



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 электронно-вычислительных машин

---

Студент:

\_\_\_\_\_ Шубенина Д. В.  
подпись, дата      Фамилия, И.О.

Преподаватель:

\_\_\_\_\_ Попов А. Ю.  
подпись, дата      Фамилия, И. О.

Москва — 2023 г.

# Содержание

<b>Цель работы</b>	<b>3</b>
<b>1 Основные сведения</b>	<b>4</b>
1.1 Модель памяти . . . . .	4
1.2 Система команд . . . . .	4
<b>2 Ход работы</b>	<b>5</b>
2.1 Задание 0 . . . . .	5
2.2 Задание 1 . . . . .	6
2.3 Задание 2 . . . . .	10
2.4 Задание 3 . . . . .	11
2.5 Задание 4 . . . . .	12
2.6 Задание 5 . . . . .	13
2.6.1 Выполнение команды . . . . .	13
2.6.2 Трасса выполнения программы . . . . .	14

# 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 дизассемблированный код общей программы, полученный в результате выполнения команды make.

Листинг 2.1 – Дизассемблированный код общей программы

```
1 SYMBOL TABLE:
2 80000000 1      d  .text  00000000 .text
3 80000040 1      d  .data  00000000 .data
4 00000000 1      df *ABS*  00000000 test.o
5 00000008 1      *ABS*  00000000 len
6 00000004 1      *ABS*  00000000 enroll
7 00000004 1      *ABS*  00000000 elem_sz
8 80000040 1      .data  00000000 _x
9 8000000c 1      .text  00000000 loop
10 8000003c 1      .text  00000000 forever
11 80000000 g      .text  00000000 _start
12 80000060 g      .data  00000000 _end
13
14 Disassembly of section .text:
15
16 80000000 <_start>:
17 80000000:      00200a13          addi    x20,x0,2
18 80000004:      00000097          auipc   x1,0x0
19 80000008:      03c08093          addi    x1,x1,60 # 80000040 <_x>
20
21 8000000c <loop>:
22 8000000c:      0000a103          lw      x2,0(x1)
23 80000010:      002f8fb3          add     x31,x31,x2
24 80000014:      0040a103          lw      x2,4(x1)
25 80000018:      002f8fb3          add     x31,x31,x2
26 8000001c:      0080a103          lw      x2,8(x1)
27 80000020:      002f8fb3          add     x31,x31,x2
28 80000024:      00c0a103          lw      x2,12(x1)
29 80000028:      002f8fb3          add     x31,x31,x2
30 8000002c:      01008093          addi    x1,x1,16
31 80000030:      fffa0a13          addi    x20,x20,-1
32 80000034:      fc0a1ce3          bne     x20,x0,8000000c <loop>
33 80000038:      001f8f93          addi    x31,x31,1
34
35 8000003c <forever>:
36 8000003c:      0000006f          jal     x0,8000003c <forever>
37
38 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 – Дизассемблированный код программы для варианта 21

```

1 SYMBOL TABLE:
2 80000000 1      d  .text  00000000 .text
3 80000054 1      d  .data  00000000 .data
4 00000000 1      df *ABS*  00000000 individual.o
5 00000008 1      *ABS*  00000000 len
6 00000004 1      *ABS*  00000000 enroll
7 00000004 1      *ABS*  00000000 elem_sz
8 80000054 1      .data  00000000 _x
9 80000014 1      .text  00000000 lp
10 8000002c 1      .text  00000000 lt1
11 80000034 1      .text  00000000 lt2
12 8000003c 1      .text  00000000 lt3
13 80000044 1      .text  00000000 lt4
14 80000050 1      .text  00000000 lp2
15 80000000 g      .text  00000000 _start
16 80000078 g      .data  00000000 _end
17
18 Disassembly of section .text:
19
20 80000000 <_start>:
21 80000000:          00000097          auipc    x1,0x0
22 80000004:          05408093          addi     x1,x1,84 # 80000054 <_x>
23 80000008:          00100a13          addi     x20,x0,1
24 8000000c:          0000af83          lw      x31,0(x1)
25 80000010:          00408093          addi     x1,x1,4
26
27 80000014 <lp>:
28 80000014:          0000a103          lw      x2,0(x1)
29 80000018:          0040a183          lw      x3,4(x1)
30 8000001c:          0080a203          lw      x4,8(x1)
31 80000020:          00c0a283          lw      x5,12(x1)

```

```

32 80000024:      01f16463      bltu    x2,x31,8000002c <lt1>
33 80000028:      00200fb3      add     x31,x0,x2
34
35 8000002c <lt1>:
36 8000002c:      01f1e463      bltu    x3,x31,80000034 <lt2>
37 80000030:      00300fb3      add     x31,x0,x3
38
39 80000034 <lt2>:
40 80000034:      01f26463      bltu    x4,x31,8000003c <lt3>
41 80000038:      00400fb3      add     x31,x0,x4
42
43 8000003c <lt3>:
44 8000003c:      01f2e463      bltu    x5,x31,80000044 <lt4>
45 80000040:      00500fb3      add     x31,x0,x5
46
47 80000044 <lt4>:
48 80000044:      01008093      addi    x1,x1,16
49 80000048:      fffa0a13      addi    x20,x20,-1
50 8000004c:      fc0a14e3      bne     x20,x0,80000014 <lp>
51
52 80000050 <lp2>:
53 80000050:      0000006f      jal     x0,80000050 <lp2>
54
55 Disassembly of section .data:
56
57 80000054 <_x>:
58 80000054:      0001          c.addi  x0,0
59 80000056:      0000          c.unimp
60 80000058:      0002          c.slli64      x0
61 8000005a:      0000          c.unimp
62 8000005c:      00000003      lb      x0,0(x0) # 0 <elem_sz-0x4>
63 80000060:      0004          .2byte  0x4
64 80000062:      0000          c.unimp
65 80000064:      0005          c.addi  x0,1
66 80000066:      0000          c.unimp
67 80000068:      0006          c.slli  x0,0x1
68 8000006a:      0000          c.unimp
69 8000006c:      00000007      .4byte  0x7
70 80000070:      0008          .2byte  0x8
71 80000072:      0000          c.unimp
72 80000074:      0009          c.addi  x0,2

```

## Листинг 2.4 – Псевдокод на языке C эквивалентной программы

```

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

```



```
6      int x20 = len/enroll;
7      int *x1 = _x;
8
9      do {
10         int x2 = x1[0];
11         x31 += x2;
12         x2 = x1[1];
13         x31 += x2;
14         x2 = x1[2];
15         x31 += x2;
16         x2 = x1[3];
17         x31 += x2;
18         x1 += enroll;
19         x20--;
20     } while(x20 != 0);
21     x31++;
22     while(1){}
23 }
```

## 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`

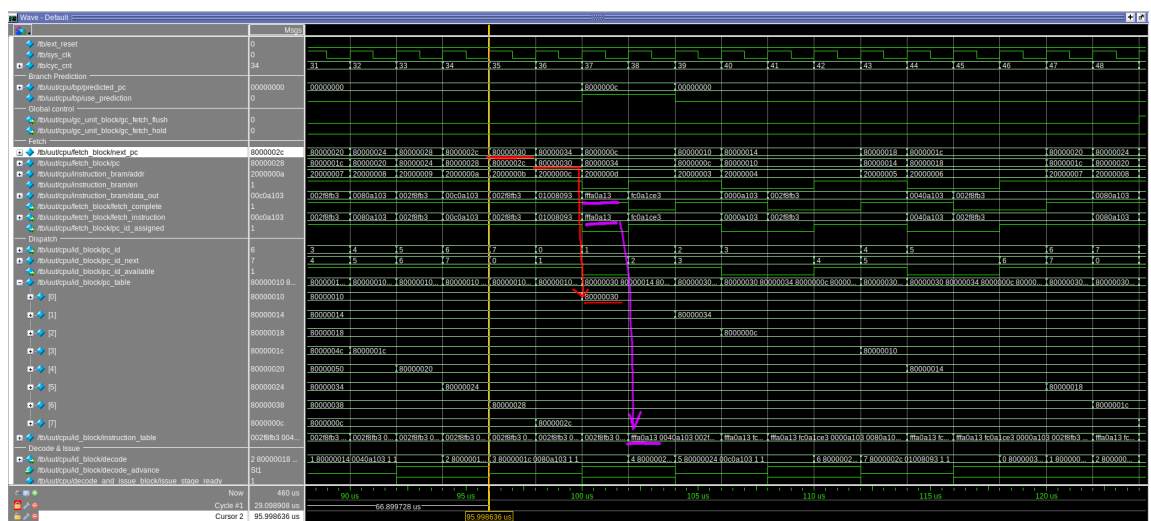


Рисунок 2.1 – Временная диаграмма выборки и диспетчеризации команды

## 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.`

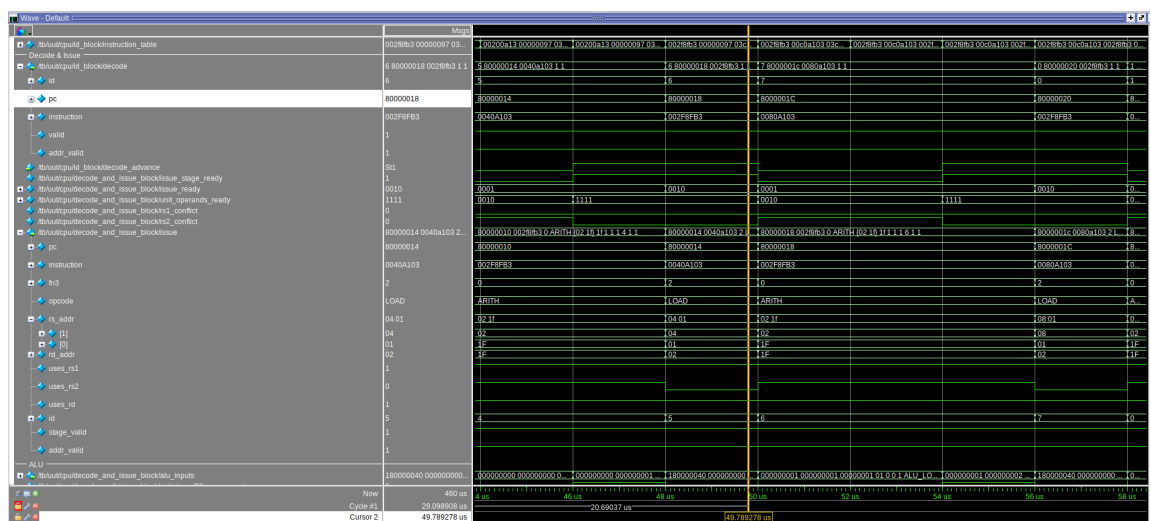


Рисунок 2.2 – Временная диаграмма выполнения стадии декодирования и планирования на выполнение команды

Из рисунка 2.2 видно, что сигналы rs1\_conflict и rs2\_conflict не выставлены. Следовательно, конфликта нет.

## 2.5 Задание 4

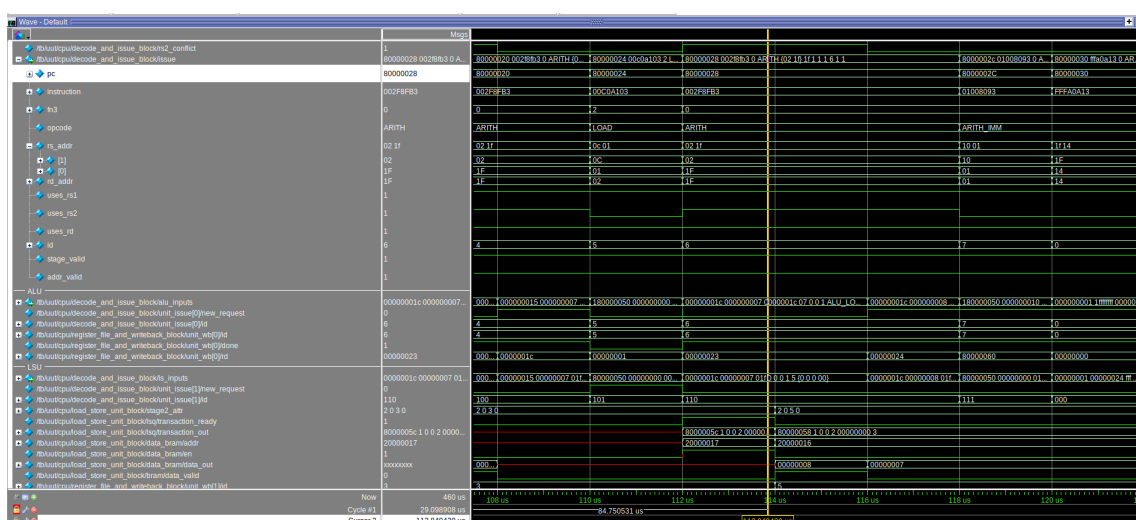
Для выполнения задания 4 необходимо получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с указанным адресом.

Вариант 21:

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

Код команды: 002f8fb3.

Команда: add x31,x31,x2.



## 2.6 Задание 5

### 2.6.1 Выполнение команды

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

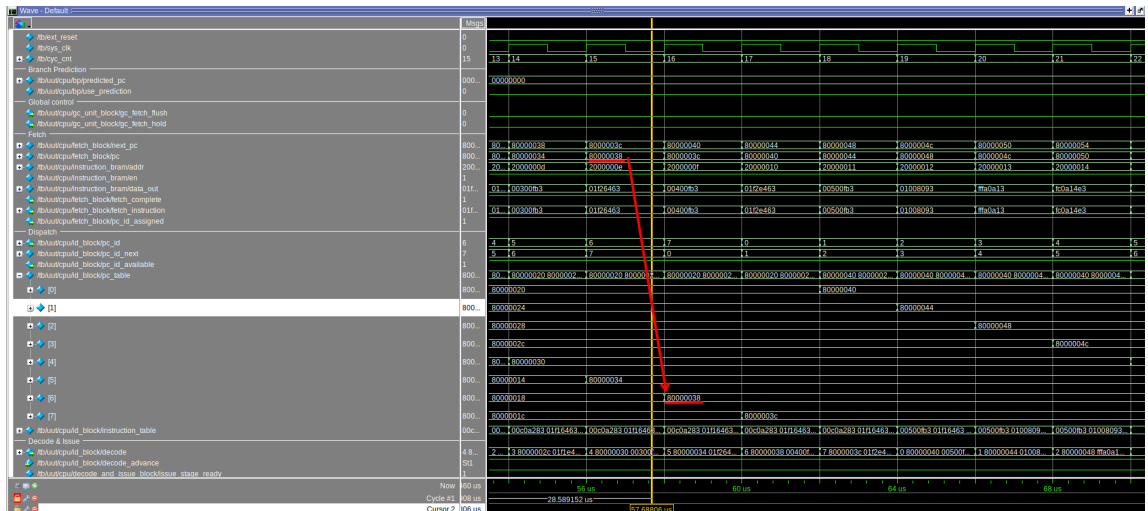


Рисунок 2.4 – Выборка и диспетчеризация

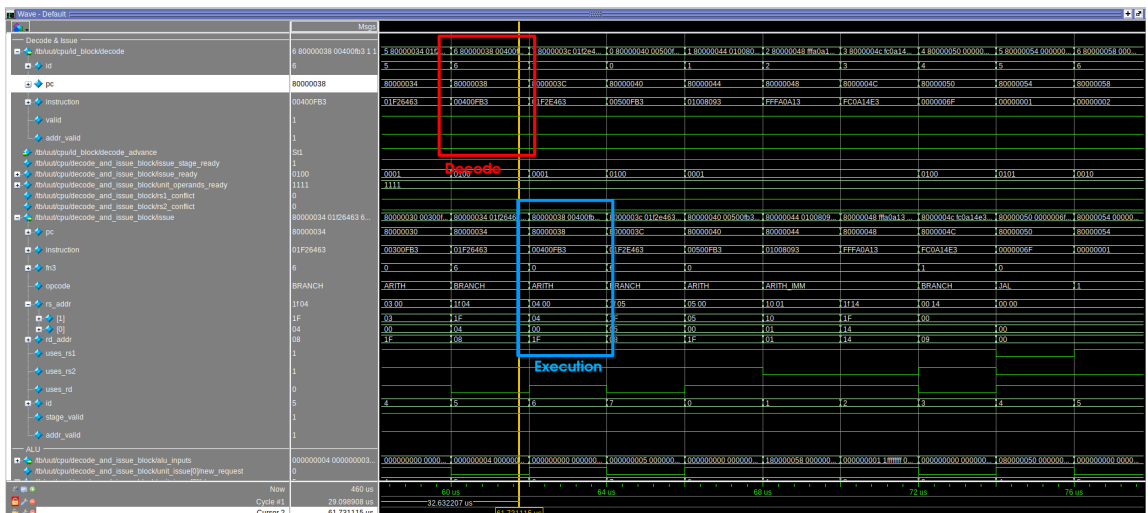


Рисунок 2.5 – Декодирование и выполнение

## 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	D	X				
8000005c	00000003	<invalid command>	7																							F	ID	D	X			
80000060	00000004	<invalid command>	0																								F	ID	D	X		
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