МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Вычислительная техника»

Дисциплина «Организация ЭВМ и систем»

**Лабораторная работа №4**

**«Применение методов оптимизации ассемблерного кода, полученного компиляторами из Си-программ»**

**Вариант 12**

Выполнил:

студент группы ИВТАПбд-31

Кондратьев П.С.

Проверила:

Лылова А. В.

Ульяновск, 2018

**Цель работы:** научиться основам построения программ на языке Assembler для архитектуры Intel.

**Порядок работы:**

1. **Запрограммировать** на языке С (или найти готовую реализацию) алгоритм, предложенный в соответствии с вариантом задания.
2. **Откомпилировать полученную программу** в любом доступном компиляторе, позволяющем получить исходный код программы на языке Assembler, без оптимизации.
3. **Откомпилировать полученную программу** в любом доступном компиляторе, позволяющем получить исходный код программы на языке Assembler, с оптимизацией кода по скорости.
4. **Проанализировать** исходный код на языке Assembler, полученный в п.2, и **произвести анализ**, результатом которого будет ответ на вопрос, чем отличается ассемблерный код программы, полученный компилятором от кода, написанного вручную программистом.
5. На основе анализа полученного кода на языке Assembler, **сделать правки данного ассемблерного кода** таким образом, чтобы в нем не прослеживались так называемые «следы» С компиляции.

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

Исходный код:

#include <stdio.h>

int main() {

int n = 5;

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

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;

}

}

3. Ассемблированный код и ассемблированный код с оптимизацией

|  |  |
| --- | --- |
| c/c++ | Assembler (по скорости) |
| 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 |
| 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; |
| c/c++ | Assembler (по скорости) |
| 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 (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 |
| tmp = mass[i]; | tmp = mass[i];  00B81550 mov eax,dword ptr mass[ebx\*4]  00B81554 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) |
| tmp = mass[i]; | tmp = mass[i];  00B8155D 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) |
| 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] |
| else | 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) |
| 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;  } |
| } | 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 |

4. Ассемблированный код и измененный программный код

В дисассемблированном коде программы, написанной на языке С присутствуют сторонние библиотеки, которые используются для объявления сегмента кода, инициализации переменных, вывода в консоль их нужно убрать.

|  |  |
| --- | --- |
| Исходный код | Без «следов» С |
| 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 | int main() {  Global start |
| int n = 5;  int mass[5] = {3, 1, 5, 2, 4}; | .data  n db 5  mas dd 3, 1, 5, 2, 4 |
| 00B81520 movaps xmm0,xmmword ptr (0B85B30h)]  int i, j, step;  int tmp; | i dv 0  j dv 0  step dv 0  tmp dv 0 |
| 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] | .code  start:  //for (step = n / 2; step > 0; step /= 2)  mov edx,2  push ebx  push esi  movups [mass],xmm0  mov [ebp-8],4  push edi  nop [eax+eax] |
| 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++) {  w:mov ebx,edx  mov i,ebx  cmp edx,5  jge k  lea esi,[mass]  mov [ebp-24h],esi |
| tmp = mass[i];  00B81550 mov eax,dword ptr mass[ebx\*4]  00B81554 mov dword ptr [tmp],eax | J:mov eax mass[ebx\*4]  mov [tmp],eax |
| 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) {  q:mov eax,ebx  cmp ebx,edx  jl g |
| tmp = mass[i];  00B8155D lea ebx,[edx\*4] | //tmp = mass[i];  lea ebx,[edx\*4] |
| 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])  l:mov edi, [esi]  mov ecx,eax  sub ecx,edx  cmp tmp,edi  jge q |
| 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];  mov mass[eax\*4],edi  sub esi,ebx  mov eax,ecx  cmp eax,edx  jge l  mov ebx, i  mov esi, [ebp-24h] |
| else  break;} | //else  // break;} |
| 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;  g:mov ecx, tmp  inc ebx  add esi,4  mov mass[eax\*4],ecx  mov i,ebx  mov [ebp-24h],esi  cmp ebx,5  jl j |
| Исходный код | Без «следов» С |
| 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;  } | XOR i i  XOR j j  XOR step step  XOR tmp tmp  //for (step = n / 2; step > 0; step /= 2)  k:=mov eax,edx  cdq  sub eax,edx  mov edx,eax  sar edx,1  test edx,edx  jg w  }  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 | ret  end start |

Вывод:

Были изучены основы построения программ на языке Assembler микропроцессоров семейства i80x60 архитектуры Intel, при изменении дисассемблированного кода.

Библиографический список

1. Asmworld: программирование на ассемблере — Режим доступа: http://asmworld.ru/ (Дата обращения 29.10.18)

2. Курс: организация ЭВМ и систем — Режим доступа: http://eos.ulstu.ru/course/view.php?id=905 (Дата обращения 29.10.18)

3. Основы ассемблера — Режим доступа: http://www.codenet.ru/progr/asm/newbee/lesson3.php (Дата обращения 29.10.18)