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» (ИУ)
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

Отчет по лабораторной работе №1 по курсу «Архитектура электронно-вычислительных машин»

Изучение принципов работы микропроцессорного ядра RISC-V

Группа: ИУ7-53Б

Студент: _____ Дьяченко А. А.
(Подпись, дата) (Фамилия И. О.)

Преподаватель: _____ Ибрагимов С. В.
(Подпись, дата) (Фамилия И. О.)

Москва, 2023 г.

СОДЕРЖАНИЕ

1 Введение

1.1 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.

1.2 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.

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 Общая для всех вариантов программа

2.1 Исследуемая программа

Исходный текст исследуемой программы представлен на листинге 1.

Листинг 1 – Исходный текст общей программы

```
1   .section .text
2   .globl _start;
3   len = 8 Размер# массива
4   enroll = 4 Количество# обрабатываемыхэлементовзаоднуитерацию
5   elem_sz = 4 Размер# одногоэлементамассива
6 _start:
7   addi x20, x0, len/enroll
8   la x1, _x
9 loop:
10  lw x2, 0(x1)
11  add x31, x31, x2
12  lw x2, 4(x1)
13  add x31, x31, x2
14  lw x2, 8(x1)
15  add x31, x31, x2
16  lw x2, 12(x1)
17  add x31, x31, x2
18  addi x1, x1, elem_sz*enroll
19  addi x20, x20, -1
20  bne x20, x0, loop
21  addi x31, x31, 1
22 forever: j forever
23
24  .section .data
25 _x: .4byte 0x1
26  .4byte 0x2
27  .4byte 0x3
28  .4byte 0x4
29  .4byte 0x5
30  .4byte 0x6
31  .4byte 0x7
32  .4byte 0x8
```

Дизассемблерный листинг исследуемой программы представлен на листинге 2.

Листинг 2 – Дизассемблерный листинг общей программы

```
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>
```

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

Листинг 3 – Псевдокод на языке C

```
#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) {}
}
```

2.2 Результаты исследования программы

Скриншот, полученный в ходе выполнения задания №2 (получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки и диспетчеризации команды с адресом 80000020 на первой итерации) представлен на рисунке 1.

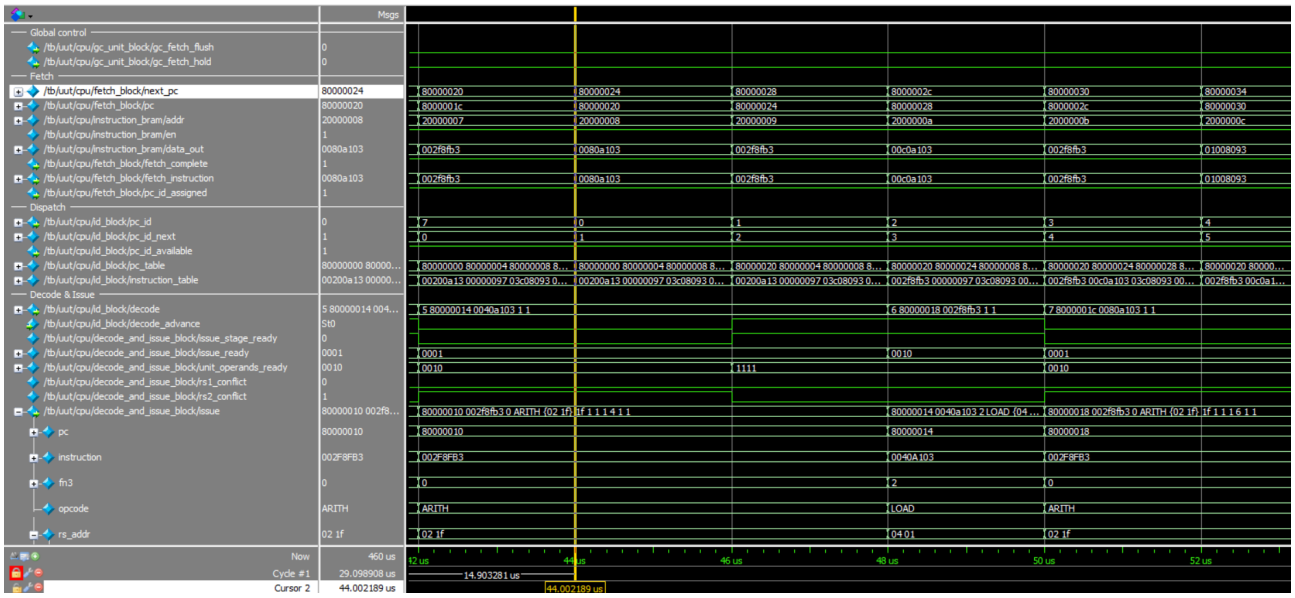

Рисунок 1 – Временная диаграмма выполнения стадий выборки и диспетчеризации

Скриншот, полученный в ходе выполнения задания №3 (получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с адресом 8000002c на первой итерации) представлен на рисунке 2.

Рисунок 2 – Временная диаграмма выполнения стадий декодирования и планирования на выполнение

Скриншот, полученный в ходе выполнения задания №4 (получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с адресом 80000014 на первой итерации) представлен на рисунке 3.

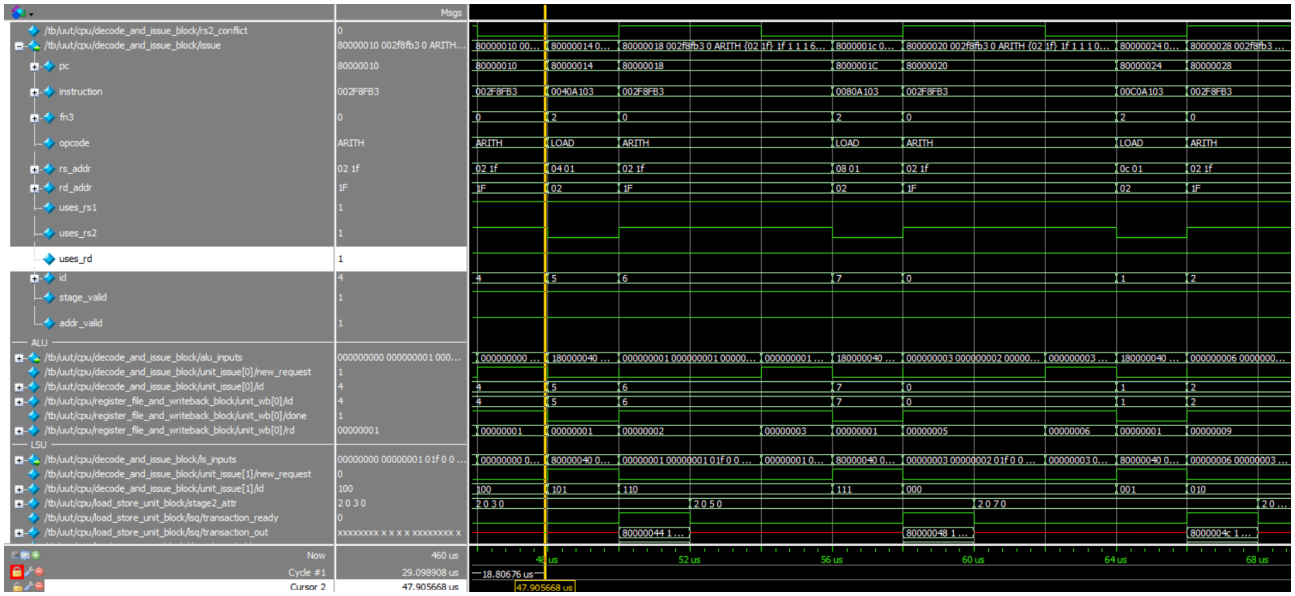

Рисунок 3 – Временная диаграмма выполнения стадии выполнения

3 Программа по варианту

Все задания выполнялись по индивидуальному варианту №6. Исходный текст исследуемой программы представлен на листинге 4.

Листинг 4 – Исходный текст индивидуальной программы

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

Код программы на языке C, соответствующей индивидуальному варианту, представлен на листинге 5.

Листинг 5 – Код программы на языке C

```c
#include <stdio.h>

int main() {
    int len = 8;
    int enroll = 2;
    int elem_sz = 4;

    int _x[] = { 1, 2, 3, 4, 5, 6, 7, 8 };

    int x20 = len / enroll;
    int* x1 = _x;
    int x31 = 0;

    while (x20 != 0) {
        int x2 = _x[0];
        int x3 = _x[4];
        x1 += elem_sz * enroll;
        x20--;

        x31 += x2;
        x31 += x3;
    }

    x31++;

    printf("%d \n", x31);

    while (1) { }

    return 0;
}
```

3.1 Трасса работы программы

Трасса работы программы индивидуального варианта представлена на рисунке 4.

Адрес	Код команды	Команда	id	Номер такта
80000000<_start>	00400a13	addi x20,x0,4	0	F ID D AL
80000004	00000097	auipc x1,0x0	1	F ID D AL
80000008	3008093	addi x1,x1,48#80000034<x>	2	F ID D AL
8000000c	000000fb3	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	808093	addi x1,x1,8	6	F ID D AL
8000001c	fffa0a13	addi x20,x20,-1	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>	000006f	jal x0,80000030<lp2>	4	F ID DX
80000034	00000001	<invalid operation>	5	F X
80000038	00000002	<invalid operation>	6	FX
80000010	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	808093	addi x1,x1,8	6	F ID D AL
8000001c	fffa0a13	addi x20,x20,-1	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	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	808093	addi x1,x1,8	5	F ID D AL
8000001c	fffa0a13	addi x20,x20,-1	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	0000a103	lw x2,0(x1)	2	F ID D M1 M2 M3
80000014	0040a183	lw x3,4(x1)	3	F ID D M1 M2 M3
80000018	808093	addi x1,x1,8	4	F ID D AL
8000001c	fffa0a13	addi x20,x20,-1	5	F ID D AL
80000020	002f8fb3	add x31,x31,x2	6	F ID D AL
80000024	003f8fb3	add x31,x31,x3	7	F ID D AL
80000028	fe0a14e3	bne x20,x0,80000010<lp>	0	F ID D B
80000010	0000a103	lw x2,0(x1)	1	F ID D X
80000014	0040a183	lw x3,4(x1)	2	F ID DX
80000018	808093	<invalid operation>	3	F X
8000001c	fffa0a13	<invalid operation>	4	FX
8000002c	001f8f93	addi x31,x31,1	2	F ID D AL
80000030<lp2>	000006f	jal x0,80000030<lp2>	3	F ID D B
80000034	00000001	<invalid operation>	4	F ID D X
80000038	00000002	<invalid operation>	5	F ID DX
8000003c	00000003	<invalid operation>	6	F X
80000040	00000004	<invalid operation>	7	FX
80000030	000006f	jal x0,80000030 <lp2>	5	F ID D AL

Рисунок 4

3.2 Сравнение значения регистров

Значение регистра x31 на момент окончания выполнения программы равно значению того же регистра, полученного в Задании №1, и равняется 25.

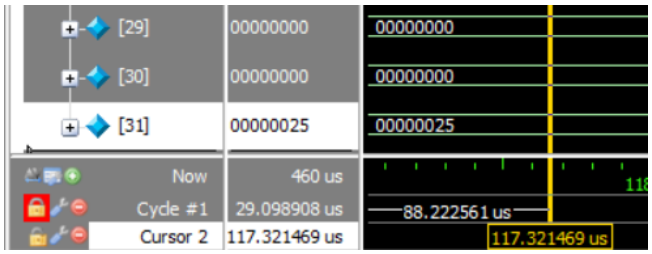

Рисунок 5 – Значение регистра x31 на момент окончания выполнения программы

3.3 Временные диаграммы сигналов

Временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом #! lw x3, 4(x1), представлены на рисунке 6.

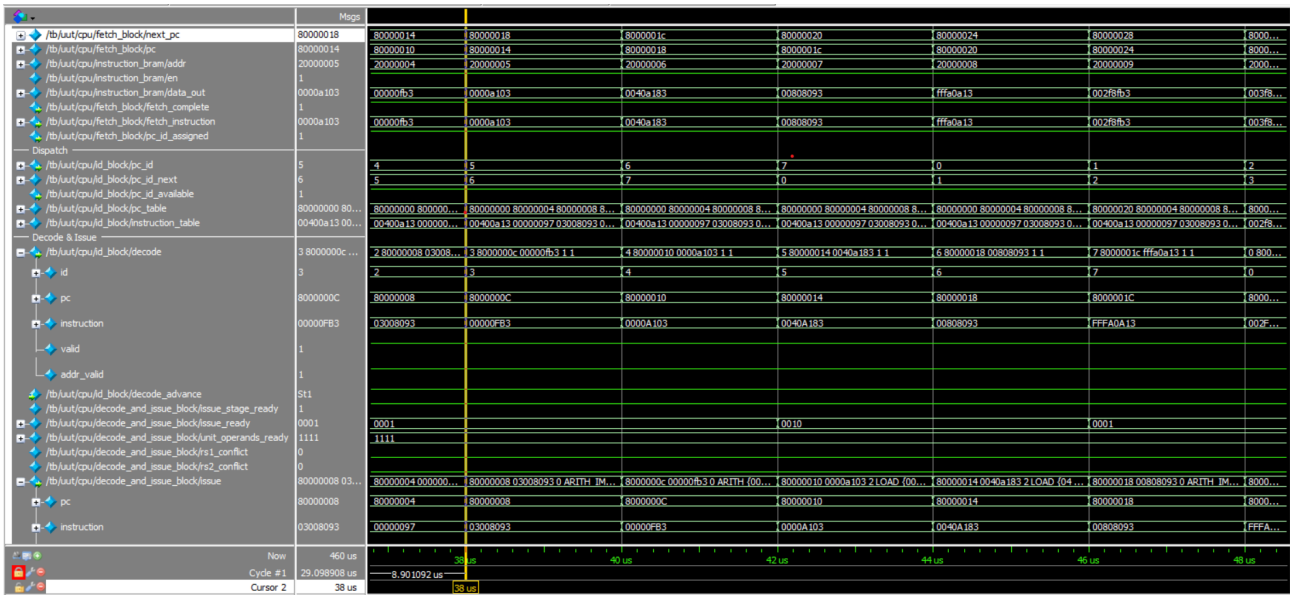

Рисунок 6 – Стадии выполнения команды lw x3, 4(x1)

4 Вывод об эффективности работы программы

Как показано в ходе работы программы, представленной на рисунке **??**, не возникает никаких конфликтов, связанных с регистрами. Все команды выполняются немедленно после завершения предыдущей команды, благодаря грамотному распределению операций между операциями доступа к памяти и арифметическими операциями.

5 Вывод

В результате выполнения лабораторной работы были изучены принципы функционирования, построения и особенности архитектуры суперскалярных конвейерных микропроцессоров.