МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Вычислительная техника»

Дисциплина «Организация ЭВМ и систем»

**Лабораторная работа №3.**

**Исследование функциональных возможностей системы команд микропроцессора**

Выполнил:

студент группы ИВТАПбд-21

Кондратьев П.С.

Проверила:

Лылова А.В.

Ульяновск, 2018

**Цель работы:** более глубокое изучение системы команд микропроцессоров семейства i80x86 с помощью исследования методов оптимизации, компиляторов языка С.

Современные компиляторы C поддерживают множество архитектурных особенностей микропроцессоров семейства i80x86. Для этого существует возможность настройки компилятора на множество режимов оптимизации кода: по размеру, по скорости и т.д. Используя методы оптимизации, компиляторы могут в той или иной мере модифицировать код программы, применяя заложенные в них шаблоны, зависящие от типа используемого микропроцессора.

Таким образом, более глубокое изучение системы команд можно построить на исследовании методов оптимизации.

|  |  |
| --- | --- |
| 12 | Сортировка массива методом Шелла. |

Исходный код:

#include <stdio.h>

void ShellSort(int n, int mass[]) {

int i, j, step;

int tmp;

for (step = n / 2; step > 0; step /= 2)

for (i = step; i < n; i++)

{

tmp = mass[i];

for (j = i; j >= step; j -= step)

{

if (tmp < mass[j - step])

mass[j] = mass[j - step];

else

break;

}

mass[j] = tmp;

}

}

int main() {

int n = 5;

int mass[5] = {3, 1, 5, 2, 4};

ShellSort(n, mass);

printf("Sorted array:\n");

for (int i = 0; i < n; i++)

printf("%d ", mass[i]);

printf("\n");

return 0;

}

Сравнительная таблица оптимизированного assembler кода:

|  |  |  |
| --- | --- | --- |
| c/c++ | Assembler (по скорости) | Assembler (по размеру кода) |
| #include <stdio.h> | #include <stdio.h> | #include <stdio.h> |
| int main() { | 00B81510 push ebp  00B81511 mov ebp,esp  00B81513 sub esp,24h  00B81516 mov eax,dword ptr [\_\_security\_cookie (0B87000h)]  00B8151B xor eax,ebp  00B8151D mov dword ptr [ebp-4],eax | 00961510 push ebp  00961511 mov ebp,esp  00961513 sub esp,24h  00961516 mov eax,dword ptr [\_\_security\_cookie (0967000h)]  0096151B xor eax,ebp  0096151D mov dword ptr [ebp-4],eax |
| int n = 5;  int mass[5] = {3, 1, 5, 2, 4}; | int n = 5;  int mass[5] = {3, 1, 5, 2, 4}; | int n = 5;  int mass[5] = {3, 1, 5, 2, 4}; |
| int i, j, step;  int tmp; | 00B81520 movaps xmm0,xmmword ptr [\_\_xmm@00000002000000050000000100000003 (0B85B30h)]  int i, j, step;  int tmp; | 00961520 movaps xmm0,xmmword ptr [\_\_xmm@00000002000000050000000100000003 (0965B30h)]  int i, j, step;  int tmp; |
| c/c++ | Assembler (по скорости) | Assembler (по размеру кода) |
| for (step = n / 2; step > 0; step /= 2) | for (step = n / 2; step > 0; step /= 2)  00B81527 mov edx,2  00B8152C push ebx  00B8152D push esi  00B8152E movups xmmword ptr [mass],xmm0  00B81532 mov dword ptr [ebp-8],4  00B81539 push edi  00B8153A nop word ptr [eax+eax] | for (step = n / 2; step > 0; step /= 2)  00961527 mov edx,2  0096152C push ebx  0096152D push esi  0096152E movups xmmword ptr [mass],xmm0  00961532 mov dword ptr [ebp-8],4  00961539 push edi  0096153A nop word ptr [eax+eax] |
| for (i = step; i < n; i++) { | for (i = step; i < n; i++) {  00B81540 mov ebx,edx  00B81542 mov dword ptr [i],ebx  00B81545 cmp edx,5  00B81548 jge main+87h (0B81597h)  00B8154A lea esi,[mass]  00B8154D mov dword ptr [ebp-24h],esi | for (i = step; i < n; i++) {  00961540 mov ebx,edx  00961542 mov dword ptr [i],ebx  00961545 cmp edx,5  00961548 jge main+87h (0961597h)  0096154A lea esi,[mass]  0096154D mov dword ptr [ebp-24h],esi |
| tmp = mass[i]; | tmp = mass[i];  00B81550 mov eax,dword ptr mass[ebx\*4]  00B81554 mov dword ptr [tmp],eax | tmp = mass[i];  00961550 mov eax,dword ptr mass[ebx\*4]  00961554 mov dword ptr [tmp],eax |
| for (j = i; j >= step; j -= step) { | for (j = i; j >= step; j -= step) {  00B81557 mov eax,ebx  00B81559 cmp ebx,edx  00B8155B jl main+71h (0B81581h) | for (j = i; j >= step; j -= step) {  00961557 mov eax,ebx  00961559 cmp ebx,edx  0096155B jl main+71h (0961581h) |
| tmp = mass[i]; | tmp = mass[i];  00B8155D lea ebx,[edx\*4] | tmp = mass[i];  0096155D lea ebx,[edx\*4] |
| if (tmp < mass[j - step]) | if (tmp < mass[j - step])  00B81564 mov edi,dword ptr [esi]  00B81566 mov ecx,eax  00B81568 sub ecx,edx  00B8156A cmp dword ptr [tmp],edi  00B8156D jge main+6Bh (0B8157Bh) | if (tmp < mass[j - step])  00961564 mov edi,dword ptr [esi]  00961566 mov ecx,eax  00961568 sub ecx,edx  0096156A cmp dword ptr [tmp],edi  0096156D jge main+6Bh (096157Bh) |
| mass[j] = mass[j - step]; | mass[j] = mass[j - step];  00B8156F mov dword ptr mass[eax\*4],edi  00B81573 sub esi,ebx  00B81575 mov eax,ecx  00B81577 cmp eax,edx  00B81579 jge main+54h (0B81564h)  00B8157B mov ebx,dword ptr [i]  00B8157E mov esi,dword ptr [ebp-24h] | mass[j] = mass[j - step];  0096156F mov dword ptr mass[eax\*4],edi  00961573 sub esi,ebx  00961575 mov eax,ecx  00961577 cmp eax,edx  00961579 jge main+54h (0961564h)  0096157B mov ebx,dword ptr [i]  0096157E mov esi,dword ptr [ebp-24h] |
| else | else  break;  } | else  break;  } |
| mass[j] = tmp; | mass[j] = tmp;  00B81581 mov ecx,dword ptr [tmp]  00B81584 inc ebx  00B81585 add esi,4  00B81588 mov dword ptr mass[eax\*4],ecx  00B8158C mov dword ptr [i],ebx  00B8158F mov dword ptr [ebp-24h],esi  00B81592 cmp ebx,5  00B81595 jl main+40h (0B81550h) | mass[j] = tmp;  00961581 mov ecx,dword ptr [tmp]  00961584 inc ebx  00961585 add esi,4  00961588 mov dword ptr mass[eax\*4],ecx  0096158C mov dword ptr [i],ebx  0096158F mov dword ptr [ebp-24h],esi  00961592 cmp ebx,5  00961595 jl main+40h (0961550h) |
| for (step = n / 2; step > 0; step /= 2)  }  return 0;  } | int i, j, step;  int tmp;  for (step = n / 2; step > 0; step /= 2)  00B81597 mov eax,edx  00B81599 cdq  00B8159A sub eax,edx  00B8159C mov edx,eax  00B8159E sar edx,1  00B815A0 test edx,edx  00B815A2 jg main+30h (0B81540h)  }  return 0;  } | int i, j, step;  int tmp;  for (step = n / 2; step > 0; step /= 2)  00961597 mov eax,edx  00961599 cdq  0096159A sub eax,edx  0096159C mov edx,eax  0096159E sar edx,1  009615A0 test edx,edx  009615A2 jg main+30h (0961540h)  }  return 0;  } |
| } | 00B815A4 mov ecx,dword ptr [ebp-4]  00B815A7 xor eax,eax  00B815A9 pop edi  00B815AA pop esi  00B815AB xor ecx,ebp  00B815AD pop ebx  00B815AE call @\_\_security\_check\_cookie@4 (0B811CCh)  00B815B3 mov esp,ebp  00B815B5 pop ebp  00B815B6 ret | 009615A4 mov ecx,dword ptr [ebp-4]  009615A7 xor eax,eax  009615A9 pop edi  009615AA pop esi  009615AB xor ecx,ebp  009615AD pop ebx  009615AE call @\_\_security\_check\_cookie@4 (09611CCh)  009615B3 mov esp,ebp  009615B5 pop ebp  009615B6 ret |

Проанализируем эти таблицу. Проведя оптимизацию с/с++ кода по скорости, памяти, Assembler код не изменился. Это показывает то что исходный код и так оптимизирован. Из-за того, что в исходном коде были сделаны асимптотические методы сжатия, оптимизация кода:

1. Реализация сортировки Шелла была вынесена из метода класса, в тело main
2. Работа с массивом стала константой (массив задается в main), нежели считывание из консоли
3. Убран вывод промежуточных данных (вывод исходного массива, отсортированного массива)
4. В реализация данного программного продукта была использована всего 1 подключенная библиотека
5. Строки pop, push (перед закрывающей скобкой программы) ожно заменить на leave. В первом случае восстанавливаются регистры edi, esi, однобайтовый leave заменяет 3 байта mov esp,ebp /pop ebp

**Вывод:**

В ходе лабораторной работы, были рассмотрены способы оптимизации кода по скорости и по памяти. Для оптимизации стоит по возможности сокращать количество операций, использовать специализированные для процессора операции или более короткие по размеру.

# Используемые ресурсы

1. how does push and pop work in assembly: [Электронный ресурс] URL: https://stackoverflow.com/questions/26026174/how-does-push-and-pop-work-in-assembly (дата обращения: 12.05.2018).
2. Команда LEAVE: [Электронный ресурс] URL: <http://www.vuzllib.su/books/6257-Assembler/175> (дата обращения: 12.05.2018).
3. Лабораторная работа Лылова А.В. «Организация ЭВМ и систем»