Хэш таблица

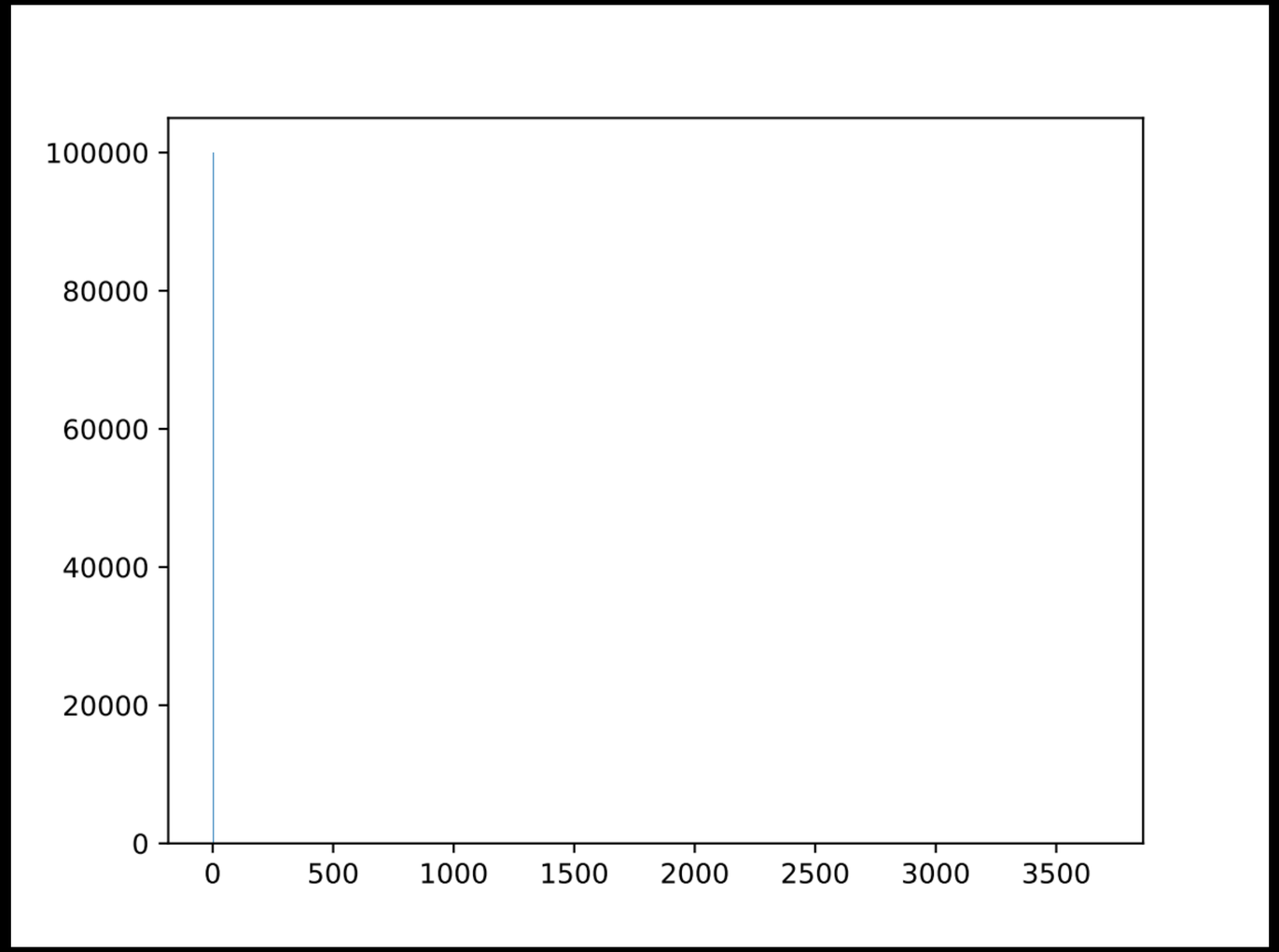
Оптимизация структуры данных с помощью ассемблерной вставки

Ванурин Сергей Максимович

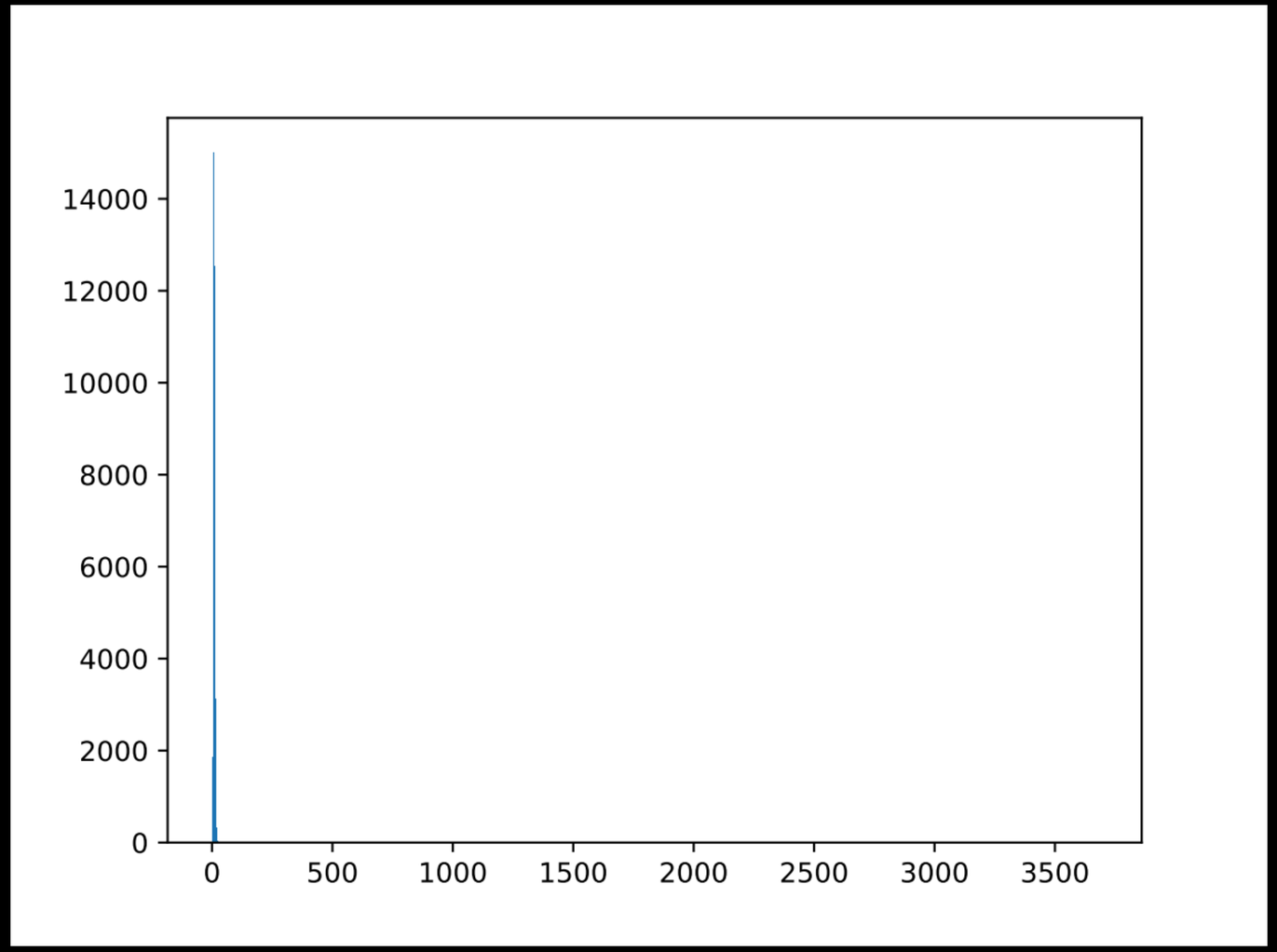
Цели исследования

- Анализ производительности хэш таблицы с различными оптимизациями
- Анализ распределения различных хэш функций
- Сравнение ассемблерных вставок и оптимизаций компилятора

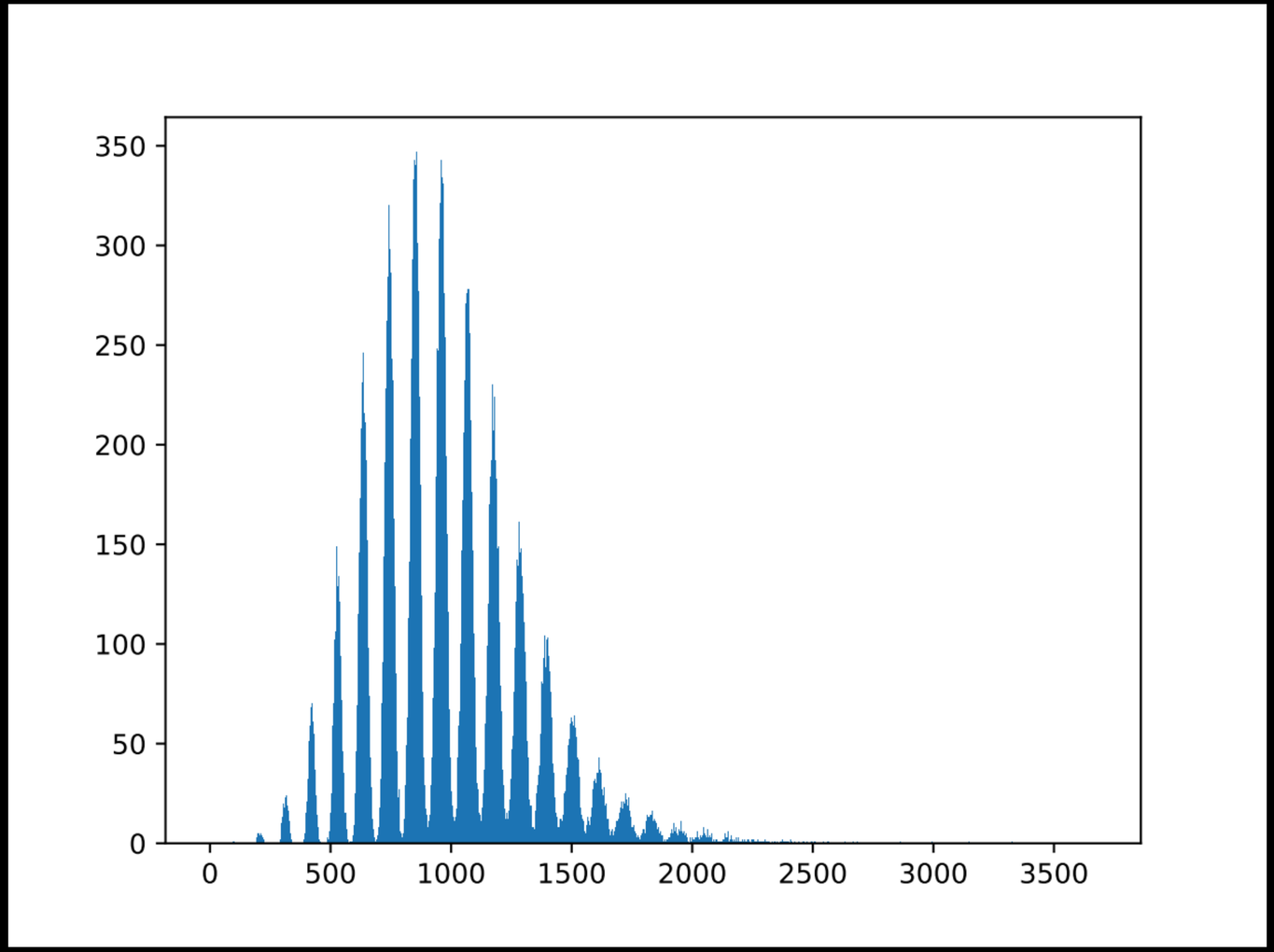
One hash


100000
80000
60000
40000
20000
0
0
500
1000
1500
2000
2500
3000
3500

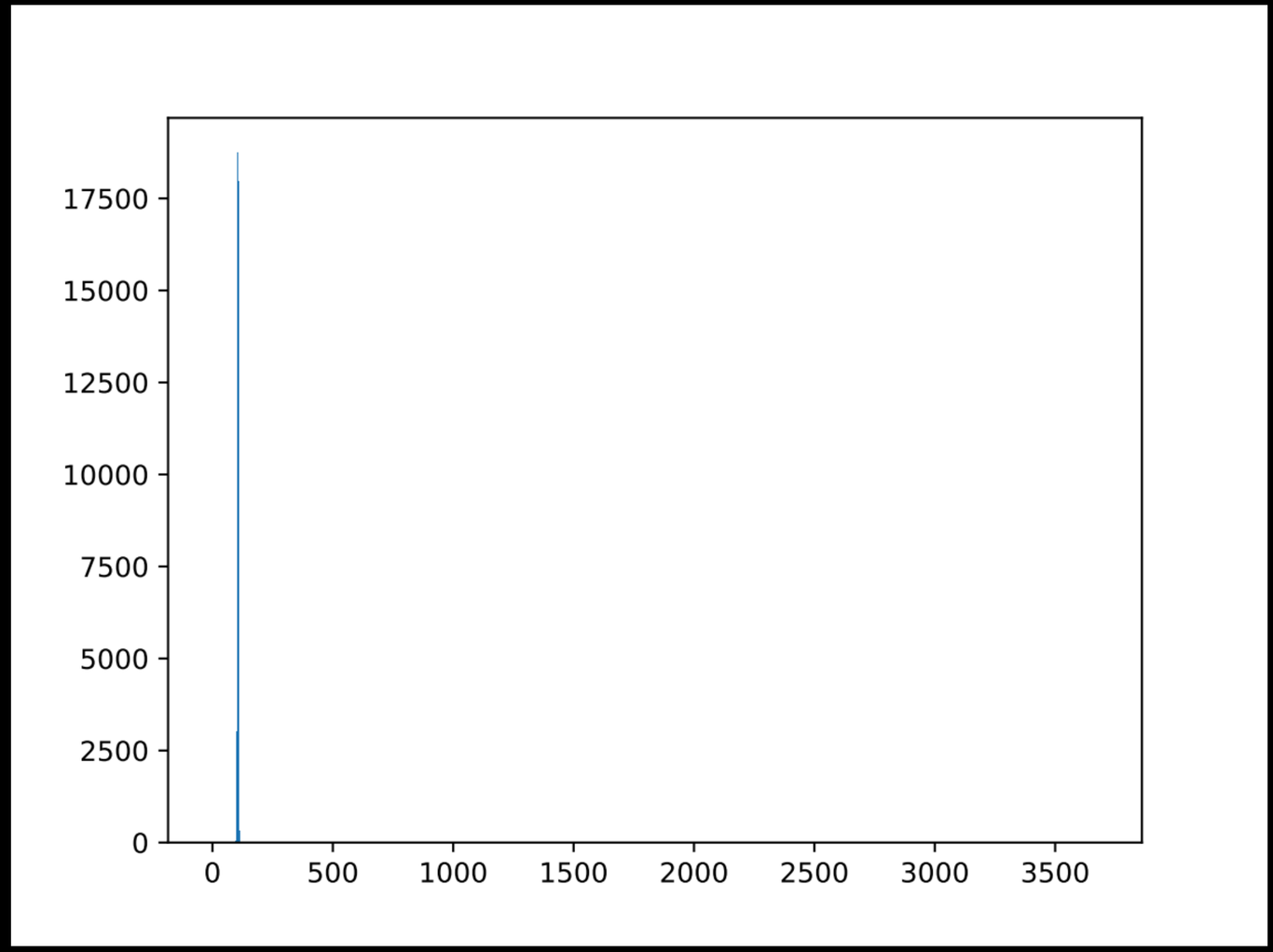
Length hash


14000
12000
10000
8000
6000
4000
2000
0
0
500
1000
1500
2000
2500
3000
3500

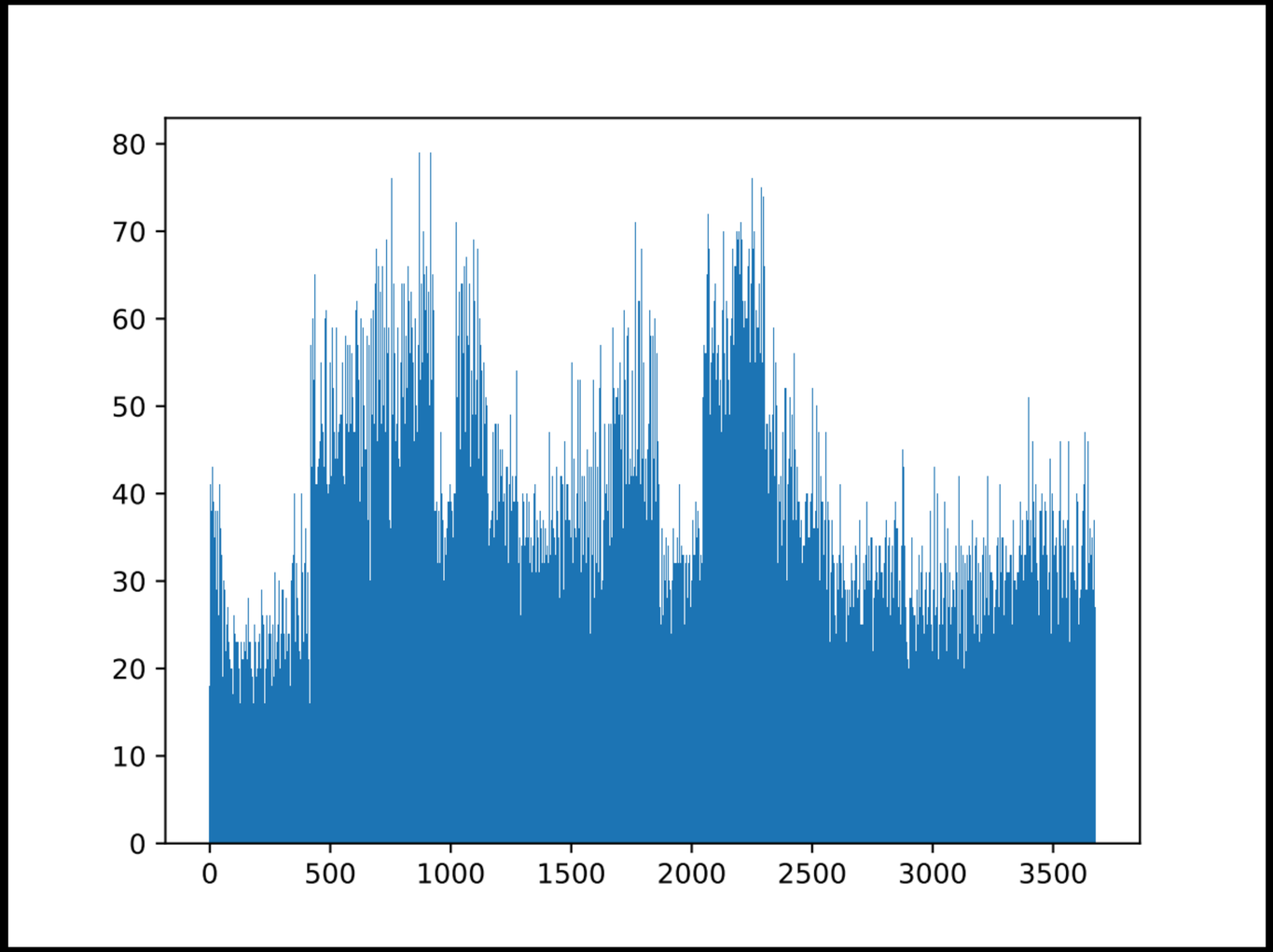
Sum hash



Sum over length hash



Xor hash



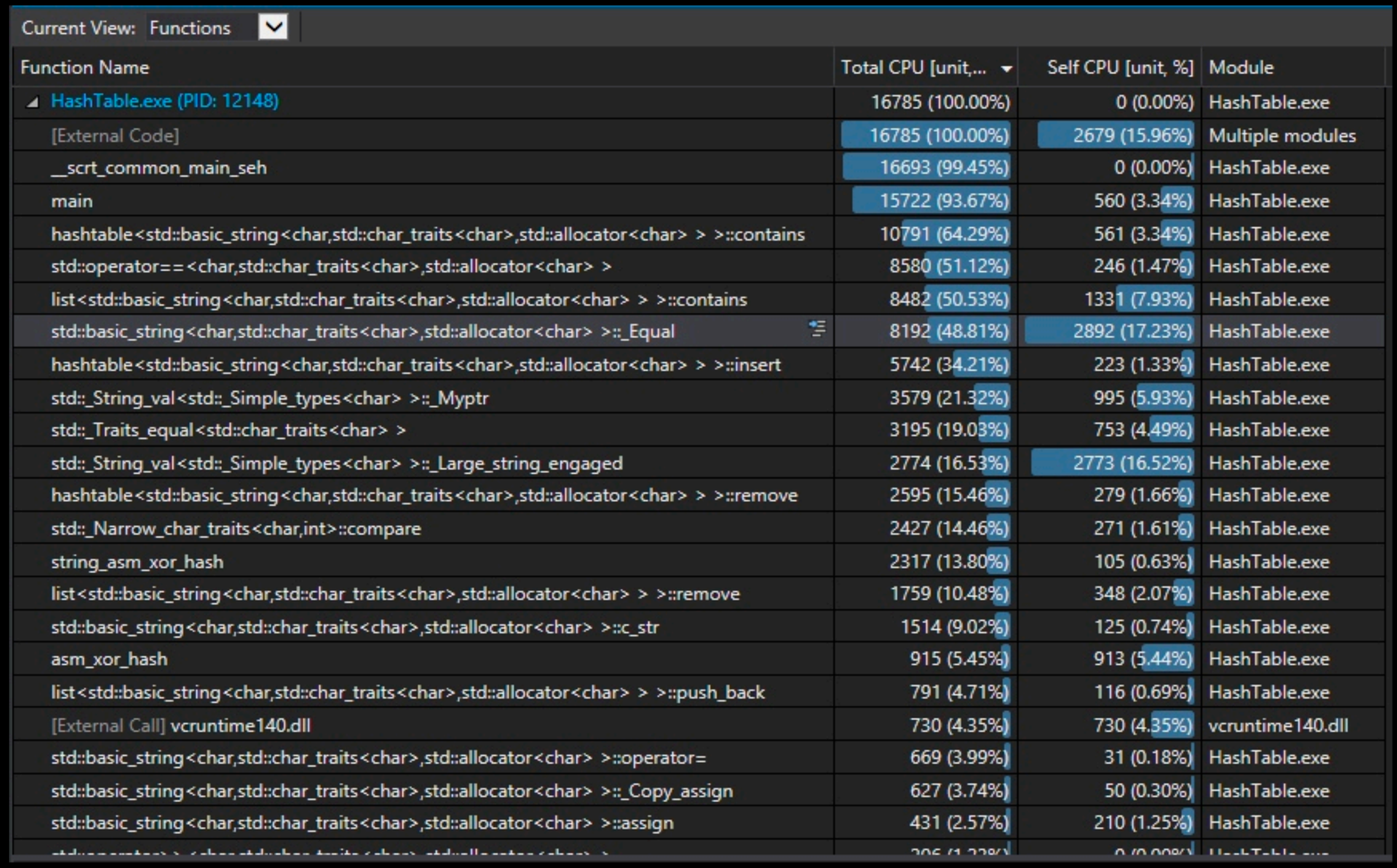
До ассемблерной оптимизации

Можем заметить, что функция хеширования занимает значительную часть времени исполнения программы.

Function Name	Total CPU [unit,...	Self CPU [unit, %]	Module
HashTable.exe (PID: 14152)	21705 (100.00%)	0 (0.00%)	HashTable.exe
[External Code]	21704 (100.00%)	2122 (9.78%)	Multiple modules
__scrt_common_main_seh	21582 (99.43%)	0 (0.00%)	HashTable.exe
main	20653 (95.15%)	587 (2.70%)	HashTable.exe
hashtable<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::contains	13202 (60.82%)	595 (2.74%)	HashTable.exe
std::operator==<char,std::char_traits<char>,std::allocator<char> >	8376 (38.59%)	252 (1.16%)	HashTable.exe
list<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::contains	8305 (38.26%)	1361 (6.27%)	HashTable.exe
hashtable<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::insert	8121 (37.42%)	218 (1.00%)	HashTable.exe
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::_Equal	8002 (36.87%)	2783 (12.82%)	HashTable.exe
xor_hash	7339 (33.81%)	2640 (12.16%)	HashTable.exe
std::_String_val<std::_Simple_types<char> >::_Myptr	4325 (19.93%)	1697 (7.82%)	HashTable.exe
hashtable<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::remove	3996 (18.41%)	268 (1.23%)	HashTable.exe
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::operator[]	3509 (16.17%)	1262 (5.81%)	HashTable.exe
std::_String_val<std::_Simple_types<char> >::_Large_string_engaged	3063 (14.11%)	3062 (14.11%)	HashTable.exe
std::_Traits_equal<std::char_traits<char> >	3052 (14.06%)	731 (3.37%)	HashTable.exe
std::_Narrow_char_traits<char,int>::compare	2294 (10.57%)	274 (1.26%)	HashTable.exe
list<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::remove	1800 (8.29%)	363 (1.67%)	HashTable.exe
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::size	1258 (5.80%)	1255 (5.78%)	HashTable.exe
[External Call] vcruntime140.dll	1140 (5.25%)	1140 (5.25%)	vcruntime140.dll
list<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::push_back	777 (3.58%)	88 (0.41%)	HashTable.exe
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::operator=	682 (3.14%)	38 (0.18%)	HashTable.exe
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::_Copy_assign	625 (2.88%)	53 (0.24%)	HashTable.exe
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::assign	462 (2.13%)	197 (0.91%)	HashTable.exe

После ассемблерной оптимизации

Доля функции хэширования значительно понизилась



Current View: Functions

Function Name	Total CPU [unit,...	Self CPU [unit, %]	Module
HashTable.exe (PID: 12148)	16785 (100.00%)	0 (0.00%)	HashTable.exe
[External Code]	16785 (100.00%)	2679 (15.96%)	Multiple modules
__scrt_common_main_seh	16693 (99.45%)	0 (0.00%)	HashTable.exe
main	15722 (93.67%)	560 (3.34%)	HashTable.exe
hashtable<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::contains	10791 (64.29%)	561 (3.34%)	HashTable.exe
std::operator==<char,std::char_traits<char>,std::allocator<char> >	8580 (51.12%)	246 (1.47%)	HashTable.exe
list<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::contains	8482 (50.53%)	1331 (7.93%)	HashTable.exe
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::_Equal	8192 (48.81%)	2892 (17.23%)	HashTable.exe
hashtable<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::insert	5742 (34.21%)	223 (1.33%)	HashTable.exe
std::_String_val<std::_Simple_types<char> >::_Myptr	3579 (21.32%)	995 (5.93%)	HashTable.exe
std::_Traits_equal<std::char_traits<char> >	3195 (19.03%)	753 (4.49%)	HashTable.exe
std::_String_val<std::_Simple_types<char> >::_Large_string_engaged	2774 (16.53%)	2773 (16.52%)	HashTable.exe
hashtable<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::remove	2595 (15.46%)	279 (1.66%)	HashTable.exe
std::_Narrow_char_traits<char,int>::compare	2427 (14.46%)	271 (1.61%)	HashTable.exe
string_asm_xor_hash	2317 (13.80%)	105 (0.63%)	HashTable.exe
list<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::remove	1759 (10.48%)	348 (2.07%)	HashTable.exe
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::c_str	1514 (9.02%)	125 (0.74%)	HashTable.exe
asm_xor_hash	915 (5.45%)	913 (5.44%)	HashTable.exe
list<std::basic_string<char,std::char_traits<char>,std::allocator<char> > >::push_back	791 (4.71%)	116 (0.69%)	HashTable.exe
[External Call] vcruntime140.dll	730 (4.35%)	730 (4.35%)	vcruntime140.dll
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::operator=	669 (3.99%)	31 (0.18%)	HashTable.exe
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::_Copy_assign	627 (3.74%)	50 (0.30%)	HashTable.exe
std::basic_string<char,std::char_traits<char>,std::allocator<char> >::assign	431 (2.57%)	210 (1.25%)	HashTable.exe

Xor hash

```cpp
unsigned int xor_hash(const std::string& data)
{
    unsigned int sum = 0;
    unsigned int data_length = data.length();

    for (int i = 0; i < data_length; i++)
    {
        sum ^= data[i];
        sum = (sum << 1) | (sum >> 31);
    }

    return sum;
}
```

```asm
.686
.MODEL tiny, c
.code

asm_xor_hash proc data:ptr byte
    xor eax, eax
    xor edx, edx

    mov ecx, dword ptr data

hash_loop:
    mov dl, [ecx]

    cmp edx, 0
    je loop_end

    xor eax, edx
    rol eax, 1
    inc ecx

    jmp hash_loop

loop_end:
    ret

asm_xor_hash endp

end
```

Уровень оптимизации	Без ассемблерной вставки	С ассемблерной вставкой	Коэффициент ускорения
Od	21,72s	16,35s	1,328
O1	10,79s	9,89s	1,09
O2	9,2s	9,25s	0,994

Условия измерений

1) Используемый компилятор: MSVC
2) 17,5 миллионов операций вставки, Поиска и удаления
3) Результатом является среднее значение среди 5 замеров