实验 2　数据表示与运算实验

姓名：林荣恩　　学号：191220060　　院系：计算机科学与技术系

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.　在 64 位计算机中运行一个 C 语言程序，在该程序中出现了以下变量的初值，在表格中填写它们对应的机器数(用十六进制表示)。在 gdb 里面可使用 x/1xw 查看 int/unsigned/float 的机器数,使用 x/1xh 查看 short/unsigned short 的机器数,使用 x/1xb 查看 char 的机器数,使用 x/1xg 查看 double 的机器数:

(1) int x=-32768
(2) short y=522
(3) unsigned z=65530
(4) char c='@'
(5) float a=-1.1
(6) double b=10.5
(7) float u=123456.789e4
(8) double v=123456.789e4

变量	x	y	z	c
机器数	0xffff8000	0x20a	0xfffa	0x40
变量	a	b	u	v
机器数	0xbf8ccccd	0x4025000000000000	0x4e932c06	0x41d26580b4800000

2.　使用命令 gcc –ggdb swap.c –o swap 编译 swap.c 代码,完成后面的实验。

1)　使用 gdb 命令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)
0x7fffffffdf70	0x7fffffffdf74	0x7fffffffdf58	0x7fffffffdf50

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	0x7fffffffdf70	0x7fffffffdf74	1	2
第一步后	0x7fffffffdf70	0x7fffffffdf74	1	3
第二步后	0x7fffffffdf70	0x7fffffffdf74	2	3
第三步后	0x7fffffffdf70	0x7fffffffdf74	2	1

2) 运行 reverse.c，并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。

得到错误结果的原因：left=right=3 时，传入 xor_swap 的指针 x 和 y 相同，指向同一位置，在异或交换的第一步 (*y=*x^*y) 中该位置上的值变为 0，且之后如何操作该位置的值都保持为 0，无法起到交换的作用。

修改后的代码见 reverse.c.

3. 编译并运行程序，使用 gdb 指令查看变量的取值，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	double 表示的二进制小数的有效位数为 52 位（十进制下至多 16 位），x 转为 double 类型后精度得以保证。
语句二	False	float 表示的二进制小数的有效位数为 23 位（十进制下至多 7 位），x 转为 float 类型后末尾几位遭截取，精度丢失，比较结果为不等。
语句三	False	同上，两数末尾几位遭截取，得到 p1=p2=3.14152974.
语句四	True	d+(f-f)=d+0=d. 比较结果为相等。
语句五	False	由于 d+f 远大于 d，且受限于 double 二进制的 52 位限制，d+f 的真实结果的后数位被截断，精度丢失，d 在 d+f 中造成的变化被消除，即“d+f=f”，故(d+f)-f=f-f=0，比较结果为不等。

4. 观察 data_rep.c 程序的运行。

1) 使用命令 gdbtui data_rep 进入 gdb 的 TUI 调试模式，之后分别输入命令：layout asm 和 layout regs，再输入命令 start 启动程序，然后使用 si 命令进行单步运行。

	机器数 (16 进制)	真值 (10 进制)		机器数 (16 进制)	真值 (10 进制)
x	0x66	102	y	0x39	57
~x	0x99	-103	!x	0x00	0
x&y	0x20	32	x&&y	0x01	1
x|y	0x7f	127	x||y	0x01	1

	机器数 (16 进制)	真值 (10 进制)	OF	SF	CF	AF
x1	0x7fffffff	2147483647	0	0	0	0
y1	0x00000001	1	0	0	0	0
sum_x1_y1	0x80000000	-2147483648	1 (有符号溢出)	1 (结果<0)	0	1 (低 4 位向高 4 位存在进位)
diff_x1_y1	0x7ffffffe	2147483646	0 (未溢出)	0 (结果>0)	0	0 (未有高 4 位向低 4 位的借位)
diff_y1_x1	0x80000002	-2147483646	0	1 (结果<0)	1(Cout=0, Cin=1)	1(高 4 位向低 4 位存在借位)
x2	0x7fffffff	2147483647	0	1	1	1
y2	0x00000001	1	0	1	1	1
sum_x2_y2	0x80000000	2147483648	1(x2,y2 最高位 0, sum_x2_y2 最高位 1)	1	0(Cout=0, Cin=0)	1

diff_x2_y2	0x7ffffffe	2147483646	0(x2,y2,sum_x2_y2 最高位均 0)	0 (无符号, 最高位=0)	0	0(未有高 4 位向低 4 位的借位)
diff_y2_x2	0x80000002	2147483650	0	1 (无符号, 最高位=1)	1(Cout=0, Cin=1)	1(高 4 位向低 4 位存在借位)

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。