ЛАБОРАТОРНАЯ РАБОТА №8
ВЛИЯНИЕ КЭШ-ПАМЯТИ НА ВРЕМЯ ОБРАБОТКИ МАССИВОВ

Цели работы

1. Исследование зависимости времени доступа к данным в памяти от их объема.
2. Исследование зависимости времени доступа к данным в памяти от порядка их обхода.

1. ВЛИЯНИЕ КЭШ-ПАМЯТИ НА СКОРОСТЬ ДОСТУПА К ДАННЫМ

На практике часто встречается задача обработки массивов данных. С точки зрения скорости выполнения программы, важно в каком порядке обрабатываются элементы массива, т.к. от этого будет зависеть, насколько эффективно будет работать кэш-память. Основными особенностями организации кэш-память, которые играют важную роль при обработке массивов данных, являются блочное кэширование данных и аппаратная предвыборка данных в кэш. Кроме того, существенно, умещается ли обрабатываемый массив в кэш-памяти.

1.1 Влияние уровней кэш-памяти

Иерархия памяти включает несколько уровней кэш-памяти разного размера и с разным временем доступа. Допустим, некоторая программа производит многократную обработку элементов массива. Если построить график зависимости среднего времени доступа к одному элементу массива от размера массива, то он должен иметь нелинейный характер. При малых размера массива, когда все данные умещаются в кэш-памяти первого уровня, время доступа к элементу будет наименьшим, и не будет меняться при увеличении размера массива. Если размер массива превысит размер кэш-памяти первого уровня, то весь массив целиком уже не сможет в нем

разместиться. Поэтому при обращении к некоторым элементам массива в кэш-памяти первого уровня будут случаться кэш-промахи, и элементы будут загружаться из кэш-памяти второго уровня (или оперативной памяти). В результате, чем больше кэш-промахов происходит, тем больше будет среднее время доступа к элементу, вплоть до времени доступа к следующему уровню иерархии памяти. В результате с увеличением размера массива среднее время доступа к элементу будет ступенчато возрастать. Таким образом, анализ графика может показать, каковы объемы различных уровней кэш-памяти, имеющихся в системе.

1.2 Влияние блочной передачи данных

Данные из оперативной памяти в кэш-память считываются целыми блоками. Размер блока равен одной или нескольким кэш-строкам. Если элементы в массиве обрабатываются последовательно один за другим, то попытка чтения первого элемента кэш-строки вызывает копирование всего блока из медленной оперативной памяти в кэш-память. Чтение нескольких последующих элементов выполняется намного быстрее, т.к. они уже находятся в быстрой кэш-памяти.

1.3 Влияние аппаратной предвыборки данных

В большинстве современных микропроцессоров реализована аппаратная предвыборка данных. Она устроена таким образом, что при последовательном обходе очередные данные считываются из оперативной памяти еще до того, как к ним произошло обращение. Кэш-контроллеры с высокой вероятностью распознают последовательный обход памяти и обеспечивают эффективную предварительную загрузку данных в кэш-память. Как следствие, вероятность кэш-промахов значительно снижается. Если же элементы массива обрабатываются в более сложном порядке, то либо кэш-контроллер его не распознает, и тогда аппаратная предвыборка работать не будет, либо может распознать неправильно, что повлечёт

вытеснение ещё нужных данных из кэш-памяти и увеличение среднего времени доступа к элементу массива.

2. ИССЛЕДОВАНИЕ ВЛИЯНИЯ КЭШ-ПАМЯТИ

Для изучения влияния кэш-памяти на скорость доступа к данным в данной лабораторной работе требуется построить программу, выполняющую обработку данных различного размера тремя характерными способами: последовательно в сторону увеличения адресов, последовательно в сторону уменьшения адресов и в случайном порядке. Исследование времени выполнения программы в зависимости от порядка обхода и объема обрабатываемых данных позволит пронаблюдать влияние кэш-памяти.

2.1 Перебор размеров массива

Для определения среднего времени доступа к данным, программа должна многократно выполнять чтение элементов массива заданного размера (N) в заданном порядке. Интересующий нас диапазон изменения размеров массива определяется следующими границами:

- минимальное значение N_{min} выбирается заведомо меньше, чем размер кэш-памяти данных 1-го уровня (например, 1 Кбайт – 256 элементов типа `int`).
- максимальное значение N_{max} выбирается заведомо больше, чем размер кэш-памяти последнего уровня (например, 32 МБайта).

Программа должна перебирать размеры массива от N_{min} до N_{max}, для каждого конкретного N определяя среднее время доступа к элементу массива. Шаг изменения N должен быть достаточно маленьким, чтобы увидеть все существенные изгибы на графике, и достаточно большим, чтобы время выполнения теста не было слишком большим. Можно использовать переменный шаг, который растет вместе с возрастанием N.

2.2 Выполнение обхода

Для каждого конкретного размера массива N программа должна выполнить многократное чтение элементов массива в заданном порядке. Чтобы во время обхода не тратить время на вычисление индекса каждого следующего элемента, используется следующий приём. Значения элементов массива заполняются таким образом, чтобы сформировать односвязный циклический список, в котором значение очередного элемента представляет собой номер следующего. Обход тогда может быть выполнен циклом следующего вида:

```
for (k=0, i=0; i<N*K; i++) k = x[k];
```

Здесь N – размер массива, K – число обходов массива.

Таким образом, для каждого конкретного N программа должна выполнять следующие действия:

1. Заполнить значения элементов, чтобы они образовали циклический односвязный список в соответствии с требуемым порядком обхода.
2. Выполнить многократный обход массива. Замерить время обхода.

2.3 Способы обхода элементов массива

В рамках лабораторной работы предлагается сравнить время доступа к данным для трех характерных способов обхода:

- Прямой обход – в сторону увеличения адресов,
- Обратный обход – в сторону уменьшения адресов,
- Обход элементов в случайном порядке.

Для каждого способа обхода в программе необходимо заполнить значения элементов массива таким образом, чтобы они образовали односвязный циклический список, захватывающий все элементы массива (рис. 1). Особое внимание следует обратить на заполнение массива для случайного обхода. Например, если в варианте для случайного массива, приведённом на рис. 1в, поменять местами значения ячеек 5 и 6, то в обходе будут участвовать только

элементы с номерами 0, 6 и 7. Необходимо, чтобы при обходе массива участвовали все элементы.

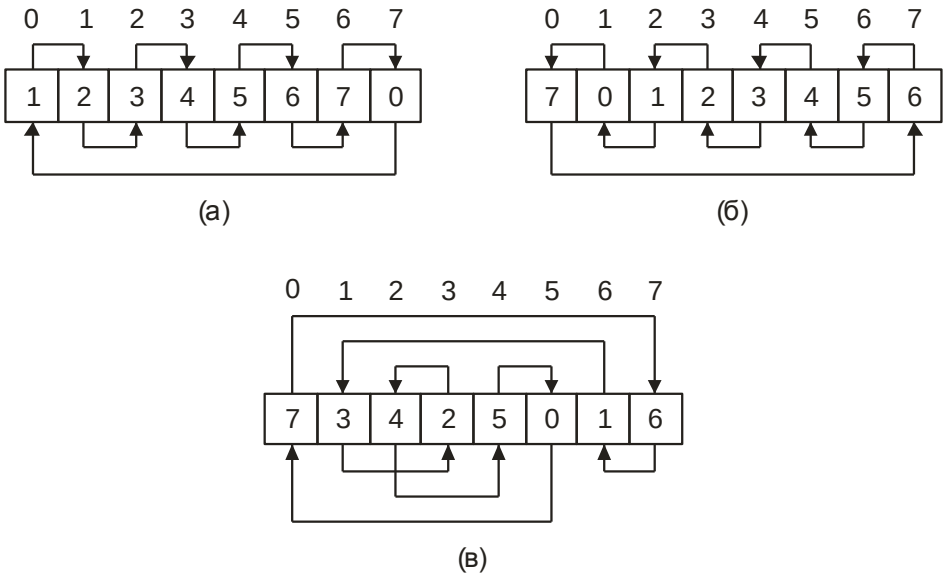

Рис. 1. Примеры способов заполнения элементов массива для выполнения прямого (а), обратного (б) и случайного (в) обходов

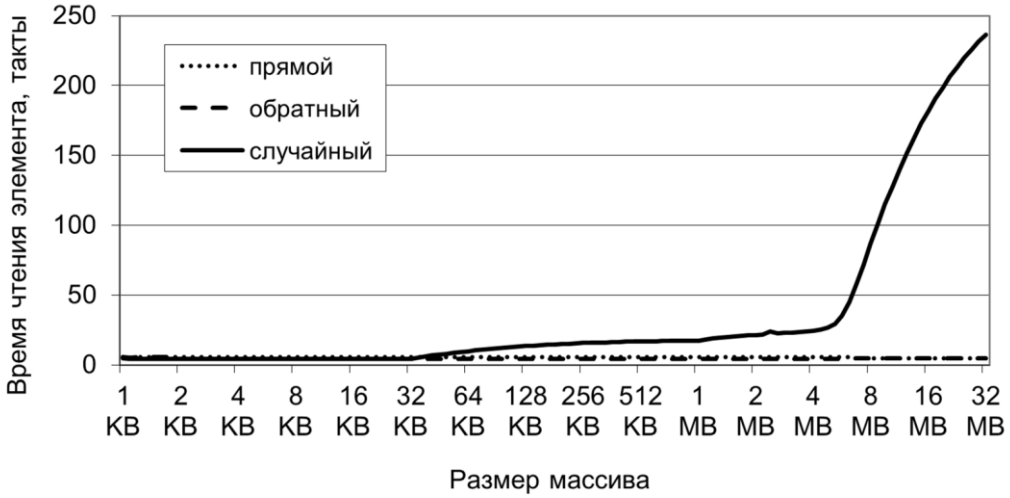

Рис. 2. Зависимость среднего времени чтения элемента массива от размера массива при различных способах обхода для процессора Intel Xeon E5420.

На рис. 2 приведен пример графиков, полученных на процессоре Intel Xeon E5420 (L1: 32 KB, L2: 6 MB). Видна разница в скорости

последовательного и случайного обходов массива. На соответствующих местах графика случайного обхода видно возрастание времени обращения к элементам.

Примечание 1. Компилировать программу нужно с ключом оптимизации –O1 чтобы исключить лишние обращения в память за счёт размещения переменных на регистрах.

Примечание 2. Перед измерением времени для каждого размера N необходимо осуществить однократный обход массива, чтобы «прогреть кэш-память», то есть выгрузить из кэш-памяти посторонние данные, разместив там (по возможности) необходимые нам данные.

Примечание 3. Результирующий график может оказаться сильно "замусоренным" высокими пиками. Такое бывает, если на машине работают посторонние "тяжелые" программы. В этом случае рекомендуется встроить в программу дополнительный цикл, который бы прогонял тест несколько раз для каждого размера массива, и автоматически сохранял минимальное время по нескольким замерам.

3. ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ

1. Написать программу, многократно выполняющую обход массива заданного размера тремя способами.
2. Для каждого размера массива и способа обхода измерить среднее время доступа к одному элементу (в тактах процессора). Построить графики зависимости среднего времени доступа от размера массива.
3. На основе анализа полученных графиков:
   - определить размеры кэш-памяти различных уровней, обосновать ответ, сопоставить результат с известными реальными значениями;
   - определить размеры массива, при которых время доступа к элементу массива при случайном обходе больше, чем при прямом или обратном; объяснить причины этой разницы во временах.

4. Составить отчет по лабораторной работе. Отчет должен содержать следующее.
    - Титульный лист.
    - Цель лабораторной работы.
    - Описание способа заполнения массива тремя способами.
    - Графики зависимости среднего времени доступа к одному элементу от размера массива и способов обхода.
    - Полный компилируемый листинг реализованной программы и команду для ее компиляции.
    - Вывод по результатам лабораторной работы.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое кэш-память? Какую проблему она решает?
2. Какой способ обхода данных в памяти является самым быстрым? Почему?
3. Какой способ обхода данных в памяти является самым медленным? Почему?
4. Для каких программ наличие кэш-памяти дает выигрыш во времени работы?
5. Назовите две основных причины, по которым случайный обход массива дольше, чем прямой и обратный.
6. Почему тип обхода не сказывается на времени доступа к памяти, если массив умещается в кэш-памяти первого уровня?
7. Чем отличаются программная и аппаратная предвыборка данных?