Национальный исследовательский ядерный университет «МИФИ»

Факультет Кибернетики и информационной безопасности

Кафедра «Компьютерные системы и технологии»

Пояснительная записка к проекту на тему «Разработка системы памяти»

Студент гр. М19-512 Петров А.Н.

Руководитель Скитев А.А.

Москва 2020

Вариант 21

РАЗРАБОТКА СИСТЕМЫ ПАМЯТИ

Цель работы: изучение основ структурного проектирования вычислительных систем, овладение современными САПР, получение навыков патентного поиска и исследовательской деятельности.

1. Необходимо разработать систему памяти, включающую в себя кэш память заданного размера. Система памяти должна принимать запросы на чтение/запись со стороны системной шины, выполнять поиск запрошенных данный в кэш памяти. В случае кэш промаха – перенаправлять запрос в контроллер оперативной памяти. Результаты запроса должны сохраняться в кэш, при необходимости вытеснив более старые данные в соответствии с заданным алгоритмом вытеснения. В случае кэш попадания или после получения данных из ОП на запрос системной шины должен быть сформирован корректный ответ. Параметры системы памяти приведены в таблице 1.

Системная шина, кэш-память и интерфейс с ОП находятся в различных тактовых доменах.

Интерфейс с системной шиной включает в себя следующие сигналы:

Входы: *sys_clk, sys_rst_n, sys_addr[15:0], sys_wr, sys_rd, sys_wdata[31:0],sys_bval[3:0].*

Выходы: *sys_rdata[31:0], sys_ack.*

Интерфейс с контроллером оперативной памяти включает в себя следующие сигналы:

Входы: *ram_clk, ram_rst_n, ram_rdata[7 : 0], ram_rack.*

Выходы: *ram_addr[11 : 0], ram_wdata[7 : 0], ram_avalid, ram_rnw.*

2. Провести патентный поиск по изобретениям и полезным моделям с глубиной поиска 25 лет (1994-2019) по заданным в таблице 1 алгоритмам и странам. Отчет по поиску оформить в соответствии с ГОСТ Р 15.011-96, в т.ч. аналитическая часть, Таблица В.6.1, в приложении рефераты патентов.

Таблица 1. Параметры задания

Тип кэш-памяти	8-ми канальная
Объем кэш-памяти, байт	2048
Разрядность шины данных ОП, байт	1
Разрядность строки кэш-памяти, байт	16
Алгоритм вытеснения	LRU
Алгоритм записи	обратная
Алгоритмы для патентного поиска	записи
Страны для патентного поиска	Россия,Франция,США

Этапы выполнения работы:

1. Подготовить обзор кэш памяти.
2. Выполнить проектирование системы памяти.
3. Провести патентный поиск.
4. Подготовить пояснительную записку по выполненной работе.

Базы данных для патентного поиска:

Роспатента	http://www.fips.ru
ЕПВ Espacenet	http://ep.espacenet.com
США USPATEULL	http://www.uspto.gov
Google Patent Search	http://www.google.com/patent
Всемирной организации интеллектуальной собственности (ВОИС) в системе PATENSCOPE	http://patentscope.wipo.int/search/en/search.jsf

Оглавление

1 Обзор кэш-памяти

Информация, которая оперативно используется при выполнении программ, хранится в основной памяти (динамическом ОЗУ). Кэш-память предназначена для хранения копии информации, с которой в данный момент работает процессор. Имеет более высокое быстродействие по сравнению с оперативной, однако меньший размер. Применение кэш-памяти позволяет повысить производительность вычислительной системы при правильной реализации. Кэш-память чаще всего реализуется статическим ОЗУ (SRAM), которое значительно дороже динамического ОЗУ, поэтому основную память в целом на них не выполняют.

На рисунке 1.1 представлена структура системы памяти с кэшированием.

Рисунок 1.1 - Структура системы памяти с кэшированием

При такой организации системы памяти при чтении сначала выполняется обращение к кэш-памяти. Если в кэше имеется копия запрашиваемых данных, то он вырабатывает сигнал Hit (попадание) и выдает данные на общую шину данных. В противном случае сигнал Hit не вырабатывается и выполняется чтение из основной памяти RAM с одновременным обновлением содержимого кэша путем записи в него нового блока данных.

Процессор CPU обращается к кэш-памяти так же, как и к ОП, формируя обычный адрес и управляющие сигналы.

Эффективность применения кэширования зависит от ряда факторов, наиболее важные из которых следующие:

1. Ёмкость.
2. Способ отображения основной памяти на кэш-память.
3. Размер блока.
4. Алгоритм замещения информации в заполненной кэш-памяти.
5. Алгоритм согласования содержимого основной и кэш-памяти.
6. Число уровней кэш-памяти.

1.1 Организация

Размер кэша всегда меньше размера ОП, поэтому необходимо, чтобы вместе с каждым блоком информации, сохраняемым в кэше, сохранялся признак по которому данные в ОП можно найти, чаще всего это адрес этого блока данных в ОП.

Разрядность адреса основной памяти можно вычислить как:

$$n = log_2(N),$$

где N – емкость оперативной памяти.

Массив ОП делится на B блоков, содержащих по K байт.

Количество блоков в основной памяти вычисляется следующим образом:

$$B = \frac{N}{K},$$

где n – N – емкость оперативной памяти, а K – размер блока ОП.

Следовательно, для адресации блоков требуется $b = \log_2 B$ разрядов.

Кэш-память данных состоит из строк, размер которых идентичен размеру блока ОП.

Число строк кэша C значительно меньше числа блоков ОП:

$$C = \frac{P}{K},$$

где P – емкость кэш-памяти, K – размер блока ОП. То есть для адресации строк кэш-памяти необходимо $c = \log_2 C$ разрядов.

Для адресации байта в строке необходимо $k = \log_2 K$ разрядов.

Общая разрядность для памяти данных кэша составит (c+k) разрядов.

На рисунке 1.2 представлена структура ОП и структура кэш-памяти.

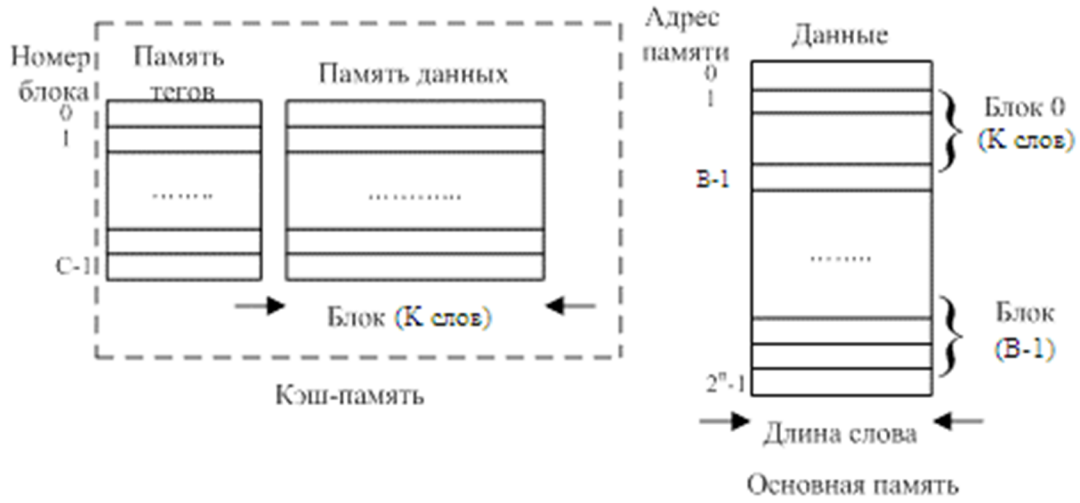

Рисунок 1.2 – Организация ОП и кэш-памяти

При размещении блоков в кэш-памяти строкам кэша могут соответствовать определенные блоки ОП. Так как число блоков ОП значительно превышает число строк кэша, строкам кэша назначается тэг, указывающий, копия какого блока хранится в них в данное время. Тэгом служит старшая часть адреса, поступающего от процессора.

Кэш-память структурно состоит из двух частей – памяти данных и памяти тэгов. Поля адреса, поступающего от процессора, играют в кэшированной памяти разные роли.

На рисунке 1.3 представлены поля адреса, поступающего от процессора.

Рисунок 1.3 - Поля адреса

Адрес, поступающий от процессора, разбивается на следующие поля:

- младшие разряды адреса, $k = \log_2 K$, отводятся под смещение. Смещение совпадает с адресом слова в строке кэша, поскольку емкости строки и блока равны и слова расположены в них идентично;
- следующие разряды, $c = \log_2 C$, отводятся под индекс, который адресует строку (группы строк) кэш-памяти.
- старшие разряды, (n-c-k), называются тэгом. По сравнению тэга адреса с хранящимся тегом определяется наличие запрашиваемого слова в хранящейся строке.

1.2 Отображение ОП на кэш-память

Отображение блока ОП на кэш состоит в копировании этого блока в строку кэш-памяти. Номер строки кэша выбирается в соответствии со следующими алгоритмами:

- прямое отображение;
- полностью ассоциативное отображение;
- наборно-ассоциативное отображение.

1.2.1 Прямое отображение

В кэш-памяти с прямым отображением адрес строки равен адресу блока в ОП взятом по модулю числа строк в кэше.

Адрес, поступающий от процессора, разбивается на тэг, на индекс, который адресует строки кэша, и на смещение, являющееся адресом слова в строке кэша.

В данном случае строка из ОП может быть отображена в единственную строку кэша, но в каждую строку кэша может быть отображено много возможных строк ОП.

На рисунке 1.4 представлено соответствие адресов блоков и строк, и организация кэш-памяти с прямым отображением.

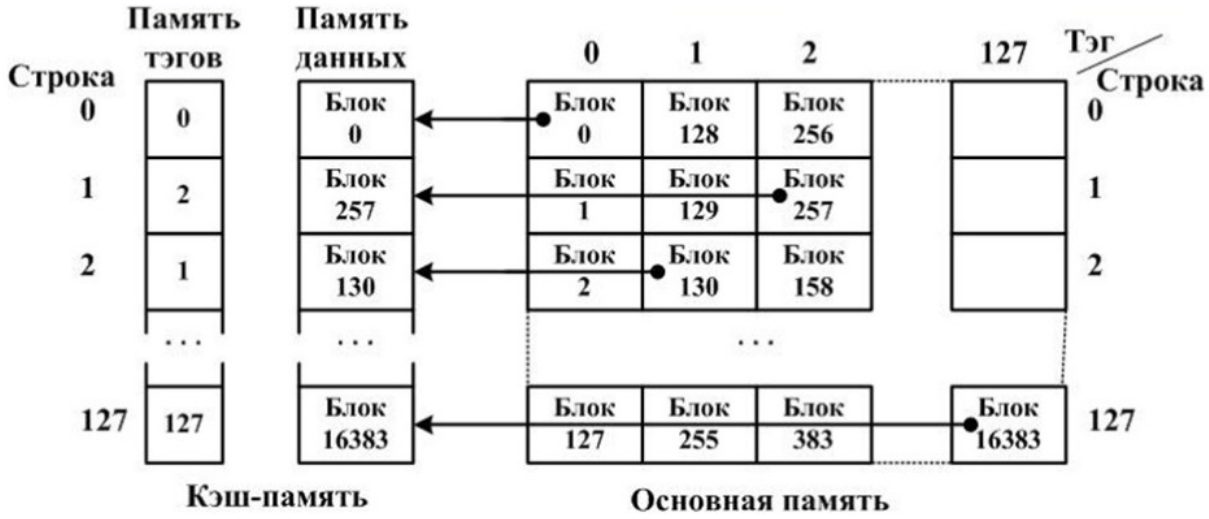

Рисунок 1.4 - Соответствие адресов блоков и строк, и организация кэш-памяти с прямым отображением

Кэш попадание вырабатывается при совпадении тэга, извлеченный по индексу из памяти тэгов и тэга пришедший с процессора. При загрузке из ОП заменяется вся строка.

Достоинством кэш-памяти с прямым отображением является минимальное время поиска нужной строки в кэше, а также ее простая организация.

Недостатком является то, что каждому блоку из ОП соответствует лишь одна строка кэш-памяти. При последовательном обращении к словам из двух различных блоков, отражаемых на одну и ту же строку кэш-памяти, будет происходить постоянное замещение данной строки. Поэтому вероятность кэш попадания в случае прямого отображения снижается.

На рисунке 1.5 показана схема организации кэш-памяти с прямым размещением.

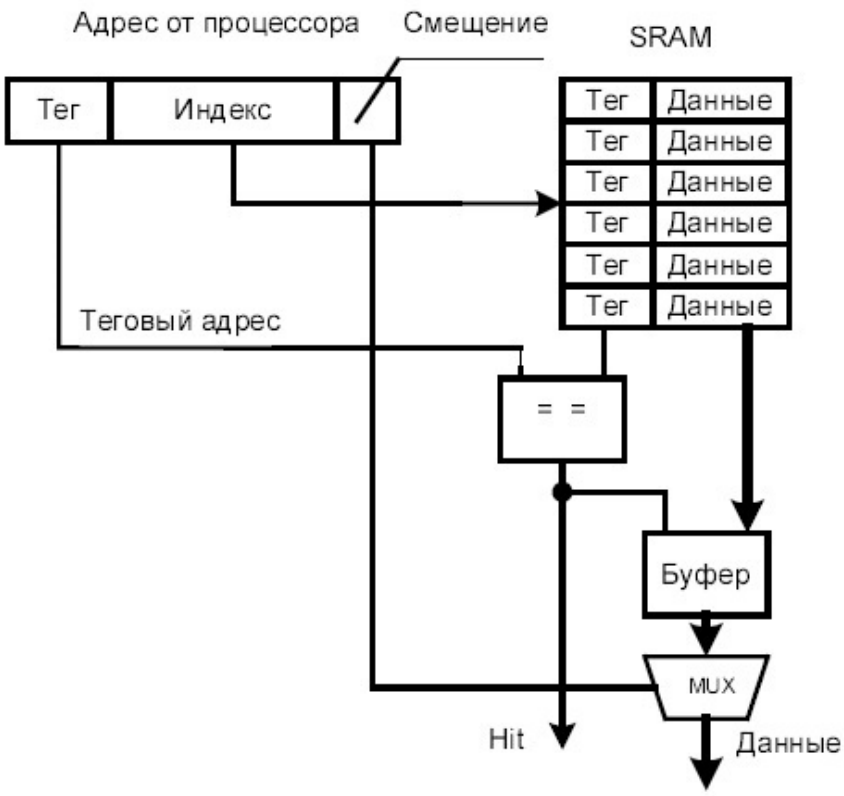

Рисунок 1.5 - Организация кэш-памяти с прямым отображением

1.2.2 Полностью ассоциативное отображение

Полностью ассоциативное отображение разрешает загрузку любого блока ОП в любую строку кэш-памяти. Логика управления кэш-памяти выделяет в адресе ОП два поля: поле тэга и поле слова. Поле тэга совпадает с адресом блока основной памяти. Для проверки наличия копии блока в кэш-памяти логика управления кэша должна одновременно проверить тэги всех строк на совпадение с полем тэга адреса.

На рисунке 1.6 представлена структура полностью ассоциативной кэш-памяти.

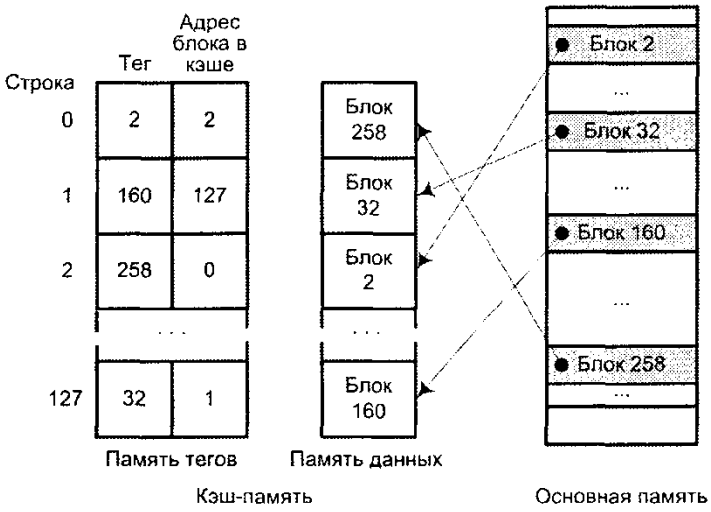

Рисунок 1.6- Структура полностью ассоциативной кэш-памяти

При любых обменах адрес запрашиваемой информации сравнивается с полями "тэг" всех строк кэш-памяти и при совпадении их в любой строке устанавливается сигнал Hit. При чтении и Hit = 1 данные выдаются на шину данных, причем смещение адресует читаемое слово, выбирая его из блока с помощью мультиплексора. Если же совпадений нет (Hit = 0), то происходит чтение из основной памяти и блок данных вместе с адресом помещается в свободную или наиболее давно не используемую строку кэш-памяти.

На рисунке 1.7 показана схема организации кэш-памяти полностью ассоциативного отображения.

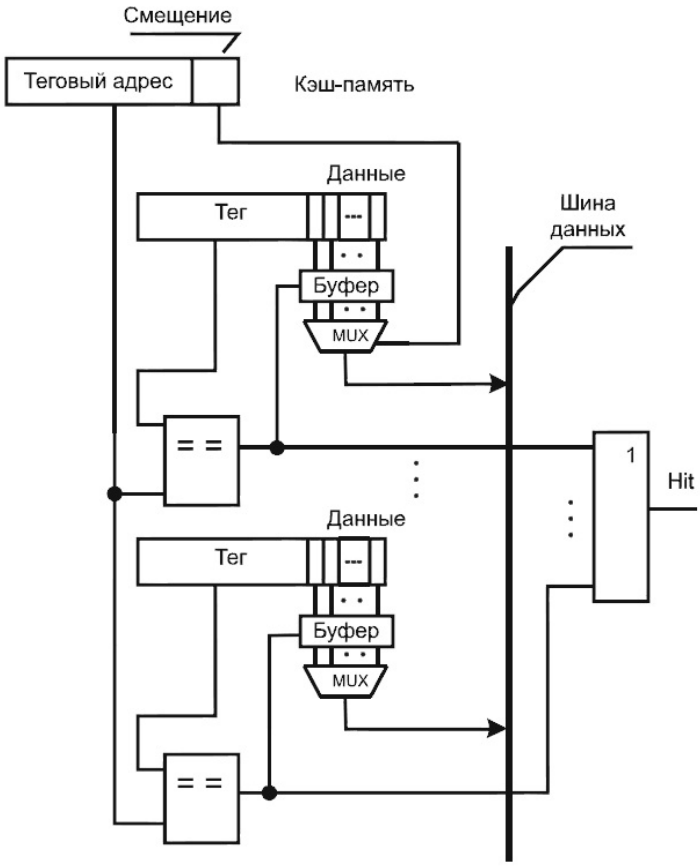

Рисунок 1.7 - Схема организации кэш-памяти полностью ассоциативного отображения

Достоинством данного отображения является максимальная эффективность использования места в кэш-памяти, а недостатком – сравнительно высокое потребление энергии и необходимость большого количества аппаратных ресусов (для каждой строчки полный набор аппаратуры для сравнения и выделения нужного слова).

1.2.3 Наборно-ассоциативное отображение

Такое отображение является одним из возможных компромиссов, сочетающим достоинства прямого и ассоциативного способов отображения. Кэш-память (как тэгов, так и данных) разбивается на v подмножеств (модулей), каждое из которых содержит k строк. Зависимость между модулем и блоками ОП такая же, как и при прямом отображении: на строки, входящие в модуль i, могут быть отображены только вполне определенные блоки основной памяти, в соответствии с соотношением $i = j \bmod v$, где j — адрес

блока ОП. В то же время размещение блоков по строкам модуля — произвольное, и для поиска нужной строки в пределах модуля используется ассоциативный принцип.

На рисунке 1.8 представлена система памяти с наборно-ассоциативным отображением.

Рисунок 1.8 - Соответствие адресов блоков и строк, и организация кэш-памяти с наборно-ассоциативным отображением

На рисунке 1.9 показана схема организации кэш-памяти с частично-ассоциативным отображением. Блок ОП можно поместить только в тот набор, номер которого равен адресу этого блока, взятому по модулю. Место блока в наборе может быть произвольным. Сравнение тэгов со старшими разрядами адреса производится только для строк, входящих в набор. По числу строк в наборе различают 2-канальные, 4-канальные и в целом n-канальные, где n – кратное двойке число. В данном примере рассматривается 2-канальная структура, для каждого канала требуется отдельная SRAM. Одновременно выбираются строки во всех каналах. Считывание идет от того канала, где имеется совпадает тег из адреса и канала. Из строки через смещение выбирается адресованное слово. При отсутствии совпадений происходит обращение к ОП и замещение строки в одном из каналов.

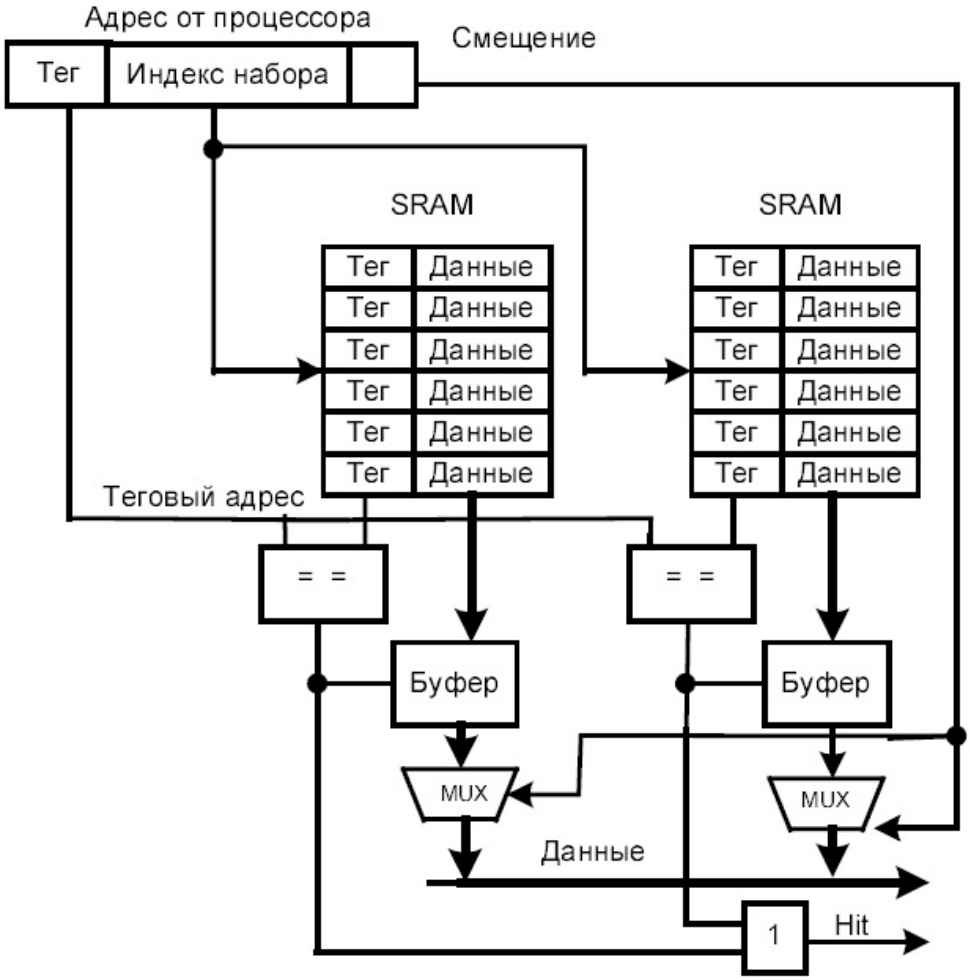

Рисунок 1.9 - Схема организации кэш памяти с наборно-ассоциативным отображением

По сравнению с кэшем с прямым размещением кэш наборно-ассоциативного типа вследствие более свободного размещения блоков в наборе позволяет сформировать в кэше лучший состав копий блоков основной памяти, т.к. имеется возможность выбрать ту или иную заменяемую страницу.

При промахе и обращении к ОП в кэш-память должен быть записан новый блок. При прямом отображении для этого может быть использована только одна определенная строка, которая и будет заменена. Для других вариантов кэша нужен какой-то алгоритм выбора удаляемой строки (алгоритм замещения). Существует множество различных алгоритмов замещения, некоторые из них рассмотрены ниже.

1.3 Алгоритмы замещения

Если в кэш-памяти не используется прямое отображение, при котором каждому блоку основной памяти соответствует одна строка кэша, необходимо иметь алгоритм вытеснения заполненной строки кэша в ОП:

1. LRU;
2. MRU;
3. FIFO;
4. RR.
5. LFU;

1.3.1 LRU

Алгоритм LRU (Least recently used) можно реализовать с помощью очереди, куда в порядке заполнения строки кэш-памяти заносятся ссылки на эти строки. При каждом обращение к строке ссылка перемещается в конец очереди. Тем самым первая строка в очереди претендент на замещение.

Альтернативно возможно реализация этого метода через сохранение «бита возраста» для строк кэша и за счет этого происходит отслеживание наименее использованных строк В подобной реализации, при каждом обращении к строке кэша меняется возраст всех остальных строк.

Данный алгоритм применяется в реализуемом кэше.

1.3.2 MRU

Алгоритм MRU (Most recently used) реализуется аналогичным образом, нолько претендентом на замещение является не первая, а последняя строка.

1.3.3 FIFO

FIFO работает по принципу «первый вошел, первый вышел». Здесь заменяется строка, дольше всего находившаяся в кэш-памяти. Так же реализуется при помощи очереди, только при обращении ссылка не перемещается в конец очереди, а остается на месте.

1.3.4 RR

В случае алгоритма RR (Random Replacement), замещаемая строка выбирается случайным образом. Реализуется, например, с помощью счетчика, содержимое которого увеличивается на единицу с каждым тактовым импульсом, вне зависимости от того, имело место попадания или промах. Значение в счетчике определяет заменяемую строку в полностью ассоциативной кэш-памяти или строку в пределах модщуля для наборно-ассоциативной кэш-памяти.

1.3.5 LFU

LFU (Least Frequently Used) — вытесняется буфер, использованный реже всех. Реализуется добавлением к каждой ячейке кэша счетчика, который увеличивается при каждом обращении. При вытеснении из кэша исключается значение с наименьшим значением счетчика. В случае многоканальной памяти к каждому каналу также добавляется счетчик для выбора канала из которого вытесняется строка.

1.4 Алгоритмы записи

В процессе вычислений центральный процессор может не только считывать имеющуюся информацию, но и записывать новую, обновляя содержимое кэш-памяти. Но многие устройства ввода/вывода умеют напрямую обмениваться информацией с основной памятью. В обоих вариантах возникает ситуация, когда содержимое основной и кэш-памяти перестает совпадать. В итоге на связанное с основной памятью устройство вывода может быть выдана «устаревшая» информация, поскольку все изменения, сделанные процессором, фиксируются только в кэш-памяти, а процессор будет использовать старое содержимое кэш-памяти вместо новых данных, загруженных в основную память из устройства ввода.

Для разрешения ситуации, когда процессор выполняет операцию записи, в системах с кэш-памятью предусмотрены методы обновления кэш-

памяти, которые можно разбить на две большие группы: метод прямой записи и метод обратной записи.

1.4.1 Метод прямой записи

В случае прямой записи обновляется, прежде всего, слово, хранящееся в основной памяти. Если в кэш-памяти существует копия данного слова, то она обновляется. При сквозной записи с отображением, если в кэш-памяти отсутствует нужная копия этого слова, то из основной памяти в кэш-память пересылается блок, содержащий обновленное слово. По методу сквозной записи без отображения этого не делается.

Главное достоинство метода прямой записи в том, что если блок кэш-памяти освобождается для хранения блока, то удаляемый блок можно не возвращать в основную память, потому что его копия там уже имеется. Время для операций записи не уменьшается.

При использовании метода буферизованной прямой записи, являющегося модификацией прямой записи, появляется некоторый эффект и при операциях записи. Здесь информация сначала записывается в кэш-память и в специальный буфер, работающий по схеме FIFO. Запись в основную память производится уже из буфера, а процессор, не дожидаясь ее окончания, может продолжать работу. При использовании буферизации процессор полностью освобождается от работы с основной памятью.

1.4.2 Метод обратной записи

При обратной записи слово заносится только в кэш-память. Если соответствующего блока в кэш-памяти нет, то блок вначале пересылается из основной памяти, после чего запись все равно выполняется исключительно в кэш-память. При замещении блока его необходимо предварительно переслать в соответствующее место основной памяти. При каждом чтении из основной памяти осуществляются две пересылки между основной и кэш-памятью.

Разновидность метода обратной записи – метод флаговой обратной записи. При изменении в каком-то блоке кэш-памяти устанавливается связанный с этим блоком бит изменения (флаг). Когда производится замещение блока, то он переписывается в основную память только тогда, когда его флаг изменения установлен, иначе строка просто перезаписывается.

1.5 Архитектура кэш-памяти

Разделять кэш-память на память данных и команд стали далеко не сразу после начала ее применения в микропроцессорах. Вначале она была смешанной согласно Неймановской архитектуре. Позже стало обычным разделять кэш-память на две части – отдельно для команд, отдельно для данных согласно Гарвардской архитектуре.

В данной работе реализуется кэш уровня 2, в котором нет функционального разделения на кэш данных и инструкций.

При раздельной кэш-памяти выборка команд и данных может производиться одновременно, при этом исключаются возможные конфликты. В некоторых вычислительных машинах помимо кэш-памяти команд и кэш-памяти данных может использоваться и адресная кэш-память.

Также иногда кэш-память реализуют не отдельными микросхемами, а резервируя области оперативной памяти (технология RAMCache) и иногда области SSD-дисков (StoreMI). В теории возможно применение для этой цели и обычных магнитных жестких дисков, однако практически данный подход не дает никаких очевидных преимуществ.

1.6 Иерархическая кэш-память

При увеличении емкости кэш-памяти происходит снижение быстродействия, поэтому общую емкость кэш-памяти увеличивают за счет иерархической организации, при которой кэш-память состоит из нескольких уровней запоминающих устройств, отличающихся емкостью и

быстродействием. Каждый последующий уровень обладает большей емкостью, но меньшим быстродействием.

Первый уровень (L1) иерархии образует наиболее скоростная кэш-память сравнительно небольшой емкости (не более 128 Кбайт). На кристалле ее располагают по возможности ближе к процессору, чтобы минимизировать длину соединяющей их шины и тем самым способствовать ускорению обмена информации.

Второй уровень (L2) обладает емкостью большей, чем у первого уровня, а быстродействие несколько ниже. Размещается на одном кристалле с процессором, за счет чего уменьшается длина связей и увеличивается быстродействие.

Для новейших многоядерных процессоров типично размещение на том же кристалле еще и кэш-памяти третьего уровня (L3).

Помимо этого иногда кеширование производится и в области ОП (AMD RAMcache, LLC) и даже в SSD (AMD StoreMI), однако данные уровни применяются для работы с магнитными жесткими дисками, а не ОП, поэтому они в разборе опущены.

При построении иерархий кэш-памяти используется один из принципов хранения – инклюзивный или эксклюзивный.

Инклюзивная архитектура предполагает копирование информации, находящейся в L1 и L2. Во время копирования информации из ОП в кэш делается, две копии, одна копия заносится в L2, а другая копия – в L1. При считывании процессором информации из кэша, она берется из L1. Если нужной информации в кэше первого уровня нет, то она ищется в L2. Если нужная информация в кэше второго уровня найдена, то она дублируется в L1 (по принципу LRU), и передается в процессор. Если нужная информация не найдена и в кэше второго уровня, то она считывается из ОП.

Эксклюзивная кэш-память предполагает уникальность информации, находящейся в L1 и L2. В этом случае при считывании информации из ОП – информация сразу заносится в L1. Когда L1 заполнен, то, информация

переносится из L1 в L2. Если при считывании процессором информации из L1 нужная информация не найдена, то она ищется в L2. Если нужная информация найдена в L2, то кэши первого и второго уровня обмениваются между собой строками. Если нужная информация не найдена и в L2, то обращение идет к ОП.

2 Разработка системы кэш-памяти

2.1 Общий алгоритм кэш-памяти

На рисунке 2.1 представлен общий алгоритм работы кэш-памяти.

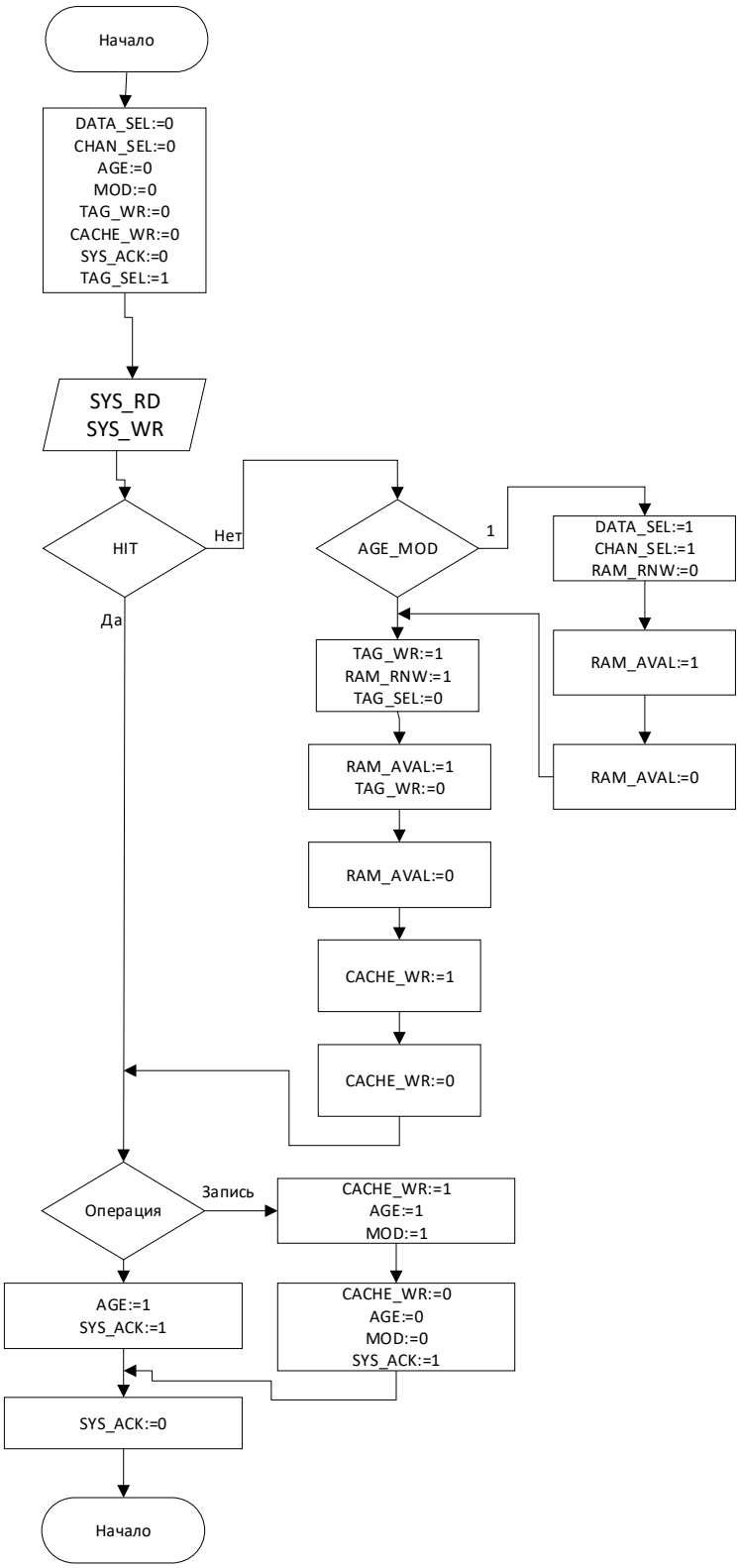

Рисунок 2.1 – Алгоритм работы кэш-памяти

2.2 Функциональная схема кэш-памяти

Согласно заданию системный адрес имеет 16 бит. Размер строки – 16 байт, следовательно размерность смещения – 4 бита.

Объем кэш-памяти составляет 2048 байт, значит всего в кэше предусмотрено 128 строк и как следствие 16 групп по 8 каналов. Следовательно размерность индекса – 4 бита.

Остальная часть адреса отводится под тэг – 8 бит.

На рисунке 2.2 представлена функциональная схема кэш-памяти.

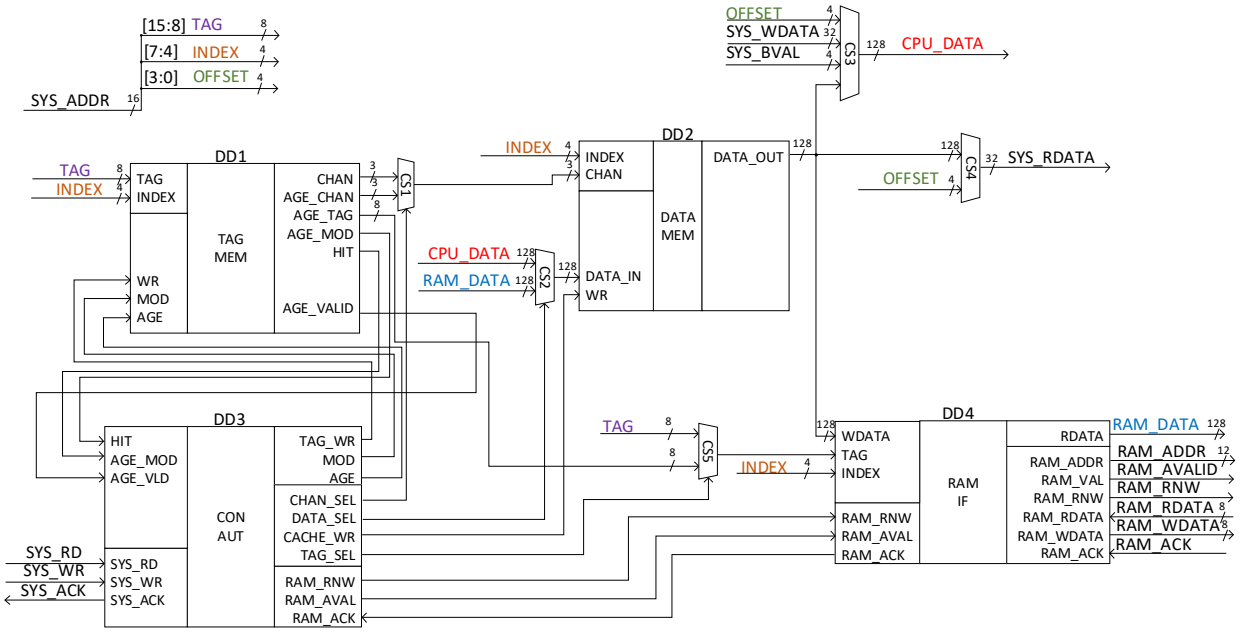

Рисунок 2.2 – Функциональная схема кэш-памяти

CS1 – схема выбора канала для памяти данных.

CS2 – схема выбора строки, записываемой в память данных.

CS3 – схема формирования строки с записанным словом при операции записи в кэш.

CS4 – выбор слова из строки при операции чтения.

CS5 – выбор тэга для формирования адреса при обращение в ОП.

2.3 Функциональная схема памяти тэгов

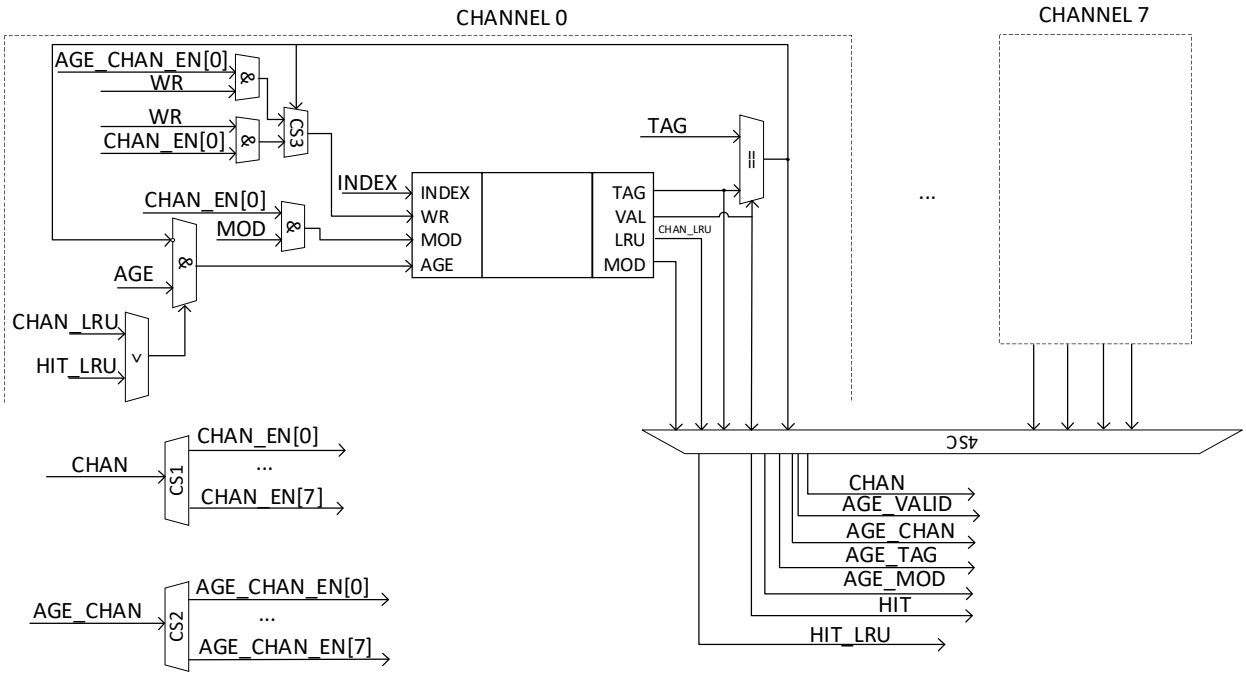

Рисунок 2.3 – Функциональная схема памяти тэгов

CS1 – схема формирования сигналов включения при выборе канала с попаданием.

CS2 – схема формирования сигналов включения при выборе канала с самой старой строкой.

CS3 – схема выбора между включением по возрасту или попаданию.

CS4 – схема выбора канала по LRU и попаданию.

2.4 Местное устройство управления

Местное устройство управления представляет собой автомат Мура с 9 состояниями. Граф состояний представлен на рисунке 2.4.

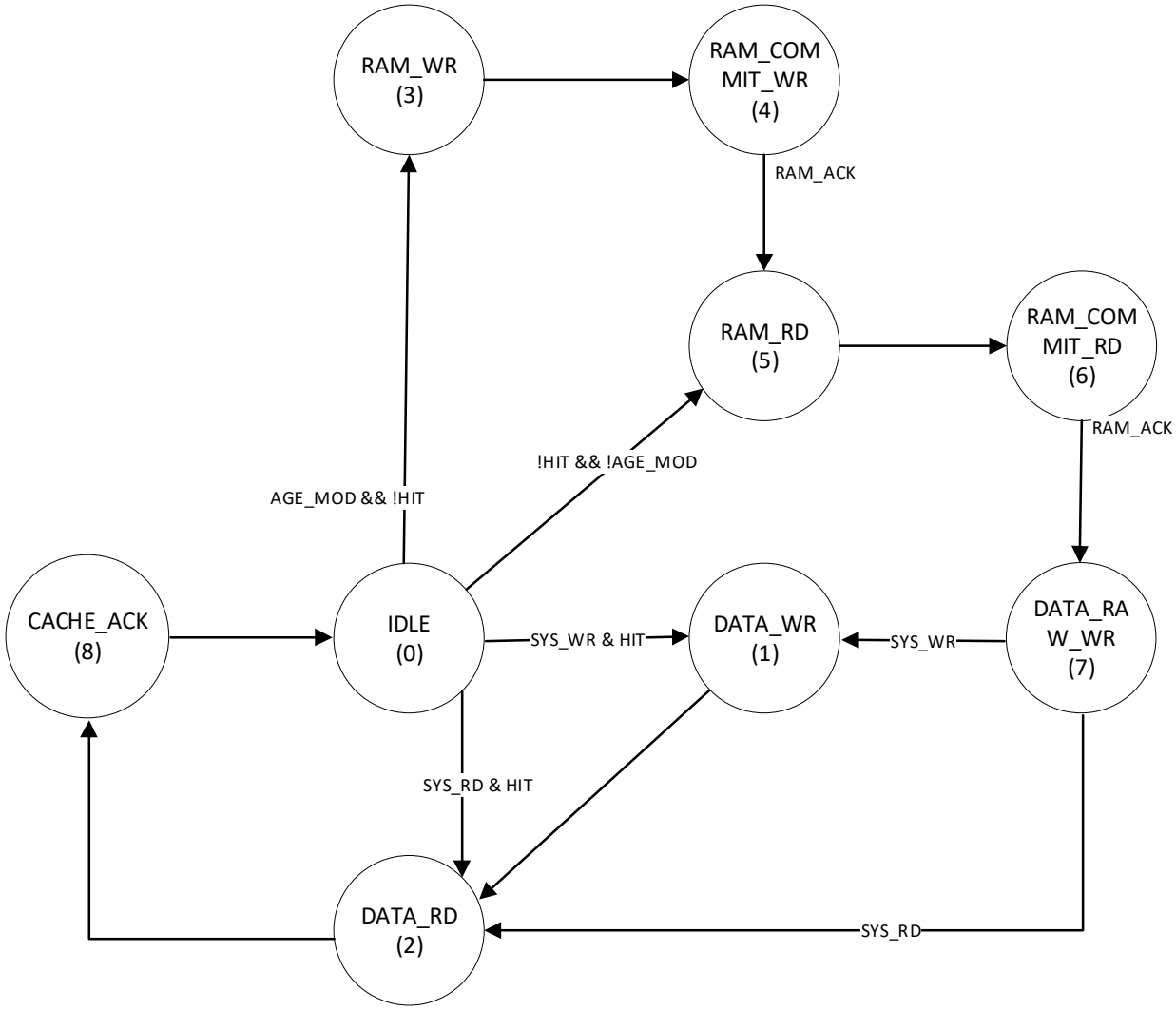

Рисунок 2.4 – Граф состояний местного устройства управления

2.5 Анализ производительности кэш-памяти в зависимости от объема отображаемых данных

В целях анализа производительности было проведено четыре теста высоконагруженного чтения. Тесты отличаются по объему памяти, отображаемому в кэш:

1) ОП равна по объему кэш-памяти

2) ОП равна двукратному объему кэш-памяти

3) ОП равна четырехкратному объему кэш-памяти

4) ОП равна 65536 байт (максимально возможный объем)

Временные параметры системы памяти представлены в таблице 2.1.

Таблица 2.1 – Временные параметры системы памяти

Параметр	Значение (нс)
Период cpu_clk	8
Период cache_clk	21
Период ram_clk	58
Задержка чтения ОП	580

В рамках каждого теста было выполнено 65536 операций. Результаты тестов показаны на таблице 2.2 .

Таблица 2.2 - Результаты тестов производительности кэш-памяти

	ОП=Кэш	ОП=2хКэш	ОП=4хКэш	ОП=65536 байт
Среднее (нс)	246,3166199	1102,498627	1537,907562	1917,359711
Медиана (нс)	240	240	1960	1976
Процент промахов (%)	0,003646851	0,498886108	0,750320435	0,968597412

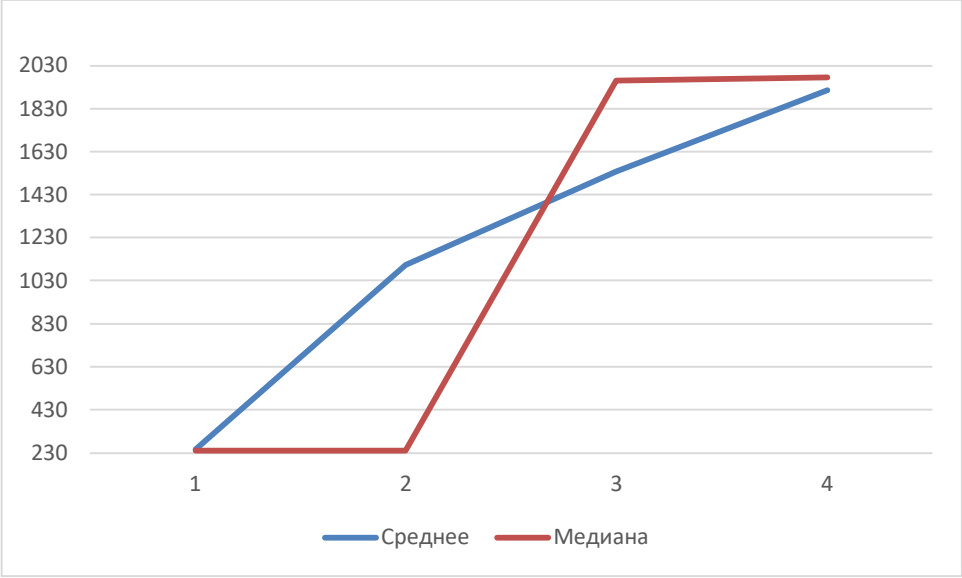

Рисунок 2.5 - Среднее и медианное время ответа системы памяти в зависимости от объема отображаемой памяти

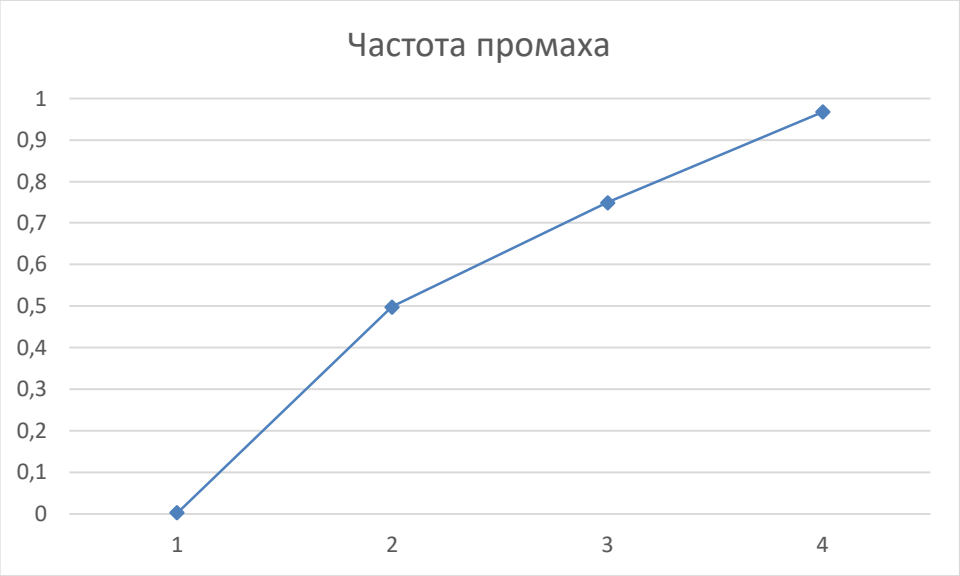

Рисунок 2.6 - Частота промаха системы памяти в зависимости от объема отображаемой памяти.

2.6 Анализ производительности кэш-памяти при автоматическом тестировании

Временные параметры системы памяти представлены в таблице 2.1.

Всего в рамках теста было выполнено 327680 операций. Результаты тестов приведены на таблицах 2.3 и 2.4.

Таблица 2.3 – Среднее время выполнения операции в нс

	Промах	Попадание
Чтение	2790.375494	237.926097
Запись	2813.836185	262.003150

Таблица 2.4 – Медианное время выполнения операции в нс

	Промах	Попадание
Чтение	3510	230
Запись	3534	270

2.7 Результаты реализации дизайна

```
Device Utilization Summary:

Slice Logic Utilization:
  Number of Slice Registers:                          710 out of
54,576    1%
    Number used as Flip Flops:                692
    Number used as Latches:                    18
    Number used as Latch-thrus:                 0
    Number used as AND/OR logics:               0
  Number of Slice LUTs:                         1,605 out of
27,288    5%
    Number used as logic:                       935 out of
27,288    3%
      Number using O6 output only:            832
      Number using O5 output only:              1
      Number using O5 and O6:                 102
      Number used as ROM:                       0
    Number used as Memory:                      620 out of
6,408    9%
      Number used as Dual Port RAM:           548
        Number using O6 output only:          480
        Number using O5 output only:            5
        Number using O5 and O6:                63
      Number used as Single Port RAM:          72
        Number using O6 output only:           40
        Number using O5 output only:            0
        Number using O5 and O6:                32
      Number used as Shift Register:            0
    Number used exclusively as route-thrus:    50
      Number with same-slice register load:    50
      Number with same-slice carry load:        0
      Number with other load:                   0

Slice Logic Distribution:
  Number of occupied Slices:                    606 out of
6,822    8%
  Number of MUXCYs used:                         32 out of
13,644    1%
  Number of LUT Flip Flop pairs used:       1,761
    Number with an unused Flip Flop:              1,150 out of
1,761   65%
    Number with an unused LUT:                    156 out of
1,761    8%
    Number of fully used LUT-FF pairs:            455 out of
1,761   25%
    Number of slice register sites lost
      to control set restrictions:                  0 out of
54,576    0%

  A LUT Flip Flop pair for this architecture represents one LUT
paired with
```

```
  one Flip Flop within a slice.  A control set is a unique
combination of
  clock, reset, set, and enable signals for a registered
element.
  The Slice Logic Distribution report is not meaningful if the
design is
  over-mapped for a non-slice resource or if Placement fails.

IO Utilization:
  Number of bonded IOBs:                                122 out of
190   64%

Specific Feature Utilization:
  Number of RAMB16BWERs:                                  2 out of
116    1%
  Number of RAMB8BWERs:                                   0 out of
232    0%
  Number of BUFIO2/BUFIO2_2CLKs:                          0 out of
32     0%
  Number of BUFIO2FB/BUFIO2FB_2CLKs:                      0 out of
32     0%
  Number of BUFG/BUFGMUXs:                                3 out of
16    18%
    Number used as BUFGs:                         3
    Number used as BUFGMUX:                       0
  Number of DCM/DCM_CLKGENs:                              0 out of
8      0%
  Number of ILOGIC2/ISERDES2s:                            0 out of
376    0%
  Number of IODELAY2/IODRP2/IODRP2_MCBs:                  0 out of
376    0%
  Number of OLOGIC2/OSERDES2s:                            0 out of
376    0%
  Number of BSCANs:                                       0 out of
4      0%
  Number of BUFHs:                                        0 out of
256    0%
  Number of BUFPLLs:                                      0 out of
8      0%
  Number of BUFPLL_MCBs:                                  0 out of
4      0%
  Number of DSP48A1s:                                     0 out of
58     0%
  Number of GTPA1_DUALs:                                  0 out of
2      0%
  Number of ICAPs:                                        0 out of
1      0%
  Number of MCBs:                                         0 out of
2      0%
  Number of PCIE_A1s:                                     0 out of
1      0%
```

```
  Number of PCILOGICSEs:                                  0 out of
2    0%
  Number of PLL_ADVs:                                     0 out of
4    0%
  Number of PMVs:                                         0 out of
1    0%
  Number of STARTUPs:                                     0 out of
1    0%
  Number of SUSPEND_SYNCs:                                0 out of
1    0%


Overall effort level (-ol):   High
Router effort level (-rl):    High
```