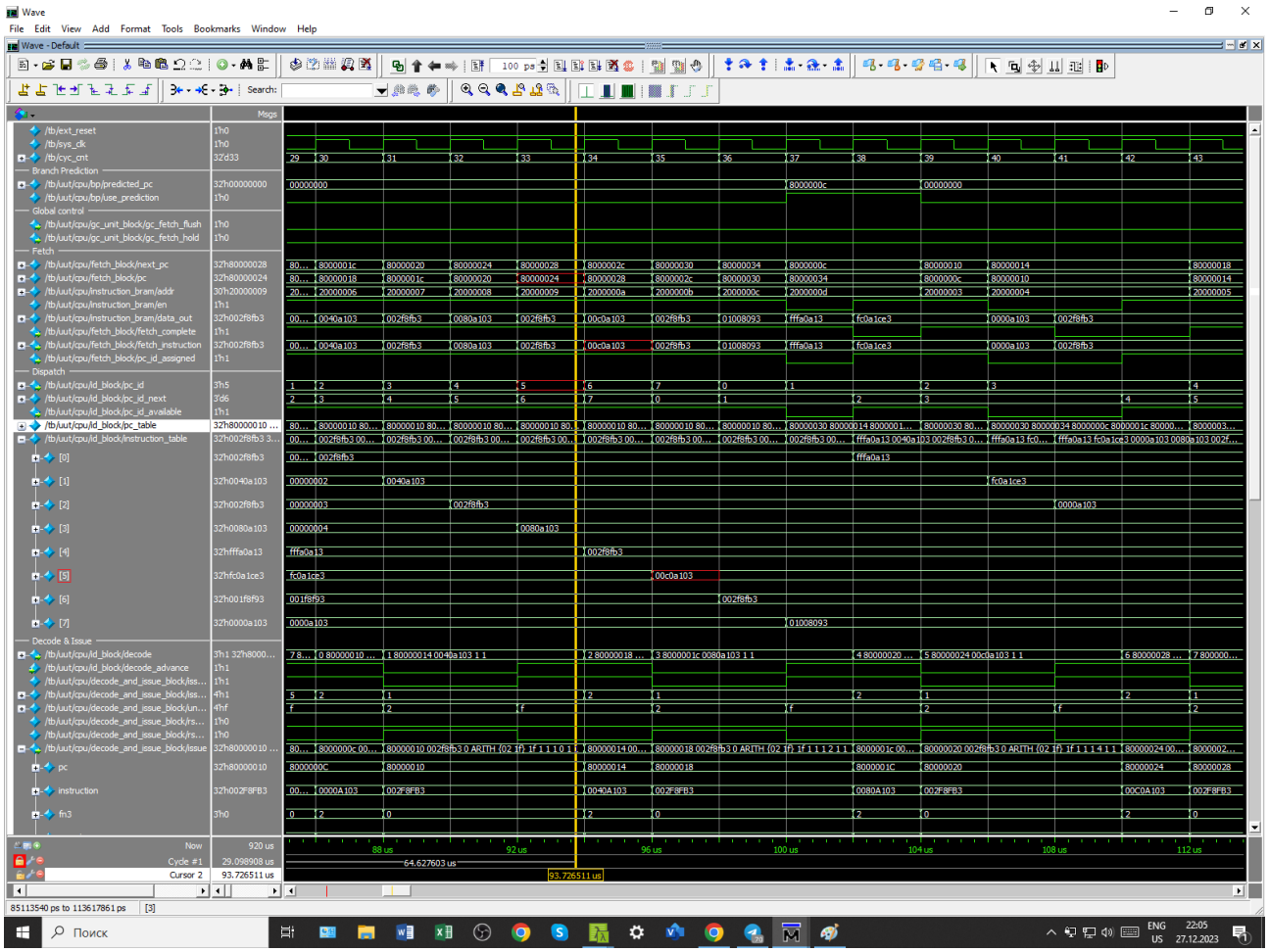


Рисунок 1 – Временная диаграмма выполнения стадий выборки и диспечеризации команды по адресу 80000024, 2-я итерация

2.3 Задание 3

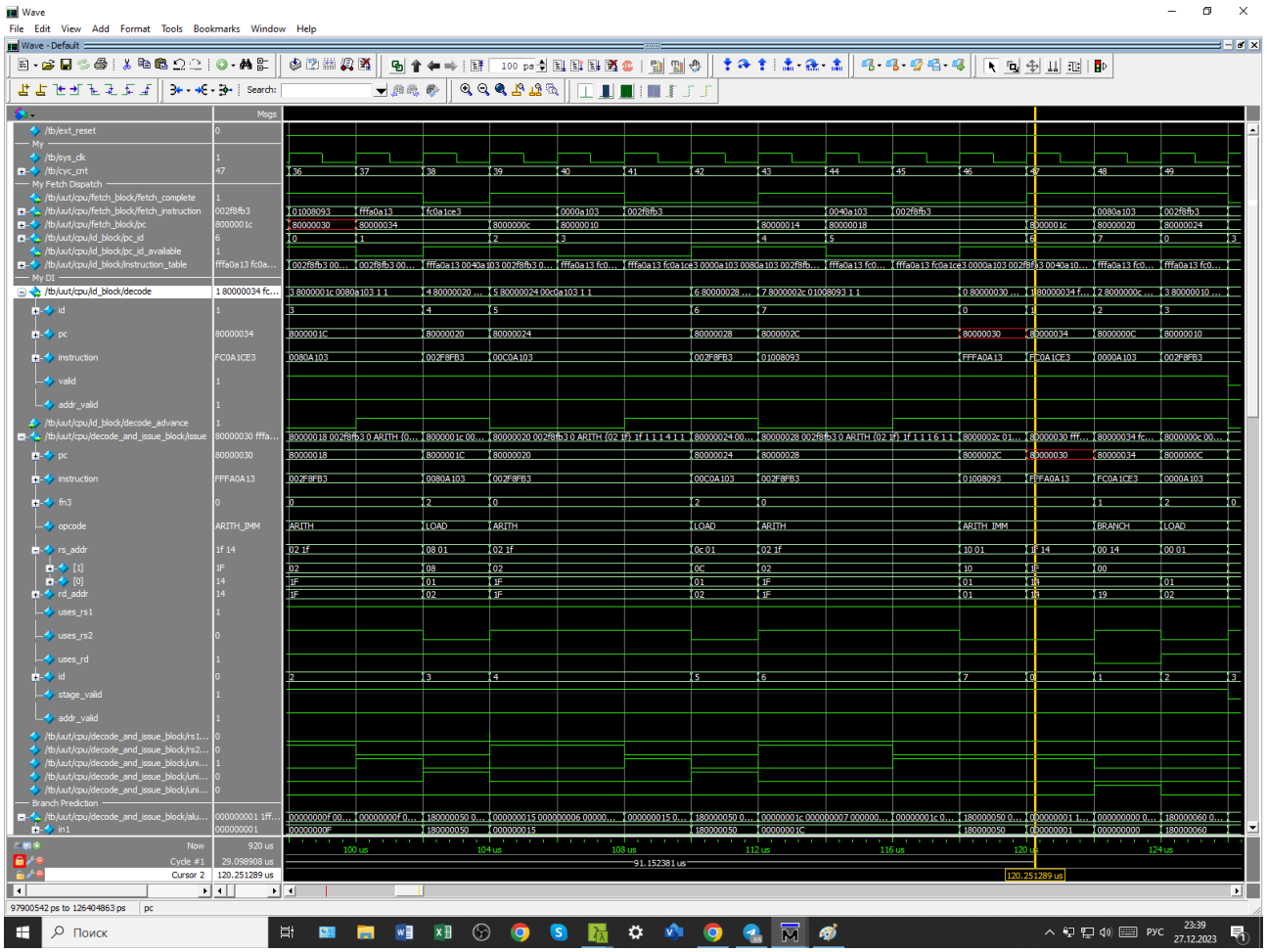


Рисунок 2 – Временная диаграмма выполнения стадии декодирования и планирования на выполнение команды по адресу 80000030, 2-я итерация

2.4 Задание 4

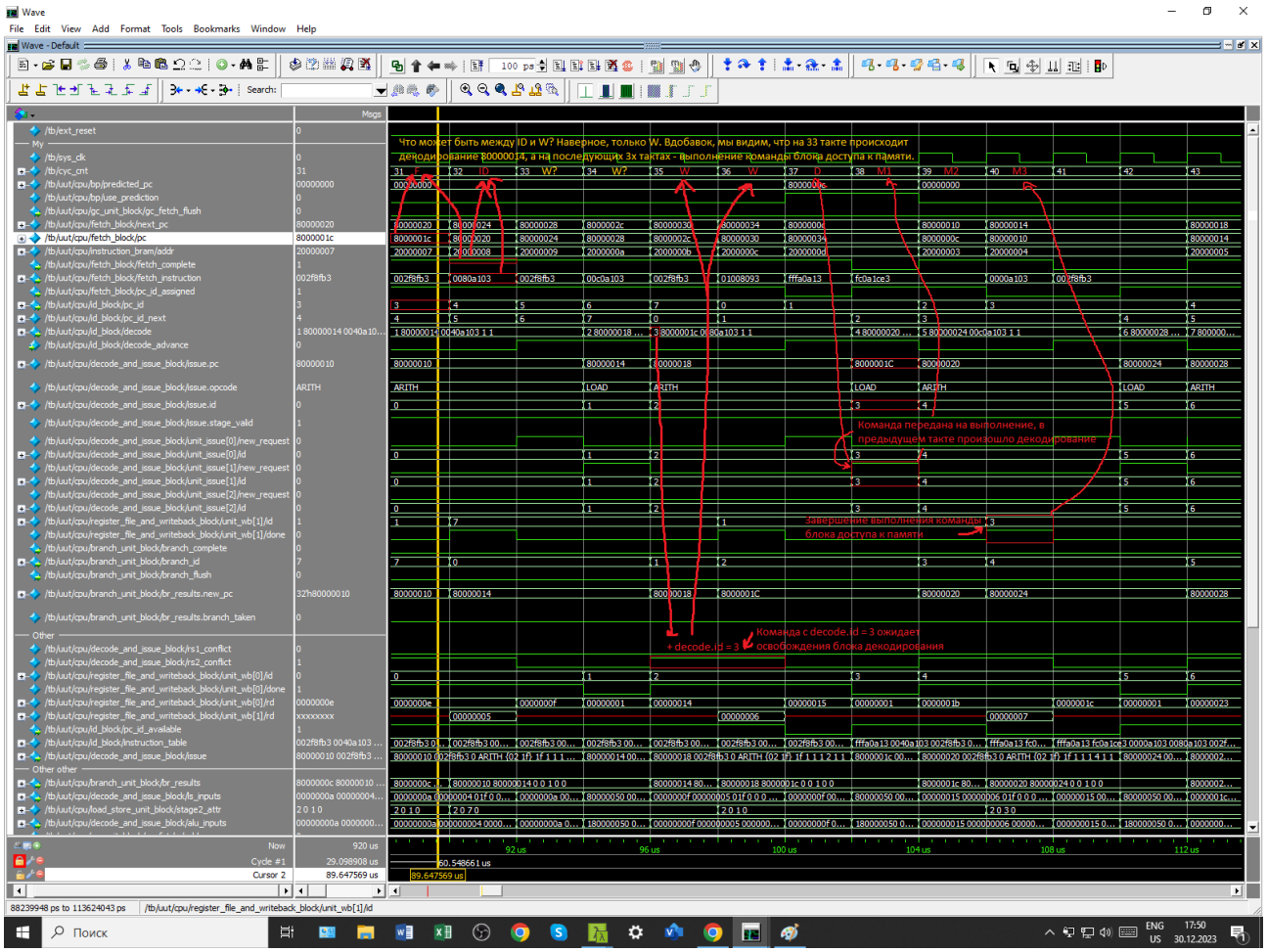


Рисунок 3 – Временная диаграмма выполнения стадии выполнения команды по адресу 8000001c, 2-я итерация

2.5 Задание 5

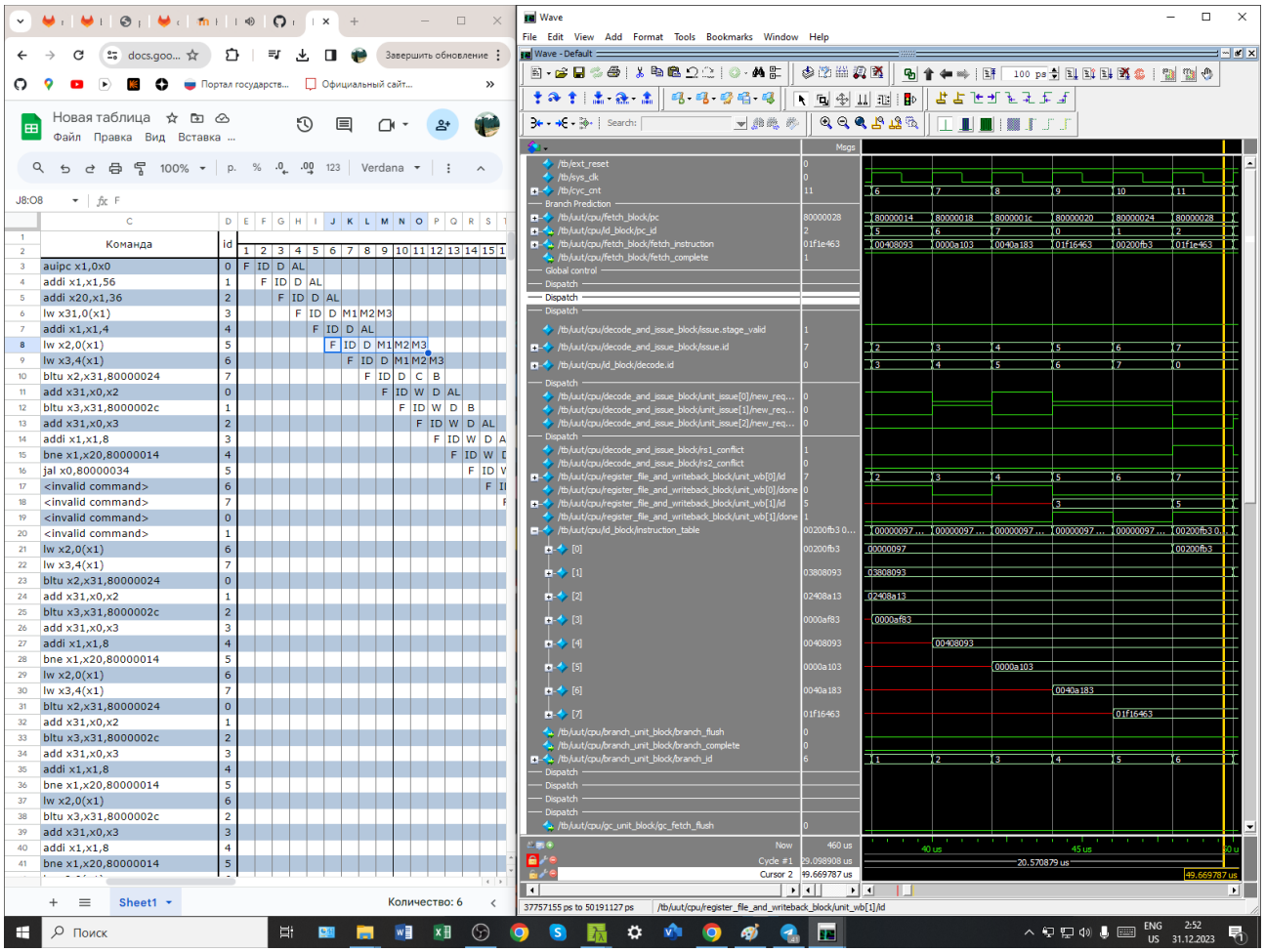


Рисунок 4 – Временная диаграмма всех стадий выполнения команды, обозначенной в тексте программы #!

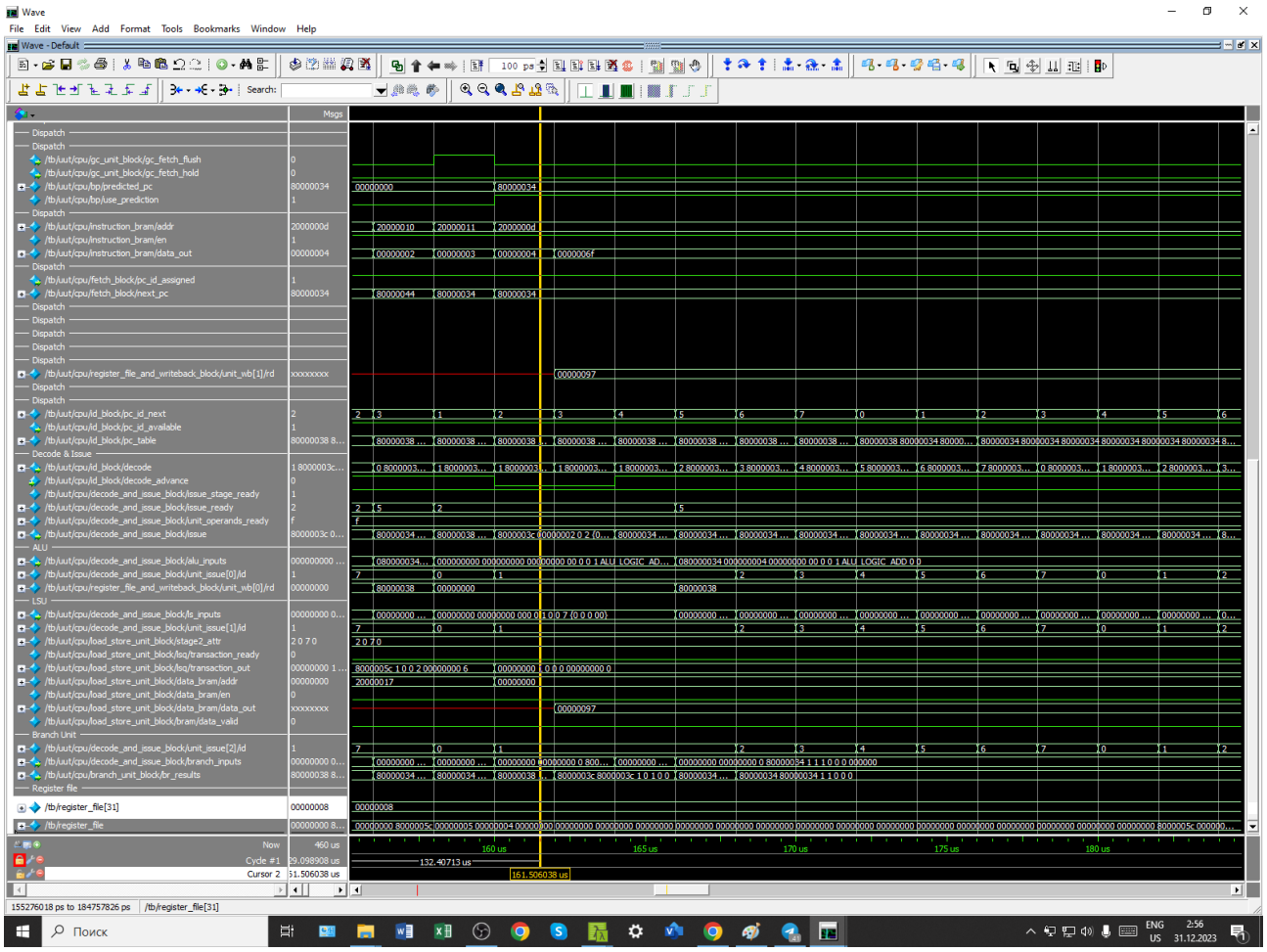


Рисунок 5 – В результате выполнения программы, регистр x31 принимает значение 8, как и ожидалось

Адрес	Код команды	Команда	id	Номер такта
80000000<_start>	00000097	auipc x1,0x0	0	F ID D AL
80000004	03808093	addi x1,x1,56	1	F ID D AL
80000008	02408a13	addi x20,x1,36	2	F ID D AL
8000000c	0000af83	lw x31,0(x1)	3	F ID D M1 M2 M3
80000010	00408093	addi x1,x1,4	4	F ID D AL
80000014<lp>	0000a103	lw x2,0(x1)	5	F ID D M1 M2 M3
80000018	0040a183	lw x3,4(x1)	6	F ID D M1 M2 M3
8000001c	01f16463	bltu x2,x31,80000024	7	F ID D C B
80000020	00200fb3	add x31,x0,x2	0	F ID W D AL
80000024	01f1e463	bltu x3,x31,8000002c	1	F ID W D B
80000028	00300fb3	add x31,x0,x3	2	F ID W D AL
8000002c	00808093	addi x1,x1,8	3	F ID W D AL
80000030	ff4092e3	bne x1,x20,80000014	4	F ID W D B
80000014<lp>	0000a103	lw x2,0(x1)	5	F ID W D M1 M2 M3
80000018	0040a183	lw x3,4(x1)	6	F ID W D M1 M2 M3
8000001c	01f16463	bltu x2,x31,80000024	7	F ID W D C B
80000020	00200fb3	add x31,x0,x2	0	F ID W W D X
80000024	01f1e463	bltu x3,x31,8000002c	1	F ID W W D X
80000028	00300fb3	add x31,x0,x3	2	F ID W W D X
8000002c	00808093	addi x1,x1,8	3	F ID W X
80000030	ff4092e3	bne x1,x20,80000014	4	F ID X
80000014<lp>	0000a103	lw x2,0(x1)	5	F X
80000024	01f1e463	bltu x3,x31,8000002c	6	FX
80000028	00300fb3	add x31,x0,x3	2	F ID D B
8000002c	00808093	addi x1,x1,8	3	F ID D X
80000030	ff4092e3	bne x1,x20,80000014	4	F ID DX
80000014<lp>	0000a103	lw x2,0(x1)	5	F X
8000002c	00808093	addi x1,x1,8	6	FX
80000030	ff4092e3	bne x1,x20,80000014	4	F ID D AL
80000014<lp>	0000a103	lw x2,0(x1)	5	F ID D B
80000018	0040a183	lw x3,4(x1)	6	F ID D M1 M2 M3
8000001c	01f16463	bltu x2,x31,80000024	7	F ID D M1 M2 M3
80000020	00200fb3	add x31,x0,x2	0	F ID D C B
80000024	01f1e463	bltu x3,x31,8000002c	1	F ID D AL X
80000028	00300fb3	add x31,x0,x3	2	F ID W? X

Рисунок 6 – Трасса выполнения неоптимизированной программы

Конфликты возникают из-за того, что происходит обращение к регистру, запись в который ещё не была завершена. Как можно заметить по псевдокоду, строка x1 += enroll (add x1, x1, elem_sz*enroll) будет выполнена вне зависимости от предшествующих условных переходов, а также значение регистра x1 не используется после строки lw x3, 4(x1) в пределах выполнения тела цикла. Следовательно, если поместить строку add x1, x1, slem_sz*enroll сразу после lw x3, 4(x1), это не повлияет на результат выполнения программы, зато даст дополнительную задержку в один такт между загрузкой значения в x2 и чтением из него, что позволит устранить конфликты.

```
1  .section .text
2          .globl _start;
3          len = 9
4          enroll = 2
5          elem_sz = 4
6
7  _start:
8          la x1, _x
9          addi x20, x1, elem_sz*len
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         add x1, x1, elem_sz*enroll
16         bltu x2, x31, lt1
17         add x31, x0, x2
18 lt1:    bltu x3, x31, lt2
19         add x31, x0, x3
20 lt2:
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 0x8
30         .4byte 0x6
31         .4byte 0x7
32         .4byte 0x5
33         .4byte 0x4
```

Листинг 6 – Оптимизированный код программы по варианту на языке ассемблера

```
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:       00808093    addi    x1,x1,8
80000020:       01f16463    bltu    x2,x31,80000028 <lt1>
80000024:       00200fb3    add     x31,x0,x2
80000028 <lt1>:
80000028:       01f1e463    bltu    x3,x31,80000030 <lt2>
8000002c:       00300fb3    add     x31,x0,x3
80000030 <lt2>:
80000030:       ff4092e3    bne     x1,x20,80000014 <lp>
80000034 <lp2>:
80000034:       0000006f    jal     x0,80000034 <lp2>

Disassembly of section .data:
80000038 <_x>:
80000038:       0001        c.addi  x0,0
8000003a:       0000        unimp
8000003c:       0002        0x2
8000003e:       0000        unimp
80000040:       00000003    lb      x0,0(x0) # 0 <enroll-0x2>
80000044:       0004        c.addi4spn      x9,x2,0
80000046:       0000        unimp
80000048:       0008        c.addi4spn      x10,x2,0
8000004a:       0000        unimp
8000004c:       0006        0x6
8000004e:       0000        unimp
80000050:       00000007    0x7
80000054:       0005        c.addi  x0,1
80000056:       0000        unimp
80000058:       0004        c.addi4spn      x9,x2,0
```

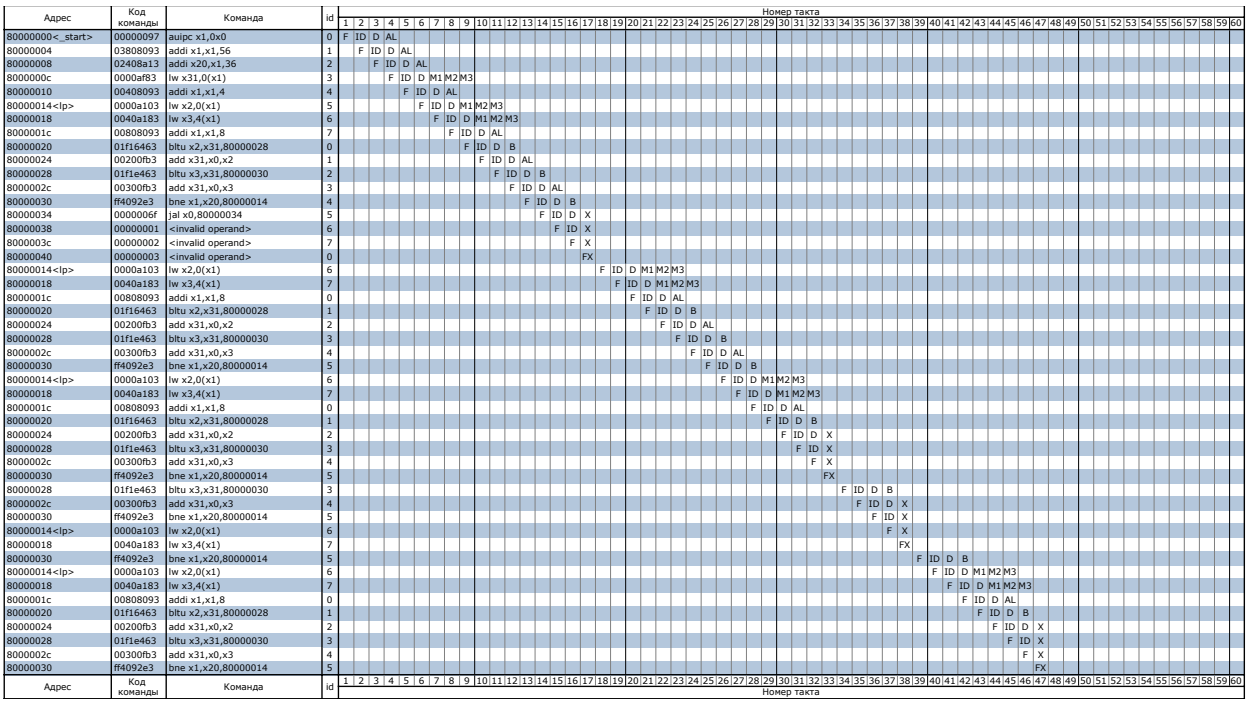
Листинг 7 – Дизассемблированный листинг оптимизированного кода программы по варианту

```
00000097
03808093
02408a13
0000af83
00408093
0000a103
0040a183
00808093
01f16463
00200fb3
01f1e463
00300fb3
ff4092e3
0000006f
00000001
00000002
00000003
00000004
00000008
00000006
00000007
00000005
00000004
```

Листинг 8 – Оптимизированный код программы по варианту в шестнадцатеричном представлении

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

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

void _start() {
    unsigned *x1 = _x; // la x1, _x
    unsigned *x20 = x1 + len; // addi x20, x1, elem_sz*len
    unsigned x31 = x1[0]; // lw x31, 0(x1)
    x1 += 1; // addi x1, x1, elem_sz*1

    do {
        // lp
        unsigned x2 = x1[0]; // lw x2, 0(x1)
        unsigned x3 = x1[1]; // lw x3, 4(x1)

        x1 += enroll; // add x1, x1, elem_sz*enroll

        if (x2 < x31) { // bltu x2, x31, lt1
        } else {
            x31 = x2; // add x31, x0, x2
        }
        // lt1
        if (x3 < x31) { // bltu x3, x31, lt2
        } else {
            x31 = x3; // add x31, x0, x3
        }
        // lt2
    } while (x1 != x20); // bne x1, x20, lp

    for (;;); // lp2: j lp2
}
```

Листинг 9 – Псевдокод оптимизированной программы по варианту



Адрес	Код команды	Команда	id	Номер такта
80000000<_start>	00000097	auipc x1,0x0	0	F ID D AL
80000004	03808093	addi x1,x1,56	1	F ID D AL
80000008	02408a13	addi x20,x1,36	2	F ID D AL
8000000c	0000af83	lw x31,0(x1)	3	F ID D M1 M2 M3
80000010	00408093	addi x1,x1,4	4	F ID D AL
80000014<lp>	0000a103	lw x2,0(x1)	5	F ID D M1 M2 M3
80000018	0040a183	lw x3,4(x1)	6	F ID D M1 M2 M3
8000001c	00808093	addi x1,x1,8	7	F ID D AL
80000020	01f16463	bltu x2,x31,80000028	0	F ID D B
80000024	00200fb3	add x31,x0,x2	1	F ID D AL
80000028	01f1e463	bltu x3,x31,80000030	2	F ID D B
8000002c	00300fb3	add x31,x0,x3	3	F ID D AL
80000030	ff4092e3	bne x1,x20,80000014	4	F ID D B
80000034	0000006f	jal x0,80000034	5	F ID D X
80000038	00000001	<invalid operand>	6	F ID X
8000003c	00000002	<invalid operand>	7	F X
80000040	00000003	<invalid operand>	0	FX
80000014<lp>	0000a103	lw x2,0(x1)	6	F ID D M1 M2 M3
80000018	0040a183	lw x3,4(x1)	7	F ID D M1 M2 M3
8000001c	00808093	addi x1,x1,8	0	F ID D AL
80000020	01f16463	bltu x2,x31,80000028	1	F ID D B
80000024	00200fb3	add x31,x0,x2	2	F ID D AL
80000028	01f1e463	bltu x3,x31,80000030	3	F ID D B
8000002c	00300fb3	add x31,x0,x3	4	F ID D AL
80000030	ff4092e3	bne x1,x20,80000014	5	F ID D B
80000014<lp>	0000a103	lw x2,0(x1)	6	F ID D M1 M2 M3
80000018	0040a183	lw x3,4(x1)	7	F ID D M1 M2 M3
8000001c	00808093	addi x1,x1,8	0	F ID D AL
80000020	01f16463	bltu x2,x31,80000028	1	F ID D B
80000024	00200fb3	add x31,x0,x2	2	F ID D X
80000028	01f1e463	bltu x3,x31,80000030	3	F ID X
8000002c	00300fb3	add x31,x0,x3	4	F X
80000030	ff4092e3	bne x1,x20,80000014	5	FX
80000028	01f1e463	bltu x3,x31,80000030	3	F ID D B
8000002c	00300fb3	add x31,x0,x3	4	F ID D X
80000030	ff4092e3	bne x1,x20,80000014	5	F ID D X
80000014<lp>	0000a103	lw x2,0(x1)	6	F X
80000018	0040a183	lw x3,4(x1)	7	FX
80000030	ff4092e3	bne x1,x20,80000014	5	F ID D B
80000014<lp>	0000a103	lw x2,0(x1)	6	F ID D M1 M2 M3
80000018	0040a183	lw x3,4(x1)	7	F ID D M1 M2 M3
8000001c	00808093	addi x1,x1,8	0	F ID D AL
80000020	01f16463	bltu x2,x31,80000028	1	F ID D B
80000024	00200fb3	add x31,x0,x2	2	F ID D X
80000028	01f1e463	bltu x3,x31,80000030	3	F ID X
8000002c	00300fb3	add x31,x0,x3	4	F X
80000030	ff4092e3	bne x1,x20,80000014	5	FX
Адрес	Код команды	Команда	id	Номер такта

Рисунок 7 – Трасса выполнения оптимизированной программы

3 Заключение

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

На основе изученных материалов был найден способ оптимизировать программу.

Цель работы была достигнута.