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»(ИУ7)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.04 «Программная инженерия»

О Т Ч Е Т
по лабораторной работе № 2

Название Изучение принципов работы микропроцессорного ядра RISC-V

Дисциплина Архитектура элекронно-вычислительных машин

Студент: _____ Золотухин А В.
подпись, дата Фамилия, И.О.

Преподаватель: _____ Попов А. Ю.
подпись, дата Фамилия, И. О.

Москва — 2022 г.

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.

В данной работе исследуется набор команд RV32I, который включает в себя основные команды 32-битной целочисленной арифметики кроме умножения и деления.

Набор команд RV32I предполагает использование 32 регистров общего назначения x0-x31 размером в 32 бита каждый и регистр pc, хранящего адрес следующей команды. Все регистры общего назначения равноправны, в любой команде могут использоваться любые из регистров. Регистр pc не может использоваться в командах.

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.

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 (кроме команды auipc) 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 для всех вариантов программа

Исследуемая программа

Исходный текст исследуемой программы представлен на рисунке 1.

```
.section .text (1)
.globl _start; (2)
len = 8 # size of array (3)
enroll = 4 #amount of processed elements in one iteration
elem_sz = 4 #size of one element in array
_start: (4)
addi x20, x0, len/enroll (5)
la x1, _x (6)
loop:
lw x2, 0(x1) (7)
add x31, x31, x2 (8)
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 (9)
addi x20, x20, -1 (10)
bne x20, x0, loop (11)
addi x31, x31, 1
forever: j forever (12)

.section .data (13)
_x:     .4byte 0x1 (14)
.4byte 0x2
.4byte 0x3
.4byte 0x4
.4byte 0x5
.4byte 0x6
.4byte 0x7
.4byte 0x8
```

Рисунок 1 – Исходный код программы

Дизассемблерный листинг исследуемой программы представлен на рисунке 2

```
80000000 <_start>:
80000000:    00200a13            addi    x20,x0,2
80000004:    00000097            auipc   x1,0x0 (1)
80000008:    03c08093            addi    x1,x1,60 # 80000040 <_x>

8000000c <loop>:
8000000c:    0000a103            lw  x2,0(x1)
80000010:    002f8fb3            add x31,x31,x2
80000014:    0040a103            lw  x2,4(x1)
80000018:    002f8fb3            add x31,x31,x2
8000001c:    0080a103            lw  x2,8(x1)
80000020:    002f8fb3            add x31,x31,x2
80000024:    00c0a103            lw  x2,12(x1)
80000028:    002f8fb3            add x31,x31,x2
8000002c:    01008093            addi    x1,x1,16
80000030:    fffa0a13            addi    x20,x20,-1
80000034:    fc0a1ce3            bne x20,x0,8000000c <loop>
80000038:    001f8f93            addi    x31,x31,1

8000003c <forever>:
8000003c:    0000006f            jal x0,8000003c <forever>
```

Рисунок 2 – Дизассемблированный код программы

Можно сказать, что данная программа эквивалентна псевдокоду на языке C, представленному на рисунке 3.

```
#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){}
}
```

Рисунок 3 – Псевдокод программы

Результаты исследования программы

Все задания выполнялись по индивидуальному варианту (5).

Скриншот, полученный в ходе выполнения задания №2 (получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки и диспетчеризации команды с адресом 8000001c на первой итерации) представлен на рисунке 4.

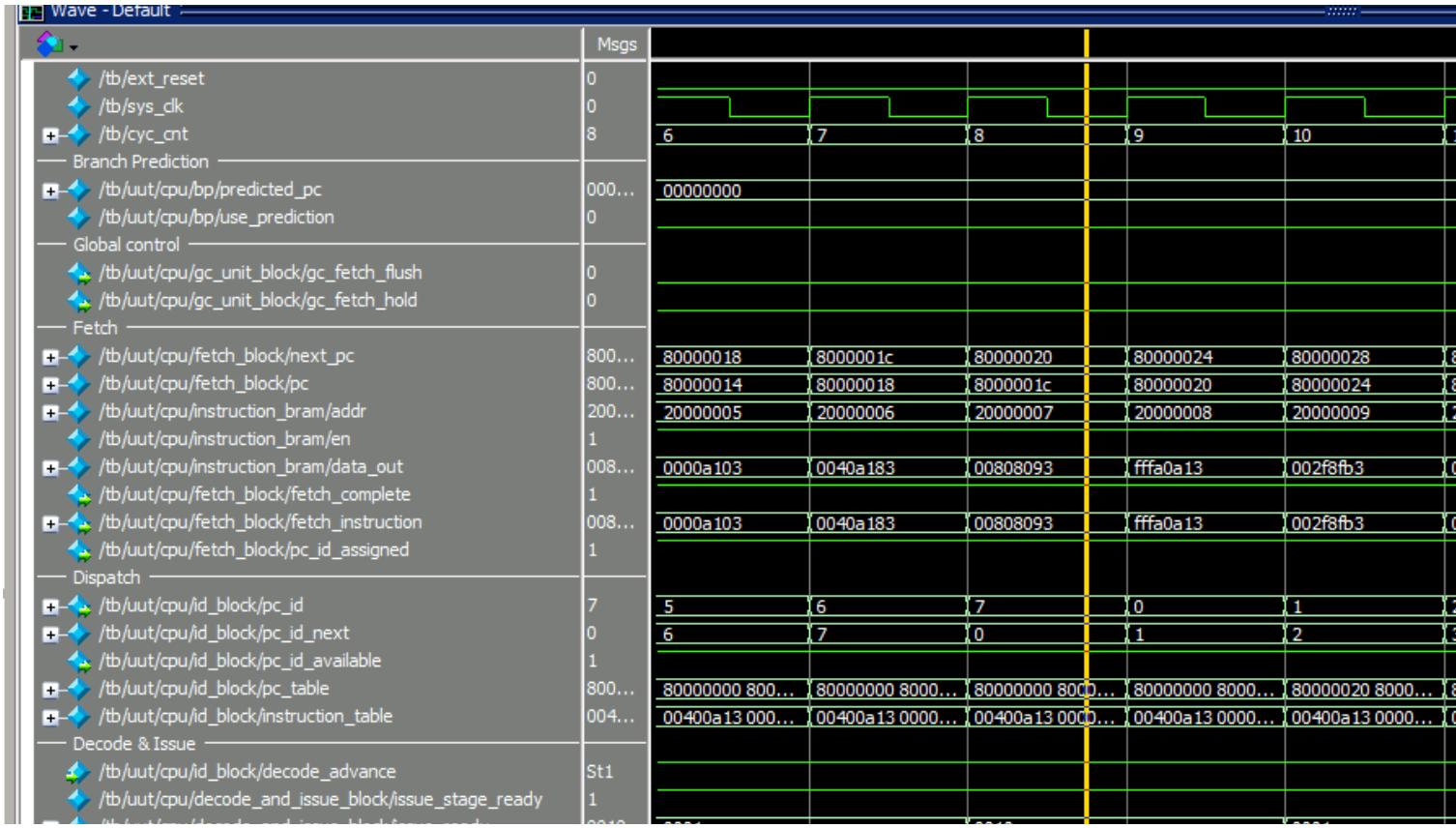


Рисунок 4 – Временная диаграмма выполнения стадий выборки и диспетчеризации

Скриншот, полученный в ходе выполнения задания №3 (получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с адресом 80000028 на первой итерации) представлен на рисунке 5.

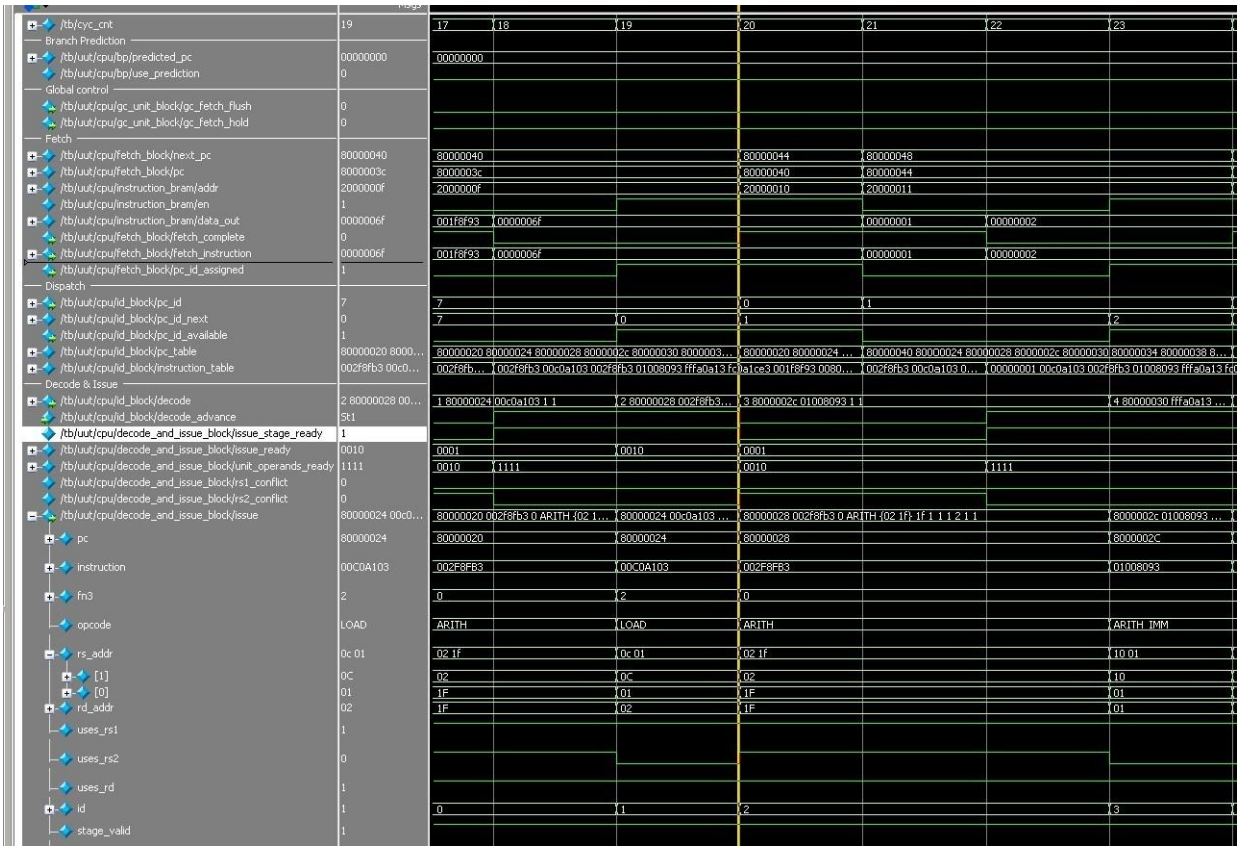


Рисунок 5 – Временная диаграмма выполнения стадии декодирования и планирования на выполнение

Скриншот, полученный в ходе выполнения задания №4 (получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с адресом 80000010 на первой итерации) представлен на рисунке 6.

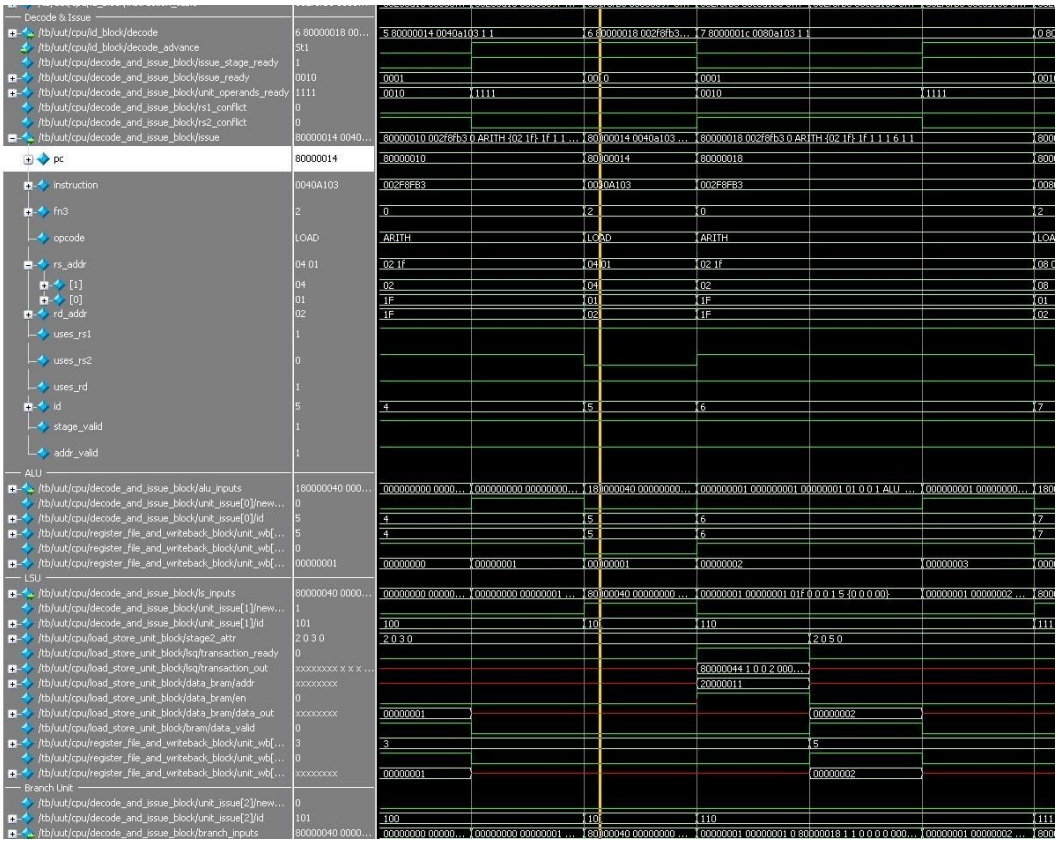


Рисунок 6 – Временная диаграмма выполнения стадии выполнения

Программа по варианту

Все задания выполнялись по индивидуальному варианту (5).

Исследуемая программа

Исходный текст исследуемой программы представлен на рисунке 7.

Вариант 5

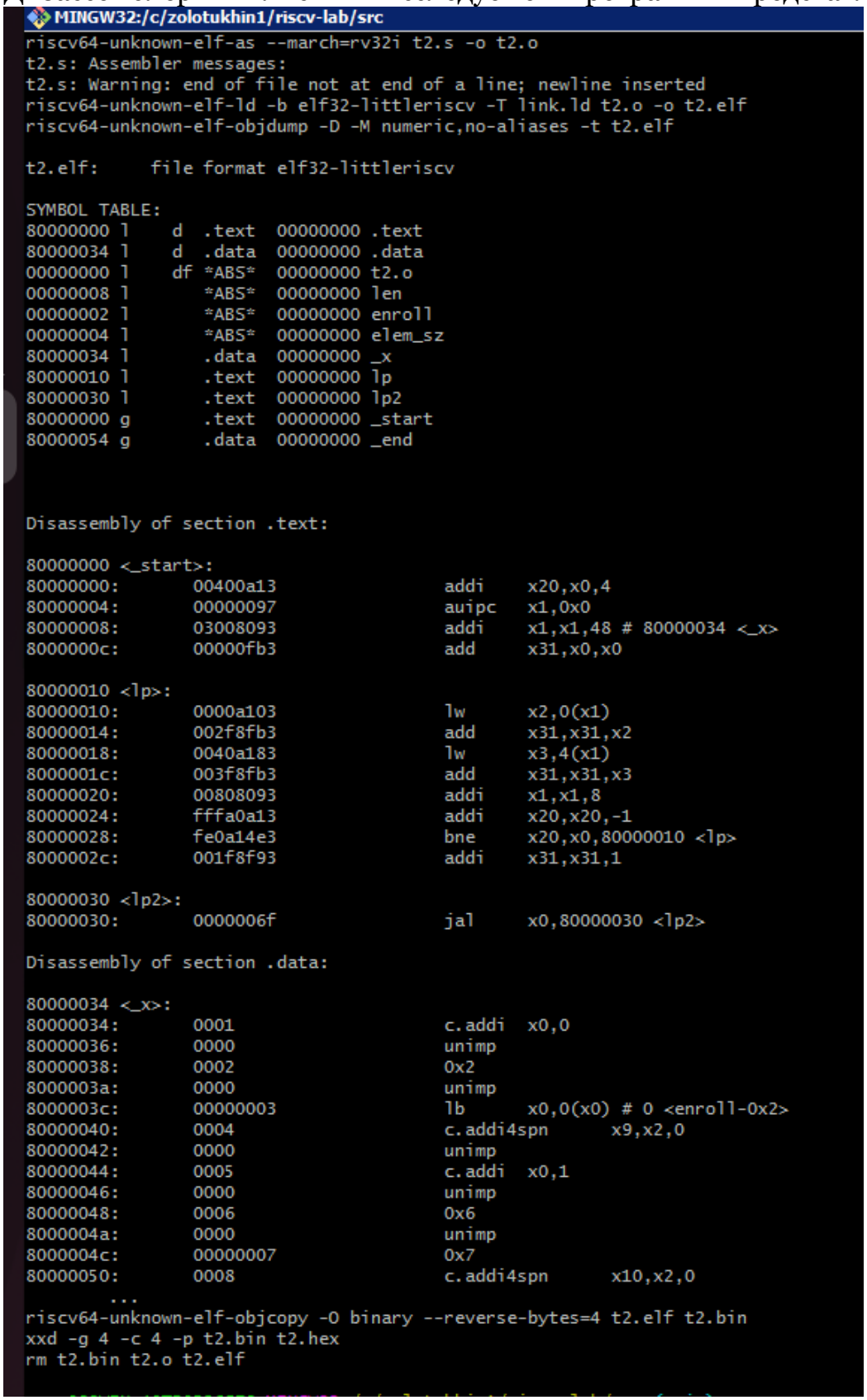
```
        .section .text
        .globl _start;
        len = 8 #Размер массива
        enroll = 2 #Количество обрабатываемых элементов за одну итерацию
        elem_sz = 4 #Размер одного элемента массива

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

Рисунок 7 – Исходный текст исследуемой программы

Дизассемблерный листинг исследуемой программы представлен на рисунке 8.



```
MINGW32:/c/zolotukhin1/riscv-lab/src
riscv64-unknown-elf-as --march=rv32i t2.s -o t2.o
t2.s: Assembler messages:
t2.s: Warning: end of file not at end of a line; newline inserted
riscv64-unknown-elf-ld -b elf32-littleriscv -T link.ld t2.o -o t2.elf
riscv64-unknown-elf-objdump -D -M numeric,no-aliases -t t2.elf

t2.elf:     file format elf32-littleriscv

SYMBOL TABLE:
80000000 l    d  .text  00000000 .text
80000034 l    d  .data  00000000 .data
00000000 l    df *ABS*  00000000 t2.o
00000008 l       *ABS*  00000000 len
00000002 l       *ABS*  00000000 enroll
00000004 l       *ABS*  00000000 elem_sz
80000034 l       .data  00000000 _x
80000010 l       .text  00000000 lp
80000030 l       .text  00000000 lp2
80000000 g       .text  00000000 _start
80000054 g       .data  00000000 _end



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:       002f8fb3                add     x31,x31,x2
80000018:       0040a183                lw      x3,4(x1)
8000001c:       003f8fb3                add     x31,x31,x3
80000020:       00808093                addi    x1,x1,8
80000024:       fffa0a13                addi    x20,x20,-1
80000028:       fe0a14e3                bne     x20,x0,80000010 <lp>
8000002c:       001f8f93                addi    x31,x31,1

80000030 <lp2>:
80000030:       0000006f                jal     x0,80000030 <lp2>

Disassembly of section .data:

80000034 <_x>:
80000034:       0001                    c.addi  x0,0
80000036:       0000                    unimp
80000038:       0002                    0x2
8000003a:       0000                    unimp
8000003c:       00000003                lb      x0,0(x0) # 0 <enroll-0x2>
80000040:       0004                    c.addi4spn      x9,x2,0
80000042:       0000                    unimp
80000044:       0005                    c.addi  x0,1
80000046:       0000                    unimp
80000048:       0006                    0x6
8000004a:       0000                    unimp
8000004c:       00000007                0x7
80000050:       0008                    c.addi4spn      x10,x2,0
        ...
riscv64-unknown-elf-objcopy -O binary --reverse-bytes=4 t2.elf t2.bin
xxd -g 4 -c 4 -p t2.bin t2.hex
rm t2.bin t2.o t2.elf
```

Рисунок 8 – Дизассемблерный листинг исследуемой программы

Трасса работы программы

Трасса работы программы представлена на рисунке 9.

				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	62	63	64	65	66	67
80000000 <_start>:	00400a13	addi x20,x0,4	0	F	ID	D	AL																																																												
80000004:	00000097	auipc x1,0x0	1		F	ID	D	AL																																																										
80000008:	03008093	addi x1,x1,48 # 80000034 <_x>	2			F	ID	D	AL																																																								
8000000c:	00000fb3	add x31,x0,x0	3				F	ID	D	AL																																																								
80000010 <lp>:	0000a103	lw x2,0(x1)	4					F	ID	D	M1	M2	M3																																																					
80000014:	002f8fb3	add x31,x31,x2	5						F	ID	D	C	C	AL																																																		
80000018:	0040a183	lw x3,4(x1)	6							F	ID	W	W	D	M1	M2	M3																																																
8000001c:	003f8fb3	add x31,x31,x3	7								F	ID	W	W	D	C	C	AL																																														
80000020:	808093	addi x1,x1,8	0									F	ID	W	W	W	W	D	AL																																													
80000024:	fffa0a13	addi x20,x20,-1	1										F	ID	W	W	W	W	D	AL																																											
80000028:	fe0a14e3	bne x20,x0,80000010 <lp>	2											F	ID	W	W	W	W	D	B																																										
8000002c:	001f8f93	addi x31,x31,1	3												F	ID	W	W	W	W	D	X																																									
80000030 <lp2>:	0000006f	jal x0,80000030 <lp2>	4													F	ID	W	W	W	W	DX																																										
80000034 <_x>:	00000001	<invalid command>	5														F	ID	W	W	W	X																																										
80000038:	00000002	<invalid command>	6																F	ID	W	X																																										
8000003c:	00000003	<invalid command>	7																	F	ID	X																																										
80000040:	00000004	<invalid command>	0																		F	X																																										
80000044:	00000005	<invalid command>	4																			FX																																										
80000010 <lp>:	0000a103	lw x2,0(x1)	5																				F	ID	D	M1	M2	M3																																						
80000014:	002f8fb3	add x31,x31,x2	6																					F	ID	D	C	C	AL																																					
80000018:	0040a183	lw x3,4(x1)	7																						F	ID	W	W	D	M1	M2	M3																																				
8000001c:	003f8fb3	add x31,x31,x3	0																							F	ID	W	W	D	C	C	AL																																			
80000020:	808093	addi x1,x1,8	1																								F	ID	W	W	W	W	D	AL																																		
80000024:	fffa0a13	addi x20,x20,-1	2																									F	ID	W	W	W	W	D	AL																																	
80000028:	fe0a14e3	bne x20,x0,80000010 <lp>	3																										F	ID	W	W	W	W	D	B																																
80000010 <lp>:	0000a103	lw x2,0(x1)	4																											F	ID	W	W	W	W	D	M1	M2	M3																													
80000014:	002f8fb3	add x31,x31,x2	5																												F	ID	W	W	W	W	D	C	C	AL																												
80000018:	0040a183	lw x3,4(x1)	6																													F	ID	W	W	W	W	W	W	D	M1	M2	M3																										
8000001c:	003f8fb3	add x31,x31,x3	7																															F	ID	W	W	W	W	W	D	C	C	AL																									
80000020:	808093	addi x1,x1,8	0																																F	ID	W	W	W	W	W	W	W	D	AL																								
80000024:	fffa0a13	addi x20,x20,-1	1																																	F	ID	W	W	W	W	W	W	W	D	AL																							
80000028:	fe0a14e3	bne x20,x0,80000010 <lp>	2																																		F	ID	W	W	W	W	W	W	W	D	B																						
80000010 <lp>:	0000a103	lw x2,0(x1)	3																																				F	ID	W	W	W	W	W	W	D	M1	M2	M3																				
80000014:	002f8fb3	add x31,x31,x2	4																																						F	ID	W	W	W	W	W	D	C	C	AL																			
80000018:	0040a183	lw x3,4(x1)	5																																							F	ID	W	W	W	W	W	W	W	D	M1	M2	M3																
8000001c:	003f8fb3	add x31,x31,x3	6																																										F	ID	W	W	W	W	W	D	C	C	AL															
80000020:	808093	addi x1,x1,8	7																																											F	ID	W	W	W	W	W	W	W	D	AL														
80000024:	fffa0a13	addi x20,x20,-1	0																																												F	ID	W	W	W	W	W	W	W	D	AL													
80000028:	fe0a14e3	bne x20,x0,80000010 <lp>	1																																													F	ID	W	W	W	W	W	W	W	D	B												
80000010 <lp>:	0000a103	lw x2,0(x1)	2																																															F	ID	W	W	W	W	W	W	D	X											
80000014:	002f8fb3	add x31,x31,x2	3																																																	F	ID	W	W	W	W	W	DX											
80000018:	0040a183	lw x3,4(x1)	4																																																		F	ID	W	W	W	W	X											
8000001c:	003f8fb3	add x31,x31,x3	5																																																					F	ID	W	X											
80000020:	808093	addi x1,x1,8	6																																																						F	ID	X											
80000024:	fffa0a13	addi x20,x20,-1	7																																																							F	X											
80000028:	fe0a14e3	bne x20,x0,80000010 <lp>	2																																																								FX											
8000002c:	001f8f93	addi x31,x31,1	3																																																									F	ID	D	AL							
80000030 <lp2>:	0000006f	jal x0,80000030 <lp2>	4																																																										F	ID	D	B						
80000034 <_x>:	00000001	<invalid command>	5																																																											F	ID	D	X					
80000038:	00000002	<invalid command>	6																																																												F	ID	DX					
8000003c:	00000003	<invalid command>	7																																																													F	X					
80000040:	00000004	<invalid command>	5																																																														FX					
80000030 <lp2>:	0000006f	jal x0,80000030 <lp2>	6																																																													F	ID	D		
				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	62	63	64	65	66	67

Рисунок 9 – Трасса работы программы

Временные диаграммы

Временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом #! (add x31, x31, x3), представлены на рисунках 10, 11, 12.

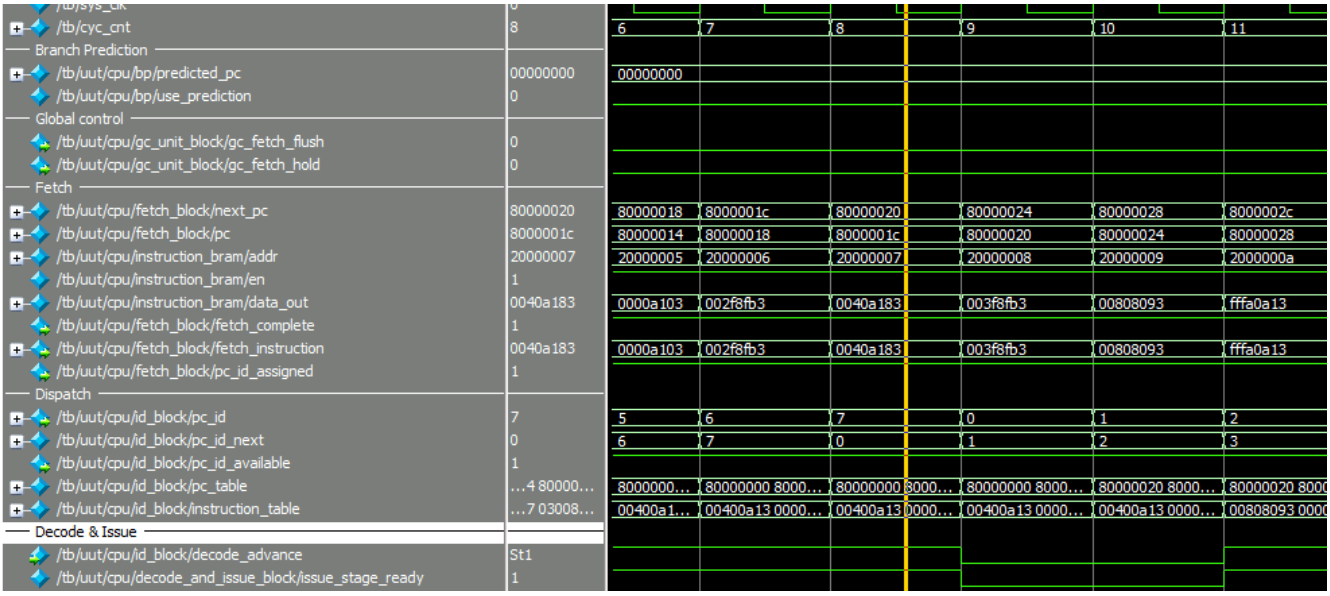


Рисунок 10 – Временная диаграмма стадии выборки и диспетчеризации

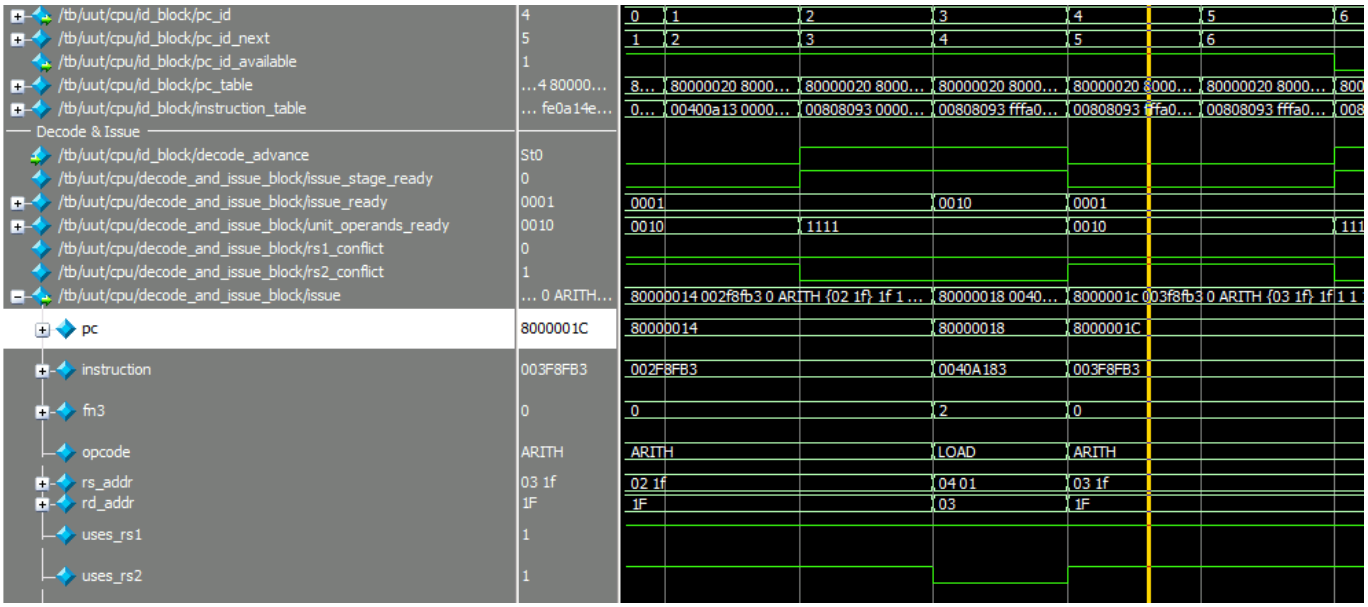


Рисунок 11 – Временная диаграмма стадии декодирования

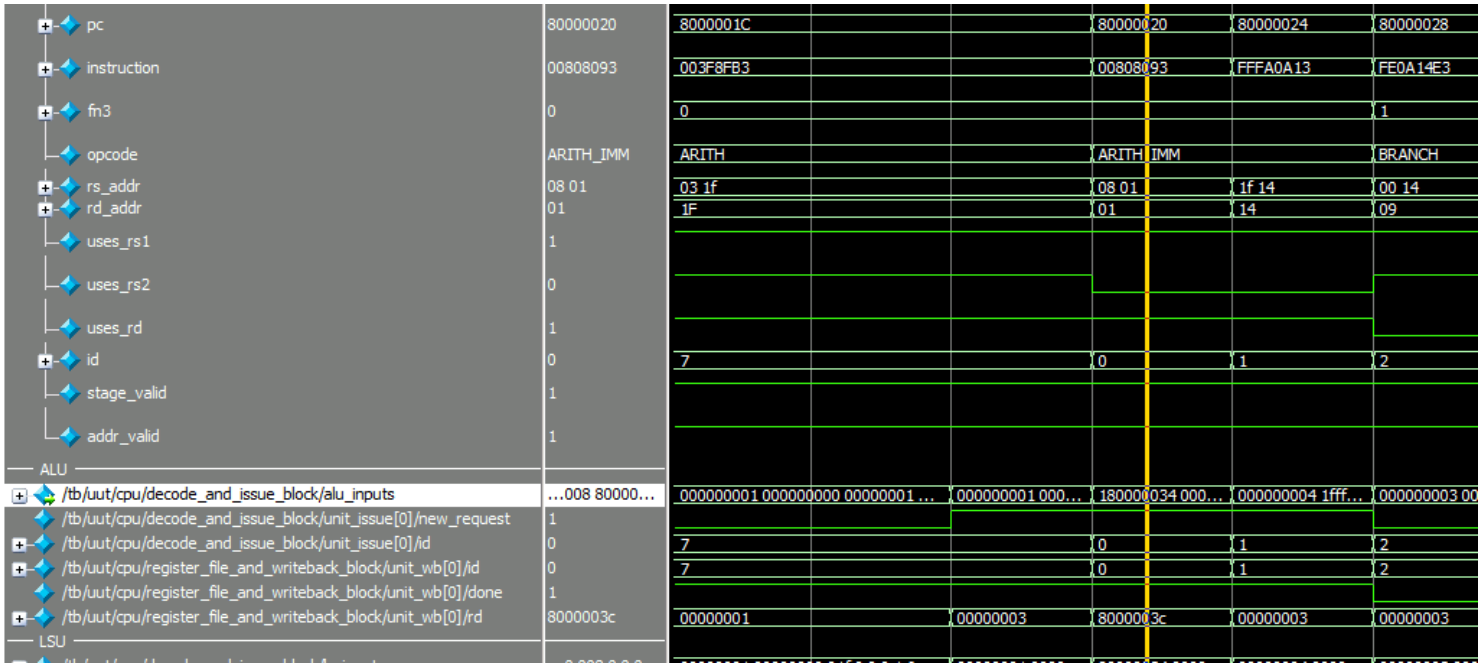


Рисунок 12 – Временная диаграмма стадии выполнения

Вывод об эффективности работы программы

Как видно на трассе работы программы, представленой на рисунке 9, конфликты по регистрам возникают.

Оптимизировать же программу можно, если сначала производить все команды загрузки, затем выполнять команды, обслуживающие цикл и в конце команды обрабатывающие данные.

Оптимизированная программа

Исходный текст оптимизированной программы представлен на рисунке 13.

```
        .section .text
        .globl _start;
        len = 8 #Размер массива
        enroll = 2 #Количество обрабатываемых элементов за одну и
    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
        .4byte 0x5
        .4byte 0x6
        .4byte 0x7
        .4byte 0x8
```

Рисунок 13 — Код оптимизированной программы

Дизассемблерный листинг оптимизированной программы представлен на рисунке 14.

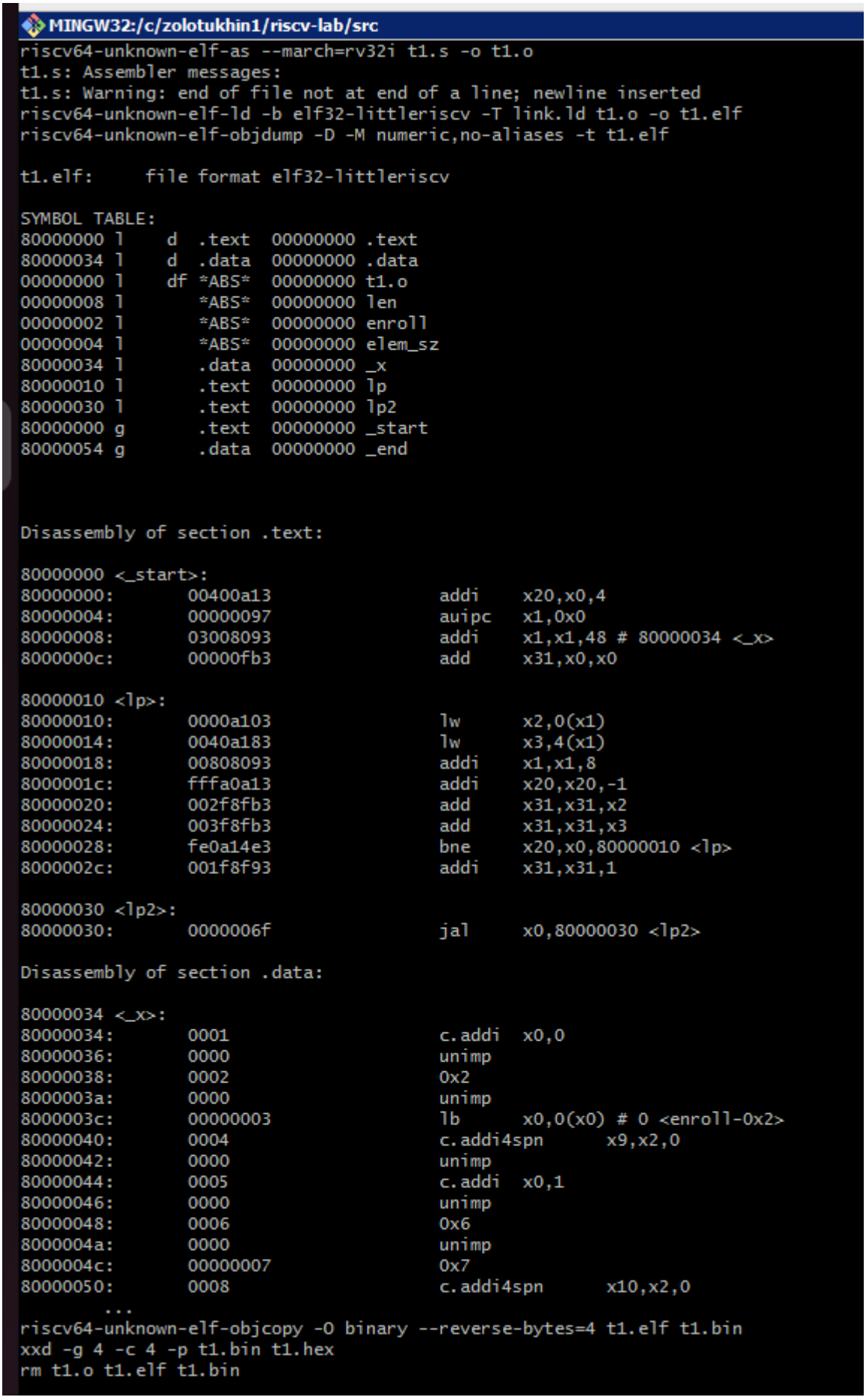
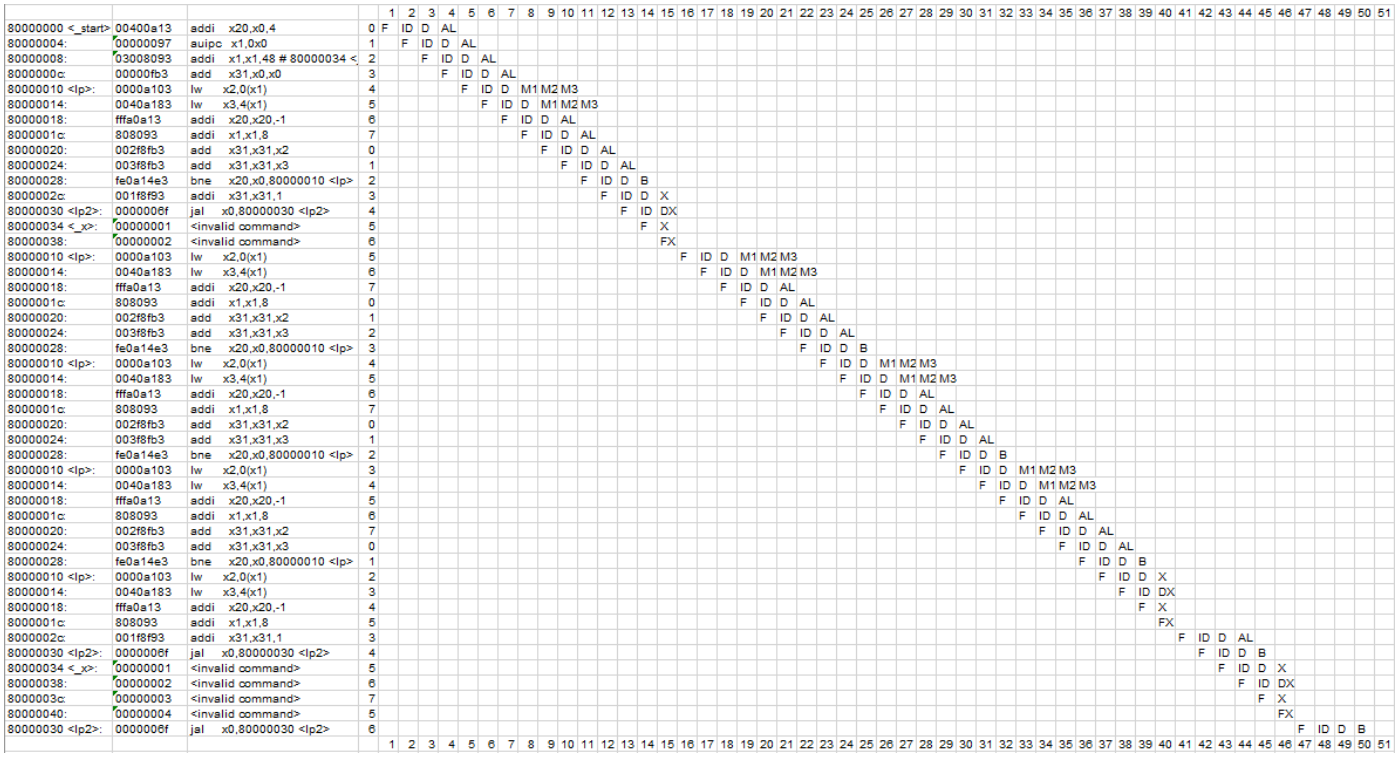


Рисунок 14 – Дизассемблированный листинг программы.

Трасса работы программы

Трасса работы оптимизированной программы представлена на рисунке 15.



				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>	00400a13	addi x20,x0,4	0	F	ID	D	AL																																														
80000004:	00000097	auipc x1,0x0	1		F	ID	D	AL																																												
80000008:	03008093	addi x1,x1,48 # 80000034 <_	2			F	ID	D	AL																																										
8000000c:	00000fb3	add x31,x0,x0	3				F	ID	D	AL																																									
80000010 <lp>:	0000a103	lw x2,0(x1)	4					F	ID	D	M1	M2	M3																																					
80000014:	0040a183	lw x3,4(x1)	5						F	ID	D	M1	M2	M3																																				
80000018:	fffa0a13	addi x20,x20,-1	6							F	ID	D	AL																																				
8000001c:	808093	addi x1,x1,8	7								F	ID	D	AL																																		
80000020:	002f8fb3	add x31,x31,x2	0									F	ID	D	AL																																
80000024:	003f8fb3	add x31,x31,x3	1										F	ID	D	AL																														
80000028:	fe0a14e3	bne x20,x0,80000010 <lp>	2											F	ID	D	B																														
8000002c:	001f8f93	addi x31,x31,1	3												F	ID	D	X																													
80000030 <lp2>:	0000006f	jal x0,80000030 <lp2>	4													F	ID	DX																													
80000034 <_x>:	00000001	<invalid command>	5														F	X																													
80000038:	00000002	<invalid command>	6															FX																													
80000010 <lp>:	0000a103	lw x2,0(x1)	5																F	ID	D	M1	M2	M3																								
80000014:	0040a183	lw x3,4(x1)	6																	F	ID	D	M1	M2	M3																							
80000018:	fffa0a13	addi x20,x20,-1	7																		F	ID	D	AL																							
8000001c:	808093	addi x1,x1,8	0																			F	ID	D	AL																						
80000020:	002f8fb3	add x31,x31,x2	1																				F	ID	D	AL																					
80000024:	003f8fb3	add x31,x31,x3	2																					F	ID	D	AL																				
80000028:	fe0a14e3	bne x20,x0,80000010 <lp>	3																						F	ID	D	B																			
80000010 <lp>:	0000a103	lw x2,0(x1)	4																							F	ID	D	M1	M2	M3																
80000014:	0040a183	lw x3,4(x1)	5																								F	ID	D	M1	M2	M3														
80000018:	fffa0a13	addi x20,x20,-1	6																									F	ID	D	AL														
8000001c:	808093	addi x1,x1,8	7																										F	ID	D	AL													
80000020:	002f8fb3	add x31,x31,x2	0																											F	ID	D	AL											
80000024:	003f8fb3	add x31,x31,x3	1																												F	ID	D	AL									
80000028:	fe0a14e3	bne x20,x0,80000010 <lp>	2																													F	ID	D	B								
80000010 <lp>:	0000a103	lw x2,0(x1)	3																														F	ID	D	M1	M2	M3					
80000014:	0040a183	lw x3,4(x1)	4																															F	ID	D	M1	M2	M3				
80000018:	fffa0a13	addi x20,x20,-1	5																																F	ID	D	AL					
8000001c:	808093	addi x1,x1,8	6																																	F	ID	D	AL				
80000020:	002f8fb3	add x31,x31,x2	7																																		F	ID	D	AL			
80000024:	003f8fb3	add x31,x31,x3	0																																			F	ID	D	AL		
80000028:	fe0a14e3	bne x20,x0,80000010 <lp>	1																																				F	ID	D	B		
80000010 <lp>:	0000a103	lw x2,0(x1)	2																																					F	ID	D	X		
80000014:	0040a183	lw x3,4(x1)	3																																						F	ID	DX		
80000018:	fffa0a13	addi x20,x20,-1	4																																							F	X		
8000001c:	808093	addi x1,x1,8	5																																								FX		
8000002c:	001f8f93	addi x31,x31,1	3																																									F	ID	D	AL		
80000030 <lp2>:	0000006f	jal x0,80000030 <lp2>	4																																									F	ID	D	B		
80000034 <_x>:	00000001	<invalid command>	5																																									F	ID	D	X		
80000038:	00000002	<invalid command>	6																																									F	ID	DX		
8000003c:	00000003	<invalid command>	7																																										F	X		
80000040:	00000004	<invalid command>	5																																											FX		
80000030 <lp2>:	0000006f	jal x0,80000030 <lp2>	6																																												F	ID	D	B
				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 15 – Трасса работы оптимизированной 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.