**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：**二进制炸弹拆除 | | **学号：202100130122** |
| **班级：计机 21.6** | **姓名： 吕鹏洋** | |
| **Email：2656678633@qq.com** | | |
| **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。 | | |
| **实验软件和硬件环境：**   由四片四核龙芯 3A 处理器构成的 16 核 CC-NUMA 结构、内可配置外可扩展结构的实验硬件平台——多路处理器计算机教学实验系统   gdb   bomb 可执行文件（MIPS） | | |
| **实验原理和方法：**  本实验设计为一个黑客拆解二进制炸弹的游戏。我们仅给黑客（同学）提供一个 MIPS 二进制可执行文件 bomb，不提供源代码（提供汇编代码帮助理解）。  程序运行中有 6 个关卡（6 个 phase），每个关卡需要用户输入正确的字符串或 数字才能通关，否则会引爆炸弹（打印出一条错误信息）！要求同学运用 GDB 调试工具，通过分析汇编代码，找到在每个 phase 程序 段中，引导程序跳转到“explode\_bomb”程序段的地方，并分析其成功跳转的条件，以此为突破口寻找应该在命令行输入何种字符串来通关。 | | |
| **实验步骤：**  **汇编语言翻译注释：**  **一：00400d6c <phase\_1>:**  **400d6c: 27bdffe0 addiu sp,sp,-32**  **400d70: afbf001c sw ra,28(sp)**  **400d74: afbe0018 sw s8,24(sp)**  **400d78: 03a0f021 move s8,sp**  **400d7c: afc40020 sw a0,32(s8)**  **400d80: 8fc40020 lw a0,32(s8)**  **400d84: 3c020040 lui v0,0x40**  **400d88: 2445276c addiu a1,v0,10092**  **400d8c: 0c10073e jal 401cf8 <strings\_not\_equal>**  **400d90: 00000000 nop**  **400d94: 10400003 beqz v0,400da4 <phase\_1+0x38>**  **400d98: 00000000 nop**  **400d9c: 0c10087c jal 4021f0 <explode\_bomb>**  **400da0: 00000000 nop**  **400da4: 03c0e821 move sp,s8**  **400da8: 8fbf001c lw ra,28(sp)**  **400dac: 8fbe0018 lw s8,24(sp)**  **400db0: 27bd0020 addiu sp,sp,32**  **400db4: 03e00008 jr ra**  **400db8: 00000000 nop**  **C语言版：**  **string input;**  **cin >> input;**  **string answer = "Let's begin now!";**  **if (input == answer)**  **return;**  **else**  **explode\_bomb();**  **二：00400dbc <phase\_2>:**  **400dbc: 27bdffc0 addiu sp,sp,-64**  **400dc0: afbf003c sw ra,60(sp)**  **400dc4: afbe0038 sw s8,56(sp)**  **400dc8: 03a0f021 move s8,sp**  **400dcc: 3c1c0042 lui gp,0x42**  **400dd0: 279cb190 addiu gp,gp,-20080**  **400dd4: afbc0010 sw gp,16(sp)**  **400dd8: afc40040 sw a0,64(s8)**  **400ddc: 27c2001c addiu v0,s8,28**  **400de0: 8fc40040 lw a0,64(s8)**  **400de4: 00402821 move a1,v0**  **400de8: 0c1006ea jal 401ba8 <read\_six\_numbers> //读入6个数字，这之后查看$s8发现从第初始位置往后28位开始存储我的输入，每个四位存储一个**  **400dec: 00000000 nop**  **400df0: 8fdc0010 lw gp,16(s8)**  **400df4: 8fc3001c lw v1,28(s8) //m[$s8+28]存有读入的第一个数，将其存入$v1**  **400df8: 24020001 li v0,1**  **400dfc: 10620004 beq v1,v0,400e10 <phase\_2+0x54> //判定$v1是否为1，不是则引爆**  **400e00: 00000000 nop**  **400e04: 0c10087c jal 4021f0 <explode\_bomb>**  **400e08: 00000000 nop**  **400e0c: 8fdc0010 lw gp,16(s8)**  **400e10: 24020001 li v0,1 //设定循环单次表达式$v0 = 1(i = 1)**  **400e14: afc20018 sw v0,24(s8) //将$v0(i)存入栈，这时s8+24=1**  **400e18: 10000023 b 400ea8 <phase\_2+0xec> //循环开始，跳至条件表达式**  **400e1c: 00000000 nop**  **400e20: 8fc20018 lw v0,24(s8) //将之前存在栈中的变量i取出，这里将v0从1恢复成了i**  **400e24: 00000000 nop**  **400e28: 2442ffff addiu v0,v0,-1 //$v0自减**  **400e2c: 00021080 sll v0,v0,0x2 //$v0 = $v0 \* 4（拓到1 int长度）**  **400e30: 27c30018 addiu v1,s8,24 //第i个数的存放位置为m[$s8 + 24 + i \* 4]（一个int型变量占4个字节），这里+24找到第一个数的前一字节**  **400e34: 00621021 addu v0,v1,v0//这里加上V0找到第i-1个数的位置并存入v0**  **400e38: 8c440004 lw a0,4(v0) //加4就找到了第i个数的位置，将第i个数的值存入$a0（为什么之前要v0自减？）**  **400e3c: 2403000c li v1,12**  **400e40: 8fc20018 lw v0,24(s8) //将之前存在栈中的变量i取出**  **400e44: 00000000 nop**  **400e48: 00621023 subu v0,v1,v0 //$v0 = $v1(12) - $v0(i)**  **400e4c: 8f83806c lw v1,-32660(gp) //读得输入的学号**  **400e50: 00021080 sll v0,v0,0x2 //$v0 = $v0 \* 4（拓到1 int长度）**  **400e54: 00621021 addu v0,v1,v0 //$v0 = $v1 + $v0**  **400e58: 8c420000 lw v0,0(v0) //将学号的倒数i位读入$v0中**  **400e5c: 00000000 nop**  **400e60: 00820018 mult a0,v0 //$a0与$v0相乘，$a0存的是第i个数，V0是学号倒数第i位**  **400e64: 00002012 mflo a0 //将结果存入$a0**  **400e68: 8fc20018 lw v0,24(s8) //将之前存在栈中的变量i取出**  **400e6c: 00000000 nop**  **400e70: 00021080 sll v0,v0,0x2 //$v0 = $v0 \* 4（拓到1 int长度）**  **400e74: 27c30018 addiu v1,s8,24 //第i个数得存放位置为m[$s8 + 24 + i \* 4]（一个int型变量占4个字节）**  **400e78: 00621021 addu v0,v1,v0 //得到第i个数的位置**  **400e7c: 8c420004 lw v0,4(v0) //将第i + 1个数的值存入$v0**  **400e80: 00000000 nop**  **400e84: 10820004 beq a0,v0,400e98 <phase\_2+0xdc> //比较$a0（第i个数与学号的倒数i位相乘的结果）与$v0(第i + 1个数)，若不相等则引爆**  **400e88: 00000000 nop**  **400e8c: 0c10087c jal 4021f0 <explode\_bomb>**  **400e90: 00000000 nop**  **400e94: 8fdc0010 lw gp,16(s8) //末尾循环体**  **400e98: 8fc20018 lw v0,24(s8)**  **400e9c: 00000000 nop**  **400ea0: 24420001 addiu v0,v0,1 //i自加**  **400ea4: afc20018 sw v0,24(s8) //将i存入栈中**  **400ea8: 8fc20018 lw v0,24(s8)//取出i，v0=i**  **400eac: 00000000 nop**  **400eb0: 28420006 slti v0,v0,6 //判断循环是否结束，相当于i != 6(i < 6) 若v0<6,则v0=1,否则v0=0**  **400eb4: 1440ffda bnez v0,400e20 <phase\_2+0x64>//如果v0！=0，则跳转到指定地址，继续循环**  **400eb8: 00000000 nop**  **400ebc: 03c0e821 move sp,s8**  **400ec0: 8fbf003c lw ra,60(sp)**  **400ec4: 8fbe0038 lw s8,56(sp)**  **400ec8: 27bd0040 addiu sp,sp,64**  **400ecc: 03e00008 jr ra**  **400ed0: 00000000 nop**  **三：00400ed4 <phase\_3>:**  **400ed4: 27bdffc8 addiu sp,sp,-56**  **400ed8: afbf0034 sw ra,52(sp)**  **400edc: afbe0030 sw s8,48(sp)**  **400ee0: 03a0f021 move s8,sp**  **400ee4: 3c1c0042 lui gp,0x42**  **400ee8: 279cb190 addiu gp,gp,-20080**  **400eec: afbc0018 sw gp,24(sp)**  **400ef0: afc40038 sw a0,56(s8)**  **400ef4: 8fc40038 lw a0,56(s8)**  **400ef8: 3c020040 lui v0,0x40**  **400efc: 24452780 addiu a1,v0,10112**  **400f00: 27c3002c addiu v1,s8,44**  **400f04: 27c20028 addiu v0,s8,40**  **400f08: 27c60024 addiu a2,s8,36**  **400f0c: afa60010 sw a2,16(sp)**  **400f10: 00603021 move a2,v1**  **400f14: 00403821 move a3,v0**  **400f18: 8f828084 lw v0,-32636(gp)**  **400f1c: 00000000 nop**  **400f20: 0040c821 move t9,v0**  **400f24: 0320f809 jalr t9**  **400f28: 00000000 nop**  **400f2c: 8fdc0018 lw gp,24(s8)**  **400f30: 28420003 slti v0,v0,3 //判定输入的数据个数是否为3**  **400f34: 10400004 beqz v0,400f48 <phase\_3+0x74>**  **400f38: 00000000 nop**  **400f3c: 0c10087c jal 4021f0 <explode\_bomb>**  **400f40: 00000000 nop**  **400f44: 8fdc0018 lw gp,24(s8)**  **400f48: 8fc2002c lw v0,44(s8) //读入输入的第一个数至$v0**  **400f4c: 00000000 nop**  **400f50: 2c430008 sltiu v1,v0,8 //判断$v0是否小于8,是则继续,否则引爆**  **400f54: 1060008e beqz v1,401190 <phase\_3+0x2bc>**  **400f58: 00000000 nop**  **400f5c: 00021880 sll v1,v0,0x2 //$v1 = $v0 \* 4，拓至1 int型变量长度**  **400f60: 3c020040 lui v0,0x40**  **400f64: 2442278c addiu v0,v0,10124**  **400f68: 00621021 addu v0,v1,v0**  **400f6c: 8c420000 lw v0,0(v0) //$v0对应case语句的地址，用x $v0查看**  **400f70: 00000000 nop**  **400f74: 00400008 jr v0 //switch语句，跳至相应的case语句**  **400f78: 00000000 nop //case 0:**  **400f7c: 24020071 li v0,113 //q的ASCII码**  **400f80: a3c20020 sb v0,32(s8) //存入1byte的数据**  **400f84: 8f82806c lw v0,-32660(gp)**  **400f88: 00000000 nop**  **400f8c: 8c43002c lw v1,44(v0) //读入学号的最后一位**  **400f90: 8fc20024 lw v0,36(s8) //读入输入数据的第三个数**  **400f94: 00000000 nop**  **400f98: 00620018 mult v1,v0**  **400f9c: 00001812 mflo v1 //$v1 = $v1 \* $v0**  **400fa0: 24020309 li v0,777 //判断是否为777(=3\*=7\*111)，是则break，否则引爆**  **400fa4: 10620081 beq v1,v0,4011ac <phase\_3+0x2d8>**  **400fa8: 00000000 nop**  **400fac: 0c10087c jal 4021f0 <explode\_bomb>**  **400fb0: 00000000 nop**  **400fb4: 8fdc0018 lw gp,24(s8)**  **400fb8: 1000008f b 4011f8 <phase\_3+0x324>**  **400fbc: 00000000 nop //case 1:**  **400fc0: 24020062 li v0,98 //b的ASCII码**  **400fc4: a3c20020 sb v0,32(s8) //存入1byte的数据**  **400fc8: 8f82806c lw v0,-32660(gp)**  **400fcc: 00000000 nop**  **400fd0: 8c43002c lw v1,44(v0) //读入学号的最后一位**  **400fd4: 8fc20024 lw v0,36(s8) //读入输入数据的第三个数**  **400fd8: 00000000 nop**  **400fdc: 00620018 mult v1,v0**  **400fe0: 00001812 mflo v1 //$v1 = $v1 \* $v0**  **400fe4: 240200d6 li v0,214 //判断是否为214(=2\*107)，是则break，否则引爆**  **400fe8: 10620073 beq v1,v0,4011b8 <phase\_3+0x2e4>**  **400fec: 00000000 nop**  **400ff0: 0c10087c jal 4021f0 <explode\_bomb>**  **400ff4: 00000000 nop**  **400ff8: 8fdc0018 lw gp,24(s8)**  **400ffc: 1000007e b 4011f8 <phase\_3+0x324>**  **401000: 00000000 nop //case 2:**  **401004: 24020062 li v0,98 //b的ASCII码**  **401008: a3c20020 sb v0,32(s8) //存入1byte的数据**  **40100c: 8f82806c lw v0,-32660(gp)**  **401010: 00000000 nop**  **401014: 8c43002c lw v1,44(v0) //读入学号的最后一位**  **401018: 8fc20024 lw v0,36(s8) //读入输入数据的第三个数**  **40101c: 00000000 nop**  **401020: 00620018 mult v1,v0**  **401024: 00001812 mflo v1 //$v1 = $v1 \* $v0**  **401028: 240202f3 li v0,755 //判断是否为755(=5\*151)，是则break，否则引爆**  **40102c: 10620065 beq v1,v0,4011c4 <phase\_3+0x2f0>**  **401030: 00000000 nop**  **401034: 0c10087c jal 4021f0 <explode\_bomb>**  **401038: 00000000 nop**  **40103c: 8fdc0018 lw gp,24(s8)**  **401040: 1000006d b 4011f8 <phase\_3+0x324>**  **401044: 00000000 nop //case 3:**  **401048: 2402006b li v0,107 //k的ASCII码**  **40104c: a3c20020 sb v0,32(s8) //存入1byte的数据**  **401050: 8f82806c lw v0,-32660(gp)**  **401054: 00000000 nop**  **401058: 8c43002c lw v1,44(v0) //读入学号的最后一位**  **40105c: 8fc20024 lw v0,36(s8) //读入输入数据的第三个数**  **401060: 00000000 nop**  **401064: 00620018 mult v1,v0**  **401068: 00001012 mflo v0 //$v0 = $v1 \* $v0**  **40106c: 10400058 beqz v0,4011d0 <phase\_3+0x2fc> //判断$v0是否等于0，是则break，否则引爆**  **401070: 00000000 nop**  **401074: 0c10087c jal 4021f0 <explode\_bomb>**  **401078: 00000000 nop**  **40107c: 8fdc0018 lw gp,24(s8)**  **401080: 1000005d b 4011f8 <phase\_3+0x324>**  **401084: 00000000 nop //case 4:**  **401088: 2402006f li v0,111 //o的ASCII码**  **40108c: a3c20020 sb v0,32(s8) //存入1byte的数据**  **401090: 8f82806c lw v0,-32660(gp)**  **401094: 00000000 nop**  **401098: 8c43002c lw v1,44(v0) //读入学号的最后一位**  **40109c: 8fc20024 lw v0,36(s8) //读入输入数据的第三个数**  **4010a0: 00000000 nop**  **4010a4: 00620018 mult v1,v0**  **4010a8: 00001812 mflo v1 //$v1 = $v1 \* $v0**  **4010ac: 240200e4 li v0,228 //判断是否为228(=2\*114=4\*57)，是则break，否则引爆 ???**  **4010b0: 1062004a beq v1,v0,4011dc <phase\_3+0x308>**  **4010b4: 00000000 nop**  **4010b8: 0c10087c jal 4021f0 <explode\_bomb>**  **4010bc: 00000000 nop**  **4010c0: 8fdc0018 lw gp,24(s8)**  **4010c4: 1000004c b 4011f8 <phase\_3+0x324>**  **4010c8: 00000000 nop //case 5:**  **4010cc: 24020074 li v0,116 //t的ASCII码**  **4010d0: a3c20020 sb v0,32(s8) //存入1byte的数据**  **4010d4: 8f82806c lw v0,-32660(gp)**  **4010d8: 00000000 nop**  **4010dc: 8c43002c lw v1,44(v0) //读入学号的最后一位**  **4010e0: 8fc20024 lw v0,36(s8) //读入输入数据的第三个数**  **4010e4: 00000000 nop**  **4010e8: 00620018 mult v1,v0**  **4010ec: 00001812 mflo v1 //$v1 = $v1 \* $v0**  **4010f0: 24020201 li v0,513 //判断是否为513(=3\*171=9\*57)，是则break，否则引爆**  **4010f4: 1062003c beq v1,v0,4011e8 <phase\_3+0x314>**  **4010f8: 00000000 nop**  **4010fc: 0c10087c jal 4021f0 <explode\_bomb>**  **401100: 00000000 nop**  **401104: 8fdc0018 lw gp,24(s8)**  **401108: 1000003b b 4011f8 <phase\_3+0x324>**  **40110c: 00000000 nop //case 6:**  **401110: 24020076 li v0,118 //v的ASCII码**  **401114: a3c20020 sb v0,32(s8) //存入1byte的数据**  **401118: 8f82806c lw v0,-32660(gp)**  **40111c: 00000000 nop**  **401120: 8c43002c lw v1,44(v0) //读入学号的最后一位**  **401124: 8fc20024 lw v0,36(s8) //读入输入数据的第三个数**  **401128: 00000000 nop**  **40112c: 00620018 mult v1,v0**  **401130: 00001812 mflo v1 //$v1 = $v1 \* $v0**  **401134: 2402030c li v0,780 //判断是否为780(=2\*390=3\*260=4\*195=5\*156=6\*130)，是则break，否则引爆**  **401138: 10620004 beq v1,v0,40114c <phase\_3+0x278>**  **40113c: 00000000 nop**  **401140: 0c10087c jal 4021f0 <explode\_bomb>**  **401144: 00000000 nop //case 7:**  **401148: 8fdc0018 lw gp,24(s8) ???**  **40114c: 24020062 li v0,98 //b的ASCII码**  **401150: a3c20020 sb v0,32(s8) //存入1byte的数据**  **401154: 8f82806c lw v0,-32660(gp)**  **401158: 00000000 nop**  **40115c: 8c43002c lw v1,44(v0) //读入学号的最后一位**  **401160: 8fc20024 lw v0,36(s8) //读入输入数据的第三个数**  **401164: 00000000 nop**  **401168: 00620018 mult v1,v0**  **40116c: 00001812 mflo v1 //$v1 = $v1 \* $v0**  **401170: 24020338 li v0,824 //判断是否为824(=2\*412=4\*206)，是则break，否则引爆**  **401174: 1062001f beq v1,v0,4011f4 <phase\_3+0x320>**  **401178: 00000000 nop**  **40117c: 0c10087c jal 4021f0 <explode\_bomb>**  **401180: 00000000 nop**  **401184: 8fdc0018 lw gp,24(s8)**  **401188: 1000001b b 4011f8 <phase\_3+0x324>**  **40118c: 00000000 nop**  **401190: 24020078 li v0,120**  **401194: a3c20020 sb v0,32(s8)**  **401198: 0c10087c jal 4021f0 <explode\_bomb>**  **40119c: 00000000 nop**  **4011a0: 8fdc0018 lw gp,24(s8)**  **4011a4: 10000014 b 4011f8 <phase\_3+0x324>**  **4011a8: 00000000 nop**  **4011ac: 00000000 nop**  **4011b0: 10000011 b 4011f8 <phase\_3+0x324>**  **4011b4: 00000000 nop**  **4011b8: 00000000 nop**  **4011bc: 1000000e b 4011f8 <phase\_3+0x324>**  **4011c0: 00000000 nop**  **4011c4: 00000000 nop**  **4011c8: 1000000b b 4011f8 <phase\_3+0x324>**  **4011cc: 00000000 nop**  **4011d0: 00000000 nop**  **4011d4: 10000008 b 4011f8 <phase\_3+0x324>**  **4011d8: 00000000 nop**  **4011dc: 00000000 nop**  **4011e0: 10000005 b 4011f8 <phase\_3+0x324>**  **4011e4: 00000000 nop**  **4011e8: 00000000 nop**  **4011ec: 10000002 b 4011f8 <phase\_3+0x324>**  **4011f0: 00000000 nop**  **4011f4: 00000000 nop**  **4011f8: 83c20028 lb v0,40(s8) //输入的第二个字符数据**  **4011fc: 83c30020 lb v1,32(s8) //将case语句中存的字符取出**  **401200: 00000000 nop**  **401204: 10620004 beq v1,v0,401218 <phase\_3+0x344> //比较，不相等则引爆**  **401208: 00000000 nop**  **40120c: 0c10087c jal 4021f0 <explode\_bomb>**  **401210: 00000000 nop**  **401214: 8fdc0018 lw gp,24(s8)**  **401218: 03c0e821 move sp,s8**  **40121c: 8fbf0034 lw ra,52(sp)**  **401220: 8fbe0030 lw s8,48(sp)**  **401224: 27bd0038 addiu sp,sp,56**  **401228: 03e00008 jr ra**  **40122c: 00000000 nop**  **00401230 <func4>:**  **401230: 27bdffd8 addiu sp,sp,-40**  **401234: afbf0024 sw ra,36(sp)**  **401238: afbe0020 sw s8,32(sp)**  **40123c: afb0001c sw s0,28(sp)**  **401240: 03a0f021 move s8,sp**  **401244: afc40028 sw a0,40(s8)**  **401248: 8fc20028 lw v0,40(s8)//v0为输入的参数**  **40124c: 00000000 nop**  **401250: 28420002 slti v0,v0,2 //判断$v0是否小于2，是则$v0 = 1,否则为0**  **401254: 14400011 bnez v0,40129c <func4+0x6c> //$v0 != 0则跳转，return 1，这两句合起来就是if(x<2)return 1;**  **401258: 00000000 nop**  **40125c: 8fc20028 lw v0,40(s8)//把输入的参数x重新赋值给v0**  **401260: 00000000 nop**  **401264: 2442ffff addiu v0,v0,-1**  **401268: 00402021 move a0,v0//a0自减，也就是x--**  **40126c: 0c10048c jal 401230 <func4> //递归f(--x)**  **401270: 00000000 nop**  **401274: 00408021 move s0,v0 //将f(x - 1)的结果存入$s0**  **401278: 8fc20028 lw v0,40(s8)**  **40127c: 00000000 nop**  **401280: 2442fffe addiu v0,v0,-2**  **401284: 00402021 move a0,v0**  **401288: 0c10048c jal 401230 <func4> //递归引用自身f(x - 2)**  **40128c: 00000000 nop**  **401290: 02021021 addu v0,s0,v0 //$v0 = f(x - 1) + f(x - 2)，返回$v0**  **401294: 10000002 b 4012a0 <func4+0x70>**  **401298: 00000000 nop**  **40129c: 24020001 li v0,1 //return 1**  **4012a0: 03c0e821 move sp,s8**  **4012a4: 8fbf0024 lw ra,36(sp)**  **4012a8: 8fbe0020 lw s8,32(sp)**  **4012ac: 8fb0001c lw s0,28(sp)**  **4012b0: 27bd0028 addiu sp,sp,40**  **4012b4: 03e00008 jr ra**  **4012b8: 00000000 nop**  **四：004012bc <phase\_4>:**  **4012bc: 27bdffd8 addiu sp,sp,-40**  **4012c0: afbf0024 sw ra,36(sp)**  **4012c4: afbe0020 sw s8,32(sp)**  **4012c8: 03a0f021 move s8,sp**  **4012cc: 3c1c0042 lui gp,0x42**  **4012d0: 279cb190 addiu gp,gp,-20080**  **4012d4: afbc0010 sw gp,16(sp)**  **4012d8: afc40028 sw a0,40(s8)**  **4012dc: 8fc30028 lw v1,40(s8)//v1是输入的数**  **4012e0: 3c020040 lui v0,0x40**  **4012e4: 244227ac addiu v0,v0,10156**  **4012e8: 00602021 move a0,v1**  **4012ec: 00402821 move a1,v0**  **4012f0: 27c20018 addiu v0,s8,24**  **4012f4: 00403021 move a2,v0**  **4012f8: 8f828084 lw v0,-32636(gp)**  **4012fc: 00000000 nop**  **401300: 0040c821 move t9,v0**  **401304: 0320f809 jalr t9**  **401308: 00000000 nop**  **40130c: 8fdc0010 lw gp,16(s8)**  **401310: 00401821 move v1,v0**  **401314: 24020001 li v0,1**  **401318: 14620005 bne v1,v0,401330 <phase\_4+0x74> //$v0 != 1则引爆**  **40131c: 00000000 nop**  **401320: 8fc20018 lw v0,24(s8)**  **401324: 00000000 nop**  **401328: 1c400005 bgtz v0,401340 <phase\_4+0x84>//v0大于0则转移**  **40132c: 00000000 nop**  **401330: 0c10087c jal 4021f0 <explode\_bomb>**  **401334: 00000000 nop**  **401338: 8fdc0010 lw gp,16(s8)**  **40133c: 00000000 nop**  **401340: 8f82806c lw v0,-32660(gp)//这里v0存入学号**  **401344: 00000000 nop**  **401348: 8c42002c lw v0,44(v0) //令$v0等于学号的最后一位**  **40134c: 00000000 nop**  **401350: 30420001 andi v0,v0,0x1 //v0&1,如果结果为0则v0为偶数，否则为奇数**  **401354: 304200ff andi v0,v0,0xff**  **401358: 10400010 beqz v0,40139c <phase\_4+0xe0> //if...else...结构，奇数（1）继续，偶数（0）跳转**  **40135c: 00000000 nop**  **401360: 8fc20018 lw v0,24(s8) //令$v0等于输入的第一个数**  **401364: 00000000 nop**  **401368: 00402021 move a0,v0 //传入参数**  **40136c: 0c10048c jal 401230 <func4> //计算f(x)**  **401370: 00000000 nop**  **401374: 8fdc0010 lw gp,16(s8)**  **401378: 00401821 move v1,v0**  **40137c: 24020008 li v0,8**  **401380: 10620013 beq v1,v0,4013d0 <phase\_4+0x114> //f(x) = $v1 != 8则引爆，则x = 5**  **401384: 00000000 nop**  **401388: 0c10087c jal 4021f0 <explode\_bomb>**  **40138c: 00000000 nop**  **401390: 8fdc0010 lw gp,16(s8)**  **401394: 1000000e b 4013d0 <phase\_4+0x114>**  **401398: 00000000 nop**  **40139c: 8fc20018 lw v0,24(s8)//之前判定的学号最后一位为偶数则跳转到这里**  **4013a0: 00000000 nop**  **4013a4: 00402021 move a0,v0 //传入参数，a0就是我输入的数**  **4013a8: 0c10048c jal 401230 <func4> //进入函数f(x)，f(x)为斐波那契数列**  **4013ac: 00000000 nop**  **4013b0: 8fdc0010 lw gp,16(s8)**  **4013b4: 00401821 move v1,v0//将计算结果存入v1**  **4013b8: 2402000d li v0,13**  **4013bc: 10620004 beq v1,v0,4013d0 <phase\_4+0x114> //v1 != 13则引爆，所以x=6**  **4013c0: 00000000 nop**  **4013c4: 0c10087c jal 4021f0 <explode\_bomb>**  **4013c8: 00000000 nop**  **4013cc: 8fdc0010 lw gp,16(s8)**  **4013d0: 03c0e821 move sp,s8**  **4013d4: 8fbf0024 lw ra,36(sp)**  **4013d8: 8fbe0020 lw s8,32(sp)**  **4013dc: 27bd0028 addiu sp,sp,40**  **4013e0: 03e00008 jr ra**  **4013e4: 00000000 nop**  **五：004013e8 <phase\_5>:**  **4013e8: 27bdffb8 addiu sp,sp,-72**  **4013ec: afbf0044 sw ra,68(sp)**  **4013f0: afbe0040 sw s8,64(sp)**  **4013f4: 03a0f021 move s8,sp**  **4013f8: afc40048 sw a0,72(s8) //$a0为输入的字符串，将其存到m[s8+72]的位置**  **4013fc: 8fc40048 lw a0,72(s8)**  **401400: 0c10071e jal 401c78 <string\_length>//跳转到计算字符串长度**  **401404: 00000000 nop**  **401408: 00401821 move v1,v0**  **40140c: 24020006 li v0,6**  **401410: 10620003 beq v1,v0,401420 <phase\_5+0x38> //判断输入的字符串长度是否为6，不是则引爆**  **401414: 00000000 nop**  **401418: 0c10087c jal 4021f0 <explode\_bomb>**  **40141c: 00000000 nop**  **401420: afc00018 sw zero,24(s8) //m[s8+24]存入0，相当于i = 0开始循环**  **401424: 10000020 b 4014a8 <phase\_5+0xc0>**  **401428: 00000000 nop**  **40142c: 8fc20018 lw v0,24(s8) //读得i**  **401430: 8fc30018 lw v1,24(s8) //读得i**  **401434: 8fc40048 lw a0,72(s8) //从栈中取出输入的字符串（从0开始编号）**  **401438: 00000000 nop**  **40143c: 00831821 addu v1,a0,v1//到达字符串的第i+1个字符(i从0开始)**  **401440: 80630000 lb v1,0(v1) //读取输入字符串的第i+1个字符**  **401444: 00000000 nop**  **401448: 306300ff andi v1,v1,0xff//这里为什么要先&11111111?**  **40144c: 3063000f andi v1,v1,0xf //取字符转化为二进制数（1byte=8bits）的后四位**  **//到这里v1存的是每个字符的二进制ASCII码后四位**  **401450: 00021080 sll v0,v0,0x2 //将i拓至1 int型变量长度，i\*4**  **401454: 27c40018 addiu a0,s8,24 //读得i的地址**  **401458: 00821021 addu v0,a0,v0//初始i（=0）向后移动循环次数\*4位**  **40145c: ac43000c sw v1,12(v0) //将字符后四位存入m[$s8 + 24 + 12 + i \* 4]**  **401460: 8fc40018 lw a0,24(s8) //读得i**  **401464: 8fc20018 lw v0,24(s8) //读得i**  **401468: 00000000 nop**  **40146c: 00021080 sll v0,v0,0x2 //将i拓至1 int型变量长度**  **401470: 27c30018 addiu v1,s8,24 //读得i的地址**  **401474: 00621021 addu v0,v1,v0**  **401478: 8c43000c lw v1,12(v0) //读得字符后四位，存入$v1中**  **40147c: 3c020041 lui v0,0x41**  **401480: 244230ec addiu v0,v0,12524 //找到这个内置字符串的开头位置**  **401484: 00621021 addu v0,v1,v0 //从开头地址往后移动$v1位**  **//这里v1存的是当前i层循环得到的字符的后四位**  **401488: 80430000 lb v1,0(v0) //取得内置字符串的第$v1个字符，也就是我们输入的第i个字符后四位（二进制）对应的那个字符**  **//内置字符串为 【isrveawhobpnutfg】0——15**  **40148c: 27c20018 addiu v0,s8,24**  **401490: 00441021 addu v0,v0,a0**  **401494: a0430004 sb v1,4(v0) //将获得的字符存入m[$s8 + 28 + i]（2i+4)**  **401498: 8fc20018 lw v0,24(s8) //读得i**  **40149c: 00000000 nop**  **4014a0: 24420001 addiu v0,v0,1 //i++**  **4014a4: afc20018 sw v0,24(s8)**  **4014a8: 8fc20018 lw v0,24(s8)//v0=i**  **4014ac: 00000000 nop**  **4014b0: 28420006 slti v0,v0,6 //$v0 = ($v0 < 6 ? 1 : 0)，为1再循环一次，即i < 6**  **4014b4: 1440ffdd bnez v0,40142c <phase\_5+0x44>//v0!=0则跳转,这两句合起来就是一个for循环循环六次之后跳出来继续执行**  **4014b8: 00000000 nop**  **4014bc: a3c00022 sb zero,34(s8) //将字符串尾设为'\0'**  **4014c0: 27c2001c addiu v0,s8,28**  **4014c4: 00402021 move a0,v0 //a0存入的是通过循环映射出来的字符串**  **4014c8: 3c020040 lui v0,0x40**  **4014cc: 244527b0 addiu a1,v0,10160 //a1取得的是需要比较是否相等的字符串，为giants**  **4014d0: 0c10073e jal 401cf8 <strings\_not\_equal> //比较$a0,$a1**  **4014d4: 00000000 nop**  **4014d8: 10400003 beqz v0,4014e8 <phase\_5+0x100>//也即是说映射出来的字符串需要为giants，否则会引爆**  **4014dc: 00000000 nop**  **4014e0: 0c10087c jal 4021f0 <explode\_bomb>**  **4014e4: 00000000 nop**  **4014e8: 03c0e821 move sp,s8**  **4014ec: 8fbf0044 lw ra,68(sp)**  **4014f0: 8fbe0040 lw s8,64(sp)**  **4014f4: 27bd0048 addiu sp,sp,72**  **4014f8: 03e00008 jr ra**  **4014fc: 00000000 nop**  **六：00401500 <phase\_6>:**  **401500: 27bdffa0 addiu sp,sp,-96**  **401504: afbf005c sw ra,92(sp)**  **401508: afbe0058 sw s8,88(sp)**  **40150c: 03a0f021 move s8,sp**  **401510: 3c1c0042 lui gp,0x42**  **401514: 279cb190 addiu gp,gp,-20080**  **401518: afbc0010 sw gp,16(sp)**  **40151c: afc40060 sw a0,96(s8)**  **401520: 3c020041 lui v0,0x41**  **401524: 24423130 addiu v0,v0,12592**  **401528: afc20020 sw v0,32(s8)**  **40152c: 27c20024 addiu v0,s8,36**  **401530: 8fc40060 lw a0,96(s8)**  **401534: 00402821 move a1,v0**  **401538: 0c1006ea jal 401ba8 <read\_six\_numbers> //分配空间加输入六个数字**  **40153c: 00000000 nop // 第一层循环的初始条件设定**  **401540: 8fdc0010 lw gp,16(s8)**  **401544: afc0001c sw zero,28(s8) //设置 i = 0**  **401548: 1000003c b 40163c <phase\_6+0x13c> // 跳至0x40163c处第一重循环条件判断处**  **40154c: 00000000 nop // 第一层循环中间循环体开始**  **401550: 8fc2001c lw v0,28(s8) // 获取i,v0=i**  **401554: 00000000 nop**  **401558: 00021080 sll v0,v0,0x2 //将i拓至1 int长度**  **40155c: 27c30018 addiu v1,s8,24**  **401560: 00621021 addu v0,v1,v0**  **401564: 8c42000c lw v0,12(v0) // 得到输入的第i个数(p $v0)**  **401568: 00000000 nop**  **40156c: 28420007 slti v0,v0,7 // 判断第i个数是否小于7**  **401570: 1040000a beqz v0,40159c <phase\_6+0x9c> //若第i个数大于7, 炸弹爆炸**  **401574: 00000000 nop**  **401578: 8fc2001c lw v0,28(s8) //取得循环变量i**  **40157c: 00000000 nop**  **401580: 00021080 sll v0,v0,0x2//i 变成1 int长度**  **401584: 27c30018 addiu v1,s8,24**  **401588: 00621021 addu v0,v1,v0**  **40158c: 8c42000c lw v0,12(v0)//v0等于输入的第i个数**  **401590: 00000000 nop**  **401594: 1c400004 bgtz v0,4015a8 <phase\_6+0xa8> // 判断第i个数是否大于0，若小于等于0则引爆**  **401598: 00000000 nop**  **40159c: 0c10087c jal 4021f0 <explode\_bomb>**  **4015a0: 00000000 nop**  **4015a4: 8fdc0010 lw gp,16(s8)**  **4015a8: 8fc2001c lw v0,28(s8)//取得之前的循环变量 v0=i**  **4015ac: 00000000 nop // 第二层循环的初始条件设定**  **4015b0: 24420001 addiu v0,v0,1 // j = i + 1**  **4015b4: afc20018 sw v0,24(s8) // 将j存至m[$s8 + 24]处**  **4015b8: 10000017 b 401618 <phase\_6+0x118> // 跳至0x401618条件判断处**  **4015bc: 00000000 nop // 第二层循环中间循环体开始**  **4015c0: 8fc2001c lw v0,28(s8) // 获取i**  **4015c4: 00000000 nop**  **4015c8: 00021080 sll v0,v0,0x2//i变成1 int长度**  **4015cc: 27c30018 addiu v1,s8,24**  **4015d0: 00621021 addu v0,v1,v0**  **4015d4: 8c43000c lw v1,12(v0) // 获取第i个数**  **4015d8: 8fc20018 lw v0,24(s8) // 获取j**  **4015dc: 00000000 nop**  **4015e0: 00021080 sll v0,v0,0x2**  **4015e4: 27c40018 addiu a0,s8,24**  **4015e8: 00821021 addu v0,a0,v0**  **4015ec: 8c42000c lw v0,12(v0) // 获取第j个数**  **4015f0: 00000000 nop**  **4015f4: 14620004 bne v1,v0,401608 <phase\_6+0x108> //判断第$v1(i)个数和第$v0(j)个数是否不相等，否则爆炸**  **4015f8: 00000000 nop**  **4015fc: 0c10087c jal 4021f0 <explode\_bomb>**  **401600: 00000000 nop**  **401604: 8fdc0010 lw gp,16(s8)**  **401608: 8fc20018 lw v0,24(s8)//$v0=j**  **40160c: 00000000 nop // 第二层循环末尾循环体**  **401610: 24420001 addiu v0,v0,1 // j++**  **401614: afc20018 sw v0,24(s8)**  **401618: 8fc20018 lw v0,24(s8) // 第二层循环条件判断处,$v0=j**  **40161c: 00000000 nop**  **401620: 28420006 slti v0,v0,6 // 当j < 6时，继续循环**  **401624: 1440ffe6 bnez v0,4015c0 <phase\_6+0xc0>//若<6,v0为1，再次执行循环，否则接着向下执行**  **401628: 00000000 nop**  **40162c: 8fc2001c lw v0,28(s8) // 获取i**  **401630: 00000000 nop // 第一层循环末尾循环体**  **401634: 24420001 addiu v0,v0,1 // i++**  **401638: afc2001c sw v0,28(s8)**  **40163c: 8fc2001c lw v0,28(s8) // 第一层循环条件判断处**  **401640: 00000000 nop**  **401644: 28420006 slti v0,v0,6**  **401648: 1440ffc1 bnez v0,401550 <phase\_6+0x50>// 当i < 6 时，继续循环**  **//设输入的第i个数为a[i]，则0<a[i]<7，并且输入的第i个数不能后他后面的某个数相等**  **/\***  **for (int i = 0;i < 6;i++) {**  **if (!(a[i] > 0 && a[i] < 7) explode\_bomb();**  **for (int j = i + 1;j < 6;j++)**  **if (a[i] == a[j]) explode\_bomb();**  **}**  **\*/**  **//第二个循环**  **40164c: 00000000 nop // 第一层循环的初始条件**  **401650: afc0001c sw zero,28(s8) // i = 0**  **401654: 10000028 b 4016f8 <phase\_6+0x1f8> //第一层循环判断处**  **401658: 00000000 nop // 第二层循环的初始条件**  **40165c: 3c020041 lui v0,0x41**  **401660: 24423130 addiu v0,v0,12592 // 令$v0=&firstnode = 0x413130**  **401664: afc20020 sw v0,32(s8) // m[$s8 + 32] = &firstnode**  **401668: 24020001 li v0,1 //设置第二层循环变量 j = 1**  **40166c: afc20018 sw v0,24(s8)**  **401670: 1000000a b 40169c <phase\_6+0x19c> // 第二层循环判断处**  **401674: 00000000 nop**  **401678: 8fc20020 lw v0,32(s8) // 取得&node**  **40167c: 00000000 nop**  **401680: 8c420008 lw v0,8(v0) // $v0 = m[$v0 + 8],即node=node->next**  **401684: 00000000 nop**  **401688: afc20020 sw v0,32(s8) // 将移动后的位置保存**  **40168c: 8fc20018 lw v0,24(s8) // 取得j**  **401690: 00000000 nop**  **401694: 24420001 addiu v0,v0,1 // j++**  **401698: afc20018 sw v0,24(s8)**  **40169c: 8fc2001c lw v0,28(s8) // 取得i**  **4016a0: 00000000 nop**  **4016a4: 00021080 sll v0,v0,0x2**  **4016a8: 27c30018 addiu v1,s8,24**  **4016ac: 00621021 addu v0,v1,v0**  **4016b0: 8c43000c lw v1,12(v0) // 取得输入的第i个数**  **4016b4: 8fc20018 lw v0,24(s8)**  **4016b8: 00000000 nop**  **4016bc: 0043102a slt v0,v0,v1 // 判断j是否小于输入的第i个数，是则循环继续**  **4016c0: 1440ffed bnez v0,401678 <phase\_6+0x178>**  **4016c4: 00000000 nop**  **4016c8: 8fc2001c lw v0,28(s8) // 取得i**  **4016cc: 00000000 nop**  **4016d0: 00021080 sll v0,v0,0x2**  **4016d4: 27c30018 addiu v1,s8,24**  **4016d8: 00621021 addu v0,v1,v0**  **4016dc: 8fc30020 lw v1,32(s8) // 将node = 链表的第a[i]个数取出**  **4016e0: 00000000 nop**  **4016e4: ac430024 sw v1,36(v0) // 将取出的数存入m[$v0+36],这里假设存入数组ans[]**  **//各个节点的值：0x0fd, 0x2d5, 0x12d, 0x3e5, 0x0d4, 0x1b0**  **//node1:0x0fd**  **//node2:0x2d5**  **//node3:0x12d**  **//node4:0x3e5**  **//node5:0x0d4**  **//node6:0x1v0**  **4016e8: 8fc2001c lw v0,28(s8)//获取第一重循环变量i**  **4016ec: 00000000 nop**  **4016f0: 24420001 addiu v0,v0,1//i++**  **4016f4: afc2001c sw v0,28(s8)**  **4016f8: 8fc2001c lw v0,28(s8) // 条件判断处**  **4016fc: 00000000 nop**  **401700: 28420006 slti v0,v0,6 // i < 6 则循环继续**  **401704: 1440ffd5 bnez v0,40165c <phase\_6+0x15c>**  **//这一段就是按照之前输入的数的顺序，将链表重新排序**  **/\***  **chainNode ans[6];**  **for (int i = 0;i < 6;i++) {**  **chainNode \*node = c.firstNode;**  **for (int j = 1;j < a[i];j++)**  **node = node->next;**  **ans[i] = node;**  **}**  **\*/**  **401708: 00000000 nop**  **40170c: 8fc2003c lw v0,60(s8)**  **401710: 00000000 nop**  **401714: afc20020 sw v0,32(s8) // 到达ans[0]**  **401718: 24020001 li v0,1//设置循环变量i=1**  **40171c: afc2001c sw v0,28(s8) // i = 1**  **401720: 10000016 b 40177c <phase\_6+0x27c>**  **401724: 00000000 nop // 循环开始**  **401728: 8fc2001c lw v0,28(s8) // 取得i**  **40172c: 00000000 nop**  **401730: 00021080 sll v0,v0,0x2**  **401734: 27c30018 addiu v1,s8,24**  **401738: 00621021 addu v0,v1,v0**  **40173c: 8c430024 lw v1,36(v0) // 取得res[i]，这里第一次是第二个节点（i从1开始)**  **401740: 8fc20020 lw v0,32(s8) // 这里第一次是第一个节点,即v0->v1**  **401744: 00000000 nop**  **401748: ac430008 sw v1,8(v0) // node->next = ans[i]**  **40174c: 8fc2001c lw v0,28(s8) // 取得i**  **401750: 00000000 nop**  **401754: 00021080 sll v0,v0,0x2**  **401758: 27c30018 addiu v1,s8,24**  **40175c: 00621021 addu v0,v1,v0**  **401760: 8c420024 lw v0,36(v0) // 取得ans[i]**  **401764: 00000000 nop**  **401768: afc20020 sw v0,32(s8) // m[s8+32]=$v0，存储当前节点ans[i]**  **40176c: 8fc2001c lw v0,28(s8) // 末尾循环体起始，取得i**  **401770: 00000000 nop**  **401774: 24420001 addiu v0,v0,1 // i++**  **401778: afc2001c sw v0,28(s8) // 存入内存中**  **40177c: 8fc2001c lw v0,28(s8) // 条件判断起始**  **401780: 00000000 nop**  **401784: 28420006 slti v0,v0,6 // 判断i < 6，是则循环继续**  **401788: 1440ffe7 bnez v0,401728 <phase\_6+0x228>**  **//这一部分好像是把链表重新按照ans中的顺序排序了**  **40178c: 00000000 nop**  **401790: 8fc20020 lw v0,32(s8)**  **401794: 00000000 nop**  **401798: ac400008 sw zero,8(v0) // node->next = null**  **40179c: 8fc2003c lw v0,60(s8)**  **4017a0: 00000000 nop**  **4017a4: afc20020 sw v0,32(s8) //v0 = firstnode**  **4017a8: afc0001c sw zero,28(s8) // i = 0**  **4017ac: 10000032 b 401878 <phase\_6+0x378>**  **4017b0: 00000000 nop**  **4017b4: 8f82806c lw v0,-32660(gp)**  **4017b8: 00000000 nop**  **4017bc: 8c42002c lw v0,44(v0)**  **4017c0: 00000000 nop**  **4017c4: 30420001 andi v0,v0,0x1**  **4017c8: 304200ff andi v0,v0,0xff**  **4017cc: 10400012 beqz v0,401818 <phase\_6+0x318> // if...else语句，判断奇偶，奇数1，不跳转；偶数 0，跳转**  **4017d0: 00000000 nop**  **4017d4: 8fc20020 lw v0,32(s8)**  **4017d8: 00000000 nop**  **4017dc: 8c430000 lw v1,0(v0)**  **4017e0: 8fc20020 lw v0,32(s8)**  **4017e4: 00000000 nop**  **4017e8: 8c420008 lw v0,8(v0)**  **4017ec: 00000000 nop**  **4017f0: 8c420000 lw v0,0(v0)**  **4017f4: 00000000 nop**  **4017f8: 0062102a slt v0,v1,v0 // 比较重排后链表前后两个数的大小，升序则爆炸**  **4017fc: 10400015 beqz v0,401854 <phase\_6+0x354>**  **401800: 00000000 nop**  **401804: 0c10087c jal 4021f0 <explode\_bomb>**  **401808: 00000000 nop**  **40180c: 8fdc0010 lw gp,16(s8)**  **401810: 10000010 b 401854 <phase\_6+0x354>**  **401814: 00000000 nop**  **401818: 8fc20020 lw v0,32(s8)//偶数跳转到这**  **40181c: 00000000 nop**  **401820: 8c430000 lw v1,0(v0)**  **401824: 8fc20020 lw v0,32(s8)**  **401828: 00000000 nop**  **40182c: 8c420008 lw v0,8(v0)**  **401830: 00000000 nop**  **401834: 8c420000 lw v0,0(v0)**  **401838: 00000000 nop**  **40183c: 0043102a slt v0,v0,v1 // 比较重排后链表前后两个数的大小，降序则爆炸**  **401840: 10400004 beqz v0,401854 <phase\_6+0x354>**  **401844: 00000000 nop**  **401848: 0c10087c jal 4021f0 <explode\_bomb>**  **40184c: 00000000 nop**  **401850: 8fdc0010 lw gp,16(s8)**  **401854: 8fc20020 lw v0,32(s8)**  **401858: 00000000 nop**  **40185c: 8c420008 lw v0,8(v0)**  **401860: 00000000 nop**  **401864: afc20020 sw v0,32(s8)**  **401868: 8fc2001c lw v0,28(s8)**  **40186c: 00000000 nop**  **401870: 24420001 addiu v0,v0,1 # i++**  **401874: afc2001c sw v0,28(s8)**  **401878: 8fc2001c lw v0,28(s8)**  **40187c: 00000000 nop**  **401880: 28420005 slti v0,v0,5 # i < 5**  **401884: 1440ffcb bnez v0,4017b4 <phase\_6+0x2b4>**  **401888: 00000000 nop**  **40188c: 03c0e821 move sp,s8**  **401890: 8fbf005c lw ra,92(sp)**  **401894: 8fbe0058 lw s8,88(sp)**  **401898: 27bd0060 addiu sp,sp,96**  **40189c: 03e00008 jr ra**  **4018a0: 00000000 nop**  **//根据输入的数重排链表，学号最后一位为奇数则需要升序排列，若为偶数需要将序排列**  **C语言版：**  **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 = 1;i < 6;i++)**  **if (node->element < node->next->element)**  **explode\_bomb();**  **}**  **else {**  **node = c.firstNode;**  **for (int i = 1;i < 6;i++)**  **if (node->element > node->next->element)**  **explode\_bomb();**  **}** | | |
| **结论分析与体会：**   1. 学习了各个指令的原理及使用方法 2. 学会了设置断点，使用GDB得到寄存器存储的值，或者了解对应的数据结构 3. 学习到循环，递归，switch case等操作在反汇编语言下的实现方法 4. 学习了对于节点，将指针的地址存入寄存器，读出寄存器中指针的值，存储到寄存器中地址后移8位的地址。从而实现了将节点连接。 5. 通过phase\_6中对链表的操作知道了对于节点，会将指针（即为节点数据的地址）存储在内存中 | | |