**《计算机原理与体系结构》实验报告**

**（版本：201203）**

**学号 16300240023 姓名 杨芗琳 学院 计算机科学技术学院完成时间 2018.3**

1. 实验名称

数的表示和缓冲器溢出

1. 实验目的
2. 熟悉windows下VS.NET开发环境
3. 熟悉IEEE754标准【1】
4. 熟悉数据在计算机内部的表示
5. 熟悉计算机的存储组织结构
6. 实验过程（编译环境：VS）

1.1精度比较

* 设计思路：以math.h中的1/sqrt(x)作为标准，令(float)x=1~100，比较1/sqrt(x)与InvSqrt(x)的大小

输出：（具体见附页）

|  |
| --- |
| i=1  math.h:1  InvSqrt(float):0.998307  差值:0.00169283 |
| i=4  math.h:0.5  InvSqrt(float):0.499154  差值:0.000846416 |

在100个数中，i=1时误差最大，为0.00169283；其次是i=4，误差为0.000846416；余下的数误差基本不超过0.0005；并且随着i的增大，误差没有明显的增长或者减小的趋势。由此我们可以认为InvSqrt函数在小数点后三位精度下是准确的。

1.2性能比较

* 设计思路：性能以运行两个函数所需要的CPU时间作为标准。分别运行两个函数，各运行100次，取多次运行结果取平均值，以达到排除cache的影响。
* 输入。以1~100作为自变量。
* 输出。（单位秒）

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1/Sqrt(x) | 0.017 | 0.014 | 0.015 | 0.014 | 0.016 | 0.021 | 0.018 | 0.017 | 0.018 | 0.015 | 0.016 | 0.018 | 0.016 | 0.015 | 0.013 |
| InvSqrt(x) | 0.015 | 0.013 | 0.017 | 0.014 | 0.017 | 0.013 | 0.016 | 0.017 | 0.013 | 0.012 | 0.014 | 0.015 | 0.014 | 0.014 | 0.017 |

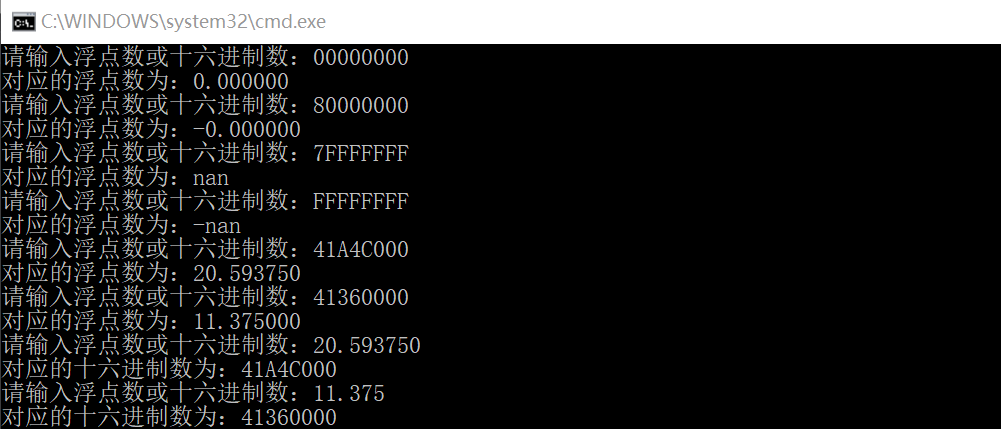
1/Sqrt(x)函数平均调用时间为0.0187秒，InvSqrt(x)函数平均调用时间为0.0123秒，故InvSqrt(x)函数性能更好。

2.实现浮点数和其16进制编码的双向转换

* 设计思路：用union建一个类，包括一个浮点数float\_type和一个整型int\_type。用同一段内存存储信息，实现浮点数和十六进制的双向转换。
* 输入。由于union类不能直接通过输入赋值，必须明确数据类型在程序中直接赋值，所以程序首先要判断输入的数据类型。所以程序首先定义一个输入的字符串data，然后判断data是一个十六进制数还是一个浮点数；确定数据类型后，把该字符串的真值存储到相应的union类中的变量中，如果是十六进制则存到int\_type中，如果是浮点数则存储到float\_type。
* 输出。前面程序已经判断了输入的数据类型，输出时以另一种数据类型输出。
* 注意要点：0，+inf，-inf，在IEEE\_754中，阶码用移码表示
* 输入输出
* 十六进制转浮点数
* +0(最小正数)：00000000
* -0(最大负数)：80000000
* 最大正数：7FFFFFFF
* 最小负数：FFFFFFFF
* 任意值，取0x41A4C000\0x41360000
* 浮点数转十六进制
* 任意值，取20.593750\11.375000
* 输出内容：

|  |
| --- |
| 请输入浮点数或十六进制数：00000000  对应的浮点数为：0.000000  请输入浮点数或十六进制数：80000000  对应的浮点数为：-0.000000  请输入浮点数或十六进制数：7FFFFFFF  对应的浮点数为：nan  请输入浮点数或十六进制数：FFFFFFFF  对应的浮点数为：-nan  请输入浮点数或十六进制数：41A4C000  对应的浮点数为：20.593750  请输入浮点数或十六进制数：41360000  对应的浮点数为：11.375000  请输入浮点数或十六进制数：20.593750  对应的十六进制数为：41A4C000  请输入浮点数或十六进制数：11.375  对应的十六进制数为：41360000 |

* 输出截图：



3.1循环不会结束。因为在IEEE754表示法中，浮点数是以定点小数的形式存储，以二进制形式表示。0.1没有一个对应的二进制定点小数，所以x永远不会等于1，循环不会结束。

3.2结果是result1=0，result2=1；因为IEEE754中，double类型的加法是先使两个的指数相同，尾数再相加。在变换指数的过程中，小数向大数看齐，所以当1的指数变成18时，它的尾数由于double精度限制变成了0，所以a+b=a，a-a=0；而第二个式子中a-a=0，0+1=1，所以result2=1.

3.3计算的是1不断除以二得到的值。最后结果为0，这个循环会结束。因为1不断除以二，最后由于double精度限制，会舍弃最后的尾数，从而得到0。

4.1//stack.c反汇编代码

|  |
| --- |
| #include<iostream>  //stack.c  #01 int add(int a, int b) {  00D61700 push ebp  00D61701 mov ebp,esp  00D61703 sub esp,44h  00D61706 push ebx  00D61707 push esi  00D61708 push edi  #02 int sum;  #03 sum = a + b;  00D61709 mov eax,dword ptr [a]  00D6170C add eax,dword ptr [b]  00D6170F mov dword ptr [sum],eax  #04 return sum;  00D61712 mov eax,dword ptr [sum]  #05 }  //省略一段  #06 int main() {  00D61730 push ebp  00D61731 mov ebp,esp  00D61733 sub esp,44h  00D61736 push ebx  00D61737 push esi  00D61738 push edi  #07 int ret = 0xDEEDBEEF;  00D61739 mov dword ptr [ret],0DEEDBEEFh  #08 ret = add(0x19, 0x82);  00D61740 push 82h  00D61745 push 19h  00D61747 call add (0D6126Ch)  00D6174C add esp,8  00D6174F mov dword ptr [ret],eax  #09 return ret;  00D61752 mov eax,dword ptr [ret]  #10 }  00D61755 pop edi  00D61756 pop esi  00D61757 pop ebx  00D61758 mov esp,ebp  00D6175A pop ebp  00D6175B ret |

寄存器：

|  |
| --- |
| EAX = 0000009B EBX = 00B63000 ECX = 00000001 EDX = 00D69588 ESI = 00D61050 EDI = 00D61050 EIP = 00D61715 ESP = 00E6FD20 EBP = 00E6FD70 EFL = 00000202 |

当程序执行第07行时，堆栈如图表1所示

low address

high address

ESP

EBP

|  |
| --- |
|  |
| return address |
| previous ebp |
| local variables(44h) |
| ebx |
| esi |
| edi |
| 0xDEEDBEEFh |
|  |

图表1

当程序运行到第03行时，堆栈如图表2所示

add()

main()

ESP

EBP

|  |
| --- |
|  |
| return address |
| previous ebp |
| local variables(44h) |
| ebx |
| esi |
| edi |
| 0xDEEDBEEFh |
| 0x82h |
| 0x19h |
| return address |
| previous ebp |
|  |
| int sum |
|  |
|  |

图表2

//retaddr.c反汇编代码

|  |
| --- |
| include<stdio.h>  using namespace std;  //retaddr.c  #01void malice() {  010B1730 push ebp  010B1731 mov ebp,esp  010B1733 sub esp,40h  010B1736 push ebx  010B1737 push esi  010B1738 push edi  #02 printf("Hey,you've been attacked.\n");  010B1739 push offset string "Hey,you've been attacked.\n" (010B6B30h)  010B173E call \_printf (010B132Ah)  010B1743 add esp,4  #03}  010B1746 pop edi  010B1747 pop esi  010B1748 pop ebx  010B1749 mov esp,ebp  010B174B pop ebp  010B174C ret  #04void foo() {  010B1700 push ebp  010B1701 mov ebp,esp  010B1703 sub esp,44h  010B1706 push ebx  010B1707 push esi  010B1708 push edi  #05 int \*ret;  #06 ret = (int\*)&ret + 2;  010B1709 lea eax,[ebp+4]  010B170C mov dword ptr [ret],eax  #07 (\*ret) = (int)malice;  010B170F mov eax,dword ptr [ret]  010B1712 mov dword ptr [eax],offset malice (010B10A5h)  #08}  010B1718 pop edi  010B1719 pop esi  010B171A pop ebx  010B171B mov esp,ebp  010B171D pop ebp  010B171E ret  #08int main() {  009C17D0 push ebp  009C17D1 mov ebp,esp  009C17D3 sub esp,40h  009C17D6 push ebx  009C17D7 push esi  009C17D8 push edi  #10 foo();  009C17D9 call foo (09C12F3h)  #11 return 0;  009C17DE xor eax,eax  #12}  009C17E0 pop edi  #13}  009C17E1 pop esi  009C17E2 pop ebx  009C17E3 mov esp,ebp  009C17E5 pop ebp  009C17E6 ret |

寄存器：

|  |
| --- |
| EAX = 0F9A1944 EBX = 006F1000 ECX = 00000001 EDX = 009C9580 ESI = 009C1050 EDI = 009C1050 EIP = 009C17D9 ESP = 0095F8DC EBP = 0095F928 EFL = 00000206 |

在程序第10行之前，堆栈情况如图表3所示

ESP

EBP

|  |
| --- |
|  |
| return address |
| previous ebp\* |
| local variables(40h) |
| ebx |
| esi |
| edi |
|  |

图表3

执行第05行之前，堆栈情况如图表4所示

|  |
| --- |
|  |
| return address |
| previous ebp\* |
| local variables(40h) |
| ebx |
| esi |
| edi |
| return address |
| previous ebp |
| local variables(44h) |
| ebx |
| esi |
| edi |
|  |

图表4

EBP

ESP

&ret

(int\*)&ret+2

执行第07行后，堆栈情况如图表5所示

EBP

|  |
| --- |
|  |
| return address |
| previous ebp\* |
| local variables(40h) |
| ebx |
| esi |
| edi |
| previous ebp |
|  |

图表5

ESP

1. 实验源代码

提交的源代码具备：可读性，有注释；交互性：输入/输出都有提示；结构化设计。

|  |
| --- |
| i=1  math.h:1  InvSqrt(float):0.998307  差值:0.00169283  i=2  math.h:0.707107  InvSqrt(float):0.70693  差值:0.000176728  i=3  math.h:0.57735  InvSqrt(float):0.576847  差值:0.000503421  i=4  math.h:0.5  InvSqrt(float):0.499154  差值:0.000846416  i=5  math.h:0.447214  InvSqrt(float):0.447141  差值:7.25687e-05  i=6  math.h:0.408248  InvSqrt(float):0.407681  差值:0.0005669  i=7  math.h:0.377964  InvSqrt(float):0.377444  差值:0.000520319  i=8  math.h:0.353553  InvSqrt(float):0.353465  差值:8.83639e-05  i=9  math.h:0.333333  InvSqrt(float):0.332953  差值:0.000380129  i=10  math.h:0.316228  InvSqrt(float):0.315686  差值:0.000541985  i=11  math.h:0.301511  InvSqrt(float):0.301036  差值:0.000475198  i=12  math.h:0.288675  InvSqrt(float):0.288423  差值:0.00025171  i=13  math.h:0.27735  InvSqrt(float):0.277312  差值:3.76105e-05  i=14  math.h:0.267261  InvSqrt(float):0.267214  差值:4.71473e-05  i=15  math.h:0.258199  InvSqrt(float):0.257751  差值:0.000448048  i=16  math.h:0.25  InvSqrt(float):0.249577  差值:0.000423208  i=17  math.h:0.242536  InvSqrt(float):0.242389  差值:0.000146791  i=18  math.h:0.235702  InvSqrt(float):0.235678  差值:2.4423e-05  i=19  math.h:0.229416  InvSqrt(float):0.229415  差值:1.14739e-06  i=20  math.h:0.223607  InvSqrt(float):0.223571  差值:3.62843e-05  i=21  math.h:0.218218  InvSqrt(float):0.218118  差值:0.000100061  i=22  math.h:0.213201  InvSqrt(float):0.213029  差值:0.000171304  i=23  math.h:0.208514  InvSqrt(float):0.208279  差值:0.000235483  i=24  math.h:0.204124  InvSqrt(float):0.203841  差值:0.00028345  i=25  math.h:0.2  InvSqrt(float):0.19969  差值:0.000310242  i=26  math.h:0.196116  InvSqrt(float):0.195802  差值:0.000314221  i=27  math.h:0.19245  InvSqrt(float):0.192154  差值:0.000296474  i=28  math.h:0.188982  InvSqrt(float):0.188722  差值:0.000260159  i=29  math.h:0.185695  InvSqrt(float):0.185485  差值:0.000210121  i=30  math.h:0.182574  InvSqrt(float):0.182422  差值:0.000152543  i=31  math.h:0.179605  InvSqrt(float):0.179511  差值:9.45777e-05  i=32  math.h:0.176777  InvSqrt(float):0.176733  差值:4.41819e-05  i=33  math.h:0.174078  InvSqrt(float):0.173998  差值:7.92146e-05  i=34  math.h:0.171499  InvSqrt(float):0.171381  差值:0.000117287  i=35  math.h:0.169031  InvSqrt(float):0.168876  差值:0.000155091  i=36  math.h:0.166667  InvSqrt(float):0.166477  差值:0.000190064  i=37  math.h:0.164399  InvSqrt(float):0.164179  差值:0.000220269  i=38  math.h:0.162221  InvSqrt(float):0.161977  差值:0.000244319  i=39  math.h:0.160128  InvSqrt(float):0.159867  差值:0.000261396  i=40  math.h:0.158114  InvSqrt(float):0.157843  差值:0.000270993  i=41  math.h:0.156174  InvSqrt(float):0.155901  差值:0.000273049  i=42  math.h:0.154303  InvSqrt(float):0.154036  差值:0.000267789  i=43  math.h:0.152499  InvSqrt(float):0.152243  差值:0.000255734  i=44  math.h:0.150756  InvSqrt(float):0.150518  差值:0.000237599  i=45  math.h:0.149071  InvSqrt(float):0.148857  差值:0.000214353  i=46  math.h:0.147442  InvSqrt(float):0.147255  差值:0.000187114  i=47  math.h:0.145865  InvSqrt(float):0.145708  差值:0.000157148  i=48  math.h:0.144338  InvSqrt(float):0.144212  差值:0.000125855  i=49  math.h:0.142857  InvSqrt(float):0.142762  差值:9.47565e-05  i=50  math.h:0.141421  InvSqrt(float):0.141356  差值:6.54161e-05  i=51  math.h:0.140028  InvSqrt(float):0.139988  差值:3.95179e-05  i=52  math.h:0.138675  InvSqrt(float):0.138656  差值:1.88053e-05  i=53  math.h:0.137361  InvSqrt(float):0.137356  差值:5.00679e-06  i=54  math.h:0.136083  InvSqrt(float):0.136083  差值:0  i=55  math.h:0.13484  InvSqrt(float):0.134834  差值:5.57303e-06  i=56  math.h:0.133631  InvSqrt(float):0.133607  差值:2.35736e-05  i=57  math.h:0.132453  InvSqrt(float):0.132397  差值:5.58794e-05  i=58  math.h:0.131306  InvSqrt(float):0.131202  差值:0.000104323  i=59  math.h:0.130189  InvSqrt(float):0.130018  差值:0.000170797  i=60  math.h:0.129099  InvSqrt(float):0.128875  差值:0.000224024  i=61  math.h:0.128037  InvSqrt(float):0.12782  差值:0.000217199  i=62  math.h:0.127  InvSqrt(float):0.126787  差值:0.000212938  i=63  math.h:0.125988  InvSqrt(float):0.125777  差值:0.00021112  i=64  math.h:0.125  InvSqrt(float):0.124788  差值:0.000211604  i=65  math.h:0.124035  InvSqrt(float):0.123867  差值:0.000168003  i=66  math.h:0.123091  InvSqrt(float):0.122961  差值:0.000130825  i=67  math.h:0.122169  InvSqrt(float):0.12207  差值:9.94653e-05  i=68  math.h:0.121268  InvSqrt(float):0.121194  差值:7.33957e-05  i=69  math.h:0.120386  InvSqrt(float):0.120334  差值:5.20796e-05  i=70  math.h:0.119523  InvSqrt(float):0.119488  差值:3.50848e-05  i=71  math.h:0.118678  InvSqrt(float):0.118656  差值:2.19271e-05  i=72  math.h:0.117851  InvSqrt(float):0.117839  差值:1.22115e-05  i=73  math.h:0.117041  InvSqrt(float):0.117036  差值:5.55813e-06  i=74  math.h:0.116248  InvSqrt(float):0.116246  差值:1.61678e-06  i=75  math.h:0.11547  InvSqrt(float):0.11547  差值:4.47035e-08  i=76  math.h:0.114708  InvSqrt(float):0.114707  差值:5.73695e-07  i=77  math.h:0.113961  InvSqrt(float):0.113958  差值:2.89083e-06  i=78  math.h:0.113228  InvSqrt(float):0.113221  差值:6.76513e-06  i=79  math.h:0.112509  InvSqrt(float):0.112497  差值:1.19209e-05  i=80  math.h:0.111803  InvSqrt(float):0.111785  差值:1.81422e-05  i=81  math.h:0.111111  InvSqrt(float):0.111086  差值:2.52575e-05  i=82  math.h:0.110432  InvSqrt(float):0.110398  差值:3.30508e-05  i=83  math.h:0.109764  InvSqrt(float):0.109723  差值:4.13731e-05  i=84  math.h:0.109109  InvSqrt(float):0.109059  差值:5.00306e-05  i=85  math.h:0.108465  InvSqrt(float):0.108406  差值:5.89266e-05  i=86  math.h:0.107833  InvSqrt(float):0.107765  差值:6.79046e-05  i=87  math.h:0.107211  InvSqrt(float):0.107134  差值:7.68527e-05  i=88  math.h:0.1066  InvSqrt(float):0.106515  差值:8.56519e-05  i=89  math.h:0.106  InvSqrt(float):0.105906  差值:9.42275e-05  i=90  math.h:0.105409  InvSqrt(float):0.105307  差值:0.000102483  i=91  math.h:0.104828  InvSqrt(float):0.104718  差值:0.000110343  i=92  math.h:0.104257  InvSqrt(float):0.104139  差值:0.000117742  i=93  math.h:0.103695  InvSqrt(float):0.103571  差值:0.000124626  i=94  math.h:0.103142  InvSqrt(float):0.103011  差值:0.000130944  i=95  math.h:0.102598  InvSqrt(float):0.102461  差值:0.000136659  i=96  math.h:0.102062  InvSqrt(float):0.10192  差值:0.000141725  i=97  math.h:0.101535  InvSqrt(float):0.101388  差值:0.000146121  i=98  math.h:0.101015  InvSqrt(float):0.100865  差值:0.000149831  i=99  math.h:0.100504  InvSqrt(float):0.100351  差值:0.000152834 |