**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 №2**

**По курсу: "Архитектура ЭВМ"**

Студент Чыонг Нгуен Вьет Уи Группа ИУ7-52Б Название предприятия МГТУ им. Н. Э. Баумана, каф. ИУ Тема Изучение принципов работы микропроцессорного ядра RISC-V

|  |  |  |
| --- | --- | --- |
| Студент: |  | Чыонг Н.В.У. |
| Преподаватель: | подпись, дата | Фамилия, И.О.  Попов A.Ю. |
|  | подпись, дата | Фамилия, И. О. |

Москва — 2022 г.

**СОДЕРЖАНИЕ**

[Введение](#_bookmark0) 3

1. [Аналитическая часть](#_bookmark1) 4
   1. [Архитектура набора команд RV32I](#_bookmark2) 4
   2. [Микроархитектура](#_bookmark3) 4
2. [Задание 1](#_bookmark5) 7

[3 Задания 2, 3, 4](#_bookmark7) 10

1. [Задание 5](#_bookmark11) 13
2. [Оптимизация](#_bookmark13) 17

[Заключение](#_bookmark16) 22

2

# ВВЕДЕНИЕ

Основной целью работы является ознакомление с принципами функциони­ рования, построения и особенностями архитектуры суперскалярных конвейерных микропроцессоров. Дополнительной целью работы является знакомство с принципа­ ми проектирования и верификации сложных цифровых устройств с использованием языка описания аппаратуры SystemVerilog и ПЛИС.

Для достижения данной цели необходимо выполнить следующие задачи:

* + ознакомиться с набором команд RV32I;
  + ознакомиться с основными принципами работы ядра Taiga: изучить операции, выполняемые на каждой стадии обработки команд;
  + на основе полученных знаний проанализировать ход выполнения программы и оптимизировать ее;

3

# Аналитическая часть

## Архитектура набора команд RV32I

RISC-V является открытым современным набором команд, который может использоваться для построения как микроконтроллеров, так и высокопроизводитель­ ных микропроцессоров. В связи с такой широкой областью применения в систему команд введена вариативность. 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 се­ бя основные команды 32-битной целочисленной арифметики кроме умножения и деления. В рамках данного набора команд мы не будем рассматривать системные ко­ манды, связанные с таймерами, системными регистрами, управлением привилегиями, прерываниями и исключениями.

Набор команд RV32I предполагает использование 32 регистров общего назна­ чения 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. Ис­ пользуется 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. Распреде­ ление областей памяти между различными устройствами (ОЗУ, ПЗУ, устройства ввода-вывода) определяется реализацией.

## Микроархитектура

В лабораторной работе рассматривается система, состоящая из вычислитель­ ного ядра Taiga и локальной памяти, реализованной с помощью блочной памяти

4

ПЛИС. Команды и данные находятся в едином адресном пространстве. Дешифратор адресов настроен таким образом, что блок памяти ПЛИС отображается в адресное пространство RISC-V с адреса 0x80000000. Память ПЛИС имеет фиксированную задержку доступа в 1 такт, в связи с чем отпадает необходимость в кеш-памяти.

Taiga является конвейерным микропроцессором с элементами суперскалярно­ сти. При конвейерной организации микропроцессора различные команды одновре­ менно проходят различные стадии своей обработки. Конвейер Taiga насчитывает 4 стадии. В скобках приведены сокращенные обозначения стадий.

1. Выборка(F) — cтадия, на которой команда извлекается из ПК. Выполняется в блоке выборки.
2. Диспетчеризация (ID) — стадия, на которой происходит запись команды в очередь команд для декодирования. Выполняется в блоке управления метаданными.
3. Декодирование и планирование на выполнение (D) — стадия на которой происходит определение типа и полей команды и определение вычислительного блока, способного ее исполнить. Выполняется в блоке декодирования и планирования на выполнение.
4. Выполнение (AL, M1..M3, в зависимости от исполнительного блока) — стадия, на которой команда передается в блок выполнения.

"Ширина"конвейера Taiga равна 1 для всех стадий, кроме стадии выполнения.

В лучшем случае, каждая стадия конвейера выполняется за один такт.

В состав рассматриваемой конфигурации Taiga входит 3 блока выполнения команд: Арифметико-логическое устройство (АЛУ), блок доступа к памяти (LSU) и блок ветвлений. АЛУ и блок ветвлений выполняют команды за 1 такт, LSU — минимум за 3.

Ниже, на рисунке [1.1](#_bookmark4) приведена структурная схема ядра Taiga.

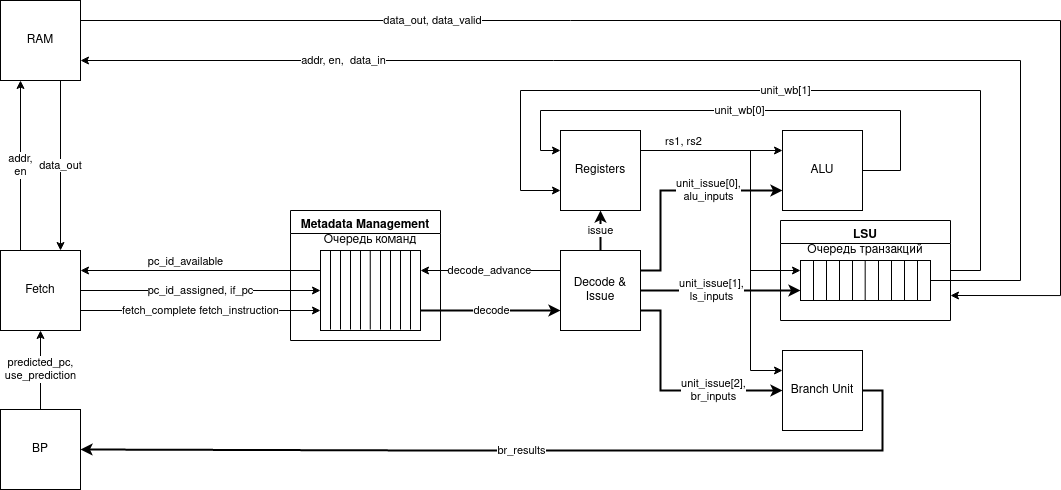


Рисунок 1.1 — Обобщенная структурная схема ядра Taiga

# Задание 1

Варивнт 20.

Листинг 2.1 — Листинг исходной программы

. s e c t i o n . te xt

. g l o b l \_start ;

l e n = 9 #Ра змер ма с сива

e n r o l l = 2 #Колич е с тво обрабатыва емых элементов з а одну ит ерацию elem\_sz = 4 #Ра змер одно г о элемента ма с сива

\_start :

l a x1 , \_x

addi x20 , x0 , ( len −1)/ e n r o l l lw x31 , 0 ( x1 )

addi x1 , x1 , elem\_sz ∗1

lp :

lw x2 , 0 ( x1 ) lw x3 , 4 ( x1 )

add x1 , x1 , elem\_sz∗ e n r o l l bltu x2 , x31 , l t 1

add x31 , x0 , x2 #! l t 1 : bltu x3 , x31 , l t 2

add x31 , x0 , x3

l t 2 :

addi x20 , x20 , −1 bne x20 , x0 , lp

lp 2 : j lp 2

. s e c t i o n . data

\_x: . 4 byte 0 x1

. 4 byte 0 x2

. 4 byte 0 x3

. 4 byte 0 x4

. 4 byte 0 x8

. 4 byte 0 x6

. 4 byte 0 x7

. 4 byte 0 x5

. 4 byte 0 x4

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

7

Листинг 2.2 — Деассемблированный листинг исходной программы

1 Disassembly o f s e c t i o n . te xt :

2

3 80000000 <\_start >:

4 80000000 : 00000097 auipc x1 , 0 x0

5 80000004 : 03 c 08093 addi x1 , x1 , 6 0 # 8000003 c <\_x>

6 80000008 : 00400 a13 addi x20 , x0 , 4

7 8000000 c : 0000 af 83 lw x31 , 0 ( x1 )

8 80000010 : 00408093 addi x1 , x1 , 4

9

10 80000014 <lp >:

11 80000014 : 0000 a103 lw x2 , 0 ( x1 )

12 80000018 : 0040 a183 lw x3 , 4 ( x1 )

13 8000001 c : 00808093 addi x1 , x1 , 8

14 80000020 : 01 f 16463 bltu x2 , x31 , 80000028 <lt 1 >

15 80000024 : 00200 fb 3 add x31 , x0 , x2

16

17 80000028 <lt 1 >:

18 80000028 : 01 f 1 e 463 bltu x3 , x31 , 80000030 <lt 2 >

19 8000002 c : 00300 fb 3 add x31 , x0 , x3

20

21 80000030 <lt 2 >:

22 80000030 : f f f a 0 a 1 3 addi x20 , x20 ,−1

23 80000034 : f e 0 a 10 e 3 bne x20 , x0 , 80000014 <lp>

24

25 80000038 <lp 2 >:

26 80000038 : 0000006 f j a l x0 , 80000038 <lp 2 >

27

28 Disassembly o f s e c t i o n . data :

29

30 8000003 c <\_x>:

31 8000003 c : 0001 c . addi x0 , 0

32 8000003 e : 0000 unimp

33 80000040 : 0002 0 x2

34 80000042 : 0000 unimp

35 80000044 : 00000003 lb x0 , 0 ( x0 ) # 0 <e n r o l l −0x2>

36 80000048 : 0004 c . addi 4 spn x9 , x2 , 0

37 8000004 a : 0000 unimp

38 8000004 c : 0008 c . addi 4 spn x10 , x2 , 0

39 8000004 e : 0000 unimp

40 80000050 : 0006 0 x6

41 80000052 : 0000 unimp

42 80000054 : 00000007 0 x7

43 80000058 : 0005 c . addi x0 , 1

44 8000005 a : 0000 unimp

45 8000005 c : 0004 c . addi 4 spn x9 , x2 , 0

Ниже, в листинге [2.3](#_bookmark6) приведен псевдокод на языке C эквивалентной програм­ мы. В программе находится максимум из элементов массива. В результате, в x31 будет находиться максимальное число из массива \_x, причем, в x31 будет помещен последний из максимумов. Тогда в результате выполнения программы в x31 будет лежать 0x8.

Листинг 2.3 — Псевдокод на языке C эквивалентной программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1  2  3  4 | #d e f i n e l e n 9  #d e f i n e e n r o l l 2  #d e f i n e elem\_sz 4 | | | |
| 5 | void \_start ( ) | | | |
| 6 | { | | | |
| 7 | i n t \_x [ ] = { 1 , 2 , 3 , 4 , 8 , 6 , 7 , 5 , 4 }; | | | |
| 8 | i n t x2 , x3 , x4 , x5 , x31 ; | | | |
| 9 |  | | | |
| 10 | i n t ∗x1 = \_x; | | | |
| 11 | i n t x20 = ( l e n − 1 ) / e n r o l l ; | | | |
| 12 | x31 = x1 [ 0 ] ; | | | |
| 13 | x1 += 1 ; | | | |
| 14 |  | | | |
| 15 | do | |  |  |
| 16 | { | |  |  |
| 17 |  | | x2 | = x1 [ 0 ] ; |
| 18 |  | | x3 | = x1 [ 1 ] ; |
| 19 |  | | x1 | += e n r o l l ; |
| 20 |  | | i f | ( x2>=x31 ) |
| 21 |  | |  | x31 = x2 ; |
| 22 |  | | i f | ( x3>=x31 ) |
| 23 |  | |  | x31 = x3 ; |
| 24 |  | x20 −= 1 ; | | |
| 25 |  | } while ( x20 ) ; | | |
| 26 |  |  | | |
| 27 |  | while ( 1 ) {} | | |
| 28 | } |  | | |

# Задания 2, 3, 4

Мой вариант 20, следовательно в соответствии с заданием 2 выполняем поиск такта, в котором выполняется выборка команды с адрессом 08000002*𝑐*, номер итерации: 2-я. на рисунке [3.1](#_bookmark8) приведена временная диаграмма, поясняющая этапы выборки и диспетчеризации.

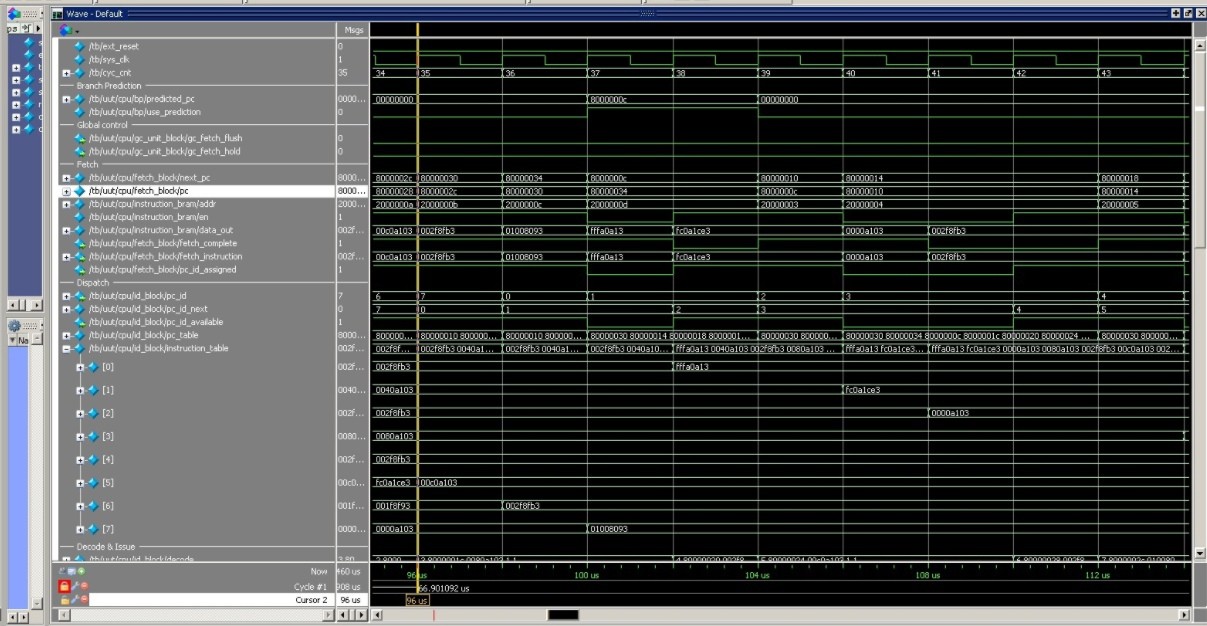


Рисунок 3.1 — Временнуя диаграмма выполнения стадий выборки и диспетчеризации команды

В соответствии с заданием 3 выполняем поиск такта, в котором выполняется декодирование и планирование команды с адрессом 080000038, номер итерации: 2-я. На рисунке [3.2](#_bookmark9) приведена временная диаграмма, поясняющая этап декодирования и планирования.

10

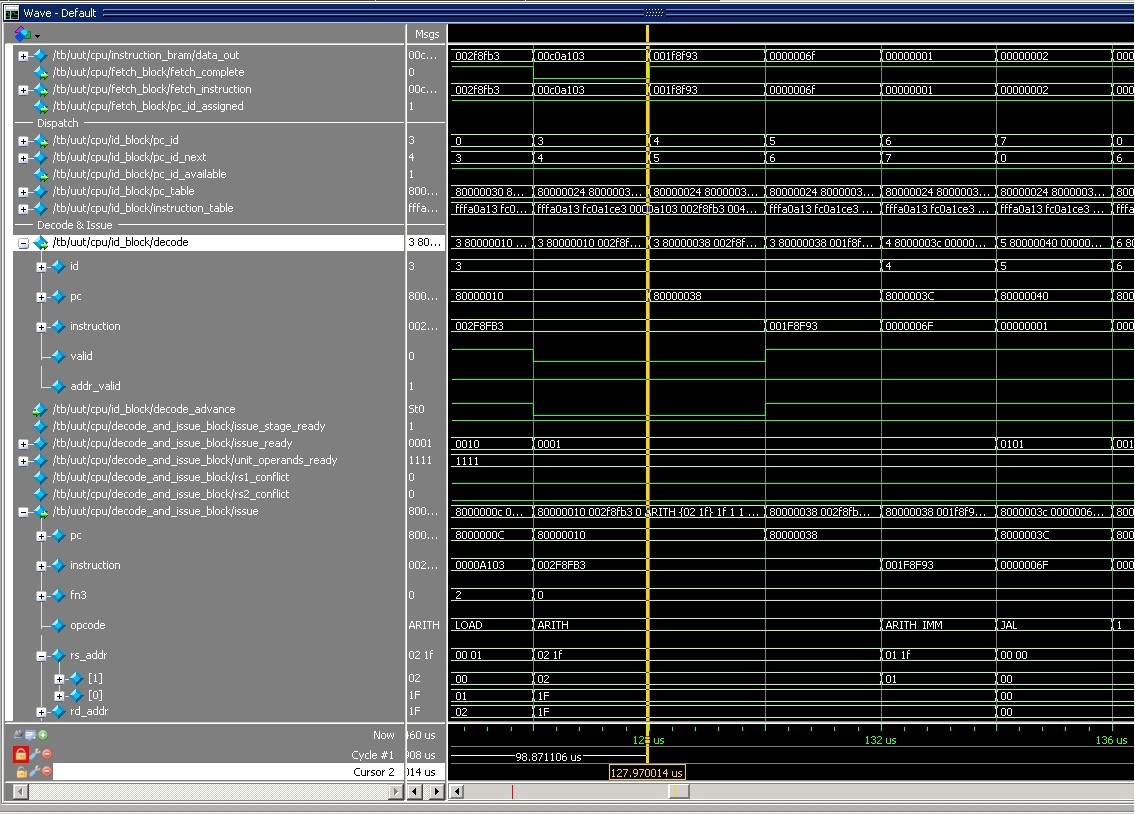


Рисунок 3.2 — Временнуя диаграмма выполнения стадии декодирования и планирования на выполнение команды

В соответствии с заданием 4 выполняем поиск такта, в котором выполняется исполнение команды с адрессом 080000024, номер итерации: 2-я. На рисунке [3.3](#_bookmark10) приведена временная диаграмма, поясняющая этап выполнения команды.

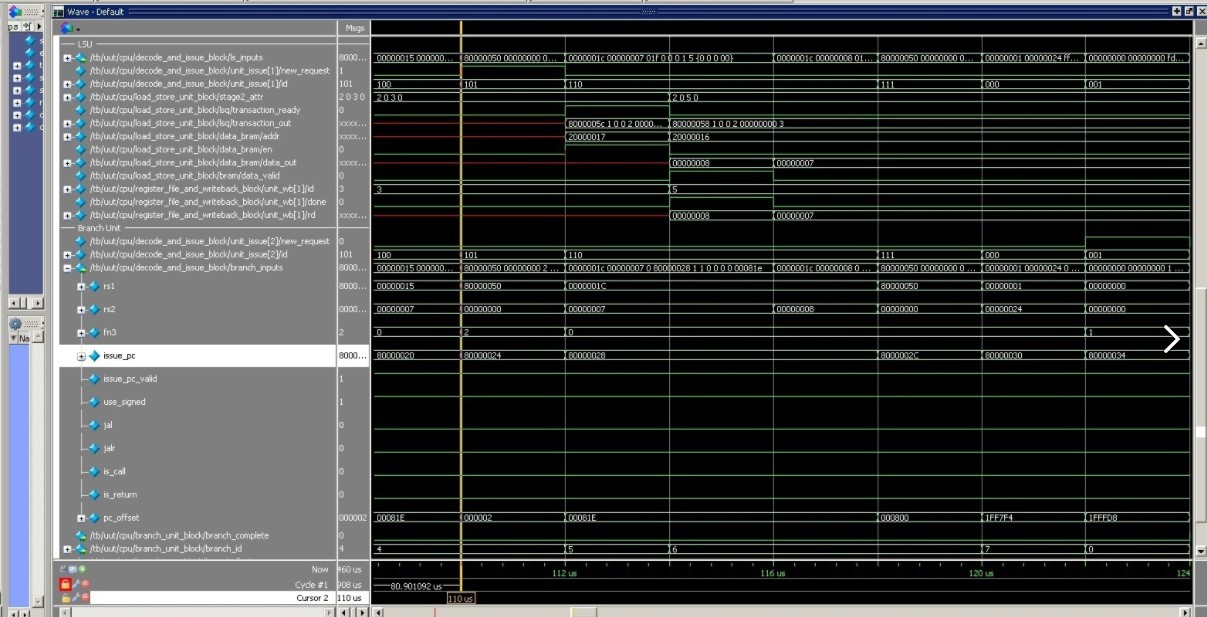


Рисунок 3.3 — Диаграмма, соответствующая этапу выполнения

# Задание 5

На рисунках 4.1-4.4 изображены временные диаграммы сигналов, соответ­ ствующих всем стадиям выполнения команды, обозначенной в тексте программы варианта 20 символом #!.

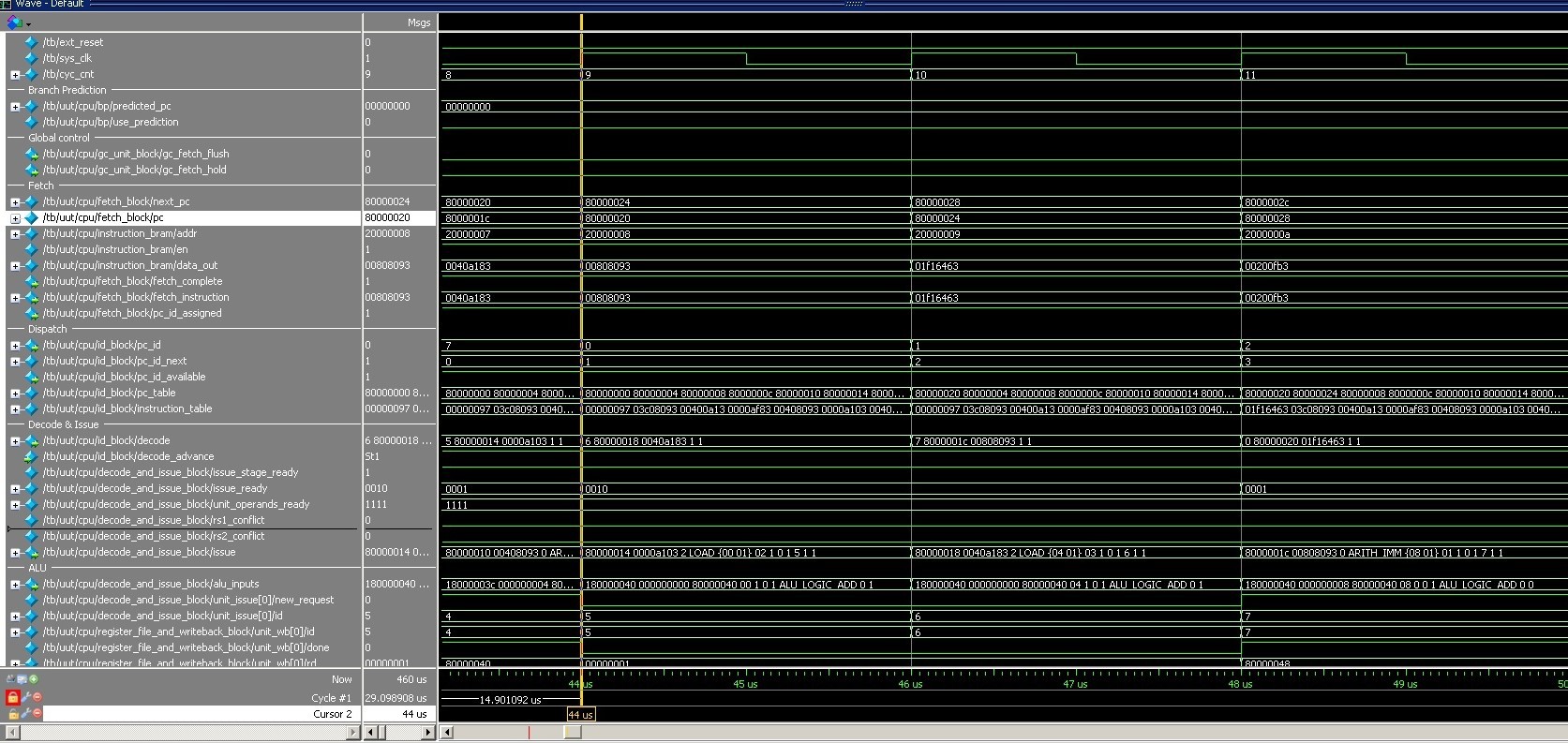


Рисунок 4.1 — Диаграмма, соответствующая этапу выборки команды, обозначенной в тексте программы символом #!

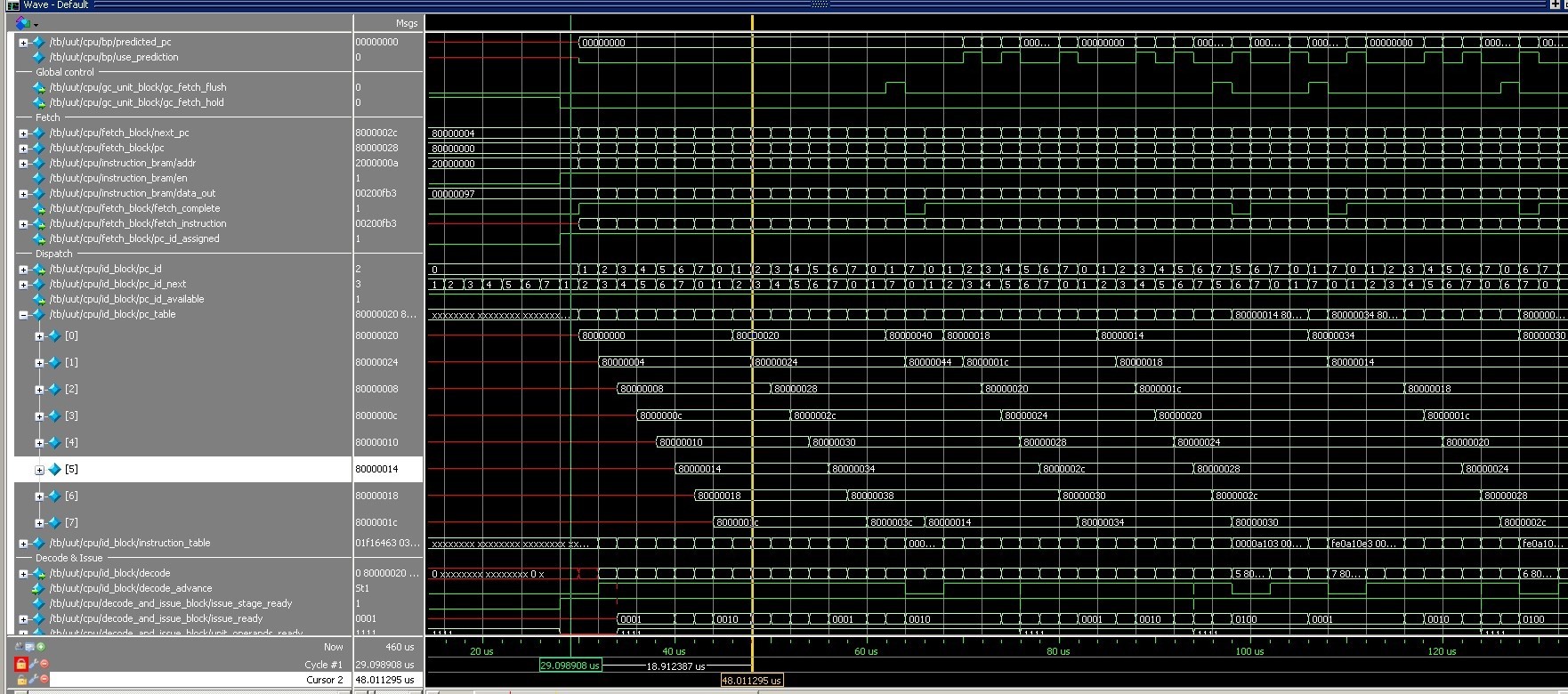


Рисунок 4.2 — Диаграмма, соответствующая этапу диспетчеризации команды, обозначенной в тексте программы символом #!

13

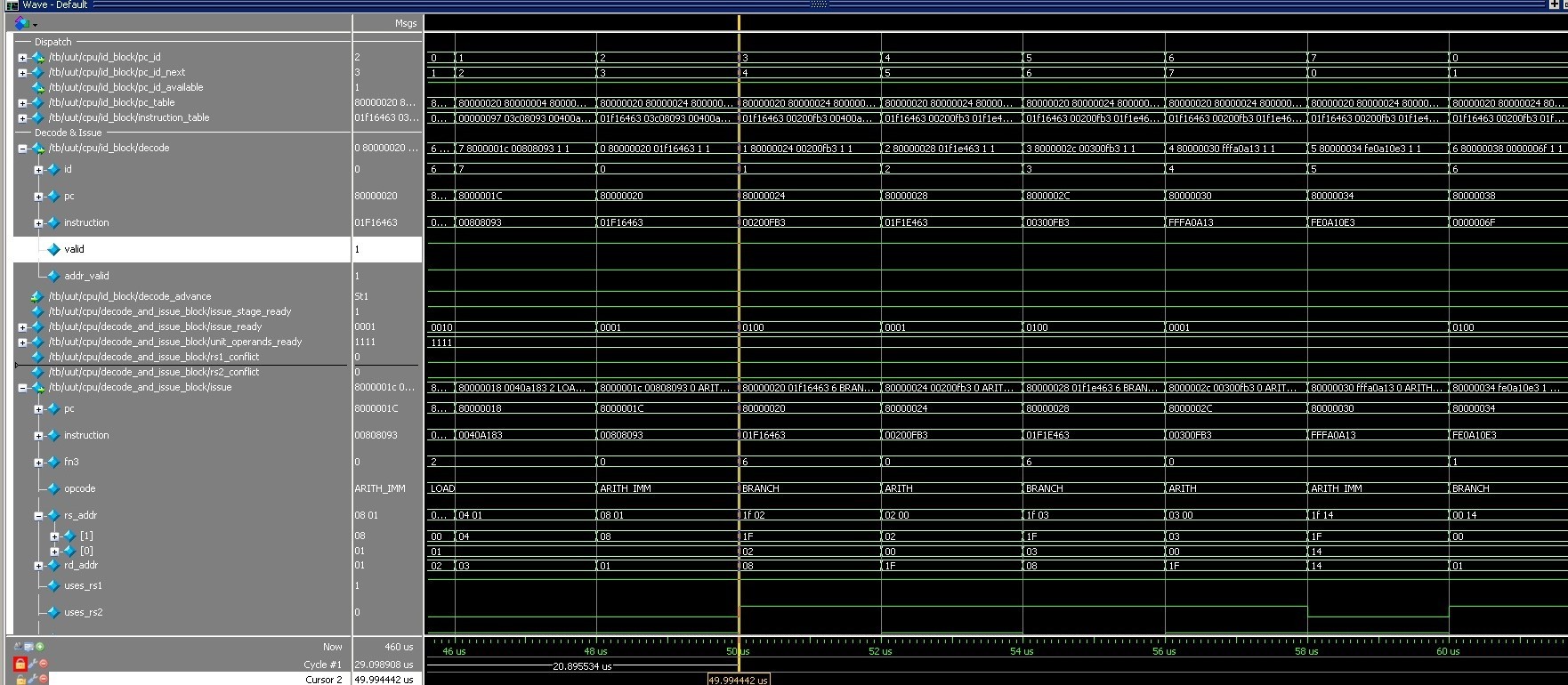


Рисунок 4.3 — Диаграмма, соответствующая этапам декодирования команды, обозначенной в тексте программы символом #!

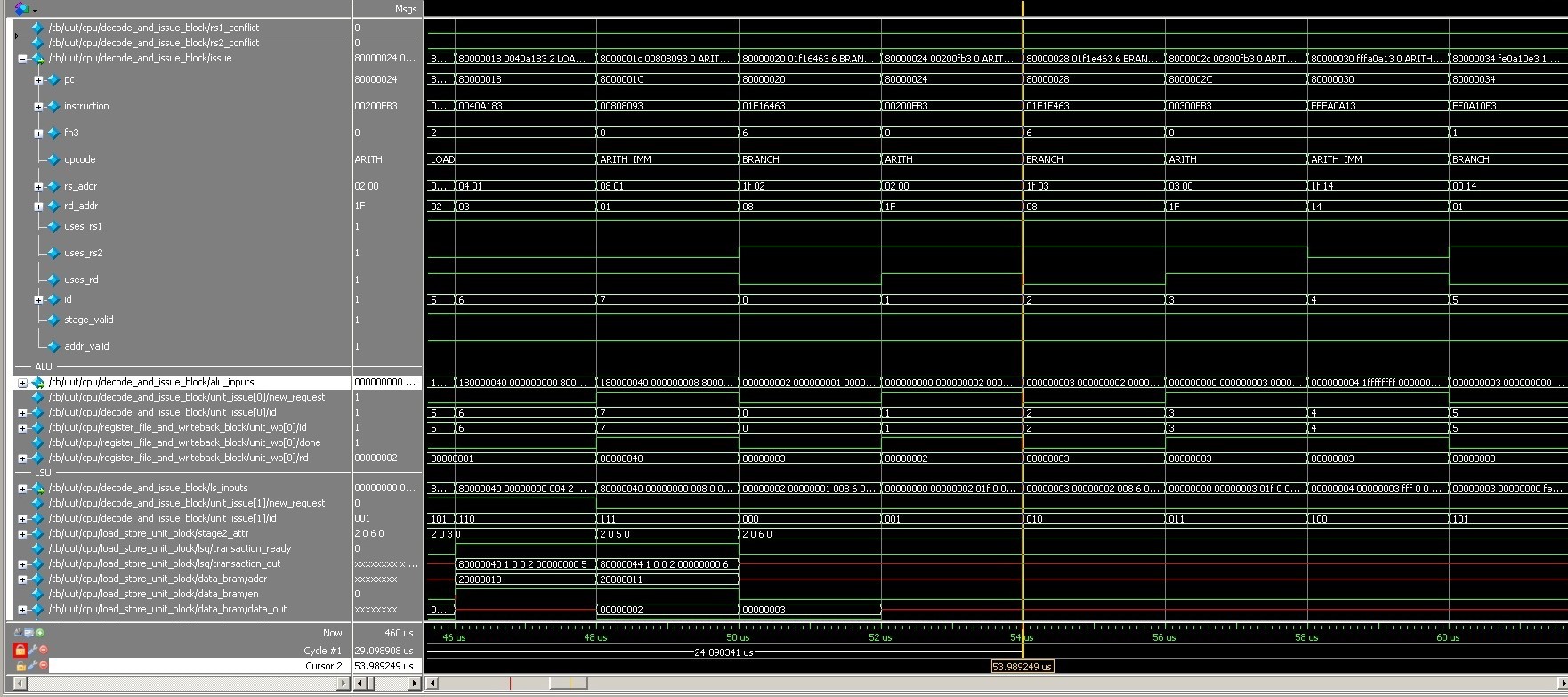


Рисунок 4.4 — Диаграмма, соответствующая этапу выполнения команды, обозначенной в тексте программы символом #!

Следующие задание: анализируя диаграмму заполнить трассу выполнения программы.

Ниже, на рисунке [4.5](#_bookmark12) приведена трасса работы программы.

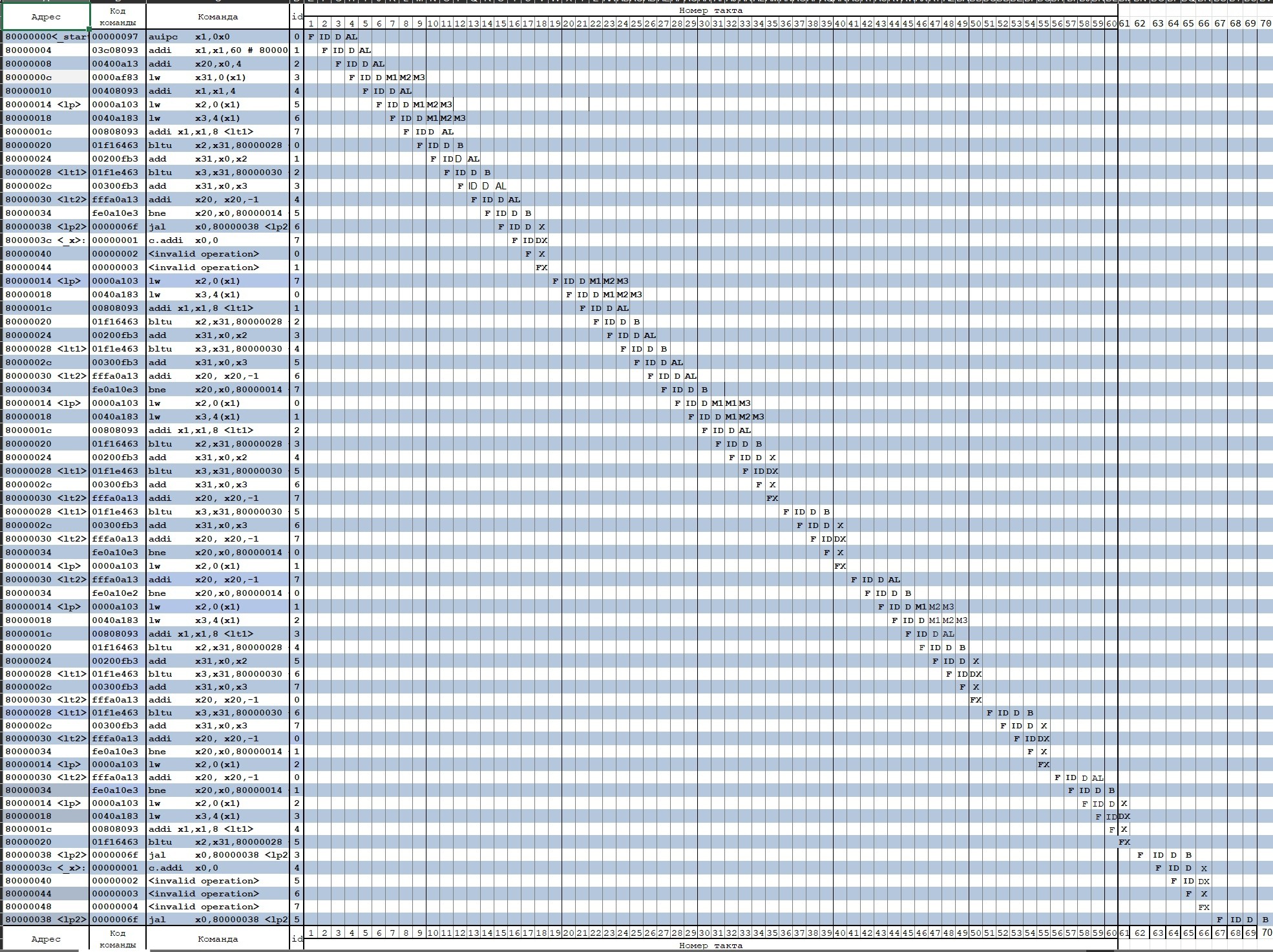


Рисунок 4.5 — Трасса работы программы

# Оптимизация

Из полученной выше трассы видно, что выполнение программы до окончания последней команды заняло 67 тактов. За время работы программы не случилось ни одного конфликта, однако случилось 7 gc\_fetch\_flush. Было принято решение убрать цикл и производить поиск максимального элемента без него. На рисунках [5.1](#_bookmark14)-[5.2](#_bookmark15), представлены код и деассемблированный листинг оптимизрованной программы.

Листинг 5.1 — Листинг оптимизрованной программы

. s e c t i o n . te xt

. g l o b l \_start ;

l e n = 9 #Ра змер ма с сива

e n r o l l = 2 #Колич е с тво обрабатыва емых элементов з а одну ит ерацию elem\_sz = 4 #Ра змер одно г о элемента ма с сива

\_start :

l a x1 , \_x

lw x31 , 0 ( x1 )

addi x1 , x1 , elem\_sz ∗1

lw x2 , 0 ( x1 ) lw x3 , 4 ( x1 ) lw x4 , 8 ( x1 ) lw x5 , 12 ( x1 ) lw x6 , 16 ( x1 ) lw x7 , 20 ( x1 ) lw x8 , 24 ( x1 ) lw x9 , 28 ( x1 )

bltu x2 , x31 , add x31 , x0 ,

l t 1 : bltu x3 , add x31 , x0 ,

bltu x4 , add x31 , x0 ,

l t 3 : bltu x5 , add x31 , x0 ,

bltu x6 , add x31 , x0 ,

bltu x7 , add x31 , x0 ,

bltu x8 ,

add x31 , x0 ,

l t 1

x2 #! x31 ,

x3 x31 ,

x4 x31 ,

x5 x31 ,

x6 x31 ,

x7 x31 ,

x8

l t 2

l t 2 :

l t 3

l t 4

l t 4 :

l t 5

l t 5 :

l t 6

l t 6 :

l t 7

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

17

|  |  |  |  |
| --- | --- | --- | --- |
| 34 | l t 7 : bltu x9 , x31 , lp 2 | | |
| 35 | add x31 , x0 , x9 | | |
| 36 | lp 2 : j lp 2 | | |
| 37 |  | | |
| 38 | . s e c t i o n . data | | |
| 39 \_x: | | . 4 byte | 0 x1 |
| 40 | | . 4 byte | 0 x2 |
| 41 | | . 4 byte | 0 x3 |
| 42 | | . 4 byte | 0 x4 |
| 43 | | . 4 byte | 0 x8 |
| 44 | | . 4 byte | 0 x6 |
| 45 | | . 4 byte | 0 x7 |
| 46 | | . 4 byte | 0 x5 |
| 47 | | . 4 byte | 0 x4 |

Листинг 5.2 — Деассемблированный листинг оптимизированной програм­ мы

1 Disassembly o f s e c t i o n . te xt :

2

3 80000000 <\_start >:

4 80000000 : 00000097 auipc x1 , 0 x0

5 80000004 : 07408093 addi x1 , x1 , 116 # 80000074 <\_x>

6 80000008 : 0000 af 83 lw x31 , 0 ( x1 )

7 8000000 c : 00408093 addi x1 , x1 , 4

8 80000010 : 0000 a103 lw x2 , 0 ( x1 )

9 80000014 : 0040 a183 lw x3 , 4 ( x1 )

10 80000018 : 0080 a203 lw x4 , 8 ( x1 )

11 8000001 c : 00 c 0 a 283 lw x5 , 1 2 ( x1 )

12 80000020 : 0100 a303 lw x6 , 1 6 ( x1 )

13 80000024 : 0140 a383 lw x7 , 2 0 ( x1 )

14 80000028 : 0180 a403 lw x8 , 2 4 ( x1 )

15 8000002 c : 01 c 0 a 483 lw x9 , 2 8 ( x1 )

16 80000030 : 01 f 16463 bltu x2 , x31 , 80000038 <lt 1 >

17 80000034 : 00200 fb 3 add x31 , x0 , x2

18

19 80000038 <lt 1 >:

20 80000038 : 01 f 1 e 463 bltu x3 , x31 , 80000040 <lt 2 >

21 8000003 c : 00300 fb 3 add x31 , x0 , x3

22

23 80000040 <lt 2 >:

24 80000040 : 01 f 26463 bltu x4 , x31 , 80000048 <lt 3 >

25 80000044 : 00400 fb 3 add x31 , x0 , x4

26

27 80000048 <lt 3 >:

28 80000048 : 01 f 2 e 463 bltu x5 , x31 , 80000050 <lt 4 >

29 8000004 c : 00500 fb 3 add x31 , x0 , x5

30

31 80000050 <lt 4 >:

32 80000050 : 01 f 36463 bltu x6 , x31 , 80000058 <lt 5 >

33 80000054 : 00600 fb 3 add x31 , x0 , x6

34

35 80000058 <lt 5 >:

36 80000058 : 01 f 3 e 463 bltu x7 , x31 , 80000060 <lt 6 >

37 8000005 c : 00700 fb 3 add x31 , x0 , x7

38

39 80000060 <lt 6 >:

40 80000060 : 01 f 46463 bltu x8 , x31 , 80000068 <lt 7 >

41 80000064 : 00800 fb 3 add x31 , x0 , x8

42

43 80000068 <lt 7 >:

44 80000068 : 01 f 4 e 463 bltu x9 , x31 , 80000070 <lp 2 >

45 8000006 c : 00900 fb 3 add x31 , x0 , x9

46

47 80000070 <lp 2 >:

48 80000070 : 0000006 f j a l x0 , 80000070 <lp 2 >

49

50 Disassembly o f s e c t i o n . data :

51

52 80000074 <\_x>:

53 80000074 : 0001 c . addi x0 , 0

54 80000076 : 0000 unimp

55 80000078 : 0002 0 x2

56 8000007 a : 0000 unimp

57 8000007 c : 00000003 lb x0 , 0 ( x0 ) # 0 <e n r o l l −0x2>

58 80000080 : 0004 c . addi 4 spn x9 , x2 , 0

59 80000082 : 0000 unimp

60 80000084 : 0008 c . addi 4 spn x10 , x2 , 0

61 80000086 : 0000 unimp

62 80000088 : 0006 0 x6

63 8000008 a : 0000 unimp

64 8000008 c : 00000007 0 x7

65 80000090 : 0005 c . addi x0 , 1

66 80000092 : 0000 unimp

67 80000094 : 0004 c . addi 4 spn x9 , x2 , 0

На рисунке 5.1 приведена трасса работы оптимизированной программы.

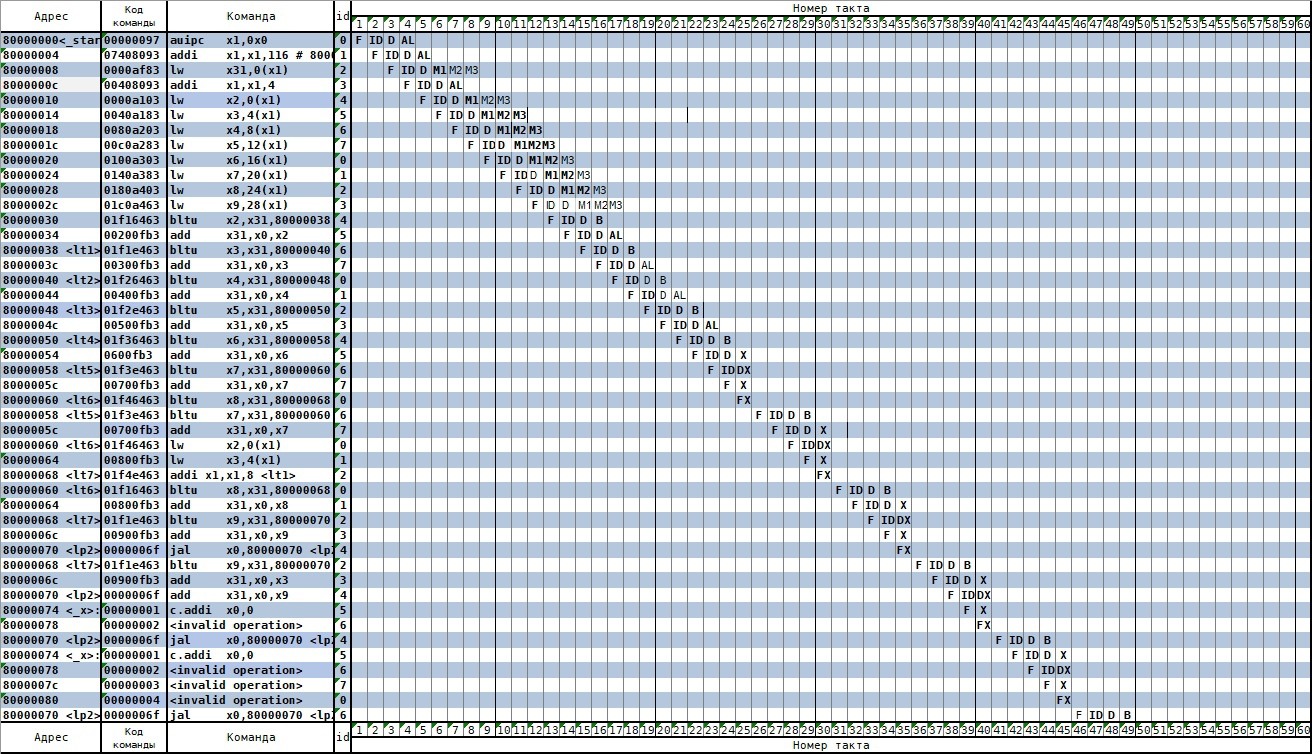


Рисунок 5.1 — Трасса работы оптимизированной программы

В итоге, можно будет уменьшить программу на 67-46 = 21 такт, то есть на 21/67 = 31будет работать быстрее.

**Вывод**

Из полученной трассы оптимизированной программы видно, что программа усеньшилась и происходит меньше gc\_fetch\_flush.

21

# ЗАКЛЮЧЕНИЕ

В данной лабораторной работе было проведено ознакомление с архитектурой ядра Taiga, а именно с порядком работы вычислительного конвейера: изучены ко­ манды RV32I, рассмотрены действия, выполняемые на каждой стадии конвейера, и данные, передаваемые между ними.

После ознакомления с теоретической стороной вопроса, был выполнен разбор этапов выполнения программы на симуляции процессора с набором инструкций RV32I. После ее анализа удалось провести оптимизацию.

В итоге, теоретические знания о порядке исполнения программ на процессорах с RISC архитектурой были закреплены на практике.

Таким образом все поставленные задачи решены, основная цель работы достигнута.

22