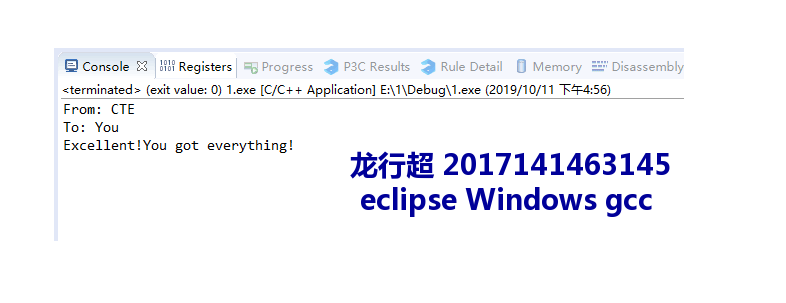
1. key3 = -1； key4 = 55；



原理解释：解开key3、key4的关键是跳过if (\*msg1 == '\0')的判断，而要跳过这个判断只有一种可能：process\_keys34(&key3, &key4);函数通过指针改变了栈帧里的返回值，栈帧从上到下是&key4、&key3、return，process\_keys34(&key3, &key4);的内容为\*(((int \*)&key3) + \*key3) += \*key4;也就是左边指向了return，从栈帧上可知，key3的值为-1，而key4的值为要跳的位置-当前栈帧的return，由反汇编窗口能轻易得到这两个值，计算出来结果为55，所以key3 = -1；key4 = 55；

1. 栈帧结构如下（从高地址到低地址，写出变量和ebp的布局）：

|  |  |
| --- | --- |
| main | ebp |
| a |
| b |
| … |
| b |
| a |
| rt |
| … |
| &b |
| &a |
| rt |
| … |
| swap2 | ebp |
| temp |
| … |
| swap1 | ebp |
| temp |
| … |

1. 输入字符串为：b8 ef be ad de 68 ff 14 40 00 c3 00 00 00 00 00 00 00 00 00 00 00 00 00 88 fe 61 00 70 fe 61 00

解题思路：植入恶意程序代码，通过修改eax的值来达成目的，详细解释如下：首先得到以下三条汇编的机器码：

mov $0xdeadbeef, %eax //修改eax寄存器的值

push $0x4014ff //将test中调用getbuf的返回地址压入栈中

ret //弹出栈顶元素并把值传给IP，与上条结合实现实现回到原来的程序执行

对应的机器码如下：

b8 ef be ad de

68 ff 14 40 00

c3

接下来利用缓冲区溢出，修改getbuf函数的返回地址，将其指向为buf缓冲区，然后执行上述三条指令，往eax里赋值，紧接着再回到getbuf函数的返回地址执行val变量的赋值，在此过程中要保证栈帧结构不被破环，所以要注意保留getbuf函数栈帧中的指向test函数的ebp保持不变，否则程序将不会得到想要的结果，故所要做的工作除了得知getbuf函数的返回地址之外，还要知道test函数栈帧的ebp的值。总结如下：

1. 写入机器码
2. 汇编窗口得到getbuf的返回地址
3. test函数栈帧的基地址
4. 获得buf缓冲区首地址

获得2，3两个东西的办法就是在getbuf函数入口处设置断点，然后运行一条char buf[16]的指令，这样的话ebp的值指向的就是test栈帧的基地址，eax的值指向的就是buf首地址，实验截图如下：

