**X86部分**

**一、填空题（20分）**

1. 已知某32位整数X，其值为-102，则其16进制补码为\_\_\_\_\_\_\_\_\_\_\_ ，另一32位整数Y的补码为0xFFFFFF68H，则X+Y的16进制补码(32位)为 \_\_\_\_\_\_ ，X-Y的16进制补码为 \_\_\_\_\_\_ 。
2. X86-64位linux系统下的float类型的数据对齐要求是 \_\_\_ 字节对齐，double类型的是 \_\_ 字节对齐。
3. 假设存在一种16位的浮点数表示，exp位数是7，frac位数是8，符号位为1，其所能表示的绝对值最小的规格化数的exp是 ，frac是 ；251的exp是 ，frac是 。
4. 给出13/8这一数字的32位浮点数（符合IEEE 754标准）表示，即exp= ；frac= 。

5）在有test.c源代码的情况下，如何编译出该源代码对应的汇编程序，请写出命令行：

6）在没有源代码，只有执行文件test的情况下，在linux系统或者macos上如何反编译出该执行文件所对应的汇编代码, 请写出命令行（写一种就可以）。

1. **判断题（10分）**

已知 int x = …; int y = …; float f = …; double d = …; unsigned int ux = x; unsigned int uy = y;

判断下面的等式（或不等式或推导）是否成立。

（1）x == (int)(float) x

（2）x == (int)(double) x

（3）f == (float)(double) f

（4）d == (float) d

（5）d > f  **->** -f > -d

（6）(d+f)-d == f

（7）(x>y) == (-x<-y)

（8）(x|-x)>>31 == -1

（9）~x+~y == ~(x+y)

（10）(int)(ux-uy) == -(y-x)

1. **简答题**

（1）（8分）有如下C语言的结构声明  
typedef struct {  
 short code;  
 long start;  
 char raw[3];  
 double data;  
} OldSensorData;

typedef struct {  
short code;  
short start;  
char raw[5];  
short sense;  
short ext;  
double data;

} NewSensorData;

1.1）在运行Linux系统的X86-64计算机上，这两个结构数据的内存layout是怎么样的，请在下面的表格里标出结构体内各个变量所占据的位置（假设起始地址是0，标出各个变量名以及size，如果有对齐用的padding空间请用不同方式标出，比如涂成灰色）；同时标出结构所占内存的大小（字节数）。

OldSensorData

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

NewSensorData

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 |  | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1.2）有如下C代码

void foo(OldSensorData \*oldData)  
{  
 NewSensorData \*newData;  
 /\* this zeros out all the space allocated for oldData \*/  
 bzero((void \*)oldData, sizeof(oldData));  
 oldData->code = 0x104f;  
 oldData->start = 0x80501ab8;  
 oldData->raw[0] = 0xe1;  
 oldData->raw[1] = 0xe2;  
 oldData->raw[2] = 0x8f;  
 oldData->raw[-5] = 0xff;  
 oldData->data = 1.5;  
 newData = (NewSensorData \*) oldData;  
 ...  
上述代码运行后，我们访问newData指向的地址，请填充如下空格（16进制数表示，注意字节序）。

(a) newData->start = 0x\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
(b) newData->raw[0] = 0x\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
(c) newData->raw[2] = 0x\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
(d) newData->raw[4] = 0x\_\_\_\_\_\_\_\_\_\_\_\_\_  
(e) newData->sense = 0x\_\_\_\_\_ \_\_\_\_

2）X86 32位体系结构中的条件跳转指令jg是用于符号数比较还是无符号数比较的？其产生跳转的成立条件是~(SF^OF)&~ZF为真，请解释为何是这一条件。（4分）

3）下图给出了一个C函数，并由gcc编译成相应的汇编代码（AT&T语法格式），请补全这段代码里头被省去的部分。（32位X86代码，10分）

编译出的代码：

…

movl 8(%ebp),%eax

movl (%ebp),%edx

leal (% , %eax),%ecx

leal (%edx,%edx,2),%

sall ,%edx

addl (%ebp), %ecx

leal 4(%edx,%eax),%eax

imull %ecx,%eax

…

int arith(int x, int y, int z)

{

int t1= x+y;

int t2 =z+t1;

int t3=x+4;

int t4=y\*48;

int t5=t3+t4;

int rval=t2\*t5;

return rval;

}

4） C语言中过程的参数个数可以是不固定的。比如定义了如下能够产生格式化输出的过程：

void my\_printf(const char \*fmt, ...)；

其参数个数大于等于1，第一个参数是一个格式字符串，可以接受形如“input string %d %d”之类的字符串作为输入，其中%d指定输出32位带符号整数，具体的输出整数值则由后续的参数指定（为简化起见，这个函数只能接受%d作为格式转换）。这个函数的汇编代码如下所示，请分析这些代码并回答如下问题：（11分）

* 这类不定参数的过程是如何传入参数的？
* my\_printf是如何确定不定参数个数的？

.section .rdata,"dr"

LC0:

.ascii "%d\0"

.text

.globl \_my\_printf

\_my\_printf:

pushl %ebp

movl %esp, %ebp

subl $16, %esp

pushl %esi

leal 12(%ebp), %esi

pushl %ebx

movl 8(%ebp), %ebx

movzbl (%ebx), %eax

testb %al, %al

je L11

L14:

cmpb $37, %al #’%’的ascii码值是37

je L5

movsbl %al,%eax

L8:

movl %eax, (%esp)

incl %ebx

call \_putchar

L15:

movzbl (%ebx), %eax

testb %al, %al

jne L14

L11:

addl $16, %esp

popl %ebx

popl %esi

popl %ebp

ret

L5:

incl %ebx

movsbl (%ebx),%eax

cmpl $100, %eax

jne L8

movl %esi, %eax

incl %ebx

movl (%eax), %eax

addl $4, %esi

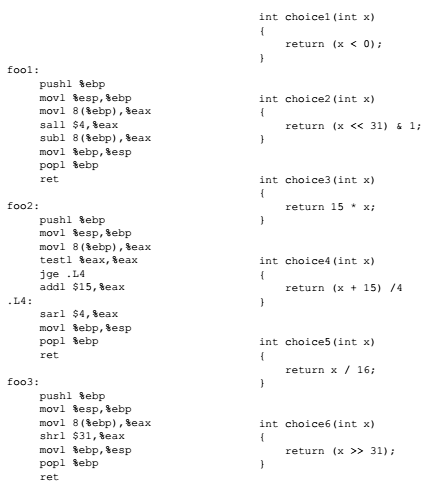
movl $LC0, (%esp)

movl %eax, 4(%esp)

call \_printf

jmp L15

5）（8分）左侧的汇编代码段foo1（2分）、foo2（3分）、foo3（3分）分别对应右侧的哪段C代码？

****