计算机科学与技术学院

计算机系统原理课程实验报告

实验题目：拆炸弹	学号：202200130048
班级： 6	姓名：陈静雯
Email：1205037094@qq.com	
实验目的： 汇编语句	
实验软件和硬件环境： Vm+abuntu	

实验原理和方法：

1. 炸弹一：string_not_equal 为一个函数，判断输入与内置字符串是否相等

2. 炸弹二：六个数字，第一位固定为 1，后面的第 i 位（从第二位开始）为学号的倒数第 i-1 位 * 输入的第 i-1 位，所以 1, 1*8=8, 8*4=32, 32*0=0, 0*0=0, 0*3=0，所以六位数为 1，8，32，0，0，0

0x00400dbc <+0>: addiu sp,sp,-64

0x00400dc0 <+4>: sw ra,60(sp)

0x00400dc4 <+8>: sw s8,56(sp)

0x00400dc8 <+12>: move s8,sp

0x00400dcc <+16>: lui gp,0x42

0x00400dd0 <+20>: addiu gp,gp,-20080

0x00400dd4 <+24>: sw gp,16(sp)

0x00400dd8 <+28>: sw a0,64(s8)

0x00400ddc <+32>: addiu v0,s8,28

0x00400de0 <+36>: lw a0,64(s8)

0x00400de4 <+40>: move a1,v0

0x00400de8 <+44>: jal 0x401ba8 <read_six_numbers> // 读入 6 个数字

0x00400dec <+48>: nop

0x00400df0 <+52>: lw gp,16(s8)

0x00400df4 <+56>: lw v1,28(s8) // 读取第一个数

```
0x00400df8 <+60>: li v0,1                    // v0=1

0x00400dfc <+64>: beq v1,v0,0x400e10 <phase_2+84>  // 将 v0 与 v1 比较，如果相等则跳转
                                                    说明第一个数一定是 1
0x00400e00 <+68>: nop

0x00400e04 <+72>: jal 0x4021f0 <explode_bomb>

0x00400e08 <+76>: nop

0x00400e0c <+80>: lw gp,16(s8)

0x00400e10 <+84>: li v0,1                    //v0=1

0x00400e14 <+88>: sw v0,24(s8)

0x00400e18 <+92>: b 0x400ea8 <phase_2+236>

0x00400e1c <+96>: nop

0x00400e20 <+100>: lw v0,24(s8)

0x00400e24 <+104>: nop

0x00400e28 <+108>: addiu v0,v0,-1            // v0=v0-1

0x00400e2c <+112>: sll v0,v0,0x2         // 逻辑左移，把 v0 左移两位，

0x00400e30 <+116>: addiu v1,s8,24          // v1=【s8+24】，为储存循环变量 i 的地址

0x00400e34 <+120>: addu v0,v1,v0           // v0=v1+v0

0x00400e38 <+124>: lw a0,4(v0)           //a0=【v0+4】，将输入的第 i 个数传入 a0

0x00400e3c <+128>: li v1,12                //v1=12

0x00400e40 <+132>: lw v0,24(s8)          // 读入循环变量 i

0x00400e44 <+136>: nop

0x00400e48 <+140>: subu v0,v1,v0             //v0=v1-v0=12-i

0x00400e4c <+144>: lw v1,-32660(gp)          //读取输入的学号

0x00400e50 <+148>: sll v0,v0,0x2         //v0 左移两位

0x00400e54 <+152>: addu v0,v1,v0           //v0=【v1+v0】，即输入学号的倒数第 i 位

0x00400e58 <+156>: lw v0,0(v0)

0x00400e5c <+160>: nop

0x00400e60 <+164>: mult a0,v0              //把 a0*v0 的结果存入寄存器
```

0x00400e64 <+168>: mflo a0　　//将寄存器的值赋给 a0，即 a0=a0*v0

0x00400e68 <+172>: lw v0,24(s8)

0x00400e6c <+176>: nop

0x00400e70 <+180>: sll v0,v0,0x2

0x00400e74 <+184>: addiu v1,s8,24

0x00400e78 <+188>: addu v0,v1,v0　　// v0=v1-v0

0x00400e7c <+192>: lw v0,4(v0)　　//v0=v0+4，取得第 i+1 个数

0x00400e80 <+196>: nop

0x00400e84 <+200>: beq a0,v0,0x400e98 <phase_2+220>
//比较 a0 与 v0 的值是否相等
（a0=学号倒数第 i 位*输入的第 i 个数，v0 为输入的第 i+1 个数）

0x00400e88 <+204>: nop

0x00400e8c <+208>: jal 0x4021f0 <explode_bomb>

0x00400e90 <+212>: nop

0x00400e94 <+216>: lw gp,16(s8)

0x00400e98 <+220>: lw v0,24(s8)

0x00400e9c <+224>: nop

0x00400ea0 <+228>: addiu v0,v0,1

0x00400ea4 <+232>: sw v0,24(s8)

0x00400ea8 <+236>: lw v0,24(s8)　//v0=内存【s8+24】位置的值，即目前的循环变量 i 的值

0x00400eac <+240>: nop

0x00400eb0 <+244>: slti v0,v0,6
// 比较，如果 v0<6,则 v0=1，如果 v0>=6，则 v0=0,循环结束

0x00400eb4 <+248>: bnez v0,0x400e20 <phase_2+100>　// v0 不为 0 则跳转到<phase_2+100>

0x00400eb8 <+252>: nop

0x00400ebc <+256>: move sp,s8

0x00400ec0 <+260>: lw ra,60(sp)

0x00400ec4 <+264>: lw s8,56(sp)

0x00400ec8 <+268>: addiu sp,sp,64

0x00400ecc <+272>: jr ra

0x00400ed0 <+276>: nop

3.炸弹三：是 switch-case 函数，不同的第一位数对应不同的 case，要选择与学号最后一位能整除的数段，不然会爆炸

0x00400ed4 <+0>: addiu sp,sp,-56

0x00400ed8 <+4>: sw ra,52(sp)

0x00400edc <+8>: sw s8,48(sp)

0x00400ee0 <+12>:move s8,sp

0x00400ee4 <+16>:lui gp,0x42

0x00400ee8 <+20>:addiu gp,gp,-20080

0x00400eec <+24>:sw gp,24(sp)

0x00400ef0 <+28>: sw a0,56(s8)

0x00400ef4 <+32>: lw a0,56(s8)

0x00400ef8 <+36>: lui v0,0x40

0x00400efc <+40>: addiu a1,v0,10112

0x00400f00 <+44>: addiu v1,s8,44

0x00400f04 <+48>: addiu v0,s8,40

0x00400f08 <+52>: addiu a2,s8,36

0x00400f0c <+56>: sw a2,16(sp)

0x00400f10 <+60>: move a2,v1

0x00400f14 <+64>: move a3,v0

0x00400f18 <+68>: lw v0,-32636(gp)

0x00400f1c <+72>: nop

0x00400f20 <+76>: move t9,v0

0x00400f24 <+80>: jalr t9 //返回输入参数个数

0x00400f28 <+84>: nop

0x00400f2c <+88>: lw gp,24(s8)

0x00400f30 <+92>: slti v0,v0,3
//v0<3, v0=1 , 否则 v0=0，即输入的参数>=3,经查看$a1 可知输入应为”%d %c %d”

0x00400f34 <+96>: beqz v0,0x400f48 <phase_3+116>

0x00400f38 <+100>: nop

0x00400f3c <+104>: jal 0x4021f0 <explode_bomb>

0x00400f40 <+108>: nop

0x00400f44 <+112>: lw gp,24(s8)

0x00400f48 <+116>: lw v0,44(s8) //输入的第一个数

0x00400f4c <+120>: nop

0x00400f50 <+124>: sltiu v1,v0,8
// v0<8,v1=1; v0>=8,v1=0 即输入的第一个数应小于 8 否则爆炸

0x00400f54 <+128>: beqz v1,0x401190 <phase_3+700>

0x00400f58 <+132>:nop

0x00400f5c <+136>: sll v1,v0,0x2

0x00400f60 <+140>: lui v0,0x40

0x00400f64 <+144>: addiu v0,v0,10124 //四行实现第一个数向某个地址的转换

0x00400f68 <+148>: addu v0,v1,v0

0x00400f6c <+152>: lw v0,0(v0)

0x00400f70 <+156>: nop

0x00400f74 <+160>: jr v0 //跳转到寄存器 v0 中的地址

0x00400f78 <+164>: nop

0x00400f7c <+168>: li v0,113 //第一个参数为 0

0x00400f80 <+172>: sb v0,32(s8) //v0 为应该输入的第二个参数，存入 s8+32

0x00400f84 <+176>: lw v0,-32660(gp) //学号

0x00400f88 <+180>: nop

0x00400f8c <+184>: lw v1,44(v0) //学号最后一位数字（8）

0x00400f90 <+188>: lw v0,36(s8) //v0 输入的第三个参数

```
0x00400f94 <+192>: nop

0x00400f98 <+196>: mult  v1,v0

0x00400f9c <+200>: mflo v1

0x00400fa0 <+204>: li v0,777                    //v1=v0 即学号的最后一位*第三个参数 = 777

0x00400fa4 <+208>: beq v1,v0,0x4011ac <phase_3+728>

0x00400fa8 <+212>: nop

0x00400fac <+216>: jal   0x4021f0 <explode_bomb>

0x00400fb0 <+220>: nop

0x00400fb4 <+224>: lw  gp,24(s8)

0x00400fb8 <+228>: b 0x4011f8 <phase_3+804>

0x00400fbc <+232>: nop

0x00400fc0 <+236>: li v0,98                          //第一个参数为 1

0x00400fc4 <+240>: sb v0,32(s8)

0x00400fc8 <+244>: lw v0,-32660(gp)

0x00400fcc <+248>: nop

0x00400fd0 <+252>: lw  v1,44(v0)

0x00400fd4 <+256>: lw  v0,36(s8)

0x00400fd8 <+260>: nop

0x00400fdc <+264>: mult  v1,v0

0x00400fe0 <+268>: mflo  v1

0x00400fe4 <+272>: li  v0,214

0x00400fe8 <+276>:  beq  v1,v0,0x4011b8 <phase_3+740>

0x00400fec <+280>: nop

0x00400ff0 <+284>: jal  0x4021f0 <explode_bomb>

0x00400ff4 <+288>: nop

0x00400ff8 <+292>: lw  gp,24(s8)

0x00400ffc <+296>:b 0x4011f8 <phase_3+804>
```

```
0x00401000 <+300>: nop
0x00401004 <+304>: li  v0,98                    //第一个参数为 2
0x00401008 <+308>: sb  v0,32(s8)
0x0040100c <+312>: lw  v0,-32660(gp)
0x00401010 <+316>: nop
0x00401014 <+320>: lw  v1,44(v0)
0x00401018 <+324>: lw  v0,36(s8)
0x0040101c <+328>: nop
0x00401020 <+332>: mult v1,v0
0x00401024 <+336>: mflo v1
0x00401028 <+340>: li   v0,755
0x0040102c <+344>: beq v1,v0,0x4011c4 <phase_3+752>
0x00401030 <+348>: nop
0x00401034 <+352>: jal 0x4021f0 <explode_bomb>
0x00401038 <+356>: nop
0x0040103c <+360>: lw  gp,24(s8)
0x00401040 <+364>: b 0x4011f8 <phase_3+804>
0x00401044 <+368>: nop
0x00401048 <+372>: li v0,107                    //第一个参数为 3
0x0040104c <+376>: sb  v0,32(s8)
0x00401050 <+380>: lw  v0,-32660(gp)
0x00401054 <+384>: nop
0x00401058 <+388>: lw  v1,44(v0)
0x0040105c <+392>: lw  v0,36(s8)
0x00401060 <+396>: nop
0x00401064 <+400>: mult v1,v0
0x00401068 <+404>: mflo v0
```

```
0x0040106c <+408>: beqz v0,0x4011d0 <phase_3+764>      //相乘为 0
0x00401070 <+412>:  nop
0x00401074 <+416>:  jal 0x4021f0 <explode_bomb>
0x00401078 <+420>: nop
0x0040107c <+424>:  lw gp,24(s8)
0x00401080 <+428>: b  0x4011f8 <phase_3+804>
0x00401084 <+432>: nop
0x00401088 <+436>: li  v0,111                          //第一个参数为 4
0x0040108c <+440>: sb  v0,32(s8)
0x00401090 <+444>: lw  v0,-32660(gp)
0x00401094 <+448>:  nop
0x00401098 <+452>: lw  v1,44(v0)
0x0040109c <+456>: lw  v0,36(s8)
0x004010a0 <+460>: nop
0x004010a4 <+464>: mult  v1,v0
0x004010a8 <+468>: mflo  v1
0x004010ac <+472>: li  v0,228
0x004010b0 <+476>: beq v1,v0,0x4011dc <phase_3+776>
0x004010b4 <+480>: nop
0x004010b8 <+484>: jal 0x4021f0 <explode_bomb>
0x004010bc <+488>: nop
0x004010c0 <+492>: lw  gp,24(s8)
0x004010c4 <+496>:  b  0x4011f8 <phase_3+804>
0x004010c8 <+500>: nop
0x004010cc <+504>: li  v0,116                          //第一个参数为 5
0x004010d0 <+508>: sb  v0,32(s8)
0x004010d4 <+512>: lw  v0,-32660(gp)
```

```
0x004010d8 <+516>: nop

0x004010dc <+520>:  lw  v1,44(v0)

0x004010e0 <+524>: lw  v0,36(s8)

0x004010e4 <+528>: nop

0x004010e8 <+532>: mult  v1,v0

0x004010ec <+536>: mflo v1

0x004010f0 <+540>: li v0,513

0x004010f4 <+544>: beq v1,v0,0x4011e8 <phase_3+788>

0x004010f8 <+548>: nop

0x004010fc <+552>: jal  0x4021f0 <explode_bomb>

0x00401100 <+556>: nop

0x00401104 <+560>: lw  gp,24(s8)

0x00401108 <+564>: b 0x4011f8 <phase_3+804>

0x0040110c <+568>: nop

0x00401110 <+572>: li  v0,118                    //第一个参数为 6

0x00401114 <+576>: sb  v0,32(s8)

0x00401118 <+580>: lw  v0,-32660(gp)

0x0040111c <+584>: nop

0x00401120 <+588>:  lwv1,44(v0)

0x00401124 <+592>: lw  v0,36(s8)

0x00401128 <+596>: nop

0x0040112c <+600>: mult v1,v0

0x00401130 <+604>: mflo v1

0x00401134 <+608>: li   v0,780

0x00401138 <+612>: beq v1,v0,0x40114c <phase_3+632>

0x0040113c <+616>: nop

0x00401140 <+620>: jal 0x4021f0 <explode_bomb>
```

```
0x00401144 <+624>: nop

0x00401148 <+628>: lw gp,24(s8)

0x0040114c <+632>: li v0,98                    //第一个参数为 7

0x00401150 <+636>: sb v0,32(s8)

0x00401154 <+640>: lw v0,-32660(gp)

0x00401158 <+644>: nop

0x0040115c <+648>: lw v1,44(v0)

0x00401160 <+652>: lw v0,36(s8)

0x00401164 <+656>: nop

0x00401168 <+660>: mult v1,v0

0x0040116c <+664>: mflo v1

0x00401170 <+668>: li v0,824

0x00401174 <+672>: beq v1,v0,0x4011f4 <phase_3+800>

0x00401178 <+676>: nop

0x0040117c <+680>: jal 0x4021f0 <explode_bomb>

0x00401180 <+684>: nop

0x00401184 <+688>: lw gp,24(s8)

0x00401188 <+692>: b 0x4011f8 <phase_3+804>

0x0040118c <+696>: nop

0x00401190 <+700>: li v0,120

0x00401194 <+704>: sb v0,32(s8)

0x00401198 <+708>: jal 0x4021f0 <explode_bomb>

0x0040119c <+712>: nop

0x004011a0 <+716>: lw gp,24(s8)

0x004011a4 <+720>: b 0x4011f8 <phase_3+804>

0x004011a8 <+724>: nop

0x004011ac <+728>: nop
```

```
0x004011b0 <+732>: b 0x4011f8 <phase_3+804>

0x004011b4 <+736>: nop

0x004011b8 <+740>: nop

0x004011bc <+744>: b 0x4011f8 <phase_3+804>

0x004011c0 <+748>: nop

0x004011c4 <+752>: nop

0x004011c8 <+756>: b 0x4011f8 <phase_3+804>

0x004011cc <+760>: nop

0x004011d0 <+764>: nop

0x004011d4 <+768>: b 0x4011f8 <phase_3+804>

0x004011d8 <+772>: nop

0x004011dc <+776>:  nop

0x004011e0 <+780>: b 0x4011f8 <phase_3+804>

0x004011e4 <+784>: nop

0x004011e8 <+788>: nop

0x004011ec <+792>: b 0x4011f8 <phase_3+804>

0x004011f0 <+796>: nop

0x004011f4 <+800>: nop

0x004011f8 <+804>: lb v0,40(s8)                    //取出输入的第二个参数

0x004011fc <+808>: lb v1,32(s8)

0x00401200 <+812>:  nop

0x00401204 <+816>:  beq  v1,v0,0x401218 <phase_3+836>       //v0 = v1，否则爆炸

0x00401208 <+820>: nop

0x0040120c <+824>: jal  0x4021f0 <explode_bomb>

0x00401210 <+828>: nop

0x00401214 <+832>: lw  gp,24(s8)

0x00401218 <+836>: move sp,s8
```

0x0040121c <+840>: lw ra,52(sp)

0x00401220 <+844>: lw s8,48(sp)

0x00401224 <+848>: addiu sp,sp,56

0x00401228 <+852>: jr ra

0x0040122c <+856>: nop

4.炸弹四：递归+斐波那契数列, f（6）= 13

0x004012bc <+0>: addiu sp,sp,-40

0x004012c0 <+4>: sw ra,36(sp)

0x004012c4 <+8>: sw s8,32(sp)

0x004012c8 <+12>: move s8,sp

0x004012cc <+16>: lui gp,0x42

0x004012d0 <+20>: addiu gp,gp,-20080

0x004012d4 <+24>: sw gp,16(sp)

0x004012d8 <+28>: sw a0,40(s8)

0x004012dc <+32>: lw v1,40(s8)

0x004012e0 <+36>: lui v0,0x40

0x004012e4 <+40>: addiu v0,v0,10156

0x004012e8 <+44>: move a0,v1

0x004012ec <+48>: move a1,v0

0x004012f0 <+52>: addiu v0,s8,24

0x004012f4 <+56>: move a2,v0

0x004012f8 <+60>: lw v0,-32636(gp) // 调用子程序输入 v0

0x004012fc <+64>: nop

0x00401300 <+68>: move t9,v0 #t9=v0

0x00401304 <+72>: jalr t9
// 跳转到 t9 并连接，计算输入参数个数并把结果存入 v0

0x00401308 <+76>: nop

0x0040130c <+80>: lw gp,16(s8)

0x00401310 <+84>: move v1,v0　　// v1=v0

0x00401314 <+88>: li v0,1　　//v0=1

0x00401318 <+92>: bne v1,v0,0x401330 <phase_4+116>
// v1！=v0 则跳转，判断 v1 是否为 1，不为 1 炸弹爆炸，因此 v1 应为 1，即输入参数的个数为 1

0x0040131c <+96>: nop

0x00401320 <+100>: lw v0,24(s8)

0x00401324 <+104>: nop

0x00401328 <+108>: bgtz v0,0x401340 <phase_4+132>
// v0>0 则跳转，【s8+24】里储存的是输入的值，可知输入的参数大小应大于 0

0x0040132c <+112>: nop

0x00401330 <+116>: jal 0x4021f0 <explode_bomb>

0x00401334 <+120>: nop

0x00401338 <+124>: lw gp,16(s8)

0x0040133c <+128>: nop

0x00401340 <+132>: lw v0,-32660(gp)　// 学号

0x00401344 <+136>: nop

0x00401348 <+140>: lw v0,44(v0)　// 学号最后一位 v0=8

0x0040134c <+144>: nop

0x00401350 <+148>: andi v0,v0,0x1　　// v0=0，$8_{(10)}=1000_{(2)}$，即 1000 与 0001 进行按位与，结果为 0000，所以学号最后一位是奇数，v0=1；学号最后一位是偶数，v0=0

0x00401354 <+152>: andi v0,v0,0xff #v0=0，

0x00401358 <+156>: beqz v0,0x40139c <phase_4+224>　//v0=0，学号最后一位是偶数，跳转

0x0040135c <+160>: nop

0x00401360 <+164>: lw v0,24(s8)

0x00401364 <+168>: nop

0x00401368 <+172>: move a0,v0

```
0x0040136c <+176>: jal 0x401230 <func4>       //进入函数

0x00401370 <+180>: nop

0x00401374 <+184>: lw gp,16(s8)

0x00401378 <+188>: move v1,v0

0x0040137c <+192>: li v0,8                    //学号最后一位奇数，则要使 f（输入）= 8

0x00401380 <+196>: beq v1,v0,0x4013d0 <phase_4+276>

0x00401384 <+200>: nop

0x00401388 <+204>: jal 0x4021f0 <explode_bomb>

0x0040138c <+208>: nop

0x00401390 <+212>: lw gp,16(s8)

0x00401394 <+216>: b 0x4013d0 <phase_4+276>

0x00401398 <+220>: nop

0x0040139c <+224>: lw v0,24(s8)          //v0 为偶数，跳到这一行，v0 为输入的参数

0x004013a0 <+228>: nop

0x004013a4 <+232>: move a0,v0            //a0=v0，此时 a0 为输入的参数

0x004013a8 <+236>: jal 0x401230 <func4>         // 跳转到<func4>

0x004013ac <+240>: nop

0x004013b0 <+244>: lw gp,16(s8)

0x004013b4 <+248>: move v1,v0              // v1=v0

0x004013b8 <+252>: li v0,13                    // v0=13

0x004013bc <+256>: beq v1,v0,0x4013d0 <phase_4+276>   // v1 和 v0 相等就跳转至程序结束

0x004013c0 <+260>: nop

0x004013c4 <+264>: jal 0x4021f0 <explode_bomb>

0x004013c8 <+268>: nop

0x004013cc <+272>: lw gp,16(s8)

0x004013d0 <+276>: move sp,s8

0x004013d4 <+280>: lw ra,36(sp)
```

0x004013d8 <+284>: lw s8,32(sp)

0x004013dc <+288>: addiu sp,sp,40

0x004013e0 <+292>: jr ra

0x004013e4 <+296>: nop

Dump of assembler code for function func4: // 斐波那契数列

0x00401234 <+4>: sw ra,36(sp)

0x00401238 <+8>: sw s8,32(sp)

0x0040123c <+12>: sw s0,28(sp)

0x00401240 <+16>: move s8,sp

0x00401244 <+20>: sw a0,40(s8) // a0，存传入的数

0x00401248 <+24>: lw v0,40(s8) // v0，取出传入的数

0x0040124c <+28>: nop

0x00401250 <+32>: slti v0,v0,2 // 若 v0<=1, v0=1，若 v0>=2,v0=0
即直到传入数值<=1 时，v0=1

0x00401254 <+36>: bnez v0,0x40129c <func4+108>
//v0=1 时跳至<func4+108>，v0！=1 时继续运行

0x00401258 <+40>: nop

0x0040125c <+44>: lw v0,40(s8)

0x00401260 <+48>: nop

0x00401264 <+52>: addiu v0,v0,-1 //v0=v0-1

0x00401268 <+56>: move a0,v0 // a0=v0

0x0040126c <+60>: jal 0x401230 <func4> //调用 func(v0-1)

0x00401270 <+64>: nop

0x00401274 <+68>: move s0,v0 //f（v0-1）递归结束回到这一行，把函数结果存入 s0

0x00401278 <+72>: lw v0,40(s8)

0x0040127c <+76>: nop

0x00401280 <+80>: addiu v0,v0,-2 //v0=v0-2

0x00401284 <+84>: move a0,v0

0x00401288 <+88>: jal 0x401230 <func4>　　//调用 func（v0-2）

0x0040128c <+92>: nop

0x00401290 <+96>: addu v0,s0,v0　　// v0=s0+v0　即 f（v0-1）+f（v0-2）得到 f（v0）

0x00401294 <+100>: b 0x4012a0 <func4+112>　　//跳转

0x00401298 <+104>: nop

0x0040129c <+108>: li v0,1　　// v0<=1 时跳转到这里，即 f（1）= 0，f（0）= 0

0x004012a0 <+112>: move sp,s8

0x004012a4 <+116>: lw ra,36(sp)

0x004012a8 <+120>: lw s8,32(sp)

0x004012ac <+124>: lw s0,28(sp)

0x004012b0 <+128>: addiu sp,sp,40

0x004012b4 <+132>: jr ra

0x004012b8 <+136>: nop

5. 炸弹五：通过内置字符串，找到对应字符串使得映射后得到“giants”

0x004013e8 <+0>: addiu sp,sp,-72

0x004013ec <+4>: sw ra,68(sp)

0x004013f0 <+8>: sw s8,64(sp)

0x004013f4 <+12>: move s8,sp

0x004013f8 <+16>: sw a0,72(s8)

0x004013fc <+20>: lw a0,72(s8)　//a0 为输入的字符串

0x00401400 <+24>: jal 0x401c78 <string_length>　// 判断输入字符串的长度，返回值放在 v0

0x00401404 <+28>: nop

0x00401408 <+32>: move v1,v0　　//v1=v0=字符串的长度

0x0040140c <+36>: li v0,6

0x00401410 <+40>: beq v1,v0,0x401420 <phase_5+56>
// v1 和 v0 相等就跳转，否则炸弹爆炸，输入的字符串的长度必须为 6

0x00401414 <+44>: nop

0x00401418 <+48>: jal 0x4021f0 <explode_bomb>

0x0040141c <+52>: nop

0x00401420 <+56>: sw zero,24(s8)　　// 初始化循环变量 i=0

0x00401424 <+60>: b 0x4014a8 <phase_5+192>

0x00401428 <+64>: nop

0x0040142c <+68>: lw v0,24(s8)

0x00401430 <+72>: lw v1,24(s8)　　//v1=v0=i

0x00401434 <+76>: lw a0,72(s8)　　//a0 输入的字符串

0x00401438 <+80>: nop

0x0040143c <+84>: addu v1,a0,v1　// v1 输入的字符串的第 i+1 个字母 (1,2,3,……)

0x00401440 <+88>: lb v1,0(v1)

0x00401444 <+92>: nop

0x00401448 <+96>: andi v1,v1,0xff　　//按位与：某数&11111111=某数

0x0040144c <+100>: andi v1,v1,0xf　　//按位与：某数&00001111=后四位

0x00401450 <+104>: sll v0,v0,0x2　　//左移 2 位

0x00401454 <+108>: addiu a0,s8,24　　//a0 = i 循环变量

0x00401458 <+112>: addu v0,a0,v0　　//v0=v0+a0，从这个地址后移 4 位

0x0040145c <+116>: sw v1,12(v0)
//【v0+12】=v1，把取出字母的二进制后四位放在【v0+12】即【s8+36】中

0x00401460 <+120>: lw a0,24(s8)

0x00401464 <+124>: lw v0,24(s8)

0x00401468 <+128>: nop

0x0040146c <+132>: sll v0,v0,0x2

0x00401470 <+136>: addiu v1,s8,24　　//v1 为 i 的地址

0x00401474 <+140>: addu v0,v1,v0　　//v0=v1+v0

0x00401478 <+144>: lw v1,12(v0)　// v1 为刚刚取得的字母的 ASCII 码二进制的后四位

0x0040147c <+148>: lui v0,0x41

0x00401480 <+152>: addiu v0,v0,12524 //v0=v0+12524 ,内置字符串的开头，断点查看内置字符串为 isrveawhobpnutfg

0x00401484 <+156>: addu v0,v1,v0
// 从开头地址往后移动 v0 位，v0 为目前取得的字母的 ascii 码二进制的后四位，比如第一个输入的字母为 a，后四位为 0001，即十进制的 1，则 v0 即为内置字符串第 2 个字母开始的位置

0x00401488 <+160>: lb v1,0(v0) // v1=v0，内置字符串相应字母的首位

0x0040148c <+164>: addiu v0,s8,24

0x00401490 <+168>: addu v0,v0,a0 //v0=v0+a0，a0 为之前的循环变量 i，即 v0=s8+24+i

0x00401494 <+172>: sb v1,4(v0)
//【v0+4】=v1，取对应的一个字母存入【v0+4】即【s8+28+i】的位置

0x00401498 <+176>: lw v0,24(s8)

0x0040149c <+180>: nop

0x004014a0 <+184>: addiu v0,v0,1 //v0++

0x004014a4 <+188>: sw v0,24(s8)

0x004014a8 <+192>: lw v0,24(s8)

0x004014ac <+196>: nop

0x004014b0 <+200>: slti v0,v0,6 //若 v0<6,v0=1，循环

0x004014b4 <+204>: bnez v0,0x40142c <phase_5+68> //若 v0=1，跳转，若 v0=0，循环结束

0x004014b8 <+208>: nop

0x004014bc <+212>: sb zero,34(s8)
//【s8+34】=0，从【s8+28】到【s8+33】分别为输入的字母映射后的内置字符串的六个字母，然后在【s8+34】即字符串的末尾置空

0x004014c0 <+216>: addiu v0,s8,28 // v0=s8+28，取得映射后的六个内置字符串的字母

0x004014c4 <+220>: move a0,v0

0x004014c8 <+224>: lui v0,0x40

0x004014cc <+228>: addiu a1,v0,10160

0x004014d0 <+232>: jal 0x401cf8 <strings_not_equal>
//映射后的六个内置字符串的字母与 a1 中的字母比较，经查看可知 a1 中的字符串为 giants

0x004014d4 <+236>: nop

0x004014d8 <+240>: beqz v0,0x4014e8 <phase_5+256>

0x004014dc <+244>: nop

0x004014e0 <+248>: jal 0x4021f0 <explode_bomb>

0x004014e4 <+252>: nop

0x004014e8 <+256>: move sp,s8

0x004014ec <+260>: lw ra,68(sp)

0x004014f0 <+264>: lw s8,64(sp)

0x004014f4 <+268>: addiu sp,sp,72

0x004014f8 <+272>: jr ra

0x004014fc <+276>: nop

6. 炸弹六：链表的重排列

0x00401500 <+0>: addiu sp,sp,-96

0x00401504 <+4>: sw ra,92(sp)

0x00401508 <+8>: sw s8,88(sp)

0x0040150c <+12>: move s8,sp

0x00401510 <+16>: lui gp,0x42

0x00401514 <+20>: addiu gp,gp,-20080

0x00401518 <+24>: sw gp,16(sp)

0x0040151c <+28>: sw a0,96(s8)

0x00401520 <+32>: lui v0,0x41

0x00401524 <+36>: addiu v0,v0,12592

0x00401528 <+40>: sw v0,32(s8)

0x0040152c <+44>: addiu v0,s8,36

0x00401530 <+48>: lw a0,96(s8)

0x00401534 <+52>: move a1,v0

0x00401538 <+56>: jal 0x401ba8 <read_six_numbers> // 读取 6 个数字，返回值存在 v0

0x0040153c <+60>: nop

0x00401540 <+64>: lw gp,16(s8) // 第一重循环

0x00401544 <+68>: sw zero,28(s8) //设置循环变量 i=0

0x00401548 <+72>: b 0x40163c <phase_6+316> //第一个循环的条件判断

0x0040154c <+76>: nop

循环判断输入是否符合条件[1, 6]：

0x00401550 <+80>: lw v0,28(s8) // v0=i

0x00401554 <+84>: nop

0x00401558 <+88>: sll v0,v0,0x2

0x0040155c <+92>: addiu v1,s8,24

0x00401560 <+96>: addu v0,v1,v0

0x00401564 <+100>: lw v0,12(v0) //v0 为取出输入的第 i 个数

0x00401568 <+104>: nop

0x0040156c <+108>: slti v0,v0,7 //v0>=7，炸弹爆炸，可知输入的六个数字<7

0x00401570 <+112>: beqz v0,0x40159c <phase_6+156>

0x00401574 <+116>: nop

0x00401578 <+120>: lw v0,28(s8) // v0=i

0x0040157c <+124>: nop

0x00401580 <+128>: sll v0,v0,0x2

0x00401584 <+132>: addiu v1,s8,24

0x00401588 <+136>: addu v0,v1,v0

0x0040158c <+140>: lw v0,12(v0)

0x00401590 <+144>: nop

0x00401594 <+148>: bgtz v0,0x4015a8 <phase_6+168>
// v0>0,则跳转,否则炸弹爆炸,可知输入的六个数字>0

0x00401598 <+152>: nop

0x0040159c <+156>: jal 0x4021f0 <explode_bomb>

0x004015a0 <+160>: nop

0x004015a4 <+164>: lw gp,16(s8)

0x004015a8 <+168>: lw v0,28(s8) // v0=i

0x004015ac <+172>: nop

0x004015b0 <+176>: addiu v0,v0,1 //v0=v0+1，设定第二重循环变量 j=i+1

0x004015b4 <+180>: sw v0,24(s8) //【s8+24】=v0=j，即 i 的值存在【s8+28】，j 的值【s8+24】

0x004015b8 <+184>: b 0x401618 <phase_6+280> //第二重循环的条件判断：j<6

0x004015bc <+188>: nop

循环判断两个数字不能重复：

0x004015c0 <+192>: lw v0,28(s8) // v0=i

0x004015c4 <+196>: nop

0x004015c8 <+200>: sll v0,v0,0x2

0x004015cc <+204>: addiu v1,s8,24 // v1=j

0x004015d0 <+208>: addu v0,v1,v0 // v0=s8+24+i

0x004015d4 <+212>: lw v1,12(v0) // v1 为取得输入的第 i 个数字

0x004015d8 <+216>: lw v0,24(s8) // v0=j

0x004015dc <+220>: nop

0x004015e0 <+224>: sll v0,v0,0x2

0x004015e4 <+228>: addiu a0,s8,24

0x004015e8 <+232>: addu v0,a0,v0

0x004015ec <+236>: lw v0,12(v0) // 取得第 j 个数字

0x004015f0 <+240>: nop

0x004015f4 <+244>: bne v1,v0,0x401608 <phase_6+264>
//比较 v1 和 v0 的值，相等就爆炸，可知前后两个数字不能重复

0x004015f8 <+248>: nop

0x004015fc <+252>: jal 0x4021f0 <explode_bomb>

0x00401600 <+256>: nop

0x00401604 <+260>: lw gp,16(s8)

0x00401608 <+264>: lw v0,24(s8) // v0=j

0x0040160c <+268>: nop

0x00401610 <+272>: addiu v0,v0,1 // j++

0x00401614 <+276>: sw v0,24(s8)

0x00401618 <+280>: lw v0,24(s8) // 取出第二重循环变量 j

0x0040161c <+284>: nop

0x00401620 <+288>: slti v0,v0,6 //j<6，v0=1，否则 v0=0

0x00401624 <+292>: bnez v0,0x4015c0 <phase_6+192> // v0=1 即 j<6 时跳转

0x00401628 <+296>: nop

0x0040162c <+300>: lw v0,28(s8)

0x00401630 <+304>: nop

0x00401634 <+308>: addiu v0,v0,1

0x00401638 <+312>: sw v0,28(s8)

0x0040163c <+316>: lw v0,28(s8)

0x00401640 <+320>: nop

0x00401644 <+324>: slti v0,v0,6 // v0<6，则 v0=1

0x00401648 <+328>: bnez v0,0x401550 <phase_6+80> //v0 不等于 0 就继续循环

0x0040164c <+332>: nop

0x00401650 <+336>: sw zero,28(s8) // i=0

0x00401654 <+340>: b 0x4016f8 <phase_6+504>

0x00401658 <+344>: nop

循环，将链表按输入的顺序存入新的数组：
0x0040165c <+348>: lui v0,0x41

0x00401660 <+352>: addiu v0,v0,12592

0x00401664 <+356>: sw v0,32(s8)
//【s8+32】=v0，给 v0 在【s8+32】新建一个数组，把链表第一个数放到数组第一个位置

0x00401668 <+360>: li v0,1 // 设置第二重循环变量 j=1

0x0040166c <+364>: sw v0,24(s8) //【s8+24】=v0=1，即 j 放在【s8+24】里

0x00401670 <+368>: b 0x40169c <phase_6+412> // 跳转 j 条件判断

0x00401674 <+372>: nop

0x00401678 <+376>: lw v0,32(s8)

0x0040167c <+380>: nop

0x00401680 <+384>: lw v0,8(v0) // v0=【v0+8】

0x00401684 <+388>: nop

0x00401688 <+392>: sw v0,32(s8) // 【s8+32+8】= v0

0x0040168c <+396>: lw v0,24(s8) //v0=【s8+24】，取得第二重循环变量 j

0x00401690 <+400>: nop

0x00401694 <+404>: addiu v0,v0,1 // j++

0x00401698 <+408>: sw v0,24(s8) // 【s8+24】=v0，更新 j 的新值

0x0040169c <+412>: lw v0,28(s8)

0x004016a0 <+416>: nop

0x004016a4 <+420>: sll v0,v0,0x2

0x004016a8 <+424>: addiu v1,s8,24 //v1=j

0x004016ac <+428>: addu v0,v1,v0 // v0=s8+24+j

0x004016b0 <+432>: lw v1,12(v0) // v1=【v0+12】，取得输入的第 i 个数

0x004016b4 <+436>: lw v0,24(s8)

0x004016b8 <+440>: nop

0x004016bc <+444>: slt v0,v0,v1 // v0<v1, v0=1

0x004016c0 <+448>: bnez v0,0x401678 <phase_6+376>
//若 j< 输入的第 i 个数，则继续第二重循环，直到 j=输入的第 i 个数，向下运行，即找到链表中应该排在第 i 的节点

0x004016c4 <+452>: nop

0x004016c8 <+456>: lw v0,28(s8) #取得第一重循环变量 i

0x004016cc <+460>: nop

0x004016d0 <+464>: sll v0,v0,0x2

0x004016d4 <+468>: addiu v1,s8,24

0x004016d8 <+472>: addu v0,v1,v0

0x004016dc <+476>: lw v1,32(s8) // v1= 新建数组的首地址

0x004016e0 <+480>: nop

0x004016e4 <+484>: sw v1,36(v0) // 【v0+36】=v1，把节点值装入新开辟数组首地址的下一个位置

0x004016e8 <+488>: lw v0,28(s8)

0x004016ec <+492>: nop

0x004016f0 <+496>: addiu v0,v0,1 // i++

0x004016f4 <+500>: sw v0,28(s8)

0x004016f8 <+504>: lw v0,28(s8)

0x004016fc <+508>: nop

0x00401700 <+512>: slti v0,v0,6

0x00401704 <+516>: bnez v0,0x40165c <phase_6+348> // i<6 继续循环，否则跳出循环

0x00401708 <+520>: nop

0x0040170c <+524>: lw v0,60(s8)

0x00401710 <+528>: nop

0x00401714 <+532>: sw v0,32(s8)

0x00401718 <+536>: li v0,1

0x0040171c <+540>: sw v0,28(s8) // 【s8+28】=v0，即循环变量储存在【s8+28】里

0x00401720 <+544>: b 0x40177c <phase_6+636>

0x00401724 <+548>: nop

循环；
0x00401728 <+552>: lw v0,28(s8) // v0=i

0x0040172c <+556>: nop

0x00401730 <+560>: sll v0,v0,0x2

```
0x00401734 <+564>: addiu v1,s8,24                    // v1=s8+24

0x00401738 <+568>: addu v0,v1,v0                     // v0=s8+24+i*4

 0x0040173c <+572>: lw v1,36(v0)
// v1=【s8+24+i*4+36】得到新链表第 i 个节点的值，第一次时这里是第二个节点，因为i=1，第一个节点为【0】

0x00401740 <+576>: lw v0,32(s8)                 // v0=【s8+32】第一次中这是第一个节点

 0x00401744 <+580>: nop

0x00401748 <+584>: sw v1,8(v0)

0x0040174c <+588>: lw v0,28(s8)

0x00401750 <+592>: nop

0x00401754 <+596>: sll v0,v0,0x2

0x00401758 <+600>: addiu v1,s8,24

0x0040175c <+604>: addu v0,v1,v0

0x00401760 <+608>: lw v0,36(v0)                 // v0 第 i 个节点

 0x00401764 <+612>: nop

0x00401768 <+616>: sw v0,32(s8)                  // 【s8+32】=v0，储存当前节点

0x0040176c <+620>: lw v0,28(s8)                 // v0=i

0x00401770 <+624>: nop

0x00401774 <+628>: addiu v0,v0,1                // i++

0x00401778 <+632>: sw v0,28(s8)

0x0040177c <+636>: lw v0,28(s8)

0x00401780 <+640>: nop

0x00401784 <+644>: slti v0,v0,6                 //v0<6，继续循环

0x00401788 <+648>: bnez v0,0x401728 <phase_6+552>

0x0040178c <+652>: nop

0x00401790 <+656>: lw v0,32(s8)                  // 由<+616>可知这里是最后一个节点

0x00401794 <+660>: nop
```

0x00401798 <+664>: sw zero,8(v0)　　// 【v0+8】=0，把最后一个节点的后面置空 NULL

0x0040179c <+668>: lw v0,60(s8)　　//给 v0 找一个位置即【s8+60】，新开辟一个空间

0x004017a0 <+672>: nop

0x004017a4 <+676>: sw v0,32(s8)

0x004017a8 <+680>: sw zero,28(s8)　　// 设置循环变量 i=0

0x004017ac <+684>: b 0x401878 <phase_6+888>

0x004017b0 <+688>: nop

循环， 判断新链表是否按升序或降序排序：
0x004017b4 <+692>: lw v0,-32660(gp)　　// 取得学号

0x004017b8 <+696>: nop

0x004017bc <+700>: lw v0,44(v0)　　//取得学号最后一位=8

0x004017c0 <+704>: nop

0x004017c4 <+708>: andi v0,v0,0x1

0x004017c8 <+712>: andi v0,v0,0xff　　//判断奇偶

0x004017cc <+716>: beqz v0,0x401818 <phase_6+792>　　//偶数跳转

0x004017d0 <+720>: nop

0x004017d4 <+724>: lw v0,32(s8)

0x004017d8 <+728>: nop

0x004017dc <+732>: lw v1,0(v0)

0x004017e0 <+736>: lw v0,32(s8)

0x004017e4 <+740>: nop

0x004017e8 <+744>: lw v0,8(v0)

0x004017ec <+748>: nop

0x004017f0 <+752>: lw v0,0(v0)

0x004017f4 <+756>: nop

0x004017f8 <+760>: slt v0,v1,v0　　//奇数降序

0x004017fc <+764>: beqz v0,0x401854 <phase_6+852>

```
0x00401800 <+768>: nop

0x00401804 <+772>: jal 0x4021f0 <explode_bomb>

0x00401808 <+776>: nop

0x0040180c <+780>: lw gp,16(s8)

0x00401810 <+784>: b 0x401854 <phase_6+852>

0x00401814 <+788>: nop

0x00401818 <+792>: lw v0,32(s8)

0x0040181c <+796>: nop

0x00401820 <+800>: lw v1,0(v0)                    // v1 为第 i 个节点

0x00401824 <+804>: lw v0,32(s8)

0x00401828 <+808>: nop

0x0040182c <+812>: lw v0,8(v0)                    // v0 指向 i+1 节点的地址

0x00401830 <+816>: nop

0x00401834 <+820>: lw v0,0(v0)                    // v0 为 i+1 节点

0x00401838 <+824>: nop

0x0040183c <+828>: slt v0,v0,v1
// 如果 v1<v0,则跳转，即前一个节点的值要小于后一个节点的值即升序排序

0x00401840 <+832>: beqz v0,0x401854 <phase_6+852>

0x00401844 <+836>: nop

0x00401848 <+840>: jal 0x4021f0 <explode_bomb>

0x0040184c <+844>: nop

0x00401850 <+848>: lw gp,16(s8)

0x00401854 <+852>: lw v0,32(s8)

0x00401858 <+856>: nop

0x0040185c <+860>: lw v0,8(v0)                    // 到达链表下一个节点

0x00401860 <+864>: nop

0x00401864 <+868>: sw v0,32(s8)

0x00401868 <+872>: lw v0,28(s8)
```

0x0040186c <+876>: nop

0x00401870 <+880>: addiu v0,v0,1 // i++

0x00401874 <+884>: sw v0,28(s8)

0x00401878 <+888>: lw v0,28(s8)

0x0040187c <+892>: nop

0x00401880 <+896>: slti v0,v0,5 // v0<5 时继续循环

0x00401884 <+900>: bnez v0,0x4017b4 <phase_6+692>

0x00401888 <+904>: nop

0x0040188c <+908>: move sp,s8

0x00401890 <+912>: lw ra,92(sp)

0x00401894 <+916>: lw s8,88(sp)

0x00401898 <+920>: addiu sp,sp,96

0x0040189c <+924>: jr ra

0x004018a0 <+928>: nop

7. 隐藏：二叉搜索树

0x00401990 <+0>: addiu sp,sp,-40

0x00401994 <+4>: sw ra,36(sp)

0x00401998 <+8>: sw s8,32(sp)

0x0040199c <+12>: move s8,sp

0x004019a0 <+16>: lui gp,0x42

0x004019a4 <+20>: addiu gp,gp,-20080

0x004019a8 <+24>: sw gp,16(sp)

0x004019ac <+28>: jal 0x401fec <read_line>

0x004019b0 <+32>: nop

0x004019b4 <+36>: lw gp,16(s8)

0x004019b8 <+40>: sw v0,28(s8) // 输入的字符串储存在 v0 里
0x004019bc <+44>: lw v0,28(s8) // 取出输入的数值

```
0x004019c0 <+48>: nop

0x004019c4 <+52>: move a0,v0                // a0=v0

0x004019c8 <+56>: move a1,zero              // a1=0

0x004019cc <+60>: li a2,10                  // a2=10

0x004019d0 <+64>: lw v0,-32656(gp)

0x004019d4 <+68>: nop

0x004019d8 <+72>: move t9,v0

0x004019dc <+76>: jalr t9

0x004019e0 <+80>: nop

0x004019e4 <+84>: lw gp,16(s8)

0x004019e8 <+88>: sw v0,24(s8)              // v0 =输入的数值

0x004019ec <+92>: lw v0,24(s8)

0x004019f0 <+96>: nop

0x004019f4 <+100>: addiu v0,v0,-1           // v0=v0-1

0x004019f8 <+104>: sltiu v0,v0,1001
// 条件判断，v0 的值必须小于 1001，否则炸弹爆炸，即输入的数值应该小于 1002

0x004019fc <+108>: bnez v0,0x401a10 <secret_phase+128>

0x00401a00 <+112>: nop

0x00401a04 <+116>: jal 0x4021f0 <explode_bomb>

0x00401a08 <+120>: nop

0x00401a0c <+124>: lw gp,16(s8)

0x00401a10 <+128>: lui v0,0x41

0x00401a14 <+132>: addiu a0,v0,12676

0x00401a18 <+136>: lw a1,24(s8)             // a1= 输入的数值

0x00401a1c <+140>: jal 0x4018a4 <fun7>

Fun7：
Dump of assembler code for function fun7:

0x004018a4 <+0>: addiu sp,sp,-32
```

```
0x004018a8 <+4>: sw ra,28(sp)
0x004018ac <+8>: sw s8,24(sp)                  // 输入的数值在【s8+24】
0x004018b0 <+12>: move s8,sp
0x004018b4 <+16>: sw a0,32(s8)                 //【s8+32】=a0
0x004018b8 <+20>: sw a1,36(s8)                 // a1=输入的数值
0x004018bc <+24>: lw v0,32(s8)
0x004018c0 <+28>: nop
0x004018c4 <+32>: bnez v0,0x4018d8 <fun7+52>   //如果 v0 不为 0 就跳转
0x004018c8 <+36>: nop
0x004018cc <+40>: li v0,-1
0x004018d0 <+44>: b 0x401978 <fun7+212>
0x004018d4 <+48>: nop
0x004018d8 <+52>: lw v0,32(s8)
 0x004018dc <+56>: nop
0x004018e0 <+60>: lw v1,0(v0)                  //v1 为树里的节点值
 0x004018e4 <+64>: lw v0,36(s8)                // 取出输入的数值
 0x004018e8 <+68>: nop
0x004018ec <+72>: slt v0,v0,v1                 // v0<v1,则 v0=1，否则 v0=0，跳转
 0x004018f0 <+76>: beqz v0,0x401924 <fun7+128>
0x004018f4 <+80>: nop
0x004018f8 <+84>: lw v0,32(s8)
0x004018fc <+88>: nop
0x00401900 <+92>: lw v0,4(v0)
0x00401904 <+96>: nop
0x00401908 <+100>: move a0,v0
0x0040190c <+104>: lw a1,36(s8)
```

```
0x00401910 <+108>: jal 0x4018a4 <fun7>

0x00401914 <+112>: nop

0x00401918 <+116>: sll v0,v0,0x1
//v0 左移一位，末尾为 0，v0=2*v0，即进入该节点的左孩子

0x0040191c <+120>: b 0x401978 <fun7+212>

0x00401920 <+124>: nop

0x00401924 <+128>: lw v0,32(s8)

 0x00401928 <+132>: nop

0x0040192c <+136>: lw v1,0(v0)

 0x00401930 <+140>: lw v0,36(s8)

 0x00401934 <+144>: nop

0x00401938 <+148>: slt v0,v1,v0                    // v0<v1 就跳转，反之不跳转

0x0040193c <+152>: beqz v0,0x401974 <fun7+208>

0x00401940 <+156>: nop

0x00401944 <+160>: lw v0,32(s8)

0x00401948 <+164>: nop

0x0040194c <+168>: lw v0,8(v0)

 0x00401950 <+172>: nop

0x00401954 <+176>: move a0,v0

0x00401958 <+180>: lw a1,36(s8)

0x0040195c <+184>: jal 0x4018a4 <fun7>          // 跳转回函数开头

0x00401960 <+188>: nop

0x00401964 <+192>: sll v0,v0,0x1                   // 左移一位

0x00401968 <+196>: addiu v0,v0,1              //末尾补 1，v0=2*v0+1，即进入该节点的右孩子

0x0040196c <+200>: b 0x401978 <fun7+212>

0x00401970 <+204>: nop

0x00401974 <+208>: move v0,zero                    // v0=0

0x00401978 <+212>: move sp,s8
```

0x0040197c <+216>: lw ra,28(sp)

0x00401980 <+220>: lw s8,24(sp)

0x00401984 <+224>: addiu sp,sp,32

0x00401988 <+228>: jr ra

0x0040198c <+232>: nop

End of assembler dump.

这个函数就相当于是一个二叉搜索树，根据 v0 和 v1 的大小关系来确定下一节点是左孩子还是右孩子。如果 v0 大则下一节点是右孩子，v1 大则下一节点是左节点.

Fun7 结束，继续 secret_phase:
0x00401a20 <+144>: nop

0x00401a24 <+148>: lw gp,16(s8)

0x00401a28 <+152>: move v1,v0 // v1=v0,即把 fun7 的返回值存到 v1 里

0x00401a2c <+156>: li v0,7 // v0=7，即二进制的 111

0x00401a30 <+160>: beq v1,v0,0x401a44 <secret_phase+180> //比较 v1 和 v0 是否相等，相等就跳转，否则炸弹爆炸.

0x00401a34 <+164>: nop

0x00401a38 <+168>: jal 0x4021f0 <explode_bomb>

0x00401a3c <+172>: nop

0x00401a40 <+176>: lw gp,16(s8)

0x00401a44 <+180>: lui v0,0x40

0x00401a48 <+184>: addiu a0,v0,10168

0x00401a4c <+188>: lw v0,-32712(gp)

0x00401a50 <+192>: nop

0x00401a54 <+196>: move t9,v0

0x00401a58 <+200>: jalr t9

0x00401a5c <+204>: nop

0x00401a60 <+208>: lw gp,16(s8)

0x00401a64 <+212>: jal 0x402264 <phase_defused>

0x00401a68 <+216>: nop

0x00401a6c <+220>: lw gp,16(s8)

0x00401a70 <+224>: move sp,s8

0x00401a74 <+228>: lw ra,36(sp)

0x00401a78 <+232>: lw s8,32(sp)

0x00401a7c <+236>: addiu sp,sp,40

0x00401a80 <+240>: jr ra

0x00401a84 <+244>: nop

实验步骤：

1.

炸弹一关键是 beqz，v0 为 1 炸弹爆炸，所以就是看 jal 跳转的那个函数，即判断字符串是否相等，故在 0x400d8c 设置断点，查看内置字符串，“Let’s begin now!”即炸弹一的答案

```
0x00400d84 <+24>:    lui     v0,0x40
0x00400d88 <+28>:    addiu   a1,v0,10092
0x00400d8c <+32>:    jal     0x401cf8 <strings_not_equal>
0x00400d90 <+36>:    nop
0x00400d94 <+40>:    beqz    v0,0x400da4 <phase_1+56>
0x00400d98 <+44>:    nop
0x00400d9c <+48>:    jal     0x4021f0 <explode_bomb>
```

```
Breakpoint 1, 0x00400d8c in phase_1 ()
(gdb) x /16c $a1
0x40276c:       76 'L'  101 'e' 116 't' 39 '\'' 115 's' 32 ' '  98 'b'  101 'e'
0x402774:       103 'g' 105 'i' 110 'n' 32 ' '  110 'n' 111 'o' 119 'w' 33 '!'
(gdb) c
Continuing.
```

2.

炸弹二是要输入六个数字

```
0x00400de4 <+40>:    move    a1,v0
0x00400de8 <+44>:    jal     0x401ba8 <read_six_numbers>
0x00400dec <+48>:    nop
```

首先 v1 要为 1，不然就爆炸

```
0x00400df0 <+52>:    lw      gp,16(s8)
0x00400df4 <+56>:    lw      v1,28(s8)
0x00400df8 <+60>:    li      v0,1
0x00400dfc <+64>:    beq     v1,v0,0x400e10 <phase_2+84>
0x00400e00 <+68>:    nop
0x00400e04 <+72>:    jal     0x4021f0 <explode_bomb>
```

然后是一个循环<phase_2+100>到<+248>，分析代码可知 a0 为第 i 个输入乘学号倒数第 i 个数字

beq a0,v0 即 a0 要与 v0 相等，否则炸弹爆炸，也可以在 0x400e84 设置断点，不断查看 a0 中的值，得到最终结果，即 1 8 32 0 0 0

```
0x00400e74 <+184>:   addiu   v1,s8,24
0x00400e78 <+188>:   addu    v0,v1,v0
0x00400e7c <+192>:   lw      v0,4(v0)
0x00400e80 <+196>:   nop
0x00400e84 <+200>:   beq     a0,v0,0x400e98 <phase_2+220>
0x00400e88 <+204>:   nop
0x00400e8c <+208>:   jal     0x4021f0 <explode_bomb>
0x00400e90 <+212>:   nop
```

```
Breakpoint 4, 0x00400e84 in phase_2 ()
(gdb) p $a0
$15 = 8
(gdb) p $v0
$16 = 8
(gdb) c
Continuing.
```

```
Breakpoint 4, 0x00400e84 in phase_2 ()
(gdb) p $a0
$18 = 32
(gdb) p $v0
$19 = 48
```

```
Breakpoint 1, 0x00400e84 in phase_2 ()
(gdb) p $a0
$2 = 0
(gdb) p $v0
$3 = 0
```

3.

Scanf 输入%d %c %d

```
Breakpoint 2, 0x00400f10 in phase_3 ()
(gdb) p $a1
$2 = 4204416
(gdb) x/s $a1
0x402780:       "%d %c %d"
(gdb) x/s $a2
0x4080000c:     "\001"
(gdb) x/s $a0
0x413334 <input_strings+160>:    "7 b 103"
(gdb) c
```

根据第一个参数，会跳到不同的程序段，学号最后一位是 8，然后要找到能被 8 整除的数字在的程序段，824/8 = 103，所以第一个数和第三个数为 7,103

```
(gdb) p $v1
$3 = 824
(gdb) p $v0
$4 = 824
```

第二个数与 v1 相等，v1 为 98，asc 码为 98 的字符即 b，所以第二个参数为 b

```
Breakpoint 3, 0x00401204 in phase_3 ()
(gdb) p $v1
$5 = 98
(gdb) p $v0
$6 = 98
```

```
0x004011f4 <+800>:   nop
0x004011f8 <+804>:   lb      v0,40(s8)
0x004011fc <+808>:   lb      v1,32(s8)
0x00401200 <+812>:   nop
0x00401204 <+816>:   beq     v1,v0,0x401218 <phase_3+836>
0x00401208 <+820>:   nop
0x0040120c <+824>:   jal     0x4021f0 <explode_bomb>
```

4.
输入的是一个数字，函数内部结构可知 f（x）=f（x-1）+f（x-2）
时斐波那契数列的递归形式，所以答案是求 f（x）=13 的 x 为多少，答案是 6

```
Breakpoint 4, 0x004013bc in phase_4 ()
(gdb) p $v1
$11 = 13
(gdb) p $v0
$12 = 13
0x004013a4 <+232>:   move    a0,v0
0x004013a8 <+236>:   jal     0x401230 <func4>
0x004013ac <+240>:   nop
0x004013b0 <+244>:   lw      gp,16(s8)
0x004013b4 <+248>:   move    v1,v0
0x004013b8 <+252>:   li      v0,13
0x004013bc <+256>:   beq     v1,v0,0x4013d0 <phase_4+276>
0x004013c0 <+260>:   nop
0x004013c4 <+264>:   jal     0x4021f0 <explode_bomb>
```

5.
输入 6 个字符，giants 对应的十进制分别为 15 0 5 11 13 1
则对应的 ASCII 码后四位分别为 1111 0000 0101 1011 1101 0001
则对应的字母分别为 o(0110 1111) p(0111 0000) u(0111 0101) k(0110 1011) m(0110 1101) q(0111 0001)
即应输入 op u(e) km q(a)

```
Breakpoint 9, 0x004014d0 in phase_5 ()
(gdb) x /s $a1
0x4027b0:       "giants"
```

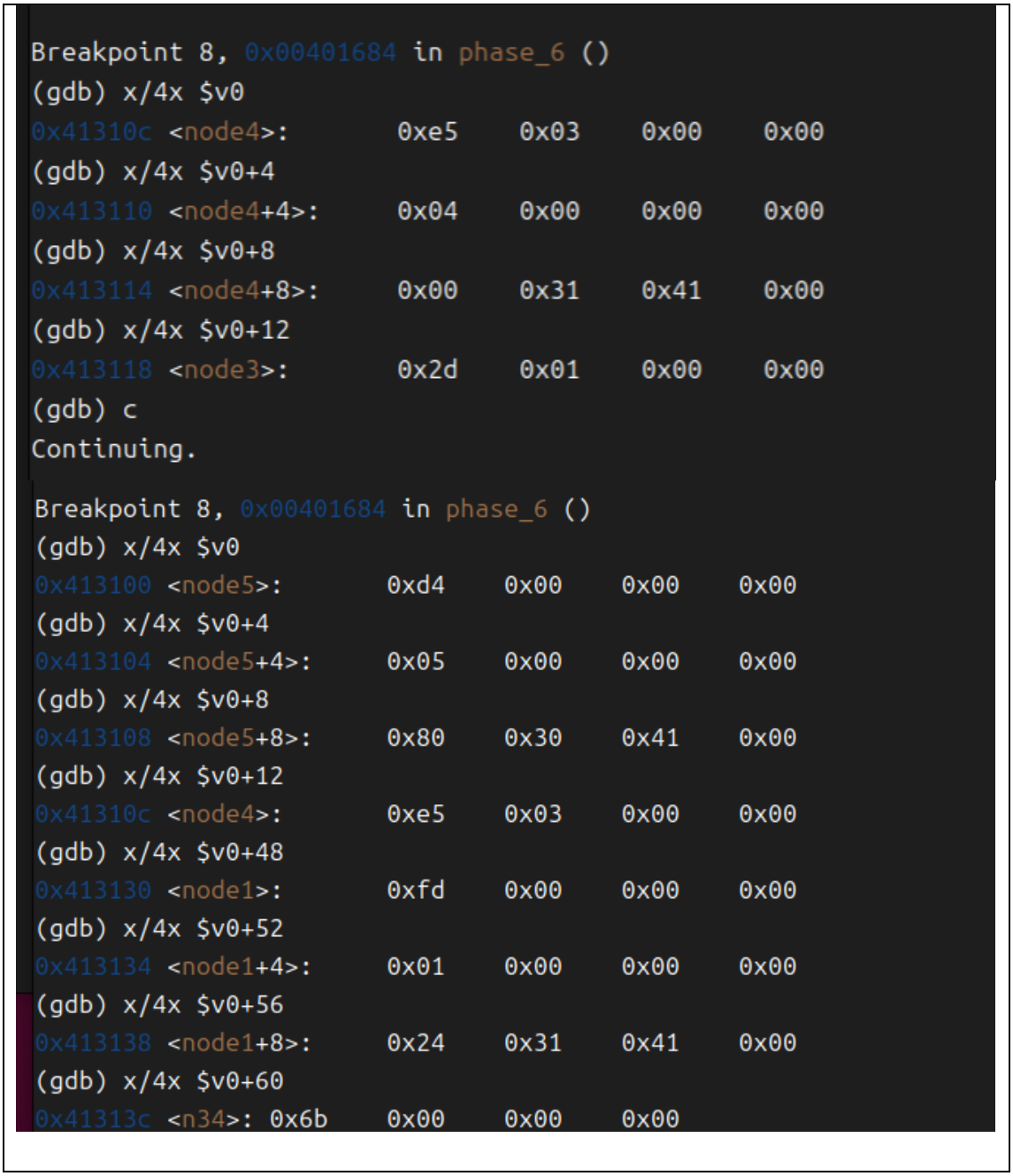
```
Breakpoint 4, 0x00401484 in phase_5 ()
(gdb) x /s $v0
0x4130ec <array.3607>:  "isrveawhobpnutfg"
```

6.

找到六个节点的地址，学号最后一位为偶数，升序排序，即按地址从小到大排序即可，六个地址分别为 fd,2d5,12d,3e5,d4,1b0,所以节点从小到大为节点5,1,3,6,2,4，最后可以使得 v1 每次都小于 v0

```
Breakpoint 8, 0x00401684 in phase_6 ()
(gdb) x/4x $v0
0x413124 <node2>:       0xd5    0x02    0x00    0x00
(gdb) b *0x00401684
Note: breakpoint 8 also set at pc 0x401684.
Breakpoint 13 at 0x401684
(gdb) b *0x00401680
Breakpoint 14 at 0x401680
(gdb) x/4x $v0+4
0x413128 <node2+4>:     0x02    0x00    0x00    0x00
(gdb) x/4x $v0
0x413124 <node2>:       0xd5    0x02    0x00    0x00
(gdb) x/4x $v0+8
0x41312c <node2+8>:     0x18    0x31    0x41    0x00
(gdb) x/4x $v0+12
0x413130 <node1>:       0xfd    0x00    0x00    0x00
(gdb)
```

```
Breakpoint 8, 0x00401684 in phase_6 ()
(gdb) x/4x $v0
0x413118 <node3>:       0x2d    0x01    0x00    0x00
(gdb) x/4x $v0+4
0x41311c <node3+4>:     0x03    0x00    0x00    0x00
(gdb) x/4x $v0+8
0x413120 <node3+8>:     0x0c    0x31    0x41    0x00
(gdb) x/4x $v0+12
0x413124 <node2>:       0xd5    0x02    0x00    0x00
(gdb)
```


Breakpoint 8, 0x00401684 in phase_6 ()
(gdb) x/4x $v0
0x41310c <node4>: 0xe5 0x03 0x00 0x00
(gdb) x/4x $v0+4
0x413110 <node4+4>: 0x04 0x00 0x00 0x00
(gdb) x/4x $v0+8
0x413114 <node4+8>: 0x00 0x31 0x41 0x00
(gdb) x/4x $v0+12
0x413118 <node3>: 0x2d 0x01 0x00 0x00
(gdb) c
Continuing.
Breakpoint 8, 0x00401684 in phase_6 ()
(gdb) x/4x $v0
0x413100 <node5>: 0xd4 0x00 0x00 0x00
(gdb) x/4x $v0+4
0x413104 <node5+4>: 0x05 0x00 0x00 0x00
(gdb) x/4x $v0+8
0x413108 <node5+8>: 0x80 0x30 0x41 0x00
(gdb) x/4x $v0+12
0x41310c <node4>: 0xe5 0x03 0x00 0x00
(gdb) x/4x $v0+48
0x413130 <node1>: 0xfd 0x00 0x00 0x00
(gdb) x/4x $v0+52
0x413134 <node1+4>: 0x01 0x00 0x00 0x00
(gdb) x/4x $v0+56
0x413138 <node1+8>: 0x24 0x31 0x41 0x00
(gdb) x/4x $v0+60
0x41313c <n34>: 0x6b 0x00 0x00 0x00

```
Breakpoint 8, 0x00401684 in phase_6 ()
(gdb) x /4x $v0
0x413118 <node3>:       0x2d    0x01    0x00    0x00
(gdb) x $v0
0x413118 <node3>:       0x2d
(gdb) c
Continuing.

Breakpoint 8, 0x00401684 in phase_6 ()
(gdb) Quit
(gdb) x /4x $v0
0x41310c <node4>:       0xe5    0x03    0x00    0x00
(gdb) c
Continuing.

Breakpoint 8, 0x00401684 in phase_6 ()
(gdb) x /4x $v0
0x413100 <node5>:       0xd4    0x00    0x00    0x00
(gdb) c
Continuing.
```

```
(gdb) p $v0
$10 = 253
(gdb) c
Continuing.

Breakpoint 10, 0x0040183c in phase_6 ()
(gdb) p $v0
$11 = 301
(gdb) p $v1
$12 = 253
(gdb) c
Continuing.

Breakpoint 10, 0x0040183c in phase_6 ()
(gdb) p $v0
$13 = 432
(gdb) p $v1
$14 = 301
(gdb) c
Continuing.
```

```
Breakpoint 10, 0x0040183c in phase_6 ()
(gdb) p $v0
$15 = 725
(gdb) c
Continuing.

Breakpoint 10, 0x0040183c in phase_6 ()
(gdb) p $v0
$16 = 997
(gdb) c
Continuing.
```

7.
要在炸弹四的答案后跟一个字符串，断点查看 0x4022f4 判断字符串是否相等的函数里的字符串，然后在六个炸弹拆完后就会进入隐藏炸弹。

```
0x004022d0 <+108>:   lw      gp,16(s8)
0x004022d4 <+112>:   move    v1,v0
0x004022d8 <+116>:   li      v0,2
0x004022dc <+120>:   bne     v1,v0,0x402354 <phase_defused+240>
0x004022e0 <+124>:   nop
0x004022e4 <+128>:   addiu   v0,s8,24
0x004022e8 <+132>:   move    a0,v0
0x004022ec <+136>:   lui     v0,0x40
0x004022f0 <+140>:   addiu   a1,v0,10416
0x004022f4 <+144>:   jal     0x401cf8 <strings_not_equal>
0x004022f8 <+148>:   nop
0x004022fc <+152>:   lw      gp,16(s8)
0x00402300 <+156>:   bnez    v0,0x402354 <phase_defused+240>
0x00402304 <+160>:   nop
```

```
(gdb) x /s $a1
0x4028b0:       "austinpowers"
(gdb)
```

要让 v1=v0=7，程序是一个二叉搜索树，v0 小于 v1 就进入左孩子，v0>v1 进入右孩子
输入一个 1，发现 v1 变化是 36->8->6->0，v0 一直小于 v1，每次左移一位，一直都是 0.
v0 为 111，说明每次左移一位加一，即比较三次都要 v0 大于 v1
36->50->107，比 107 大一点仍会爆炸
且 v0 要小于 1002，所以 v0 最大为 1001，故输入 1001


0x004019ec <+92>: lw v0,24(s8)
0x004019f0 <+96>: nop
0x004019f4 <+100>: addiu v0,v0,-1
0x004019f8 <+104>: sltiu v0,v0,1001
0x004019fc <+108>: bnez v0,0x401a10 <secret_phase+128>
0x00401a00 <+112>: nop
0x00401a04 <+116>: jal 0x4021f0 <explode_bomb>
0x00401a08 <+120>: nop
0x00401928 <+132>: nop
0x0040192c <+136>: lw v1,0(v0)
0x00401930 <+140>: lw v0,36(s8)
0x00401934 <+144>: nop
0x00401938 <+148>: slt v0,v1,v0
0x0040193c <+152>: beqz v0,0x401974 <fun7+208>
0x00401940 <+156>: nop
0x00401944 <+160>: lw v0,32(s8)
如果v0<v1就跳转
0x00401954 <+176>: move a0,v0
0x00401958 <+180>: lw a1,36(s8)
0x0040195c <+184>: jal 0x4018a4 <fun7>
0x00401960 <+188>: nop
0x00401964 <+192>: sll v0,v0,0x1
0x00401968 <+196>: addiu v0,v0,1
0x0040196c <+200>: b 0x401978 <fun7+212>
0x00401970 <+204>: nop
0x00401974 <+208>: move v0,zero
左移一位加一即右孩子
0x00401a0c <+124>: lw gp,16(s8)
0x00401a10 <+128>: lui v0,0x41
0x00401a14 <+132>: addiu a0,v0,12676
0x00401a18 <+136>: lw a1,24(s8)
0x00401a1c <+140>: jal 0x4018a4 <fun7>
0x00401a20 <+144>: nop
0x00401a24 <+148>: lw gp,16(s8)
0x00401a28 <+152>: move v1,v0
0x00401a2c <+156>: li v0,7
0x00401a30 <+160>: beq v1,v0,0x401a44 <secret_phase+180>
0x00401a34 <+164>: nop

```
Breakpoint 6, 0x004018e0 in fun7 ()
(gdb) p $v1
$24 = 36
(gdb) p $v0
$25 = 4272492
(gdb) c
Continuing.

Breakpoint 6, 0x004018e0 in fun7 ()
(gdb) p $v1
$26 = 50
(gdb) p $v0
$27 = 4272444
(gdb) c
Continuing.

Breakpoint 6, 0x004018e0 in fun7 ()
(gdb) p $v1
$28 = 107
(gdb) c
Continuing.
```

```
Breakpoint 4, 0x00401a30 in secret_phase ()
(gdb) p $v1
$29 = 7
(gdb) p $v0
$30 = 7
(gdb) c
Continuing.
```

8.全部过程

```
quiet@quiet-linux:~/桌面/bomb$ qemu-mipsel -g 12345 ./bomb
Please input your ID_number:
202200130048
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Let's begin now!
Phase 1 defused. How about the next one?
1 8 32 0 0 0
That's number 2.  Keep going!
7 b 103
Halfway there!
6 austinpowers
So you got that one.  Try this one.
opekma
Good work!  On to the next...
5 1 3 6 2 4
Curses, you've found the secret phase!
But finding it and solving it are quite different...
1001
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
```

结论分析与体会：

常用 mips 指令

addiu 相加

lw $r1, 0($r8) 取 r8 寄存器中的字加上 16 位偏移量存到 r1 寄存器中

sw 从 r1 中读取数据按 16 位偏移量 offset 写入寄存器 r8

beq $r0, $r0, PROG2 相等跳转

bgez $r1,LABEL2 若 r1 大于等于 0，则跳转

sll $r0,$r0,0 r0 逻辑左移,按符号位扩展

sub $r5,$r4,$r3 r3 减 r4，并将其存入 r5

等等