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

Название Изучение принципов работы микропроцессорного ядра RISC-V

Дисциплина Архитектура элекронно-вычислительных машин

Студент: _____ Шубенина Д. В.
подпись, дата Фамилия, И.О.

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

Москва — 2023 г.

Содержание

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.

1 Основные 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.

1.1 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.

1.2 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.

2 Ход работы

2.1 Задание 0

Ниже приведен дизассмблированный код общей программы, полученный в результате выполнения команды make.

Листинг 2.1 – Дизассемблированный код общей программы

```
SYMBOL TABLE:
80000000 l    d  .text  00000000 .text
80000040 l    d  .data  00000000 .data
00000000 l    df *ABS*  00000000 test.o
00000008 l       *ABS*  00000000 len
00000004 l       *ABS*  00000000 enroll
00000004 l       *ABS*  00000000 elem_sz
80000040 l       .data  00000000 _x
8000000c l       .text  00000000 loop
8000003c l       .text  00000000 forever
80000000 g       .text  00000000 _start
80000060 g       .data  00000000 _end

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 <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>

Disassembly of section .data:
```

```
39
40 80000040 <_x>:
41 80000040:        0001                    c.addi  x0,0
42 80000042:        0000                    c.unimp
43 80000044:        0002                    c.slli64        x0
44 80000046:        0000                    c.unimp
45 80000048:        00000003                lb      x0,0(x0) # 0 <elem_sz-0x4>
46 8000004c:        0004                    .2byte  0x4
47 8000004e:        0000                    c.unimp
48 80000050:        0005                    c.addi  x0,1
49 80000052:        0000                    c.unimp
50 80000054:        0006                    c.slli  x0,0x1
51 80000056:        0000                    c.unimp
52 80000058:        00000007                .4byte  0x7
53 8000005c:        0008                    .2byte  0x8
```

2.2 Задание 1

Листинг 2.2 – Исходный текст программы для варианта 21

```
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, x0, (len-1)/enroll
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    lw x4, 8(x1)
16    lw x5, 12(x1)
17    bltu x2, x31, lt1
18    add x31, x0, x2
19 lt1:    bltu x3, x31, lt2
20    add x31, x0, x3
21 lt2:    bltu x4, x31, lt3
22    add x31, x0, x4 #!
23 lt3:    bltu x5, x31, lt4
24    add x31, x0, x5
25 lt4:
26    add x1, x1, elem_sz*enroll
27    addi x20, x20, -1
28    bne x20, x0, lp
29 lp2: j lp2
30
31    .section .data
32 _x: .4byte 0x1
33    .4byte 0x2
34    .4byte 0x3
35    .4byte 0x4
36    .4byte 0x5
37    .4byte 0x6
38    .4byte 0x7
39    .4byte 0x8
40    .4byte 0x9
```

Листинг 2.3 – Дизассеблированный код программы для варианта 21

```
SYMBOL TABLE:
80000000 l    d  .text  00000000 .text
80000054 l    d  .data  00000000 .data
00000000 l    df *ABS*  00000000 individual.o
00000008 l       *ABS*  00000000 len
00000004 l       *ABS*  00000000 enroll
00000004 l       *ABS*  00000000 elem_sz
80000054 l       .data  00000000 _x
80000014 l       .text  00000000 lp
8000002c l       .text  00000000 lt1
80000034 l       .text  00000000 lt2
8000003c l       .text  00000000 lt3
80000044 l       .text  00000000 lt4
80000050 l       .text  00000000 lp2
80000000 g       .text  00000000 _start
80000078 g       .data  00000000 _end

Disassembly of section .text:

80000000 <_start>:
80000000:       00000097                auipc   x1,0x0
80000004:       05408093                addi    x1,x1,84 # 80000054 <_x>
80000008:       00100a13                addi    x20,x0,1
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:       0080a203                lw      x4,8(x1)
80000020:       00c0a283                lw      x5,12(x1)
80000024:       01f16463                bltu    x2,x31,8000002c <lt1>
80000028:       00200fb3                add     x31,x0,x2

8000002c <lt1>:
8000002c:       01f1e463                bltu    x3,x31,80000034 <lt2>
80000030:       00300fb3                add     x31,x0,x3

80000034 <lt2>:
80000034:       01f26463                bltu    x4,x31,8000003c <lt3>
80000038:       00400fb3                add     x31,x0,x4

8000003c <lt3>:
8000003c:       01f2e463                bltu    x5,x31,80000044 <lt4>
80000040:       00500fb3                add     x31,x0,x5

80000044 <lt4>:
```

```
80000044:       01008093                addi    x1,x1,16
80000048:       fffa0a13                addi    x20,x20,-1
8000004c:       fc0a14e3                bne     x20,x0,80000014 <lp>

80000050 <lp2>:
80000050:       0000006f                jal     x0,80000050 <lp2>

Disassembly of section .data:

80000054 <_x>:
80000054:       0001                    c.addi  x0,0
80000056:       0000                    c.unimp
80000058:       0002                    c.slli64        x0
8000005a:       0000                    c.unimp
8000005c:       00000003                lb      x0,0(x0) # 0 <elem_sz-0x4>
80000060:       0004                    .2byte  0x4
80000062:       0000                    c.unimp
80000064:       0005                    c.addi  x0,1
80000066:       0000                    c.unimp
80000068:       0006                    c.slli  x0,0x1
8000006a:       0000                    c.unimp
8000006c:       00000007                .4byte  0x7
80000070:       0008                    .2byte  0x8
80000072:       0000                    c.unimp
80000074:       0009                    c.addi  x0,2
```

Листинг 2.4 – Псевдокод на языке C эквивалентной 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){}
}
```

2.3 Задание 2

Для выполнения задания 2 необходимо получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки и диспетчеризации команды с указанным адресом.

Вариант 21:

Адрес команды: 80000030, 2-я итерация

Код команды: fffa0a13

Команда: addi x20, x20, -1

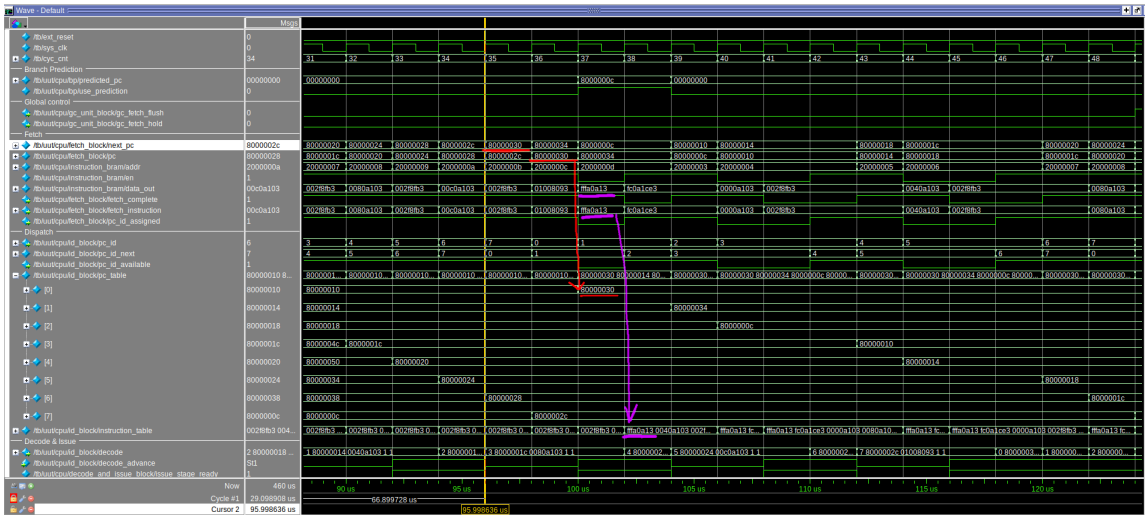

Рисунок 2.1 – Временная диаграмма выборки и диспетчеризации команды

2.4 Задание 3

Для выполнения задания 3 необходимо получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с указанным адресом.

Вариант 21:

Адрес команды, номер итерации: 80000018, 1-я.

Код команды: 002f8fb3.

Команда: add x31,x31,x2.

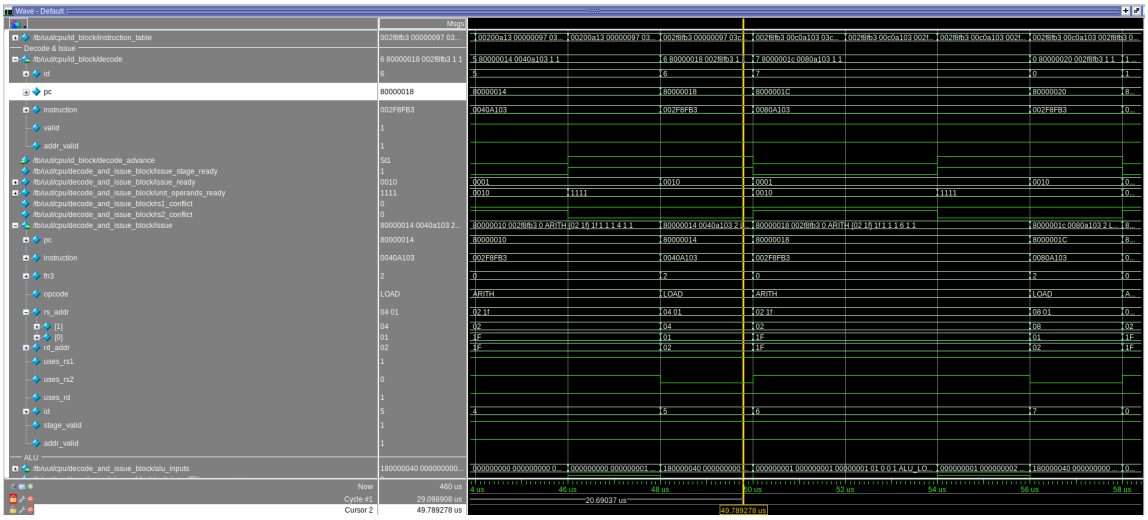

Рисунок 2.2 – Временная диаграмма выполнения стадии декодирования и планирования на выполнение команды

Из рисунка 2.2 видно, что сигналы rs1_conflict и rs2_conflict не выставлены. Следовательно, конфликта нет.

2.5 Задание 4

Для выполнения задания 4 необходимо получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с указанным адресом.

Вариант 21:

Адрес команды, номер итерации: 80000028, 2-я.

Код команды: 002f8fb3.

Команда: add x31,x31,x2.

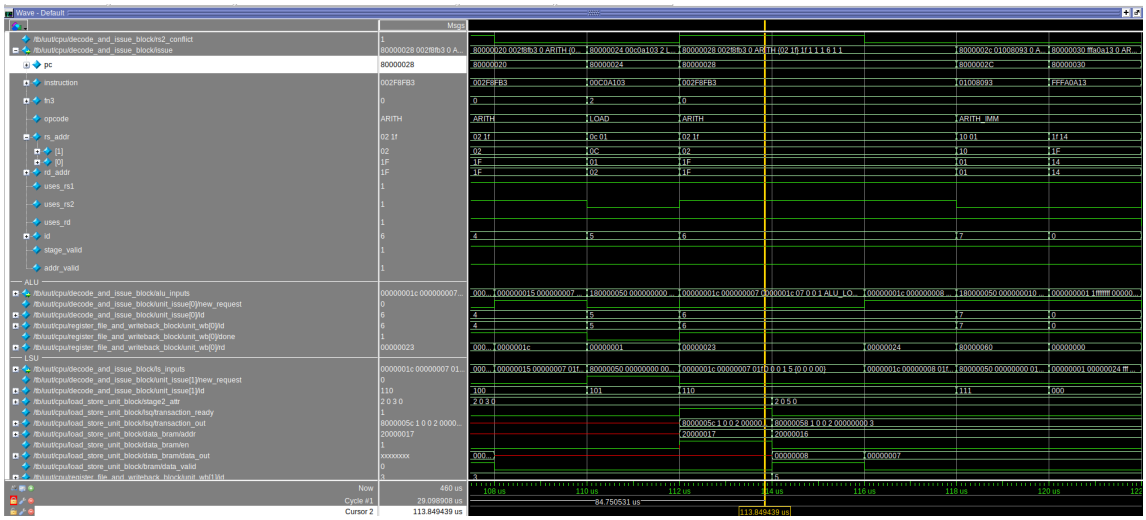

Рисунок 2.3 – Выполнение команды с адресом 80000028

2.6 Задание 5

2.6.1 Выполнение команды

Ниже приведены временные диаграммы этапов выполнения команды add x31,x0,x4 (адрес команды 80000038).

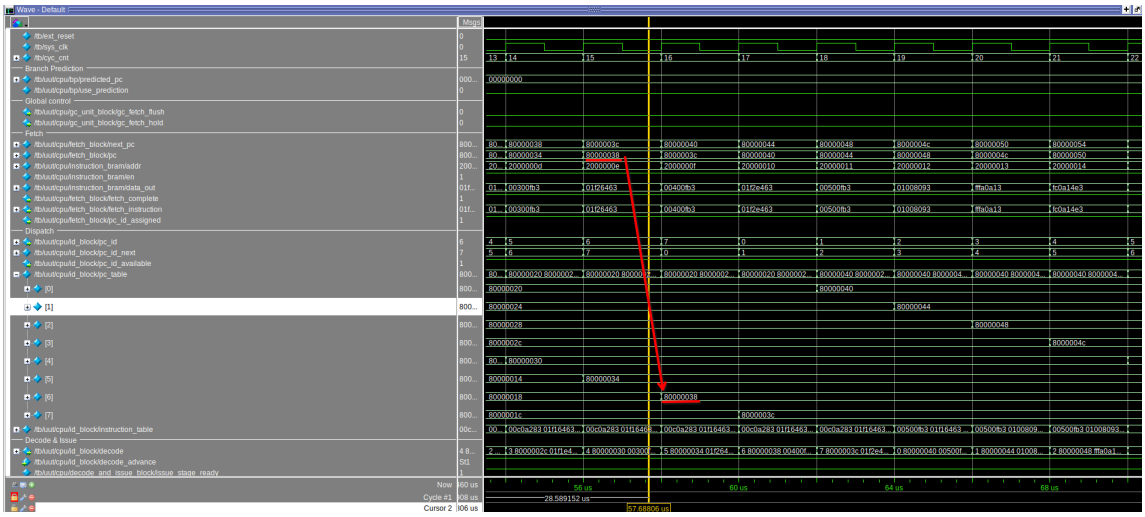

Рисунок 2.4 – Выборка и диспетчеризация

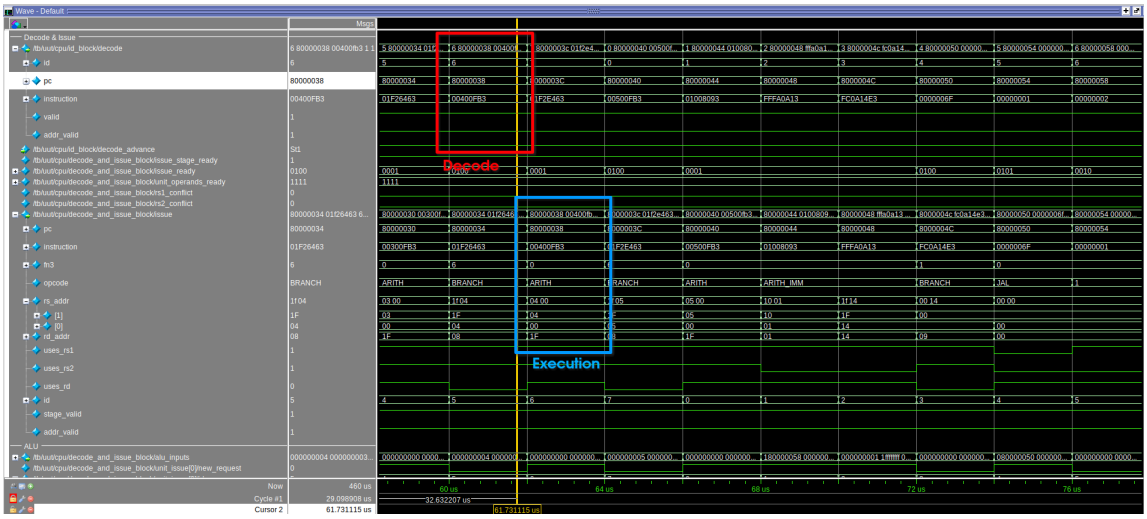

Рисунок 2.5 – Декодирование и выполнение

2.6.2 Трасса выполнения 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
80000000<_start>	00000097	auipc x1,0x0	0	F	ID	D	AL																									
80000004	05408093	addi x1,x1,84#8000054<_x>	1		F	ID	D	AL																								
80000008	00100a13	addi x20,x0,1	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	0080a203	lw x4,8(x1)	7								F	ID	D	M1	M2	M3																
80000020	00c0a283	lw x5,12(x1)	0									F	ID	D	M1	M2	M3															
80000024	01f16463	bltu x2,x31,8000002c<lt1>	1										F	ID	D	B																
80000028	00200fb3	add x31,x0,x2	2											F	ID	D	AL															
8000002c<lt1>	01f1e463	bltu x3,x31,80000034<lt2>	3												F	ID	D	B														
80000030	00300fb3	add x31,x0,x3	4													F	ID	D	AL													
80000034<lt2>	01f26463	bltu x4,x31,8000003c<lt3>	5														F	ID	D	B												
80000038	00400fb3	add x31,x0,x4	6															F	ID	D	AL											
8000003c<lt3>	01f2e463	bltu x5,x31,80000044<lt4>	7																F	ID	D	B										
80000040	00500fb3	add x31,x0,x5	0																	F	ID	D	AL									
80000044	01008093	addi x1,x1,16	1																		F	ID	D	AL								
80000048	fffa0a13	addi x20,x20,-1	2																			F	ID	D	AL							
8000004c	fc0a14e3	bne x20,x0,80000014<lp>	3																				F	ID	D	B						
80000050<lp2>	0000006f	jal x0,80000050 <lp2>	4																					F	ID	D	B					
80000054	00000001	<invalid command>	5																						F	ID	D	X				
80000058	00000002	<invalid command>	6																							F	ID	DX				
8000005c	00000003	<invalid command>	7																								F	X				
80000060	00000004	<invalid command>	0																									FX				
80000050<lp2>	0000006f	jal x0,80000050 <lp2>	6																										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
				Номер такта																												

Рисунок 2.6 – Трасса выполнения программы

При составлении трассы, изображенной на рисунке 2.6, не было обнаружено ни одного конфликта, программа в оптимизации не нуждается.

3 Заключение

В ходе выполнения лабораторной работы были изучены основные особенности архитектуры процессора RISC-V, а также основные инструкции и регистры процессора. Были получены навыки работы с ModelSim и составления трассы выполнения программы по временным диаграммам, получаемом с помощью этого ПО.