Домашняя работа №3

Кэш-память

Цель работы: закрепление материала по теме «кэш-память» путем решения задач по данной теме.

Порядок выполнения работы:

1. Решение приложенных задач согласно варианту.
2. Приведение описания решения.

Содержание отчета

1. Условия задачи (вместо заголовка «Теоретическая часть» будет заголовок «Условие задачи»);
2. Подробное решение задачи.

Примечания:

1. Распределение схем согласно вариантам.
2. Ответы на вопросы в конце документа.
3. В поле «Ссылка на отчет» требуется ссылка на ответ, а не на диск, где лежит отчет. Начиная с этой работы, если ссылка на отчет приложена неверно, то отчет не принимается.
4. «Шаблон отчета» можно найти в сообщении с тз дз2 (ничего не было обновлено с прошлой дз) .

Условия задач

Вариант 1. Предположим, что у вас есть система с 1 тактом на инструкцию. Доступ к данным осуществляется только через инструкции по загрузке и сохранению. Эти обращения составляют 25% от общего числа инструкций. Штраф за промах составляет 50 тактов (общее число тактов на операцию), а коэффициент промахов 5%.

Определите ускорение, полученное при отсутствии промахов кэша, по сравнению со случаем, когда есть промахи.

В ответе нужно представить число без дополнительных переменных (хотя в самом решении могут использоваться переменные без значения). Ожидается использование следующих переменных: количество инструкций, время одного такта, количество обращений по инструкциям, количество обращений по данным, коэффициент промахов, штраф за промах. Можно вводить и другие переменные с их описанием.

Варианты 2 и 3. Имеется следующее определение глобальных переменных и функций

Вариант	Глобальные переменные	Функции

2	`unsigned int size = 1024 * 1024;`
	`double x[size];`
	`double y[size];`
	`double z[size];`
	`double xx[size];`
	`double yy[size];`
	`double zz[size];`
		`void f(double w)`
		`{`
		`  for (unsinged int i=0; i<size; ++i)`
		`  {`
		`  x[i] = xx[i] * w + x[i];`
		`  y[i] = yy[i] * w + y[i];`
		`  z[i] = zz[i] * w + z[i];`
		`  }`
		`}`

3	`unsigned int size = 1024 * 1024;`
	`double x[size];`
	`double y[size];`
	`double z[size];`
	`double xx[size];`
	`double yy[size];`
	`double xz[size];`
		`void f(double w) {`
		`  for (unsinged int i=0; i<size; ++i)`
		`  {`
		`  x[i] = xx[i] * w + x[i];`
		`  }`
		`  for (unsinged int i=0; i<size; ++i)`
		`  {`
		`  y[i] = yy[i] * w + y[i];`
		`  }`
		`  for (unsinged int i=0; i<size; ++i)`
		`  {`
		`  z[i] = zz[i] * w + z[i];`
		`  }`
		`}`

Рассмотрим систему с L1 кэшем данных с ассоциативностью 4-way размером 32 КБ и размером строки 64 байта. Кэш L2 представляет собой 8-way ассоциативный кэш размером 1 МБ и размером строки 64 байта. Алгоритм вытеснения: LRU. Массивы последовательно хранятся в памяти, и первый из них начинается с адреса, кратного 1024.

Определите процент попаданий (число попаданий к общему числу обращений) для кэшей L1 и L2 для выполнения предложенной функции.

В ответе нужно представить два числа, равных % попаданий для L1 и L2 кэшей.

Вариант 4. Имеем следующий фрагмент кода:

```
struct element
{
    double x, y, ax, ay, vx, vy, a, b;
};

void f(element arr [], int n, double asqr)
{
    for (int i=0; i<n; ++i)
    {
        arr[i].x += arr[i].vx * asqr + 0.5 * arr[i].ax * asqr * asqr;
    }
    for (int i=0; i<n; ++i)
    {
        arr[i].y += arr[i].vy * asqr + 0.5 * arr[i].ay * asqr * asqr;
    }
}
```

Функция f выполняется для массива из 1000 элементов в системе с кэшем данных L1 размером 32 КБ и 4-way ассоциативностью. Размер блока составляет 64 байта. Предположим, что массив arr выровнен по адресу, кратному 64.

Необходимо определить количество кэш-промахов и процент промахов (число промахов к общему числу обращений).

В ответе нужно представить два числа: количество кэш-промахов и % промахов.

Вариант 5. Имеется следующий фрагмент кода:

```
double a[256][256], b[256][256], c[256][256], d[256][256];

for (int i=0; i<256; ++i)
{
    for (int j=0; j<256; ++j)
    {
        a[i][j] = b[i][j] + c[i][j];
    }
}

for (int i=0; i<256; ++i)
{
    for (int j=0; j<256; ++j)
    {
        d[i][j] = b[i][j] - c[i][j];
    }
}
```

Также имеется система с полностью ассоциативным кэшем данных размером 16 КБ (размер линии 64 байта). Штраф за промахи равняется 16 тактов. Пусть промахи записи в кэше непосредственно отправляются в буфер записи минуя кэш без дополнительных задержек.

Определите частоту попадания для приведенного кода, предполагая, что переменные i и j назначены регистрам процессора. Под частотой попаданий понимаем число попаданий к общему числу ~~промахов~~ (обращений).

В ответе нужно представить число, равное частоте попаданий.

FAQ

1. **Q:** В 4ом варианте дз учитывать переменные i, n, asqr, или они как и в 5 варианте лежат в регистрах?
   A: Нас интересуют больше обращения к массиву, так что положим, что i, n, asqr лежат в регистрах.

2. **Q:** Штраф за промах составляет 50 тактов (общее число тактов на операцию). Что значит фраза в скобках? То есть у нас в эти 50 тактов входят и обращение к кэшу, и после этого обращение к памяти?
   A: Да, последний вопрос и есть ответ на первый)

3. **Q:** В разделе "Условие задачи" предполагается копипаста из самого условия или требуются какие-то дополнения?
   A: Копипаста условия задачи согласно варианту

5. **Q:** Определить по сравнению, это означает отношение или разность? (также к вопросу, что имеется в виду в первой задаче под ускорением)
   A: Отношение

6. **Q:** Вариант 2,3. Перед тем как зайти в первый цикл кэш у нас пустой? Или при объявлении в глобальной памяти x,y,z,xx,yy,zz они кэшировались?
   A: Полагаем, что изначально кэш пустой

7. **Q:** Вопросы про алгоритмы вытеснения.
   A: Если алгоритм вытеснения важен, то он явно указан в условии. Если его нет, то для решения задачи это не важно.

8. **Q:** Вопрос такой, чем отличаются количество инструкций, количество обращений по инструкциям и количество обращений по данным?
   A: Количество инструкций это Instruction Count (IC). Для понимания предлагаю воспринимать «количество инструкций» как «количество команд», где под командой понимается «инструкция + данные». Количество обращений как раз идет про уже отдельно про инструкции внутри «команды» и отдельно про данные.

9. **Q:** Вариант 5. Указано, что промахи записи отправляются в буфер записи, минуя кэш. Это означает, что мне нужно беспокоиться только из-за промахов чтения?
   A: Это значит, что если у тебя происходит промах, то у тебя нет штрафа за него (без дополнительных задержек).