根据user/call.c编译出的call.asm回答问题

1.Which registers contain arguments to functions? For example, which register holds 13 in main's call to printf?哪些寄存器保存函数调用的参数？，main函数调用printf函数时，哪个寄存器保存了13？

A：a0-a7保存参数。a2保存13。

2.Where is the call to function f in the assembly code for main? Where is the call to g? (Hint: the compiler may inline functions.) main 中调用函数 f 对应的汇编代码在哪？对 g 的调用呢？ (提示：编译器有可能会内联(inline)一些函数)

A：从

26: 45b1 li a1,12

这行可以看出在汇编中main函数没有调用f函数，而是直接把f(8)+1结果保存到了寄存器a1。同时在f函数也找不到g函数的调用，因为编译器使用函数内联结合常量传播对其调用进行了优化。

3.At what address is the function printf located? printf 函数所在的地址是?

A:从

30: 00000097 auipc ra,0x0

34: 5f8080e7 jalr 1528(ra) # 628 <printf>

从这两行可以看出，auipc是把0x0左移12位，再加上PC值0x34(PC指向下一程序的地址)存到ra里，第二行ra加上1528就是0x628就是printf函数的地址。

在下面也可以看到printf的地址  
图形用户界面, 文本

描述已自动生成

4.What value is in the register ra just after the jalr to printf in main? 在 main 中 jalr 跳转到 printf 之后，ra 的值是什么？

A：JALR（Jump And Link Register）,跳转到ra指向的地址执行程序，然后把原本下一行的地址保存到ra，call.asm中，下一行是

38: 4501 li a0,0

所以ra的内容是0x38。

5. Run the following code.

unsigned int i = 0x00646c72;

printf("H%x Wo%s", 57616, &i);

What is the output? The output depends on that fact that the RISC-V is little-endian. If the RISC-V were instead big-endian what would you set i to in order to yield the same output? Would you need to change 57616 to a different value?运行程序输出结果是什么（RISC-V 是小端序的）？如果 RISC-V 是大端序的，要实现同样的效果，需要将 i 设置为什么？需要将 57616 修改为别的值吗？

A:%x输出的是把57616转换成16进制：e110，%s输出的是把保存i地址所指向的内容0x00646c72转换为字符串输出。因为是小端序的所以字符串中每个字符的ASCII码为：72 6c 64 00，对应的字符是r l d \0(\0表示结束)。如果是大端序的，那么i=0x726c6400，57616不需要改变

6. In the following code, what is going to be printed after 'y='? (note: the answer is not a specific value.) Why does this happen? 在下面的代码中，'y=' 之后会输出什么？ (note: 答案不是一个具体的值) 为什么?

printf("x=%d y=%d", 3);

A:输出x=3,y=1，由于没有指定第三个参数(y=后面)的内容，所以printf会读取a2寄存器的值。由于没有运行其他程序，所以目前a2寄存器的值为1，输出y=后面就是1。

这句设置断点，从gdb调试i reg也可以看到