В таблице 1.1 приведена структура команды разрабатываемого микро-ЭВМ.

Таблица 1.1 – Структура команды микро-ЭВМ

|  |  |  |
| --- | --- | --- |
| Первое слово | | |
| Биты 29-24 | Биты 23-15 | |
| КОП | Адрес операнда блока памяти  (MEM) | |
| Второе слово | | |
| Биты 14-11 | | Биты 10-0 |
| Адрес операнда в памяти общего назначения (R) | | Безразличные биты |

В таблице 1.2 представлен список команд микро-ЭВМ с соответствующими типами адресации и кодами операций.

Таблица 1.2 – Архитектура системы команд микро-ЭВМ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Команда | Первое слово | | Второе слово | |
| 1 | 2 | 3 | | 4 | |
| 1 | MOV (MEM → R1) | 000000 | MEM | R1 | XXXXXXXXXXX |
| 2 | MOV (R1 → MEM) | 000001 | MEM | R1 | XXXXXXXXXXX |
| 3 | MOV (R1 → R2) | 000010 | R1 | R2 | XXXXXXXXXXX |
| 4 | PUSH (R1) | 000011 | R1 | XXXX | XXXXXXXXXXX |
| 5 | POP (R1) | 000100 | R1 | XXXX | XXXXXXXXXXX |
| 6 | SUB(MEM, R1) → MEM | 000101 | MEM | R1 | XXXXXXXXXXX |
| 7 | SUB(R2, R1) → R2 | 000110 | R2 | R1 | XXXXXXXXXXX |
| 8 | XOR(MEM, R1) → MEM | 000111 | MEM | R1 | XXXXXXXXXXX |
| 9 | XOR(R2, R1) → R2 | 001000 | R2 | R1 | XXXXXXXXXXX |
| 10 | NAND(MEM, R1) → MEM | 001001 | MEM | R1 | XXXXXXXXXXX |
| 11 | NAND(R2, R1) → R2 | 001010 | R2 | R1 | XXXXXXXXXXX |
| 12 | ROR(MEM, R1) → MEM | 001011 | MEM | R1 | XXXXXXXXXXX |
| 13 | ROR(R2, R1) → R2 | 001100 | R2 | R1 | XXXXXXXXXXX |
| 14 | JMP(MEM) | 001101 | MEM | XXXX | XXXXXXXXXXX |
| 15 | JAZ(MEM) | 001110 | MEM | XXXX | XXXXXXXXXXX |
| 16 | HLT | 111111 | XXXXXXXXX | XXXX | XXXXXXXXXXX |
| Примечание – X обозначает безразличный бит, MEM - адрес операнда блока памяти, R1 и R2 – номера регистров из РОН. | | | | | |

**2 РАЗРАБОТКА ОСНОВНЫХ КОМПОНЕНТОВ МИКРО-ЭВМ**

**2.1 Блок памяти**

Так как пристанская архитектура требует использования принципа общей памяти, то было принято решение: объединить блок ПЗУ (постоянного запоминающего устройства) и блок ОЗУ (Оперативного запоминающего устройства) в один компонент. Условное графическое обозначение (УГО) блока памяти представлено на рисунке 2.1.

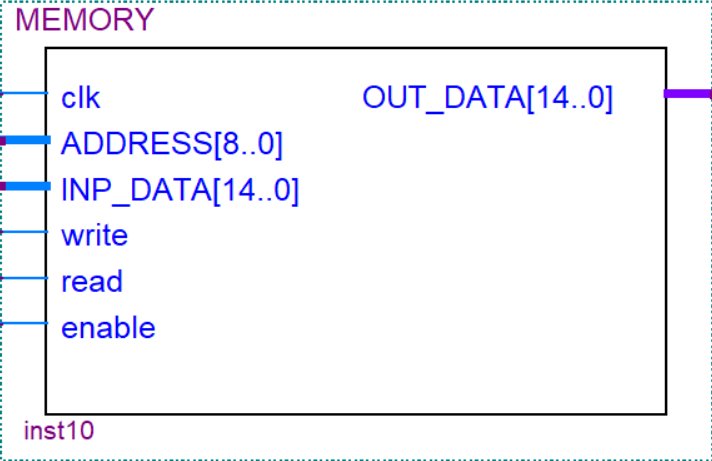


Рисунок 2.1 – УГО блока памяти

Данный блок имеет следующие входные сигналы:

– clk – тактовый сигнал;

– ADDRESS[8..0] – шина адреса для доступа к необходимой ячейке;

– INP\_DATA[14..0] – шина данных для записи информации в память;

– write – сигнал для разрешения записи в память;

– read – сигнал для разрешения чтения из памяти;

– enable – сигнал, разрешающий работу блока.

Также выходные сигналы:

– OUT\_DATA[14..0] – шина для вывода данных из памяти;

Так как ПЗУ является асинхронным, а ОЗУ синхронным, следовательно для вывода данных из ОЗУ необходимо использовать синхросигнал, в отличии от ПЗУ, где данные выходят постоянно.

Общий объем памяти равен 29 = 512, так как разрядность шины данных равна 9 битам. Для регулирования обращения к ОЗУ или ПЗУ используется старший бит адреса, если он равен единице, то обращение будет происходить к ОЗУ. Также в ПЗУ нельзя записывать данные. Для работы блока необходимо подать сигнал enable вместе с сигналом clk.

Чтение осуществляется за 2 такта. Для получение необходимых данных нужно подавать необходимый адрес на протяжении 2 тактов, а также read на протяжении 2-го такта для вывода данных.

Для записи необходимо также подавать адрес на протяжении 2 тактов, а также сигнал write.

Моделирование блока памяти представлено на рисунке 2.2.

**3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ**

**3.1 Функциональное моделирование программы для тестирования**

Общая функциональная схема разработанной микро-ЭВМ представлена в приложении ГУИР.400201.110 Э2.18. Программа для тестирования приведена в таблице 3.1.

Таблица 3.1 – Программа для тестирования микро-ЭВМ

|  |  |  |  |
| --- | --- | --- | --- |
| Счетчик команд | Символьный вид | Бинарный вид | Пояснение |
| 1 | 2 | 3 | 4 |
| 00 | MOV $256, R0 | 000000100000000  000000000000000 | [$256] = -122 |
| 01 | MOV $259, R1 | 000000100000000  000000000000000 | [$259] = 1488 |
| 02 | MOV R0, R3 | 000010000000000  001100000000000 | [R0] = -122 |
| 03 | NAND R0, R3 | 001010000000000  001100000000000 | [R0] = -122,  [R3] = -122 |
| 04 | JAZ $0 | 001110000000000  000000000000000 | ZF = 0, SF = 0  Перехода не будет |
| 05 | SUB $352, R1 | 000101101100000  000100000000000 | [$352] = 56  [R1] = 1488 |
| 06 | JAZ $90 | 000111001011010  000000000000000 | ZF = 0, SF = 1  Переход будет |
| 07 | XOR $260, R1 | 000111000000100  000100000000000 | [$260] = 22  [R1] = 1488 |
| 08 | PUSH R3 | 000011000000011  000000000000000 | [R3] = -122 |
| 09 | MOV R3, $261 | 000001100000101  001100000000000 | [R3] = -122 |
| 10 | MOV $288, R4 | 000000100100000  010000000000000 | [$288] = 91 |
| 11 | MOV $289, R5 | 000000100100001  010100000000000 | [$288] = 234 |
| 12 | MOV R5, R7 | 000010000000101  011100000000000 | [R5] = 234 |
| 13 | PUSH R4 | 000011000000100  000000000000000 | [R4] = 91 |
| 14 | PUSH R7 | 000011000000111  000000000000000 | [R7] = 234 |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| 15 | POP R0 | 000100000000000  000000000000000 | [TOP] = 234 |
| 16 | PUSH R3 | 000011000000011  000000000000000 | [R3] = -122 |
| 17 | JMP $144 | 001101010010000  000000000000000 | Переход будет |
| 18 | ROR $400, R1 | 001011110010000  000100000000000 | [$400] = 1,  [R1] = 1488 |
| 19 | SUB R7, R5 | 000110000000111  010100000000000 | [R7] = 234,  [R5] = 234 |
| 20 | ROR R1, R4 | 001100000000001  010000000000000 | [R1] = 1488,  [R4] = 91 |
| 21 | NAND $401, R7 | 001001110010001  011100000000000 | [$401] = -2,  [R7] = 0 |
| 22 | XOR $402, R1 | 000111110010010  000100000000000 | [$402] = -100,  [R1] = -8960 |
| 23 | MOV R7, $410 | 000001110011010  011100000000000 | [R7] = 0 |
| 24 | HLT | 11111010010000  000000000000000 | Остановка работы ЭВМ |

Дамп части памяти, которая содержит данные о выполняемых командах представлена на рисунке 3.1.

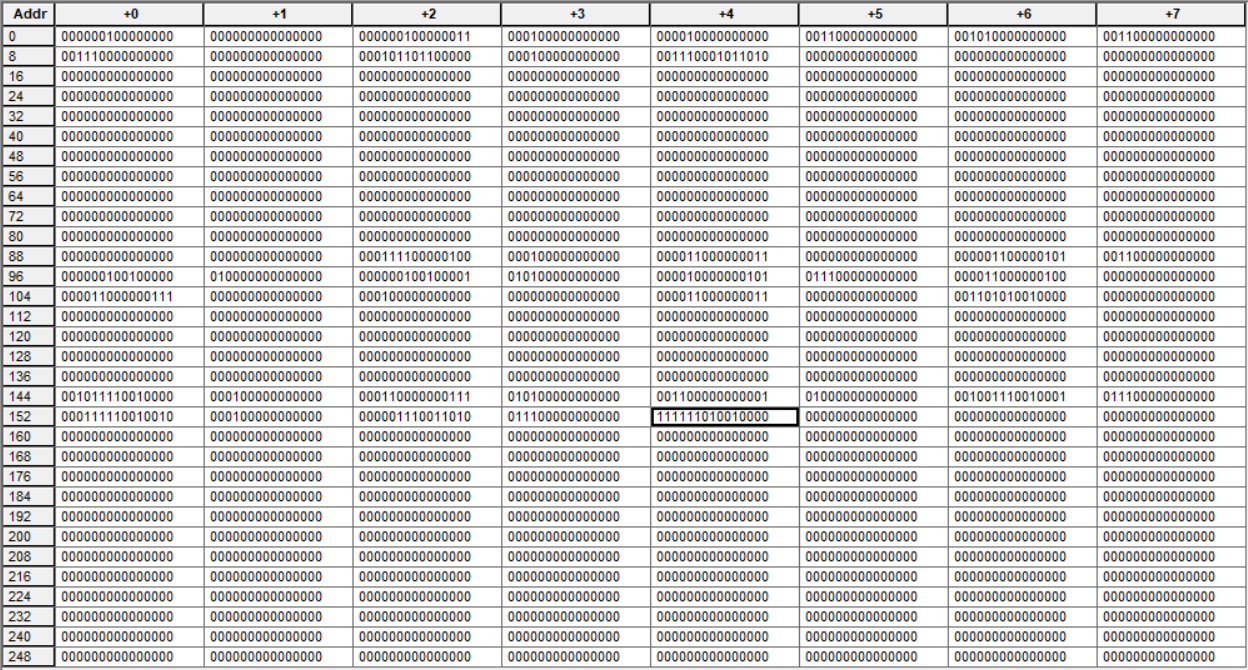


Рисунок 3.1 – Дамп части памяти, содержащую программу

Дамп части памяти, которая содержит операнды для выполнения программы, а также результаты операций программы, до и после выполнения программы представлен на рисунках 3.2 и 3.3 соответственно.

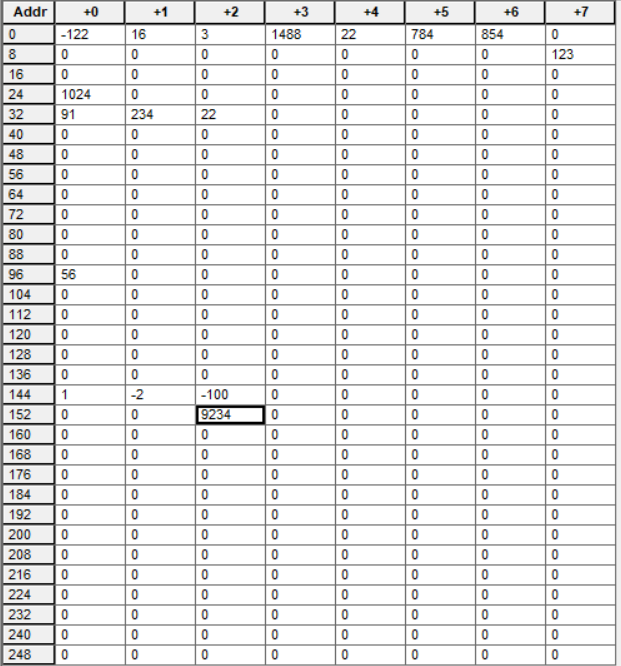


Рисунок 3.2 – Дамп части памяти с операндами и результатами до выполнения



Рисунок 3.3 – Дамп части памяти с операндами и результатами после выполнения

Результаты моделирования программы представлены на рисунках 3.4 – 3.28.

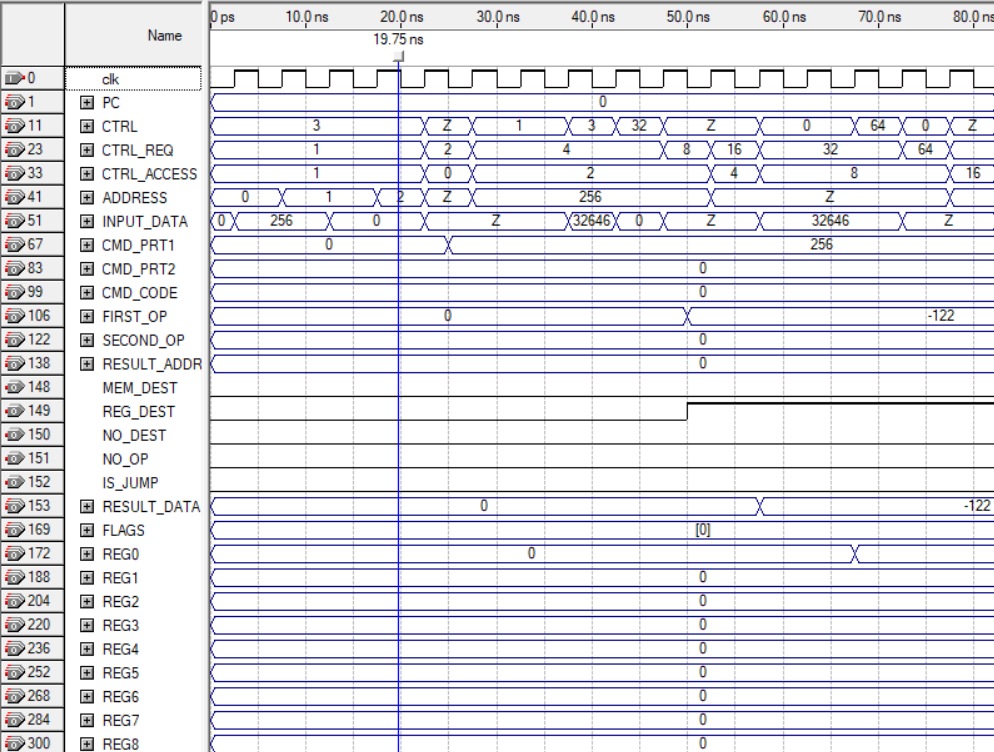


Рисунок 3.4 – Моделирование первой команды

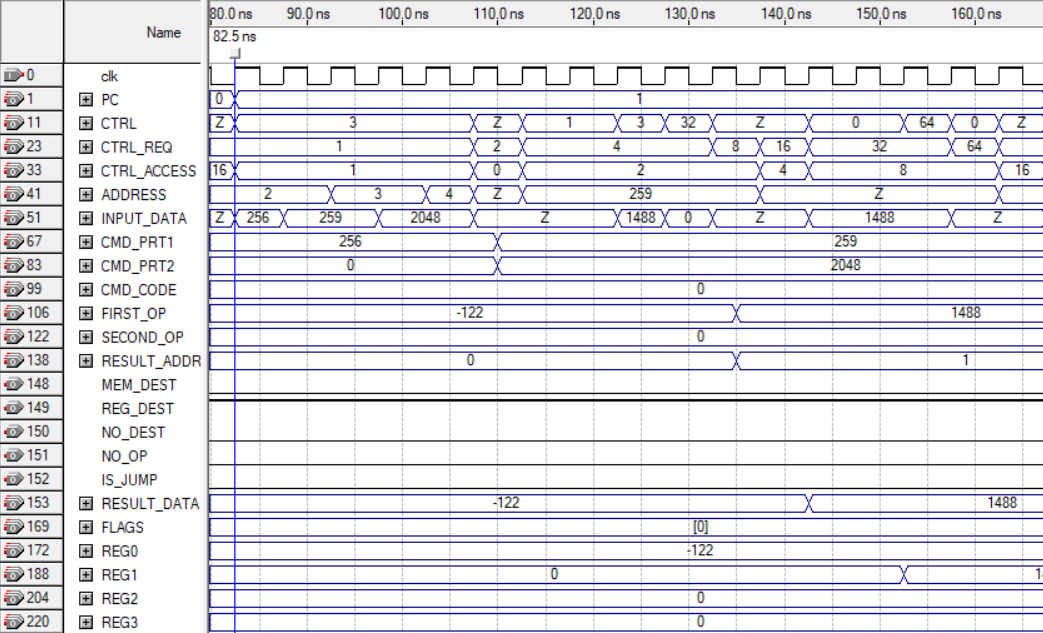


Рисунок 3.5 – Моделирование второй команды

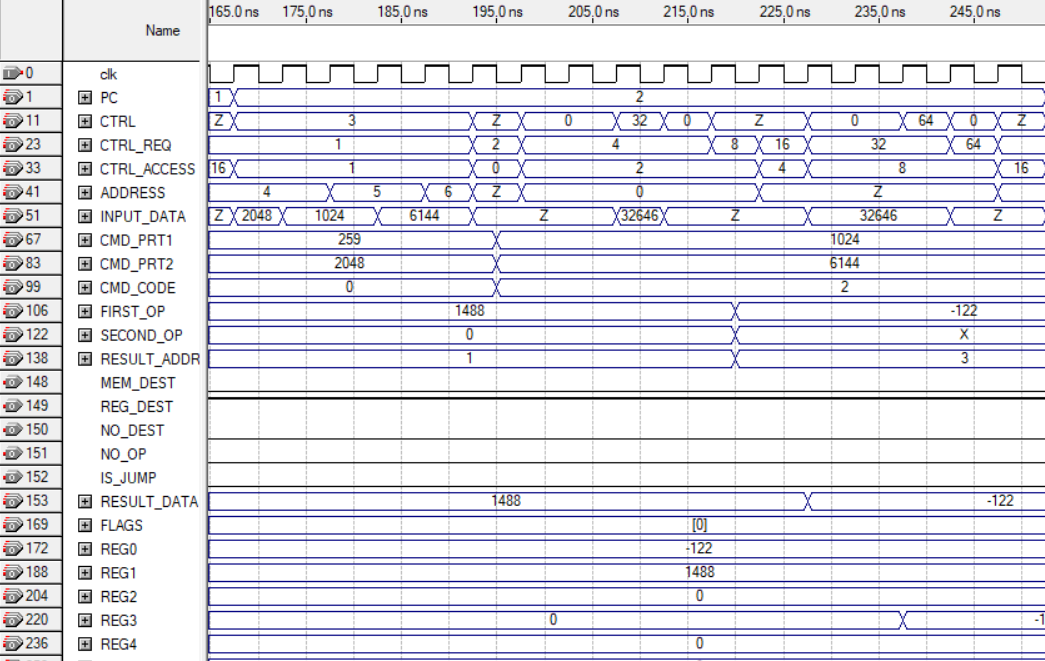


Рисунок 3.6 – Моделирование третей команды

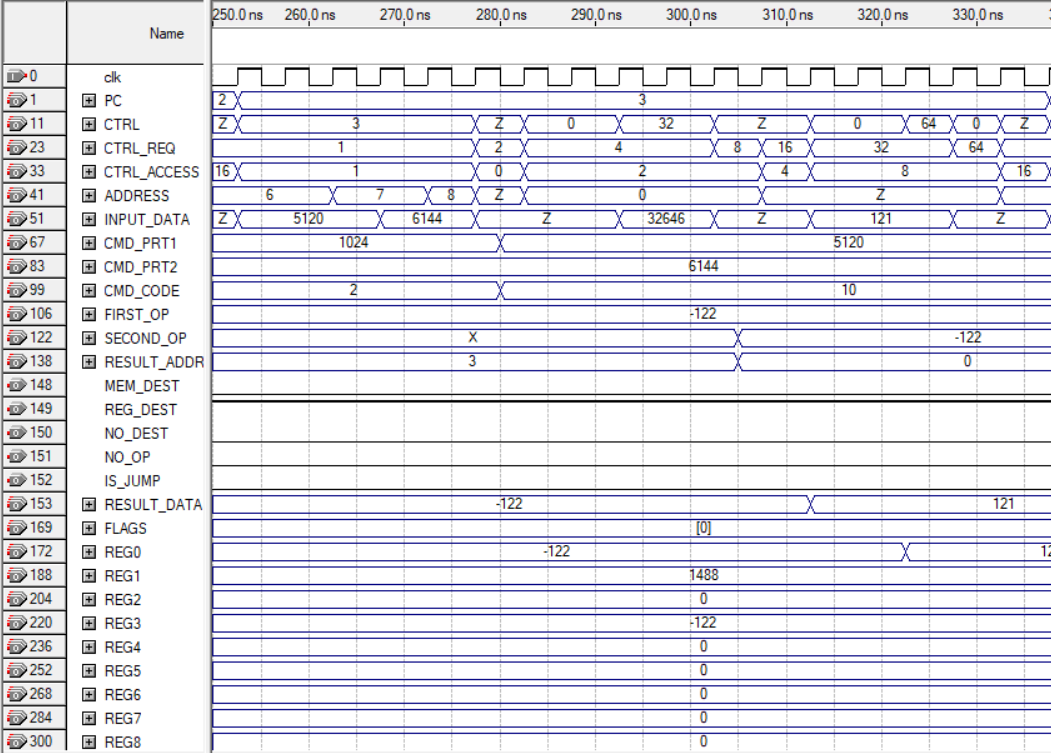


Рисунок 3.7 – Моделирование четвертой команды

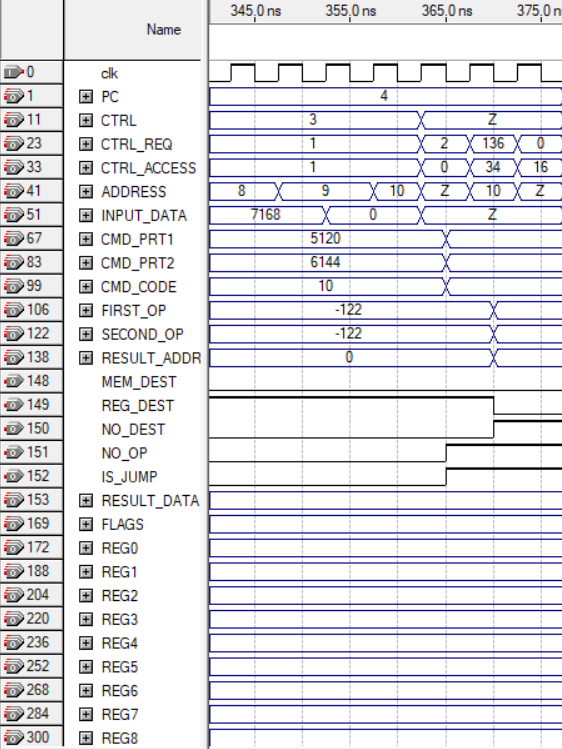


Рисунок 3.8 – Моделирование пятой команды

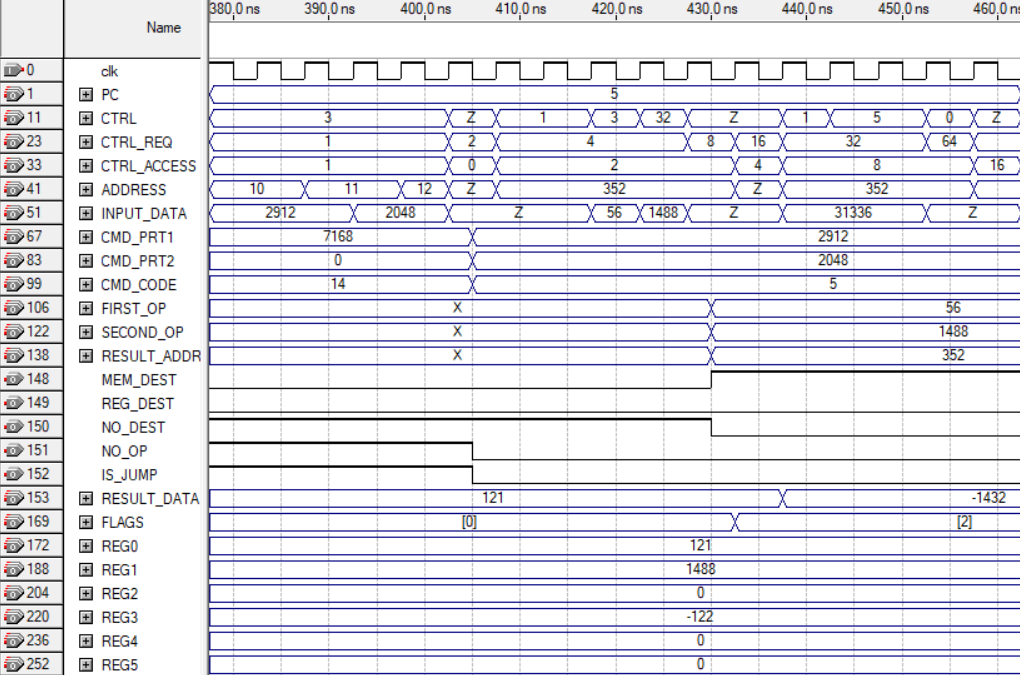


Рисунок 3.9 – Моделирование шестой команды

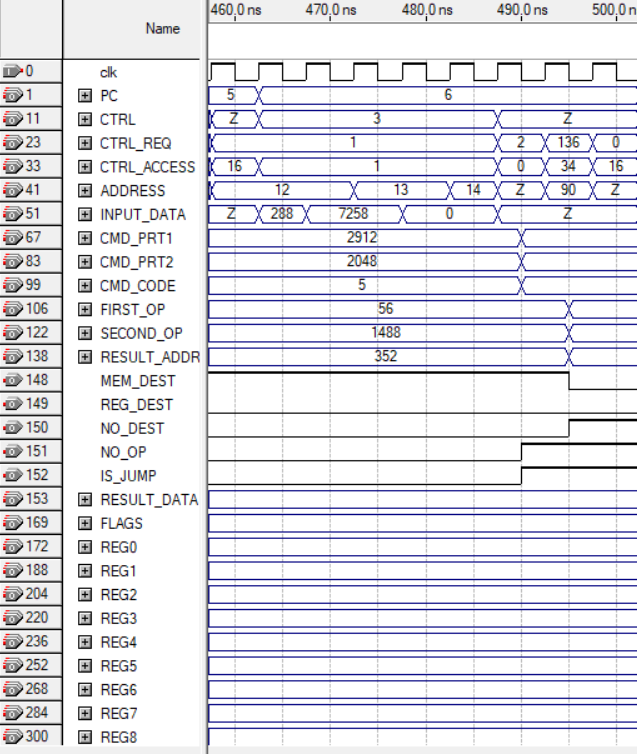


Рисунок 3.10 – Моделирование седьмой команды

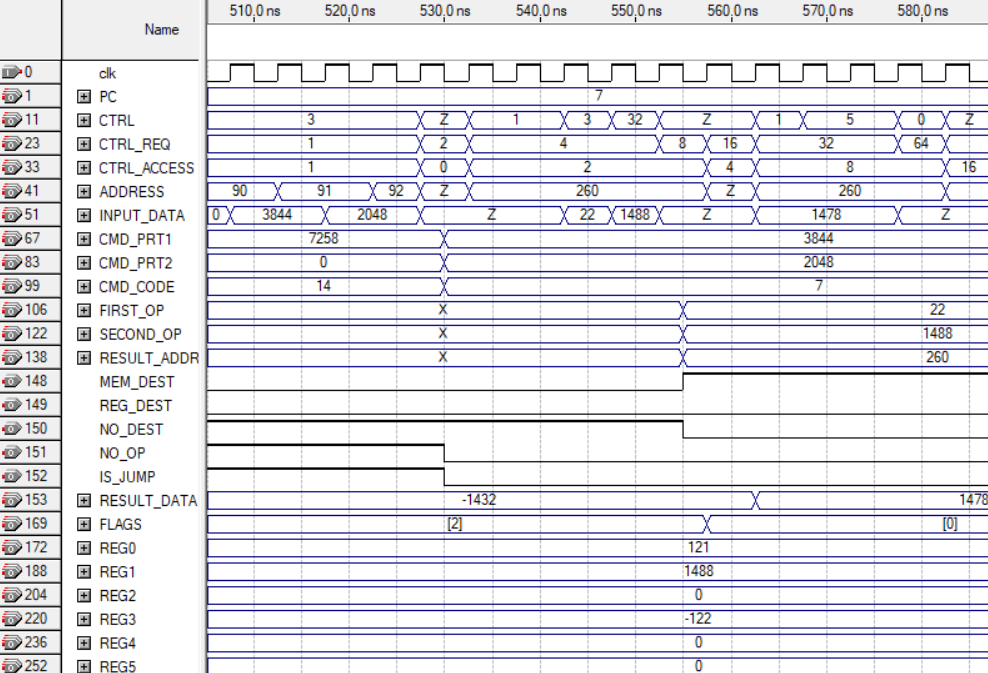
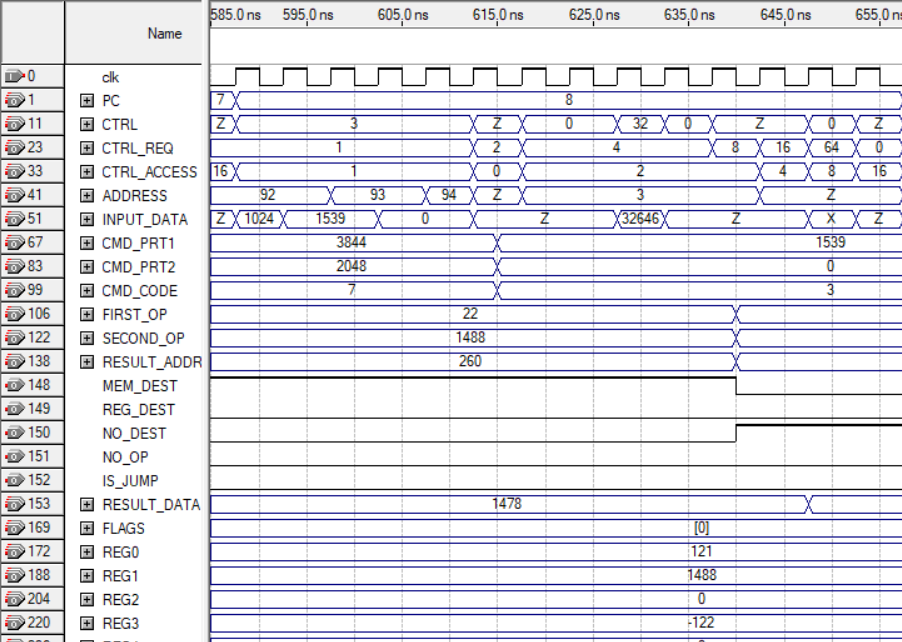


Рисунок 3.11 – Моделирование восьмой команды



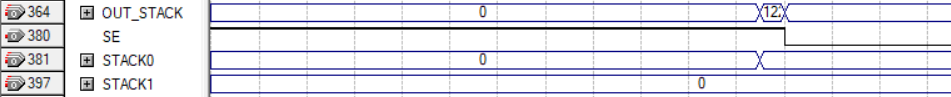


Рисунок 3.12 – Моделирование девятой команды

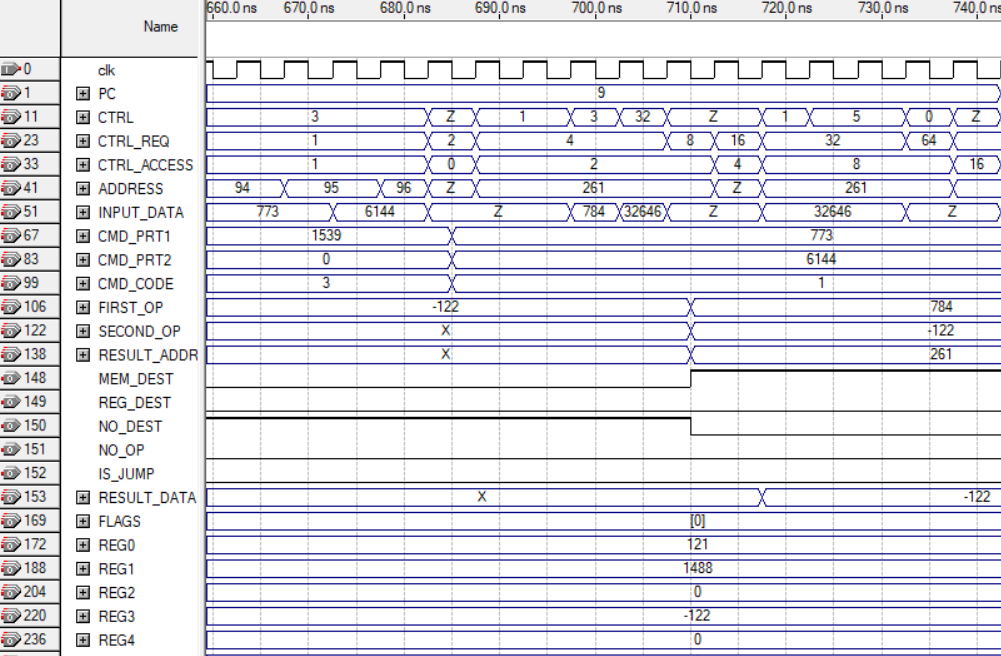


Рисунок 3.13 – Моделирование десятой команды

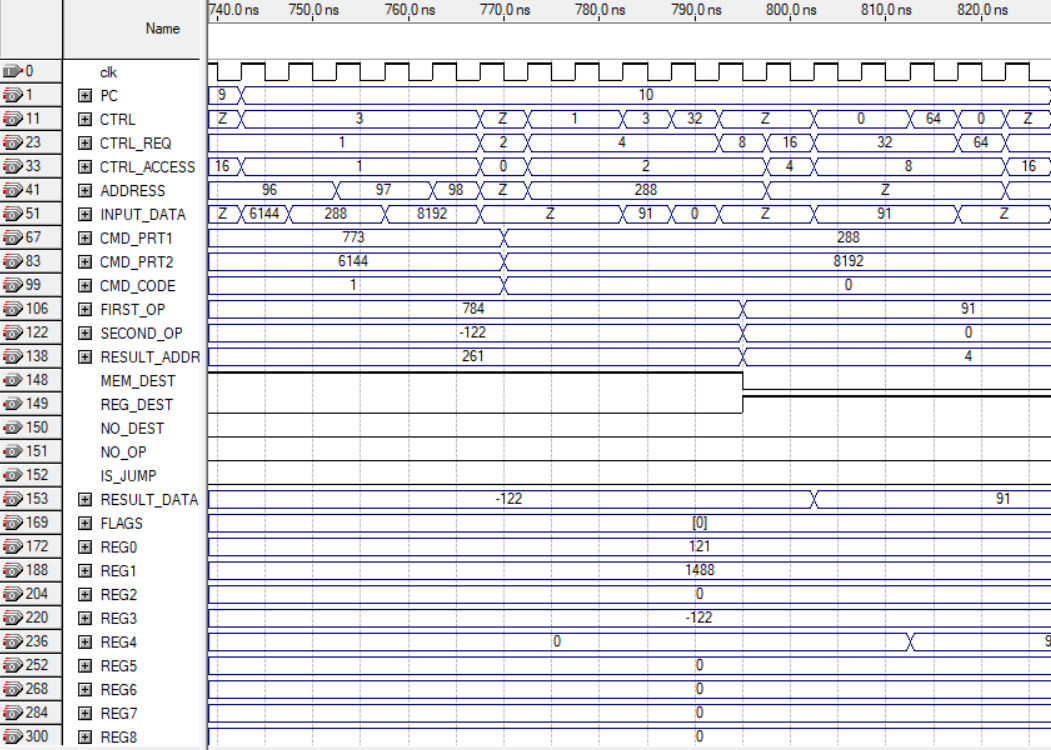


Рисунок 3.14 – Моделирование одиннадцатой команды

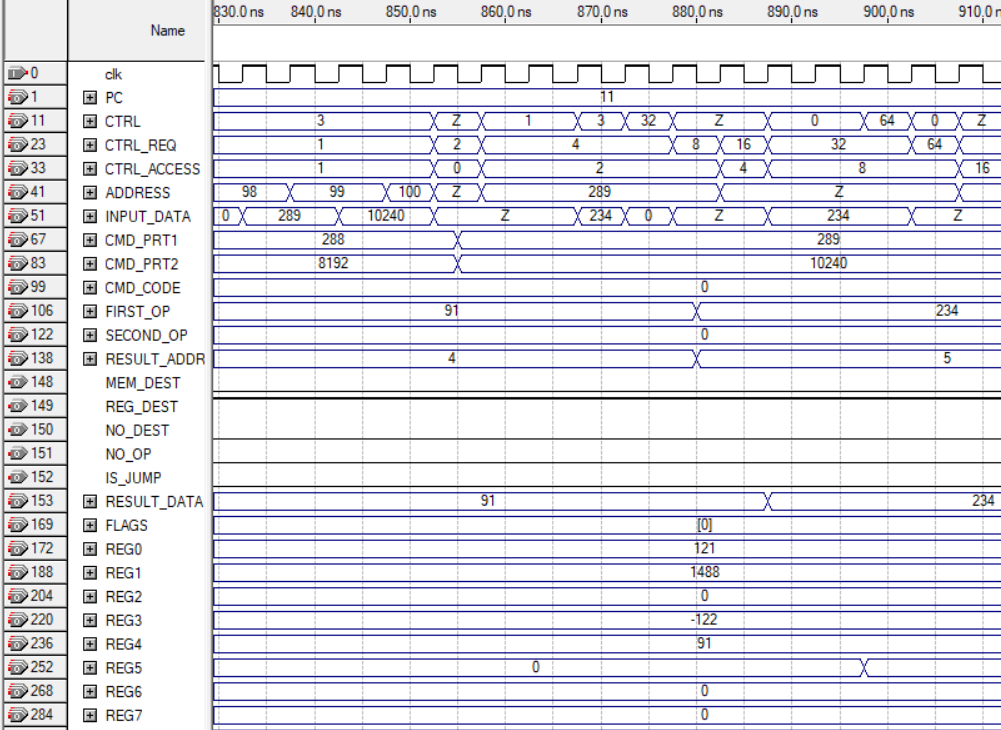


Рисунок 3.15 – Моделирование двенадцатой команды

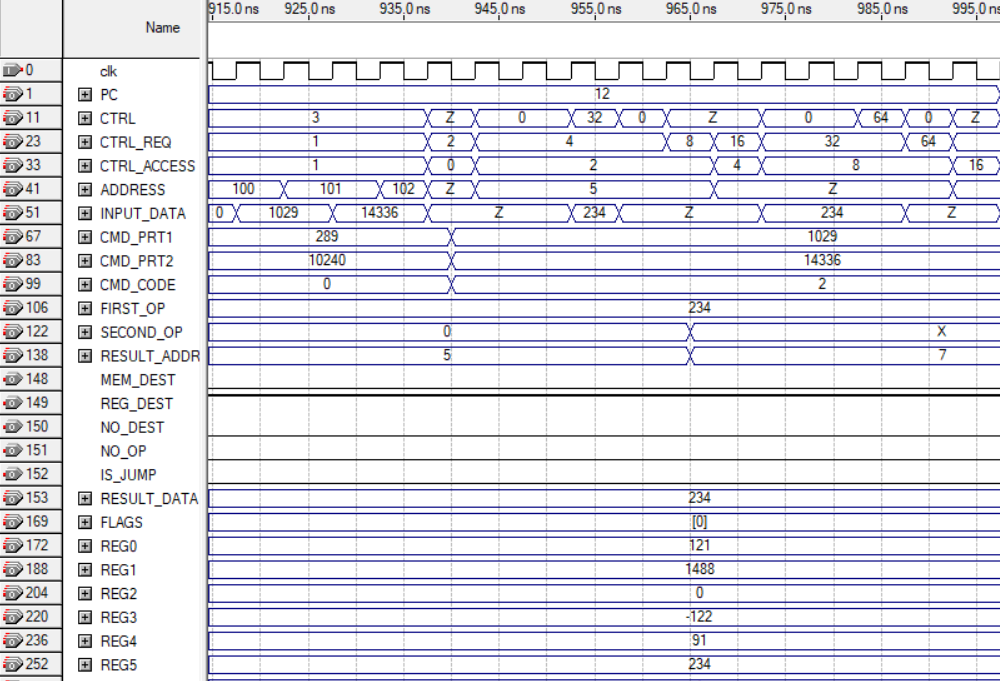
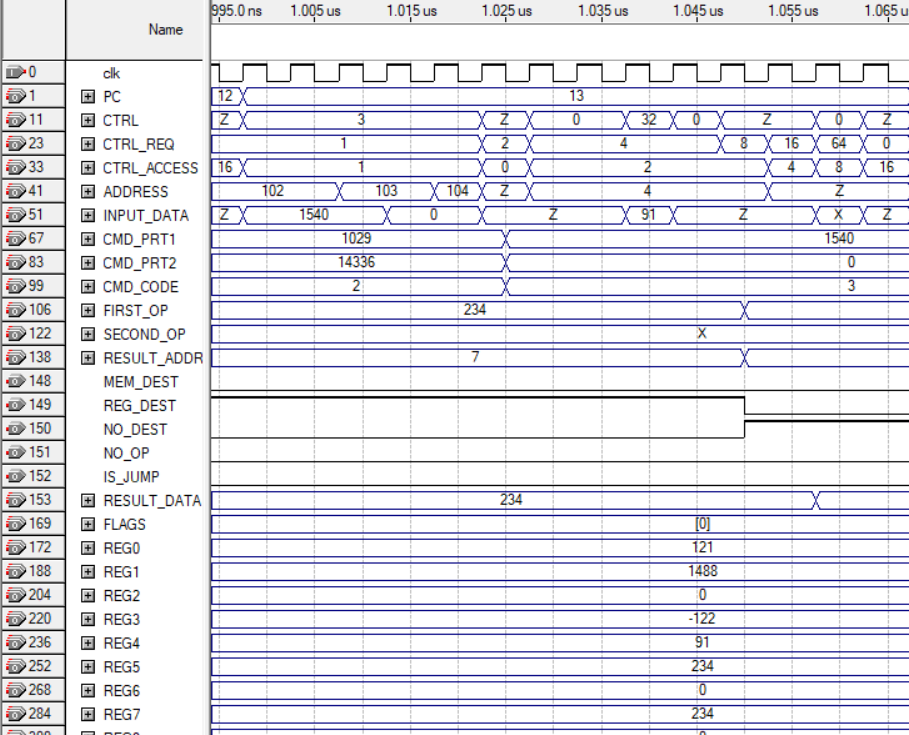


Рисунок 3.16 – Моделирование тринадцатой команды



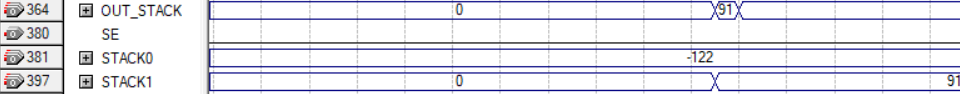
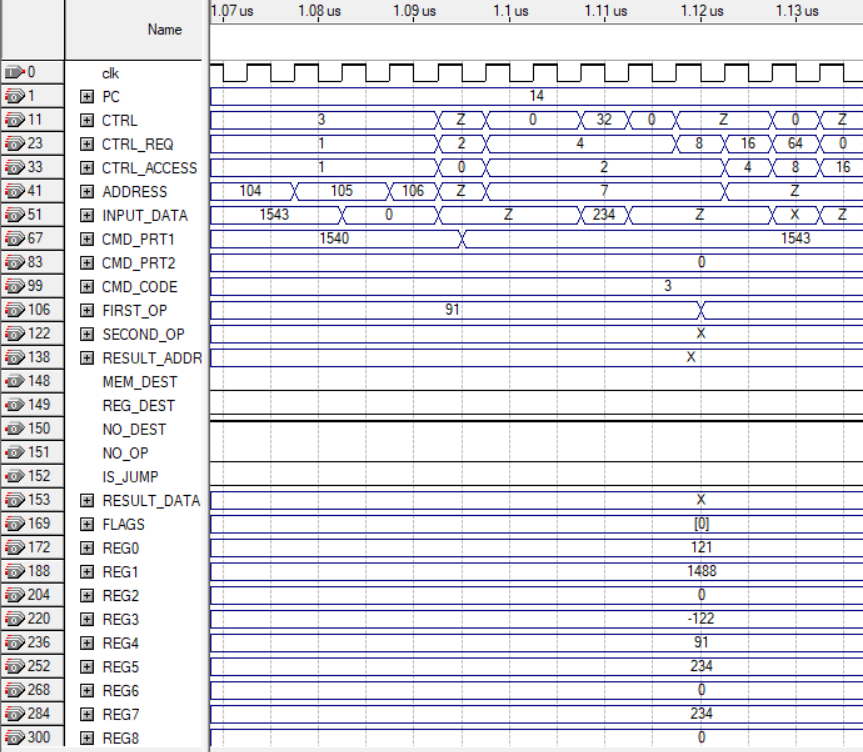


Рисунок 3.17 – Моделирование четырнадцатой команды



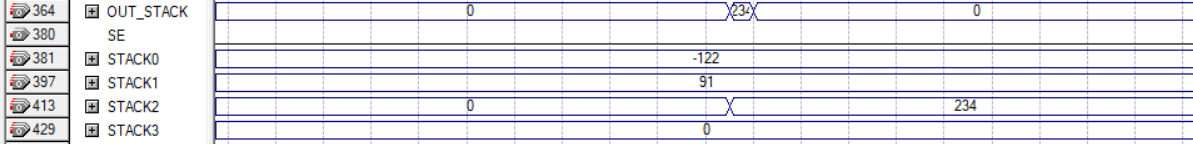
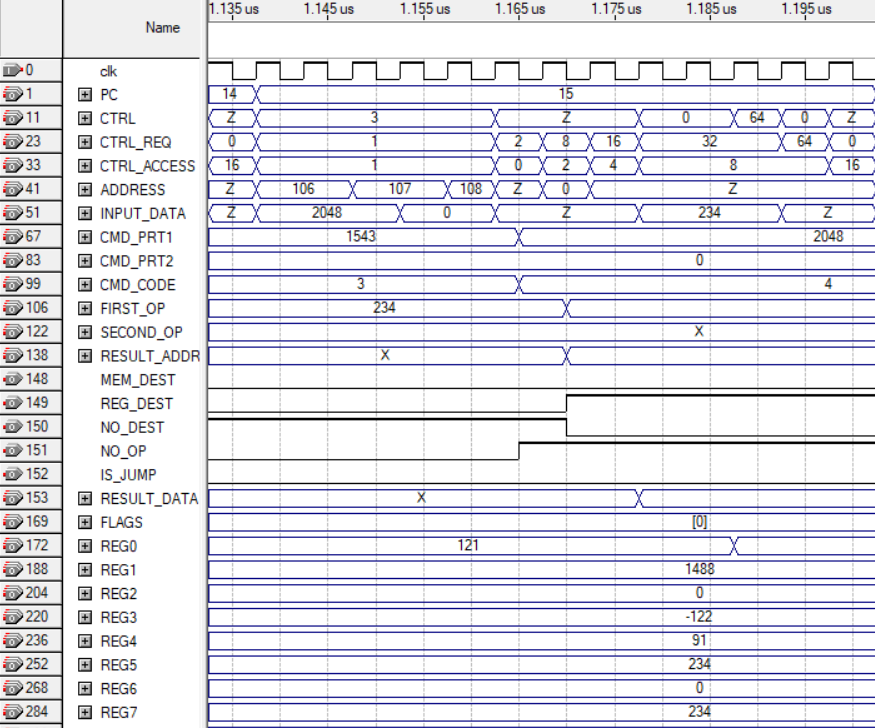


Рисунок 3.18 – Моделирование пятнадцатой команды



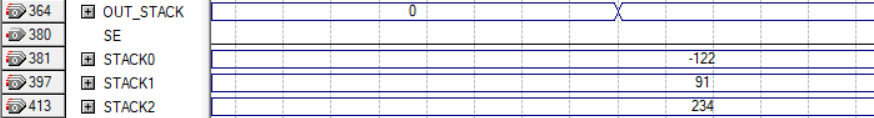
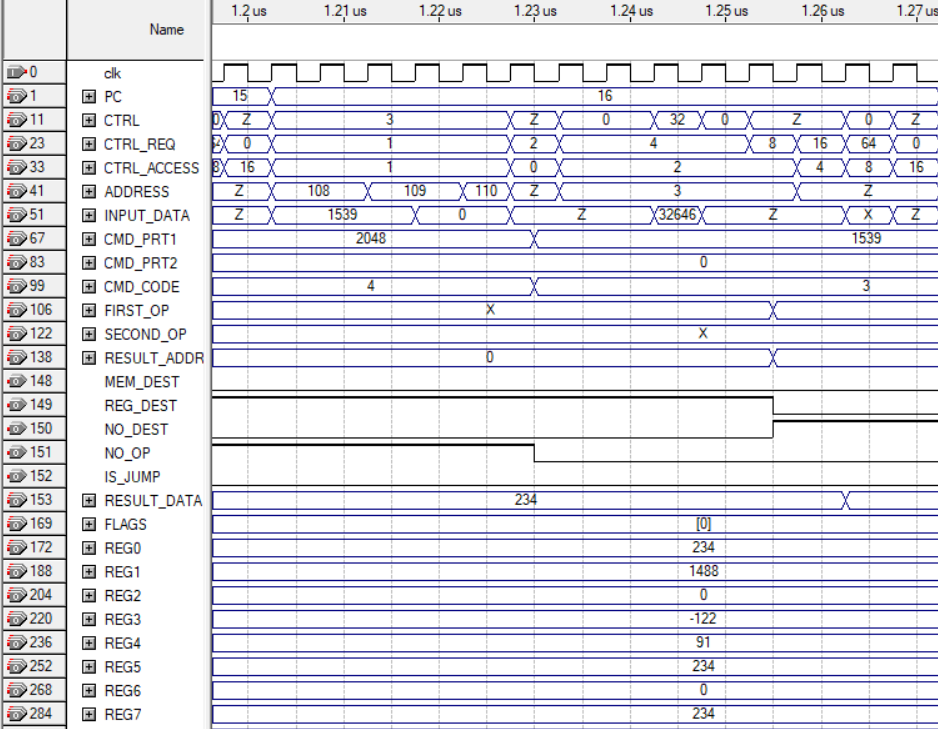


Рисунок 3.19 – Моделирование шестнадцатой команды



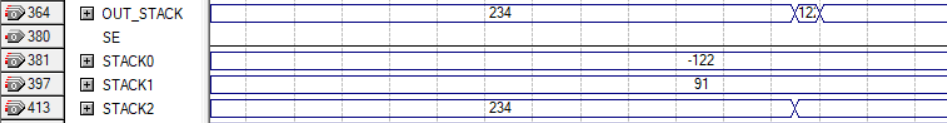


Рисунок 3.20 – Моделирование семнадцатой команды

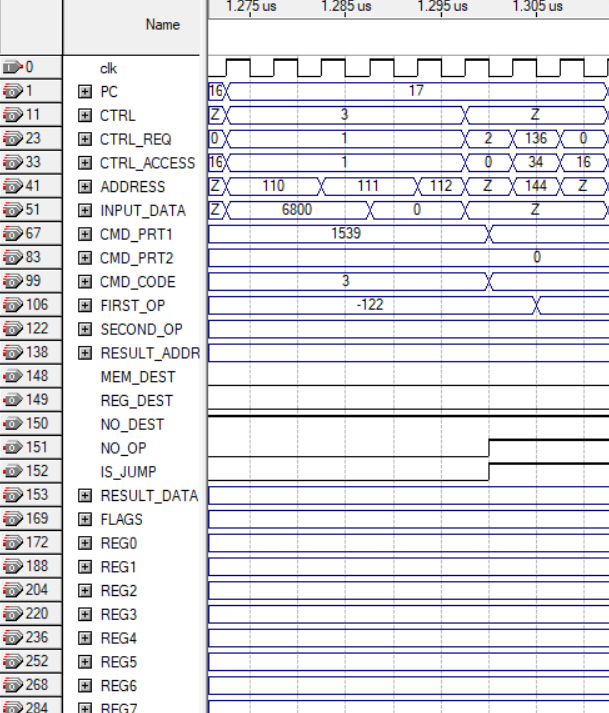


Рисунок 3.21 – Моделирование восемнадцатой команды

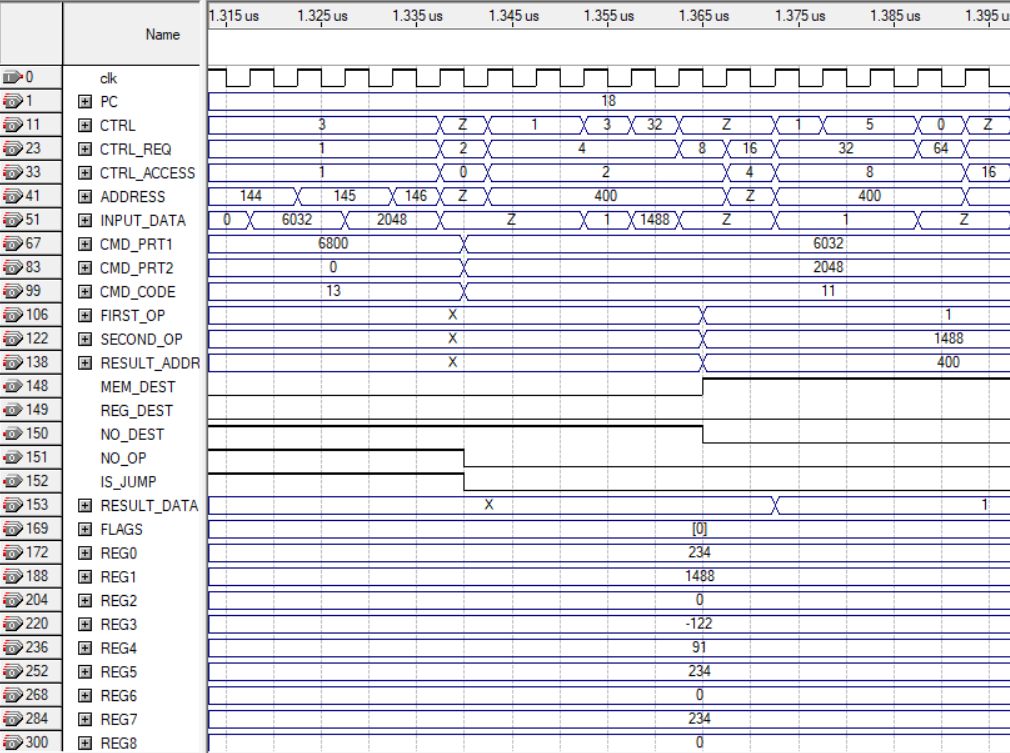


Рисунок 3.22 – Моделирование девятнадцатой команды

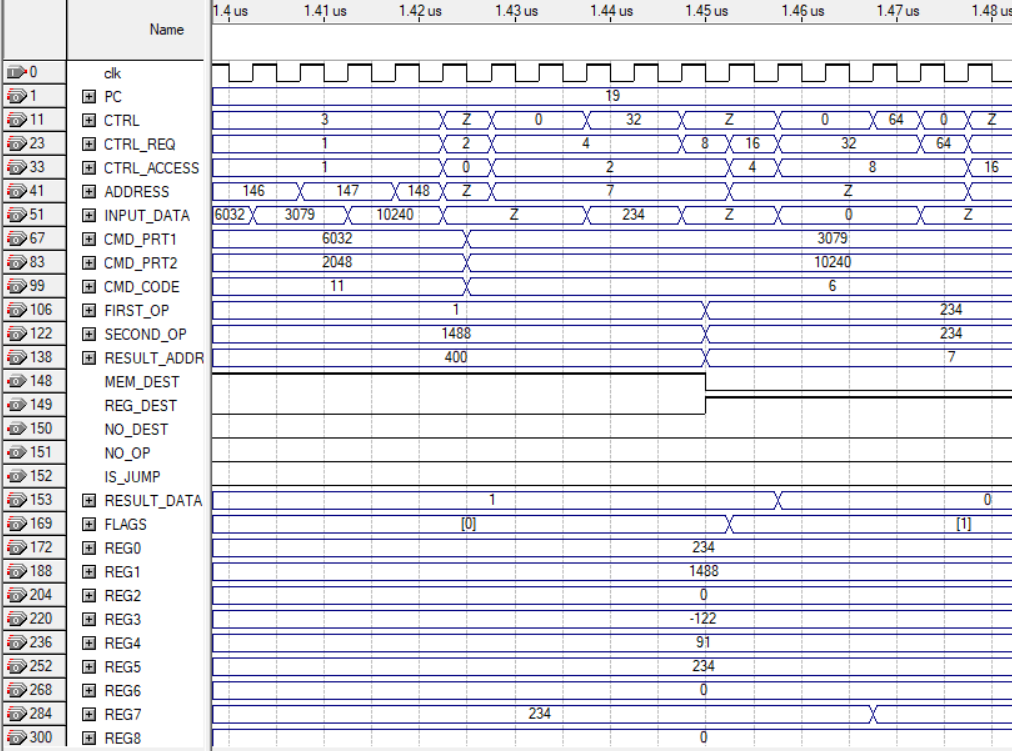


Рисунок 3.23 – Моделирование двадцатой команды

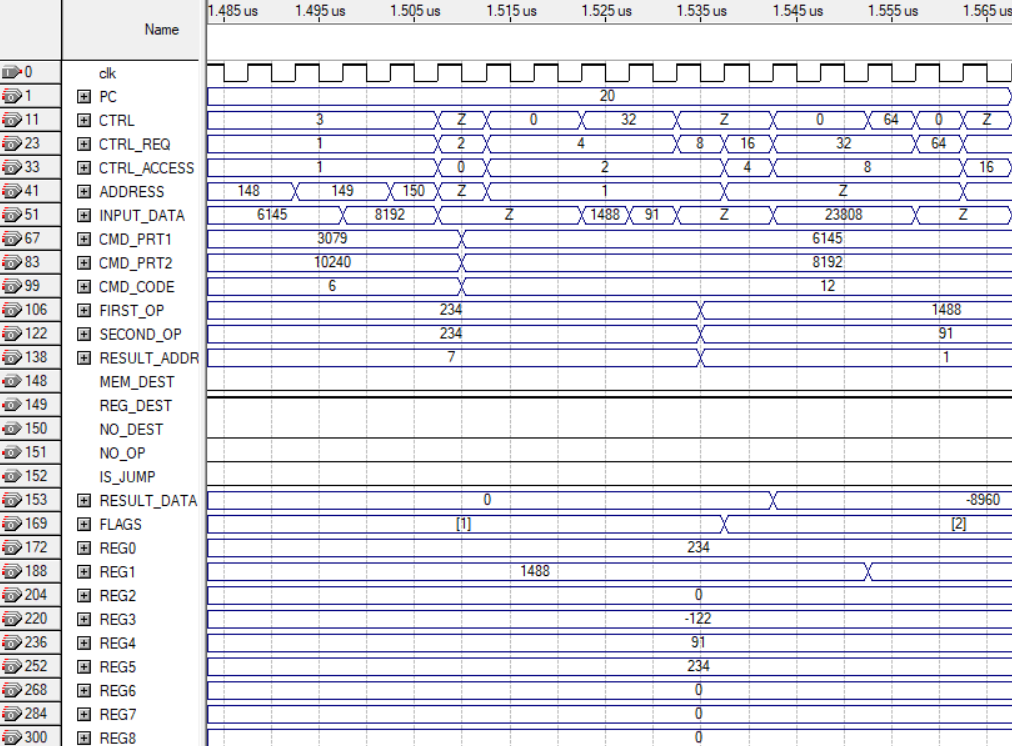


Рисунок 3.24 – Моделирование двадцать первой команды

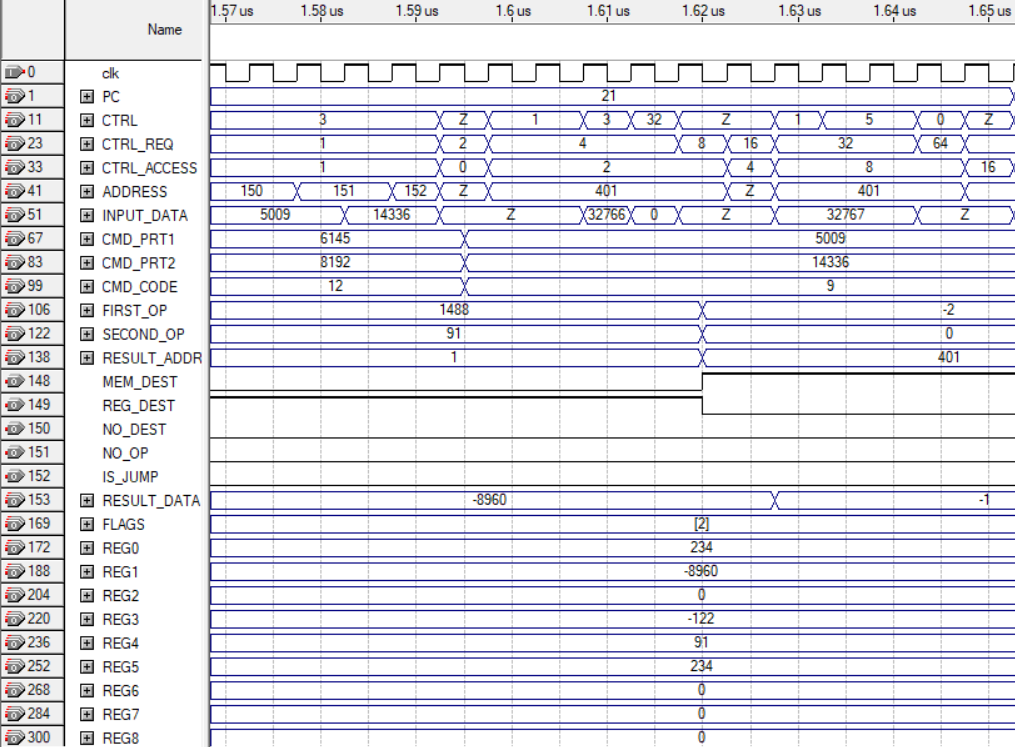


Рисунок 3.25 – Моделирование двадцать второй команды

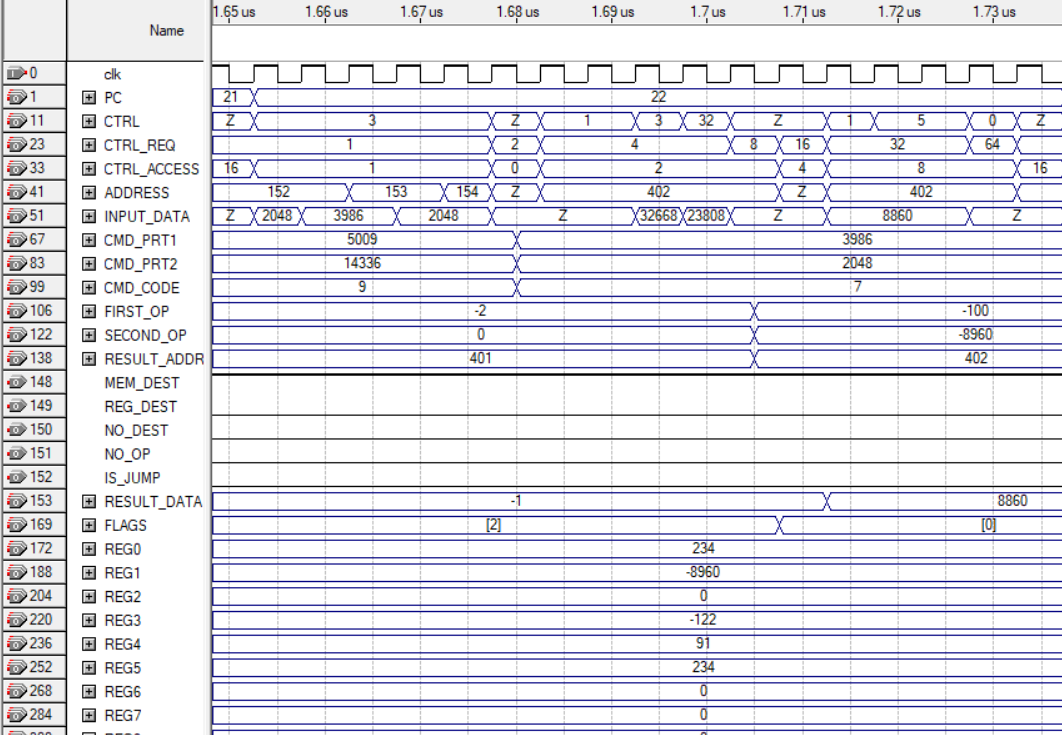


Рисунок 3.26 – Моделирование двадцать третей команды

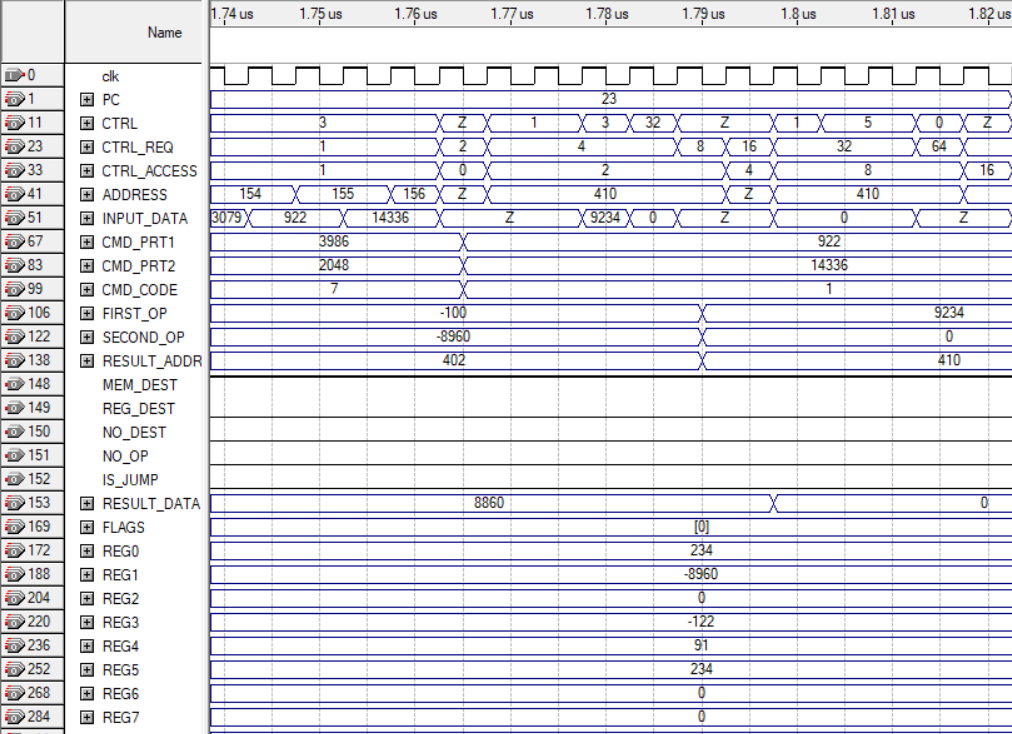


Рисунок 3.27 – Моделирование двадцать четвертой команды

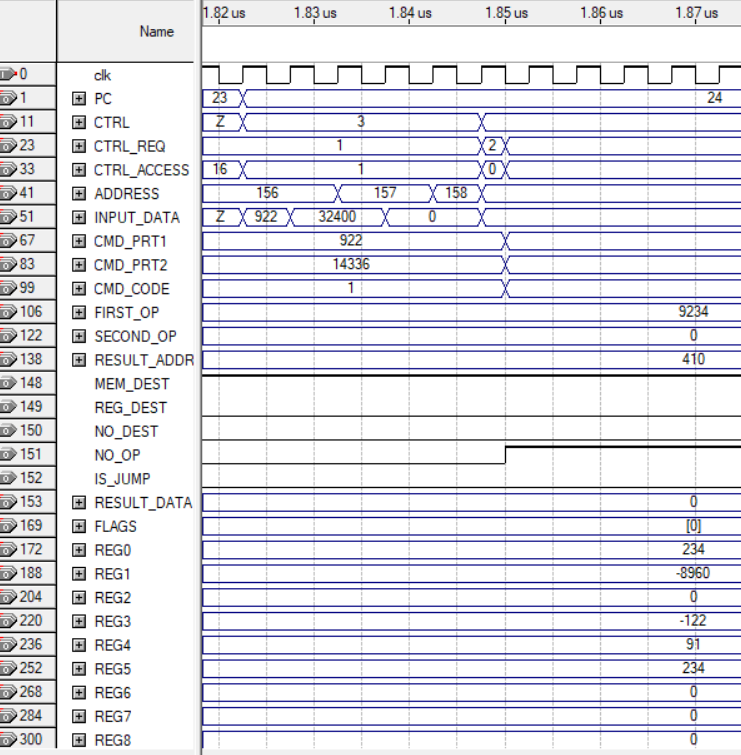


Рисунок 3.28 – Моделирование двадцать пятой команды