|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 3 |

**Название:**

Организация памяти конвейерных суперскалярных электронных вычислительных машин

**Дисциплина:** Архитектура ЭВМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | И.С. Климов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.Ю. Попов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2021

**Содержание**

[**Введение** 3](#_Toc92119508)

[**1.** **Задание 1** 4](#_Toc92119509)

[**2.** **Задание 2** 5](#_Toc92119510)

[**3.** **Задание 3** 6](#_Toc92119511)

[**4.** **Задание 4** 10](#_Toc92119512)

[**5.** **Задание 5** 12](#_Toc92119513)

[**6.** **Задание 6** 14](#_Toc92119514)

[**7.** **Задание 7** 16](#_Toc92119515)

[**8.** **Задание 8** 18](#_Toc92119516)

[**Контрольные вопросы** 20](#_Toc92119517)

[**1.** **Вопрос 1** 20](#_Toc92119518)

[**2.** **Вопрос 2** 20](#_Toc92119519)

[**3.** **Вопрос 3** 20](#_Toc92119520)

[**4.** **Вопрос 4** 21](#_Toc92119521)

[**5.** **Вопрос 5** 21](#_Toc92119522)

# **Введение**

**Цель работы -** освоение принципов эффективного использования подсистемы памяти современных универсальных ЭВМ, обеспечивающей хранение и своевременную выдачу команд и данных в центральное процессорное устройство.

Для достижения цели были поставлены и решены следующие **задачи**:

* ознакомиться с теоретическим материалом, касающимся особенностей функционирования подсистемы памяти современных конвейерных суперскалярных ЭВМ;
* изучить возможности программы PCLAB;
* изучить средства идентификации микропроцессоров;
* провести исследования времени выполнения тестовых программ;
* сделать выводы о архитектурных особенностях используемых ЭВМ.

# **Задание 1**

**Задача**: ознакомиться с возможностями программы PCLAB в Разделе 2 методических указаний. Запустить программу PCLAB 1.0. Изучить идентификационную информацию на вкладке «Идентификация процессора».

Программа PCLAB предназначена для исследования производительности x86 совместимых ЭВМ c IA32 архитектурой, работающих под управлением операционной системы Windows (версий 95 и старше). Исследование организации ЭВМ заключается в проведении ряда экспериментов, направленных на построение зависимостей времени обработки критических участков кода от изменяемых параметров. Набор реализуемых программой экспериментов позволяет исследовать особенности построения современных подсистем памяти ЭВМ и процессорных устройств, выявить конструктивные параметры конкретных моделей ЭВМ. Процесс сбора и анализа экспериментальных данных в PCLAB основан на процедуре профилировки критического кода, т.е. в измерении времени его обработки центральным процессорным устройством.

На листинге 1.1 представлена идентификационная информация.

Листинг 1.1 – Информация о процессоре, полученная с помощью PCLAB

Vendor ID: "Genuine Intel"; CPUID level 10

Дополнительные функции Intel:

Верисия 00010677:

Type 0 − Original OEM

Family 6 − Pentium Pro

Model 7 − Pentium III /Pentium III Xeon − external L2 cache

Stepping 7

Reserved 4

Extended brand string: " Intel (R) Core (TM) 2 Quad CPU Q8200 @ 2. 33GHz"

CLFLUSH instruction cache line size: 8

Initial APIC ID: 3

Hyper threading siblings: 4

. . .

TLB and cache info:

b1: unknown TLB/ cache descriptor

b0: дескриптор TLB−команд, 4K страницы, асс. 4−направ., 128 элементов

05: unknown TLB/ cache descriptor

f0: unknown TLB/ cache descriptor

57: unknown TLB/ cache descriptor

56: unknown TLB/ cache descriptor

7d: unknown TLB/ cache descriptor

30: L1 кэш−команд, 32 KB, асс. 8−направ., длина с троки 64 байта

b4: unknown TLB/ cache descriptor

2c: L1 кэш−данных, 32 KB, асс. 8−направ., длина с троки

Processor serial: 0001−0677−BFEB−FBFF−0008−E39D

# **Задание 2**

**Задача**: на основании идентификационной информации о микропроцессоре ЭВМ, используемой при проведении лабораторной работы, определить следующие параметры: размер линейки кэш-памяти верхнего уровня и объем физической памяти. Результаты занести в отчет.

**Размер** линейки кэш-памяти верхнего уровня: 64 байт. **Объем** физической памяти: 4 Гб.

# **Задание 3**

**Задача**: ознакомиться с описанием эксперимента «Исследование расслоения динамической памяти» на вкладке «Описание эксперимента». Провести эксперимент. По результатам эксперимента определить: количество банков динамической памяти; размер одной страницы динамической памяти; количество страниц в динамической памяти. Сделать выводы о использованном способе наращивания динамической памяти. Результаты занести в отчет.

**Цель эксперимента**: определение способа трансляции физического адреса, используемого при обращении к динамической памяти.

**Исходные данные**:

* Размер линейки кэша: 64 байт;
* Объем физической памяти: 4 Гбайт.

В таблице 3.1 приведены настраиваемые параметры.

Таблица 3.1 – Настраиваемые параметры

|  |  |  |  |
| --- | --- | --- | --- |
| Эксперимент | № | Значение | Описание |
| 1 | 1 | 32 Кбайт | Максимальное расстояние между читаемыми блоками |
| 2 | 128 байт | Шаг увеличения расстояния между читаемыми 4-х байтовыми ячейками |
| 3 | 1 Мбайт | Размер массива |
| 2 | 1 | 32 Кбайт | Максимальное расстояние между читаемыми блоками |
| 2 | 64 байт | Шаг увеличения расстояния между читаемыми 4-х байтовыми ячейками |
| 3 | 1 Мбайт | Размер массива |

На рисунках 3.1 – 3.3 приведены зависимости времени обращения к памяти от расстояния между читаемыми блоками данных.

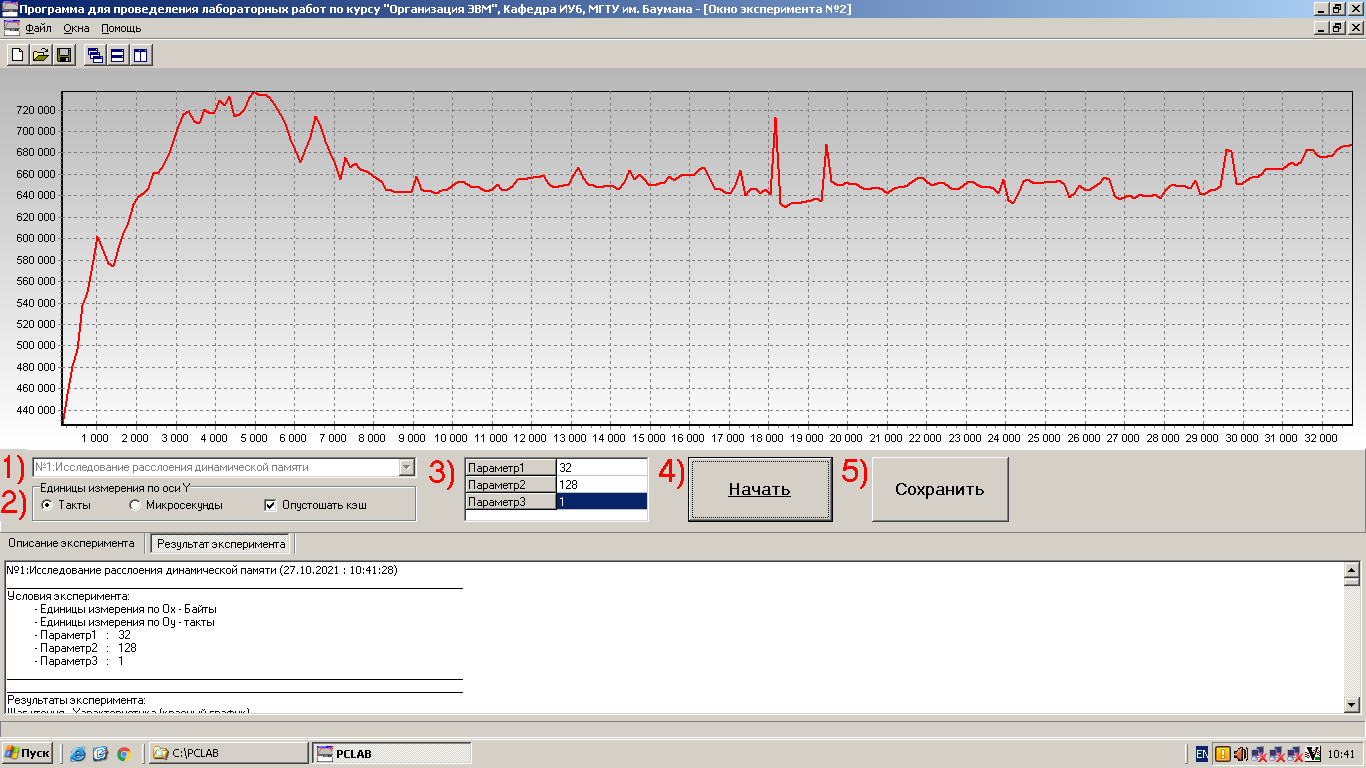


Рисунок 3.1 – Результаты исследования расслоения динамической памяти

(часть 1)

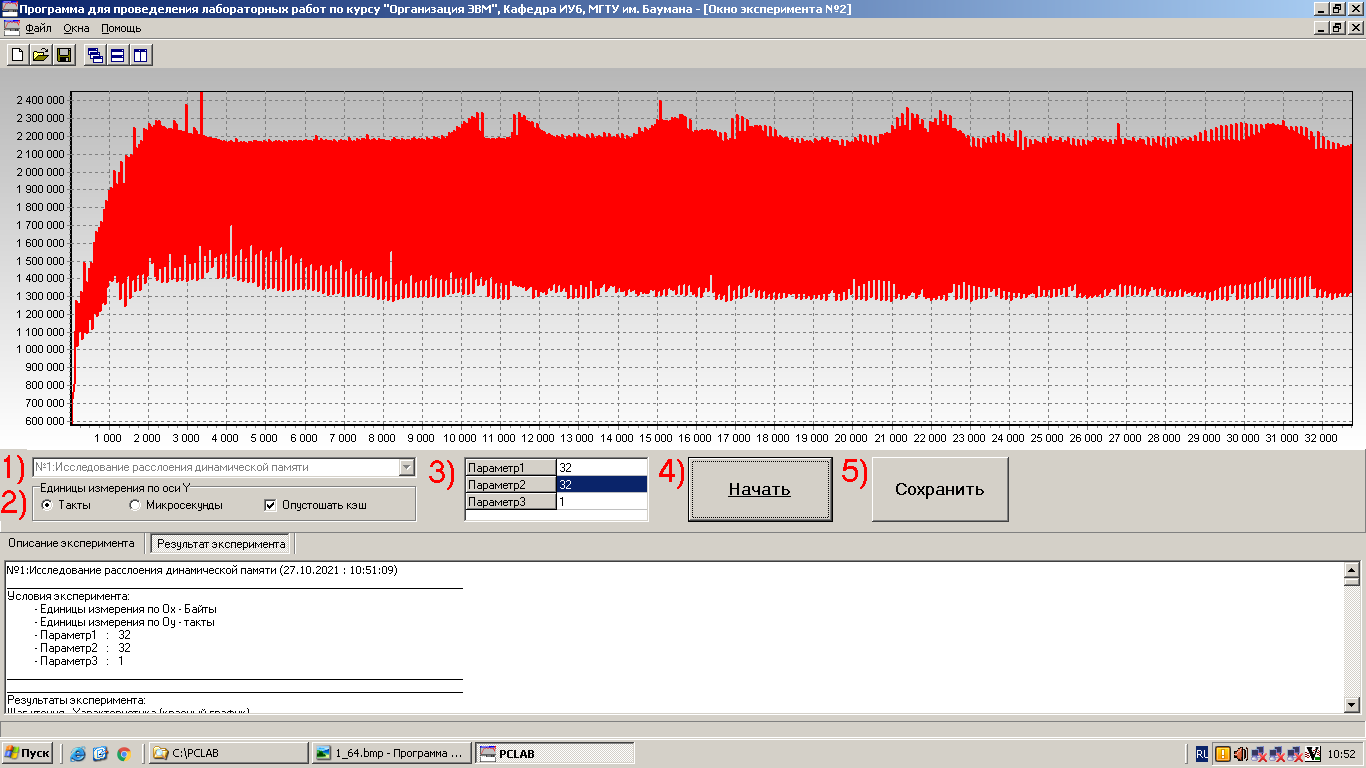


Рисунок 3.1 – Результаты исследования расслоения динамической памяти

(часть 2)

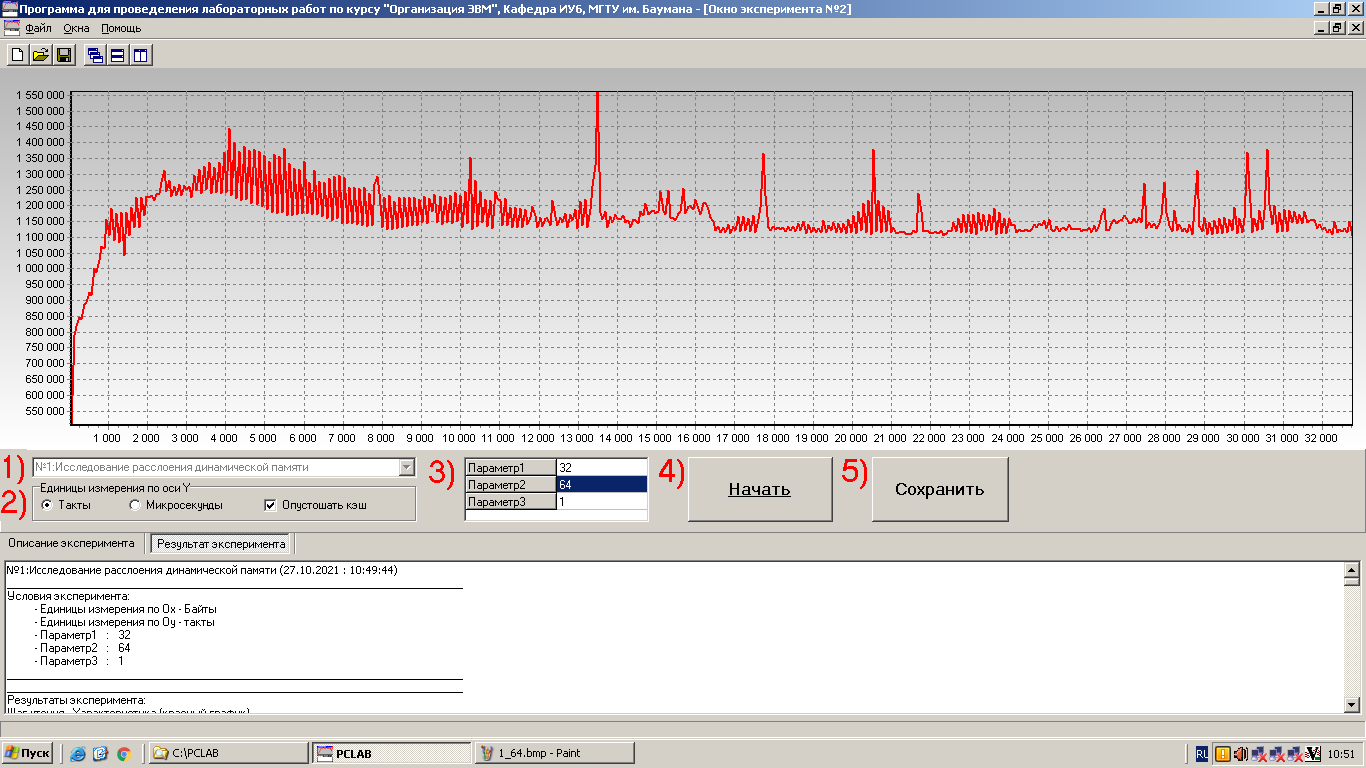


Рисунок 3.3 – Результаты исследования расслоения динамической памяти

(часть 3)

Для более наглядного представления на рисунке 3.4 приведена блочно-циклическая схема расслоения памяти.



Рисунок 3.4 – Блочно-циклическая схема расслоения памяти

**Результаты:**

* количество банков динамической памяти: *B = T1 / P = 1024 / 128 = 8*;
* размер одной страницы динамической памяти: *PS = T2 / B = 4096 / 8 = 512 байт*;
* количество страниц в динамической памяти: *C = V / (PS \* B \* P) = 232 / 219 = 8192 байт*,

где

* B —количество банок памяти;
* P —объем данных, являющийся минимальной порцией обмена кэш-памяти верхнего уровня с оперативной памятью;
* PS —размер страницы DRAM памяти;
* T1 —размер блока в одной банке памяти;
* T2 —размер страницы одной банки памяти;
* V —объем физического пространства ОП;
* C —количество страниц в ОП.

Таким образом, можно сделать **выводы**:

1. Память расслоена, доступ к ней осуществляется с разным временем доступа (в зависимости от размера запрашиваемого блока);
2. Экспериментально были получены значения количества банков ОП, размер страница банка, размер страницы DRAM;
3. Данные следует размещать так, чтобы они укладывали в одну страницу;
4. Данные нужно выравнивать по размеру линейки кэша.
5. Данные необходимо обрабатывать так, чтобы минимизировать количество последовательных обращений в одну банку ОП.

# **Задание 4**

**Задача**: ознакомиться с описанием эксперимента «Сравнение эффективности ссылочных и векторных структур данных». Провести эксперимент. По результатам эксперимента определить: отношение времени работы алгоритма, использующего зависимые данные, ко времени обработки аналогичного алгоритма обработки независимых данных. Сделать выводы об эффективности ссылочных и векторных структур данных и способах ее повышения. Результаты занести в отчет.

**Цель эксперимента**: оценить влияние зависимости команд по данным на эффективность вычислений.

В таблице 4.1 приведены настраиваемые параметры.

Таблица 4.1 – Настраиваемые параметры

|  |  |  |
| --- | --- | --- |
| № | Значение | Описание |
| 1 | 1 Мбайт | Количество элементов в списке |
| 2 | 32 Кбайт | Максимальная фрагментация списка |
| 3 | 1 Кбайт | Шаг увеличения фрагментации |

На рисунке 4.1 представлена зависимость времени выполнения поиска минимального значения для массива и односвязного списка.

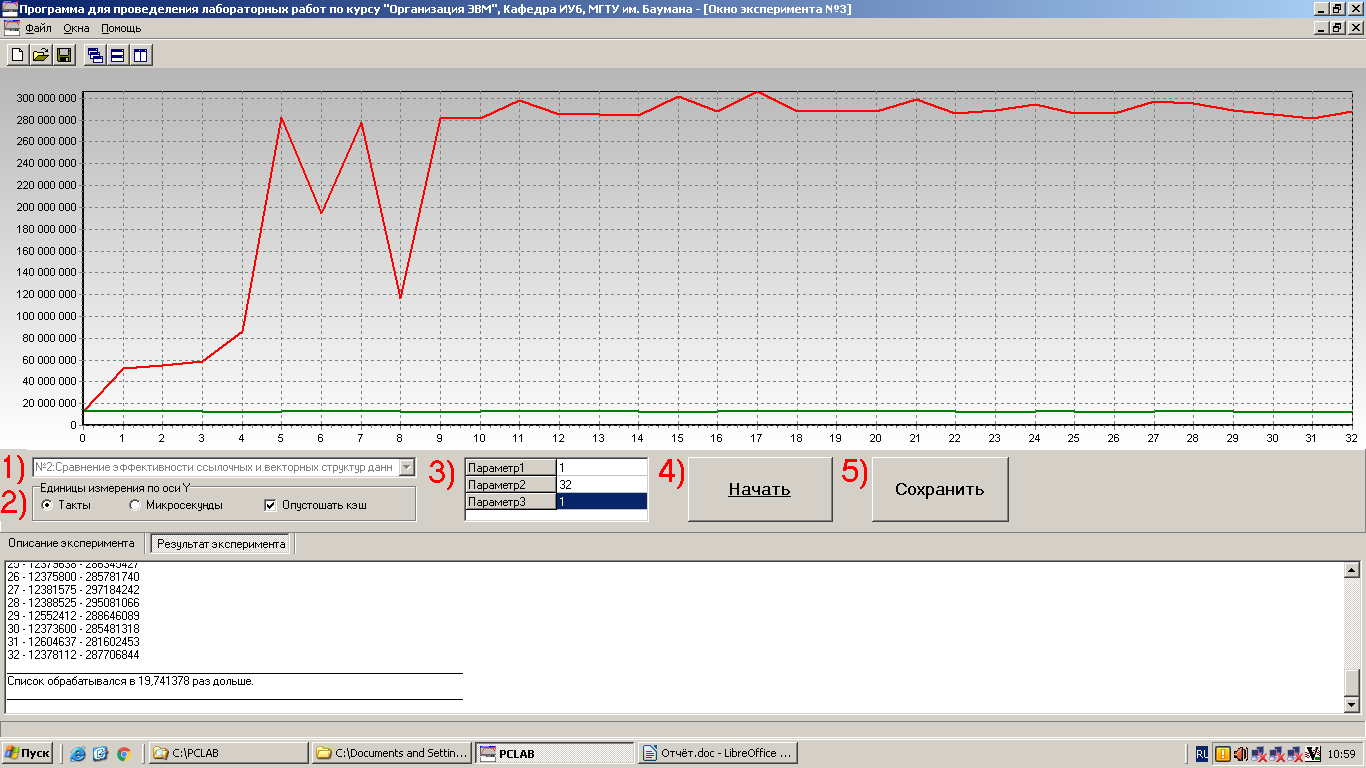


Рисунок 4.1 – Сравнение эффективности ссылочных и векторных структур данных

Из полученного графика видна проблема сематического разрыва. Поэтому следует использовать структуры данных с учётом технологического фактора (для машины выгоднее использовать массив). **Результат**: односвязный список обрабатывался в **19,741378** раз дольше.

**Выводы**:

1. Связанные данные следует организовывать так, чтобы при работе программы они были как можно ближе друг к другу расположены в ОП;
2. Использование подходящих структур данных приводит к значительному повышению производительности системы.

# **Задание 5**

**Задача**: для ЭВМ, используемой при проведении лабораторной работы определить следующие параметры: степень ассоциативности и размер TLB данных. Ознакомиться с описанием и провести эксперимент «Исследование эффективности программной предвыборки». По результатам эксперимента определить: отношение времени последовательной обработки блока данных ко времени обработки блока с применением предвыборки; время и количество тактов первого обращения к странице данных. Сделать выводы об эффективности предвыборки и способах ее повышения. Результаты занести в отчет.

**Цель эксперимента**: выявить способы ускорения вычислений благодаря применению предвыборки данных.

**Исходные данные**:

* степень ассоциативности TLB данных – 4ячейки;
* размер TLB данных – 128групп.

В таблице 5.1 приведены настраиваемые параметры.

Таблица 5.1 – настраиваемые параметры

|  |  |  |
| --- | --- | --- |
| № | Значение | Описание |
| 1 | 512 байт | Шаг увеличения расстояния между читаемыми данными |
| 2 | 128 Кбайт | Размер массива |

На рисунке 5.1 представлены зависимости времени обращения к памяти от расстояния между читаемыми блоками данных.

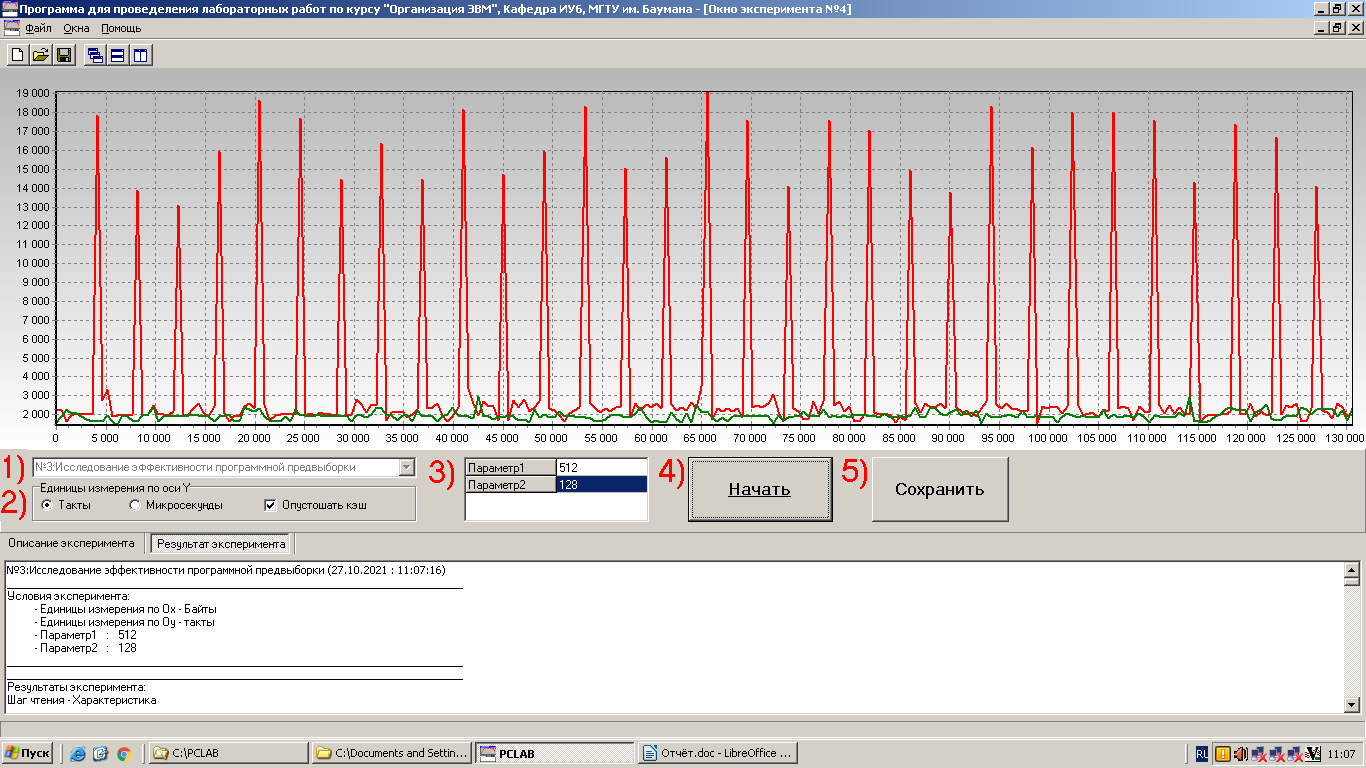


Рисунок 5.1 – Исследование эффективности программной предвыборки

Красный график показывает время или количество тактов работы алгоритма без предвыборки. Зеленый график – с использованием предвыборки.

Время обращения к первому элементу в таблицы в 20 раз больше так как неизвестно, где эта страница находится в памяти. Имеется только логический адрес, а требуется физический. Поэтому следует использовать предвыборку. В данном примере ускорение почти в 2 раза.

**Результаты**:

* обработка без загрузки таблицы страниц в TLB производилась в **1,8456621** раз дольше;
* время первого обращения к странице данных – **15000** тактов.

**Выводы**:

1. Для исключения задержек, связанных с получением физического адреса начала страницы, имеет смысл предварительно загрузить страницы в TLB перед работой с большими массивами данных.

# **Задание 6**

**Задача**: ознакомиться с описанием и провести эксперимент «Исследование способов эффективного чтения оперативной памяти». По результатам эксперимента определить: отношение времени обработки блока памяти неоптимизированной структуры ко времени обработки блока структуры, обеспечивающей эффективную загрузку и параллельную обработку данных. Сделать выводы о способах повышения эффективности чтения оперативной памяти.

**Цель эксперимента**: исследование возможности ускорения вычислений благодаря использованию структур данных, оптимизирующих механизм чтения оперативной памяти.

**Исходные данные**:

* адресное расстояние между банками памяти: 128 байт;
* размер буфера чтения: 4 Кбайт.

В таблице 6.1 представлены настраиваемые параметры.

Таблица 6.1 – Настраиваемые параметры

|  |  |  |
| --- | --- | --- |
| № | Значение | Описание |
| 1 | 1 Мбайт | Размер массив |
| 2 | 128 ед. | Количество потоков данных |

На рисунке 6.1 представлены зависимости времени чтения данных от количества одновременно обрабатываемых массивов для неоптимизированной структуры (красный график) и структуры, обеспечивающей эффективную загрузку и параллельную обработку данных (зеленый график).

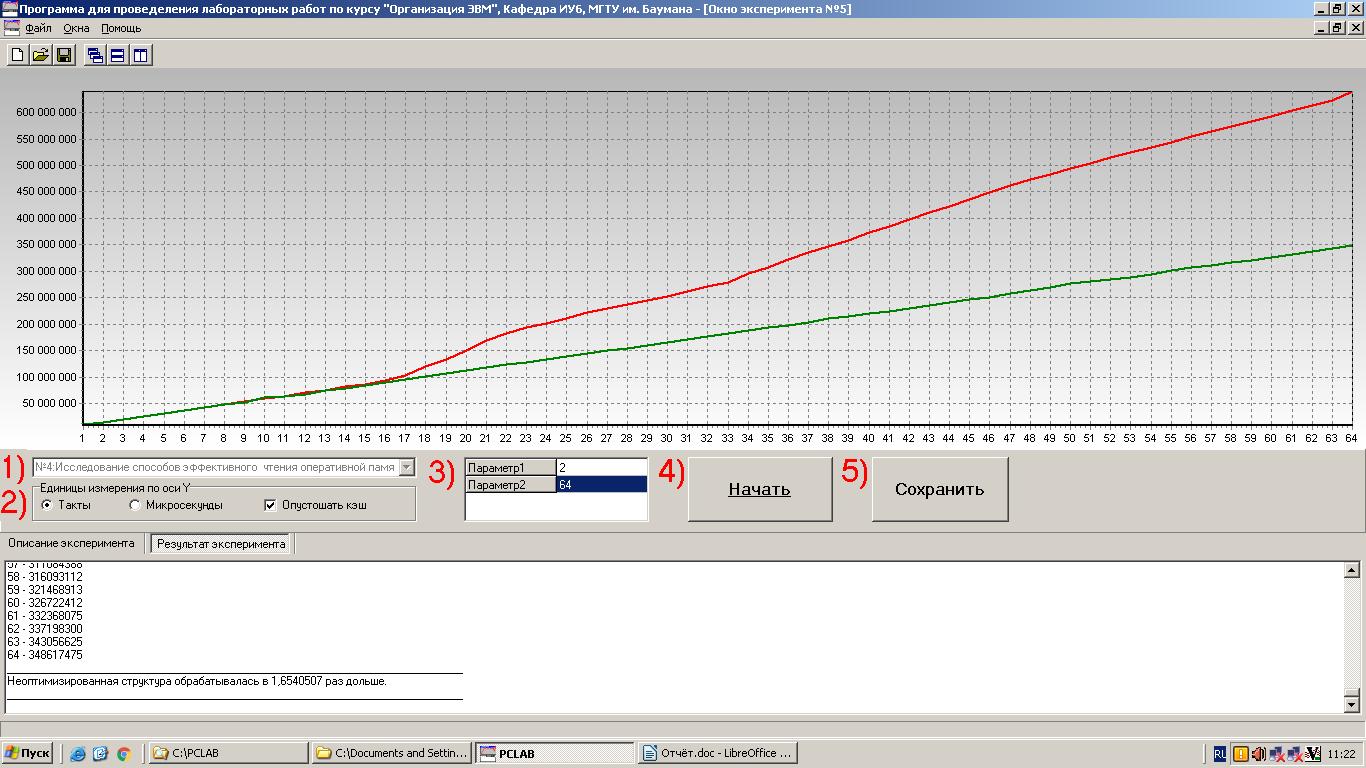


Рисунок 6.1 – Исследование способов эффективного чтения оперативной памяти

**Результаты**: отношение времени обработки блока памяти неоптимизированной структуры ко времени обработки блока структуры, обеспечивающей эффективную загрузку и параллельную обработку данных – **1.65**.

**Выводы**:

1. упорядочив данные определённым образом, можно ускорить приложение (сгруппировав данные, которые используется вместе);
2. следует переупорядочивать данные, выравнивая их по размеру кэш-линии, тем самым исключая несвоевременную передачу данных;
3. нужно размещать данные как можно ближе друг к другу (стараться не обращаться к диспетчеру кучи за памятью, использовать особенности выравнивания данных).

# **Задание 7**

**Задача**: для ЭВМ, используемой при проведении лабораторной работы

определить следующие параметры: размер банка кэш-памяти данных первого и второго уровня, степень ассоциативности кэш-памяти первого и второго уровня, размер линейки кэш-памяти первого и второго уровня. Ознакомиться с описанием и провести эксперимент «Исследование конфликтов в кэш-памяти». По результатам эксперимента определить: отношение времени обработки массива с конфликтами в кэш-памяти ко времени обработки массива без конфликтов. Сделать выводы о способах устранения конфликтов в кэш-памяти.

**Цель эксперимента**: исследование влияния конфликтов кэш-памяти на эффективность вычислений.

**Исходные данные**:

* размер банка кэш-памяти данных первого и второго уровня: 32 Кбайт;
* степень ассоциативности кэш-памяти первого и второго уровня: 8 ячеек;
* размер линейки кэш памяти первого и второго уровня: 64 байт.

В таблице 7.1 приведены настраиваемые параметры.

|  |  |  |
| --- | --- | --- |
| № | Значение | Описание |
| 1 | 128 Кбайт | Размер банка кэш-памяти |
| 2 | 128 байт | Размер линейки кэш-памяти |
| 3 | 32 ед. | Количество читаемых линеек |

На рисунке 7.1 представлены зависимости времени обращения к памяти от расстояния между читаемыми блоками данных.

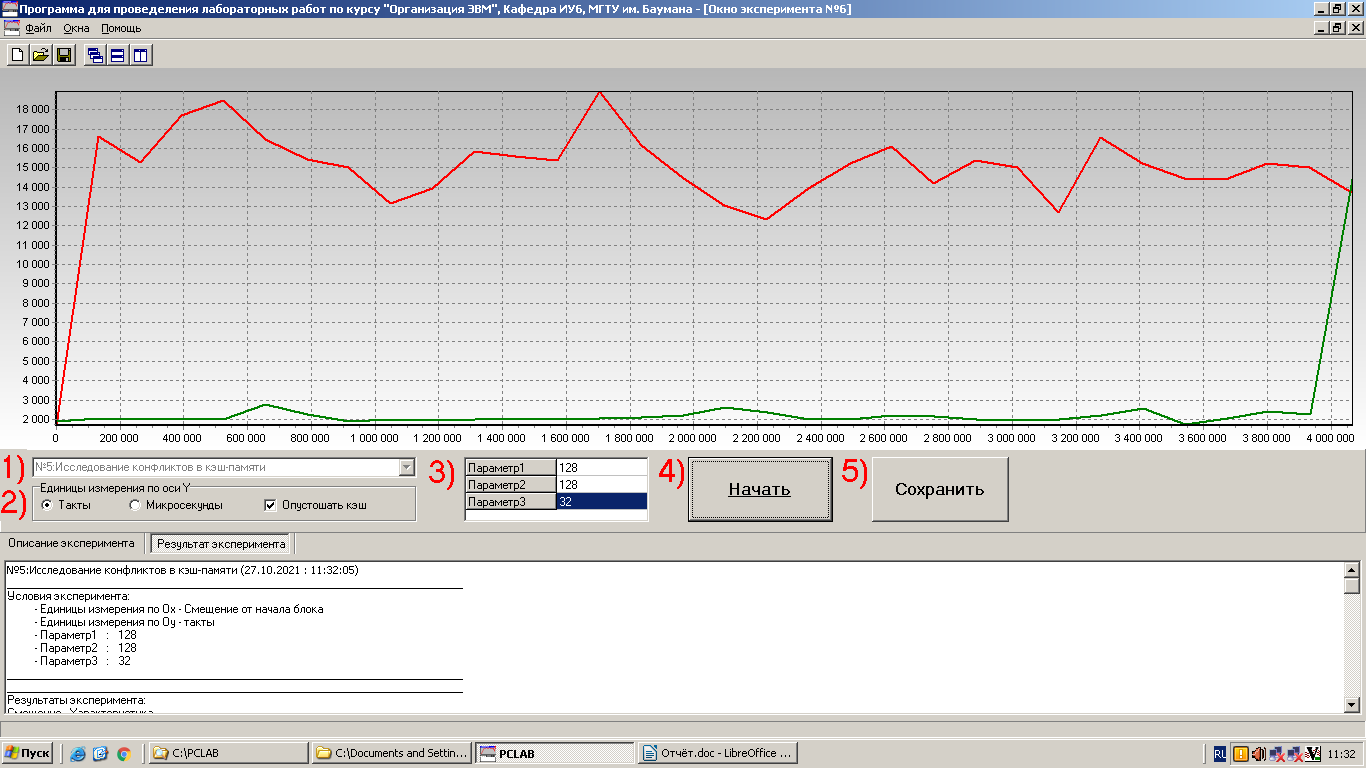


Рисунок 7.1 – Исследование конфликтов в кэш-памяти

**Результаты**: отношение времени обработки массива с конфликтами в кэш-памяти ко времени обработки без конфликтов – **5,9547263**.

На рисунке 7.2 приведена кэш-память с четырехканальным частично-ассоциативным отображением, помогающая понять суть эксперимента.

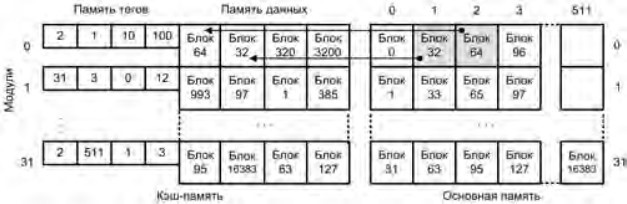


Рисунок 7.2 – Кэш-память с четырехканальным частично-ассоциативным отображением

**Выводы**:

1. Кэш память ускоряет процессор примерно в 6 раз;
2. Данные следует выравнивать по адресам, кратным размеру линейки кэша;
3. Данные нужно обрабатывать так, чтобы уменьшить количество последовательных обращений к блокам памяти, соответствующих одному набору (модулю).

# **Задание 8**

**Задача**: ознакомиться с описанием и провести эксперимент «Исследование алгоритмов сортировки». По результатам эксперимента определить: отношение времени сортировки массивов алгоритмом QuickSort ко времени сортировки алгоритмом Counting-Radix, а также ко времени сортировки Counting-Radix алгоритмом, оптимизированным под 8-процессорную вычислительную систему. Сделать выводы о наиболее эффективном алгоритме сортировки.

**Цель эксперимента**: исследование способов эффективного использования памяти и выявление наиболее эффективных алгоритмов сортировки, применимых в вычислительных системах.

В таблице 8.1 приведены настраиваемые параметры.

Таблица 8.1 – Настраиваемые параметры

|  |  |  |
| --- | --- | --- |
| № | Значение | Описание |
| 1 | 1 Мбайт | Количество 64-разрядных элементов массива |
| 2 | 8 Кбайт | Шаг увеличения размера массива |

На рисунке 8.1 приведены зависимости времени сортировок (Quick Sort, Radix-Counting Sort, оптимизированный Radix-Counting Sort) от размера исходного массива.

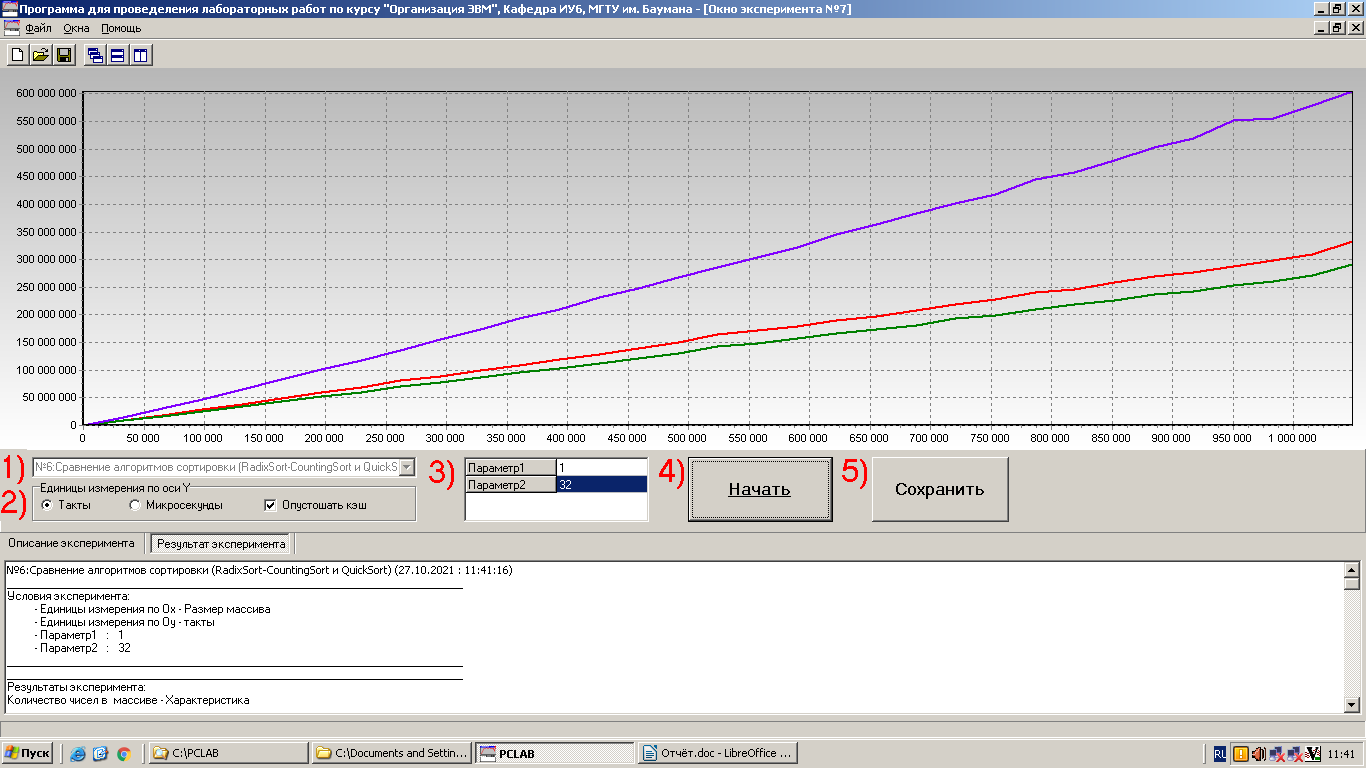


Рисунок 8.1 – Исследование алгоритмов сортировки

Фиолетовый график показывает время или количество тактов работы алгоритма QuickSort, красный – неоптимизированного алгоритма Radix-Counting, зеленый – оптимизированного под 8-процессорную вычислительную систему алгоритма Radix-Counting.

**Результаты**:

* отношение времени сортировки массива алгоритмом QuickSort ко времени сортировки алгоритмом Radix-Counting Sort – **1.8379134**;
* отношение времени сортировки массива алгоритмом QuickSort ко времени сортировки алгоритмом Radix-Counting Sort, оптимизированным под 8-процессорную вычислительную систему – **2.0982673**.

**Выводы**:

1. Существует алгоритм поразрядной сортировки со сложностью, меньшей линейной вычислительной сложности *O(n / log(n))*;
2. Следует выбирать алгоритмы на основе типа входных данных, которые позволяют решить задачу наиболее эффективно.

# **Контрольные вопросы**

Ниже представлены ответы на контрольные вопросы к лабораторной работе.

## **Вопрос 1**

**Назовите причины расслоения оперативной памяти.**

Расслоение памяти позволяет повысить пропускную способность оперативной памяти за счет ее компоновки из нескольких банков. Процедуры обращений к нескольким банкам памяти при таком построении можно совместить.

## **Вопрос 2**

**Как в современных процессорах реализована аппаратная предвыборка?**

В суперскалярных процессорах суть предвыборки заключается в том, чтобы выгоднее загрузить вычислительный конвейер (производится считывание команд из памяти, опережающее ход вычислений). Команды после выборки размещаются в быстродействующем буфере предвыборки. Он может быть организован по принципу очереди, тогда команды поступают в порядке выполнения в программе. Высокое быстродействие буфера и наличие в нем значительного количества команд позволяет одновременно нагружать все конвейеры процессора.

Другой пример аппаратной предвыборки: суть предвыборки заключается в загрузке физического адреса не одной страницы, а сразу нескольких (8 или 16) за одно обращение к памяти (это возможно за счет расслоения памяти). Таким образом, уменьшается время получения физического адреса страницы оперативной памяти при загрузке его в TLB.

## **Вопрос 3**

**Какая информация храниться в TLB?**

У машины со страничной организацией виртуальной памяти в TLB хранятся физические адреса начала страниц в физической памяти (фреймов).

## **Вопрос 4**

**Какой тип ассоциативной памяти используется в кэш-памяти второго уровня современных ЭВМ и почему?**

В кэш-памяти второго уровня современных ЭВМ используется частично-ассоциативный тип. Он сочетает достоинства прямого и полностью ассоциативного способов отображения: экономия памяти тэгов (по сравнению с полностью ассоциативным кэшем) и уменьшение количества попаданий в один и тот же набор линеек кэша (по сравнению с прямым отображением).

## **Вопрос 5**

**Приведите пример программной предвыборки.**

На листинге 9.1 представлена часть программы из эксперимента 3.

Листинг 9.1 – Пример программной выборки

**int** \*p = get\_array(arr\_size); *// Get huge array from enternal world*

**int** tmp;

**for** (**int** a = 0; a < arr\_size; a += SIZEOF\_PAGE)

tmp = \*(**int** \*)(**int**(p) + a);

**for** (**int** a = 0; a < arr\_size; a++)

{

**int** elem = \*(**int** \*)(**int**(p) + a);

… *// Array processing*

}