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 «Программное обеспечение ЭВМ и информационные технологии»(ИУ7)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.04 «Программная инженерия»

О Т Ч Е Т
по лабораторной работе № 3

Название Организация памяти конвейерных суперскалярных ЭВМ

Дисциплина Архитектура элекронно-вычислительных машин

Студент: _____ Зайцева А. А.
подпись, дата Фамилия, И.О.

Преподаватель: _____ Попов А. Ю.
подпись, дата Фамилия, И. О.

Москва — 2021 г.

Цель работы

Основной целью работы является освоение принципов эффективного использования подсистемы памяти современных универсальных ЭВМ, обеспечивающей хранение и своевременную выдачу команд и данных в центральное процессорное устройство. Работа проводится с использованием программы для сбора и анализа производительности PCLAB.

В ходе работы необходимо 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.

Исследования расслоения динамической памяти

Исходные данные: размер линейки кэш-памяти верхнего уровня (128), объем физической памяти.

Настраиваемые параметры: максимальное расстояния между читаемыми блоками (32К), шаг увеличения расстояния между читаемыми 4-х байтовыми ячейками (128Б), размер массиваи (1М).

Графики полученных характеристик представлены на рисунке 1.

Рисунок 1 – Результат исследования расслоения динамической памяти

График показывает количество тактов работы алгоритма. Ось абсцисс отражает шаг приращения адреса читаемых данных. Ось ординат отображает количество тактов.

Следует учесть, что шаг чтения, меньший размера линейки кэш-памяти приводит к получению результирующей кривой, имеющей пилообразный характер: каждое второе обращение будет выполняться не к динамической памяти, а к кэш-памяти. Это явление показано на рисунке 2.

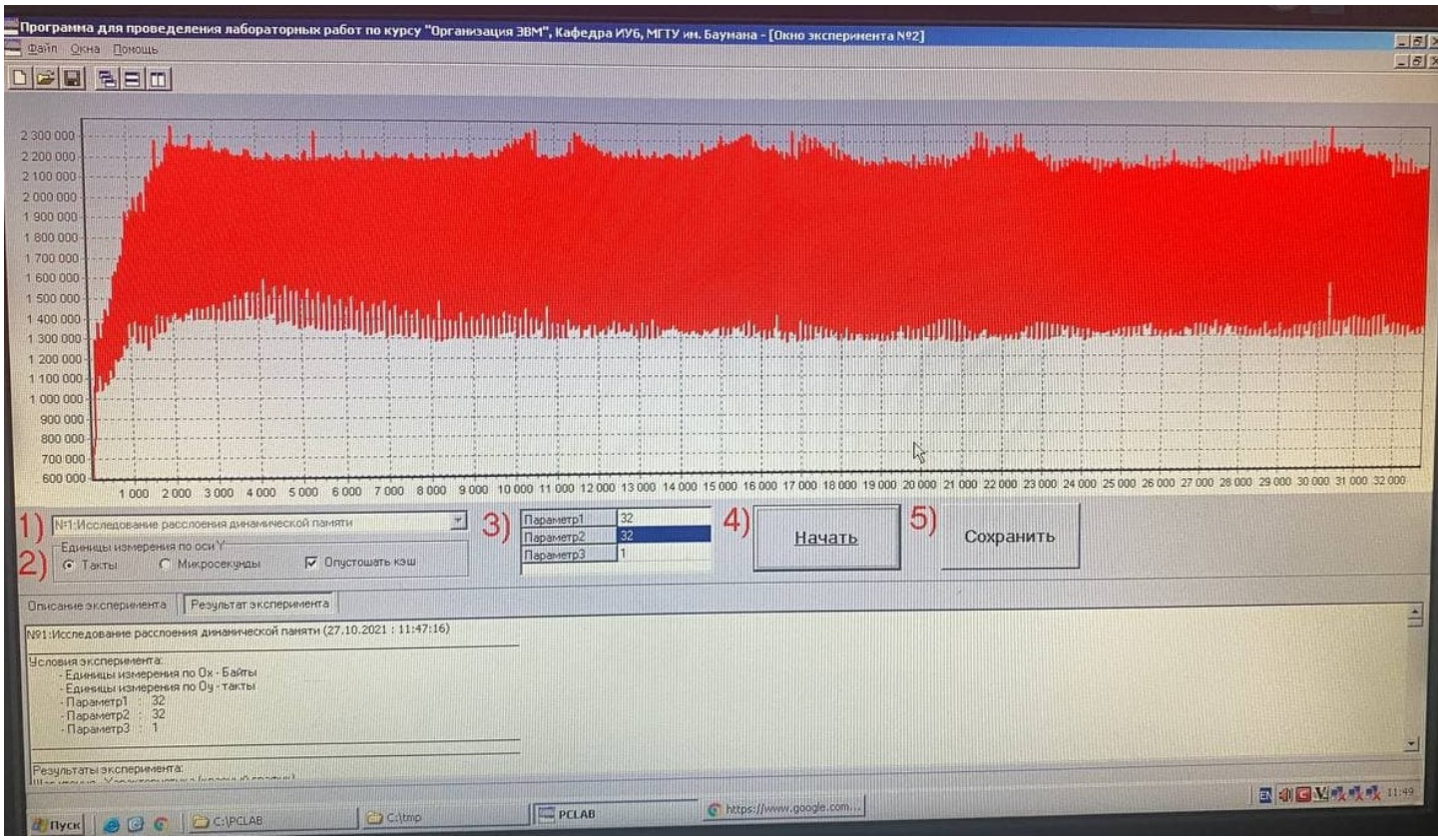

Рисунок 2 – Результат исследования расслоения динамической памяти с меньшим шагом увеличения расстояния между читаемыми 4-х байтовыми ячейками (32Б)

Результаты эксперимента:количество банков динамической памяти, размер одной страницы динамической памяти, количество страниц в динамической памяти.

По графику можно определить несколько параметров.

1. Минимальный шаг чтения динамической памяти, при котором происходит постоянное обращение к одному и тому же банку, соответствует первому локальному экстремуму полученной функции (точка T1=1024 байт).

По полученному значению шага T1 можно определить количество банков памяти: Б = T1/П = 1024/128=8, где П – объем данных, являющийся минимальной порцией обмена кэш-памяти верхнего уровня с оперативной памятью и соответствует размеру линейки кэшпамяти верхнего уровня.

2. При достижения глобального экстремума, после которого рост локальных экстремумов не происходит, определяется характерная точка T2 (4096 байт). Соответствующи данной точке шаг чтения является наихудшим при обращении к динамической памяти, т.к. приводит к постоянному закрытию и открытию страниц динамической памяти. Таким образом шаг T2 соответствует расстоянию (в байтах) между началом двух последовательных страниц одного банка. Зная ко-

личество банков, определяем размер одной страницы: РС = Т2/Б = 4096/8=512.

3. Зная параметры РС и Б, а также полный объем памяти О определяем количество страниц физической оперативной памяти: С = О/(РС*Б*П)=О/(512*8*128)

Вывод

Оперативная память расслоена и неоднородна, поэтому обращение к последовательно расположенным данным требует различного времени из-за наличия открытыя и закрытыя страниц динамической памяти. При этом чем больше адресное расстояние, тем больше время доступа. В связи с этим для создания эффективных программ необходимо учитывать расслоение памяти и размещать рядом данные для непосредственной обработки.

Сравнение эффективности ссылочных и векторных структур

Настраиваемые параметры: количество элементов в списке (1M), максимальная фрагментации списка (32K), шаг увеличения фрагментации (1K).

Графики полученных характеристик представлены на рисунке 3.

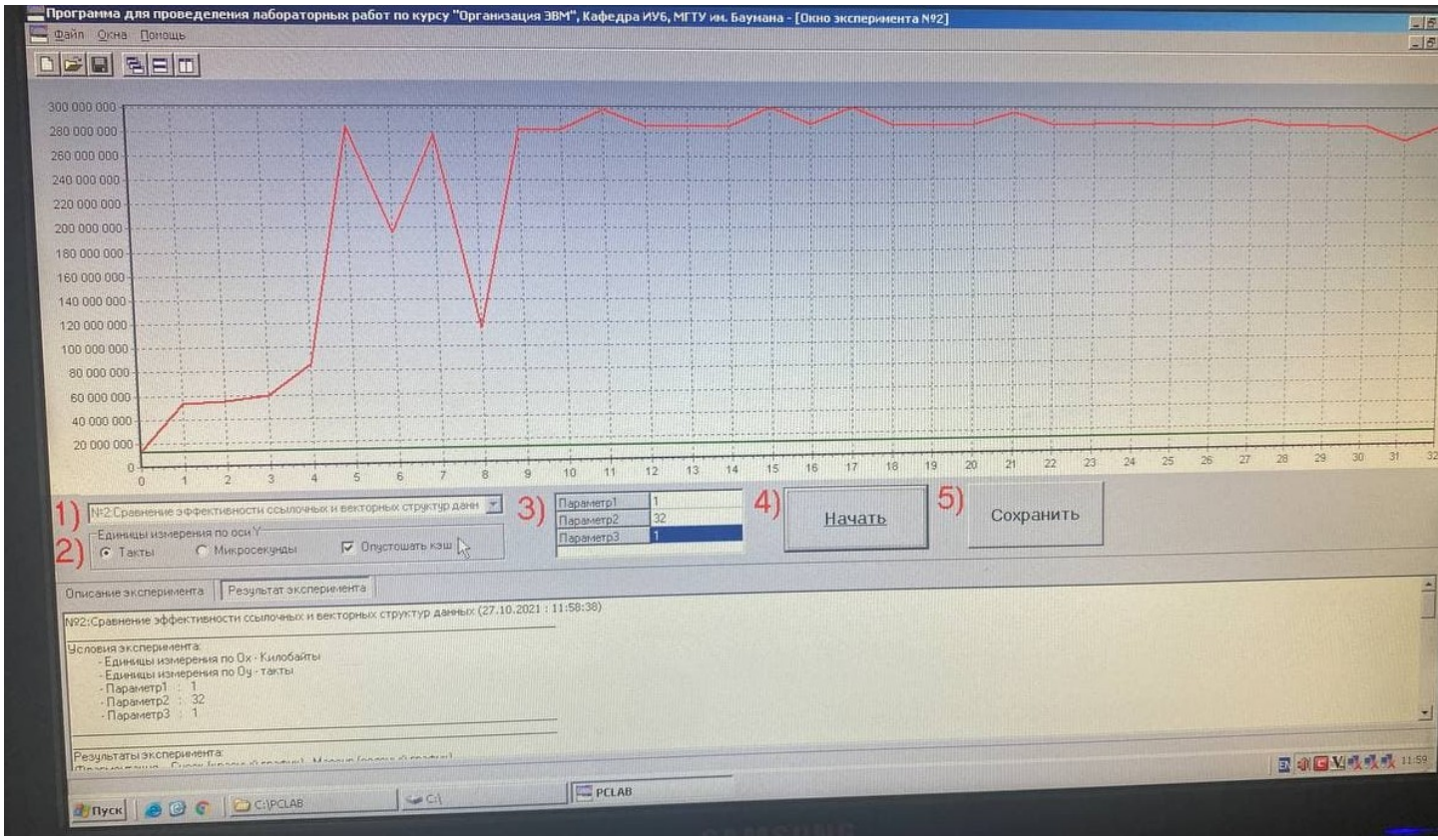

Рисунок 3 – Результат исследования эффективности ссылочных и векторных структур

Красный график (верхний) показывает количество тактов работы алгоритма, использующего список. Зеленый график (нижний) показывает количество тактов работы алгоритма, использующего массив. Ось абсцисс отражает фрагментацию списка.

Результаты эксперимента:отношение времени работы алгоритма, использующего зависимые данные, ко времени обработки аналогичного алгоритма обработки независимых данных = 20

Вывод

Видна проблема семантического разрыва: машина не присоблена к работе со ссылочными структурами. Использовать структуры данных надо с учётом скрытых технологических констант. Если алгоритм предполагает возможность использования массива, а списки не дают существенной разницы, то использование массива вполне оправдано.

Исследование эффективности программной предвыборки

Исходные данные: степень ассоциативности и размер TLB данных.

Настраиваемые параметры: шаг увеличения расстояния между читаемыми-данными (512Б), размер массиваи (128К).

Графики полученных характеристик представлены на рисунке 4.

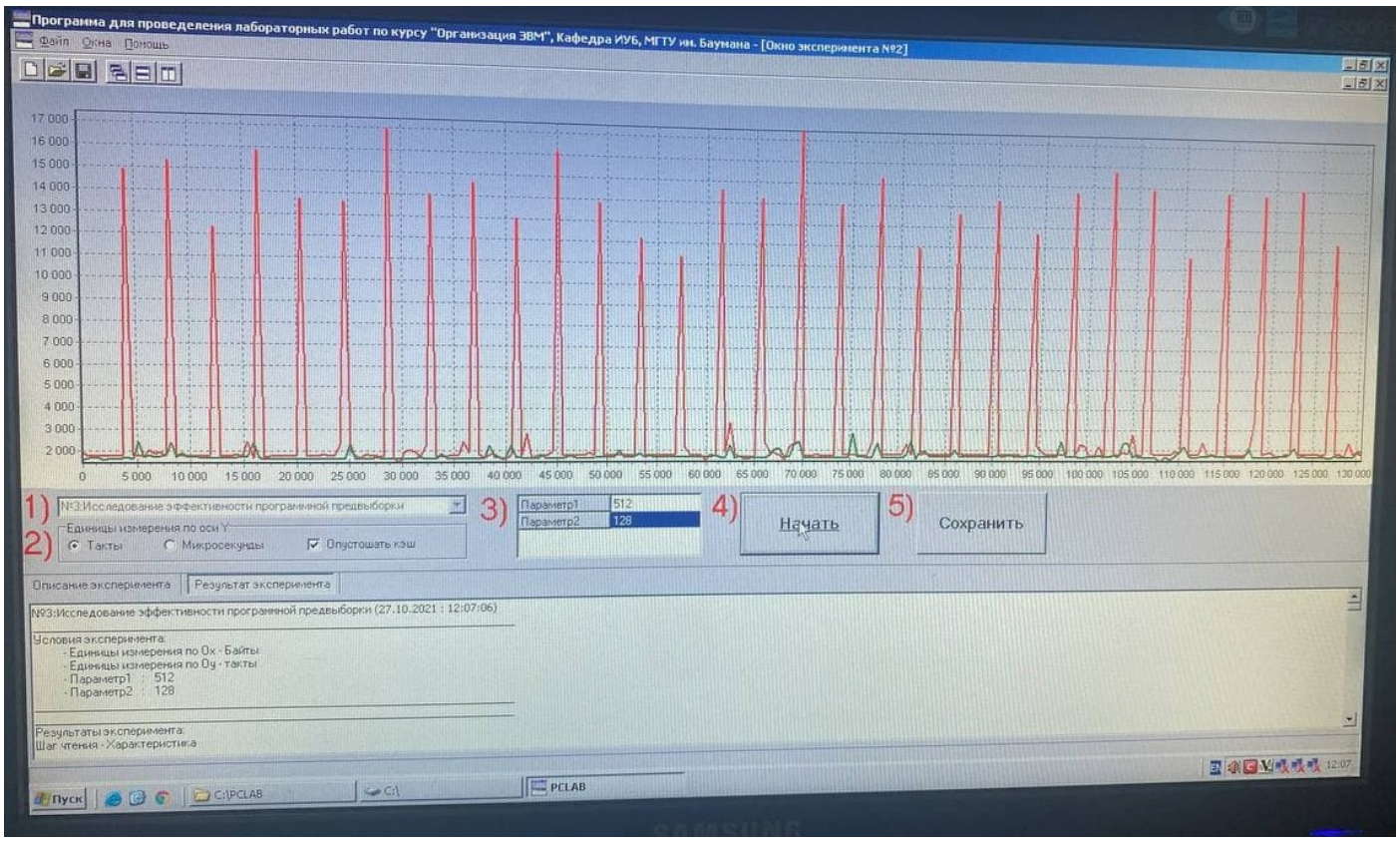

Рисунок 4 – Результат исследования эффективности предвыборки

Красный график (верхний с острыми пиками) показывает количество тактов работы алгоритма без предвыборки. Зеленый график (нижний без значимых пиков) показывает количество тактов работы алгоритма с использованием предвыборки. Ось абсцисс отражает смещение читаемых данных от начала блока.

Результаты эксперимента: отношение времени последовательной обработки блока данных ко времени обработки блока с применением предвыборки = 1,92; количество тактов первого обращения к странице данных = 15000

Вывод

Обработка больших массивов информации сопряжена с открытием большого количества физических страниц памяти. При первом обращении к странице памяти наблюдается увеличенное время доступа к данным в 20 раз, так как оно при отсутствии информации в TLB вызывает двойное обращение к оперативной памяти: сначала за информацией из таблицы страниц, а далее за востребованными данными. Поэтому для ускорения работы программы можно использовать предвыборку. Например, пока процессор занят некоторыми расчетами и не обращается к памяти, можно заблаговременно провести все указанные действия благодаря дополнительному запросу небольшого количества данных из оперативной памяти.

Также стоит стараться не использовать в программе массивы, к которым обращение выполняется только один раз.

Исследование способов эффективного чтения оперативной памяти

Исходные данные: адресное расстояние между банками памяти, размер буфера чтения.

Настраиваемые параметры: размер массива (2M), количество потоков данных (64).

Графики полученных характеристик представлены на рисунке 5.

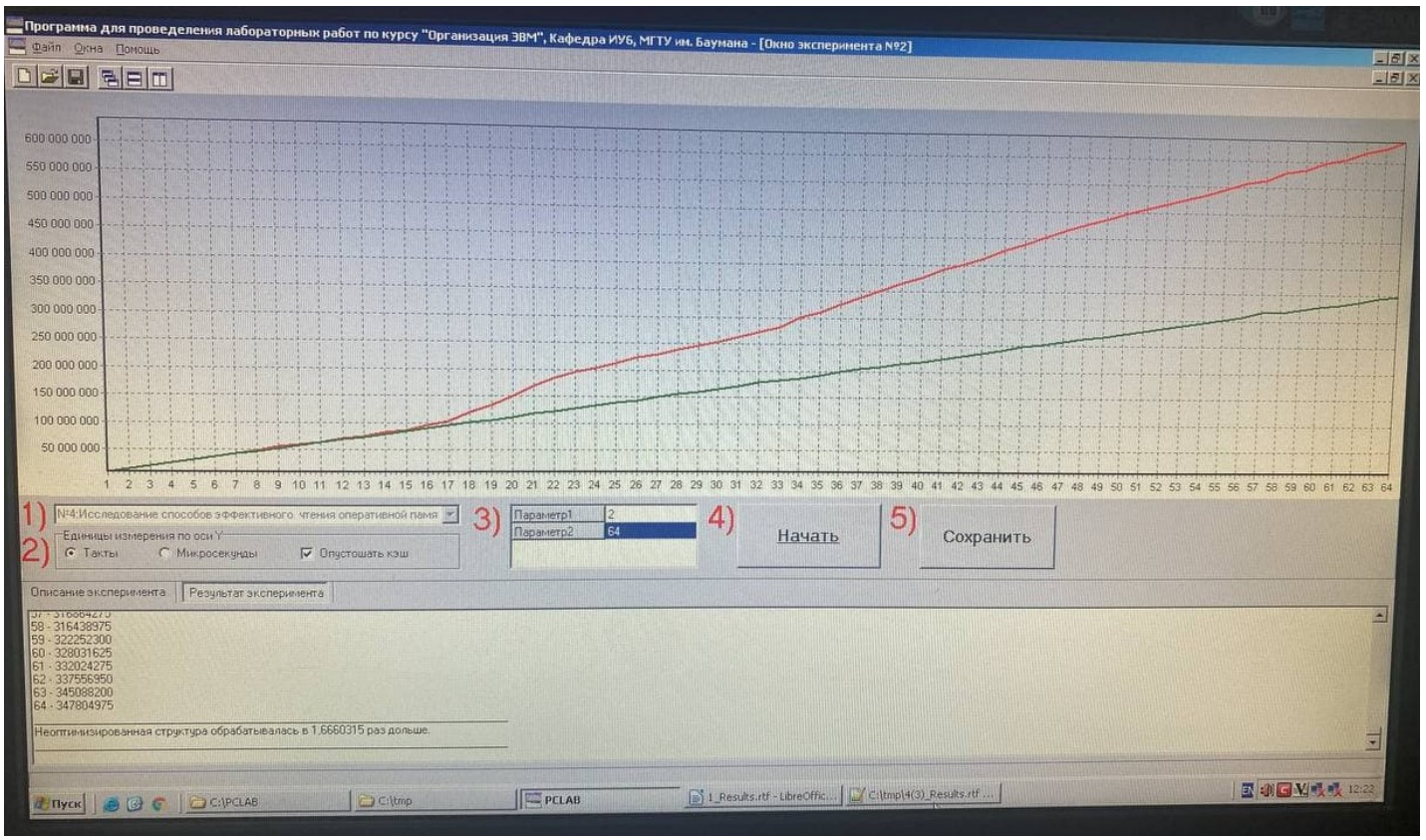

Рисунок 5 – Результат исследования оптимизирующих структур данных

Красный график (верхний) показывает количество тактов работы алгоритма, использующего неоптимизированную структуру. Зеленый график (нижний) показывает количество тактов работы алгоритма с использованием оптимизированной структуры. Ось абсцисс отражает количество одновременно обрабатываемых массивов.

Результаты эксперимента: отношение времени обработки блока памяти неоптимизированной структуры ко времени обработки блока структуры, обеспечивающей эффективную загрузку и параллельную обработку данных = 1,7.

Вывод

Эффективная обработка нескольких векторных структур данных без их дополнительной оптимизации не использует в должной степени возможности аппаратных ресурсов.

Для создания структур данных, оптимизирующих их обработку, необходимо передавать в каждом пакете только востребованную для вычислений информацию.То есть для ускорения алгоритмов необходимо правильно упорядочивать данные.

Исследование конфликтов в кэш-памяти

Исходные данные: размер банка кэш-памяти данных первого и второго уровня, степень ассоциативности кэш-памяти первого и второго уровня, размер линейки кэшпамяти первого и второго уровня.

Настраиваемые параметры: размер банка кэш-памяти (128K), размер линейки кэш-памяти (128б), количество читаемых линеек (32).

Графики полученных характеристик представлены на рисунке 6.

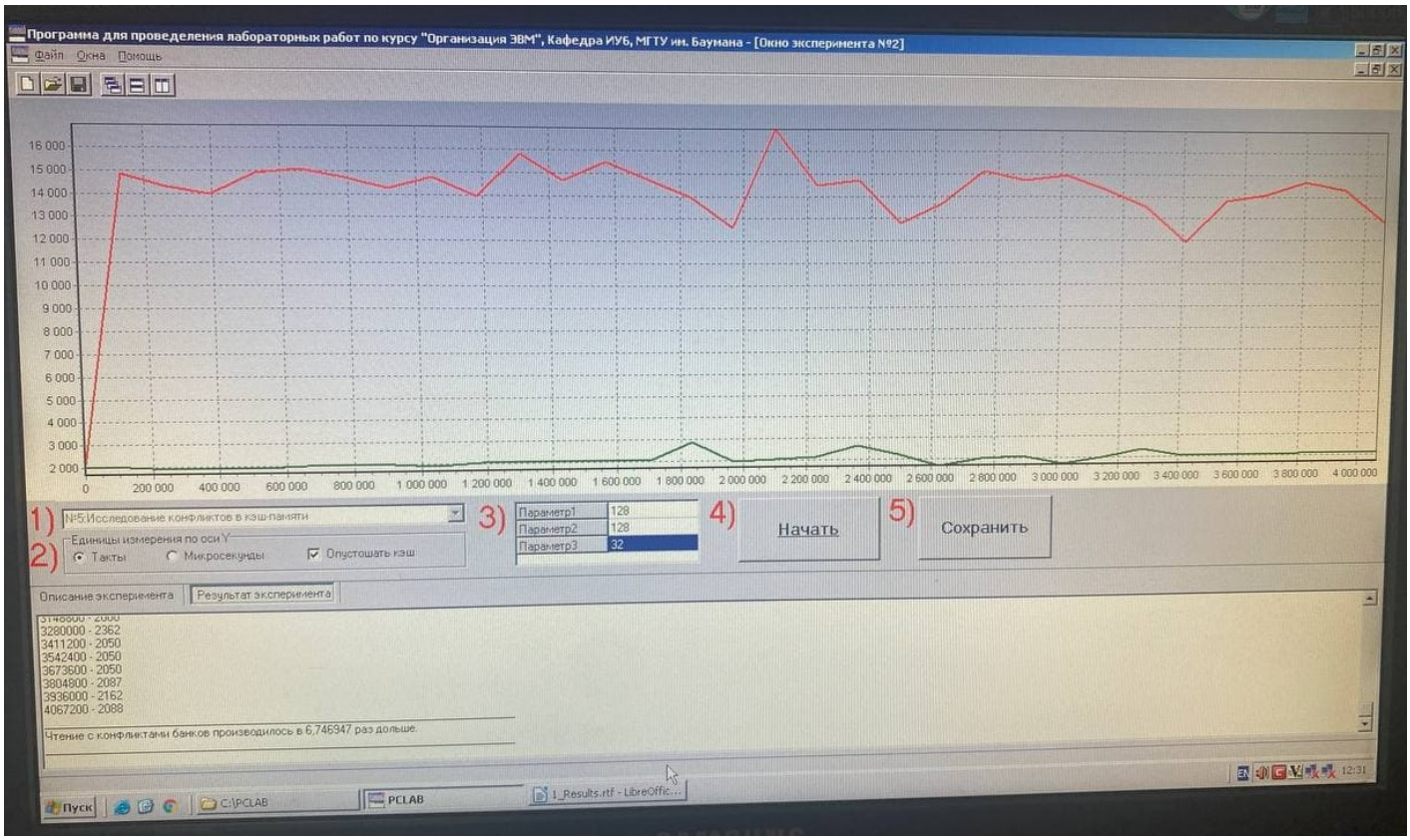

Рисунок 6 – Результат исследования конфликтов в кэш-памяти

Красный график (верхний) показывает количество тактов работы процедуры, читающей данные с конфликтами в кэш-памяти. Зеленый график (нижний) показывает количество тактов работы процедуры, не вызывающей конфликтов в кэш-памяти. Ось абсцисс отражает смещение читаемой ячейки от начала блока данных

Результаты эксперимента: отношение времени обработки массива с конфликтами в кэш-памяти ко времени обработки массива без конфликтов = 6,7.

Вывод

Попытка читать данные из оперативной памяти с шагом, кратным размеру банка, приводит к их помещению в один и тот же набор. Если же количество запросов превосходит степень ассоциативности кэш-памяти, т.е. количество банков или количество линеек в наборе, то наблюдается постоянное вытеснение данных из кэш-памяти, причем больший ее объем остается незадействованным. Кэш память ускоряет работу процессора в 6.7 раз

Сравнение алгоритмов сортировки

Исходные данные: количество процессоров вычислительной системы, размер пакета, количество элементов в массиве, разрядность элементов массива

Настраиваемые параметры: количество 64-х разрядных элементов массивов (1M), шаг увеличения размера массива (32K).

Графики полученных характеристик представлены на рисунке 7.

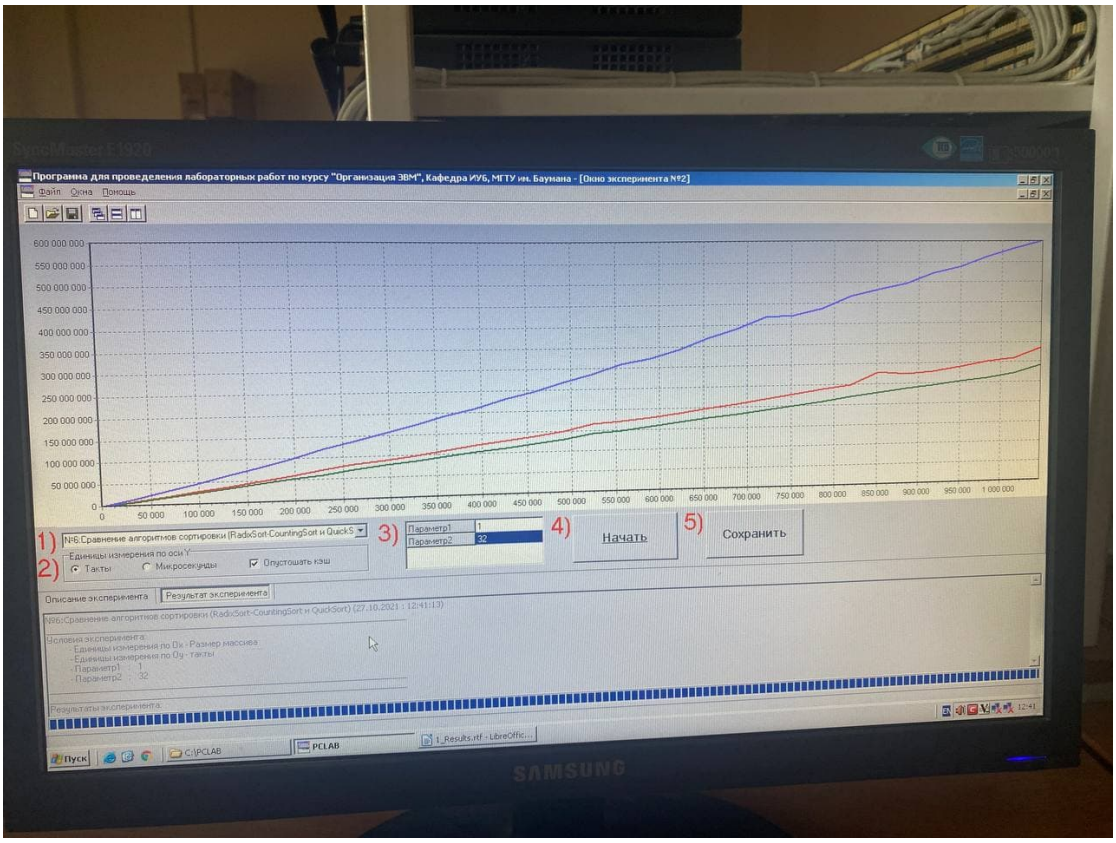

Рисунок 7 – Результат исследования алгоритмов сортировки

Фиолетовый график (верхний) показывает количество тактов работы алгоритма QuickSort. Красный график (средний) показывает количество тактов работы неоптимизированного алгоритма Radix-Counting. Зеленый график (нижний) показывает количество тактов работы оптимизированного под 8-процессорную вычислительную систему алгоритма Radix-Counting. Ось абсцисс отражает количество 64-разрядных элементов сортируемых массивов.

Результаты эксперимента: отношение времени сортировки массива алгоритмом QuickSort ко времени сортировки алгоритмом Radix-Counting Sort (= 1,4) и ко времени сортировки Radix-Counting Sort, оптимизированной под 8-процессорную вычислительную систему (=1,8).

Вывод

Существует алгоритм сортировки менее чем линейной вычислительной сложности.

Общий вывод

В результате выполнения лабораторной работы были изучены принципы эффективного использования подсистемы памяти современных универсальных ЭВМ.

В ходе работы проработан теоретический материал, касающийся особенностей функционирования подсистемы памяти современных конвейерных суперскалярных ЭВМ, изучены возможности программы PCLAB, изучены средства идентификации микропроцессоров, проведены исследования времени выполнения тестовых программ, сделаны выводы об архитектурных особенностях используемых ЭВМ.