ВВЕДЕНИЕ

Персональный компьютер быстро вошел в нашу жизнь. Современные вычислительные машины представляют одно из самых значительных достижений человеческой мысли, влияние, которого на развитие научно-технического прогресса трудно переоценить. Область применения ЭВМ огромна и непрерывно расширяется. Компьютеры применяются в таких сферах как управление производством, транспорт и связь, информационно-вычислительная техника, военная техника, бытовая техника, обучение, медицина, научное исследование и многих других сферах нашей жизни.

Еще несколько лет назад было редкостью увидеть какой-нибудь персональный компьютер – они были, но были очень дорогие, и даже не каждая фирма могла иметь у себя в офисе компьютер. Помимо того, что компьютеры были очень дорогие, они были очень медленные и не могли выполнять функции, которые сейчас нам кажутся. Тридцать лет назад человек и подумать не мог, что он сможет взять компьютер с собой в путешествие, что он сможет работать, находясь за тысячи километров от рабочего места. Все это стало возможно благодаря тому, что производители стали делать более компактные и легкие, более функциональные и дешевые компьютеры. Чтобы этого достичь некоторые компоненты постепенно заменялись более удобными, современными и миниатюрными составляющими. В связи с этим огромную актуальность приобрели специализированные и легко встраиваемые микропроцессорные устройства.

Такие устройства проектируются в расчёте на выполнение ограниченного набора функций. При этом основными критериями для проектировщика служат: компактность, быстродействие, минимальные аппаратные затраты, надёжность, цена и другие факторы.

Задачей данного курсового проект является разработка схемы микро-ЭВМ на ПЛИС согласно выданному варианту.

На разработку схемы устройства оказывает влияние тип архитектуры, что является основным фактором при реализации устройства. Различаются Принстонскую и Гарвардскую архитектуру вычислительных машин. Разрабатываемое устройство соответствует Принстонскому типу архитектуры, которую часто называют архитектурой фон Неймана. Данная архитектура характеризуется использованием общей оперативной памяти для хранения программ и данных. Для обращения к этой памяти используется общая системная шина, по которой в процессор поступают и команды, и данные. Эта архитектура имеет ряд важных достоинств. Наличие общей памяти позволяет оперативно перераспределять ее объем для хранения отдельных массивов команд, данных и реализации стека в зависимости от решаемых задач. Таким образом, обеспечивается возможность более эффективного использования имеющегося объема оперативной памяти в каждом конкретном случае применения микропроцессоров. Использование общей шины для передачи команд и данных значительно упрощает отладку,

тестирование и текущий контроль функционирования системы, повышает ее надежность. Однако ей присущи и существенные недостатки. Основным из них является необходимость последовательной выборки команд и обрабатываемых данных по общей системной шине. При этом общая шина ограничивает производительность цифровой системы.

Разрабатываемая микро-ЭВМ будет обладать возможностью выполнять 16 различных команд. В рамках данного курсового проекта предусмотрено расширение функциональных возможностей.

В разрабатываемом устройстве так же будет присутствовать кэш-память. Кэш-память напрямую влияет на скорость вычислений и помогает процессору работать с более равномерной загрузкой.

1 РАЗРАБОТКА СТРУКТУРЫ МИКРО-ЭВМ

Задание, согласно варианту, приведено в таблицах 1.1 – 1.4.

Таблица 1.1 – Задание часть первая

№ варианта	Тип архитектуры	Разрядность шин		Память		
		Адреса	Данных	ПЗУ	ОЗУ	Тип адресации
54	Принстонская	14	9	Синхронная	Синхронная	Косвенная регистровая

Таблица 1.2 – Задание часть вторая

Команда условного перехода	РОН	КЭШ		
		k	Алгоритм замещения строк	Синхронизация с памятью
JAZ	8	2	LFU	Сквозная с отображением

Таблица 1.3 – Задание часть третья

АЛУ				Арбитраж шин
Арифметические команды	Логические команды		Сдвиговые команды	
SUB	NOT	NXOR	ROR	Децентрализованный параллельный

Таблица 1.4 – Задание часть четвертая

Стек		Схема предсказания Переходов			КПДП	Конвейер
Объем	Направление Роста	Тип автомата	Бит	Тип шаблона		
10	Вверх	A5	5	PC		

1.1 Функциональный состав микро-ЭВМ

В основу архитектуры современных персональных компьютеров положен магистрально-модульный принцип. Модульный принцип позволяет потребителю самому комплектовать нужную ему конфигурацию компьютера и производить при необходимости ее модернизацию. Модульная организация компьютера опирается на магистральный (шинный) принцип обмена информацией между устройствами. По этой причине было решено выбрать именно эту архитектуру для реализации курсового проекта.

Набор функциональных модулей микро-ЭВМ может отличаться в зависимости от задач, выполняемых устройством, и назначения устройства. Однако можно выделить несколько модулей, характерных для каждого устройства данного типа: модуль памяти, модули выборки и выполнения команд, а также модуль кэш-памяти.

Блок памяти, разрабатываемой микро-ЭВМ, состоит из ПЗУ (ROM) и ОЗУ (RAM), объединённых в один общий блок, что не позволяет разделить шины команд и данных, данный аспект приводит к уменьшению производительности устройства в целом. Разрядность ячеек в модуле соответствует ширине шины данных, заданной по варианту, что будет описано при разработке архитектуры системы команд.

Блок выборки команд отвечает за считывание команд из памяти и наращивание значения счетчика команд.

Блок исполнения команд включает в себя блок декодирования команд, устройство управления, арифметико-логическое устройство, стек и блок регистров общего назначения. Далее приведено описание каждого структурного блока.

Устройство управления обеспечивает выполнение последовательности микроопераций в соответствии с кодом текущей команды и организует выборку команд в соответствии с выполняемой программой. Помимо вышеперечисленного на этот блок отводится задача тактирования команд и управления их различными стадиями.

Одним из основных узлов вычислительной системы является арифметико-логическое устройство. Назначением блока АЛУ является формирование результата арифметических и логических команд над операндами, попадающими на вход. На вход АЛУ, помимо операндов, подается некоторый набор управляющих сигналов, за счет чего на выходе формируется требуемое значение, которое зависит от выполняемой операции.

Для хранения данных при работе микро-ЭВМ может использоваться стек. Стек – это память с линейно упорядоченными ячейками и специальным механизмом доступа, исключающим необходимость указания адреса при обращении. Для доступа к регистрам стека используются команды PUSH и POP.

Важной составляющей современных процессоров является блок внутренней памяти, реализованный в виде набора программно-доступных регистров, называемых регистрами общего назначения. При наличии данного блока операнды команд могут размещаться в одной из двух запоминающих сред – в ОЗУ или в РОН. Применение данного блока позволяет сократить время выполнения операций, за счет быстрого доступа к операндам.

Блок кэш-памяти является неотъемлемой частью современных микро-ЭВМ. Данный модуль используется процессором для уменьшения среднего времени доступа к памяти. Достигается это посредством хранения часто используемых данных из основной памяти, т.е. когда процессору необходимо обратится в память для чтения или записи данных, он первоначально

проверяет, доступна ли их копия в кэше. В случае успеха проверки процессор производит операцию, используя кэш, что значительно быстрее использования основной памяти устройства. Данные между кэшем и основной памятью передаются блоками фиксированного размера.

1.2 Разработка системы команд

В разрабатываемом устройстве применяется строго фиксированная длина команды, что увеличивает количество памяти, необходимое для хранения кода программы, но повышает быстродействие системы, т.к. не требуется вычисление адреса памяти, по которому хранится команда, и упрощает представление архитектуры системы команд.

Общая структура команды для прямой регистровой адресаций приведена в таблице 1.5.

Общая структура команды для косвенной регистровой адресаций приведена в таблице 1.6.

Общая структура команды для непосредственной адресации приведена в таблице 1.7.

Общая структура команды для прямой адресации приведена в таблице 1.8.

Структура команды HLT приведена в таблице 1.9.

Таблица 1.5 – Структура команды с прямой регистровой адресацией

Безразличные биты	Адрес регистра	Адрес регистра	КОП
17 бит	3 бита	3 бита	4 бита

Таблица 1.6 – Структура команд с косвенной регистровой адресацией

Безразличные биты	Адрес регистра	Адрес регистра	Адрес регистра	КОП
14 бит	3 бита	3 бита	3 бита	4 бита

Таблица 1.7 – Структура команды с непосредственной адресацией

Безразличные биты	Адрес памяти	КОП
9 бит	14 бит	4 бита

Таблица 1.8 – Структура команды с прямой адресацией

Безразличные биты	Адрес памяти	Адрес регистра	КОП
6 бит	14 бит	3 бита	4 бита

Таблица 1.9 – Структура команды HLT

Безразличные биты	КОП
23 бита	4 бита

В таблице 1.10 представлен список команд микро-ЭВМ с соответствующими типами адресации и кодами операций.

Таблица 1.10 – Архитектура системы команд микро-ЭВМ

Команда	Адресация операнда	КОП
MOV Reg, [Reg]	Косвенная регистровая	0000
SUB Reg, [Reg]	Косвенная регистровая	0001
NOT [Reg]	Косвенная регистровая	0010
NXOR Reg, [Reg]	Косвенная регистровая	0011
ROR Reg, [Reg]	Косвенная регистровая	0100
JMP Adr	Непосредственная	0101
JAZ Adr	Непосредственная	0110
MOV Adr, Reg	Прямая	0111
MOV Reg, Adr	Прямая	1000
SUB Reg, Reg	Прямая регистровая	1001
NOT Reg	Прямая регистровая	1010
NXOR Reg, Reg	Прямая регистровая	1011
ROR Reg, Reg	Прямая регистровая	1100
PUSH Reg	Прямая регистровая	1101
POP Reg	Прямая регистровая	1110
HLT	–	1111

1.3 Взаимодействие блоков микро-ЭВМ при выполнении программ

Выполнение команды начинается с ее считывания из памяти и заполнения специальных регистров. Команда записывается в регистр IR, состоящий из трех регистров, последовательно считывающих часть команды с общей шины данных, после чего сохраняются адреса операндов. Значение счетчика команд IP постепенно наращивается при считывании команды и увеличивается после чтения последней части команды, при выполнении перехода, значение счетчика устанавливается в указанное в команде значение.

Значение регистра IR поступает на декодер для определения выполняемых инструкций. Если считанная команда не является командой HLT, то управление передается на один из блоков, отвечающих за выполнение заданной последовательности действий. При определении команды HLT работа микро-ЭВМ приостанавливается.

Когда выполнение операции завершено блок команды устанавливает в единичное значение флаг успешного выполнения команды, после чего блок управления завершает выполнение команды и обнуляется в случае поступления новой команды начинается ее выполнение.

2 РАЗРАБОТКА ОСНОВНЫХ УСТРОЙСТВ МИКРО-ЭВМ

С функциональными схемами блоков можно ознакомиться в конце пояснительной записки.

2.1 Блок памяти

Блок памяти представляет собой объединение синхронных ROM и RAM памяти с общим адресным пространством, которое используется как для хранения операндов, так и для хранения команд. Адресное пространство разбито на две равные части, первая половина памяти отведена для команд, вторая половина для данных. Это реализовано путем обработки старшего бита адреса ячейки и определением к какому типу данных относится данное значение. Блок реализован при помощи стандартных модулей lpm_ram_dq и lpm_rom. Разрядность каждой ячейки памяти в данных блоках была выбрана исходя из варианта задания, и равна 9 битам. Схема внутренней структуры блока памяти приведена на рисунке 2.2.

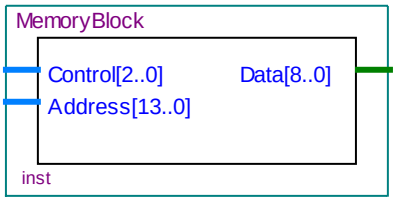

Рисунок 2.1 – УГО блока памяти

Входные сигналы:

- Address – шина адреса данных для чтения или записи;
- Data – входная шина данных;
- Control – шина из трёх сигналов: сигнала, разрешающего считывание данных, сигнала, разрешающего запись данных и тактирующего сигнала clock.

Выходной сигнал:

- Data – шина вывода данных.

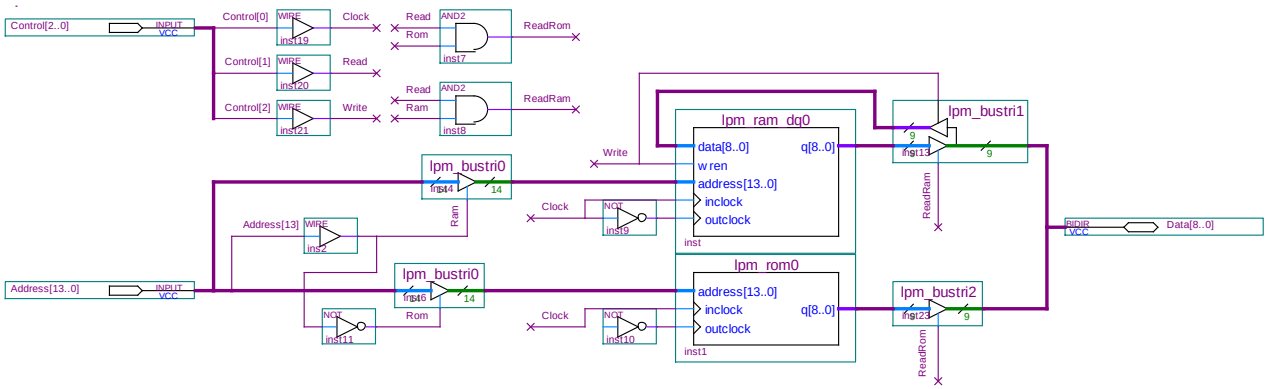

Рисунок 2.2 – Внутренняя структура блока памяти

2.2 Блок стек-памяти

Блок стека состоит из 10 элементов lpm_dff, декодера для выбора нужной ячейки и указателя на вершину стека.

Текущий указатель стека указывает на последний добавленный элемент. При выполнении команды push происходит проверка регистра указателя. Если регистр указывает на последний элемент стека, то формируется сигнал о переполнении, а сам указатель на вершину своего значения не изменяет. При выполнении команды pop так же осуществляется проверка значения регистра указателя. Если указатель на вершину отображает отсутствие элементов в стеке, то формируется сигнал об ошибке. Схема внутренней структуры блока стек-памяти приведена в конце ПЗ (Э1).

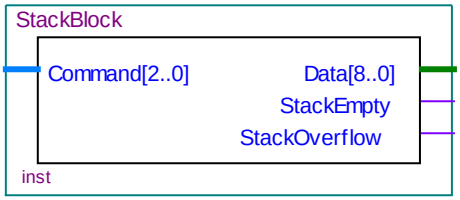

Рисунок 2.3 – УГО Стека

Входные сигналы:

- Data – входная шина данных;
- Control – шина из трех сигналов: сигнала push, сигнала pop и тактирующего сигнала.

Выходные сигналы:

- Data – шина вывода данных;
- StackOverflow – флаг переполнения стека;
- StackEmpty – флаг пустого стека.

2.3 Блок регистров общего назначения

Блок регистров общего назначения схож с реализацией стека. Он состоит из 8 запоминающих элементов lpm_dff и декодера, который определяет номер регистра. В силу того, что ширина шины адреса превышает ширину шины данных, в командах с косвенной регистровой адресацией было решено использовать сразу два регистра для хранения адреса операнда. Схема внутренней структуры блока РОН приведена в конце ПЗ (Э2).

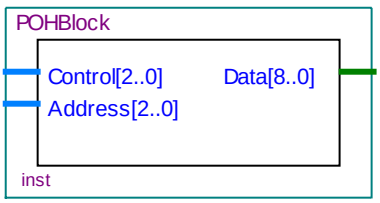

Рисунок 2.4 – УГО РОН

Входные сигналы:

- Address – шина адреса регистра;
- Data – входная шина данных;
- Control – шина из трёх сигналов: сигнала, разрешающего считывание данных, сигнала, разрешающего запись данных и тактирующего сигнала clock.

Выходной сигнал:

- Data – шина вывода данных.

2.4 Арифметико-логическое устройство

Блок АЛУ должен выполнять заданные в условии команды. АЛУ состоит из декодера, необходимого для выбора исполняемой команды, и блоков команд. Схема общей внутренней структуры блока АЛУ приведена в конце ПЗ (Э3).

Задача блока SUB состояла в выполнении арифметической команды SUB, суть которой заключается в вычислении разности двух операндов.

Команда NOT реализована с использованием стандартного блока lpm_inv.

Команда NXOR была реализована с помощью стандартного логического элемента XNOR.

Команда циклического сдвига ROR была реализована с помощью элемента lpm_clshift.

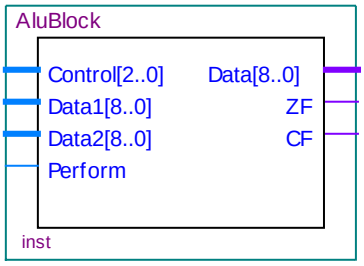

Рисунок 2.5 – УГО АЛУ

Входные сигналы:

- Data1 – первый операнд;
- Data2 – второй операнд;
- Control – код выполняемой команды;
- Perform – разрешающий сигнал;

Выходные сигналы:

- CF – вывод флага CF;
- ZF – вывод флага ZF;
- Data – шина вывода результата вычислений.

2.5 Организация кэш-памяти процессора

В связи с построением проекта на базе Принстонской архитектуры, блок кэш-памяти физически не разделен на отдельные блоки для данных и команд, что усложняет процесс его реализации. Задача блока кэш-памяти состоит в чтении и хранение данных и команд, что способствует ускорению работы, в связи с быстрым доступом к хранящимся данным.

Схемы внутренней структуры блока кэш-памяти и организации многомерности приведены в конце ПЗ (Э4 – Э6).

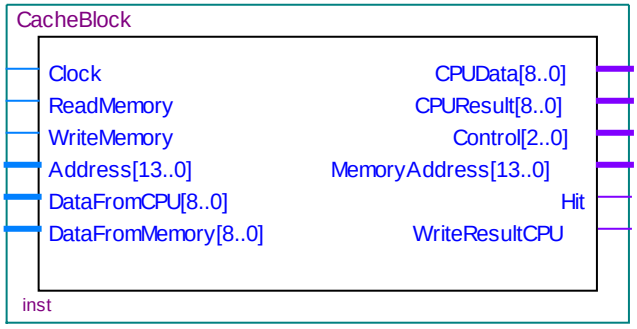

Рисунок 2.6 – УГО блока кэш-памяти

Входные сигналы:

- Address – шина адреса данных;
- DataFromMemory – входная шина считанных данных;
- DataFromCPU – входная шина данных для записи;
- ReadMemory – сигнал, разрешающий считывание данных;
- WriteMemory – сигнал, разрешающий запись данных;
- Clock – тактирующий сигнал.

Выходное сигналы:

- CPUData – шина вывода данных;
- MemoryAddress – шина адреса в памяти;
- Control – шина из трёх сигналов для управления: сигнала разрешающего чтение, сигнала разрешающего запись, и тактирующего сигнала;
- Hit – сигнал определения необходимых данных;
- WriteResultCPU – сигнал записи изменённого значения;
- CPUResult – шина для вывода изменяемых значений.

2.6 Управляющий блок. CPU

Входные сигналы:

- Clock – тактирующий сигнал;
- Data – шина входных данных из блока памяти.

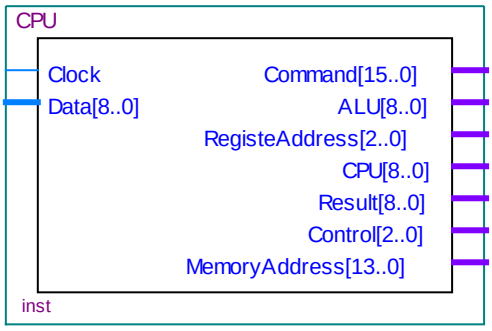

Рисунок 2.6 – УГО блока CPU

Выходные сигнал:

- Command – шина вывода команды;
- CPU – шина вывода данных;
- RegisterAddress – шина адреса регистра, содержащего операнд или адрес операнда в памяти;
- Control – шина управляющих команд для блока памяти;
- ALU – шина вывода результата вычислений операций блоком АЛУ;
- MemoryAddress – шина адреса для блока памяти;
- Result – шина для записи данных в память.

В состав общей схемы устройства входят блок CPU и блок памяти RAM и ROM с общим адресным пространством, в следствии чего в разрабатываемой микро-ЭВМ нет отдельных шин для данных и команд, что существенно замедляет работу устройства. Управляющий блок включает в себя блок регистров специального назначения, блоки команд и интерфейсы для их реализации, каждая команда имеет собственный блок управления, котором формируются соответствующие управляющие сигналы и реализован полный цикл выполнения команды. Задачей блока CU является формирование управляющих сигналов для реализации команд и управления всеми входящими в микро-ЭВМ модулями, такими как стековая память, арифметико-логическое устройство, регистры общего назначения, кэш память и другие, а также изменение состояния служебных регистров.

Со схемой устройства управления, а также с общей схемой микро-ЭВМ можно ознакомится в конце ПЗ (Э7 – Э8).

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ

3.1 Функциональное моделирование блоков стека и РОН

Блоки стека и регистров общего назначения имеют общую структуру и принцип работы.

На рисунке 3.1. представлен результат моделирование стека. Можно заметить, что при попытке извлечения данных из пустого стека выставляется флаг Empty. При попытке записи в заполненный стек выставляется флаг переполнения Overflow. На первых трех тактах происходит внесение данных в стек-память, на последующих трех, начиная с 35нс, происходит вынесение данных. Как видно из моделирования стековая память работает по принципу первый пришел, последний ушел.

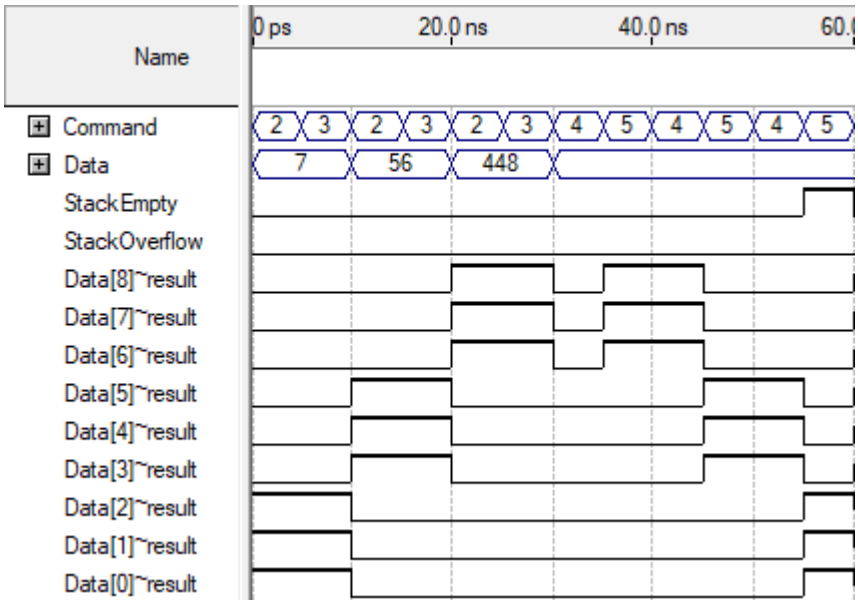

Рисунок 3.1 – Функциональное моделирование стека

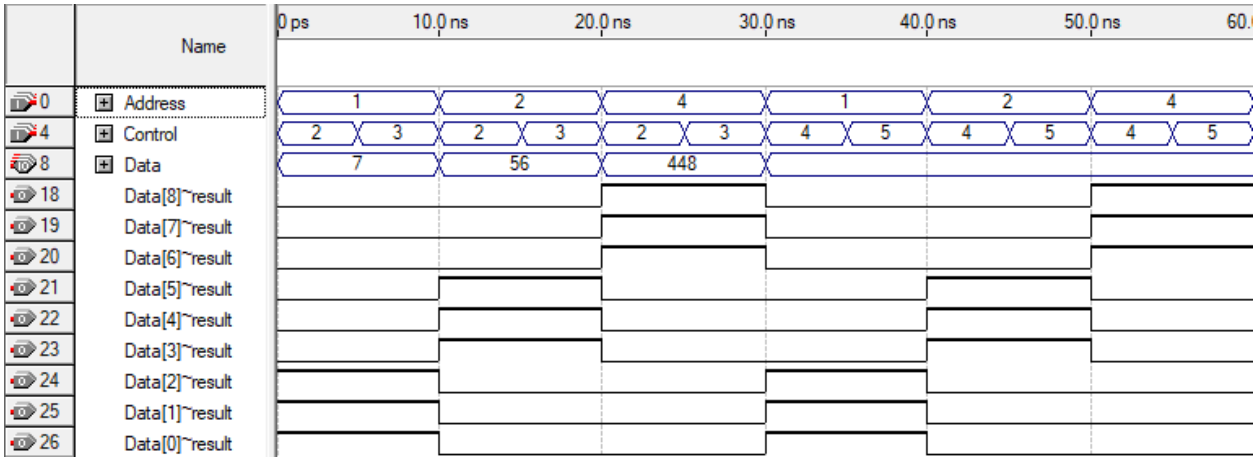

Рисунок 3.2 – Функциональное моделирование блока РОН

На рисунке 3.2. представлен результат моделирование РОН. Блок работает следующим образом: по наличию тактирующего сигнала и сигнала

запись в регистр, декодер выставляет сигнал разрешения записи на регистр, номер которого соответствует пришедшему адресу, а в случае чтения данных из регистра, декодер также анализирует пришедший адрес и выставляет сигнал разрешения чтения данных. Исполнение записи можно увидеть на первых трех тактах моделирования, исполнение чтения на последующих трех.

3.2 Функциональное моделирование блока памяти

Рисунок 3.3 отображает результат моделирования блока памяти. На первых трех тактах происходит чтение данных из первой половины адресного пространства, т.е. чтение команд, на последующих двух происходит чтение данных, что видно по адресам, на шестом такте происходит запись данных в память. На рисунках 3.4 и 3.5 представлен дамп памяти до и после моделирования.

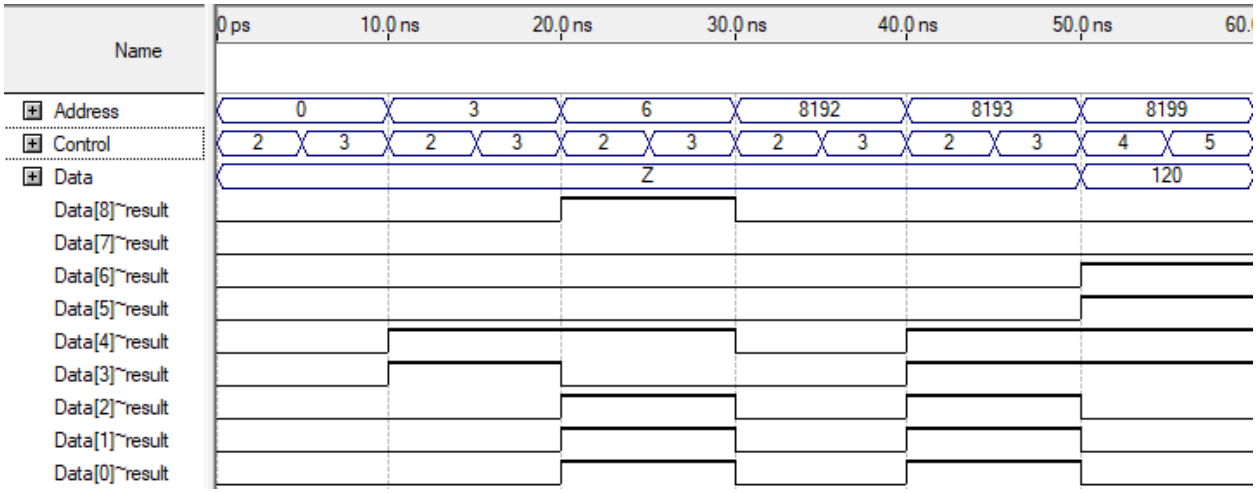

Рисунок 3.3 – Функциональное моделирование блока памяти

8192	31	16	123	14	0	0	0	0
8200	0	0	0	0	0	0	0	0
8208	0	0	0	0	0	0	0	0
8216	0	0	0	0	0	0	0	186

Рисунок 3.4 Дамп памяти до моделирования

8192	31	16	123	14	0	0	0	120
8200	0	0	0	0	0	0	0	0
8208	0	0	0	0	0	0	0	0
8216	0	0	0	0	0	0	0	186
8224	0	0	0	0	0	0	0	0

Рисунок 3.5 Дамп памяти после моделирования

3.3 Функциональное моделирование АЛУ

Рисунок 3.6 отображает результат моделирования блока АЛУ. На первом такте реализована команда SUB. На втором такте реализована команда

NOT. На третьем такте реализована команда NXOR. На четвертом такте реализована команда ROR.

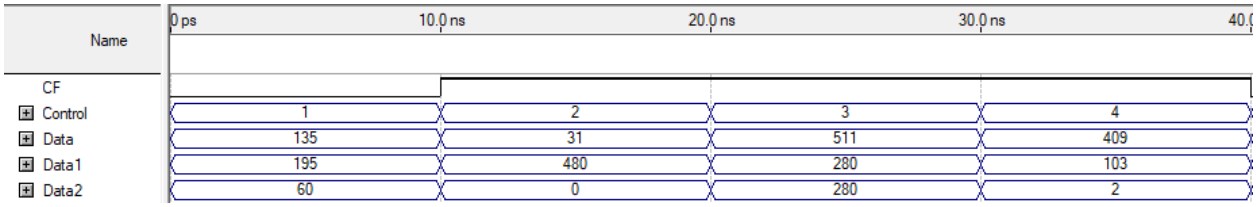

Рисунок 3.6 – Функциональное моделирование блока АЛУ

3.4 Моделирование общей схемы

Как уже было сказано ранее в общую схему микро-ЭВМ входят блок памяти, а также блок CPU, состоящий в свою очередь из блоков АЛУ, стек-памяти, кэша и устройства управления. Взаимодействие блоков начинается по подаче тактирующего сигнала Clock на блок CPU, который в свою очередь идет на устройство управления, где и формируются управляющие сигналы. После начала тактирования, устройством управления формируется сигнал о чтение команд из блока кэша, который в свою очередь формирует сигнал чтения команд из памяти. После выполнения последовательного чтения всей команды, в течении которого регистр IR загружает первую команду, происходит её декодирование и исполнение. Как уже было сказано ранее исполнением команды занимается специально разработанный блок. После того как команда завершена, блок исполнения команды выдает сигнал о завершении функционирования, после чего указатель IP наращивается и происходит последовательное чтение следующей команды из кэш-памяти.

Для моделирования системы использовалась программа, список команд которой приведен в таблице 3.1.

Таблица 3.1 – Список команд программы

№	Запись в символьном представлении	Запись в шестнадцатеричной системе	Запись в двоичной системе
1	MOV Reg1, $8192	100018	00000010000000000000011000
2	MOV $8222, Reg1	100F17	00000010000000011110010111
3	PUSH Reg1	1D	00000000000000000000011101
4	MOV Reg2, $8193	1000A8	00000010000S000000010101000
5	JMP $16	105	00000000000000000100000101
6	POP Reg6	6E	00000000000000000001101110
7	SUB Reg1, Reg2	119	00000000000000000100011001

Продолжение таблицы 3.1

8	NOT Reg6	6A	00000000000000000000001101010
9	ROR Reg1, Reg2	11C	00000000000000000000100011100
10	JAZ $32	205	00000000000000000001000000101
11	NXOR Reg1, Reg6	31B	00000000000000000001100011011
12	SUB Reg6, [Reg1, Reg2]	1911	00000000000000011001000010001
13	NOT [Reg1, Reg2]	112	00000000000000000000100010010
14	NXOR Reg6, [Reg1, Reg2]	1913	00000000000000011001000010011
15	ROR Reg6, [Reg1, Reg2]	1914	00000000000000011001000010100
16	MOV Reg4, [Reg1, Reg2]	1110	00000000000000010001000010000
17	HLT	F	00000000000000000000000001111

На рисунке 3.7 приведено моделирование чтения и выполнение команды MOV Register, Address, на 40нс происходит последовательное чтение команды в кэш-память, после чего на 155нс начинается исполнение команды MOV, последовательно происходит чтение данных в кэш-память, это видно исходя из моделирования, после чего на 240нс происходит вывод необходимых данных на общую шину данных CPU и последующее формирование управляющих сигналов для РОН, а также запись необходимого значения в первый регистр.

В дальнейшем чтение команды будет опускаться, на рисунках будет отображено непосредственно выполнение команды.

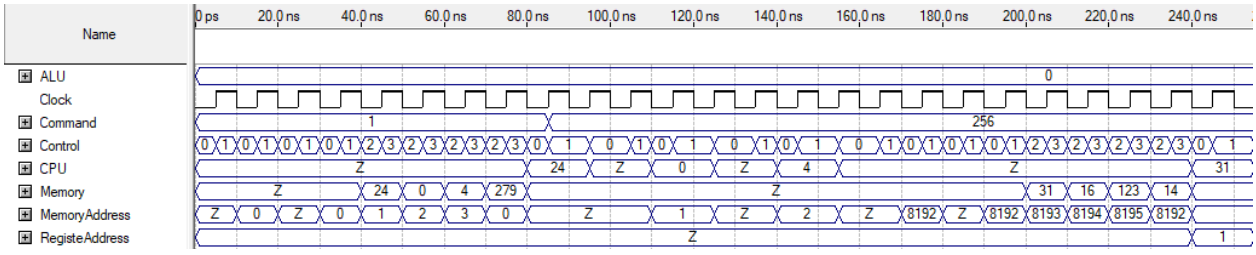

Рисунок 3.7 – Функциональное моделирование чтения и выполнение команды MOV Register, Address

На рисунке 3.8 приведено моделирование команды MOV Address, Register на 450нс происходит запись данных из первого регистра в память, в силу того, что запись строка кэша по варианту соответствует сквозной с отображением на 470нс происходит запись строки с обновленным значением в кэш-память.

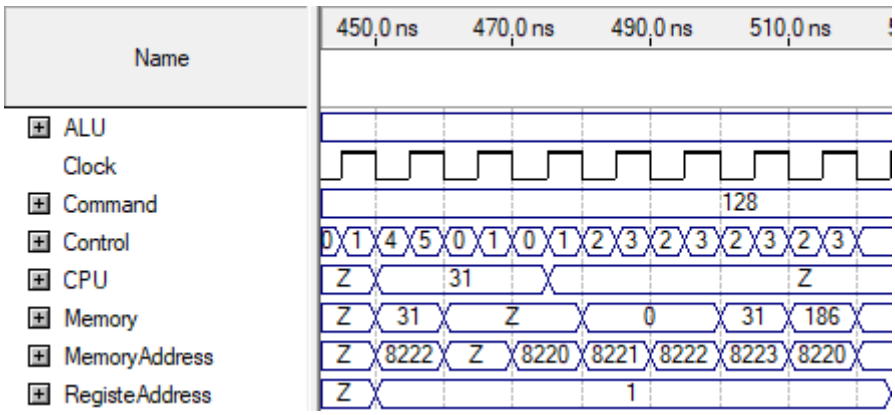

Рисунок 3.8 – Функциональное моделирование команды MOV Address, Register

На рисунке 3.9 приведено моделирование команды PUSH Register, после чтения команды на 705нс происходит занесение значения из первого регистра в стек.

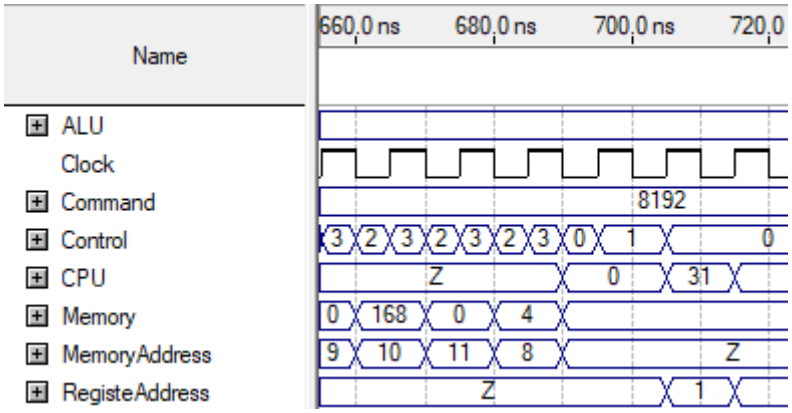

Рисунок 3.9 – Функциональное моделирование команды PUSH Register

На рисунке 3.10 приведено моделирование команды JMP Address, сразу после обработки команды начинается чтение последующей из памяти.

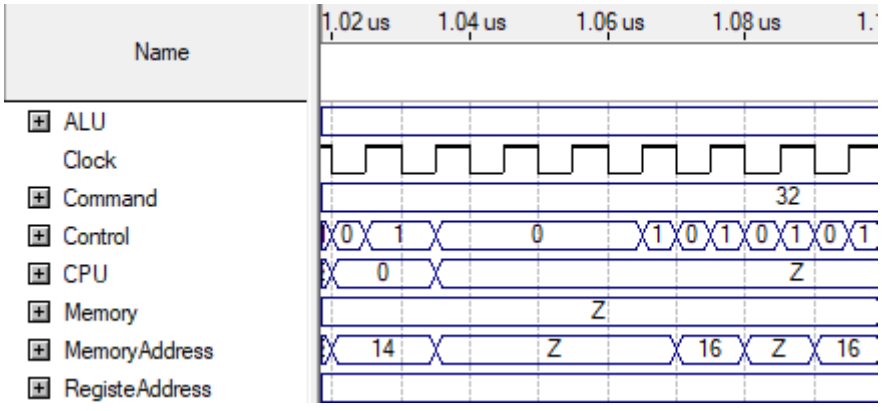

Рисунок 3.10 – Функциональное моделирование команды JMP Address

На рисунке 3.11 приведено моделирование команды POP Register, на 1225нс происходит вынесение данных из стековой памяти в шестой регистр.

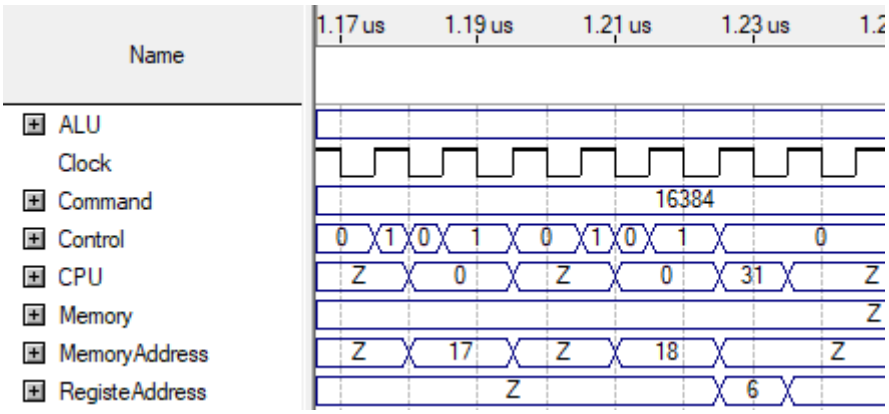

Рисунок 3.11 – Функциональное моделирование команды POP Register

На рисунке 3.12 приведено моделирование команды SUB Register, Register, как видно из моделирования на 1415нс происходит последовательное чтение данных из регистров, после чего происходит вычисление их разности.

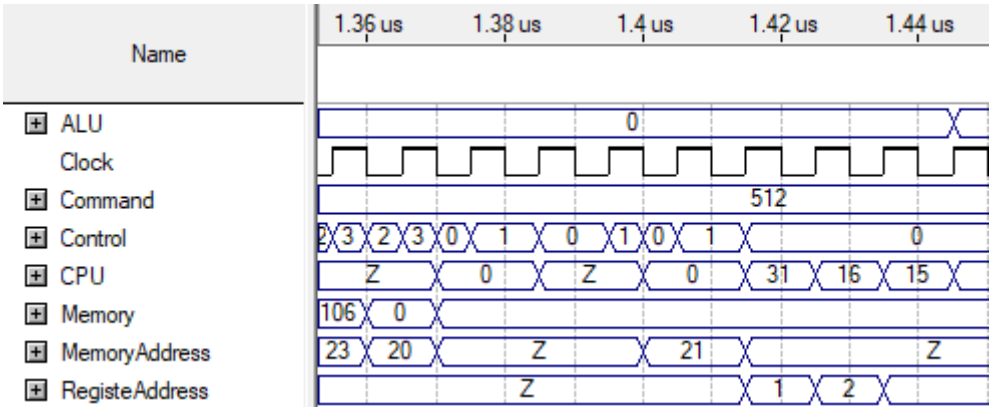

Рисунок 3.12 – Функциональное моделирование команды SUB Register, Register

На рисунке 3.13 приведено моделирование команды NOT Register, Register, на 1625нс происходит чтение данных из регистра, после чего с задержкой в один такт мы получаем инвертированное значение прочитанных данных.

На рисунке 3.14 приведено моделирование команды ROR Register, Register, на 1765нс происходит последовательное чтение данных из регистров, после чего происходит циклический сдвиг данных из второго регистра, на количество разрядов, равное значению трех младших бит второго регистра.

На рисунке 3.15 приведено моделирование команды JAZ Address, сразу после обработки команды начинается чтение последующей из памяти.

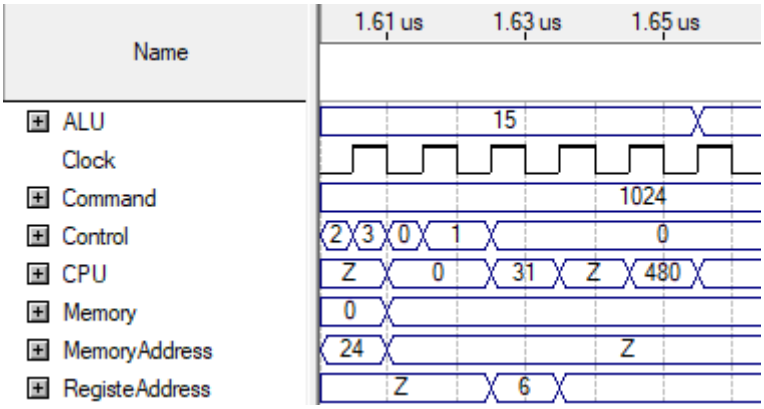

Рисунок 3.13 – Функциональное моделирование команды NOT Register, Register

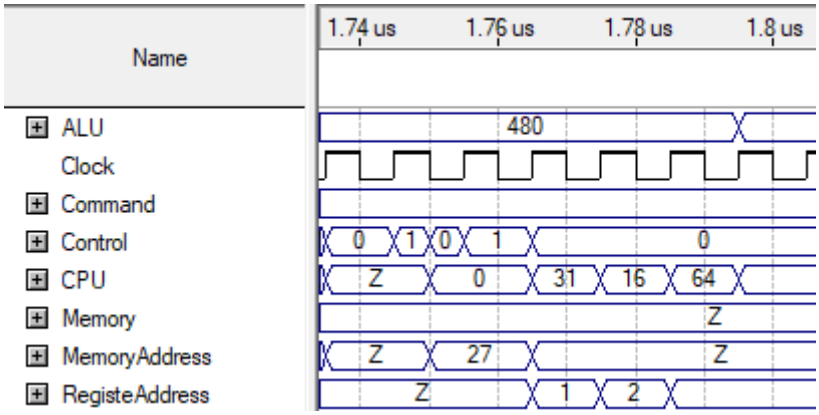

Рисунок 3.14 – Функциональное моделирование команды ROR Register, Register

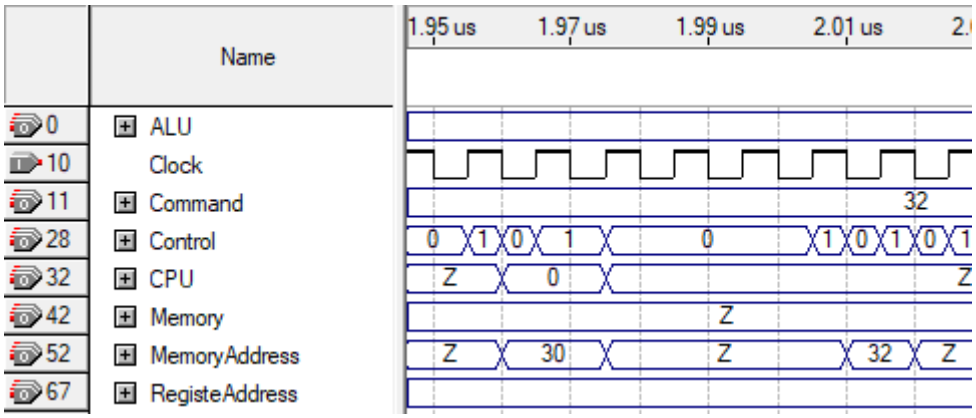

Рисунок 3.15 – Функциональное моделирование команды JAZ Address

На рисунке 3.16 приведено моделирование команды NXOR Register, Register, на 2165нс происходит последовательное чтение данных из регистров, после чего выполняется вычисление результата, как видно из моделирования при подаче двух одинаковых значений мы получаем максимально возможное значения, что свидетельствует о правильности выполнения команды.

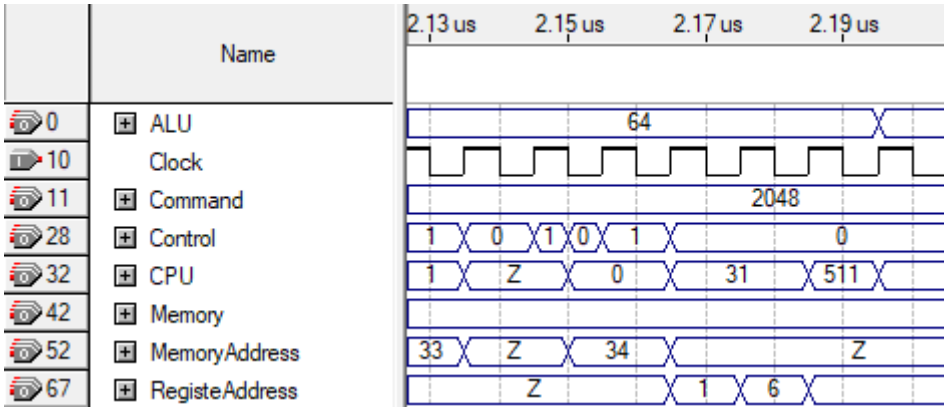

Рисунок 3.16 – Функциональное моделирование
команды NXOR Register, Register

Команды АЛУ при разных типах адресации имеют одинаковую структура выполнения, поэтому далее будет приведено описание только для команды SUB при косвенной регистровой адресации.

На рисунке 3.17 приведено моделирование команды SUB Register, [Register, Register], на 2375нс происходит последовательное чтение данных из регистров, после чего из полученного из первых двух регистров адреса считываются данные и на 2505нс происходит вывод результата вычисления.

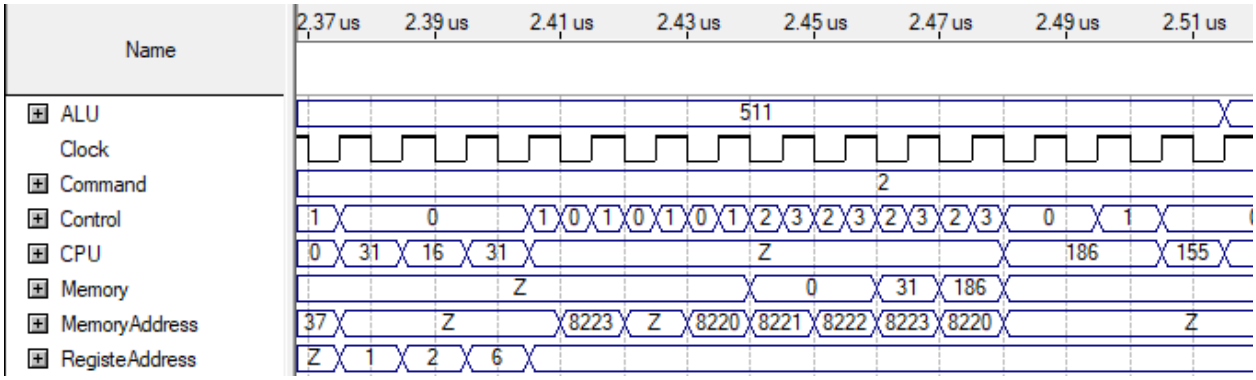

Рисунок 3.17 – Функциональное моделирование
команды SUB Register, [Register, Register]

8192	31	16	123	14	0	0	0	0
8200	0	0	0	0	0	0	0	0
8208	0	0	0	0	0	0	0	0
8216	0	0	0	0	0	0	0	186
8224	0	0	0	0	0	0	0	0

Рисунок 3.18 Дамп памяти до моделирования

8192	31	16	123	14	0	0	0	0
8200	0	0	0	0	0	0	0	0
8208	0	0	0	0	0	0	0	0
8216	0	0	0	0	0	0	31	186
8224	0	0	0	0	0	0	0	0

Рисунок 3.19 Дамп памяти после моделирования

ЗАКЛЮЧЕНИЕ

В рамках данного курсового проекта была разработана микро-ЭВМ, обладающая возможность выполнить 16 различных команд. Данная ЭВМ спроектирована так, что может быть положена за основу для дальнейшего расширения функциональности. ЭВМ может быть доработана такими составляющими, как конвейер, предсказатель переходов, арбитраж, контроллер прямого доступа к памяти, так же имеется возможность расширение команд АЛУ, возможна оптимизация работы путем пересмотра блоков команд в управляющем блоке и уменьшения количества тактов, затрачиваемых на выполнение операций.

Функциональное моделирование отдельных блоков и всей системы в целом показывает, что команды выполняются верно. Электрические схемы блоков и их частей представлены в конце пояснительной записки.