



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 №2 по курсу "Архитектура ЭВМ"

Тема Изучение принципов работы микропроцессорного ядра RISC-V

Студент Лемешев А. П.

Группа ИУ7-52Б

Преподаватель Дубровин Е.Н.

# Оглавление

<b>Введение</b>	<b>2</b>
<b>1 Основные теоретические сведения</b>	<b>3</b>
<b>2 Эксперименты</b>	<b>4</b>
2.1 Задание 1 . . . . .	4
2.2 Задание 2 . . . . .	6
2.3 Задание 3 . . . . .	7
2.4 Задание 4 . . . . .	7
2.5 Задание 5 . . . . .	8
<b>Заключение</b>	<b>12</b>

# Введение

Основной целью данной лабораторной 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

Изучение архитектуры суперскалярных конвейерных микропроцессоров используется синтезируемое описание микропроцессорного ядра Taiga, реализующего систему команд RV32I семейства RISC-V. Данное описание выполнено на языке описания аппаратуры SystemVerilog.

Термин RISC-V 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.

## 2 Эксперименты

### 2.1 Задание 1

Дизассемблировать программу по индивидуальному варианту.

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

Рис. 2.1: Код неоптимизированной программы

Создается массив из 8 элементов. Он последовательно заполняется числами от 1 до 8. Потом все элементы суммируются и в регистр x31 записывается накопленное значение. Затем содержимое регистра x31 инкрементируется.

```

1  main.elf: file format elf32-littleriscv
2
3  SYMBOL TABLE:
4  80000000 l d .text 00000000 .text
5  80000040 l d .data 00000000 .data
6  00000000 l df *ABS* 00000000 main.o
7  00000008 l *ABS* 00000000 len
8  00000004 l *ABS* 00000000 enroll
9  00000004 l *ABS* 00000000 elem_sz
10 80000040 l .data 00000000 _x
11 80000010 l .text 00000000 lp
12 8000003c l .text 00000000 lp2
13 80000000 g .text 00000000 _start
14 80000060 g .data 00000000 _end

```

Рис. 2.2: Таблица символов

```

17 Disassembly of section .text:
18 80000000 <_start>:
19 80000000: 00000097 auipc x1,0x0
20 80000004: 04008093 addi x1,x1,64 # 80000040 <_x>
21 80000008: 02008a13 addi x20,x1,32
22 8000000c: 00000fb3 add x31,x0,x0
23 80000010 <lp>:
24 80000010: 0000a103 lw x2,0(x1)
25 80000014: 0040a183 lw x3,4(x1)
26 80000018: 002f8fb3 add x31,x31,x2
27 8000001c: 003f8fb3 add x31,x31,x3
28 80000020: 0080a203 lw x4,8(x1)
29 80000024: 00c0a283 lw x5,12(x1)
30 80000028: 004f8fb3 add x31,x31,x4
31 8000002c: 005f8fb3 add x31,x31,x5
32 80000030: 01008093 addi x1,x1,16
33 80000034: fd409ee3 bne x1,x20,80000010 <lp>
34 80000038: 001f8f93 addi x31,x31,1
35 8000003c <lp2>:
36 8000003c: 0000006f jal x0,8000003c <lp2>

```

Рис. 2.3: Дизассемблированная секция текста

```

39  Disassembly of section .data:
40  80000040 <_x>:
41  80000040: 0001      c.addi x0,0
42  80000042: 0000      unimp
43  80000044: 0002      0x2
44  80000046: 0000      unimp
45  80000048: 00000003  lb x0,0(x0) # 0 <elem_sz-0x4>
46  8000004c: 0004      c.addi4spn x9,x2,0
47  8000004e: 0000      unimp
48  80000050: 0005      c.addi x0,1
49  80000052: 0000      unimp
50  80000054: 0006      0x6
51  80000056: 0000      unimp
52  80000058: 00000007  0x7
53  8000005c: 0008      c.addi4spn x10,x2,0

```

Рис. 2.4: Дизассемблированная секция данных

## 2.2 Задание 2

Получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки и диспетчеризации команды с адресом 80000000 на 2-ой итерации.

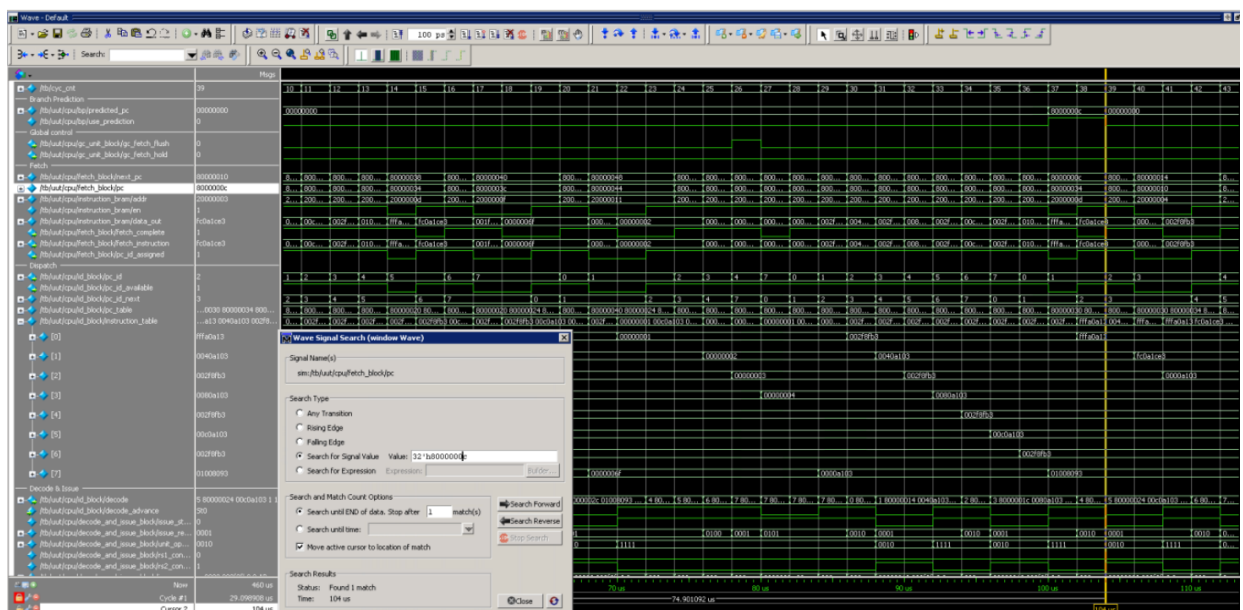


Рис. 2.5: Временная диаграмма выполнения стадий выборки и диспетчеризации команды 80000000с на 2-й итерации

### 2.3 Задание 3

Получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с адресом 80000018 на 2-ой итерации.

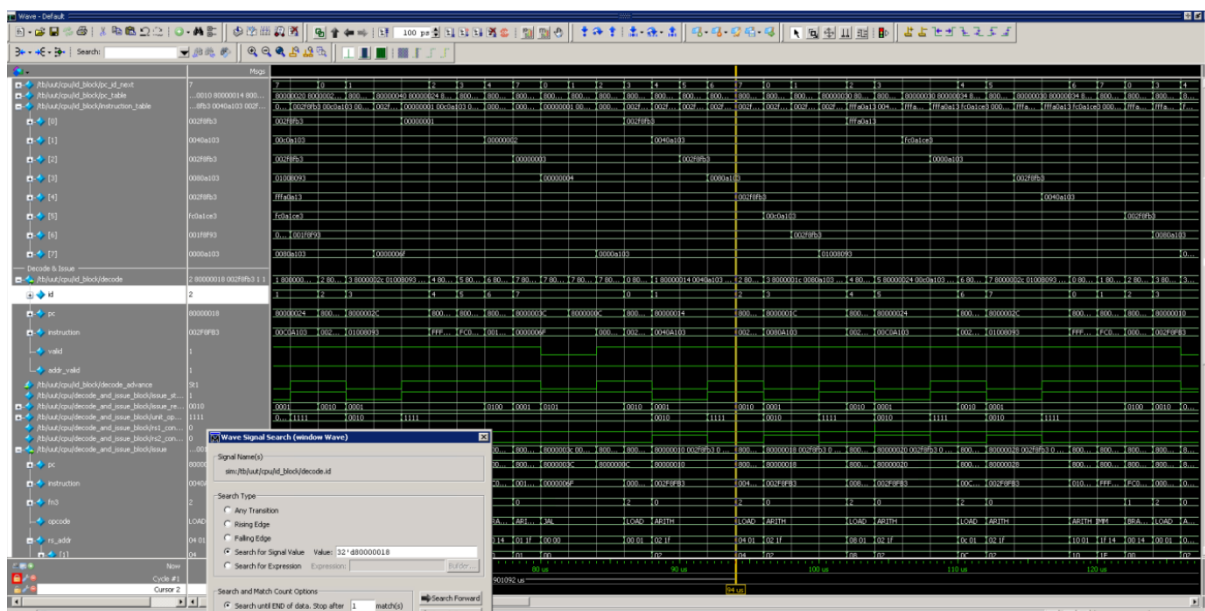


Рис. 2.6: Временная диаграмма выполнения стадий декодирования и планирования на выполнение команды 80000018 на 2-й итерации

## 2.4 Задание 4

Получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с адресом 8000002с на 1-ой итерации.



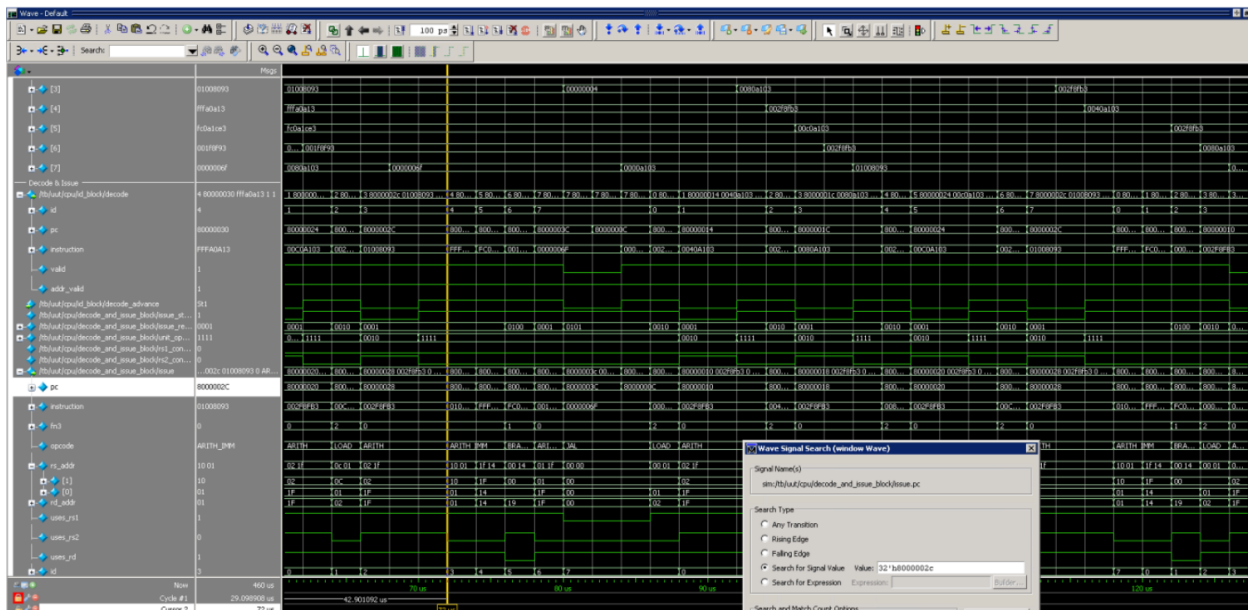


Рис. 2.7: Временная диаграмма выполнения стадии выполнения команды 8000002с на 1-й итерации

## 2.5 Задание 5

Оптимизировать программу.

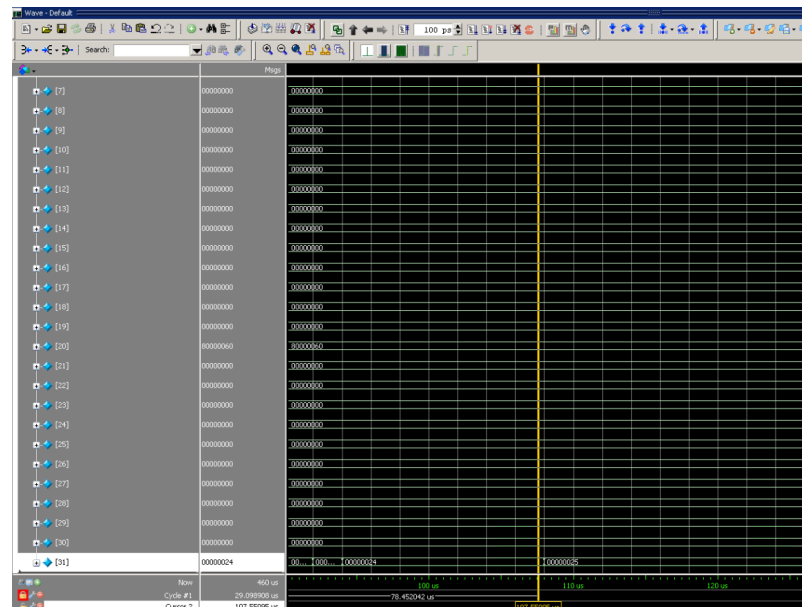


Рис. 2.8: Результат выполнения программы

Значение регистра x31 в конце выполнения программы равно  $25h = 37$ , как и предполагалось ранее.

[illegible]

Рис. 2.9: Трасса работы неоптимизированной программы

Конфликты возникают, когда после выполнения инструкции *lw* выполняется инструкция *add*. Оптимизируем программу, перенеся все команды *lw* выше инструкций *add*.

```

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

```

Рис. 2.10: Код оптимизированной программы

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			
00000000<_start>	00000097	auipc x1,0x0	0	F	ID	D	AL																																									
00000004	03c00093	addi x1,x1,60#0000003c<_x>	1		F	ID	D	AL																																								
00000008	00200a13	addi x20,x1,32	2			F	ID	D	AL																																							
0000000c<lp>	0000a103	lw x2,0(x1)	3				F	ID	D	M1	M2	M3																																				
00000010	0040a183	lw x3,4(x1)	4					F	ID	D	M1	M2	M3																																			
00000014	0080a203	lw x4,8(x1)	5						F	ID	D	M1	M2	M3																																		
00000018	00c0a283	lw x5,12(x1)	6							F	ID	D	M1	M2	M3																																	
0000001c	002f8fb3	add x31,x31,x2	7								F	ID	D	AL																																		
00000020	003f8fb3	add x31,x31,x3	8									F	ID	D	AL																																	
00000024	004f8fb3	add x31,x31,x4	1										F	ID	D	AL																																
00000028	005f8fb3	add x31,x31,x5	2											F	ID	D	AL																															
0000002c	01000093	addi x1,x1,16	3												F	ID	D	AL																														
00000030	fd409ee3	bne x1,x20,8000000c<lp>	4													F	ID	D	B																													
00000034	001f8f93	addi x31,x31,1	5														F	ID	D	X																												
00000038	0000006f	jal x0,80000038<lp2>	6															F	ID	X																												
0000003c	00000001	<invalid operation>	7																F	X																												
00000040	00000002	<invalid operation>	8																	FX																												
0000000c<lp>	0000a103	lw x2,0(x1)	6																		F	ID	D	M1	M2	M3																						
00000010	0040a183	lw x3,4(x1)	7																			F	ID	D	M1	M2	M3																					
00000014	0080a203	lw x4,8(x1)	8																				F	ID	D	M1	M2	M3																				
00000018	00c0a283	lw x5,12(x1)	1																					F	ID	D	M1	M2	M3																			
0000001c	002f8fb3	add x31,x31,x2	2																					F	ID	D	AL																					
00000020	003f8fb3	add x31,x31,x3	3																						F	ID	D	AL																				
00000024	004f8fb3	add x31,x31,x4	4																							F	ID	D	AL																			
00000028	005f8fb3	add x31,x31,x5	5																								F	ID	D	AL																		
0000002c	01000093	addi x1,x1,16	6																										F	ID	D	AL																
00000030	fd409ee3	bne x1,x20,8000000c<lp>	7																												F	ID	D	B														
0000000c<lp>	0000a103	lw x2,0(x1)	8																														F	ID	D	X												
00000010	0040a183	lw x3,4(x1)	1																															F	ID	X												
00000014	0080a203	lw x4,8(x1)	2																																F	X												
00000018	00c0a283	lw x5,12(x1)	3																																	FX												
00000034	001f8f93	add x31,x31,1	1																																													
00000038	0000006f	jal x0,80000038<lp2>	2																																													
0000003c	00000001	<invalid operation>	3																																													
00000040	00000002	<invalid operation>	4																																													
00000044	00000003	<invalid operation>	5																																													
00000048	00000004	<invalid operation>	6																																													
00000038<lp2>	0000006f	jal x0,8000003c<forever>	4																																													
00000038<lp2>	0000006f	jal x0,8000003c<forever>	5																																													
Адрес	Код команды	Команда	id	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			

Рис. 2.11: Трасса работы оптимизированной программы

Благодаря оптимизации программы получилось избавиться от конфлик-  
тов.

# Заключение

В ходе лабораторной работы были изучены принципы функционирования и построения, а также особенности архитектуры суперскалярных конвейерных микропроцессоров на примере микропроцессорного ядра Taiga, реализующего систему команд семейства RISC-V. Таким образом, цель данной работы была достигнута.