

Test the speed of copying raw memory of various utilities.

```
#include <Core/Core.h>

using namespace Upp;

const int N = 10000;
const int M = 3*1024*1024;

struct S{
    S()noexcept=default;
    S(const S&)noexcept=default;
    char buff[M];
};

CONSOLE_APP_MAIN
{
    S s;

    int64 t = 0;
    for(int i=0; i<N; ++i)
    {
        {
            RTIMING("the memory copy utility likely used by std::vector");
            char buff[M];
            new(buff)S(s);
            for(int i=0; i < M; ++i)
                t += buff[i];
        }
        {
            RTIMING("the memory copy utility used by Upp::Vector");
            char buff[M];
            memcpy_t((S*)buff, &s, 1);
            for(int i=0; i < M; ++i)
                t -= buff[i];
        }
        {
            RTIMING("memcpy function");
            char buff[M];
            memcpy(buff, &s, M);
            for(int i=0; i < M; ++i)
                t += buff[i];
        }
    }
}
```

```
    for(int i=0; i<M; ++i)
        t -= s.buff[i];
}

RLOG(t);
}
```

Typical output

0
TIMING memcpy function: 12.25 s - 1.22 ms (12.25 s / 10000), min: 1.00 ms, max: 3.00 ms,
nesting: 0 - 10000
TIMING the memory copy utility used by Upp::Vector: 8.72 s - 871.98 us (8.72 s / 10000), min:
0.00 ns, max: 3.00 ms, nesting: 0 - 10000
TIMING the memory copy utility likely used by std::vector: 11.63 s - 1.16 ms (11.63 s / 10000),
min: 1.00 ms, max: 5.00 ms, nesting: 0 - 10000

It's confirmed Upp::memcpy_t with SIMD optimization is significantly faster. So it has to be
because of memory allocation overhead.
