

# 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**НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика** 

#### ОТЧЕТ

### по лабораторной работе № 1 вариант № 3

Название Изучение принципов работы микропроцессорного ядра RISC-V Дисциплина Организация ЭВМ и систем

Студент гр. ИУ6-74Б		М.А. Гейне
•	(Подпись, дата)	(И.О.Фамилия)
Преподаватель		_А.Ю. Попов_
•	(Подпись, дата)	(И.О.Фамилия)

# Содержание

	Задание	3
1	Сборка и изучение программы	4
2	Изучение процессов выборки и диспетчеризации	6
3	Изучение процесса декодирования команды	7
4	Изучение процесса исполнения команды	8
5	Трассировка программы	9
	Выволы	15

**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.

### Задание

В ходе выполнения лабораторной работы необходимо:

- 1. Провести сборку программы по индивидуальному варианту, изучить её исходный код;
- 2. Изучить процесс выборки и диспетчеризации команды;
- 3. Изучить процесс декодирования команды;
- 4. Изучить процесс выполнения команды;
- 5. Провести анализ выполнения программы по индивидуальному варианту. Провести трассировку программы. Выявить пути для оптимизации, оптимизировать и привести трассировку оптимизированной программы.

### 1 Сборка и изучение программы

В соответствии с индивидуальным вариантом была выбрана программа, исходный код которой приведён ниже.

```
.section .text
        .globl start;
        len = 8 #Размер массива
        enroll = 1 #Количество обрабатываемых элементов за одну
        elem sz = 4 #Размер одного элемента массива
start:
        la x1, x
        addi x20, x1, elem sz*(len-1) #Адрес последнего элемента
lp:
        1w x2, 0(x1)
        add x31, x31, x2 #!
        addi x1, x1, elem sz*enroll
       bne x1, x20, lp
        addi x31, x31, 1
lp2: j lp2
        .section .data
        .4byte 0x1
х:
        .4byte 0x2
        .4byte 0x3
        .4byte 0x4
        .4byte 0x5
        .4byte 0x6
        .4byte 0x7
        .4byte 0x8
```

По результатам сборки был получен следующий дизассемблерный листинг.

```
SYMBOL TABLE:
80000000 1
           d .text
                             00000000 .text
80000024 1
             d .data
                             00000000 .data
             df *ABS*
00000000 1
                             00000000 task.o
00000008 1
                *ABS*
                             00000000 len
00000001 1
                *ABS*
                             00000000 enroll
00000004 1
                *ABS*
                             00000000 elem sz
80000024 1
                             00000000 x
                .data
8000000c 1
                             00000000 lp
                .text
80000020 1
                .text
                             00000000 lp2
80000000 g
                             00000000 start
                .text
```

```
Disassembly of section .text:
80000000 < start>:
80000000:
                 00000097
                                             auipc
                                                          x1,0x0
80000004:
                 02408093
                                             addi
                                                          x1, x1, 36
→ # 80000024 < x>
80000008:
                 01c08a13
                                                          x20, x1, 28
                                             addi
8000000c <lp>:
800000c:
                 0000a103
                                                       x2,0(x1)
                                             lw
80000010:
                 002f8fb3
                                             add
                                                        x31,x31,x2
80000014:
                 00408093
                                                         x1, x1, 4
                                             addi
80000018:
                 ff409ae3
                        x1,x20,8000000c <lp>
            bne
8000001c:
                 001f8f93
                                             addi
                                                         x31,x31,1
80000020 <1p2>:
80000020:
                  0000006f
                       x0,80000020 <1p2>
            jal
Disassembly of section .data:
80000024 < x>:
80000024:
                  0001
                                               c.addi
                                                              x0,0
80000026:
                 0000
                                               unimp
80000028:
                 0002
                                               0x2
8000002a:
                 0000
                                               unimp
8000002c:
                 0000003
                                             lb
                                                        x0,0(x0) #
\rightarrow 0 <enroll-0x1>
80000030:
                 0004
                             x9,x2,0
            c.addi4spn
\hookrightarrow
80000032:
                 0000
                                               unimp
80000034:
                  0005
                                               c.addi
                                                              x0,1
80000036:
                 0000
                                               unimp
80000038:
                                               0x6
                 0006
8000003a:
                  0000
                                               unimp
8000003c:
                 00000007
                                             0x7
80000040:
                 0008
                               x10, x2, 0
            c.addi4spn
```

Изучив исходный код программы был составлен псевдокод, приведённый ниже.

```
DATA:
x := [1, 2, 3, 4, 5, 6, 7, 8]
```

```
begin
    x1 := указатель на _x[0]
    x20 := указатель на _x[7]

do
    x2 := значение по x1
    x31 := x31 + x2
    x1++
    while (x1!=x20)

x31++
end
```

Изучив алгоритм работы программы было установлено, что в конце её выполнения x31 = 29.

### 2 Изучение процессов выборки и диспетчеризации

В процессе выборки команды происходит выборка из памяти команд по адресу, который записан в РС. Выбранные данные помещаются на линию *fetch\_instruction*. Выборка команды занимает один такт, однако может быть приостановлена, если далее по конвейеру данные некому обработать.

В рамках процесса диспетчеризации выбранные данные на предыдущем шаге помещаются в очередь команд, получая при этом id<8. В случае, если очередь полностью заполнена, выда-ётся сигнал, приостанавливающий выборку.

По индивидуальному варианту была сохранена временная диаграмма выборки и диспетчеризации команды с адресом **8000014, 1-я итерация**, приведённая на рисунке 1.

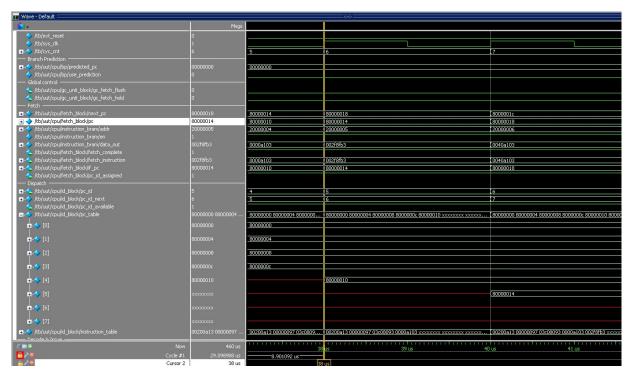


Рисунок 1 – Временная диаграмма выборки и диспетчеризации

#### 3 Изучение процесса декодирования команды

Из очереди команд данные поступают на декодер, где происходит декодирование команды. В рамках этого процесса определяются код операции, исполнительный блок, исходные регистры и регистры назначения, специфические для команды характеристики. Декодирование производится комбинационно и занимает один такт. Следующим тактом производится планирование на выполнение. Однако в некоторых случаях можно наблюдать конфликты при выполнении операции, например при параллельной попытке доступа к одному регистру разных исполнительных блоков. В таком случае команда не выдаётся на исполнение, новая команда не выбирается из очереди.

По индивидуальному варианту была сохранена временная диаграмма декодирования команды с адресом 80000020, 1-я ите-

#### рация, приведённая на рисунке 2.

rve - Default	-						
•	Msgs		<u> </u>				
tb/uut/cpu/id_block/decode		3 8000001c 0080a103	4 80000020 002f8fb3 1 1	5 80000024 00:0a103 1 1			6 80000028 002f8fb3 1 1
±- <b>/&gt;</b> id		3	14	15			6
→ pc	8000	8000001C	180000020	180000024			180000028
j <b>→</b> bc	8000	8000001C	18000020	180000024			180000028
	002F	0080A103	002F8FB3	100C0A103			002F8F83
- ◆ valid							
⊢-y valid							
- ◆ addr_valid							
/tb/uut/cpu/id_block/decode_advance	9:1						
/tb/uut/cpu/decode_and_issue_block/issue_stage_ready	1		<del>                                     </del>				
/tb/uut/cpu/decode_and_issue_block/issue_ready	0010	0001	00010	0001			0010
/tb/uut/cpu/decode_and_issue_block/unit_operands_ready		1111	į	0010		1111	
/tb/uut/cpu/decode_and_issue_block/rs1_conflict			!				
/tb/uut/cpu/decode_and_issue_block/rs2_conflict	8000						
/tb/uut/cpu/decode_and_issue_block/issue			8000001c 0080a103 2 LOAD {0		F) IF 1 1 1 4 1 1		80000024 00x0a103 2 LOAD {0.
r∳ pc	8000	80000018	8000001C	80000020			80000024
	008	002F8FB3	0080A103	X002F8FB3			I00C0A103
				1 302 31 35			1555507100
		0	2	10			12
_ <b>∜</b> opcode	LOAD	ARITH	ILOAD	IARITH			ILOAD
opcode	LOAD	ARLIN	COAD	ARTIH			TEOMO
-👉 rs_addr	08 01	02 1f	08 01	02 If			Oc 01
D-4-[1]	08	02	108	102			100
<b>→</b> [0]		1F	01	1F			01
		1F	02	I IF			102
uses_rs1							
			l				
⊢∜ uses_rs2							
id		2	13	4			15
stage_valid			ľ	Î.			ľ
─ <b>今</b> addr_valid							
/tb/uut/cpu/decode_and_issue_block/alu_inputs	1800	00000000f 000000006	180000050 000000000 800000	000000015 000000006 00000015	06 0 0 1 ALU LOGIC ADD 0 0	000000015 000000007 000000	180000050 000000000 800000
/tb/uut/cpu/decode_and_issue_block/unit_issue[0]/new_request							
/tb/uut/cpu/decode_and_issue_block/unit_issue[0]/id		2	)3	(4			5
/tb/uut/cpu/register_fie_and_writeback_block/unit_wb[0]/id	3	2	3	4			5
/ http://dock.com/register_fie_and_writeback_block/unit_wb[0]/done / http://com/register_fie_and_writeback_block/unit_wbf0]/rd	nnn	00000015	00000001	10000001b		T0000001c	100000001
	ow 460 us	1 1 1 1 1 1 1					
S Cycle :		72,901092 us	us 10	4 us 10	06 us 10	8 us 1	10 us

Рисунок 2 – Временная диаграмма декодирования

На диаграмме можно наблюдать, что после декодирования и выдачи на исполнение был обнаружен конфликт (выставлен сигнал  $rs2\_conflict$ ). По этой причине сигнал  $decode\_advance$  снят, новая команда не выбирается из очереди, а уже поступившая на декодер - не подаётся на исполнение. После снятия сигнала о конфликте процесс возобновляется.

### 4 Изучение процесса исполнения команды

После декодирования команда отдаётся на исполнение. Исполнительный блок определяется сигналом  $new\_request[i]$ , где i=0,1,2 - номер блока. Блоку АЛУ соответствует i=0, блоку обращения к памяти - i=1, блоку ветвления - i=2.

На АЛУ выполняются арифметические операции. Блок является комбинационным, результат предоставляет в течение 1 так-

та. Блок ветвления предоставляет результат за 1 такт. В случае ошибки предсказания адреса в следующем после поступления на исполнение такте будет выдан сигнал сброса, который приведёт к очистке уже выбранной очереди команд. Блок обращения к памяти выполняет запрос за 3 такта. Однако если бы было обращение к внешней памяти, то на его работу потребовалось бы значительно больше тактов.

По индивидуальному варианту была сохранена временная диаграмма выполнения команды с адресом **8000008**, приведённая на рисунке 3.

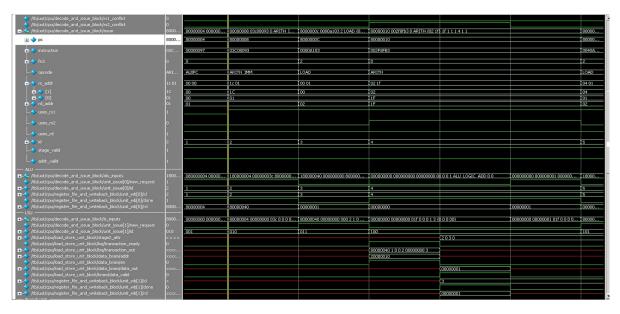


Рисунок 3 — Временная диаграмма выполнения

### 5 Трассировка программы

На основе полученных знаний была проведена трассировка индивидуальной программы. В ходе выполнения было подтверждено, что  $\mathbf{x31} = \mathbf{29}$ . Результат трассировки представлен на рисунке 4.

Адрес	Код команды	Команда	10 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 52 53 54 55 56 57 58 59 60 61
0000000 <start>:</start>	00000097	auipc x1.0x0	0 F ID D AL	
000004:	02408093	addi x1,x1,36 # 80000024 <_x>	1 FID D AL	
000008:	01c08a13	addi x20,x1,28	2 F ID D AL	
00000c <lp>:</lp>	0000a103	lw x2,0(x1)	3 F ID D M1 M2 M3	
000010:	002f8fb3	add x31,x31,x2	4 FIDDCCAL	
000014:	00408093	addi x1,x1,4	5 F ID W W D AL	
0000018:	ff409ae3	bne x1,x20,8000000c <lp></lp>	6 FIDWWDB	
00001c:	001f8f93	addi x31,x31,1	7 FIDWWD	x
0000020 <lp2>:</lp2>	0000000f	jal x0,80000020 <lp2></lp2>	0 FIDWW	DX
0000024	00000001	<invalid operation=""></invalid>	1 FIDW	
0000028	00000002	<invalid operation=""></invalid>	2 FID	*
000002c	00000003	<invalid operation=""></invalid>	3 F	x
0000030	00000004	<invalid operation=""></invalid>		EX
00000c <lp>:</lp>	0000a103	lw x2.0(x1)	0	F ID D N1 M2 M3
000010:	002f8fb3	add x31,x31,x2	1	FID C ClaL
000014:	00408093	addi x1.x1.4	2	F ID W W D AL
000018:	ff409ae3	bne x1,x20,8000000c <lp></lp>	3	FIDWWDB
00000c <lp>:</lp>	0000a103	lw x2,0(x1)	4	FIDWWDM1M2M3
000010:	002f8fb3	add x31.x31.x2	5	FID W W D C C AL
000014:	00408093	addi xl.xl.4	6	FIDWWWWDAL
000018:	ff409ae3	bne x1,x20,8000000c <lp></lp>	7	FIDWWWWDB
00000c <lp>:</lp>	0000a103	lw x2,0(x1)	0	FID W W W D NIM2M3
000010:	002f8fb3	add x31,x31,x2	1	FIDWWWWDCCAL
000014:	00408093	addi x1.x1.4	2	FIDWWWWWDAL
000014:	ff409ae3	bne x1,x20,8000000c <lp></lp>	3	FIDWWWWWDB
00000c <lp>:</lp>	0000a103	lw x2.0(x1)	4	FIDWWW W W D MINZNS
000010:	002f8fb3	add x31,x31,x2	5	FIDWWWWWDCCAL
000014:	00408093	addi x1.x1.4	6	FIDWWWWWDAL
000014:	ff409ae3	bne x1,x20,8000000c <lp></lp>	7	FIDW W W W W D B
000016: 00000c <lp>:</lp>	0000a103	lw x2.0(x1)		FIDWWWWWWDMIM2M3
000010:	002f8fb3	add x31,x31,x2	1	FIDWWWWWDCCAL
000014:	00408093	addi x1,x1,4	2	FIDWWWWWDDAL
000014:	ff409ae3	bne x1,x20,8000000c <lp></lp>	3	FIDWWWWWWWDAL
00000c <lp>:</lp>	0000a103	lw x2.0(x1)	3	FIDWWWWWWDMIN
00000C <cp>:</cp>	0000a103	add x31,x31,x2	5	FIDWWWWDCCIAL
000010:	00408093	addi x1.x1.4	5	FIDWWWWDCCAL
000014:	ff409ae3	bne x1,x20,8000000c <lp></lp>	7	FIDWWWWWWWDB
		lw x2.8(x1)	/	FIDWWWWWDX
00000c <lp>:</lp>	0000a103		0	
000010:	002f8fb3	add x31,x31,x2	1	F ID W W W W DX
000014:	00408093	addi x1,x1,4	2	F ID W W W W X
000018:	ff409ae3	bne x1,x20,8000000c <lp></lp>	3	F ID W W X
00000c <lp>:</lp>	0000a103	lw x2,0(x1)	4	F ID X
000010:	002f8fb3	add x31,x31,x2	5	F X
000020 <lp2>:</lp2>	0000006f	jal x0,80000020 <lp2></lp2>	6	FX
00001c:	001f8f93	addi x31,x31,1	1	F ID D AL
000020 <lp2>:</lp2>	0000006f	jal x0,80000020 <lp2></lp2>	2	F ID D B
300024	00000001	<invalid operation=""></invalid>	3	F ID D X
000028	00000002	<invalid operation=""></invalid>	4	F ID DX
30002c	00000003	<invalid operation=""></invalid>	5	F X
000030	00000004	<invalid operation=""></invalid>	6	FX
000020 <lp2>:</lp2>	0000006f	jal x0,80000020 <lp2></lp2>	3	F ID
000020 <lp2>:</lp2>	0000006f	jal x0,80000020 <lp2></lp2>	4	F
000020 <lp2>:</lp2>	0000006f	jal x0,80000020 <lp2></lp2>	5	
000020 <lp2>:</lp2>	0000006f	jal x0,80000020 <lp2></lp2>	6	
Адрес		Команда	1 2 2 4 5 6 7 9 0 101112	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 52 53 54 55 56 57 58 59 60

Рисунок 4 — Трассировка программы

Была получена временная диаграмма команды **add x31, x31, x2** с адресом **80000010**, приведённая на рисунке 5.

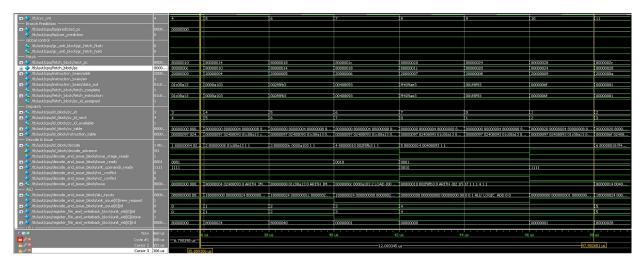


Рисунок 5 — Временная диаграмма обработки команды

В таблице видно, что после выборки данных из памяти в следующей команде случается конфликт. Это происходит из-за того, что в этой команде используется регистр, в который записываются данные из памяти. Чтобы избежать конфликта было решено производить выборку нового аргумента перед началом

очередной итерации цикла и после обработки предыдущего аргумента. Таким образом, к моменту выполнения команды с конфликтом пройдет достаточное количество тактов для выполнения пересылки из памяти. Однако на первой итерации будет также наблюдаться конфликт.

В соответствии с приведёнными выше предложениями по оптимизации была составлена оптимизированная программа, текст которой приведён ниже.

```
.section .text
        .globl start;
        len = 8 #Размер массива
        enroll = 1 #Количество обрабатываемых элементов за одну
        elem sz = 4 #Размер одного элемента массива
start:
        la x1, x
        addi x20, x1, elem sz*(len-1) #Адрес последнего элемента
                1w \times 2, 0 (\times 1)
lp:
        addi x1, x1, elem sz*enroll
        add x31, x31, x2 #!
                1w x2, 0(x1)
        bne x1, x20, lp
        addi x31, x31, 1
lp2: j lp2
        .section .data
        .4byte 0x1
x:
        .4byte 0x2
        .4byte 0x3
        .4byte 0x4
        .4byte 0x5
        .4byte 0x6
        .4byte 0x7
        .4byte 0x8
```

По результатам сборки был получен следующий дизассемблерный листинг.

```
SYMBOL TABLE:
                               00000000 .text
80000000 1
              d .text
80000024 1
              d .data
                               00000000 .data
00000000 1
              df *ABS*
                               00000000 task.o
00000008 1
                 *ABS*
                               00000000 len
0000001 1
                  *ABS*
                               00000000 enroll
                  *ABS*
00000004 1
                               00000000 elem sz
80000024 1
                  .data
                               00000000 x
                               00000000 lp
800000c 1
                  .text
80000020 1
                  .text
                               00000000 lp2
80000000 g
                               00000000 start
                  .text
80000044 g
                               00000000 end
                  .data
Disassembly of section .text:
80000000 < start>:
80000000:
                00000097
                                             auipc
                                                          x1,0x0
80000004:
                  02408093
                                             addi
                                                          x1, x1, 36
    # 80000024 < x>
80000008:
                  01c08a13
                                                          x20, x1, 28
                                             addi
8000000c <lp>:
800000c:
                 0000a103
                                             lw
                                                        x2,0(x1)
80000010:
                                                         x31,x31,x2
                  002f8fb3
                                             add
80000014:
                 00408093
                                                          x1, x1, 4
                                             addi
80000018:
                  ff409ae3
            bne
                        x1,x20,8000000c <1p>
800001c:
                  001f8f93
                                                         x31, x31, 1
                                             addi
80000020 <1p2>:
80000020:
                  0000006f
                        x0,80000020 <1p2>
            jal
Disassembly of section .data:
80000024 < x>:
80000024:
                  0001
                                               c.addi
                                                              x0,0
80000026:
                  0000
                                               unimp
80000028:
                                               0x2
                  0002
8000002a:
                  0000
                                               unimp
8000002c:
                  0000003
                                             lb
                                                        x0,0(x0) #
\rightarrow 0 <enroll-0x1>
80000030:
                  0004
            c.addi4spn
                               x9, x2, 0
80000032:
                  0000
                                               unimp
80000034:
                  0005
                                               c.addi
                                                              x0,1
80000036:
                  0000
                                               unimp
80000038:
                  0006
                                               0x6
8000003a:
                  0000
                                               unimp
```

0x7

8000003c: 00000007 80000040: 0008 → c.addi4spn x10, x2, 0

Была проведена повторная трассировка программы, результат которой отображен на рисунке 6

Адрес	Код команды	Команда	id			ер такта			
		1000 1000 N N	1 2 3		26 27 28 29	30 31 32 33 34 35 36 37 38 3	9 40 41 42 4	3 44 45 46 47 48 49 50 5	1 52 53 54 55 56 5
0000000<_start>:	00000097	auipc x1,0x0	0 F ID D						
000004:	02808093	addi x1,x1,40 # 80000028 <_x>		D D AL					
0000008:	01c08a13	addi x20,x1,28	2 F	ID D AL					
000000c:	0000a103	lw x2,0(x1)	3	F ID D M1 M2 M3					
0000010 <lp>:</lp>	00408093	addi x1,x1,4	5	F ID D AL					
0000014:	002f8fb3	add x31,x31,x2	5	F ID D C AL					
0000018:	0000a103	lw x2,0(x1)	7	FID W D M1 M2 M3					
000001c:	ff409ae3	bne x1,x20,80000010 <lp></lp>	7	F ID W D B					
0000020:	001f8f93	addi x31,x31,1	0	F ID W D X					
0000024 <lp2>:</lp2>	0000006f	jal x0,80000024 <lp2></lp2>	1	F ID W DX					
0000028	00000001	<invalid operation=""></invalid>	3	F ID X					
000002c	00000002	<invalid operation=""></invalid>	3	FX					
0000030	00000003	<invalid operation=""></invalid>	4	FX					
0000010 <lp>:</lp>	00408093	addi x1.x1.4	1	F ID D AL					
0000014:	002f8fb3	add x31,x31,x2		F ID D AL					
0000014:	0000a103	lw x2.0(x1)	3	F ID D M1 M2 M3					
000001c:	ff409ae3	bne x1,x20,80000010 <lp></lp>	4	F ID D B					
000001c.	00408093	addi x1,x1,4	5	F ID D AL					
0000010 <tp>.</tp>	002f8fb3	add x31.x31.x2	0	F ID D AL					
0000014:	0000a103	lw x2,0(x1)	6 7	F ID D M1 M2 M					
	ff409ae3			F ID D B					
000001c:		bne x1,x20,80000010 <lp></lp>	0						
0000010 <lp>:</lp>	00408093	addi x1,x1,4	1	F ID D A					
0000014:	002f8fb3	add x31,x31,x2	2	F ID C					
0000018:	0000a103	lw x2,0(x1)	3		D M1 M2 M3				
000001c:	ff409ae3	bne x1,x20,80000010 <lp></lp>	5	F F	ID D B				
0000010 <lp>:</lp>	00408093	addi x1,x1,4	5		F ID D AL				
0000014:	002f8fb3	add x31,x31,x2	6		F ID D				
0000018:	0000a103	lw x2,0(x1)	7			D M1 M2 M3			
000001c:	ff409ae3	bne x1,x20,80000010 <lp></lp>	0		F	ID D B			
0000010 <lp>:</lp>	00408093	addi x1,x1,4	1			F ID D AL			
0000014:	002f8fb3	add x31,x31,x2	2			F ID D AL			
0000018:	0000a103	lw x2,0(x1)	3			F ID D M1 M2 M3			
000001c:	ff409ae3	bne x1,x20,80000010 <lp></lp>	4			FIDDB			
0000010 <lp>:</lp>	00408093	addi x1.x1.4	5			F ID D AL			
0000014:	002f8fb3	add x31,x31,x2	6			FIDDAL			
0000018:	0000a103	lw x2,0(x1)	6 7			F ID D M	1 M 2 M 3		
000001c:	ff409ae3	bne x1,x20,80000010 <lp></lp>	0			FIDI			
000001C.	00408093	addi x1,x1,4	7				DDX		
0000010 <tp>:</tp>	00408093	add x31,x31,x2	0				IDDX		
0000014:	0000a103	lw x2.0(x1)	3				F X		
0000018:	ff409ae3	bne x1,x20,80000010 <lp></lp>	4				FX		
0000010:	001f8f93		2					D D AL	
		addi x31,x31,1							
0000024 <lp2>:</lp2>	0000006f	jal x0,80000024 <lp2></lp2>	3 4					F ID D B	
0000028	00000001	<invalid operation=""></invalid>	4					F ID D X	
100002c	00000002	<invalid operation=""></invalid>	5					F ID DX	
000030	00000003	<invalid operation=""></invalid>						F X	
1000034	00000004	<invalid operation=""></invalid>	7					FX	
000024 <lp2>:</lp2>	0000006f	jal x0,80000024 <lp2></lp2>	5					F ID D E	3
1000024 <lp2>:</lp2>	0000006f	jal x0,80000024 <lp2></lp2>	6					FIDE	D B
0000024 <lp2>:</lp2>	0000006f	jal x0,80000024 <lp2></lp2>	7					FI	D D B
0000024 <lp2>:</lp2>	0000006f	jal x0,80000024 <lp2></lp2>	0						F ID D B
0000024 <lp2>:</lp2>	0000006f	ial x0.80000024 <lp2></lp2>	1						F ID D B
		,		4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 2					

Рисунок 6 – Трассировка оптимизированной программы

Была получена временная диаграмма команды add x31, x31, х2 с адресом 80000014, приведённая на рисунке 7.

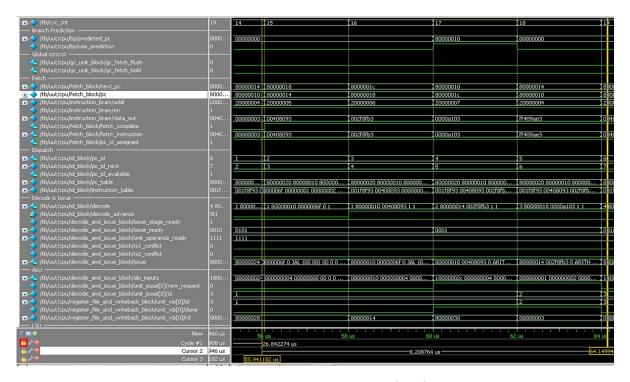


Рисунок 7 – Временная диаграмма обработки команды

Из таблицы видно, что конфликты больше не возникают и команды не находятся в ожидании лишние такты. В результате оптимизации удалось сократить время выполнения основной части программы с 60 тактов до 48, т.е. на 12 тактов.

### Выводы

В ходе выполнения лабораторной работы были изучены принципы работы микропроцессорного ядра RISC-V.

Изучены процессы выборки, диспетчеризации, декодирования и выполнения команд.

Проведена трассировка программы, предоставленной по индивидуальному заданию. В ходе трассировки выявлены конфликты в выполнении команд.

Предложены варианты оптимизации программы. В соответствии с ними программа была изменена.

Проведена трассировка оптимизированной программы. Оптимизация позволила избежать конфликтов и увеличить скорость исполнения программы на 12 тактов.