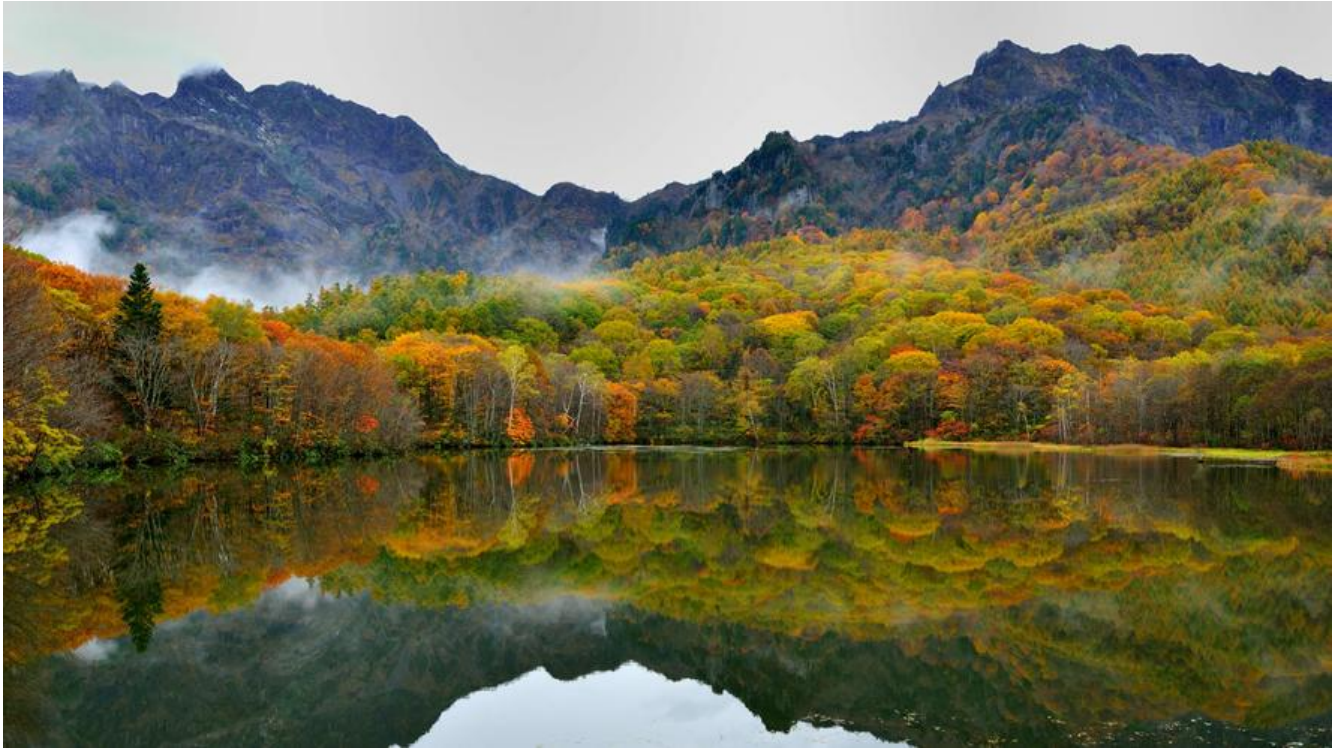
链滴

（转载）编译器能帮我们把代码优化到什么程度？

作者：zhaozhizheng
原文链接：https://ld246.com/article/1614579605550
来源网站：链滴



转载

TODO: 用while写法的程序会不会循环展开?

一个简单的累加求和程序:

```
TYPE S=0;
for(int i = 0;i < SIZE; i++) {
    S += a[i];
}
```

很多人都觉得这个程序写得不好，编译器不能生成很好的汇编代码。于是有了以下的几种“优化”:

```
#include <iostream>
using namespace std;

void main(int argc,char **argv)
{
#define TYPE int
#define SIZE 10000

    TYPE* a=new TYPE[SIZE];
    for(int i = 0; i<SIZE; ++i){
        a[i] = i;
    }
    //求和，通常版本
    TYPE S=0;
    for(int i = 0;i < SIZE; i++) {
        S += a[i];
    }
    cout<<S<<endl;
```

原文链接: （转载）编译器能帮我们把代码优化到什么程度?

```
    TYPE S2 = 0;
    //版本1：认为中间产生的变量i是多余的，改为用移动指针
    TYPE* end = a + SIZE;
    for( ; a != end; ) {
        S2 += *(a++);
    }
    cout<<S2<<endl;

    //版本1中把a移到了数组的最后，现在移回原来的位置
    a = end - SIZE;

    //版本2：认为循环次数太多了，可以改为减少循环次数
    TYPE S3 = 0;
    for(int i = 0; i < SIZE; ){ //仅当SIZE为偶数时
        S3 += a[i++];
        S3 += a[i++];
    }
    cout<<S3<<endl;

    //版本3：认为版本2中会使CPU不能乱序执行，降低了效率，应该改为汇编，把中间结果放到独立寄存器中
    //谢谢 menzi11 的文章，让我认识到程序中相关的数据会让CPU不能乱序执行。
    //这里用伪汇编代替
    TYPE S4 = 0;

    register TYPE r1 = 0;
    register TYPE r2 = 0;
    for(int i = 0; i < SIZE; ){ //仅当SIZE为偶数时
        r1 += a[i++];
        r2 += a[i++];
    }

    cout<<r1 + r2<<endl;
}
```

上面的几种版本都合理，但是这些优化都是建立在编译器不能生成高效的汇编代码的假设上的。

下面来看下编译器生成的结果（vs2010，release）：

```
for(int i = 0;i < SIZE; i++) {
        S += a[i];
013B1040  mov         ebx,dword ptr [eax+4]  //把a[0],a[4],a[8]...累加到ebx中
013B1043  add         ecx,dword ptr [eax-8]  //把a[1],a[5],a[9]...累加到ecx中
013B1046  add         edx,dword ptr [eax-4]  //把a[2],a[6],a[10]...累加到edx中
013B1049  add         esi,dword ptr [eax]    //把a[3],a[7],a[11]...累加到esi中
013B104B  add         dword ptr [ebp-4],ebx
013B104E  add         eax,10h
013B1051  dec         dword ptr [ebp-8]
013B1054  jne         main+40h (13B1040h)
    }
    cout<<S<<endl;
013B1056  mov         eax,dword ptr [ebp-4]
013B1059  add         eax,esi
```

```
013B105B  add         eax,edx
013B105D  mov         edx,dword ptr [__imp_std::endl (13B204Ch)]
013B1063  add         ecx,eax              //上面的3条add指令把ebx，ecx，edx，edi都加到ecx中，
ecx是累加结果
```

可见编译器生成的代码是最好的代码，消灭了中间变量i，减少了循环次数，消灭了会造成CPU不能序执行的因素。

BTW：

有人可能会有疑问：要是size不是偶数，编译器能生成类似的高效汇编代码不？

当SIZE = 9999时：

```
//当SIZE = 9999时，编译器把中间结果放到三个寄存器中，perfect
    for(int i = 0;i < SIZE; i++) {
        S += a[i];
01341030  add         ecx,dword ptr [eax-8]
01341033  add         edx,dword ptr [eax-4]
01341036  add         esi,dword ptr [eax]
01341038  add         eax,0Ch
0134103B  dec         ebx
0134103C  jne         main+30h (1341030h)
    }
```

当SIZE = 9997 时：

```
//当SIZE = 9997时，有点复杂，先把a[0]到a[9995]累加到ecx和edx中
//再把a[9996]入到edi中，最后把ecx，edi都加到edx中
//edx压栈，调用operator<< 函数
    for(int i = 0;i < SIZE; i++) {
00D31024  xor         eax,eax
        S += a[i];
00D31026  add         ecx,dword ptr [esi+eax*4]
00D31029  add         edx,dword ptr [esi+eax*4+4]
00D3102D  add         eax,2
00D31030  cmp         eax,270Ch
00D31035  jl          main+26h (0D31026h)
    for(int i = 0;i < SIZE; i++) {
00D31037  cmp         eax,270Dh
00D3103C  jge         main+41h (0D31041h)
        S += a[i];
00D3103E  mov         edi,dword ptr [esi+eax*4]
    }
    cout<<S<<endl;
00D31041  mov         eax,dword ptr [__imp_std::endl (0D3204Ch)]
00D31046  add         edx,ecx
00D31048  mov         ecx,dword ptr [__imp_std::cout (0D32050h)]
00D3104E  push        eax
00D3104F  add         edx,edi
00D31051  push        edx
00D31052  call        dword ptr [__imp_std::basic_ostream<char,std::char_traits<char> >::opera
or<< (0D32048h)]
```

上面的分析都是SIZE，即数组的大小是已知情况下，那个数组大小是未知情况下，编译器又会怎样？

```
TYPE mySum(TYPE* a, int size){
    TYPE s = 0;
    for(int i = 0; i < size; ++i){
        s += a[i];
    }
    return s;
}
```

生成的汇编代码：

```
//先累加a[0] 到 a[size-2]
    TYPE s = 0;
00ED100C  xor         esi,esi
    for(int i = 0; i < size; ++i){
00ED100E  xor         eax,eax
00ED1010  cmp         ebx,2
00ED1013  jl          mySum+27h (0ED1027h)
00ED1015  dec         ebx
        s += a[i];
00ED1016  add         ecx,dword ptr [edi+eax*4]     //a[0],a[2],a[4]...加到ecx中
00ED1019  add         edx,dword ptr [edi+eax*4+4]   //a[1],a[3],a[5]...加到edx中
00ED101D  add         eax,2
00ED1020  cmp         eax,ebx
00ED1022  jl          mySum+16h (0ED1016h)
00ED1024  mov         ebx,dword ptr [size]
    for(int i = 0; i < size; ++i){
00ED1027  cmp         eax,ebx                       //判断最后一个元素有没有加上
00ED1029  jge         mySum+2Eh (0ED102Eh)
        s += a[i];
00ED102B  mov         esi,dword ptr [edi+eax*4]     //当size是奇数是会执行，偶数时不会执行
00ED102E  add         edx,ecx
    }
```

总结

C++的编译器生成的汇编代码在绝大多数情况下都和人写出的最好的汇编代码相当。

关键的一点是编译器会不断升级，适应新的cpu指令，体系等，手写的汇编代码则通常悲剧了。

知道编译器能优化到什么程度，编译器到底怎样优化，是程序员很重要的素质。