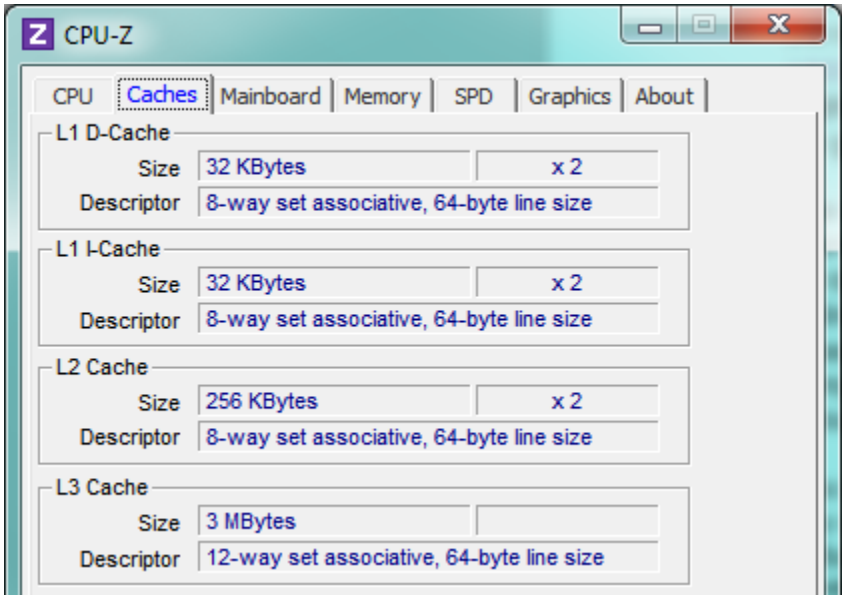
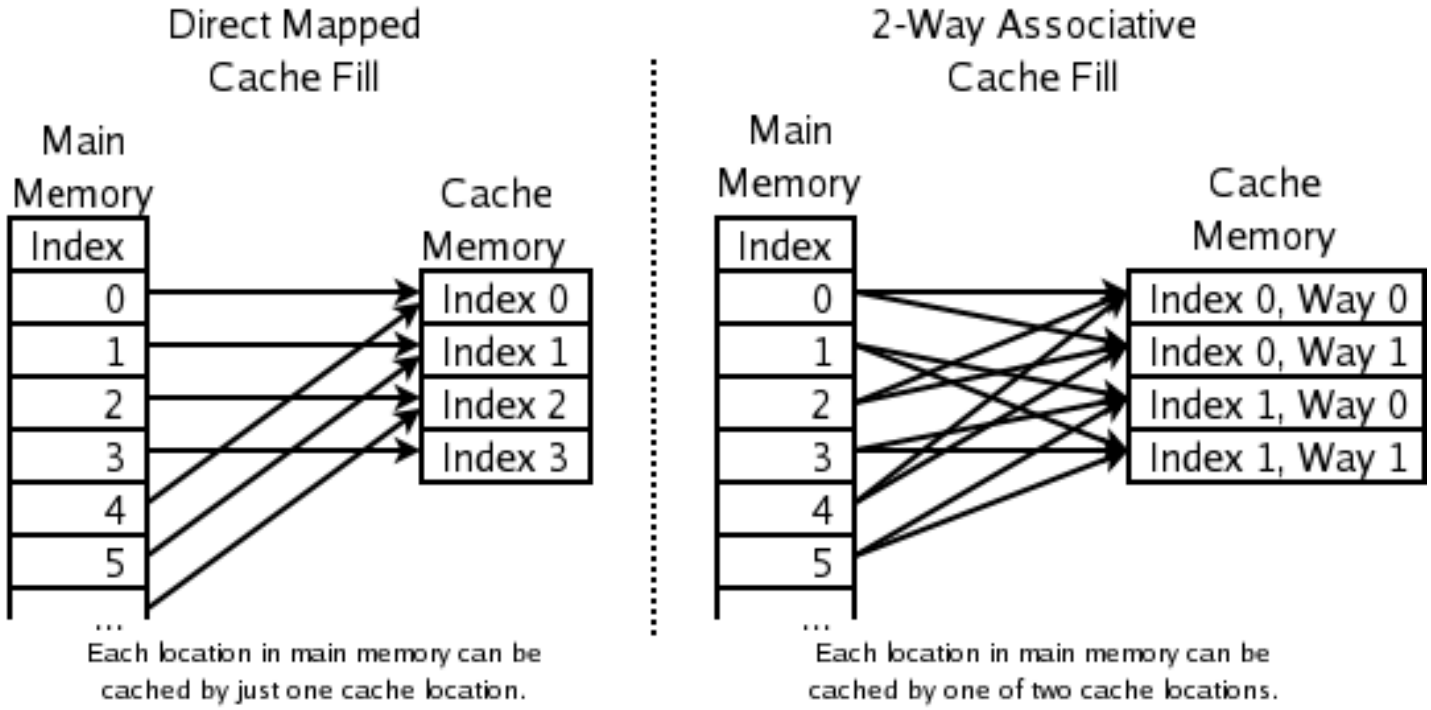
Кеш процессора

- Несколько видов кешей:
  - кеш инструкций;
  - кеш данных (L1, L2, L3);
  - буфер ассоциативной трансляции (TLB).
- Кеш обычно организован в кеш-линии (часто 64 байта для x86-процессоров).
- Номер линии получается делением адреса на 64.



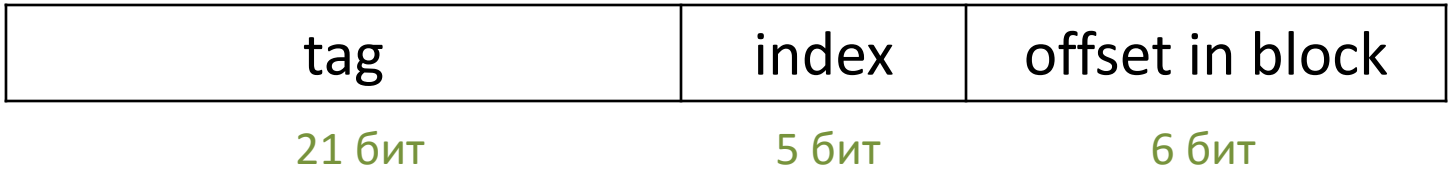
Ассоциативность кеша

- *direct mapped* — место определено однозначно
- *fully associative* — конкретный блок из памяти может быть помещён в любое место в кеше
- на практике используется компромиссное решение (2, 4, 8-канальный кеш)



Структура кеша

- Разбиение адреса (от старших битов к младшим)



- **Пример: Pentium 4.**

  «four-way set associative L1 data cache, 8 KB in size, with 64-byte cache blocks»

  – 8 КБ / 64 Б = 128 — число блоков (blocks)
  – 128 / 4 = 32 — число наборов (sets), число различных индексов
  – размер блока 64 байта = 2^6, число возможных смещений (offset) в блоке — 64
  – 21 бит — тэг

Упражнение

- **Пример: Pentium 4.**

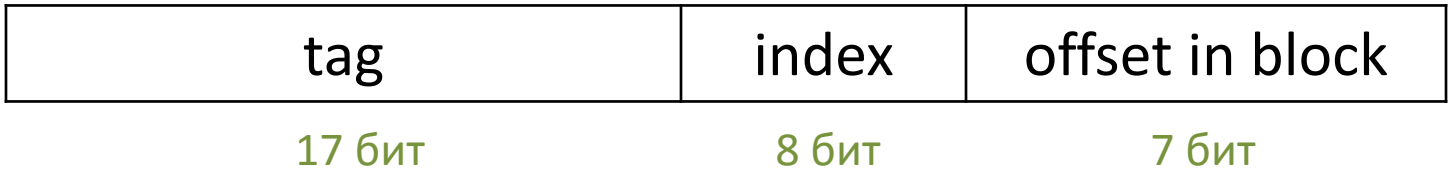
  «eight-way set associative L2 cache — 256 KB in size, with 128-byte cache blocks»

tag	index	offset in block

Упражнение

- **Пример: Pentium 4.**

  «eight-way set associative L2 cache — 256 KB in size, with 128-byte cache blocks»



Критический шаг

- (critical stride) = (number of sets) * (line size) =
  = (total cache size) / (number of ways)

От перестановки мест слагаемых...

```
float a, b, c, d, y;

y = a + b + c + d;

y = (a + b) + (c + d);
```

```
float a = -1.0E8, b = 1.0E8, c = 1.23456, y;
y = a + b + c;

(a + b) + c = 1.23456
a + (b + c) = 0
```

Автоматическая векторизация

```
const int size = 1024;
int a[size], b[size];
// ...
for (int i = 0; i < size; i++) {
    a[i] = b[i] + 2;
}
```

- SSE2: можно считывать сразу четыре элемента из b[], загружать в 128-битный регистр, прибавлять вектор (2,2,2,2)
- Необходимо выравнивание данных по 16 байт.

Задача транспонирования матрицы

```
void transpose(double a[SIZE][SIZE]) {
   int r, c; double temp;
   for (r = 1; r < SIZE; r++) { // loop through rows
        for (c = 0; c < r; c++) { // loop columns below diagonal
               std::swap(a[r][c], a[c][r]); // swap elements
        }
   }
}
void test () {
   __declspec(__align(64)) // align by cache line size
   double matrix[SIZE][SIZE]; // define matrix
   transpose(matrix); // call transpose function
}
```

Задача транспонирования матрицы

- Матрица 64x64 на Pentium 4
- 8 kb = 8192 bytes, 4 ways, line size of 64
- В кеш-линию помещаются 8 double‘ов по 8 байт каждый. Критический шаг 8192 / 4 = 2048 байт = 4 строки матрицы.

Matrix size	Total kilobytes	Time per element
63 x 63	31	11.6
64 x 64	32	16.4
65 x 65	33	11.8
127 x 127	126	12.2
128 x 128	128	17.4
129 x 129	130	14.4
511 x 511	2040	38.7
512 x 512	2048	230.7
513 x 513	2056	38.1