ICS Lab2 说明文档

16302010026 陈涛

Phase_1:

根据函数名知道需要做字符串的比较，所以需要输入一个字符串，而且在调用函数对比字符串前，把一个地址存到了寄存器 esi，通过打印地址里面字符串得到字符串：

```
(gdb) p (char *) 0x4023b0
$1 = 0x4023b0 "Houses will begat jobs, jobs will begat houses."
```

所以第一题答案为

Houses will begat jobs, jobs will begat houses.

Phase_2:

根据函数名<read_six_numbers>知，需要输入 6 个整数。

根据汇编：

```
0x0000000000400ec7 <+30>:    cmpl   $0x0,(%rsp)
0x0000000000400ecb <+34>:    jns    0x400ed2 <phase_2+41>
0x0000000000400ecd <+36>:    callq  0x40141d <explode_bomb>
0x0000000000400ed2 <+41>:    mov    %rsp,%rbp
```

知道第一个数为非负数

再根据汇编中的循环：

```
0x0000000000400ed2 <+41>:    mov    %rsp,%rbp
0x0000000000400ed5 <+44>:    mov    $0x1,%ebx
0x0000000000400eda <+49>:    mov    %ebx,%eax
0x0000000000400edc <+51>:    add    0x0(%rbp),%eax
0x0000000000400edf <+54>:    cmp    %eax,0x4(%rbp)
0x0000000000400ee2 <+57>:    je     0x400ee9 <phase_2+64>
0x0000000000400ee4 <+59>:    callq  0x40141d <explode_bomb>
0x0000000000400ee9 <+64>:    add    $0x1,%ebx
0x0000000000400eec <+67>:    add    $0x4,%rbp
0x0000000000400ef0 <+71>:    cmp    $0x6,%ebx
0x0000000000400ef3 <+74>:    jne    0x400eda <phase_2+49>
```

明白 e1+1=e2；e2+2=e3；e3+3=e4；以此类推，由于第一个数为非负数，所以我取第一个为 1，则获得 6 个数为：1,2,4,7,11,16

所以第二题答案为：

1 2 4 7 11 16

Phase_3:

发现在 scanf 函数前使用了地址 0x4025af，输出地址：

```
(gdb) p (char *) 0x4025af
$2 = 0x4025af "%d %d"
```

可知第三题需要输入两个整数。

仔细看汇编，明白 switch 语句，且发现：

```
0x0000000000400f47 <+54>:    mov    (%rsp),%eax
0x0000000000400f4a <+57>:    jmpq   *0x402420(,%rax,8)
```

以第一个参数的值乘以 8，加上地址 0x402420 存储的值为跳转地址。

输出地址 0x402420 的值，为 4198286，即 0x400f8e

```
(gdb) p *0x402420
$4 = 4198286
```

根据汇编：

```
0x0000000000400f8e <+125>:   mov    $0x1c7,%eax
0x0000000000400f93 <+130>:   cmp    0x4(%rsp),%eax
0x0000000000400f97 <+134>:   je     0x400f9e <phase_3+141>
0x0000000000400f99 <+136>:   callq  0x40141d <explode_bomb>
0x0000000000400f9e <+141>:   mov    0x8(%rsp),%rax
```

发现 0x400f8e 处直接将常数 0x1c7 与第二个参数作比较，相等即可。所以为了方便，直接将第一个参数赋值为 0，那么直接跳到 0x400f8e 处，第二个参数赋值为 0x1c7，即 455 即可。

所以第三题一种答案为：

0 455

Phase_4:

表示 6 题中被这一题坑的最惨（被参数位置的变化坑到）。

用第三题的方法知道这一题需要输入两个整数。

根据汇编：

```
0x000000000040101e <+43>:    mov    (%rsp),%eax
0x0000000000401021 <+46>:    sub    $0x2,%eax
0x0000000000401024 <+49>:    cmp    $0x2,%eax
0x0000000000401027 <+52>:    jbe    0x40102e <phase_4+59>
0x0000000000401029 <+54>:    callq  0x40141d <explode_bomb>
0x000000000040102e <+59>:    mov    (%rsp),%esi
```

发现其中一个参数是在 2~4 之间，由于前面的原因，一直以为这是第一个参数，所以一直炸。后面调整位置，第二个参数改为 2~4 之间，终于不再炸。

可是进入 func4 查看代码，发觉太烦，看不懂。

```
0x0000000000401036 <+67>:    callq  0x400fb8 <func4>
0x000000000040103b <+72>:    cmp    0x4(%rsp),%eax
0x000000000040103f <+76>:    je     0x401046 <phase_4+83>
0x0000000000401041 <+78>:    callq  0x40141d <explode_bomb>
0x0000000000401046 <+83>:    mov    0x8(%rsp),%rax
```

于是就在地址 0x40103b 那加了断点，获取 func4 的返回值。得到：第二个参数是 2 的时候，返回值 66，输入 3 返回 99，输入 4，返回 132.选了 3 作为输入。

所以第四题的一种答案是：

99 3

另：在做完之后听说有 secret phase，由于找不到入口，便向其他人询问了 secret phase 的密码，知道是在第四题的答案后面加字符串，

所以第四题我的答案是：

99 3 DrEvil

Phase_5:

第五题查看地址 0x4025af，得知本次需要输入两个整数。

根据汇编：

```
0x0000000000401090 <+48>:    mov    (%rsp),%eax
0x0000000000401093 <+51>:    and    $0xf,%eax
0x0000000000401096 <+54>:    mov    %eax,(%rsp)
0x0000000000401099 <+57>:    cmp    $0xf,%eax
0x000000000040109c <+60>:    je     0x4010cd <phase_5+109>
```

了解到第一个参数二进制表示的低四位不能为 1111，而且第一个参数在与 0xf 进行按位与运算后将结果保存为参数，所以第一个参数只要其低四位的二进制不为 1111 即可，大小无影响，姑且设为 0~14.

根据汇编：

```
0x00000000004010a8 <+72>:    add    $0x1,%edx
0x00000000004010ab <+75>:    cltq
0x00000000004010ad <+77>:    mov    0x402460(,%rax,4),%eax
0x00000000004010b4 <+84>:    add    %eax,%ecx
0x00000000004010b6 <+86>:    cmp    $0xf,%eax
0x00000000004010b9 <+89>:    jne    0x4010a8 <phase_5+72>
0x00000000004010bb <+91>:    movl   $0xf,(%rsp)
0x00000000004010c2 <+98>:    cmp    $0xf,%edx
Type <return> to continue, or q <return> to quit---
0x00000000004010c5 <+101>:   jne    0x4010cd <phase_5+109>
0x00000000004010c7 <+103>:   cmp    0x4(%rsp),%ecx
```

了解到这里有一个循环，而且参数%edx 从 1 加到了 15，也就是说循环的次数必须等于 15 次，且在最后一次循环结束时，%eax 的值必须为 15。循环中，还将%eax 中的值加入%ecx 中，在循环结束之后与第二个参数做对比。

在循环中不难发现%eax 的值在变化，每次都从不同地址中取值，通过打印

这些地址中的值，发现是一个数组（起始位置为 0x402460）中的值，且该数组为{10,2,14,7,8,12,15,11,0,4,1,13,3,9,6,5}

于是采取逆向分析，将%edx=15，%eax=15 代入循环逆向寻找答案，最终得到%eax 的初始值为 5,所以第一个参数取 5 即可,再将循环中的值相加,获得%ecx 是 115，即第二个参数取 115.

所以第五题答案是：

5 115

Phase_6:

本题由函数可知需要输入六个整数。

根据汇编：

```
0x0000000000401117 <+43>:    mov    %r12,%rbp
0x000000000040111a <+46>:    mov    (%r12),%eax
0x000000000040111e <+50>:    sub    $0x1,%eax
0x0000000000401121 <+53>:    cmp    $0x5,%eax
0x0000000000401124 <+56>:    jbe    0x40112b <phase_6+63>
0x0000000000401126 <+58>:    callq  0x40141d <explode_bomb>
0x000000000040112b <+63>:    add    $0x1,%r13d
0x000000000040112f <+67>:    cmp    $0x6,%r13d
0x0000000000401133 <+71>:    je     0x401172 <phase_6+134>
0x0000000000401135 <+73>:    mov    %r13d,%ebx
0x0000000000401138 <+76>:    movslq %ebx,%rax
0x000000000040113b <+79>:    mov    (%rsp,%rax,4),%eax
0x000000000040113e <+82>:    cmp    %eax,0x0(%rbp)
0x0000000000401141 <+85>:    jne    0x401148 <phase_6+92>
-Type <return> to continue, or q <return> to quit---
0x0000000000401143 <+87>:    callq  0x40141d <explode_bomb>
0x0000000000401148 <+92>:    add    $0x1,%ebx
0x000000000040114b <+95>:    cmp    $0x5,%ebx
0x000000000040114e <+98>:    jle    0x401138 <phase_6+76>
0x0000000000401150 <+100>:   add    $0x4,%r12
0x0000000000401154 <+104>:   jmp    0x401117 <phase_6+43>
```

由这里的嵌套循环知：输入的六个参数各不相同，而且必须在 1~6 之间，所以可以确定六个参数为 1,2,3,4,5,6.但排列顺序尚未可知。

根据汇编：

```
0x0000000000401156 <+106>:   mov    0x8(%rdx),%rdx
0x000000000040115a <+110>:   add    $0x1,%eax
0x000000000040115d <+113>:   cmp    %ecx,%eax
0x000000000040115f <+115>:   jne    0x401156 <phase_6+106>
0x0000000000401161 <+117>:   mov    %rdx,0x20(%rsp,%rsi,2)
0x0000000000401166 <+122>:   add    $0x4,%rsi
0x000000000040116a <+126>:   cmp    $0x18,%rsi
0x000000000040116e <+130>:   jne    0x401177 <phase_6+139>
0x0000000000401170 <+132>:   jmp    0x40118b <phase_6+159>
0x0000000000401172 <+134>:   mov    $0x0,%esi
0x0000000000401177 <+139>:   mov    (%rsp,%rsi,1),%ecx
0x000000000040117a <+142>:   mov    $0x1,%eax
0x000000000040117f <+147>:   mov    $0x6032f0,%edx
0x0000000000401184 <+152>:   cmp    $0x1,%ecx
0x0000000000401187 <+155>:   jg     0x401156 <phase_6+106>
0x0000000000401189 <+157>:   jmp    0x401161 <phase_6+117>
```

知道每个参数会对应一个地址，并将地址存入栈中，但是由于参数先后次序不知，所以不知道栈中地址的先后顺序，通过这段汇编可以确定每个参数对应的地址，以及该地址所存储的值：

参数值	对应的地址（地址被存入栈）	地址对应的值
1	0x6032f0	817
2	0x603300	888
3	0x603310	548
4	0x603320	347
5	0x603330	826
6	0x603340	729

再根据汇编：

```
0x000000000040118b <+159>:   mov    0x20(%rsp),%rbx
0x0000000000401190 <+164>:   lea    0x20(%rsp),%rax
0x0000000000401195 <+169>:   lea    0x48(%rsp),%rsi
0x000000000040119a <+174>:   mov    %rbx,%rcx
0x000000000040119d <+177>:   mov    0x8(%rax),%rdx
Type <return> to continue, or q <return> to quit---
0x00000000004011a1 <+181>:   mov    %rdx,0x8(%rcx)
0x00000000004011a5 <+185>:   add    $0x8,%rax
0x00000000004011a9 <+189>:   mov    %rdx,%rcx
0x00000000004011ac <+192>:   cmp    %rsi,%rax
0x00000000004011af <+195>:   jne    0x40119d <phase_6+177>
0x00000000004011b1 <+197>:   movq   $0x0,0x8(%rdx)
0x00000000004011b9 <+205>:   mov    $0x5,%ebp
0x00000000004011be <+210>:   mov    0x8(%rbx),%rax
0x00000000004011c2 <+214>:   mov    (%rax),%eax
0x00000000004011c4 <+216>:   cmp    %eax,(%rbx)
0x00000000004011c6 <+218>:   jge    0x4011cd <phase_6+225>
0x00000000004011c8 <+220>:   callq  0x40141d <explode_bomb>
0x00000000004011cd <+225>:   mov    0x8(%rbx),%rbx
0x00000000004011d1 <+229>:   sub    $0x1,%ebp
0x00000000004011d4 <+232>:   jne    0x4011be <phase_6+210>
```

第 0x401119d 到 0x4011af 行是将栈值做一些变化，而接下来则是比较一下值的大小。这段汇编合起来的效果就是表明：参数的先后顺序是按照他们对应地址所对应的值的大小排序，顺序是从大到小。根据上面的表格可知：值从大到小为：888>826>817>729>548>347,所以对应参数顺序为：2,5,1,6,3,4。

所以第六题答案为：

2 5 1 6 3 4

Secret_phase:

其实原先无法进入这个题目，后面向大佬求教了一下，发现了进入这题的密码。

```
0x000000000040124c <+24>:    mov    %rax,%rbx
0x000000000040124f <+27>:    lea    -0x1(%rax),%eax
0x0000000000401252 <+30>:    cmp    $0x3e8,%eax
0x0000000000401257 <+35>:    jbe    0x40125e <secret_phase+42>
0x0000000000401259 <+37>:    callq  0x40141d <explode_bomb>
0x000000000040125e <+42>:    mov    %ebx,%esi
0x0000000000401260 <+44>:    mov    $0x603110,%edi
```

根据这段汇编可以知道,输入的参数在 1~1001 之间,且打印出地址 0x603110 的值为 36。

根据汇编：

```
0x0000000000401260 <+44>:    mov    $0x603110,%edi
0x0000000000401265 <+49>:    callq  0x4011f6 <fun7>
0x000000000040126a <+54>:    cmp    $0x2,%eax
```

在出 fun7 之后，返回值与 2 进行比较，所以知道，参数在进入 fun7 之后返回值为 2。

查看 fun7 汇编：

```
0x00000000004011f6 <+0>:     sub    $0x8,%rsp
0x00000000004011fa <+4>:     test   %rdi,%rdi
0x00000000004011fd <+7>:     je     0x40122a <fun7+52>
0x00000000004011ff <+9>:     mov    (%rdi),%edx
0x0000000000401201 <+11>:    cmp    %esi,%edx
0x0000000000401203 <+13>:    jle    0x401212 <fun7+28>
0x0000000000401205 <+15>:    mov    0x8(%rdi),%rdi
0x0000000000401209 <+19>:    callq  0x4011f6 <fun7>
0x000000000040120e <+24>:    add    %eax,%eax
0x0000000000401210 <+26>:    jmp    0x40122f <fun7+57>
0x0000000000401212 <+28>:    mov    $0x0,%eax
0x0000000000401217 <+33>:    cmp    %esi,%edx
0x0000000000401219 <+35>:    je     0x40122f <fun7+57>
0x000000000040121b <+37>:    mov    0x10(%rdi),%rdi
0x000000000040121f <+41>:    callq  0x4011f6 <fun7>
0x0000000000401224 <+46>:    lea    0x1(%rax,%rax,1),%eax
0x0000000000401228 <+50>:    jmp    0x40122f <fun7+57>
0x000000000040122a <+52>:    mov    $0xffffffff,%eax
0x000000000040122f <+57>:    add    $0x8,%rsp
0x0000000000401233 <+61>:    retq
```

发现是一个递归函数，第一次进入 fun7，由于返回值需要 2，所以第一次进入 fun7 之后，发现参数必须小于 36，且只能通过 0x401209 来调用 fun7，有可能使返回值为 2。且调用 fun7 之后，返回值乘以 2 输出，所以第二次调用 fun7 的返回值为 1

通过这种逆向分析,总共经过 3 次调用 fun7,发现参数为 22(地址 0x603170)时，符合条件。

所以 secret_phase 的答案为：

22

综上，答案整理为：

Phase_1: Houses will begat jobs, jobs will begat houses.
Phase_2: 1 2 4 7 11 16
Phase_3: 0 455
Phase_4: 99 3 DrEvil
Phase_5: 5 115
Phase_6: 2 5 1 6 3 4
Secret_phase: 22