**实验项目名称 数据表示实验01**

## 一、 实验目的

数据表示实验：

理解补码的基本特性，能分析简单C语言程序中与数据表示相关的执行行为，并通过实际代码执行验证真实计算机中数据表示的编码体系，掌握获取C语言变量机器码的方法，培养数据表示层面基本的软硬件协同系统观。

浮点数表示实验：

## 二、 主要仪器设备、实验环境

Dev C++集成开发环境

## 三、 实验任务

数据表示实验：

1、如果给你10个信封和1000张1元的纸币，如何将1000元钱分装到10个信封中才能保证不管给出1~1000的任何一个数字，都可以从10个信封中直接拿走若干信封？请在下面10个信封上写上具体钱数。

2、给出C语言代码，结合数据表示相关知识，分析程序功能。根据自己的分析和理解完成要求程序的输出。

|  |
| --- |
| #include <stdio.h>  int main() {  int n;  printf("%d", sizeof(long long ));  while (1) {  printf("请输入一个数(输入0退出程序)：");  scanf("%d", &n);  if (!n)  break;  int i = 1;  int t = 1;  while (n) {  if (n % 2) {  printf("拿第%d个信封:%d\n", i, t);  }  n = n / 2;  i++;  t \*= 2;  }  }  return 0;  } |

3、在C语言开发环境中编译运行以上代码，验证自己的分析结果，如有不同，请思考原因，并找出依据。

## 四、 实验过程及结果分析

（1）machine\_code\_output.c的输出：

a= 4294967295 = -1 = 0xfffffff

b= 2147483648 = -2147483648 = 0x80000000

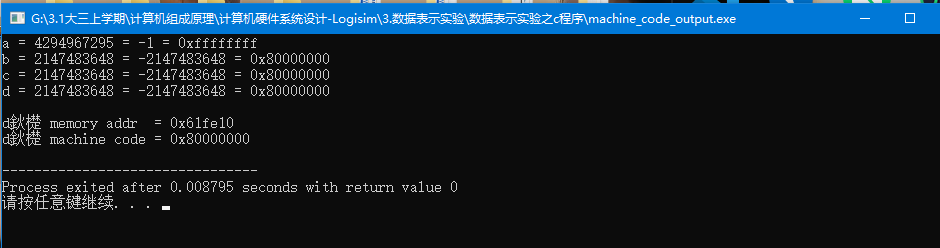
c= 2147483648 = -2147483648 = 0x80000000

d= 2147483648 = -2147483648 = 0x80000000

d’s memory addr = 0x61fe10

d’s machine code = 0x80000000

编译运行结果：



结果分析：

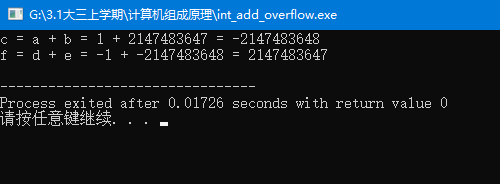
b底层存储为0x80000000,取负值后，依然为0x8000000,所以c和b 的输出一模一样。

（2）int\_add\_overflow.c的输出：

c=a+b= 1 + 2147483647 = -2147483648

f=e+d= -1 + -2147483647 = 2147483647

编译运行结果：



结果分析：

a+b 正溢出，最高位符号位由0变为1，导致结果变为负数，即0x00000001+0x0fffffff=0x10000000,表示的数即为-2147483648。

d+e 负溢出，发生最高位进位情况，发生进位的数由于长度限制被丢弃，即

0x111111111+0x100000000=0x01111111,表示的数即为2147483647。

（3）get\_var\_size.c的输出：

sizeof ( char ): 1

sizeof ( short ): 2

sizeof ( int ): 4

sizeof ( unsigned int ): 4

sizeof ( long ): 4

sizeof ( long long ): 8

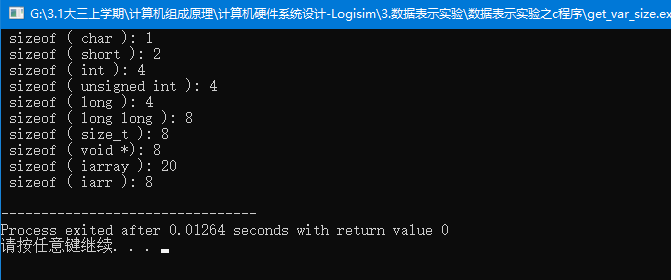
sizeof ( size\_t ): 8

sizeof ( void \*): 8

sizeof ( iarray ): 20

sizeof ( iarr ): 8

编译运行结果：



## 五、实验思考

（1）machine\_code\_output.c中，变量b和c是否相等，为什么？

相等，b底层存储为0x80000000,取负值后，依然为0x8000000,所以c和b 的输出一模一样。

（2）machine\_code\_output.c中，变量d的内存地址输出的值是否为一个确定的值，其值能否被4整除，为什么？

能被4整除，内存对齐。

（3）C语言中int、short、char、long型变量在内存中存储的机器码是采用什么编码？

补码

（4）C语言中int中的存储方式是？（小端还是大端方式？存储方式是由什么决定的？）

小端。

存储方式由CPU架构决定。