СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc133281500)

[1. РАЗРАБОТКА ОБЩЕЙ СТРУКТУРЫ МИКРО-ЭВМ 6](#_Toc133281501)

[1.1. Функциональный состав микро – ЭВМ 6](#_Toc133281502)

[1.2. Разработка системы команд 7](#_Toc133281503)

[1.3. Описание взаимодействия всех блоков микро-ЭВМ при выполнении команд программы 8](#_Toc133281504)

[2. РАЗРАБОТКА ОСНОВНЫХ УСТРОЙСТВ МИКРО-ЭВМ 9](#_Toc133281505)

[2.1. Блоки памяти ПЗУ и ОЗУ 9](#_Toc133281506)

[2.2. Блок стека 10](#_Toc133281507)

[2.3. Блок регистров общего назначения 12](#_Toc133281508)

[2.4. Блок АЛУ 13](#_Toc133281509)

[2.4.1 Операция ADDC 14](#_Toc133281510)

[2.4.2 Операция NOT 14](#_Toc133281511)

[2.4.3 Операция OR 15](#_Toc133281512)

[2.4.4 Операция SRA 15](#_Toc133281513)

[2.5 Устройство управления 15](#_Toc133281514)

[3. ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ 17](#_Toc133281515)

[ЗАКЛЮЧЕНИЕ 26](#_Toc133281516)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc133281517)

# **ВВЕДЕНИЕ**

Целью данного курсового проекта является разработка микро-ЭВМ на ПЛИС с заданными по варианту свойствами. Одним из таких свойств является тип архитектуры. Разрабатываемое в данном курсовом проекте устройство будет иметь Гарвардскую архитектуру. Данный тип архитектуры предусматривает отдельные адресные пространства для команд и данных, в то время как Принстонская архитектура предусматривает наличие общего адресного пространства.

В разрабатываемом устройстве ПЗУ предназначено для хранения команд, а ОЗУ – для хранения данных.

ЦП проектируемой микро-ЭВМ является главным устройством, благодаря которому будет функционировать проектируемая микро-ЭВМ. Он будет включать в себя следующие устройства и блоки:

– АЛУ;

– блок РОН;

– стековое устройство;

– массив служебных регистров (в их числе и регистры флагов).

Для проверки работы разработанной микро-ЭВМ будет использована микропрограмма, включающая в себя все команды разработанной архитектуры микро-ЭВМ. Разработка данного курсового проекта осуществляется с помощью программы Altera Quartus II 9.1.

# **РАЗРАБОТКА ОБЩЕЙ СТРУКТУРЫ МИКРО-ЭВМ**

## **Функциональный состав микро – ЭВМ**

Разрабатываемая микро-ЭВМ будет иметь Гарвардскую архитектуру, одним из принципов которой разделение памяти команд и памяти данных: команды и данные хранятся в разных блоках памяти. Также необходимо разработать блок регистров общего назначения, стек, арифметико-логическое устройство, блок управления. Также необходимо будет реализовать дополнительные блоки для корректной работы устройства: блок регистров флагов.

Исходя из блоков, которые перечислены выше, было принято решение разработать такие функциональный блоки:

1. Блок памяти данных.
2. Блок памяти команд.
3. Блок регистров общего назначения.
4. Блок арифметико-логического устройства.
5. Стек.
6. Блок управления.

Структурная схема представлена в приложении ГУИР.400201.314 С1. Блок управления отправляет адрес в ПЗУ и принимает инструкции для команд. При командах mov блок управления посылает сигналы в ОЗУ и в РОН, данные пересылаются между блоками ОЗУ и РОН. При командах push, pop блок управления посылает сигналы на блоки стек и РОН. Данные блоки обмениваются данными. При командах АЛУ устройство управления посылает сигналы в РОН и ОЗУ, чтобы получить операнды, после чего вместе с сигналом нужной команды отправляет их в блок АЛУ. Результат команды АЛУ отправляется в ОЗУ или в РОН, в зависимости от адресации операции.

ПЗУ и ОЗУ находятся в разных адресных пространствах так как устройство разрабатывается в соответствии с Гарвардской архитектурой.

ЦП проектируемой микро-ЭВМ обеспечивает выборку команды и ее декодирование, а также отвечает за определение текущего исполнительного адреса и выполнение очередной команды, тем самым координируя работу всего устройства. Также в состав данного блока входит блок регистров общего назначения, используемых для хранения данных, АЛУ, стек.

Арифметико-логическое устройство используется для выполнения арифметических, логических и сдвиговых команд. Результатом выполнения команды на АЛУ может являться как результирующее значение, выставляемое на шину данных, так и установка соответствующих флагов.

Стек используется для сохранения в нем значений, которые впоследствии могут быть считаны в порядке, обратном порядку их занесения в стек. Доступ к стеку происходит при выполнении команд PUSH или POP. После выполнения любой из данных команд указатель стека изменяет свое значение.

## **Разработка системы команд**

В ходе проектирования микро-ЭВМ использовалась архитектура системы команд (далее АСК) представленная в таблице 1.1. Для кодирования команды требуется 2 ячейки памяти по 11 бит.

Таблица 1.1 - Структура команды микро-ЭВМ.

|  |  |  |  |
| --- | --- | --- | --- |
| Первое слово команды | | | |
| 10-7 | 6-3 | | 2-0 |
| Код операции | Адрес регистра 1 | | Безразличные биты |
| Второе слово команды | | | |
| 10-4 | | 3-0 | |
| Адрес памяти | | | |
| Безразличные биты | | Адрес регистра 2 | |

Старшие 4 бита первого слова команды содержат код команды, биты с 6 по 3 первого содержат адрес первого регистра, с 2 по 0 бит – безразличные биты, биты с 0 по 10 второго слова ­­– адрес второго операнда, это может быть адрес памяти (с 0 по 10 бит) либо адрес второго регистра (с 0 по 3 бит).

Таким образом, система команд имеет следующий вид (см. таблицу 1.2).

Таблица 1.2 – Коды операций команд микро-ЭВМ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Мнемоническая запись | Слово 1 | | | Слово 2 | |
| MOV adr, reg | 10-7 | 6-3 | 2-0 | 10-0 | |
| 0000 | reg | x | adr | |
| MOV reg, adr | 10-7 | 6-3 | 2-0 | 10-0 | |
| 0001 | reg | x | adr | |
| PUSH reg | 10-7 | 6-3 | 2-0 | 10-0 | |
| 0010 | reg | x | x | |
| POP reg | 10-7 | 6-3 | 2-0 | 10-0 | |
| 0011 | reg | x | x | |
| JMP adr | 10-7 | 6-0 | | 10-0 | |
| 0100 | x | | adr | |
| JAZ adr | 10-7 | 6-0 | | 10-0 | |
| 0101 | x | | adr | |
| ADDC reg1, reg2 | 10-7 | 6-3 | 2-0 | 10-4 | 3-0 |
| 0110 | reg1 | x | x | reg2 |
| NOT reg | 10-7 | 6-3 | 2-0 | 10-0 | |
| 0111 | reg | x | x | |
| OR reg1, reg2 | 10-7 | 6-3 | 2-0 | 10-4 | 3-0 |
| 1000 | reg1 | x | x | reg2 |
| SRA reg1, reg2 | 10-7 | 6-3 | 2-0 | 10-4 | 3-0 |

Продолжение таблицы 1.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1001 | reg1 | | x | x | reg2 |
| ADDC adr, reg | 10-7 | 6-3 | | 2-0 | 10-0 | |
| 1010 | reg | | x | adr | |
| NOT adr | 10-7 | 6-3 | | 2-0 | 10-0 | |
| 1011 | x | | x | adr | |
| OR adr, reg | 10-7 | 6-3 | | 2-0 | 10-0 | |
| 1100 | reg | | x | adr | |
| SRA adr, reg | 10-7 | 6-3 | | 2-0 | 10-0 | |
| 1101 | reg1 | | x | adr | |
| MOV reg1, reg2 | 10-7 | 6-3 | | 2-0 | 10-4 | 3-0 |
| 1110 | reg1 | | x | x | reg2 |
| HLT | 10-7 | | 6-0 | | 10-0 | |
| 1111 | | x | | x | |

## **Описание взаимодействия всех блоков микро-ЭВМ при выполнении команд программы**

Основную работу устройства выполняет блок устройства управления. Он управляет работой всех устройств и организовывает выполнение команд, представленных в таблице 1.2 посредством блока управления.

Взаимодействие начинается со считывания команды. В ЦП находится регистр с текущим адресом считываемой команды, значение которого инкрементируется при каждом последующем считывании команды из памяти. Команды JMP, JAZ предполагают загрузку в данный регистр нового значения, передаваемого в качестве операнда в составе команды. Гарвардская архитектура разрабатываемой микро-ЭВМ предполагает наличие различных адресных пространств для команд и данных. Размер считываемой команды зависит от операции и может быть равен одному слову или двум словам, а размер шины данных – 11 бит. На первом такте происходит декодирование считанной команды. Из команды выделяется код операции и в зависимости от него выполняются переходы по этапам выполнения команды.

Блок памяти команд отвечает за хранение команд устройства.

Блок памяти данных отвечает за хранение данных устройства.

Регистры общего назначения и стек являются временными хранилищами данных при выполнении микропрограммы устройства.

Блок арифметико-логических операций отвечает за выполнение арифметических и логических команд реализуемых в данной микро-ЭВМ.

Блок регистров флагов отвечает за хранение регистров флагов, которые необходимы при выполнении команд устройства и условного перехода.

# **РАЗРАБОТКА ОСНОВНЫХ УСТРОЙСТВ МИКРО-ЭВМ**

В данном разделе описаны устройства микро-ЭВМ, такие как ПЗУ, ОЗУ, стек, РОН, АЛУ.

## **Блоки памяти ПЗУ и ОЗУ**

Было принято решение использовать имеющиеся блоки lmp\_ram\_dq и lpm\_rom. Блоки памяти должны быть асинхронными по условию задания.



Рис. 2.1. Условное графическое обозначение блока памяти команд

Входные пины:

* Addr[10..0] – адрес ПЗУ для чтения;
* MainBus[1..0] – управляющие сигналы.

Выходной пин:

* Out[10..0] – данные, которые поступают с ПЗУ на шину команд.



Рис. 2.2. Условное графическое обозначение блока памяти данных

Входные пины:

* Addr[10..0] – адрес ПЗУ для чтения;
* MainBus[2..0] – управляющие сигналы;
* Data[10..0] – данные, поступающие в ОЗУ для записи.

Выходной пин:

* Out[10..0] – данные, которые поступают с ОЗУ на шину команд.

Так как по условию блоки асинхронные, запись в блок RAM происходит в такт прихода сигнала записи, чтение происходит с задержкой в пол такта.

Схемы блоков памяти представлена в приложениях ГУИР.400201.314 Э1, ГУИР.400201.314 Э2.

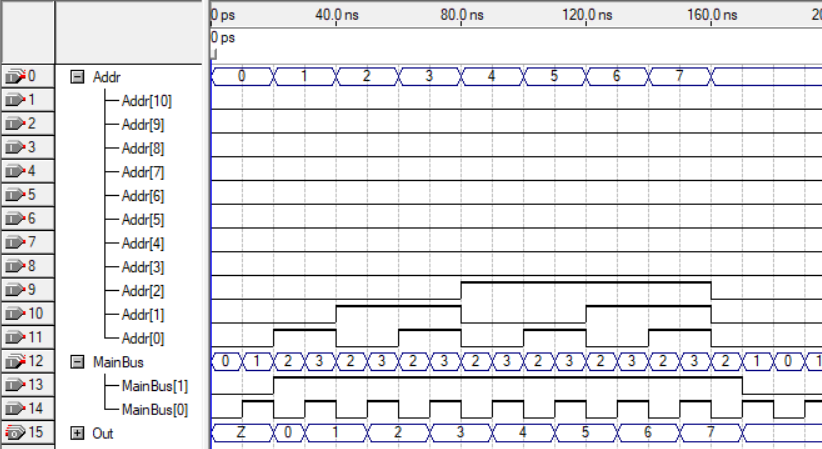


Рис. 2.3. Временная диаграмма блока ПЗУ

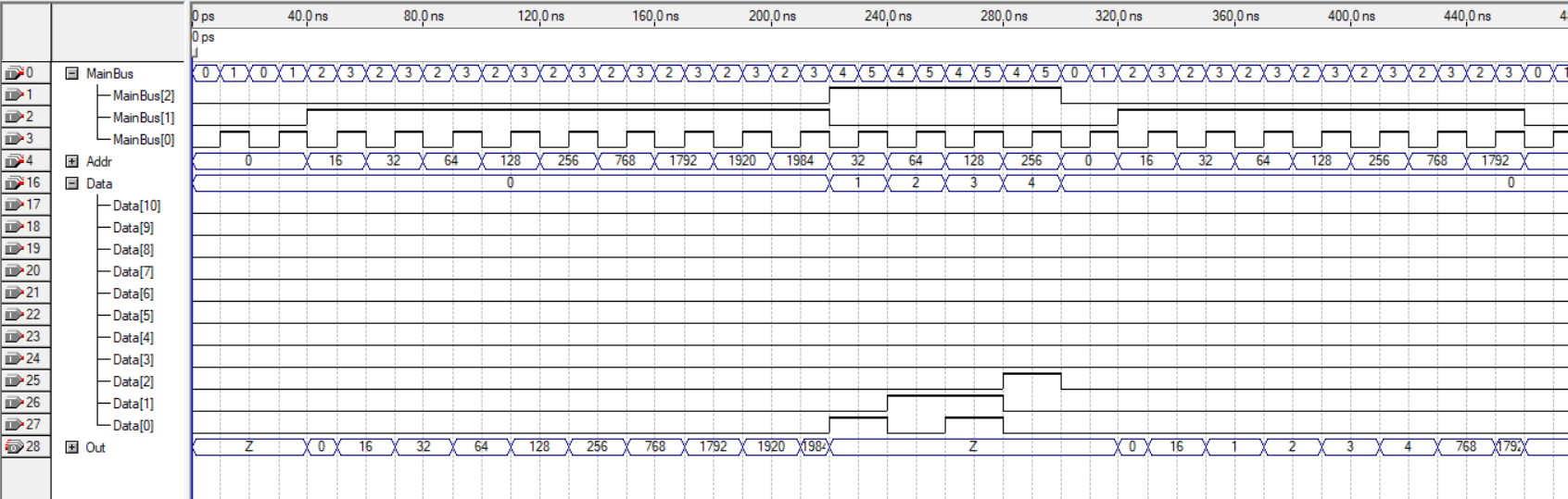


Рис. 2.4. Временная диаграмма блока ОЗУ

## **Блок стека**

Блок стек имеет 12 регистров и направление вниз, по условию задания.

Для работы со стеком предусмотрены 2 сигнала — push и pop. По этим сигналам стек обновляет свой SP (Stack Pointer) и записывает или выдаёт значение соответственно.



Рис. 2.5. Условное графическое обозначение блока стека

Входные пины:

* MainBus[2..0] – управляющие сигналы;
* IN[10..0] – данные, которые будут записаны в вершину при команде push.

Выходной пин:

* OUT[10..0] - данные, которые будут выданы при команде pop.

Указатель вершины имеет направление вниз - при увеличении количества элементов стека адрес вершины стека уменьшается. Само устройство работает 1 такт, после подачи сигнала push/pop.

Для того чтобы можно было корректно прочитать данные и стереть их с регистра, после команды pop, или же успешно записать в вершину стека новые данные и после переместить её на следующий элемент cуществует проверка при подаче команды pop, если стек пустой, то указатель не увеличивается, если поступает команда push, когда установился флаг overflow, то указатель не уменьшается и запись новых данных не производится.

Второй счетчик отвечает за перемещение указателя вершины стека. Схема блока стека представлена в приложении ГУИР.400201.314 Э3.

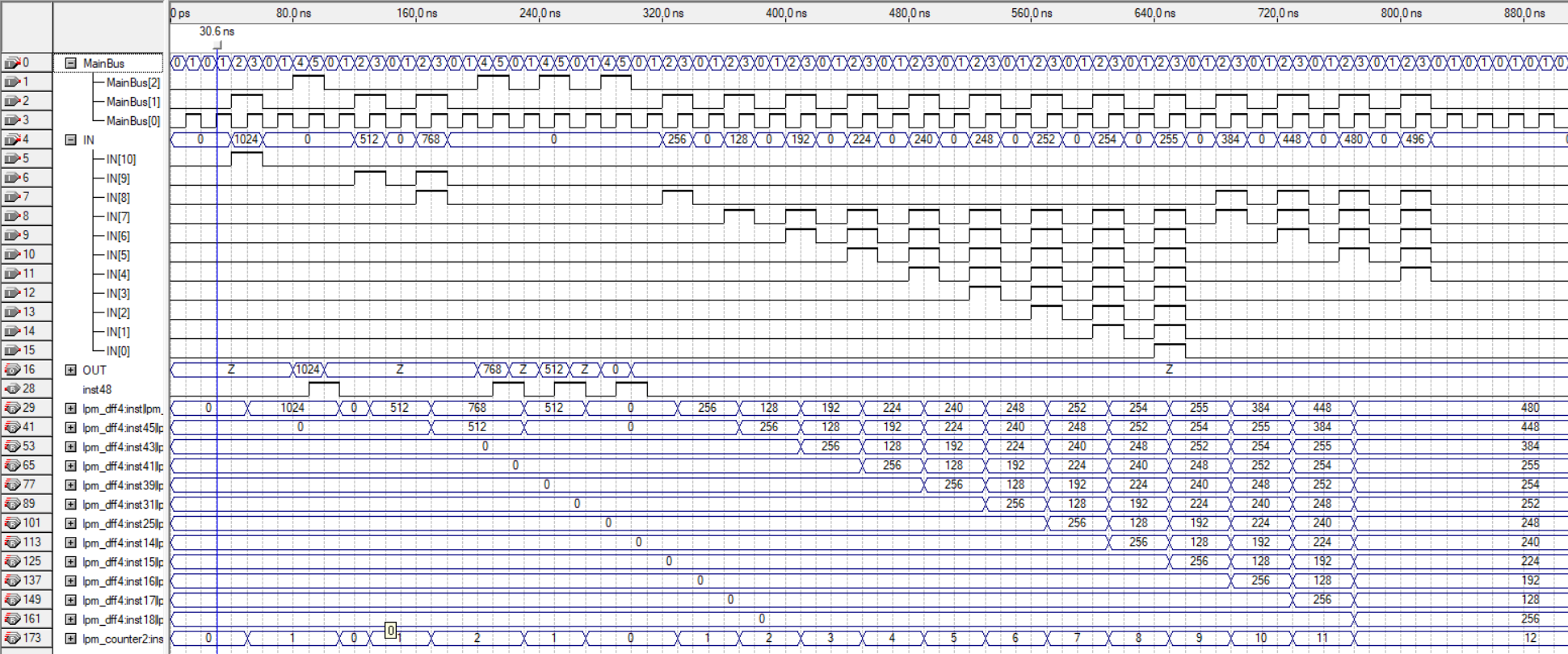


Рис. 2.6. Временная диаграмма блока стека

## **Блок регистров общего назначения**

Блок регистров общего назначения (РОН) используется для записи / чтения операндов, которые впоследствии могут использоваться для вычисления операций, либо записываться в другие регистры или память.

Сам РОН представляет из себя 12 регистров размером по 11 бит.



Рис. 2.7. Условное графическое обозначение блока регистров общего назначения

Входные пины:

* clk – тактирующий сигнал устройства;
* write – сигнал для записи в регистр, адрес которого подан на входной пин ADR[3..0];
* read - сигнал для чтения с регистра, адрес которого подан на входной пин address[2..0];
* ADR[3..0] – пин, на вход которого подается адрес выбранного регистра;
* write\_data[10..0] – данные, которые будут записаны в ячейку памяти, если подан сигнал write.

Выходные пины:

* OUT[10..0] – данные, которые будут выданы на общую шину данных, если был подан сигнал read.

Принцип работы: в блок поступает сигнал read, на мультиплексоре выбирается регистр с учетом пришедшего в данный момент адреса регистра по шине ADR[3..0], и информация попадает на выход устройства.

В блок приходит сигнал write, на дешифраторе появляется сигнал enable, который позволяет выбрать один из выходов дешифратора установить в единицу и в зависимости от того, какой выход активен, в тот регистр и будут записаны данные, которые поступают с шины write\_data[10..0].



Рис. 2.8. Условное графическое обозначение блока регистра

Схема блока регистров общего назначения представлена в приложении ГУИР.400201.314 Э4.

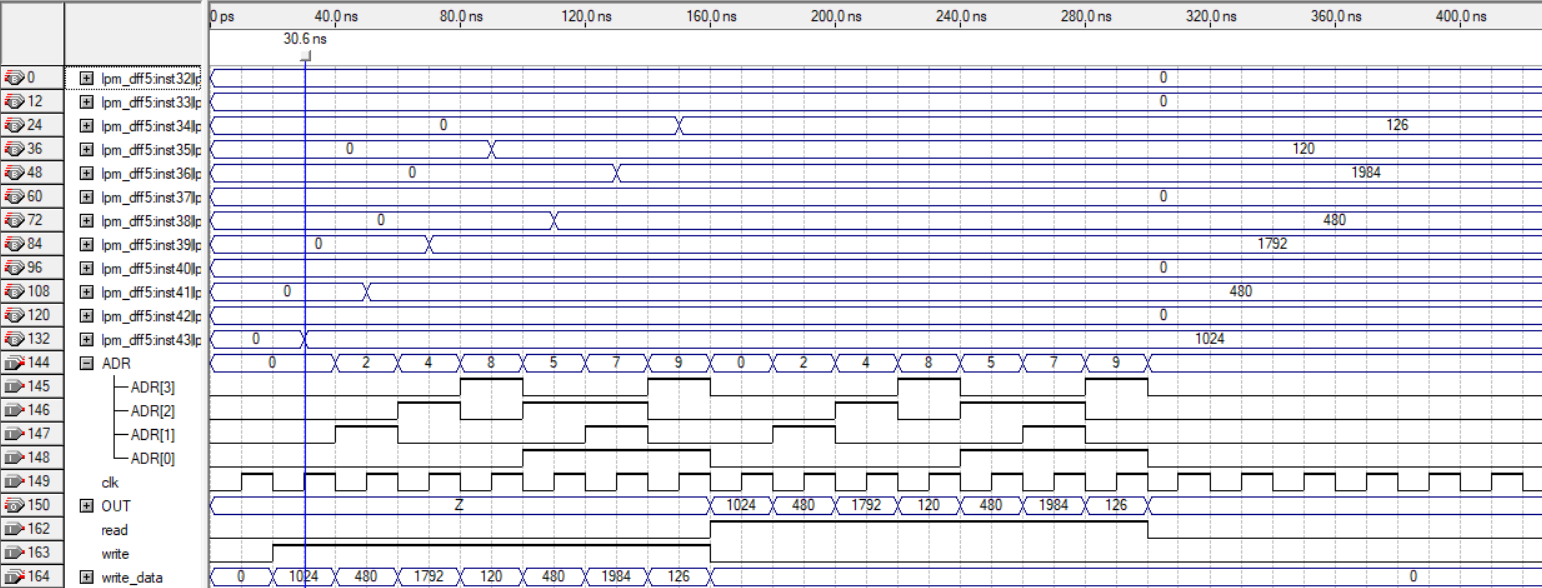


Рис. 2.9. Временная диаграмма блока РОН

## **Блок АЛУ**

Блок арифметико-логического устройства должен обязательно уметь выполнять команды:

* ADDC (сложение с учетом переноса) – сложение с учетом флага переноса (CF)
* NOT – инверсия
* OR – логическое «ИЛИ»
* SRA (Shift Right Arithmetic) – арифметический сдвиг вправо



Рис. 2.10. Условное графическое обозначение блока арифметико-логического устройства

Входные пины:

* operand1[10..0] – первый операнд;
* operand2[10..0] – второй операнд или количество бит для сдвига в SRA;
* commands[5..0] – управляющие сигналы.

Выходные пины:

* ZFlag – вывод флага нуля;
* CFlag – вывод флага переноса;
* OUT[10..0] – выходные данные.

### **Операция ADDC**

Этакомандавыполняет сложение с переносом, а именно содержимое 11 бит источника и 11 бит аккумулятора с учетом флага переноса.

На рисунке 2.11. представлено условно-графическое обозначение команды ADDC.



Рис. 2.11. Условное графическое обозначение блока команды ADDC

### **Операция NOT**

Инвертирует присланное значение. Схема блока реализации этой команды представлен на рисунке 2.12.



Рис. 2.12. Условное графическое обозначение блока команды NOT

### **Операция OR**

Выполняет побитовое «ИЛИ» между значениями operand1[10..0] и operand2[10..0].



Рис. 2.13. Условное графическое обозначение блока команды OR

### **Операция SRA**

Выполняет арифметический сдвиг вправо на заданное количество бит. В операции участвует 4 младших бита второго операнда и весь первый операнд.



Рис. 2.14. Условное графическое обозначение блока команды SRA

Схема блока арифметико-логического устройства представлена в приложении ГУИР.400201.314 Э5.

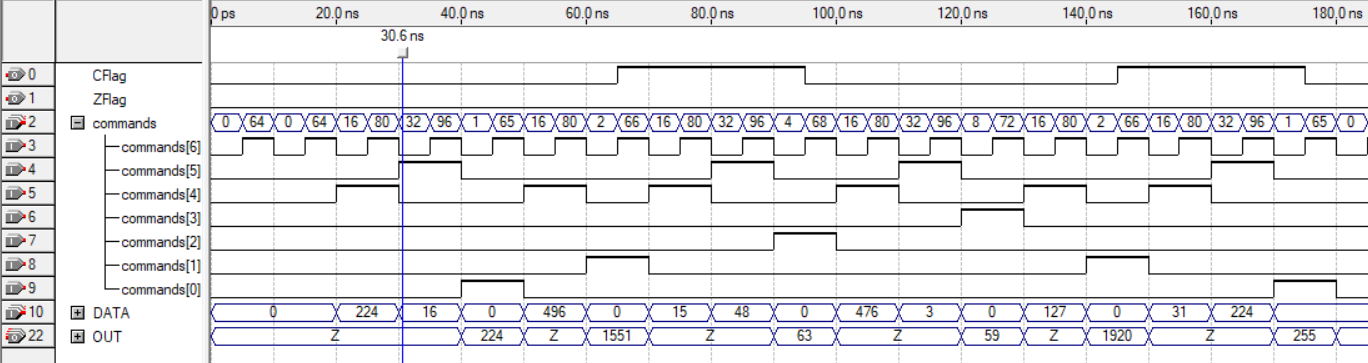


Рис. 2.15. Временная диаграмма блока АЛУ

## **Устройство управления**

Электрическая схема блока устройства управления представлена в приложении ГУИР.400201.314 Э6.

Одной из составных частей устройства управления является блок выборки и декодирования команд.

После считывания слова из ПЗУ его значение инкрементируется. При выполнении команды JMP, JAZ значение операнда, полученное из самой команды, записывается в счетчик команд.

Размер команды составляет 22 бита, поэтому осуществляется последовательное чтение 2 слов из ПЗУ. Для всех команд выделяется значение РОН из первого и второго слов команды, даже если данное значение впоследствии не анализируется (для команд JMP и JAZ, HLT). К моменту декодирования второго слова команды уже известен код операции. Далее происходит выполнение выбранной команды, а именно, считывание данных из регистров, ОЗУ, стека, либо запись в соответствующие блоки, в зависимости от поступившей операции.

Таким образом, устройство управления осуществляет выборку команды и ее декодирование, а также отвечает за координацию выполнения команд и обеспечивает взаимодействие с блоком РОН. Условно-графическое обозначение устройства управления представлено на рисунке 2.16.



Рисунок 2.16 – Условно-графическое обозначение устройства управления

Устройство управления имеет следующие входы и выходы:

1. CLK – тактовый сигнал;
2. CARRY\_FLAG – значение флага Carry, поступающее с АЛУ;
3. ZERO\_FLAG – значение флага Zero, поступающее с АЛУ;
4. DATA[10..0] – входная шина данных;
5. DATA\_COMMAND[10..0] – входная шина команд;
6. DATA\_OUT[10..0] – выходная шина данных;
7. ADDRESS\_RAM[10..0] – выход на шину адреса данных;
8. ADDRESS\_ROM[10..0] – выход на шину адреса команд;
9. MAIN\_BUS\_RAM[2..0] – шина управления ОЗУ;
10. MAIN\_BUS\_ROM[1..0] – шина управления ПЗУ;
11. MAIN\_BUS\_STACK[2..0] – шина управления стека;
12. MAIN\_BUS\_ALU[6..0] – шина управления АЛУ;
13. MAIN\_BUS\_RON[6..0] – шина управления РОН.

# **ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ**

Общая схема разработанной микро-ЭВМ представлена в приложении ГУИР.400201.314 Э7. Для общего моделирования системы была разработана следующая микропрограмма, включающая в себя все команды разработанной микро-ЭВМ.

Таблица 3.1 – Микропрограмма для моделирования

|  |  |  |  |
| --- | --- | --- | --- |
| Номер ячейки ПЗУ | Запись на языке ассемблера | Запись в бинарном виде | Запись в шестнадцатеричном виде |
| 0 | Mov reg0 adr1 | 00010000000 | 080 |
| 1 | 00000000001 | 001 |
| 2 | Mov reg1 adr8 | 00010001000 | 088 |
| 3 | 00000001000 | 008 |
| 4 | Mov reg2 adr9 | 00010010000 | 090 |
| 5 | 00000001001 | 009 |
| 6 | Mov reg3 adr10 | 00010011000 | 098 |
| 7 | 00000001010 | 00A |
| 8 | Mov reg4 adr11 | 00010100000 | 0A0 |
| 9 | 00000001011 | 00B |
| 10 | Mov reg5 adr12 | 00010101000 | 0A8 |
| 11 | 00000001100 | 00C |
| 12 | Mov reg6 adr13 | 00010110000 | 0B0 |
| 13 | 00000001101 | 00D |
| 14 | Mov reg7 adr14 | 00010111000 | 0B8 |
| 15 | 00000001110 | 00E |
| 16 | Mov reg8 adr15 | 00011000000 | 0C0 |
| 17 | 00000001111 | 00F |
| 18 | Mov reg9 adr32 | 00011001000 | 0C8 |
| 19 | 00000100000 | 020 |
| 20 | Mov reg10 adr33 | 00011010000 | 0D0 |
| 21 | 00000100001 | 021 |
| 22 | Mov reg11 adr34 | 00011011000 | 0D8 |
| 23 | 00000100010 | 022 |
| 24 | Mov adr31 reg1 | 00000001000 | 008 |
| 25 | 00000011111 | 01F |
| 26 | Push reg8 | 00101000000 | 140 |
| 27 | 00000000000 | 000 |
| 28 | Pop reg9 | 00111001000 | 1C8 |
| 29 | 00000000000 | 000 |
| 30 | Jmp 256 | 01000000000 | 200 |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| 31 |  | 00100000000 | 100 |
| 256 | Not reg5 | 01110101000 | 3A8 |
| 257 | 00000000000 | 000 |
| 258 | Addc reg6 reg10 | 01100110000 | 330 |
| 259 | 00000001010 | 00A |
| 260 | Or reg11 reg1 | 10001011000 | 458 |
| 261 | 00000000001 | 001 |
| 262 | Sra reg9 reg0 | 10011001000 | 4C8 |
| 263 | 00000000000 | 000 |
| 264 | Jaz 100 | 01010000000 | 280 |
| 265 | 00001100100 | 064 |
| 100 | Not adr20 | 10110000000 | 580 |
| 101 | 00000010100 | 014 |
| 102 | Addc reg3 adr30 | 10100011000 | 518 |
| 103 | 00000011110 | 01E |
| 104 | Or reg2 adr55 | 11000010000 | 610 |
| 105 | 00000110111 | 037 |
| 106 | Sra reg7 adr2 | 11010111000 | 6B8 |
| 107 | 00000000010 | 002 |
| 108 | Mov reg4 reg5 | 11100100000 | 720 |
| 109 | 00000000101 | 005 |
| 110 | Hlt | 11110000000 | 780 |
| 111 | 00000000000 | 000 |

Дамп памяти ПЗУ представлен на рисунке 3.1, а дампы памяти ОЗУ до и после моделирования – на рисунке 3.2.

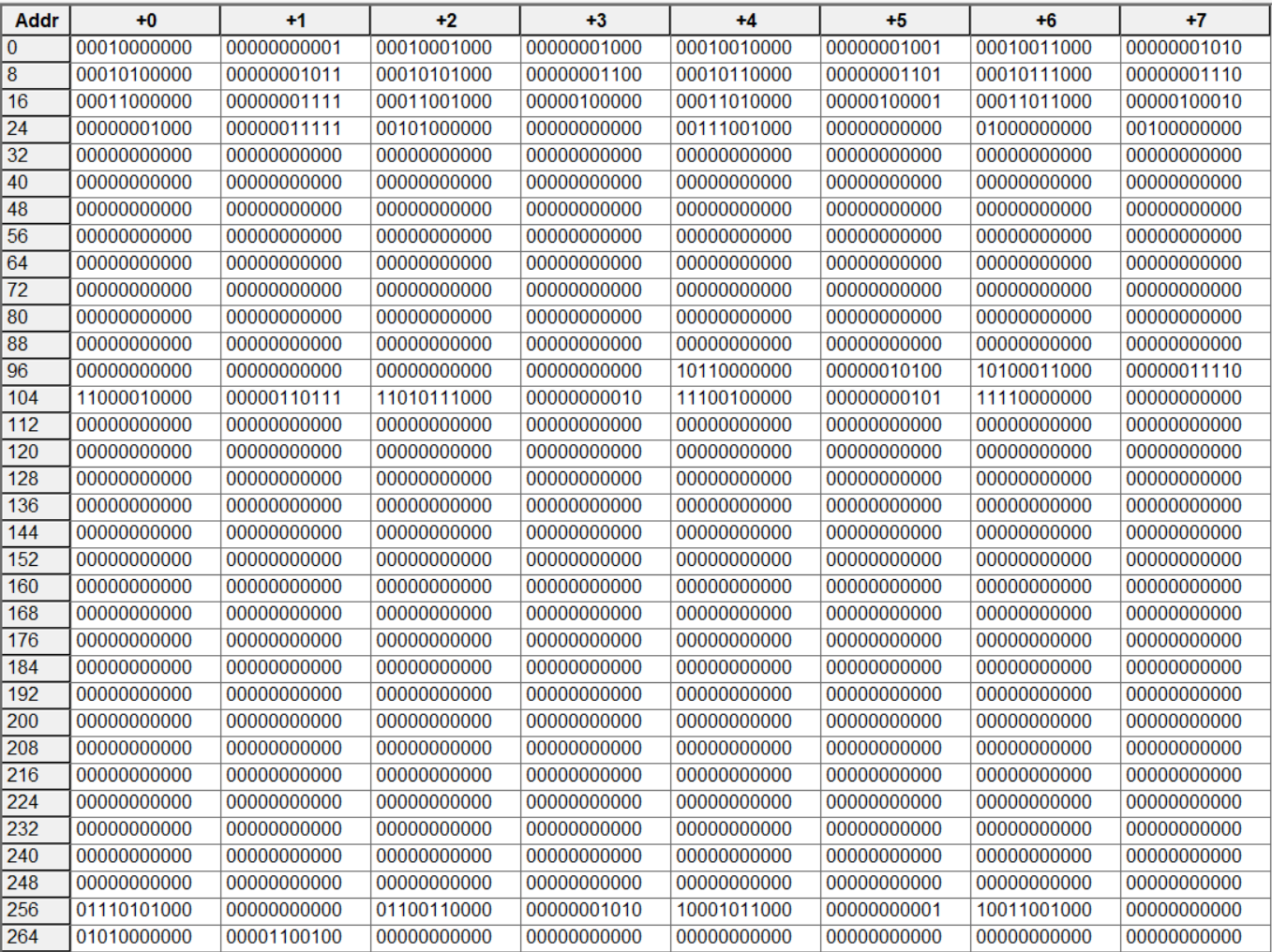


Рисунок 3.1 – Дамп памяти ПЗУ

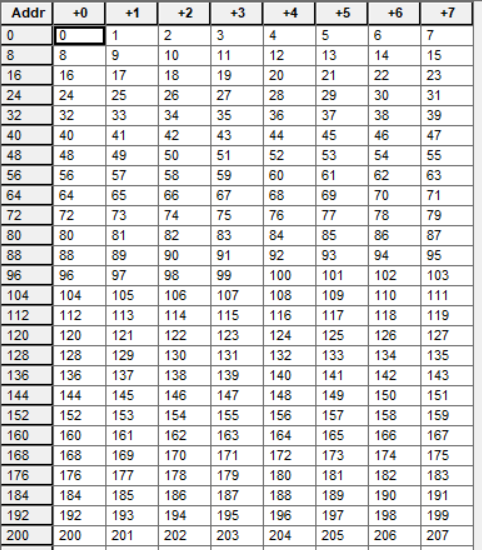
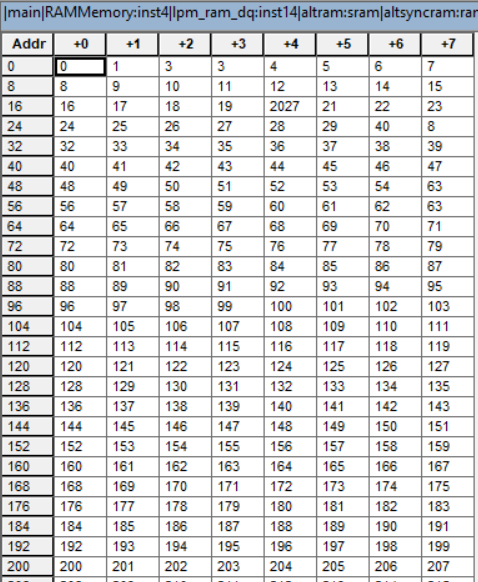
 

Рисунок 3.2 – Дампы памяти ОЗУ: a – до моделирования; б – после моделирования

На следующих рисунках изображены фрагменты общей временной диаграммы, отражающие выполнение команд микропрограммы.



Рисунок 3.3 – Общее моделирование: команды mov reg0 adr1, mov reg1 adr8, mov reg2 adr9, mov reg3 adr10, mov reg4 adr11

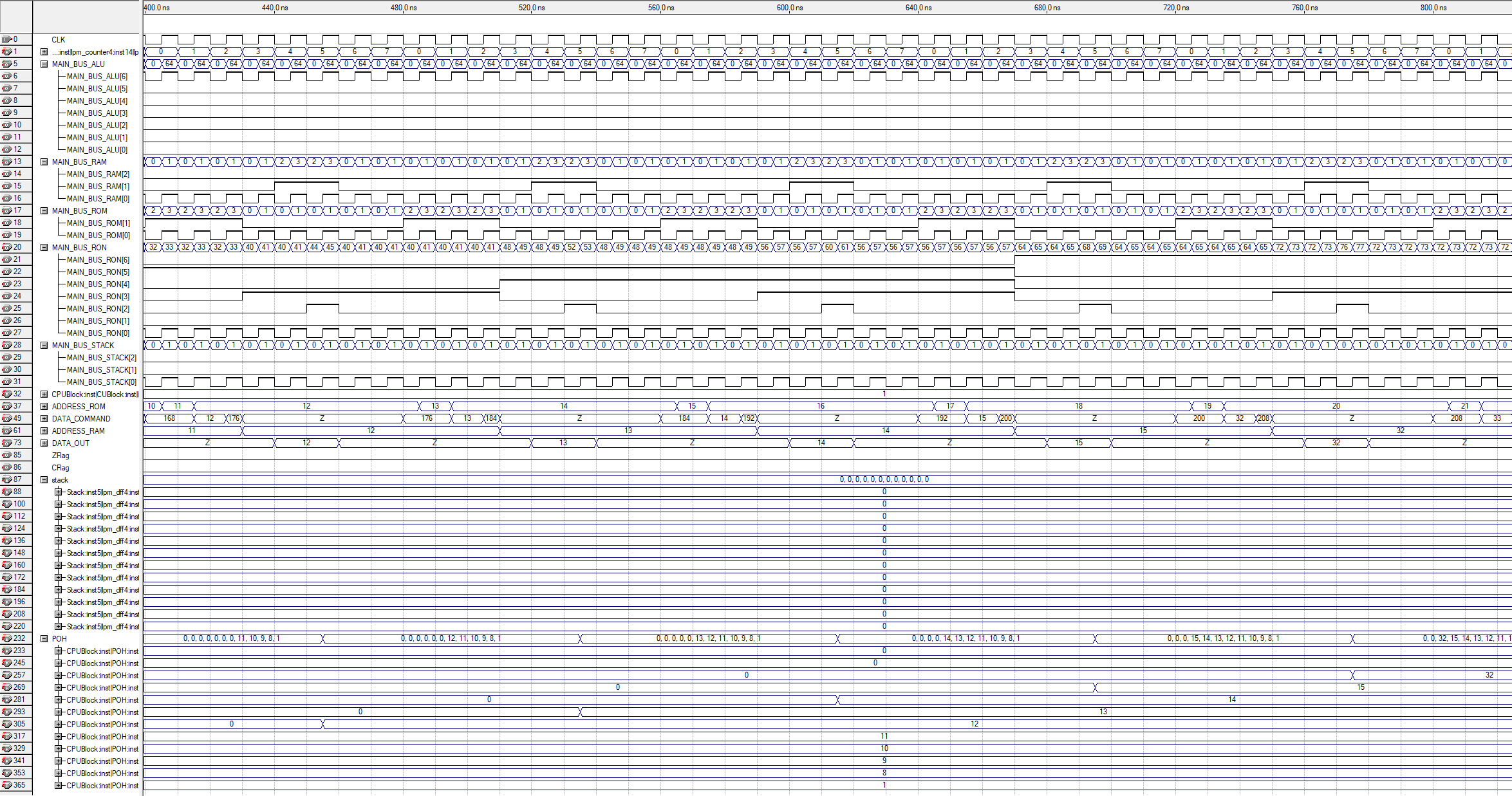


Рисунок 3.4 – Общее моделирование: команды mov reg5 adr12, mov reg6 adr13, mov reg7 adr14, mov reg8 adr15, mov reg9 adr32

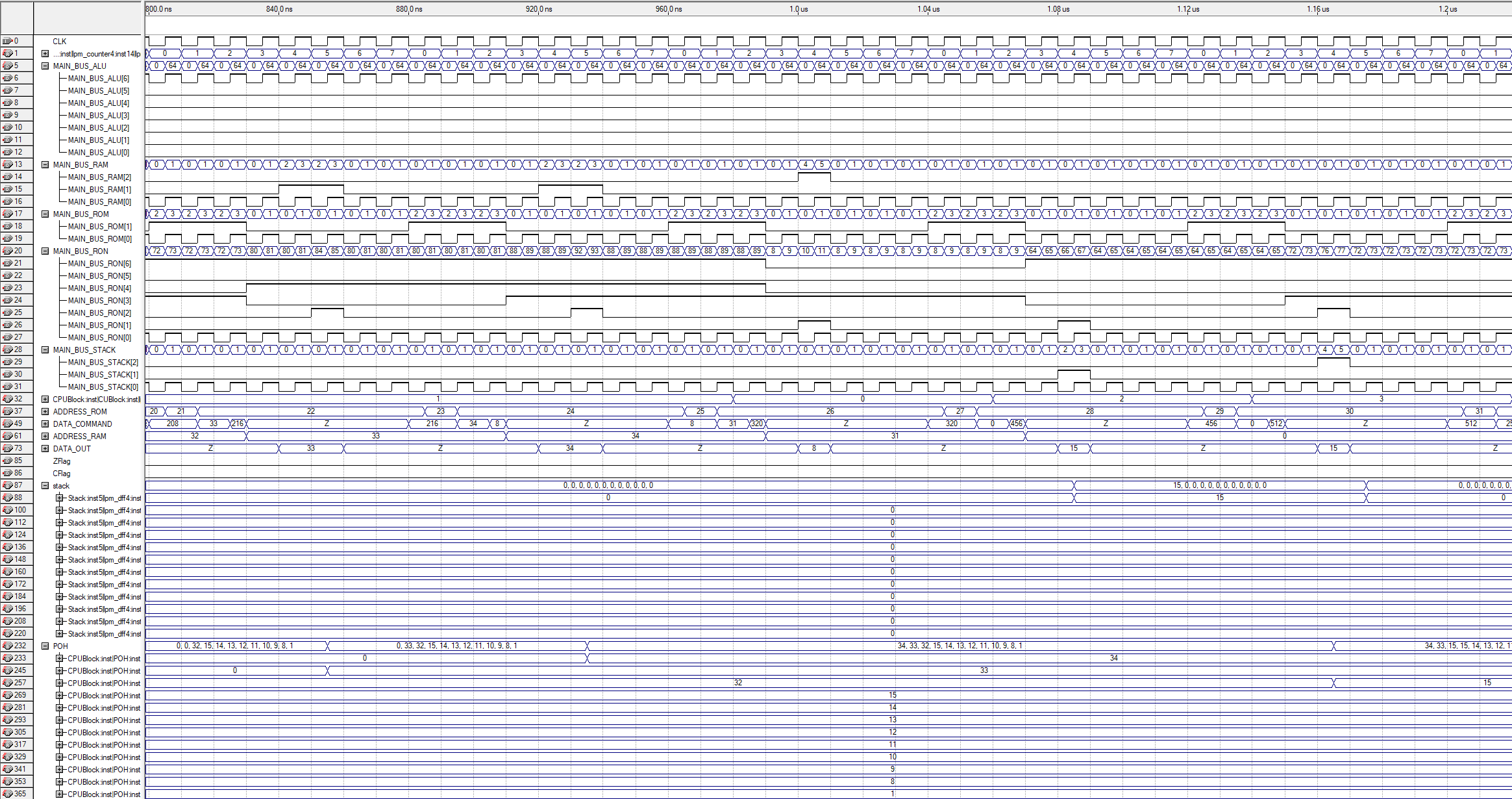


Рисунок 3.5 – Общее моделирование: команды mov reg10 adr33, mov reg11 adr34, mov adr 31 reg1, push reg8, pop reg9

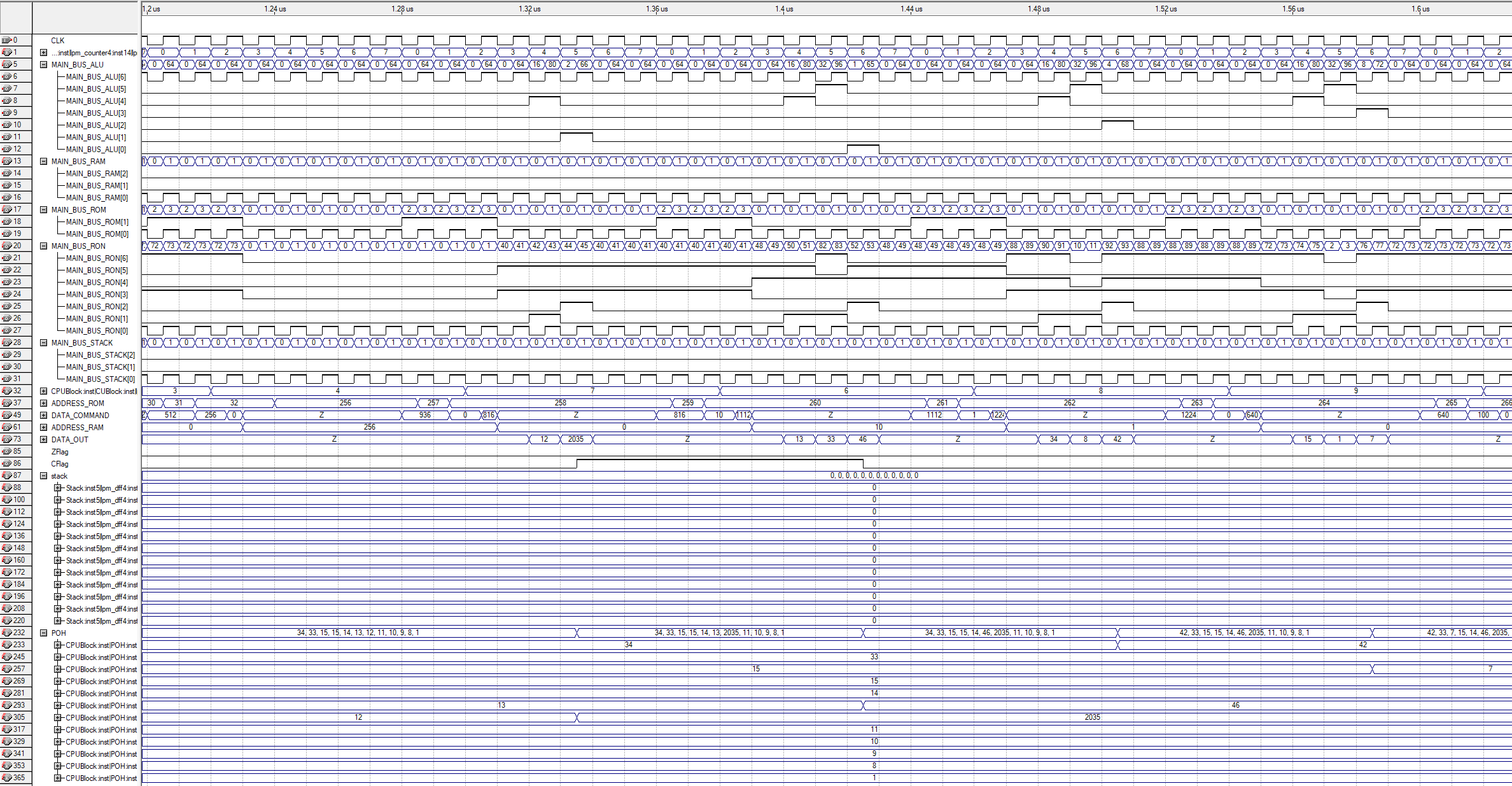


Рисунок 3.6 – Общее моделирование: команды jmp 256, not reg5, addc reg6 reg10, or reg11 reg1, sra reg9 reg0

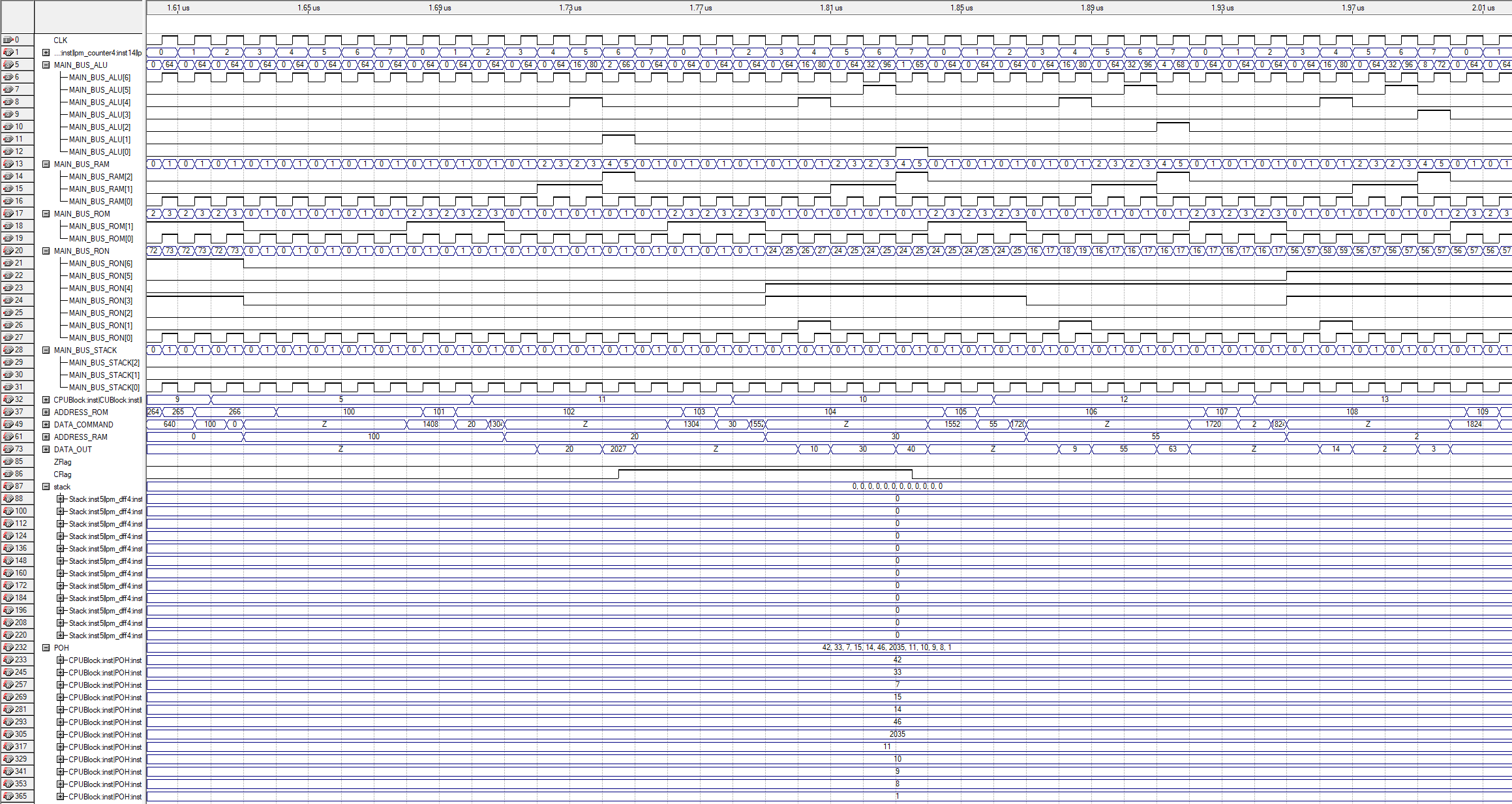


Рисунок 3.7 – Общее моделирование: команды jaz 100, not adr20, addc adr30 reg3, or adr55 reg2, sra adr2 reg7

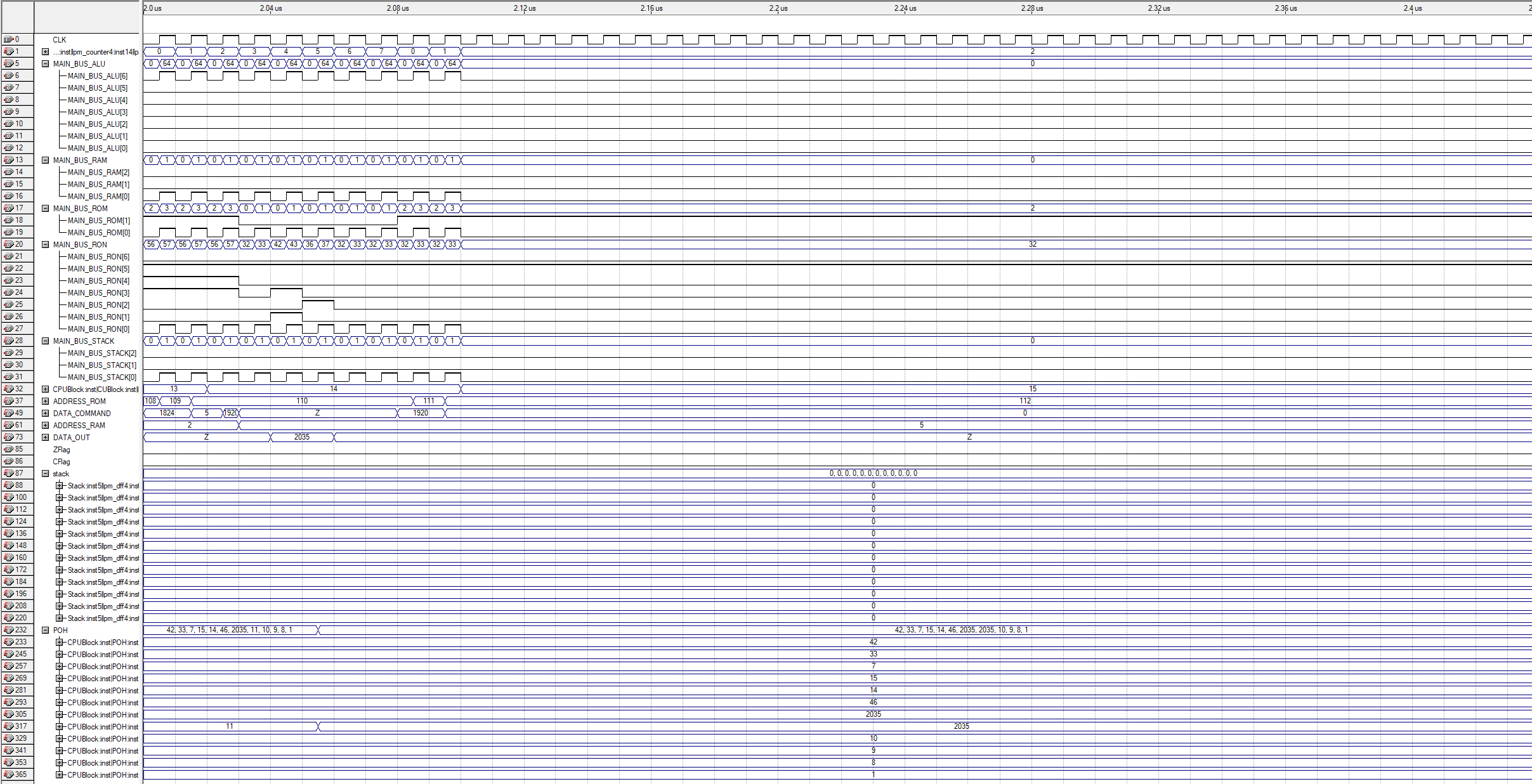


Рисунок 3.8 – Общее моделирование: команды mov reg4 reg5, hlt

# **ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте была разработана микро-ЭВМ с гарвардской архитектурой, которая предполагает наличие раздельного адресного пространства для команд и данных. Блок памяти содержит модули ПЗУ и ОЗУ размером по 2048 байт. Также разработанное устройство содержит 12 одиннадцатиразрядных регистров общего назначения. В качестве отдельных блоков в устройстве расположены стек, состоящий из 12 регистров, с направлением роста вниз и арифметико-логическое устройство. Разработанное АЛУ позволяет выполнять 4 команды: ADDC, NOT, OR, SRA.

Было проведено функциональное моделирование отдельных блоков, а также всей системы с помощью разработанной для этого микропрограммы, содержащей все реализованные команды. Результаты моделирования свидетельствуют о том, что все команды выполняются верно.

Разработанная система может быть расширена путем добавления конвейера, кэша, предсказателя переходов и контроллера прямого доступа к памяти (КПДП).

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. У. Столлингс, Структурная организация и архитектура компьютерных систем/ У. Столлингс. 5-е изд. – М.: "Вильямс ", 2001. Пер. с англ. – 892 с.
2. Р. Грушвицкий, Проектирование систем на микросхемах программируемой логики / Р. Грушвицкий. – Спб.: "Питер", 2002. – 608 с.
3. Самаль Д.И – Структурная и функциональная организация ЭВМ. -БГУиР, кафедра ЭВМ, 2013.
4. Зотов\_В.Ю. – Проектирование Встраиваемых Микропроцессорных Систем На Основе ПЛИС Фирмы XILINX, 2006.
5. Архитектура вычислительных машин [Электронный ресурс] - Режим доступа: <https://prog-cpp.ru/comp-architecture/>.
6. И.И. Глецевич, В.А. Прытков, А.В. Отвагин - Методические указания по дипломному проектированию для студентов специальности 40 02 01 «Вычислительные машины, системы и сети» всех форм обучения, Минск, 2009, 99 с.