# ИССЛЕДОВАНИЕ АЛГОРИТМА ЗАМЕЩЕНИЯ LAST LEVEL CACHE НА ПРИМЕРЕ ПРОЦЕССОРОВ INTEL

*Гагуа Д.Р., Кудырко А.А.*

*Белорусский государственный университет информатики и радиоэлектроники*
*г. Минск, Республика Беларусь*

*Перцев Д.Ю. – ст. преподаватель*

Представлены проблемы, связанные с определением степени ассоциативности кэш-памяти процессора. Рассмотрены причины их появления и возможные способы их устранения.

В микроархитектуре x86 кэш играет существенную роль в ускорении взаимодействия АЛУ процессора и оперативной памяти. Типовой процессор включает иерархию уровней кэшей, каждый из которых построен на основе множественно-ассоциативной архитектуры. Зная, как организован доступ на каждом уровне кэша, возможно существенно ускорить вычисления. Например, код перемножения матриц большого объема при оптимизации с учетом правил доступа в кэш может быть ускорен более чем в 5 раз.

Основной способ определения степени ассоциативности кэш памяти сводится к следующему псевдокоду (Nmax - максимальная предполагаемая степень ассоциативности):

```
for (int N = 2; N < Nmax; N++) {
    1) init
    2) t = 0;
       for (int k = 0; k < 1000000; k++) {
           t = arr[t];
       }
}
```

Измеряя время работы шага 2 и сравнивая результат между собой, можно зафиксировать резкое увеличение времени работы цикла, т.е. “словить” время перехода между уровнями кэша либо выход в ОЗУ. Пример этапа инициализации init для N = 2 показан на рис.1.

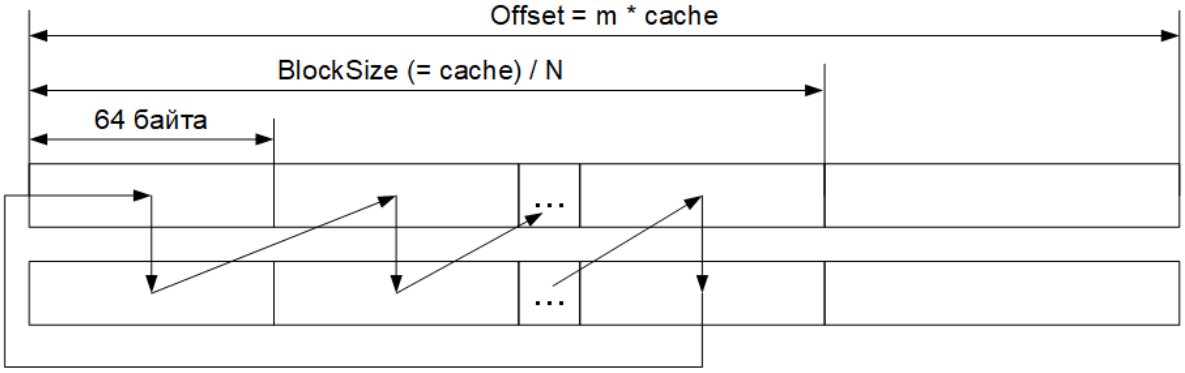


Рисунок 1 – Пример инициализации кэша

Однако проведенные эксперименты показывают, представленный алгоритм работает только при переходе между уровнями кэш-памяти и прекращает показывать корректные результаты при доступе в ОЗУ для большинства x86-процессоров, начиная с процессоров Intel Core2Duo. Это связано с тем, что начиная с данного поколения процессоров, компания Intel перешла на использование Smart Cache для последнего уровня (Last Level Cache) и схема организации изменилась (рис. 2).

Last Level Cache делится на фрагменты одинакового размера (slice). Предполагается, что данные из ОЗУ могут попасть в произвольный фрагмент, однако контроллер памяти "пытается" учитывать, для какого ядра предназначены данные и размещает их в максимально близкий к потребителю. С учетом этого, традиционный алгоритм замещения LRU (последний используемый) был замещен на хэш-функции, на основе которой вычисляется, в какой фрагмент и по какому адресу требуется разместить результат чтения памяти. Данное заключение, в том числе, подтверждается разными статьями [1-4]. Однако из-за того, что способ вычисления хэш-функции не документирован, такие изменения приводят к тому, что схема перемещения по памяти, описанная на рис. 1, перестает

работать. Статьи [1, 3] описывают подходы к анализу и способу вычисления хэш-функции.

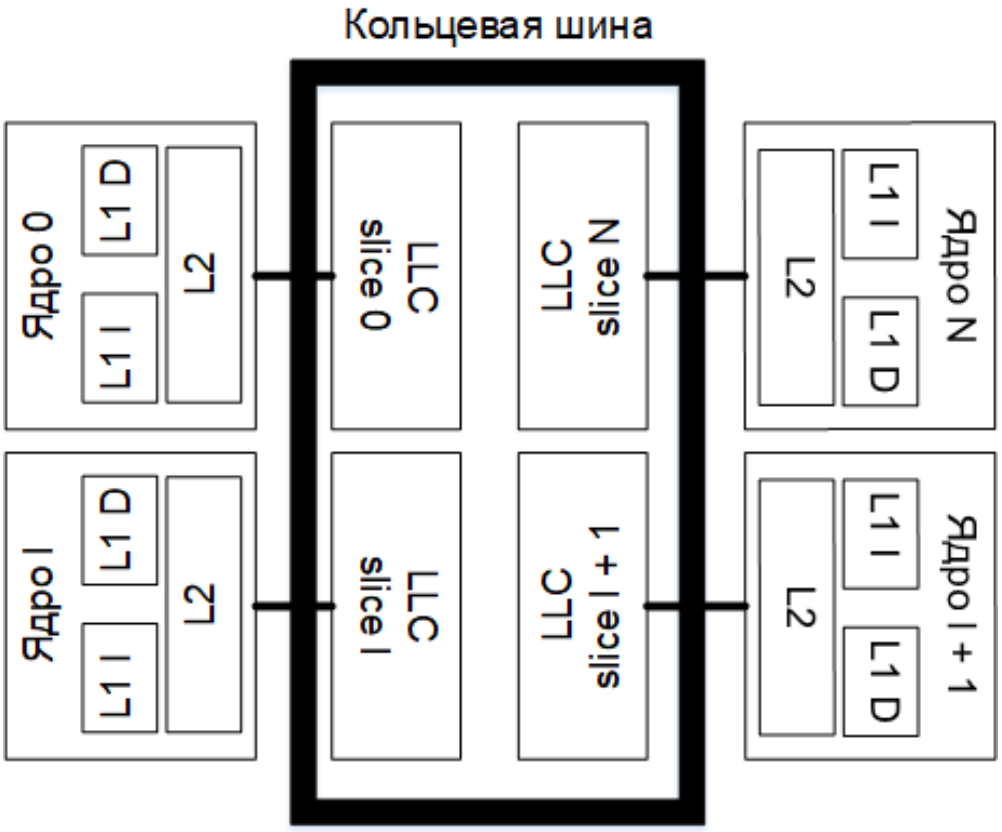


Рисунок 2 – Схема аппаратной реализации LLC процессора Intel

***Список использованных источников:***

1. Zhipeng W., Zehan C., Mingyu Ch.Cracking Intel Sandy Bridge's Cache Hash Function / W. Zhipeng [et al.]

2. Yuval Ya., Qian G., Fangfei L., Ruby B. L., Gernot H. Mapping the Intel Last-Level Cache / Ya Yuval [et al.]

3. Cl´ementine M., Nicolas Le S., Christoph N., Olivier H., Aur´elien F. Reverse Engineering Intel Last-Level Cache Complex Addressing Using Performance Counters / M. Cl´ementine [et al.]

4. Farshin A., Roozbeh A., Kostić D., Maguire G. Q. Jr. Make the Most out of Last Level Cache in Intel Processors / A. Farshin [et al.]