实验2 数据表示和运算实验

一、实验目的

1. 了解并学习计算机的数据表示方式，了解并学习计算机的算术运算方式，理解不同数据类型的运算属性。
2. 了解并学习gdb的使用方法，并运用其进行内存、寄存器检查。

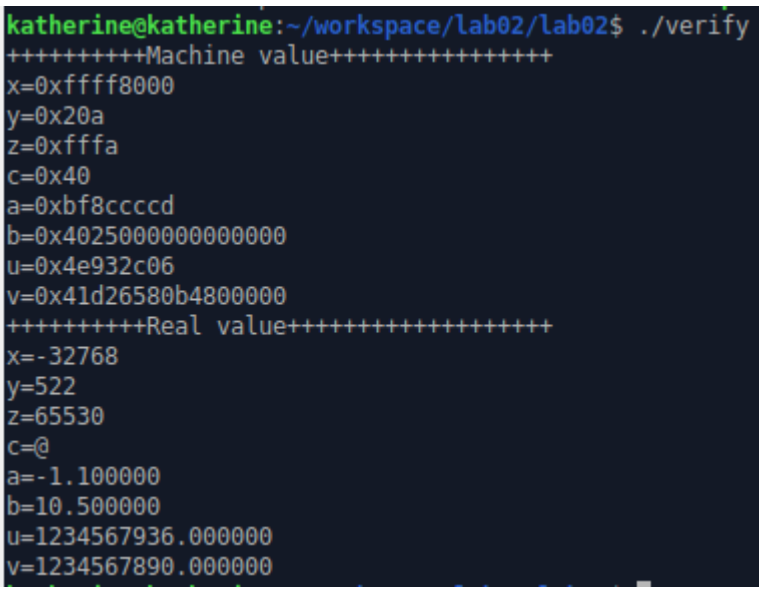
二、实验内容

1.

运行一个C语言程序，并查看以下变量的机器数

变量	x	y	z	c
机器数	0xffff8000	0x020a	0x0000fffa	0x40
变量	a	b	u	v
机器数	0xbf8ccccd	0x4025000000000000	0x4e932c06	0x41d26580b4800000

运行代码获得的输出如下：



可以看到输出与gdb查看的结果一致。

2.

1）使用gdb查看程序变量的取值,填写下面两个表格

a的存放地址(&a)	b的存放地址(&b)	x的存放地址(&x)	y的存放地址(&y)
0xbffff094	0xbffff098	0xbffff088	0xbffff08c

函数中

执行步数	x的值(机器值，用十六进制)	y的值(机器值，用十六进制)	*x的值(程序中的真值，用十进制)	*y的值(程序中的真值，用十进制
第一步前	0xbffff094	0xbffff098	1	2
第一步后	0xbffff094	0xbffff098	1	3
第二步后	0xbffff094	0xbffff098p	2	3
第三步后	0xbffff094	0xbffff098	2	1

2）运行reverse.c 说明输出这种结果的原因，修改代码以得到正确的逆序数组

运行结果如下：

```
katherine@katherine:~/workspace/lab02/lab02$ vim reverse.c
katherine@katherine:~/workspace/lab02/lab02$ gcc reverse.c -o reverse
katherine@katherine:~/workspace/lab02/lab02$ ./reverse
7650321
```

可以观察到数组最中间的元素变成了0，其余都正常逆序了。
输出这种结果的原因是，当数组元素个数是奇数时，在最后一次循环中，left=right,根据性质a^a=0,将数组最中间的元素设置为0.要得到正确的数组，只需要把`left<=right`改成`left=right`.

修改后的程序运行结果如下，可以看到数组恢复了正常逆序。

```
katherine@katherine:~/workspace/lab02/lab02$ vim reverse.c
katherine@katherine:~/workspace/lab02/lab02$ gcc reverse.c -o reverse
katherine@katherine:~/workspace/lab02/lab02$ ./reverse
7654321
```

3. 编译并运行程序，使用gdb查看变量的值，解释语句输出为False的原因并填写在表格中

	输出 True/False	**原因**
语句一	True	/
语句二	False	用gdb查看变量的值发现(int)xf==INT_MIN，在强制类型转换的时候发生了溢出
语句三	False	用gdb查看变量的值发现p1=p2=3.14159274，单精度浮点表示的是近似值，p1和p2在位层面是同样的表示（用以2为基数的科学计数法来表示最接近的数），所以p1=p2
语句四	True	/
语句五	False	用gdb查看变量的值发现result2=0，因为d远小于f，在d+f时由于float精度的限制，大小的变化被忽略不计，所以最终结果相当于f-f是0

4.观察下面data_rep.c程序的运行

1）完成表格

	机器数（十六进制）	真值（十进制）		机器数（十六进制）	真值（十进制）
x	0x66	102	y	0x39	57
~x	0x99n	-103	!x	0x0	0
x & y	0x20	32	x && y	0x1	1
x | y	0x7f	127	x || y	0x1	1

	机器数（十六进制）	真值（十进制）	OF	SF	CF	AF
x1	0x7fffffff	2147483647	0	0	0	0
y1	0x1	1	0	0	0	0
sum_x1_y1	0x80000000	-2147483648	1（运算结果产生了溢出，超出有符号整数能表示的范围）	1（运算结果的最高位是1，因为是负数）	0	1（辅助进位，半字节产生了进位）
diff_x1_y1	0x7ffffffe	2147483646	0（运算结果没有溢出）	0（运算结果最高位是0）	0	0（没有产生辅助进位）
diff_y1_x1	0x80000002	-2147483646	0	1（运算结果是负数，最高位是1）	1（减法时产生借位）	1（产生辅助进位）
x2	0x7fffffff	2147483647	0	1	1	1
y2	0x1	1	0	1	1	1
sum_x2_y2	0x80000000	2147483648	1（运算结果超出有符号整数的表示范围，但因为数据类型是无符号，所以还是正数）	1	0（加法时没有产生进位）	1
diff_x2_y2	0x7ffffffe	2147483646	0（运算结果没有超出有符号整数的表示范围）	0（运算结果最高位是0）	0	0（没有产生辅助进位）
diff_y2_x2	0x80000002	2147483650	0	1（运算结果最高位是1，但因为数据类型是无符号整数，所以表现为正数）	1（减法时产生借位）	1（产生了辅助进位）

2）写出上面表格中每个标识位变化的原因，可直接在上表中注明

仅写出变化的原因，如果较上一步没有变化，则不写原因。