山东大学 计算机科学与技术 学院

计算机系统原理 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202200400053 | 姓名：王宇涵 | | 班级： 2班 |
| 实验题目： 拆炸弹 | | | |
| 实验学时：2 | | 实验日期： 2023-11-15 | |
| 实验目的：  (1) 熟悉MIPS 指令集；  (2) 根据反汇编程序可以分析程序的功能和执行流程；  (3) 熟悉GDB 调试工具，帮助程序理解。 | | | |
| 硬件环境：  龙芯实验平台（MIPS） | | | |
| 软件环境：  1.bomb 可执行文件（MIPS）bomb 汇编文件  2.利用QEMU+GDB在ubuntu系统中配置MIPS调试运行环境 | | | |
| 实验步骤与内容： 实验一:  1. 进入gdb调试界面,通过disas phase\_1命令观察phase\_1代码(注意,默认分页浏览,因此我们需要设置set pagination off来显示所有代码),发现<string\_not\_equal>的函数,便知道输入的是一行字符串 2. 观察 beqz v0,0x400da4 <phase\_1+56>,发现这是判断语句,决定了炸弹是否引爆,因此我们只需要查看函数的两个参数就可以知道目标字符串是什么 3. 断点设在<string\_not\_equal>函数处,查看a0和a1,发现分别是输入的字符串和目标字符串,即可得到最后的结果Let’s begin start!     输入Let’s begin start!成功通过第一关 实验二:  1. 观察phase\_2代码,发现<read\_six\_numbers>,得知,我们需要输入6个数字(注意不能是连续的数字,如输入123456,应该输入1 2 3 4 5 6)   我们第一次输入1 2 3 4 5 6,观察代码发现第一个判断处beq v1,v0,0x400e10 <phase\_2+84>,在此处设置断点,此时v0=1,v1是从地址s8+28读入的数字,好奇地查看s8+28地址为首的数字串,发现正好是我们输入的数,则v1是我们输入的第一个数字.    又看到v1和1不相等就爆炸,则输入的第一个数字为1,我们测试输入为1,没有爆炸,则继续调试   1. 看到li v0,1和 sw v0,24(s8)指令,是将1存储在s8+24地址处,后指令直接跳转到   <+232>处,此处是先读入s8+24处的数,如果v0<6就将v0设置成1,若v0不为0则跳转到<+100>处,我们可以猜测出这是进行一次循环,循环变量为i,i<6时每次都重复执行<+100>到<+232>处的代码,并进行i++   1. 观察<+100>到<+124>处的代码,先取出循环变量i,将i扩展成字节,再取出s8+28+4\*i的数,这是取出输入的第i个数存储到a0中 2. 观察<+128>到<+168>处的代码,先取出循环变量i,并进行操作v0=12-i,这让我很不解,后看到lw v1 -32660(gp),发现从这个地址开始的12位数正好是我们的学号     这让我恍然大悟,原来取出的是我们学号的倒数第i位,并将取出学号倒数第i位乘输入的第i个数字,结果存入a0   1. 观察<+172>到<+192>处的代码,发现也是取输入的数字,但是此时没有对i进行-1,因此取出的是输入的第i+1个数字 2. 判断输入的第i+1个数字是否等于学号倒数第i位乘以输入的第i个数字,如果相等就继续循环,否则爆炸 3. 因此我们知道,输入第一个数字为1,输入的第二个数字必须为第一个数字乘学号的倒数第1位,即3,以此类推,答案为1 3 15 0 0 0  实验三  1. 发现没有规定的输入函数,先尝试输入1 2 3 4 5 6,看到第一个判断slti v0,v0,3     发现此时v0=3,不知道是输入的3还是长度?尝试输入1 2 3 4 5,发现结果仍是3,    接着输入1 2,发现结果为2    那就有可能是数字长度,我们接着测试输入5    则可以确定是我们输入的数字长度,但是不会超过3,且必须等于3,小于3会爆炸,则猜测只会存储输入的前三个数字  输入3,4,5,发现读取函数lw v0,44(s8),此时猜测连续存储在s8+44地址处    打印发现不是,只有3,那往前打印试试    发现5,4,3存储在S8+36,S8+40,S8+44处,则是倒序存储,可能是以栈的形式存储,注意此时4为字符,提示我们第二个输入应该输入字符  继续比较,发现读取第一个数字>=8就会爆炸,因此我们输入0~7才能不爆炸,此时我们输入的是3,继续调试,发现对V0进行了一堆地址赋值的操作,并且最后找到了对应地址,直接jr v0跳转,发现来到了<+372>位置   1. 进入<+372>发现因为学号一共12位,第12位为4\*11+V0的位置,所以v1存储了学号的最后一位,V0读取了输入的第三个数字,将两者相乘存入v0,比较v0和0的关系,若相等则跳转,否则爆炸,因此第三个数字一定是0 2. 重新输入3,4,0,进行调试,进入了<+804>,取出输入的第二个字符,将其与之前对应的分支存储的107进行比较,若相等则成功,否则爆炸,因此我们第二个字符最后应该输入k,它的askii码值为107,则最终答案为3 k 0 3. 通过分析,我们可以知道输入的第一个参数在0-7之间,且第一个参数决定了存储在S8+32的值,第一个参数和学号的最后一位决定了第三个参数的取值,第一个参数也决定了第二个参数的取值,通过调试,发现第一个参数输入0,1,2,3,4,5,6,7,分别跳转到<+168>,<+236>,<+304>,<+372>, <+436>,<+504>,<+572>,<+632>的位置,对应不同的分支,某些输入情况下学号最后一位决定了某些分支是不可行的,比如第一个参数输入7,<+668>下学号最后一位为3,但是乘积无法满足824(不是3的倍数)  实验四  1. 观察代码,发现没有规定的输入函数,则测试输入1 2 3 4 5 6,发现能通过第一个判断条件li v0,1,bne v1,v0,0x401330 <phase\_4+116>,此时观察lw v0,24(s8),打印S8+24处的值     发现有一个1,不知道是系统自带的还是输入的1,还是长度为1,我们重新换一组输入进行测试,输入3 4,此时结果为    则应该是输入的第一个数字   1. 我们继续进行调试,发现判断bgtz v0,0x401340 <phase\_4+132>,则若v0<=0,会爆炸,因此输入的第一个数字必须大于0 2. 观察发现<+132>到<+156>取出学号的最后一位(通过地址-32660(gp)可以判断),并且判断奇偶性,如果是奇数则不跳转,是偶数则跳转,我的学号最后一位是3,则不跳转 3. 观察<+164>到<+176>,先取出了输入的第一个数字放入了a0,然后调用了函数fun4,利用ni指令跳过函数内部,发现vi的值已经发生了改变     测试不同的输入,发现调用完函数v0的值均不同   1. 先不看函数内部,继续观察流程,发现需要判断v0和8的大小,如果相等则成功,若不相等则失败,因此我们需要输入一个值,这个值使得调用完func4之后,v0的值变为8 2. 观察func4函数内部,发现先取出了输入的数字,并进行判断,如果小于2就跳到函数末尾(将v0赋值为1并返回子函数),否则继续执行,将(v0-1)传入fun4得到结果v0,并存入s0, 将(v0-2)传入fun4得到结果v0,并将s0和v0相加,返回v0 3. 这给了我很大的暗示,让我想到了斐波那契数列(1,1,2,3,5,8...),因此题目变成了,输入x,使得斐波那契数列中f(x)=8,则x=5.  实验五  1. 首先看到函数<string-length>,便判断输入一串字符串,又看到判断beq v1,v0,0x401420 <phase\_5+56>,则知道输入字符串的长度一定为6,因此我们输入测试abcdef 2. 进行调试,发现先将0存入了s8+24处,直接跳转到判断s8+24地址处的值是否<6,若小于6跳转回上方,否则到代码末尾,可知道存入了循环变量i,初始值为0,且i=6时跳出循环 3. 观察循环部分,从<+68>开始,看到了第一个不同的地址处lw a0,72(s8),好奇打印,发现就是我们存储的字符串     则存储字符串的首地址为s8+72,继续看代码,发现提取了s8+72+i的位置的元素,也就是我们输入的第i个元素,将这个字符存入v1,并进行与00001111的相与操作提取该字符二进制的后四位存入v1,后将第i个字符的后四位存入4\*i+s8+36的位置,这里测试样例i=0时该字符为’a’,后四位为1,因此存入了s8+40的位置   1. <+120>开始,提取了第i个字符的后四位,放入了v1,并出现了一个新的地址lui v0,0x41,   addiu v0,v0,12524,好奇地进行打印    发现有一个字符串”isrveawhobpnutfg”且一共16位,正好对应了二进制的后四位  后在(基址+第i个字符的后四位)处提取字符存入s8+28+i的地址处,这里后四位为0001,因此提取了字符’s’,  一次循环结束,则新字符串的存储位置首地址为s8+28.   1. 所有循环结束后, sb zero,34(s8)指令将顺序存储字符串最后一位赋予0,并找到顺序存储字符串首地址s8+28赋给a0,此时a1的值是一个新的地址,我们打印一下会有惊喜的发现     首先是字符串”giants”正好6位,是将要比较的字符串.后来还有一串字符  Wow!You've defused the secret stage! 发现了秘密阶段!小彩蛋!难道有秘密炸弹?   1. 不难发现,每次循环都会通过我们输入第i个元素的后四位作为偏移量找到给定字符串对应的字符,并将其顺序存放在特定位置,最后构成一个六字符字符串,并与giants作比较,相同成功,否则爆炸,因此我们需要找出给定字符串isrveawhobpnutfg中6个字符分别的位置为15,0,5,11,13,1,我们找到后四位分别为这6个数的字符即可,一种组合是OPUKMQ,成功!  实验六  1. 通过<read\_six\_numbers>函数可以推断出输入六个数字,我们输入1 2 3 4 5 6作为测试样例,进行调试,看到lw v0,12(v0),好奇地查看s8+36处的数字,发现每隔4位进行存储我们输入的数字,此时进行了循环,i=0循环6次,则意为取出我们输入的第i个数存放到v0,进行判断是否小于7,如果大于7就爆炸,因此每一个数都不能大于7,下方代码同理,每一个数都不能小于等于0,因此,我们输入的数都在1~6之间才可以,测试样例正常可以继续调试      1. 到达<+176>,此处将i++后存入了s8+24处,又进行了一次循环,设这个变量为j,则j从i+1开始,j<6结束,在这一次循环中,我们先取出输入的第i个数,和输入的第j个数,如果两个相等,则爆炸,如果不相等,则继续循环,因此我们输入的任意两个数不能相等,因此我们只能输入1,2,3,4,5,6六个数(顺序不定) 2. 跳出第一个循环,进入<+348>,发现一个新的空间为0x41+12592,将这个地址送入了s8+32处,接着继续设置循环j=1,此时取出第i个输入的数,j<a[i]的时候再继续循环,由于输入的数为[1:6],因此不会越界,每次循环中,都先找到s8+32处,打印发现这是一个链表结点     接着找到链表的下一个结点,把这个结点存入S8+32处,便于接着循环.在j循环结束后,我们找到s8+60+4i的地址,然后将找到的最后一个结点元素存入这个地方,则相当于顺序存储了一个数组,首地址为s8+60.该数组元素res[i]=每一次j循环结束最后的结点   1. 跳出第二个循环,进入<+524>处,先读取了数组的首元素的地址,然后放到s8+32处,然后进行循环,i=1到5,先通过地址v1=s8+60+4i取数组第i个结点的地址(从第二个元素开始),然后将这个结点存入s8+32处结点p的下一个结点处np,接着更新s8+32处结点p=np,则这个操作是把数组中的元素存入链表中.循环结束后再将s8+32处结点赋值为数组中的第一个元素,即链表的firstNode. 2. 进入最后一个循环<+680>,i=0到5,先拿出学号的最后一位,再进行判断奇数和偶数,我的学号最后一位是奇数1,因此进入<+724>段,发现先取出头结点的值v1,再取出头结点下一个结点的值v0, 更新头结点以保证遍历所有元素,如果v1<v0则爆炸,因此排列完的链表是必须降序排列的! 3. 综上,我们需要通过输入数字来确定排列顺序,每个数字对应了将链表的第i个元素从头往后依此放入数组中,从而使得排列完的链表是降序排列的,通过调试我们得知链表的六个元素分别为0x0fd, 0x2d5, 0x12d, 0x3e5, 0x0d4, 0x1b0,因此最后的结果为4 2 6 3 1 5,表示将链表第4个元素放入数组第一个位置,将链表第2个元素放入数组第二个位置…最终成功!  隐藏炸弹 在发现secret\_phase的时候,我在想是否有相应的阶段呢?进行打印发现确实有    通过和同学交流,发现在实验四的答案5后加上空格和austinpowers,即可开启秘密炸弹,于是我迫不及待地进入了秘密炸弹的破解中.  先观察代码,发现read\_line函数,则我测试输入了”abcd”字符串,继续进行调试,发现到了<+40> sw v0,28(s8),打印调试v0的值发现恰好是我们所输入的值,因此我们的输入存入了s8+28处    接下来看到设置参数a0,a1,a2,通过-32656(gp)进入了一个函数,改变了v0的值,将此时的v0存入了s8+24的位置,我们打印此时v0的值      发现为0  我们换个输入1234,发现v0存放的就是我们输入的数字      并将v0-1和1001比较大小,如果小于则跳转,否则爆炸,因此我们可以推断,我们应该输入一行数字,且数字<=1001,函数的作用可能是把字符串转化为数字,并判断是不是字母.  继续调试,发现进入了一个函数<fun7> ,此时的参数1: 0x41+12676,参数2: 输入的数字, 经过调试发现参数1的地址处存放了一个结点    可以发现fun7函数改变了v0的值,此时函数将v0和7进行比较,如果相等就过关了,进入<phase\_defused>,否则失败,因此我们需要通过fun7函数找出输入的数字使得v0=7  查看fun7函数,发现先判断结点值是否为0,如果为0就返回-1, 否则继续  后将输入的数字a和结点值b做比较,如果a<b就不跳转, 如果a>=b就不跳转跳转   * 不跳转: 发现找到了结点偏移量为4的位置, 把第一个参数更新为这个位置,进行函数递归, 返回值为v0左移一位 * 跳转: 此时a>=b,进行判断,如果b<a,即a>b则不跳转: 找到了结点偏移量为8的位置, 把第一个参数更新为这个位置,进行函数递归,返回值为v0左移一位+1; 如果b>=a,即a=b,则跳转:返回0   因此我们得知这是一个**二叉搜索树**  当输入元素等于当前节点元素值时，返回0  当输入元素大于当前节点元素值时，将节点移至右节点接着调用得到返回值:v0左移一位+1  当输入元素小于当前节点元素值时，将节点移至左节点接着调用得到返回值:v0左移一位  因此我们要得到7,则需要111 的二进制序列, 则需要连续递归右子树3次,我们找右子树即可, 但是树的结构未知的情况下,我们肯定输入的数越大越好,因此我们输入1001,进行调试打印,发现结点依此为36 , 50  ,107 ,1001 直接通过了!  因此最终答案为1001  但是树的结构到底是什么呢?我经过调试,输入不同的值得出了结果    可以发现最多递归3次,因此我们最终的答案只能是1001  那么如果输入不是结点的值(进入空结点)会发生什么呢?    经过调试发现此时空结点v0的值是0,结合以上信息,可以得出v0==0的判断是用来判断是否为空结点,否则返回-1 最终答案 学号  202200400053  炸弹一  Let’s begin start!  炸弹二  1 3 15 0 0 0  炸弹三  3 k 0  炸弹四  5  炸弹五  OPUKMQ  炸弹六  4 2 6 3 1 5  秘密炸弹 (开启在拆第四个炸弹的时候输入5 austinpowers)  1001 | | | |
| 结论分析与体会：  本次拆炸弹实验是目前经历最具挑战性的实验之一,一开始完全不知道从何下手,对着一行行汇编语言代码束束手无策,直到一步步找断点,不断调试,不断猜测,不断理解,最后弄清楚汇编代码的含义,并自己手写对应的C++程序来深化理解,这个过程是很难得和有趣的的.在拆炸弹的过程中,我对老师精妙的设计感到十分敬佩,其间也出现了一些小彩蛋,如” Wow!You've defused the secret stage”也让挑战多了一份乐趣.  我认为难点有很多,包括如何找出值的存储位置,如何判断循环条件,如何理解每一行代码是在干什么,如何理解函数和子函数的关系等等…我在做题的过程中也收获了很多经验,对汇编语言有了更深层次的理解,包括格式存储,进制转换,寄存器作用,断点设置,空间利用等等.  在我通过自己的努力获得成功的提示符的时候,会产生由衷的成就感,我为自己感到自豪,也让我知道努力和坚持总是会有收获的! | | | |

附源代码

## 实验一:

0x00400d6c <+0>: addiu sp,sp,-32

0x00400d70 <+4>: sw ra,28(sp)

0x00400d74 <+8>: sw s8,24(sp)

0x00400d78 <+12>: move s8,sp

0x00400d7c <+16>: sw a0,32(s8)

0x00400d80 <+20>: lw a0,32(s8)

0x00400d84 <+24>: lui v0,0x40

0x00400d88 <+28>: addiu a1,v0,10092

0x00400d8c <+32>: jal 0x401cf8 <strings\_not\_equal>//判断字符串是否相等,相等则将v0置为1,否则置为0

0x00400d90 <+36>: nop

0x00400d94 <+40>: beqz v0,0x400da4 <phase\_1+56>//若v0不为0则跳到phase\_2,否则爆炸

0x00400d98 <+44>: nop

0x00400d9c <+48>: jal 0x4021f0 <explode\_bomb>

0x00400da0 <+52>: nop

0x00400da4 <+56>: move sp,s8

0x00400da8 <+60>: lw ra,28(sp)

0x00400dac <+64>: lw s8,24(sp)

0x00400db0 <+68>: addiu sp,sp,32

0x00400db4 <+72>: jr ra

0x00400db8 <+76>: nop

### 个人C++理解代码

string s;cin>>s;

if(s!="Let's begin now!")

explode\_bomb();

else

phase\_2();

## 实验二

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>//读入六个数字

0x00400dec <+48>: nop

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

0x00400df4 <+56>: lw v1,28(s8)//首位数字的地址是s8+24,这是读入第一个数字

0x00400df8 <+60>: li v0,1

0x00400dfc <+64>: beq v1,v0,0x400e10 <phase\_2+84>//如果第一个数字不等于1就爆炸

0x00400e00 <+68>: nop

0x00400e04 <+72>: jal 0x4021f0 <explode\_bomb>

0x00400e08 <+76>: nop

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

0x00400e10 <+84>: li v0,1//建立循环

0x00400e14 <+88>: sw v0,24(s8)//循环变量储存在s8+24中

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

0x00400e2c <+112>: sll v0,v0,0x2

0x00400e30 <+116>: addiu v1,s8,24

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

0x00400e38 <+124>: lw a0,4(v0)//将输入的第i个数字存储到a0

0x00400e3c <+128>: li v1,12//202200400053一共12位学号!

0x00400e40 <+132>: lw v0,24(s8)

0x00400e44 <+136>: nop

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

0x00400e4c <+144>: lw v1,-32660(gp)//取出学号的首位的地址

0x00400e50 <+148>: sll v0,v0,0x2//扩展成字节

0x00400e54 <+152>: addu v0,v1,v0//取出学号倒数第i位的地址

0x00400e58 <+156>: lw v0,0(v0)//取出学号倒数第i位

0x00400e5c <+160>: nop

0x00400e60 <+164>: mult a0,v0

0x00400e64 <+168>: mflo a0//将取出学号倒数第i位乘输入的第i个数字,结果存入a0

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

0x00400e7c <+192>: lw v0,4(v0)//此时没有对i进行-1,因此取出的是输入的第i+1个数字

0x00400e80 <+196>: nop

0x00400e84 <+200>: beq a0,v0,0x400e98 <phase\_2+220>//判断输入的第i+1个数字是否等于学号倒数第i位乘以输入的第i个数字,如果相等就继续循环,否则爆炸

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//循环变量i+1

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

0x00400ea8 <+236>: lw v0,24(s8)

0x00400eac <+240>: nop

0x00400eb0 <+244>: slti v0,v0,6//循环周期为6

0x00400eb4 <+248>: bnez v0,0x400e20 <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

### 个人C++理解代码

//设输入的数字是a[0..5],输入学号为num[0..11]

for(int i=1;i<6;i++)

{

if(a[i]!=a[i-1]\*num[12-i])

{

explode\_bomb();

return;

}

}

phase\_3();

## 实验三

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

0x00400f28 <+84>: nop

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

0x00400f30 <+92>: slti v0,v0,3//v0小于3设为1

0x00400f34 <+96>: beqz v0,0x400f48 <phase\_3+116>//如果v0为0才能跳转,否则爆炸,则v0必须>=3

0x00400f38 <+100>: nop

0x00400f3c <+104>: jal 0x4021f0 <explode\_bomb>

0x00400f40 <+108>: nop

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

0x00400f48 <+116>: lw v0,44(s8)

0x00400f4c <+120>: nop

0x00400f50 <+124>: sltiu v1,v0,8//v0<8 v1=1

0x00400f54 <+128>: beqz v1,0x401190 <phase\_3+700>// v1=1不跳转,v1=0爆炸,则v0不能>=8

0x00400f58 <+132>: nop

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

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

0x00400f64 <+144>: addiu v0,v0,10124

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

0x00400f6c <+152>: lw v0,0(v0)//进行一番操作,输入的第一个数决定了v0成为哪个地址,并跳转到对应的地址

0x00400f70 <+156>: nop

0x00400f74 <+160>: jr v0

0x00400f78 <+164>: nop

0x00400f7c <+168>: li v0,113//第一个参数输入0跳转到这个位置

0x00400f80 <+172>: sb v0,32(s8)

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

0x00400f88 <+180>: nop

0x00400f8c <+184>: lw v1,44(v0)

0x00400f90 <+188>: lw v0,36(s8)

0x00400f94 <+192>: nop

0x00400f98 <+196>: mult v1,v0

0x00400f9c <+200>: mflo v1

0x00400fa0 <+204>: li v0,777//第一个参数为0,学号最后一位乘第三个参数等于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//第一个参数输入1跳转到这个位置

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//第一个参数为1,学号最后一位乘第三个参数等于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 //第一个参数输入2跳转到这个位置

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//第一个参数为2,学号最后一位乘第三个参数等于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//输入3跳转到这个位置

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>//第一个参数为3,学号最后一位乘第三个参数等于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//输入4跳转到这个位置

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//第一个参数为4,学号最后一位乘第三个参数等于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//输入5跳转到这个位置

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>//第一个参数5,学号最后一位乘第三个参数等于513

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//输入6跳转到这个位置

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

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

0x0040111c <+584>: nop

0x00401120 <+588>: lw v1,44(v0)

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

0x00401128 <+596>: nop

0x0040112c <+600>: mult v1,v0

0x00401130 <+604>: mflo v1

0x00401134 <+608>: li v0,780//第一个参数6,学号最后一位乘第三个参数等于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//输入7跳转到这个位置

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//第一个参数7,学号最后一位乘第三个参数等于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//确定第二个参数

0x004011f8 <+804>: lb v0,40(s8)//第二个参数

0x004011fc <+808>: lb v1,32(s8)//之前传入的乘积应该等于的值

0x00401200 <+812>: nop

0x00401204 <+816>: beq v1,v0,0x401218 <phase\_3+836>//如果v0=v1则成功,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

### 个人理解C++代码

string s;

cin>>s;

int d;//学号的最后一位

if(strlen(s)<3)

explode\_bomb()

else

{

int a=s[0];

char b=s[1];

int c=s[2];

switch (a)

{

case 0:

{

if(d\*c!=777||b!=113)

explode\_bomb();

else

phase\_4();

}

case 1:

{

if(d\*c!=214||b!=98)

explode\_bomb();

else

phase\_4();

}

case 2:

{

if(d\*c!=755||b!=98)

explode\_bomb();

else

phase\_4();

}

case 3:

{

if(d\*c!=0||b!=107)

explode\_bomb();

else

phase\_4();

}

case 4:

{

if(d\*c!=228||b!=111)

explode\_bomb();

else

phase\_4();

}

case 5:

{

if(d\*c!=513||b!=116)

explode\_bomb();

else

phase\_4();

}

case 6:

{

if(d\*c!=780||b!=118)

explode\_bomb();

else

phase\_4();

}

case 7:

{

if(d\*c!=824||b!=98)

explode\_bomb();

else

phase\_4();

}

default

{

explode\_bomb();

}

}

}

## 实验四

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)

0x004012fc <+64>: nop

0x00401300 <+68>: move t9,v0

0x00401304 <+72>: jalr t9//t9函数作用是判断传入的参数个数是否满足要求

0x00401308 <+76>: nop

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

0x00401310 <+84>: move v1,v0

0x00401314 <+88>: li v0,1

0x00401318 <+92>: bne v1,v0,0x401330 <phase\_4+116>//若不等于1就爆炸

0x0040131c <+96>: nop

0x00401320 <+100>: lw v0,24(s8)//取得输入的第一个数字

0x00401324 <+104>: nop

0x00401328 <+108>: bgtz v0,0x401340 <phase\_4+132>

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)//取出学号首地址

0x00401344 <+136>: nop

0x00401348 <+140>: lw v0,44(v0)//取出学号的最后一位

0x0040134c <+144>: nop

0x00401350 <+148>: andi v0,v0,0x1

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

0x00401358 <+156>: beqz v0,0x40139c <phase\_4+224>//判断最后一位奇偶性,若是奇数,则不跳转,偶数跳转

0x0040135c <+160>: nop

0x00401360 <+164>: lw v0,24(s8)//取出输入的数

0x00401364 <+168>: nop

0x00401368 <+172>: move a0,v0

0x0040136c <+176>: jal 0x401230 <func4>//传入输入的数,v0发生改变

0x00401370 <+180>: nop

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

0x00401378 <+188>: move v1,v0

0x0040137c <+192>: li v0,8

0x00401380 <+196>: beq v1,v0,0x4013d0 <phase\_4+276>//判断v0的值是否为8,为8成功,不为8爆炸

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)

0x004013a0 <+228>: nop

0x004013a4 <+232>: move a0,v0

0x004013a8 <+236>: jal 0x401230 <func4>//传入输入的数,v0发生改变

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>//判断v0的值是否为13,为13成功,不为13爆炸

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

### Func4

0x00401230 <+0>: addiu sp,sp,-40

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)//此时a0保留的是输入的数

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

0x0040124c <+28>: nop

0x00401250 <+32>: slti v0,v0,2//如果小于2,v0=1,否则为0

0x00401254 <+36>: bnez v0,0x40129c <func4+108>//v0=1跳转v0=0不跳转,即输入的数小于2跳转否则不跳转

0x00401258 <+40>: nop

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

0x00401260 <+48>: nop

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

0x00401268 <+56>: move a0,v0

0x0040126c <+60>: jal 0x401230 <func4>//将v0-1后调用子函数func4

0x00401270 <+64>: nop

0x00401274 <+68>: move s0,v0//将计算后的结果存入s0

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

0x0040127c <+76>: nop

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

0x00401284 <+84>: move a0,v0

0x00401288 <+88>: jal 0x401230 <func4>/将v0-2后调用子函数func4

0x0040128c <+92>: nop

0x00401290 <+96>: addu v0,s0,v0//将计算后的结果与s0相加存入v0

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

0x00401298 <+104>: nop

0x0040129c <+108>: li v0,1//跳转后将v0值赋为1,返回子函数地址

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

### 个人理解C++代码

int func4(int x)

{

if(x<2)

return 1;

else

return func4(x-1)+func4(x-2);

}

//设学号最后一位为d

int x;

cin>>x;

if(x<=0)

explode\_bomb();

else

{

if(d&1)

{

if(func(x)==8)

phase\_5();

else

explode\_bomb();

}

else

{

if(func(x)==13)

phase\_5();

else

explode\_bomb();

}

}

## 实验五

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)

0x00401400 <+24>: jal 0x401c78 <string\_length>

0x00401404 <+28>: nop

0x00401408 <+32>: move v1,v0

0x0040140c <+36>: li v0,6

0x00401410 <+40>: beq v1,v0,0x401420 <phase\_5+56>//字符串长度必须为6,否则爆炸

0x00401414 <+44>: nop

0x00401418 <+48>: jal 0x4021f0 <explode\_bomb>

0x0040141c <+52>: nop

0x00401420 <+56>: sw zero,24(s8)//将0存入S8+24,即循环变量i=0

0x00401424 <+60>: b 0x4014a8 <phase\_5+192>

0x00401428 <+64>: nop

0x0040142c <+68>: lw v0,24(s8)//从此处开始循环,读入循环变量i

0x00401430 <+72>: lw v1,24(s8)

0x00401434 <+76>: lw a0,72(s8)

0x00401438 <+80>: nop

0x0040143c <+84>: addu v1,a0,v1

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

0x00401444 <+92>: nop

0x00401448 <+96>: andi v1,v1,0xff

0x0040144c <+100>: andi v1,v1,0xf//提取第i个字符的后四位

0x00401450 <+104>: sll v0,v0,0x2//将i扩展成int

0x00401454 <+108>: addiu a0,s8,24

0x00401458 <+112>: addu v0,a0,v0//v0=4\*i+s8+24

0x0040145c <+116>: sw v1,12(v0)//将提取第i个字符的后四位存储在4\*i+s8+36的位置

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

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

0x00401478 <+144>: lw v1,12(v0)//取出后四位放入v1

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

0x00401480 <+152>: addiu v0,v0,12524

0x00401484 <+156>: addu v0,v1,v0//找到一个神奇的地址v0+第i个字符的后四位

0x00401488 <+160>: lb v1,0(v0)//取出值

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

0x00401490 <+168>: addu v0,v0,a0

0x00401494 <+172>: sb v1,4(v0)//将字符串第i+1位置的元素存入s8+28+i的地址处

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

0x0040149c <+180>: nop

0x004014a0 <+184>: addiu v0,v0,1//循环变量i++

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

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

0x004014ac <+196>: nop

0x004014b0 <+200>: slti v0,v0,6

0x004014b4 <+204>: bnez v0,0x40142c <phase\_5+68>//循环判断

0x004014b8 <+208>: nop

0x004014bc <+212>: sb zero,34(s8)//将字符串末尾置为0

0x004014c0 <+216>: addiu v0,s8,28//整理后字符串地址

0x004014c4 <+220>: move a0,v0//存入a0

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

0x004014cc <+228>: addiu a1,v0,10160//一个新的地址

0x004014d0 <+232>: jal 0x401cf8 <strings\_not\_equal>

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

### 个人理解C++代码

string s;

string ans="giants";

string temp="isrveawhobpnutfg"

char str[6];

cin>>s;

if(strlen(s)!=6)

explode\_bomb();

else

{

for(int i=0;i<6;i++)

{

char c=s[i];

int index=c&15;

str[i]=temp[index];

}

if(str==ans)

phase\_6()

else

explode\_bomb();

}

## 实验六

expe6

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>

0x0040153c <+60>: nop

0x00401540 <+64>: lw gp,16(s8)

0x00401544 <+68>: sw zero,28(s8)

0x00401548 <+72>: b 0x40163c <phase\_6+316>

0x0040154c <+76>: nop

0x00401550 <+80>: lw v0,28(s8)//取i

0x00401554 <+84>: nop

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

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

0x00401560 <+96>: addu v0,v1,v0//s8+24+4\*i

0x00401564 <+100>: lw v0,12(v0)//取出第i个数

0x00401568 <+104>: nop

0x0040156c <+108>: slti v0,v0,7//数小于7置为1

0x00401570 <+112>: beqz v0,0x40159c <phase\_6+156>//如果等于0则爆炸

0x00401574 <+116>: nop

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

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)//取出第i个数

0x00401590 <+144>: nop

0x00401594 <+148>: bgtz v0,0x4015a8 <phase\_6+168>//如果比0大才能跳转,否则爆炸

0x00401598 <+152>: nop

0x0040159c <+156>: jal 0x4021f0 <explode\_bomb>

0x004015a0 <+160>: nop

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

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

0x004015ac <+172>: nop

0x004015b0 <+176>: addiu v0,v0,1//i++

0x004015b4 <+180>: sw v0,24(s8)//i+1存入s8+24处,记为j

0x004015b8 <+184>: b 0x401618 <phase\_6+280>

0x004015bc <+188>: nop

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

0x004015c4 <+196>: nop

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

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

0x004015d0 <+208>: addu v0,v1,v0

0x004015d4 <+212>: lw v1,12(v0)//取出输入的第i个数

0x004015d8 <+216>: lw v0,24(s8)//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)//取出输入的第j个数

0x004015f0 <+240>: nop

0x004015f4 <+244>: bne v1,v0,0x401608 <phase\_6+264>//若两个相等,则爆炸

0x004015f8 <+248>: nop

0x004015fc <+252>: jal 0x4021f0 <explode\_bomb>

0x00401600 <+256>: nop

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

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

0x0040160c <+268>: nop

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

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

0x00401618 <+280>: lw v0,24(s8)//j

0x0040161c <+284>: nop

0x00401620 <+288>: slti v0,v0,6//j<6循环

0x00401624 <+292>: bnez v0,0x4015c0 <phase\_6+192>

0x00401628 <+296>: nop

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

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

0x00401648 <+328>: bnez v0,0x401550 <phase\_6+80>

0x0040164c <+332>: nop

0x00401650 <+336>: sw zero,28(s8)//j=0 i=6

0x00401654 <+340>: b 0x4016f8 <phase\_6+504>

0x00401658 <+344>: nop

0x0040165c <+348>: lui v0,0x41

0x00401660 <+352>: addiu v0,v0,12592//新地址

0x00401664 <+356>: sw v0,32(s8)//将新地址存入s8+32处

0x00401668 <+360>: li v0,1

0x0040166c <+364>: sw v0,24(s8)//j=1s

0x00401670 <+368>: b 0x40169c <phase\_6+412>

0x00401674 <+372>: nop

0x00401678 <+376>: lw v0,32(s8)//读取新地址

0x0040167c <+380>: nop

0x00401680 <+384>: lw v0,8(v0)//找到新地址+8的元素

0x00401684 <+388>: nop

0x00401688 <+392>: sw v0,32(s8)//将新地址+8的元素存入新地址

0x0040168c <+396>: lw v0,24(s8)//读入j

0x00401690 <+400>: nop

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

0x00401698 <+408>: sw v0,24(s8)/j++

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

0x004016a0 <+416>: nop

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

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

0x004016ac <+428>: addu v0,v1,v0

0x004016b0 <+432>: lw v1,12(v0)//取出第i个数

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

0x004016b8 <+440>: nop

0x004016bc <+444>: slt v0,v0,v1//如果j<a[i]就继续循环

0x004016c0 <+448>: bnez v0,0x401678 <phase\_6+376>

0x004016c4 <+452>: nop

0x004016c8 <+456>: lw v0,28(s8)//i

0x004016cc <+460>: nop

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

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

0x004016d8 <+472>: addu v0,v1,v0//v0=s8+24+4i

0x004016dc <+476>: lw v1,32(s8)//节点元素

0x004016e0 <+480>: nop

0x004016e4 <+484>: sw v1,36(v0)//元素存入s8+60+4i

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)//i++

0x004016fc <+508>: nop

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

0x00401704 <+516>: bnez v0,0x40165c <phase\_6+348>

0x00401708 <+520>: nop

0x0040170c <+524>: lw v0,60(s8)//读取数组首元素

0x00401710 <+528>: nop

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

0x00401718 <+536>: li v0,1

0x0040171c <+540>: sw v0,28(s8)//i=1

0x00401720 <+544>: b 0x40177c <phase\_6+636>

0x00401724 <+548>: nop

0x00401728 <+552>: lw v0,28(s8)

0x0040172c <+556>: nop

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

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

0x00401738 <+568>: addu v0,v1,v0

0x0040173c <+572>: lw v1,36(v0)//v1=s8+60+4i,取数组结点的地址

0x00401740 <+576>: lw v0,32(s8)//链表首元素地址

0x00401744 <+580>: nop

0x00401748 <+584>: sw v1,8(v0)//存入s8+32+4i

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=s8+60+4i,取数组节点

0x00401764 <+612>: nop

0x00401768 <+616>: sw v0,32(s8)//存入s8+32

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

0x00401770 <+624>: nop

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

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

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

0x00401780 <+640>: nop

0x00401784 <+644>: slti v0,v0,6

0x00401788 <+648>: bnez v0,0x401728 <phase\_6+552>

0x0040178c <+652>: nop

0x00401790 <+656>: lw v0,32(s8)

0x00401794 <+660>: nop

0x00401798 <+664>: sw zero,8(v0)

0x0040179c <+668>: lw v0,60(s8)

0x004017a0 <+672>: nop

0x004017a4 <+676>: sw v0,32(s8)//修复头结点!

0x004017a8 <+680>: sw zero,28(s8)//i=0

0x004017ac <+684>: b 0x401878 <phase\_6+888>

0x004017b0 <+688>: nop

0x004017b4 <+692>: lw v0,-32660(gp)//学号最后一位

0x004017b8 <+696>: nop

0x004017bc <+700>: lw v0,44(v0)

0x004017c0 <+704>: nop

0x004017c4 <+708>: andi v0,v0,0x1//判断奇数偶数

0x004017c8 <+712>: andi v0,v0,0xff

0x004017cc <+716>: beqz v0,0x401818 <phase\_6+792>//偶数跳

0x004017d0 <+720>: nop

0x004017d4 <+724>: lw v0,32(s8)//头结点地址

0x004017d8 <+728>: nop

0x004017dc <+732>: lw v1,0(v0)//头结点值

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

0x004017e4 <+740>: nop

0x004017e8 <+744>: lw v0,8(v0)//第二个结点值

0x004017ec <+748>: nop

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

0x004017f4 <+756>: nop

0x004017f8 <+760>: slt v0,v1,v0

0x004017fc <+764>: beqz v0,0x401854 <phase\_6+852>//若v1<v0,则爆炸,则必须降序排列

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)

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

0x00401828 <+808>: nop

0x0040182c <+812>: lw v0,8(v0)

0x00401830 <+816>: nop

0x00401834 <+820>: lw v0,0(v0)

0x00401838 <+824>: nop

0x0040183c <+828>: slt v0,v0,v1

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)

0x00401860 <+864>: nop

0x00401864 <+868>: sw v0,32(s8)///第二个结点存入第一个结点

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

0x0040186c <+876>: nop

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

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

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

0x0040187c <+892>: nop

0x00401880 <+896>: slti v0,v0,5//i<5

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

### 个人理解C++代码

int a[6];

for (int i = 0;i < 6;i++)

cin>>a[i];

for (int i = 0;i < 6;i++) {

if (!(i > 0 && i < 7) explode\_bomb();

for (int j = i + 1;j < 6;j++)

if (a[i] == a[j]) explode\_bomb();

}

chain c = {0x0fd, 0x2d5, 0x12d, 0x3e5, 0x0d4, 0x1b0};

chainNode res[6];

for (int i = 0;i < 6;i++) {

chainNode \*node = c.firstNode;

for (int j = 1;j < a[i];j++)

node = node->next;

res[i] = node;

}

chainNode \*node = res[0];

for (int i = 1;i < 6;i++)

{

node->next = res[i];

node = res[i];

}

node->next = null;

if (ID & 1)

{

node = c.firstNode;

for (int i = 0;i < 5;i++)

{

if (node->element < node->next->element)

explode\_bomb();

node=node->next;

}

}

else

{

node = c.firstNode;

for (int i = 0;i < 5;i++)

{

if (node->element >node->next->element)

explode\_bomb();

node=node->next;

}

}

## 秘密炸弹

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>//读取输入的字符串或数字

0x004019b0 <+32>: nop

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

0x004019b8 <+40>: sw v0,28(s8)//将输入存入28(s8)

0x004019bc <+44>: lw v0,28(s8)

0x004019c0 <+48>: nop

0x004019c4 <+52>: move a0,v0 //参数1 : 输入

0x004019c8 <+56>: move a1,zero//参数2: 0

0x004019cc <+60>: li a2,10 //参数3 : 10

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

0x004019d4 <+68>: nop

0x004019d8 <+72>: move t9,v0

0x004019dc <+76>: jalr t9 //进入某一个函数,改变v0的值

0x004019e0 <+80>: nop

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

0x004019e8 <+88>: sw v0,24(s8) //将此时的v0存入s8+24位置

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

0x004019f0 <+96>: nop

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

0x004019f8 <+104>: sltiu v0,v0,1001 //输入-1<1001 设为1 ,正常跳转,否则爆炸

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 //参数1: 0x41+12676,a0处存的东西为<ni>36

0x00401a18 <+136>: lw a1,24(s8)////参数2: 输入数字

0x00401a1c <+140>: jal 0x4018a4 <fun7> //进入fun7改变v0

0x00401a20 <+144>: nop

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

0x00401a28 <+152>: move v1,v0

0x00401a2c <+156>: li v0,7 //将此时v0与7进行比较

0x00401a30 <+160>: beq v1,v0,0x401a44 <secret\_phase+180> //相等就过了

0x00401a34 <+164>: nop

0x00401a38 <+168>: jal 0x4021f0 <explode\_bomb>//不相等就爆炸

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

### Func7

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

0x004018a8 <+4>: sw ra,28(sp)

0x004018ac <+8>: sw s8,24(sp)

0x004018b0 <+12>: move s8,sp

0x004018b4 <+16>: sw a0,32(s8) // 地址存入s8(32)

0x004018b8 <+20>: sw a1,36(s8) //输入数字存入s8(36)

0x004018bc <+24>: lw v0,32(s8) // 取出对应的结点值

0x004018c0 <+28>: nop

0x004018c4 <+32>: bnez v0,0x4018d8 <fun7+52> //必须v0要不等于0(其实也就是不为空)才能正常执行函数,否则直接设为-1,返回

0x004018c8 <+36>: nop

0x004018cc <+40>: li v0,-1

0x004018d0 <+44>: b 0x401978 <fun7+212>

0x004018d4 <+48>: nop

0x004018d8 <+52>: lw v0,32(s8) //取出对应的结点值

0x004018dc <+56>: nop

0x004018e0 <+60>: lw v1,0(v0) //偏移量为0赋给v1

0x004018e4 <+64>: lw v0,36(s8) //读取输入数字

0x004018e8 <+68>: nop

0x004018ec <+72>: slt v0,v0,v1 //如果输入数字小于结点值 就不跳转,否则跳转

0x004018f0 <+76>: beqz v0,0x401924 <fun7+128>

0x004018f4 <+80>: nop

0x004018f8 <+84>: lw v0,32(s8) //读取结点

0x004018fc <+88>: nop

0x00401900 <+92>: lw v0,4(v0) //偏移量为4赋给v0, 找到左孩子

0x00401904 <+96>: nop

0x00401908 <+100>: move a0,v0 // 第一个参数更新为左孩子

0x0040190c <+104>: lw a1,36(s8) // 第二个参数仍然为输入的数

0x00401910 <+108>: jal 0x4018a4 <fun7> //递归

0x00401914 <+112>: nop

0x00401918 <+116>: sll v0,v0,0x1 //递归完成后, 将v0左移一位 返回v0

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

0x00401920 <+124>: nop //跳转到这

0x00401924 <+128>: lw v0,32(s8) // //读取结点

0x00401928 <+132>: nop

0x0040192c <+136>: lw v1,0(v0) //读取结点值

0x00401930 <+140>: lw v0,36(s8) //读取输入的数

0x00401934 <+144>: nop

0x00401938 <+148>: slt v0,v1,v0 //如果结点值小于输入的数, 就不进行跳转(即不相等)

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 //第一个参数更新为右孩子

0x00401958 <+180>: lw a1,36(s8) //第二个参数仍然为输入的数

0x0040195c <+184>: jal 0x4018a4 <fun7>

0x00401960 <+188>: nop

0x00401964 <+192>: sll v0,v0,0x1 ////递归完成后, 将v0左移一位+1.返回v0

0x00401968 <+196>: addiu v0,v0,1

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

0x00401970 <+204>: nop

0x00401974 <+208>: move v0,zero //将v0置为0返回

0x00401978 <+212>: move sp,s8//相当于结束了函数

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