



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»

## Отчёт по лабораторной работе №2 по курсу «Архитектура ЭВМ»

Тема Изучение принципов работы микропроцессорного ядра RISC-V

Студент Карпова Е.О.

Группа ИУ7-52Б

Преподаватель Дубровин Е.Н.

# 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.

# 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.

## 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.

## 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.

# Общая программа для всех вариантов

## Исследуемая программа

Код программы представлен в листингах 1 – 2.

Листинг 1 — Код программы для всех вариантов

```
.section .text
.globl _start;
len = 8
enroll = 4
elem_sz = 4

_start:
    addi x20, x0, len/enroll
    la x1, _x

loop:
    lw x2, 0(x1)
    add x31, x31, x2
    lw x2, 4(x1)
    add x31, x31, x2
    lw x2, 8(x1)
    add x31, x31, x2
    lw x2, 12(x1)
    add x31, x31, x2
    addi x1, x1, elem_sz*enroll
    addi x20, x20, -1
    bne x20, x0, loop
    addi x31, x31, 1

forever: j forever

.section .data
_x:    .4byte 0x1
      .4byte 0x2
      .4byte 0x3
```

Листинг 2 — Код программы для всех вариантов (продолжение)

```
.4byte 0x4  
.4byte 0x5  
.4byte 0x6  
.4byte 0x7  
.4byte 0x8
```

Дизассемблерный код представлен на листинге 3.

Листинг 3 — Дизассемблированный код общей программы

Disassembly of section .text:

```
80000000 <_start>:  
80000000: 00200a13 addi    x20,x0,2  
80000004: 00000097 auipc   x1,0x0  
80000008: 03c08093 addi    x1,x1,60 # 80000040 <x>  
  
8000000c <lp>:  
8000000c: 0000a103 lw      x2,0(x1)  
80000010: 002f8fb3 add    x31,x31,x2  
80000014: 0040a183 lw      x3,4(x1)  
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: fffa0a13 addi   x20,x20,-1  
80000034: fc0a1ce3 bne   x20,x0,8000000c <lp>  
80000038: 001f8f93 addi   x31,x31,1  
  
8000003c <lp2>:  
8000003c: 0000006f jal    x0,8000003c <lp2>
```

Можно сказать, что данная программа эквивалентна следующему псевдокоду на языке С, представленному на листинге 4.

Листинг 4 — Псевдокод общей программы

```
#define len 8
#define enroll 4
#define elem_sz 4
int _x[]={1,2,3,4,5,6,7,8};
void _start() {
    int x20 = len/enroll;
    int *x1 = _x;

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

Адрес	Код команды	Команда	id	Номер такта		
				1	2	3
0x00000000<start>	0x2000a13	addi x20,x0,2	0	F	ID D AL	
0x00000004	0x0000097	auipc x1,0x0	1	F	ID D AL	
0x00000008	0x3c06993	addi x1,1,60<#00000040->	2	F	ID D AL	
0x0000000c<loop>	0x0000a103	lw x2,0(x1)	3	F	ID D M1M2M3	
0x00000010	0x2f1fb3	add x31,x31,x2	4	F	ID D C CAL	
0x00000014	0x040a103	lw x2,4(x1)	5	F	ID W W D M1M2M3	
0x00000018	0x2f1fb3	add x31,x31,x2	6	F	ID W W D C CAL	
0x0000001c	0x080a103	lw x2,8(x1)	7	F	ID W W W W D M1M2M3	
0x00000020	0x2f1fb3	add x31,x31,x2	8	F	ID W W W W D C CAL	
0x00000024	0x000a103	lw x2,12(x1)	1	F	ID W W W W W D M1M2M3	
0x00000028	0x2f1fb3	add x31,x31,x2	2	F	ID W W W W W D C CAL	
0x0000002c	0x0000093	addi x1,1,16	3	F	ID W W W W W D C CAL	
0x00000030	0xffffa013	addi x20,x20,-1	4	F	ID W W W W W W D AL	
0x00000034	fc0alc3	lne x20,x0,80000000<loop>	5	F	ID W W W W W W D B	
0x00000038	0x01f8f93	addi x1,x31,1	6	F	ID W W W W W W D X	
0x0000003c	0x000006f	jal x0,8000003c<forever>	7	F	ID W W W W W W DX	
0x00000040	1	<invalid operation>	8	F	ID W W W X	
0x00000044	0x0000002	<invalid operation>	1	F	ID W X	
0x00000048	0x0000003	<invalid operation>	2	F	ID X	
0x0000004c	0x0000004	<invalid operation>	3	F	X	
0x00000050	0x0000006	<invalid operation>	4	FX		
0x00000060<loop>	0x0000a103	lw x2,0(x1)	7	F	ID D M1M2M3	
0x00000064	0x2f1fb3	add x31,x31,x2	0	F	ID D C CAL	
0x00000068	0x040a103	lw x2,4(x1)	1	F	ID W W D M1M2M3	
0x00000072	0x2f1fb3	add x31,x31,x2	2	F	ID W W D C CAL	
0x00000076	0x080a103	lw x2,8(x1)	3	F	ID W W W W D M1M2M3	
0x00000080	0x2f1fb3	add x31,x31,x2	4	F	ID W W W W W D C CAL	
0x00000084	0x000a103	lw x2,12(x1)	5	F	ID W W W W W D M1M2M3	
0x00000088	0x2f1fb3	add x31,x31,x2	6	F	ID W W W W W D C CAL	
0x00000092	0x0000093	addi x1,x1,16	7	F	ID W W W W W W D AL	
0x00000096	0xffffa013	addi x20,x20,-1	0	F	ID W W W W W W D B	
0x0000009c<loop>	0x0000a103	lw x2,0(x1)	2	F	ID W W W W W W D X	
0x000000a0	0x2f1fb3	add x31,x31,x2	3	F	ID W W W W W W DX	
0x000000a4	0x040a103	lw x2,4(x1)	4	F	ID W W W X	
0x000000a8	0x2f1fb3	add x31,x31,x2	5	F	ID W W X	
0x000000bc	0x000a103	lw x2,8(x1)	6	F	ID X	
0x000000c0	0x2f1fb3	add x31,x31,x2	7	F	X	
0x000000c4	0x000a103	lw x2,12(x1)	8	FX		
0x000000c8	0x01f8f93	addi x1,x31,1	3	F	ID D AL	
0x000000cc	0x000006f	jal x0,8000003c<forever>	4	F	ID D B	
0x000000d0	0x0000001	<invalid operation>	5	F	ID D X	
0x000000d4	0x0000006	<invalid operation>	6	F	ID DX	
0x000000d8	0x0000003	<invalid operation>	7	F	X	
0x000000dc	0x0000006	jal x0,8000003c<forever>	8	FX		
0x000000e0	0x0000006	jal x0,8000003c<forever>	0	F	ID D B	
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
				43	44	45
				46	47	48
				49	49	50
				51	52	53
				54	55	56
				57	58	59
				60		

Рисунок 1 – Трасса выполнения программы

# Результаты исследования программы

Задания выполнялись по варианту 6.

## Задание №2

Получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки и диспетчеризации команды с адресом 80000020 на первой итерации.

Результат представлен на рисунке 2

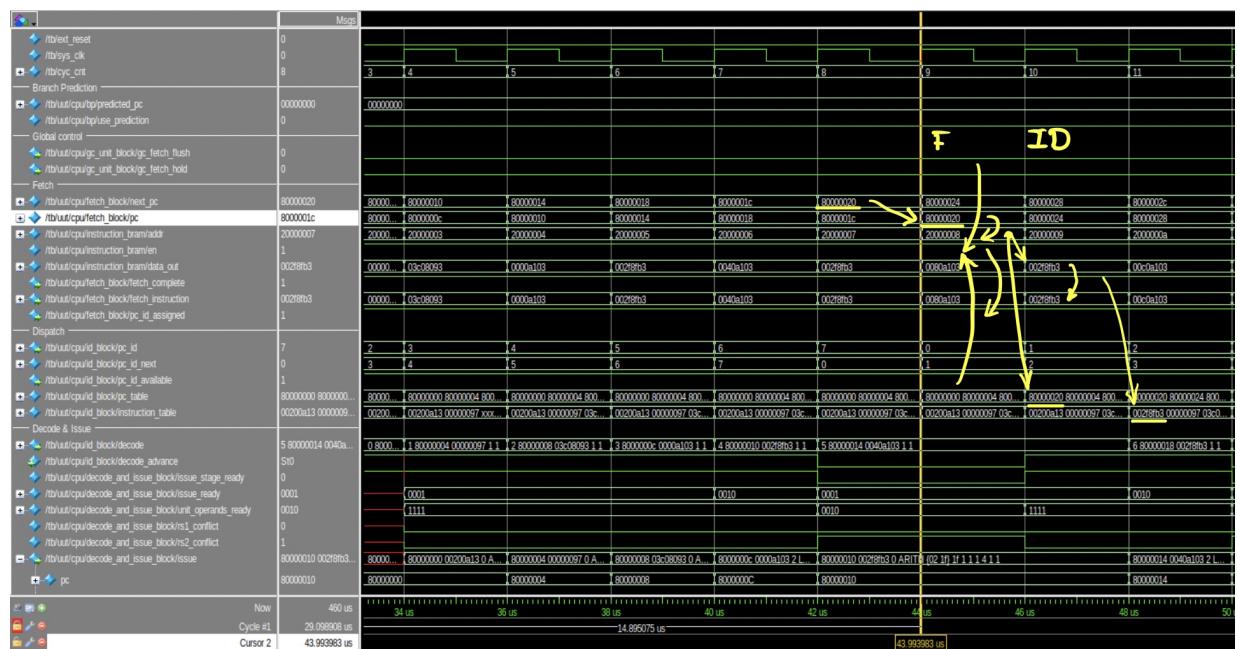


Рисунок 2 – Задание №2

## Задание №3

Получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с адресом 80000002c на первой итерации.

Результат представлен на рисунке 3

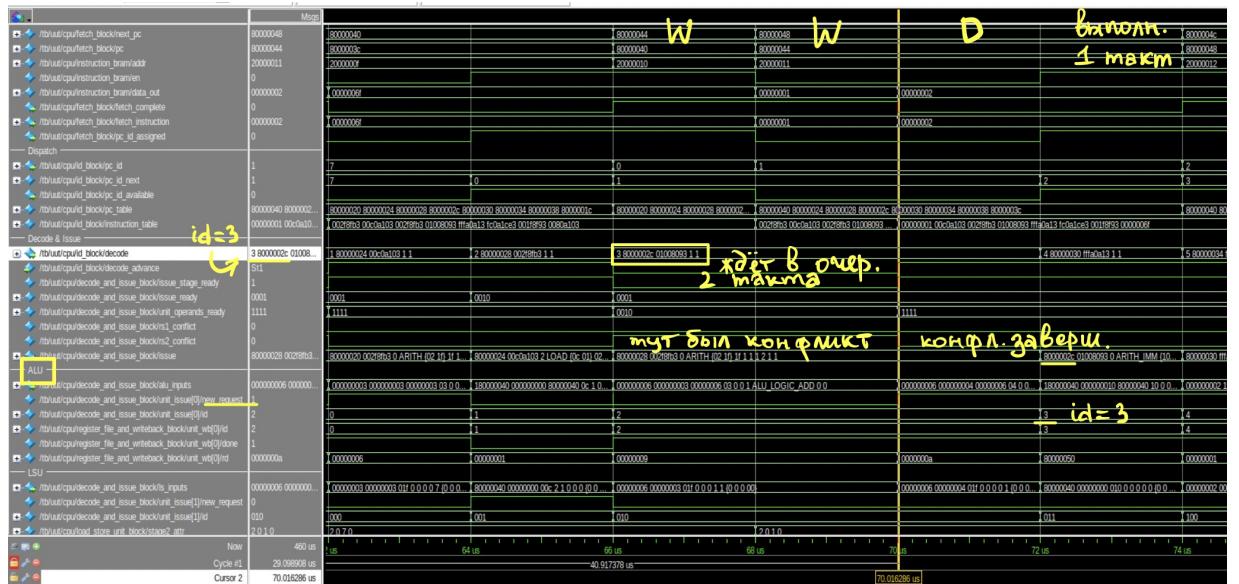


Рисунок 3 – Задание №3

## Задание №4

Получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с адресом 80000014 на первой итерации.

Результат представлен на рисунке 4

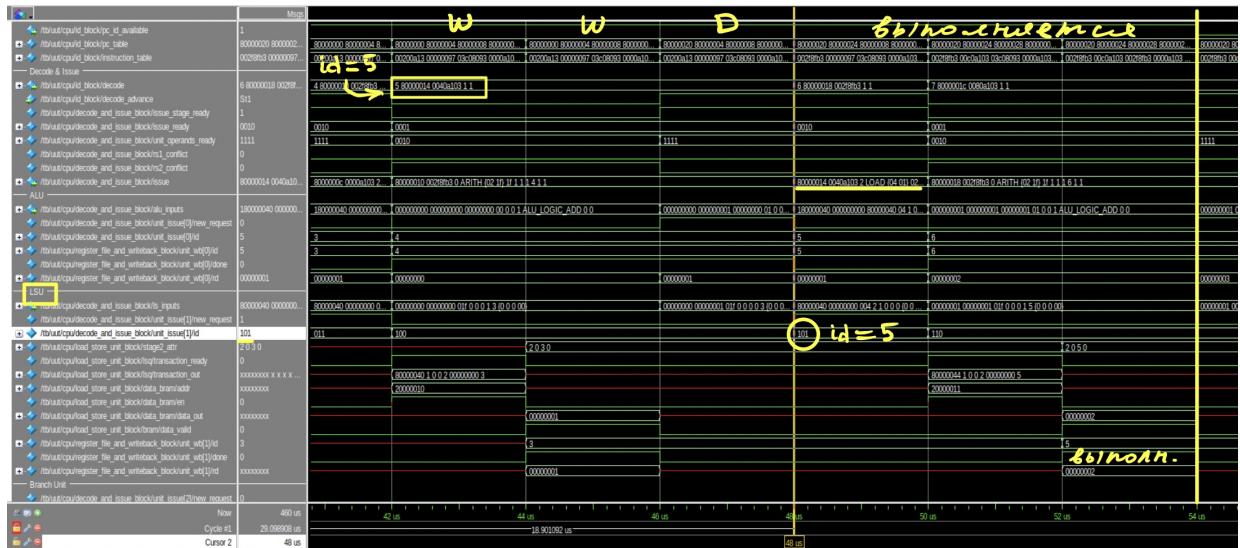


Рисунок 4 – Задание №4

# Программа по варианту

## Исследуемая программа

Код программы представлен в листинте 5

Листинг 5 — Код программы 6 варианта

```
.section .text
.globl _start;
len = 8 #Размер массива
enroll = 2 #Количество обрабатываемых элементов
за одну итерацию
elem_sz = 4 #Размер одного элемента массива

_start:
    addi x20, x0, len/enroll
    la x1, _x
    add x31, x0, x0
lp:
    lw x2, 0(x1)
    lw x3, 4(x1) # !
    addi x1, x1, elem_sz*enroll
    addi x20, x20, -1
    add x31, x31, x2
    add x31, x31, x3
    bne x20, x0, lp
    addi x31, x31, 1
lp2: j lp2

.section .data
_x:
    .4byte 0x1
    .4byte 0x2
    .4byte 0x3
    .4byte 0x4
```

Листинг 6 — Код программы 6 варианта (продолжение)

```
.4byte 0x5  
.4byte 0x6  
.4byte 0x7  
.4byte 0x8
```

Дизассемблерный код представлен на листинге 7.

Листинг 7 — Дизассемблированный код 6 варианта

```
Disassembly of section .text:  
80000000 <_start>:  
80000000: 00400a13 addi    x20,x0,4  
80000004: 00000097 auipc  x1,0x0  
80000008: 03008093 addi    x1,x1,48 # 80000034 <_x>  
8000000c: 00000fb3 add     x31,x0,x0  
80000010 <lp>:  
  
80000010: 0000a103 lw      x2,0(x1)  
80000014: 0040a183 lw      x3,4(x1)  
80000018: 00808093 addi    x1,x1,8  
8000001c: fffa0a13 addi    x20,x20,-1  
80000020: 002f8fb3 add     x31,x31,x2  
80000024: 003f8fb3 add     x31,x31,x3  
80000028: fe0a14e3 bne   x20,x0,80000010 <lp>  
8000002c: 001f8f93 addi    x31,x31,1  
80000030 <lp2>:  
  
80000030: 0000006f jal   x0,80000030 <lp2>
```

Листинг 8 — Дизассемблированный код 6 варианта (продолжение)

```
Disassembly of section .data:  
80000034 <_x>:  
80000034: 0001    c.addi   x0,0  
80000036: 0000    c.unimp  
80000038: 0002    c.slli64      x0  
8000003a: 0000    c.unimp  
8000003c: 00000003      lb      x0,0(x0) # 0 <enroll-0x2>  
80000040: 0004    0x4  
80000042: 0000    c.unimp  
80000044: 0005    c.addi   x0,1  
80000046: 0000    c.unimp  
80000048: 0006    c.slli   x0,0x1  
8000004a: 0000    c.unimp  
8000004c: 00000007      0x7  
80000050: 0008    0x8  
...  
.
```

Можно сказать, что данная программа эквивалентна следующему псевдокоду на языке С, представленному на листинге 9.

Листинг 9 — Псевдокод программы 6 варианта

```
// суммирует все элементы массива и прибавляет 1
len = 8; // длина массива = 8
enroll = 2; // сколько элементов обрабатаем за итерацию
elem_sz = 4; // размер элемента

x[len] = {1,2,3,4,5,6,7,8} // массив

x20 = len / enroll; // количество итераций
x1 = &x; // адрес массива
x31 = 0;

do {
    x2 = x1[0]; // 0 элемент
    x3 = x1[1]; // 4 элемент

    x1 += elem_sz * enroll; // сдвиг по массиву

    x20 -= 1 // уменьшение количества итераций
    x31 += x2
    x31 += x3
} while (x20 != 0);

x31 += 1 // Результат = 37

while (1) {}; // бесконечный цикл
```

## Трасса работы программы

Трасса работы представлена на рисунке 5.

Рисунок 5 – Трасса выполнения программы

## Результат работы программы

Результат работы программы представлен на рисунке 6.



Рисунок 6 – Результат работы программы

## Временные диаграммы

Временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом #! (lw x3, 4(x1)) представлены на рисунках 7 — 9.

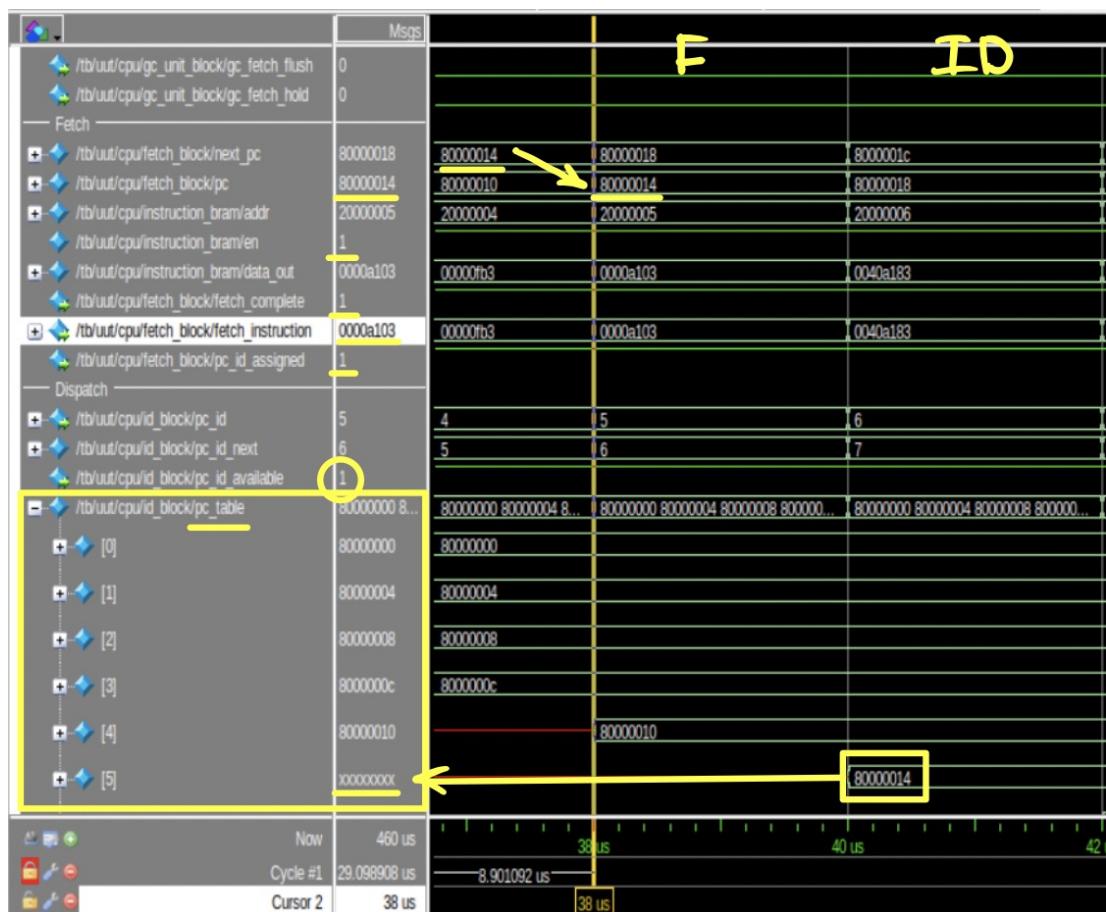


Рисунок 7 – Временные диаграммы сигналов - F & ID

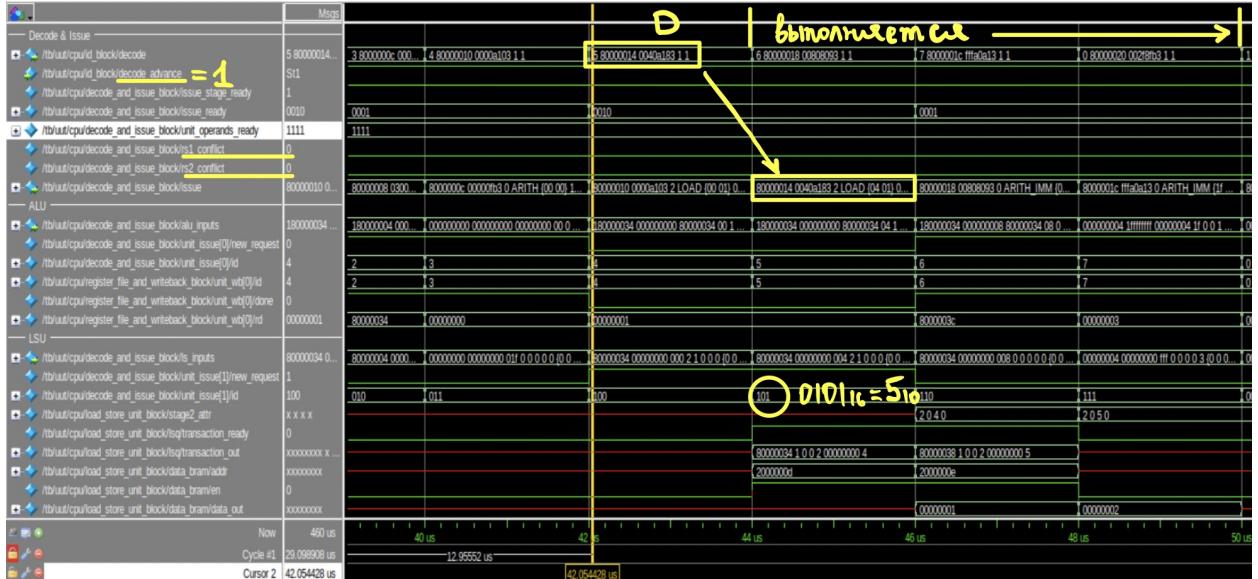


Рисунок 8 – Временные диаграммы сигналов - D & M

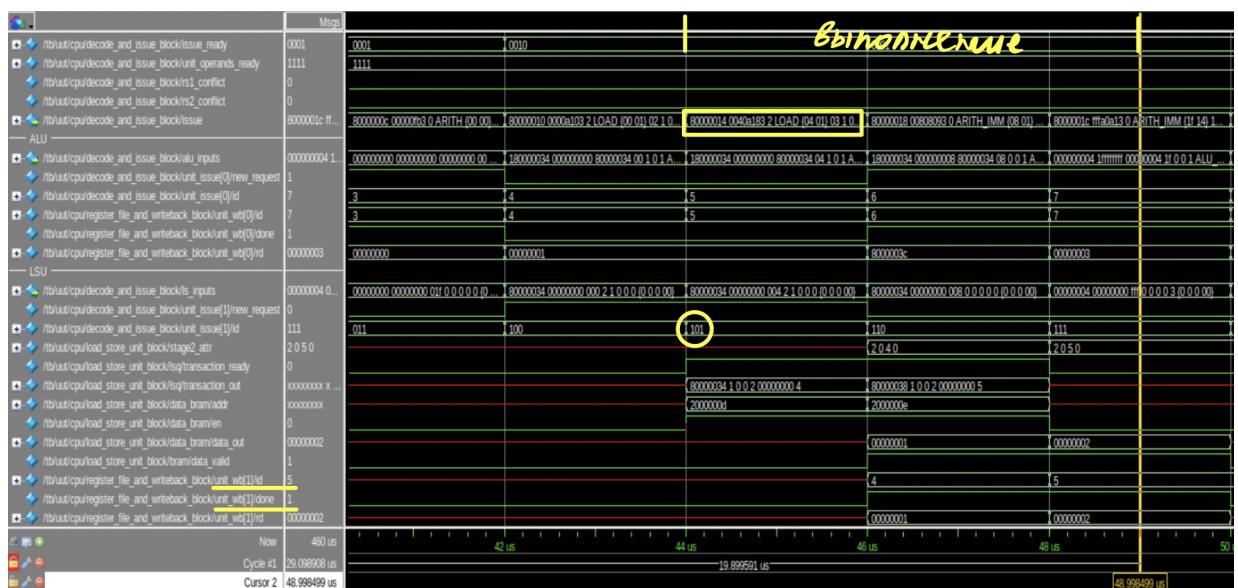


Рисунок 9 – Временные диаграммы сигналов - M

## Вывод и предложение по оптимизации

Как видно на трассе работы программы, задержек и простоев не возникает, следовательно нет более оптимального варианта для данной программы.

# **Вывод**

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.