



Лабораторная работа № 4
Измерение объема кеш-памяти L2 центрального процессора.



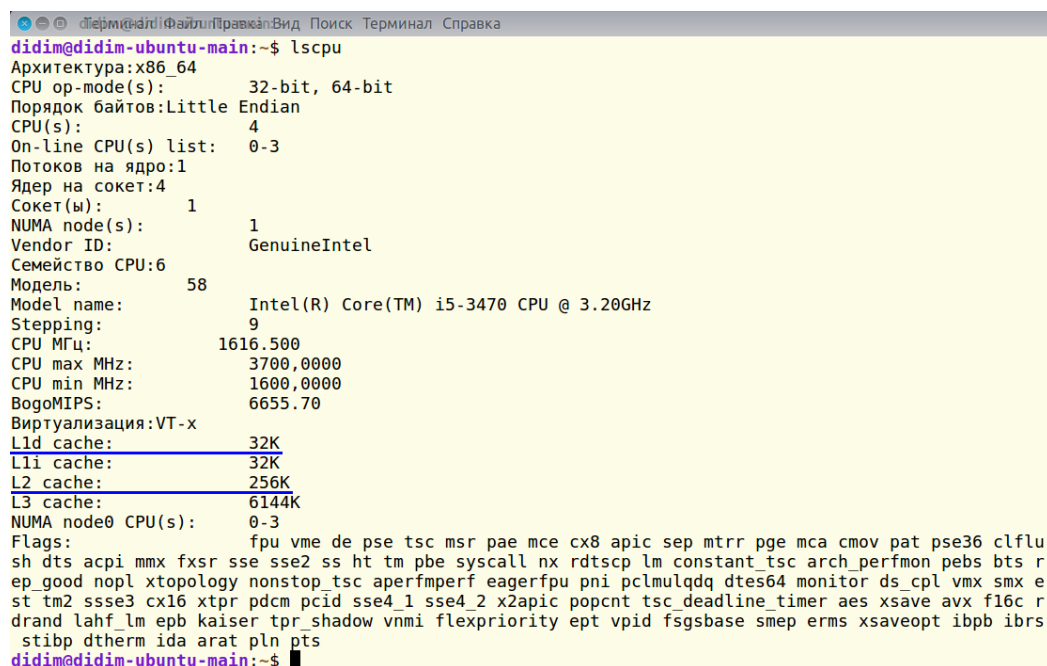
Выполнил:
студент гр. БВТ11
Волков А. А.
Проверил:
преподаватель
Васильев С. А.

Цель работы: определить объем кеш-памяти L2 центрального процессора.

Методика и результаты исследования: для проведения измерения объема кеш-памяти была разработана программа, измеряющая время обхода одномерного массива а также время доступа к каждому элементу массива. Обход массива осуществляется в прямом и обратном направлениях, а также случайным образом (при этом гарантируется однократный обход всех элементов). Размер массива варьируется от 24 kiB (что меньше объема кеша данных L1) до 512 kiB (что больше объема кеша L2, см рис. 1). Таким образом, по изменению времени доступа к одному элементу массива, теоретически, можно определить скорость обращения к кеш-памяти, и соответственно объемы кеш-памяти 1-го и 2-го уровней.

Однако, вследствие особенностей архитектуры современных ЦП, таких как перегруппировка блоков инструкций, предварительное кеширование инструкций и данных а также спекулятивное исполнение кода, предложенный способ измерения объема кеш-памяти оказался неэффективным, поскольку полученные данные не репрезентативны и не отражают реальных скоростных характеристик кеш-памяти.

Ниже представлен листинг кода программы, результаты ее работы (рис. 2), а также графики зависимости времени обхода массива и времени доступа к элементу от размера массива при различном кол-ве итераций обхода массива и различным шаге изменения размера массива (рис. 3-10).



```
didim@didim-ubuntu-main:~$ lscpu
Архитектура:x86_64
CPU op-mode(s):      32-bit, 64-bit
Порядок байтов:Little Endian
CPU(s):              4
On-line CPU(s) list: 0-3
Потоков на ядро:1
Ядер на сокет:4
Сокет(ы):            1
NUMA node(s):        1
Vendor ID:            GenuineIntel
Семейство CPU:6
Модель:              58
Model name:           Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz
Stepping:             9
CPU МГц:              1616.500
CPU max MHz:          3700,0000
CPU min MHz:          1600,0000
BogoMIPS:             6655.70
Виртуализация:VT-x
L1d cache:            32K
L1i cache:            32K
L2 cache:             256K
L3 cache:             6144K
NUMA node0 CPU(s):   0-3
Flags:                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflu
sh dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts r
ep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx e
st tm2 sse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx f16c r
drand lahf_lm epb kaiser tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms xsaveopt ibpb ibrs
stibp dtherm ida arat pln pts
didim@didim-ubuntu-main:~$
```

Рис 1. Характеристики исследуемого процессора

Листинг кода программы:

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <time.h>

#define ITERATIONS 10000
#define BLK_START 48 // 24 kiB
#define BLK_END 1025 // 512 kiB
#define BLK_SIZE 64 // 512 bytes
#define EMPTY_FLAG -1

uint64_t rdtsc (void) {
    uint64_t lo, hi;
    asm volatile ("rdtsc\n" : "=a" (lo), "=d" (hi));
    return (hi << 32) | lo;
}

uint64_t arrayWalk (const uint64_t* array, const uint64_t size) {
    uint64_t steps, index, time, avgTime = 0;

    for (int iter = 0; iter < ITERATIONS; iter++) {
        steps = index = 0;
        time = rdtsc();
        while (steps++ < size)
            index = array[index];
        avgTime += rdtsc() - time;
    }

    return avgTime / ITERATIONS;
}

int main(void) {
    srand(time(0));
    uint64_t timeStd, timeRev, timeRand;

    for (int size = BLK_START; size < BLK_END; size++) {
        const uint64_t currSize = size * BLK_SIZE;
        uint64_t *data = (uint64_t *) malloc(currSize * sizeof(uint64_t));

        data[currSize - 1] = 0;
        for (int i = 0; i < currSize - 1; i++)
            data[i] = i + 1;
        timeStd = arrayWalk(data, currSize);

        data[0] = currSize - 1;
        for (int i = 1; i < currSize; i++)
            data[i] = i - 1;
        timeRev = arrayWalk(data, currSize);

        for (int i = 0; i < currSize; i++)
            data[i] = EMPTY_FLAG;
```

```

int done, index;
for (int next = 0; next < currSize; next++) {
    done = 0;
    while (!done) {
        index = rand() % currSize;
        if (index != next && data[index] == EMPTY_FLAG) {
            data[index] = next;
            done = 1;
        }
    }
}

timeRand = arrayWalk(data, currSize);

double timeStdSingle = (double) timeStd / (double) currSize;
double timeRevSingle = (double) timeRev / (double) currSize;
double timeRandSingle = (double) timeRand / (double) currSize;

printf("%5d | %-8ld | %-8ld | %-8ld | %-6.3f | %-6.3f | %-6.3f\n",
    size, timeStd, timeRev, timeRand,
    timeStdSingle, timeRevSingle, timeRandSingle);

//free(data);
}

return 0;
}

```

```
didim@didim-ubuntu-main:~/bin/c$ ./test
```

48	53786	35673	32102	17.508	11.612	10.450
49	32788	32779	32876	10.455	10.452	10.483
50	33399	33441	33522	10.437	10.450	10.476
51	34110	34072	34062	10.450	10.439	10.436
52	34778	34775	34855	10.450	10.449	10.473
53	35448	35396	35341	10.450	10.435	10.419
54	36124	36115	36162	10.453	10.450	10.464
55	36777	36777	36906	10.448	10.448	10.485
56	37437	37451	37560	10.446	10.449	10.480
57	38153	38104	38445	10.459	10.445	10.539
58	38792	38784	39447	10.450	10.448	10.627
59	39482	39483	39598	10.456	10.456	10.487
60	40126	40118	40322	10.449	10.447	10.501

• • •

1015	679853	680008	1750928	10.466	10.468	26.954
1016	680504	680654	1777124	10.465	10.468	27.330
1017	681182	681364	1776079	10.466	10.468	27.287
1018	681839	681990	1715823	10.465	10.468	26.336
1019	682541	682705	1785309	10.466	10.468	27.375
1020	683196	683327	1758175	10.466	10.468	26.933
1021	683866	684035	1785864	10.466	10.468	27.330
1022	684533	684701	1779808	10.466	10.468	27.211
1023	685237	685371	1791899	10.466	10.468	27.369
1024	685847	686037	1761499	10.465	10.468	26.878

```
didim@didim-ubuntu-main:~/bin/c$ █
```

Рис. 2. Результат работы программы

Рис. 3. Время обхода массива
10 000 итераций, шаг изменения размера: 512 bytes

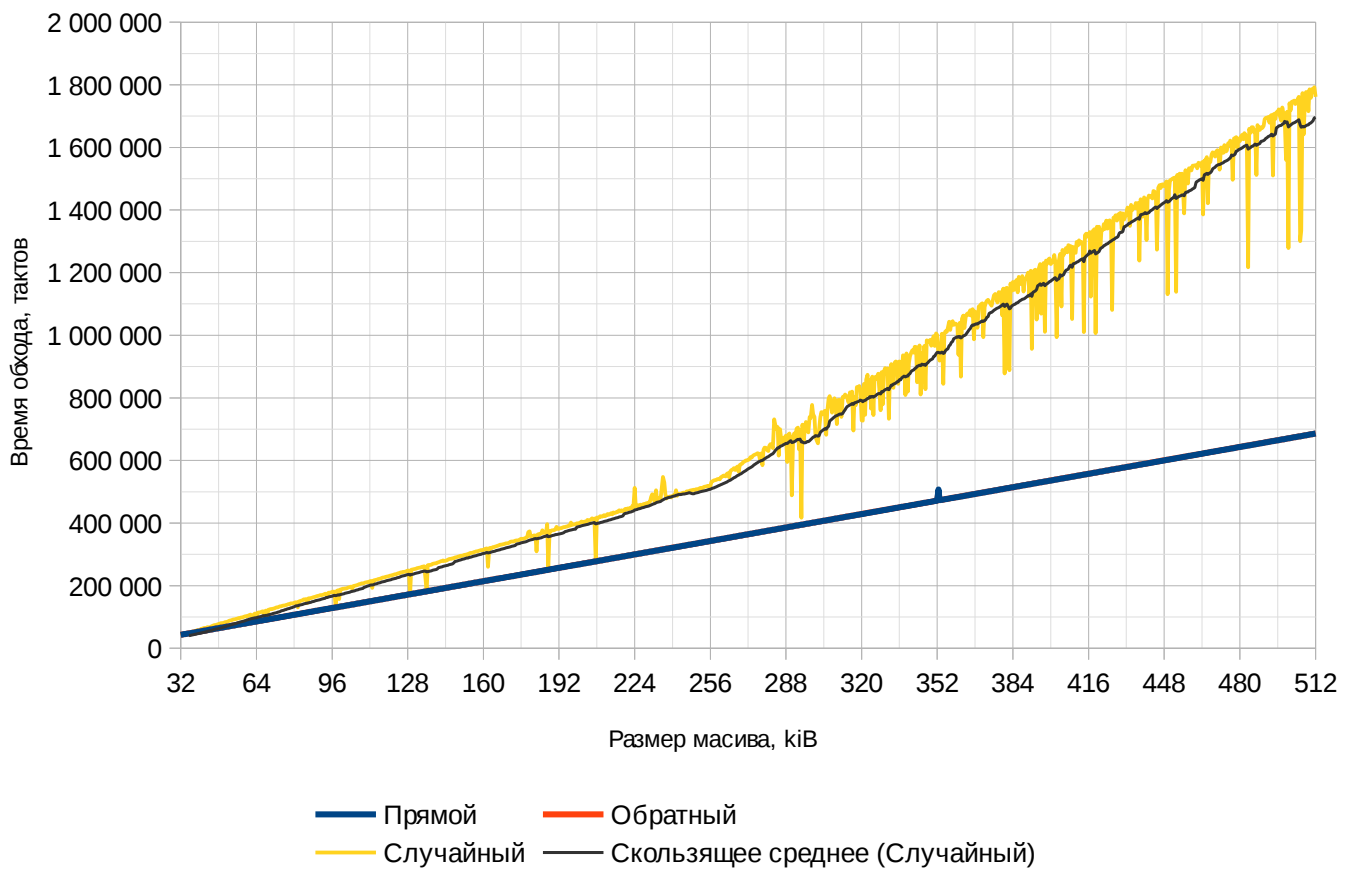


Рис. 4. Время доступа к элементу массива
10 000 итераций, шаг изменения размера: 512 bytes

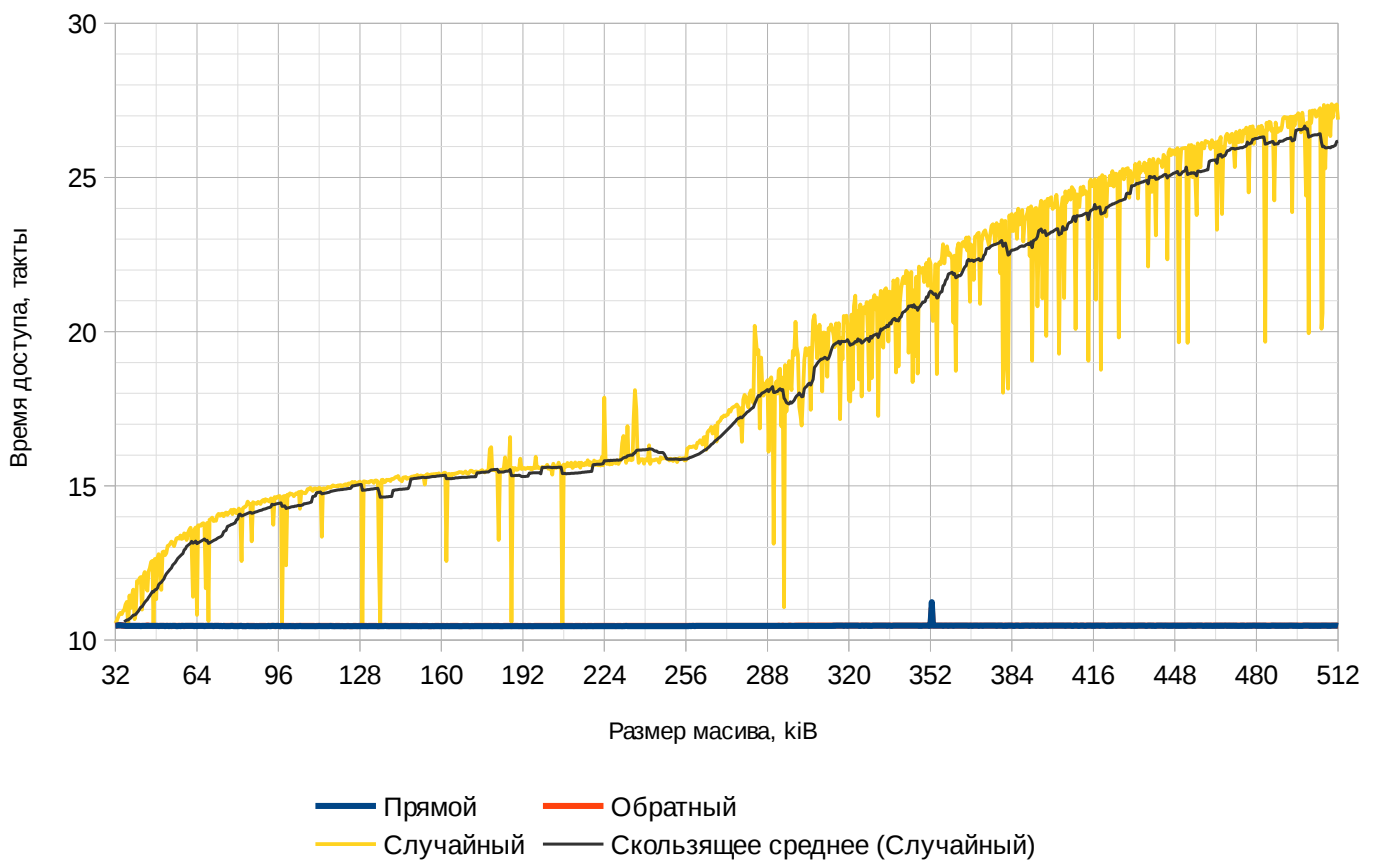


Рис. 5. Время обхода массива
100 000 итераций, шаг изменения размера: 512 bytes

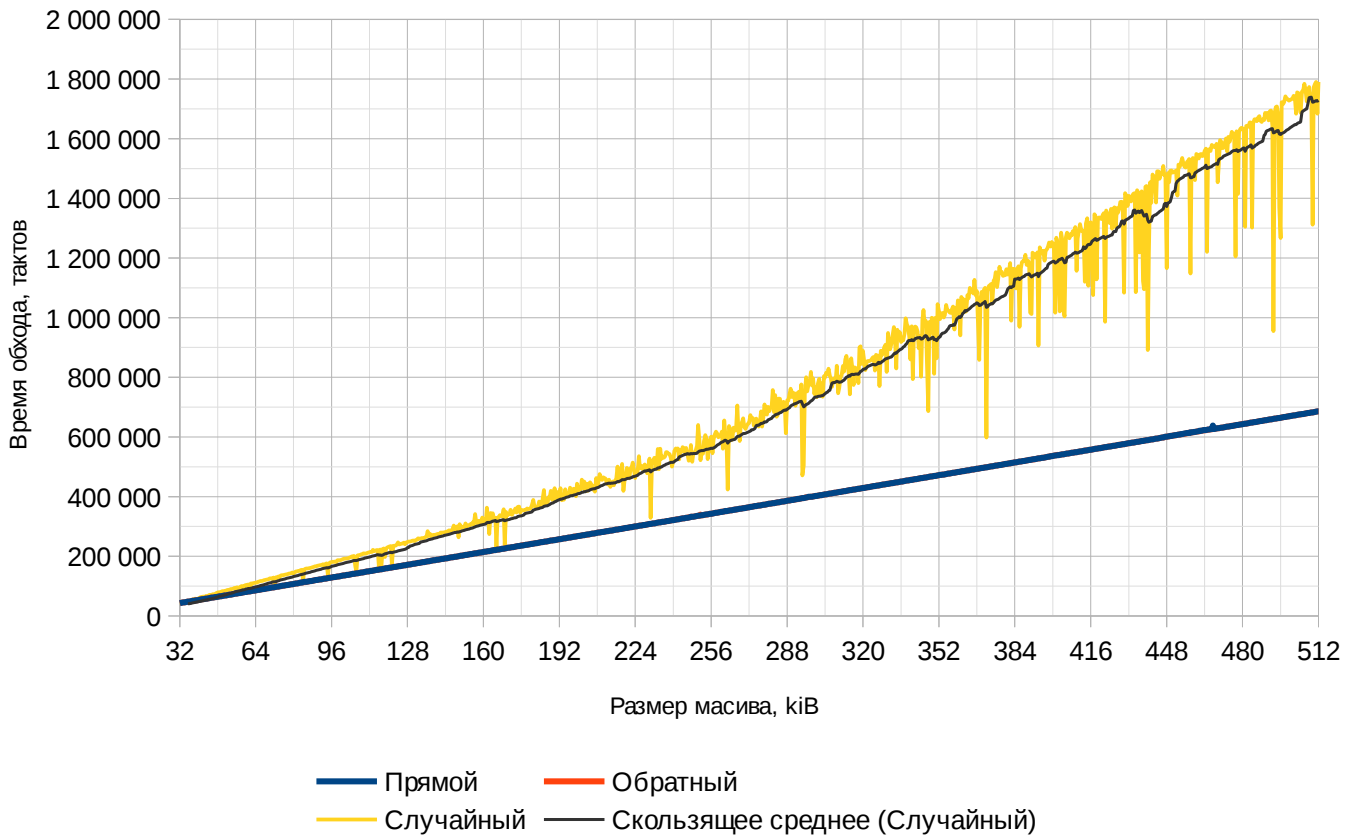


Рис. 6. Время доступа к элементу массива
100 000 итераций, шаг изменения размера: 512 bytes

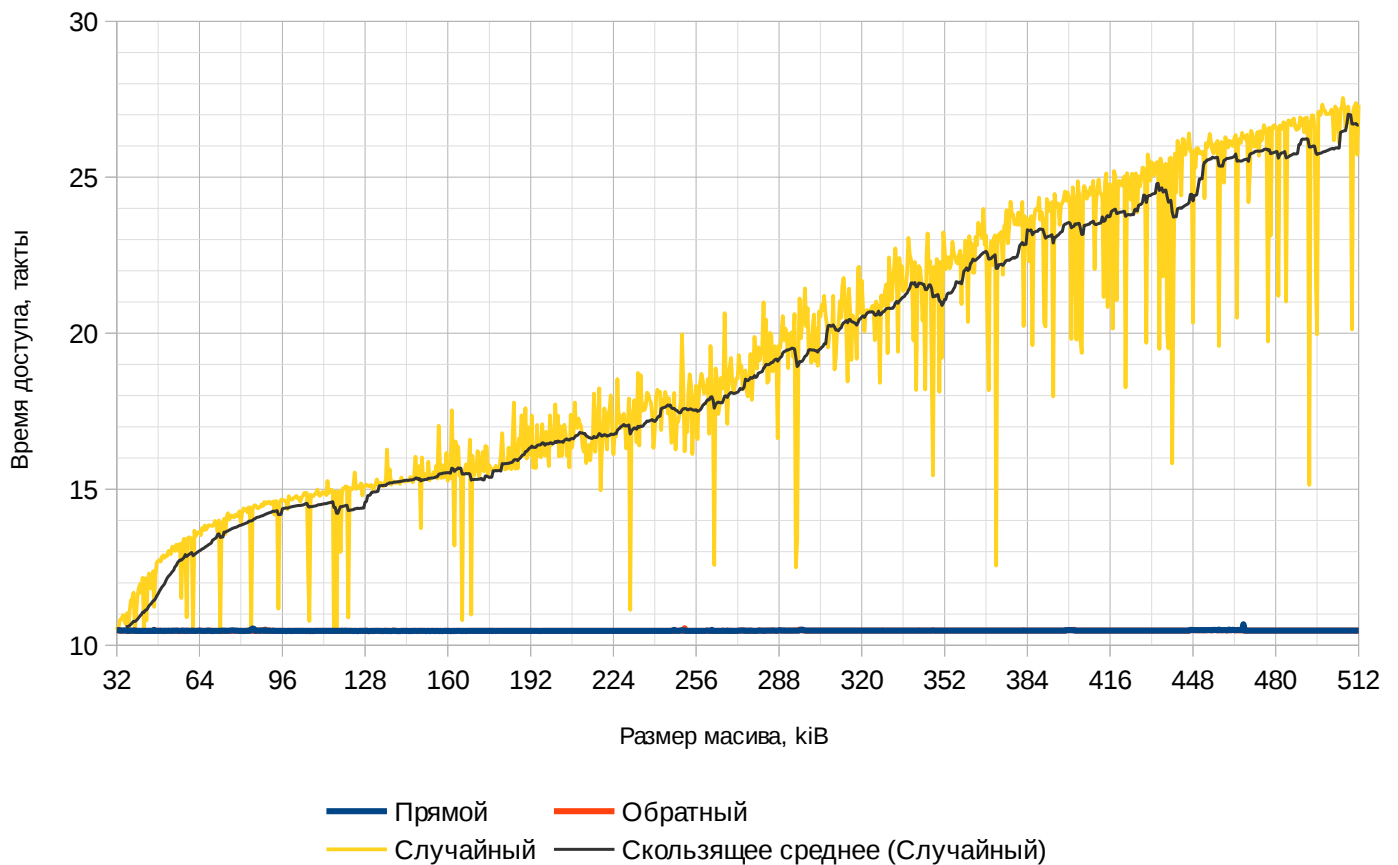


Рис. 7. Время обхода массива
10 000 итераций, шаг изменения размера: 1 kiB

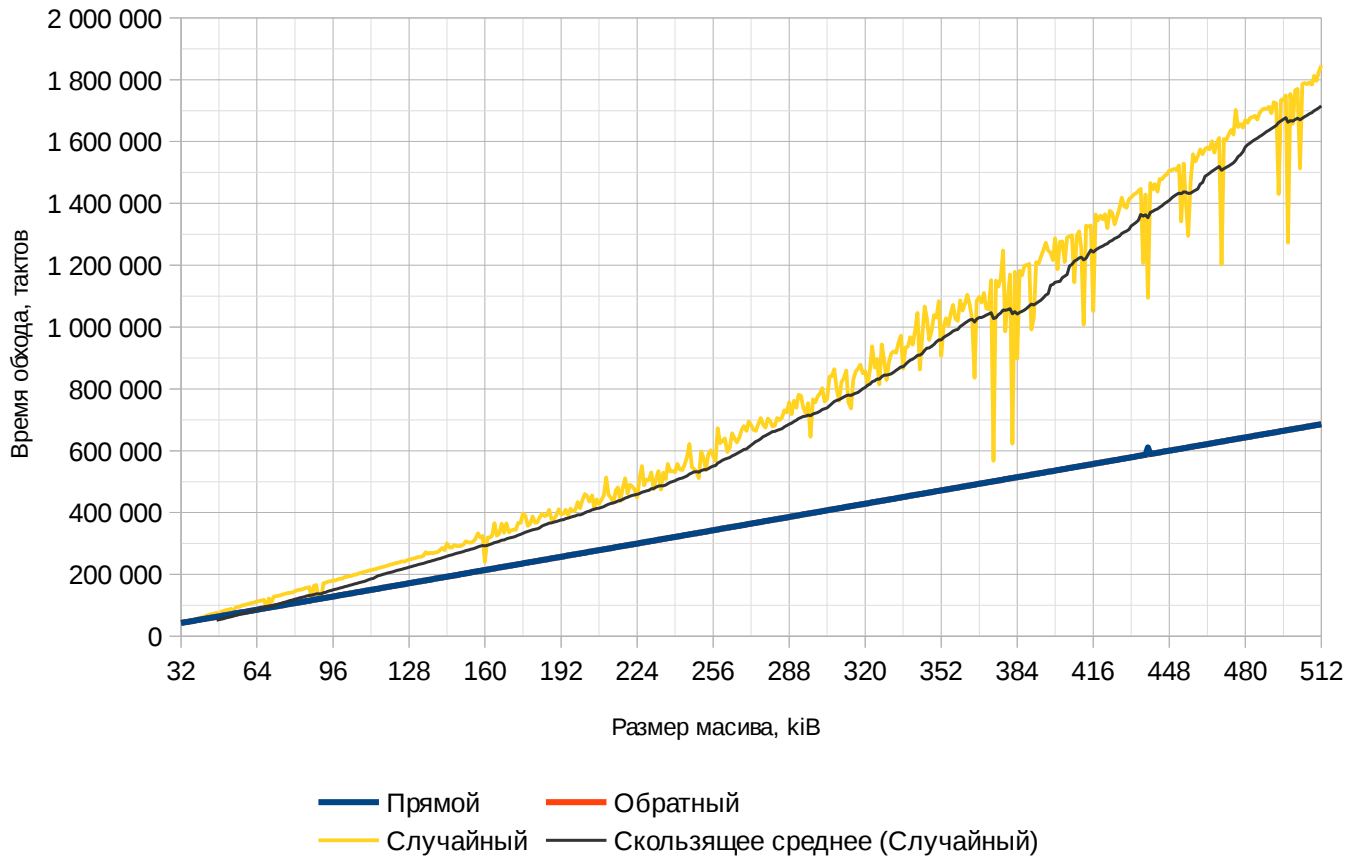


Рис. 8. Время доступа к элементу массива
10 000 итераций, шаг изменения размера: 1 kiB

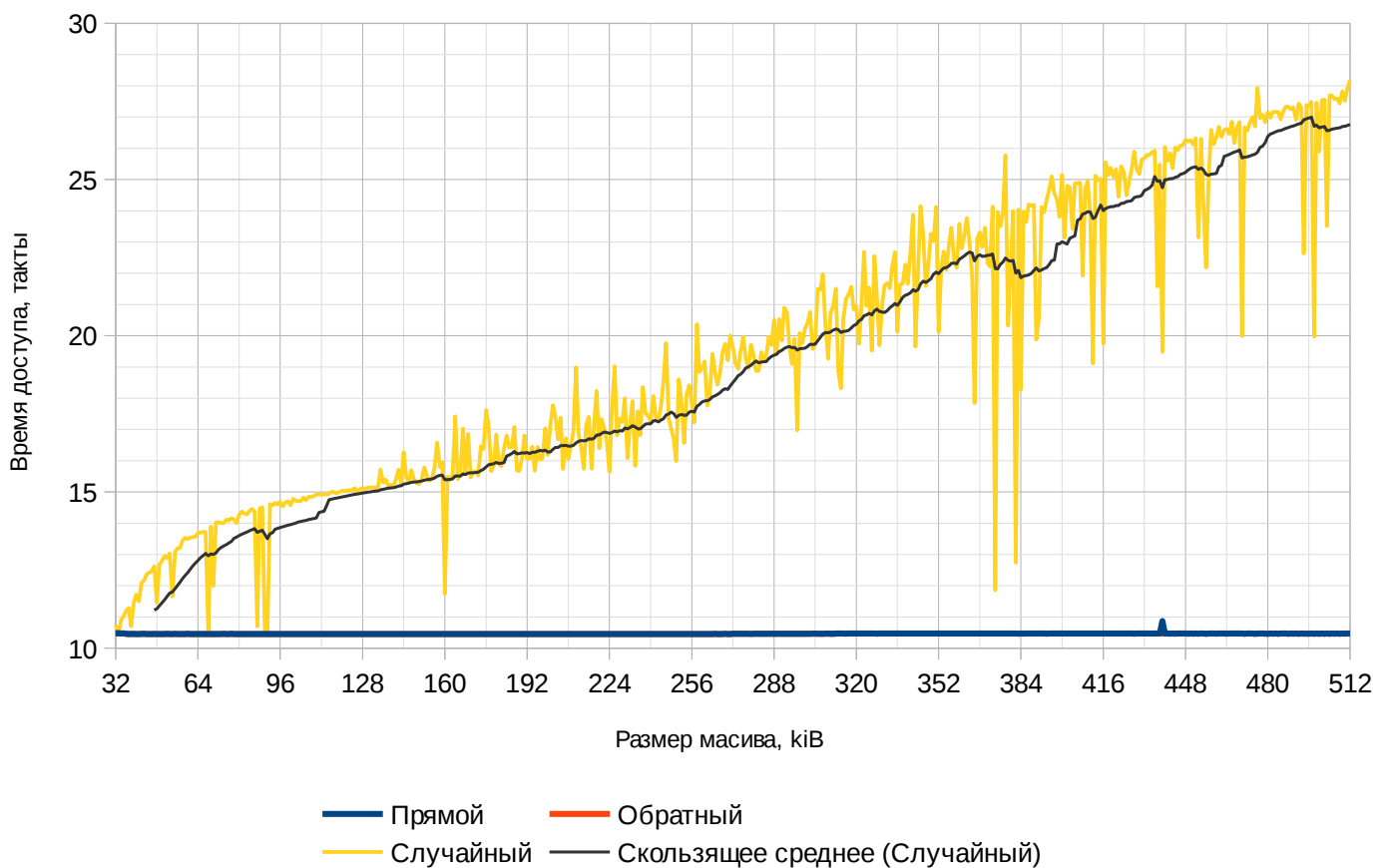


Рис. 9. Время обхода массива
100 000 итераций, шаг изменения размера: 1 kiB

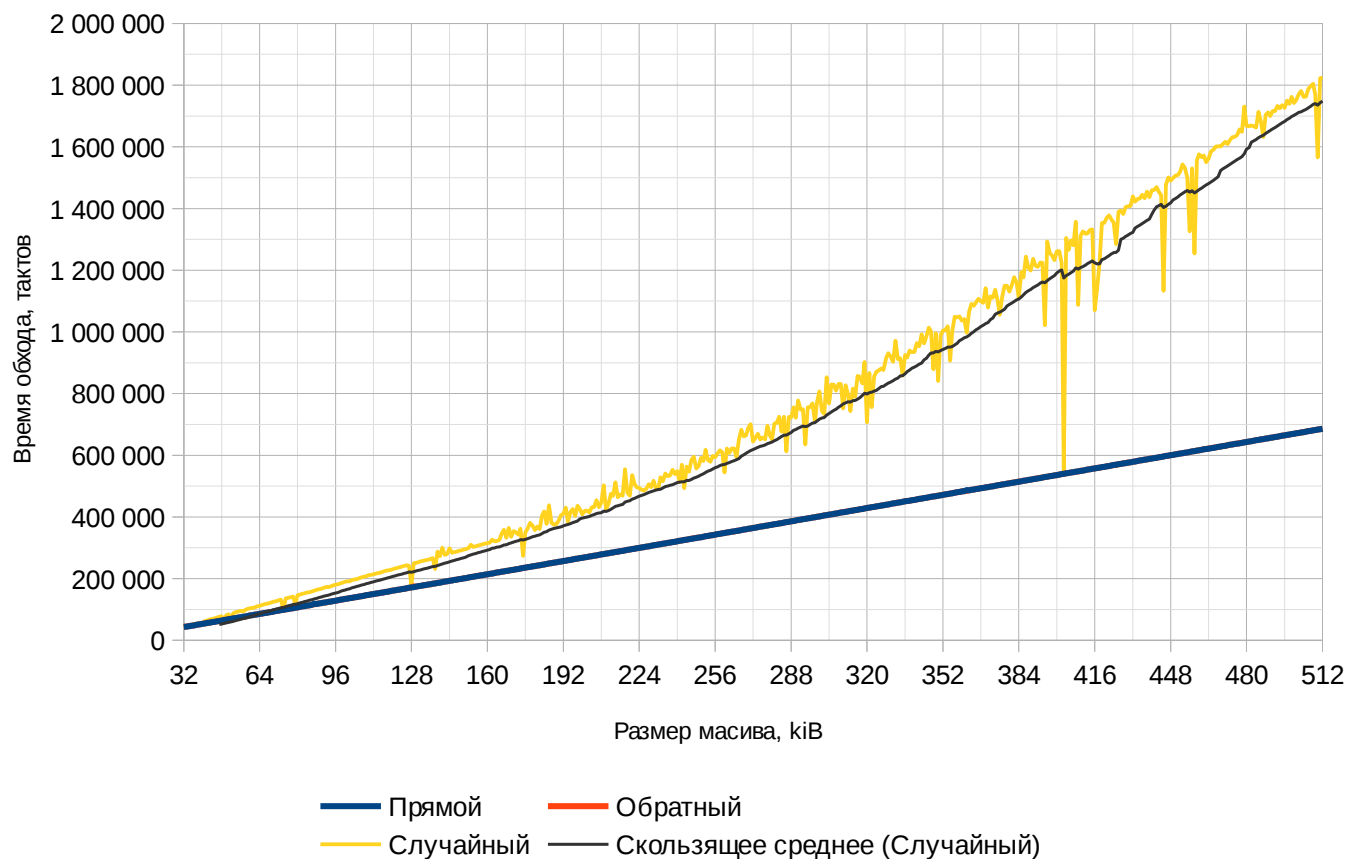


Рис. 10. Время доступа к элементу массива
100 000 итераций, шаг изменения размера: 1 kiB

